Fast Bilateral-Space Stereo for Synthetic Defocus

Jonathan T. Barron
Andrew Adams
YiChang Shih
Carlos Hernández

Google

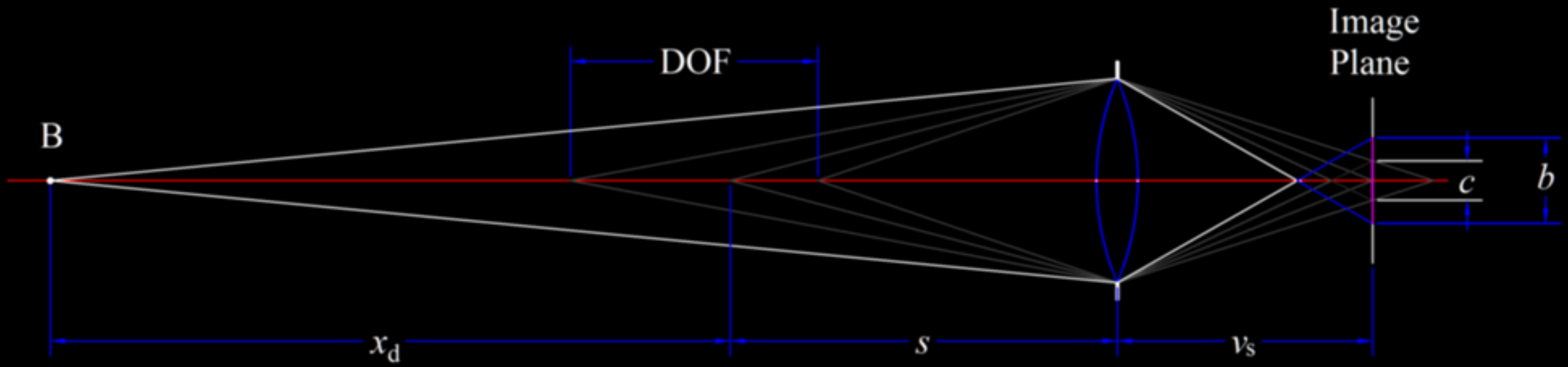
Shallow Depth of Field

Photo by Bryan Maltais

Deep Depth of Field

Photo by Bryan Maltais

Real Shallow Depth of Field



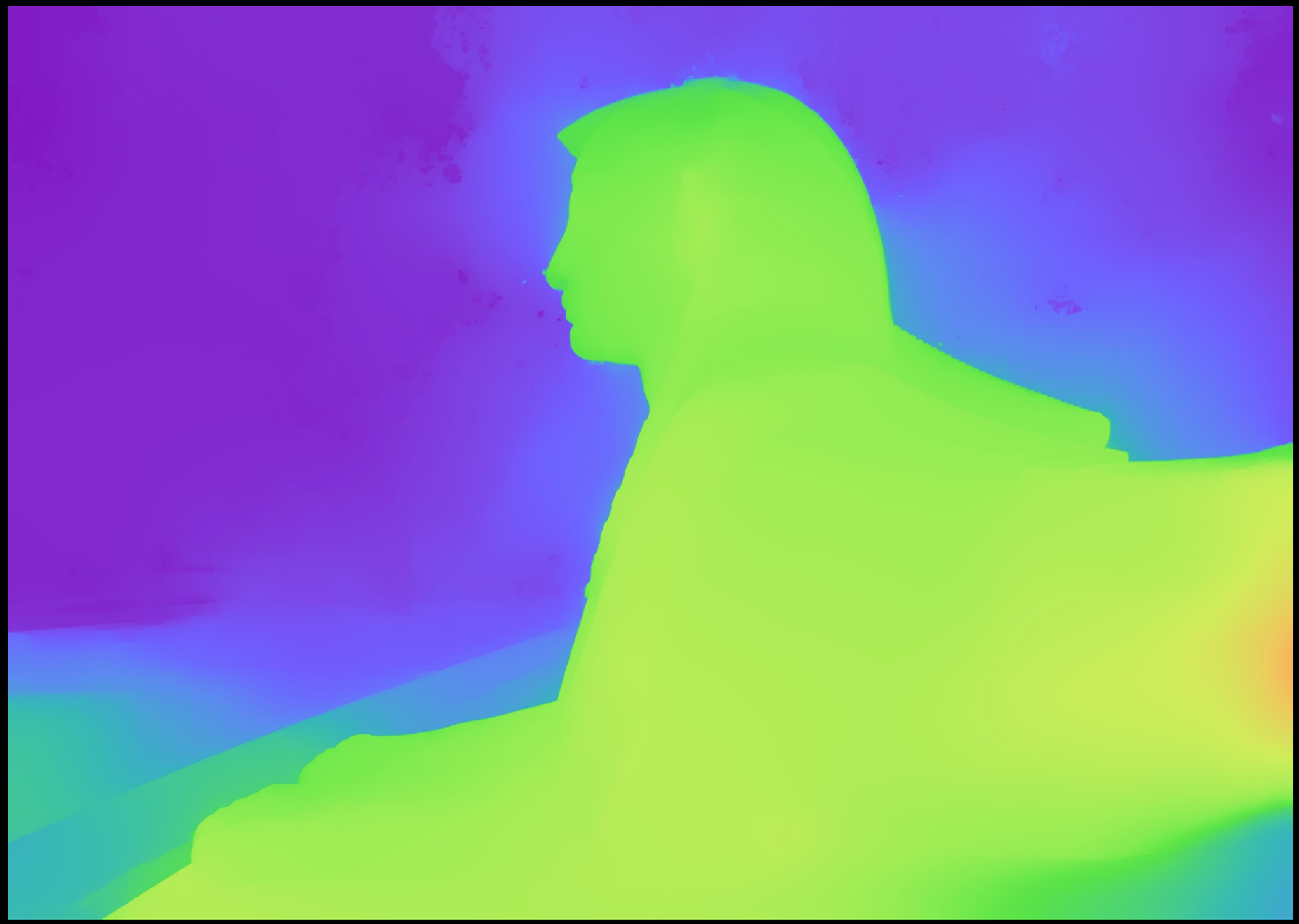
Synthetic Shallow Depth of Field

Stereo Pair

Synthetic Shallow Depth of Field

Stereo Pair

Synthetic Shallow Depth of Field



Depth Map

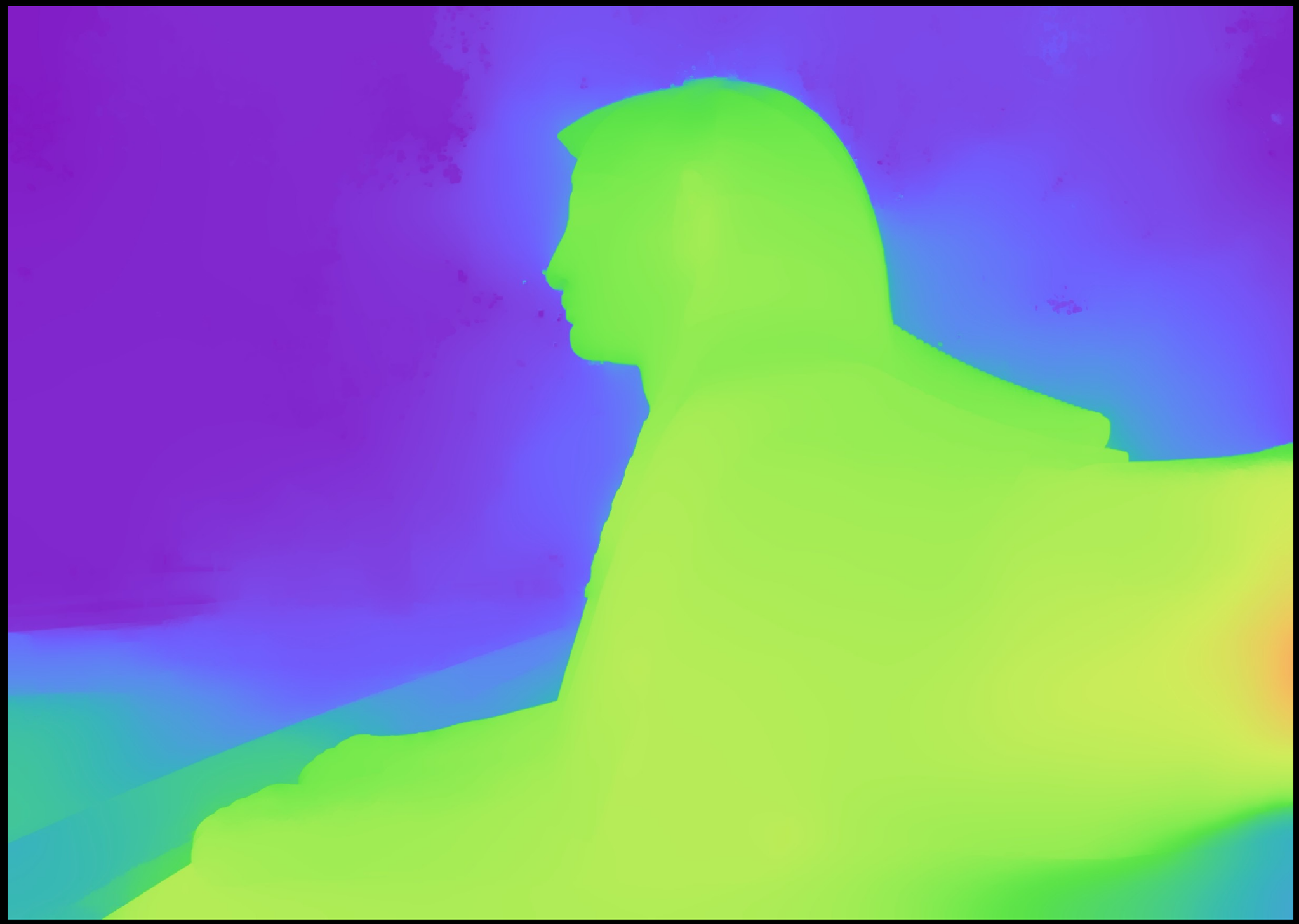
Synthetic Shallow Depth of Field

Rendered Image

Synthetic Shallow Depth of Field

Input Image

Synthetic Shallow Depth of Field



Depth Map

“Isn’t stereo solved?”

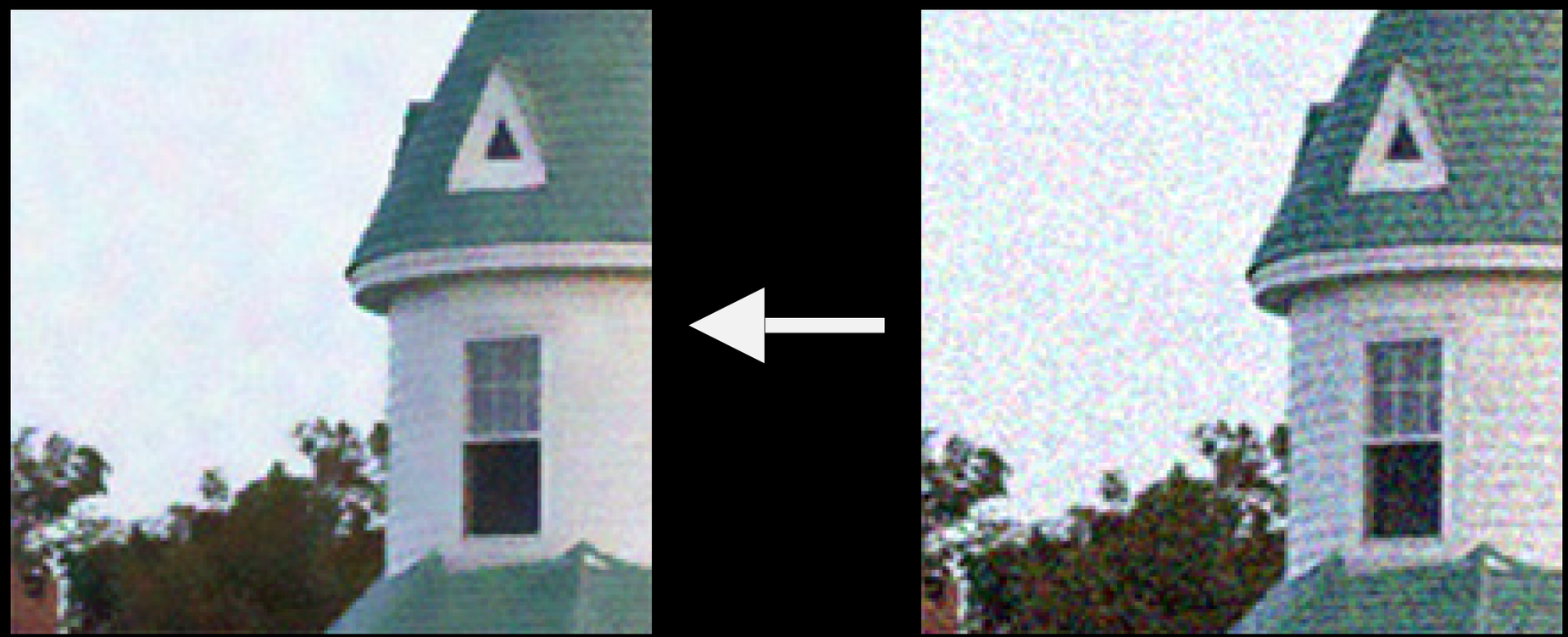
Speed: Must be fast, even on a mobile device, and even on huge images

Our algorithm is 10-100x faster than the state-of-the-art

Quality: State-of-the-art stereo algorithms only roughly follow edges in the input image

Our algorithm cannot produce depth maps which are mis-aligned to the input image

Review: Bilateral Filtering



$\text{output } y$ $\text{input } x$

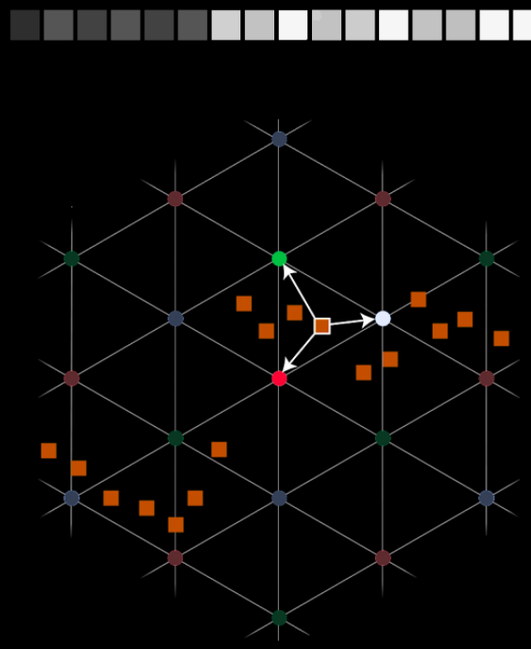
Blur within objects, not across edges

$$y = Ax$$

$$A_{i,j} = \exp\left(-\frac{\|[x_i, y_i] - [x_j, y_j]\|^2}{\sigma_{xy}^2} - \frac{\|[r_i, g_i, b_i] - [r_j, g_j, b_j]\|^2}{\sigma_{rgb}^2}\right)$$

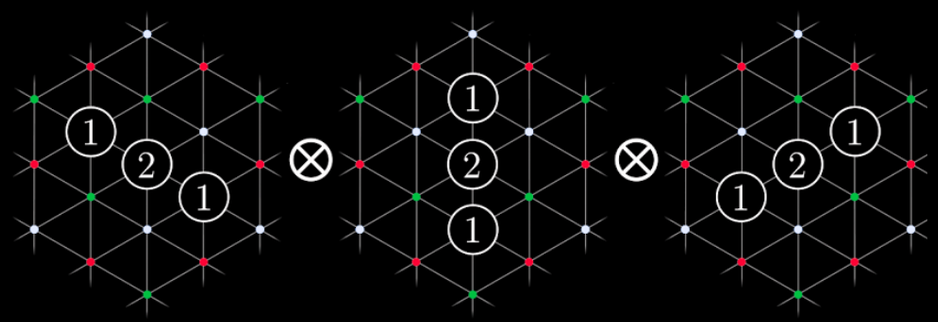
Review: Fast Bilateral Filtering

Input



Splat

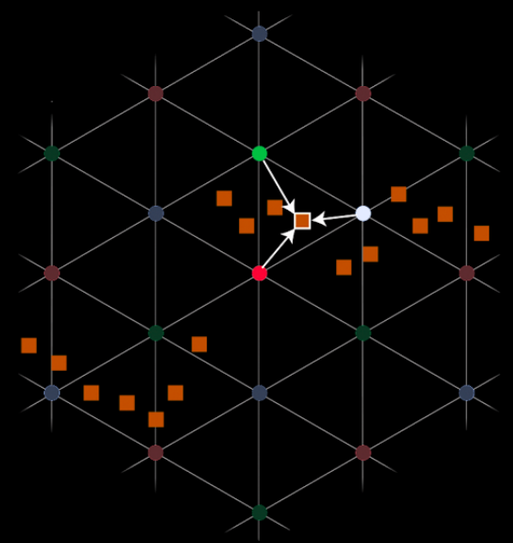
resample from pixels
into “bilateral-space”



Blur

apply a series of blurs
in bilateral-space

Output



Slice

resample back
into pixel-space

local, cheap filtering in
“bilateral-space”

≡

Non-local, edge-aware
filtering in pixel-space

A. Adams, J. Baek, and M. A. Davis
Fast high-dimensional filtering using the permutohedral lattice
Computer Graphics Forum, 2010

Past Work

To quickly apply a bilateral filter:

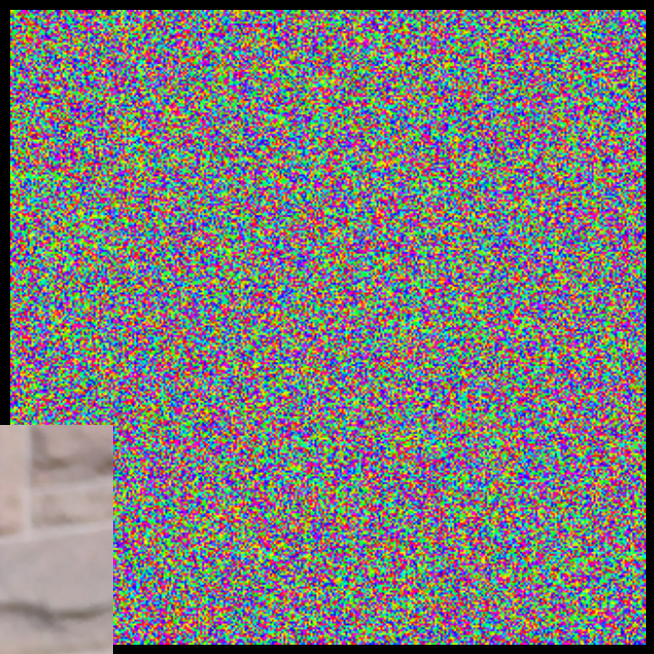
1) splat the signal
2) blur the splatted signal
3) slice out the blurred signal

Our Insight

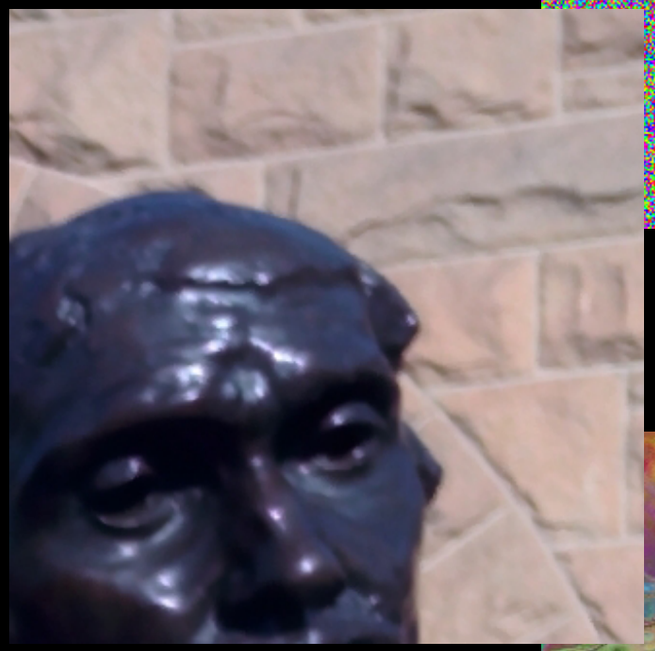
To quickly solve an optimization problem based on bilateral filtering:

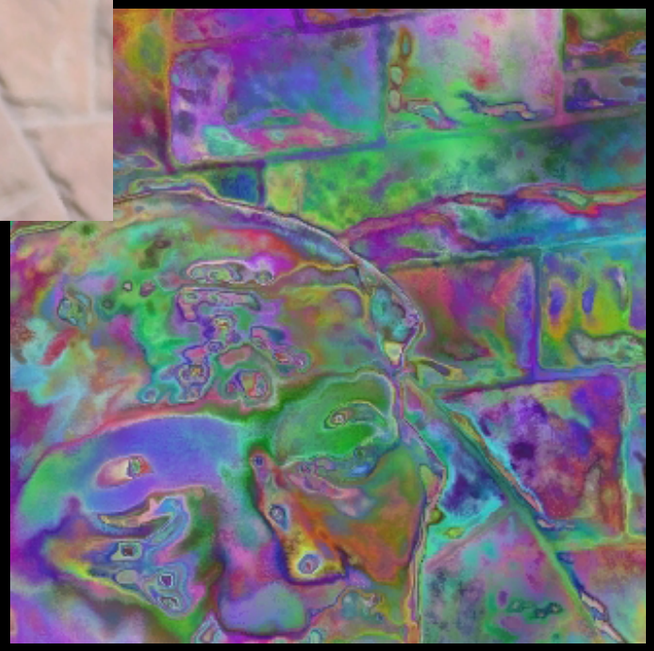
1) splat the problem
2) solve the splatted problem
3) slice out the solution

Bilateral-Space = Reparametrization



Instead of solving in “pixel space”
(each color is a parameter)



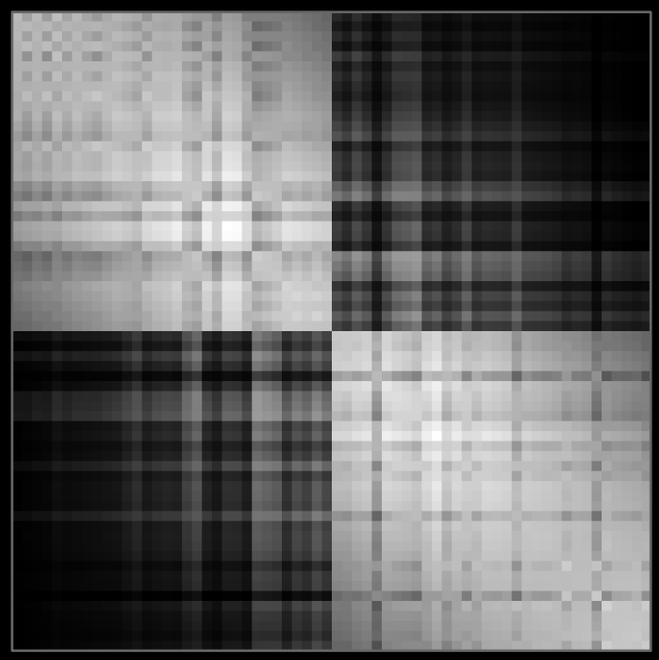


solve in “bilateral-space”
(each color is a parameter)

+ Much smaller space
+ Long-range affinities are cheap
+ Solution must be bilateral-smooth

Bilateral-Space = Matrix Decomposition

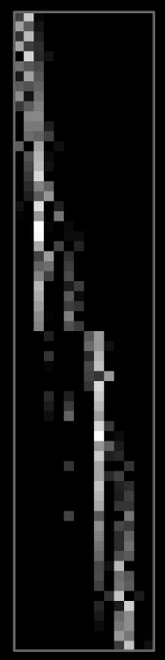
$$A = S^{\mathrm{T}} B S$$



A

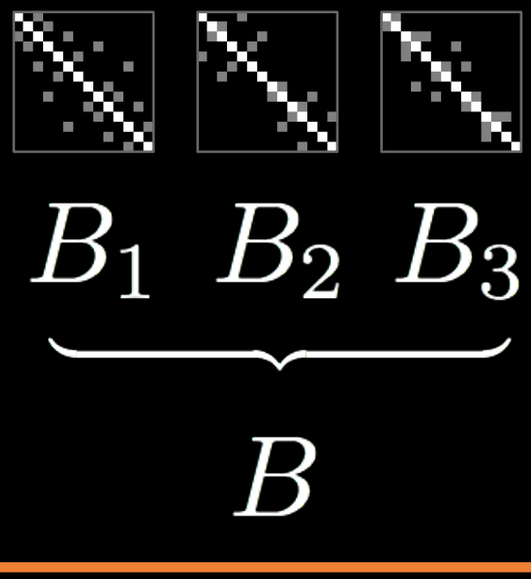
Affinity Matrix
(Huge, Dense)

$=$



S^{T}

“Slice” Matrix
(Skinny, Sparse)



$B_1 \quad B_2 \quad B_3$

B

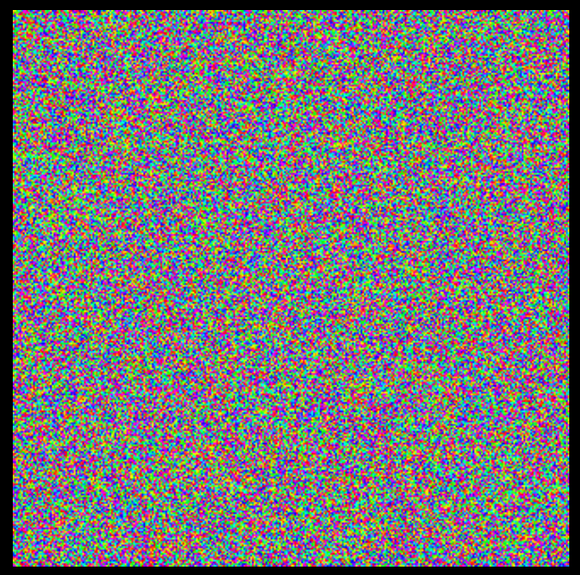
“Blur” Matrices
(Small, Sparse)

S

“Splat” Matrix
(Skinny, Sparse)

Pixel-space optimization

$$\underset{\mathbf{x}}{\text{minimize}} \quad \underbrace{\sum_i \sum_j A_{i,j} \left(x_i - x_j\right)^2}_{\text{The disparity should be bilateral-smooth}} + \lambda \underbrace{\sum_i f_i(x_i)}_{\text{The warped right image should match the left image}}$$

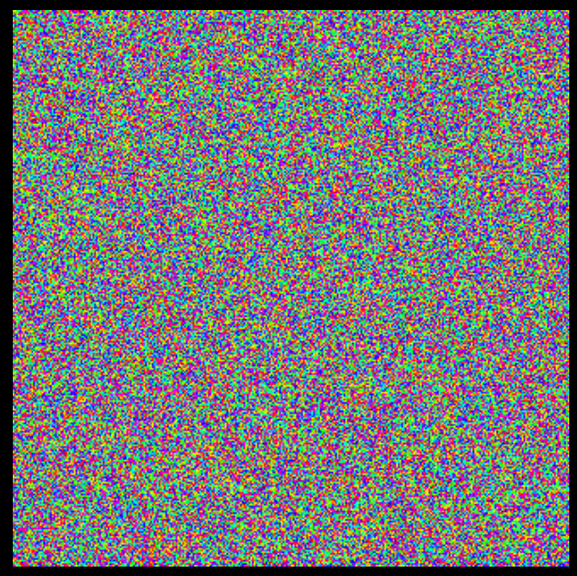


- As many parameters as there are pixels (4-40 million)
- Loss function is expensive to evaluate and minimize

Slow at low resolutions, intractable at high resolutions

Pixel-space optimization

$$\underset{\mathbf{x}}{\text{minimize}} \quad \sum_i \sum_j A_{i,j} \left(x_i - x_j\right)^2 + \lambda \sum_i f_i(x_i)$$



$\mathbf{m} := S\mathbf{1}$

$\mathbf{n} := \vec{1}$

While not converged

$\mathbf{n} := \sqrt{(\mathbf{n} \times \mathbf{m})/(\bar{B}\mathbf{n})}$

$\forall x \quad g_j(x) = \sum_{(w,i) \in S_j} w f_i(x)$

$f_i(p_i) = \max(0, p_i - u_i) + \max(0, l_i - p_i)$

$g_j(v_j) = \sum_{y=0}^{v_j} \sum_{x=0}^{y} u''_j(x) + \sum_{y=v_j}^{D} \sum_{x=y}^{\infty} l''_j(x)$

$\mathbf{p} = S^{\mathrm{T}}\mathbf{v}$

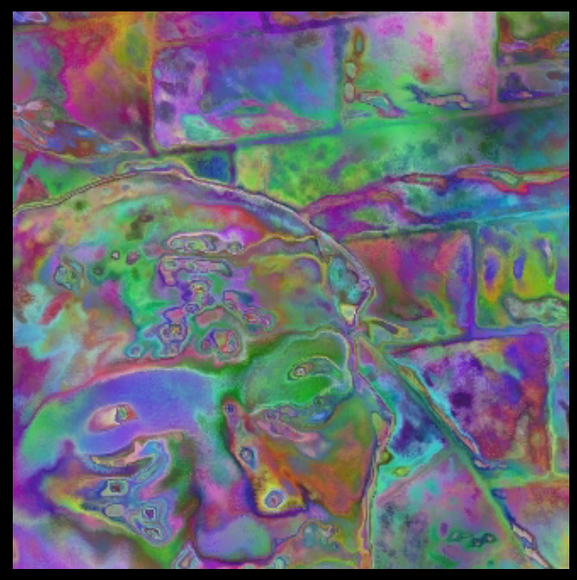
$(C'_n \bar{B} C'_n)\mathbf{1} = (C_n \bar{B} C_n)\mathbf{1} = \mathbf{m}$

$C'_n \approx C_n = \mathrm{diag}(\mathbf{n})$

$C'_s \approx C_s = \mathrm{diag}(\mathbf{m})$

Bilateral-space optimization

$$\underset{\mathbf{z}}{\text{minimize}} \quad \mathbf{z}^{\mathrm{T}}(D - CBC)\mathbf{z} + \lambda \sum_j g_j(z_i)$$

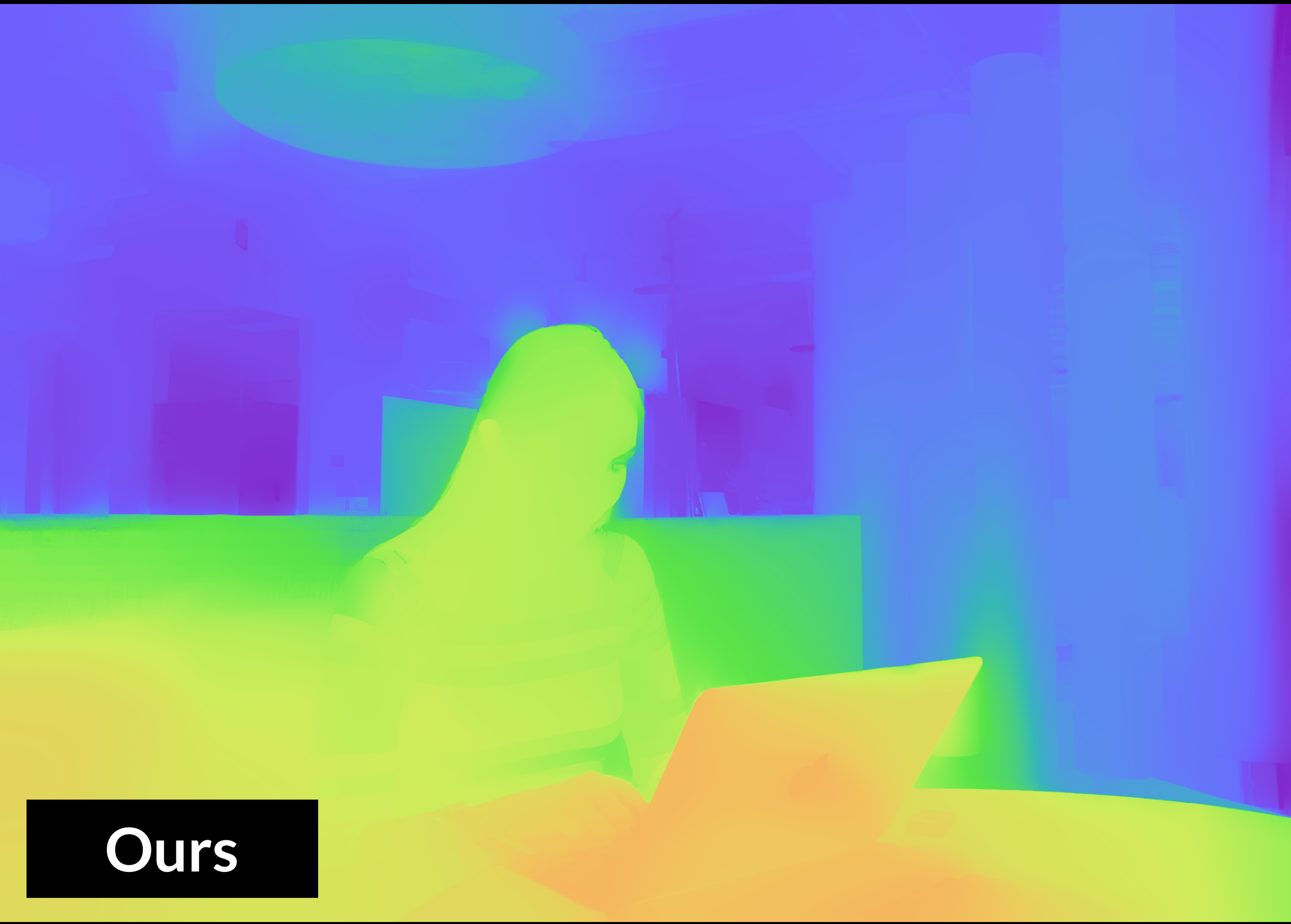


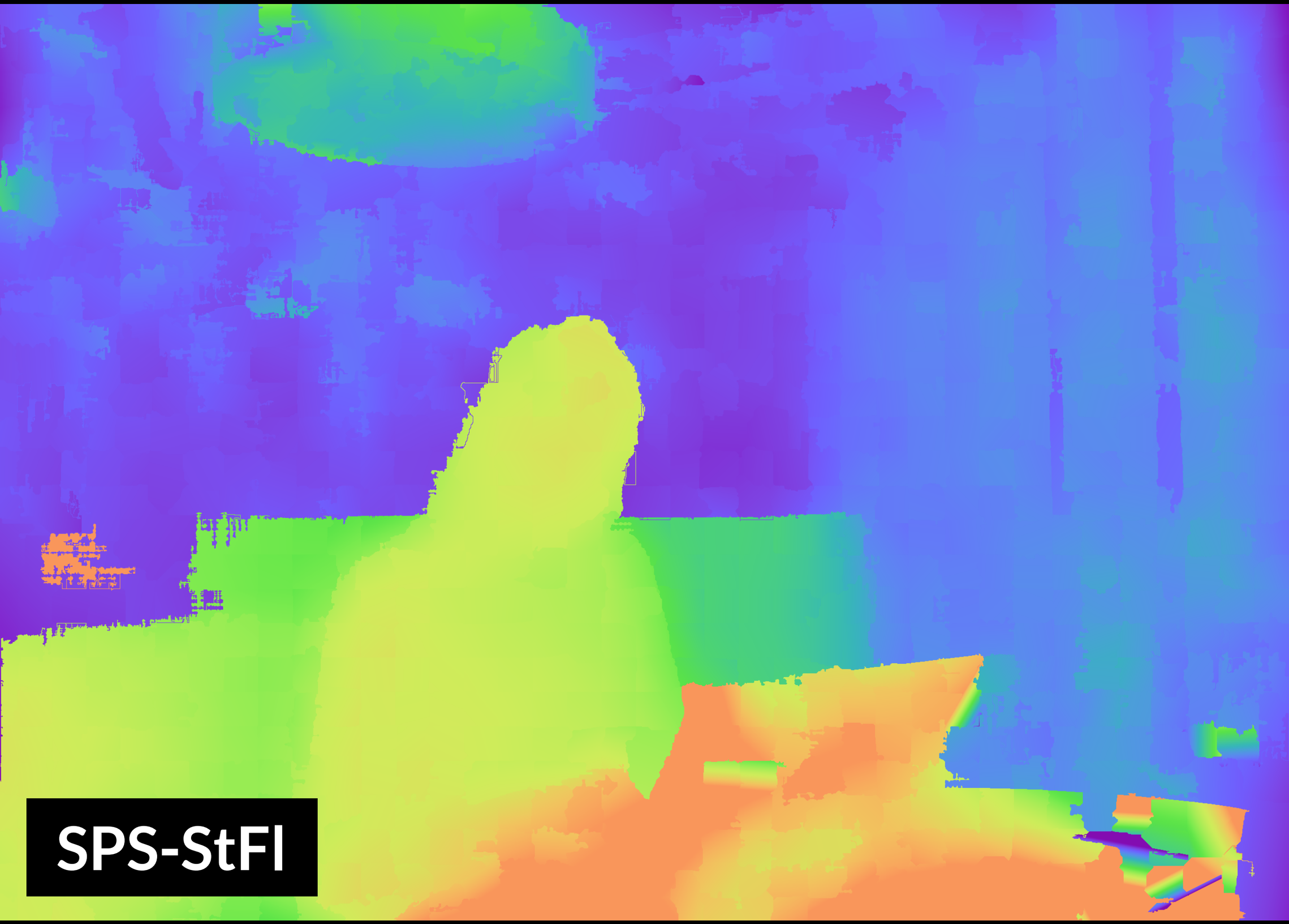
+ Far fewer parameters (50-100x)
+ Loss function is very cheap to evaluate and minimize

Results


Input 1


Input 2


Ours


SPS-StFI

Ours

Ours

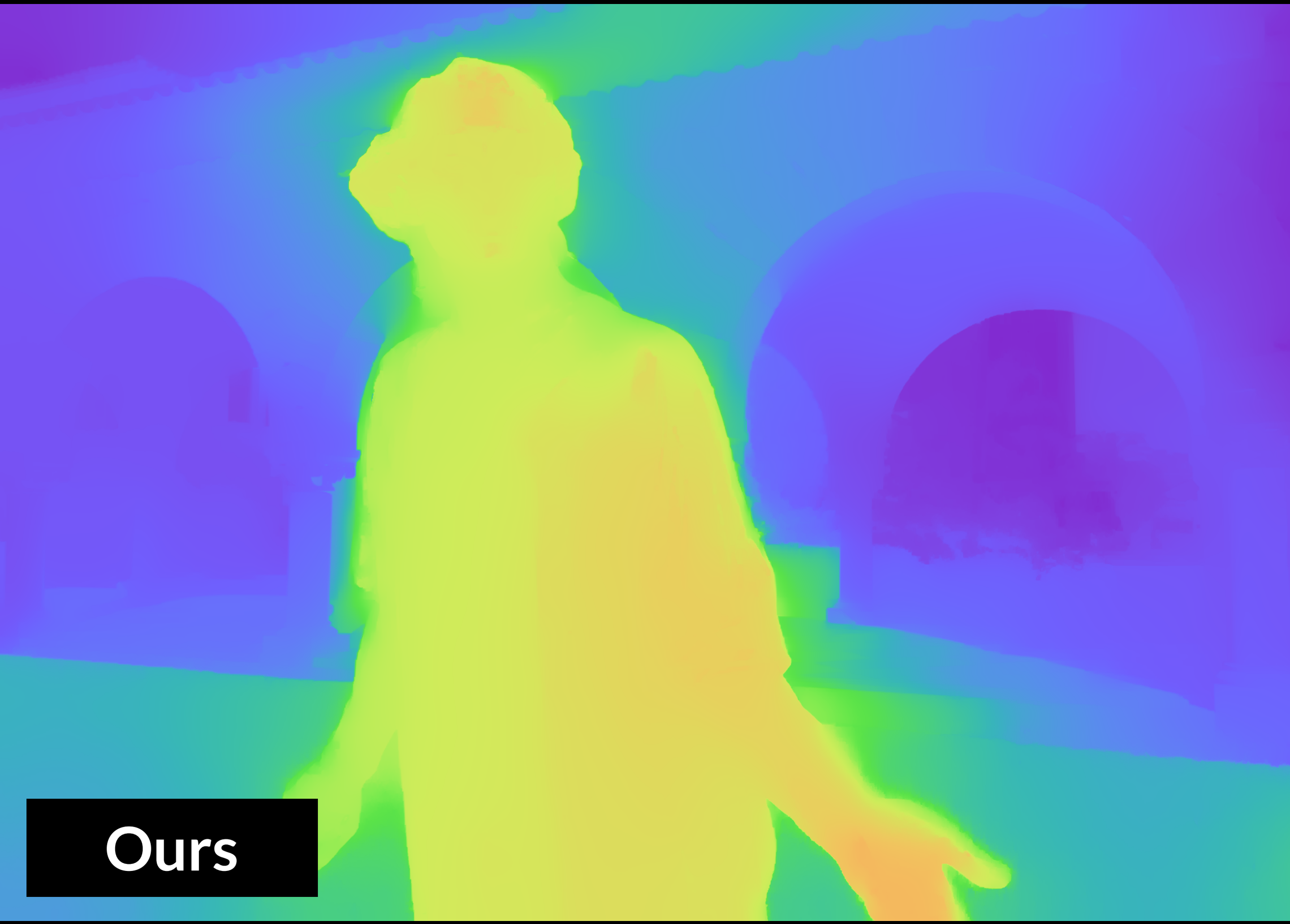
Input

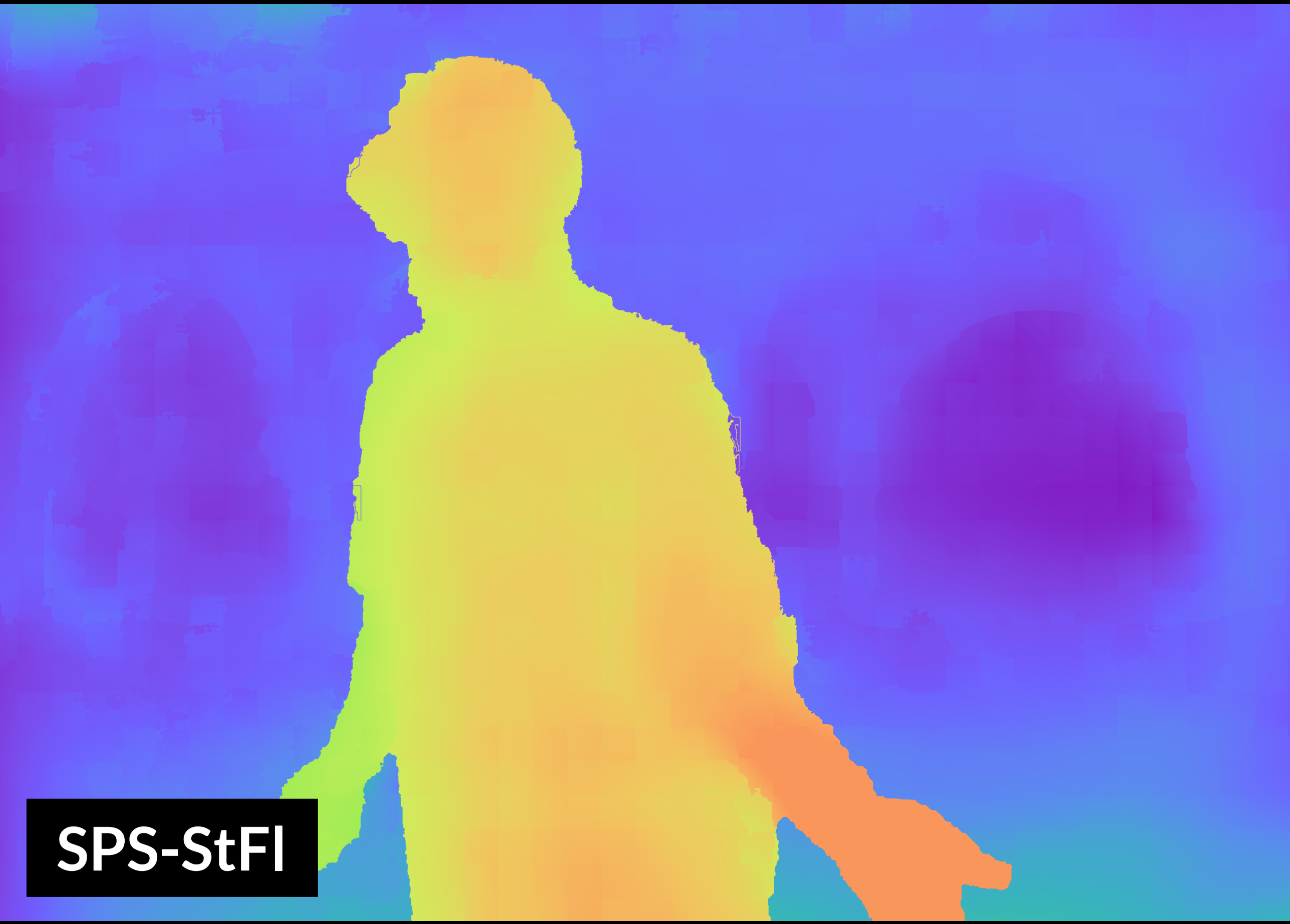

SPS-StFI

Ours


Input 1


Input 2


Ours


SPS-StFI

Ours

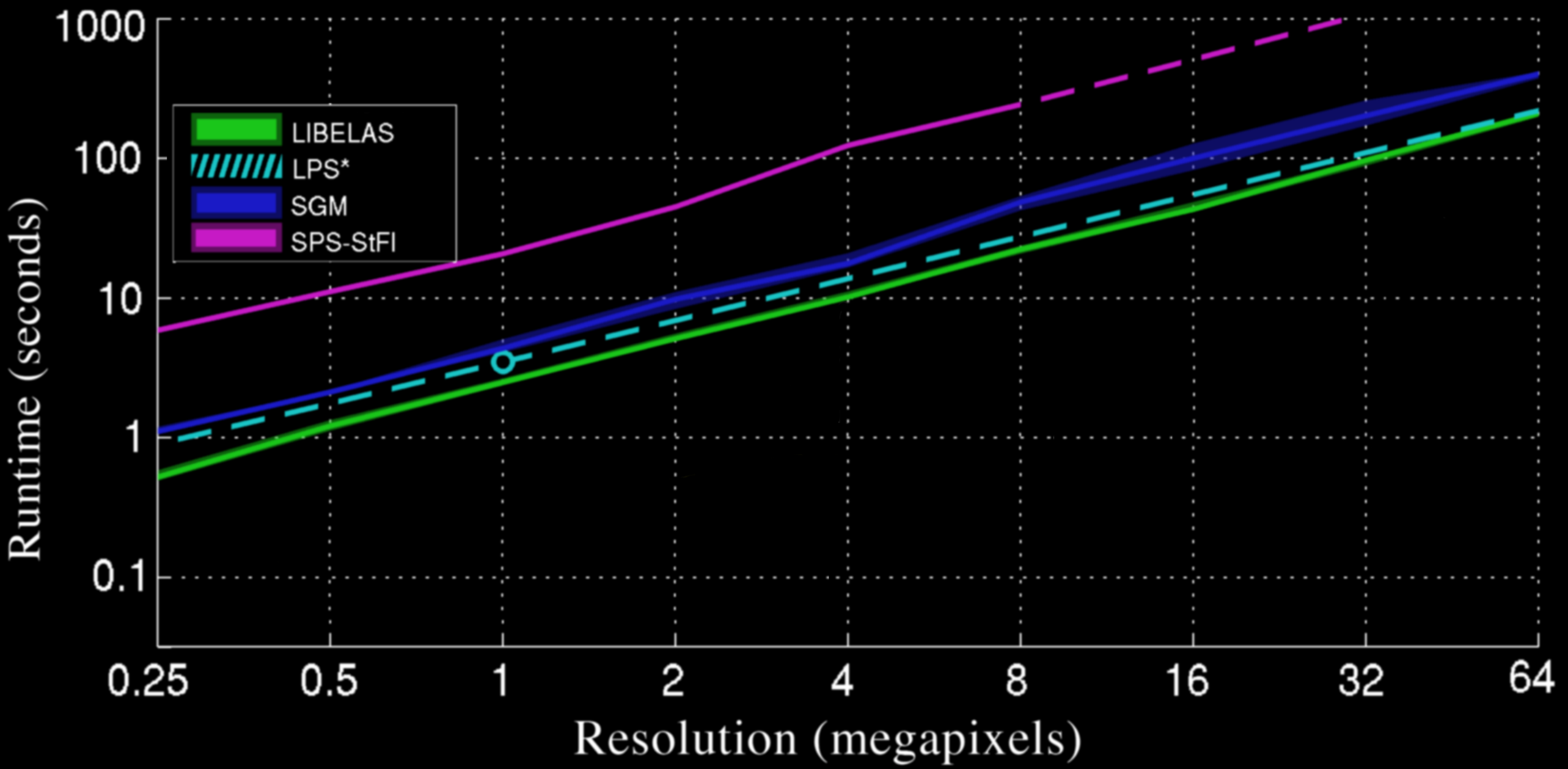
Ours


Input

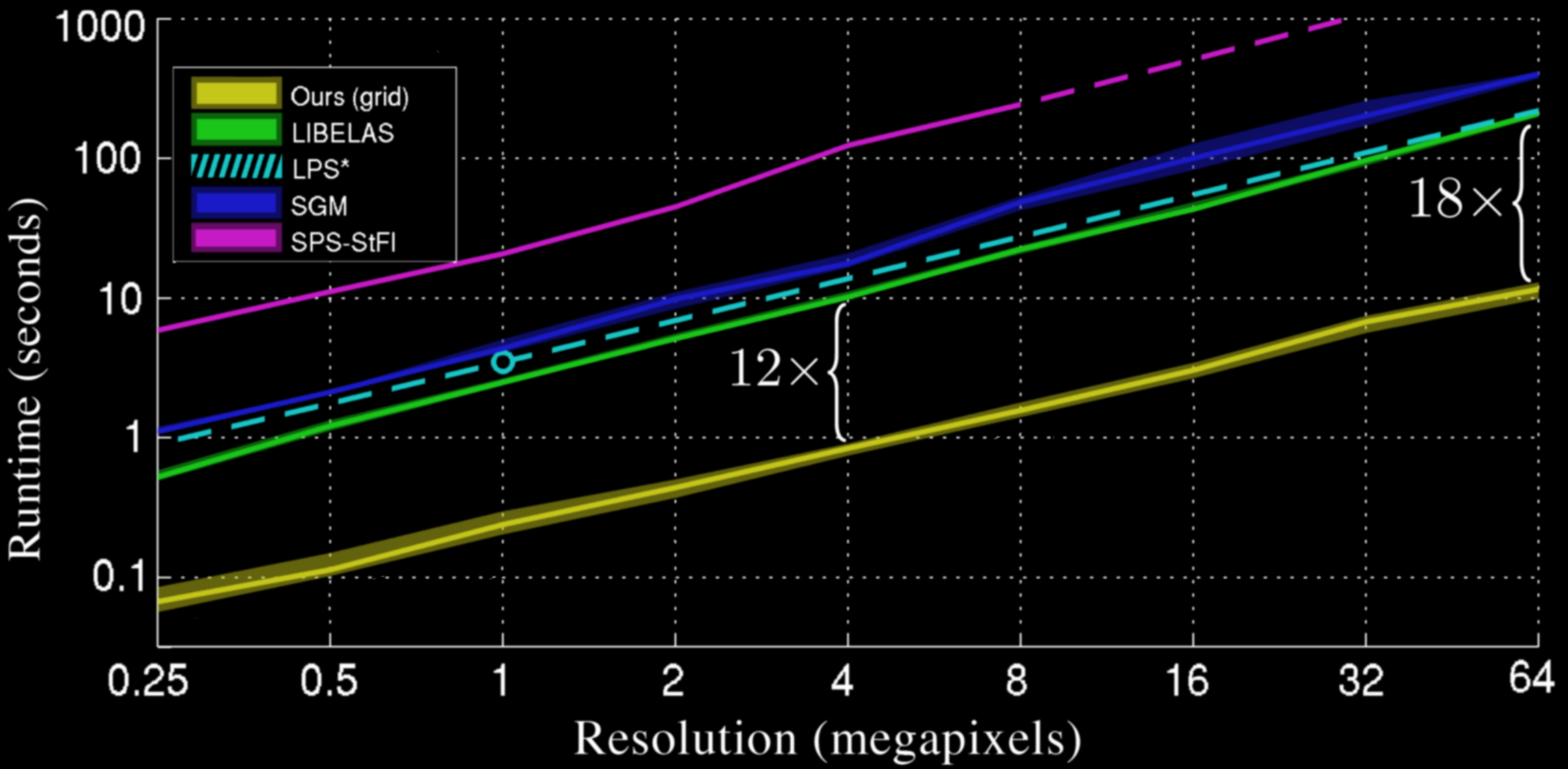

SPS-StFI

Ours

Speed


1000
100
10
1
0.1
Runtime (seconds)
LIBELAS
LPS*
SGM
SPS-StFl
0.25
0.5
1
2
4
8
16
32
64
Resolution (megapixels)

Speed


1000
100
10
1
0.1
Runtime (seconds)
Ours (grid)
LIBELAS
LPS*
SGM
SPS-StFl
18×
12×
0.25
0.5
1
2
4
8
16
32
64
Resolution (megapixels)

Lens Blur
in the Google Camera App

Use it!

Depths are embedded in the Lens Blur metadata

Video from http://depthy.me/

Conclusions

A new way to solve optimization problems based on bilateral filtering
- Fast
- Scalable
- Applicable to many vision / graphics problems

A fast and edge-aware stereo algorithm
- Great for graphics / mobile applications
- Already on millions of phones

Thanks!

barron@google.com