Progressive Multigrid Eigensolvers for Multiscale Spectral Segmentation

Michael Maire[1,2] and Stella X. Yu[3]

[1]TTI Chicago [2]California Institute of Technology
[3]University of California at Berkeley / ICSI

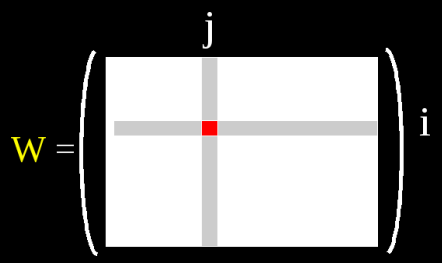
Spectral Segmentation

Build a weighted graph G = (V, E, W) from the image

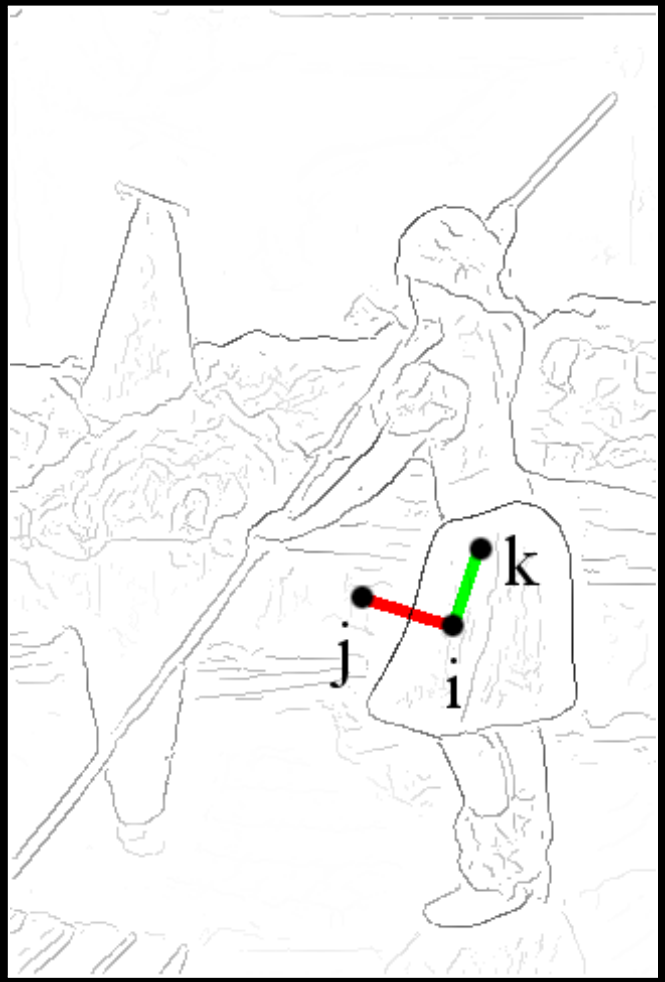
Spectral Segmentation

Build a weighted graph G = (V, E, W) from the image

- Define W using Intervening Contour



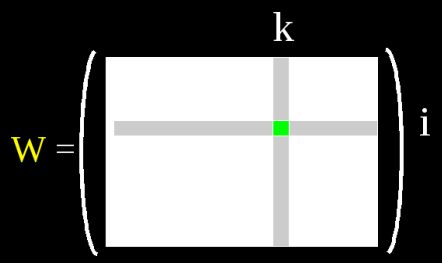
(i, j) low affinity



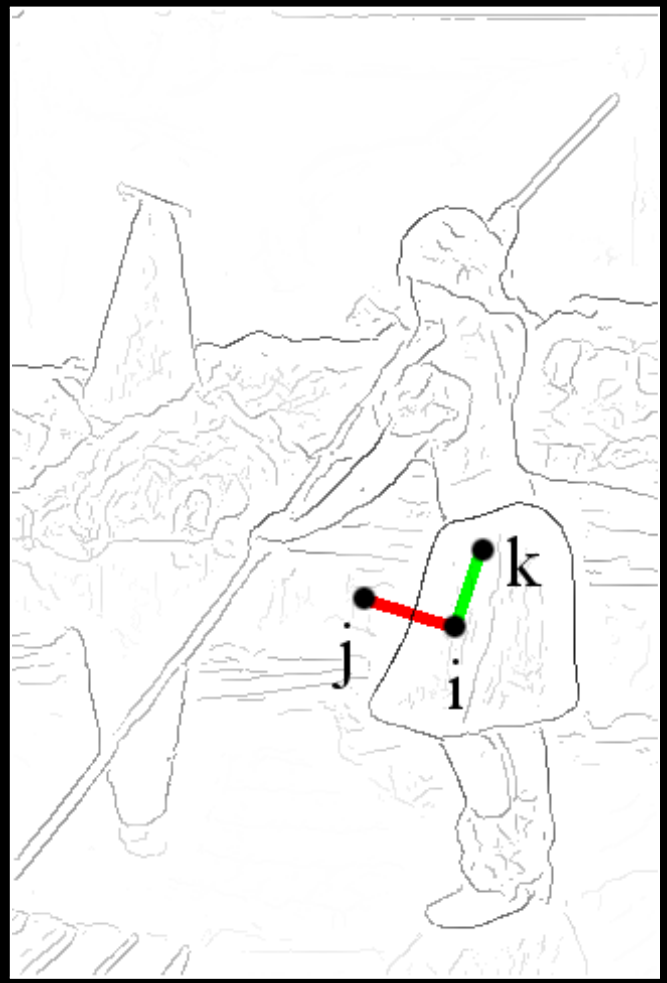
Spectral Segmentation

Build a weighted graph G = (V, E, W) from the image

- Define W using Intervening Contour



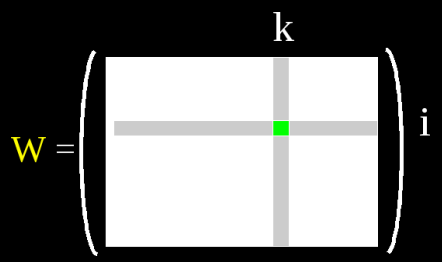
(i, k) high affinity



Spectral Segmentation

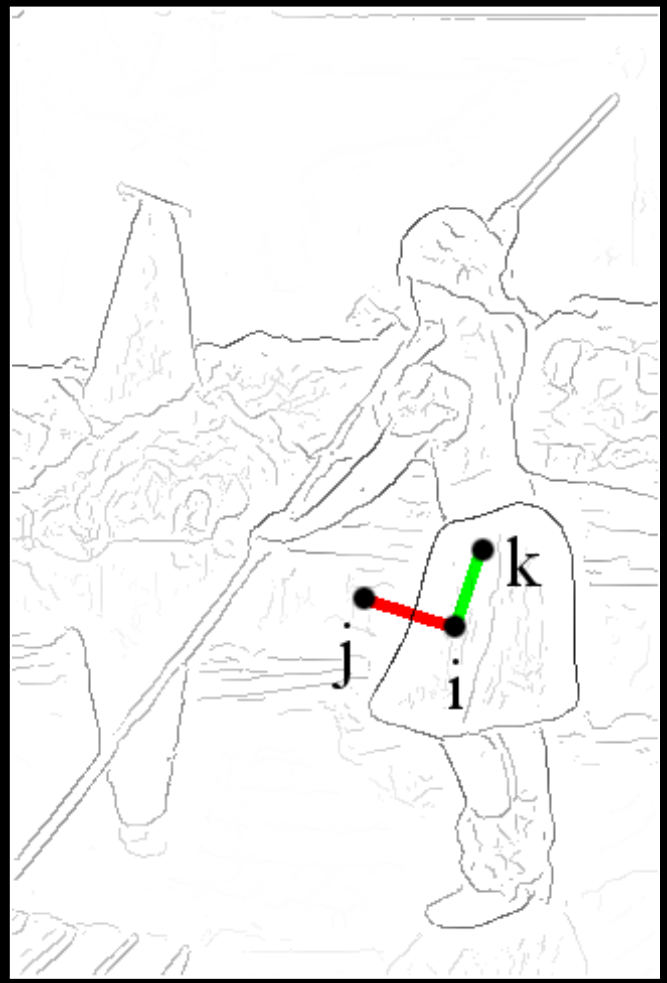
Build a weighted graph G = (V, E, W) from the image

- Define W using Intervening Contour



(i, k) high affinity

- Normalized Cuts
  [Shi & Malik 1997]



Normalized Cuts

- Graph $G = (V, E, W)$
- Split into A, B disjoint, $A \cup B = V$

Normalized Cuts

- Graph $G = (V, E, W)$
- Split into A, B disjoint, $A \cup B = V$

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

$$assoc(A, V) = \sum_{u \in A, v \in V} w(u, v)$$

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

Normalized Cuts

- Graph $G = (V, E, W)$
- Split into A, B disjoint, $A \cup B = V$

$$cut(A, B) = \sum_{u \in A, v \in B} w(u, v)$$

$$assoc(A, V) = \sum_{u \in A, v \in V} w(u, v)$$

$$Ncut(A, B) = \frac{cut(A, B)}{assoc(A, V)} + \frac{cut(A, B)}{assoc(B, V)}$$

- General case: partition using smallest eigenvectors of

$$(D - W)z = \lambda Dz$$

where $D_{ii} = \sum_j W_{ij}$

Normalized Cuts: Embedding

Image

Eigenvector

Normalized Cuts: Embedding

Image

Eigenvector

Normalized Cuts: Embedding

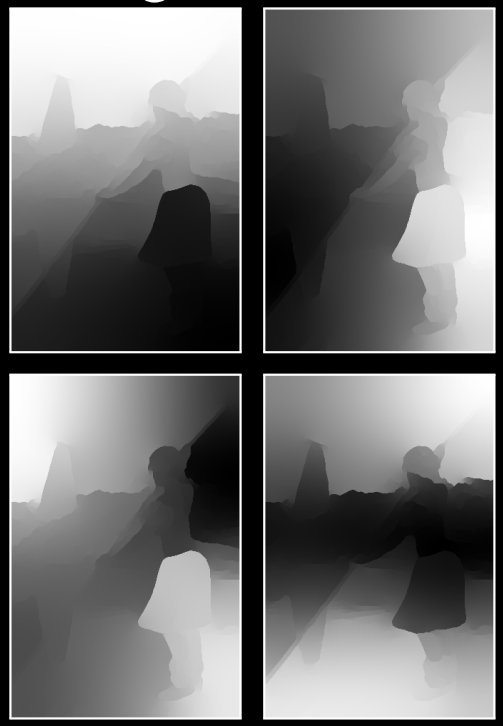
Image

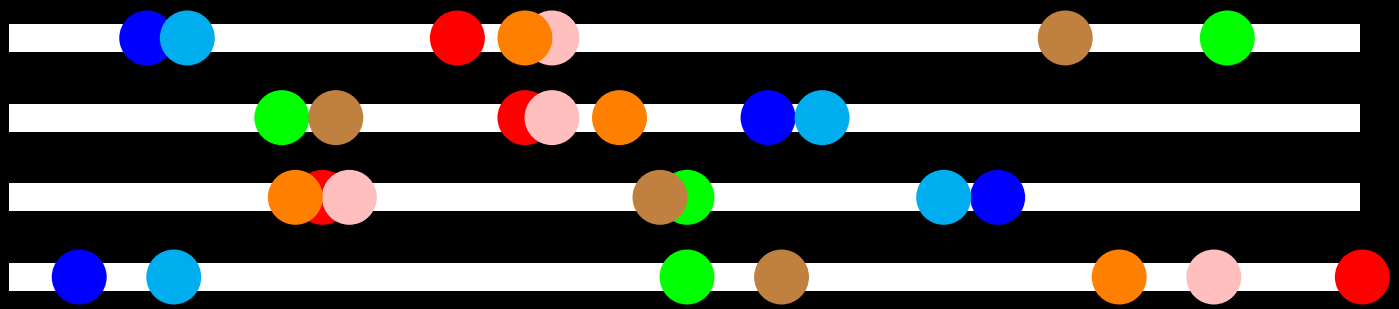
Eigenvector

Normalized Cuts: Embedding

Image

Eigenvectors

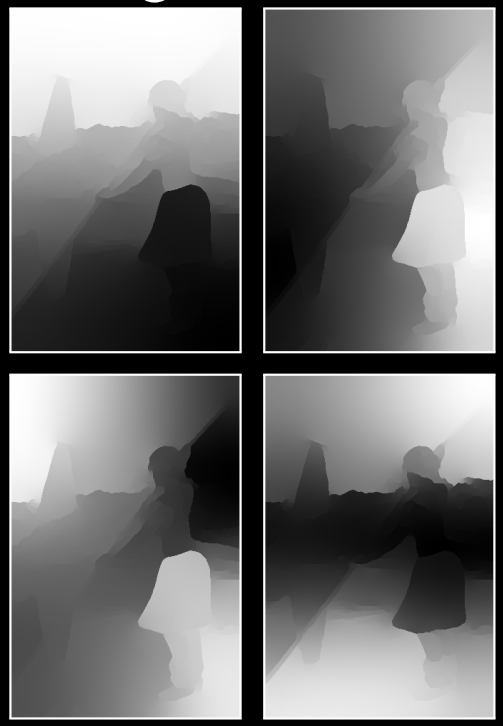


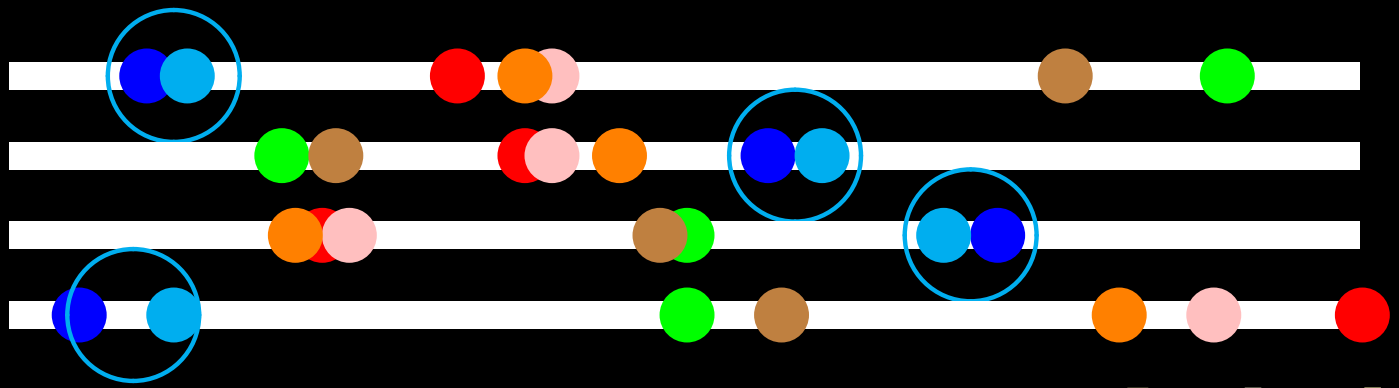


Normalized Cuts: Embedding

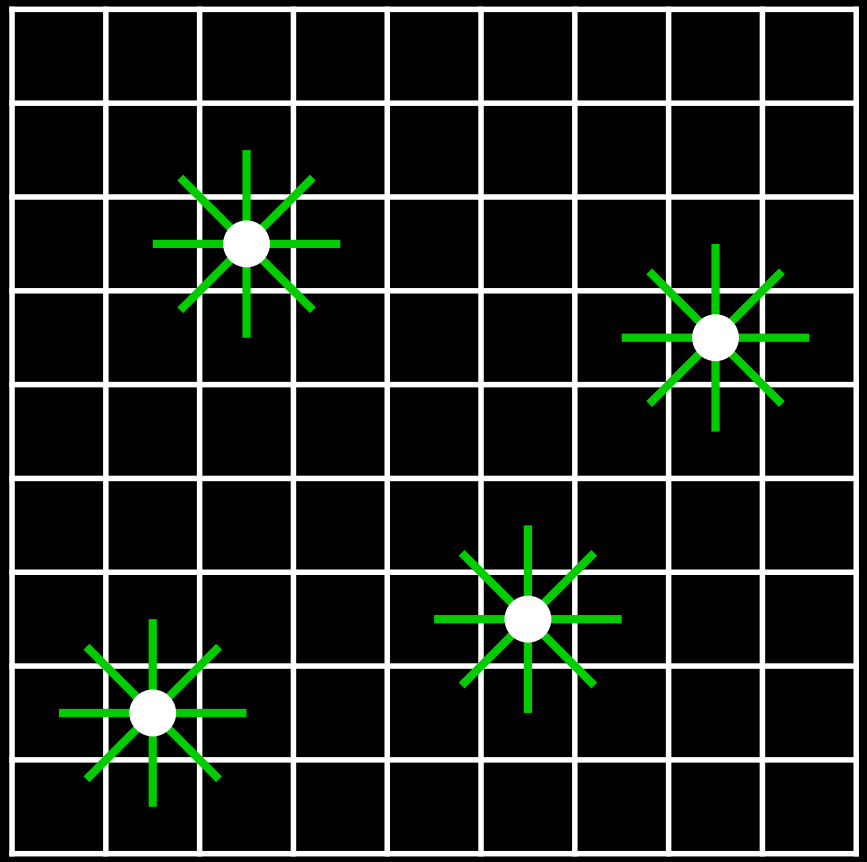
Image

Eigenvectors



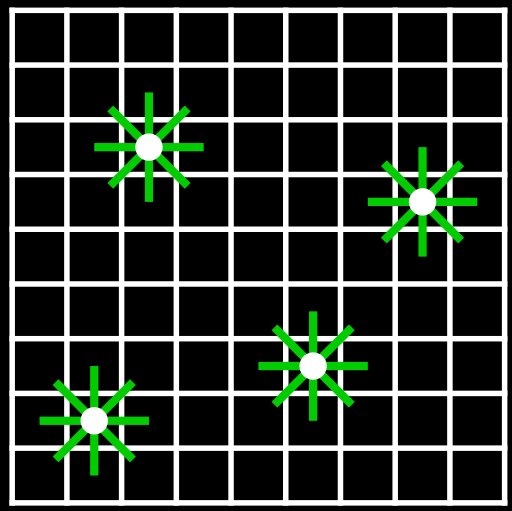


Single Scale Unconstrained Problem



W

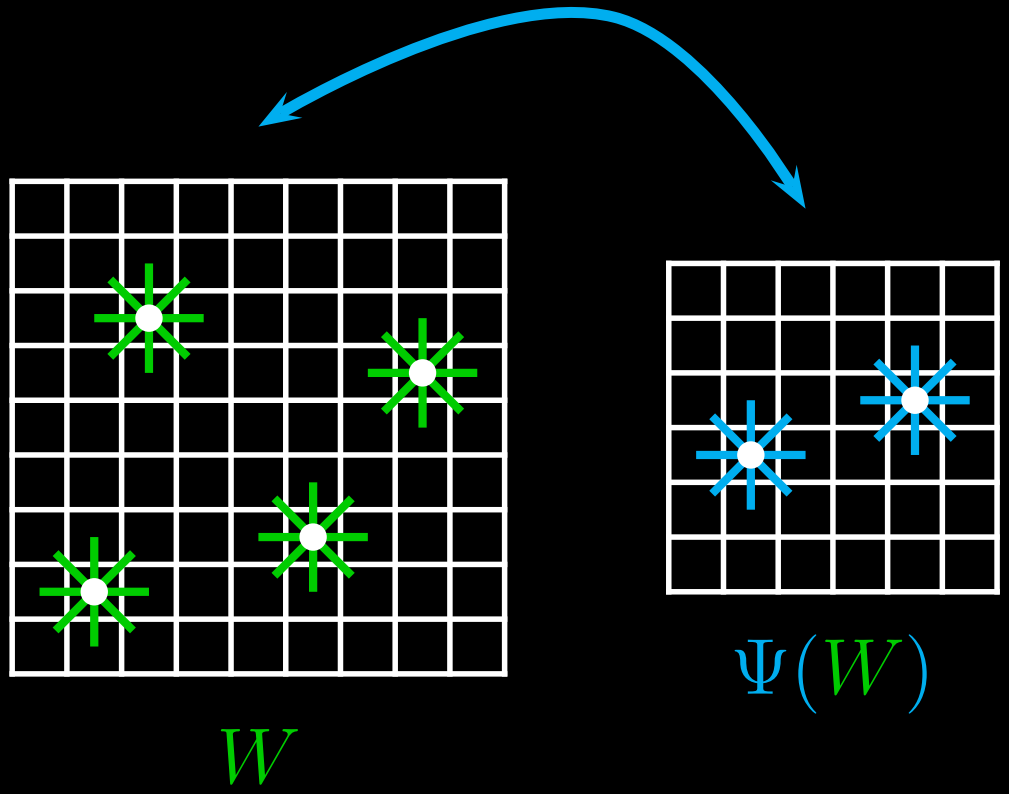
Multigrid Solver



W

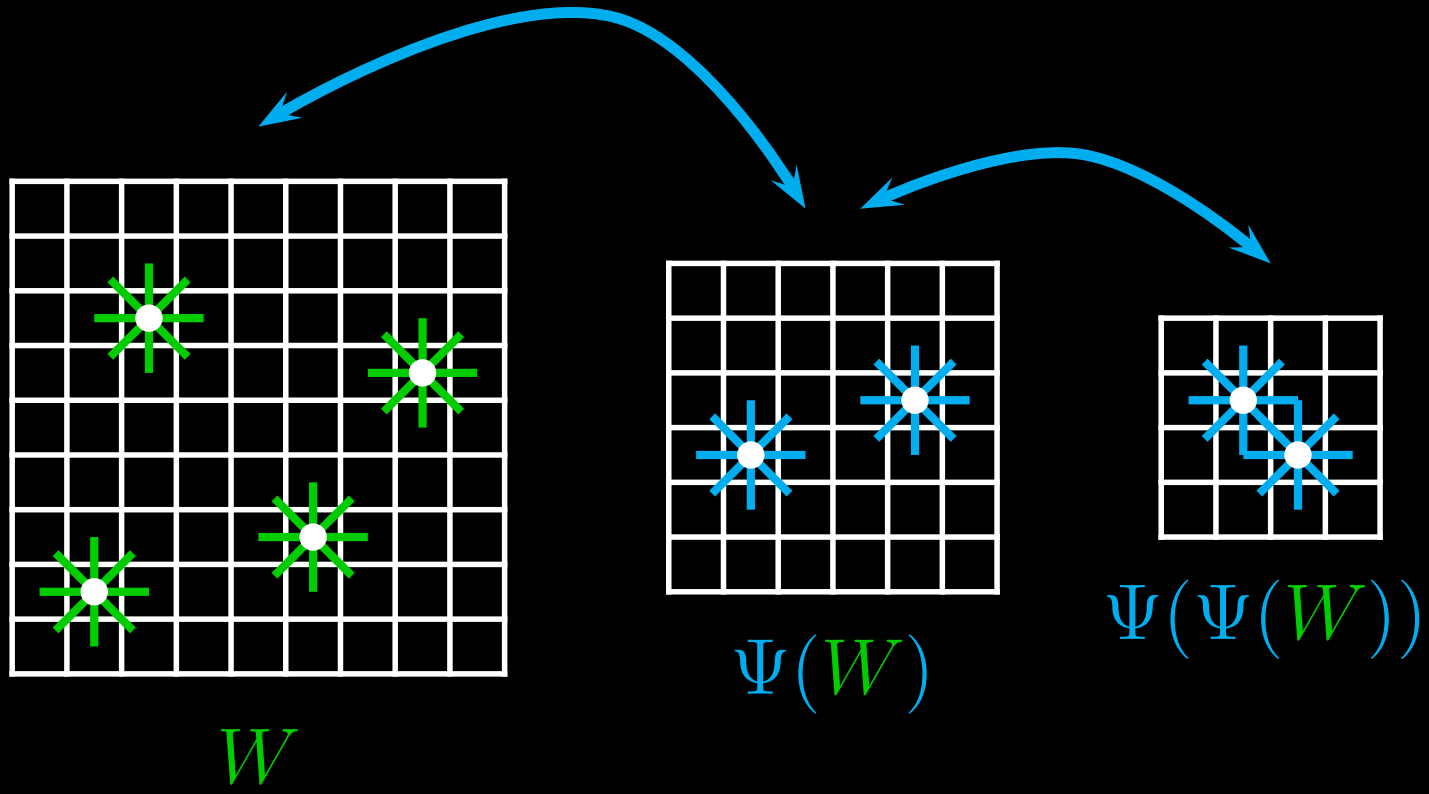
[Chennubhotla and Jepson, NIPS 2005]
[Brezina et al., Num. Linear Alg. w/App., 2008]
[Kushnir, Galun, and Brandt, PAMI 2010]

Multigrid Solver



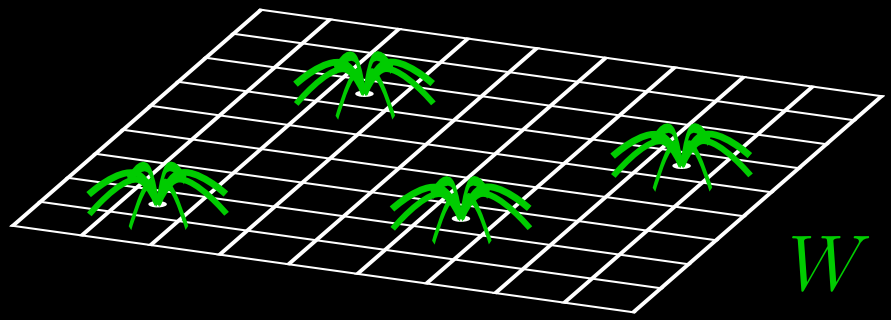
[Chennubhotla and Jepson, NIPS 2005]
[Brezina et al., Num. Linear Alg. w/App., 2008]
[Kushnir, Galun, and Brandt, PAMI 2010]

Multigrid Solver



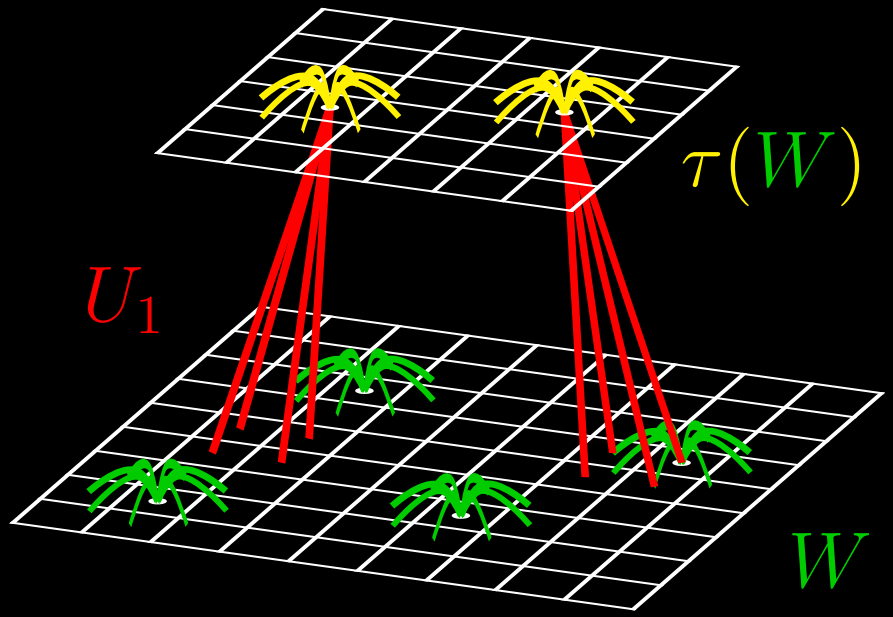
[Chennubhotla and Jepson, NIPS 2005]
[Brezina et al., Num. Linear Alg. w/App., 2008]
[Kushnir, Galun, and Brandt, PAMI 2010]

Multirange Problem



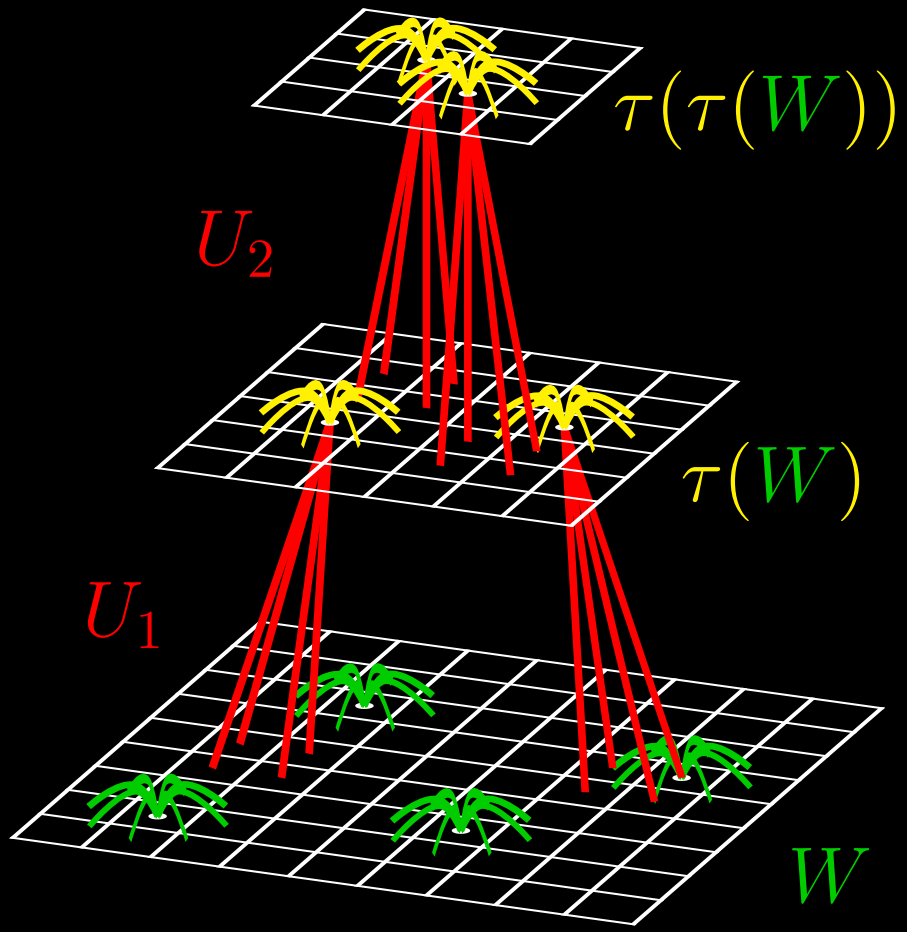
[Cour, Benezit, and Shi, CVPR 2005]

Multirange Problem



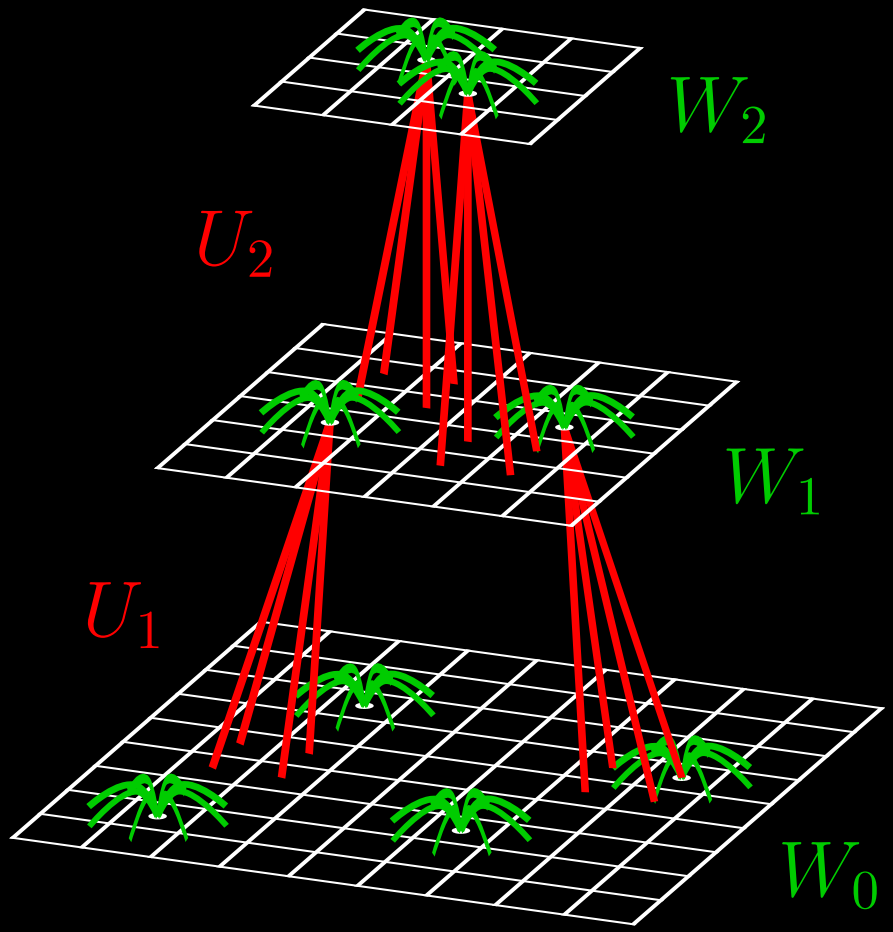
[Cour, Benezit, and Shi, CVPR 2005]

Multirange Problem



[Cour, Benezit, and Shi, CVPR 2005]

Multiscale Problem



Multigrid and Multiscale

Multigrid and Multiscale

multigrid: efficient solver computation

Multigrid and Multiscale

multigrid: efficient solver computation

multiscale: efficient problem representation

Multigrid and Multiscale

multigrid: efficient solver computation

multiscale: efficient problem representation

use both!

Multigrid and Multiscale

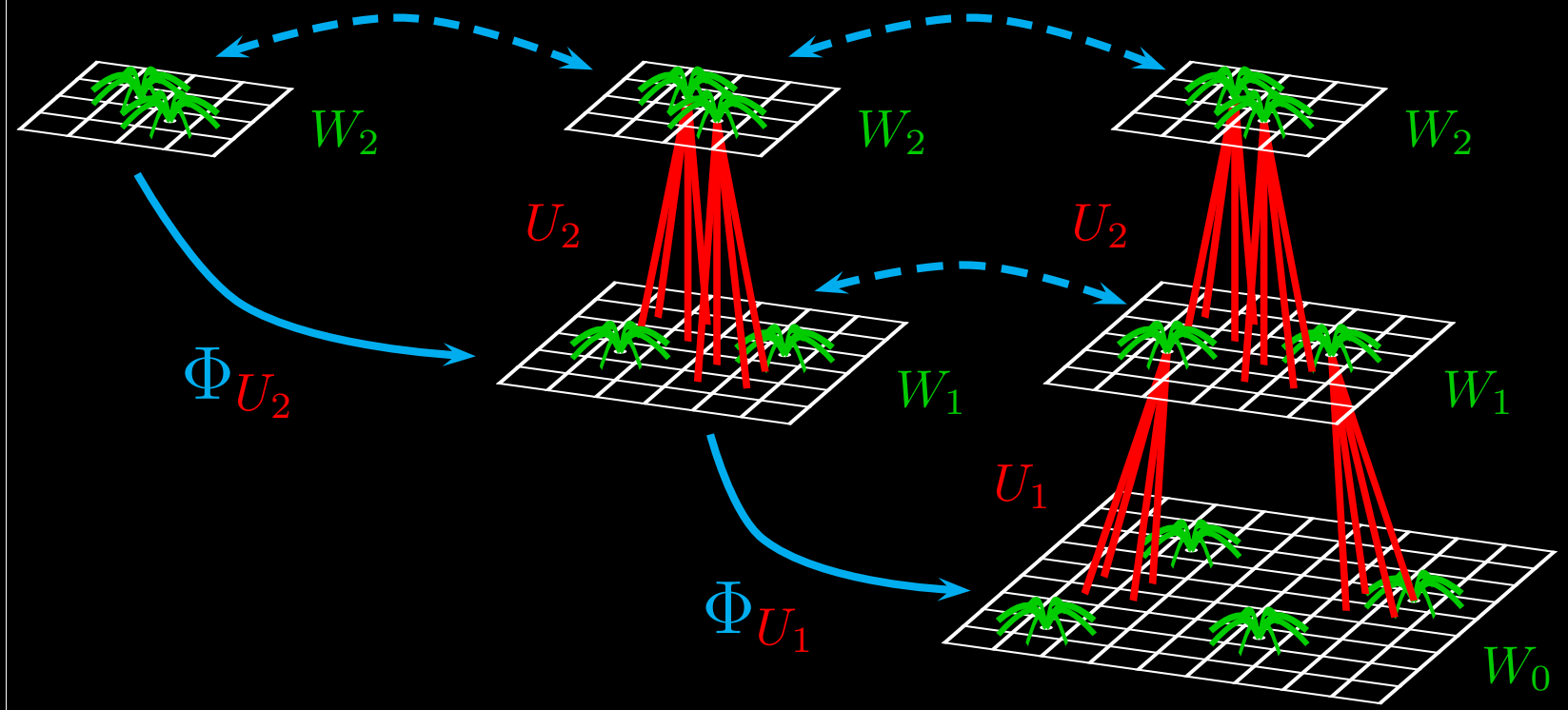
multigrid: efficient solver computation

multiscale: efficient problem representation

use both!

multiscale structure shapes multigrid strategy

Progressive Multigrid Multiscale



Algorithm Overview

Algorithm Overview

$$W = \begin{bmatrix} W_2 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_0 \end{bmatrix} \quad U = \begin{bmatrix} 0 & U_2 & U_1 \end{bmatrix} \quad Z = \begin{bmatrix} Z_2 \\ Z_1 \\ Z_0 \end{bmatrix}$$

Algorithm Overview

$$W = \begin{bmatrix} W_2 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_0 \end{bmatrix} \quad U = \begin{bmatrix} 0 & U_2 & U_1 \end{bmatrix} \quad Z = \begin{bmatrix} Z_2 \\ Z_1 \\ Z_0 \end{bmatrix}$$

- Solve unconstrained problem: $(W_2, 0)$ for $\bar{Z}_2$

Algorithm Overview

$$W = \begin{bmatrix} W_2 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_0 \end{bmatrix} \quad U = [0 \quad U_2 \quad U_1] \quad Z = \begin{bmatrix} Z_2 \\ Z_1 \\ Z_0 \end{bmatrix}$$

- Solve unconstrained problem: $(W_2, 0)$ for $\bar{Z}_2$
- Look at constraint: $U_2^* \; [Z_2; \; Z_1] = 0$

Algorithm Overview

$$W = \begin{bmatrix} W_2 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_0 \end{bmatrix} \quad U = [0 \;\; U_2 \;\; U_1] \quad Z = \begin{bmatrix} Z_2 \\ Z_1 \\ Z_0 \end{bmatrix}$$

- Solve unconstrained problem: $(W_2, 0)$ for $\bar{Z}_2$
- Look at constraint: $U_2^* \; [Z_2; \; Z_1] = 0$
- Rewrite as: $[\widehat{U_2}; \; \widetilde{U_2}]^* \; [Z_2; \; Z_1] = 0$

Algorithm Overview

$$W = \begin{bmatrix} W_2 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_0 \end{bmatrix} \quad U = [0 \;\; U_2 \;\; U_1] \quad Z = \begin{bmatrix} Z_2 \\ Z_1 \\ Z_0 \end{bmatrix}$$

- Solve unconstrained problem: $(W_2, 0)$ for $\bar{Z}_2$
- Look at constraint: $U_2^* \; [Z_2; \; Z_1] = 0$
- Rewrite as: $[\widehat{U_2}; \; \widetilde{U_2}]^* \; [Z_2; \; Z_1] = 0$
- Least-squares interpolate:

$$\widetilde{Z}_1 = \widetilde{U_2}(\widetilde{U_2}^* \widetilde{U_2})^{-1}(-\widehat{U_2}^* \bar{Z}_2)$$

Algorithm Overview

$$W = \begin{bmatrix} W_2 & 0 & 0 \\ 0 & W_1 & 0 \\ 0 & 0 & W_0 \end{bmatrix} \quad U = [0 \;\; U_2 \;\; U_1] \quad Z = \begin{bmatrix} Z_2 \\ Z_1 \\ Z_0 \end{bmatrix}$$

- Solve unconstrained problem: $(W_2, 0)$ for $\bar{Z}_2$
- Look at constraint: $U_2^* \; [Z_2; \; Z_1] = 0$
- Rewrite as: $[\widehat{U_2}; \; \widetilde{U_2}]^* \; [Z_2; \; Z_1] = 0$
- Least-squares interpolate:

  $$\widetilde{Z}_1 = \widetilde{U_2}(\widetilde{U_2}^* \widetilde{U_2})^{-1}(-\widehat{U_2}^* \bar{Z}_2)$$

- Use $[\bar{Z}_2; \; \widetilde{Z}_1]$ as initial guess for solving:

  $(\mathrm{Diag}(W_2, W_1), [0 \;\; U_2])$ for $[\bar{Z}_2; \; \bar{Z}_1]$

Implementation

Implementation

- Work with m eigenvectors simultaneously

Implementation

- ~~Work with m eigenvectors simultaneously~~
- Work with $n \times 2m$ intermediate representation:

  $n = \#$ nodes

  $m = \#$ eigenvectors

Implementation

- ~~Work with m eigenvectors simultaneously~~
- Work with $n \times 2m$ intermediate representation:
  $n = \#$ nodes
  $m = \#$ eigenvectors
- n increases coarse-to-fine

Implementation

- ~~Work with m eigenvectors simultaneously~~
- Work with $n \times 2m$ intermediate representation:
  $n = \#$ nodes
  $m = \#$ eigenvectors
- n increases coarse-to-fine
- Randomized Matrix Approximation
  [Halko, Martinsson, and Tropp, SIREV, 2011]

Implementation

- ~~Work with m eigenvectors simultaneously~~
- Work with $n \times 2m$ intermediate representation:
  $n = \#$ nodes
  $m = \#$ eigenvectors
- n increases coarse-to-fine
- Randomized Matrix Approximation
  [Halko, Martinsson, and Tropp, SIREV, 2011]
  - Randomized algorithms for linear algebra problems
  - Exponentially small failure probability
  - Simple implementation
  - Same computational complexity
  - Better hardware parallelization properties

Randomized Matrix Approximation

Randomized Matrix Approximation

- **Fixed Rank Problem**

  Given: $n \times n$ sparse matrix M

  Find: $n \times l$ dense matrix A, where $l = 2m \ll n$

  Such that: range of A approximates range of M

Randomized Matrix Approximation

- **Fixed Rank Problem**

  Given: $n \times n$ sparse matrix M

  Find: $n \times l$ dense matrix A, where $l = 2m \ll n$

  Such that: range of A approximates range of M

- **Randomized Subspace Iteration**

  Draw draw $n \times l$ $\mathrm{Gaussian\ matrix}$ Ω

  $Y \leftarrow (MM^*)^q M\Omega$

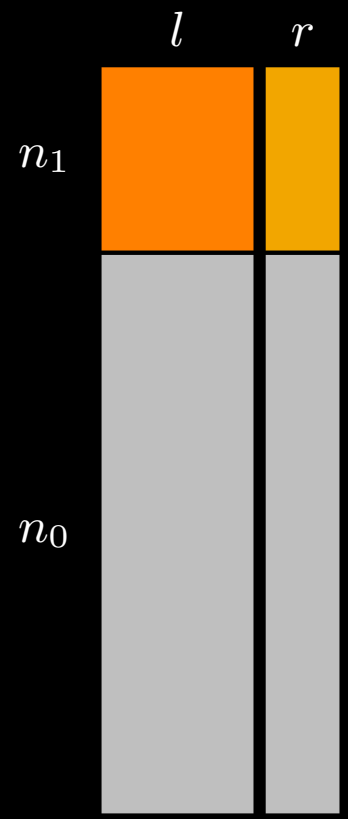
  $A \leftarrow \mathrm{QR\text{-}Orthonormalize}(Y)$

Randomized Matrix Approximation

- **Fixed Rank Problem**
  - Given: $n \times n$ sparse matrix M
  - Find: $n \times l$ dense matrix A, where $l = 2m \ll n$
  - Such that: range of A approximates range of M
- **Randomized Subspace Iteration**
  - Draw draw $n \times l$ $\mathrm{Gaussian\ matrix}$ Ω
  - $Y \leftarrow (MM^*)^q M\Omega$
  - $A \leftarrow \mathrm{QR\text{-}Orthonormalize}(Y)$
- **Eigensolver**
  - $B \leftarrow A^*MA$ — $l \times l$ matrix
  - $(V, \Lambda) \leftarrow \mathrm{Eigs}(B, m)$ — small eigenproblem
  - $Z \leftarrow AV$

Matrix Approximation + Interpolation

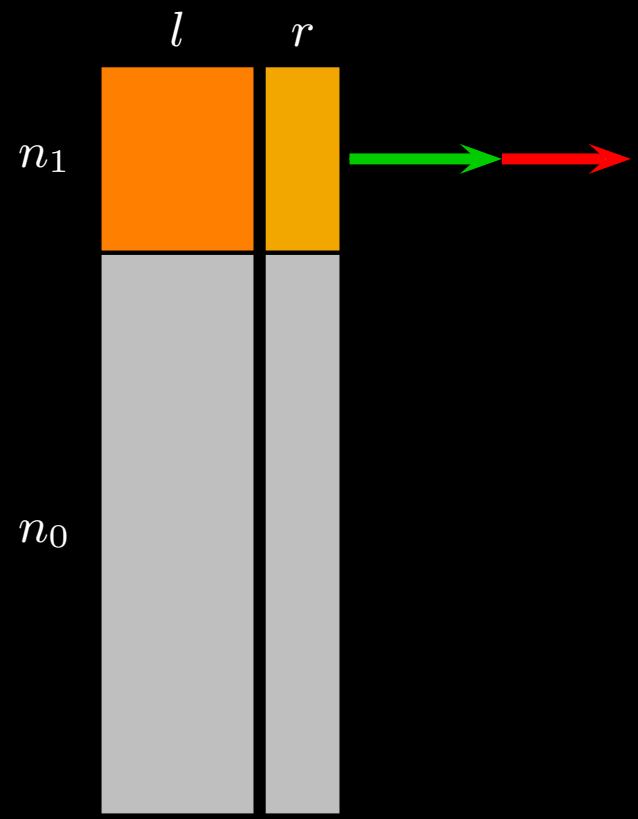
$$M = QPQ$$



Initialize A

Matrix Approximation + Interpolation

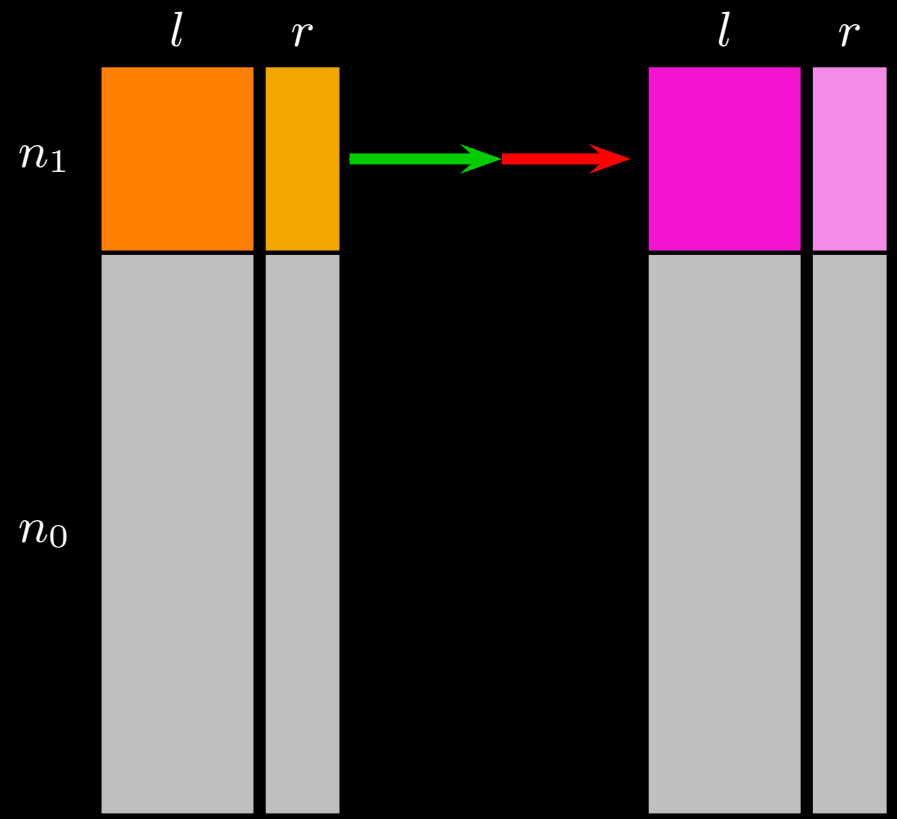
$$M = QPQ$$



Apply M (Coarse)

Matrix Approximation + Interpolation

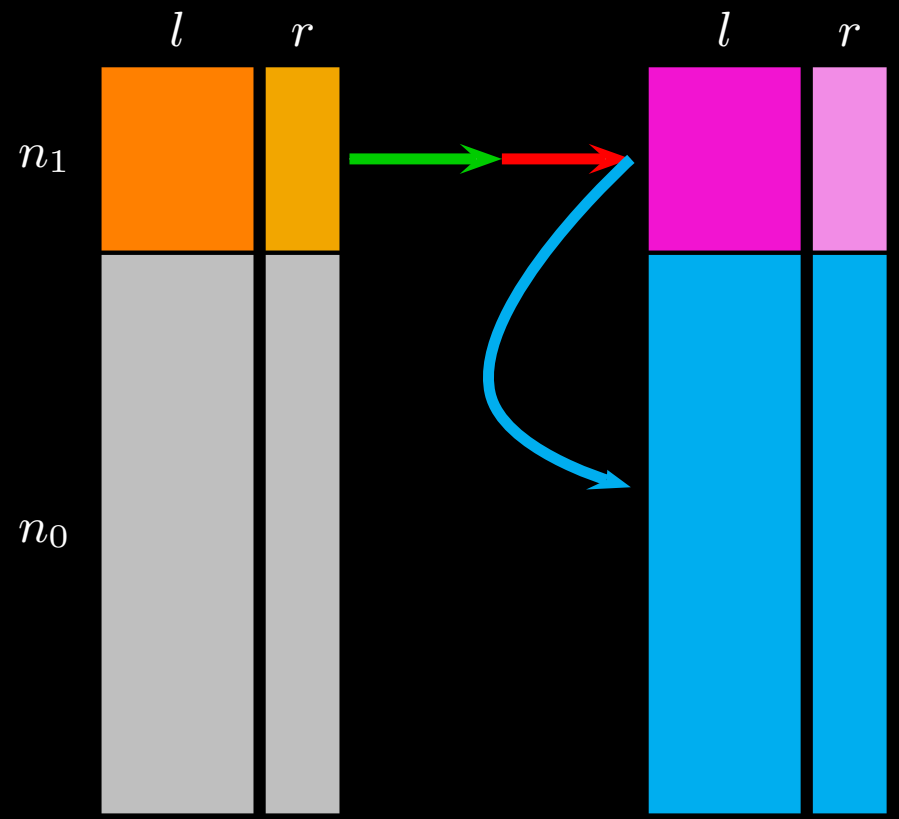
$$M = QPQ$$



Converge A (Coarse)

Matrix Approximation + Interpolation

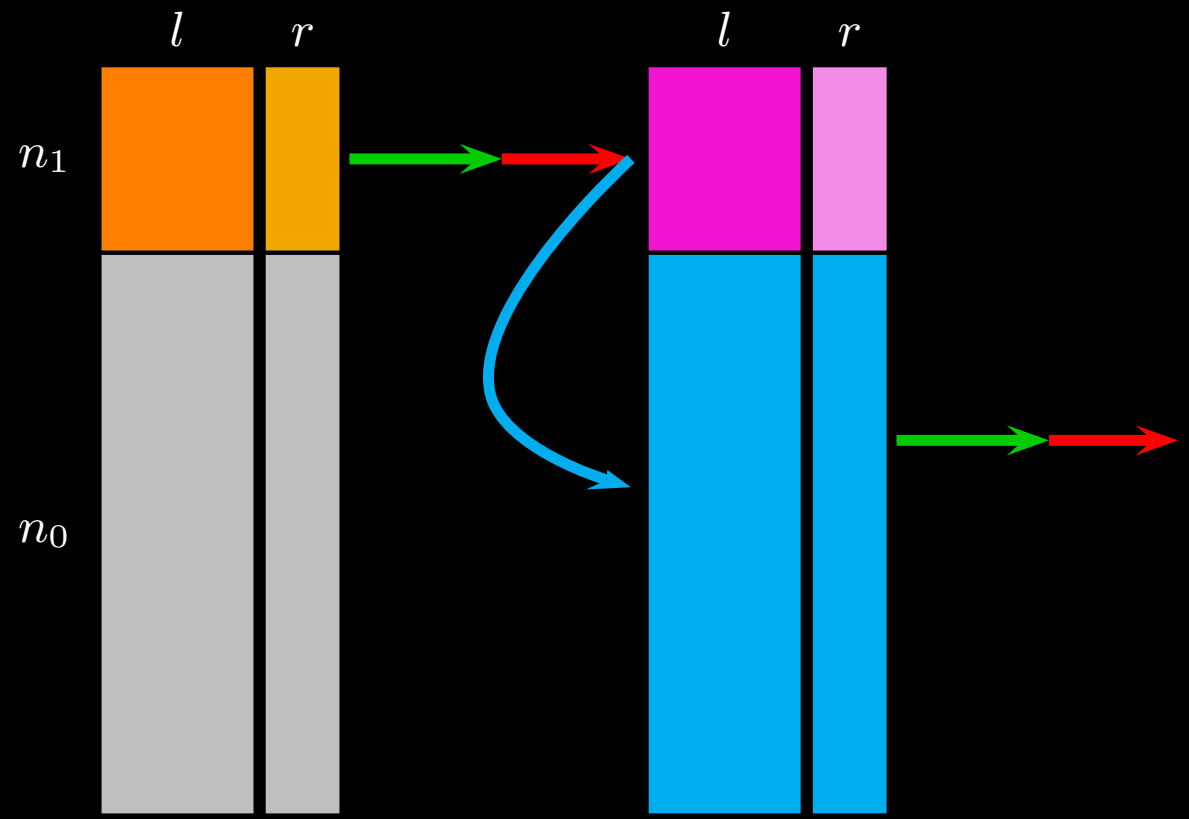
$$M = QPQ$$



Interpolate

Matrix Approximation + Interpolation

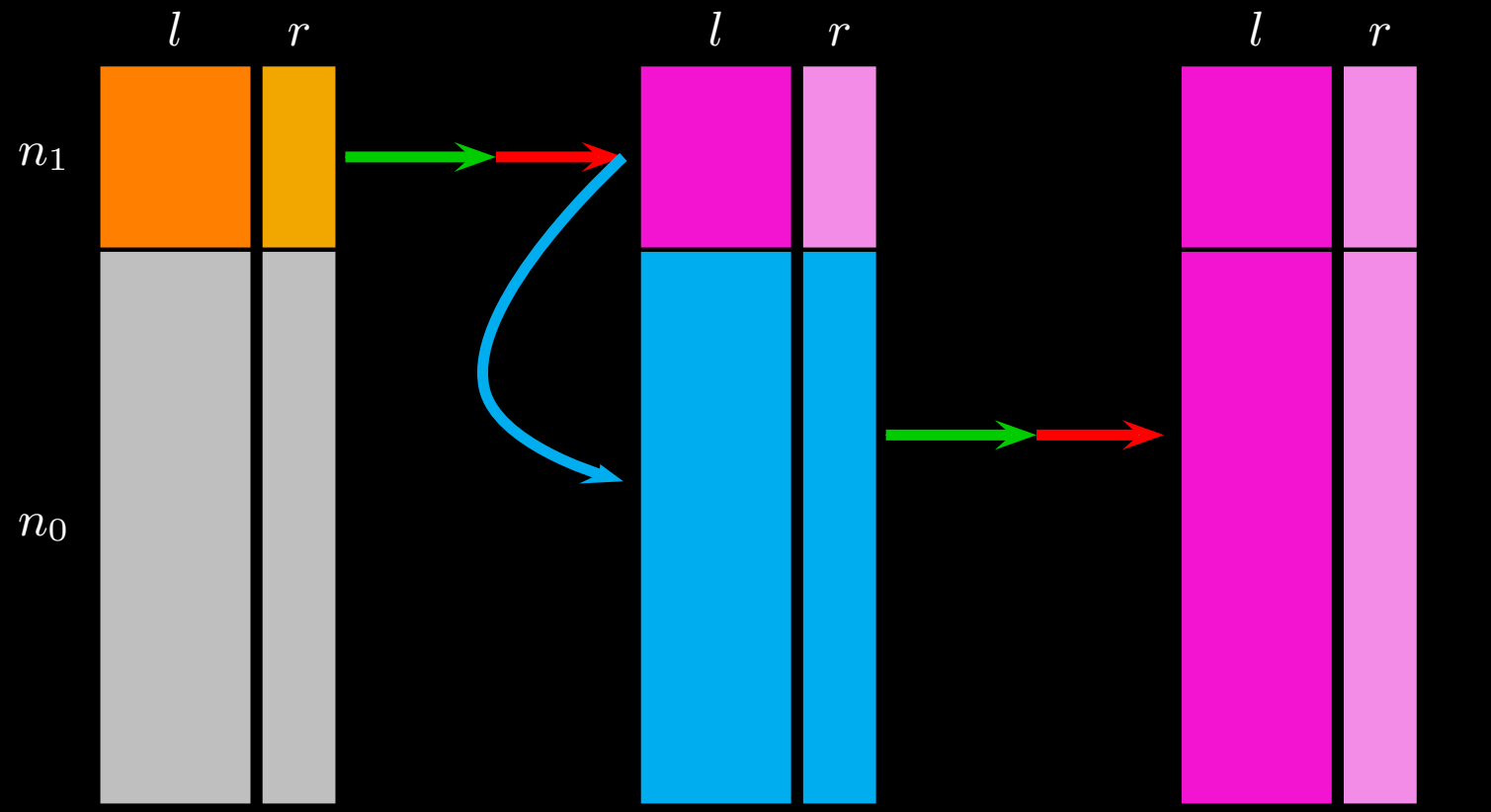
$$M = QPQ$$



Apply M (Fine)

Matrix Approximation + Interpolation

$$M = QPQ$$



Converge A

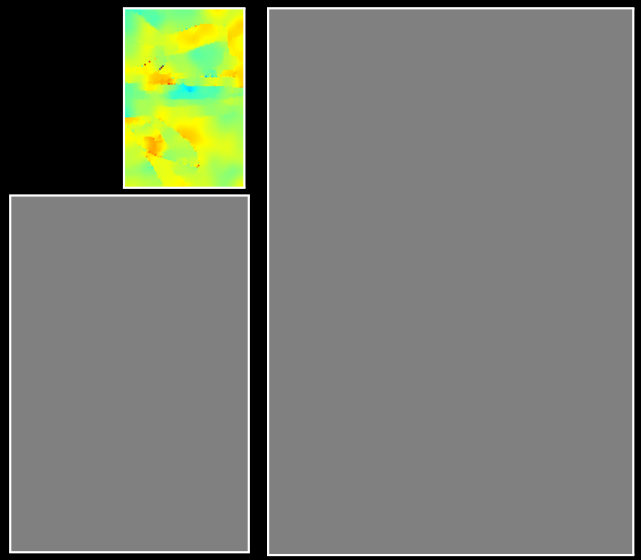
Eigenvector Convergence Comparison

Image

Eigenvector 7

Eigenvector Convergence Comparison

1 coarse: 0.46 sec



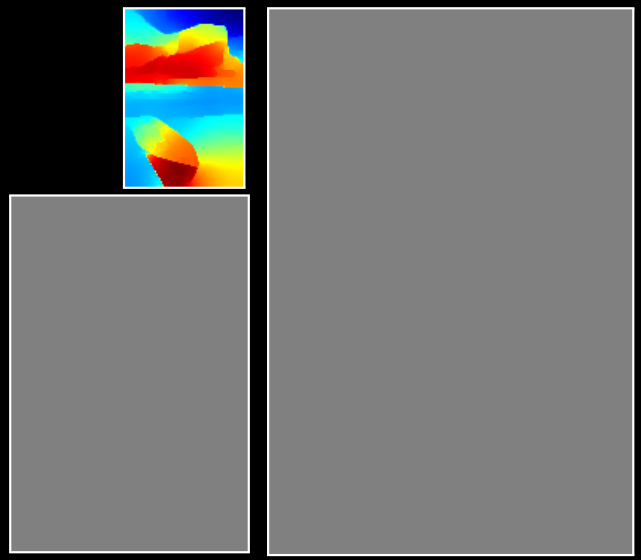
Image

Progressive Multigrid

Eigenvector 7

Eigenvector Convergence Comparison

20 coarse: 3 sec



Image

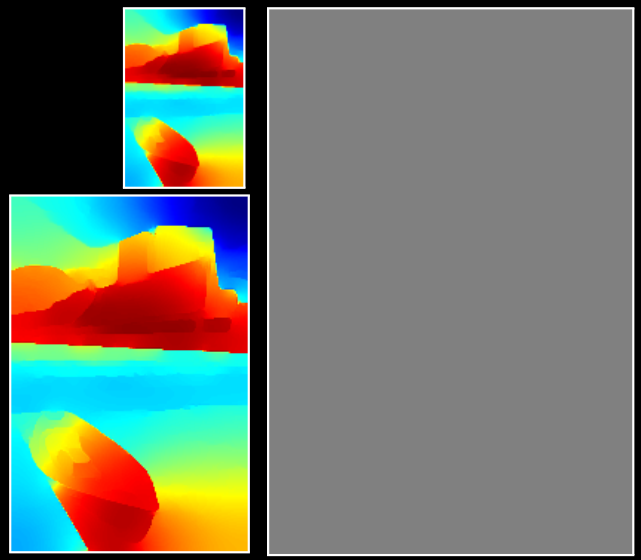
Progressive Multigrid

Eigenvector 7

Eigenvector Convergence Comparison

20 c, 1 med: 5 sec

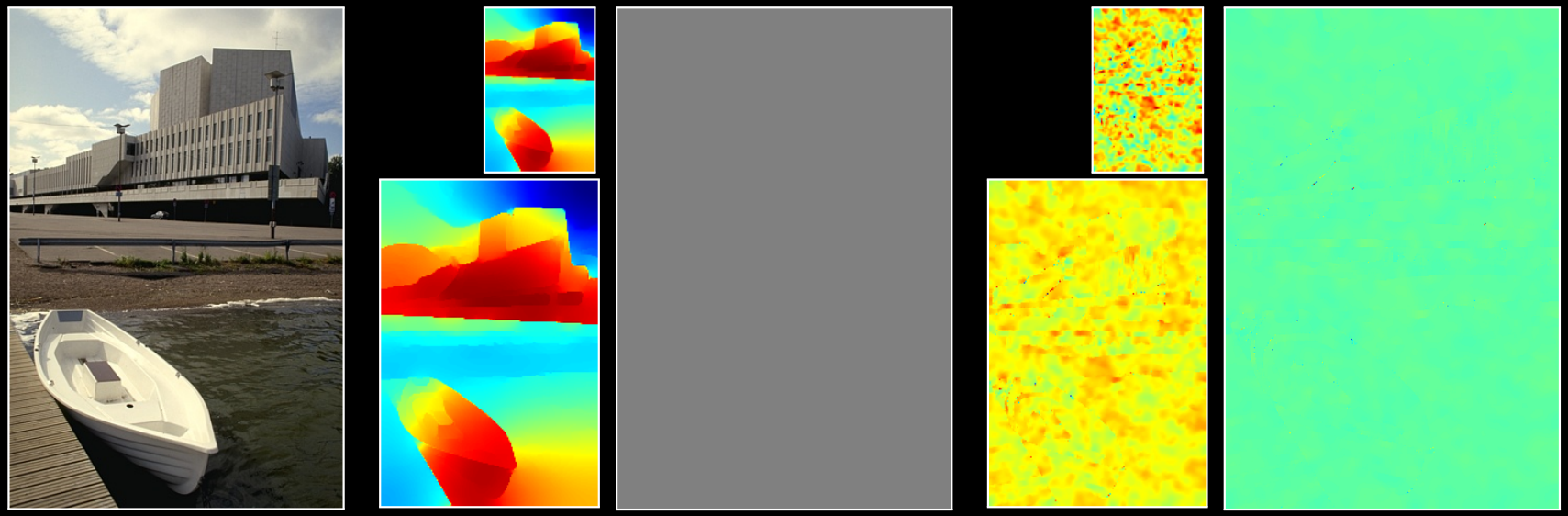
Image



Progressive Multigrid

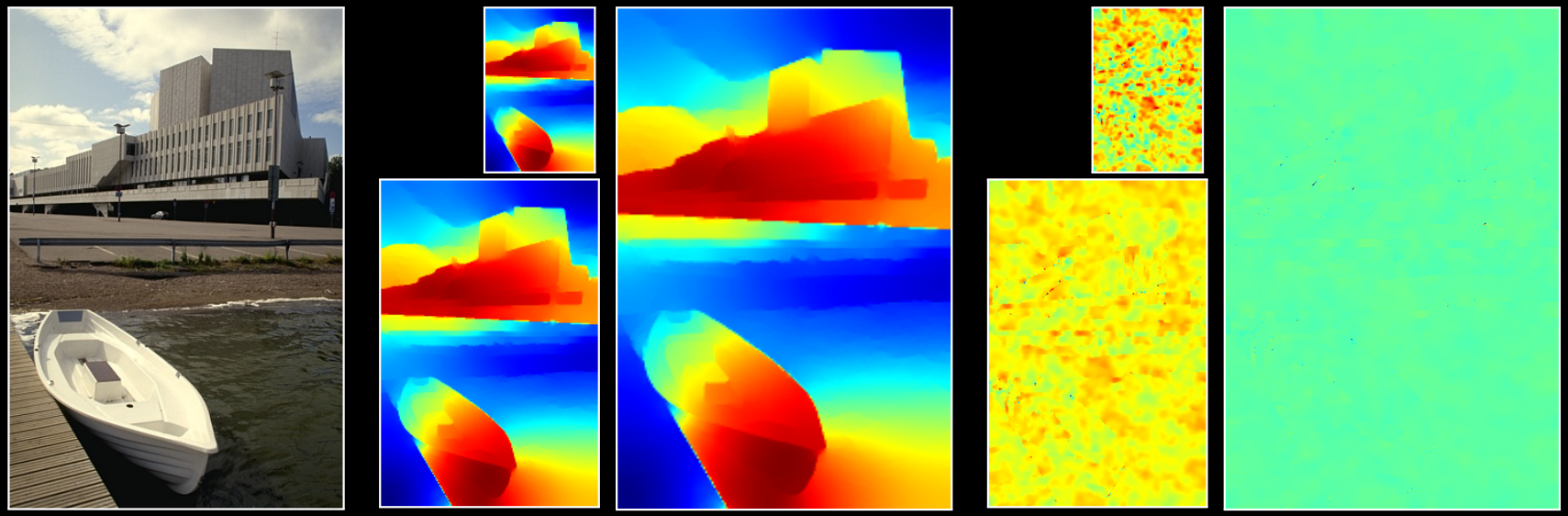
Eigenvector 7

Eigenvector Convergence Comparison



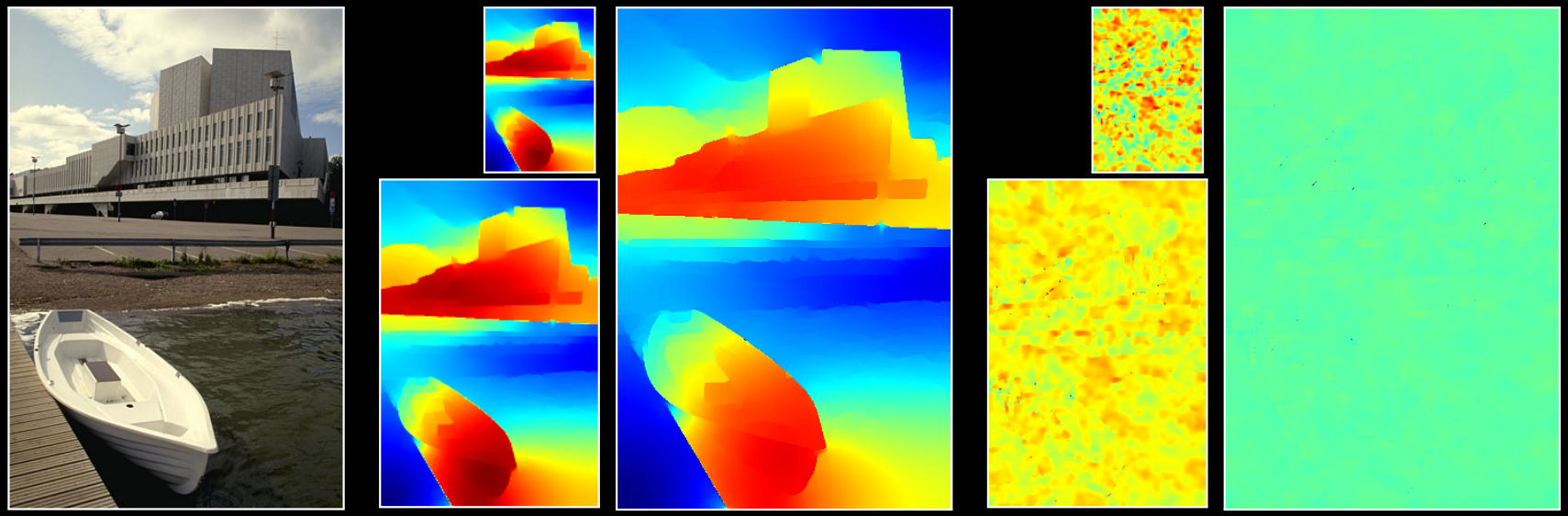
Eigenvector 7

Eigenvector Convergence Comparison



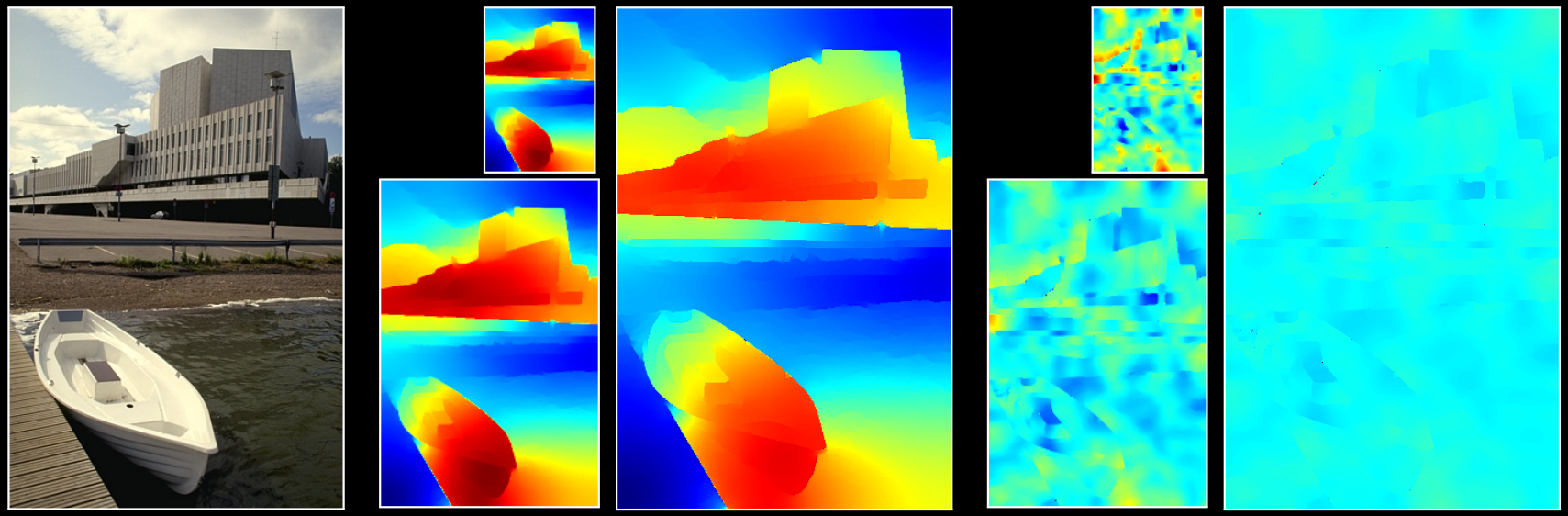
Eigenvector 7

Eigenvector Convergence Comparison



Eigenvector 7

Eigenvector Convergence Comparison



Eigenvector 7

Eigenvector Convergence Comparison

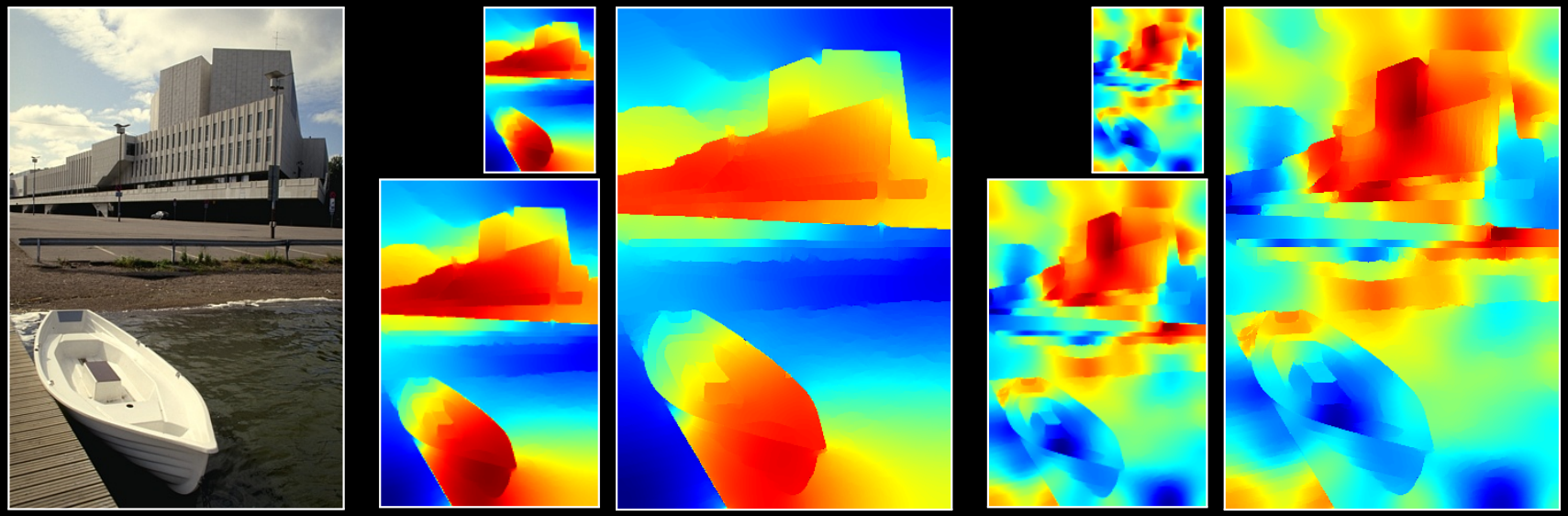
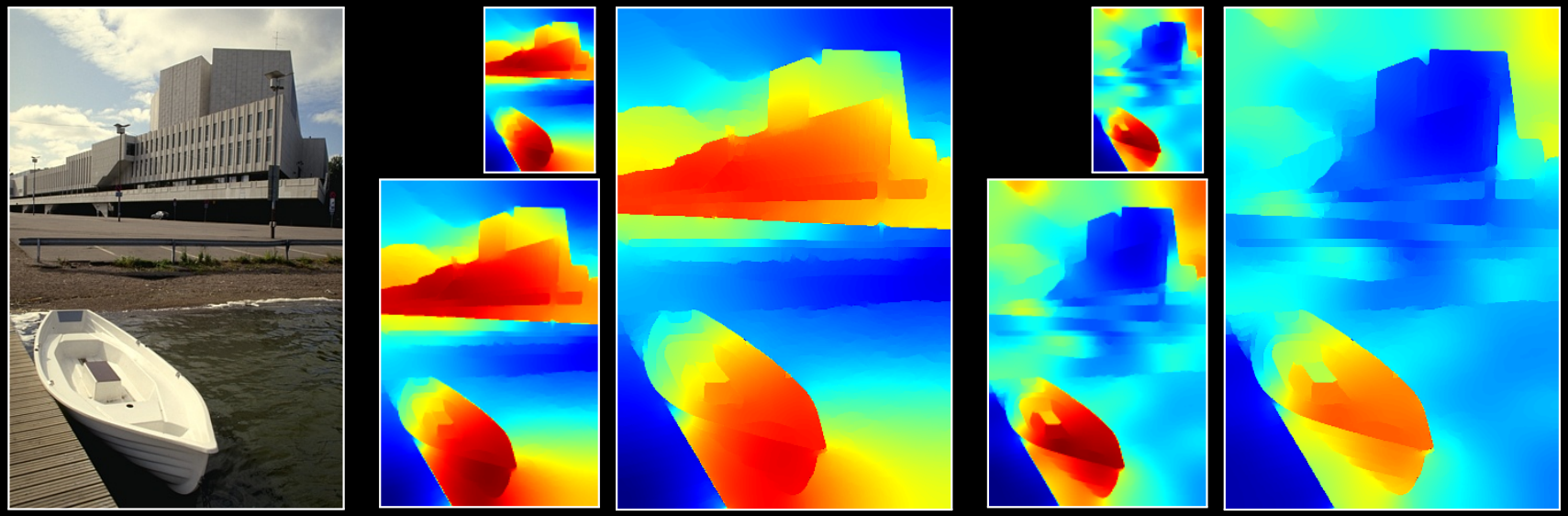


Image | Progressive Multigrid | Baseline Solver

Eigenvector 7

Eigenvector Convergence Comparison

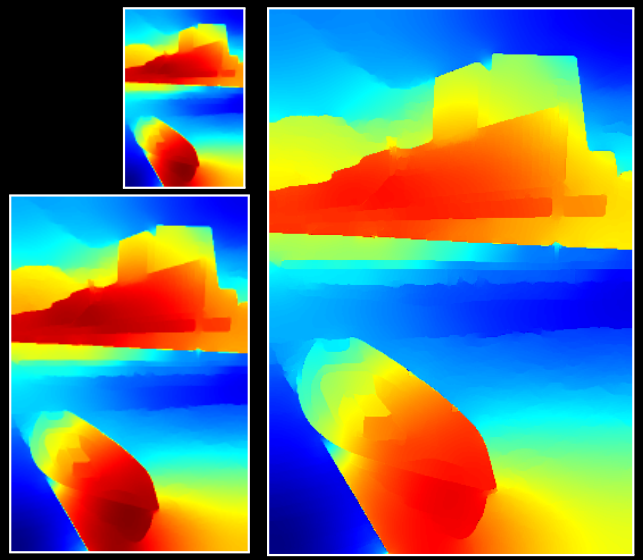


Eigenvector 7

Eigenvector Convergence Comparison

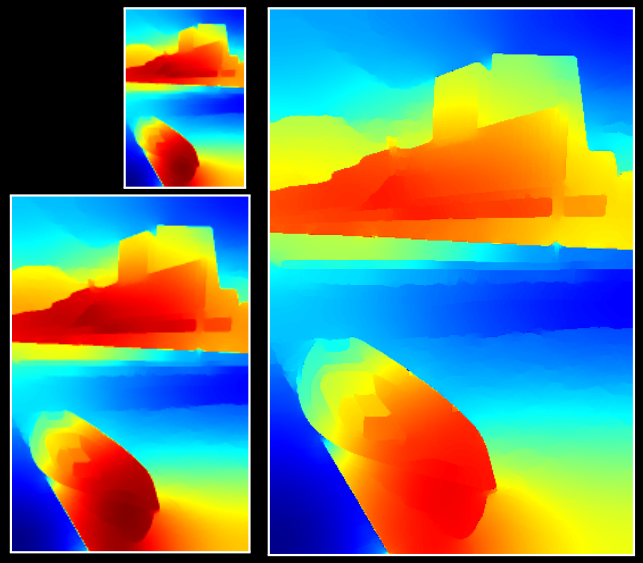
Image

. . . , 3 fine: **27 sec**



Progressive Multigrid

225 fine: **760 sec**



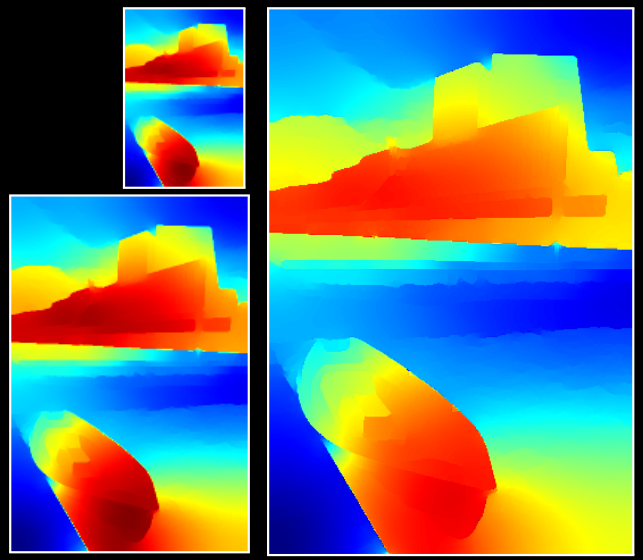
Baseline Solver

Eigenvector 7

Eigenvector Convergence Comparison

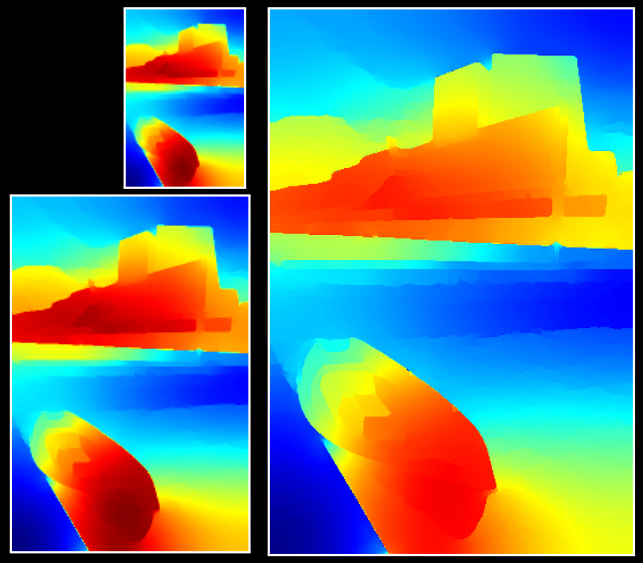
Image

. . . , 3 fine: **27 sec**



Progressive Multigrid

225 fine: **760 sec**



Baseline Solver

Eigenvector 7

$> 10\times$ speedup

Multiscale Spectral Pb

Image

sPb

M-sPb Coarse

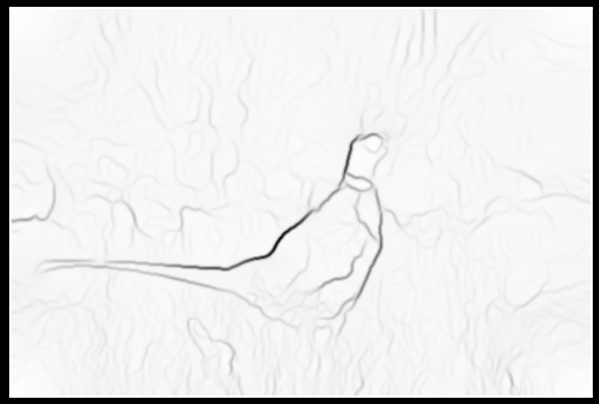
M-sPb Medium

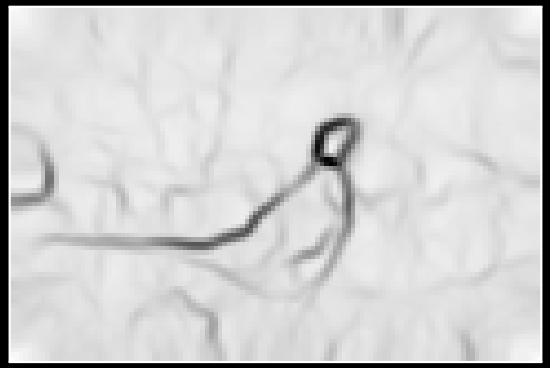
M-sPb Fine

Multiscale Spectral Pb

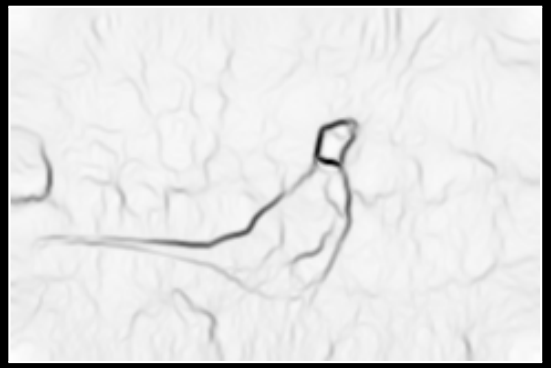
Image

sPb

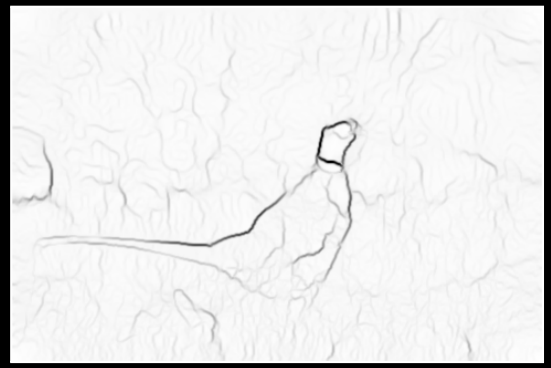




M-sPb Coarse



M-sPb Medium

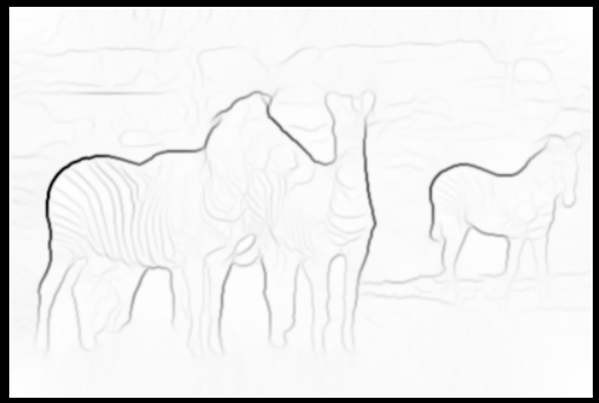


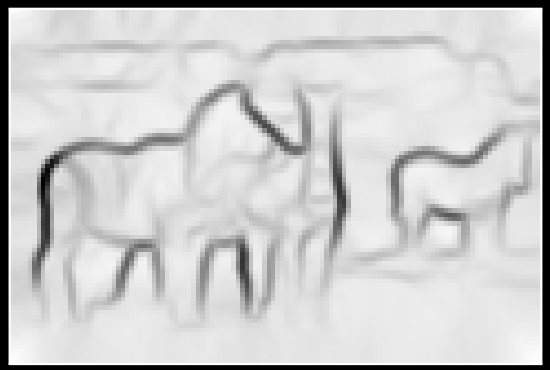
M-sPb Fine

Multiscale Spectral Pb

Image

sPb





M-sPb Coarse

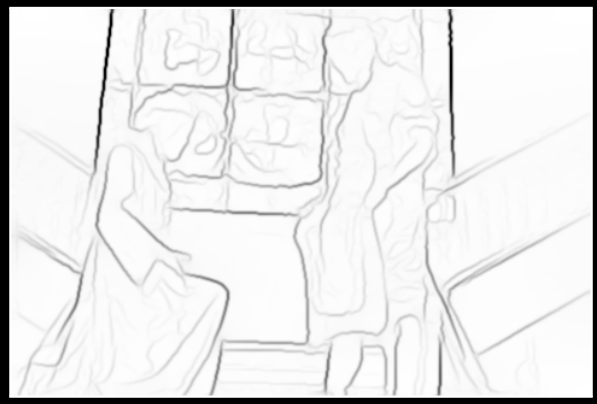
M-sPb Medium

M-sPb Fine

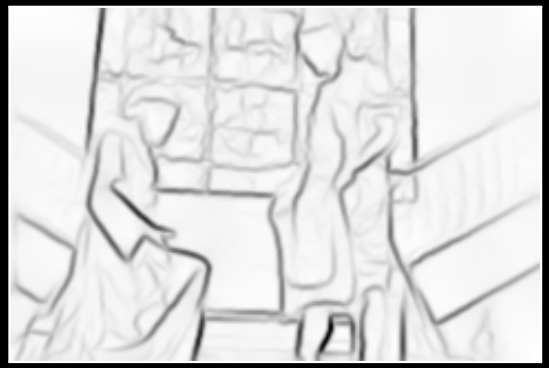
Multiscale Spectral Pb

Image

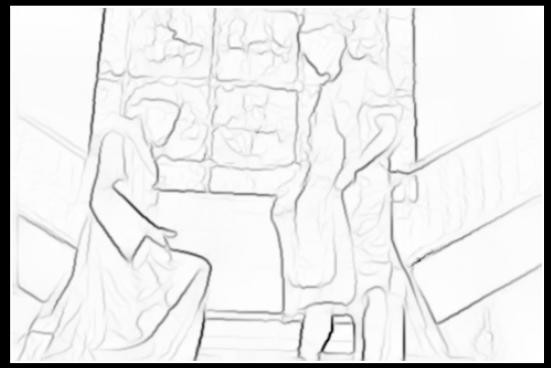
sPb



M-sPb Coarse



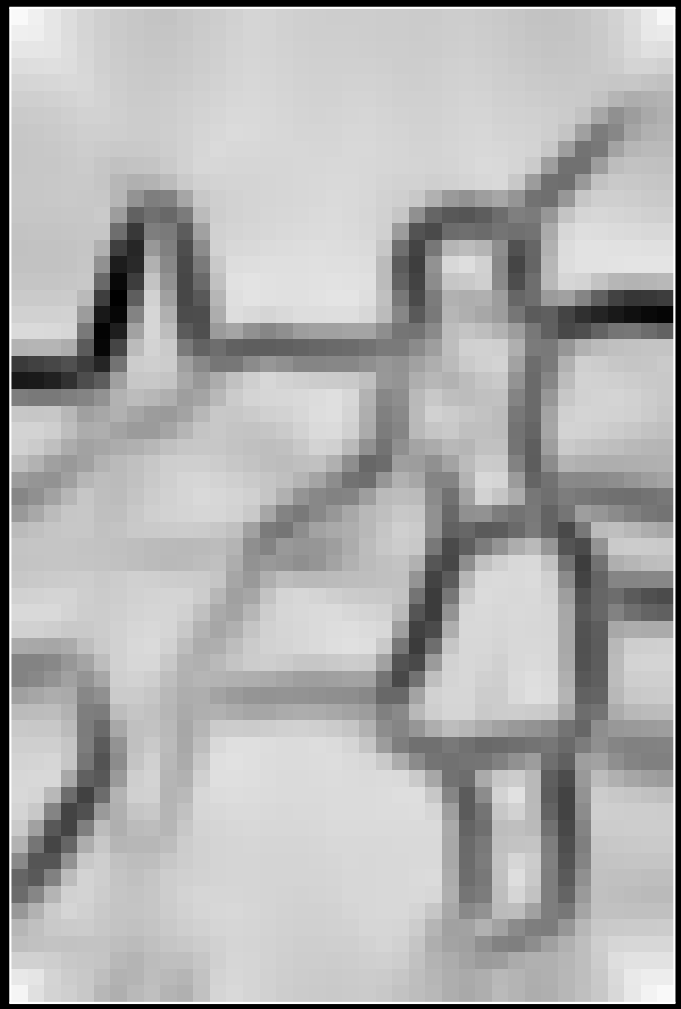
M-sPb Medium



M-sPb Fine

Multiscale Alignment for Free

Image



Multiscale sPb 1

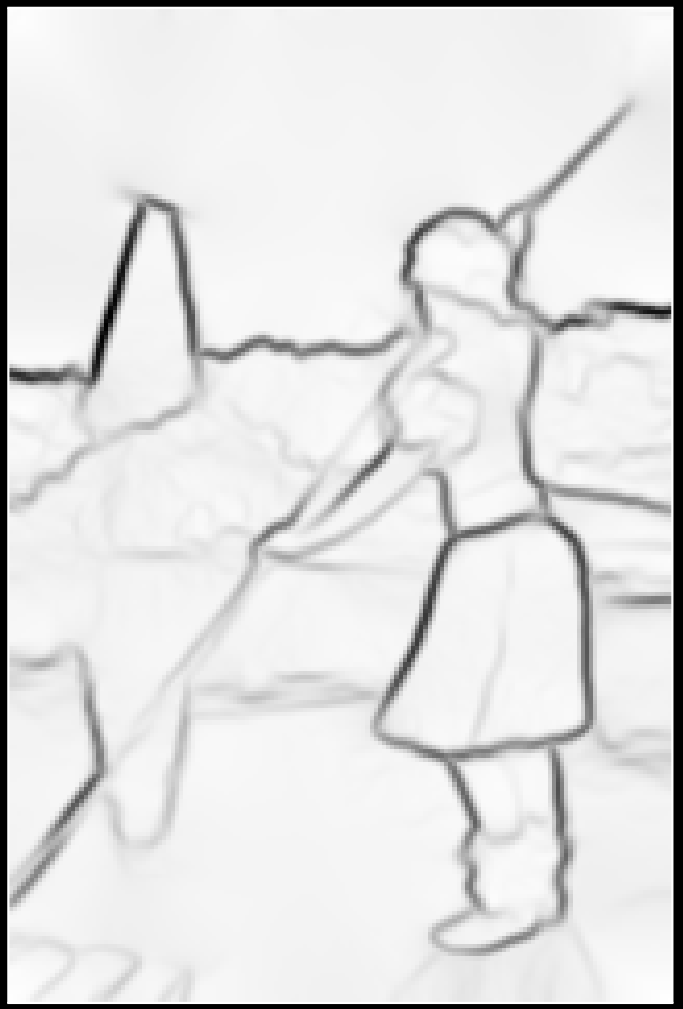
Multiscale Alignment for Free

Image

Multiscale sPb 2

Multiscale Alignment for Free

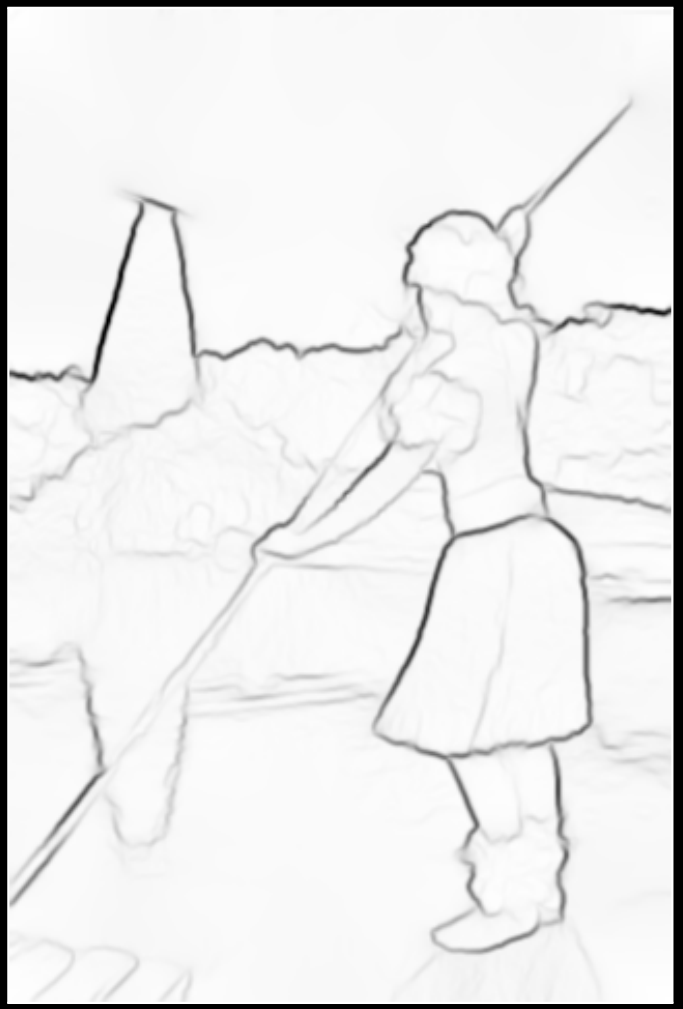
Image



Multiscale sPb 3

Multiscale Alignment for Free

Image



Multiscale sPb 4

Multiscale Alignment for Free

Image

Multiscale sPb 5

Thank You