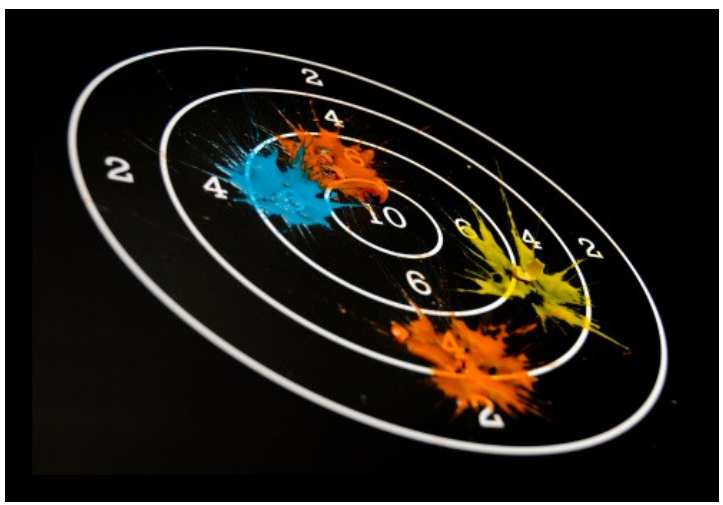
# Visualization, Lecture

## Volume Visualization: Splatting





Thomas Torsney-Weir
t.d.torsney-weir@swansea.ac.uk



Computer Science
Gwyddor Cyfrifiadur

# Splatting

Instead of asking which data samples contribute to a pixel value, ask, to which pixel values does a data sample contribute?

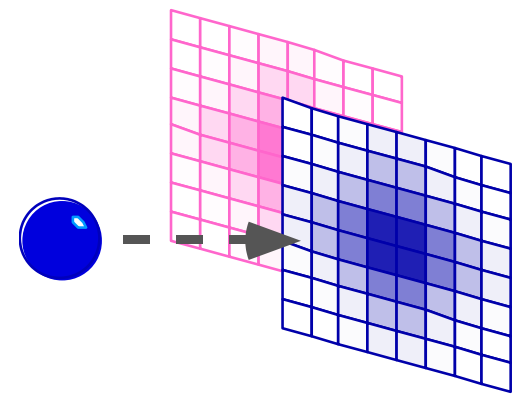
- Ray casting: pixel value computed from multiple data samples
- Splatting: multiple pixel values (partially) computed from a single data sample

Overview:

- object-order (FOR each voxel DO …)
- high-quality
- Needs to be parallelized

Idea: contribute every voxel to the image

- projection from voxel: splat
- composite in image space



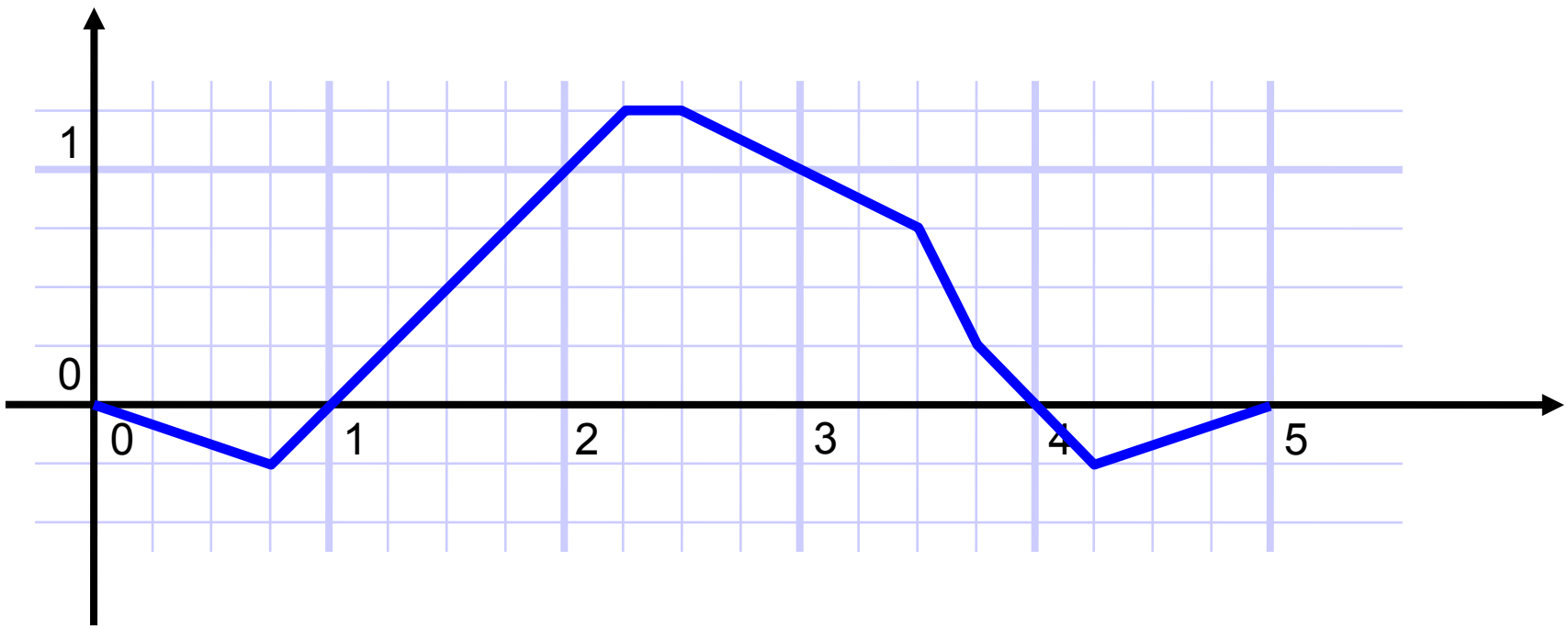
# Why?

- Marching cubes and DVR need grids
  - Marching cubes for isosurfacing
  - DVR for transfer function
- Some data are point clouds
- Generating the mesh can be difficult
- Storing the mesh can be costly

# Splatting and Reconstruction – Premise

Premise:

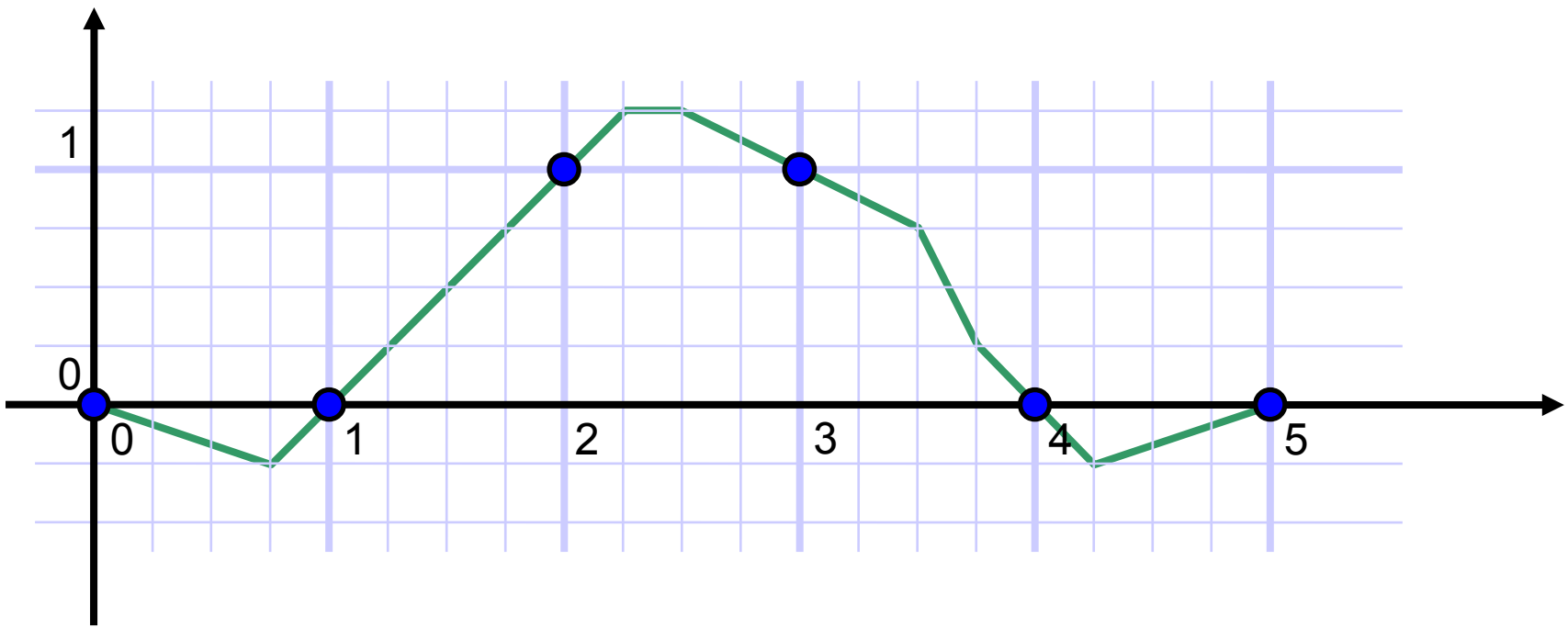
- before measurement: original distribution
  $\varphi: R^3 \rightarrow R$



# Splatting and Reconstruction – Premise

Premise:

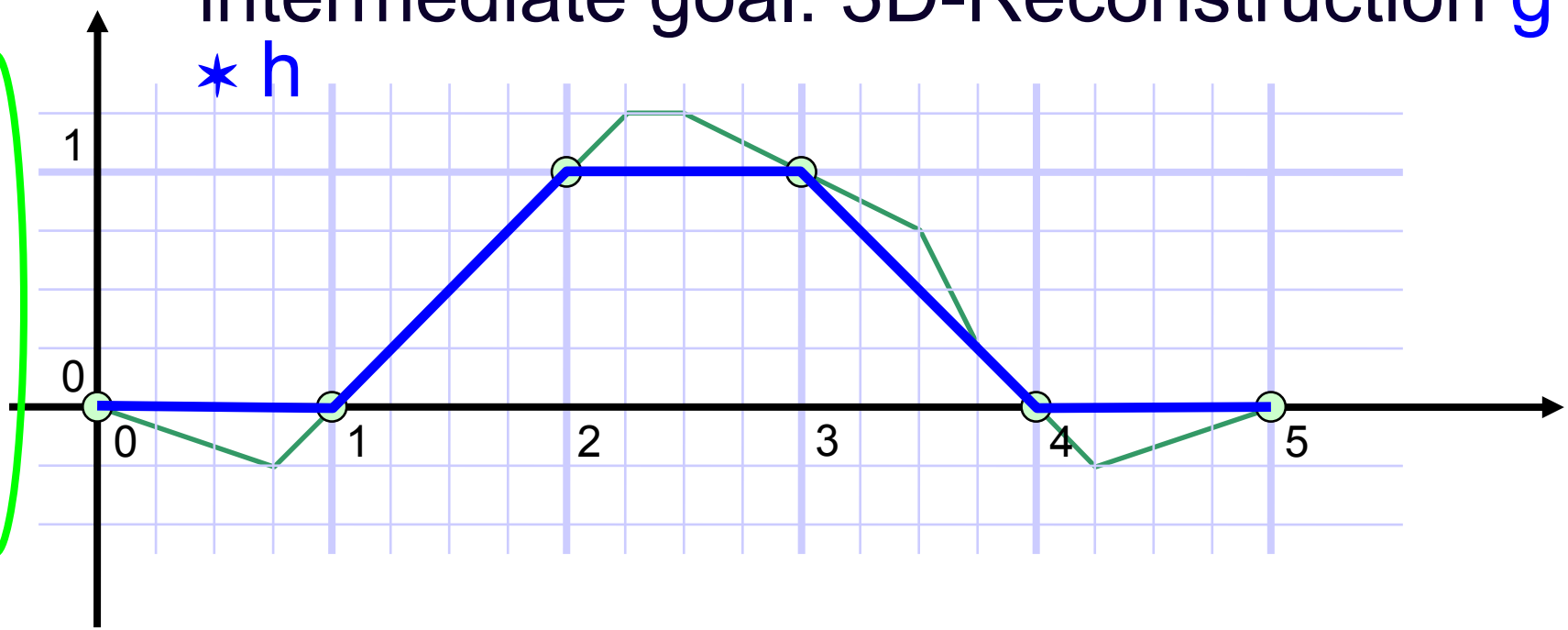
- before measurement: original distribution
  $\varphi: R^3 \rightarrow R$
- measurement = sampling: samples
- $f = \varphi \cdot comb_{3D}$



# Splatting and Reconstruction – Basics

Basic concepts, idea:

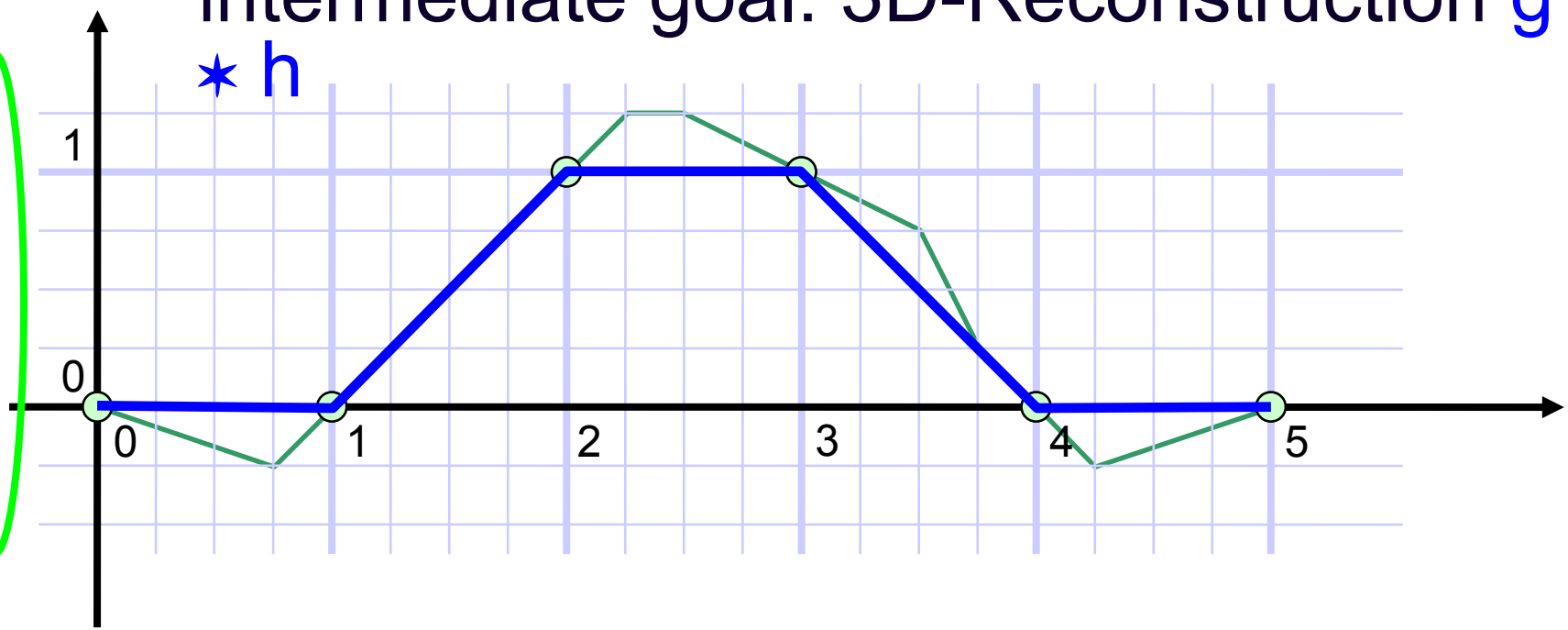
- before measurement: original distribution $\varphi: R^3 \rightarrow R$
- measurement = sampling: samples $f = \varphi \cdot comb_{3D}$
- intermediate goal: 3D-Reconstruction $g = f * h$



# Splatting and Reconstruction – Basics

Basic concepts, idea:

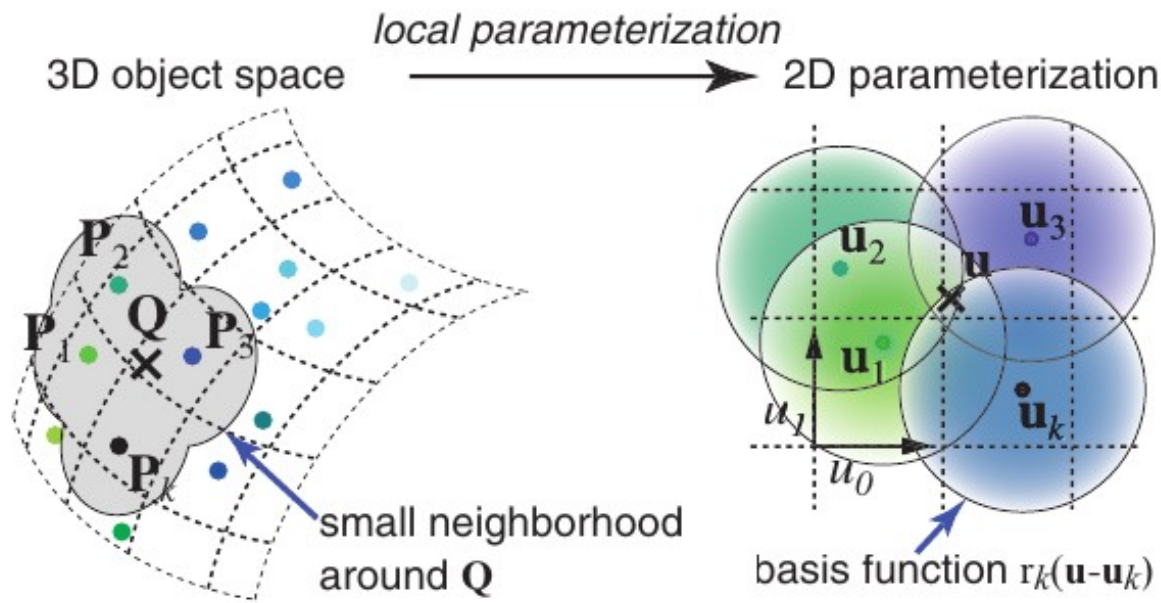
- before measurement: original distribution $\varphi: R^3 \rightarrow R$
- measurement = sampling: samples $f = \varphi \cdot comb_{3D}$
- intermediate goal: 3D-Reconstruction $g = f * h$



# Splatting and Reconstruction – Basics

Basic concepts, idea:

- intermediate goal: 3D-Reconstruction $g = f * h$
- $h$ is the kernel or basis function




Zwicker, Matthias, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. "Surface Splatting." In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 371–378. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001. https://doi.org/10.1145/383259.383300.

# Splatting and Reconstruction – Basics

Basic concepts, idea:

- intermediate goal: 3D-Reconstruction $g = f * h$
- $h$ is the kernel or basis function
- Computes the function (for each pixel)

$$g(\vec{x}) = \sum_{i=1}^{N} h_i(\vec{x})\,\alpha_i$$

Zwicker, Matthias, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. “Surface Splatting.” In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 371–378. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 2001. https://doi.org/10.1145/383259.383300.

# Splatting and Reconstruction – Basics

Basic concepts, idea:

- intermediate goal: 3D-Reconstruction g = f $\ast$ h
- h is the kernel or basis function
- Computes the function (for each pixel)

$$g(\vec{x}) = \sum_{i=1}^{N} h_i(\vec{x})\,\alpha_i$$

Final pixel value

Zwicker, Matthias, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. “Surface Splatting.” In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 371–378. SIGGRAPH ’01. New York, NY, USA: Association for Computing Machinery, 2001. https://doi.org/10.1145/383259.383300.

# Splatting and Reconstruction – Basics

Basic concepts, idea:

- intermediate goal: 3D-Reconstruction $g = f * h$
- $h$ is the kernel or basis function
- Computes the function (for each pixel)

$$g(\vec{x}) = \sum_{i=1}^{N} h_i(\vec{x})\,\alpha_i$$

Number of basis functions (number of samples)

Zwicker, Matthias, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. "Surface Splatting." In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 371–378. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001. https://doi.org/10.1145/383259.383300.

# Splatting and Reconstruction – Basics

Basic concepts, idea:

- intermediate goal: 3D-Reconstruction $g = f * h$
- $h$ is the kernel or basis function
- Computes the function (for each pixel)

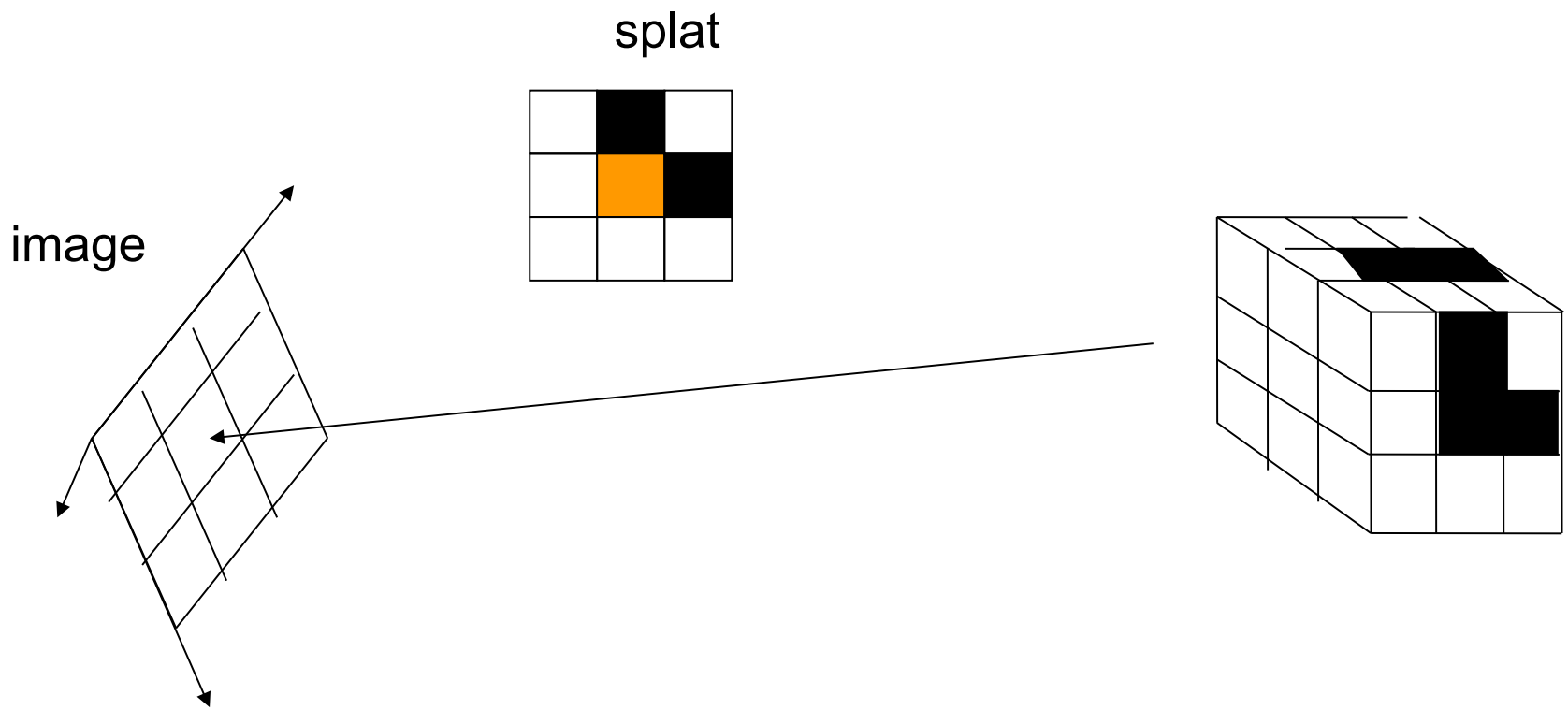
$$g(\vec{x}) = \sum_{i=1}^{N} h_i(\vec{x})\,\alpha_i$$

Effect of sample on pixel

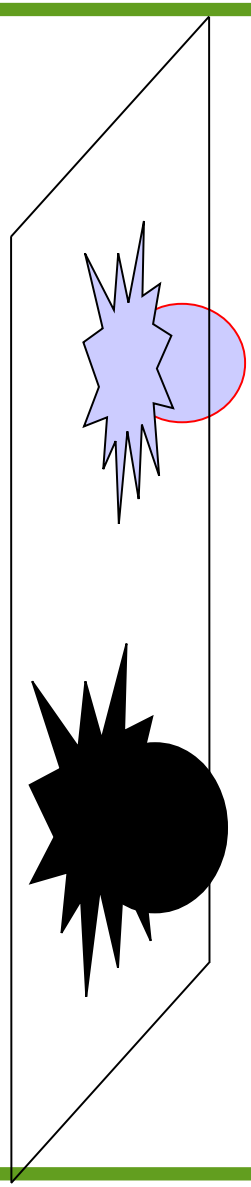
Zwicker, Matthias, Hanspeter Pfister, Jeroen van Baar, and Markus Gross. "Surface Splatting." In Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, 371–378. SIGGRAPH '01. New York, NY, USA: Association for Computing Machinery, 2001. https://doi.org/10.1145/383259.383300.

# Splatting

- Project each sample (voxel) from the volume into the image plane



# Splatting



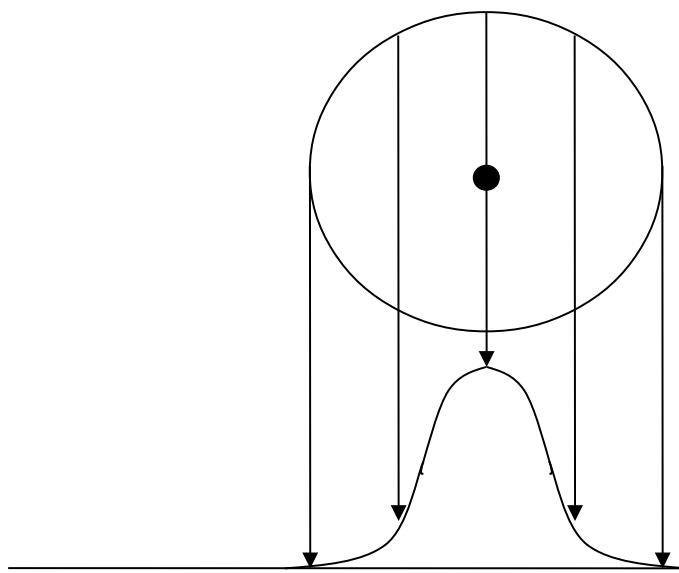
# Splatting

- Discretization via 2D splats

$$\text{Splat}(x, y) = \int w(x, y, z)\, dz$$

from the original 3D kernel

- The 3D rotationally symmetric filter kernel is integrated to produce a 2D filter kernel



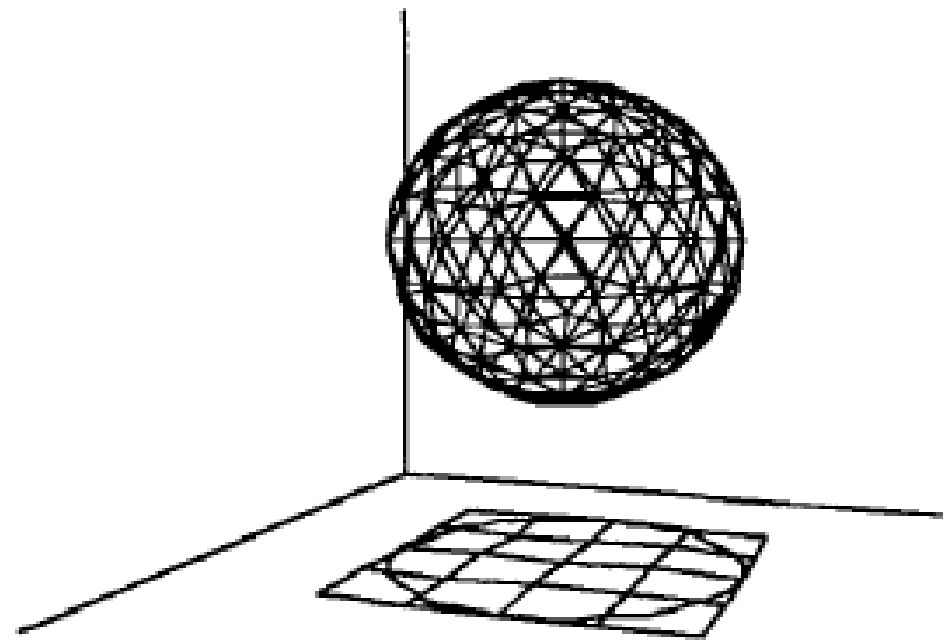
3D filter kernel

Integrate along one dimension

2D filter kernel

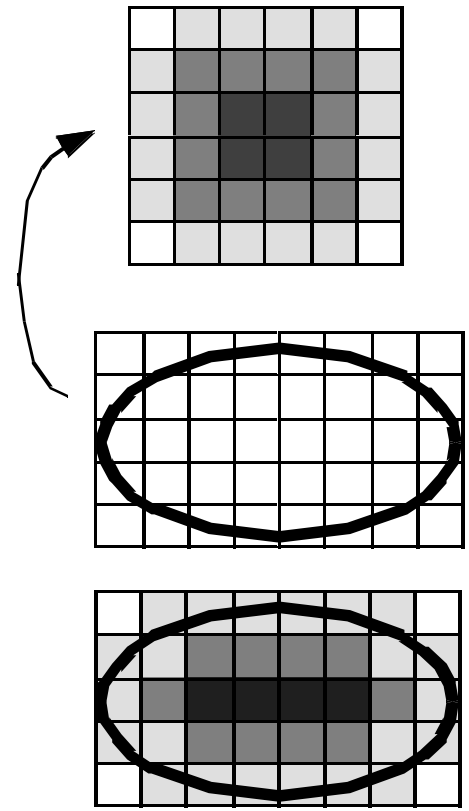
# Splatting

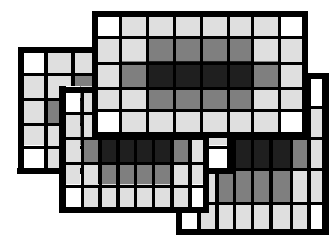
- Draw each voxel as a cloud of points (footprint) that spreads the voxel contribution across multiple pixels
- Footprint: splatted (integrated) kernel
- Approximate the 3D kernel $\boldsymbol{h}(x,y,z)$ extent by a sphere



# Splatting

- Larger footprint increases blurring and used for high pixel-to-voxel ratio
- Footprint geometry
  - Orthographic projection: footprint is independent of the view point
  - Perspective projection: footprint is elliptical
- Pre-integration of footprint
- For perspective projection: additional computation of the orientation of the ellipse

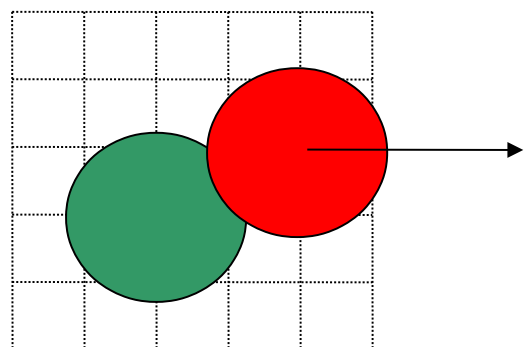


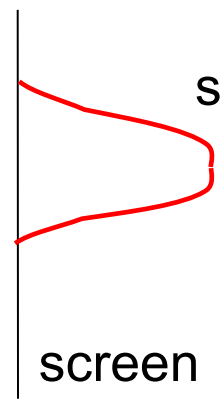


# Splatting

- Volume = field of 3D interpolation kernels
  - One kernel at each grid voxel
- Each kernel leaves a 2D footprint on screen
- Weighted footprints accumulate into image

voxel kernels





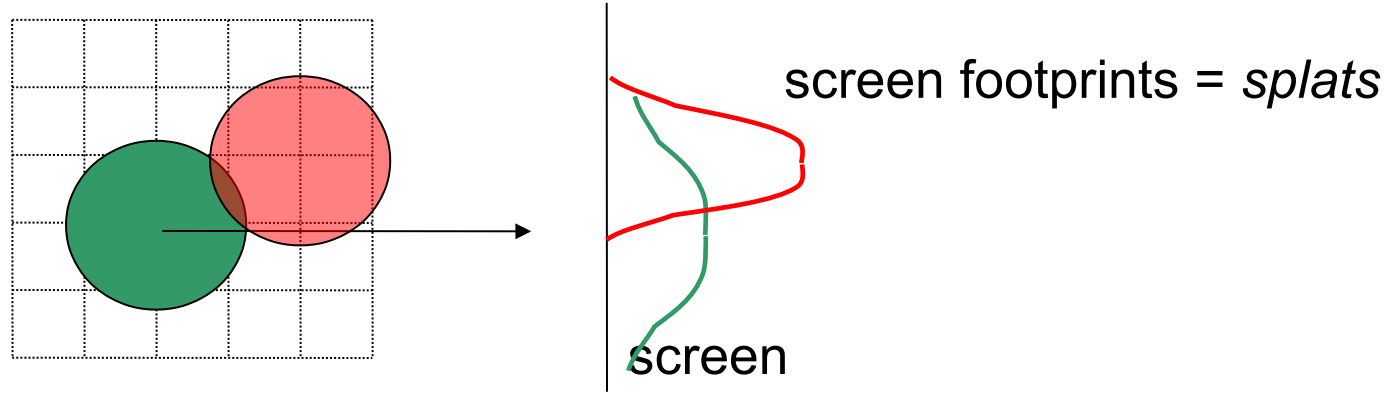
screen footprints = *splats*

screen

# Splatting

- Volume = field of 3D interpolation kernels
  - One kernel at each grid voxel
- Each kernel leaves a 2D footprint on screen
- Weighted footprints accumulate into image

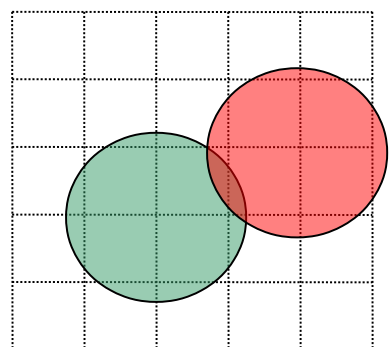
voxel kernels

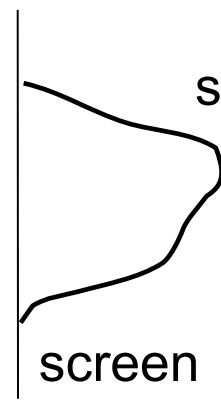


# Splatting

- Volume = field of 3D interpolation kernels
  - One kernel at each grid voxel
- Each kernel leaves a 2D footprint on screen
- Weighted footprints accumulate into image

voxel kernels





screen footprints = *splats*

screen

# Splatting

- Voxel kernels are added within sheets
- Sheets are composited front-to-back
- Sheets = volume slices most perpendicular to the image plane (analogously to stack of slices)

volume slices

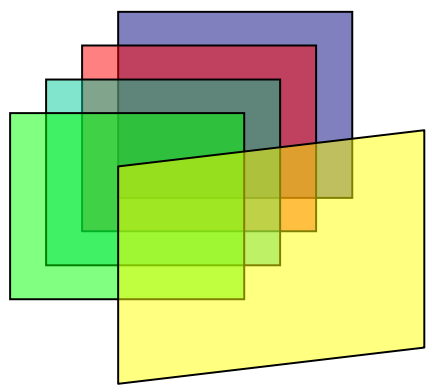
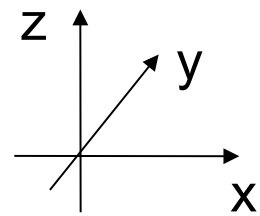


image plane at 30°




volume slices

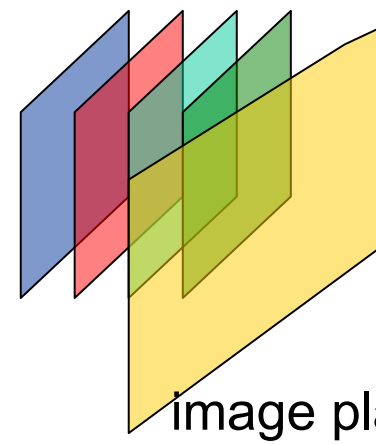
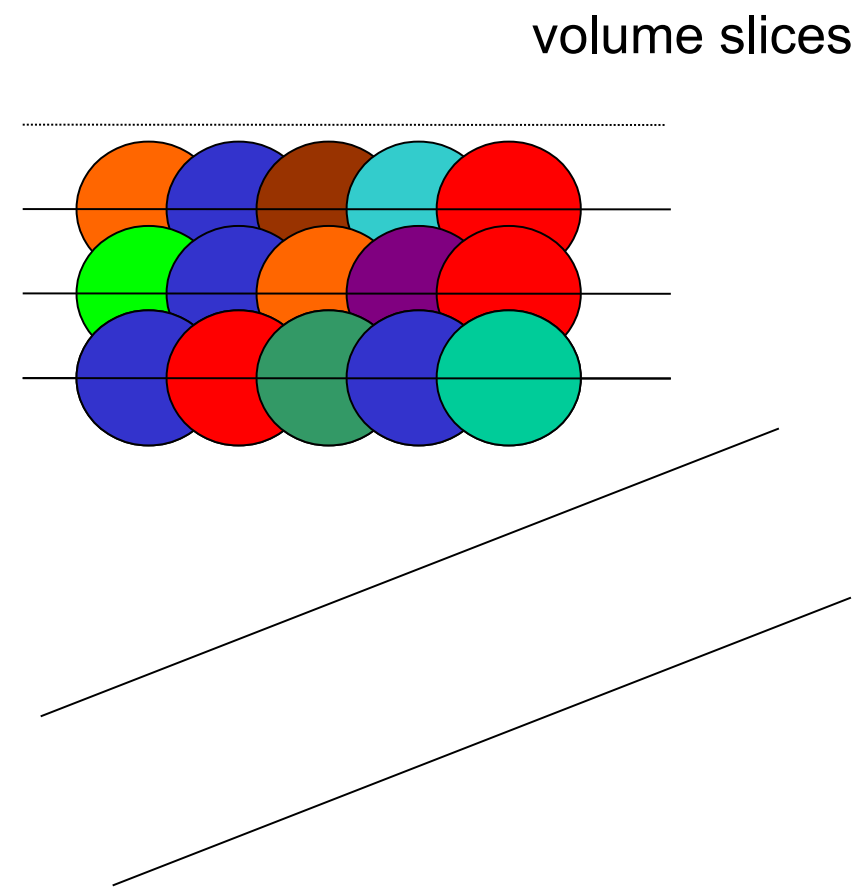


image plane at 70°

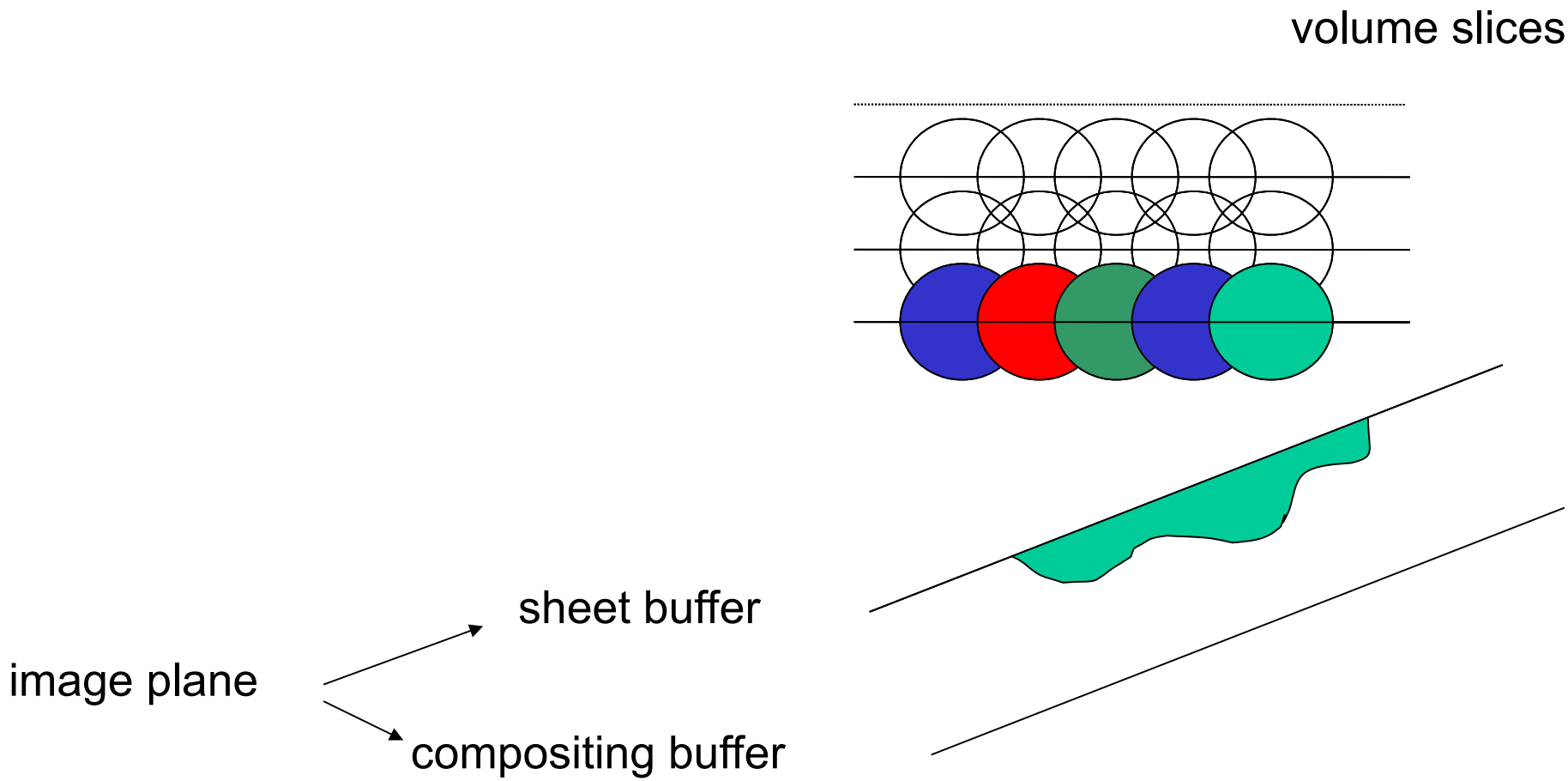
# Splatting

- Core algorithm for splatting
- Volume
  - Represented by voxels
  - Slicing
- Image plane:
  - Sheet buffer
  - Compositing buffer



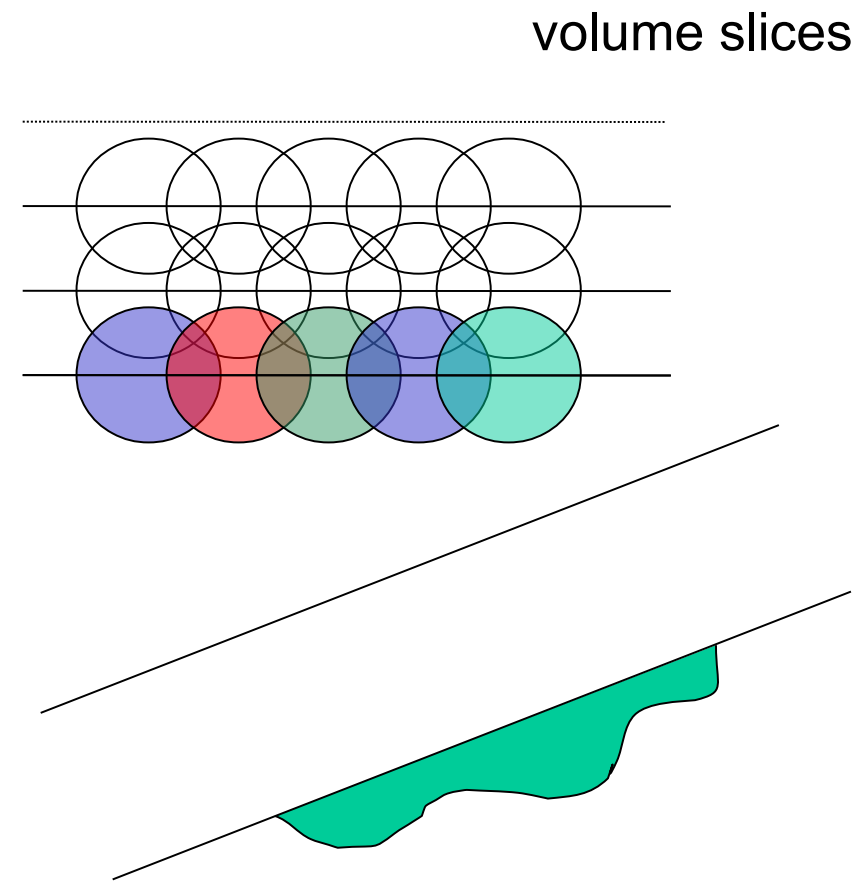
# Splatting

- Add voxel kernels within first sheet



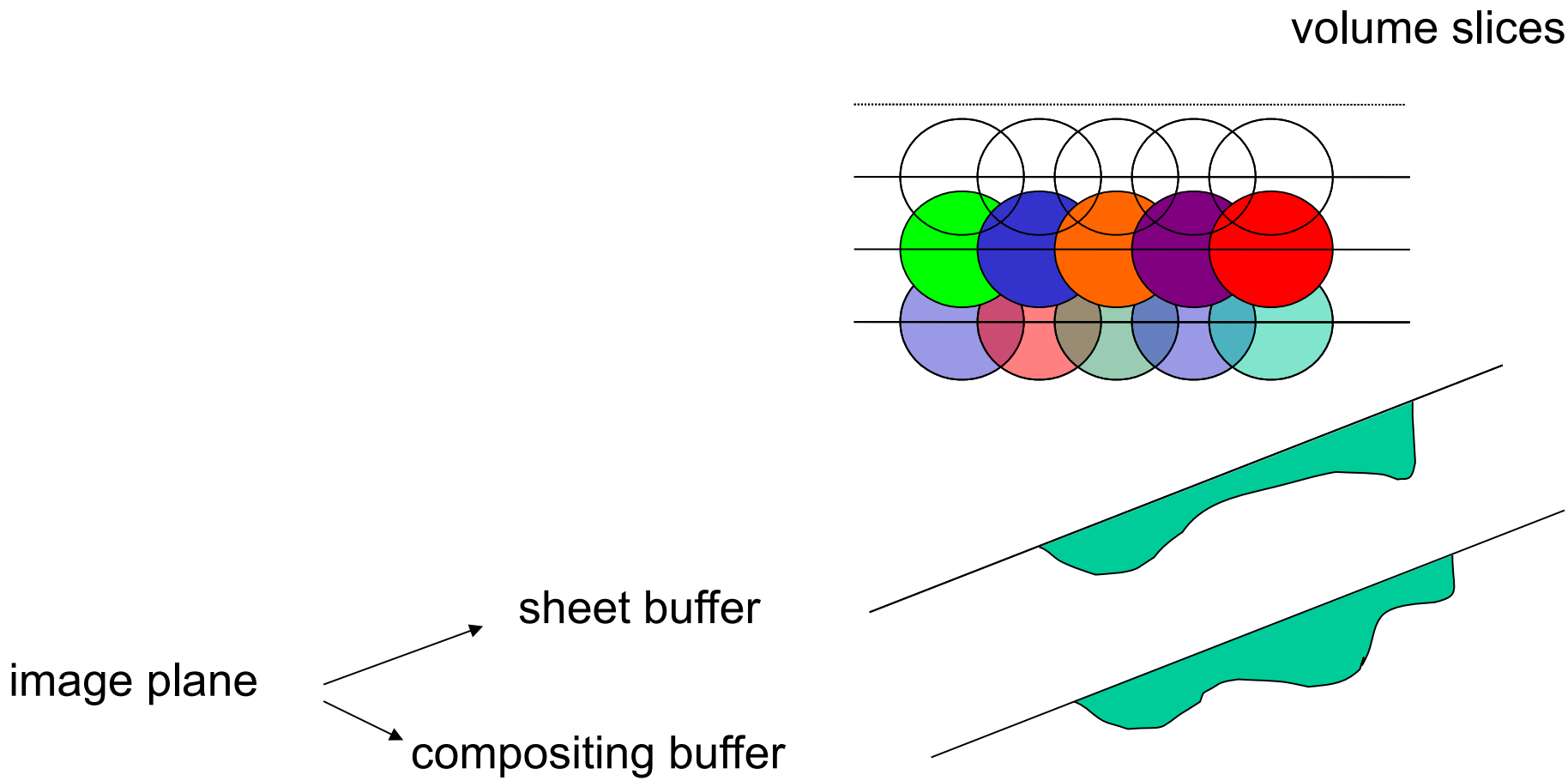
# Splatting

- Transfer to compositing buffer



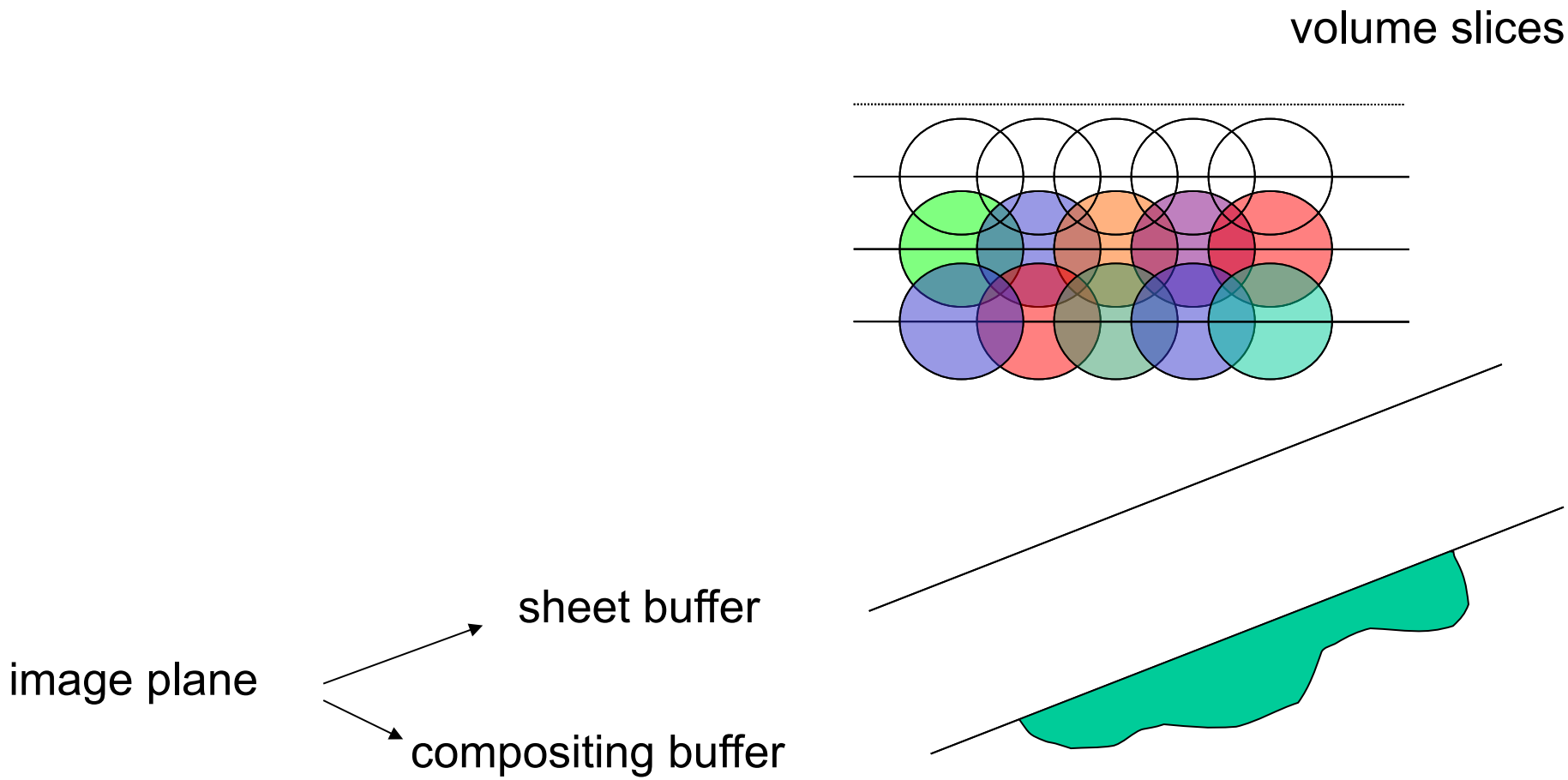
# Splatting

- Add voxel kernels within second sheet



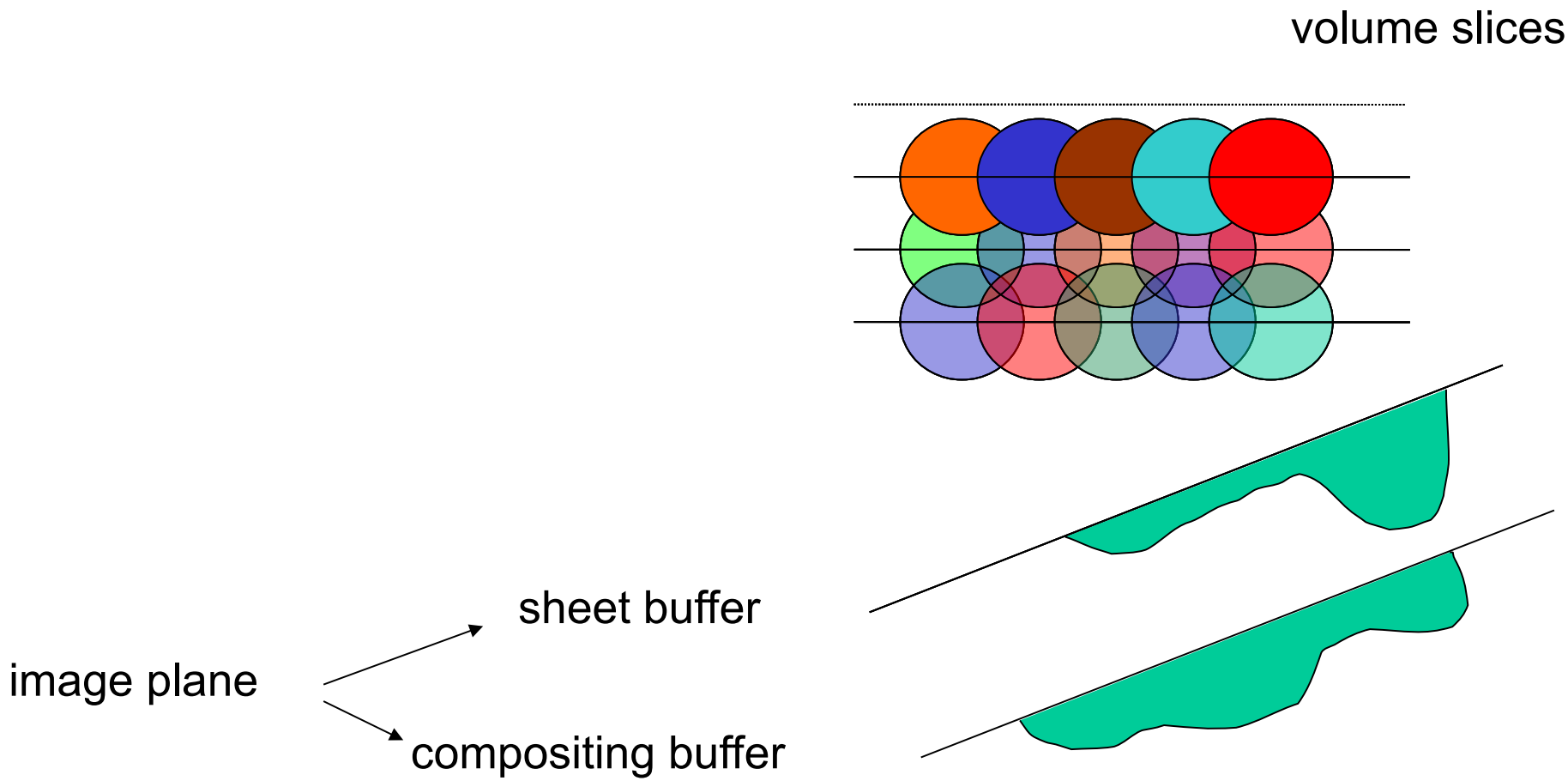
# Splatting

- Composite sheet with compositing buffer



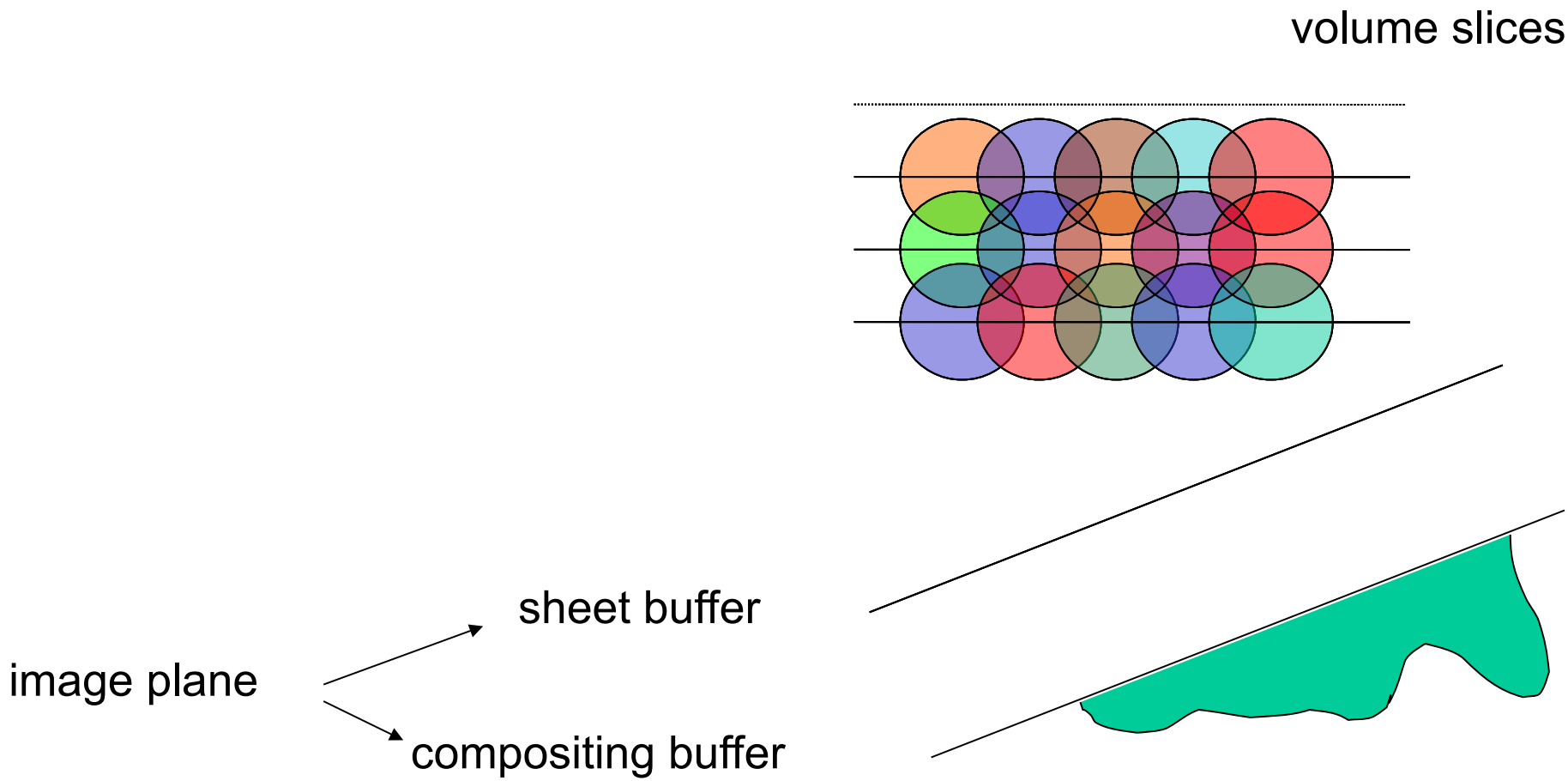
# Splatting

- Add voxel kernels within third sheet



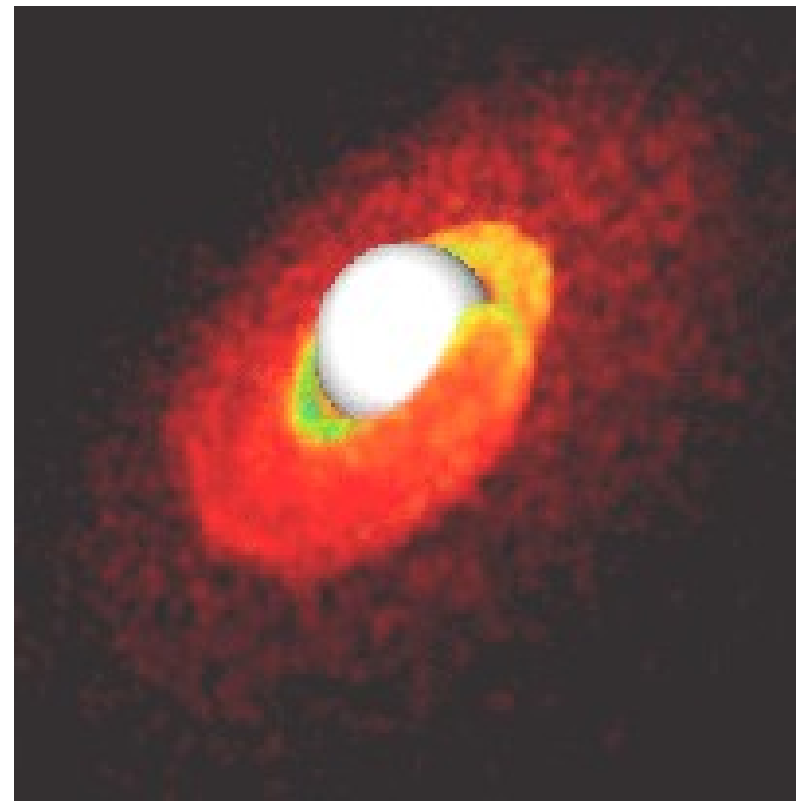
# Splatting

- Composite sheet with compositing buffer



# Splatting

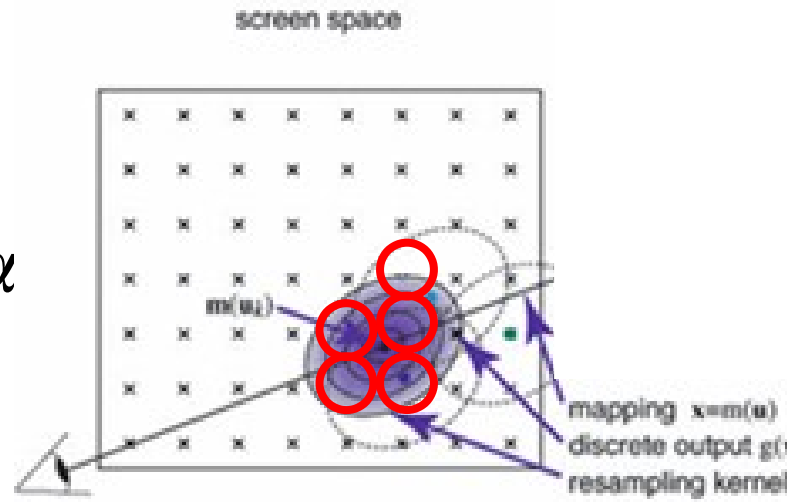
- Simple extension to volume data without grids
  - Scattered data with kernels
  - Example: SPH (smooth particle hydrodynamics)
  - Needs sorting of sample points

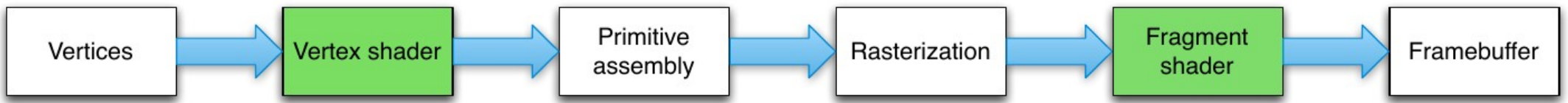


# Acceleration

$$g(\vec{x}) = \sum_{i=1}^{N} h_i(\vec{x})\,\alpha$$

Splat hits region?



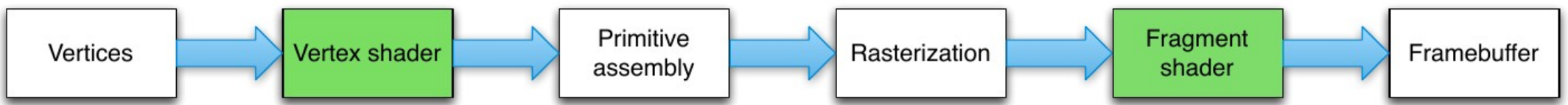

GPU pipeline

# Acceleration

$$g(\vec{x}) = \sum_{i=1}^{N} h_i(\vec{x})\,\alpha_i$$

Filter influential samples

Render splat




GPU pipeline

# Splatting – Conclusion

Pros:

- high-quality
- easy to parallelize
- Works for unstructured data
- perspective projection possible
- adaptive rendering possible

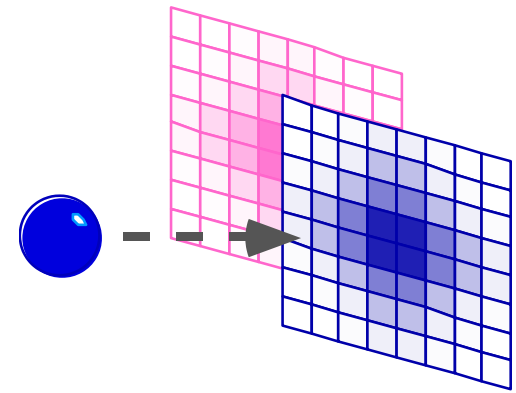
Cons:

- DVR is usually faster
- yields somewhat blurry images (in original)

# Splatting vs Ray Casting

**Splatting:**

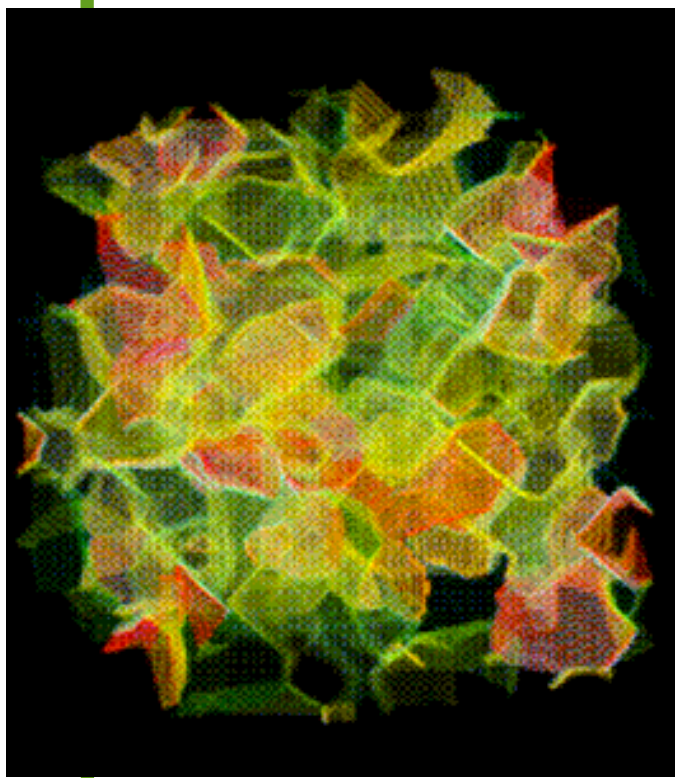
- Object-order: FOR each voxel (x,y,z) DO
  - sample volume at (x,y,z) using filter kernel
  - project reconstruction result to x-y image plane (leaving footprint)
- FOR each pixel (x,y) DO:
    - composite (color, opacity) result of all footprints

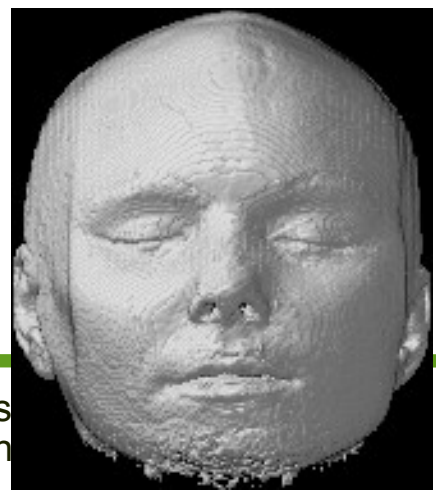


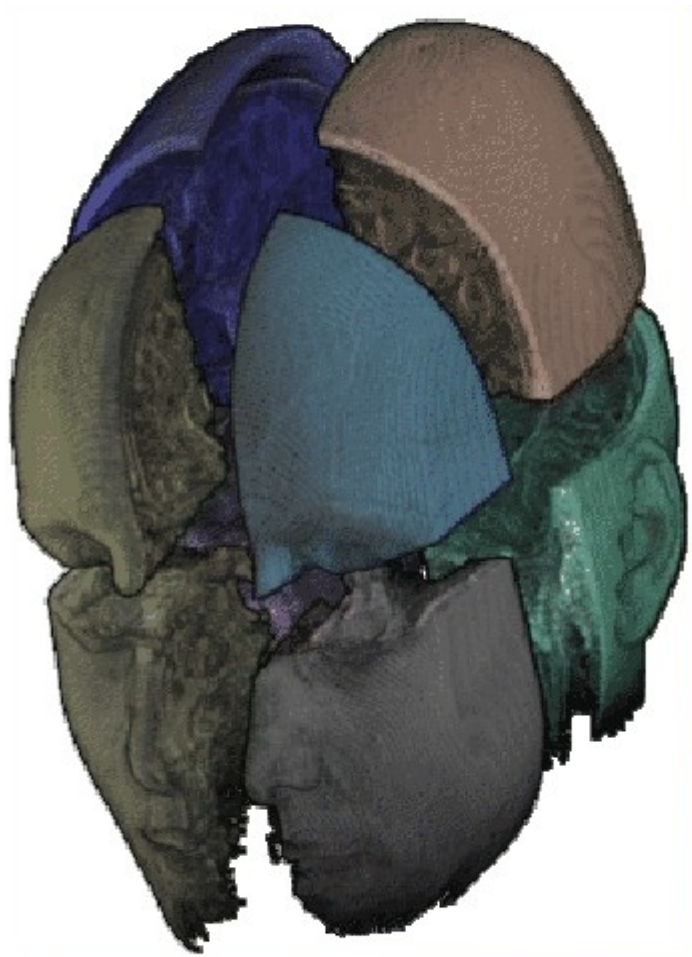
**Ray Casting:**

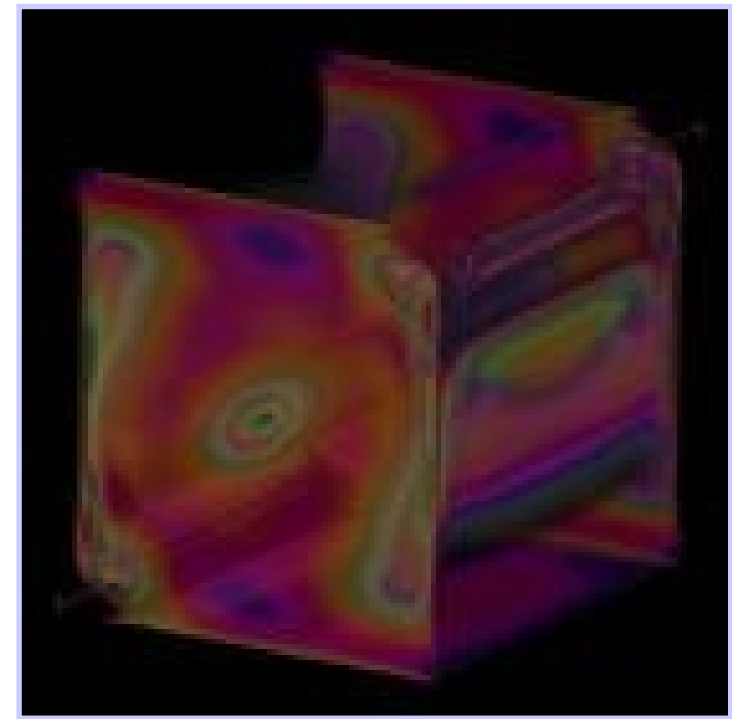
- Image-order: FOR each pixel (x,y) DO
  - cast ray into volume
  - FOR each sample point along ray (x,y,z)
    - Sample volume at (x,y,z) using filter kernel
    - composite (color, opacity) in image space at pixel (x,y)

# Splatting – Images









# Further Reading

For more details, please see,

**Data Visualization, Principles and Practice, Chapter 9, Image Visualization**, by A Telea, AK Peters 2008

And

**Footprint Evaluation for Volume Rendering**, by Lee Westover, in *ACM Computer Graphics* Volume 24, Number 4, August 1990, pages, 367-376

# Acknowledgements

We thank the following people for the lecture material:

- Robert S. Laramee
- Roberto Scopigno, Claudio Montani
- Peggy Li
- Roger Crawfis
- Hanspeter Pfister
- Philippe Lacroute
- Lukas Mroz
- Meister E. Gröller
- Torsten Möller
- Helwig Hauser
- Daniel Weiskopf