# MIC-GPU: High-Performance Computing for Medical Imaging on Programmable Graphics Hardware (GPUs)

## Various Pipelines and Load Balancing

Klaus Mueller

Computer Science
Center for Visual Computing
Stony Brook University

## Iterative Pipeline

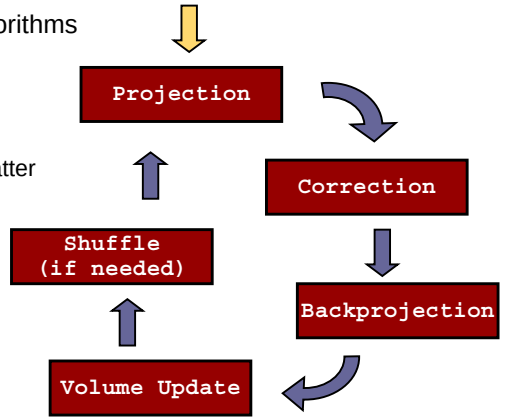
Two stack representation

Kernel selection depends on algorithms

Projection/Backprojection
- attenuation only
- attenuation + emission
- attenuation + emission + scatter

Correction
- subtraction (algebraic)
- division (EM)

Update
- addition (algebraic)
- multiply (EM)

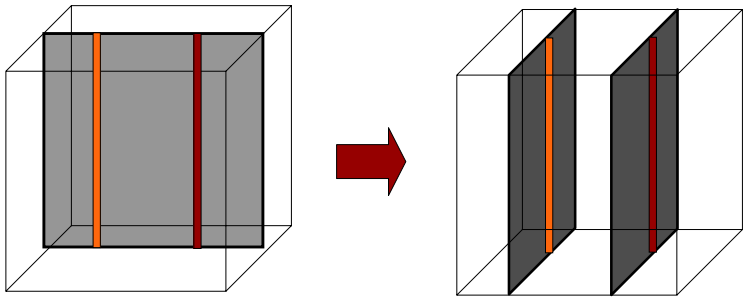





## Shuffle

Maintain data consistency

Data transfer/copy



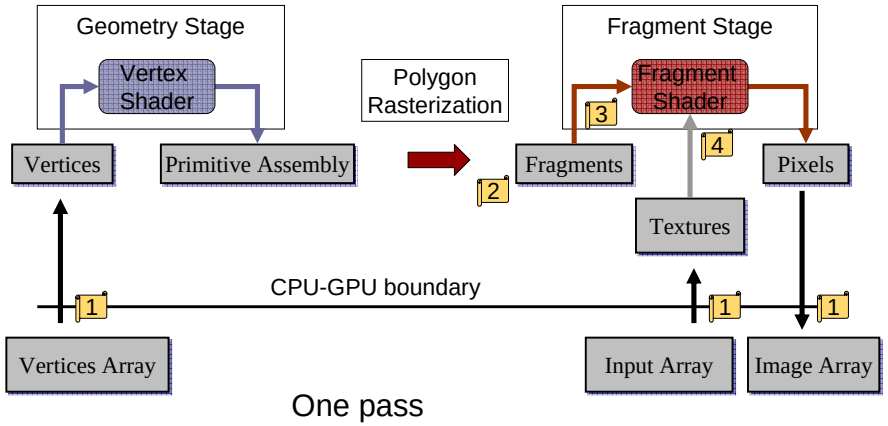


## Feldkamp (FDK) Filtered-Backprojection

Three consecutive steps
- projection space filtering
- backprojection
- volume space weighting

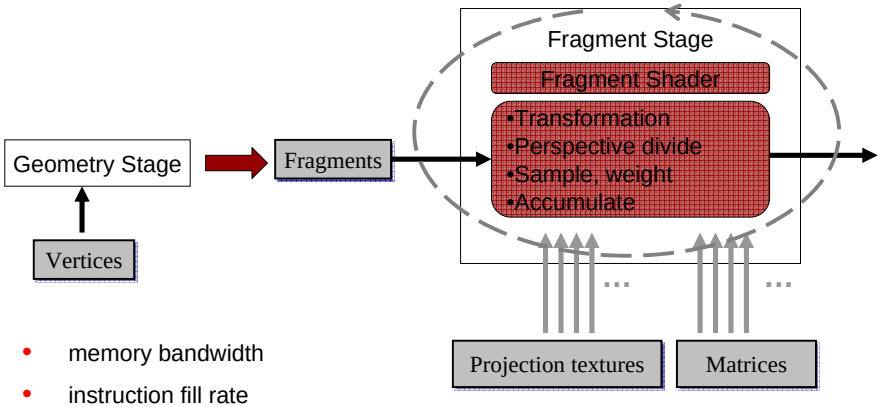
Projection space filtering using FFT
- FFT is sequential
- GPU-based FFT
  - [Govindaraju'06]
  - [Sumanaweera'05]
- good for >10k elements
- FFTW

# Graphics Pipeline Revisited





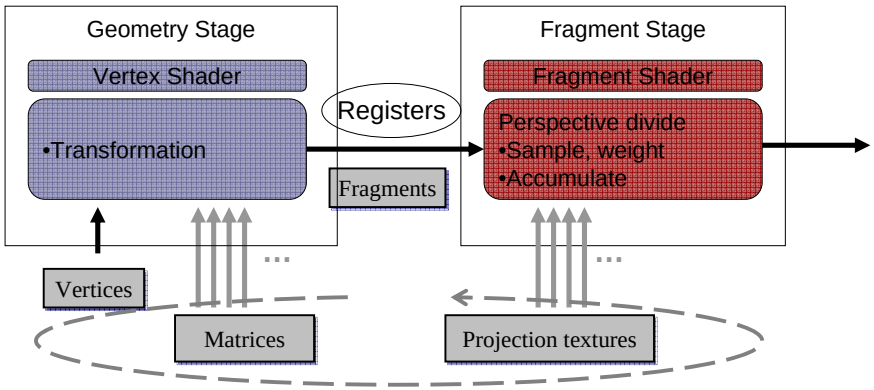

# Pipeline: GPGPU




- memory bandwidth
- instruction fill rate
- more projections, less passes & writes
- limitation of # of texture units ➔ 3D texture, high latency



# Pipeline: Graphics



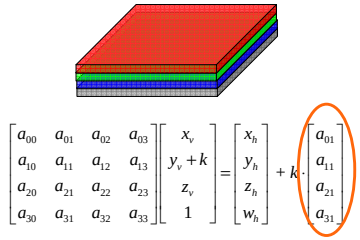

- number of projections in one pass limited by the registers (maximum = 8)
- more balanced pipeline



# Acceleration via RGBA

## Volume packing

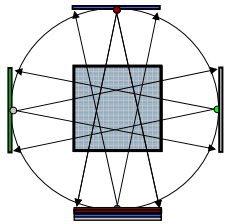
- adjacent 4 volume slices ➔ RGBA
- constant *Y* offset
- 1 matrix + 1 offset vector
- extract channel values
- reduce # of write-to-textures



$$\begin{bmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{10} & a_{11} & a_{12} & a_{13} \\ a_{20} & a_{21} & a_{22} & a_{23} \\ a_{30} & a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x_v \\ y_v + k \\ z_v \\ 1 \end{bmatrix} = \begin{bmatrix} x_h \\ y_h \\ z_h \\ w_h \end{bmatrix} + k \cdot \begin{bmatrix} a_{01} \\ a_{11} \\ a_{21} \\ a_{31} \end{bmatrix}$$

## Projection packing

- symmetry in projection layout
  - four orthogonal views
  - [φ, φ+90°, φ+180°, φ+270°]
  - no re-interpolation ➔ 4-fold speedup
- need all projections

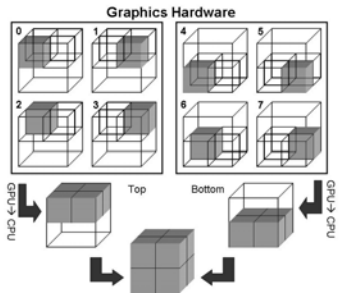


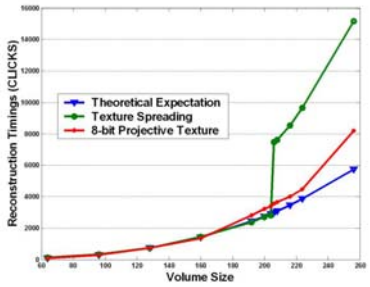
# Memory Management

Scalability

Texture compression is lossy

Volume partition and stitch



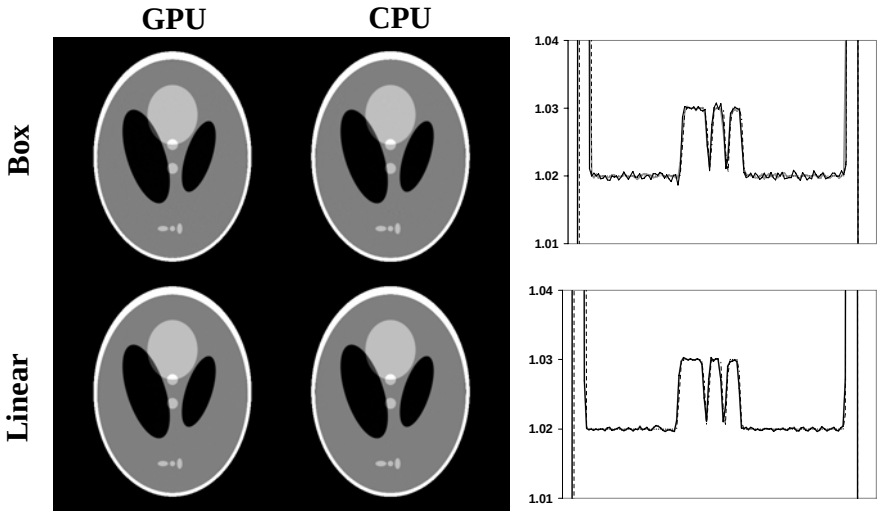




| Vol. | Projs. | Partition | Time |
|---|---|---|---|
| $128^3$ | 160 | Yes/No | 1.7s/1.7s |
| $256^3$ | 160 | Yes/No | 12s/45s |

NVIDIA Geforce FX5900

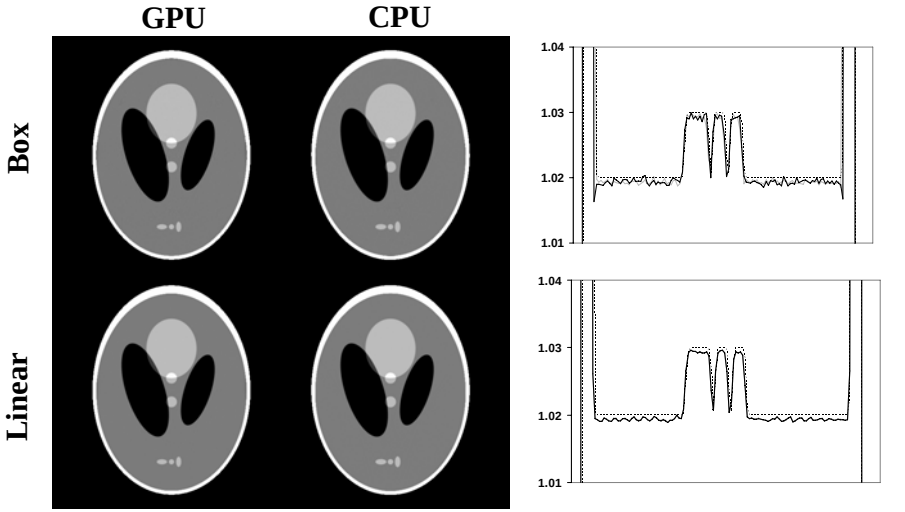


# FDK: Reconstruction Quality Parallel-Beam



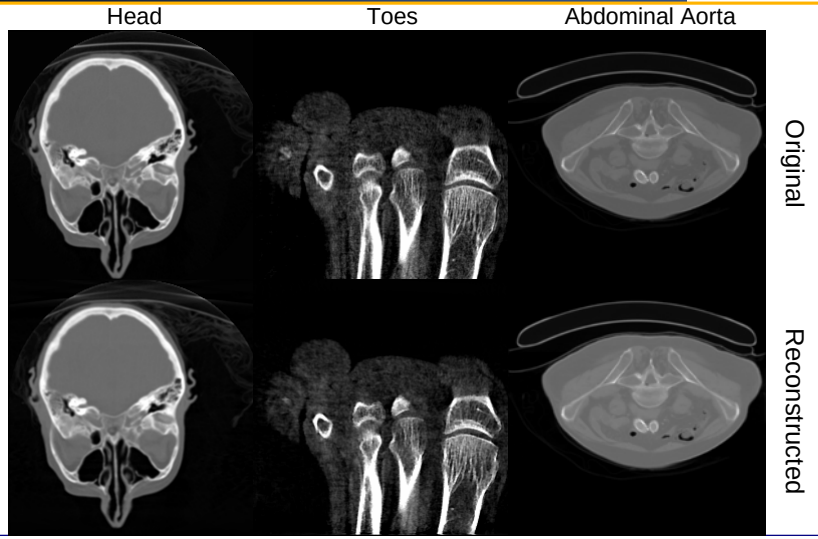



# FDK: Reconstruction Quality Cone-Beam






# FDK: Medical Datasets



# Iterative: Performance

NVIDIA 8800 GTX, 768MB memory

| Platform | Algorithm | Volume | Projections | 1 iter. | 3 iter. |
|---|---|---|---|---|---|
| SGI | SART | $128^3$ | 80 | 1.1min | 3.1min |
| GPU | SART | $128^3$ | 80 | 2s | 6.1s |
| GPU | OS-EM | $128^3$ | 80 | 5s | 15s |
| GPU | SART | $256^3$ | 160 | 7.5s | 22s |
| GPU | SIRT | $256^3$ | 160 | 5.0s | 15s |

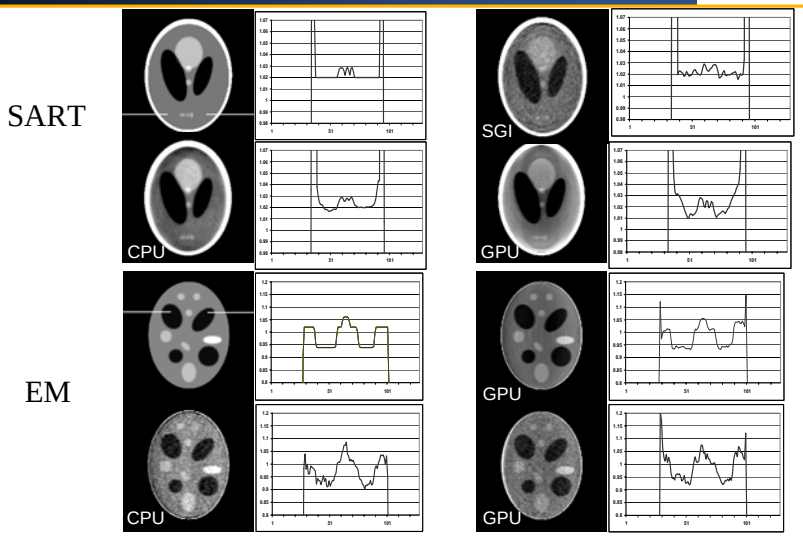


# Iterative: Performance

| Algorithm | Volume | Projections | 1 iter. | 3 iter. |
|---|---|---|---|---|
| Emission SART | $128^3$ | 80 | 8.5s | 25s |
| Emission SART | $256^3$ | 160 | 12s | 35s |



# Iterative: Reconstruction Quality






# Course Schedule

| | |
|---|---|
| 1:30 – 2:00: | Introduction |
| 2:00 – 2:30: | GPU architecture, programming model, and programming facilties |
| 2:30 – 3:00: | GPU programming examples (image processing) |
| | *Coffee Break* |
| 3:30 – 4:00: | CT reconstruction pipeline components |
| 4:00 – 4:30: | GPU-acceleration of individual components |
| 4:30 – 5:00: | Various CT reconstruction pipelines, load balancing and load estimation |
| 5:00 – 5:30: | Reconstruction visualization and final remarks |