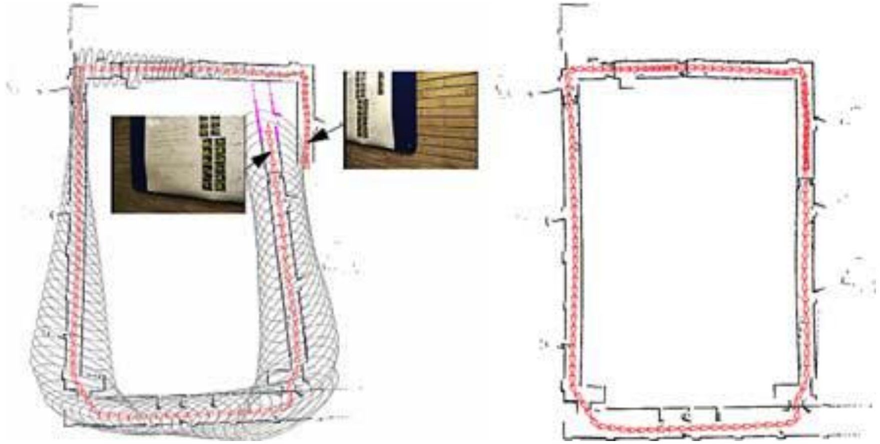
Julia SVDS for Loop Closure

Timmy Galvin

Loop Closure

Loop Closure

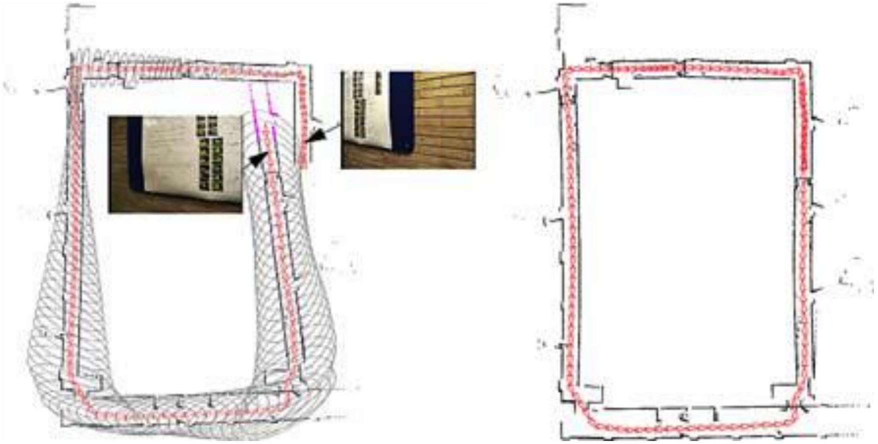
- Fundamental problem in navigation – computation/storage



Paul Newman, Research in Mobile Robotics. Oxford Mobile Robotics Research Group. 2005. http://www.soue.org.uk/souenews/issue4/mobilerobots.html

Loop Closure

- Fundamental problem in navigation – computation/storage



- Place recognition for a previously visited location
  - Simultaneous Localization and Mapping (SLAM)
- Place recognition for an external database

Paul Newman, Research in Mobile Robotics. Oxford Mobile Robotics Research Group. 2005. http://www.soue.org.uk/souenews/issue4/mobilerobots.html

Loop Closure Technique

- 1. Extract image features

Loop Closure Technique

- 1. Extract image features

- 2. Generate a scene descriptor
  - Based on vocabulary of features
  - Can be extremely sparse

$z = (w_1, 0, \ldots, 0, w_2, 0 \ldots 0, \ldots.)$

Loop Closure Technique

- 1. Extract image features

- 2. Generate a scene descriptor
  - Based on vocabulary of features
  - Can be extremely sparse

↓

$z = (w_1, 0, \ldots, 0, w_2, 0 \ldots 0, \ldots.)$

- 3. Find images with high similarity

↓

$\mathrm{dot}(z_1, z_2) > \mathrm{threshold}$

Loop Closure with Scene Sequences

- Use sequences of scenes rather than individual scenes

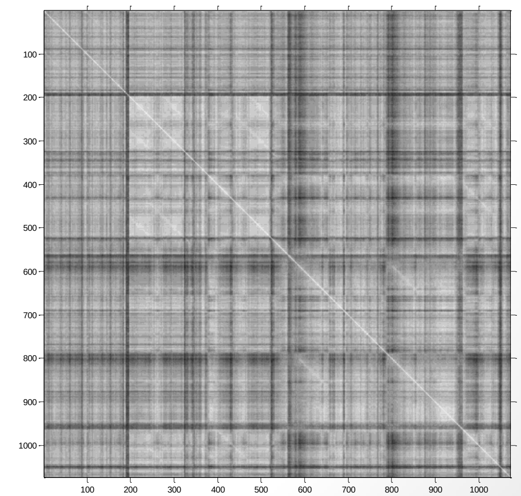
Loop Closure with Scene Sequences

- Use sequences of scenes rather than individual scenes
- Compute a similarity matrix

$$S(i,j) = \frac{\sum_{i=1}^{|\nu|} z_i z_j}{\sqrt{\sum_{i=1}^{|\nu|} z_i^2}\sqrt{\sum_{i=1}^{|\nu|} z_j^2}}$$

$$z_i = [0\ 1\ 0\ 0\ 1\ 1\ 0\ \ldots]$$

$$z_j = [0\ 0\ 1\ 0\ 0\ 1\ 1\ \ldots]$$



Loop Closure with Scene Sequences

- Use sequences of scenes rather than individual scenes

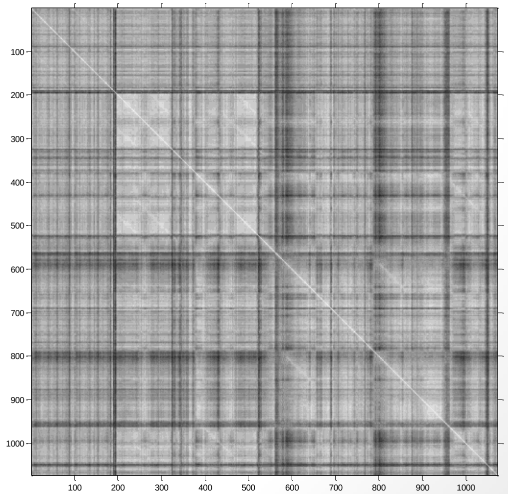
- Compute a similarity matrix

$$S(i,j) = \frac{\sum_{i=1}^{|\nu|} z_i z_j}{\sqrt{\sum_{i=1}^{|\nu|} z_i^2}\sqrt{\sum_{i=1}^{|\nu|} z_j^2}}$$

$$z_i = [0\ 1\ 0\ 0\ 1\ 1\ 0\ ...]$$

$$z_j = [0\ 0\ 1\ 0\ 0\ 1\ 1\ ...]$$

- Find local sequences (off-diagonal traces)
  - Modified Smith-Waterman algorithm



Loop Closure with Scene Sequences

- Use sequences of scenes rather than individual scenes

- Compute a similarity matrix

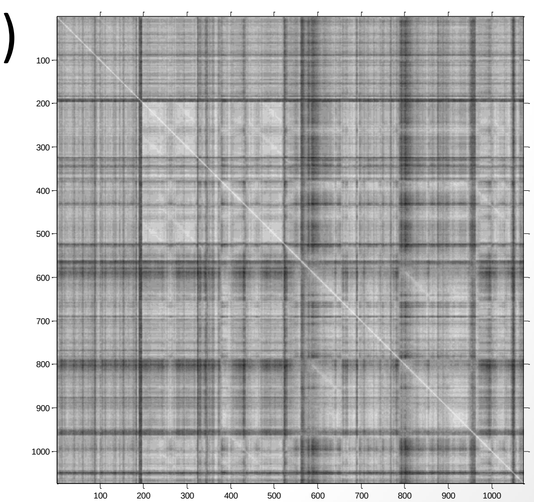
$$S(i,j) = \frac{\sum_{i=1}^{|\nu|} z_i z_j}{\sqrt{\sum_{i=1}^{|\nu|} z_i^2}\sqrt{\sum_{i=1}^{|\nu|} z_j^2}}$$

$$z_i = [0\ 1\ 0\ 0\ 1\ 1\ 0\ ...]$$

$$z_j = [0\ 0\ 1\ 0\ 0\ 1\ 1\ ...]$$

- Find local sequences (off-diagonal traces)
  - Modified Smith-Waterman algorithm

- Problem: rectangular pattern due to dominant features (common mode similarity) – false positives



Dominant Features

K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *International Journal of Computer Vision,* vol. 74, pp. 261-286, 2007.

Dominant Features

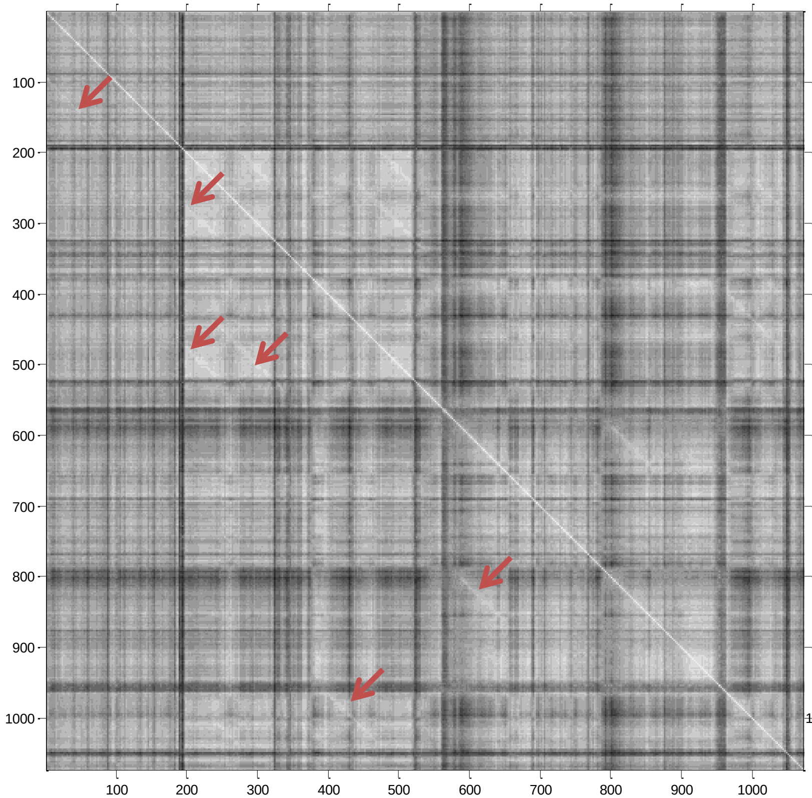
- To remove dominant features → rank reduction
- Using singular value decomposition:

$$S' = \sum_{i=r^*}^{n} u_i \lambda_i v_i^T \qquad r^* = \arg\max_r H(M,r)$$

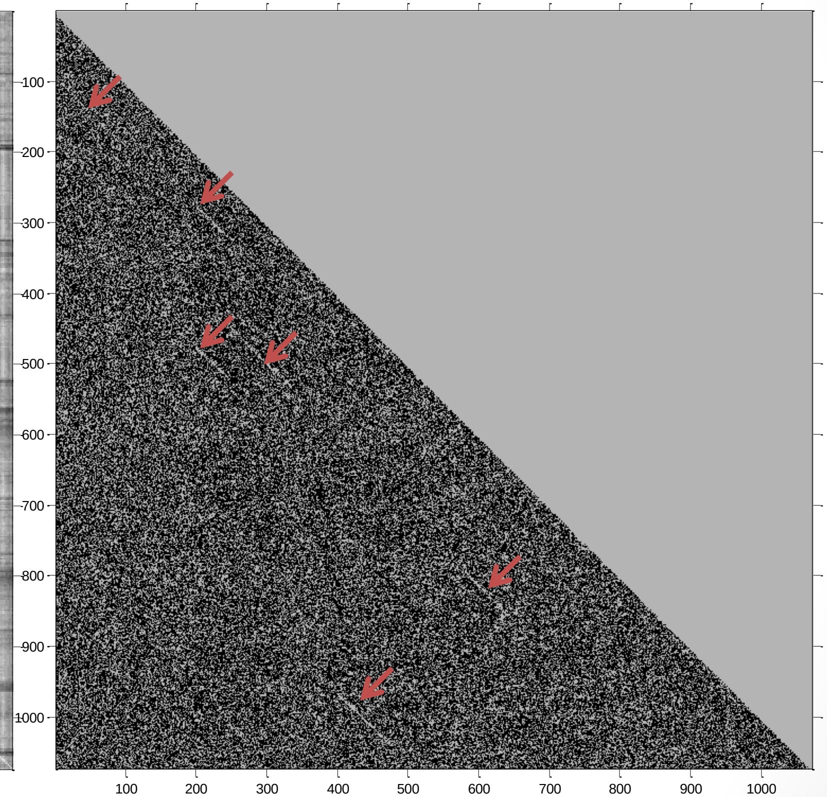
$$H(M,r) = \frac{-1}{\log(n)} \sum_{k=r}^{n} \rho(k,r) \log(\rho(k,r)) \qquad \rho(i,r) = \frac{\lambda_i}{\sum_{k=r}^{n} \lambda_k}$$

K. L. Ho and P. Newman, "Detecting loop closure with scene sequences," *International Journal of Computer Vision,* vol. 74, pp. 261-286, 2007.

Reduction

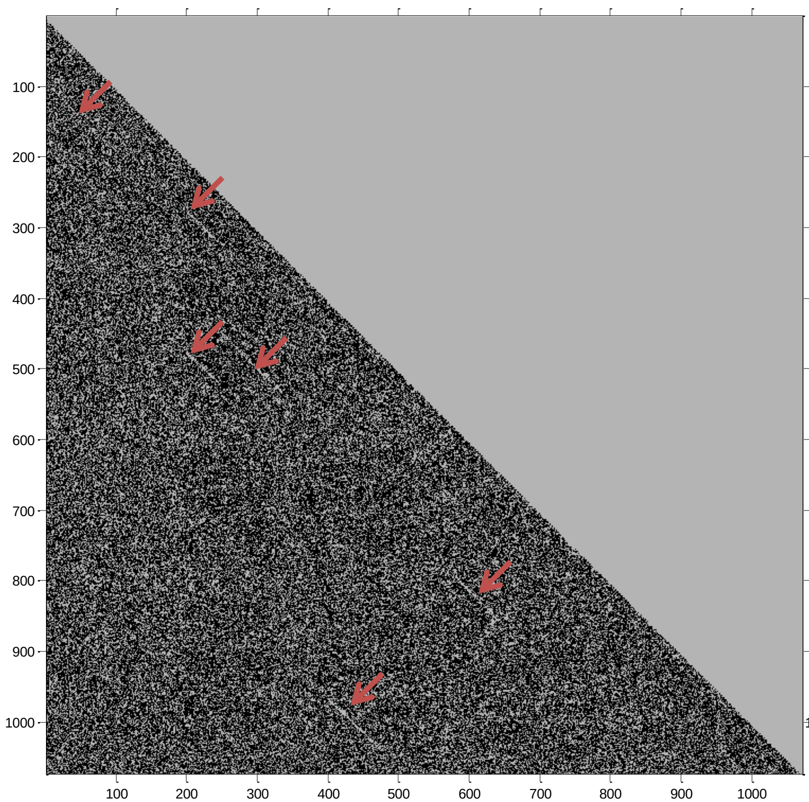


Similarity matrix

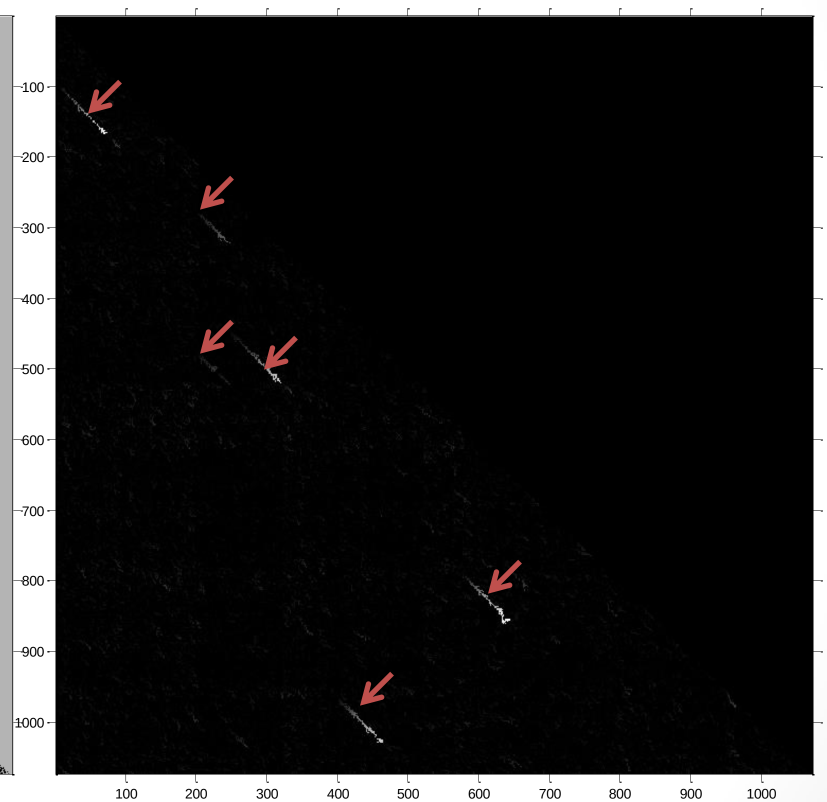


After rank reducing

Smith-Waterman



After rank reducing



After Smith-Waterman

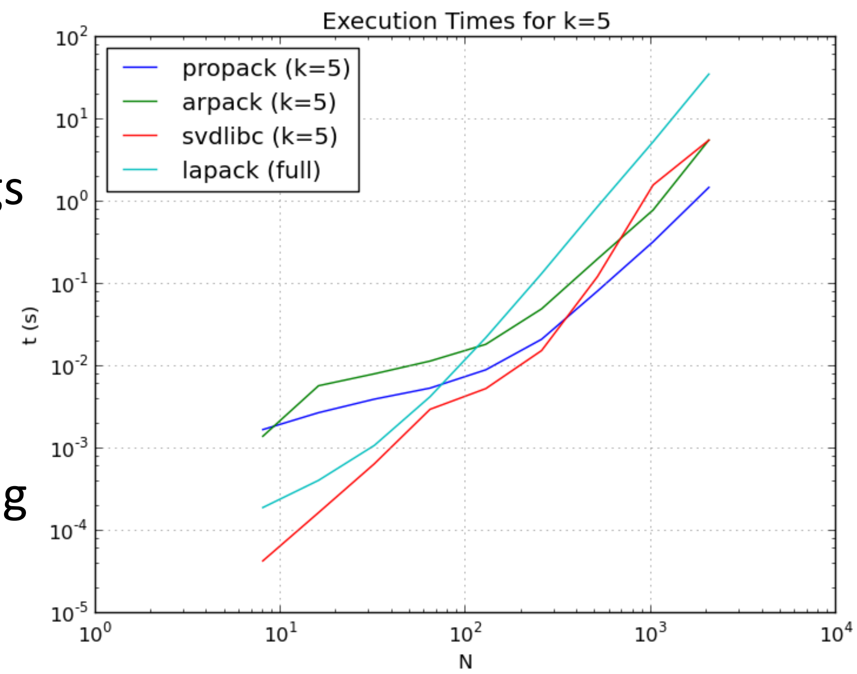
Towards Real-time

- SVD: huge bottleneck
  - 60% of the step-by-step run-time
  - Scales poorly as scene size increases

- Replace with SVDS
  - Calculates k most significant singular values/vectors
  - Better performance for large, sparse matrices
  - Singular value tolerance can be specified

SVDS and Julia

- Available SVDS
  - ARPACK – manipulating eigs on Hermitian matrix $A^{\mathsf{T}}A$
  - PROPACK – using Golub-Kahan-Lanczos (GKL) with implicit restarting



- Julia SVDS in the works:
  - Currently at basic GKL bidiagonalization
  - Part of a larger IterativeSolvers package effort
    - Provide Julia implementation of ARPACK methods

Golub-Kahan-Lanczos

- Decompose *A* iteratively into:

$$A = PBQ^*$$

- Yielding a bidiagonal *B:*

$$B_n = \begin{bmatrix} \alpha_1 & \beta_1 & & \\ & \ddots & \ddots & \\ & & \alpha_{n-1} & \beta_{n-1} \\ & & & \alpha_n \end{bmatrix}$$

- Perform SVD of *B:*

$$B = X\Sigma Y^*$$

- *A* is now decomposed as:

$$A = PX\Sigma Q^* Y^* = U\Sigma V^*$$

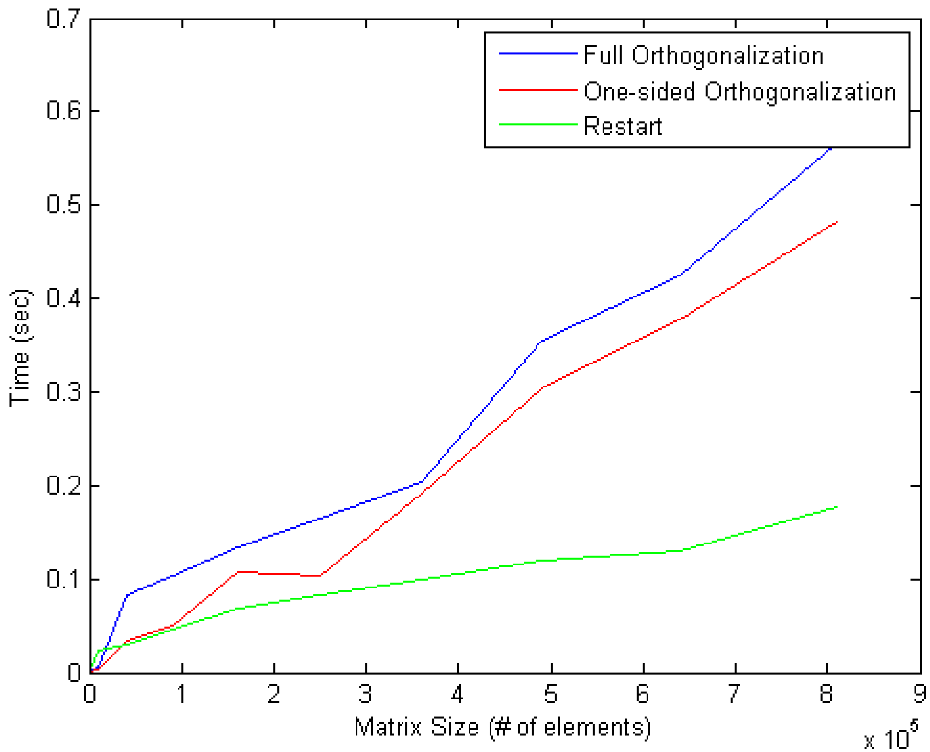
Problems with GKL

- Loss of orthogonality of left Lanczos vectors (P) and of right right Lanczos vectors (Q) through iterations

- Solutions
  - Full orthoginalization
    - High computational cost
    - Cost grows as the iterative method proceeds
  - Partial orthoginalization
    - Perform corrections when orthogonality drops below a threshold
  - Restarting
    - Restarts the computation after a fixed number of iterations to limit the number of Lanczos steps (and size of P and Q)

Performance

- Calculating 10 most significant singular values of matrices with sparsity between .01 and .1
- Comparison of different modifications to GKL



Summary

- Reimplemented versions of Golub-Kahan-Lanczos bidiagonalization for calculation of the partial SVD
  - Orthogonalization methods
  - Restarting

- To be done:
  - Cleaned up and optimized
  - Further tested against existing implementations
  - Comparison of restart methods (implicit versus thick)