IDI – simulation and evaluation

with OpenCL

Fabian Trost

IDI – Background (1/2)

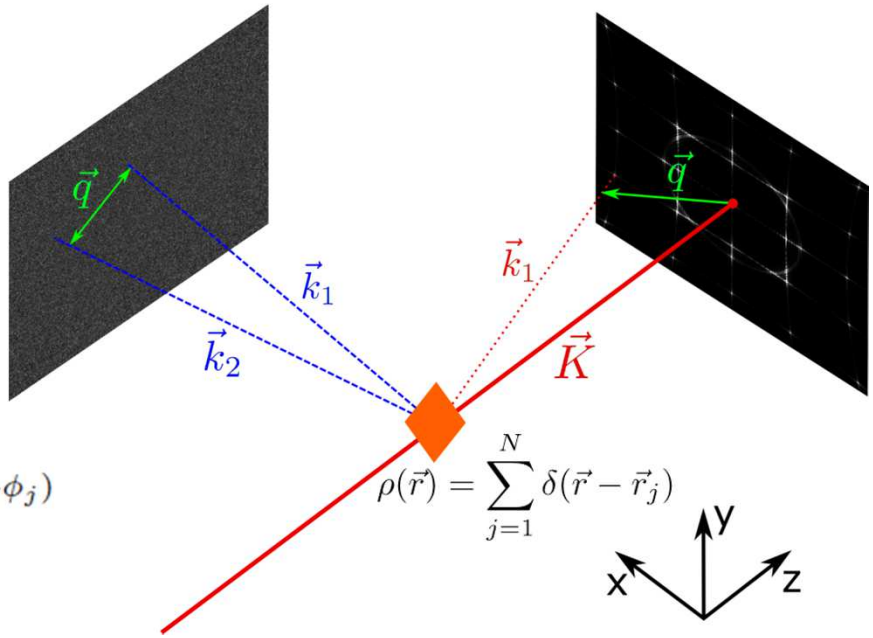
IDI

- q is determined by distance

$$\vec{q} = \vec{k}_1 - \vec{k}_2$$

- wave - function

$$\psi(\vec{k}) = \sum_{j=1}^{N} e^{i(\vec{k}\cdot\vec{r}_j + \phi_j)}$$



CDI

- q is determined by position

$$\vec{q} = \vec{K} - \vec{k}_1$$

- Diffraction pattern

$$I \propto |\mathfrak{F}[\rho(\vec{r})](\vec{q})|^2$$

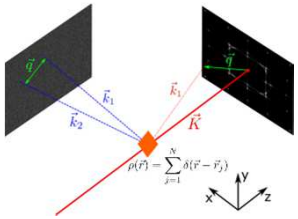
- random phases

$$\phi = [0, 2\pi) \quad \Rightarrow \quad \langle e^{i(\phi_j - \phi_l)} \rangle = \delta_{j,l}$$

- information within time-averaged intensity is lost

$$\langle I(\vec{k}) \rangle = \sum_{j,l=1}^{N} e^{i\vec{k}\cdot(\vec{r}_j - \vec{r}_l)} \langle e^{i(\phi_j - \phi_l)} \rangle = N$$

IDI – Background (2/2)



IDI

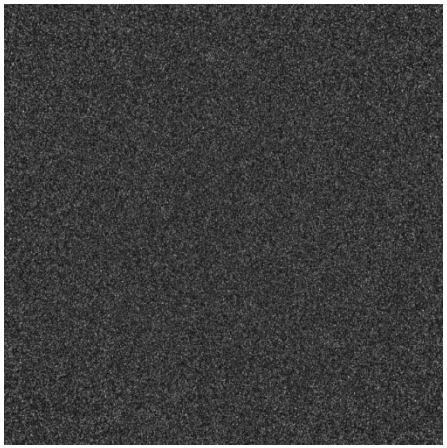
- Intensity correlation

$$\langle I(\vec{k}_1), I(\vec{k}_2)\rangle = \sum_{j,j',l,l'=1}^{N} (1-\delta_{j,j',l,l'}) \left(e^{i(\vec{k}_1(\vec{r}_j-\vec{r}_l)+(\vec{k}_2(\vec{r}_{j'}-\vec{r}_{l'}))} \underbrace{\langle e^{i(\varphi_j-\varphi_l+\varphi_{j'}-\varphi_{l'})}\rangle}_{\text{only} \neq 0 \text{ for } (j=l' \wedge l=j') \vee (j=l \wedge j'=l')} \right),$$

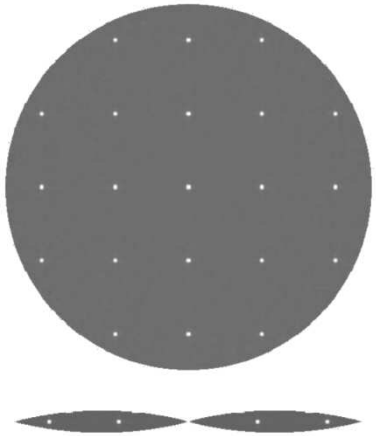
- thus we obtain:

$$\left\langle I(\vec{k}_1), I(\vec{k}_2)\right\rangle = (N^2-N) + \sum_{j\neq l}^{N} e^{i(\vec{k}_1-\vec{k}_2)(\vec{r}_j-\vec{r}_l)} = (N^2-2N) + \left| \sum_{j}^{N} e^{i\vec{q}\cdot\vec{r}_j} \right|^2$$

CDI



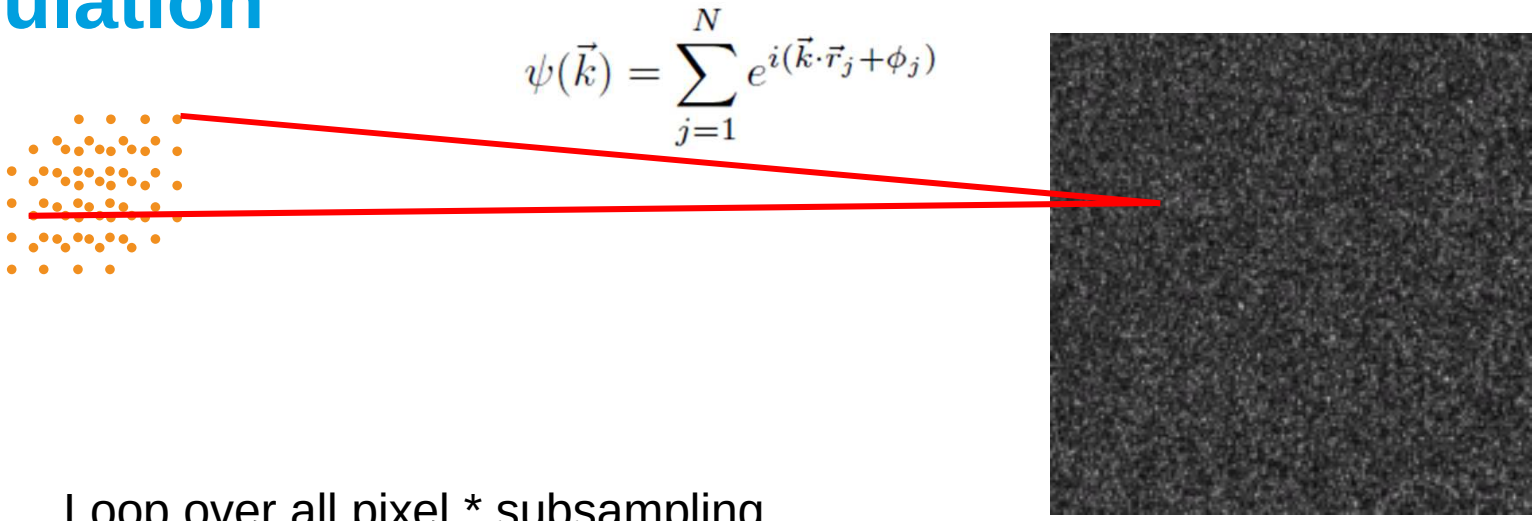
Auto-correlation



- 3D information within single exposure

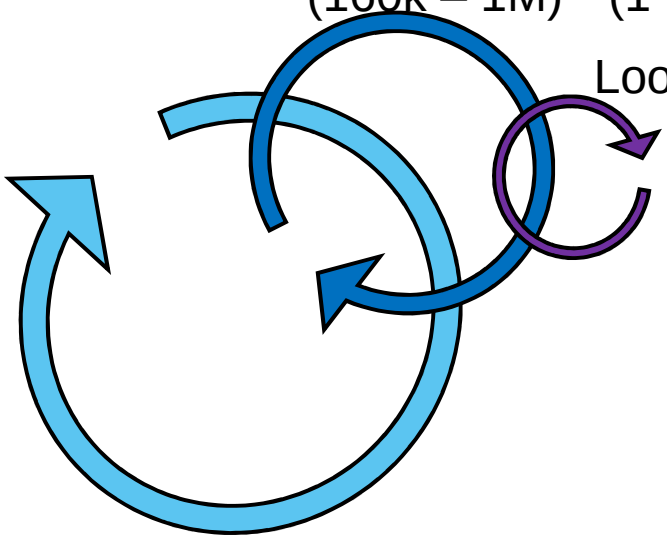
Simulation

$$\psi(\vec{k}) = \sum_{j=1}^{N} e^{i(\vec{k}\cdot\vec{r}_j+\phi_j)}$$



Loop over all pixel * subsampling
(160k – 1M) * (1 to 25)

Loop over all emitter
(10^3 – 100^3)



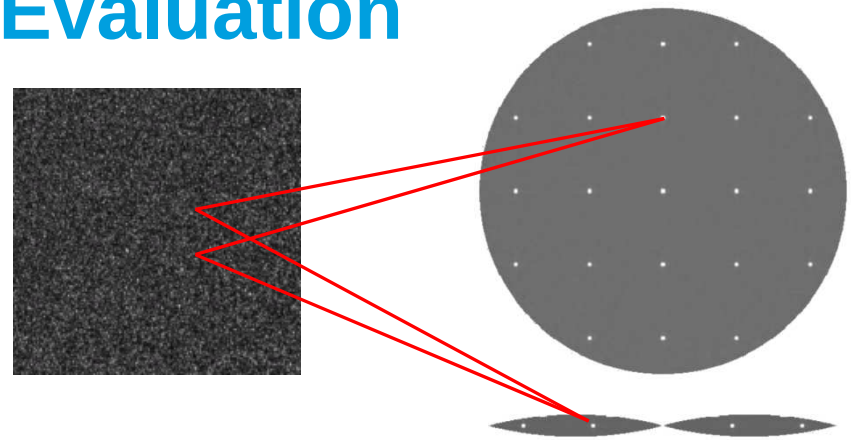
Simulate around
1000 – 10000 pattern

Implementation:

- one OpenCl – thread for each pixel
- runtime ~ pixel * emitter -> linear
  ~ $O(10^{15\ +/-\ 2})$
- Requires double precision but no atomics
- required memory < 1GB

=> no problem at all with current setup
but faster is always better.

Evaluation



Autocorrelation:
- Correlate each pixel with each pixel
- O(1/2 n^2 – 1)
- For 1 megapixel ~ 5 * 10^{11} Operations
- Results are stored in 3d - array

Implementation autocorrelation:
- one OpenCl – thread for each pixel
- runtime ~ O(n^2) n is the number of pixel
- required memory dependent on result container shape up to 3.5GB (1000*1000*500*4 ~ 1.9GB; 1200*1200*600*4 ~ 3.2GB)

Rotate and merge:
- Rotate whole container and add to an second one
  needs to be performed for each pattern > 100 000 times
  -> need for double precision
- Runtime ~ O(n) n is the number of voxels * pattern
- Memory needed: one small container (32bit) one big container (64 bit) approx. up to 10GB

Examples

LCLS LU56, fluorescing zinc (MOF)

- approx. 145000 images
  (very bright)
- Detector ePix 704x768 = 0.54MPix
- Time to correlate all patterns 9.8 days
  (with one Tesla K20Xm)
- Required GPU memory ~ 3GB

LCLS LR17, fluorescing iron (Hemoglobin)

- approx. 120000 images
  (quite dark)
- Detector Jungfrau 1024x1024= 1MPix
- Time to correlate all patterns 3 days
  (with one Tesla K20Xm and CPU)
- Required GPU memory ~ 5.5GB
  (cropped output container)

Current situation

Maxwell max-cfelg (4 + 2 nodes) with 2 x Tesla K20Xm each

- Supports only up to OpenCL 1.2
  -> not as fun to debug

- No hardware implementation of atomic add for floats / doubles
  -> requires work arounds

- Only 5.7 GB memory
  -> restricts the size of the output volume