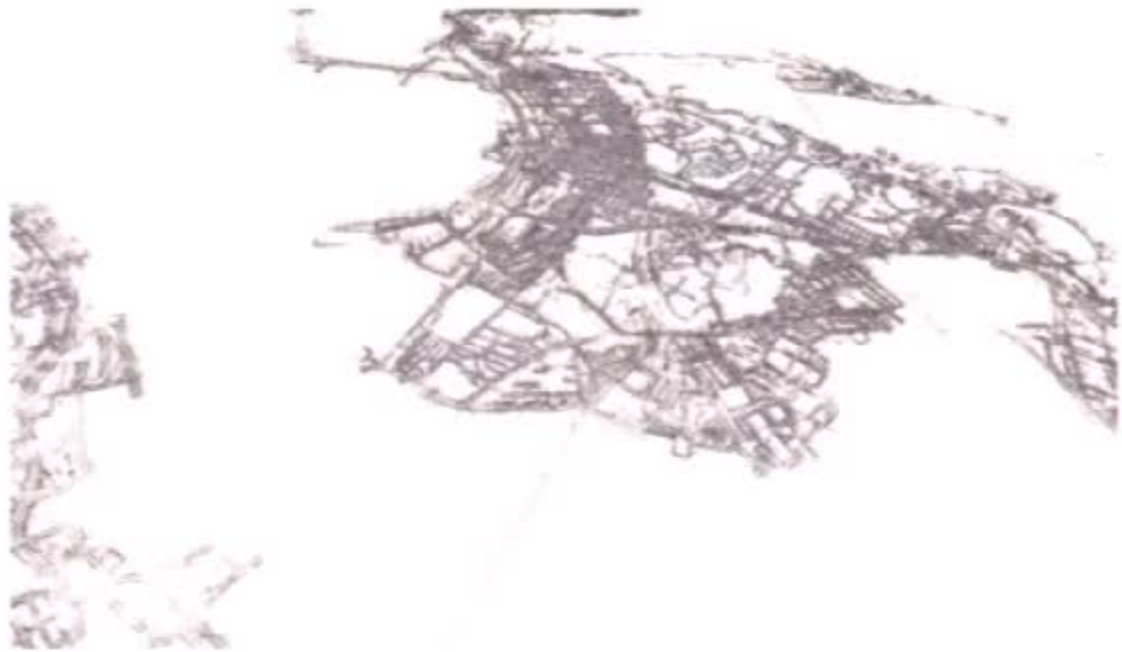
Solving NNLS Problems In Computer Vision

Sameer Agarwal, Google Inc.

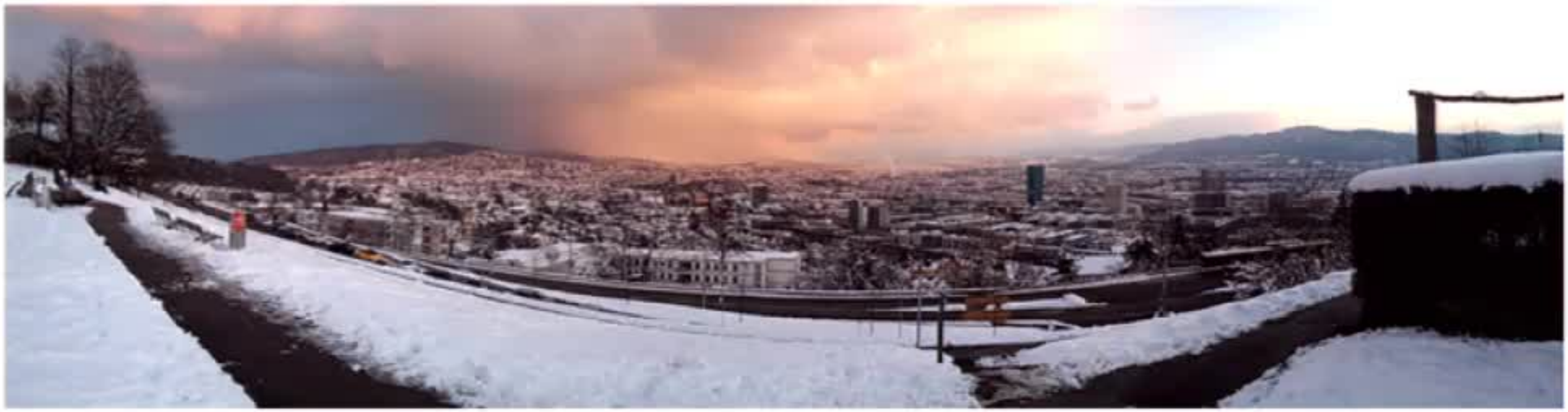
Street View 3D Reconstruction



Klingner, Martin & Roseborough

Mesh Smoothing

Photosphere Panorama Stitching



Running on your Android phone right now!

LensBlur

Photo without Lens Blur

Photo by Sascha Haeberling

Photo with Lens Blur

Nonlinear Least Squares

$$\min_x \frac{1}{2}\|F(x)\|^2 \qquad F(x) = [f_1(x), \ldots, f_n(x)]^\top$$

$$F(x + \Delta x) \approx F(x) + J(x)\Delta x \qquad J_{ij}(x) = \frac{\partial f_i(x)}{\partial x_j}$$

$$\min_{\Delta x} \frac{1}{2}\|F(x) + J(x)\Delta x\|^2 + \mu\|D(x)\Delta x\|^2$$

If $\|F(x + \Delta x)\| < \|F(x)\|$ then

$x = x + \Delta x, \qquad \mu = \mu/2$

else

$\mu = 2\mu$

Levenberg-Marquardt

Block Diagonal Preconditioners

$$H_\mu = [J^\top J + \mu D^\top D] = \begin{bmatrix} B & E \\ E^\top & C \end{bmatrix}$$

$$\begin{bmatrix} B & E \\ E^\top & C \end{bmatrix} \begin{bmatrix} \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} v \\ w \end{bmatrix}$$

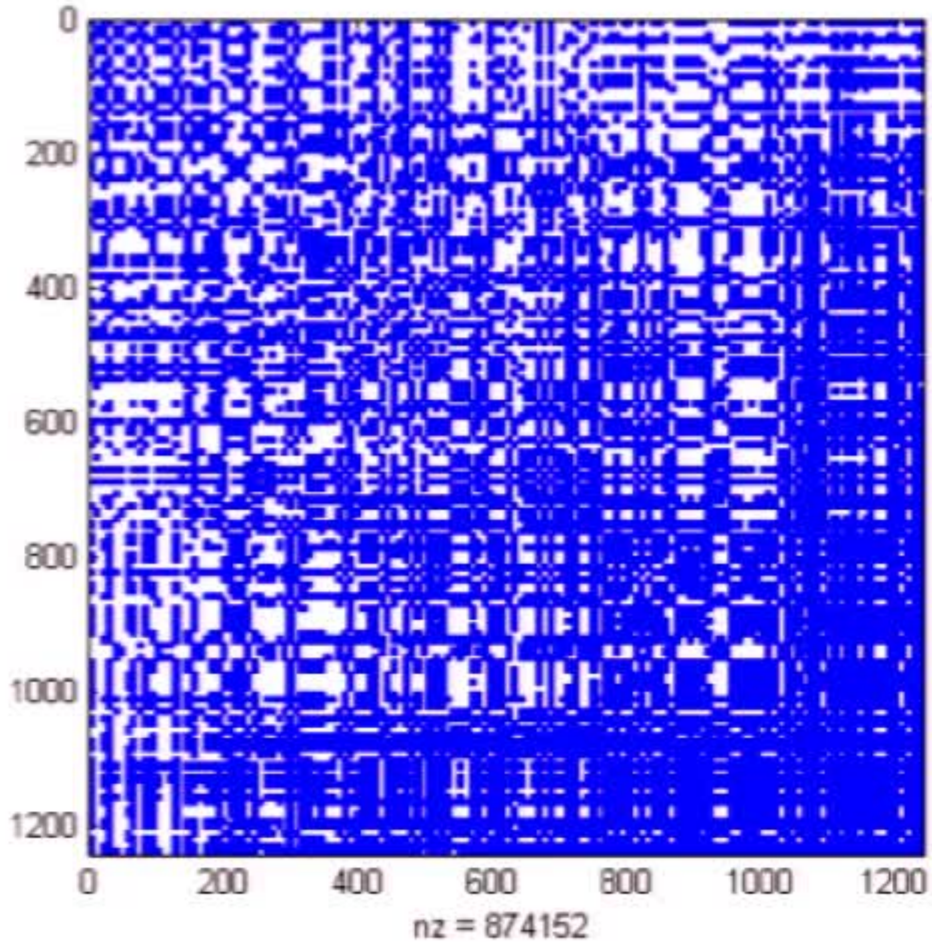
$$\underbrace{\boxed{\left[B - EC^{-1}E^\top\right]}}_{S_\mu} \Delta y = v - EC^{-1}w$$

S_μ : Schur Complement of C in H

Two simple choices:

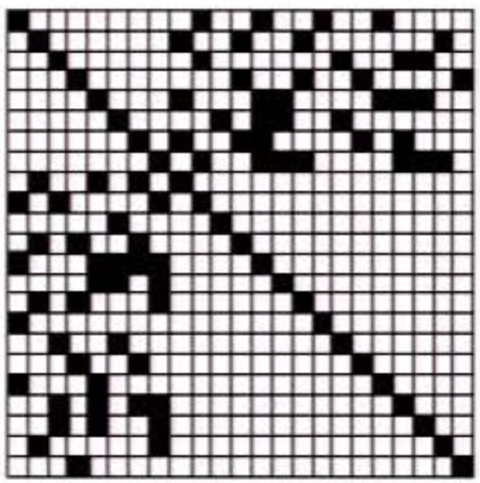
1. B
2. Block Diagonal of S_μ

Can we do better?


0
200
400
600
800
1000
1200
0
200
400
600
800
1000
1200
nz = 874152

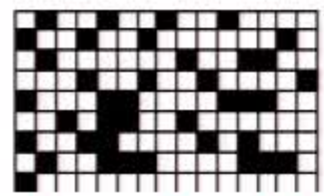
Visibility Based Clustering

$$J'J = $$



$$S = B - EC^{-1}E^\top$$

$$V = $$

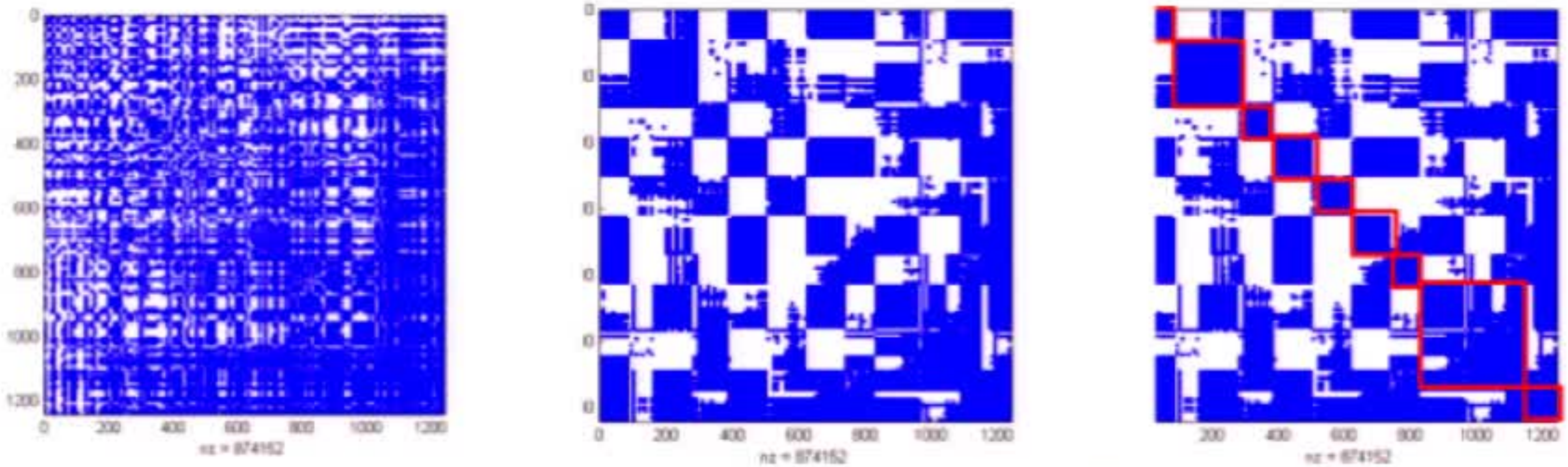


v_i: Binary vector of point visibilities.

$$\max \sum_{i \in I} \max_{j \in C} \frac{v_i^\top v_j}{||v_i||||v_j||} - \alpha|C|$$

- Assumes that "similar" cameras are tightly coupled.
- Clustering is independent of the numerical entries in S.
- Objective richer than just 0-1 pattern of S.

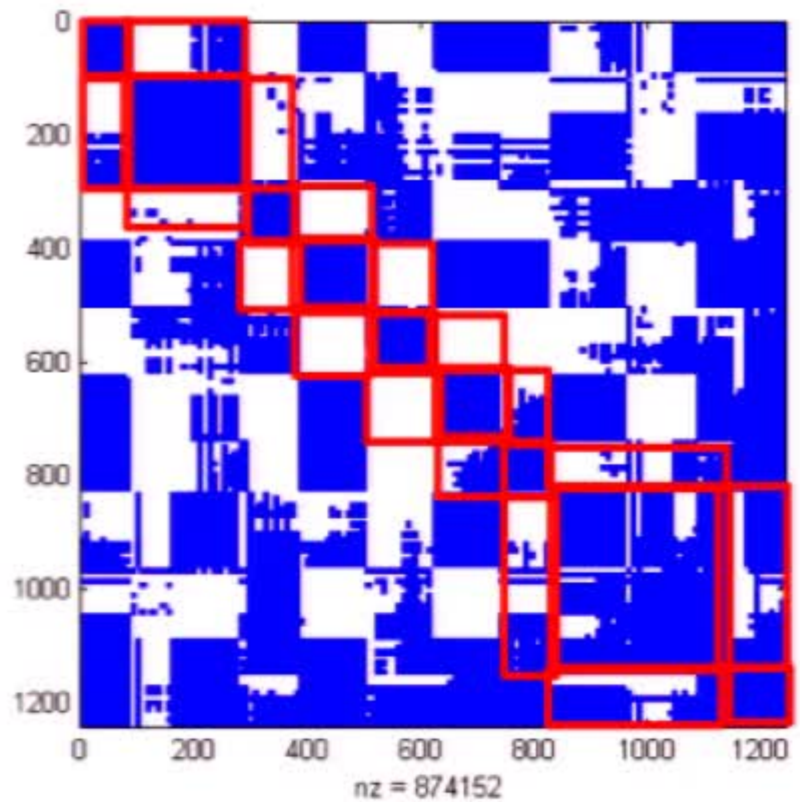
Cluster-Jacobi



- Cluster cameras using visibility.

Can we do better still?

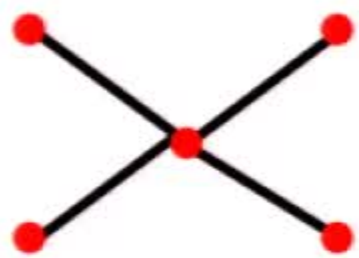
- Permute rows and columns of S so that cameras in a cluster are together
- Extract diagonal blocks corresponding to clusters.



Not a particularly useful band diagonal!

We need another permutation!

Degree-2 MST

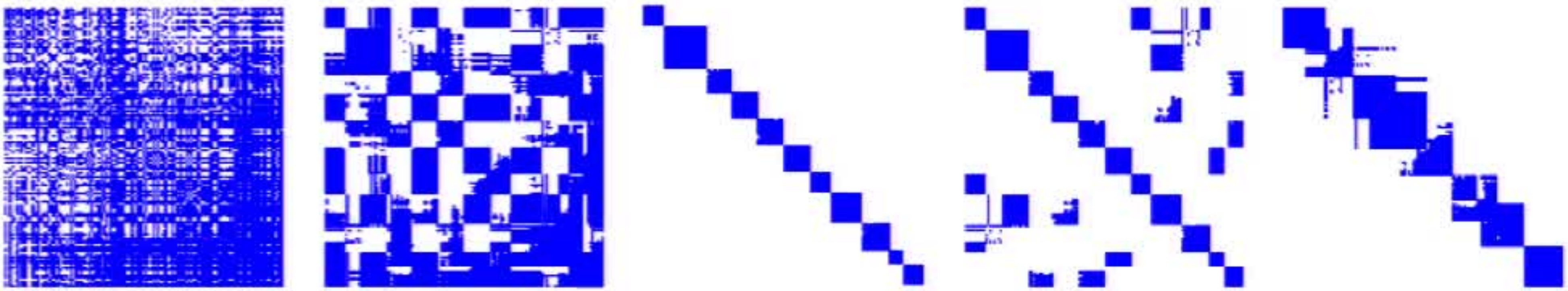


Degree-2 Maximum Spanning Trees don't always exist.

Even if they did, finding them is NP-Hard.

So, we will use greedy Kruskal's algorithm to find
Approximate Degree-2 Maximum Spanning Forests

Cluster-Tridiagonal



S | $S_1 = P_1 S P_1^\top$ | cluster-jacobi | *degree* 2 tree | cluster-tridiagonal

1. Cluster cameras using visibility & permute.
2. Use greedy Kruskal's algorithm to find degree 2 maximum spanning tree.
3. Permute the S by traversing the tree.
4. Extract block tridiagonal M.
5. **If M is indefinite, scale off diagonal blocks by 1/2.**