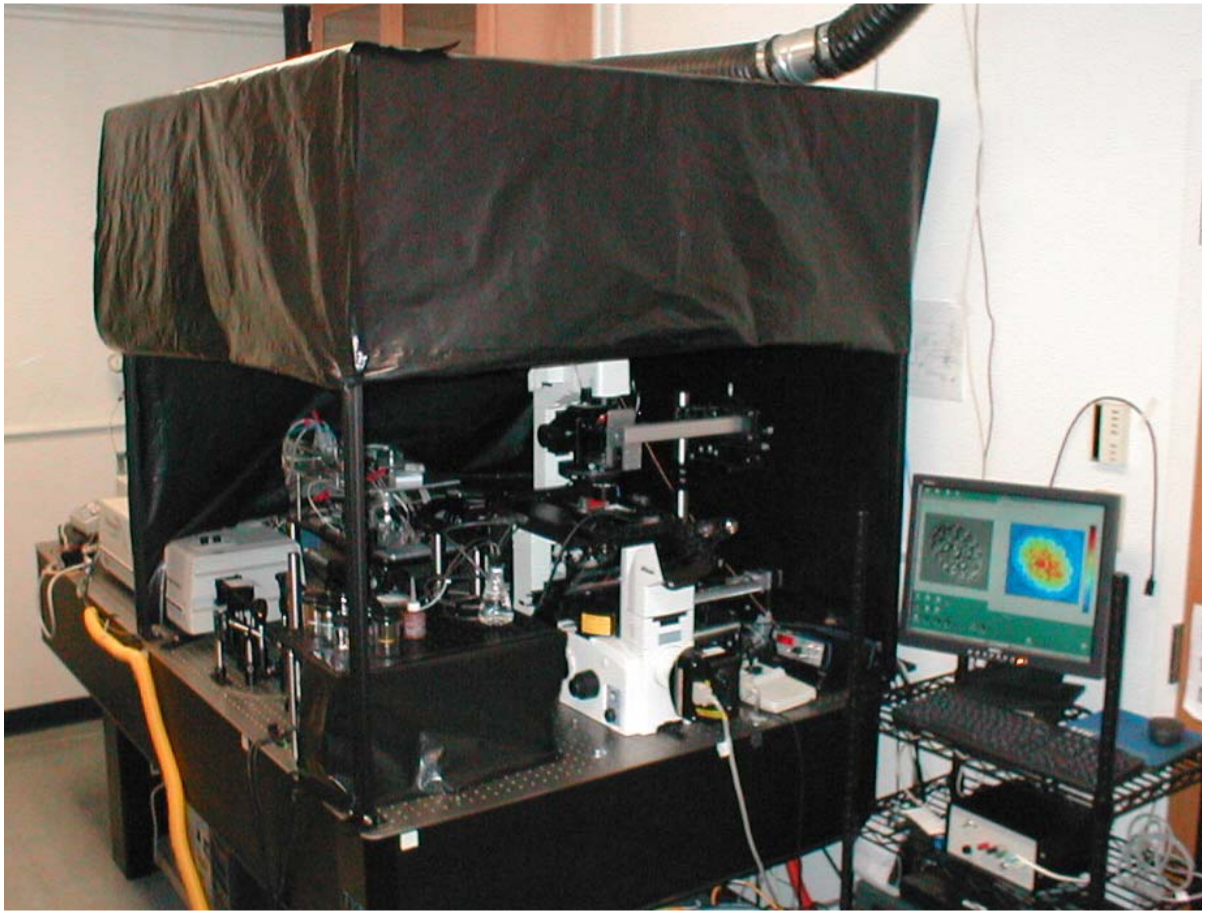
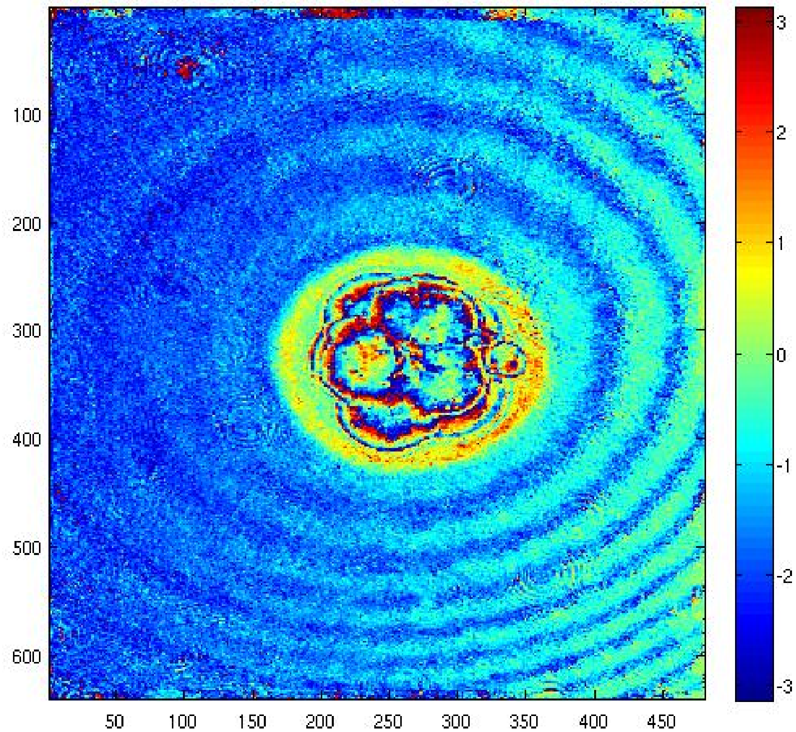
Phase Unwrapping and Affine Transformations using CUDA

Perhaad Mistry, Sherman Braganza,
David Kaeli, Miriam Leeser
ECE Department
Northeastern University
Boston, MA

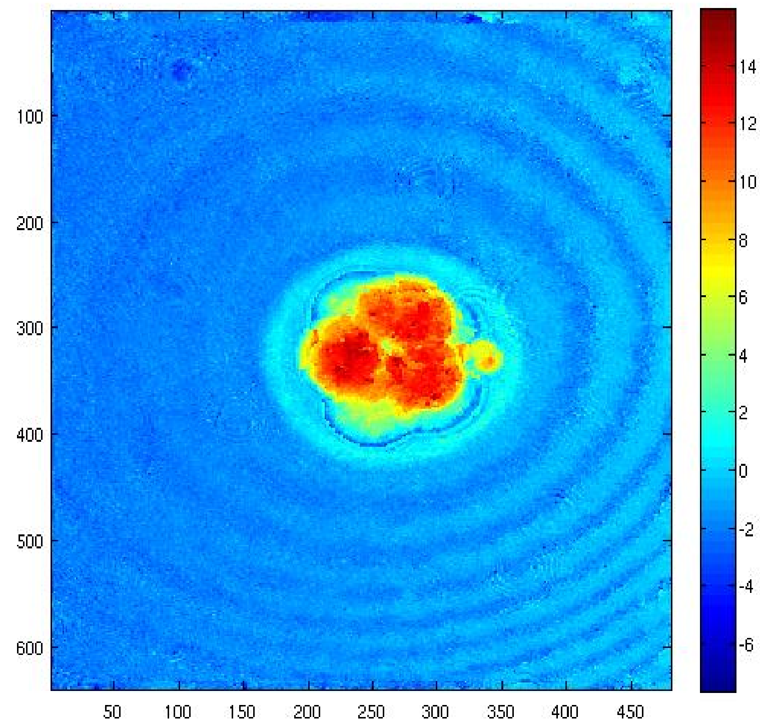
Keck 3D Fusion Microscope



Motivation



Wrapped Phase Image



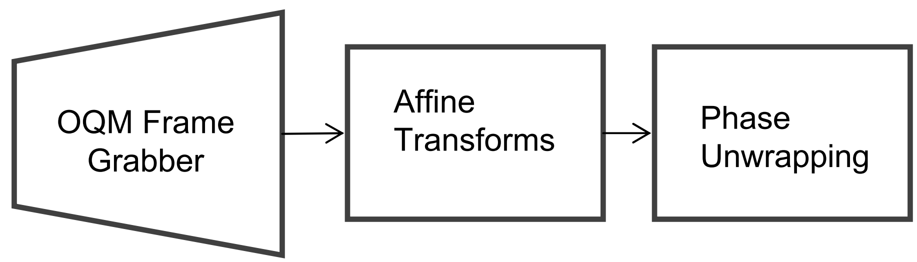
Unwrapped Phase Image

Motivation

- Phase Unwrapping used for research in In-Vitro Fertilization (IVF) in Optical Quadrature Microscopy
- Cell counts obtained after unwrapping determine viability of embryos
- Results required at close to real time to see changes in sample
- Improve cost-effectiveness of OQM and quality of research

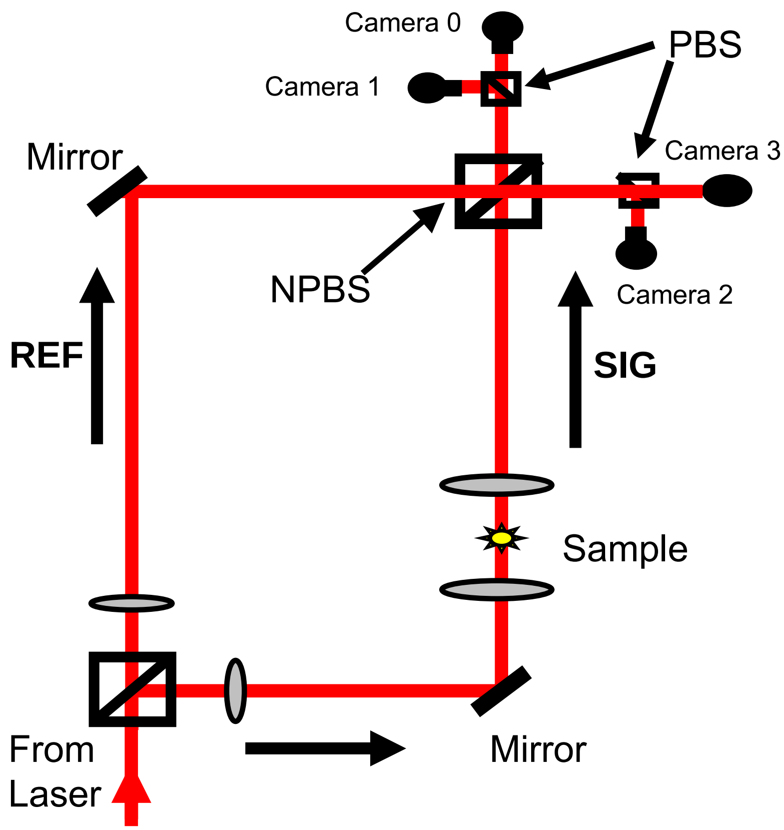
Outline

- Optical Quadrature Microscopy
- Affine Transformations
- Minimum L^P Norm Phase Unwrapping
- MEX Interface Usage
- Performance
- Future Work
- Conclusion



OQM Imaging System Layout

OQM Imaging



Optical Quadrature Microscopy Layout

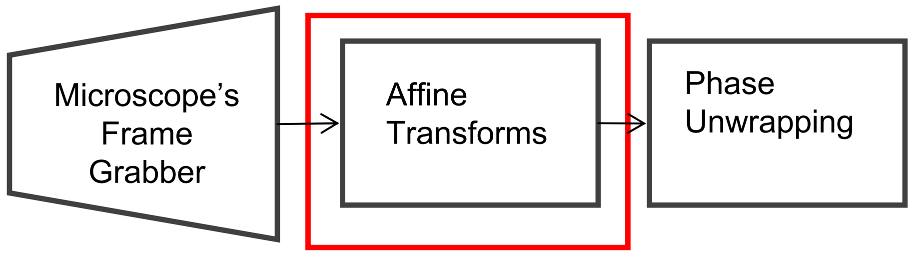
- Pattern determined by phase difference between laser beams
- Phase calculated using magnitudes captured from four cameras
- Mixed (M), signal only(S), reference(R) and dark (D) images
- Phase difference measured by angle(E) ranges from –π to π

$$E = \frac{1}{4}\sum_{n=0}^{n=3} i^n * \frac{M_n - S_n - R_n}{\sqrt{R_n}}$$

- angle(E) yields phase that is required for unwrapping

Affine Transformation

- Align images to fix sample's position in different images
- 2x2 and 2x1 matrices obtained from microscope
- Image pixels divided among blocks of threads
- Each thread calculates one pixel
- Implemented using CUDA since result needed for Phase Unwrapping



Algorithm for Affine Transform [1]

(x,y) = f(BlockId, ThreadID)

Load Data = Image[x][y]

Includes - Rotation (θ), Scaling (s), Shearing (sh)

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s(x)*\cos(\theta) & sh(y)*\sin(\theta) \\ -sh(x)*\sin(\theta) & s(y)*\cos(\theta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a \\ b \end{pmatrix}$$

Store Image[x'][y'] = Data

(Uncoalesced Writes)

[1] *A. K. Jain,* Fundamentals of digital image processing, Prentice Hall

Noise Removal & Phase Information

- Separate "noise" only image available (Dark Image)
- Dark voltage image subtracted from the mixed, signal and reference images
- Subtraction of signal image (S_n) and reference image (R_n) due to fixed pattern levels of individual cameras

$M_n = M_n - D_n$ and so on for S_n and R_n

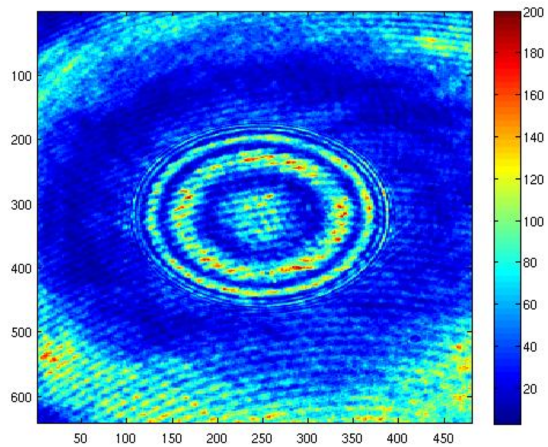
$$E = \frac{1}{4}\sum_{n=0}^{n=3} i^n * \frac{M_n - S_n - R_n}{\sqrt{R_n}}$$

n: camera number

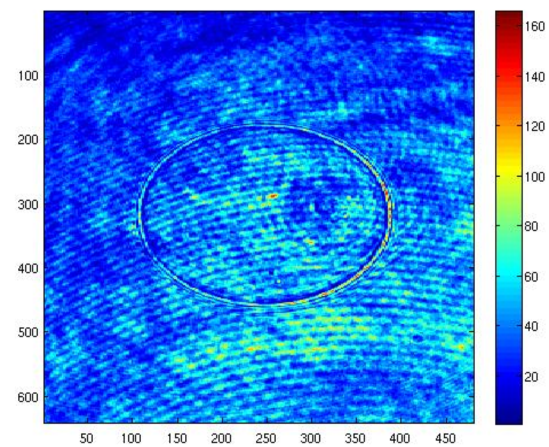
$$phase = angle(E)$$

- Square root of R_n to balance intensities for detection

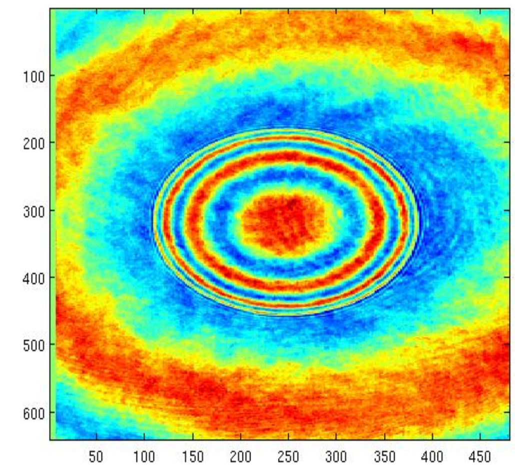
Example Images for Affine Transform



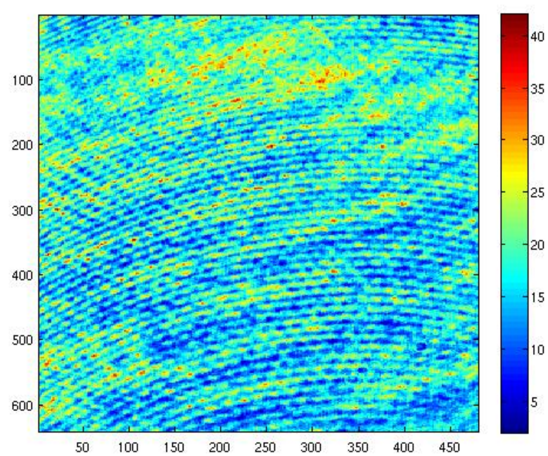
Mixed Image



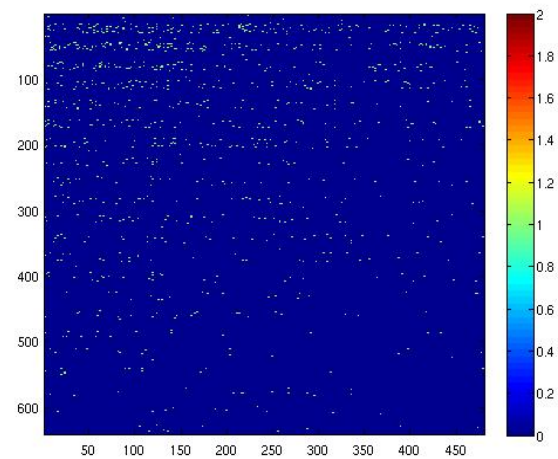
Signal only Image



After Affine Transform and Noise Removal



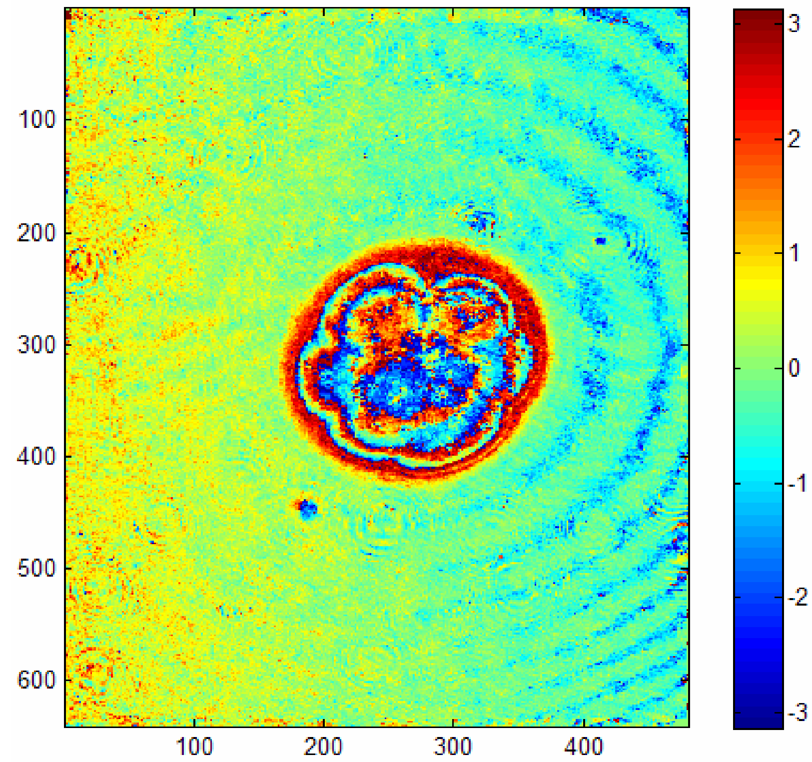
Reference Image



Dark Image

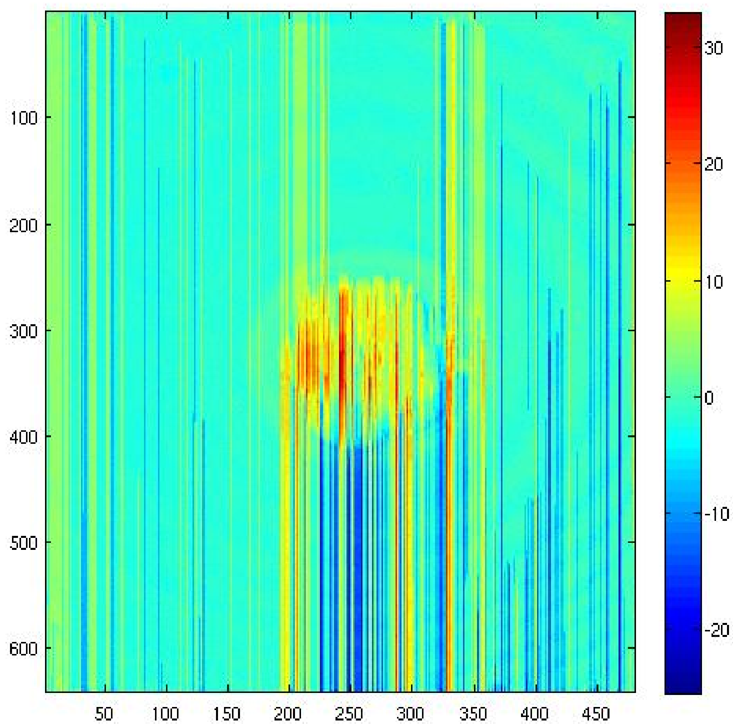
Phase Unwrapping

- Optical phase difference from *interferometric* image is shown
- Parameter of interest is Optical path difference which can be more than π
- Unwrapping along a line:
  - Sum till a gradient of π then, add 2π
- In 2D if noise is present
  - Path dependencies while summing
  - Causes accumulation errors

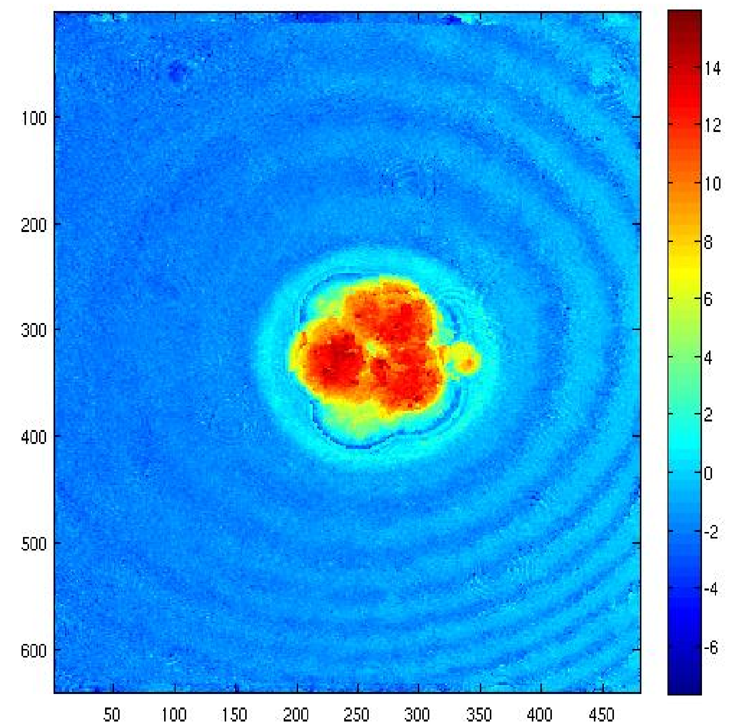


Wrapped Image
(note range of values)

MATLAB's Unwrap v/s Minimum L^P Norm



Unwrap Using MATLAB's unwrap() function



Unwrap Using Minimum L^p Norm Algorithm **[1]**

[1] *Dennis C. Ghiglia, Mark D. Pritt:* Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software

Minimum L^P Norm Algorithm

- Minimize difference between gradients of wrapped and unwrapped data [1]
- Nonlinear PDE since U and V are data dependant weights
- PDE solved iteratively using the Preconditioned Conjugate Gradient Method

$$c_{i,j} = (\phi_{i+1,j} - \phi_{i,j})U_{i,j} + (\phi_{i+1,j} - \phi_{i,j})V_{i,j}$$
$$-(\phi_{i,j} - \phi_{i-1,j})U_{i-1,j} - (\phi_{i,j} - \phi_{i,j-1})V_{i,j}$$
$$c_{i,j} = \Delta^x_{i,j}U(i,j) - \Delta^x_{i-1,j}U(i-1,j) +$$
$$\Delta^y_{i,j}V(i,j) - \Delta^y_{i-1,j}V(i-1,j)$$

where

$\phi_{x,y}$ is the unwrapped phase at (x,y)

$\Delta^x_{x,y}$ and $\Delta^y_{x,y}$ denote the wrapped phase in the x and y direction respectively

Can be expressed as $Q\phi = c$

[1] *Dennis C. Ghiglia, Mark D. Pritt.* Two-Dimensional Phase Unwrapping: Theory, Algorithms, and Software, Wiley New York

Preconditioned Conjugate Gradient

- Solve un-weighted least square phase unwrapping problem
- Minimize ε^2
- Preconditioning using a DCT needed
- Conjugate gradient method called after DCT preconditioning

$$\varepsilon^2 = \sum_{i=0}^{M-2} \sum_{j=0}^{N-2} (\phi_{i+1,j} - \phi_{i,j} - \Delta^x_{i,j})^2 + \sum_{i=0}^{M-2} \sum_{j=0}^{N-2} (\phi_{i+1,j} - \phi_{i,j} - \Delta^y_{i,j})^2$$

where $\phi_{i,j}$ is the wrapped phase at (i,j) and

$\Delta^x_{i,j}$ *is* the unwrapped phase at (i,j)

After discretizing the equation

and taking the 2D Fourier Transform

$$\Phi_{m,n} = \frac{\mathrm{P}_{m,n}}{2\cos(\pi m/M) + 2\cos(\pi n/N) - 4}$$

where Φ and P are the Fourier Transformed versions of ϕ and Δ respectively

Better DCT Implementation

- Implementing N point DCT using DFT uses 2N point DFT
- Another method seen in [1]
  - Rearrange x(n) into v(n) such that

$$v(n) = x(2n) \qquad 0 \le n \le \left[\frac{N-1}{2}\right]$$
$$= x(2N-2n-1) \qquad \left[\frac{N+1}{2}\right] \le n \le N-1$$

  - DCT(x(n)) = 2*DFT(W_{4N}^{k} * Real(v(n)))
- For 2D DCT: 4x less work since N*N instead of 2N*2N
- PCG is ~ 90% of L^P Norm Execution time
- Shuffle kernel before CUFFT call

[1] Makhoul J. A fast cosine transform in one and two dimensions.

Minimum L^P Norm Algorithm

for *k ← 0* **to** *L^P Norm Count*

Calculate Data Derived Weights

for *j ← 0* **to** *PCG Iteration Count*

DCT To DFT Steps (Shuffle Kernel)

Execute CUFFT Call

Point-Wise Complex Multiplication to Get DCT Result

Scaling Step

Execute CUFFT Call to do iDCT

Conjugate Gradient (Point Wise Matrix Operations)

end for

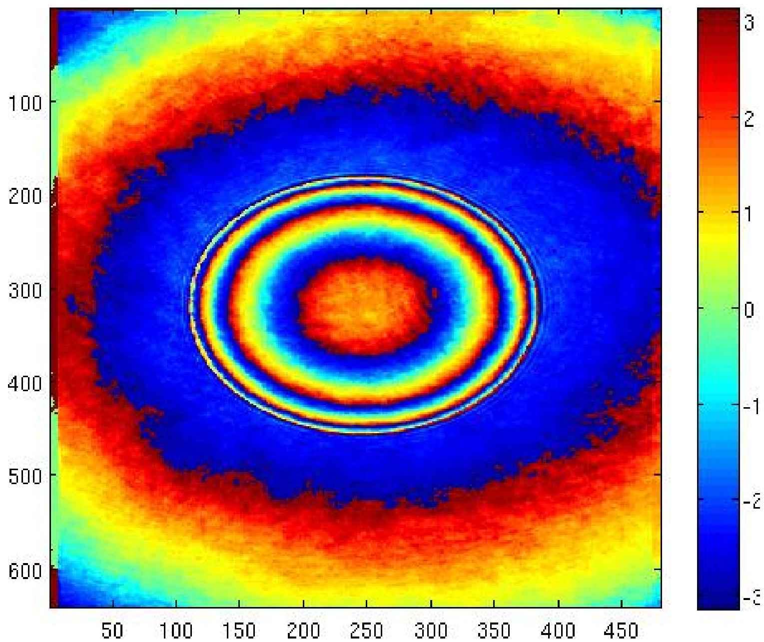
if *Convergence* **exit**

end for

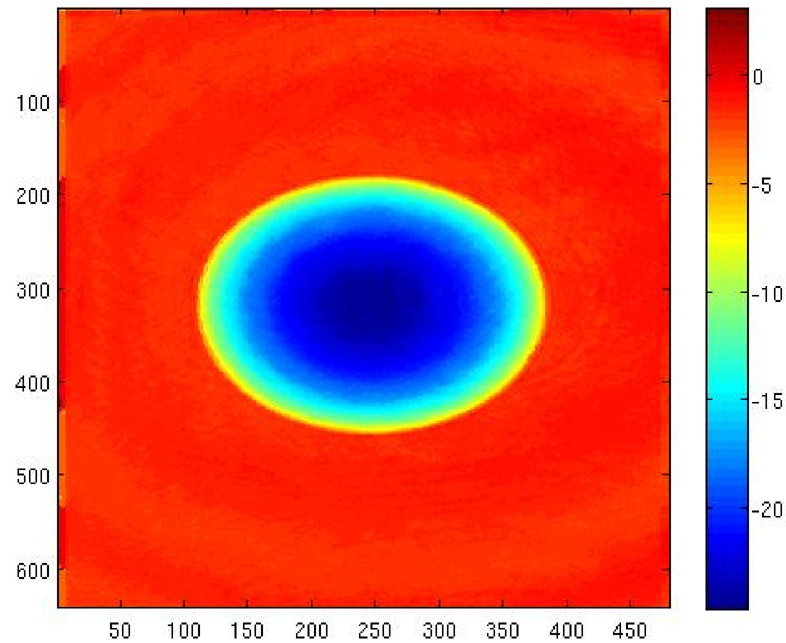
Preconditioning before CG

PCG Algorithm

Example Images – Phase Unwrapping



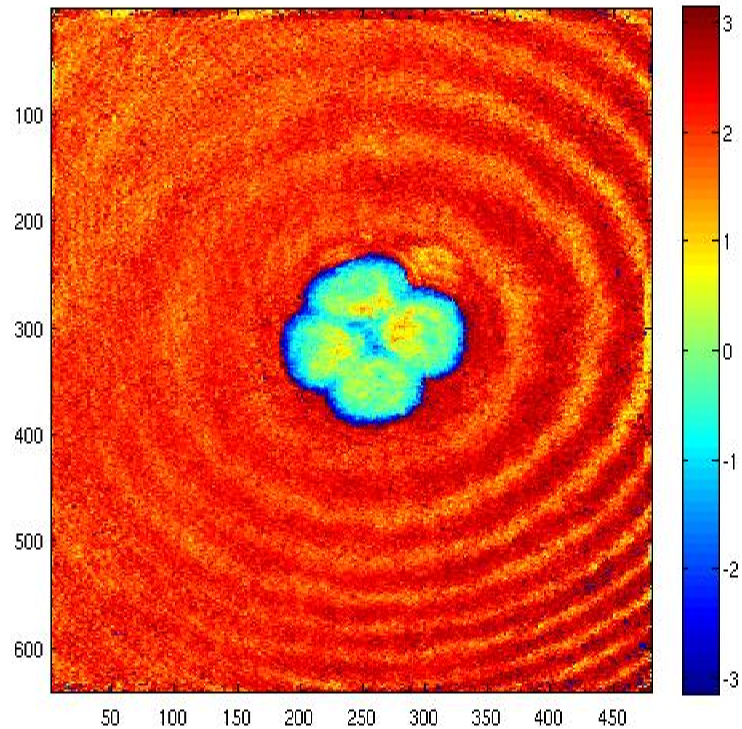
Wrapped Phase Data



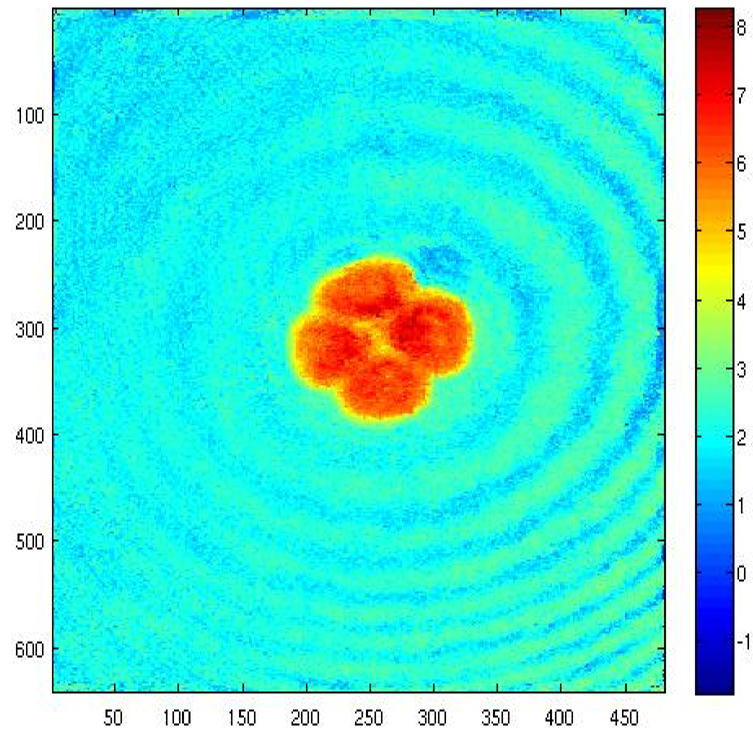
Unwrapped Phase Data

Images of Glass Bead and water

Example Images – Mouse Embryo



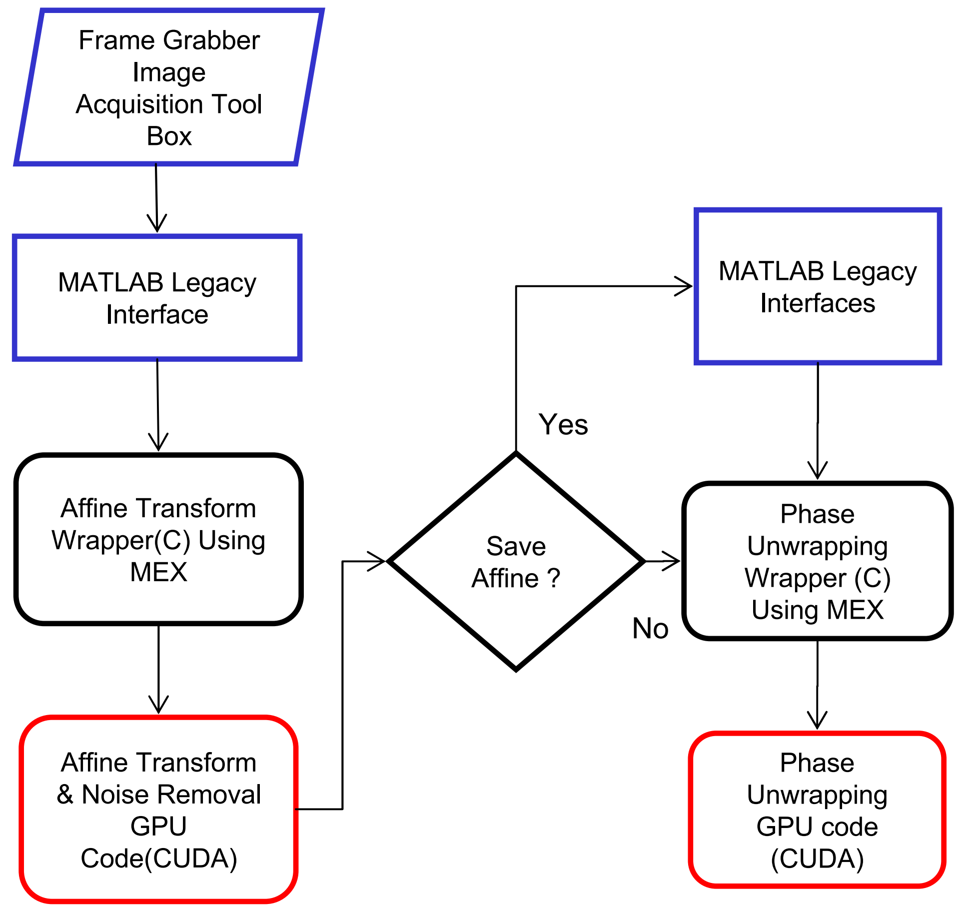
Wrapped Phase Data



Unwrapped Phase Data

MATLAB External Interface (MEX)

- MEX produces linked objects from C code
- MATLAB present in our system due to the frame grabbers
- Gateway function in C makes MATLAB data (mxArray) visible to our linked C code



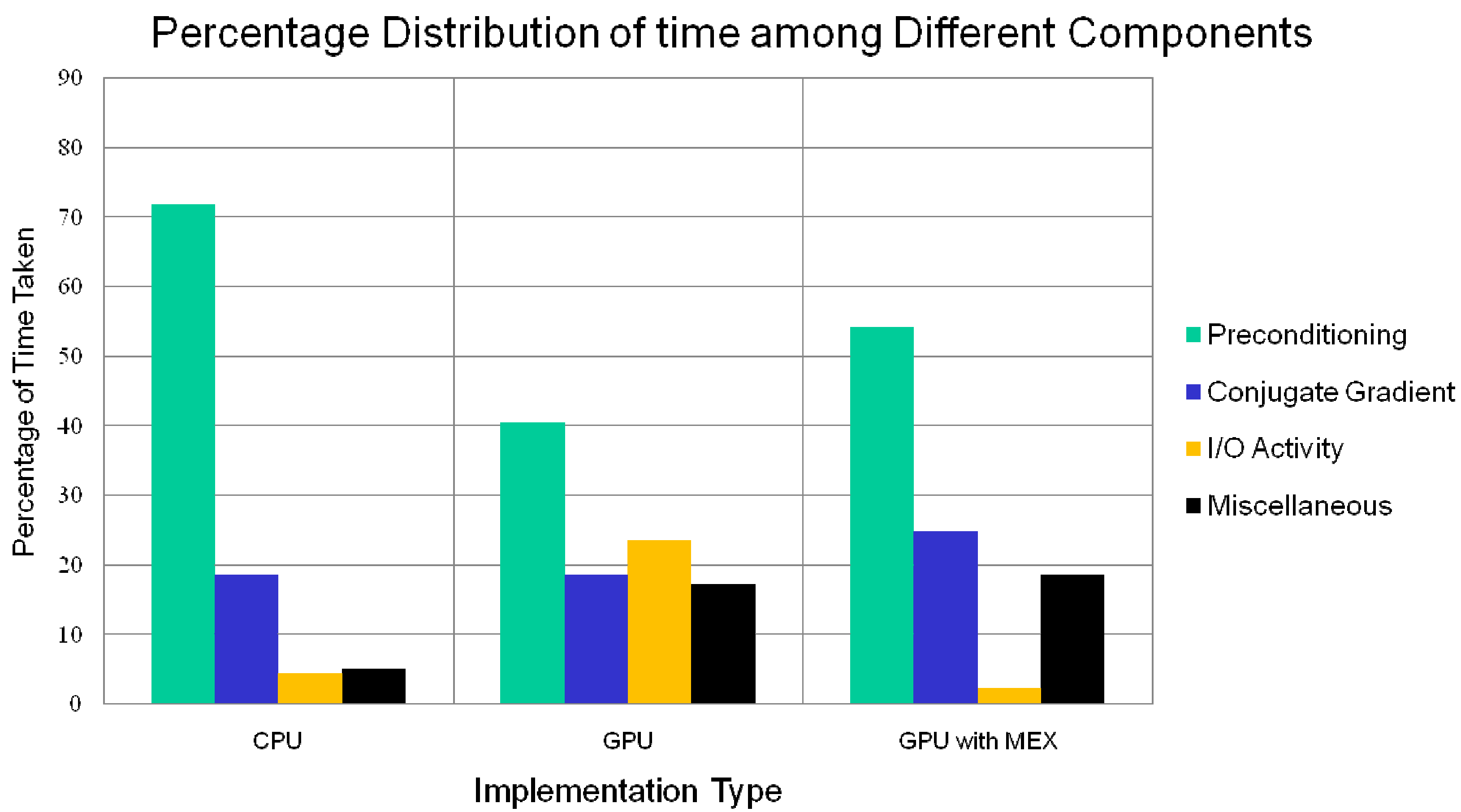
OQM Processing Control Flow

Performance (Phase Unwrapping)

	Baseline Time	GPU Time	GPU Speedup	MEX and GPU Time(sec)	Mex & GPU Speedup
Preconditioning	11.17	1.2	9.3X	1.2	9.3X
Conjugate Grad.	2.89	0.55	5.25X	0.55	5.25X
IO Activity	0.7	0.7	1X	0.02	NA
Miscellaneous	0.79	0.51	1.53X	0.41	1.92X
Total for Unwrap	15.55	2.97	5.24X	2.16	**7.20X**

Baseline : Serial implementation using
FFTW for the Fourier Transform

Performance Results



Future Work

- Study implementation of Conjugate Gradient
- Study scatter operations in CUDA
  - Present literature shows improvements only for larger data sizes (may be better on G200)
- Multi-Gpu version for imaging scenarios where images may be grabbed at faster rates
  - Presently only one image at a time acquired.
- Phase Unwrapping also required for applications like Synthetic Aperture Radar (SAR)

Conclusions

- Implemented the computationally intensive Minimum L^p Norm Phase Unwrapping and Affine Transforms on the GPU
- Not a batch-oriented process
  - Latency was critical, single image used at a time
- Reducing latency throughout the system improves quality of research and time to discovery in Optical Science Laboratory

Thank You

Questions?

Perhaad Mistry

(pmistry@ece.neu.edu)