Spatio-temporal Sampling for Reconstructing Distribution Effects

Philipp Slusallek   *Karol Myszkowski*

Gurprit Singh

Multi-dimensional adaptive sampling of distribution effects

Fourier Analysis of Light Transport

Temporal reconstruction of distribution effects

Multi-dimensional adaptive spatio-temporal sampling

Hachisuka et al. [2008]

Multi-dimensional adaptive sampling of distribution effects

Fourier Analysis of Light Transport

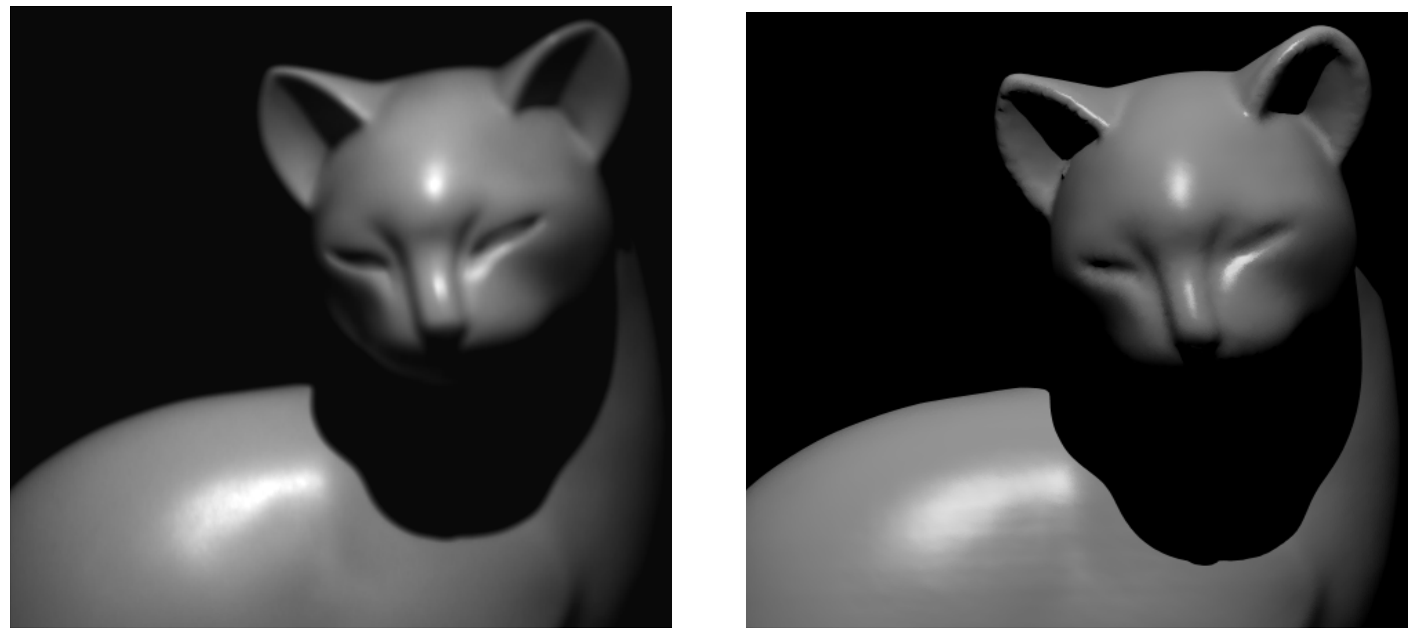
Light Transport

Understanding, manipulating and computing signals

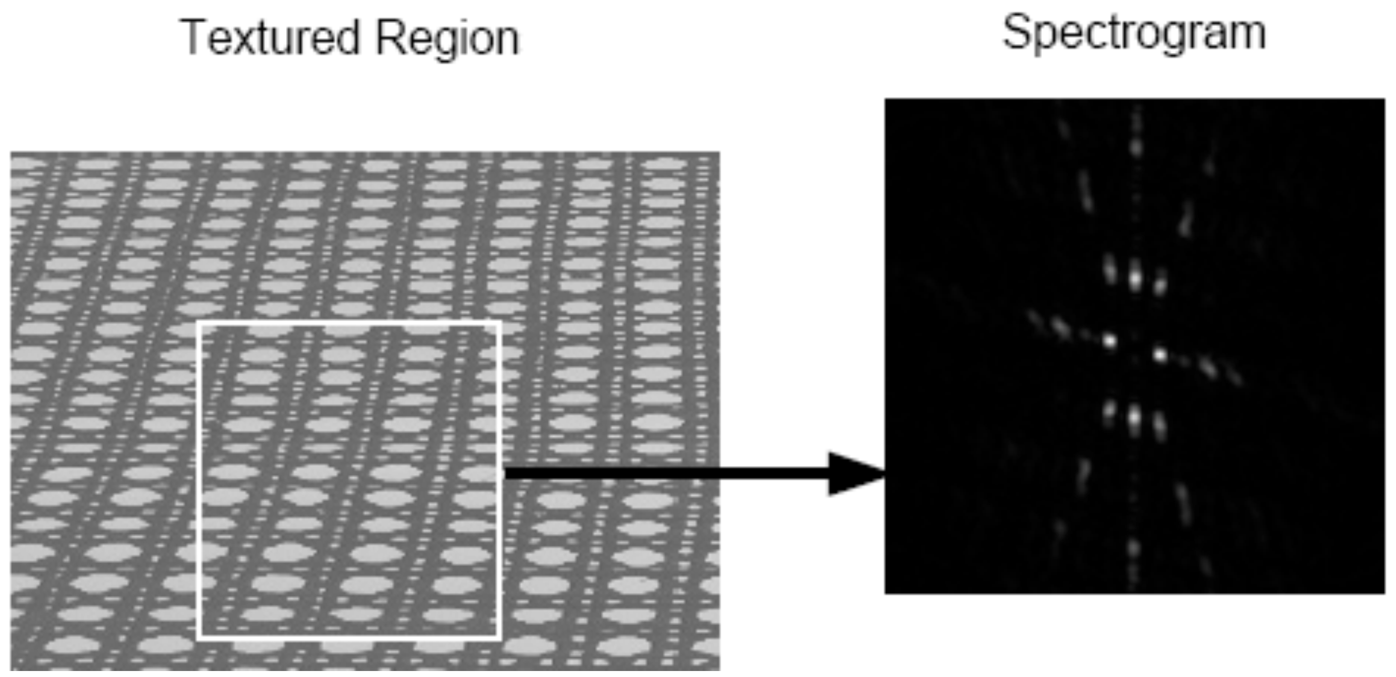
- Discontinuites
  - where things change
- Gradients
  - Useful for interpoloation
- **Frequency content (today's main course)**
  - Useful for sampling
  - Useful for inverse problems
    - Sometimes useful as basis functions
  - Statistics

And all these capture perceptual properties

Frequency contents matter in vision



Inverse lighting

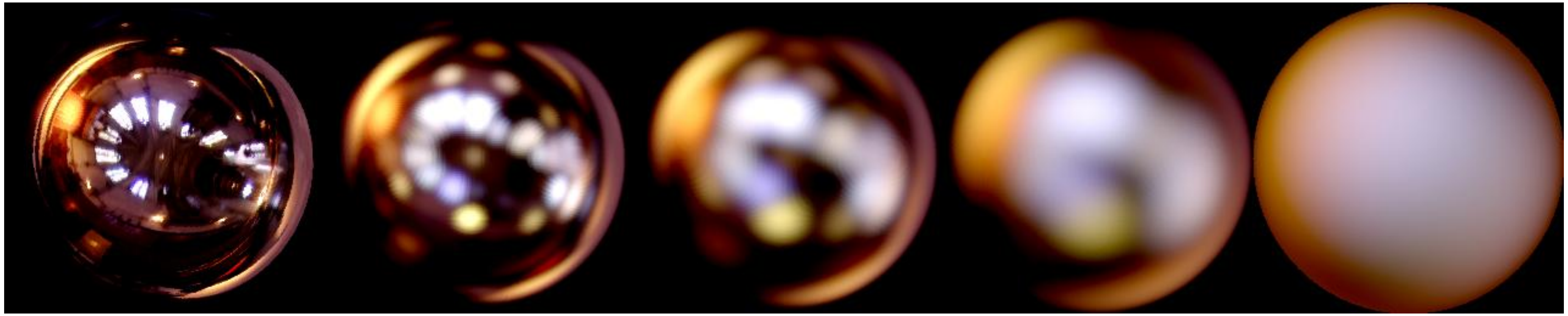


Shape from texture

Shape from (de)focus

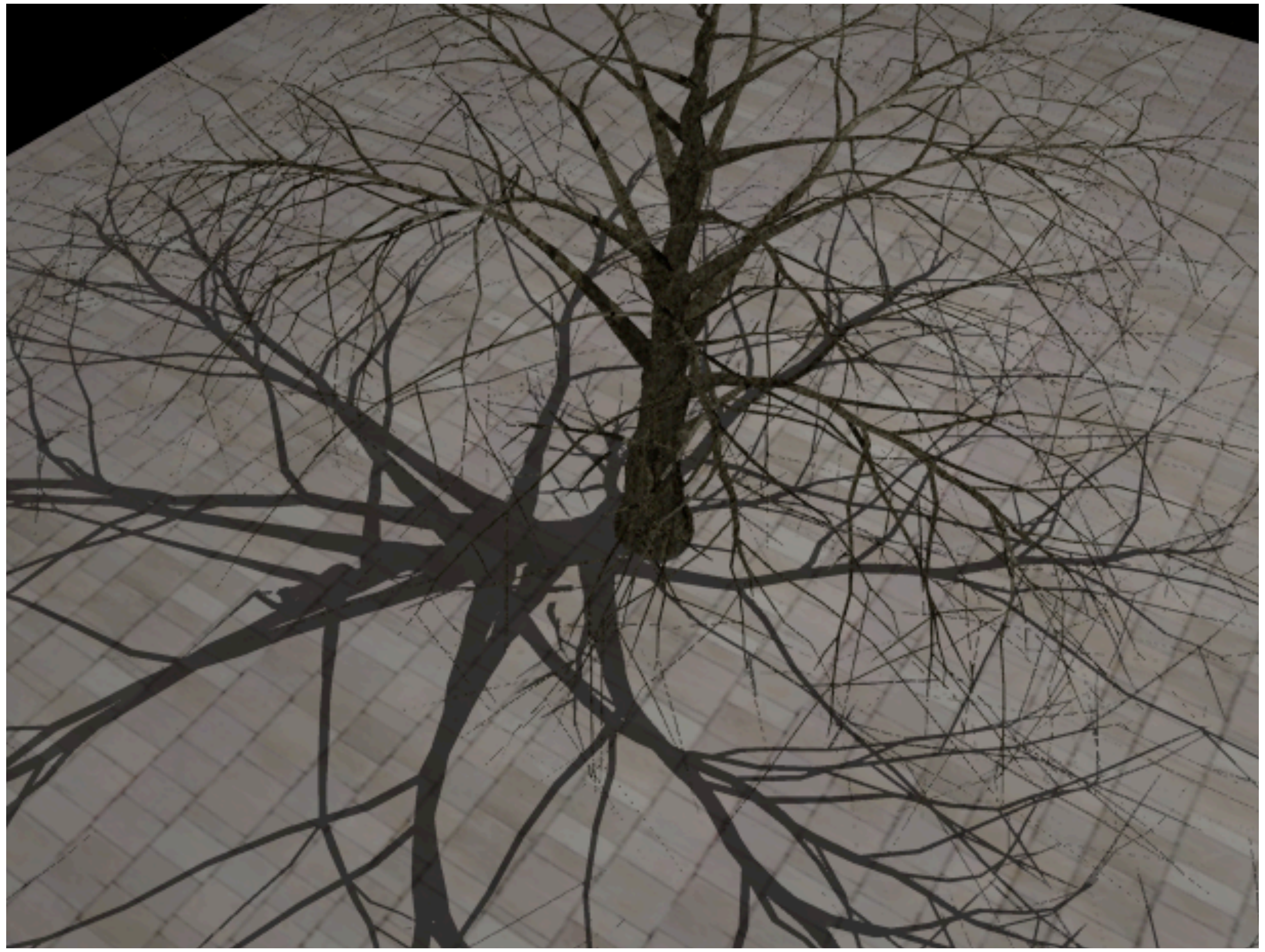
Illumination effects

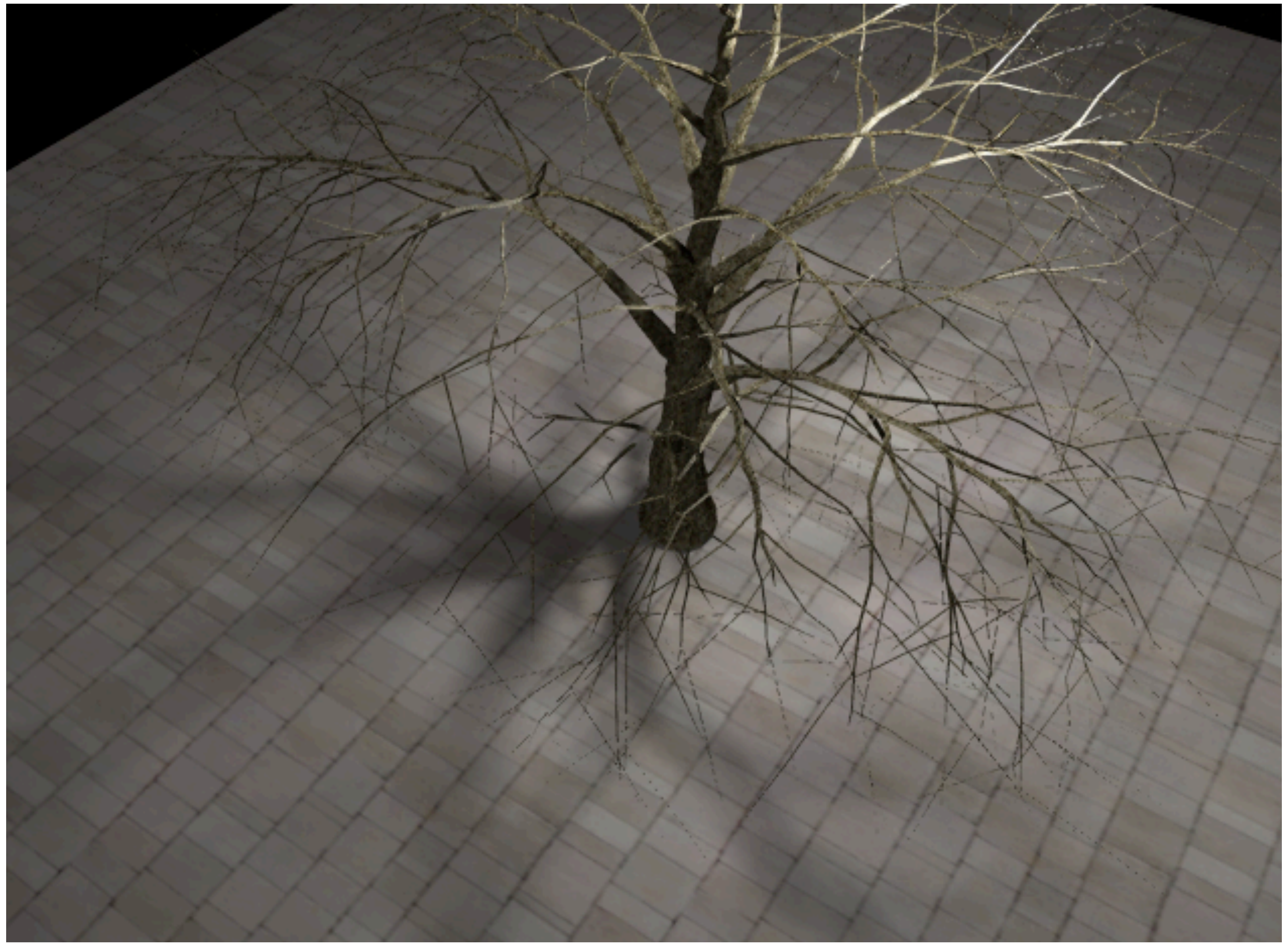
Blurry reflections



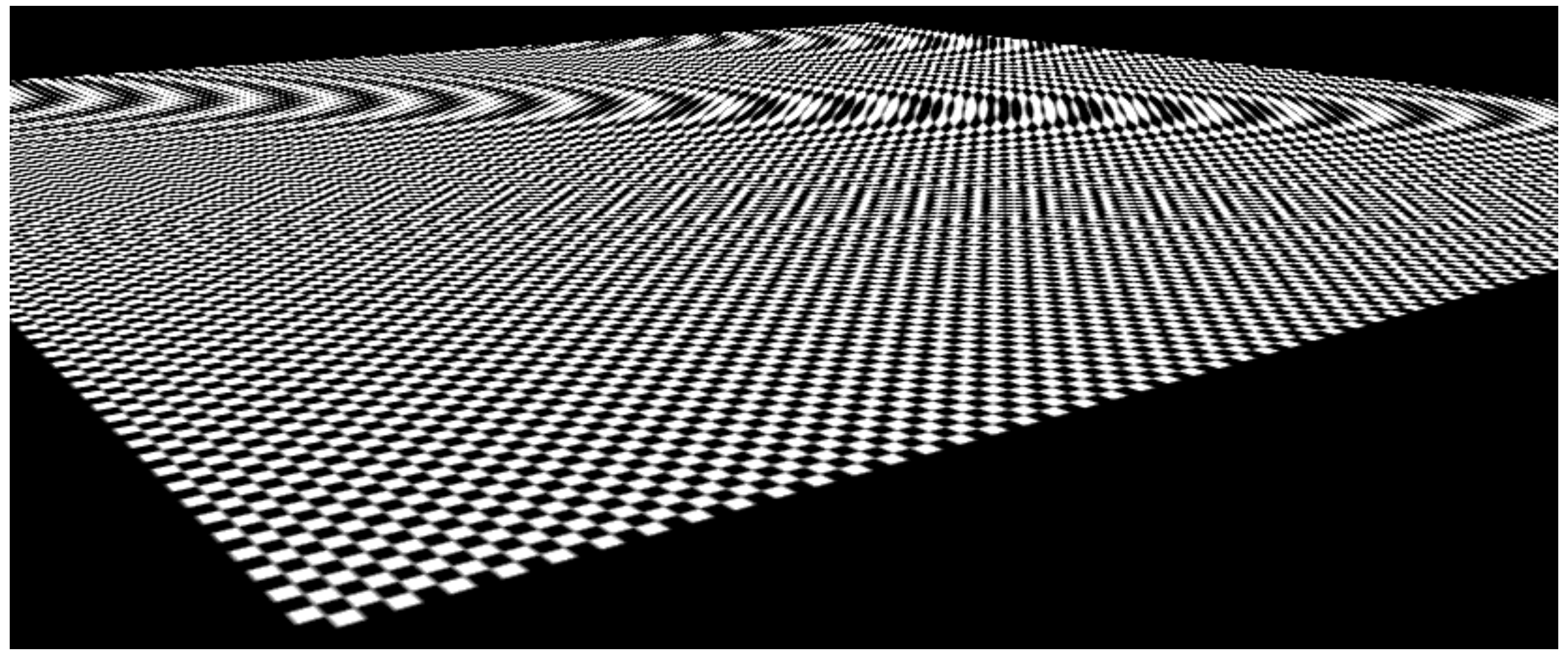
Illumination effects

Shadow boundaries:





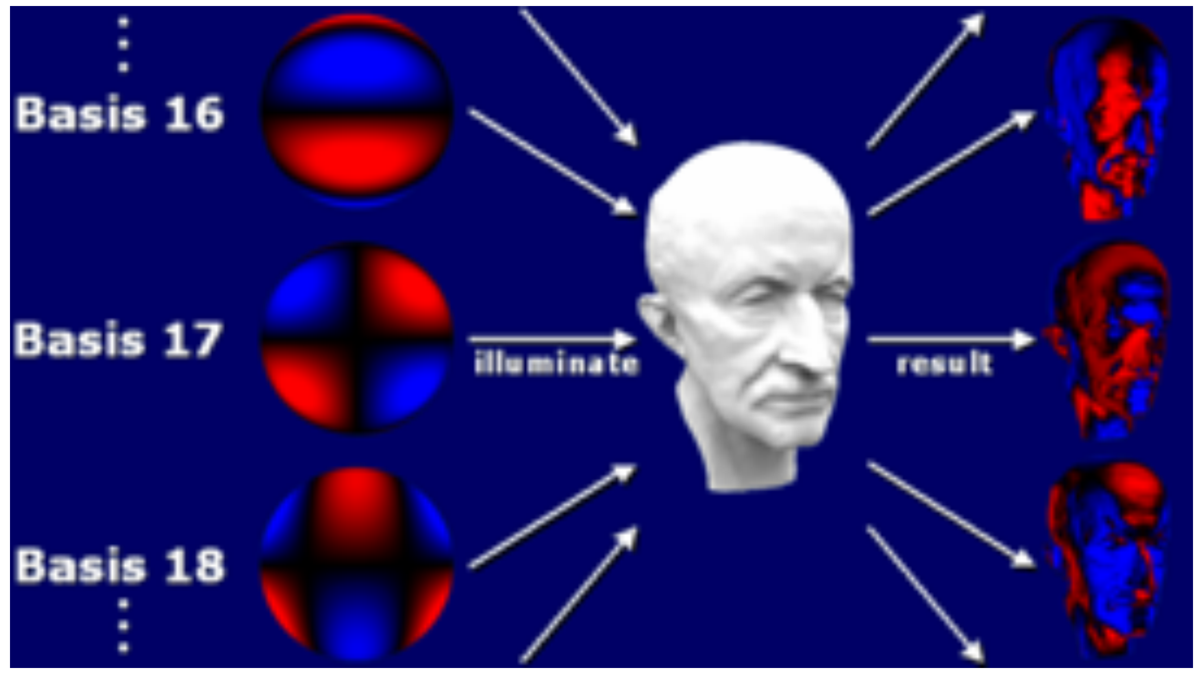
Frequency contents matter in graphics



Sampling, antialiasing

Texture filtering

Light Field Sampling

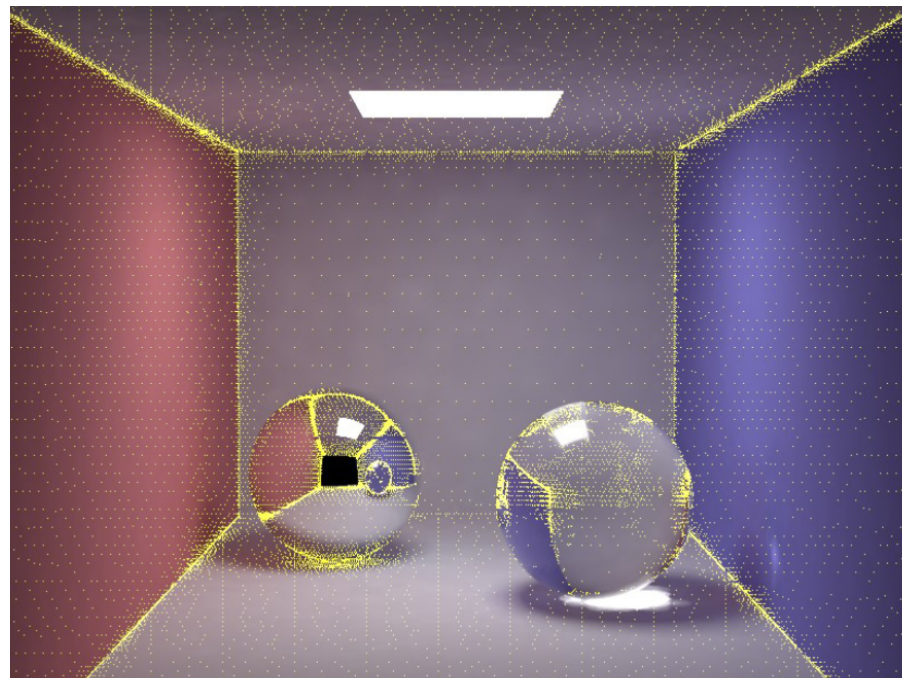


Fourier-like basis

Precomputed radiance transfer

Wavelet radiosity

Spherical harmonics



Low frequency assumption

Irradiance caching

How does light interactions in a scene explain the frequency content?

How does light interactions in a scene explain the frequency content?

Theoretical framework:

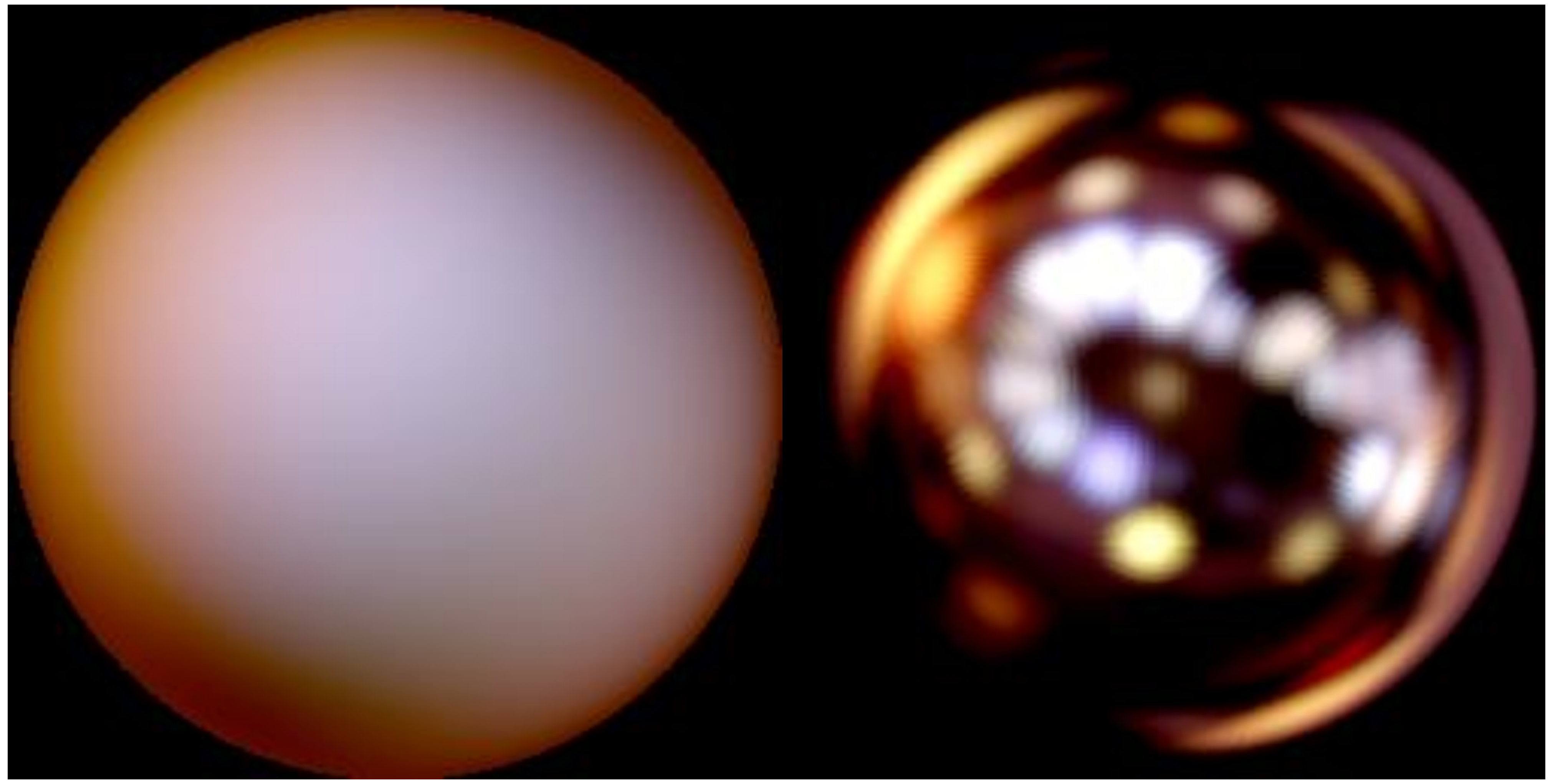
Understanding the frequency content of the radiance function

Mathematical equations of the light transport → Fourier spectrum of the Illumination in the scene

Spatial and Angular frequency

Spatial frequency

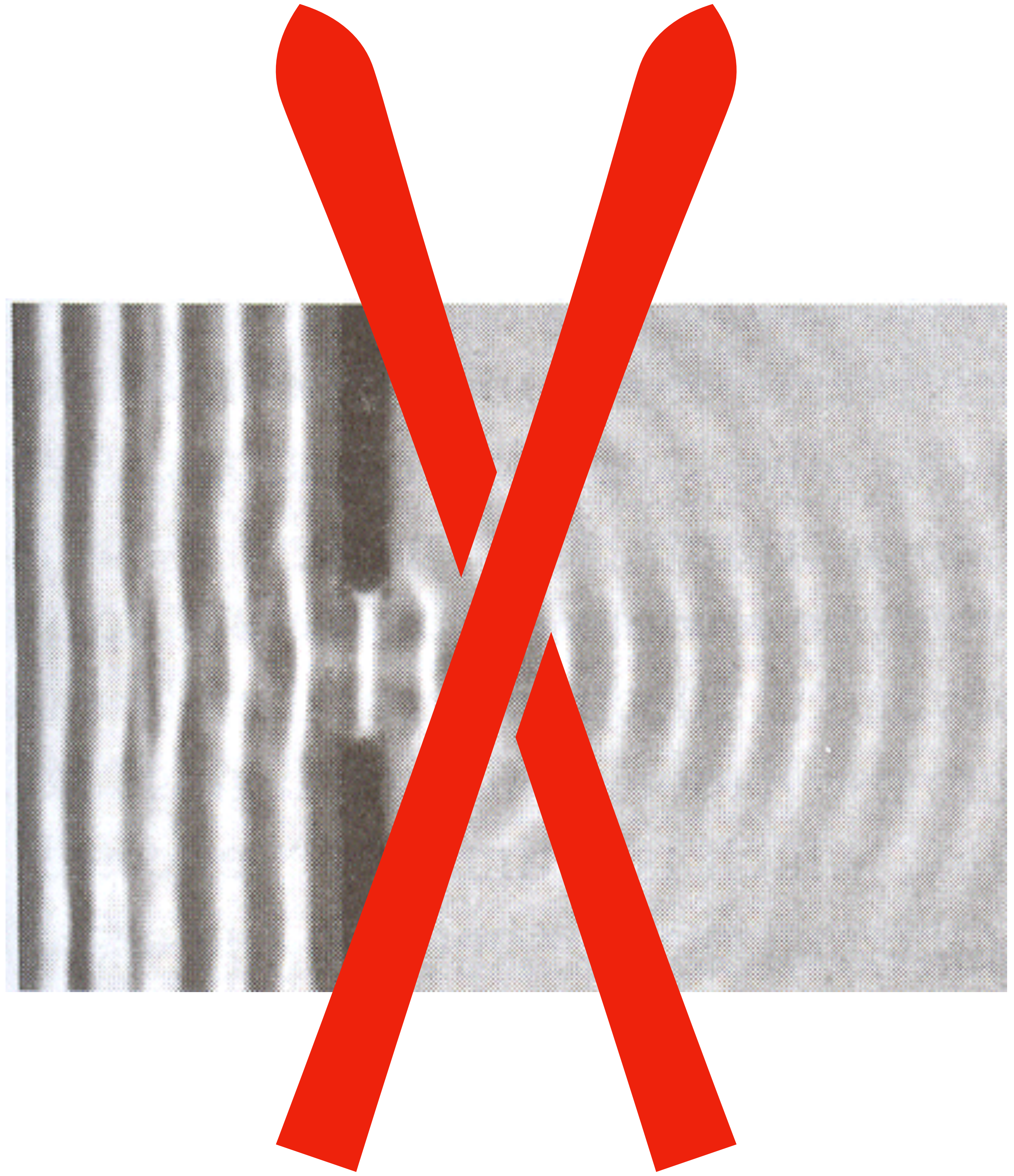
(e.g., shadows, textures)

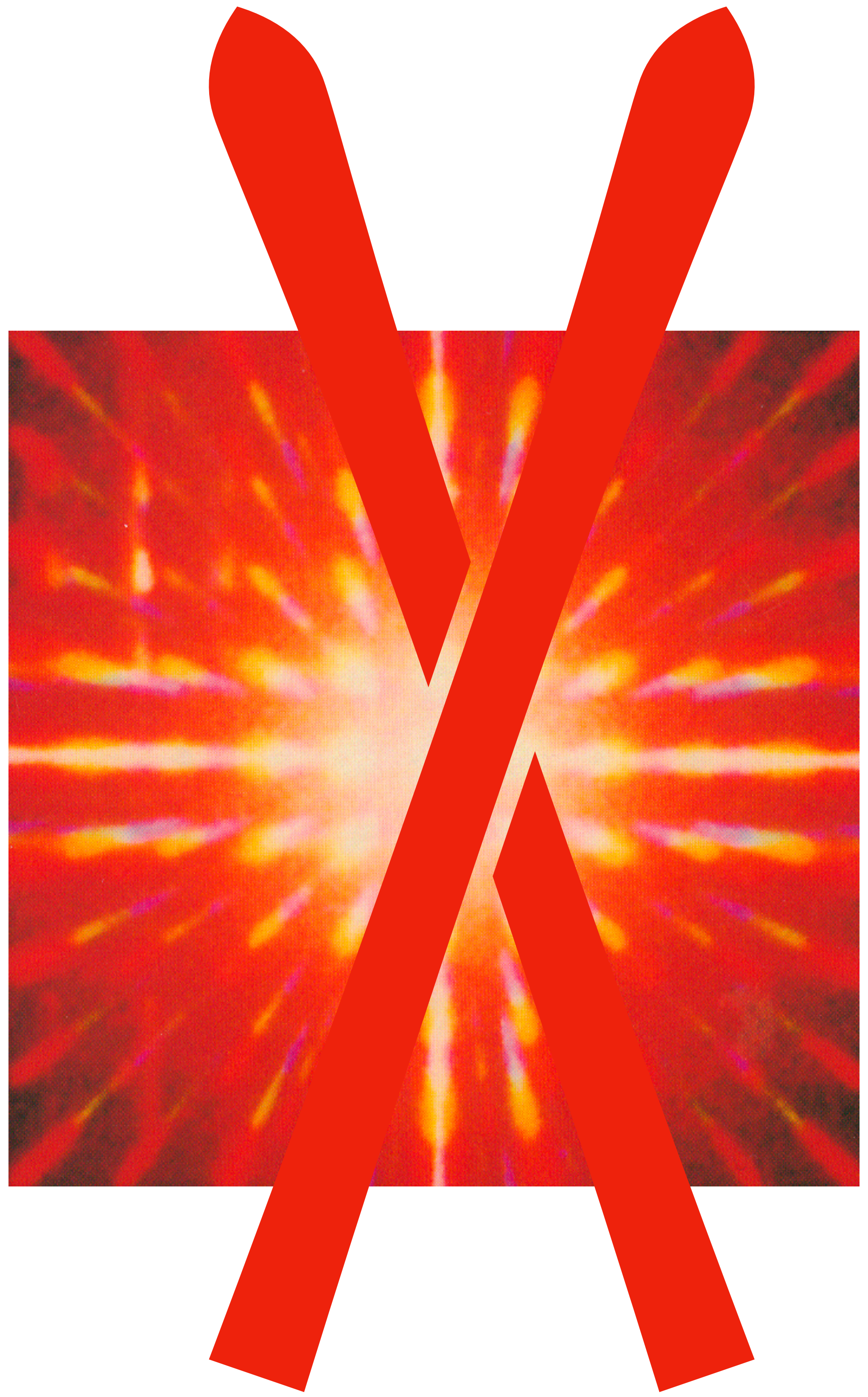


Angular frequency

(e.g., blurry highlights)

Disclaimer: no Fourier optics



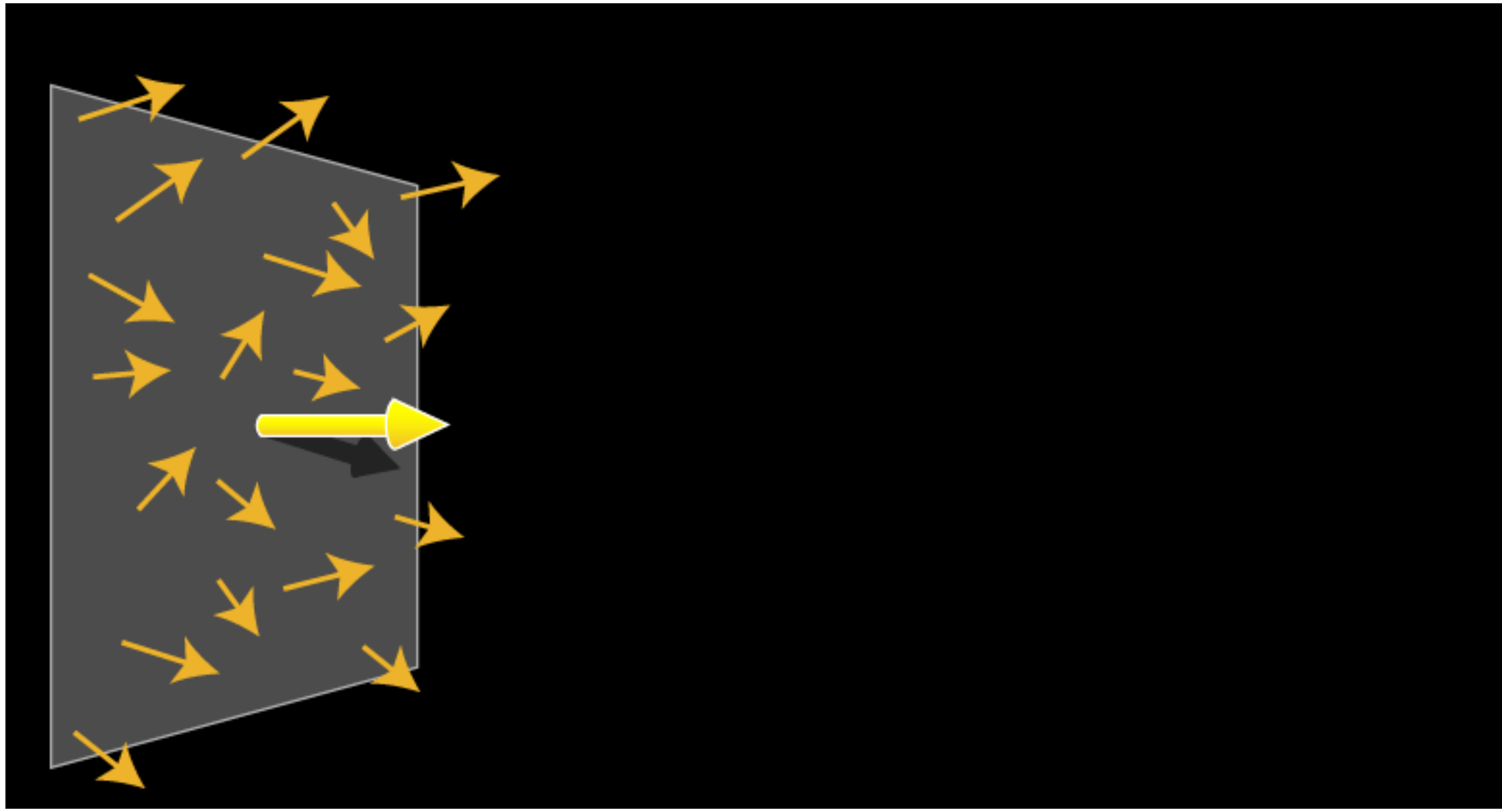


Only geometrical optics

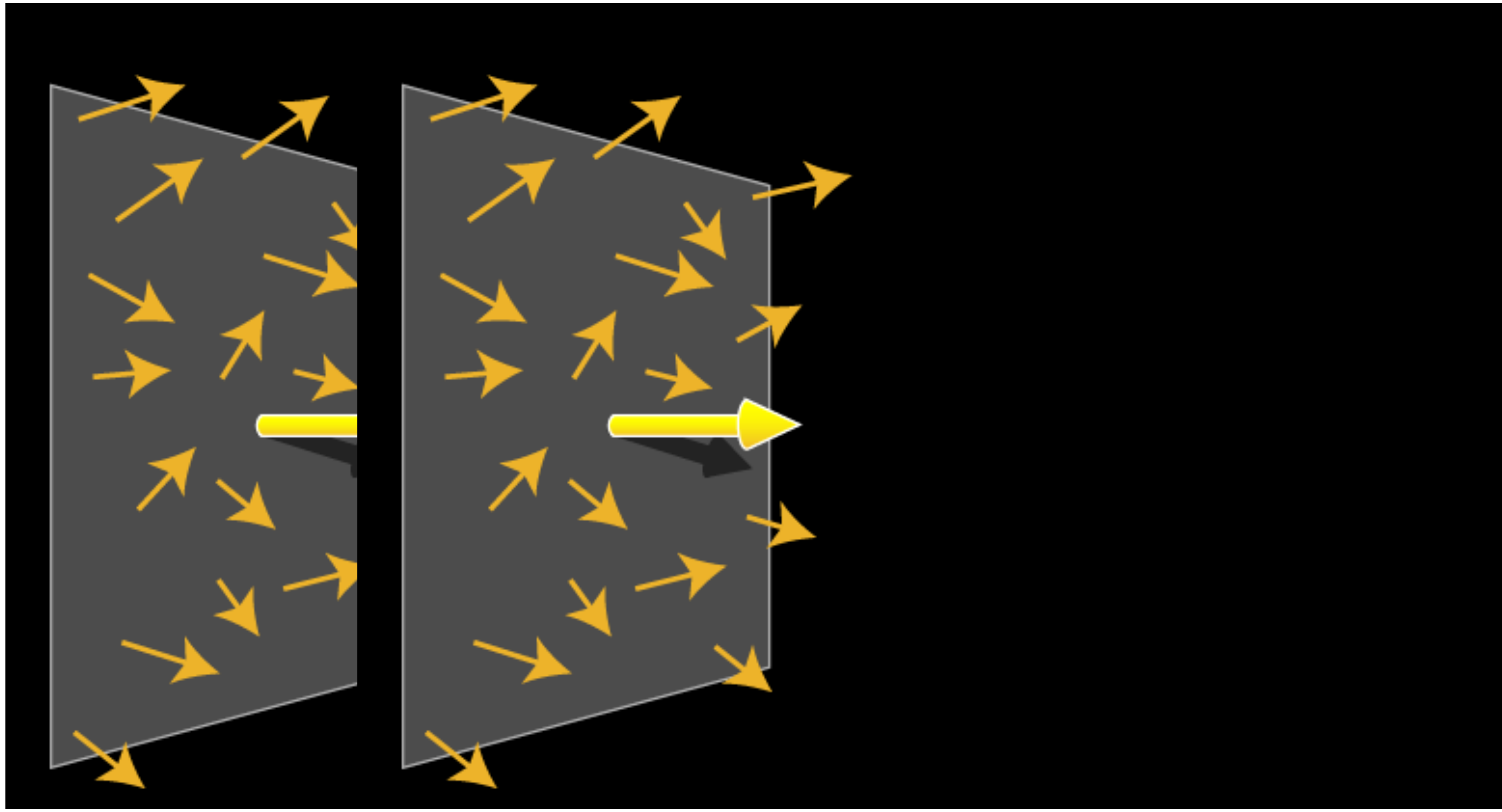
Light transport in a scene

Light transport in a scene

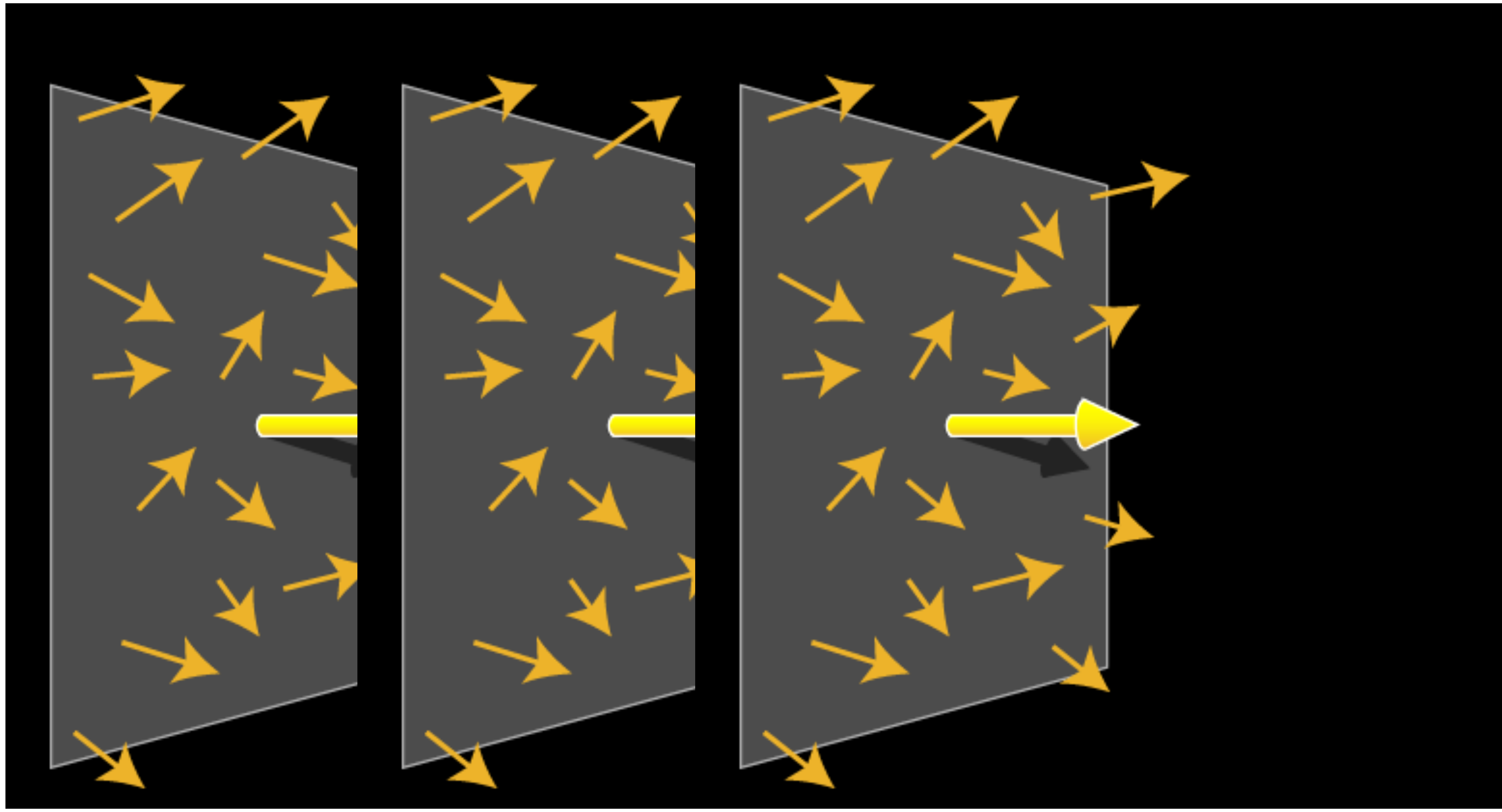
Flatland: Light transport in a scene



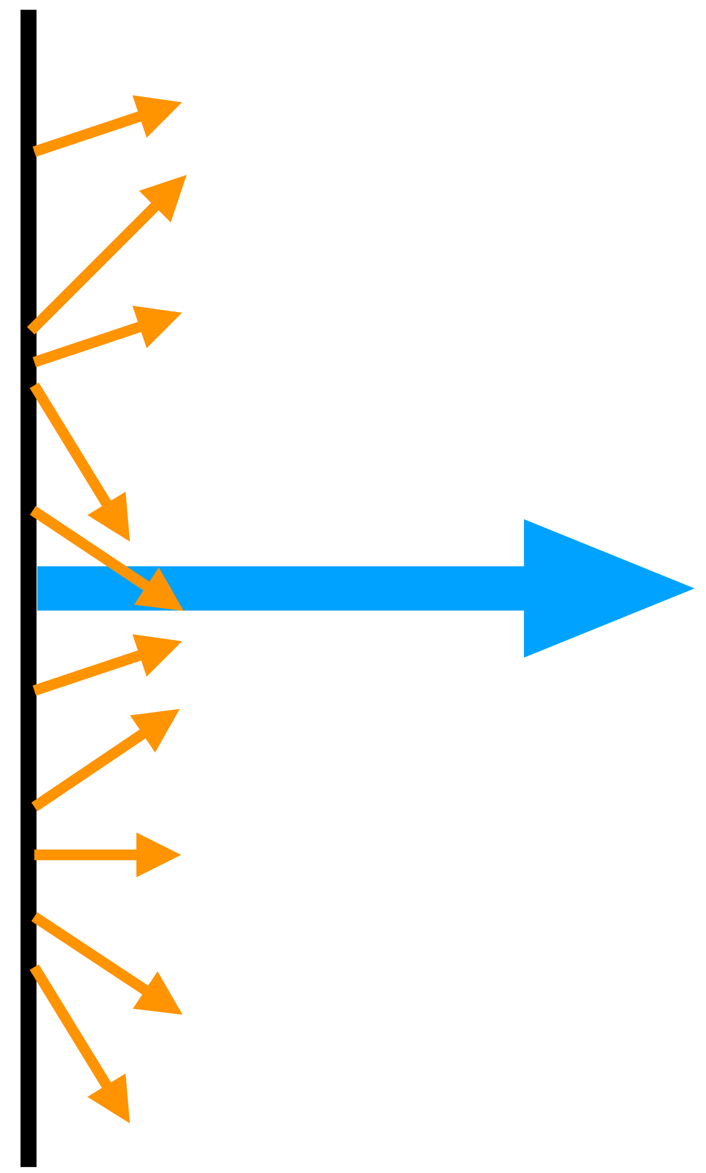
Flatland: Light transport in a scene



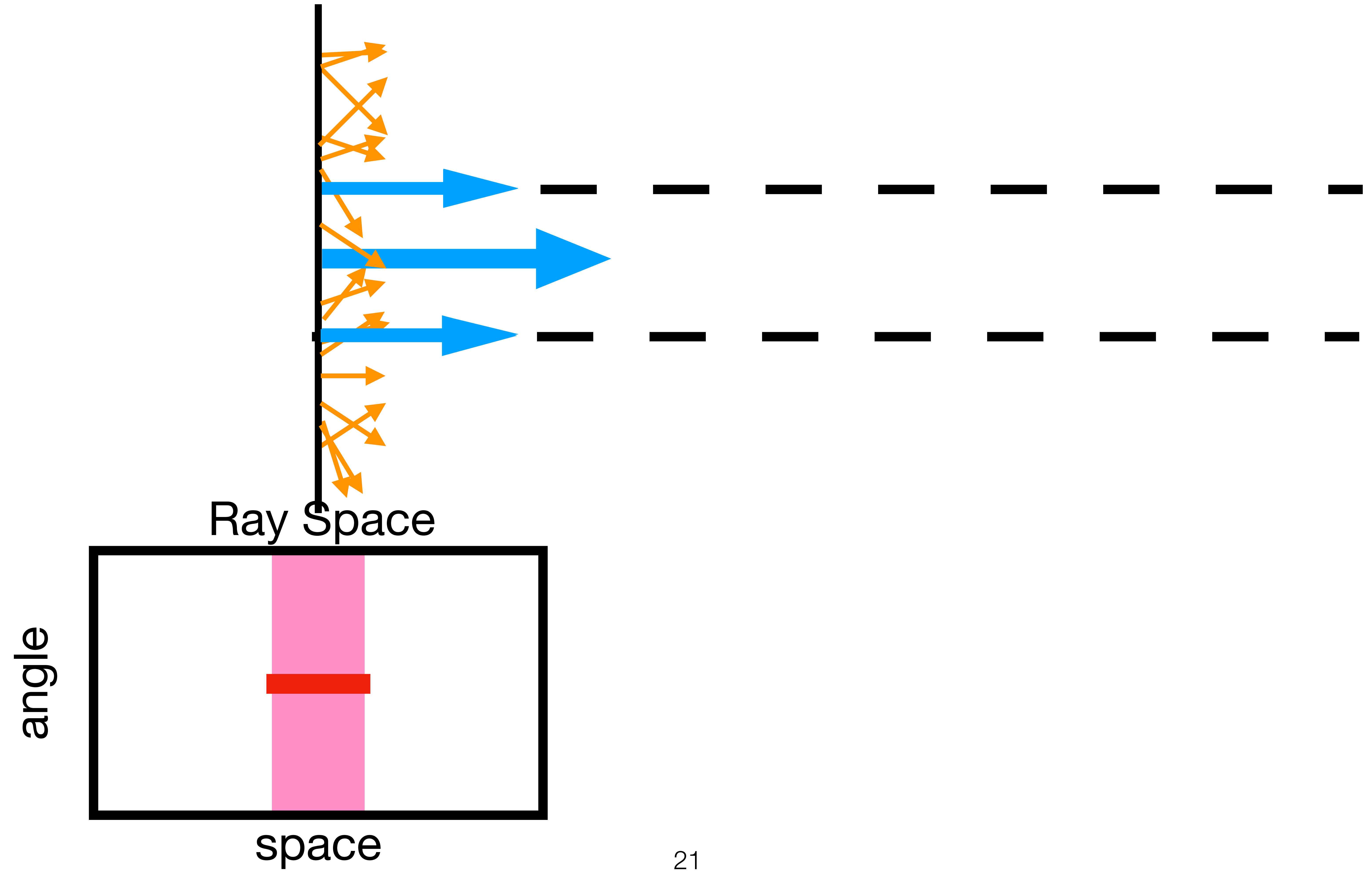
Flatland: Light transport in a scene



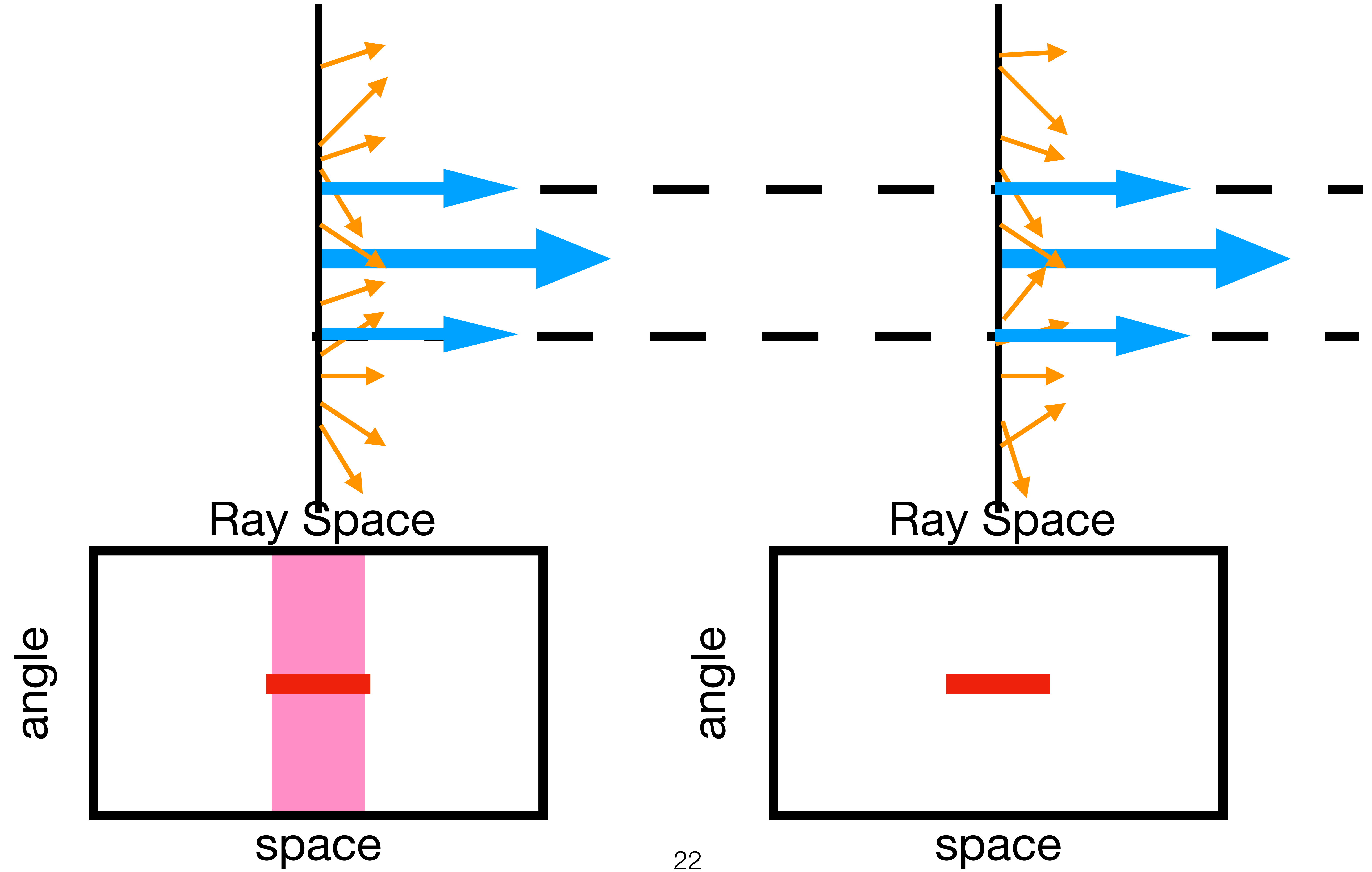
Flatland: Light transport



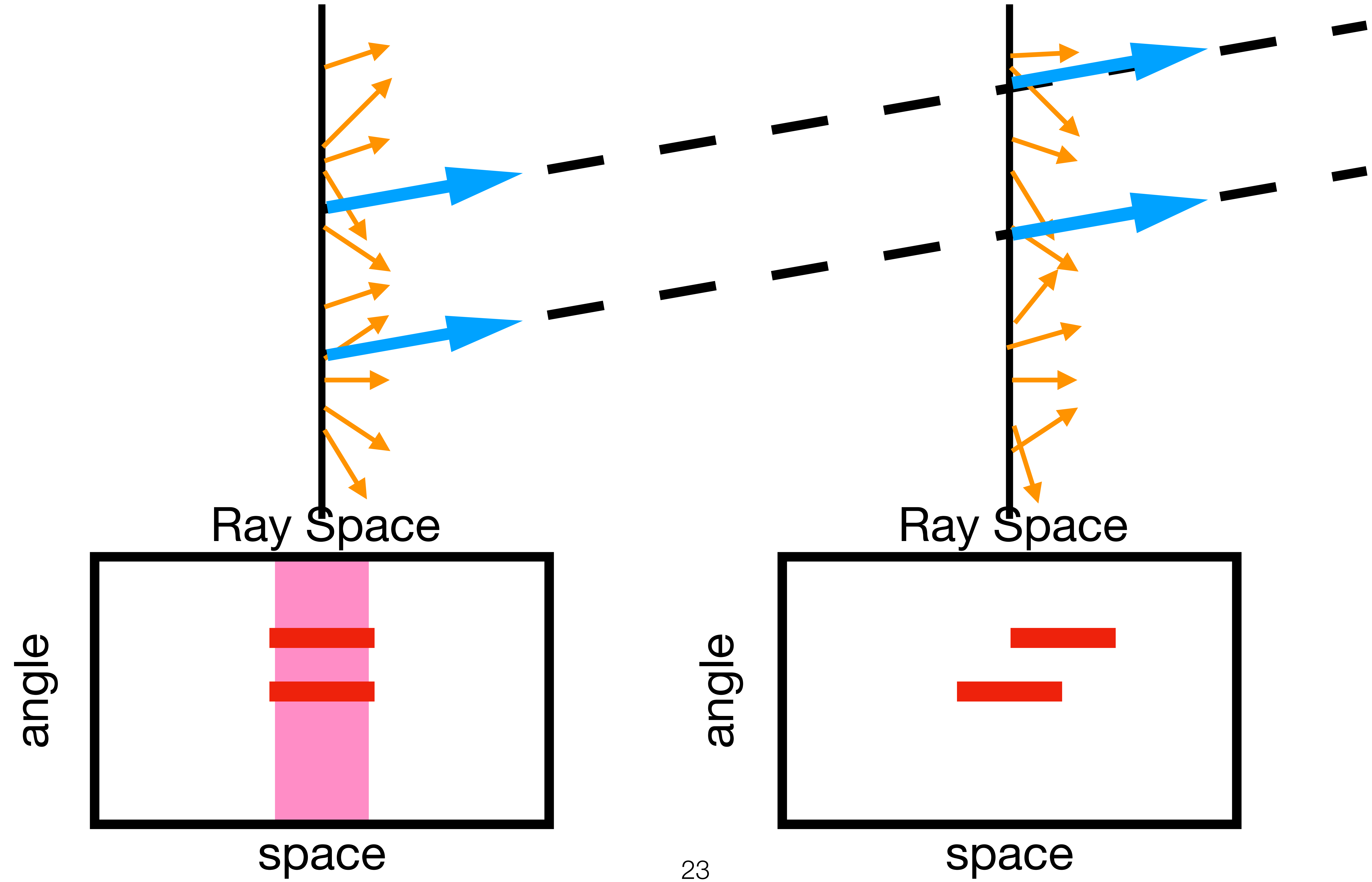
Flatland: Light transport



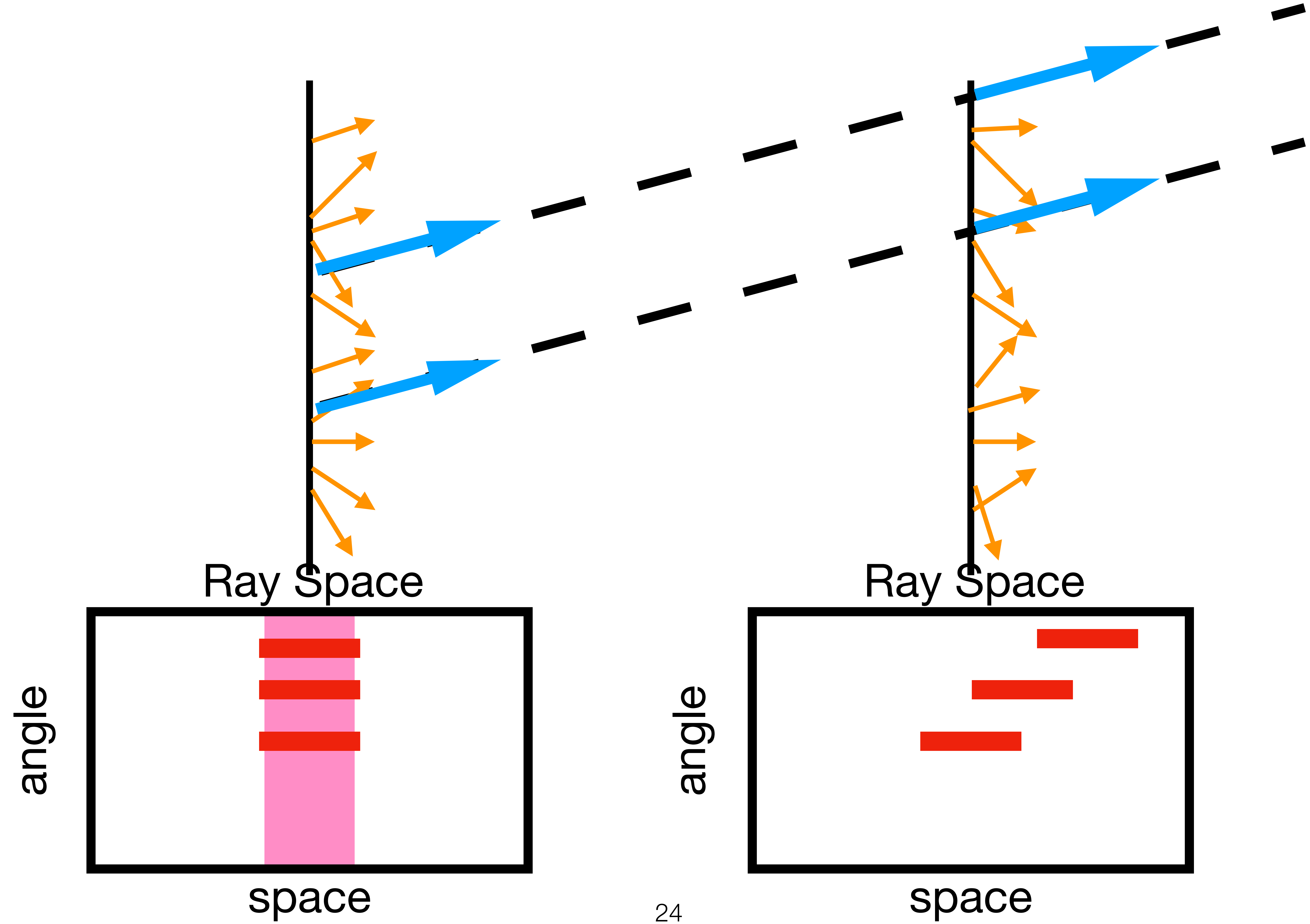
Flatland: Light transport



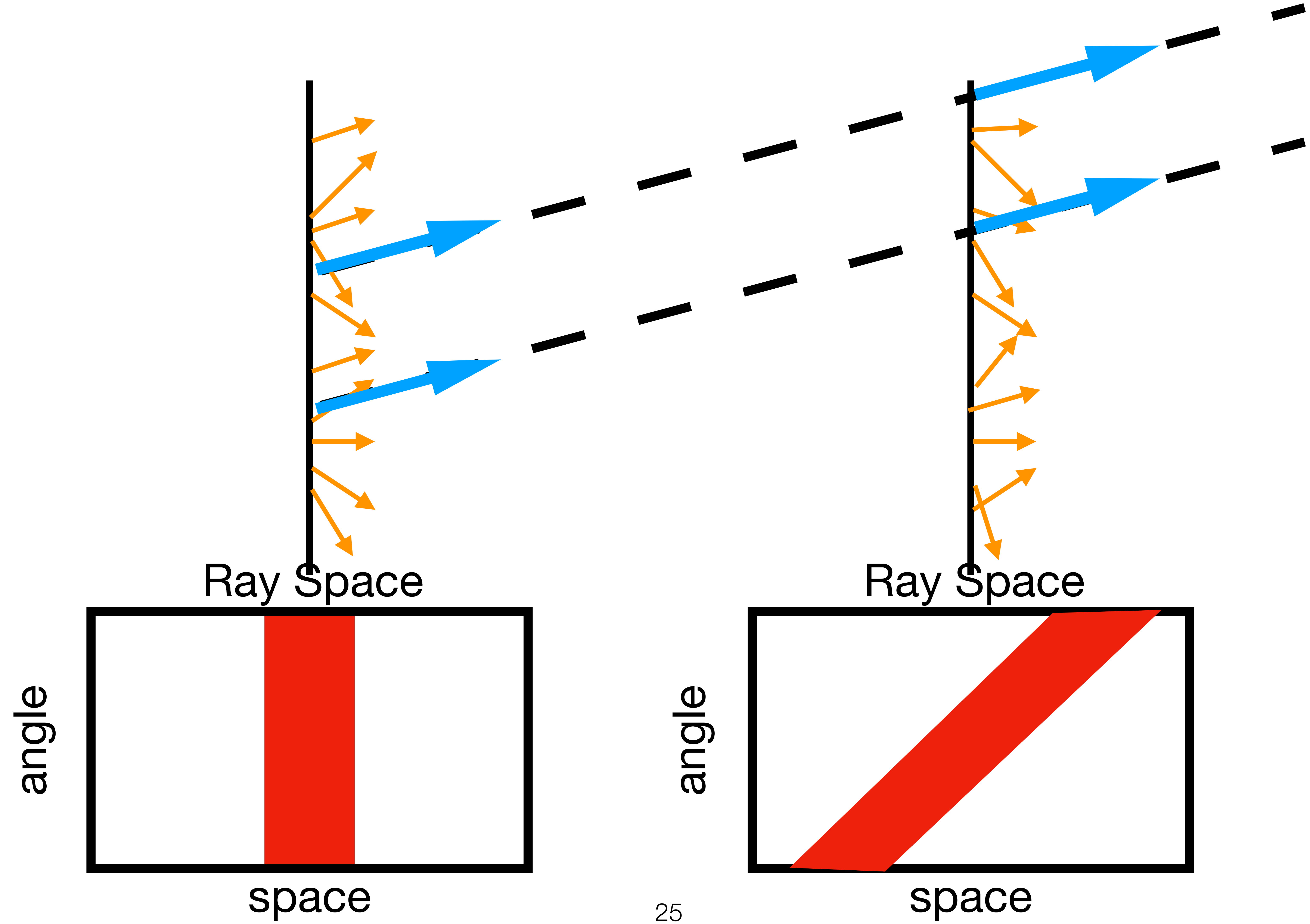
Flatland: Light transport



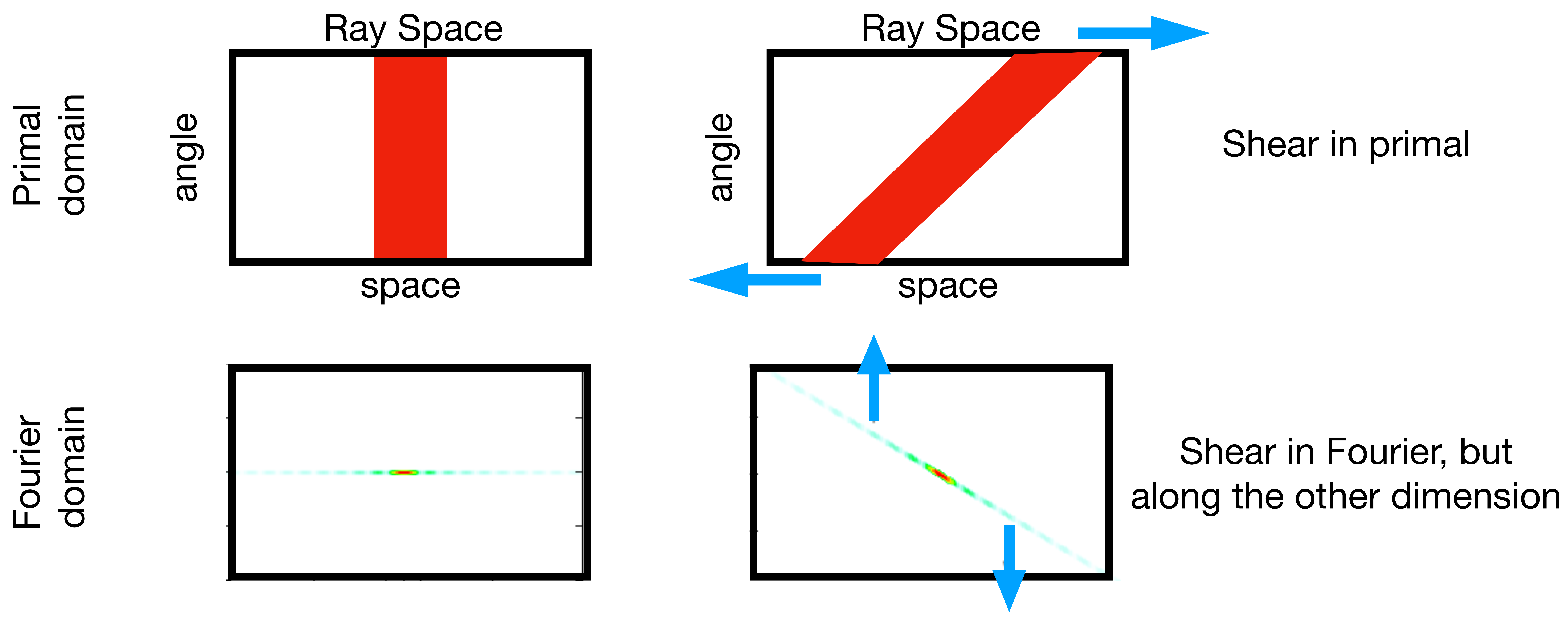
Flatland: Light transport



Flatland: Light transport

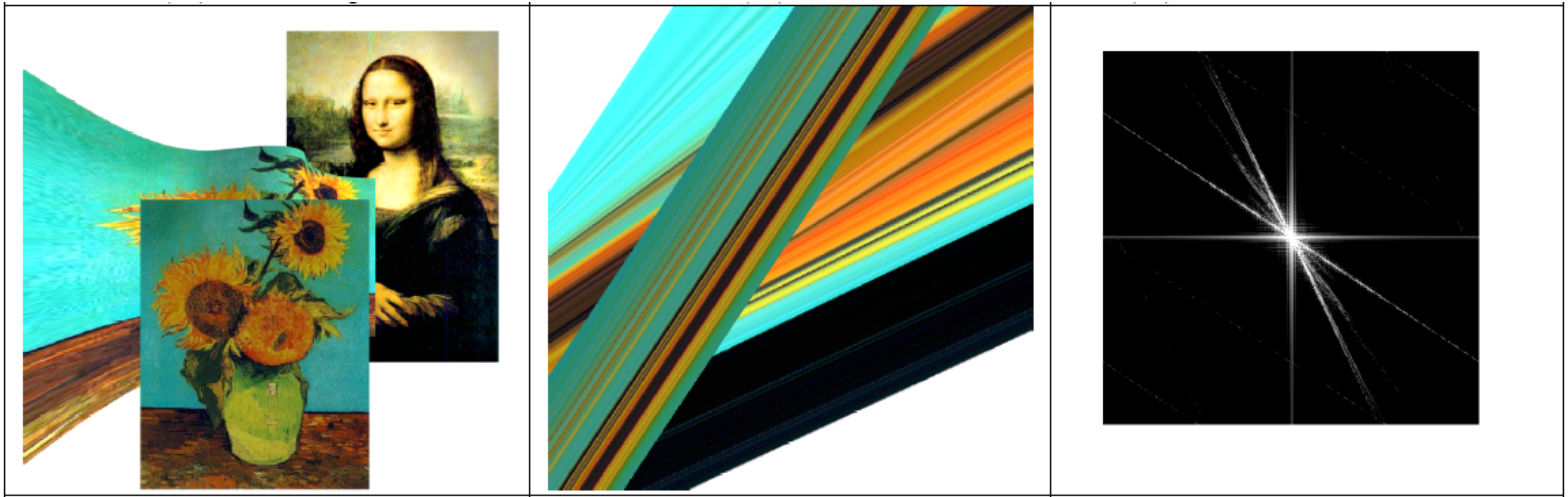


Flatland: Light transport



Transport --> Shear

Consistent with literature [see Plenoptic Sampling by Chai et al. 2000]



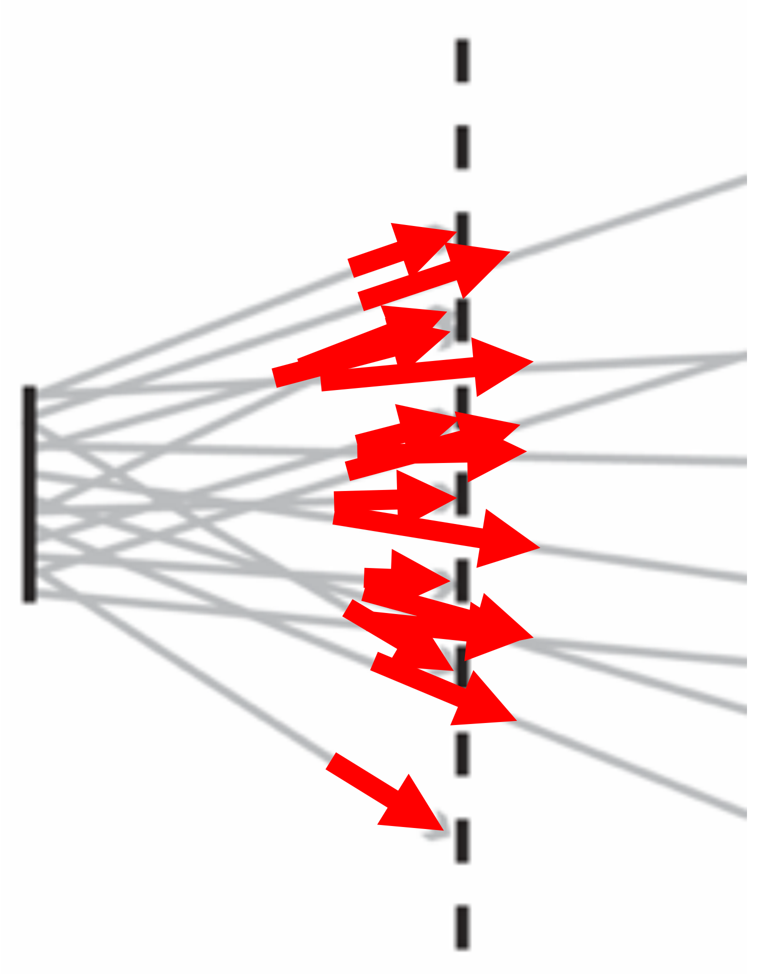
(d1)Scene image	(d2) EPI	(d3) Fourier transform of EPI

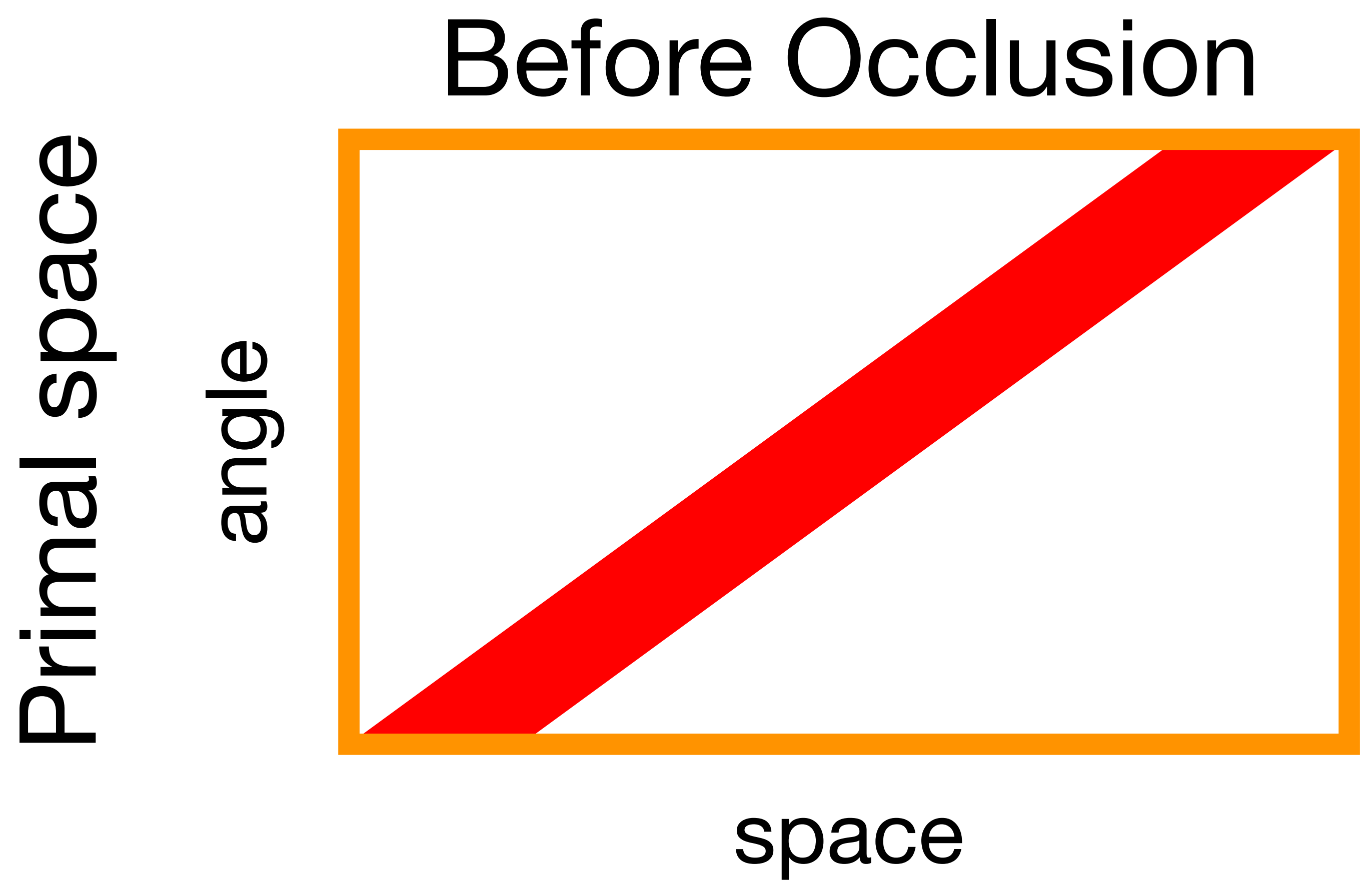
Occlusions

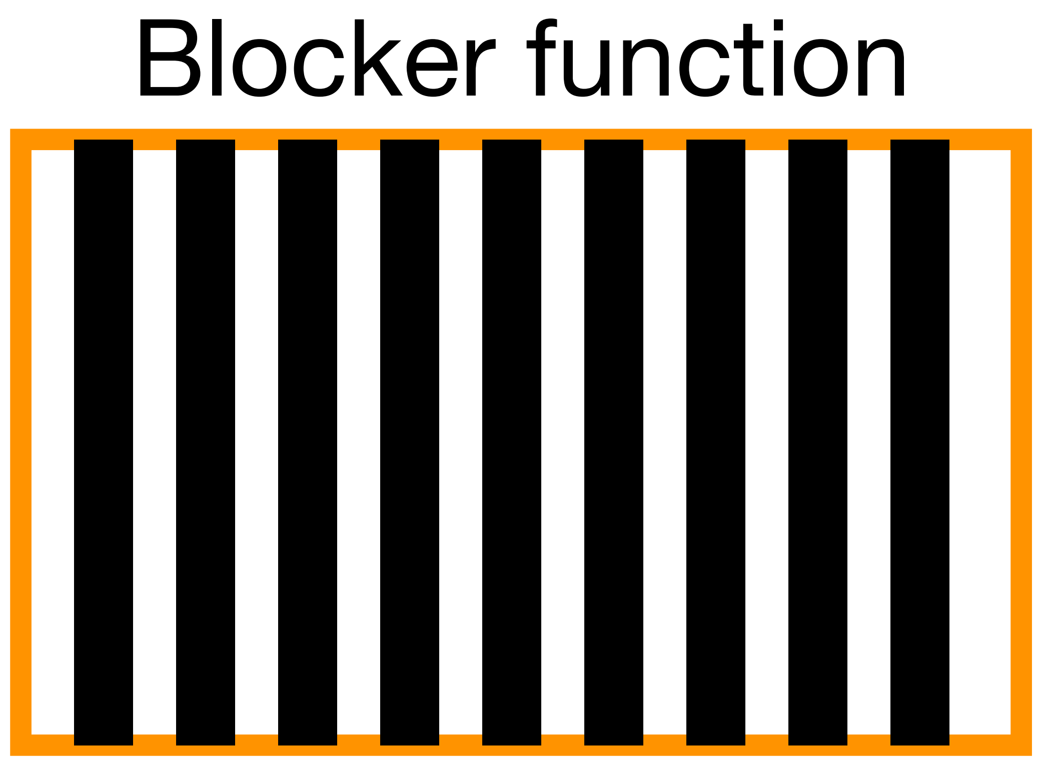
Consider planar occluders

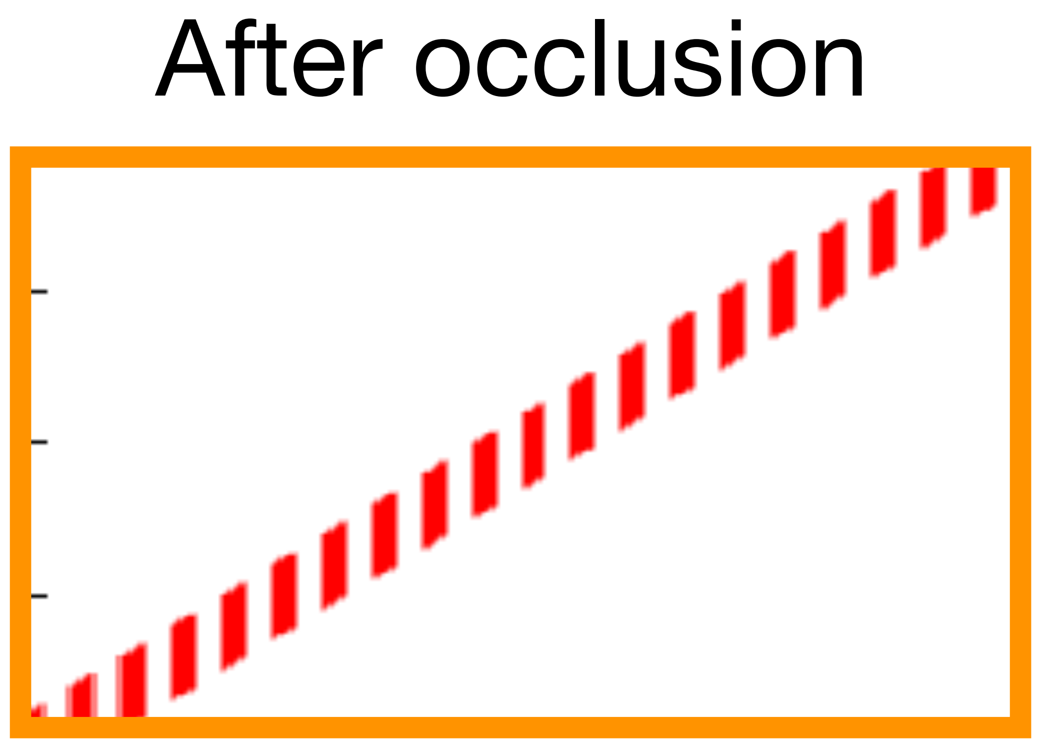
Multiplication by binary function

- mostly in space

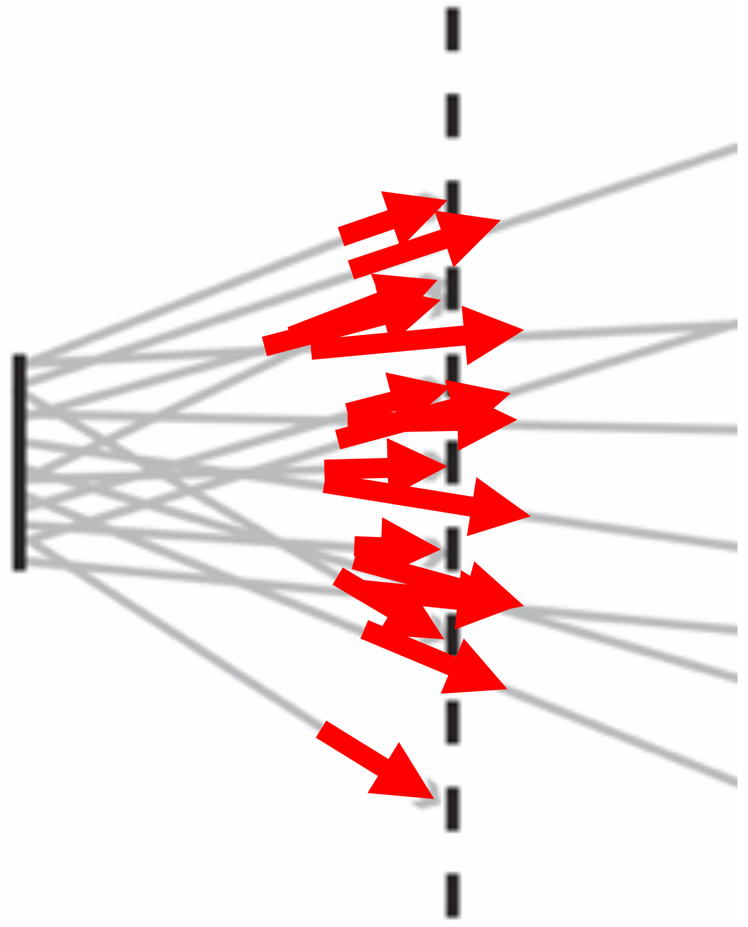




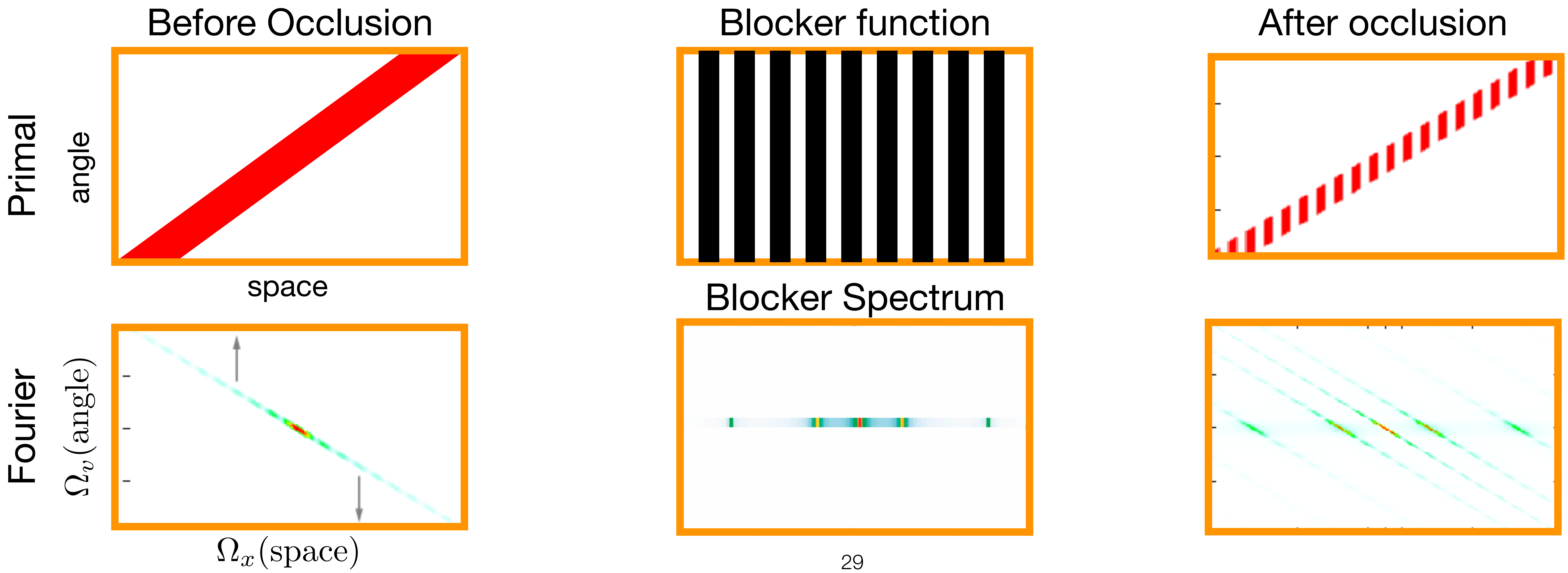




Occlusions



Multiplication in Primal domain is Convolution in Fourier domain



Main Transforms: Summary

	Transformations	Effects
Transport	Shear	
Occlusion	Convolution/Multiplication	Adds spatial frequencies
BRDF	Multiplication/Convolution	Removes angular frequencies
Curvature	Shear	

Frequency analysis of light transport [Durand et al. 2005]

Reconstructing Motion Blur

Motion blur

Objects move while the camera shutter is open

Image is "blurred" over time

Expensive for special effects

Necessary to remove "strobing" in animation

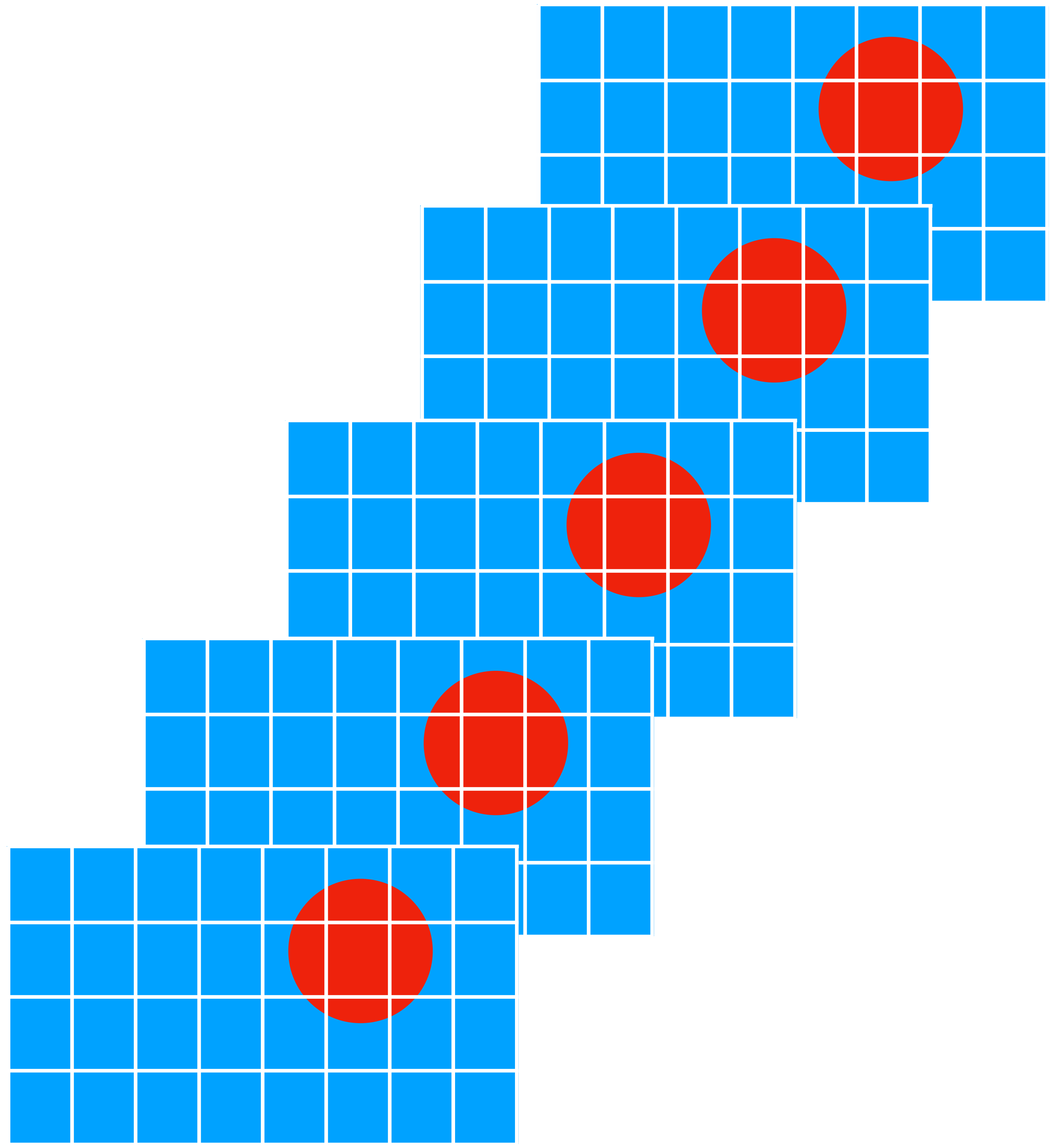
Garfield: A tale of two kitties

Rhythm & Hues Studios

The Incredibles

Pixar Animation Studios
Walt Disney Pictures

Motion blur: Simple approach



$t = 0.1$

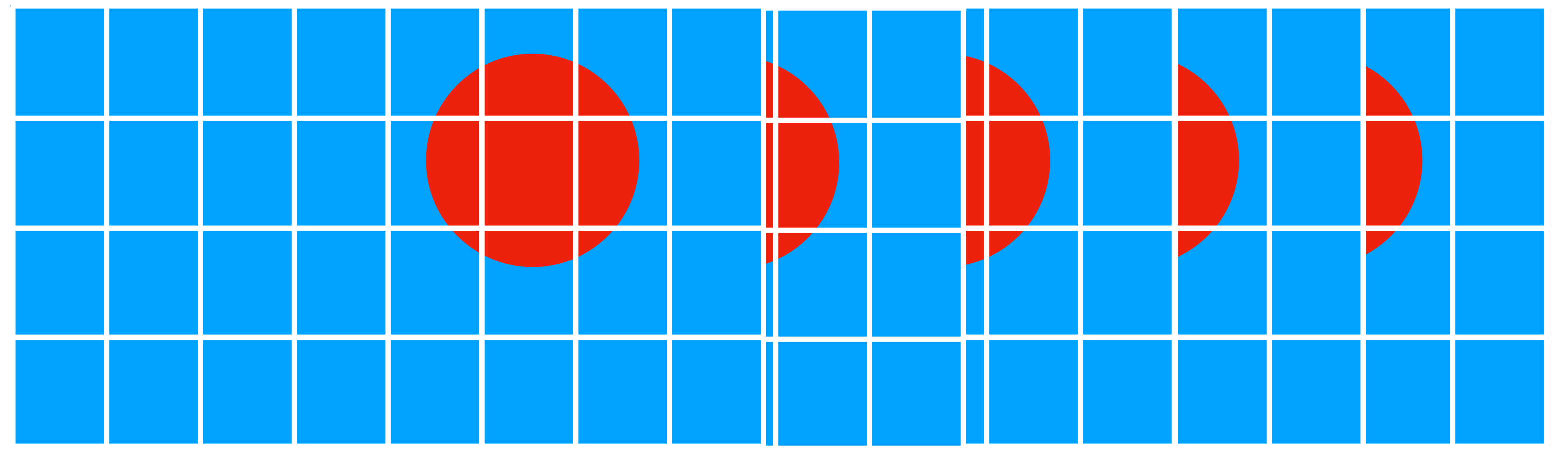
$t = 0.3$

$t = 0.5$

$t = 0.7$

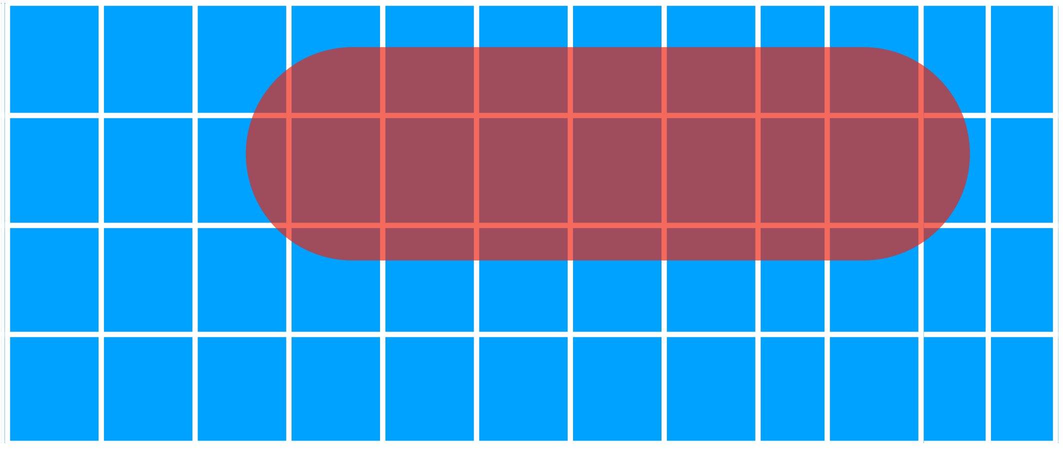
$t = 0.9$

Motion blur: Simple approach



$t = 0.5$

Motion blur: Simple approach



$$t \in [0, 1]$$

Simple approach

The simple approach is expensive

Can we do better?

Observation

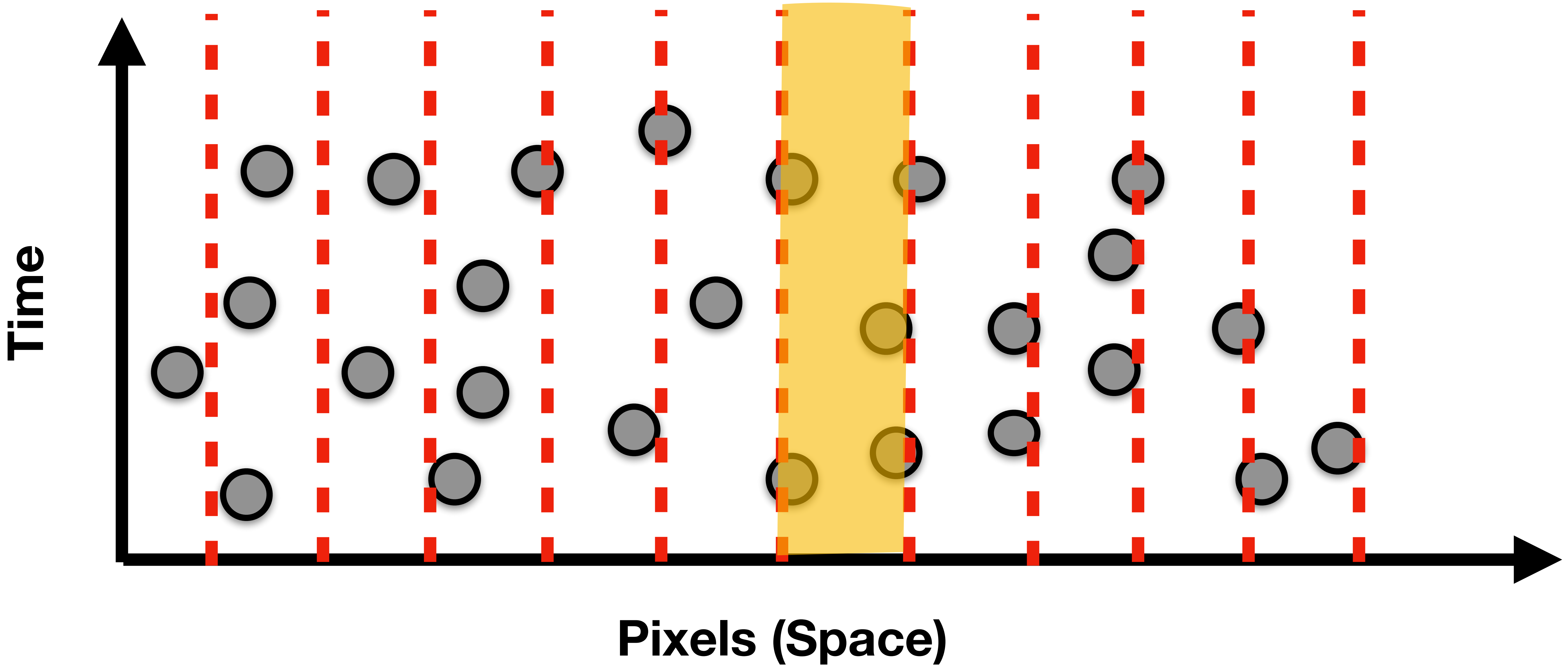
Motion blur is expensive

Motion blur removes spatial complexity

Standard Method

Use axis-aligned pixel filters at each pixel

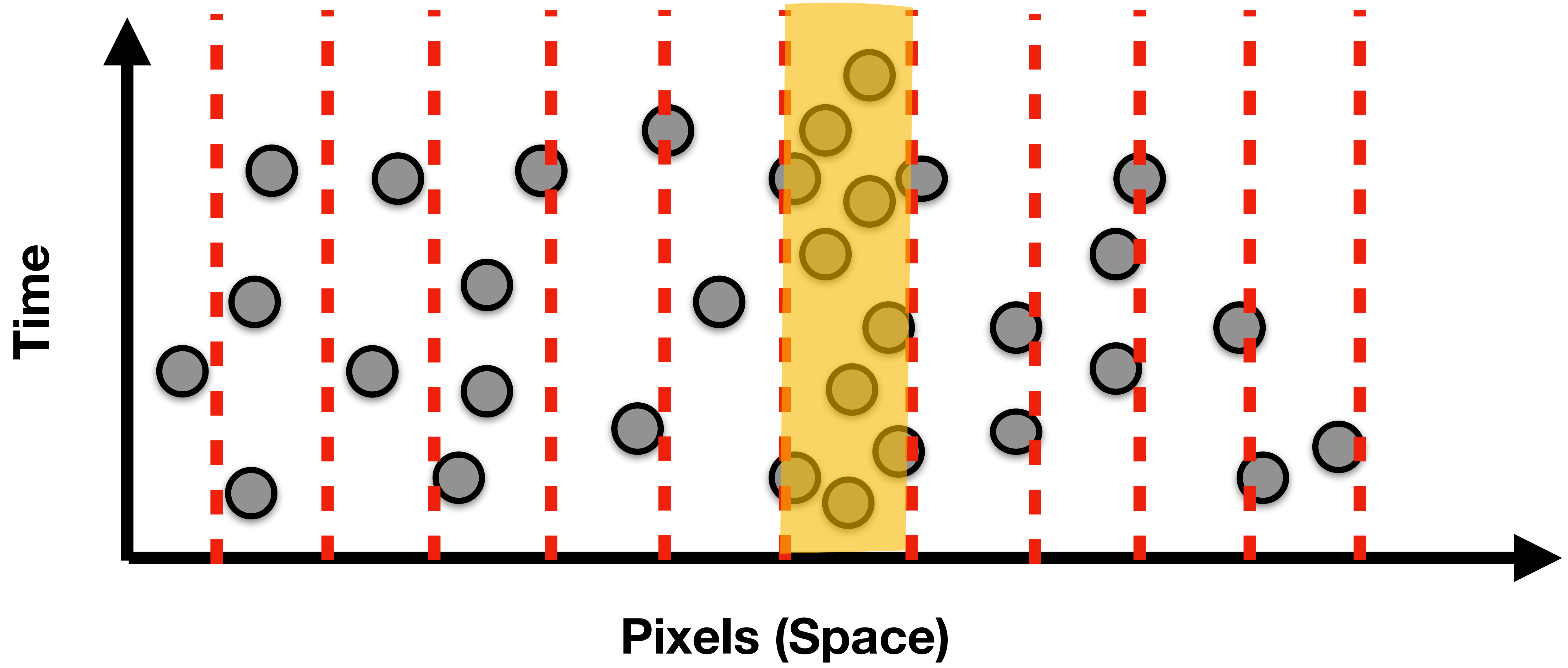
Requires many samples



Standard Method

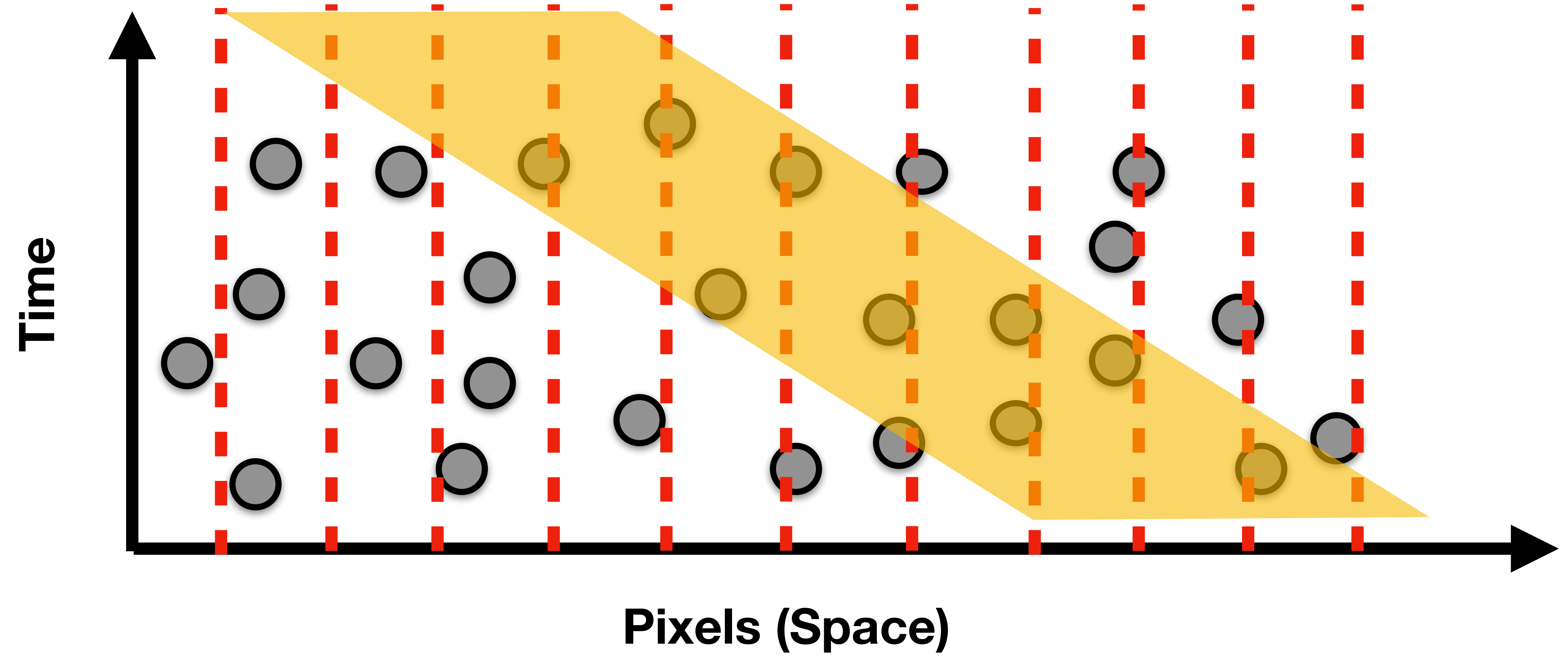
Use axis-aligned pixel filters at each pixel

Requires many samples



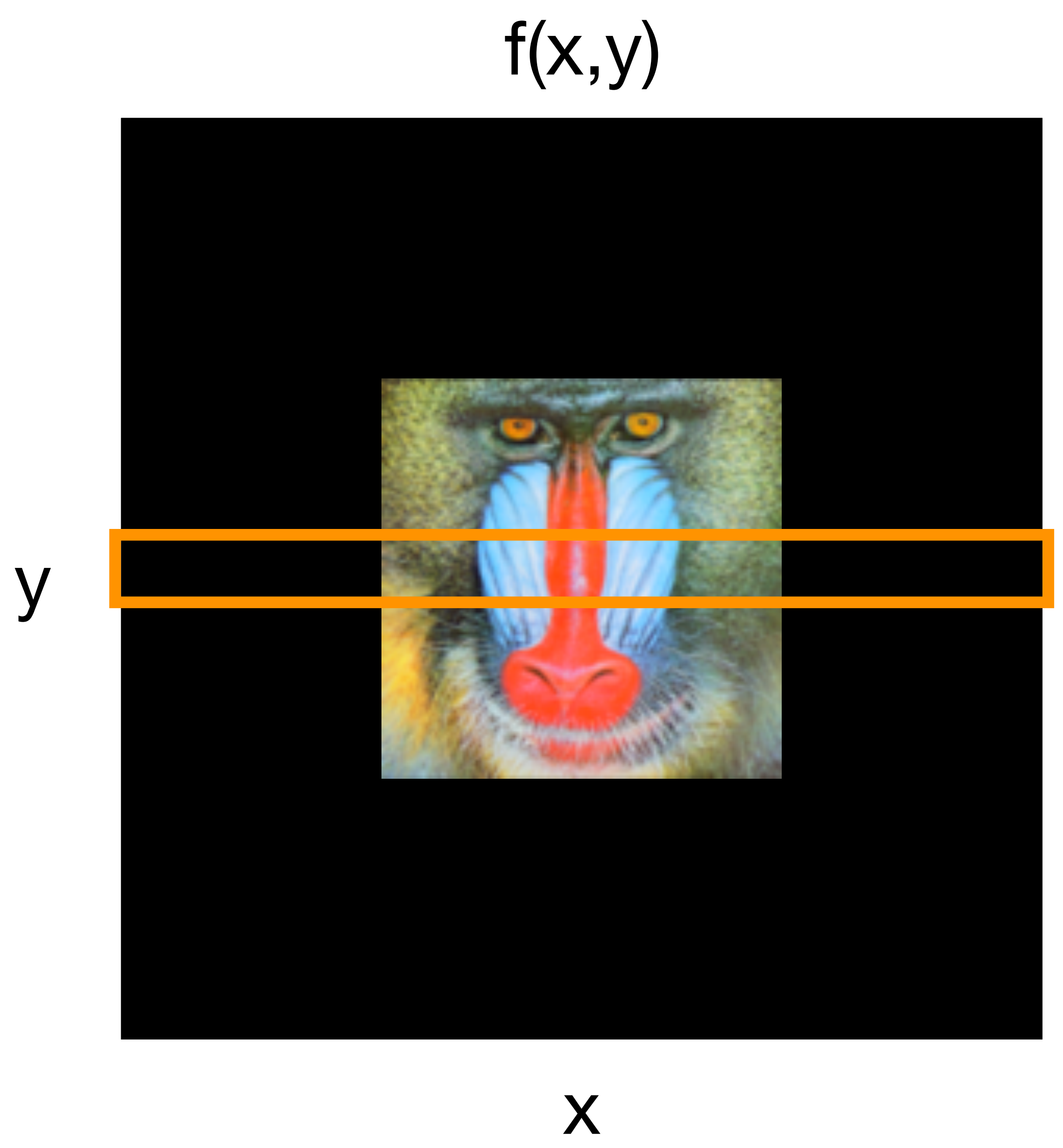
Filter shearing based on frequency analysis of light transport

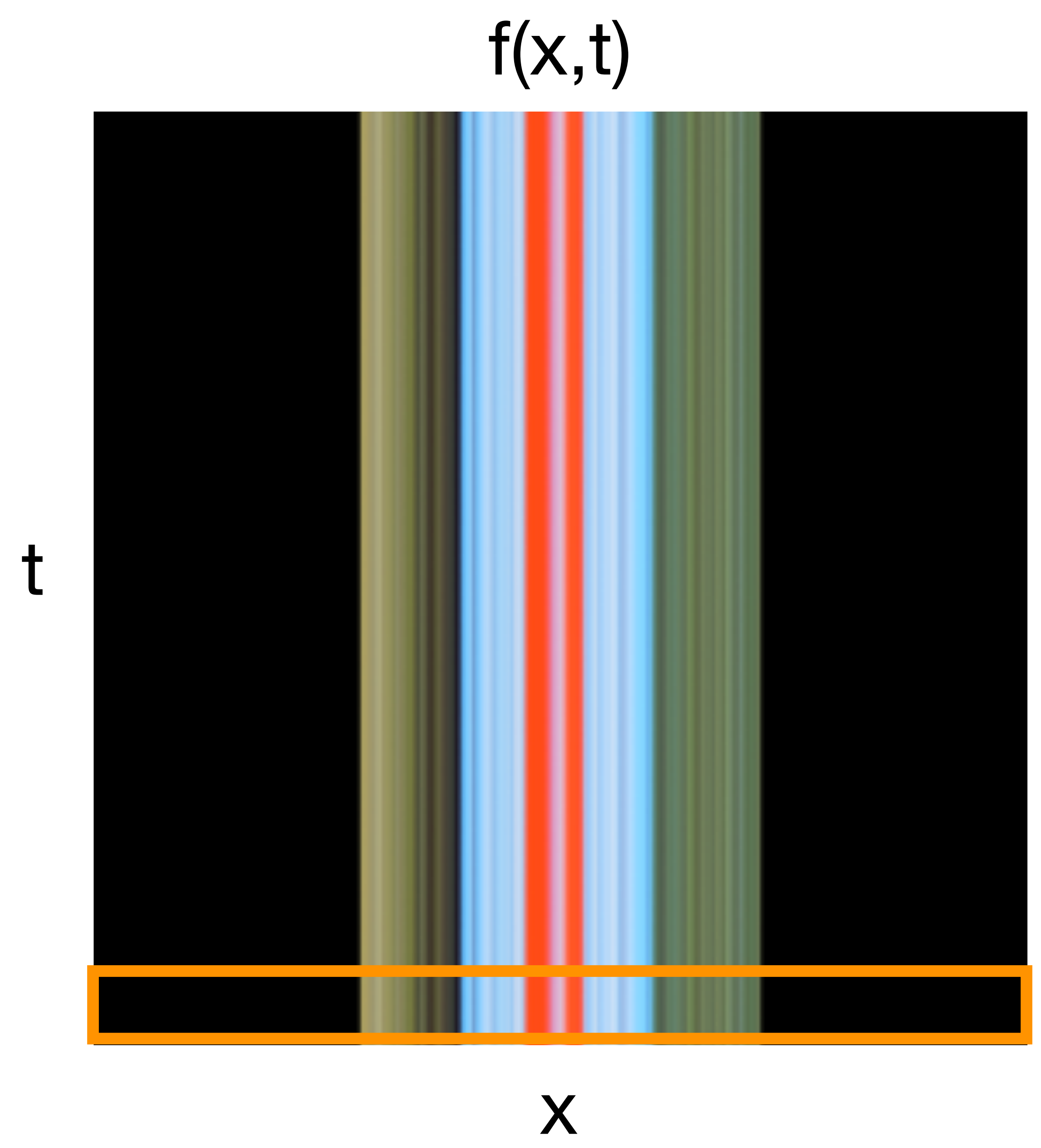
We will look at how to reuse nearby pixel samples to reconstruct using filters derived using the frequency analysis of light transport



Basic Example

No velocity: static scene $t \in [0, 1)$

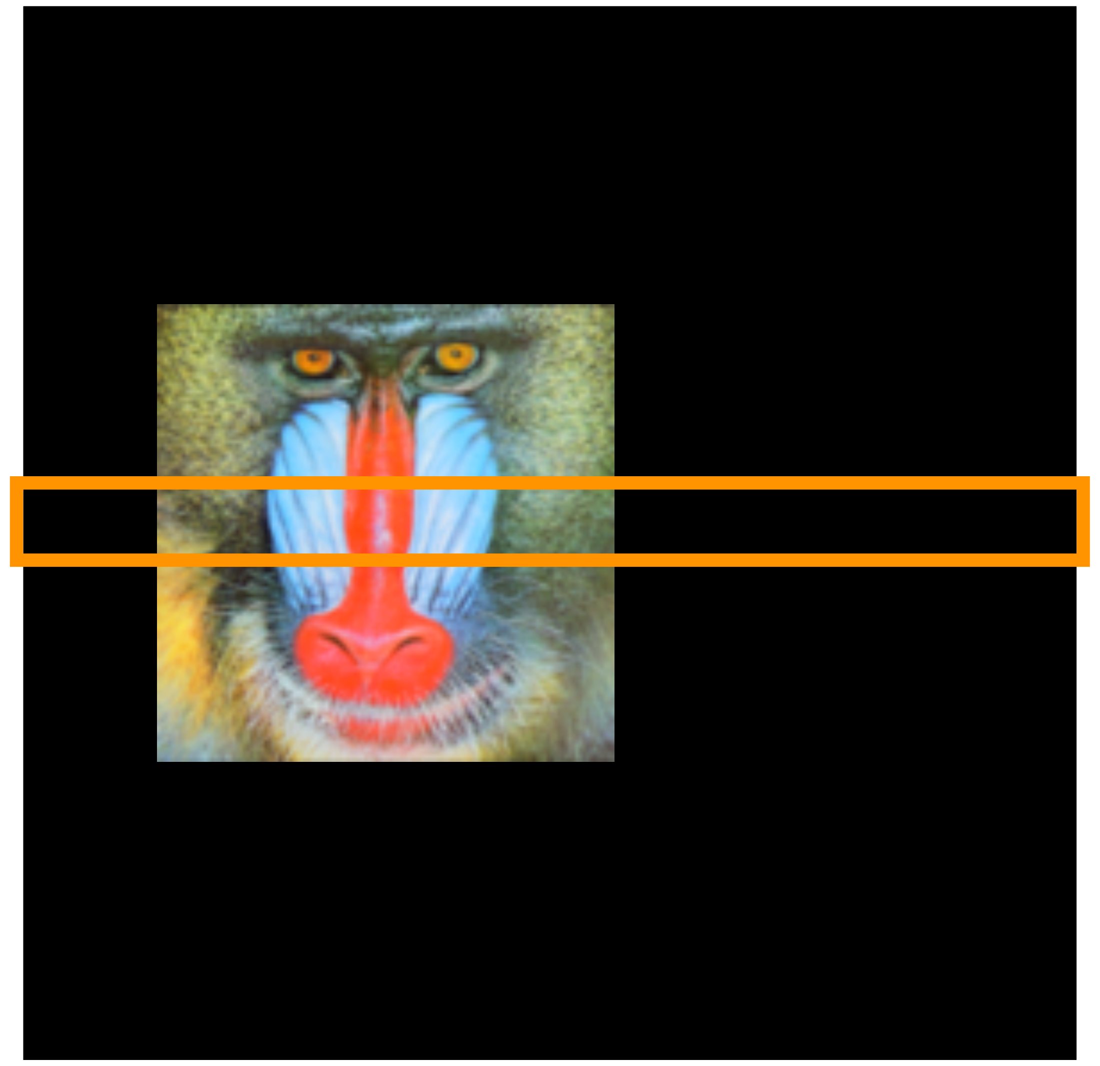




Basic Example

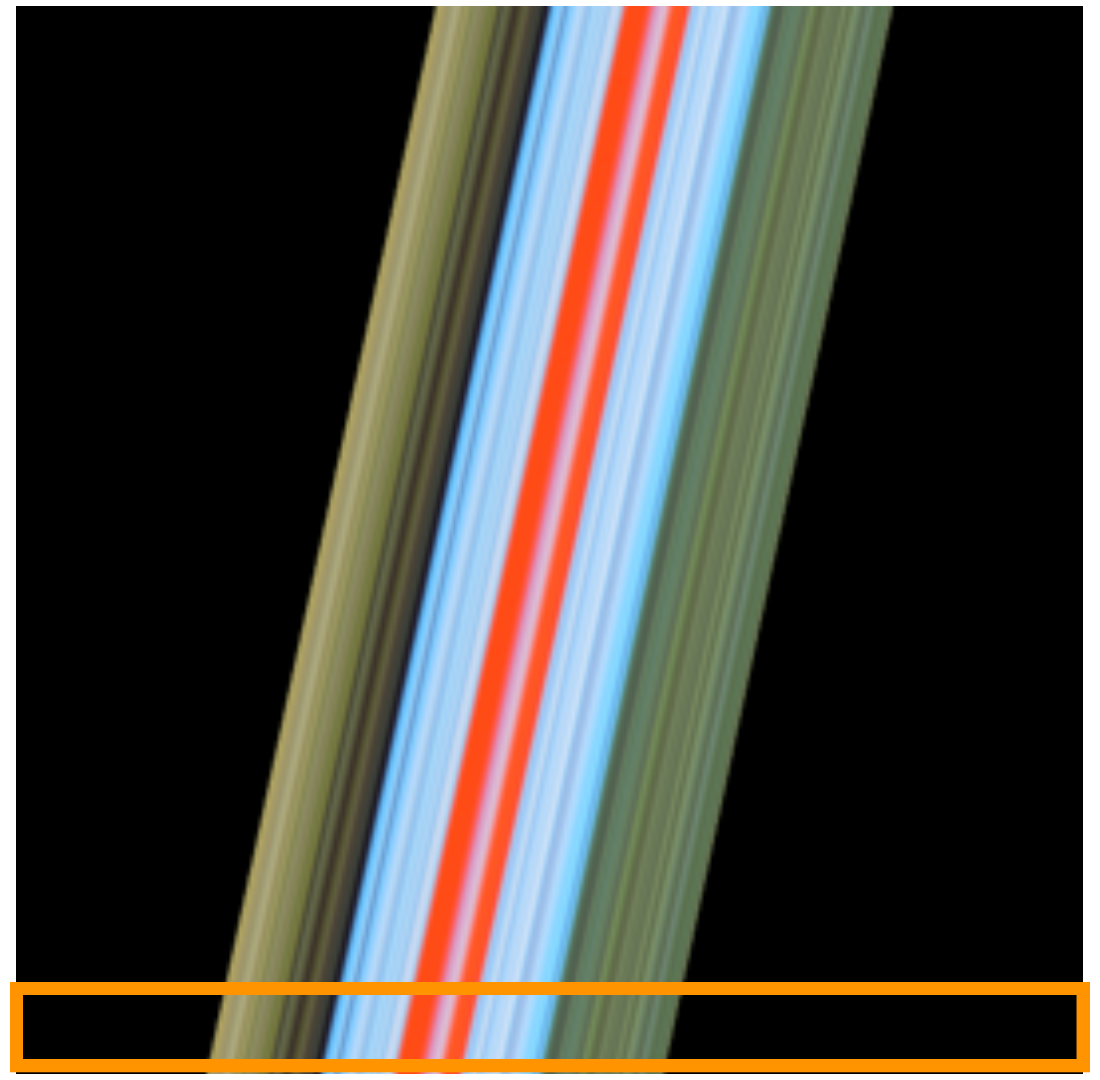
Low velocity $t \in [0, 1)$

f(x,y)

y 

x

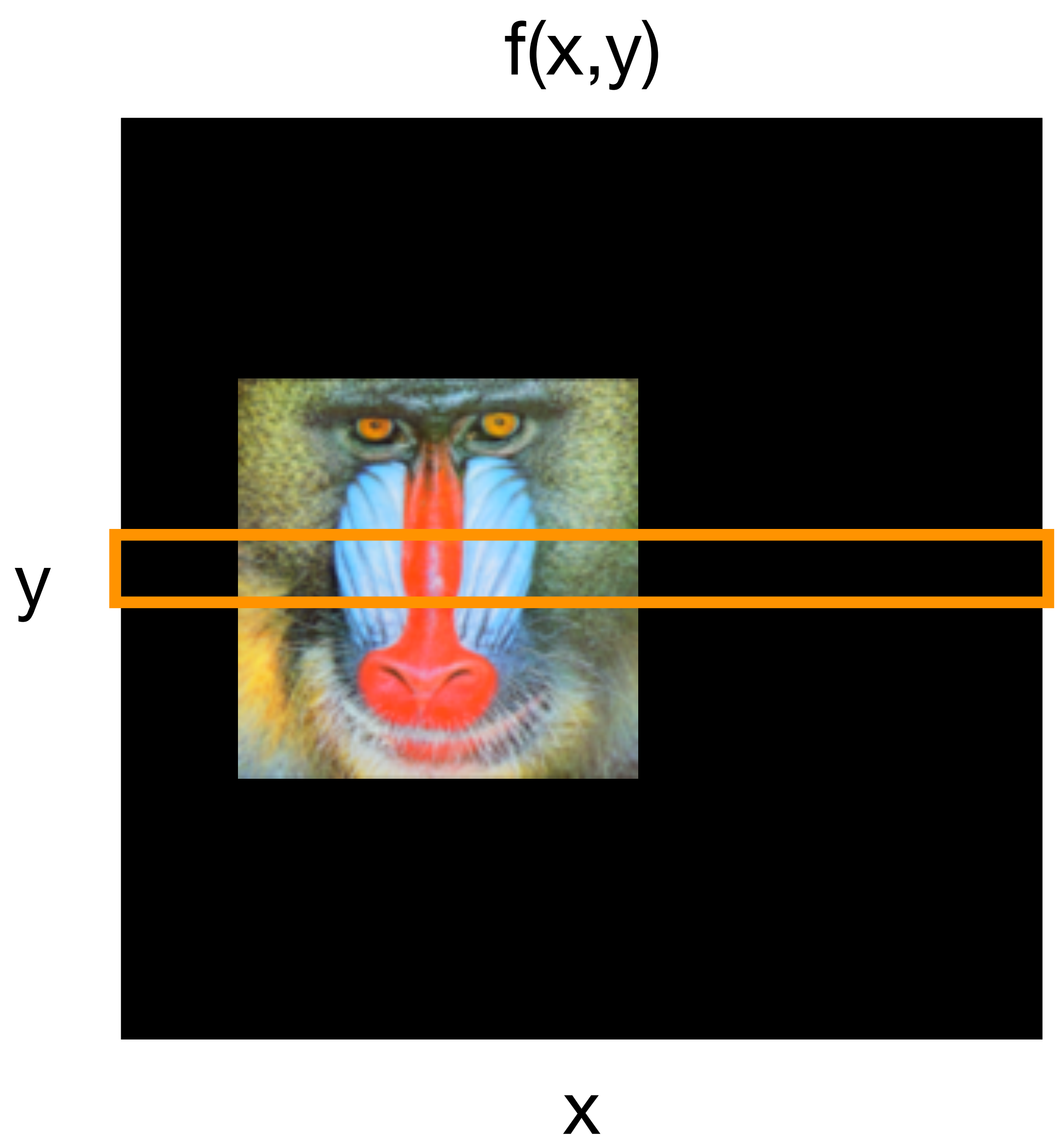
f(x,t)

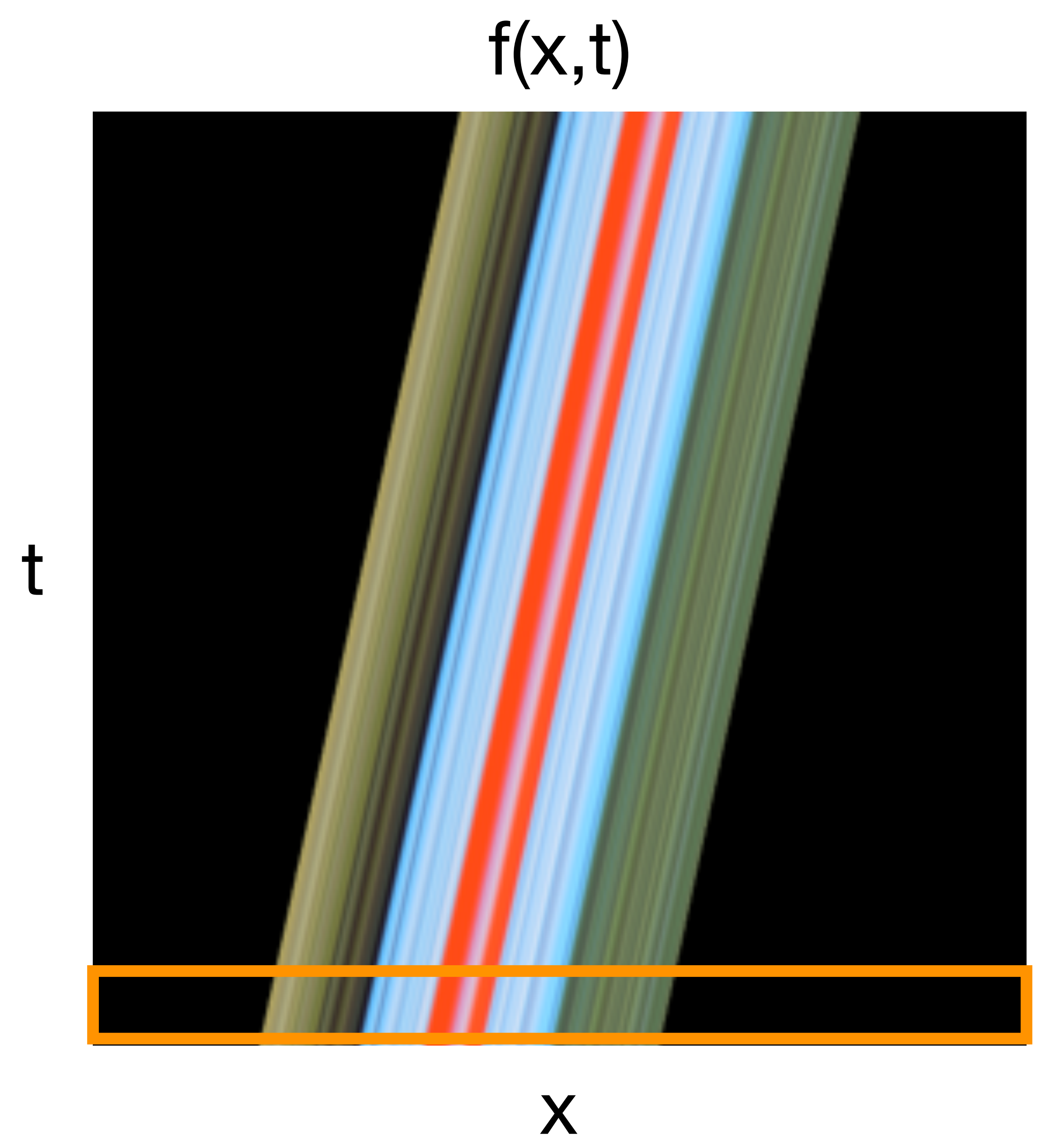
t 

x

Basic Example

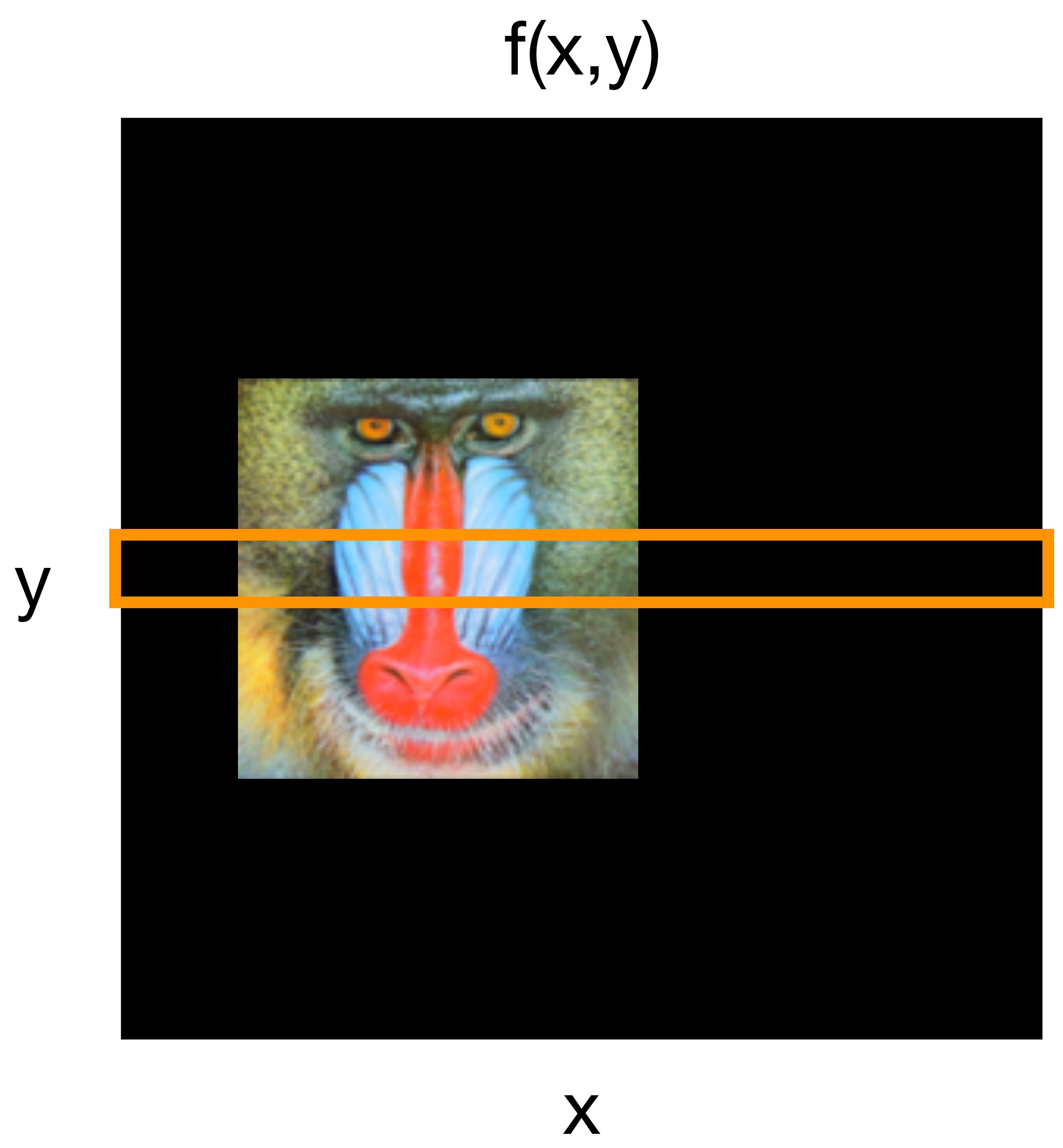
Low velocity $t \in [0, 1)$

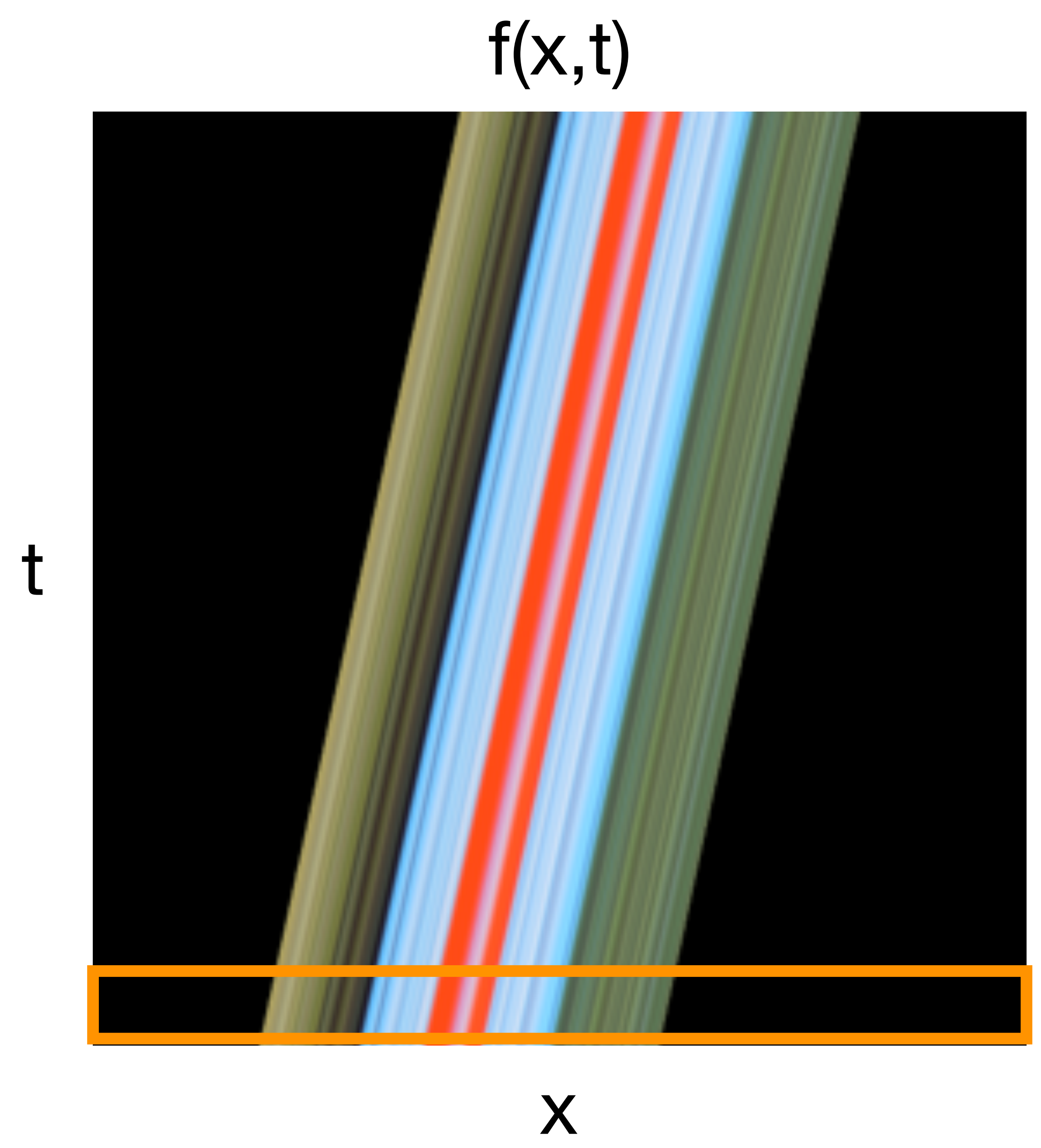




Basic Example

Low velocity $t \in [0, 1)$

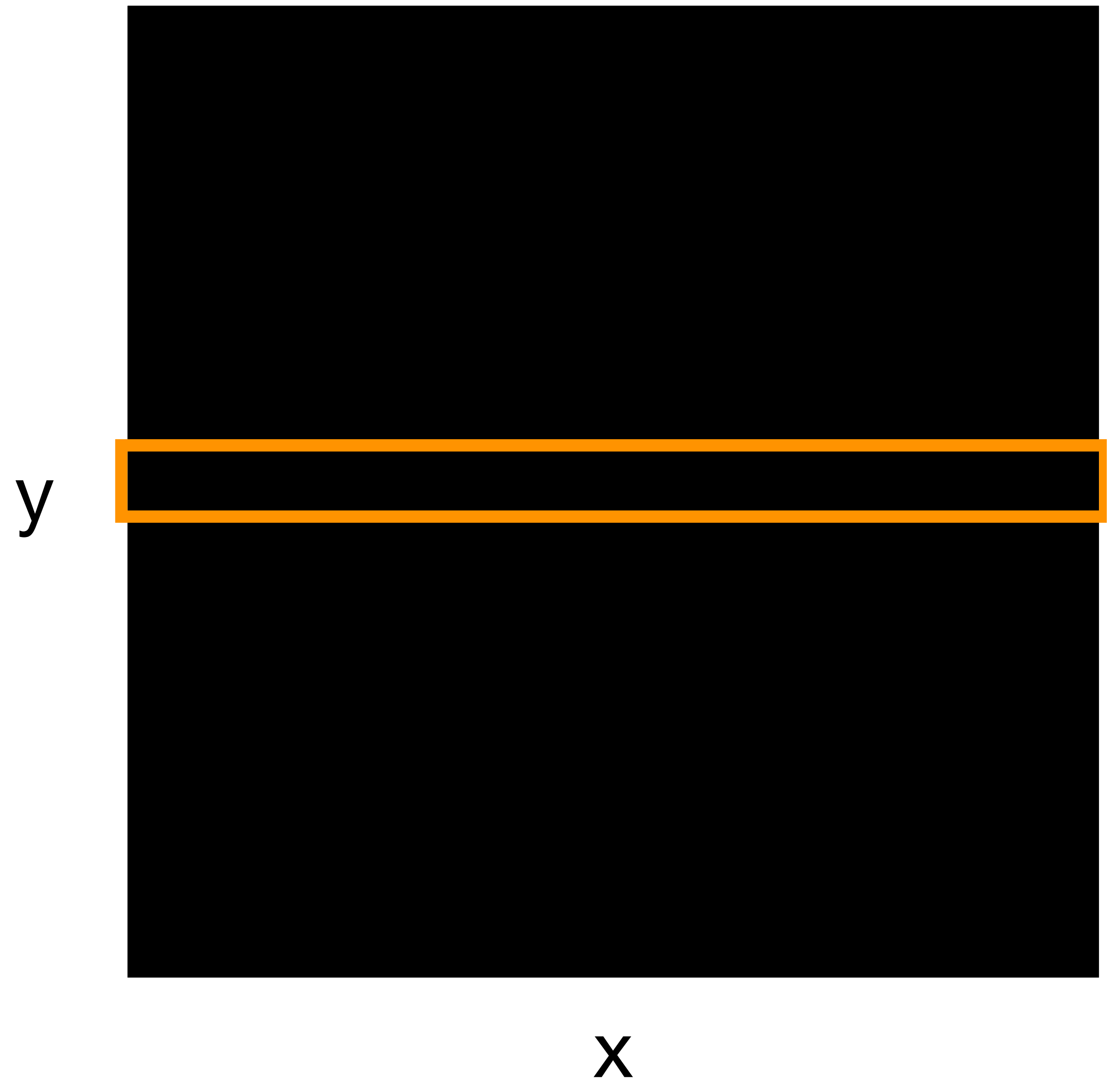




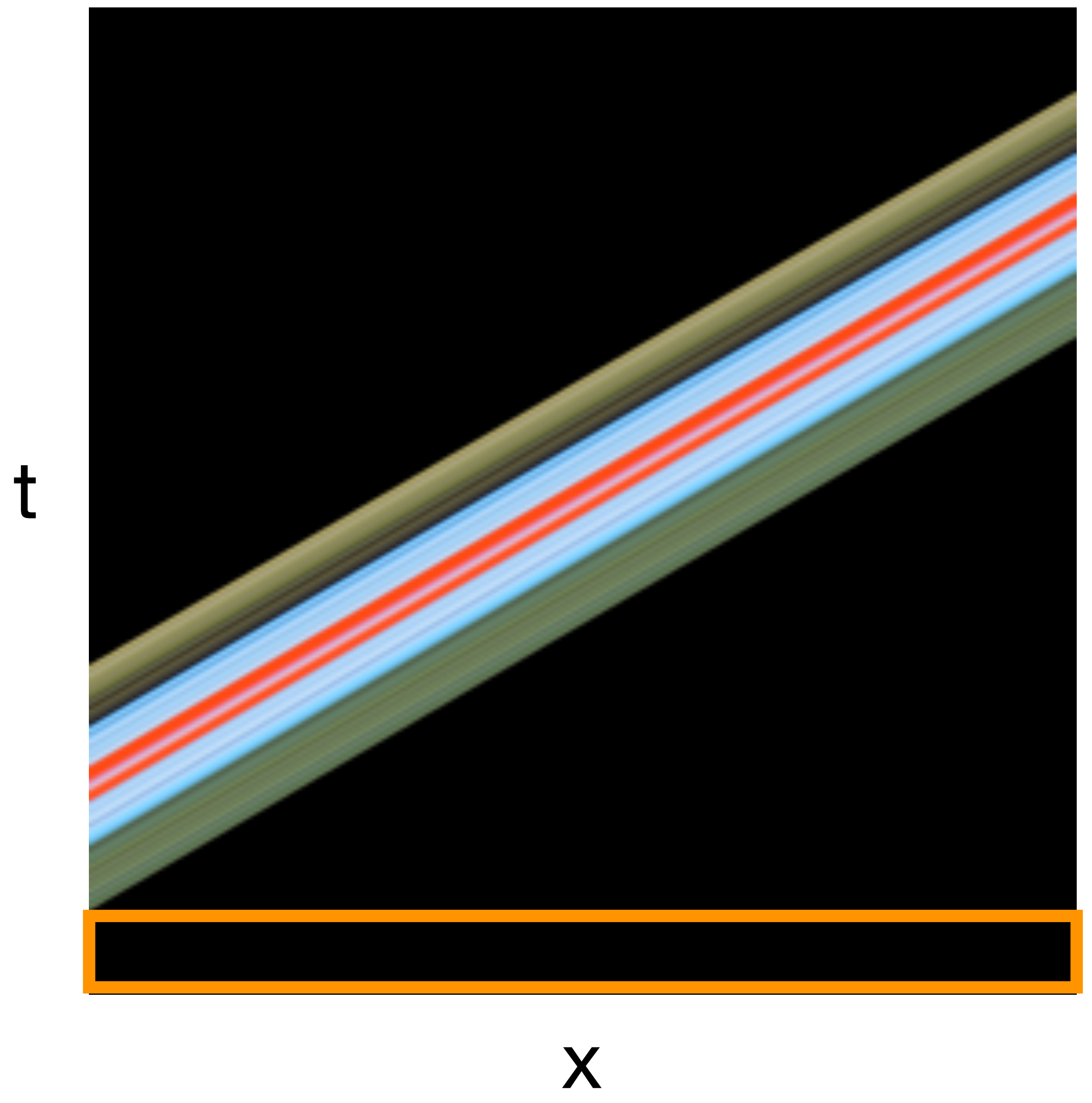
Basic Example

High velocity $t \in [0, 1)$

f(x,y)

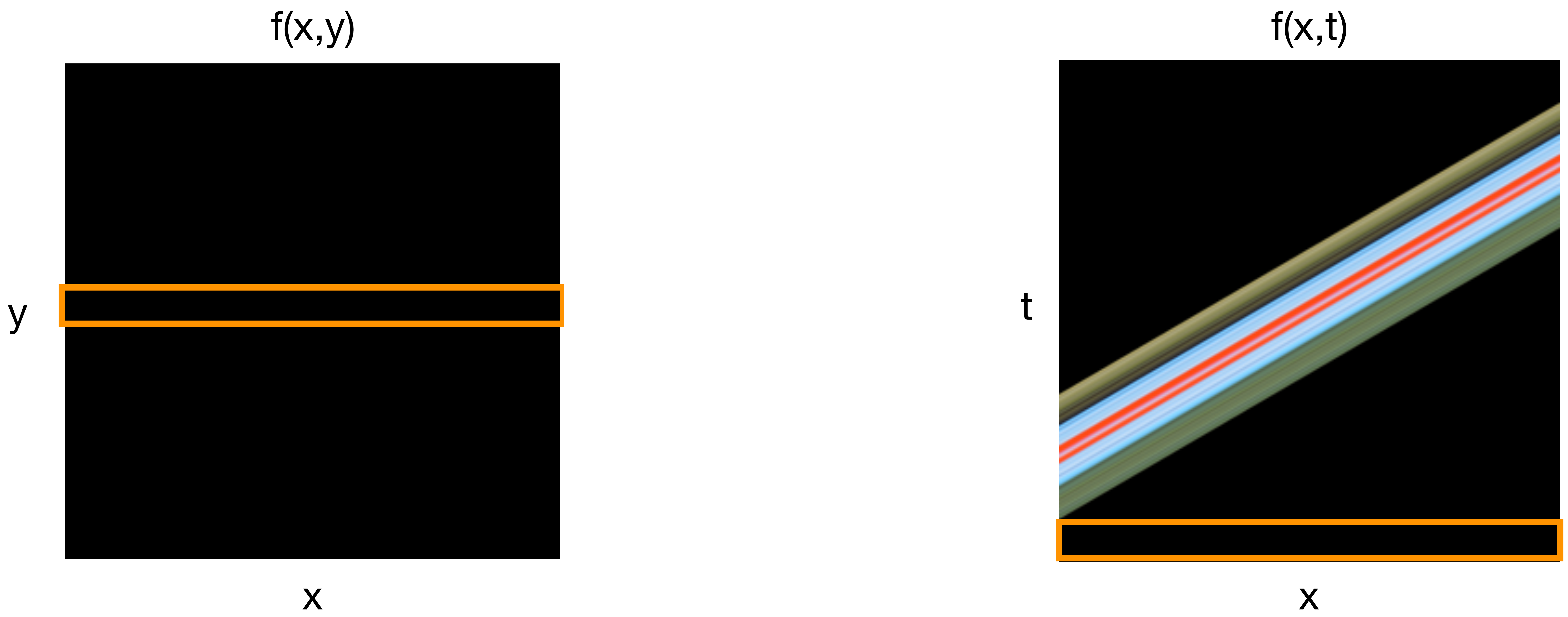


f(x,t)



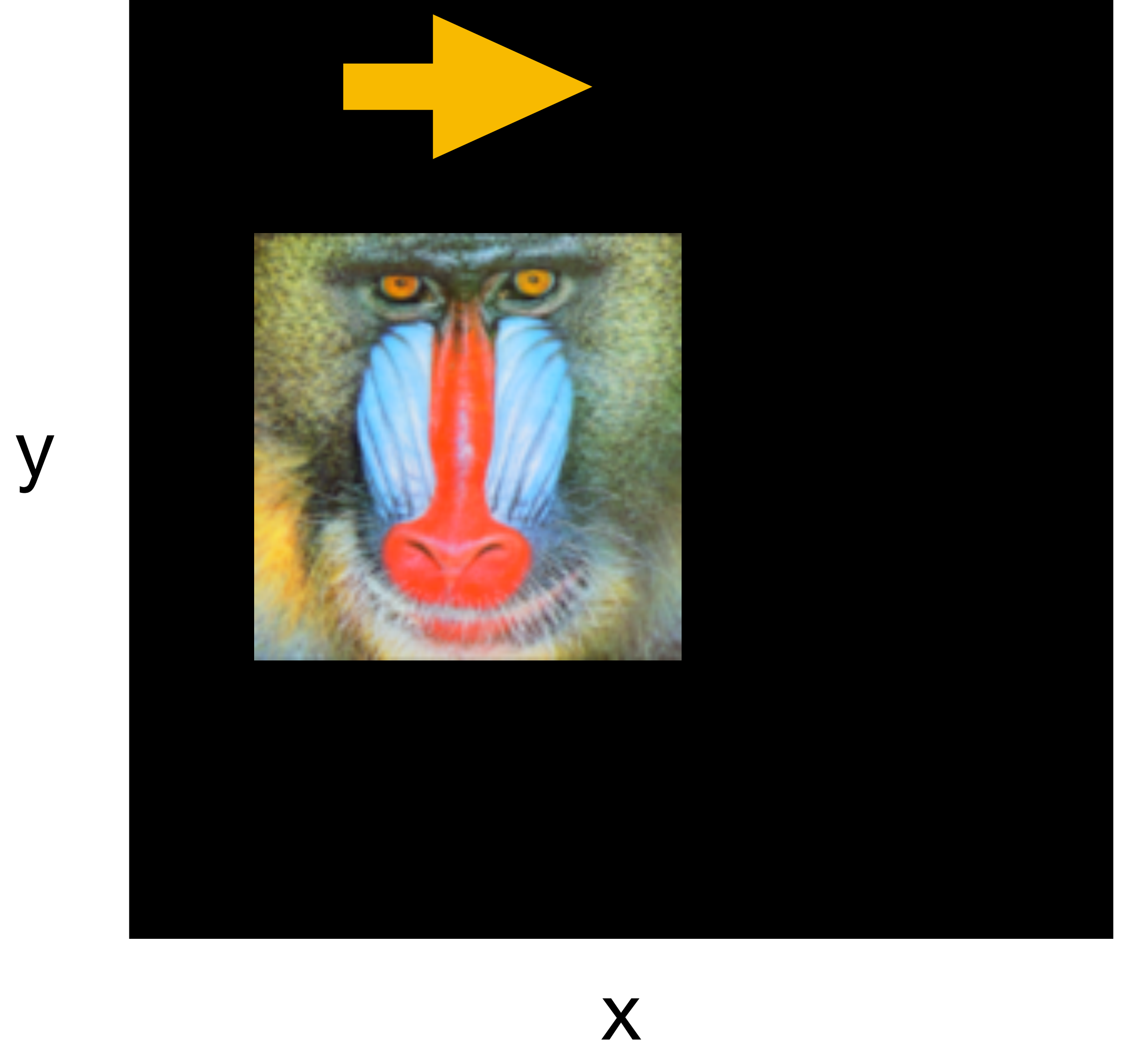
Basic Example

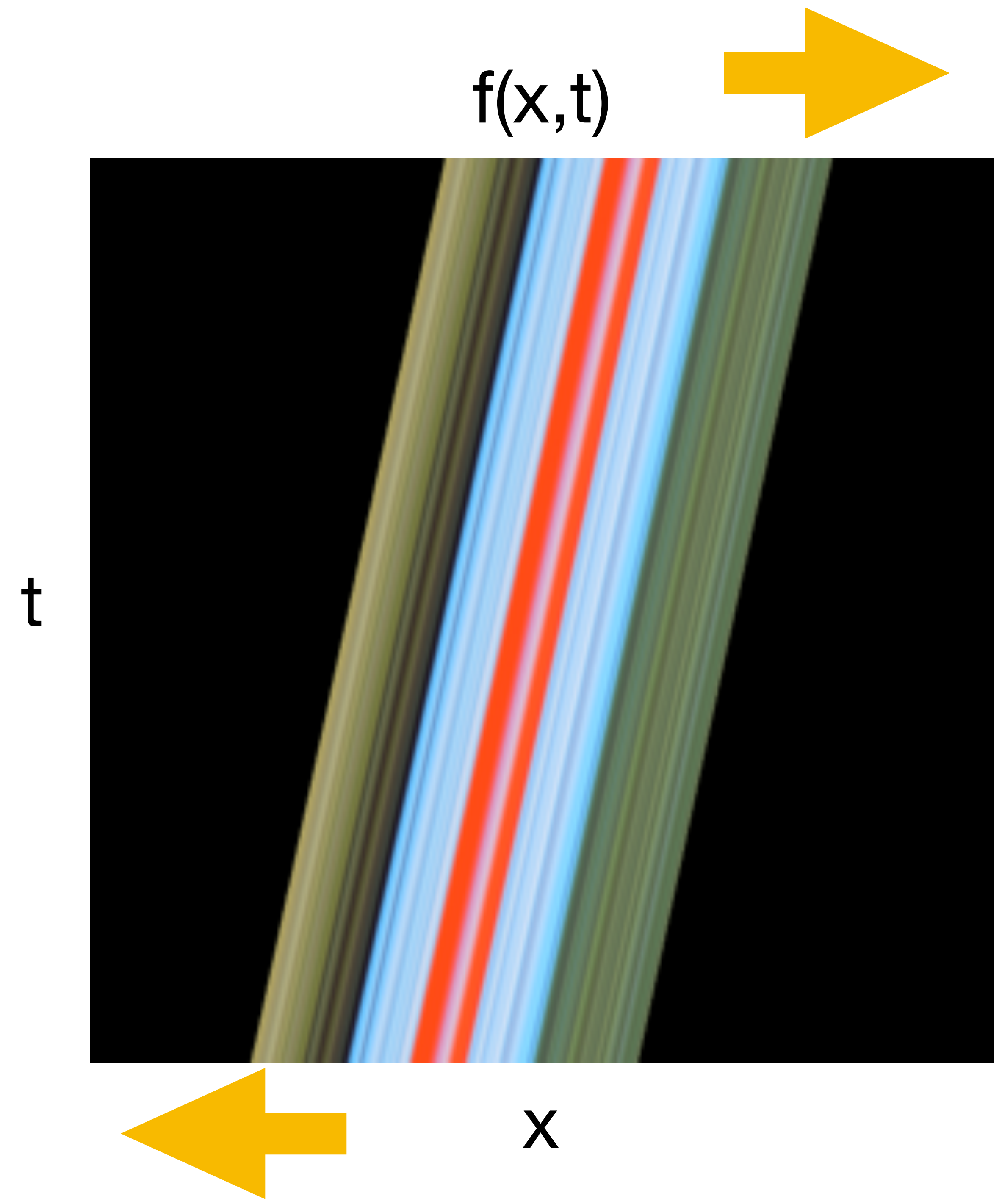
High velocity $t \in [0, 1)$



Shear in space-time

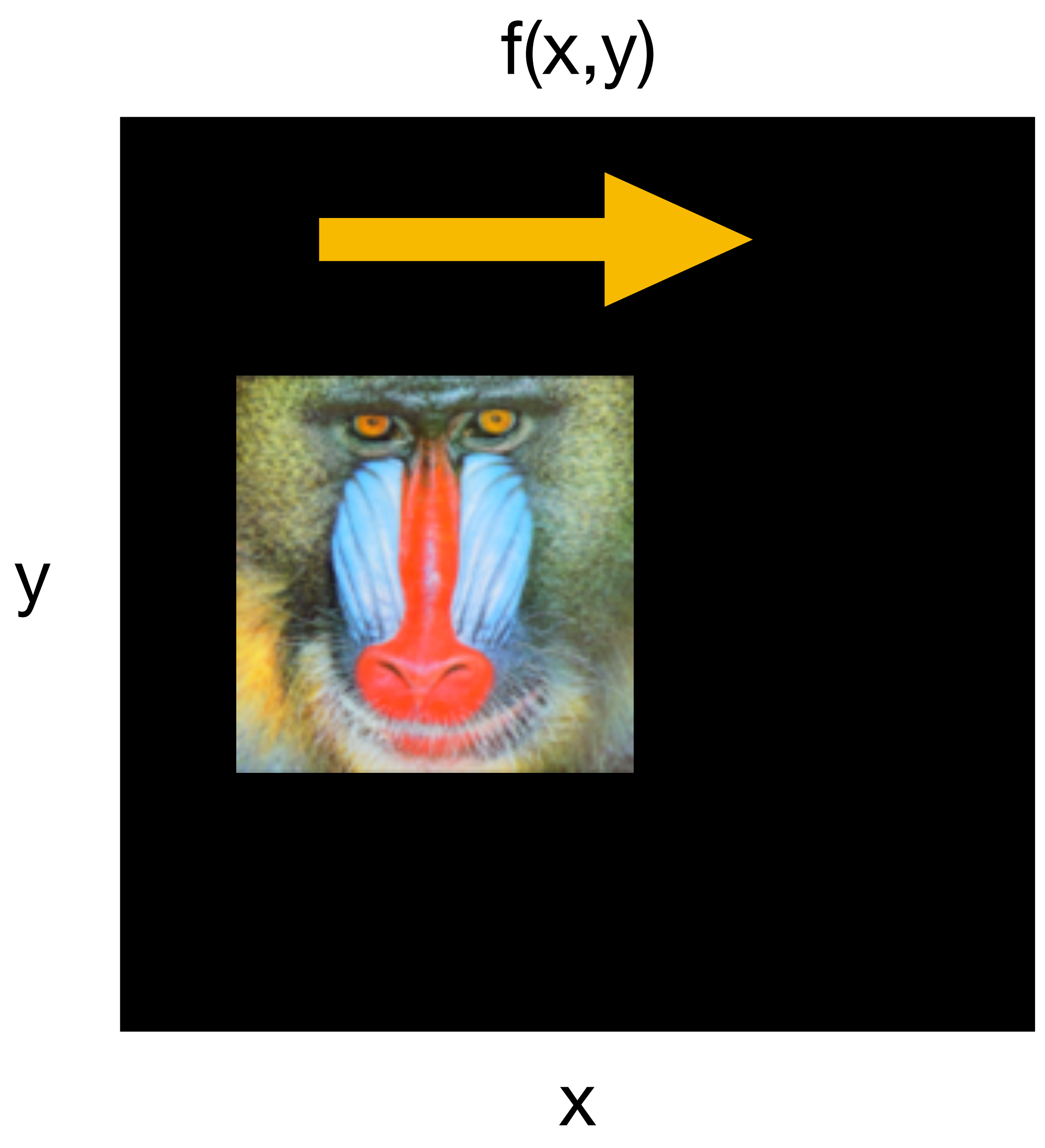
Object moving with low velocity $t \in [0, 1)$

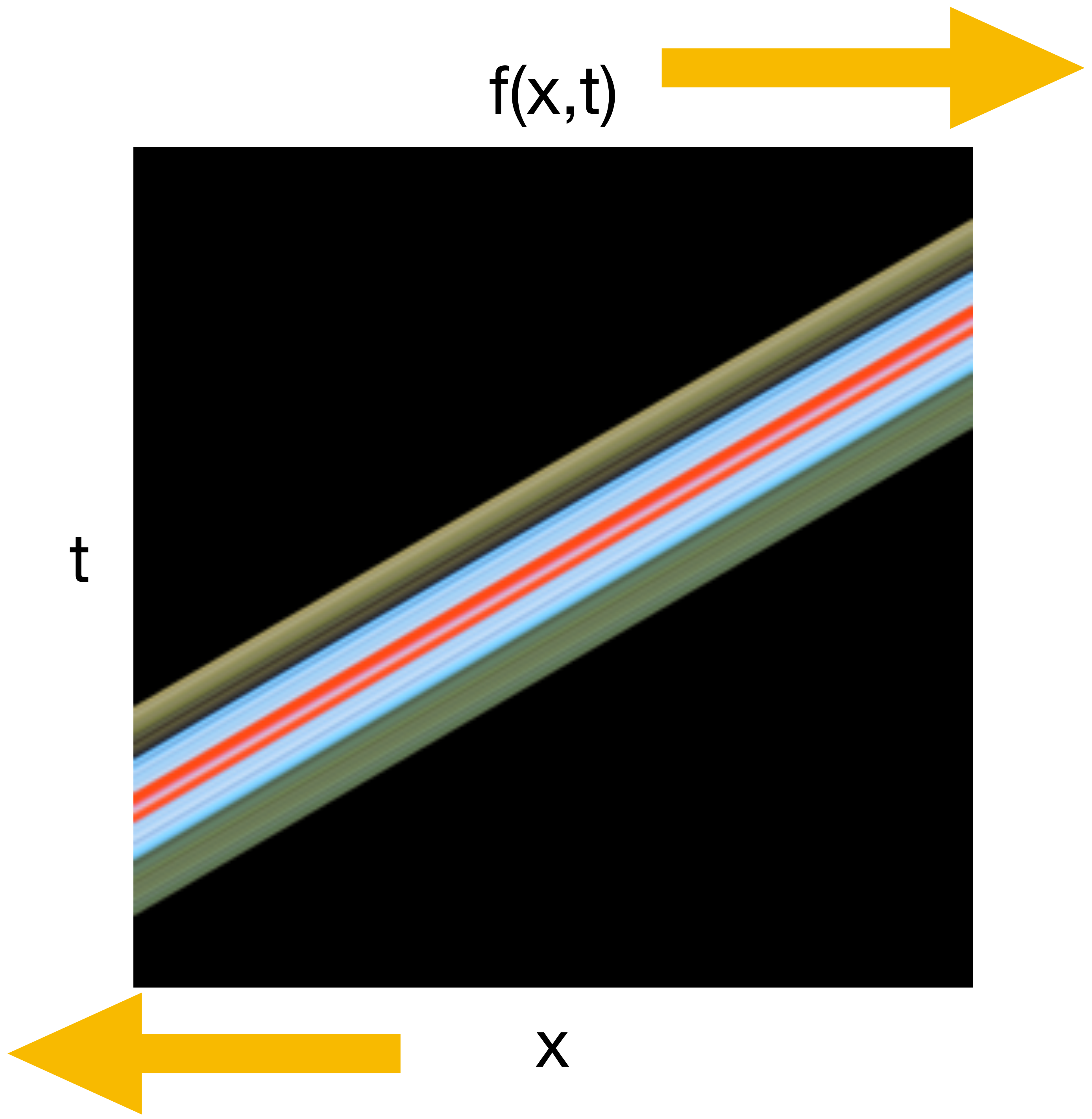




Large shear in space-time

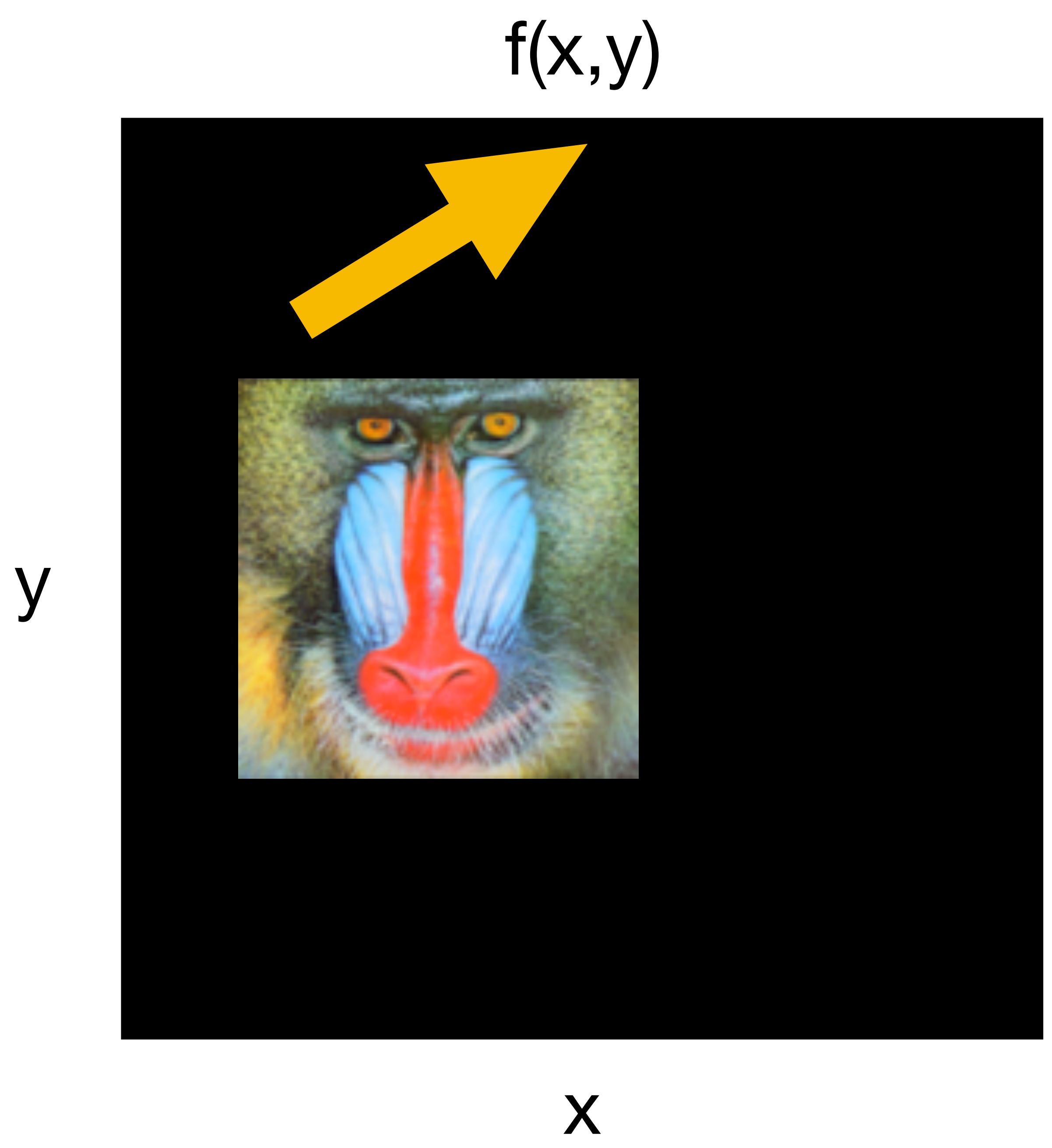
Object moving with high velocity $t \in [0, 1)$

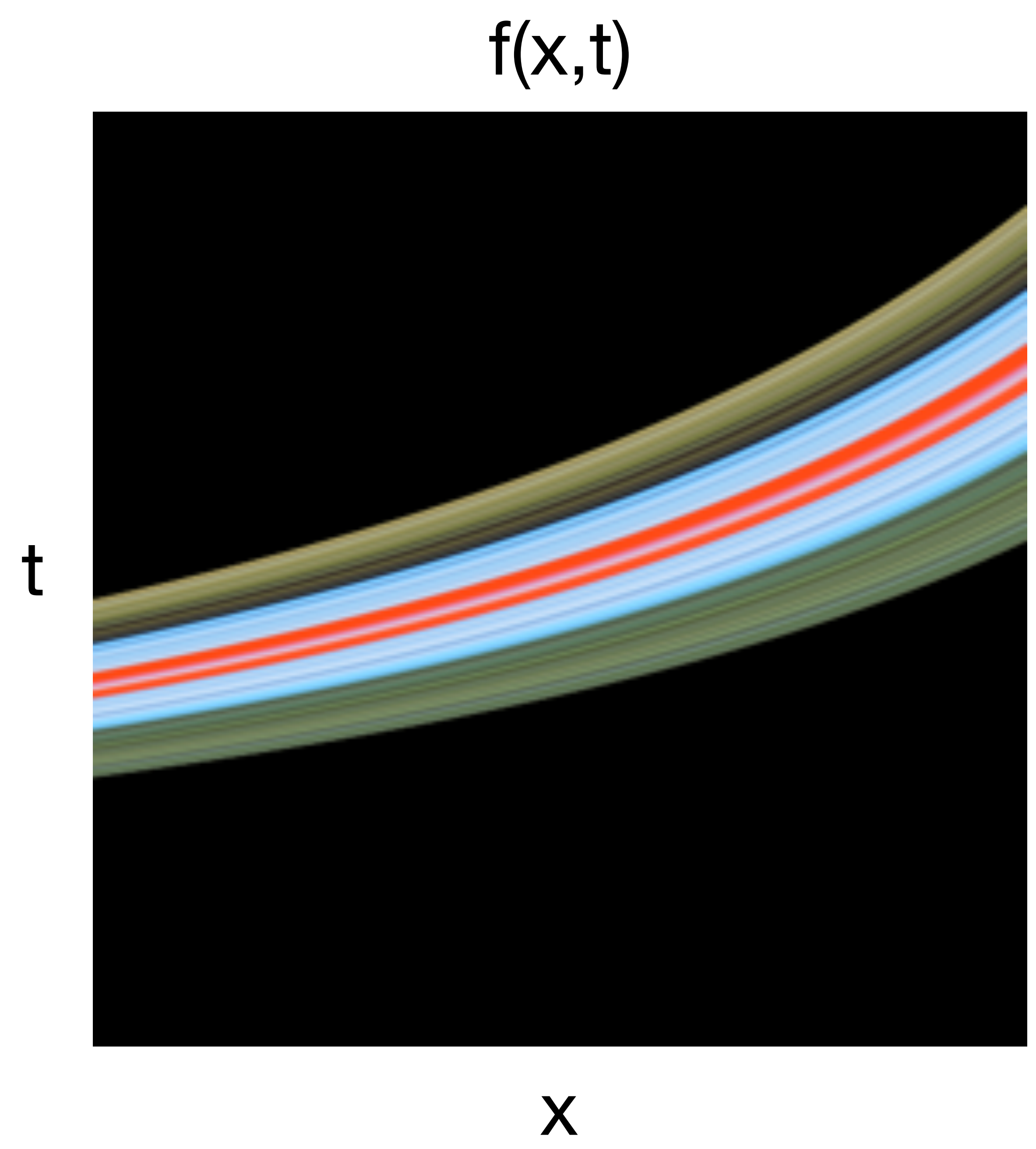




Shear in space-time

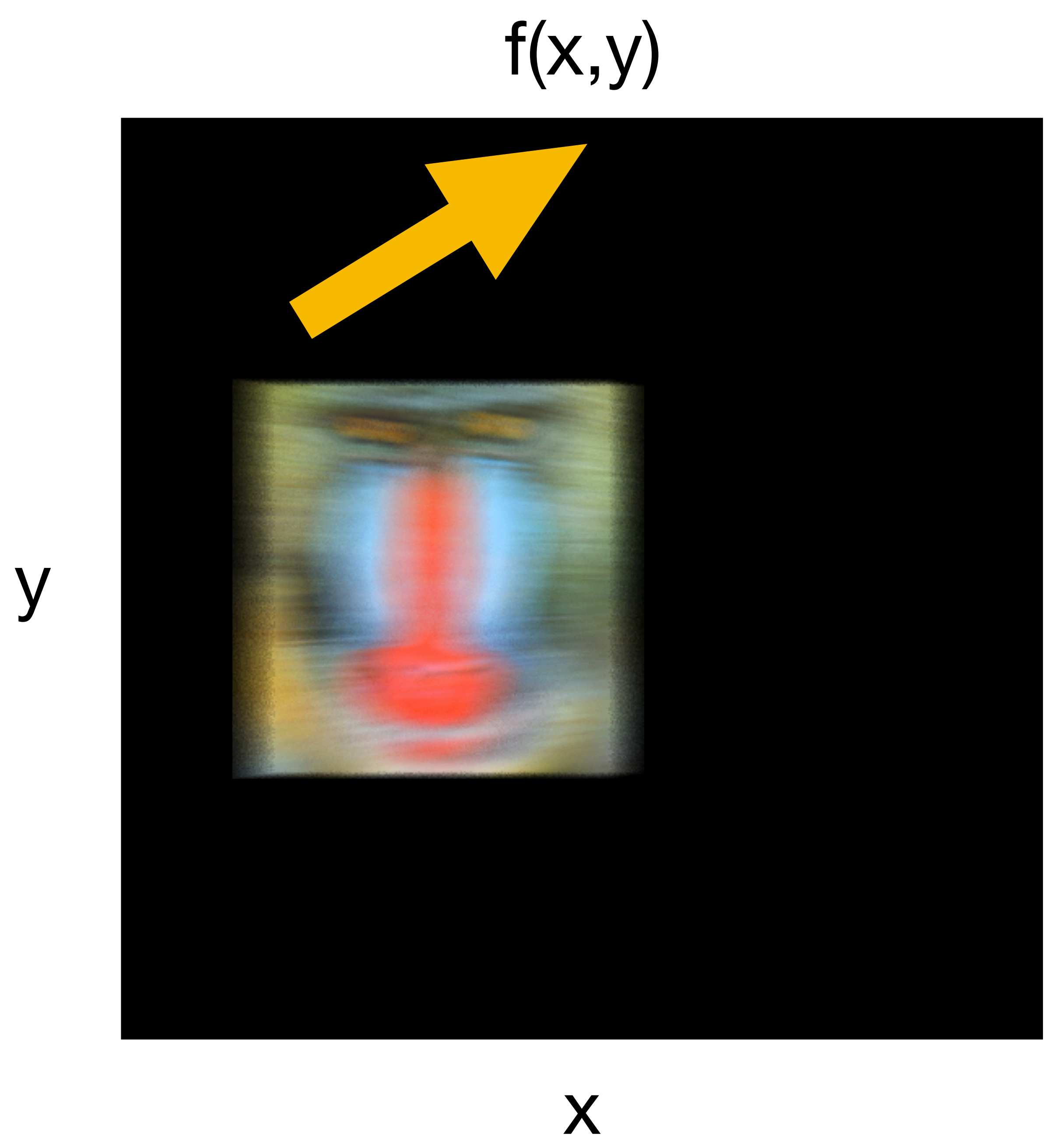
Object moving away from the camera $t \in [0, 1)$

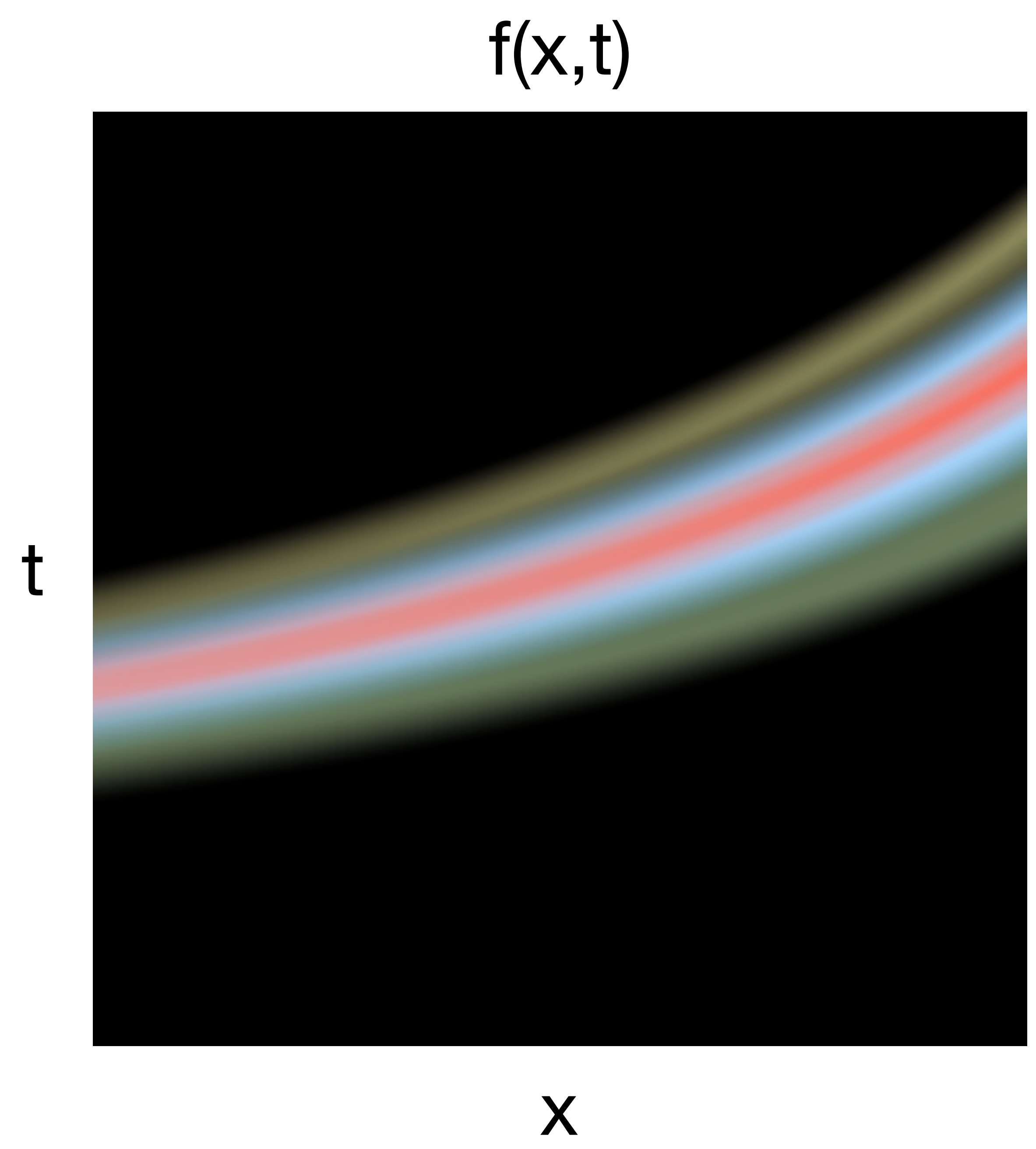




Camera shutter filter

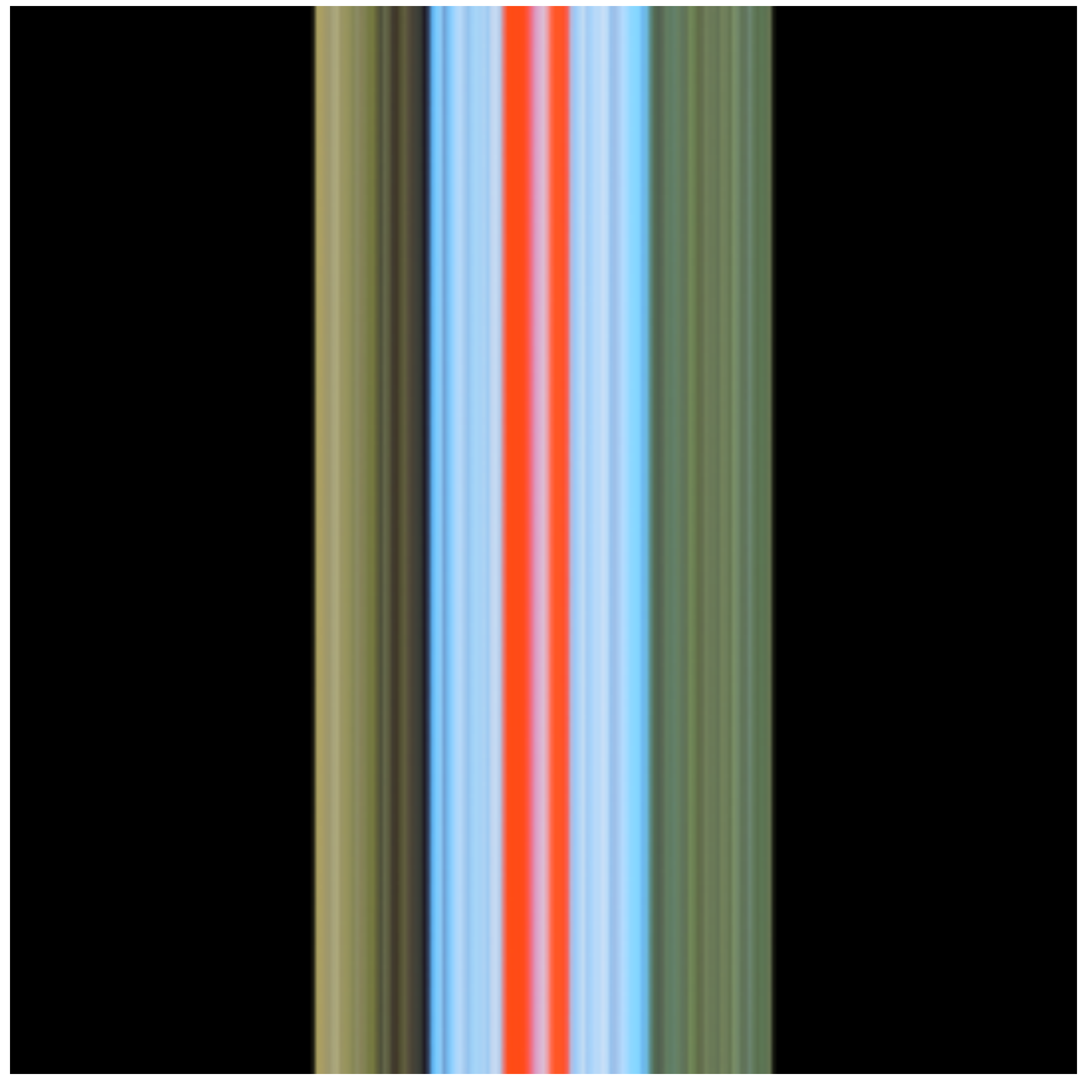
Applying shutter blur across time $t \in [0, 1)$

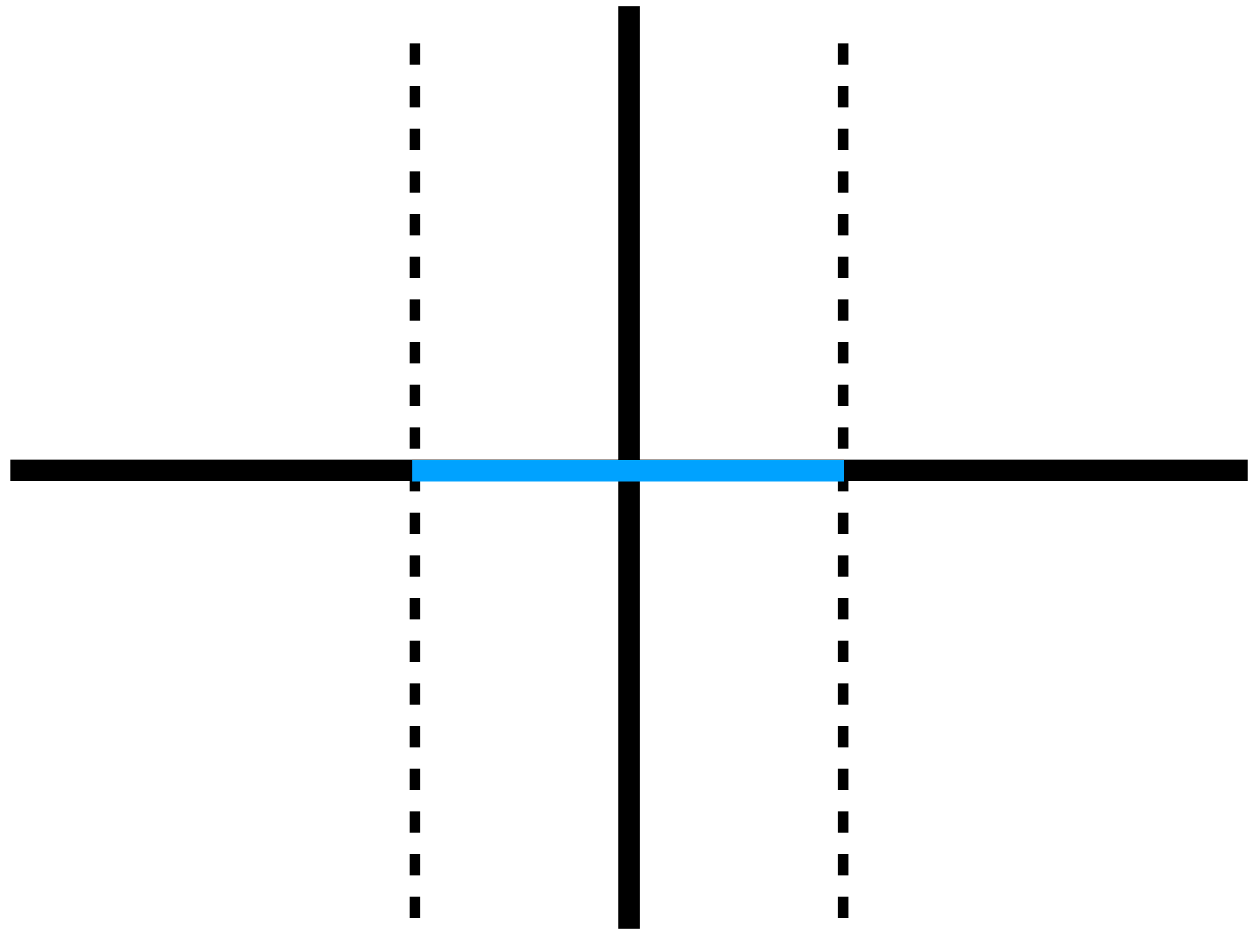




Basic example: Fourier domain

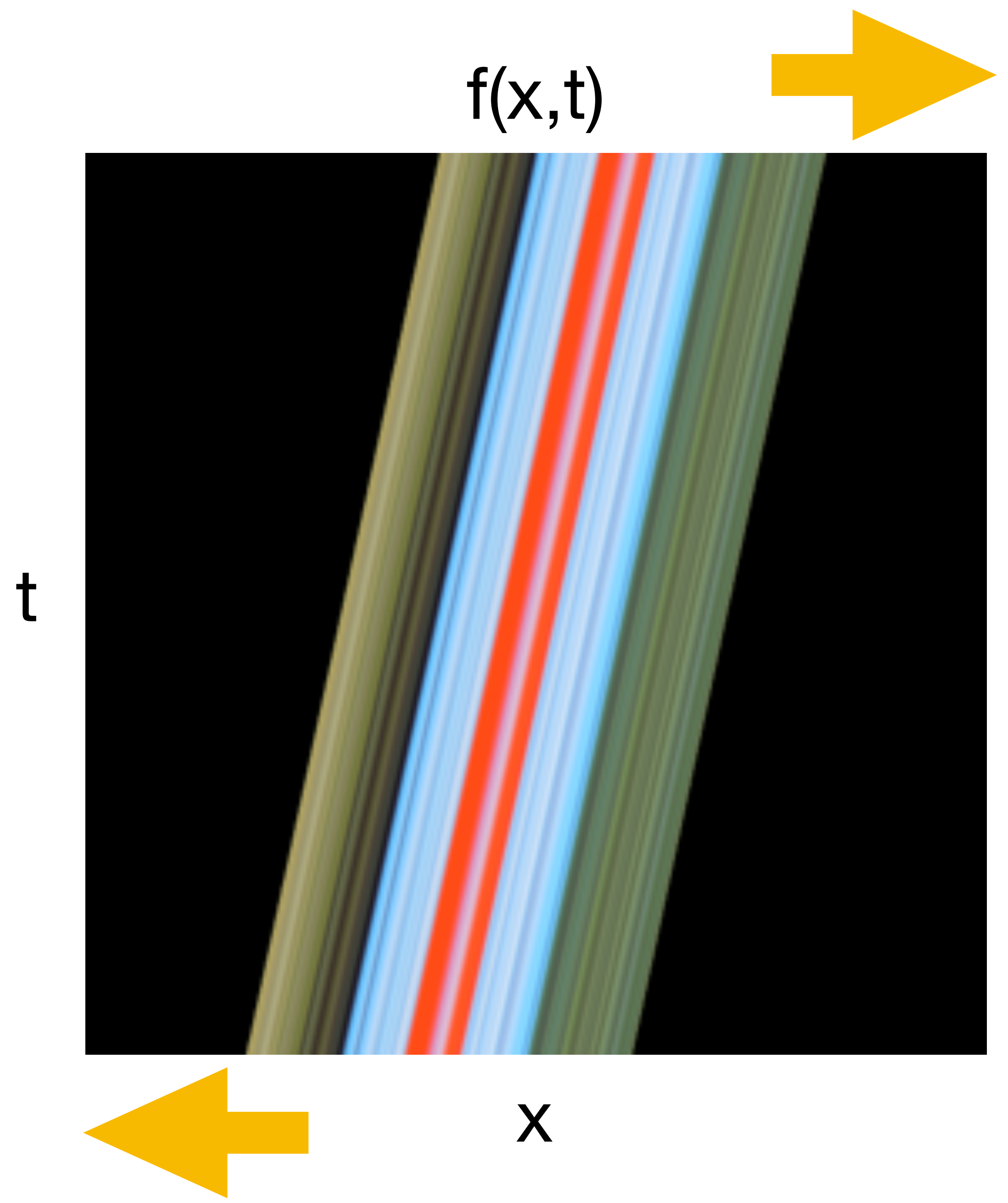
Fourier spectrum, zero velocity $t \in [0, 1)$



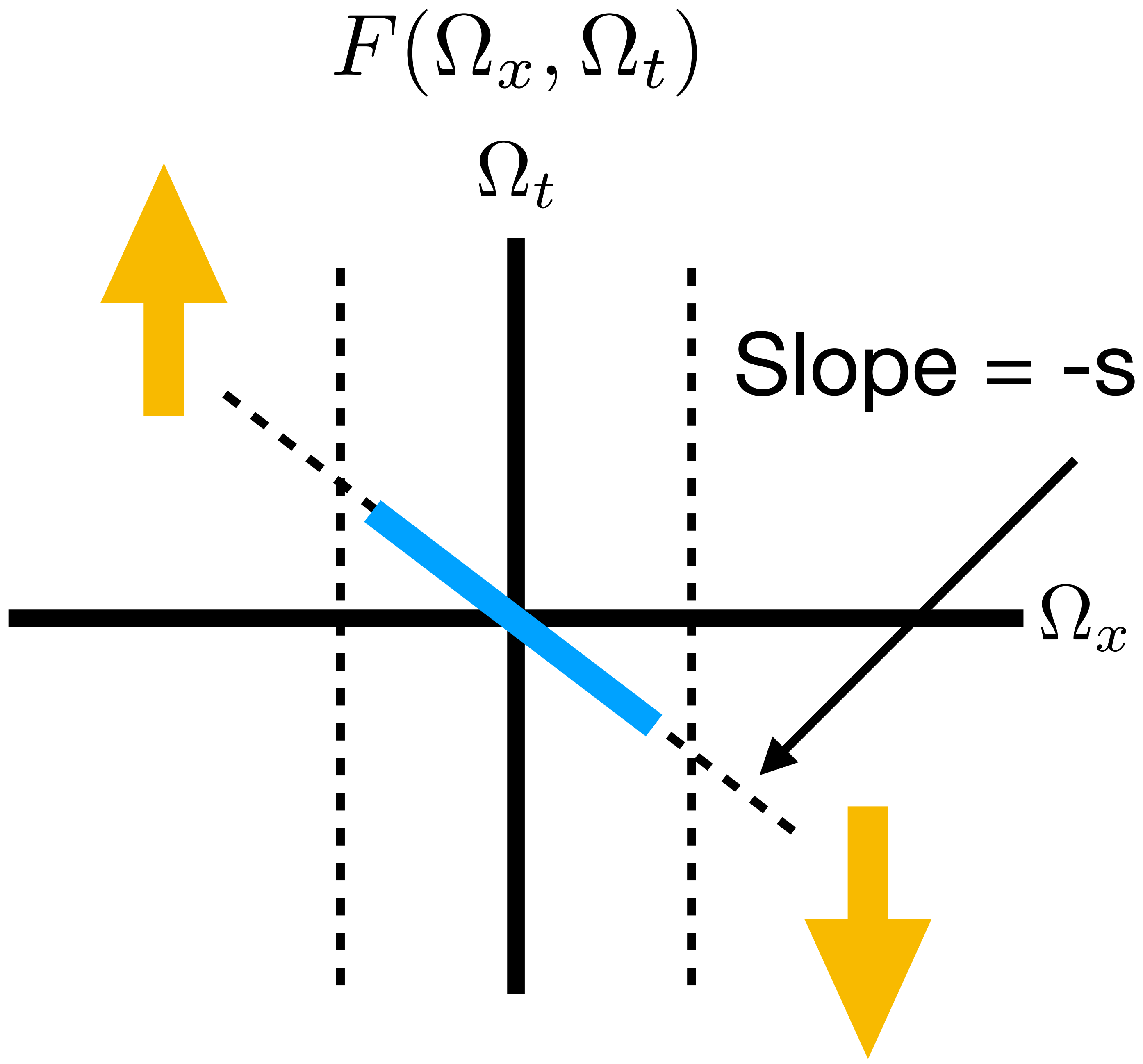


Basic example: Fourier domain

Low velocity, small shear in both domains



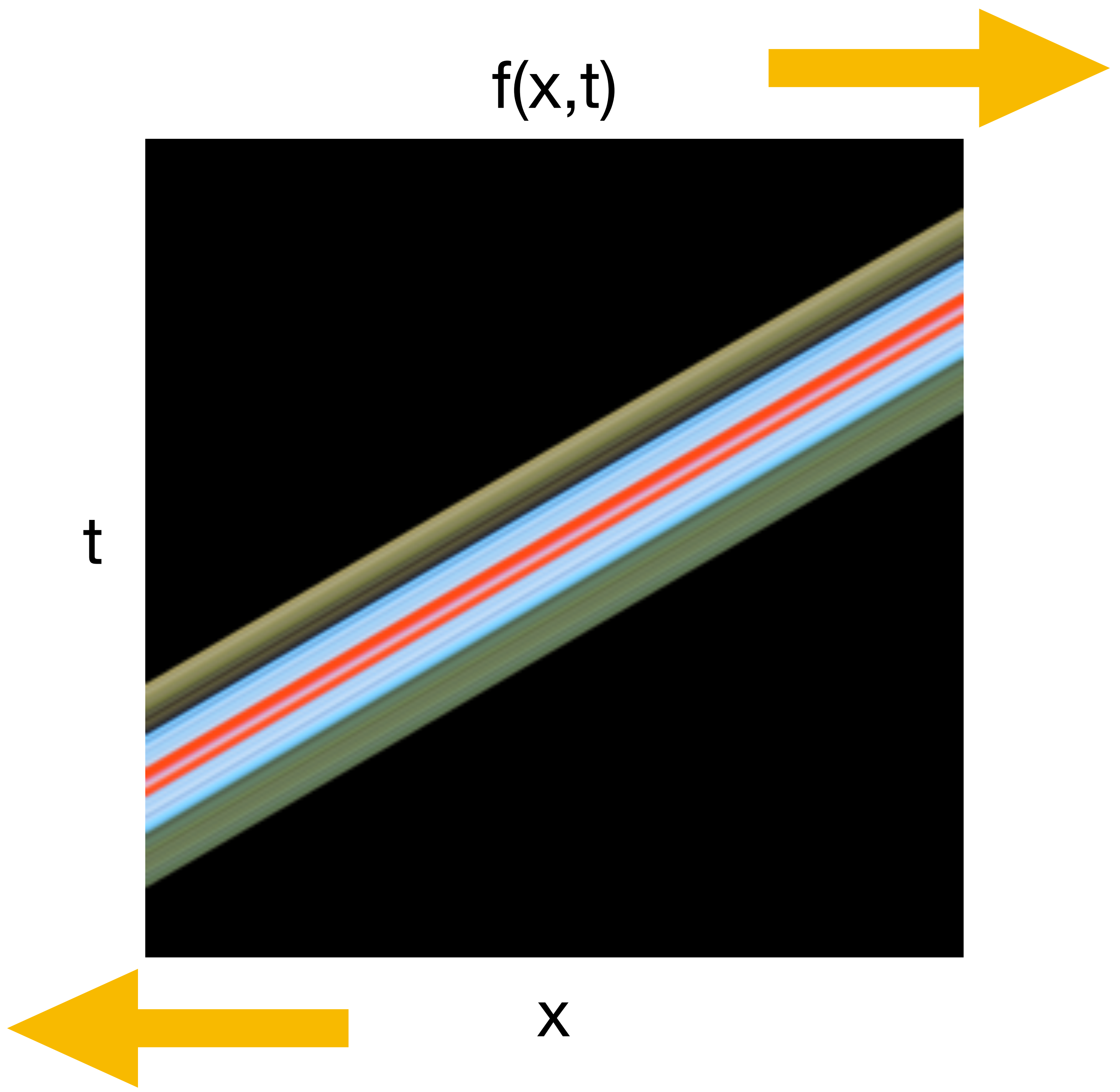
$t \in [0, 1)$

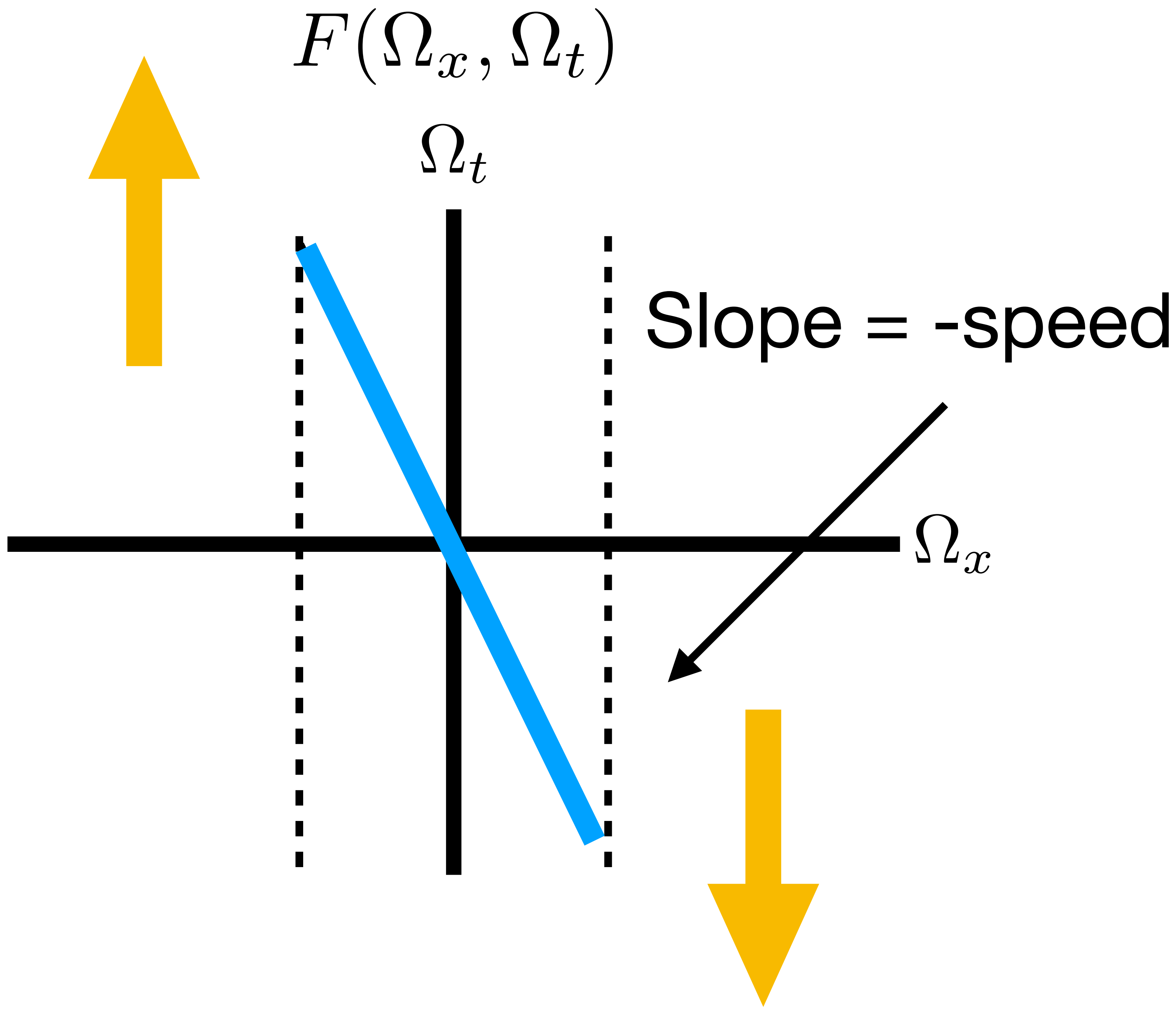


Basic example: Fourier domain

Low velocity, small shear in both domains

$t \in [0, 1)$



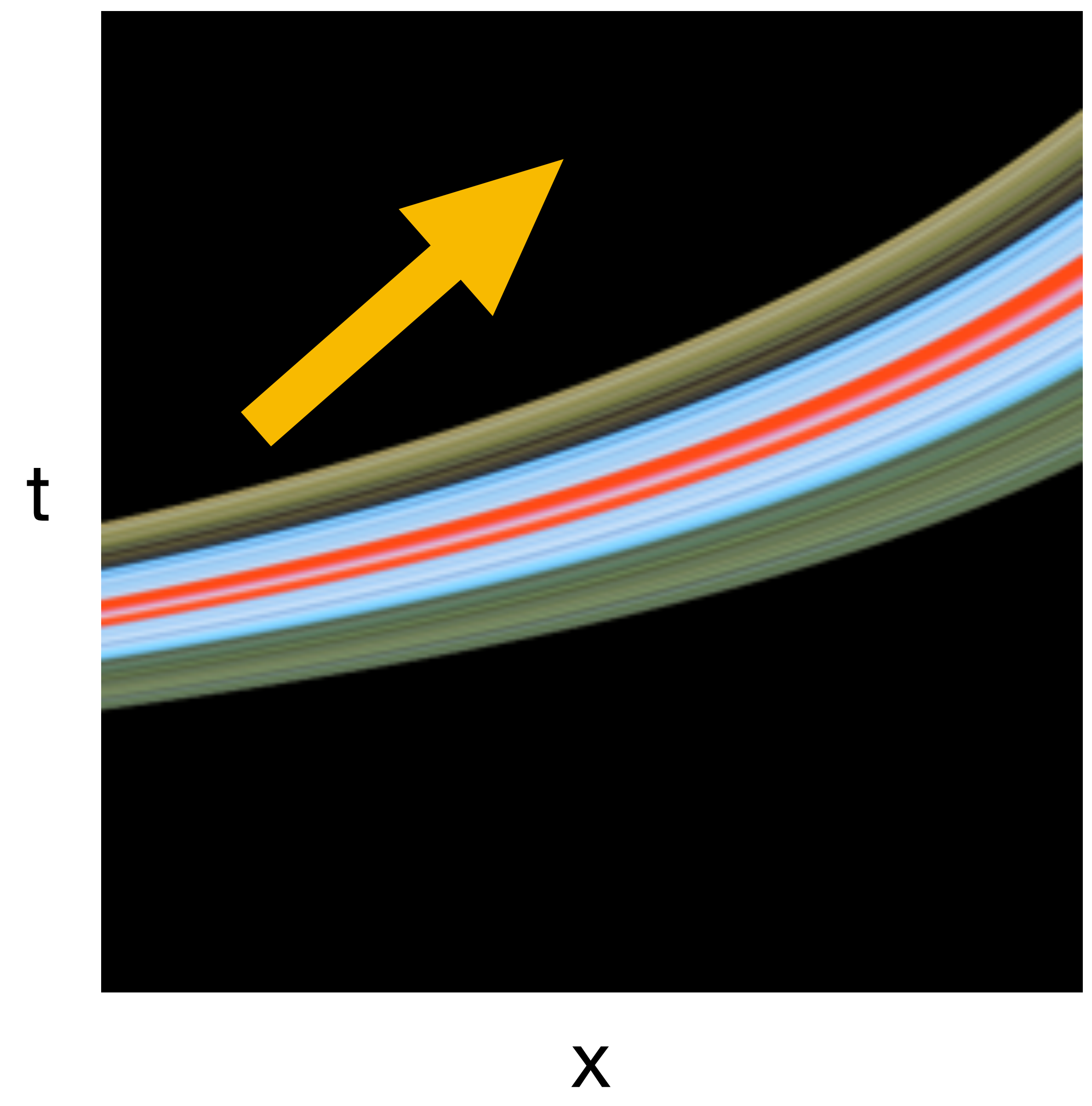


Basic example: Fourier domain

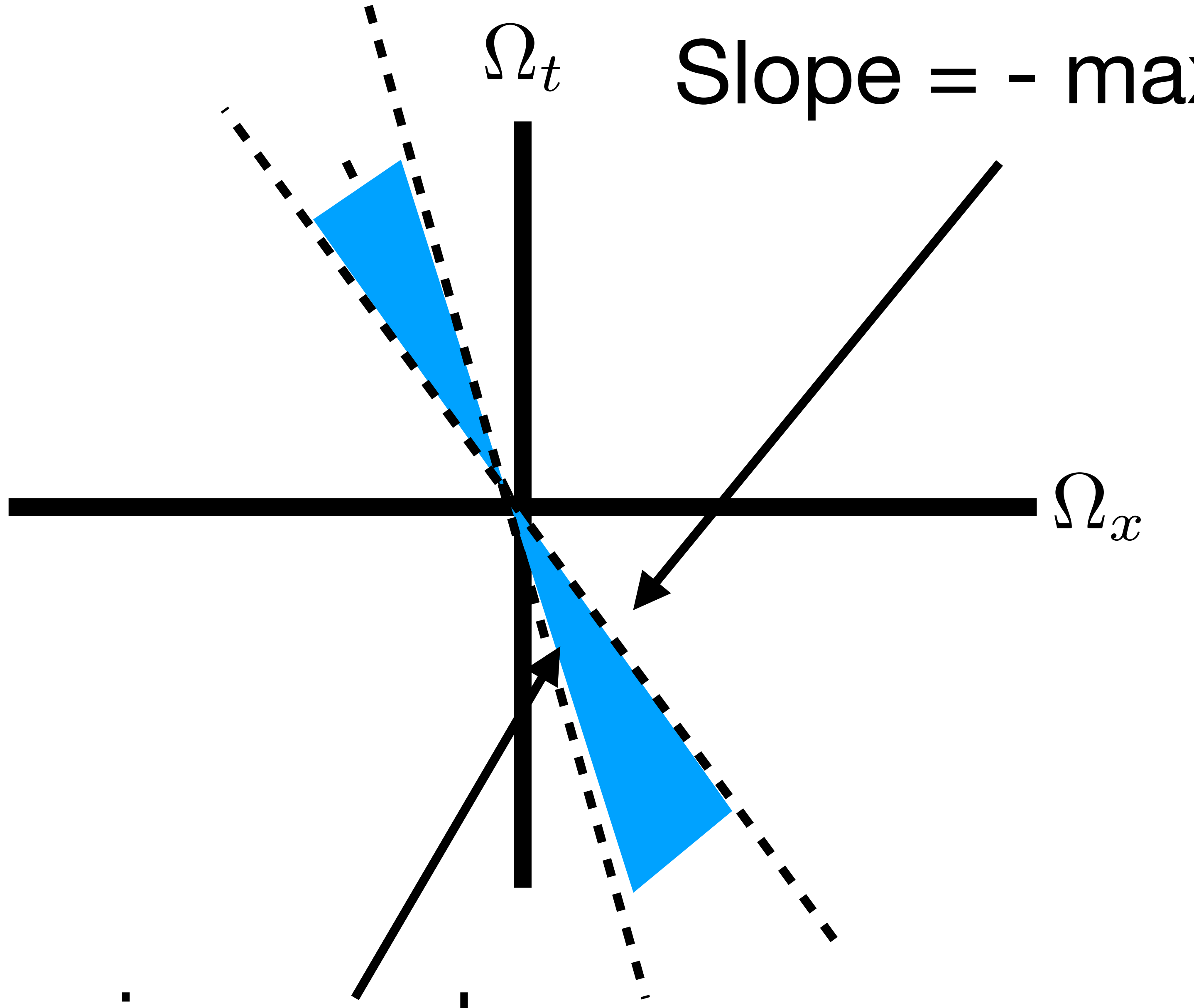
Due to camera motion, slopes are varying

$t \in [0, 1)$

f(x,t)



$F(\Omega_x, \Omega_t)$



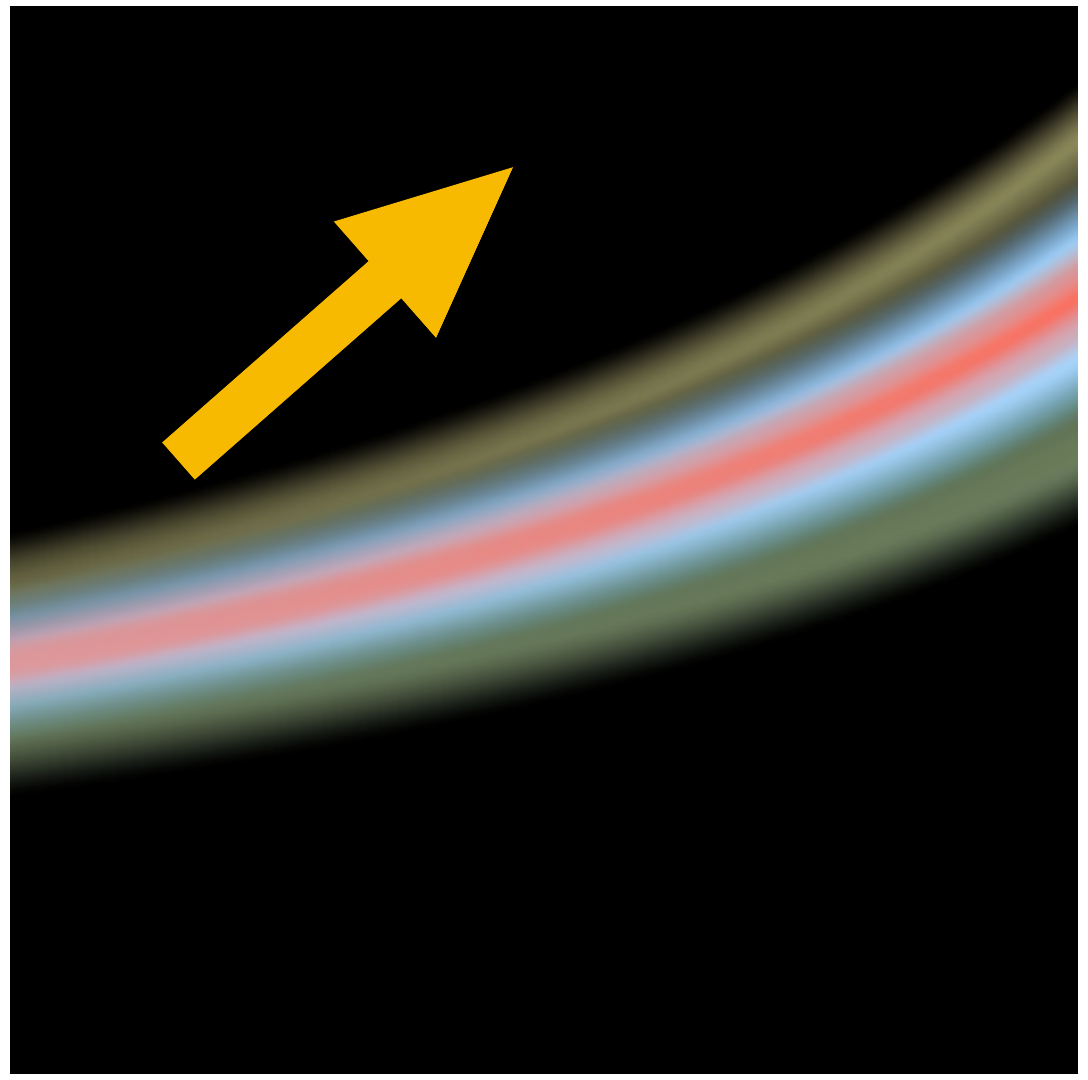
Basic example: Fourier domain

When shutter blur is applied, only low frequencies matter

$t \in [0,1)$

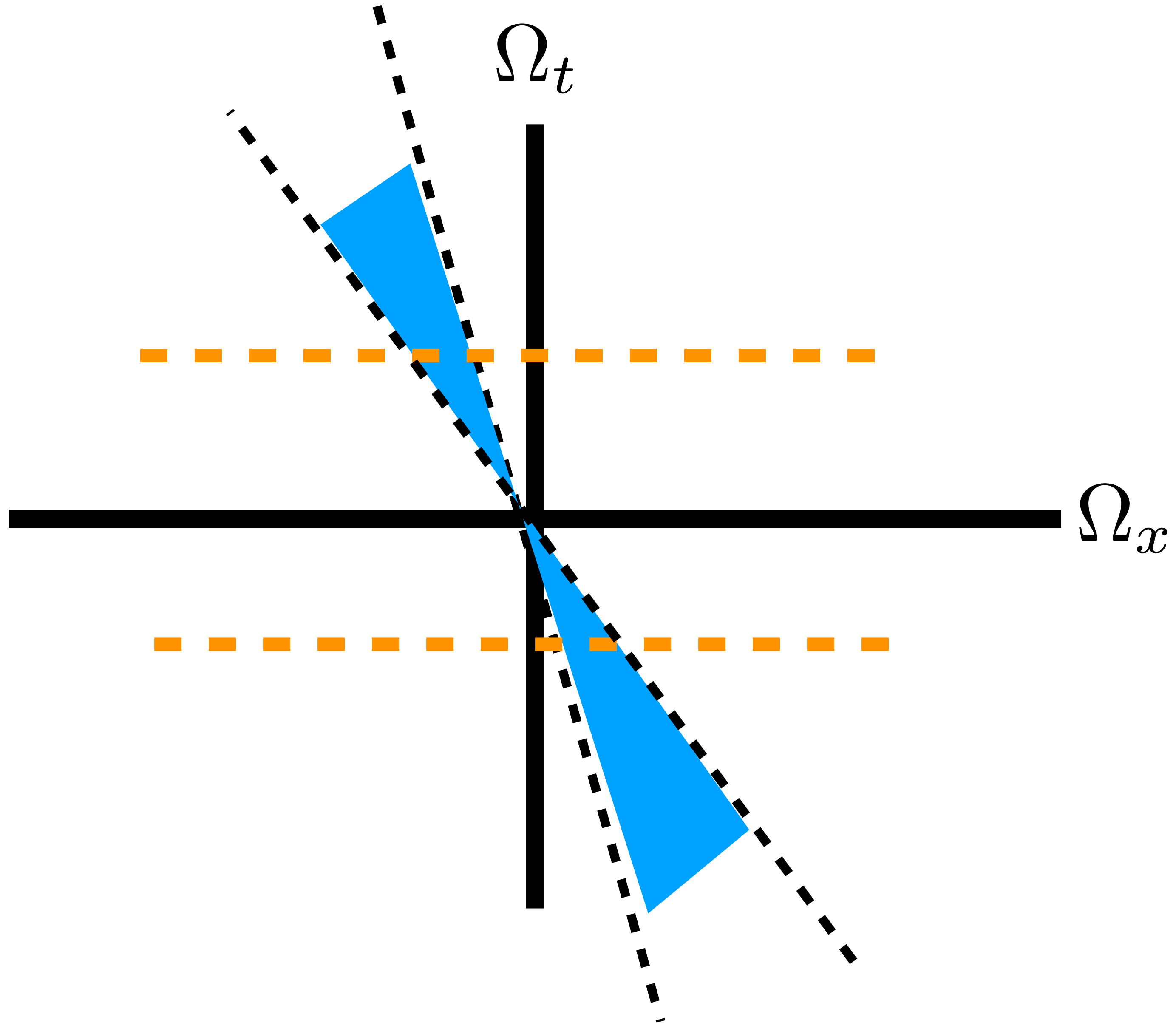
f(x,t)

t



x

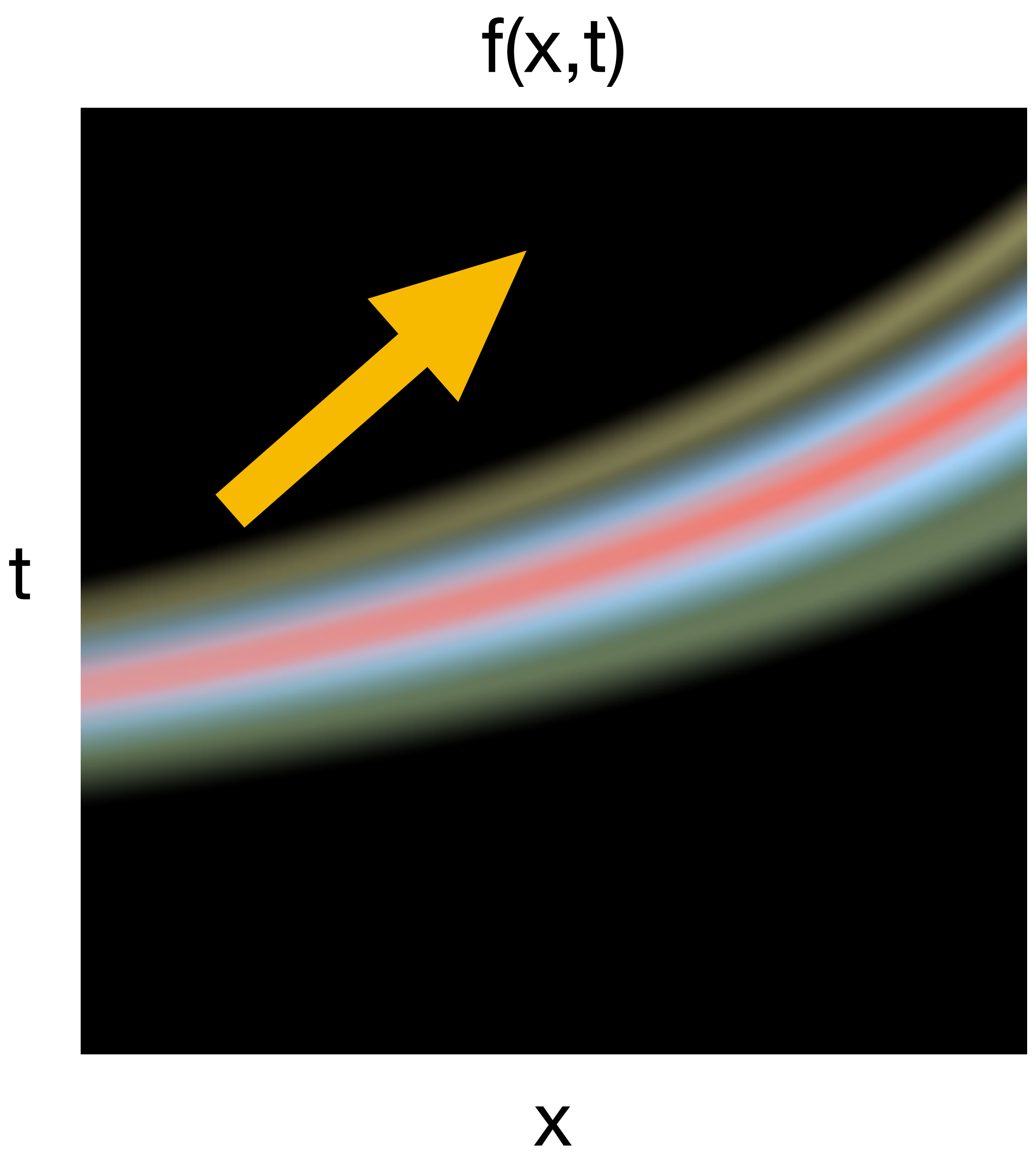
$F(\Omega_x, \Omega_t)$

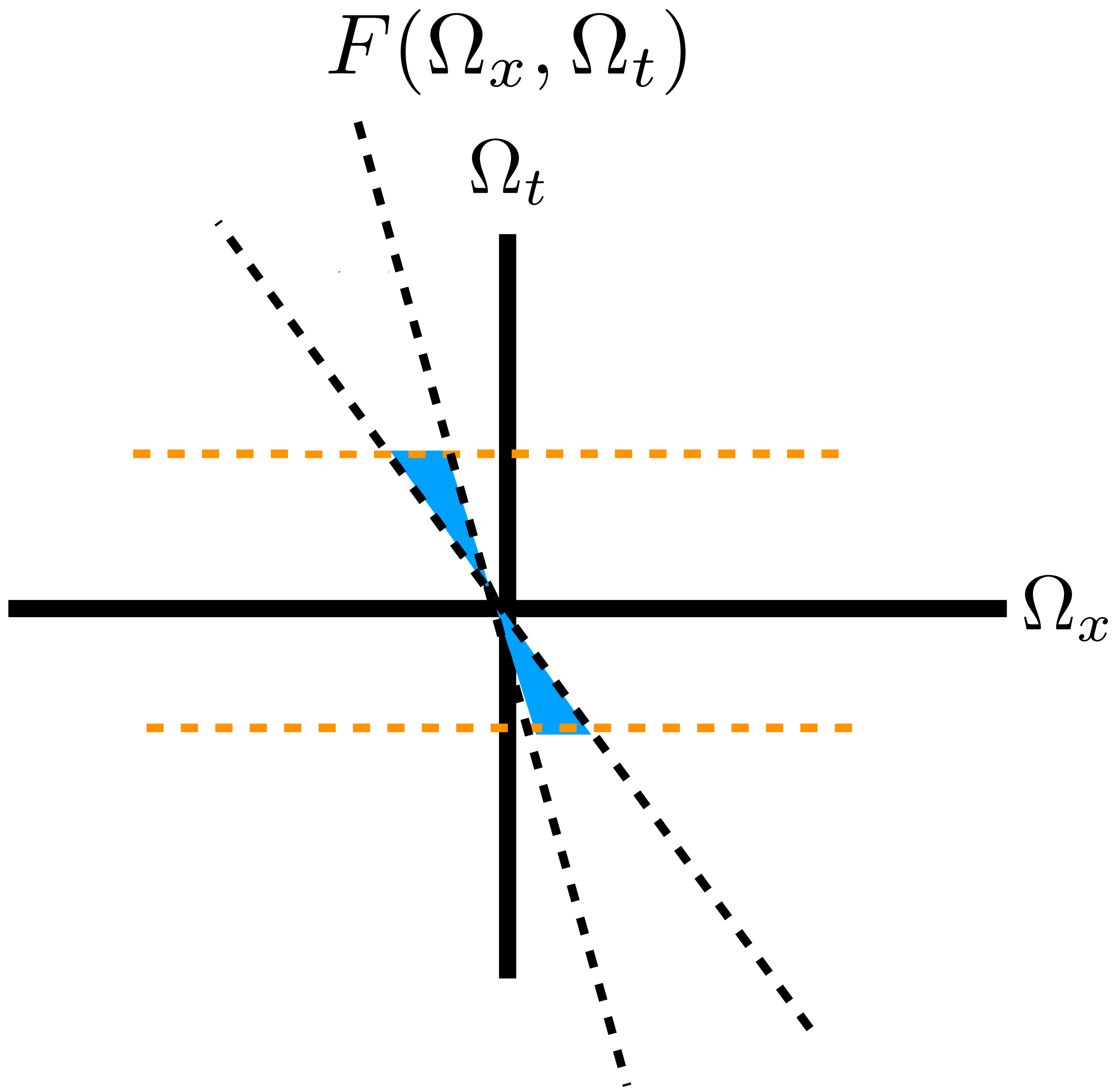


Basic example: Fourier domain

When shutter blur is applied, only low frequencies matter

$t \in [0, 1)$

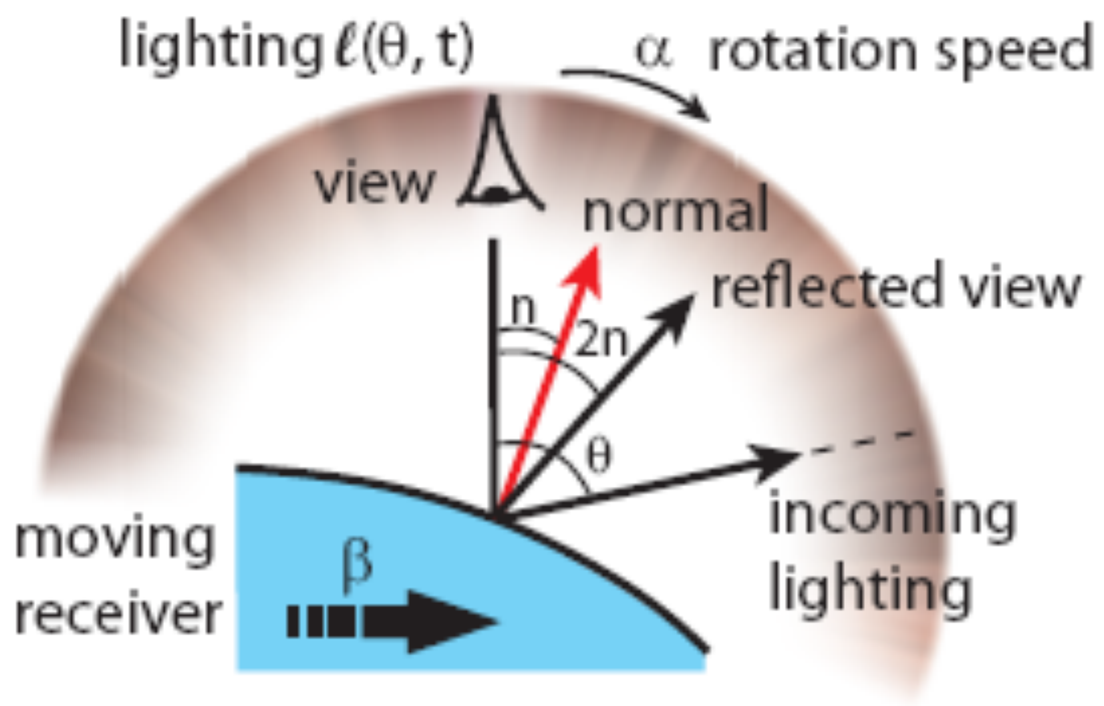


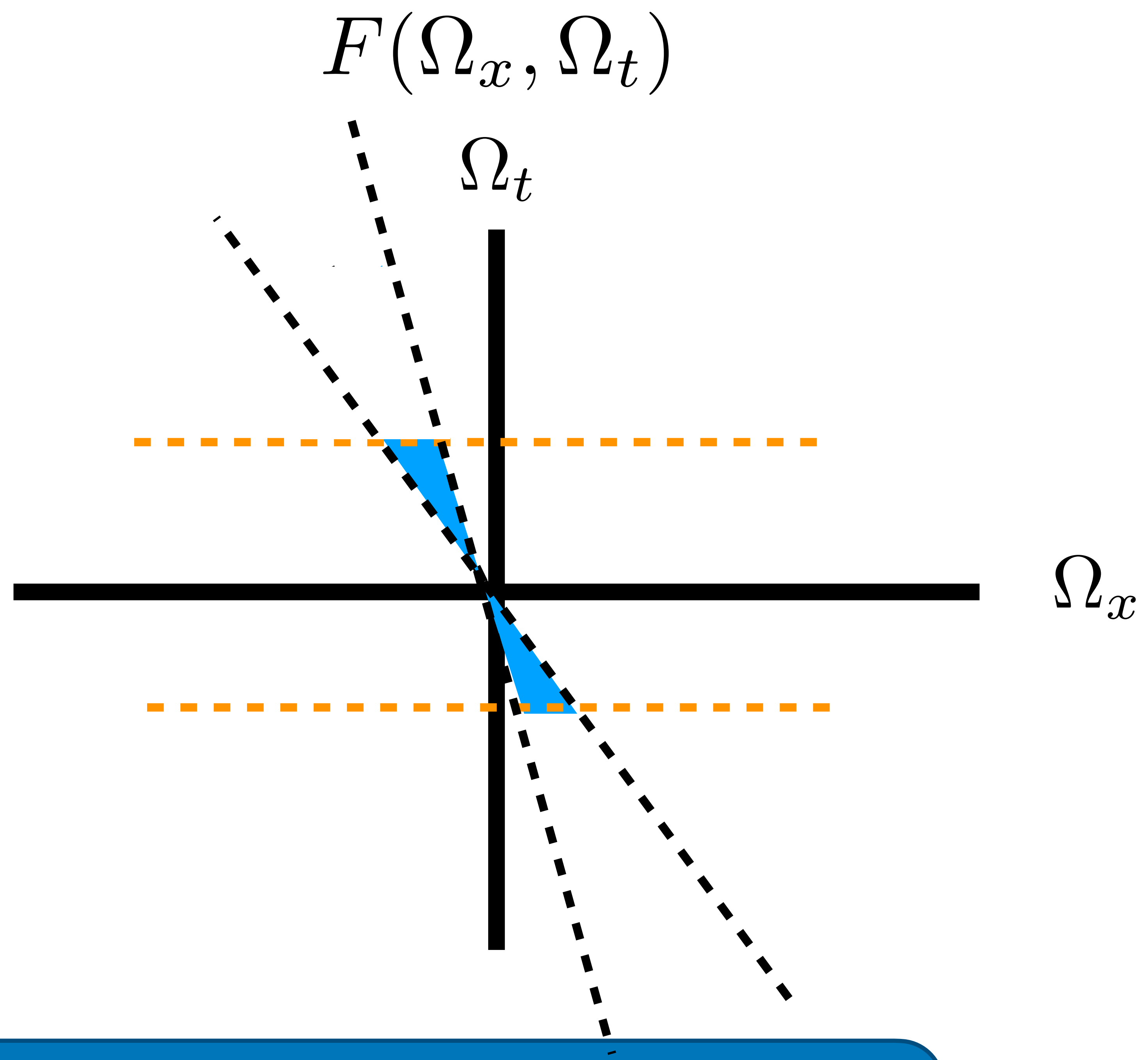


Main Insights

Common case = double wedge spectra

Shutter indirectly removes spatial frequencies





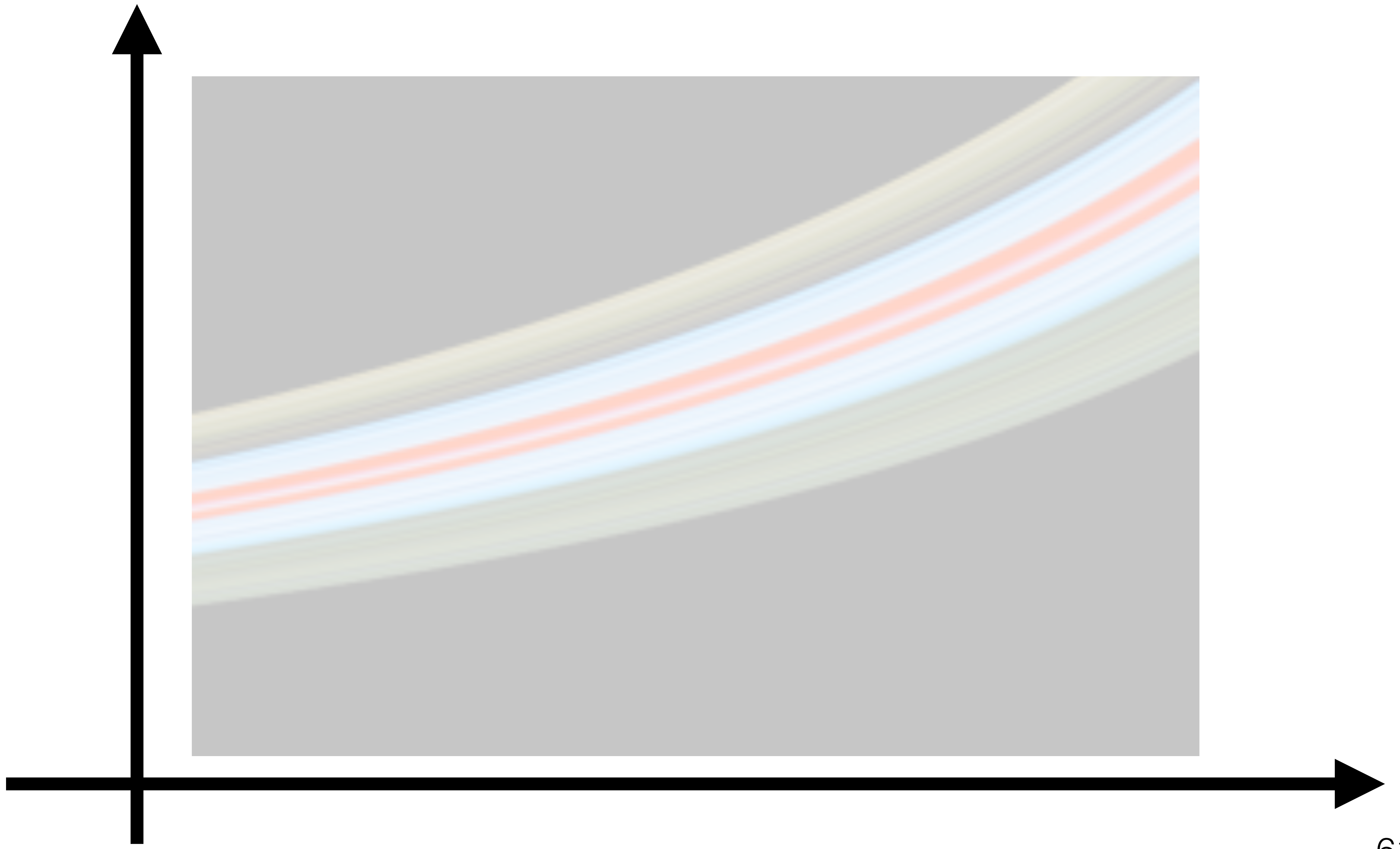
Sampling and Filtering Goals

Minimal sampling rate to prevent aliasing

Derive shape of the reconstruction filters

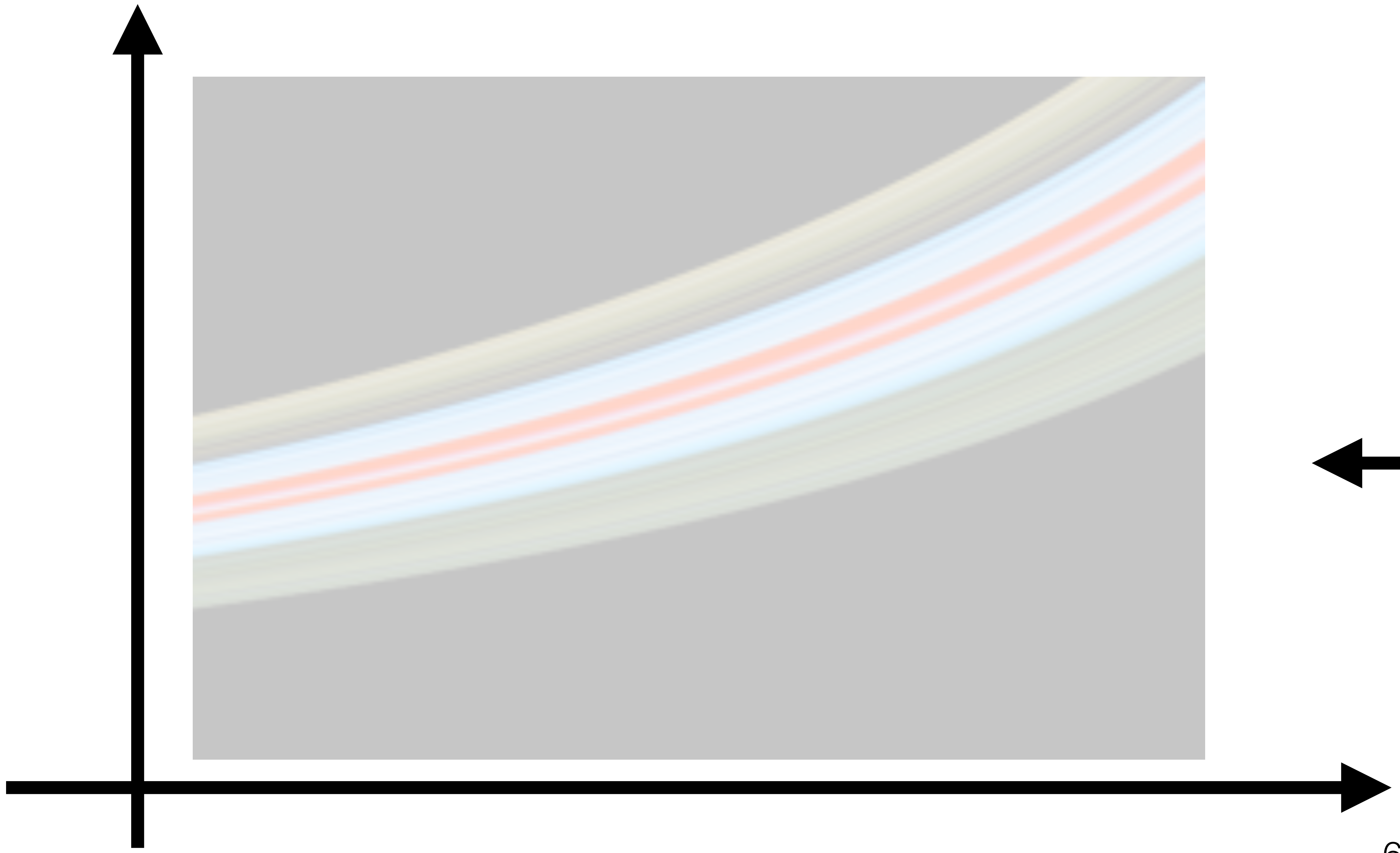
Sampling in Fourier Domain

Sampling produces replicas in the Fourier domain

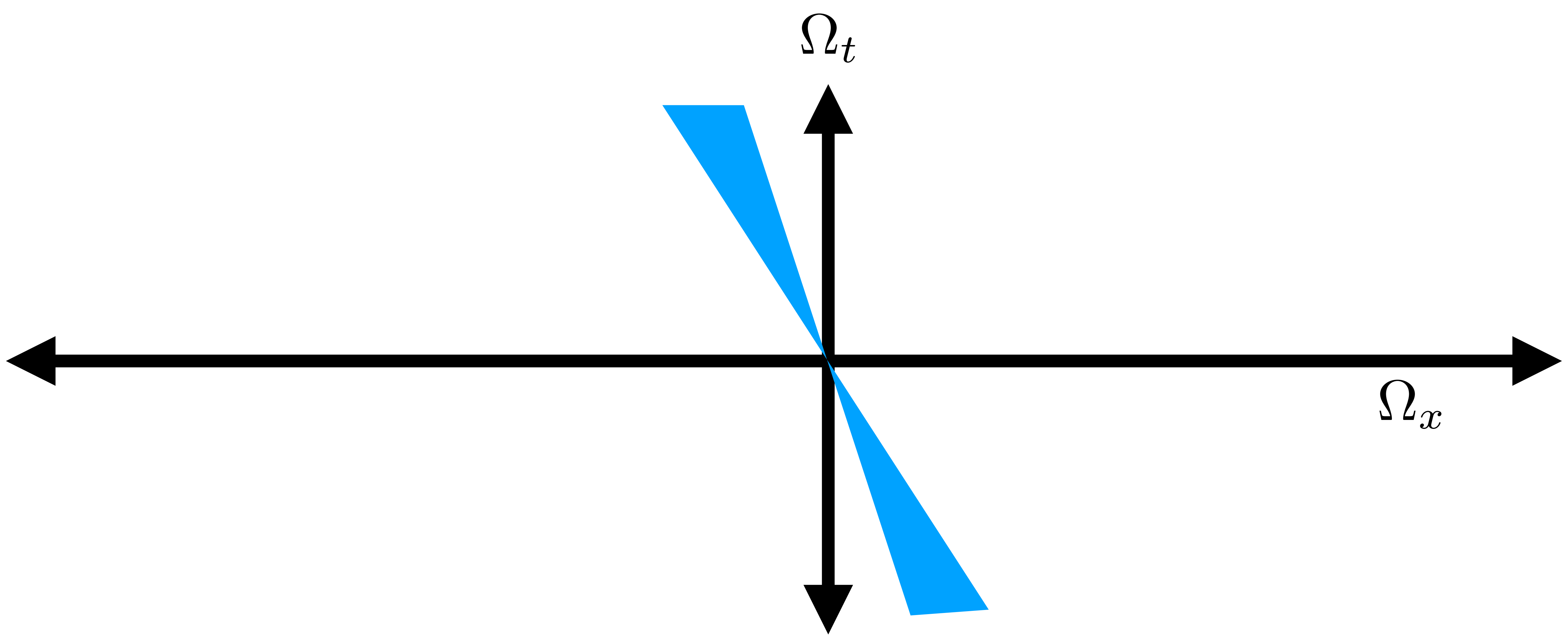


Sampling in Fourier Domain

Let's say the corresponding image has a Fourier spectrum as shown on the right side

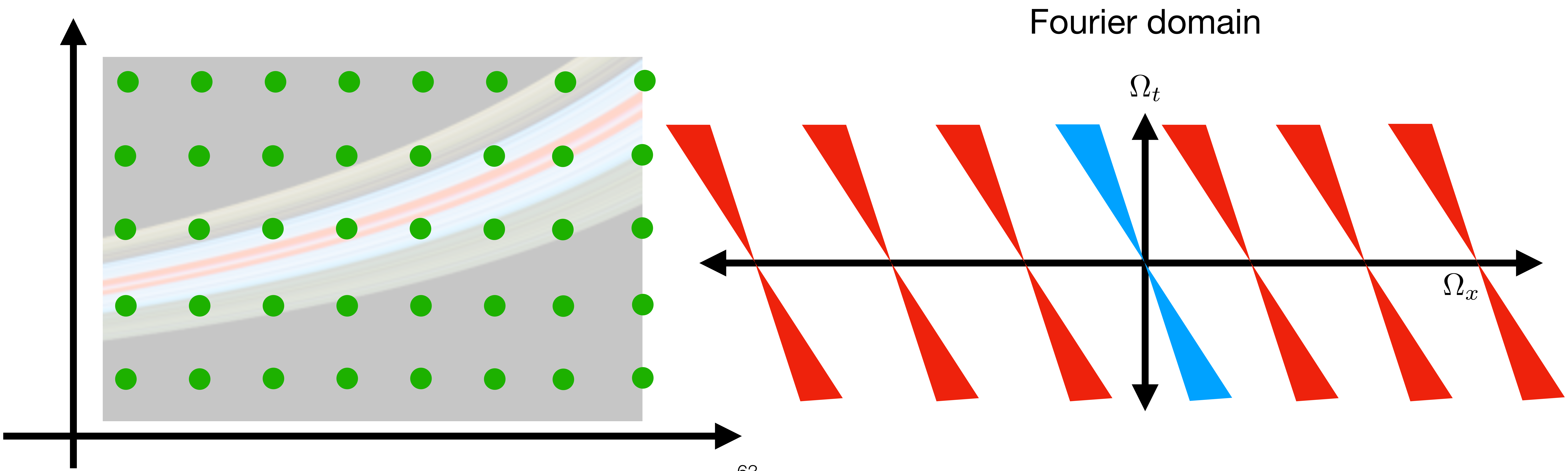


Fourier domain



Sampling in Fourier Domain

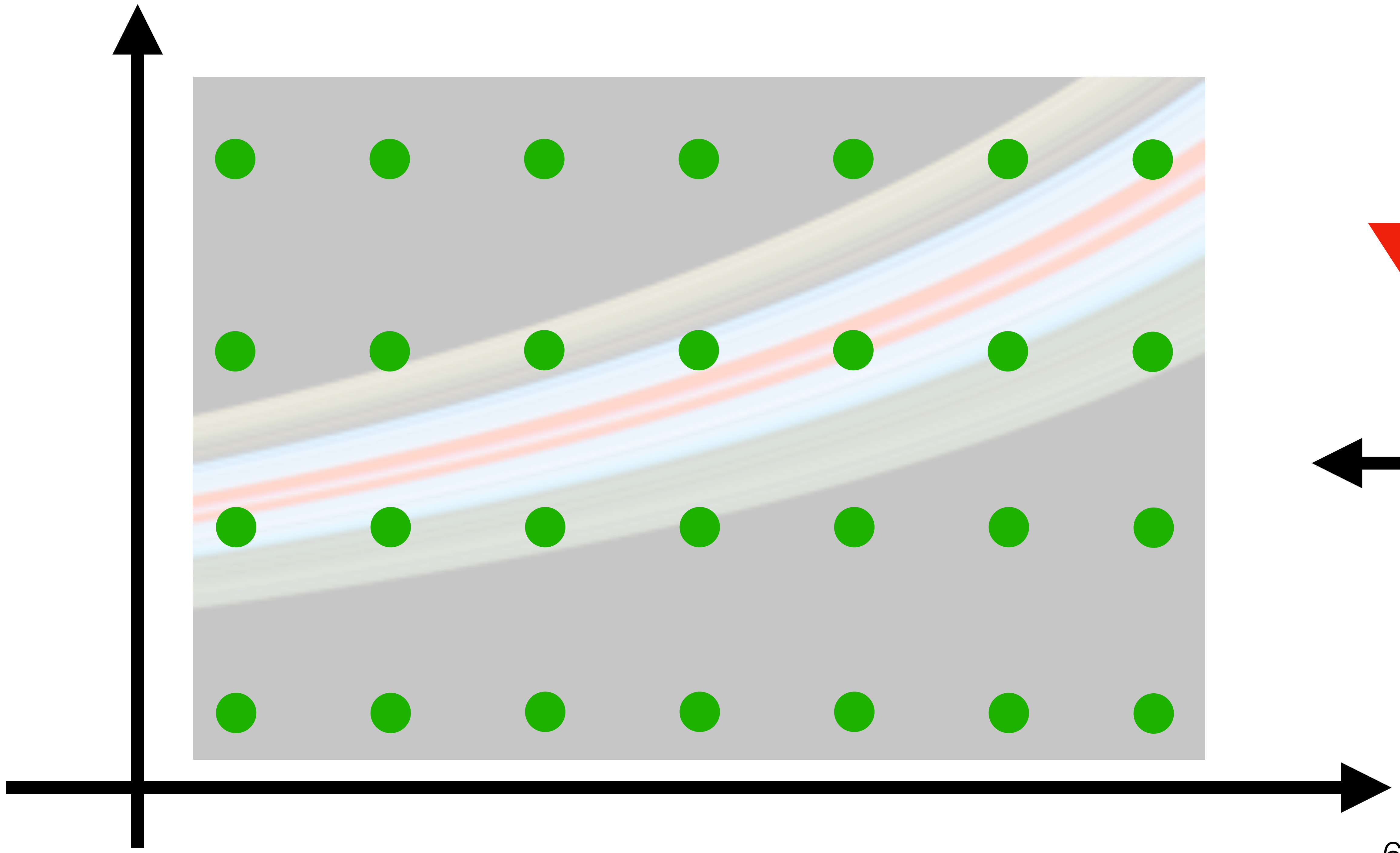
Sampling produces replicas in the Fourier domain



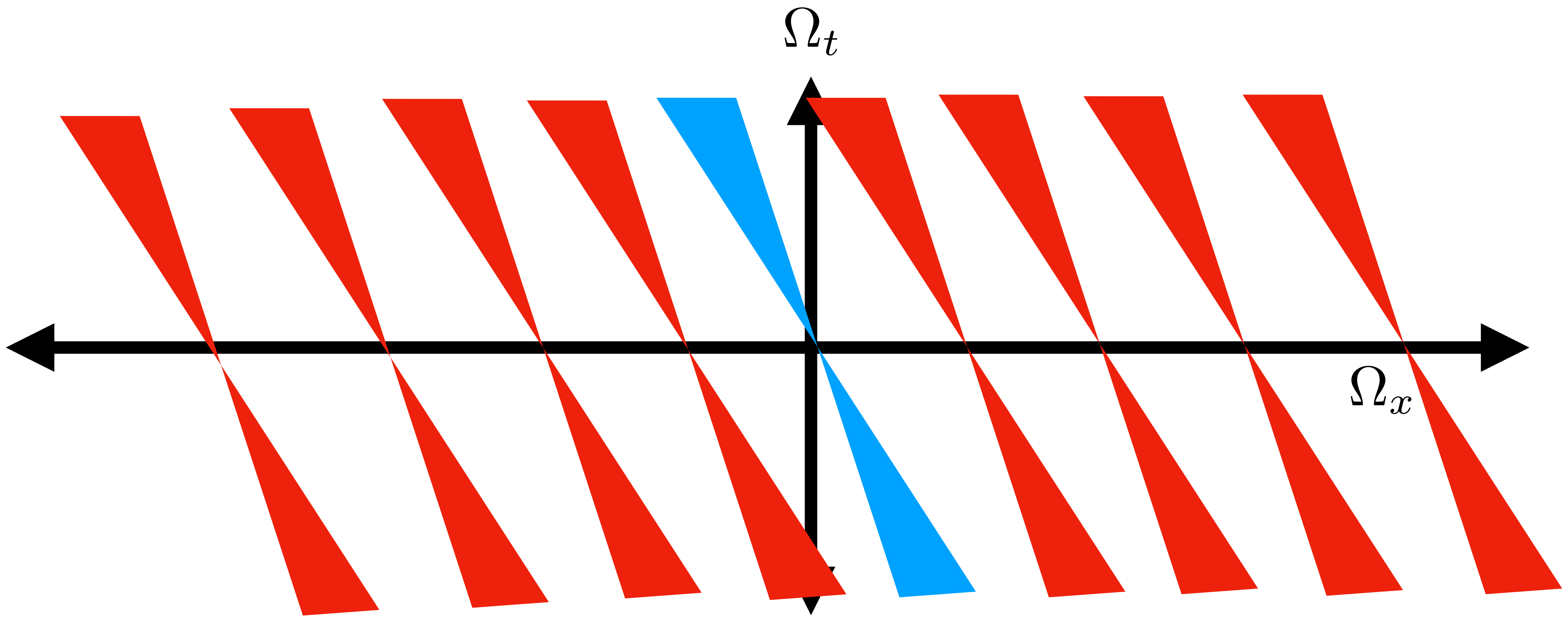
Sampling in Fourier Domain

Sampling produces replicas in the Fourier domain

Sparse sampling produces denser replicas

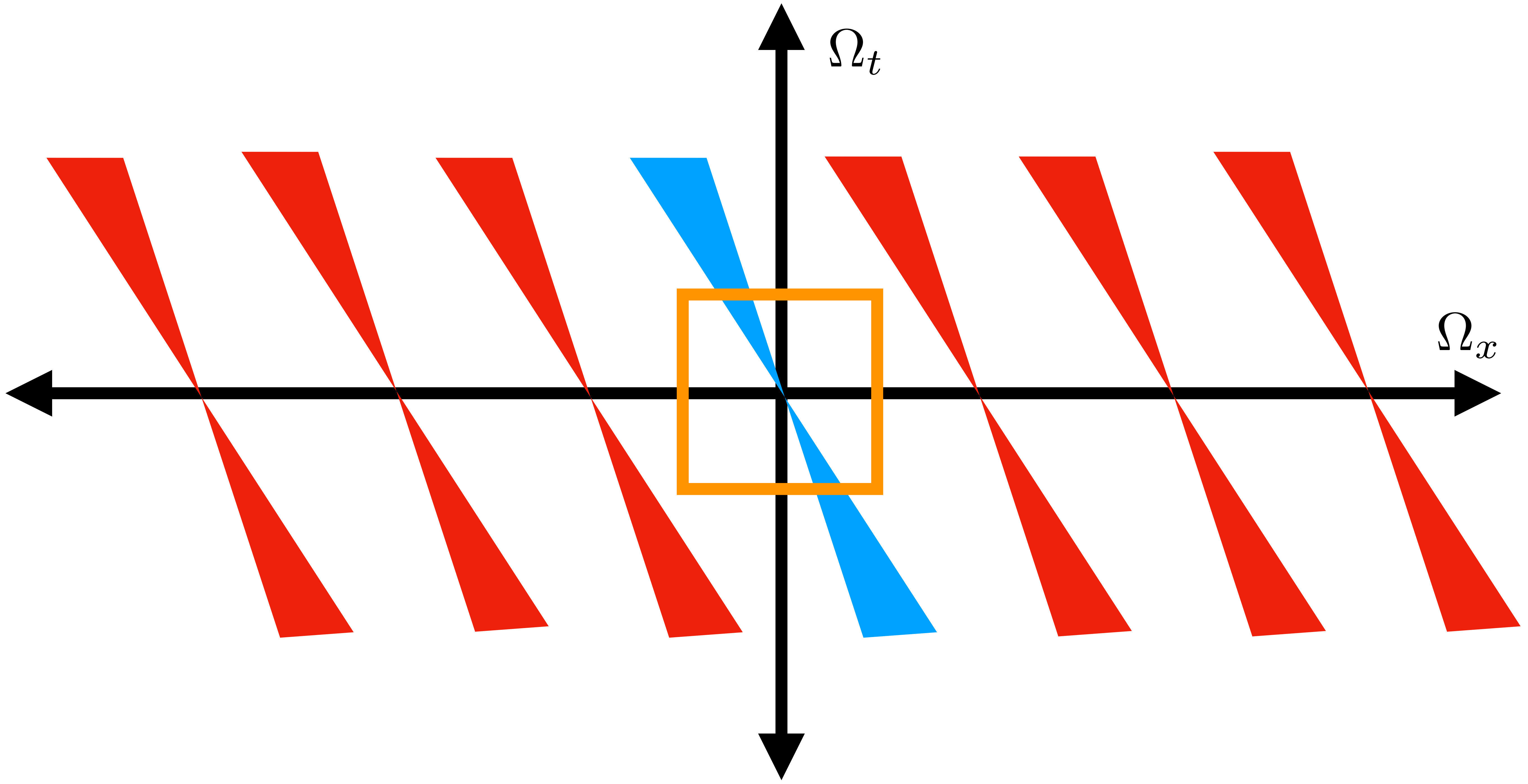


Fourier domain



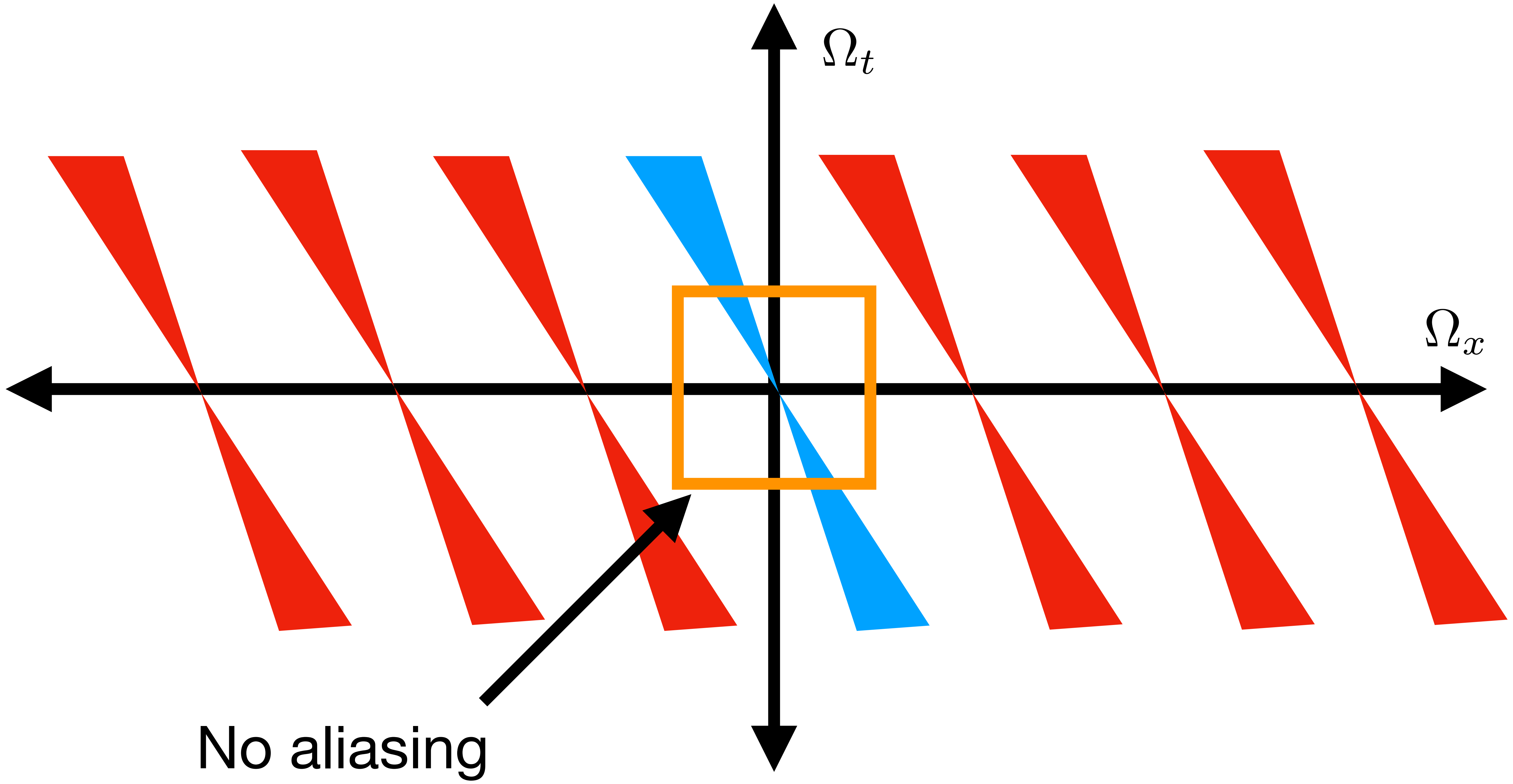
Standard Reconstruction Filtering

Standard filer, dense sampling, slow



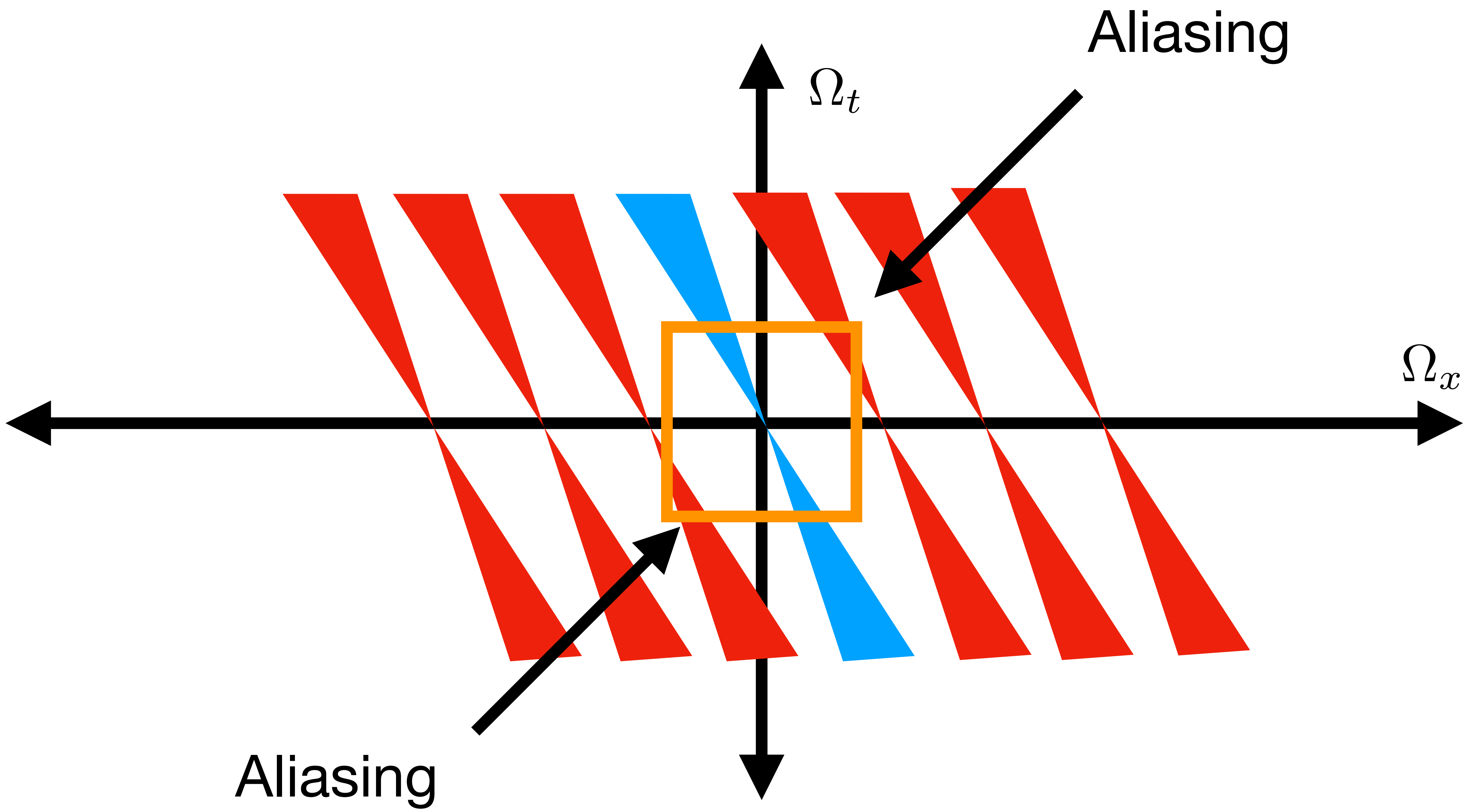
Standard Reconstruction Filtering

Standard filer, dense sampling, slow



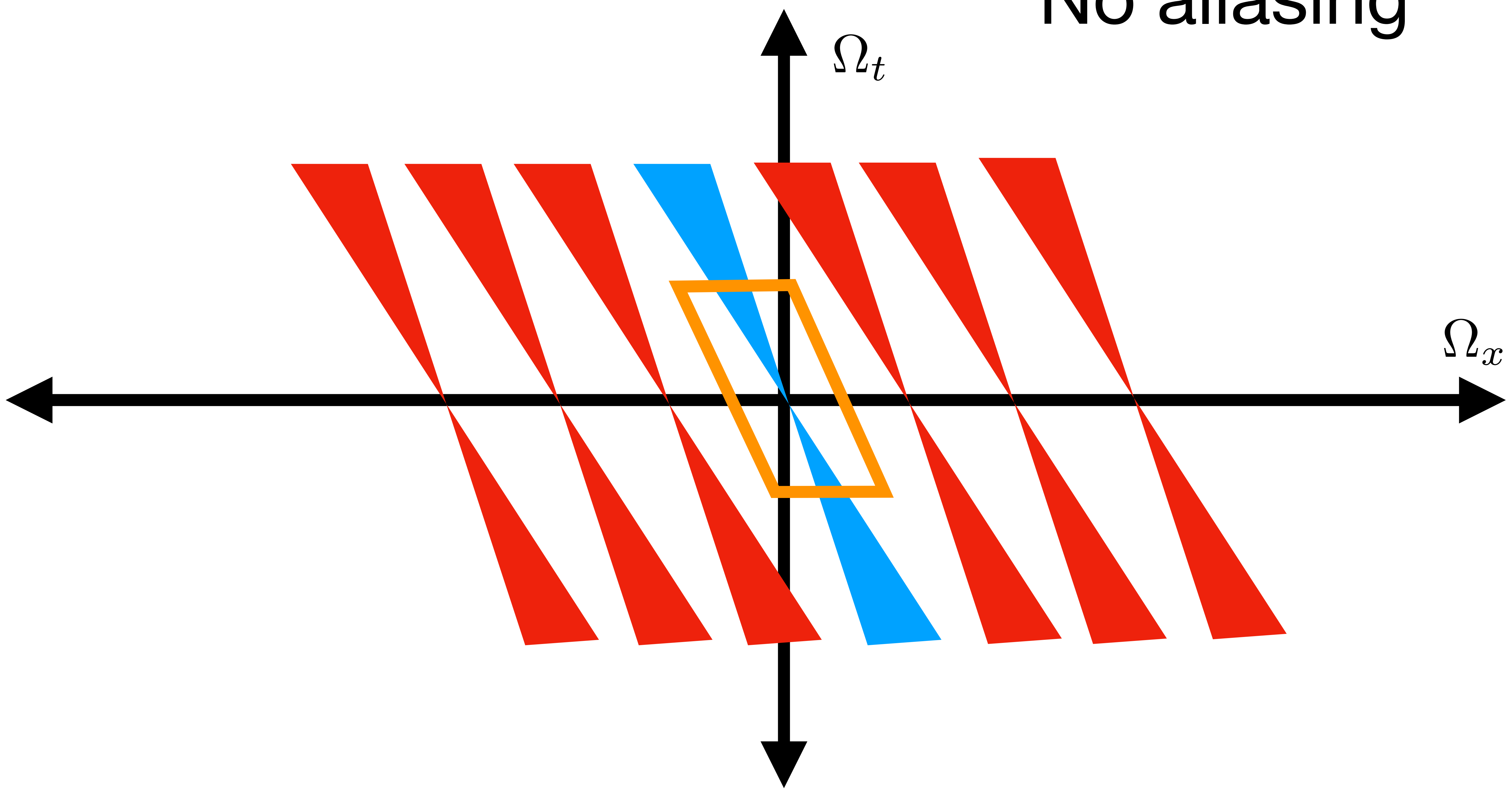
Standard Reconstruction Filtering

Standard filer, sparse sampling, fast



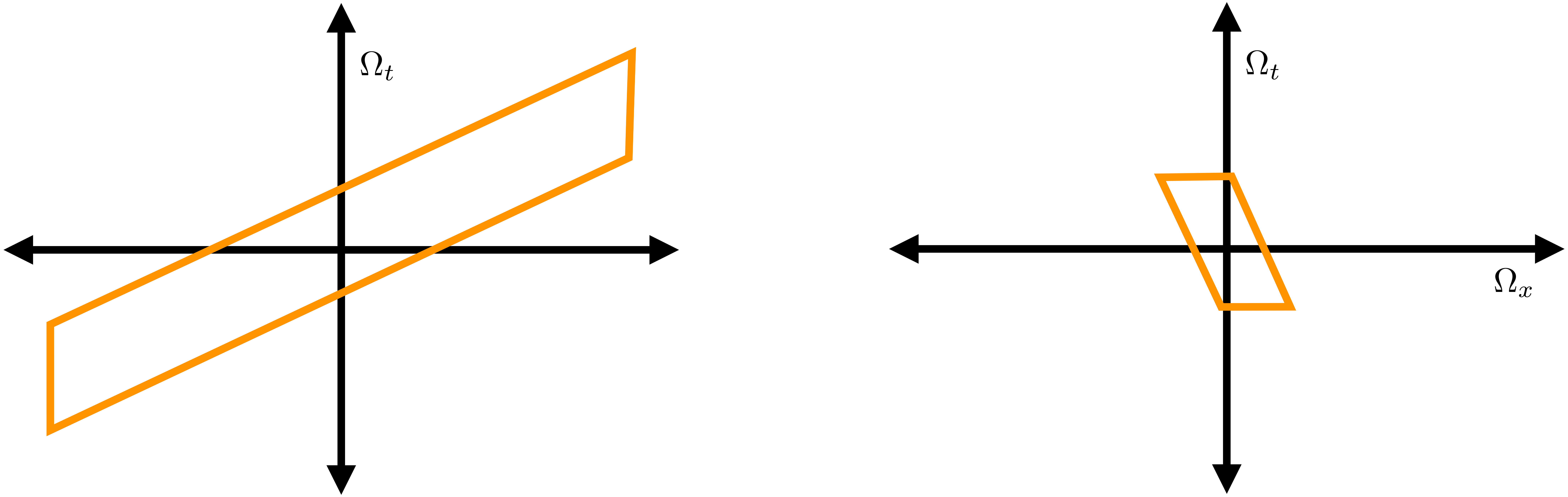
Sheared Reconstruction Filter

Standard filer, sparse sampling, fast



Sheared Reconstruction Filter

Compact shape in Fourier = wide space-time



Main Insights

Sheared filter allows for many fewer samples

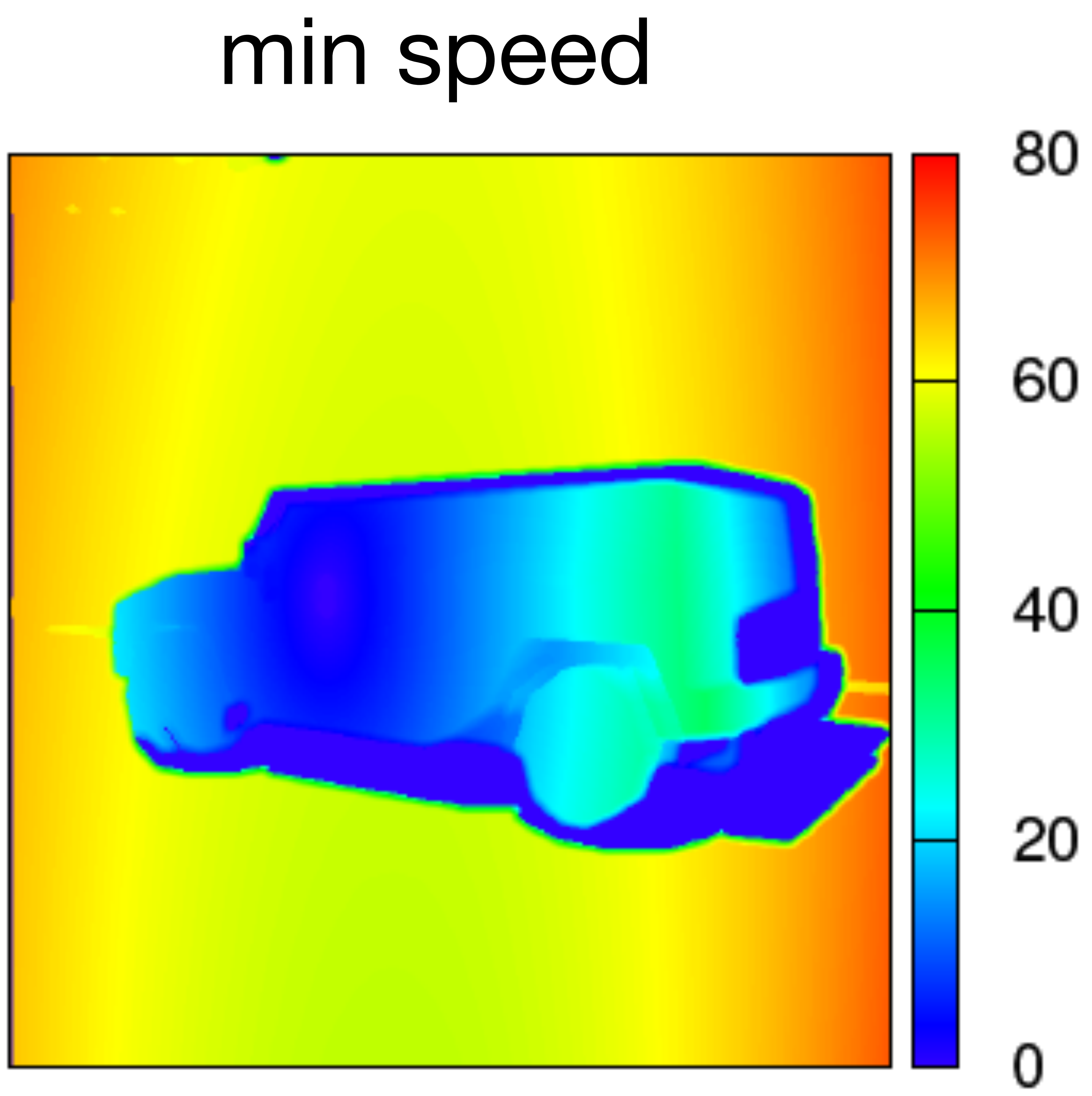
Filters in action: Car example

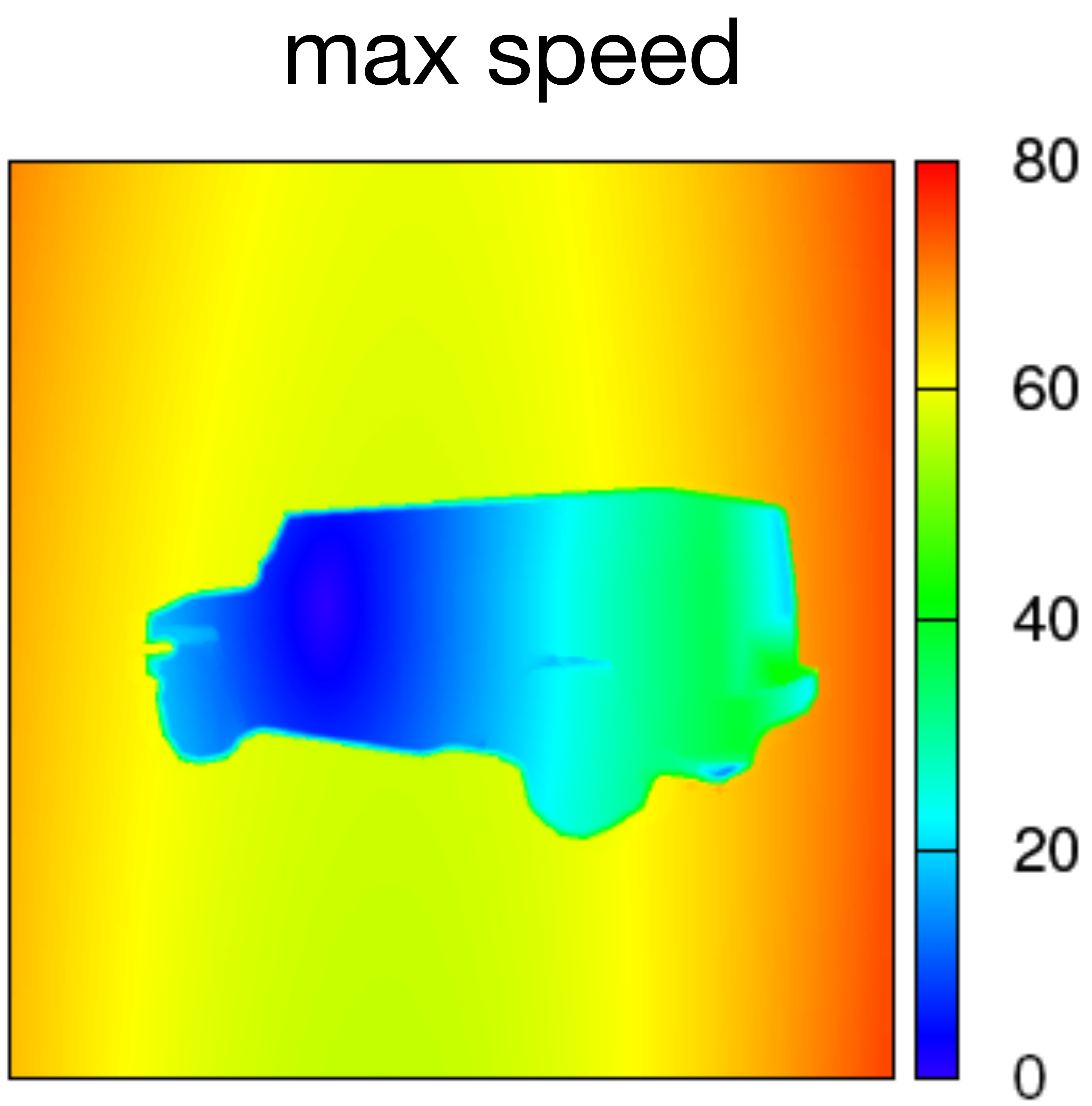
Static

Motion blurred

Implementation: stage 1

Sparse sampling to compute velocity bounds

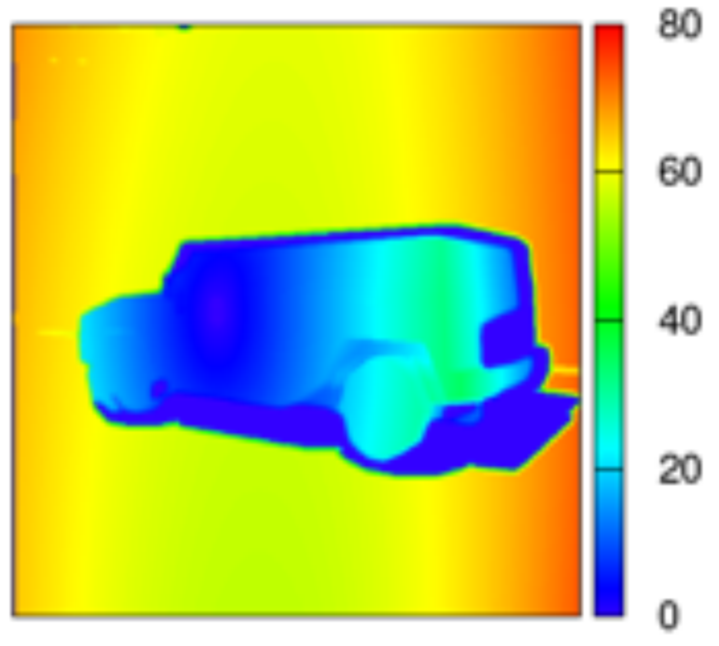




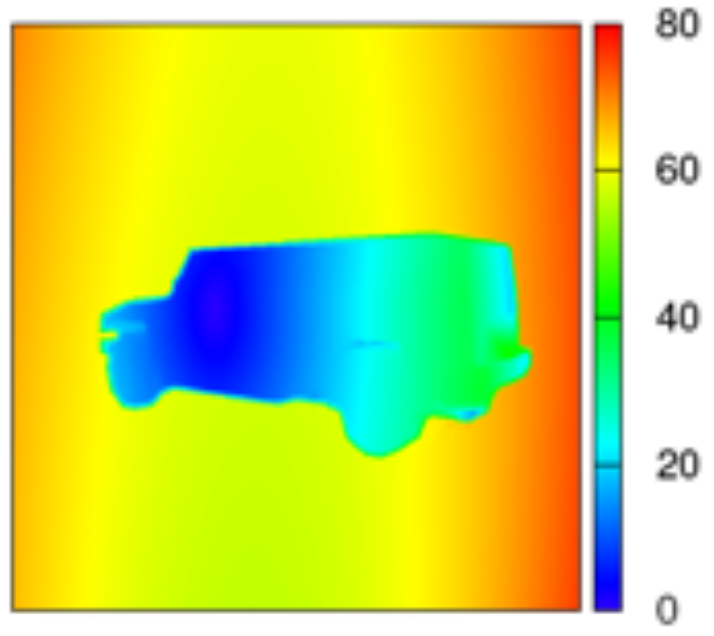
Implementation: stage 1

Calculate filter widths and sampling rates

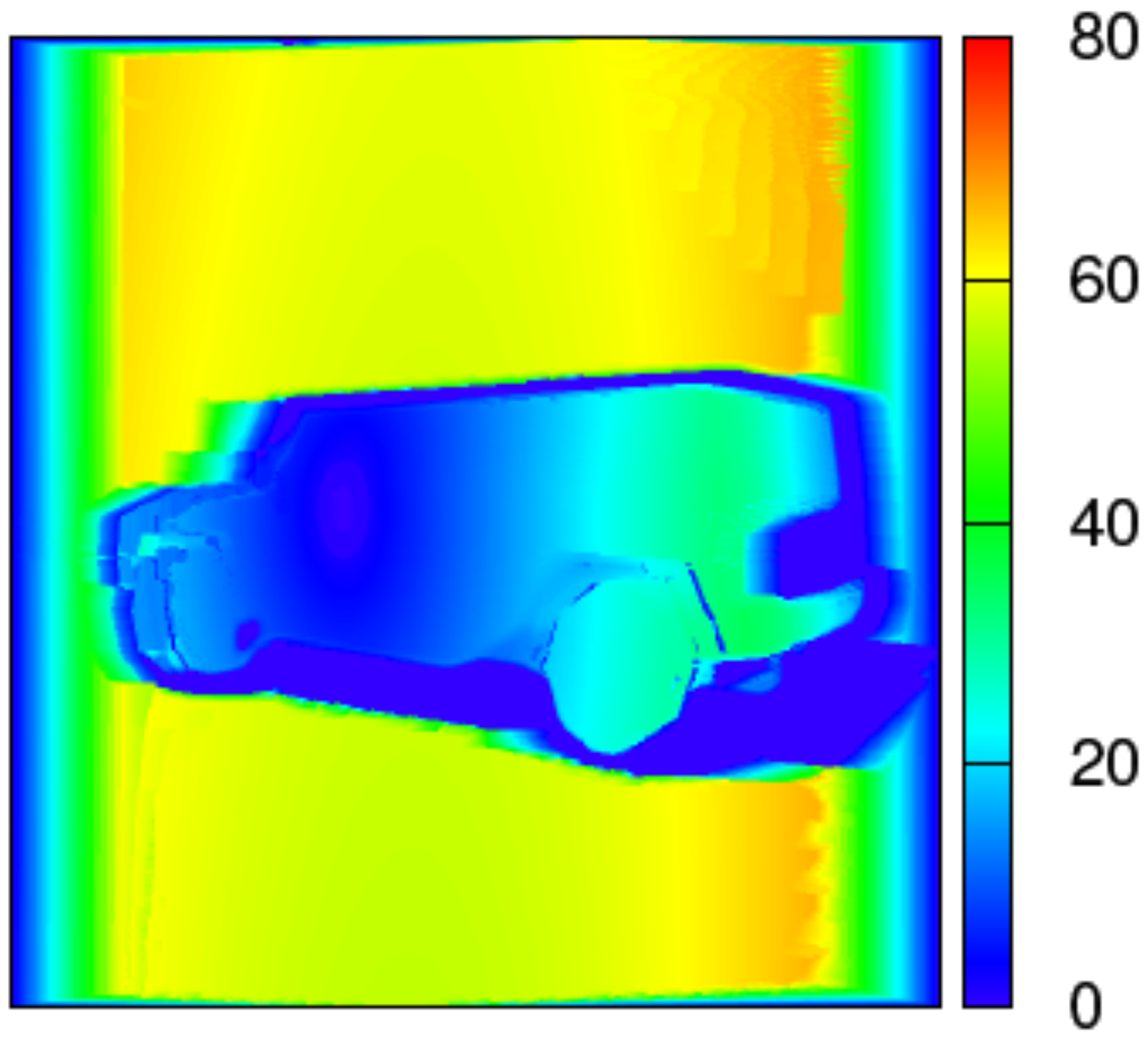
min speed



max speed

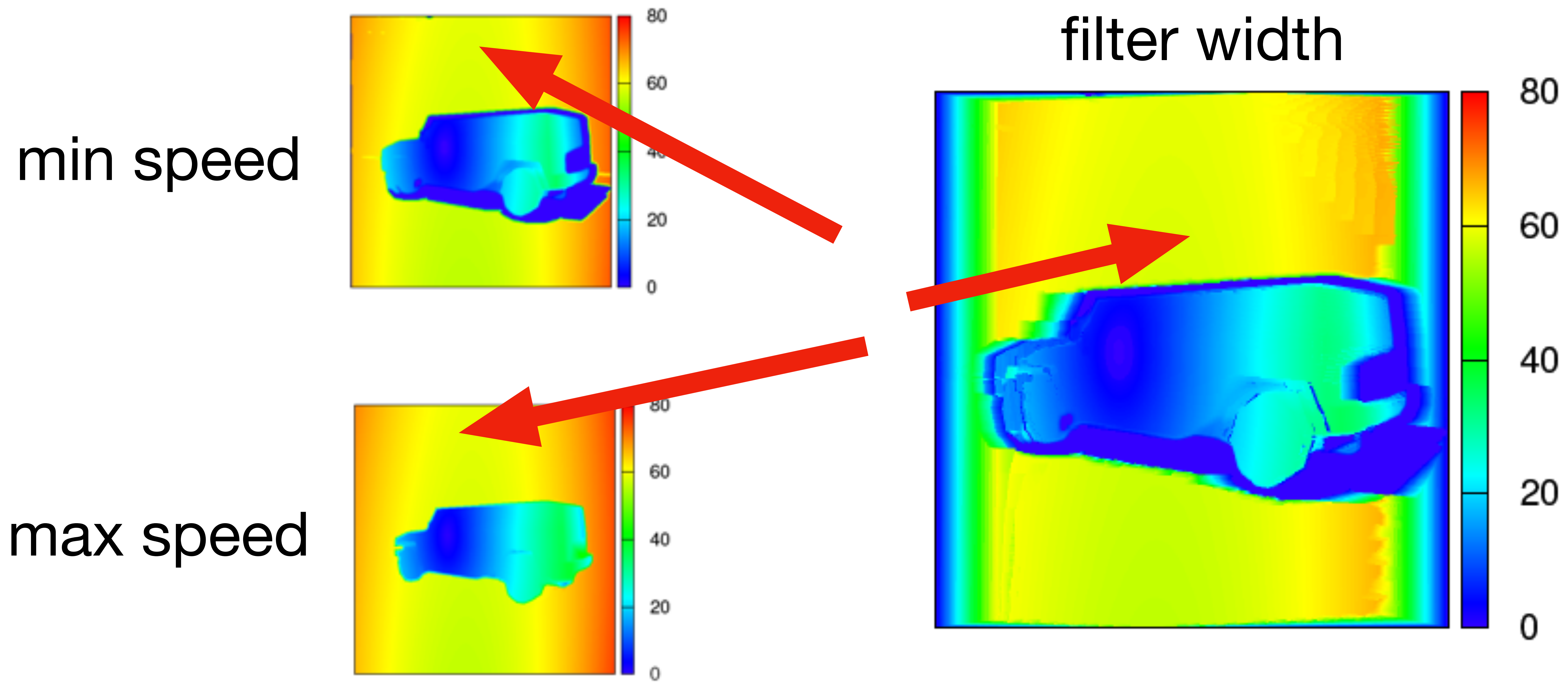


filter width



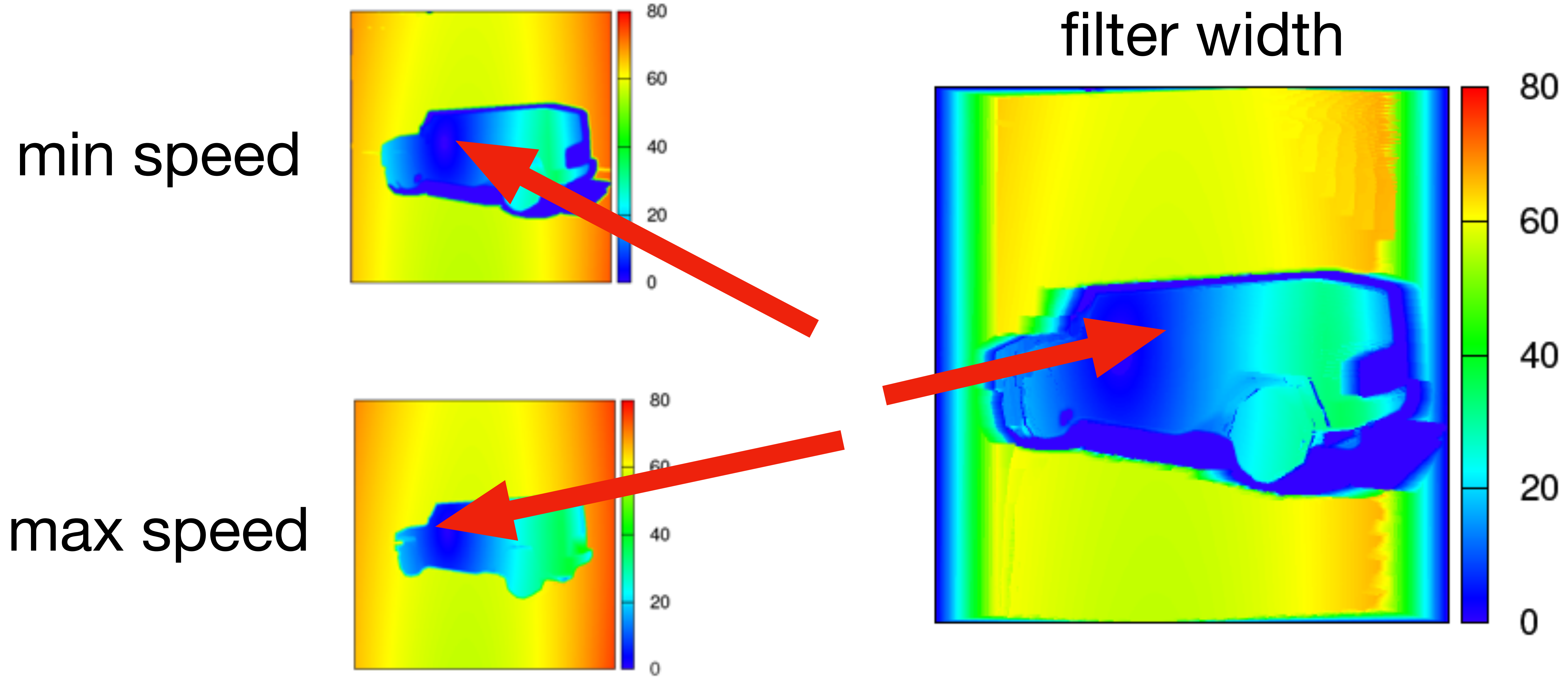
Implementation: stage 1

Uniform velocities, wide filter, low samples



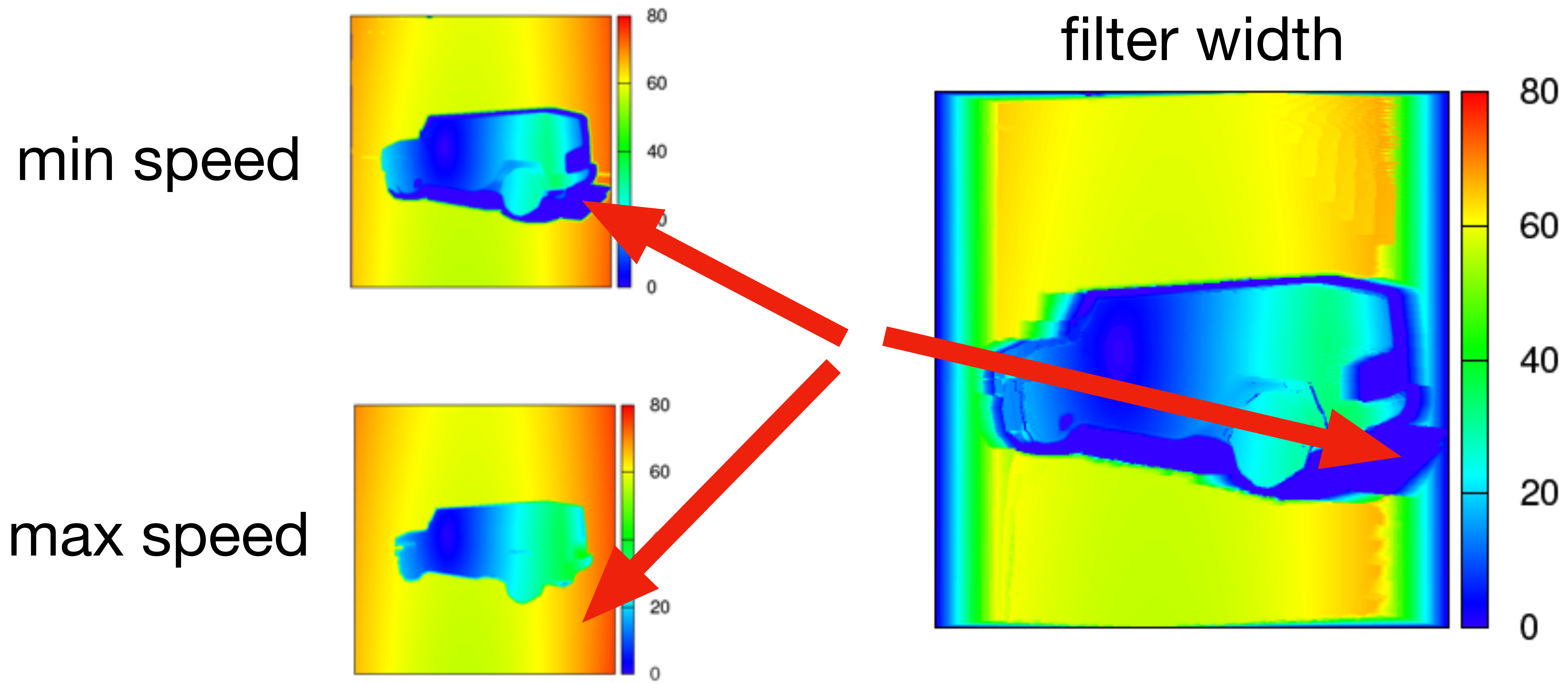
Implementation: stage 1

Static surface, small filter, low samples



Implementation: stage 1

Varying velocities, small filter, high samples

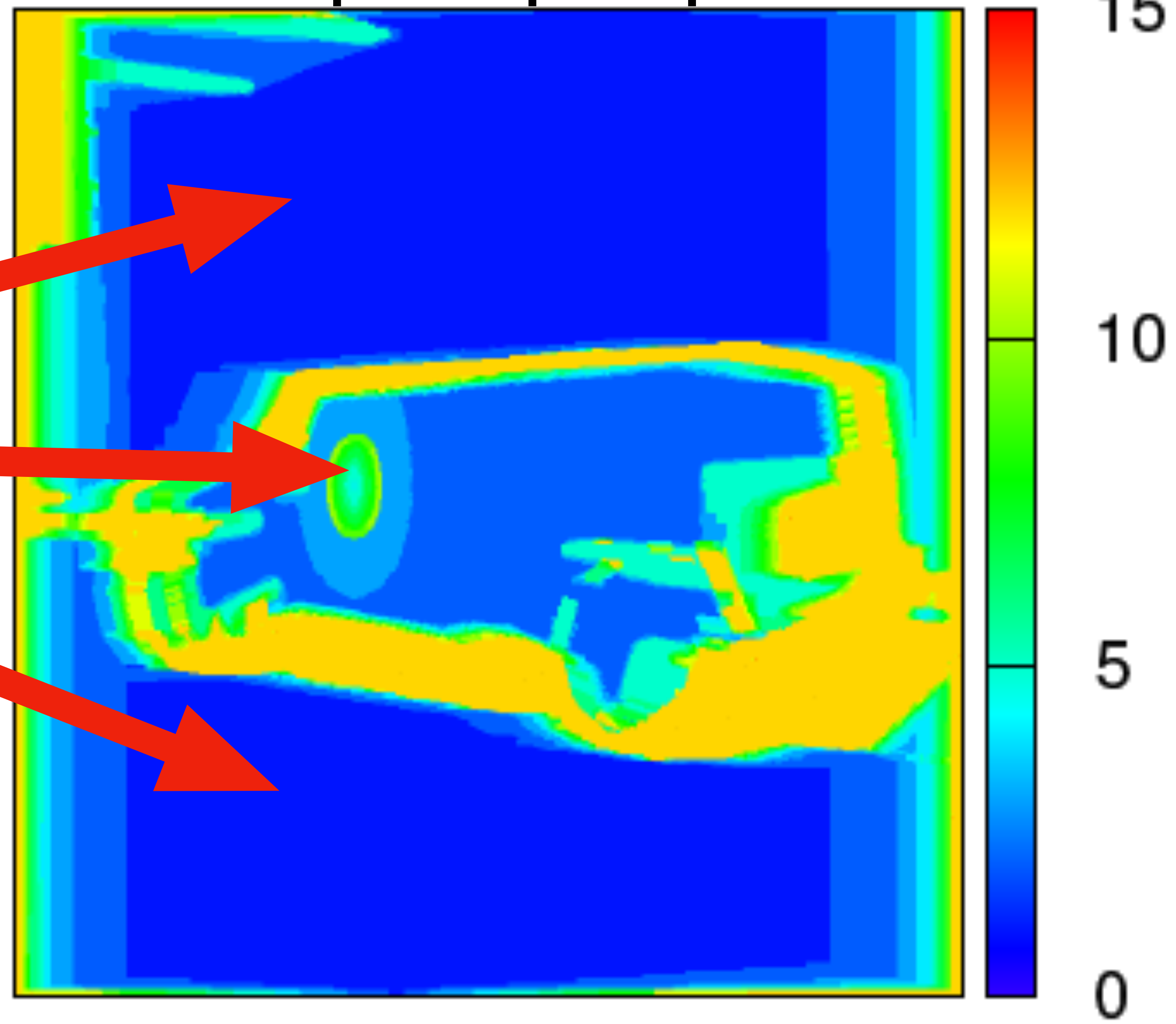


Implementation: stage 2

Then, compute sampling densities

Uniform velocities = low sample count

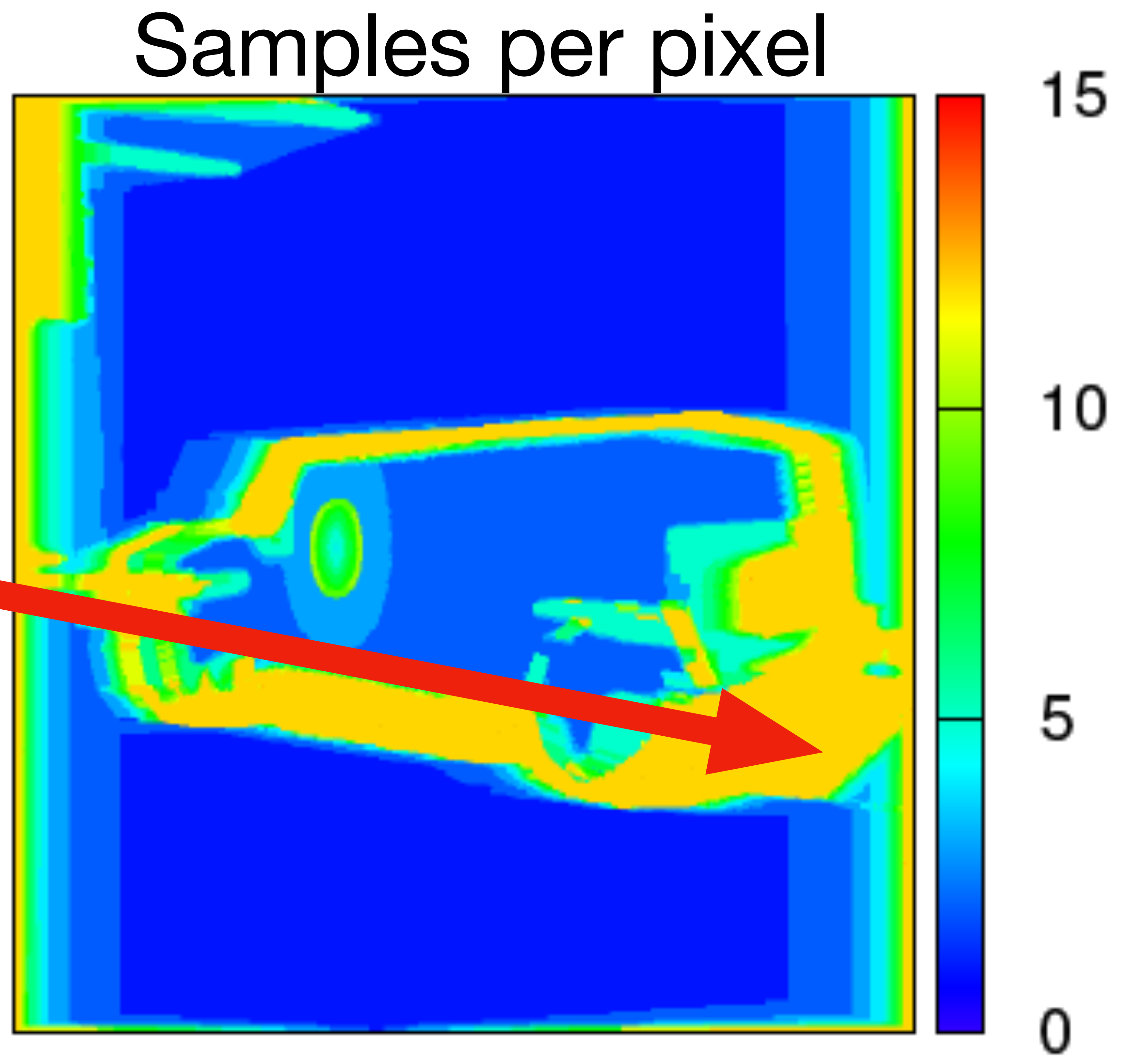
Samples per pixel



Implementation: stage 2

Then, compute sampling densities

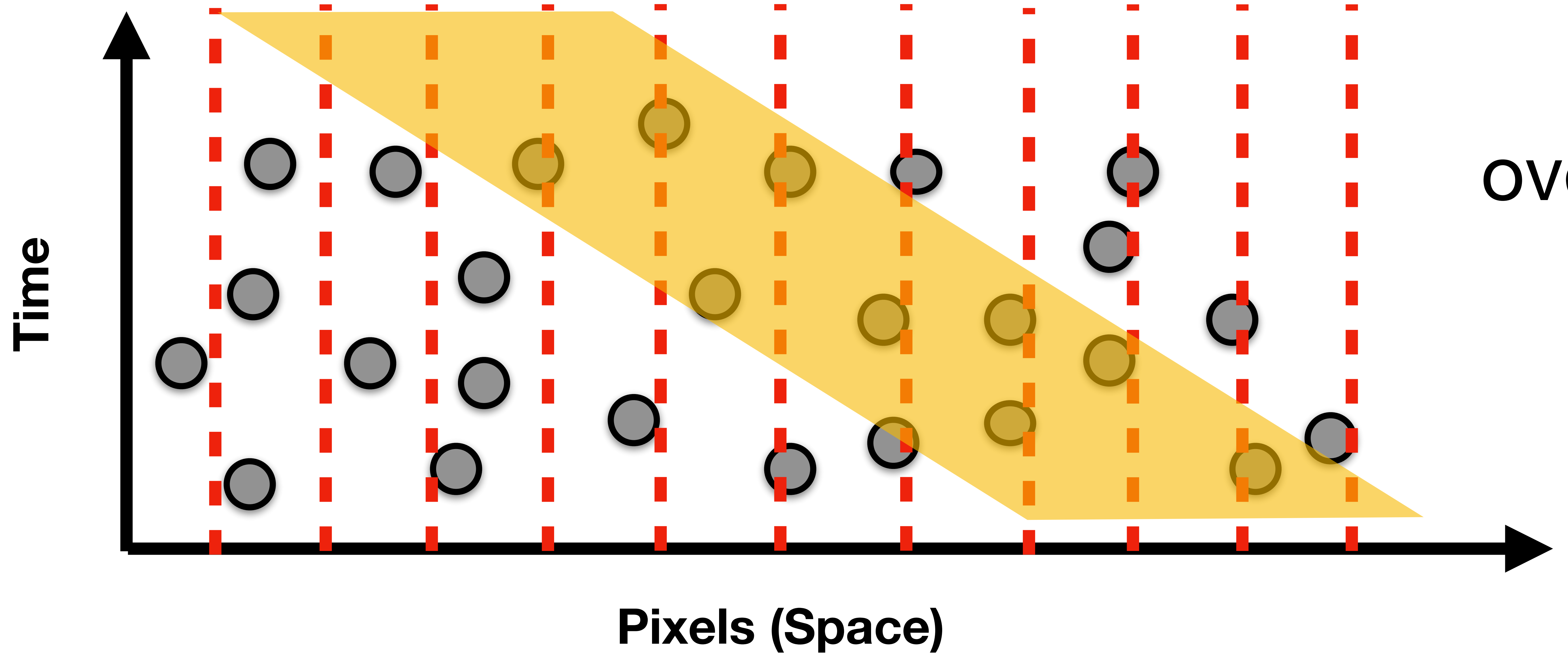
Varying velocities = high sample count



Implementation: stage 3

Render sample locations in space-time

Apply sheared filters to nearby samples



Sheared filters overlaps samples across multiple pixels

Implementation: stage 3

Filters stretched along the direction of motion

Preseve frequencies orthogonal to the motion

Filter shapes

Results

Sheared Filters
4spp

Stratified Sampling
4spp

Multi-dimensional adaptive sampling of distribution effects

Fourier Analysis of Light Transport

Temporal reconstruction of distribution effects

Temporal Light-Field Reconstruction for Rendering Distribution Effects

Lehtinen et al. [2011]

Slides courtesy: Jakko Lehtinen

Pinhole image

Requires dense sampling of 5D function:

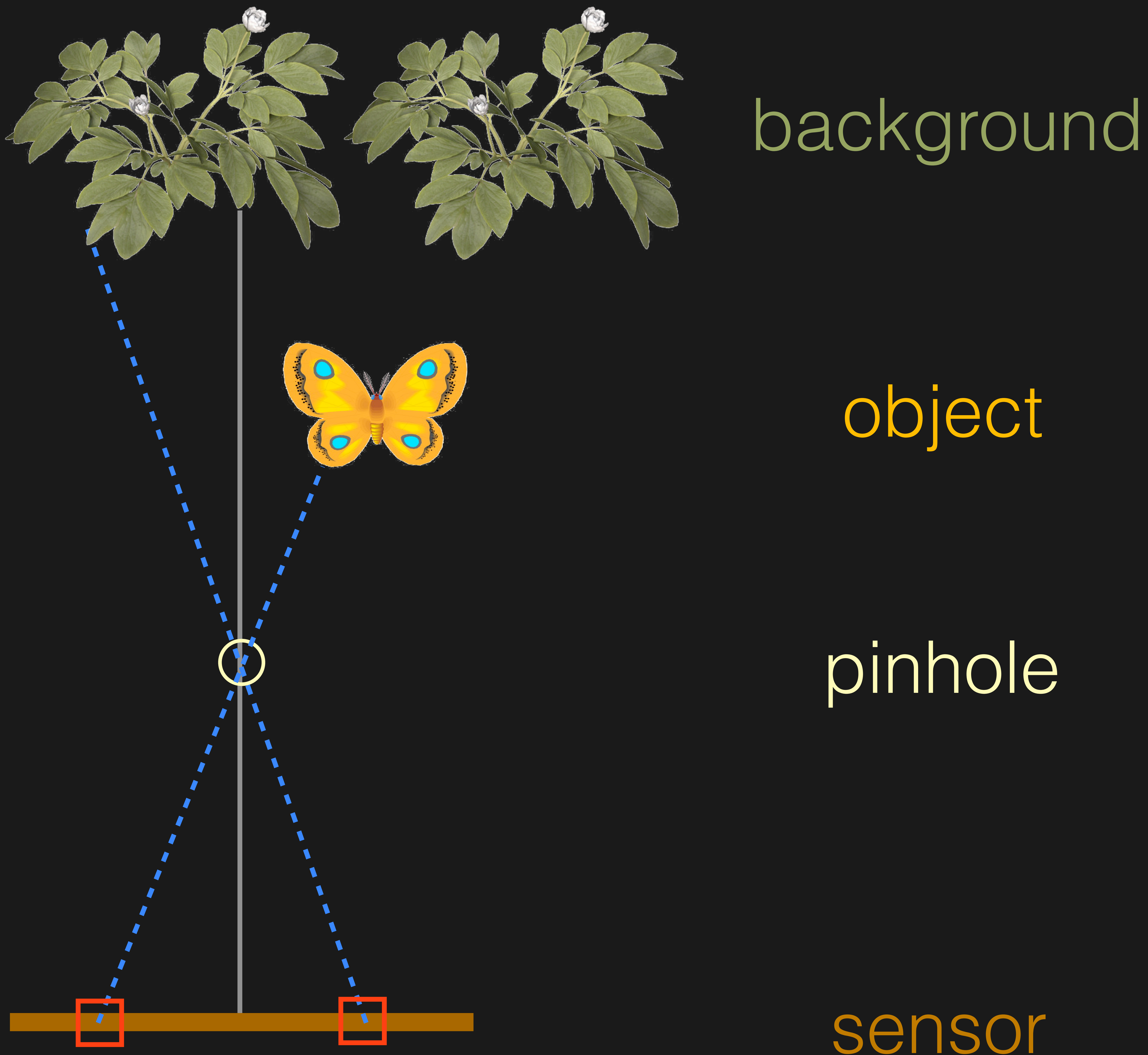
- Pixel area (2D)
- Lens aperture (2D)
- Time (1D)

With motion blur and depth of field

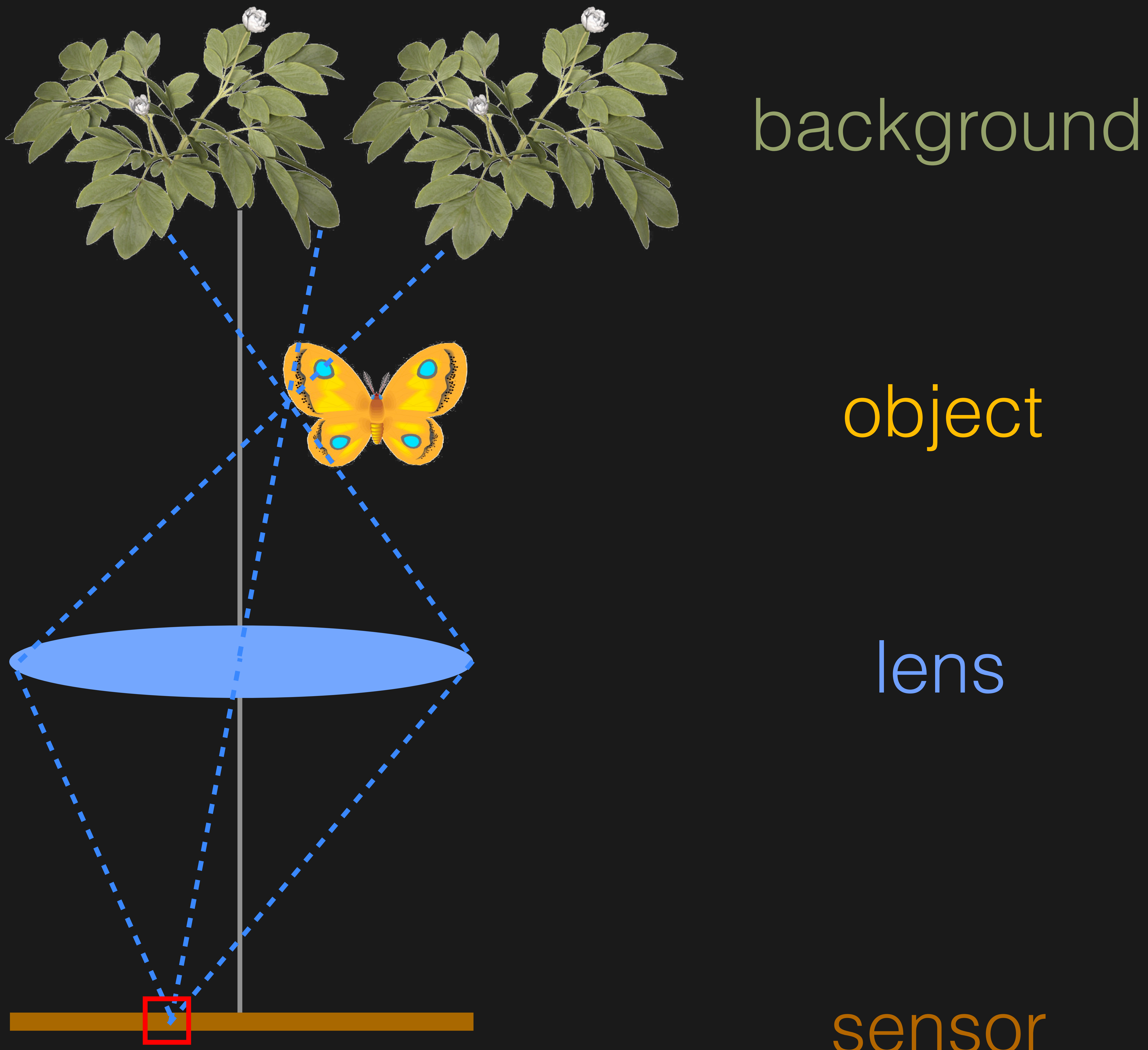
Motion blur and depth of field 1 sample per pixel

Our reconstruction

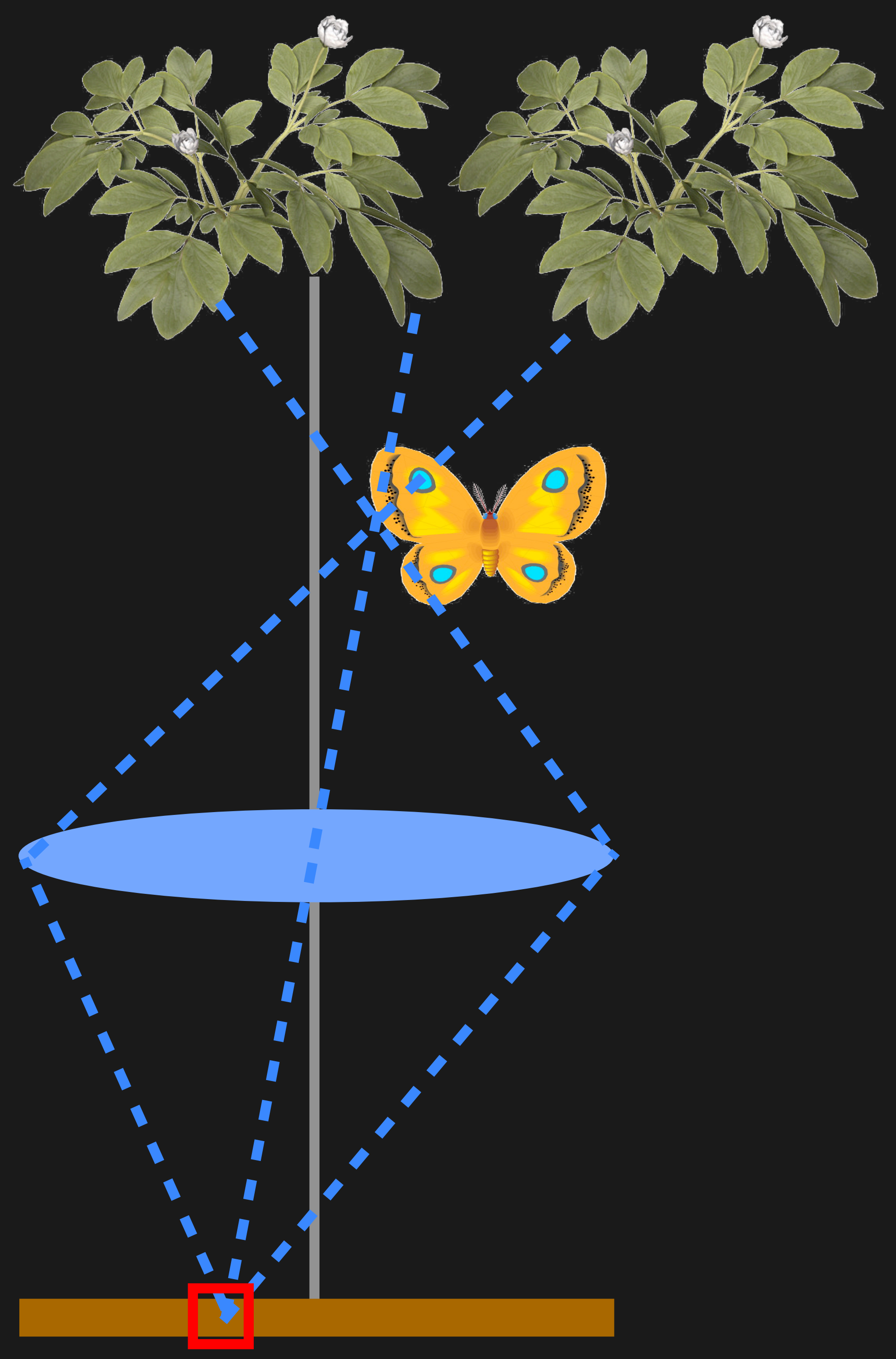
Pinhole camera model



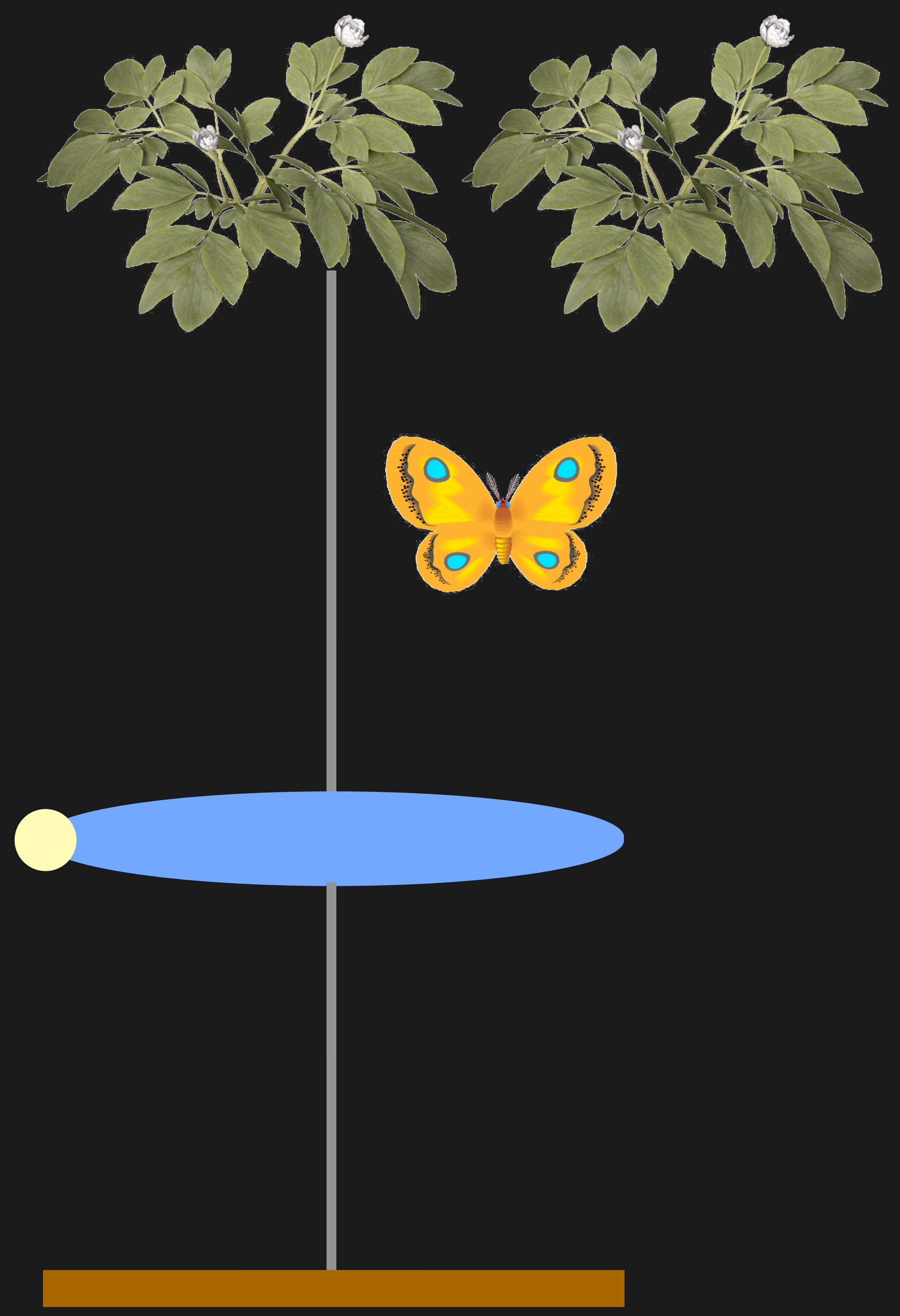
Thin lens camera model



Depth of field



Depth of field



1 scanline


Lens u
Screen x


1 pixel
Lens u
Screen x


Lens u
Screen x

Light field [Levoy 1996]
Lens u
Output:
integration over lens
Screen x

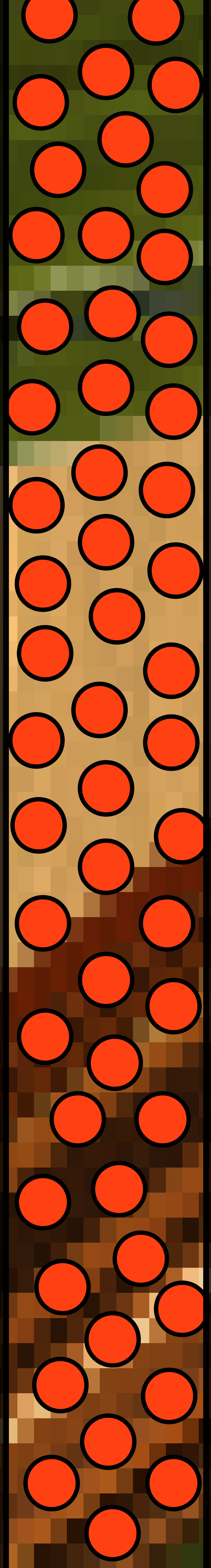
Monte Carlo sampling

Low sample density leads to noise

Monte Carlo sampling

Need many samples to capture the signal:

computationally expensive



Lens u

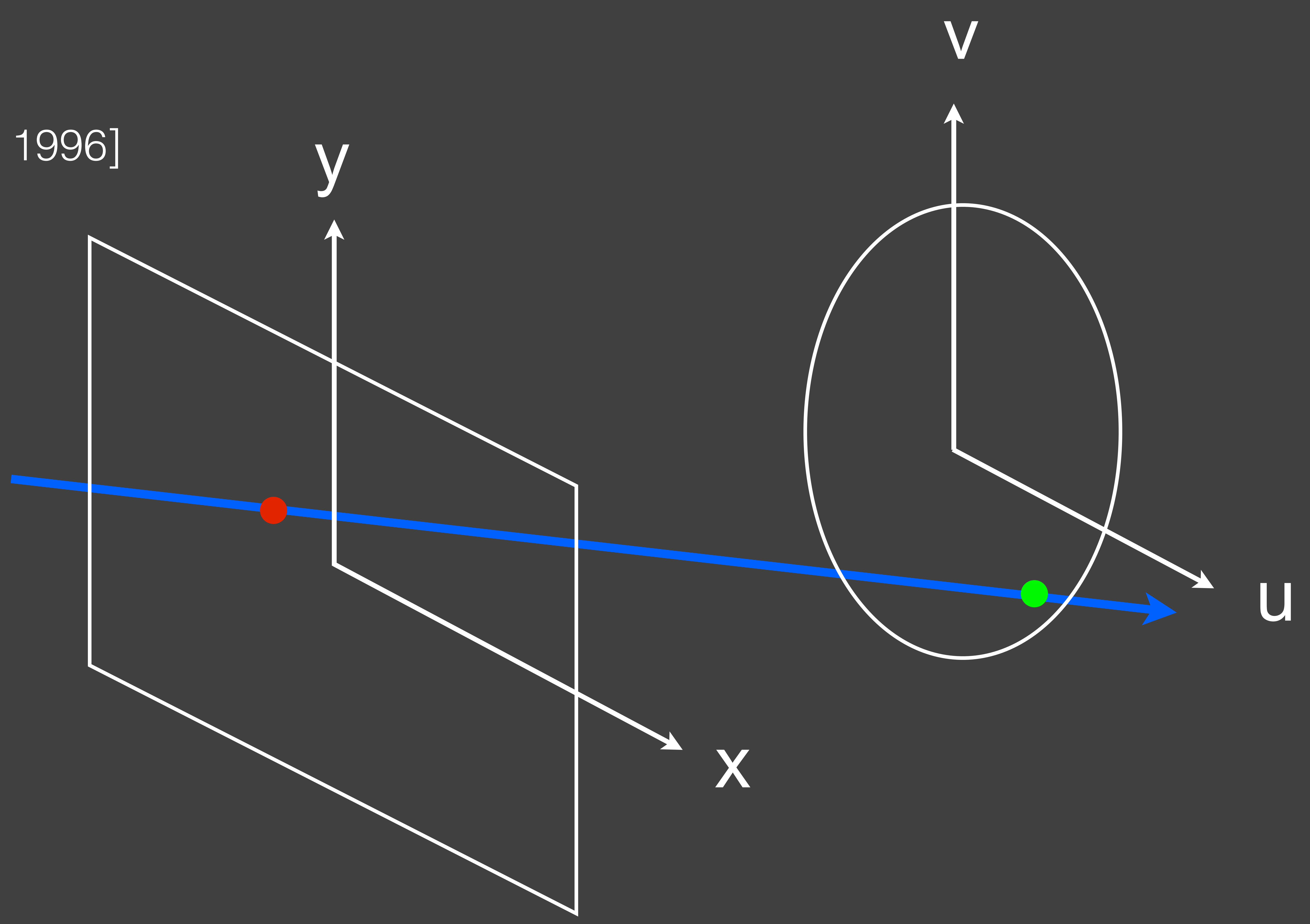
Screen x

Temporal light fields

Traditional light field is 4D [Levoy 1996]

x,y over sensor (2D)
u,v over lens (2D)

Add time dimension for moving geometry (5D)




Lens u
Screen x


Lens u
Screen x

The Integrand is Anisotropic

[Chai00, Durand05, Hachisuka08, Soler09, Egan09, ...]

Multi-dimensional Adaptive Sampling [Hachisuka 08]

Frequency Analysis and Sheared Reconstruction [Egan 09]

Our approach

Start with **sparse** input sampling

Our approach

Start with **sparse** input sampling

Perform **dense** reconstruction using sparse input samples

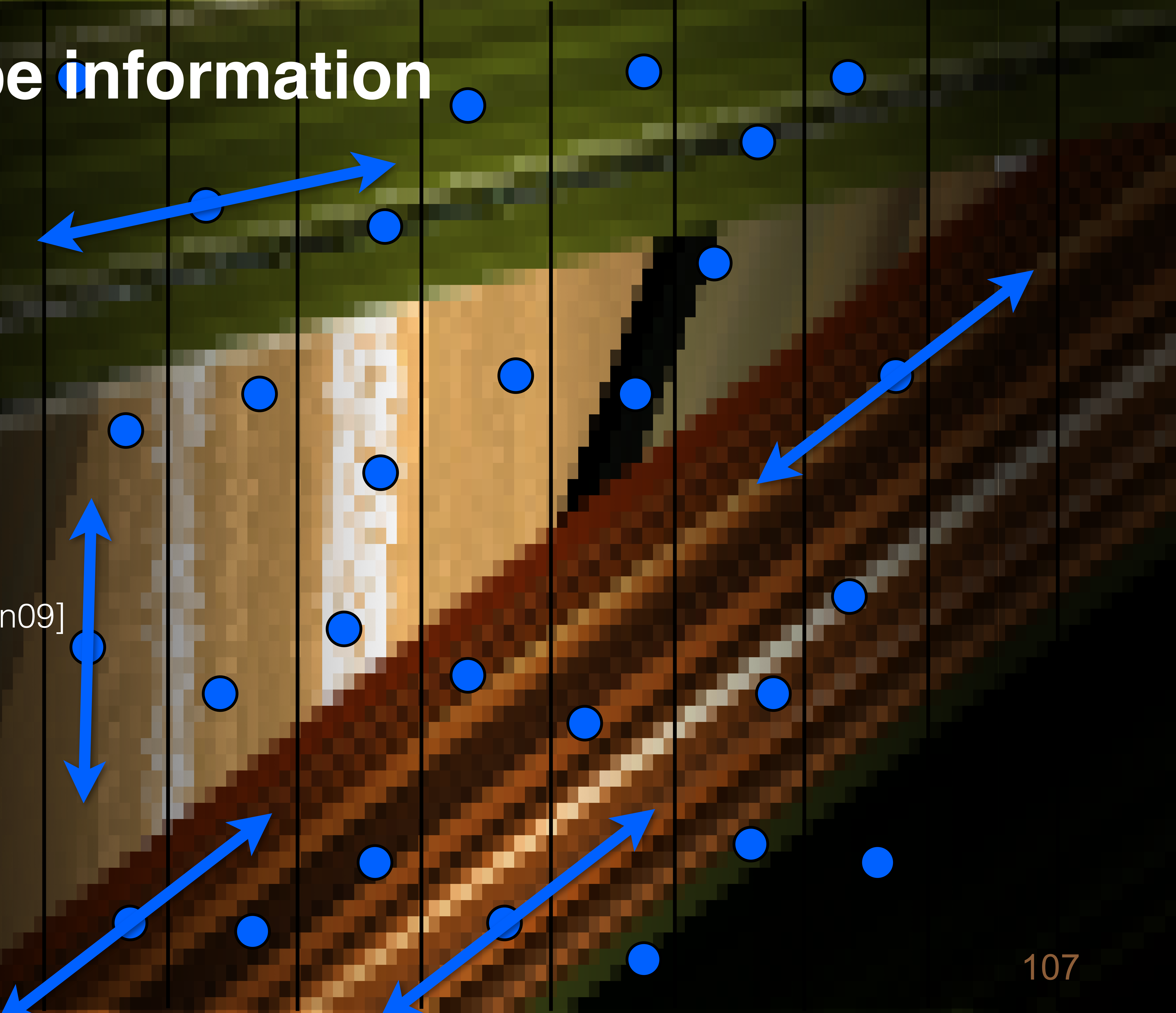
Standard Monte-Carlo integration using dense reconstruction

Our input has **slope information**

For defocus, proportional to inverse **depth 1/z** [Chai00]

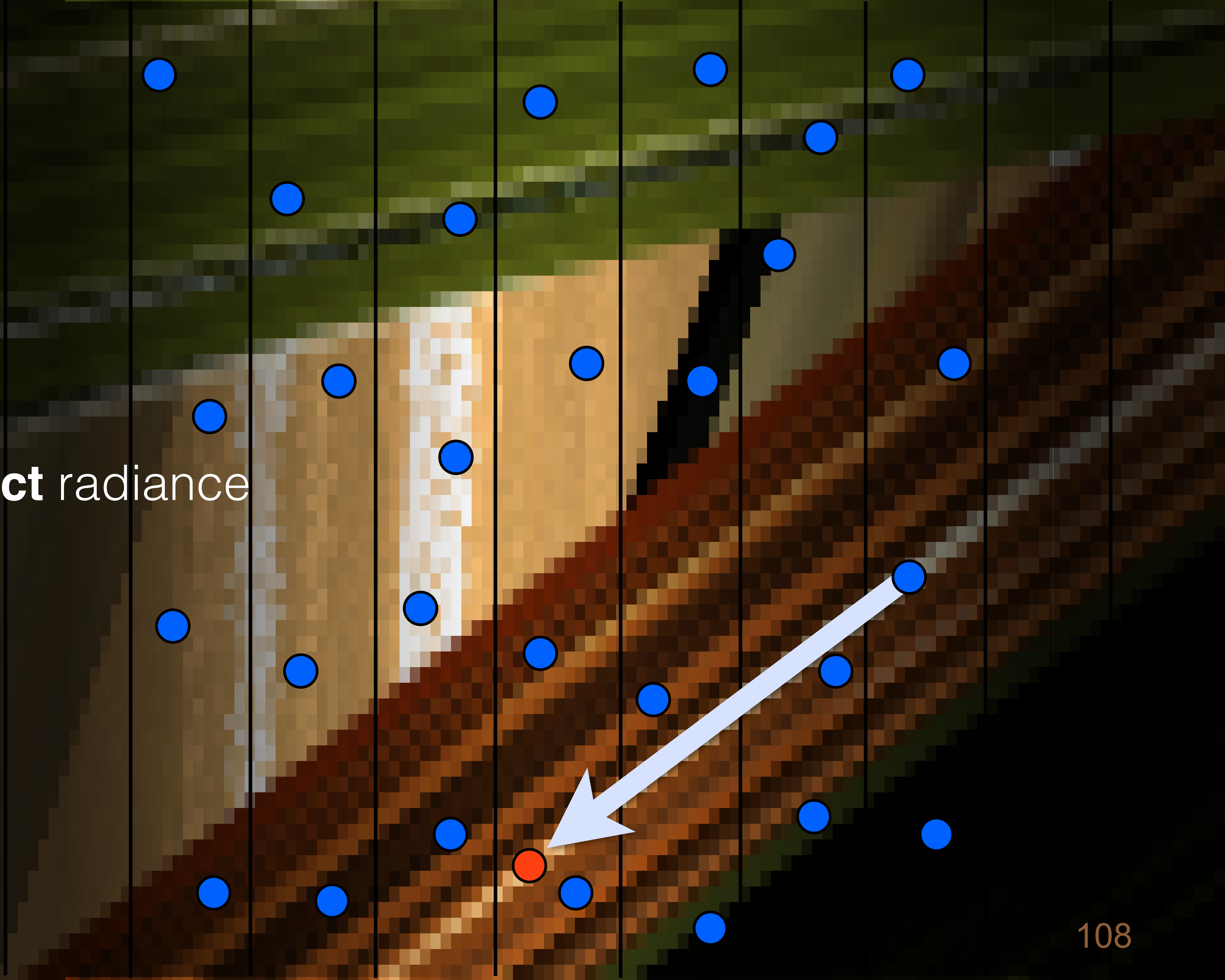
For motion, proportional to inverse **velocity 1/v** [Egan09]

Easy to output from any renderer.



What is the radiance at the red location?

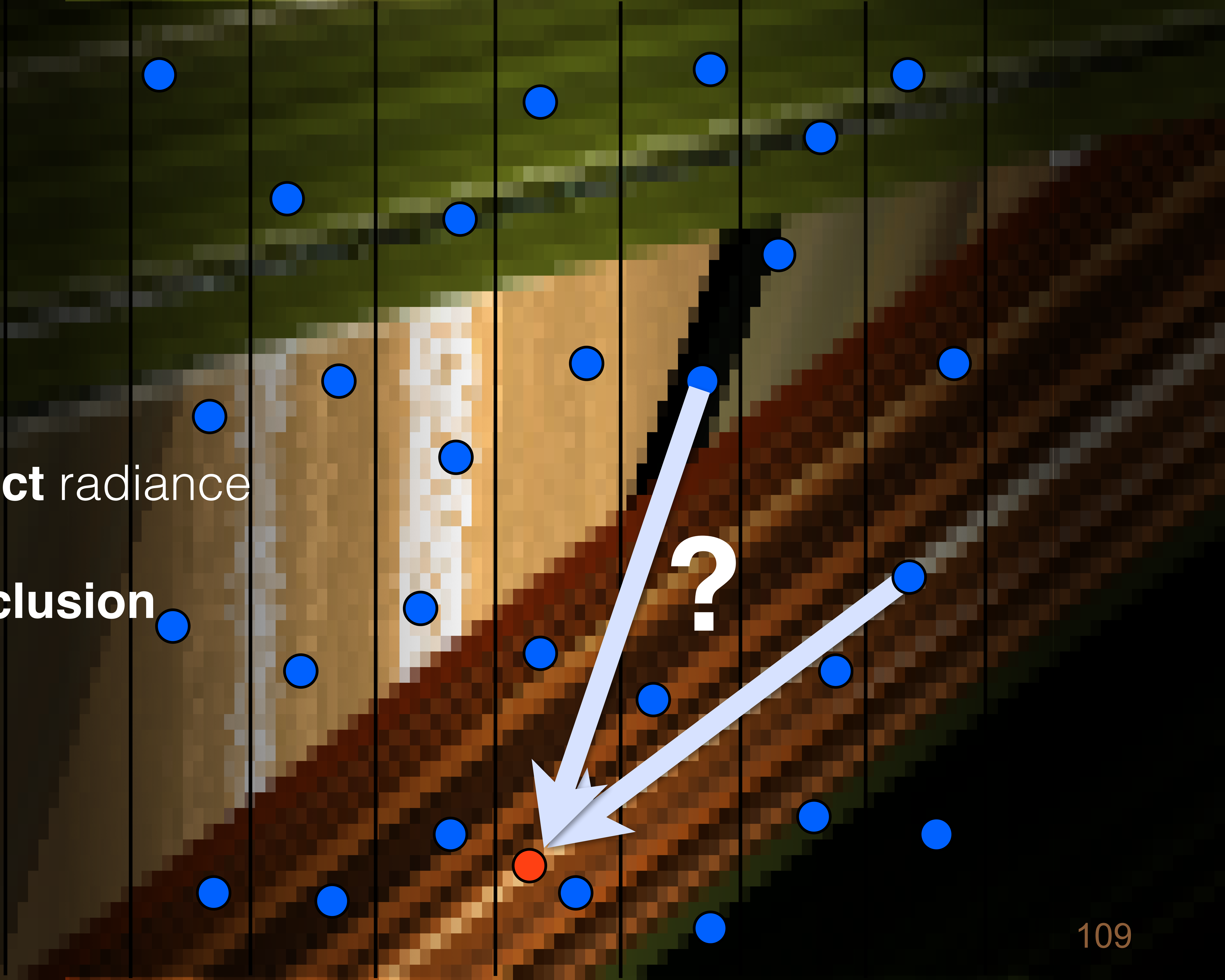
Use slope to **reproject** radiance



What is the radiance at the red location?

Use slope to **reproject** radiance

Must account for **occlusion**



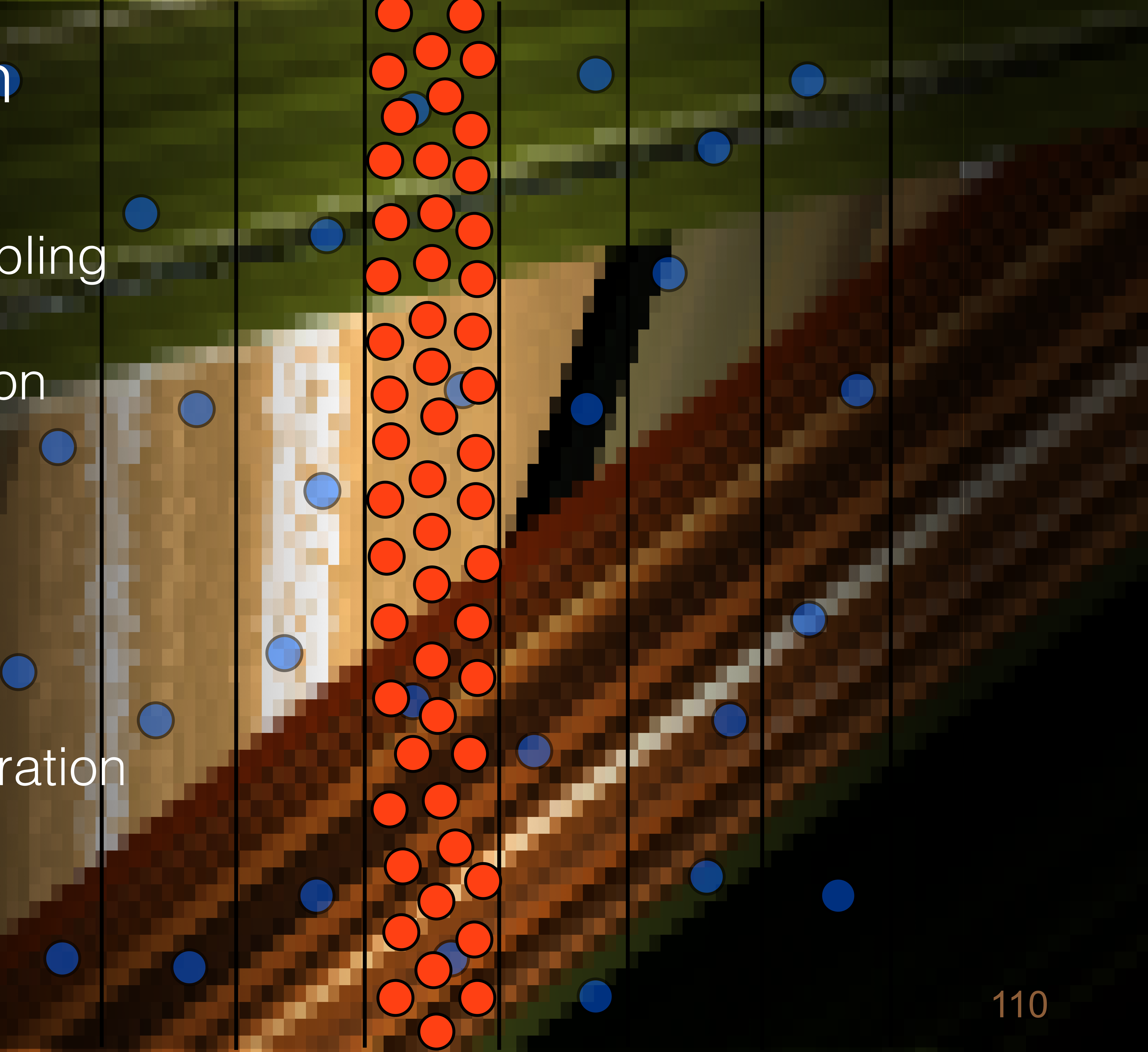
Recap: our approach

Start with **sparse** input sampling

Perform **dense** reconstruction using sparse input samples

- Use **slopes** to reproject
- Account for **visibility**

Standard Monte-Carlo integration using dense reconstruction

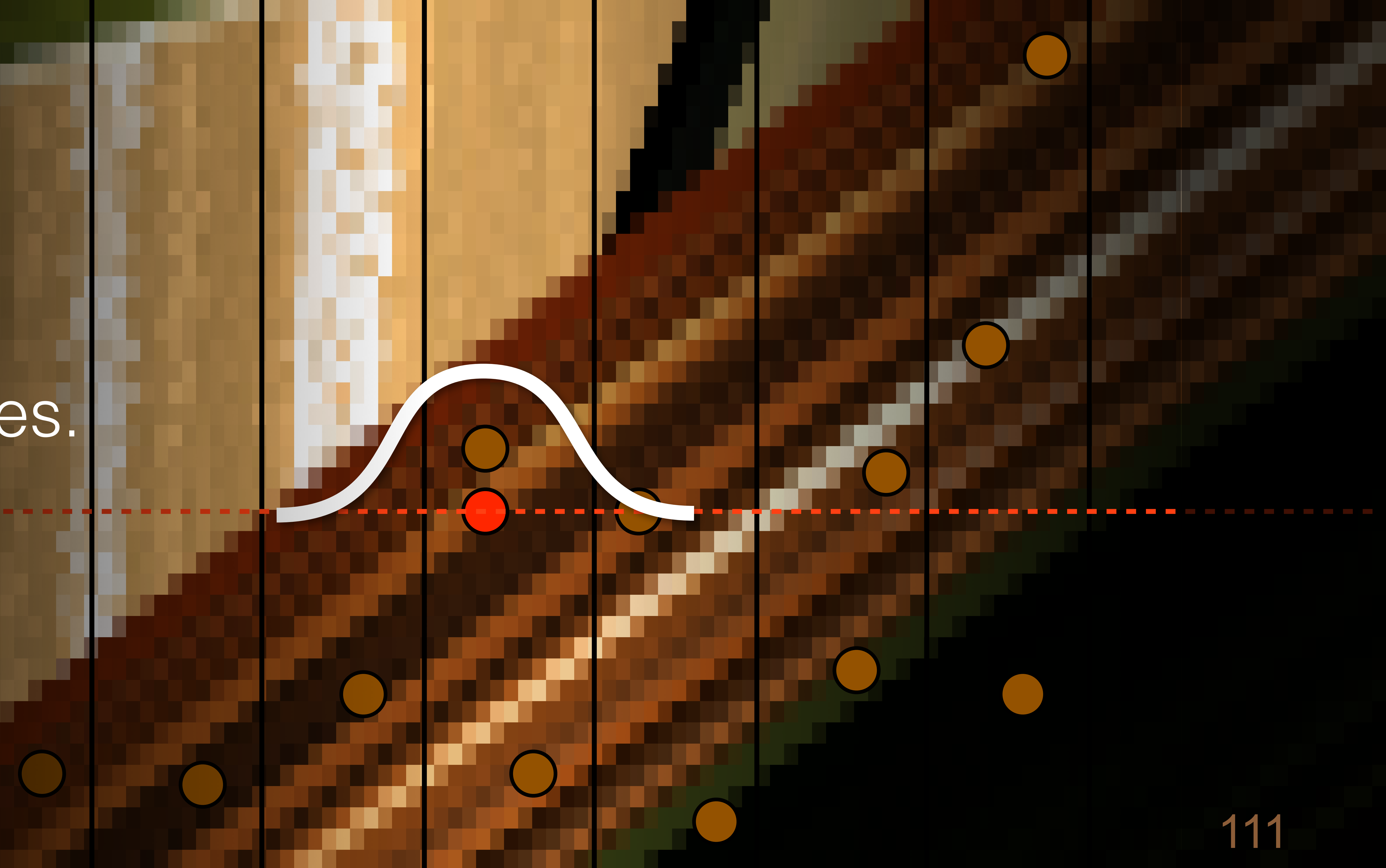


Reprojection and filtering

Simplify visibility by reprojecting into **screen space**.

Reproject to u, v, t of reconstruction location.

Pixel filter over **visible** samples.

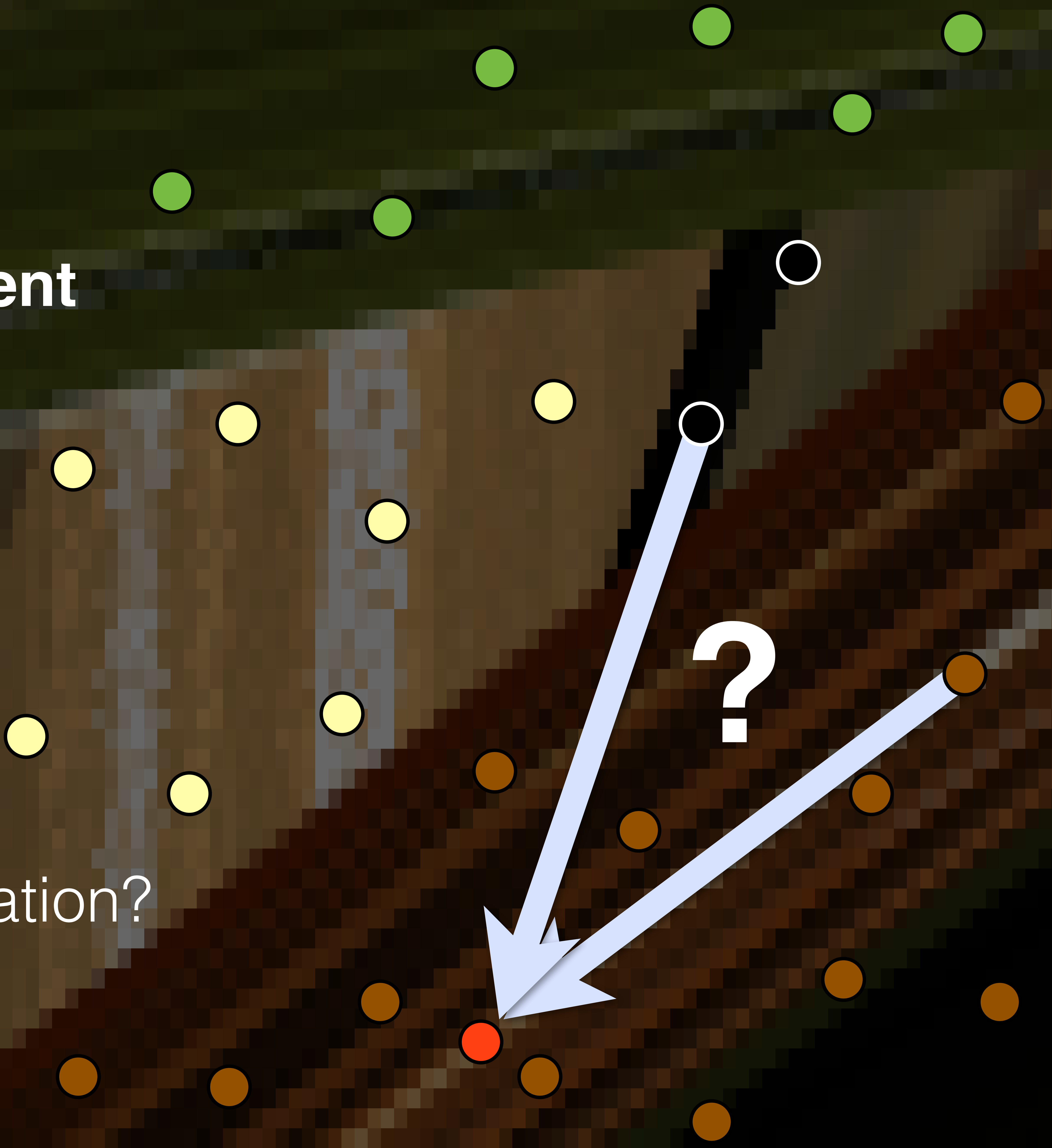


Visibility

Cluster samples into **apparent surfaces** to resolve *z*-order

SameSurface algorithm

Determining **coverage**:
Does the apparent surface cover my reconstruction location?



Visibility: SameSurface

Input:

sparse **points** with **slopes**

Visibility: SameSurface

The trajectories of samples originating from a single **apparent surface** never intersect.

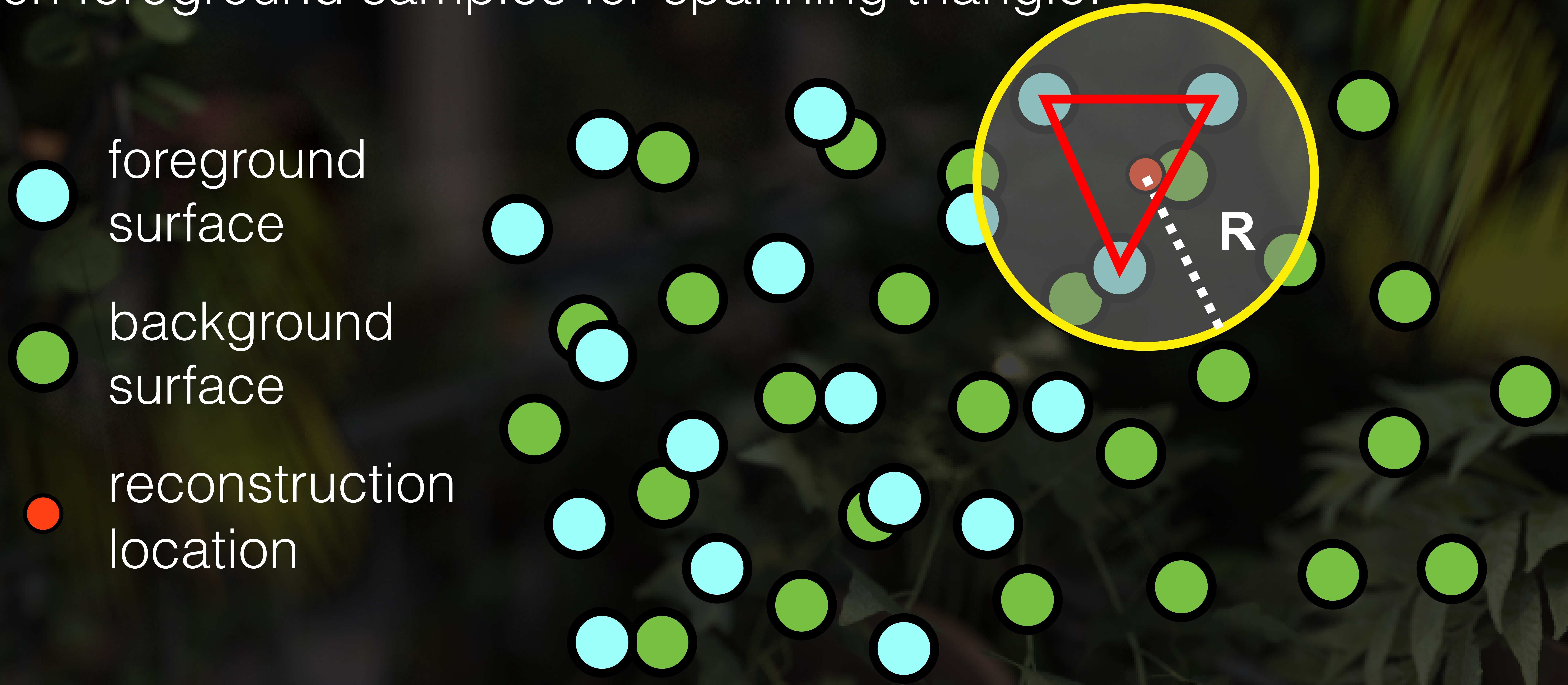
Visibility: SameSurface

Visibility events
show up as **intersections**

Visibility: Coverage

Does foreground apparent surface cover reconstruction location?

Search foreground samples for spanning triangle.



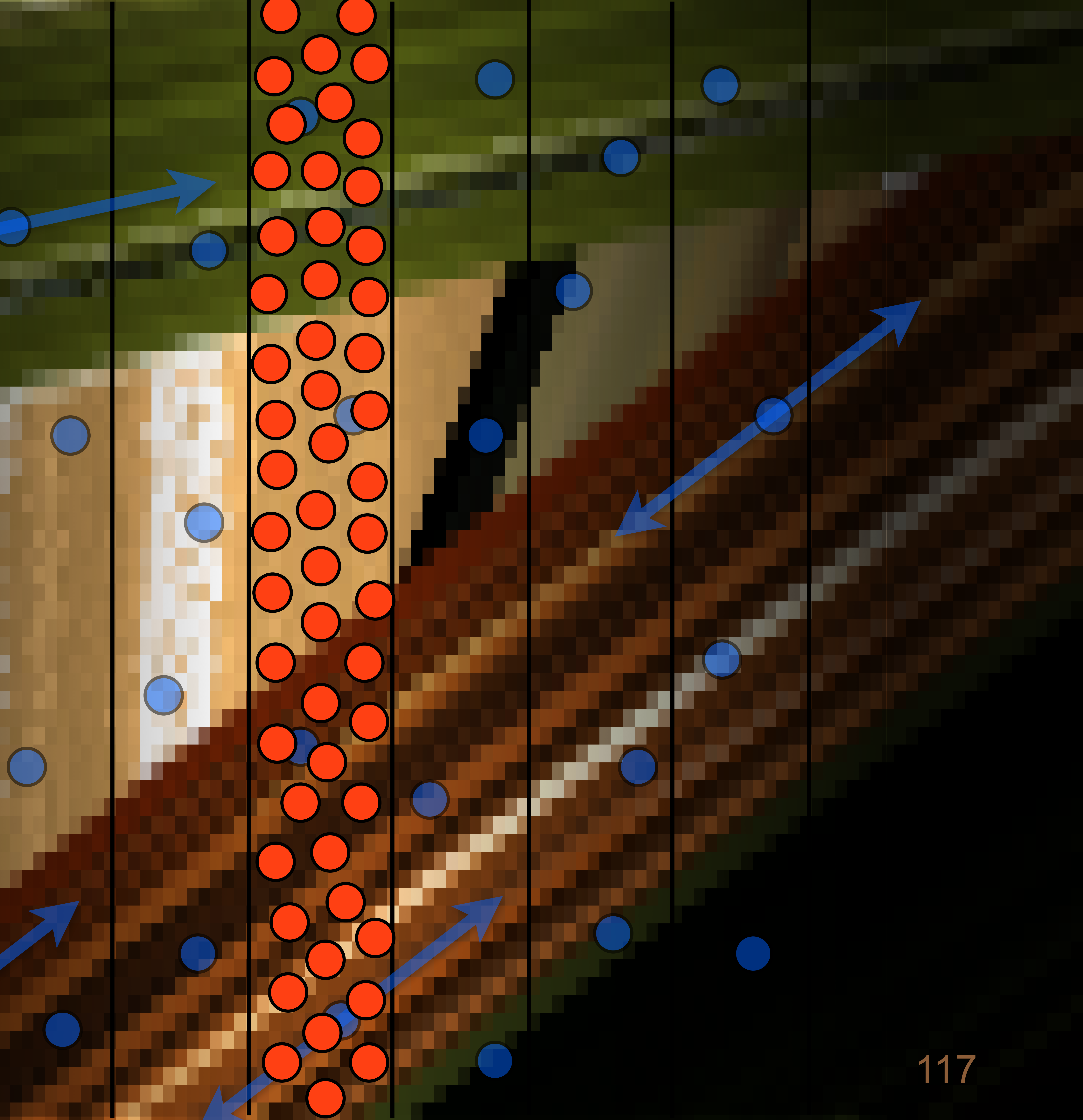
Recap: our approach

Start with **sparse** input sampling

Perform **dense** reconstruction using sparse input samples

- Use **slopes** to reproject
- Account for **visibility**

Standard Monte-Carlo integration using dense reconstruction



Observations

We only need sample radiance, depth, and velocity (i.e., **slopes**).
Reconstruction is **independent** of the original renderer.

We can **discard** the scene.

Observations

We only need sample radiance, depth, and velocity (i.e., **slopes**).
Reconstruction is **independent** of the original renderer.

We can **discard** the scene.

Need efficient **sample search**:

Fast motion and large defocus can lead to a single sample contributing to **hundreds** of pixels.

Build a **hierarchy** over input samples.

Extension to soft shadows

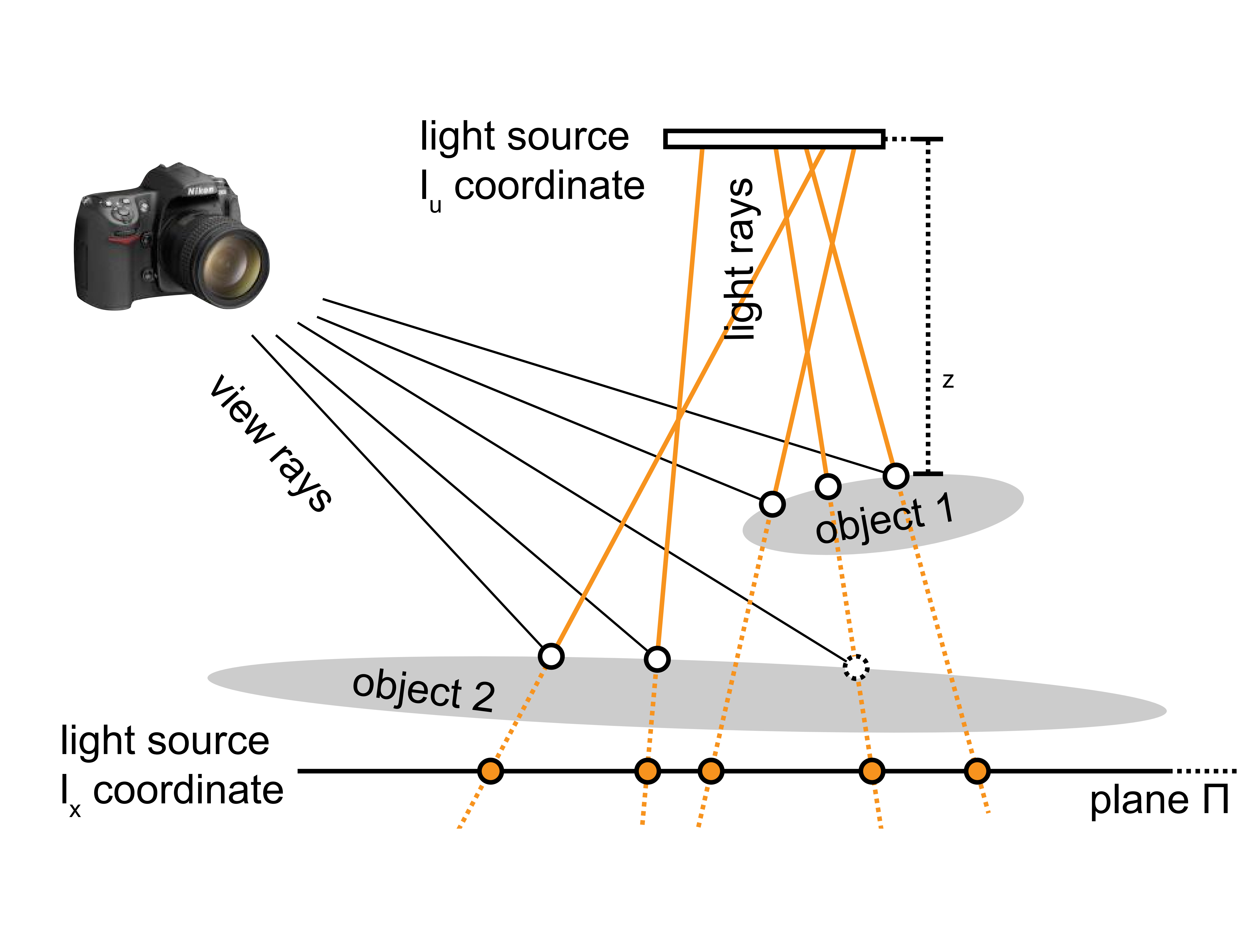
An **area light** is very much like a **lens**.

lens ~ light, sensor ~ virtual plane
Reconstruct **z** instead of radiance

Egan et al. [2010] reconstruct
far field **binary visibility** only.

7D path-tracing style reconstruction
avoiding **combinatorial explosion**

Reconstruct scene point (**5D**)
Reconstruct shadow **z** shade (**2D**)



Results

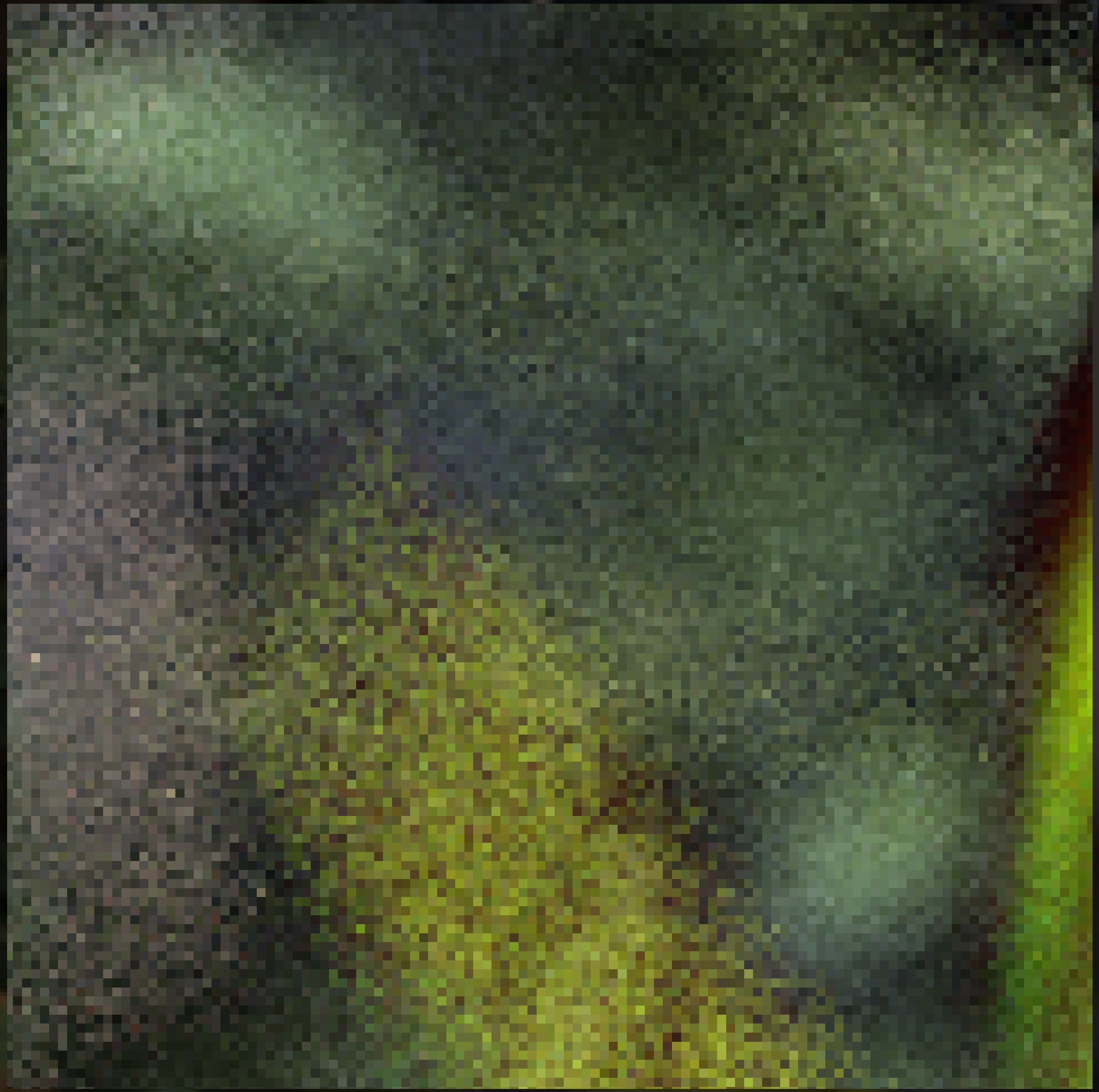
Implementation

Multithreaded **CPU**
GPU, excluding hierarchy construction

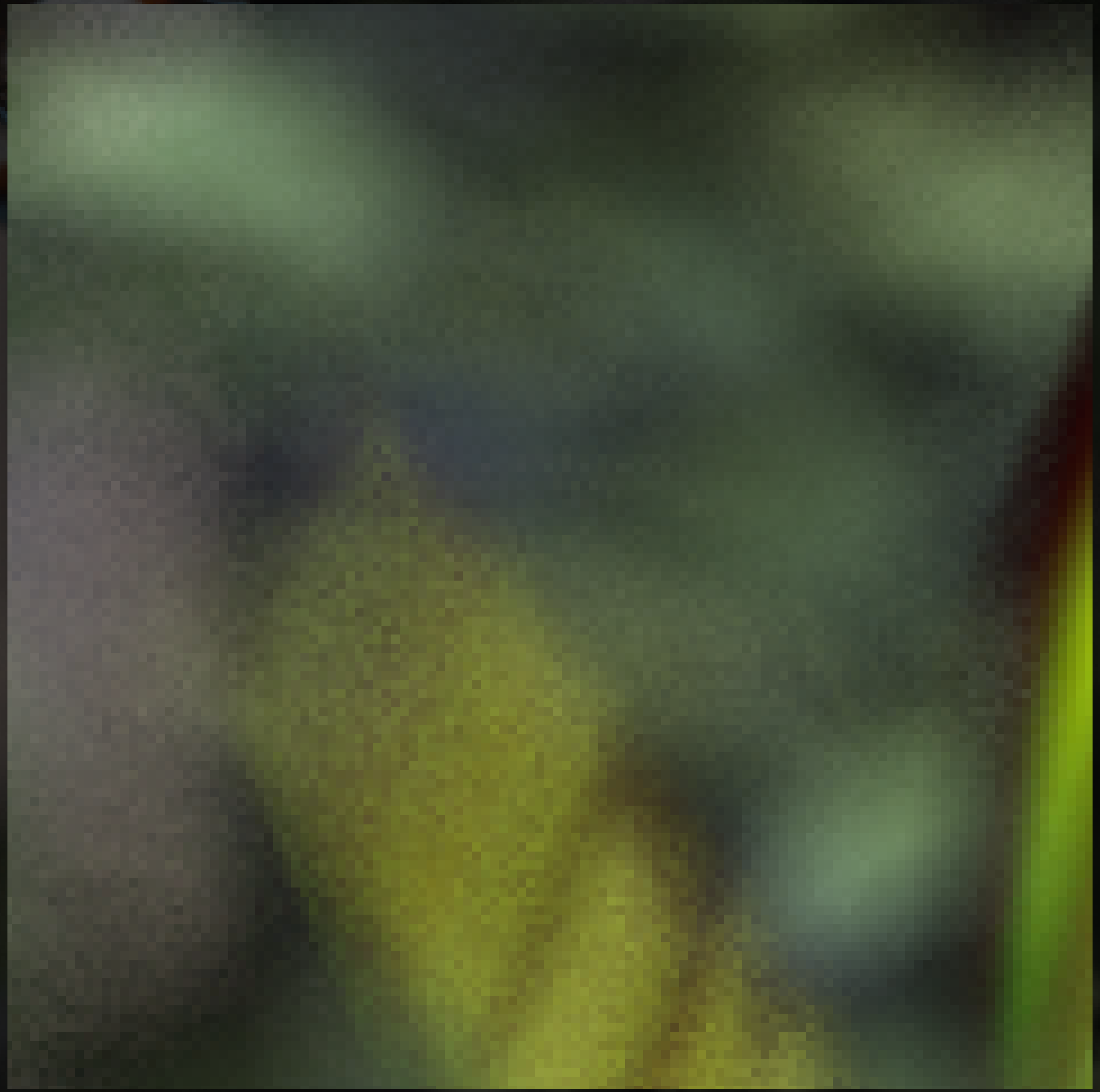
Common sample buffer format accepts outputs from:
- **PBRT**
- **Pixie** (Open source RenderMan)
- Custom ray tracer


Input: 16 spp
1072 sec (PBRT)

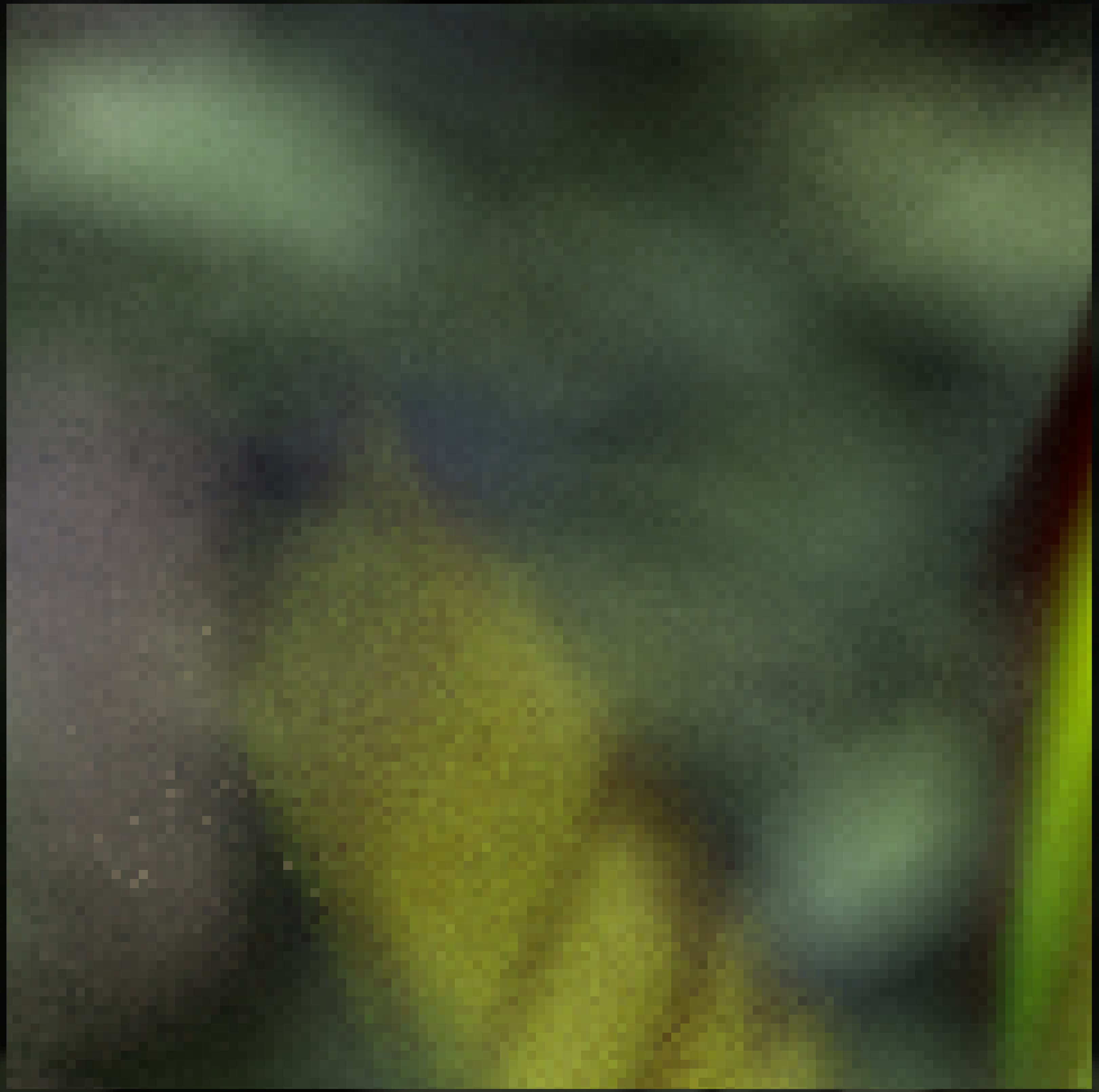
Our result: 16 spp + reconstruction at 128spp
1072 sec (PBRT) + 10 sec (reconstruction)



Input: 16 spp



Our result at 128 spp
using same input



Reference: 256 spp
(16x time)

Our result: 16 spp + reconstruction at 128spp
1072 sec (PBRT) + 10 sec (reconstruction)


Input: 16 spp
771 sec (PBRT)

Our result: 16 spp + reconstruction at 128spp
771 sec (PBRT) + 10 sec (reconstruction)

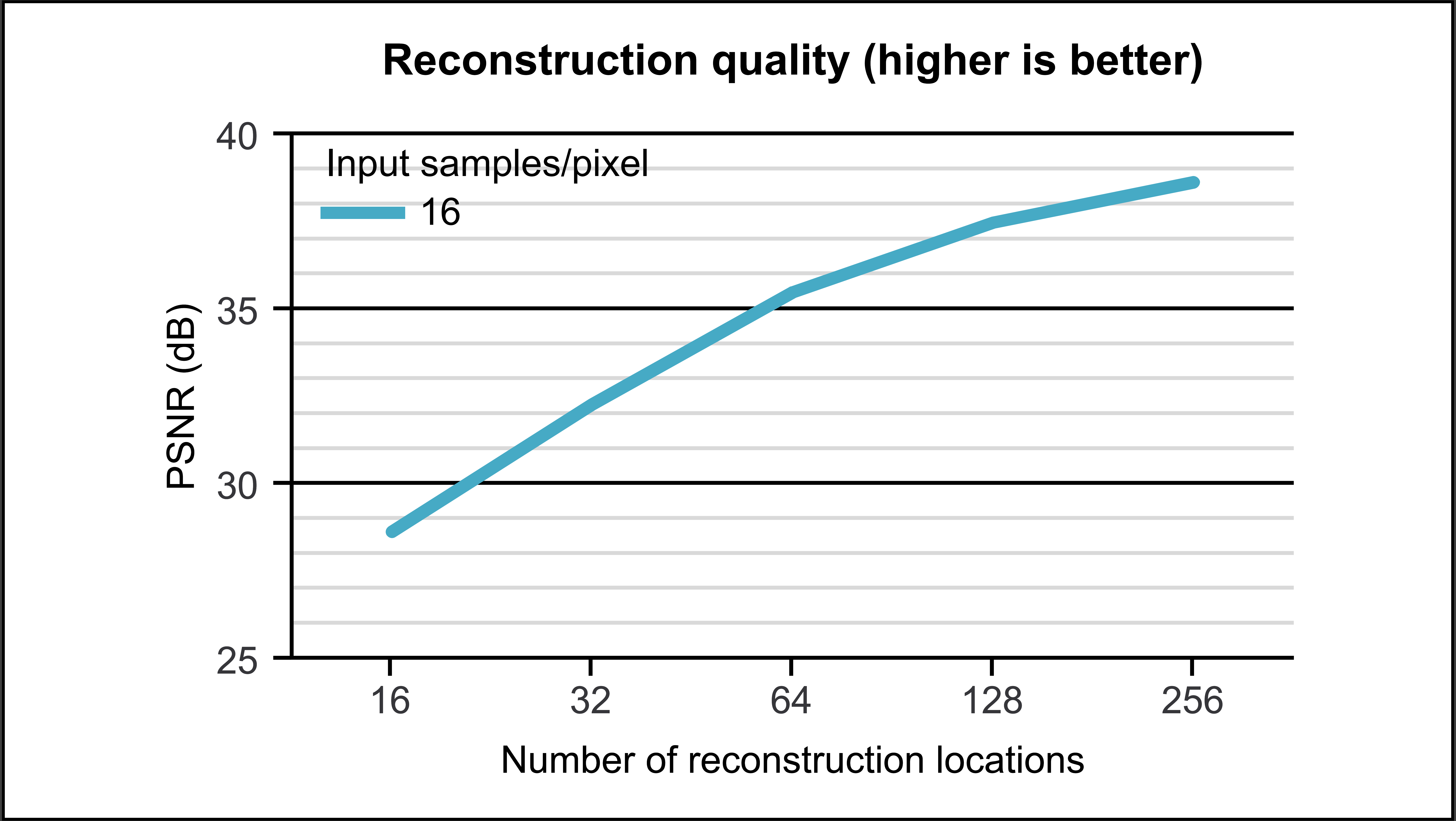
Our result: 16 spp + reconstruction at 128spp
771 sec (PBRT) + 10 sec (reconstruction)

Input: 16 spp

Our result at 128 spp
using same input

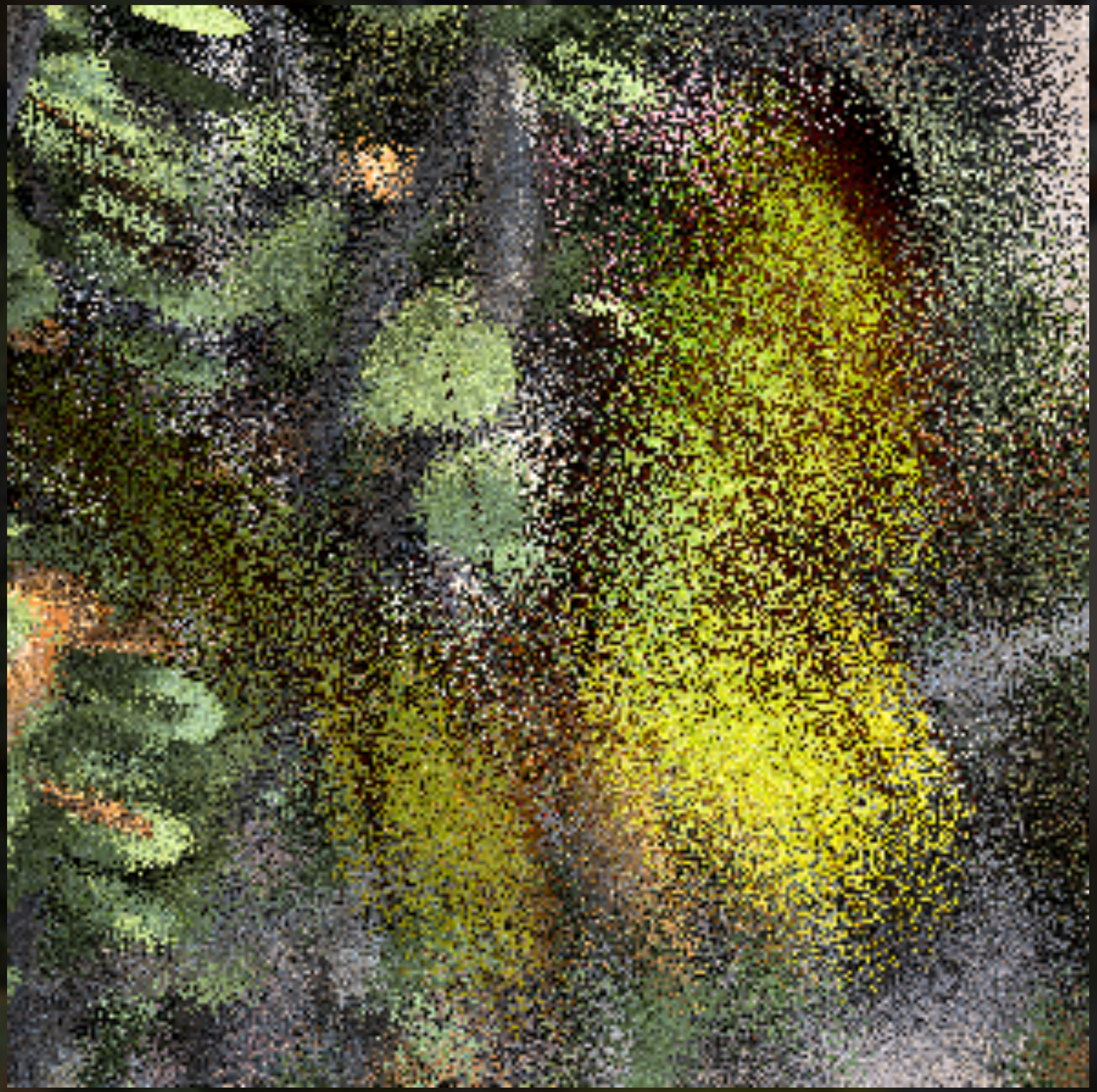
Reference: 256 spp
(16x time)

Comparison to reference




Motion blur and depth of field
1 sample per pixel

Proposed reconstruction



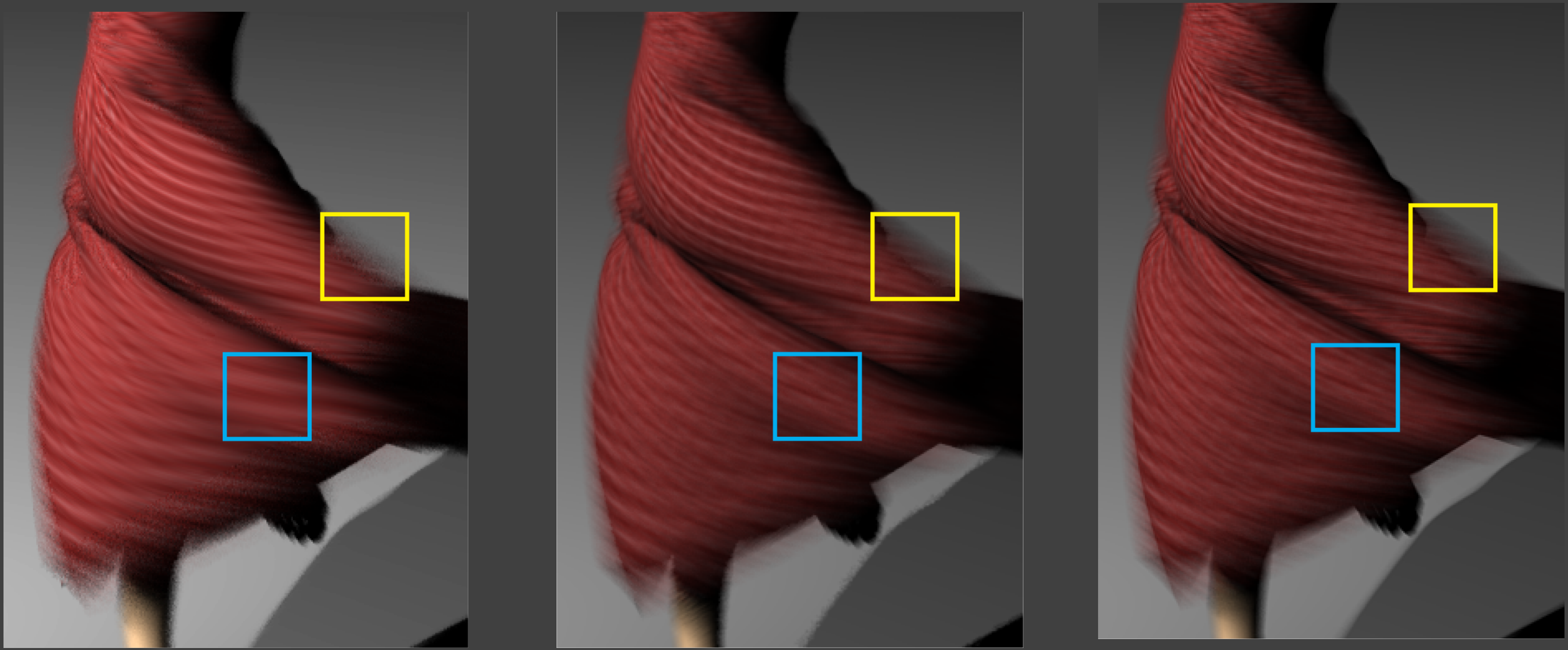
Input: 1 spp

Proposed result:
1 spp -> 128 spp

Reference 256 spp
(256x time)

Our reconstruction

Comparison to Egan et al. [2009]

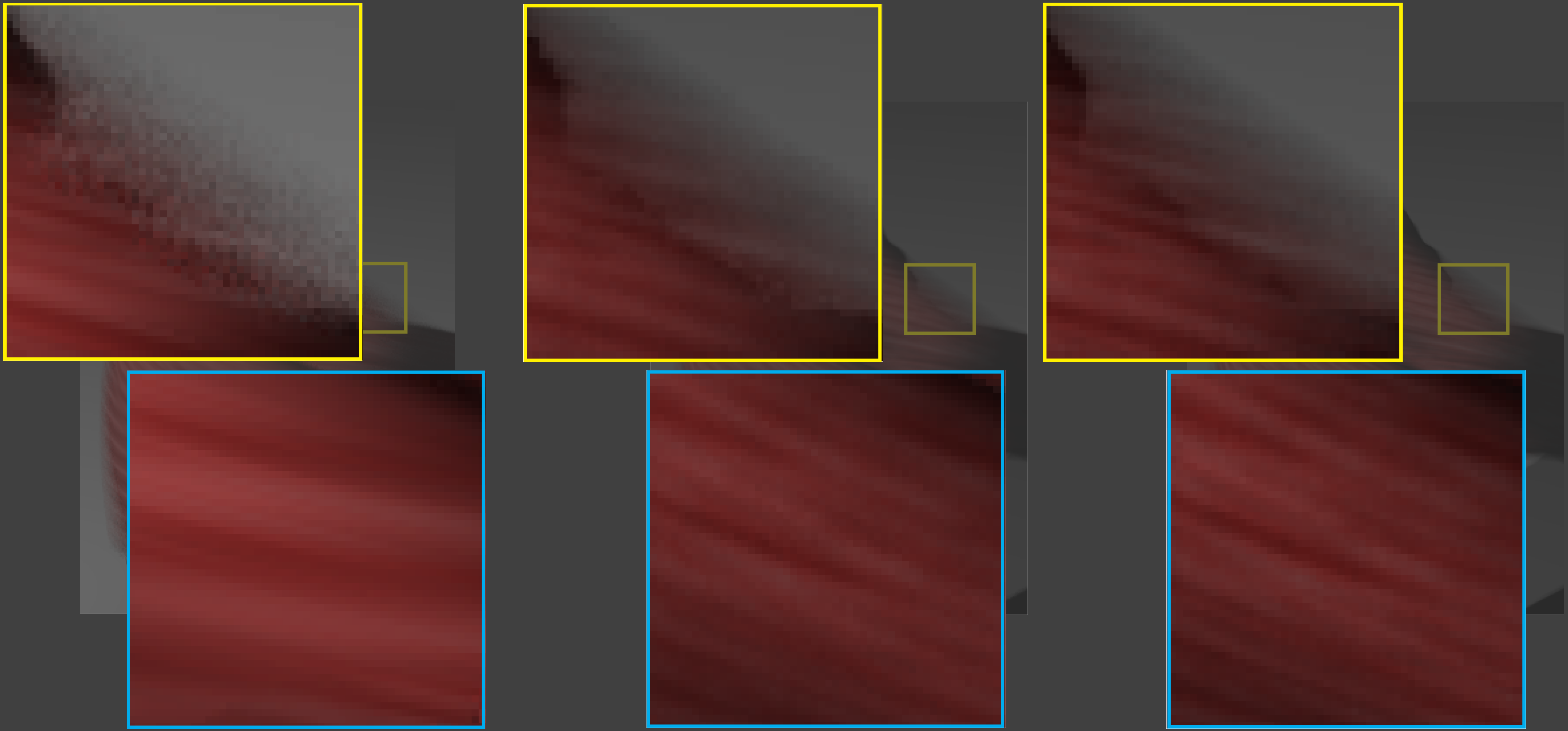


Egan et al. [2009]
8 samples / pixel

Proposed method
4 samples / pixel

Reference
256 samples / pixel

Comparison to Egan et al. [2009]

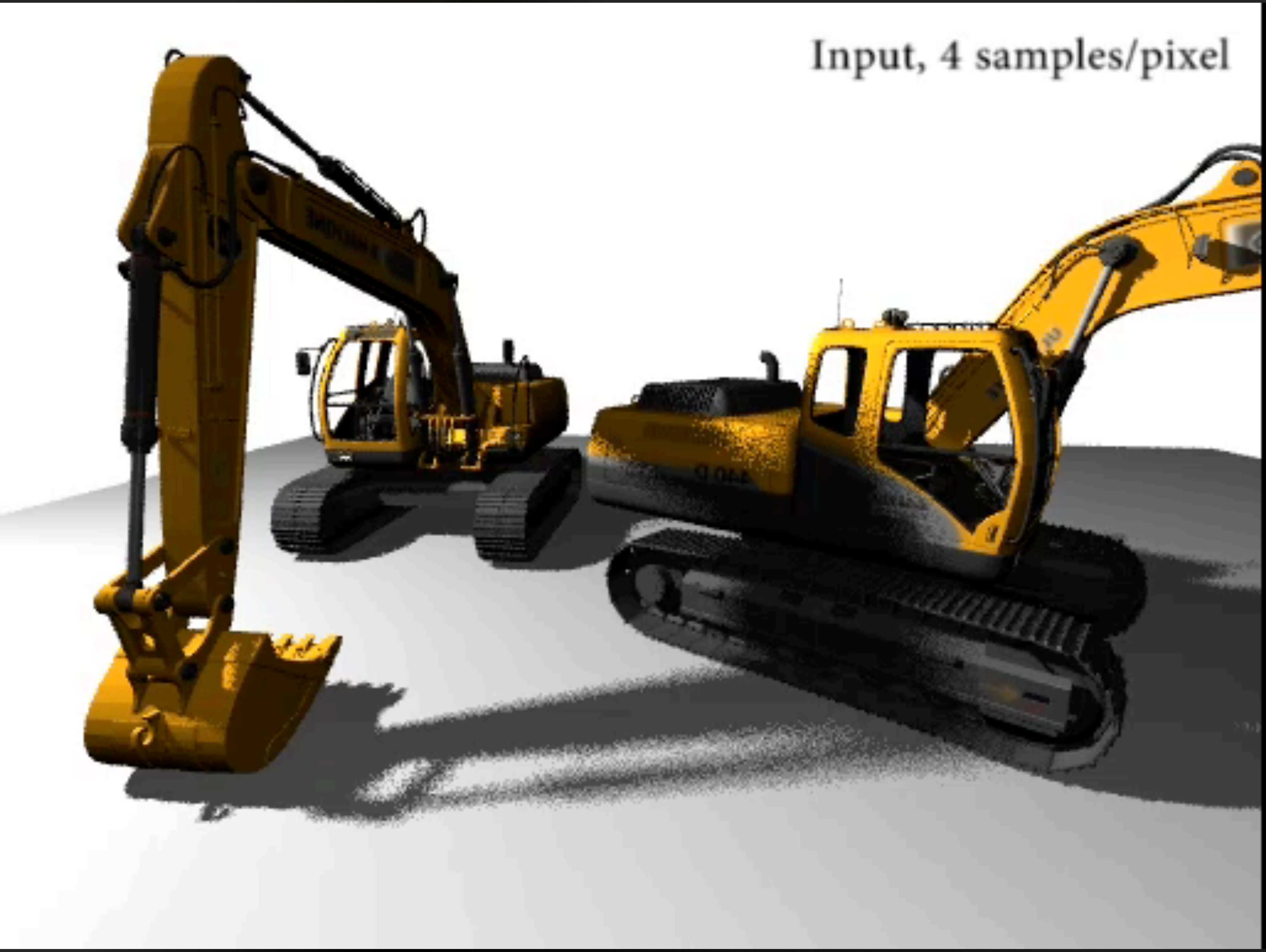


Egan et al. [2009]
8 samples / pixel

Our method
4 samples / pixel

Reference
256 samples / pixel

Soft shadows, 4 spp



7D soft shadows with motion and defocus, 4 spp

Acknowledgments

Thanks to everyone below for making the slides available online.

Fredo Durand and colleagues [Frequency analysis of light transport 2005]

Toshiya Hachisuka and colleagues [Multi-dimensional adaptive sampling and reconstruction for ray tracing 2008]

Kevin Egan and colleagues [Frequency Analysis and Sheared Reconstruction for Rendering Motion Blur 2009]

Jakko Lehtinen and colleagues [Temporal Light Field Reconstruction for Rendering Distribution Effects 2011]