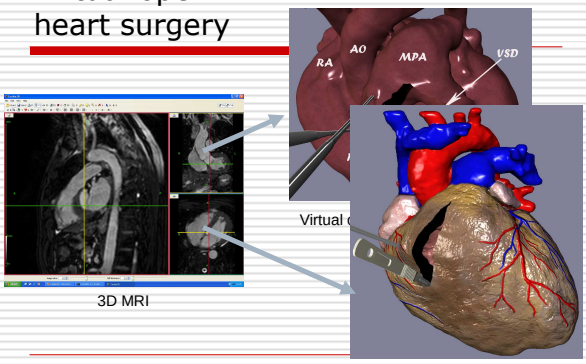
Introduction to general purpose computation on graphics hardware

- Virtual open heart surgery
- Parallel magnetic resonance imaging

Virtual open heart surgery

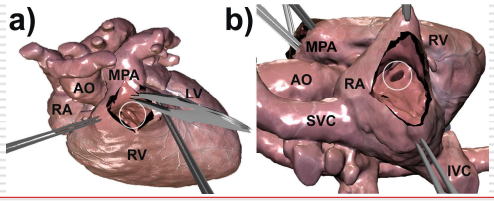


Congenital heart disease

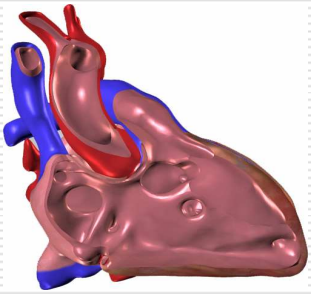
- Complex <u>individual</u> morpholgy
- Repeated surgery often necessary
- Surgical outcome will influence the entire life-time

Training of cardiac surgery

- 3D MRI of a volunteer
  - (Several) VSDs added manually in post-processing
- How can we best access the VSD?
  - Trans-ventricular or trans-atrial incision?



"Configurable" septal defects



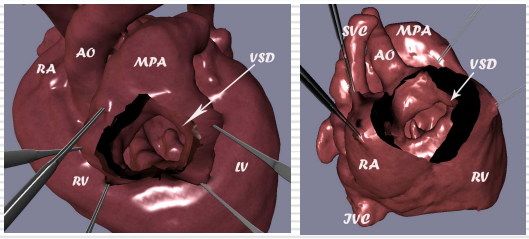
Challenge

- How to model and compute tissue elasticity sufficiently fast in a very complex organ such as the heart?
  - Real-time interaction is <u>essential</u>
- Answer
  - Implement the simulation engine on the GPU
  - GPU responsible for computing deformable model, visualization, and haptic feedback

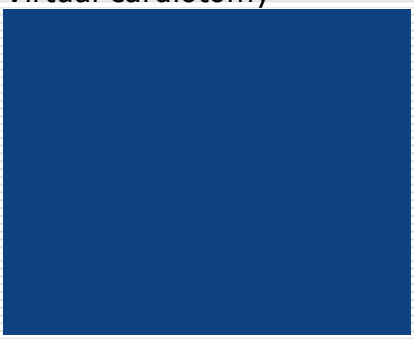
Surgical training

Virtual cardiotomy

- 2 months old boy
  - Double outlet right ventricle
  - VSD / septum deviates to the right
  - Resembling Taussig-Bing heart
  - Intramural course of coronary arteries
- **Biventricular repair possible? Switch or intracardiac repair?**

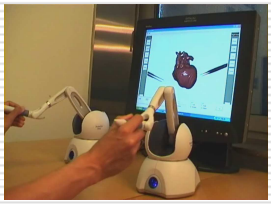


Virtual cardiotomy



Virtual Surgery – Surgical Simulation

- Setup
  - State of the art pc with high-end graphics card
    - $3000
  - Two Phantom Omnis for force feedback
    - 2 x $2500



- So I have told you why.
- Now a little about how…

Motivation: Computation on the GPU

- *Floating point* operations per second (MAD instructions)
  - Geforce 7900 GTX : **255** GFlops (measured)
  - Radeon X1900XTX : **241** GFlops (measured)
  - 3.8 GHz Intel Xeon : **7.4** GFlops (theoretically)

Explicit model

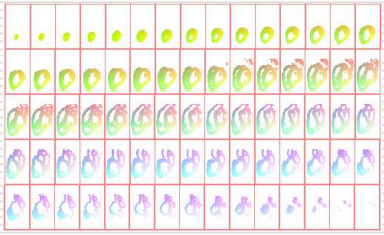
- Newtonian motion: $\mathbf{M}\ddot{\mathbf{u}} + \mathbf{C}\dot{\mathbf{u}} + \mathbf{K}\mathbf{u} = \mathbf{f}$
- Spring-mass models

$$\hat{\mathbf{F}}_i = \sum_{j \in N(\mathbf{P}_i)} k_{ij}\left(\left\|\mathbf{P}_i\mathbf{P}_j\right\| - l_{ij}^0\right)\frac{\mathbf{P}_i\mathbf{P}_j}{\left\|\mathbf{P}_i\mathbf{P}_j\right\|}$$

- The acceleration of a particle is determined by spring forces and damping forces
- With explicit time integration
  - Verlet integration updates particle positions from
    - Positions of the two previous iterations
    - Elastic force vector **F**

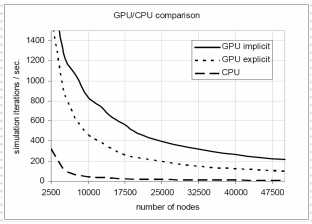
Parallel computation

- One processor pr pixel (conceptually)
- Image (rgb) corresponds to positions (xyz)
- Kernel invoked on all pixels (at the same time)

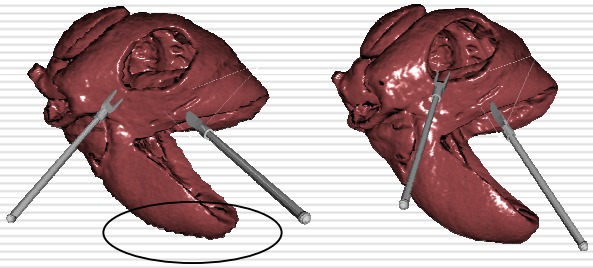


Performance

- Geforce 6800 Ultra
- GPU 30 times faster as the CPU
- But we are now three GPU generations further on…

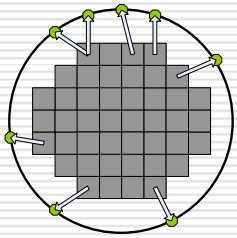


Visualisation



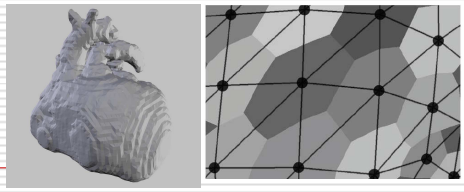
Decoupling of simulation and visualisation

- Smooth geometry
  - Circle (black/green)
- Represented by
  - Simulation node (grey)
  - Offset vector (arrow)
    - in "Tangent space"



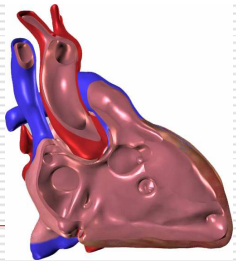
Picking – force feedback, cutting, etc.

- To identify which mass points that are touched
  - Render the "simulation surface" from the tip of the instrument
    - Shading indicates the particle "coordinates" of the nearest particle in GPU memory



Summary (Geforce 7900GTX)

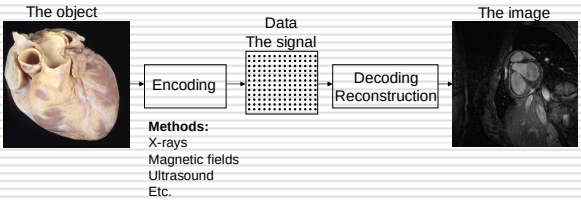
- Simulation (spring-mass)
  - 30.000 nodes / 18 springs
  - Updated at 500 Hz
- Visualisation
  - 80.000 faces
  - Updated at 25 Hz
- Force feedback
  - Every simulation step



GPU accelerated reconstruction in parallel magnetic resonance imaging

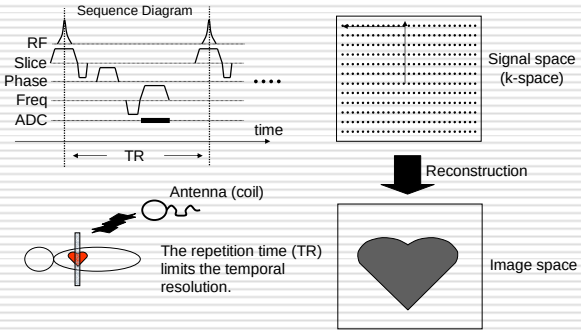
- Applicable to much more than cardiac imaging although this will be our example
- Slides on MR physics courtesy of Michael Schacht Hansen, UCL

The basic imaging experiment



- Spatial resolution and field of view is limited by the number of data points.
- Total acquisition time is determined by:
  - Acquisition time for one data point
  - The ability to acquire multiple data points simultaneously.

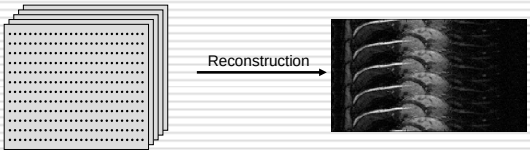
MR acquisition



Important points about TR and total acquisition time

- For simple imaging experiments:
  - Total acquisition time is proportional to TR and spatial resolution.
- Minimum TR is ~3 ms on most systems.
- An example:
  A simple sequence with TR=10ms and an image matrix of 256x256 points takes about 2.56 seconds to acquire.

How can we speed up the MRI experiment?



The purpose of the experiment is to fill the data space (k-space).

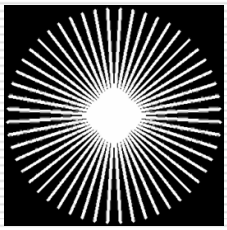
- Speed up the acquisition of each data point
- Acquire fewer data points

SENSE (parallel imaging)

- Undersample k-space
- Record signal from multiple coils
  - Accelerate (undersample) 'n' times
  - Acquire data with 'n' coils
  - Reconstruct image by solving 'n' equation of 'n' unknowns for each image pixel.

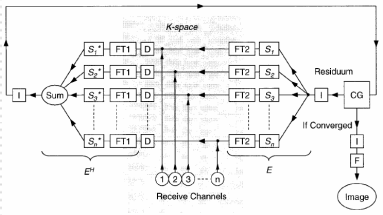
Non-Cartesian imaging

- We can use higher acceleration factors (more undersampling) if samples are not acquired on a Cartesian grid.



Iterative SENSE

- General case (Non-Cartesian sampling)
- For an N by N image
- Solve N^2 equations with N^2 unknowns
  - Linear system A**x**=**b**



Conjugate Gradient

- Each loop consists of an Fourier transform followed by an inverse Fourier transform
- Standard conjugate gradient consists of matrix-vector multiplies, i.e. **discrete Fourier transform**
- *Use a Non-equidistant Fast Fourier Transform instead!*

NFFT^{B} algorithm

Input: Complex frequency domain coefficients f_j corresponding to the non-equispaced samples $\mathbf{x}_j$.

Output: Approximate complex time domain coefficients $\hat{f}_{\mathbf{k}}$ corresponding to the equispaced grid cells $\mathbf{k}$.

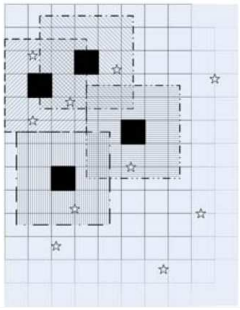
1. **Convolution**. Compute $g_{\mathbf{l}}$.
   Compute the convolution of the complex coefficients f_j at the non-equispaced samples $\mathbf{x}_j$ onto the equispaced grid cells $\mathbf{k}$.
2. **FFT**. Compute $\hat{g}_{\mathbf{k}}$.
   Compute the inverse fast Fourier transform of $g_{\mathbf{l}}$.
3. **Kernel rolloff correction**. Compute $\hat{f}_{\mathbf{k}}$.
   Divide $\hat{g}_{\mathbf{k}}$ with the coefficients of the Fourier transformed convolution kernel.

Memory access pattern

- Parallelization of NFFT, 1st try
- Memory bandwidth
  - $M+2W^dM$
  - M: number of samples
  - W: kernel size
  - d: dimensionality

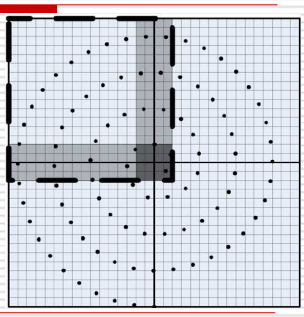
Memory access pattern

- Parallelization of NFFT, 2nd try
- Memory bandwidth
  - $\underline{W^d M+|I_N|}$
  - W: kernel size
  - d: dimensionality
  - M: number of samples
  - $|I_N|$: number of grid cells

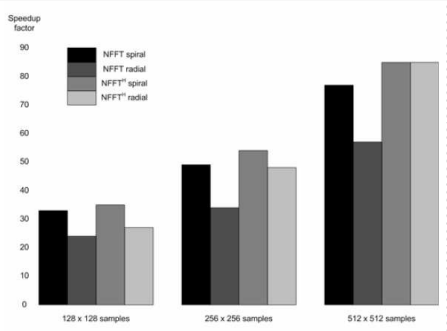


Memory access pattern

- Parallelization of NFFT, 3rd try
- Memory bandwidth
  - $\underline{nM+|I_N|,\ n<<W^d}$
  - W: kernel size
  - d: dimensionality
  - M: number of samples
  - $|I_N|$: number of grid cells

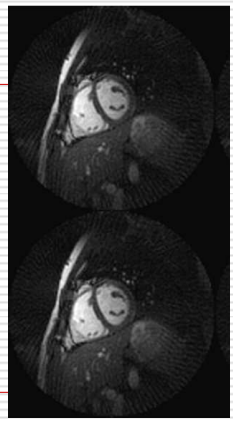


Speedup factor



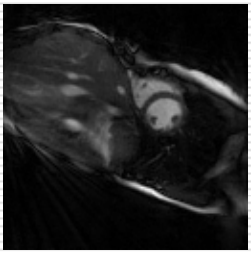
In-vivo example

- Top
  - Reference image, CPU
  - Discrete Fourier transform
- Bottom
  - GPU implementation
  - Oversampling factor 2, kernel size 4



Real-time imaging

- 8-fold undersampling
  - Acquisition time 40 ms per frame
  - Reconstruction time 25 ms per frame
- First time that real-time reconstruction of non-Cartesian *kt*-SENSE have been demonstrated!



Schedule (tentative)

- 09:45-10:00
  - Registration, welcome
- **10:00-11:00**
  - **An introduction to parallel computing on GPUs exemplified by surgical simulation and MRI reconstruction**
- **11:00-11.30**
  - **A first look at Cuda**
- 11.30-13:00
  - Lunch
- 13:00-14:30
  - Cuda in more detail
  - Performance issues
  - Examples from the Cuda SDK
- 14:30-15:00
  - Coffee break
- 15:00-16:30
  - Simple Lattice Boltzmann and cellular automata models in Cuda