Using Eigen to Sparse Matrix Solutions

Why, What, and How?

Why?

Eigen (C++ Algebra framework)
introduced some time ago a module for sparse matrices.
With version 3.2 (last month) it was released as a „stable release“.

Now, lets see what one can do with it….

Why-2?

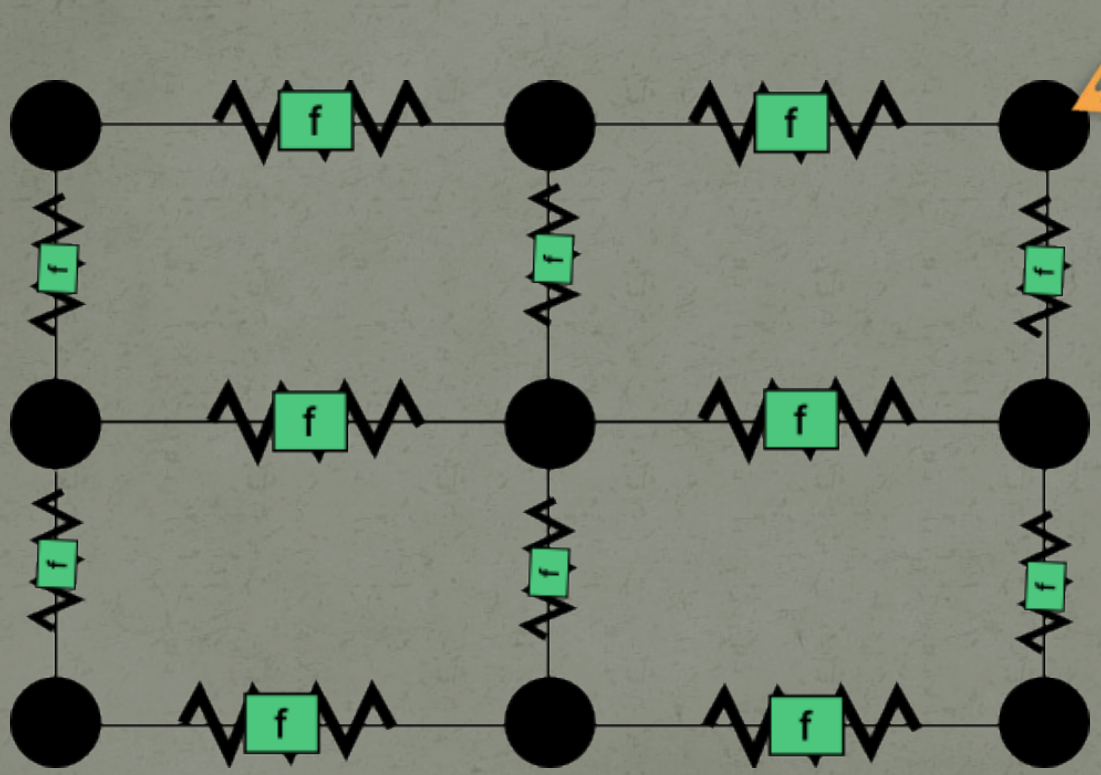
Input:
20% of all pixels are set.

Output:
Full reconstruction

Source: http://www.**wired**.com/magazine/2010/02/ff_algorithm/

What?

- Modelnig neighbours with graphs

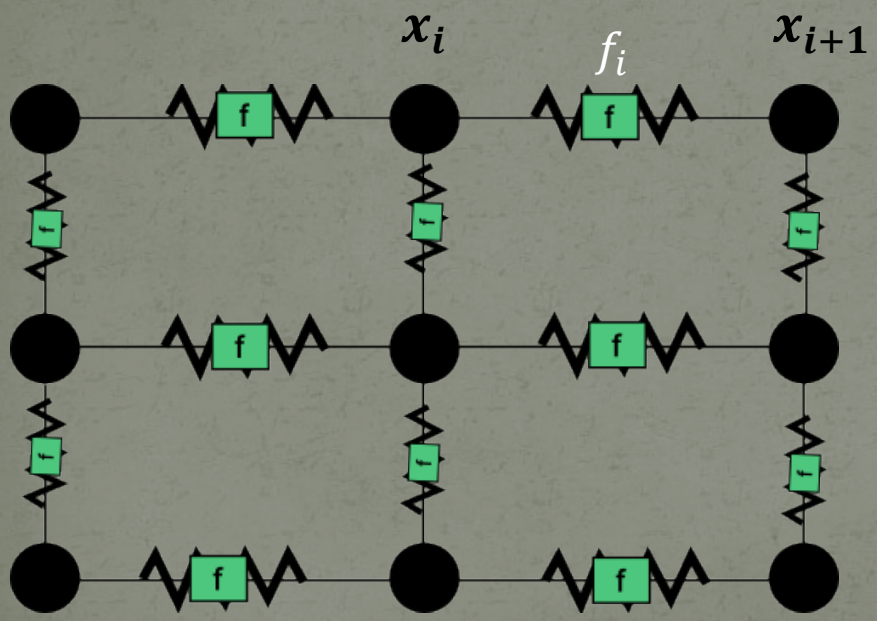


Pixel Value

„Spring“ between values

What?

- Called „Factor Graph“.
- Need to model the „factor functions“ f_i



$$f_i = \| x_i - x_{i+1} \|_2^2$$

$$\| x_i - x_j \|_2^2 = (x_i - x_j)^T (x_i - x_j)$$

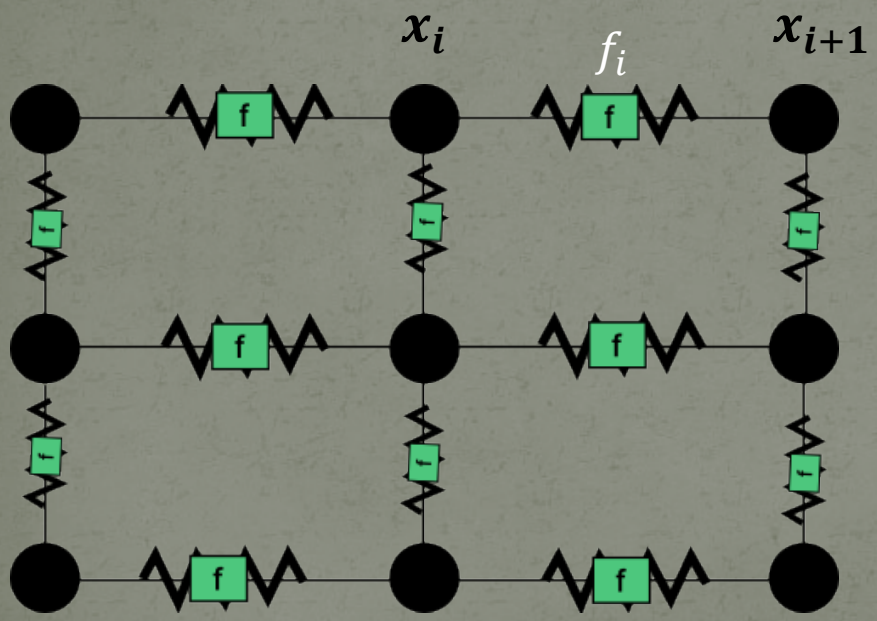
$$= x_i^T x_i - 2 x_i^T x_j + x_j^T x_j$$

$$= (x_i, x_j)^T \begin{pmatrix} 1 & -2 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix}$$

Build a cost function: $\sum_i f_i = x^T P x$, with $x = (x_1, x_2, \cdots, x_n)^T$

What?

- Consider the measurements $\hat{x}_i$



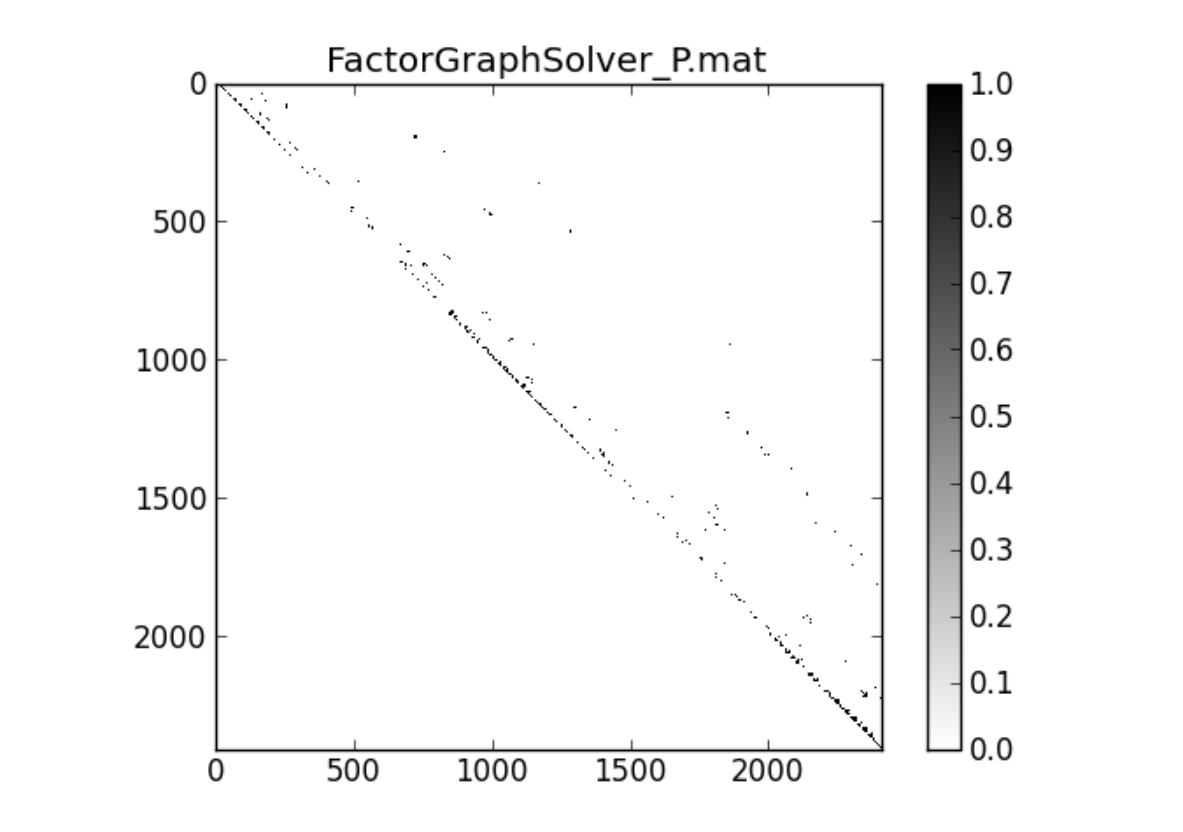
$$m_i = \| x_i - \hat{x}_i \|_2^2$$

$$\| x_i - x_j \|_2^2 = (x_i - x_j)^T (x_i - x_j)$$

$$= x_i^T x_i - \underbrace{2 x_i^T \hat{x}_i}_{c^T x_i} + \hat{x}_i^T \hat{x}_i$$

Total cost: $\sum_i f_i + m_i = x^T \tilde{P} x + c^T x$, with $x = (x_1, x_2, \cdots, x_n)^T$

Matrix P



$$x^{opt} = -(P + P^T)^{-1} c$$

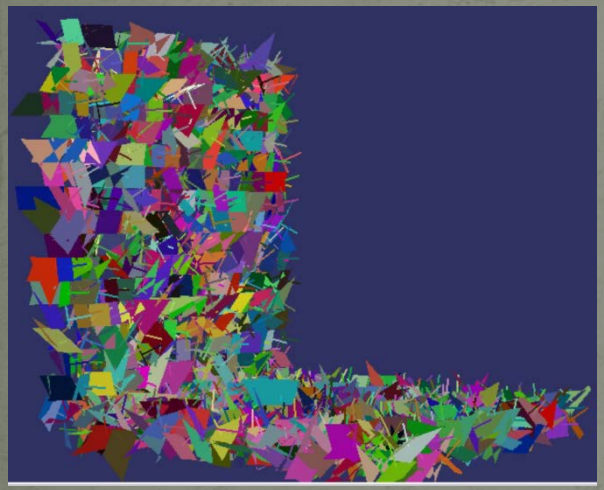
Results: Filling Missing Data

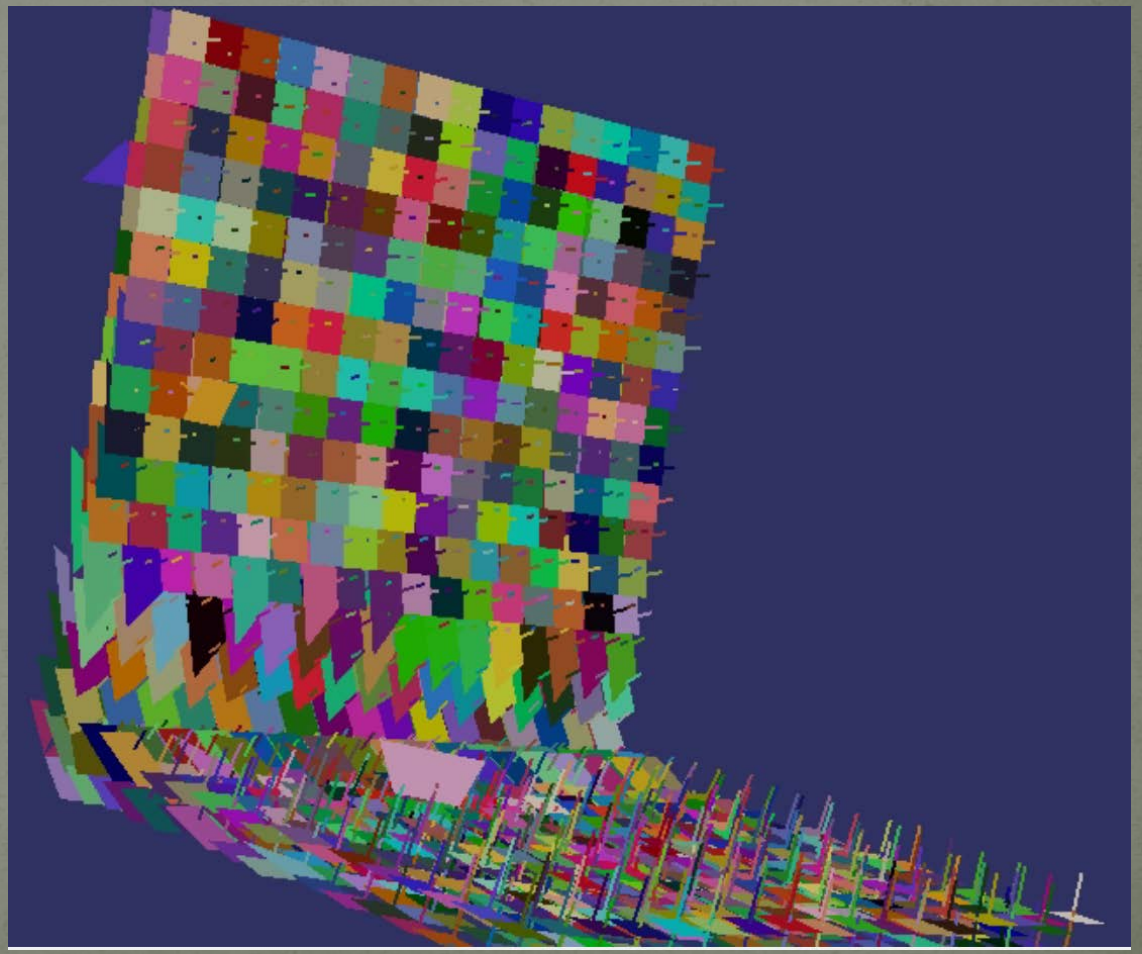
Input

Filtered

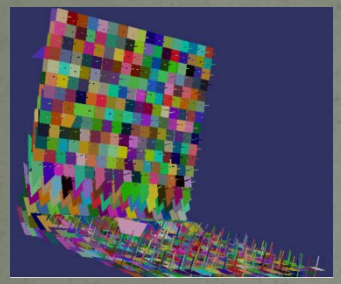
Original

Results: 3D filtering





Results: 3D filtering



- P of size: 3700 x 3700
- Direct inversion: 5 sec
- Eigen Optimized QR: 1 sec
- Eigen Sparse LLT: **4 ms** (!)

- P of size 22500 x 22500
- Duration Eigen-LLT: **5sec**