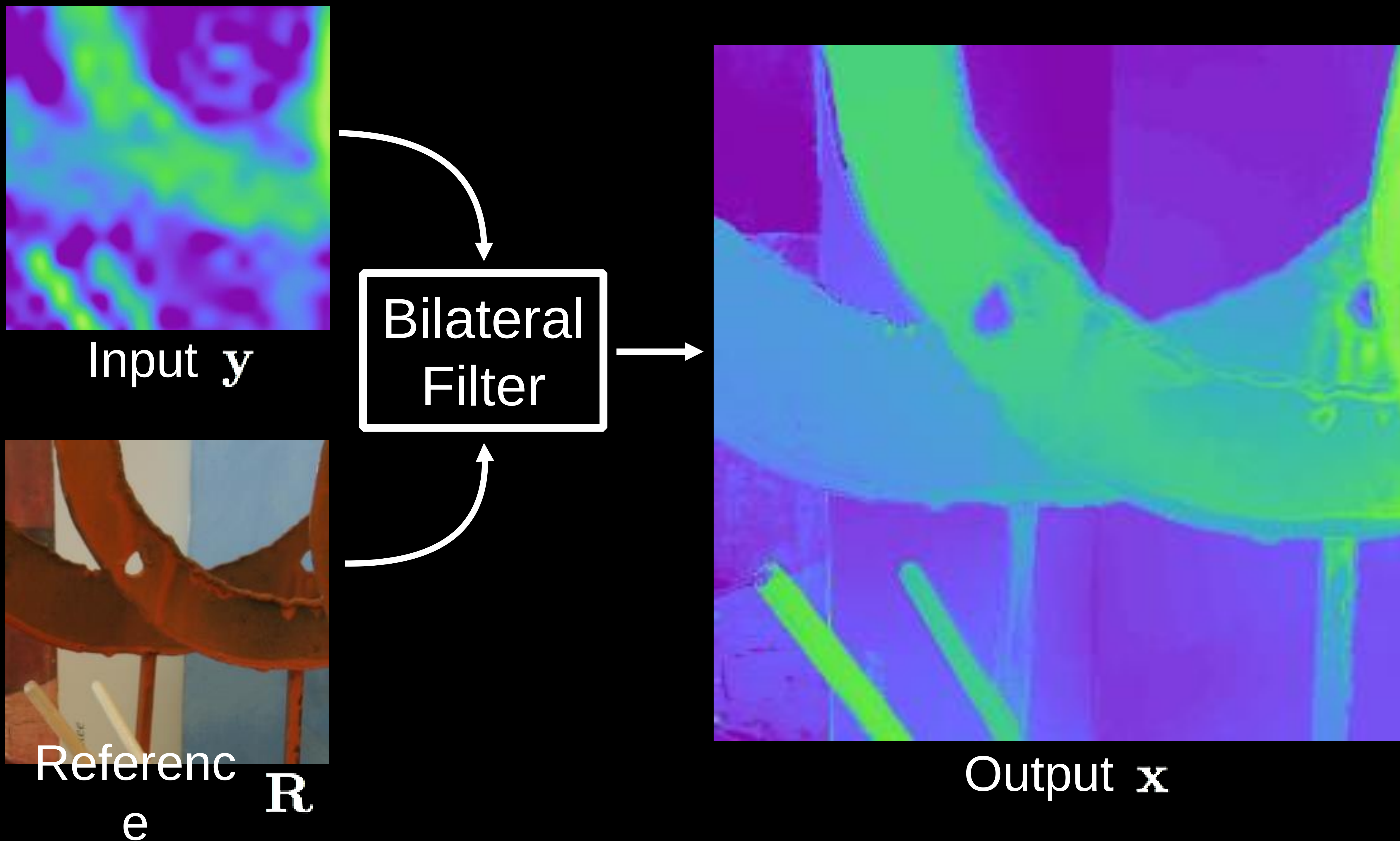
The Fast Bilateral Solver

Jonathan T. Barron Ben Poole
Google Research Stanford
University

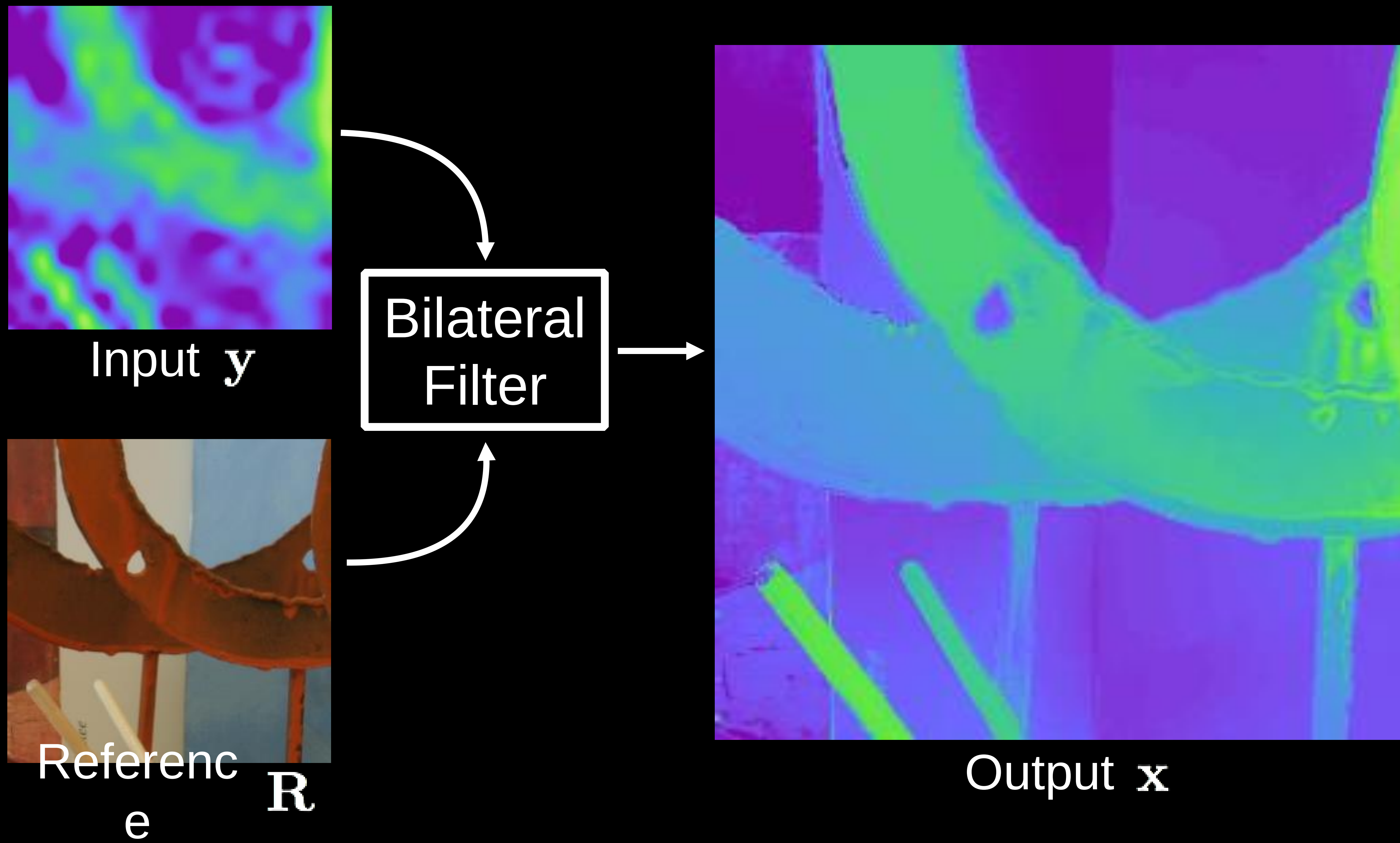
ECCV 2016

Bilateral Filter



Blur the input while respected edges in the reference image.

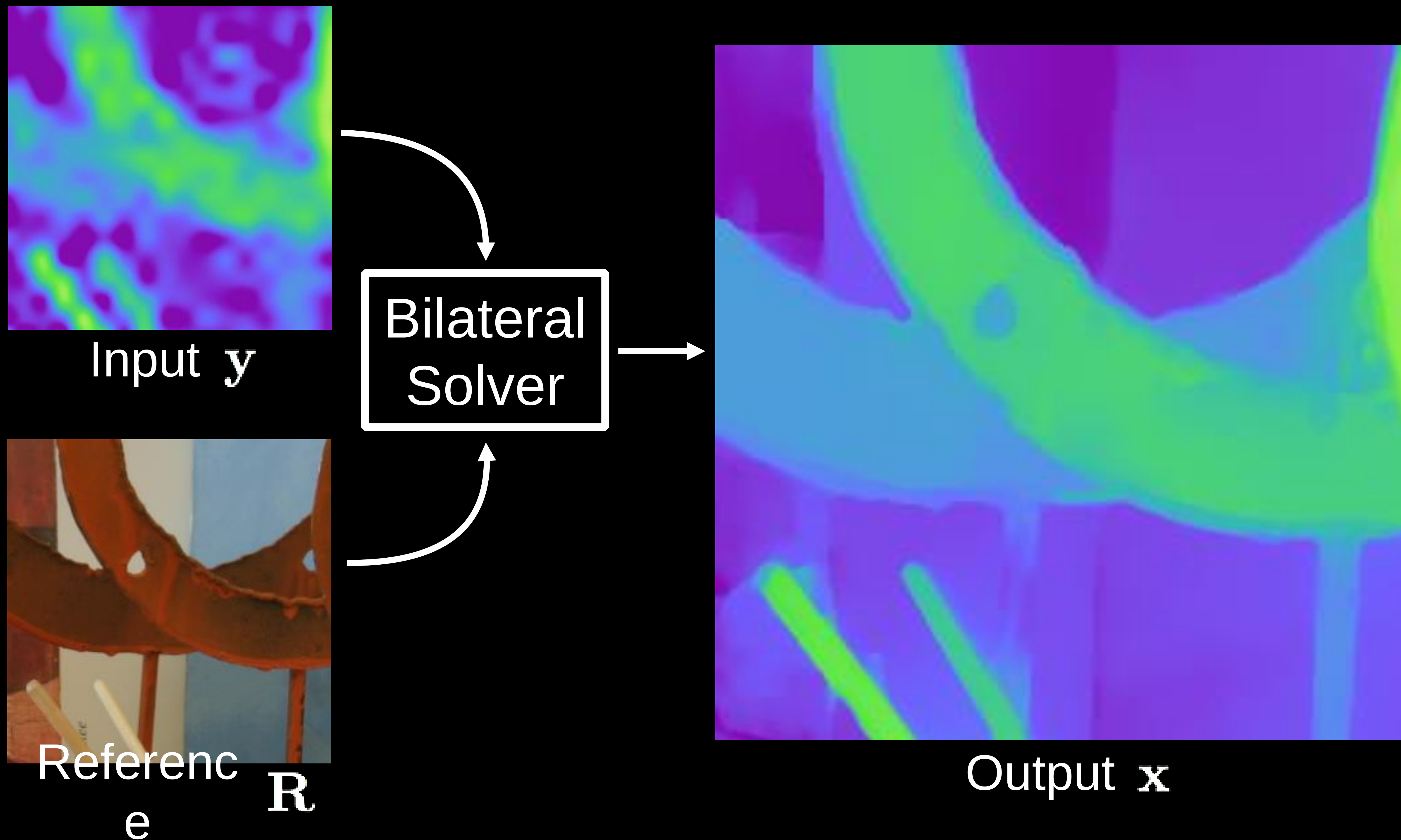
Bilateral Filter



$$\mathbf{x} \leftarrow \mathbf{W}\mathbf{y}$$

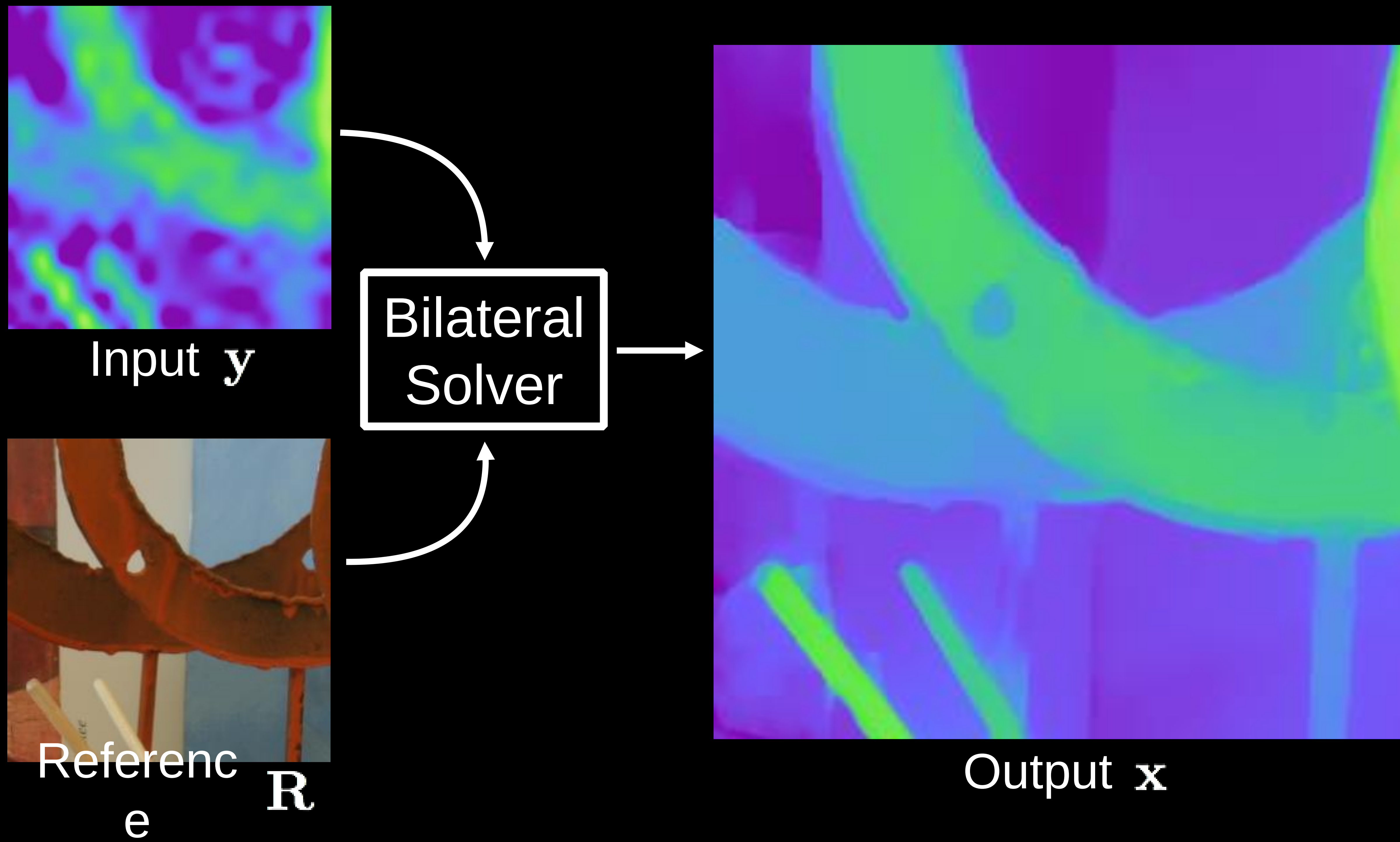
$$W_{i,j} = \exp\left(- \sum_{d} \frac{(R_{i,d} - R_{j,d})^2}{2\sigma_d^2} \right)$$

Bilateral Solver



Find the image that is as smooth as possible with respect to the reference image, and as close as possible to the input.

Bilateral Solver



$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i (x_i - y_i)^2$$

Bilateral Solver

The bilateral filter is
one gradient descent step
in the bilateral solver.

(assuming: step size = 1, λ = 1, initial state = $\mathbf{y}$)

Bilateral Solver

Bilateral Solver

Fast - Can be reformulated to be roughly as fast as a fast bilateral filter.

Bilateral Solver

Fast - Can be reformulated to be roughly as fast as a fast bilateral filter.

General - Can be easily generalized into weighted, robust, low rank, and differentiable variants.

Bilateral Solver

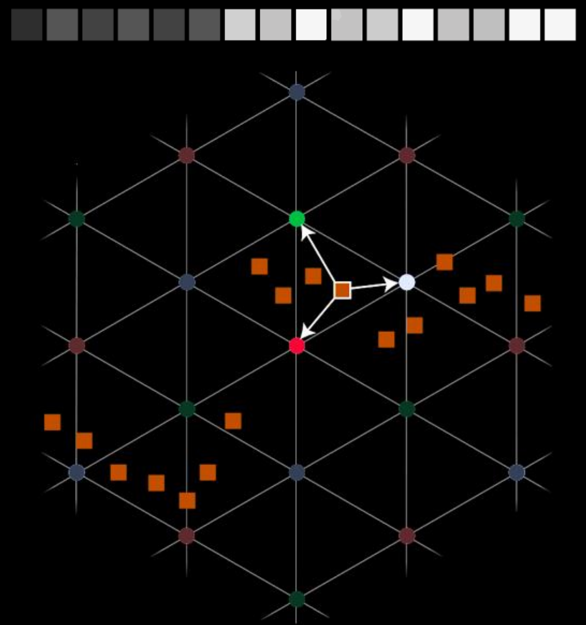
Fast - Can be reformulated to be roughly as fast as a fast bilateral filter.

General - Can be easily generalized into weighted, robust, low rank, and differentiable variants.

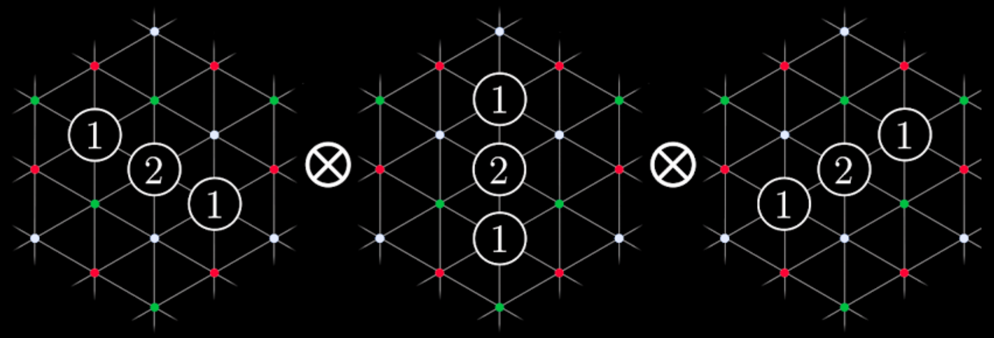
Effective - Performs well on a variety of tasks:
depth superresolution
colorization
stereo
semantic segmentation, etc.

Review: **Bilateral-Space Optimization**

Input

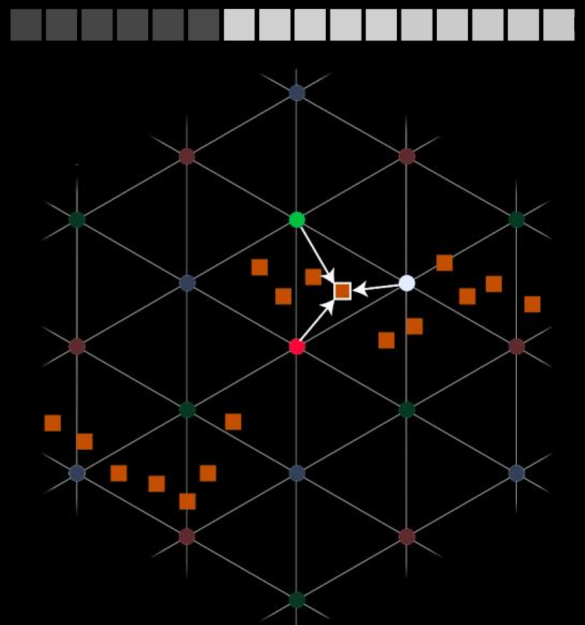


Splat
resample from pixels into “bilateral-space”



Blur
apply a series of blurs in bilateral-space

Output

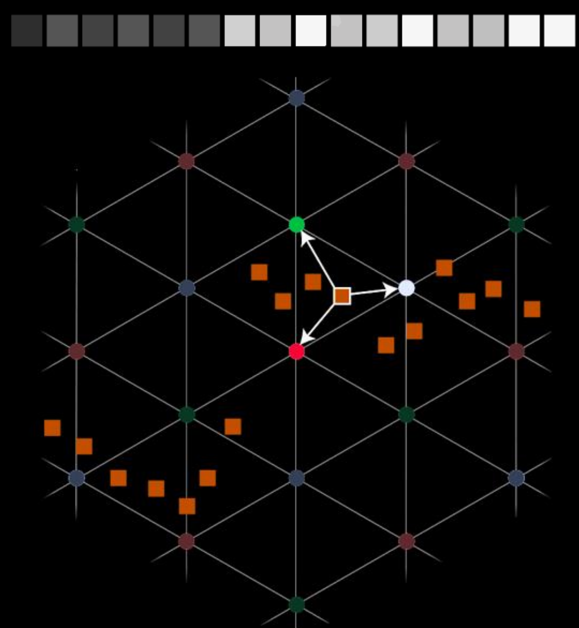


Slice
resample back into pixel-space

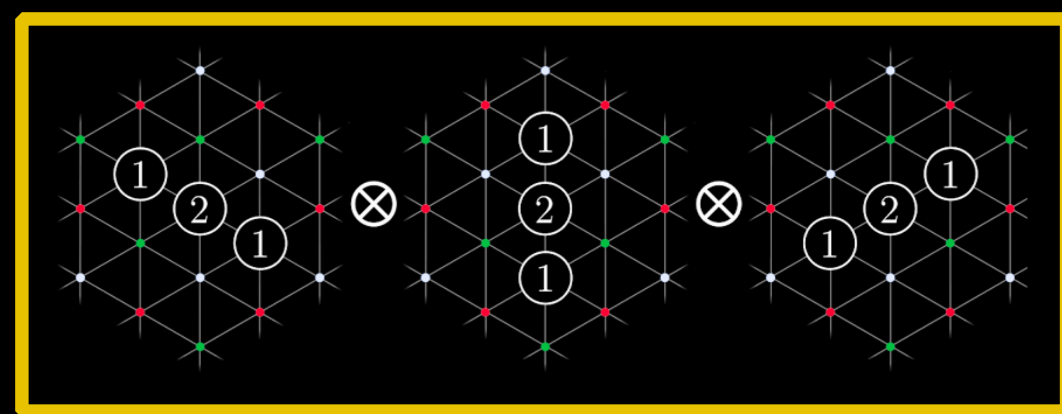
Adams et al, Eurographics 2010

Review: **Bilateral-Space Optimization**

Input

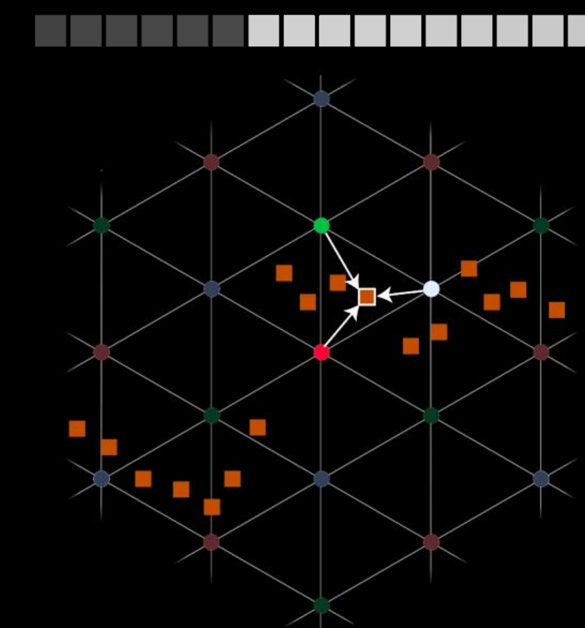


Splat
resample from pixels
into “bilateral-space”



Blur
apply a series of blurs in
bilateral-space

Output



Slice
resample back into
pixel-space

Adams et al, Eurographics 2010

We can solve optimization problems based on
bilateral kernels in this “bilateral space”

Barron et al, CVPR 2015
Märki et al, CVPR 2016

Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_{d} \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$

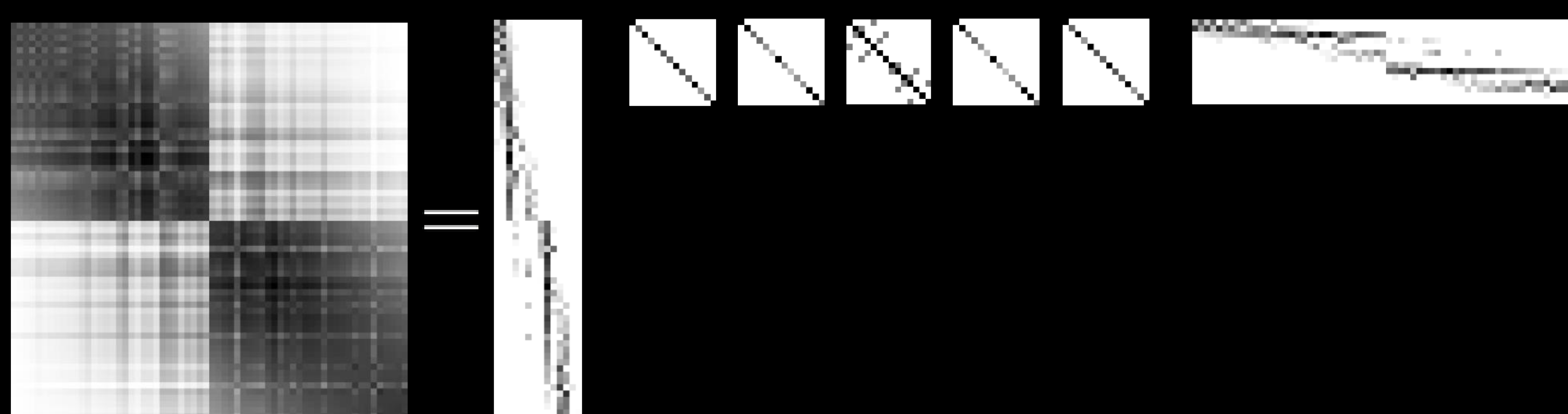
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_{d} \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



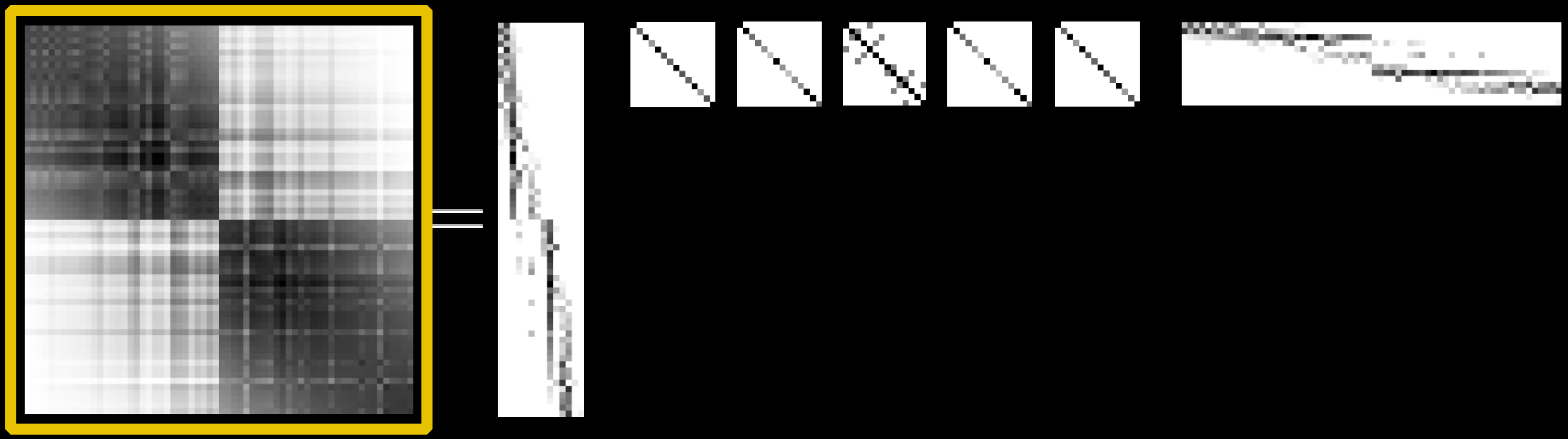
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



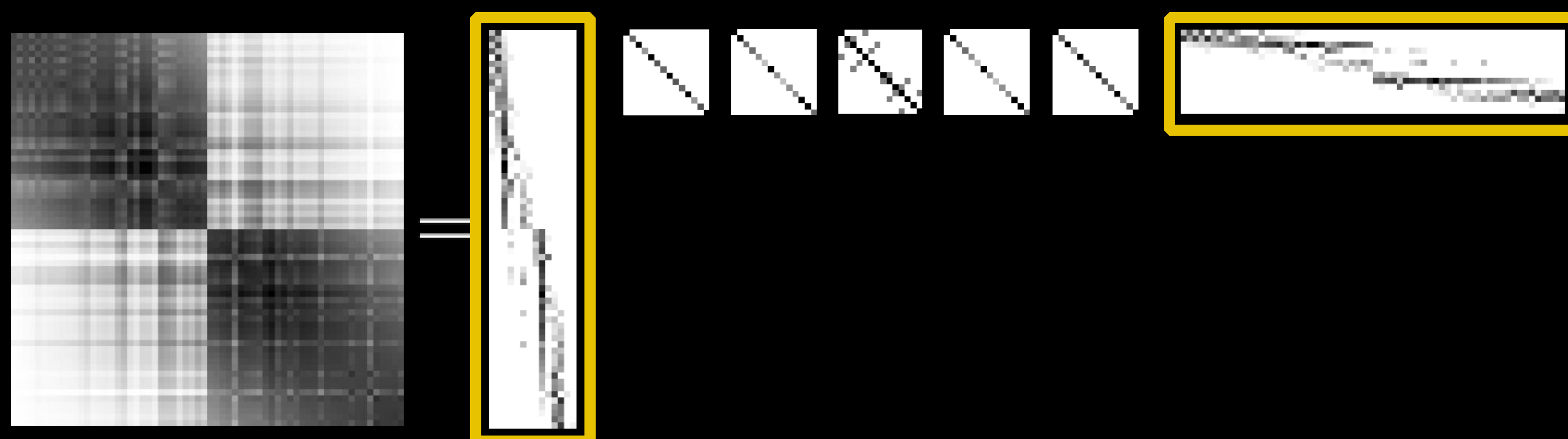
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathrm{D}_{\mathrm{m}}^{-1}\mathrm{D}_{\mathrm{n}}\mathbf{B}\mathrm{D}_{\mathrm{n}}\mathrm{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



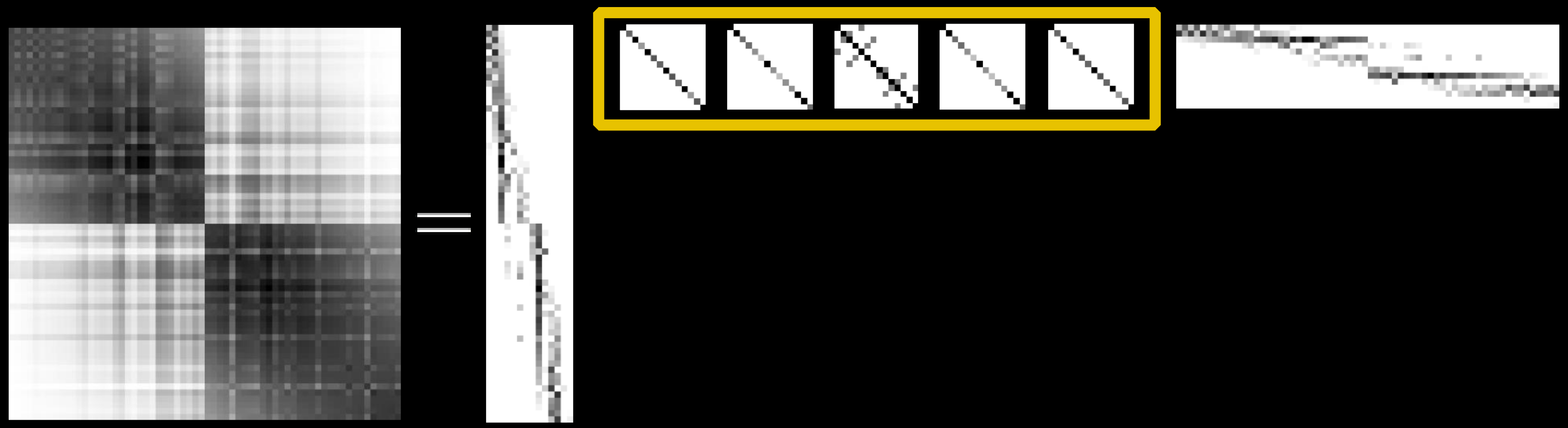
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathrm{D}_{\mathrm{m}}^{-1}\mathrm{D}_{\mathrm{n}}\mathrm{B}\mathrm{D}_{\mathrm{n}}\mathrm{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



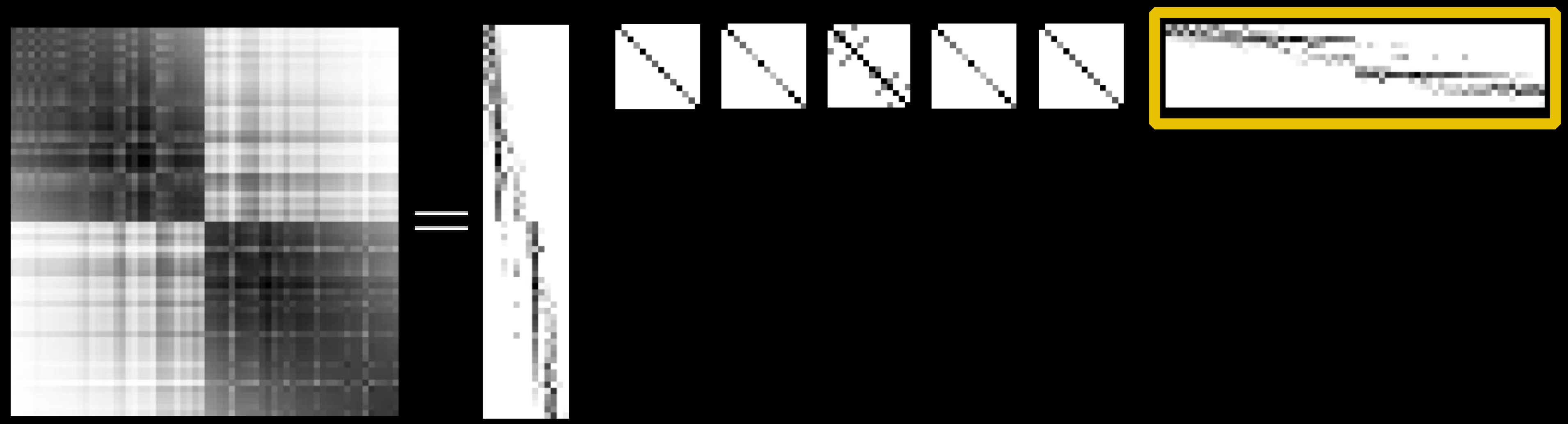
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_{d} \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



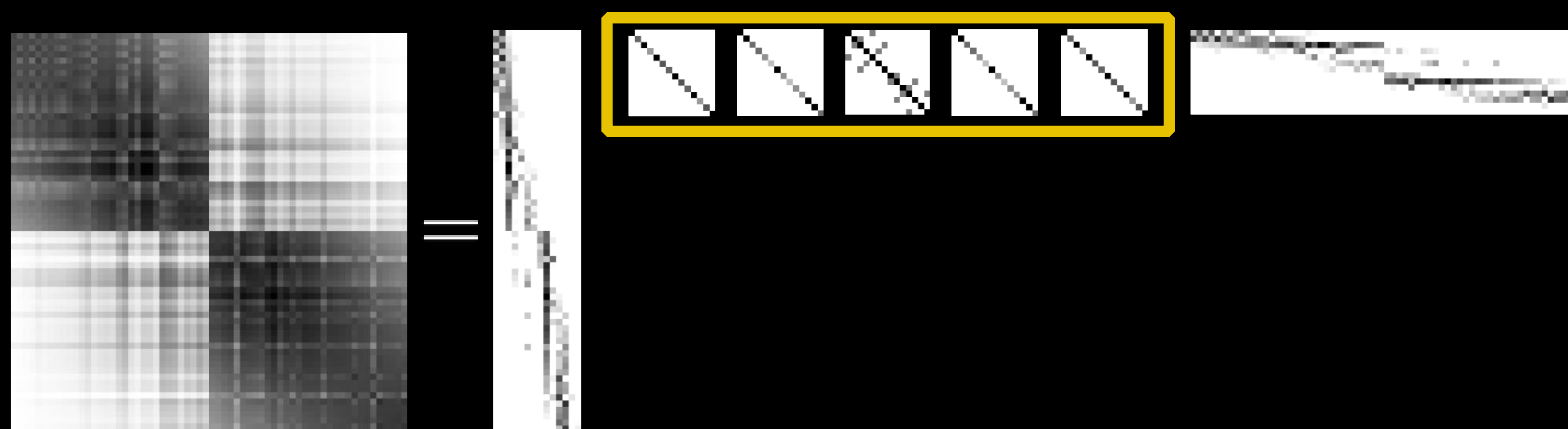
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_{d} \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathrm{D}_{\mathrm{m}}^{-1}\mathrm{D}_{\mathrm{n}}\mathbf{B}\mathrm{D}_{\mathrm{n}}\mathrm{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



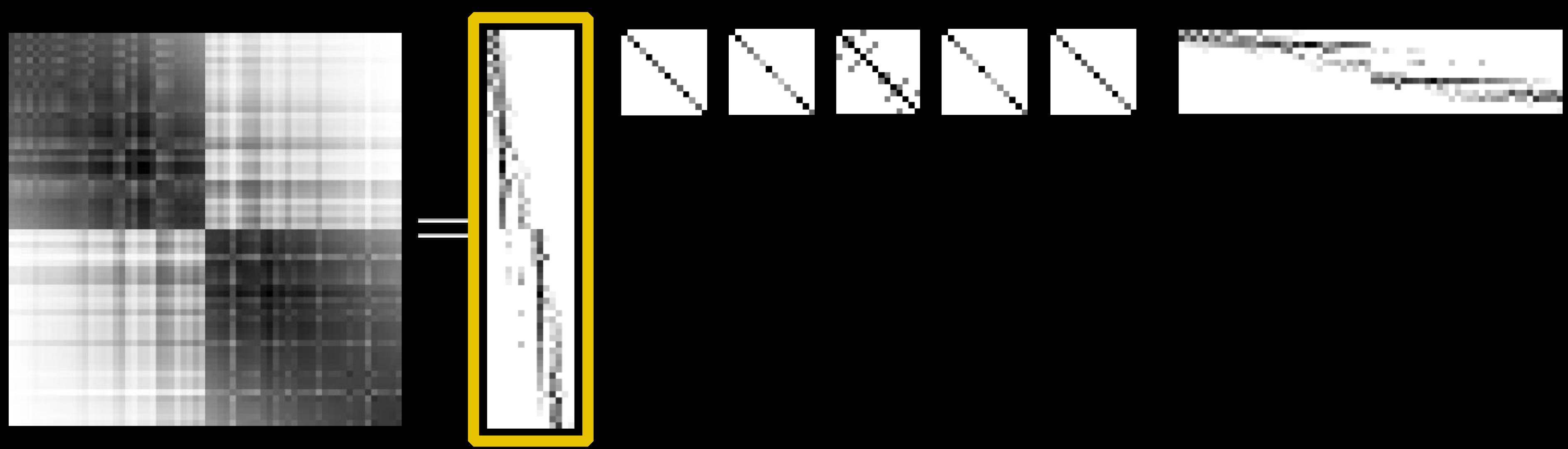
Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_{d} \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$



Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2}\sum_{i,j} W_{i,j}\left(x_i - x_j\right)^2 + \sum_i (x_i - y_i)^2$$

Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(-\sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2}\right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\mathbf{D}_{\mathrm{m}}^{-1}\mathbf{S}$$

$$\mathbf{A} = \lambda\left(\mathbf{D}_{\mathrm{m}} - \mathbf{D}_{\mathrm{n}}\mathbf{B}\mathbf{D}_{\mathrm{n}}\right) + \mathbf{S}\vec{1}$$

$$\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}}\left(\mathrm{A}^{-1}\left(\mathbf{S}\mathbf{y}\right)\right)$$

Bilateral Solver

If we know that :

$$W_{i,j} = \exp\left(- \sum_d \frac{(R_{i,d} - R_{j,d})^2}{2\sigma^2} \right)$$

We can show that:

$$\mathbf{W} = \mathbf{S}^{\mathrm{T}} \mathbf{D}_{\mathrm{m}}^{-1} \mathbf{D}_{\mathrm{n}} \mathbf{B} \mathbf{D}_{\mathrm{n}} \mathbf{D}_{\mathrm{m}}^{-1} \mathbf{S}$$

$$\mathbf{A} = \lambda \left(\mathbf{D}_{\mathrm{m}} - \mathbf{D}_{\mathrm{n}} \mathbf{B} \mathbf{D}_{\mathrm{n}} \right) + \mathbf{S}\vec{1}$$

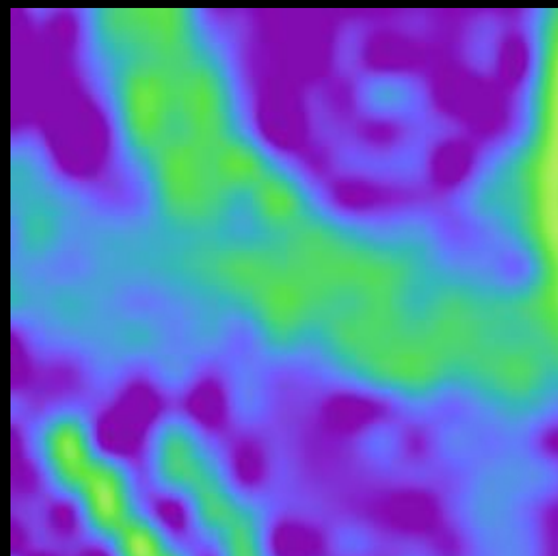
$$\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{S}\mathbf{y} \right) \right)$$

(see paper for details)

Depth Superresolution

task from Ferstl et al ICCV 2013, data from the Middlebury stereo dataset

Depth Superresolution

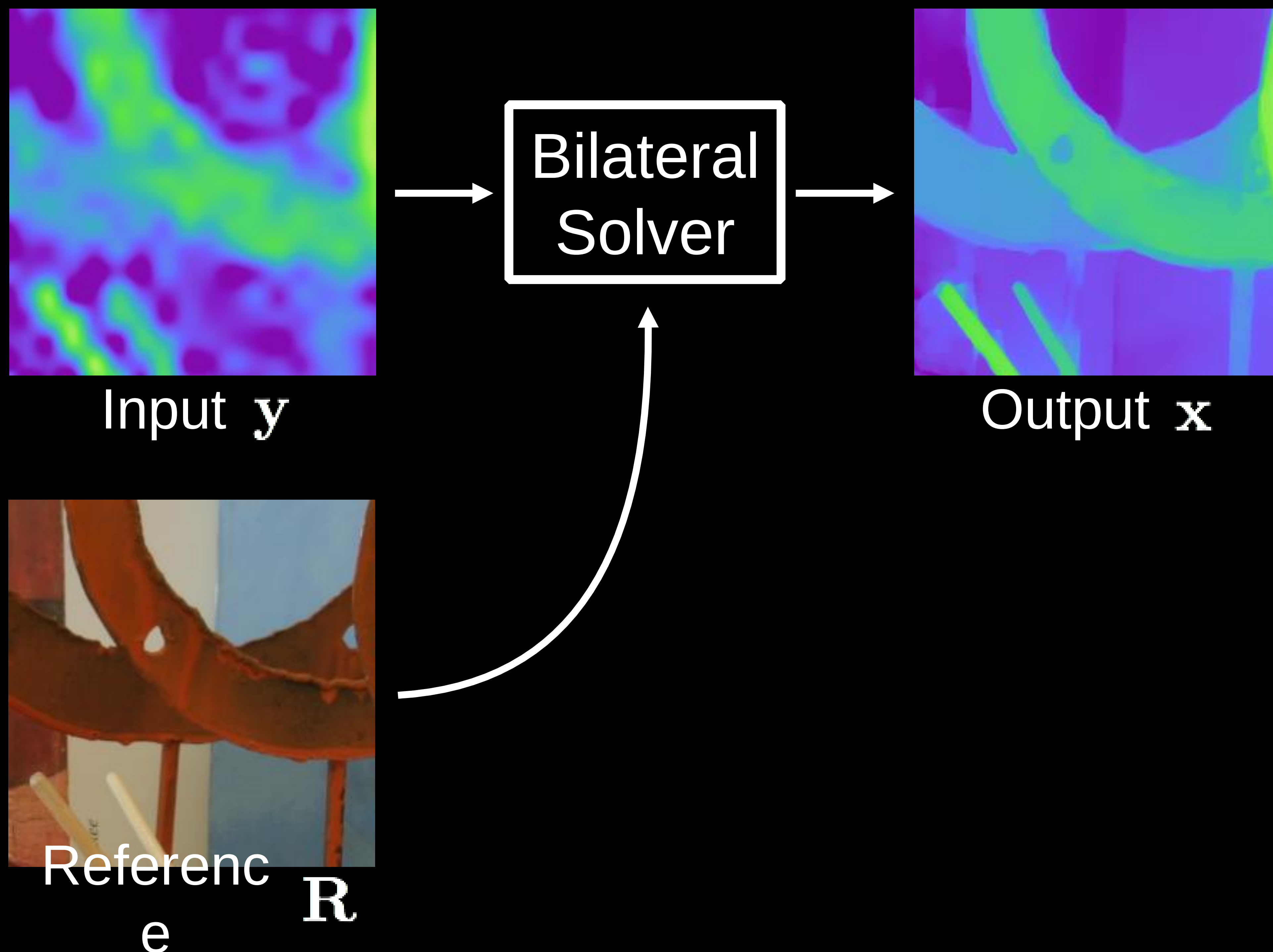


Input $\mathbf{y}$

Reference $\mathbf{R}$

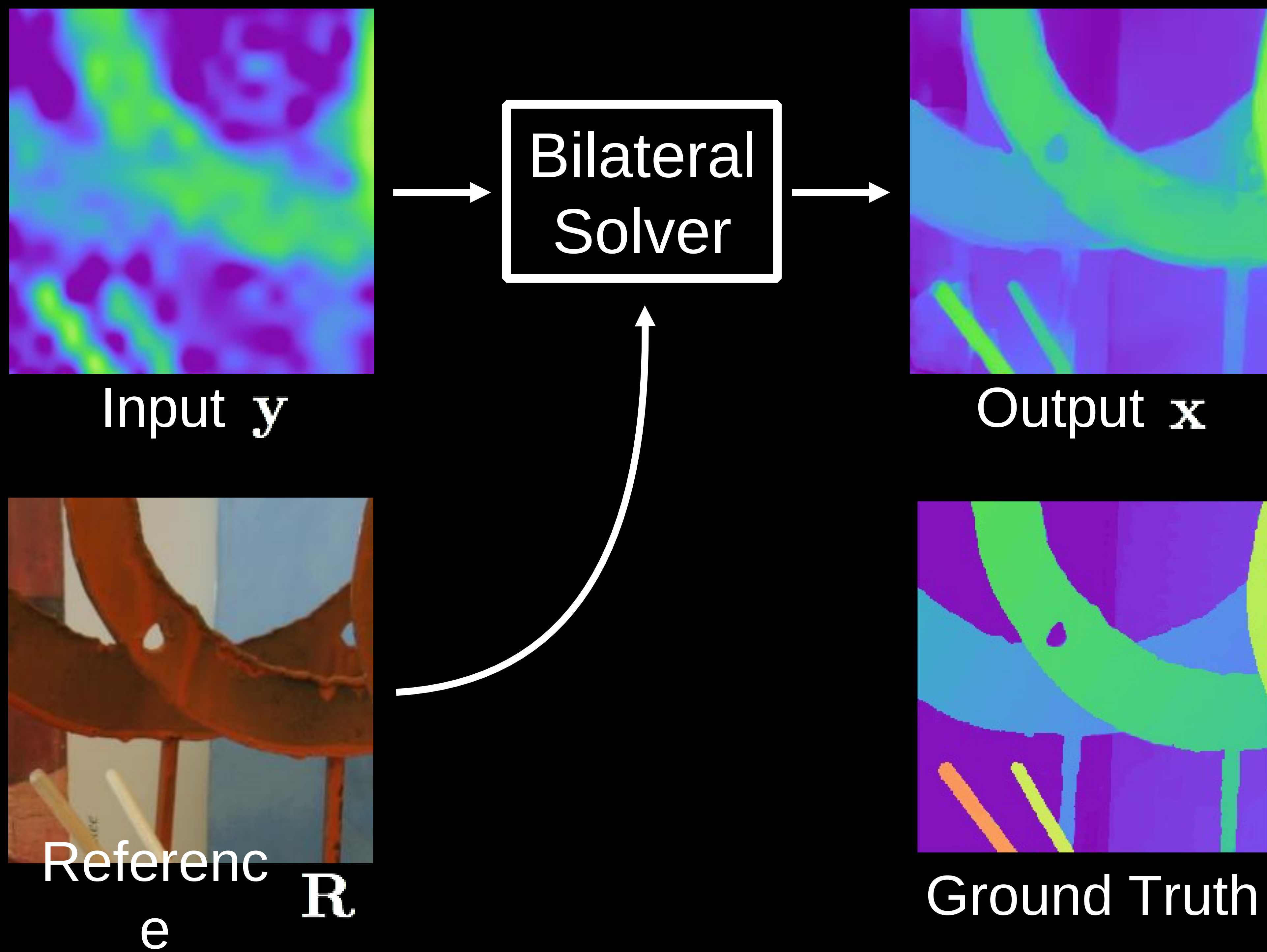
task from Ferstl et al ICCV 2013, data from the Middlebury stereo dataset

Depth Superresolution

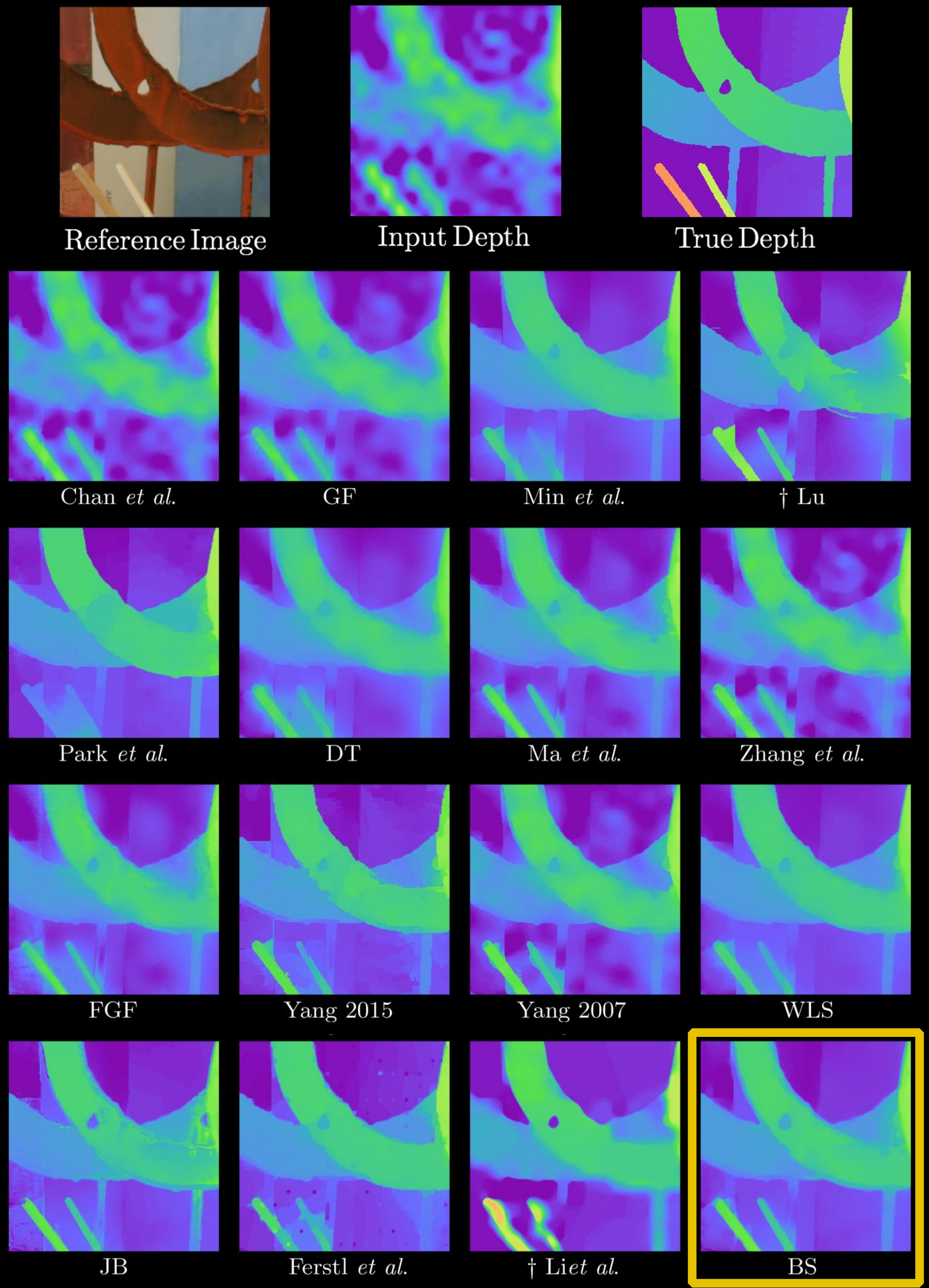


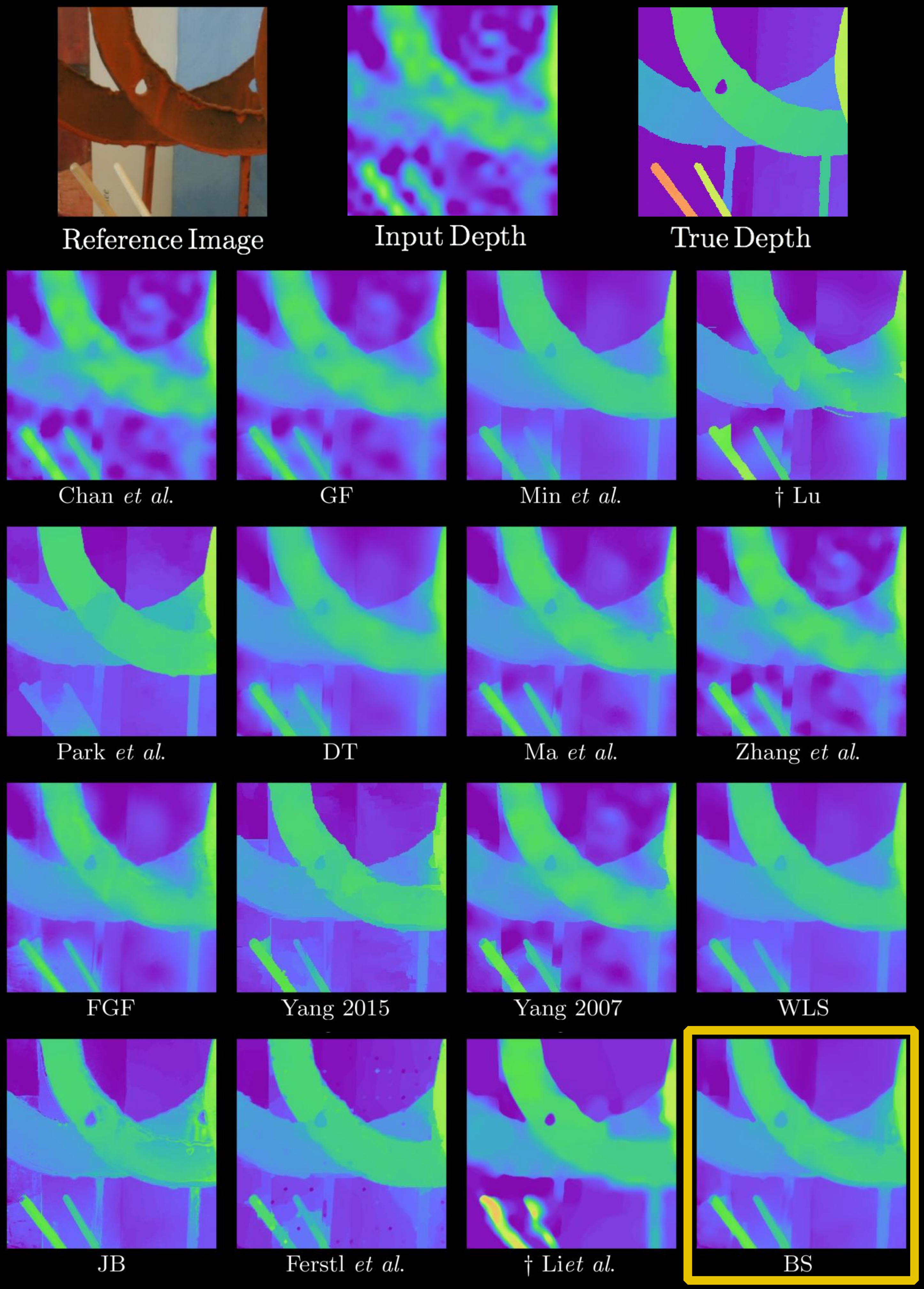
task from Ferstl et al ICCV 2013, data from the Middlebury stereo dataset

Depth Superresolution

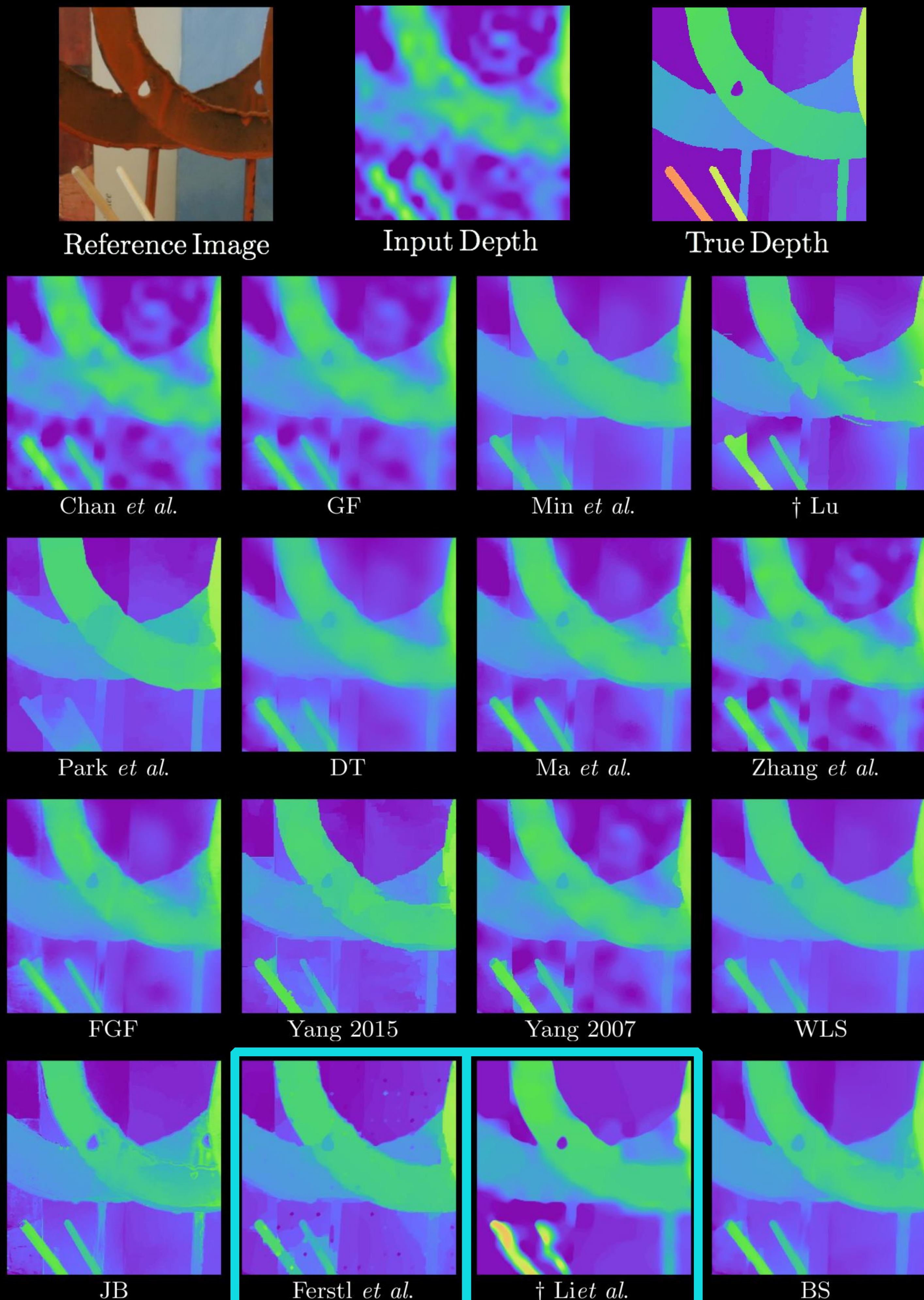


task from Ferstl et al ICCV 2013, data from the Middlebury stereo dataset

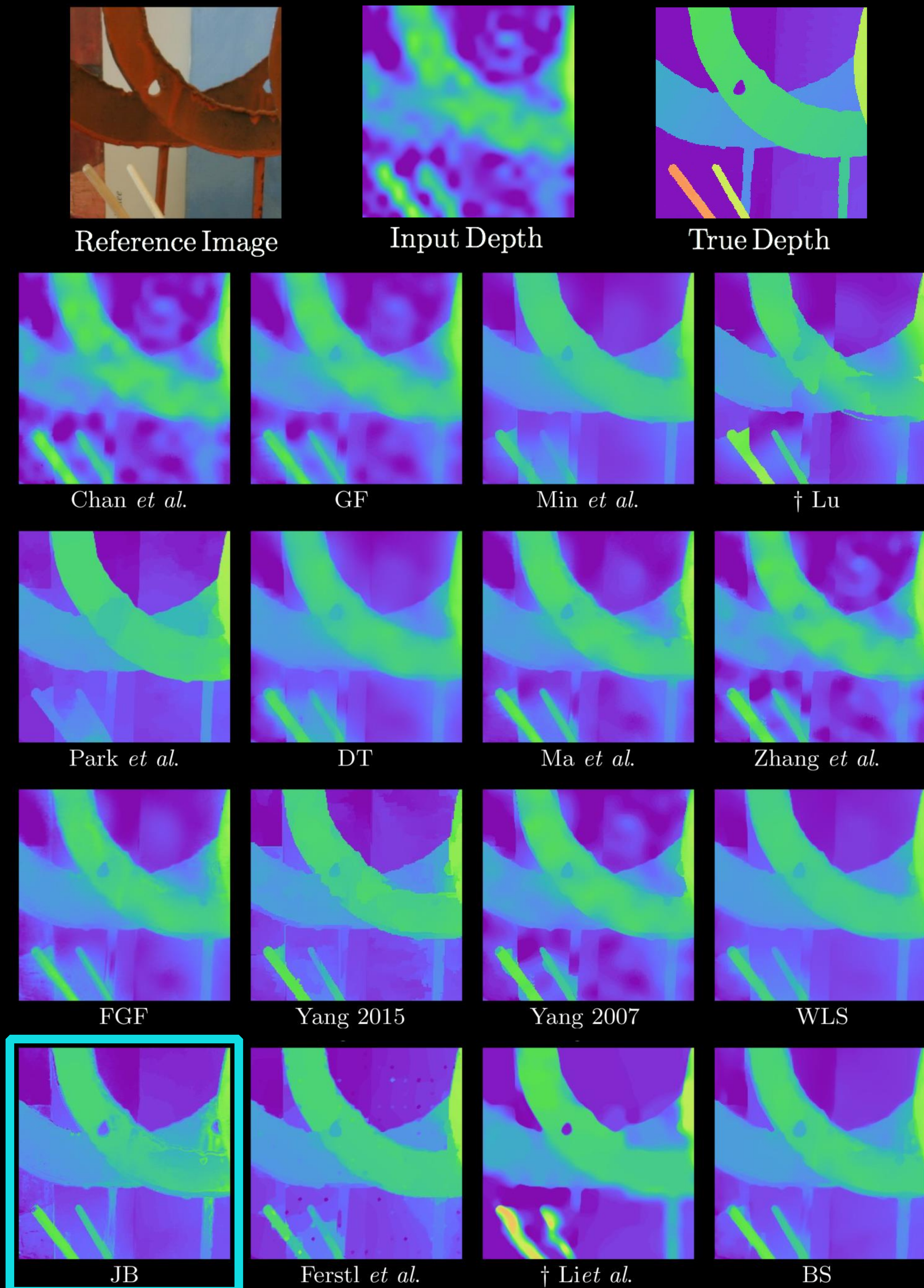

Reference Image
Input Depth
True Depth
Chan et al.
GF
Min et al.
† Lu
Park et al.
DT
Ma et al.
Zhang et al.
FGF
Yang 2015
Yang 2007
WLS
JB
Ferstl et al.
† Liet al.
BS



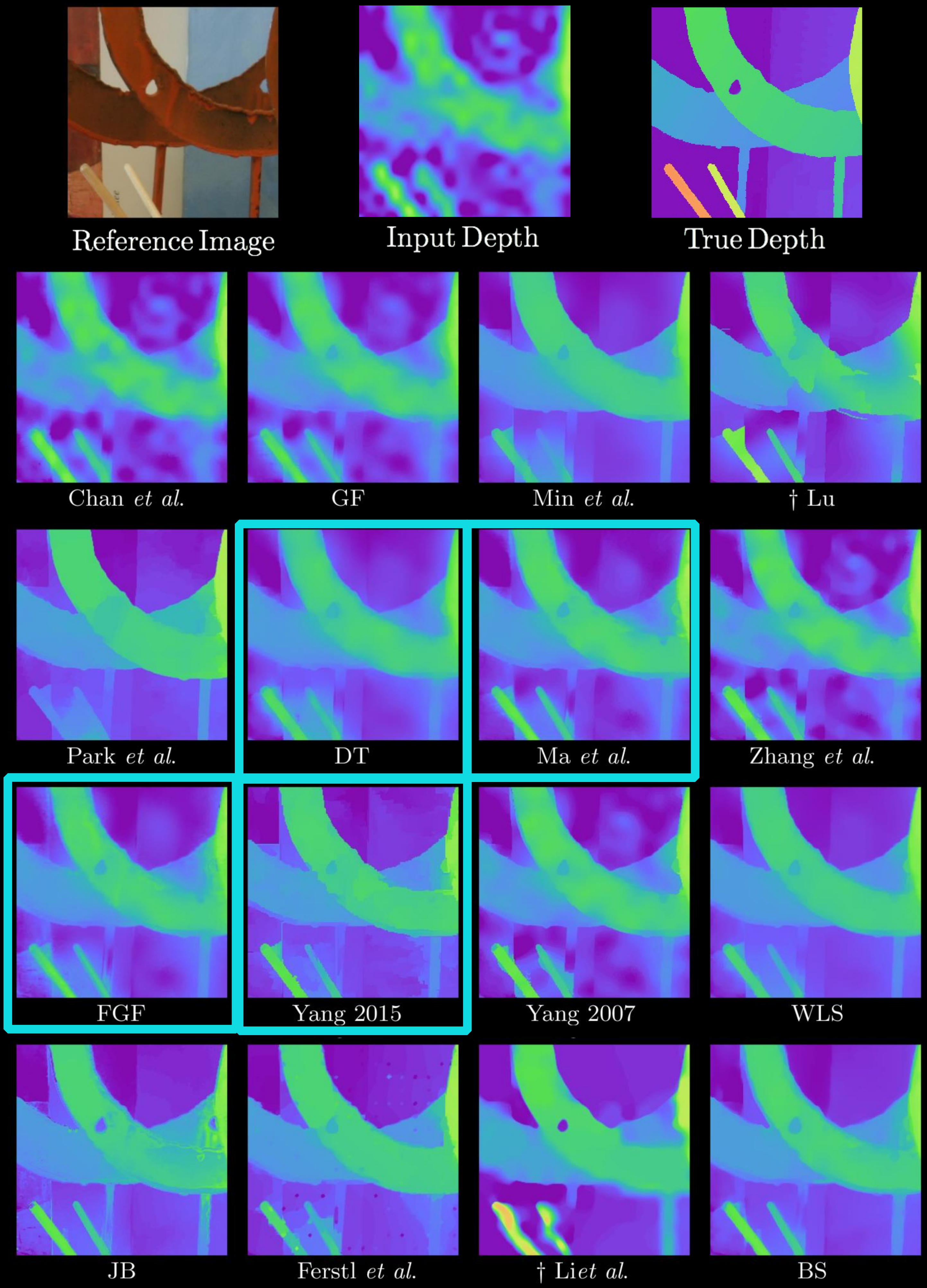
Method	Err	Time (sec)
Nearest Neighbor	7.26	0.003
Bicubic	5.91	0.007
Kiechle *et al.*	5.86	450
Bilinear	5.16	0.004
Liu *et al.*	5.10	16.60
Shen *et al.*	4.24	31.48
Diebel & Thrun	3.98	–
Chan *et al.*	3.83	3.02
Guided Filter	3.76	23.89
Min *et al.*	3.74	0.383
Lu & Forsyth	3.69	20
Park *et al.*	3.61	24.05
Domain Transform	3.56	0.021
Ma *et al.*	3.49	18
GuidedFilter (Matlab)	3.47	0.434
Zhang *et al.*	3.45	1.346
Fast Guided Filter	3.41	0.225
Yang 2015	3.41	0.304
Yang *et al.* 2007	3.25	–
Farbman *et al.*	3.19	6.11
Joint Bilateral Upsample	3.14	1.98
Ferstl *et al.*	2.93	140
Li *et al.*	2.56	700
Kwon *et al.*	1.21	300
Bilateral Solver	2.70	0.234



Method	Err	Time (sec)
Nearest Neighbor	7.26	0.003
Bicubic	5.91	0.007
Kiechle *et al.*	5.86	450
Bilinear	5.16	0.004
Liu *et al.*	5.10	16.60
Shen *et al.*	4.24	31.48
Diebel & Thrun	3.98	–
Chan *et al.*	3.83	3.02
Guided Filter	3.76	23.89
Min *et al.*	3.74	0.383
Lu & Forsyth	3.69	20
Park *et al.*	3.61	24.05
Domain Transform	3.56	0.021
Ma *et al.*	3.49	18
GuidedFilter (Matlab)	3.47	0.434
Zhang *et al.*	3.45	1.346
Fast Guided Filter	3.41	0.225
Yang 2015	3.41	0.304
Yang *et al.* 2007	3.25	–
Farbman *et al.*	3.19	6.11
Joint Bilateral Upsample	3.14	1.98
Ferstl *et al.*	2.93	140
Li *et al.*	2.56	700
Kwon *et al.*	1.21	300
Bilateral Solver	2.70	0.234



Method	Err	Time (sec)
Nearest Neighbor	7.26	0.003
Bicubic	5.91	0.007
Kiechle *et al.*	5.86	450
Bilinear	5.16	0.004
Liu *et al.*	5.10	16.60
Shen *et al.*	4.24	31.48
Diebel & Thrun	3.98	–
Chan *et al.*	3.83	3.02
Guided Filter	3.76	23.89
Min *et al.*	3.74	0.383
Lu & Forsyth	3.69	20
Park *et al.*	3.61	24.05
Domain Transform	3.56	0.021
Ma *et al.*	3.49	18
GuidedFilter (Matlab)	3.47	0.434
Zhang *et al.*	3.45	1.346
Fast Guided Filter	3.41	0.225
Yang 2015	3.41	0.304
Yang *et al.* 2007	3.25	–
Farbman *et al.*	3.19	6.11
Joint Bilateral Upsample	3.14	1.98
Ferstl *et al.*	2.93	140
Li *et al.*	2.56	700
Kwon *et al.*	1.21	300
Bilateral Solver	2.70	0.234



Method	Err	Time (sec)
Nearest Neighbor	7.26	0.003
Bicubic	5.91	0.007
Kiechle *et al.*	5.86	450
Bilinear	5.16	0.004
Liu *et al.*	5.10	16.60
Shen *et al.*	4.24	31.48
Diebel & Thrun	3.98	–
Chan *et al.*	3.83	3.02
Guided Filter	3.76	23.89
Min *et al.*	3.74	0.383
Lu & Forsyth	3.69	20
Park *et al.*	3.61	24.05
Domain Transform	3.56	0.021
Ma *et al.*	3.49	18
GuidedFilter (Matlab)	3.47	0.434
Zhang *et al.*	3.45	1.346
Fast Guided Filter	3.41	0.225
Yang 2015	3.41	0.304
Yang *et al.* 2007	3.25	–
Farbman *et al.*	3.19	6.11
Joint Bilateral Upsample	3.14	1.98
Ferstl *et al.*	2.93	140
Li *et al.*	2.56	700
Kwon *et al.*	1.21	300
Bilateral Solver	2.70	0.234

Bilateral Solver

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i (x_i - y_i)^2$$

$$\equiv$$

$$\mathbf{A} = \lambda \left(\mathbf{D}_\mathrm{m} - \mathbf{D}_\mathrm{n} \mathbf{B} \mathbf{D}_\mathrm{n}\right) + \mathbf{S}\vec{1}$$

$$\mathbf{x} \leftarrow \mathbf{S}^\mathrm{T} \left(\mathrm{A}^{-1} \left(\mathbf{S}\mathbf{y}\right)\right)$$

Weighted Bilateral Solver

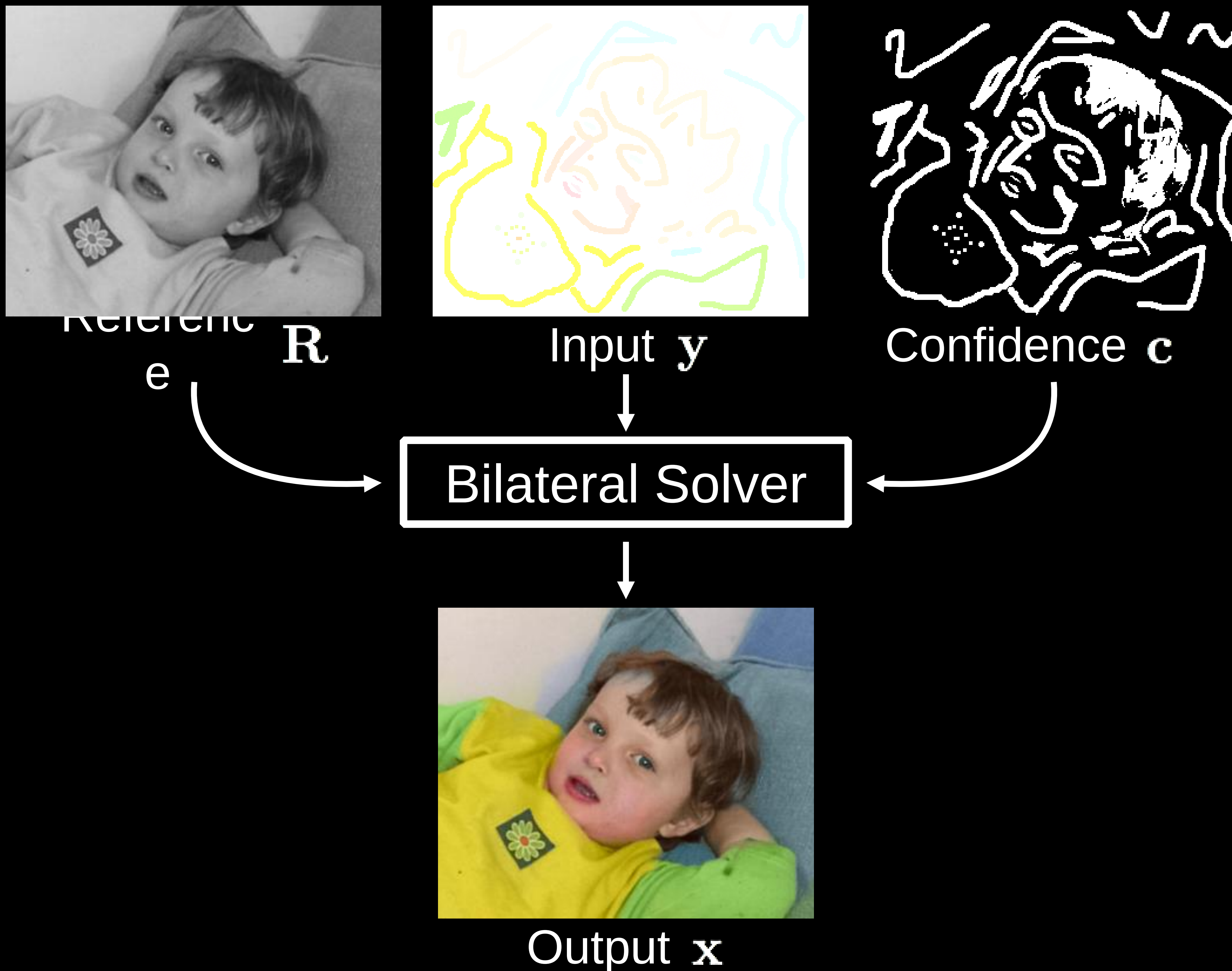
$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i c_i (x_i - y_i)^2$$

$$\equiv$$

$$\mathbf{A} = \lambda \left(\mathbf{D}_{\mathrm{m}} - \mathbf{D}_{\mathrm{n}} \mathbf{B} \mathbf{D}_{\mathrm{n}}\right) + \mathbf{Sc}$$

$$\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{Sy}\right)\right)$$

Weighted Bilateral Solver



Weighted Bilateral Solver

Bilateral Solver
0.85 sec/megapixel

Levin *et al*
81.0 sec/megapixel

95× speedup

Levin *et al,* Colorization using optimization. SIGGRAPH 2004

Weighted Bilateral Solver

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i c_i (x_i - y_i)^2$$

$$\equiv$$

$$\mathbf{A} = \lambda \left(\mathbf{D}_{\mathrm{m}} - \mathbf{D}_{\mathrm{n}} \mathbf{B} \mathbf{D}_{\mathrm{n}}\right) + \mathbf{Sc}$$

$$\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{Sy}\right)\right)$$

Robust Bilateral Solver

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i \rho(x_i - y_i)$$

$$\equiv$$

$$\begin{aligned}
&\textbf{while} \text{ not converged :} \\
&\qquad \mathbf{c} \leftarrow \frac{\rho'(\mathbf{x} - \mathbf{y})}{\mathbf{x} - \mathbf{y}} \\
&\qquad \mathbf{A} = \lambda \left(\mathbf{D}_{\mathrm{m}} - \mathbf{D}_{\mathrm{n}} \mathbf{B} \mathbf{D}_{\mathrm{n}}\right) + \mathbf{Sc} \\
&\qquad \mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{Sy}\right)\right)
\end{aligned}$$

Robust Bilateral Solver

$$\mathbf{x} \leftarrow \arg\min_{\mathbf{x}} \frac{\lambda}{2} \sum_{i,j} W_{i,j} \left(x_i - x_j\right)^2 + \sum_i \rho(x_i - y_i)$$

$$\equiv$$

$$\left.\begin{aligned}
&\textbf{while}\ \text{not converged}: \\
&\quad \mathbf{c} \leftarrow \frac{\rho'(\mathbf{x} - \mathbf{y})}{\mathbf{x} - \mathbf{y}} \\
&\quad \mathbf{A} = \lambda \left(\mathbf{D}_{\mathrm{m}} - \mathbf{D}_{\mathrm{n}} \mathbf{B} \mathbf{D}_{\mathrm{n}}\right) + \mathbf{S}\mathbf{c} \\
&\quad \mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{S}\mathbf{y}\right)\right)
\end{aligned}\right\} \text{can be done in bilateral space}$$

Stereo

MC-CNN

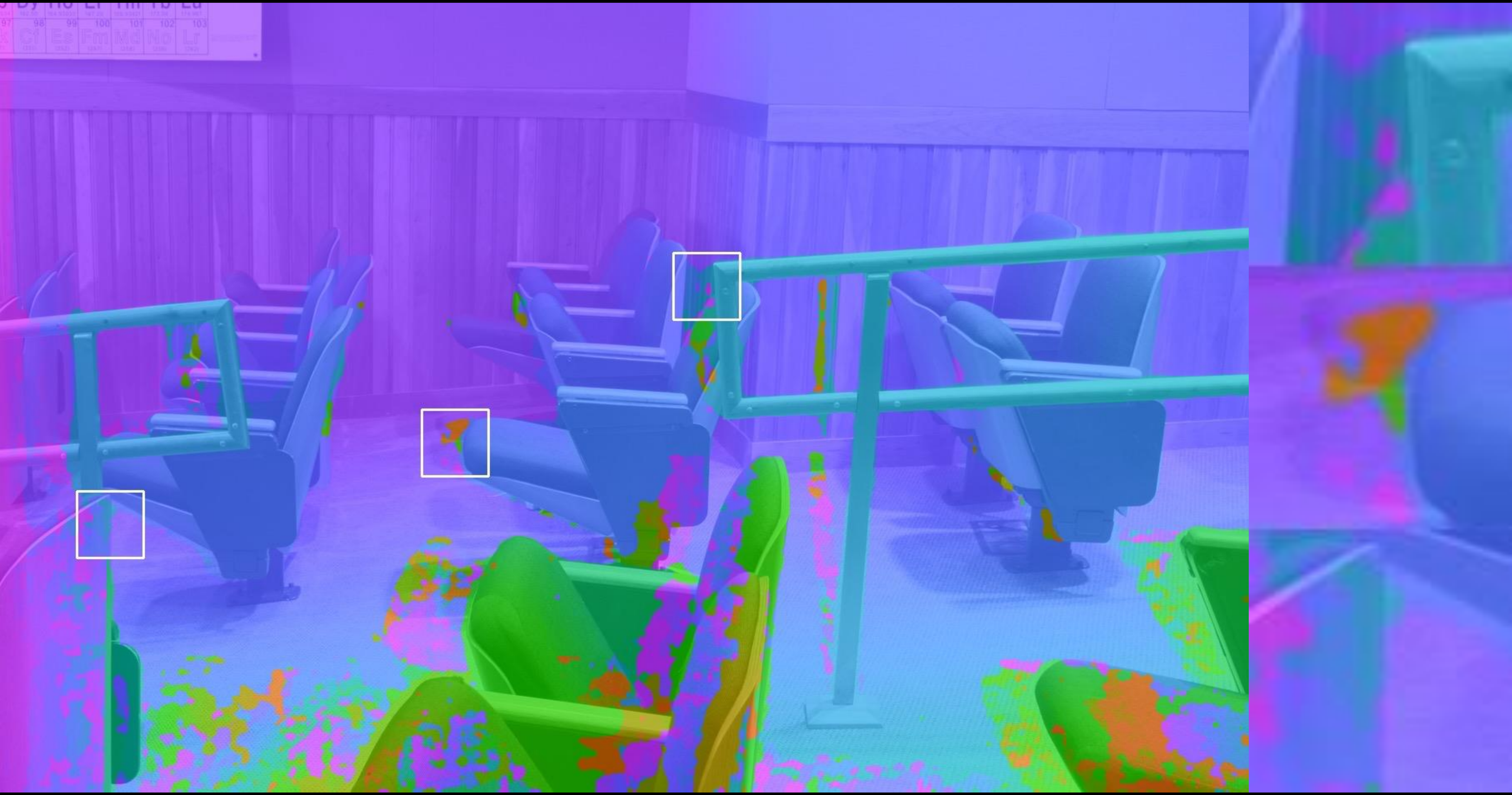
Zbontar & LeCun, CVPR 2015

Stereo

MC-CNN + Robust Bilateral Solver

Zbontar & LeCun, CVPR 2015

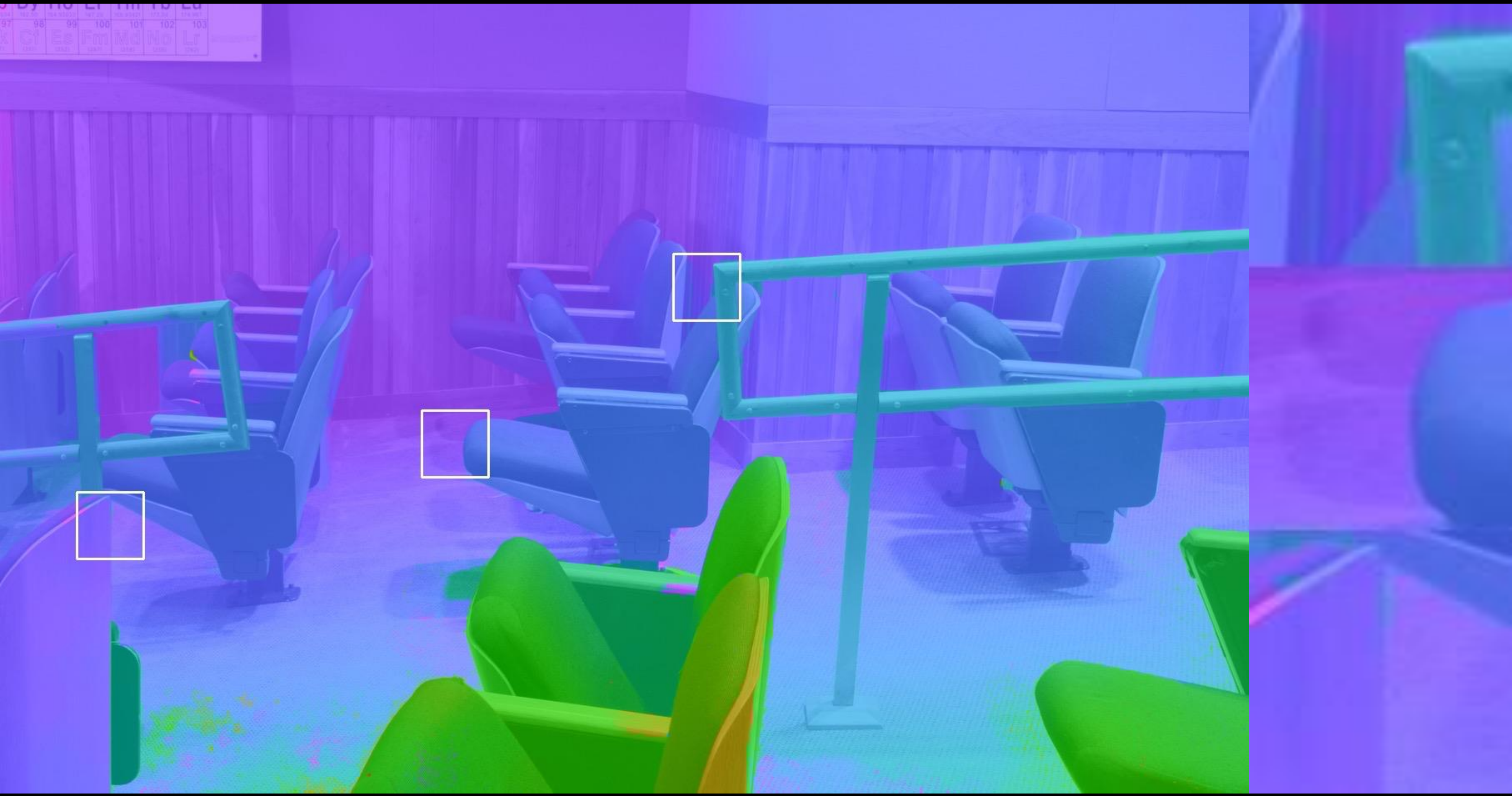
Stereo



MC-CNN

Zbontar & LeCun, CVPR 2015

Stereo



MC-CNN + Robust Bilateral Solver

Zbontar & LeCun, CVPR 2015

Stereo

Middlebury Test Set V3

Method	MAE	RMSE
MC-CNN	17.9	55.0
MC-CNN + RBS	8.19	29.9

(Lowest test-set MAE and RMSE at submission time)

Stereo

Middlebury Test Set V3

Method	MAE	RMSE
MC-CNN	17.9	55.0
MC-CNN + RBS	8.19	29.9

Middlebury Training Set V3

Method	MAE	RMSE
MC-CNN	5.93	18.36
MC-CNN + TF	5.67	16.18
MC-CNN + FGF	5.91	16.32
MC-CNN + WMF	5.30	15.62
MC-CNN + DT	5.69	16.53
MC-CNN + RBS	2.81	8.44

Stereo

Middlebury Test Set V3

Method	MAE	RMSE
MC-CNN	17.9	55.0
MC-CNN + RBS	8.19	29.9

Middlebury Training Set V3

Method	MAE	RMSE
MeshStereo	3.83	10.75
MeshStereo + TF	3.81	9.91
MeshStereo + FGF	3.96	10.03
MeshStereo + WMF	3.87	10.10
MeshStereo + DT	3.77	10.12
MeshStereo + RBS	3.22	8.72

Stereo

Middlebury Test Set V3

Method	MAE	RMSE
MC-CNN	17.9	55.0
MC-CNN + RBS	8.19	29.9

Middlebury Training Set V3

Method	MAE	RMSE
TMAP	3.98	11.55
TMAP + TF	3.94	10.90
TMAP + FGF	4.17	10.79
TMAP + WMF	4.11	11.01
TMAP + DT	3.86	10.92
TMAP + RBS	3.31	9.44

Stereo

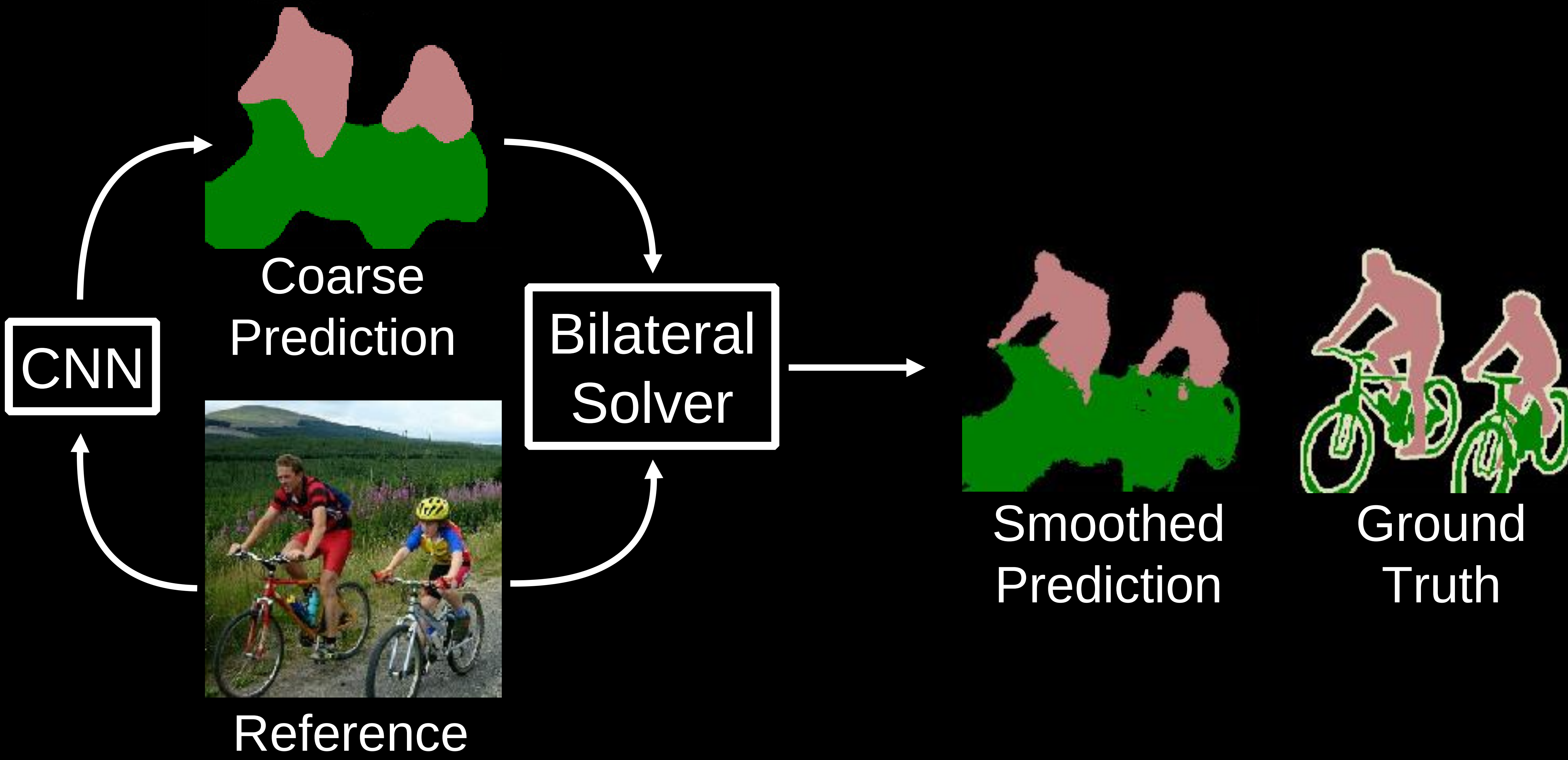
Middlebury Test Set V3

Method	MAE	RMSE
MC-CNN	17.9	55.0
MC-CNN + RBS	8.19	29.9

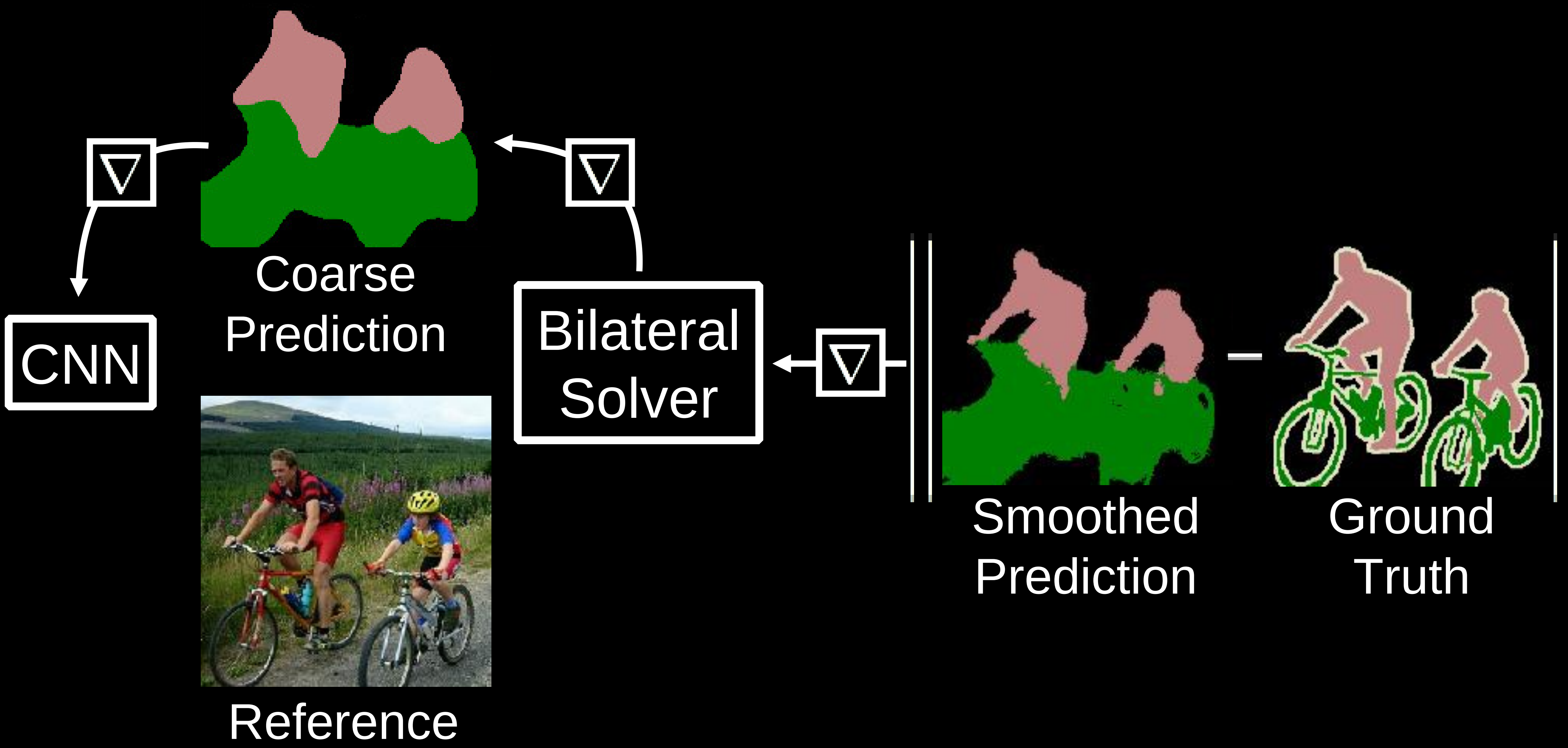
Middlebury Training Set V3

Method	MAE	RMSE
SGM	3.85	10.68
SGM + TF	3.82	9.55
SGM + FGF	4.05	9.66
SGM + WMF	3.97	9.99
SGM + DT	3.85	9.90
SGM + RBS	3.44	9.21

Bilateral Solver as a “Layer”



Bilateral Solver as a "Layer"



Bilateral Solver as a “Layer”

Forward: $\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}}\left(\mathrm{A}^{-1}\left(\mathbf{S}\mathbf{y}\right)\right)$

Bilateral Solver as a "Layer"

Forward: $\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{S} \mathbf{y} \right) \right)$

Backward: $\nabla_{\mathbf{y}} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{S} \nabla_{\mathbf{x}} \right) \right)$

Bilateral Solver as a “Layer”

Forward: $\mathbf{x} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{S}\mathbf{y} \right) \right)$

Backward: $\nabla_{\mathbf{y}} \leftarrow \mathbf{S}^{\mathrm{T}} \left(\mathrm{A}^{-1} \left(\mathbf{S} \nabla_{\mathbf{x}} \right) \right)$

No annoying bookkeeping, “unrolling”, etc.

Semantic Segmentation

Reference

Semantic Segmentation

Deeplab

Accuracy (IOU)	Runtime (ms)
Deeplab	
62.3%	58

Chen et al, ICLR 2015

Semantic Segmentation

Deeplab + DenseCRF

	Accuracy (IOU)	Runtime (ms)
Deeplab	62.3%	58
Deeplab + DenseCRF	67.6%	58 +918

Chen et al, ICLR 2015

Semantic Segmentation

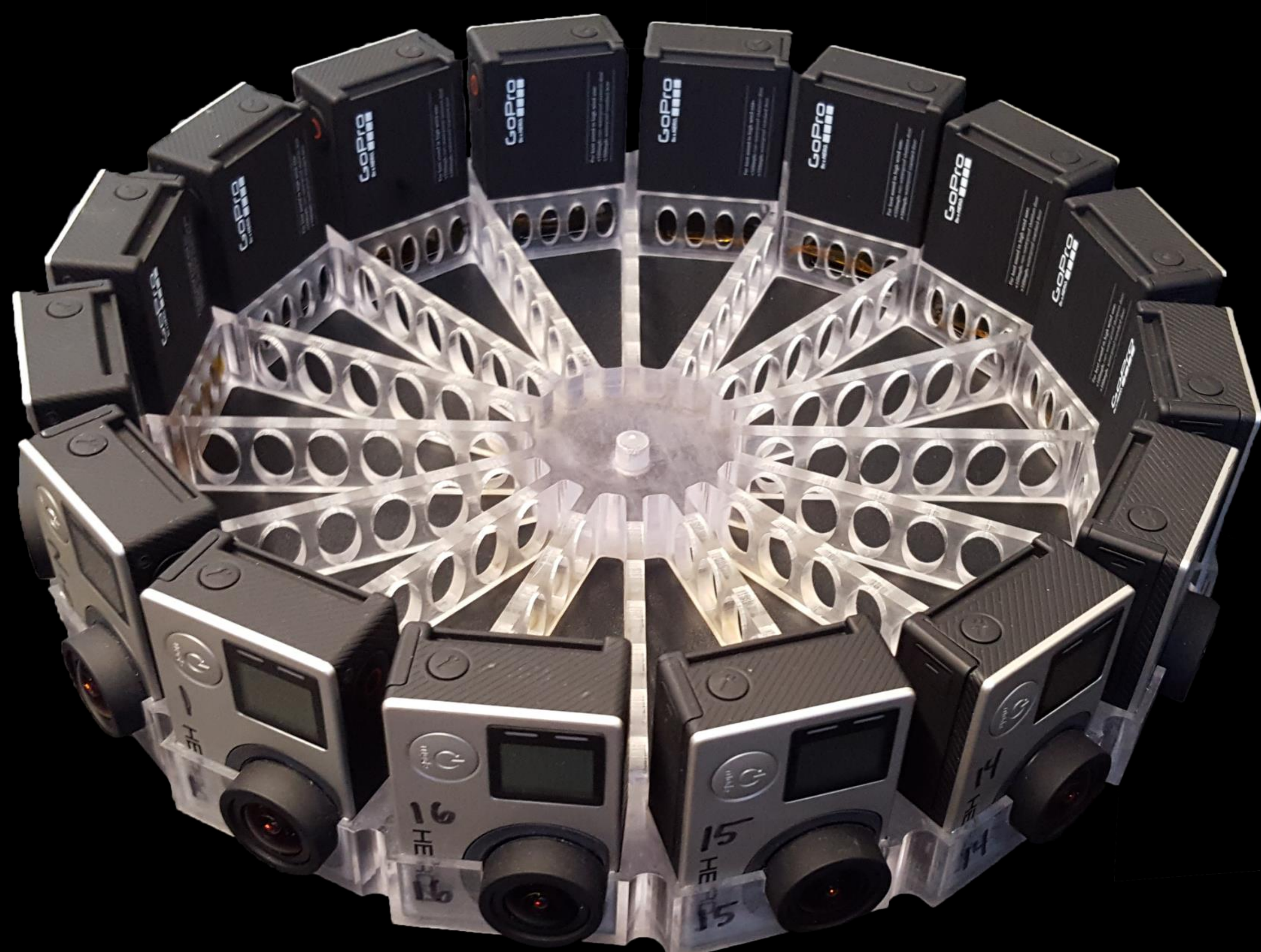
Deeplab + Bilateral Solver

Accuracy (IOU)	Runtime (ms)
Deeplab	
62.3%	58
Deeplab + DenseCRF	
67.6%	58 +918
Deeplab + Bilateral Solver	
66.0%	58 + 85

70% as accurate
11x faster

Preview: Optical Flow for VR

Anderson et al, Jump: Virtual Reality Video, SIGGRAPH Asia 2016



Preview: Optical Flow for VR

Anderson et al, Jump: Virtual Reality Video, SIGGRAPH Asia 2016

Preview: Optical Flow for VR

Anderson et al, Jump: Virtual Reality Video, SIGGRAPH Asia 2016

To Conclude

The bilateral solver is a simple, fast, and effective tool.

To Conclude

The bilateral solver is a simple, fast, and effective tool.

Nearly as fast as a bilateral filter, but significantly more accurate.

To Conclude

The bilateral solver is a simple, fast, and effective tool.

Nearly as fast as a bilateral filter, but significantly more accurate.

Works well on a variety of tasks: depth superresolution, colorization, stereo, semantic segmentation, etc.

To Conclude

The bilateral solver is a simple, fast, and effective tool.

Nearly as fast as a bilateral filter, but significantly more accurate.

Works well on a variety of tasks: depth superresolution, colorization, stereo, semantic segmentation, etc.

Can be easily integrated into deep learning pipelines.

Code Available*

*not the exact same code that was used for the paper

github.com/poolio/bilateral_solver

Thanks!