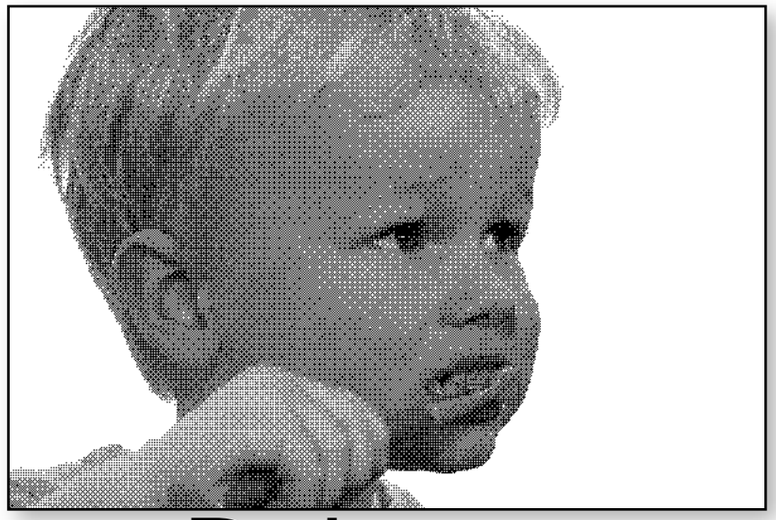
Image-Based Rendering

http://grail.cs.washington.edu/projects/slf/

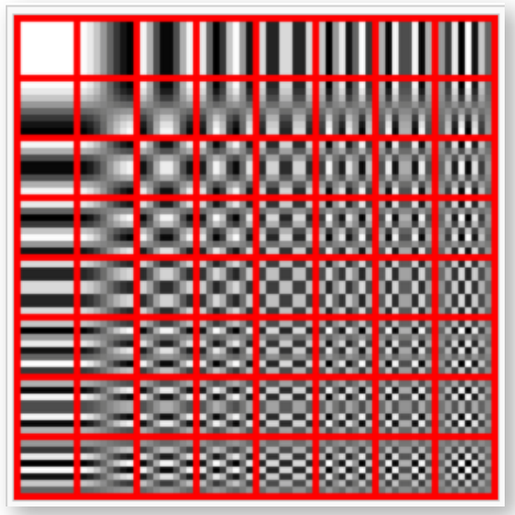
Adrien Treuille

Overview

Announcements



Dithering



Compression

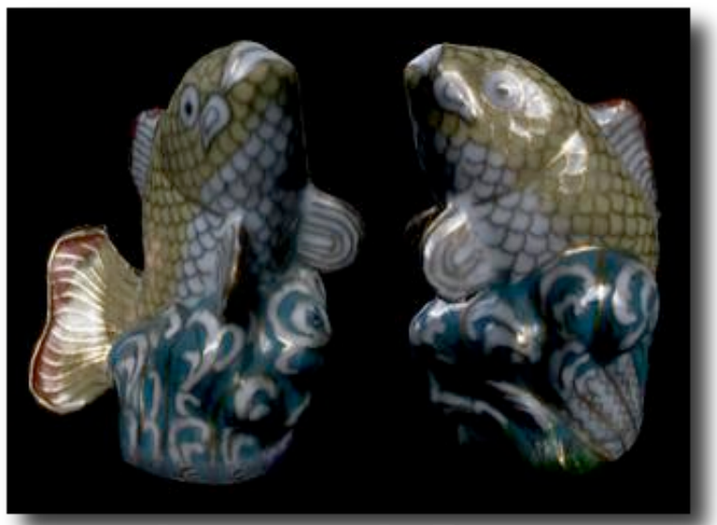
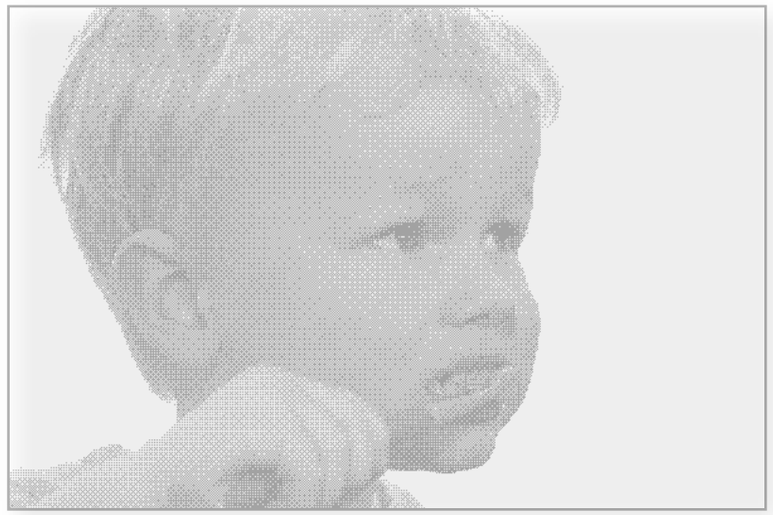


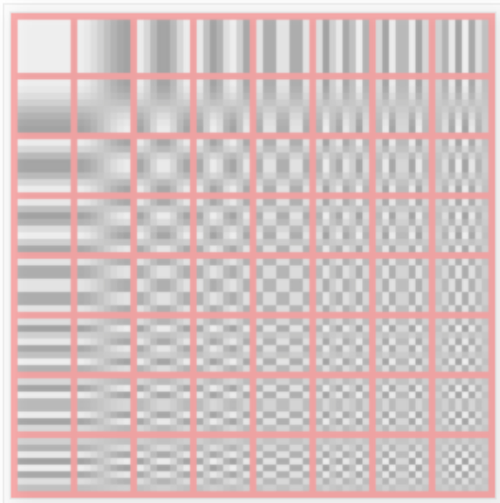
Image-based Rendering

Overview

Announcements



Dithering



Compression

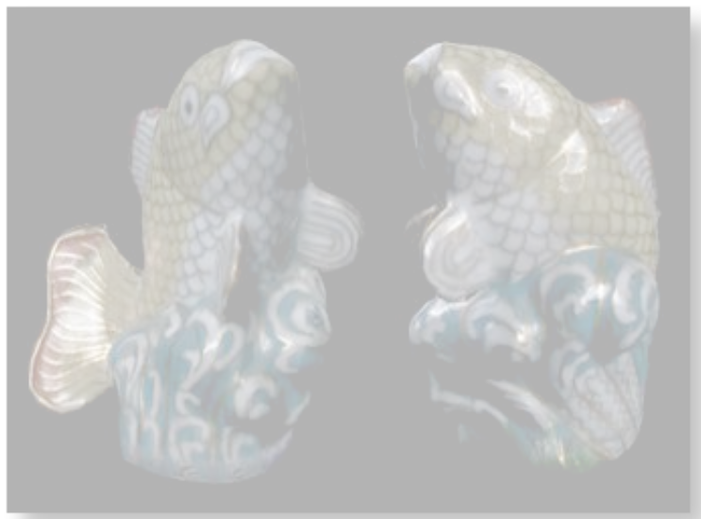
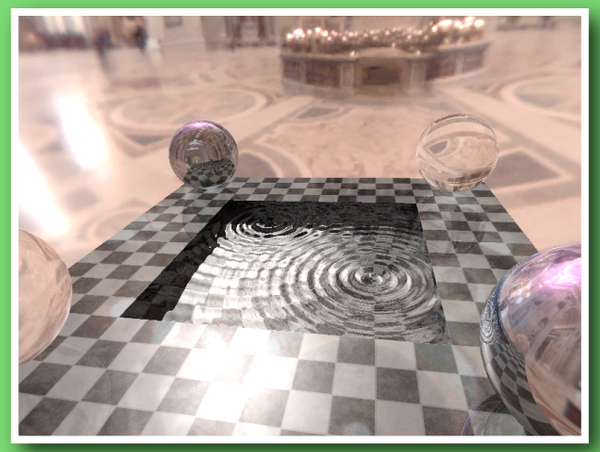


Image-based Rendering

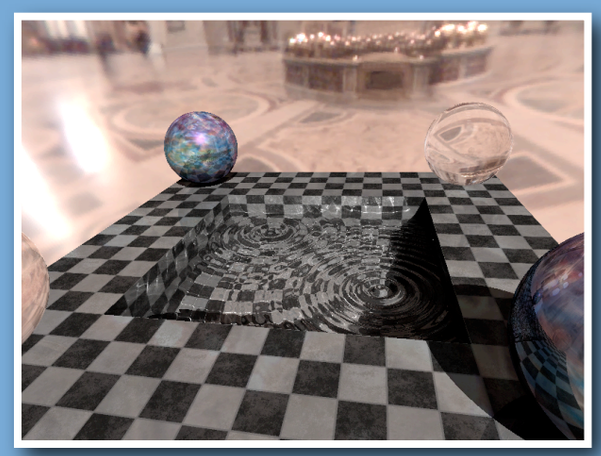
Project 3&4 Grading



Project 3

- Requirement A
- Requirement B
- Requirement C

50% BACK



Project 4

- Requirement A
- Requirement B
- Requirement C
- Requirement D
- Requirement E
- Requirement F

Project 4 Competition

Top 4 Artifacts get an IPod Touch!

Artifact can be movie/image/anything else...

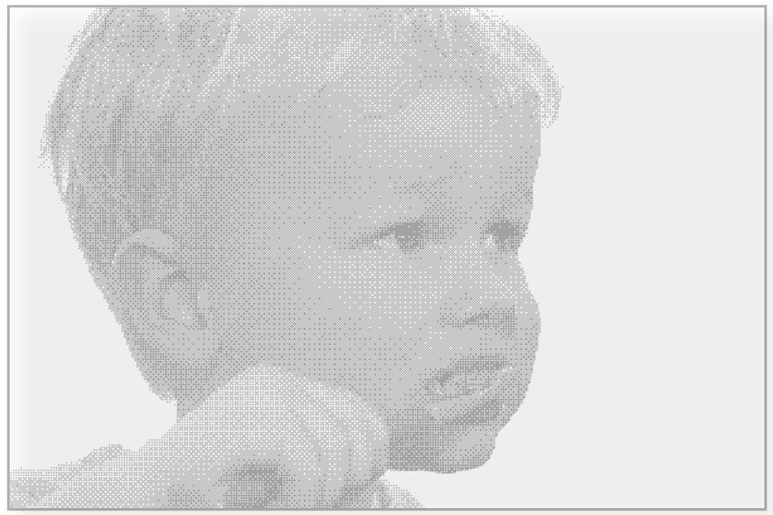
(decided by vote of TAs + Graphics Lab)

Thank you AMD...

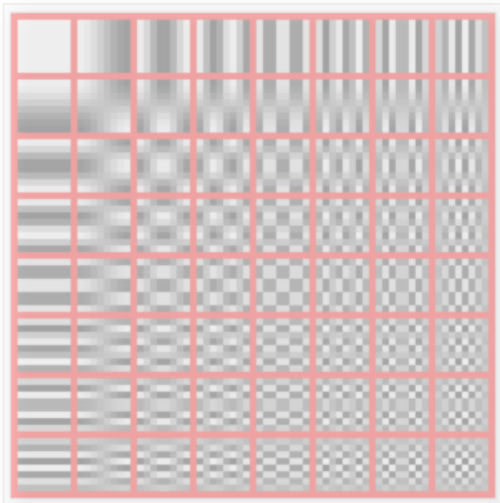
AMD

Overview

Announcements



Dithering



Compression

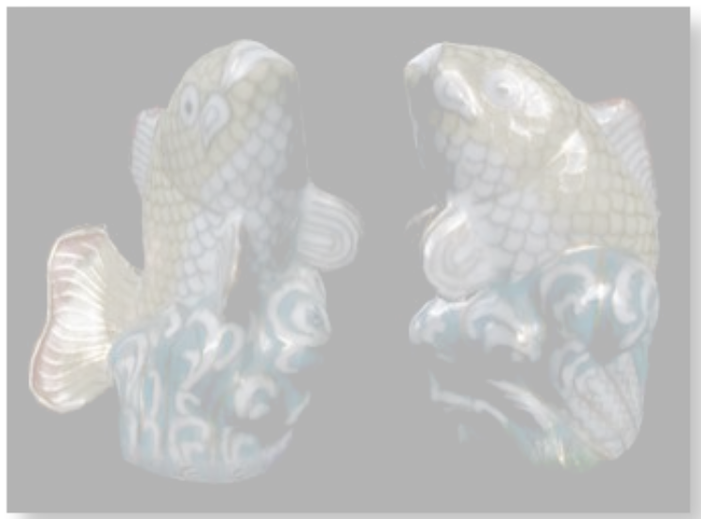
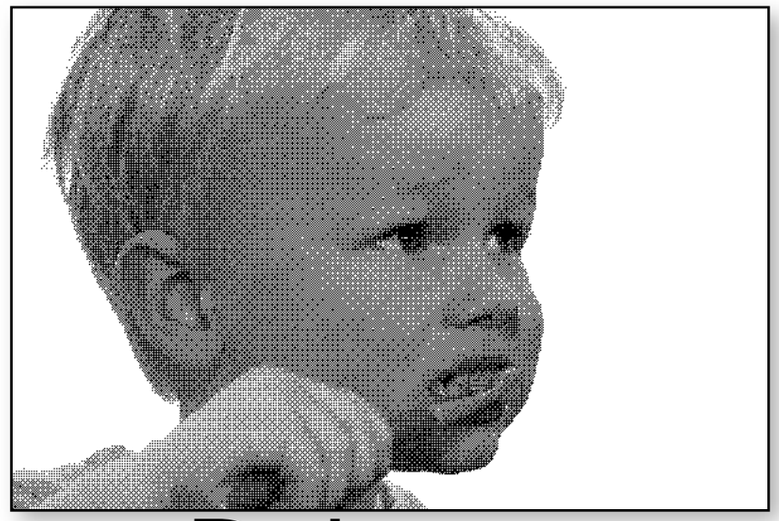


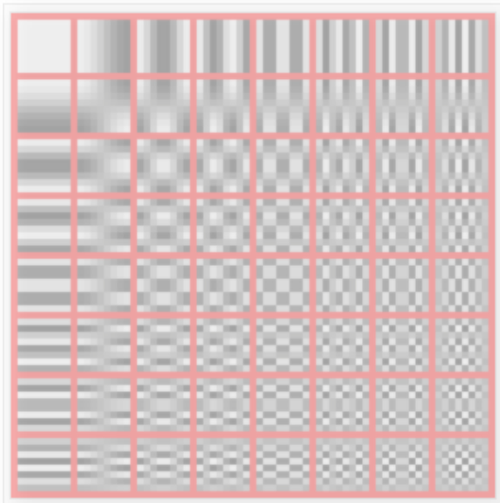
Image-based Rendering

Overview

Announcements



Dithering



Compression

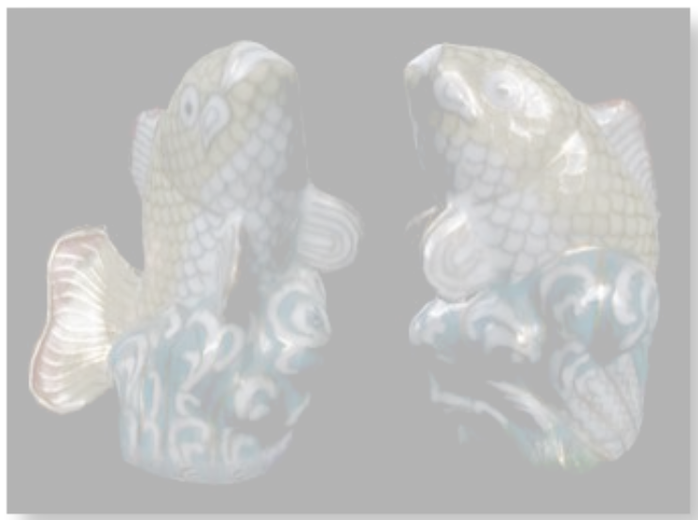
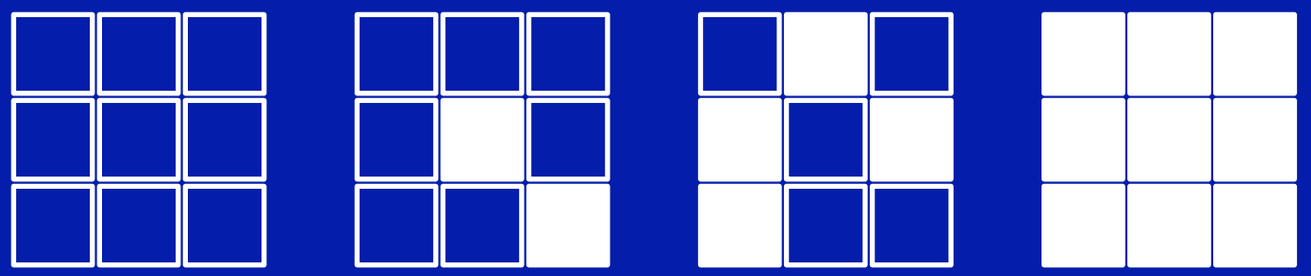


Image-based Rendering

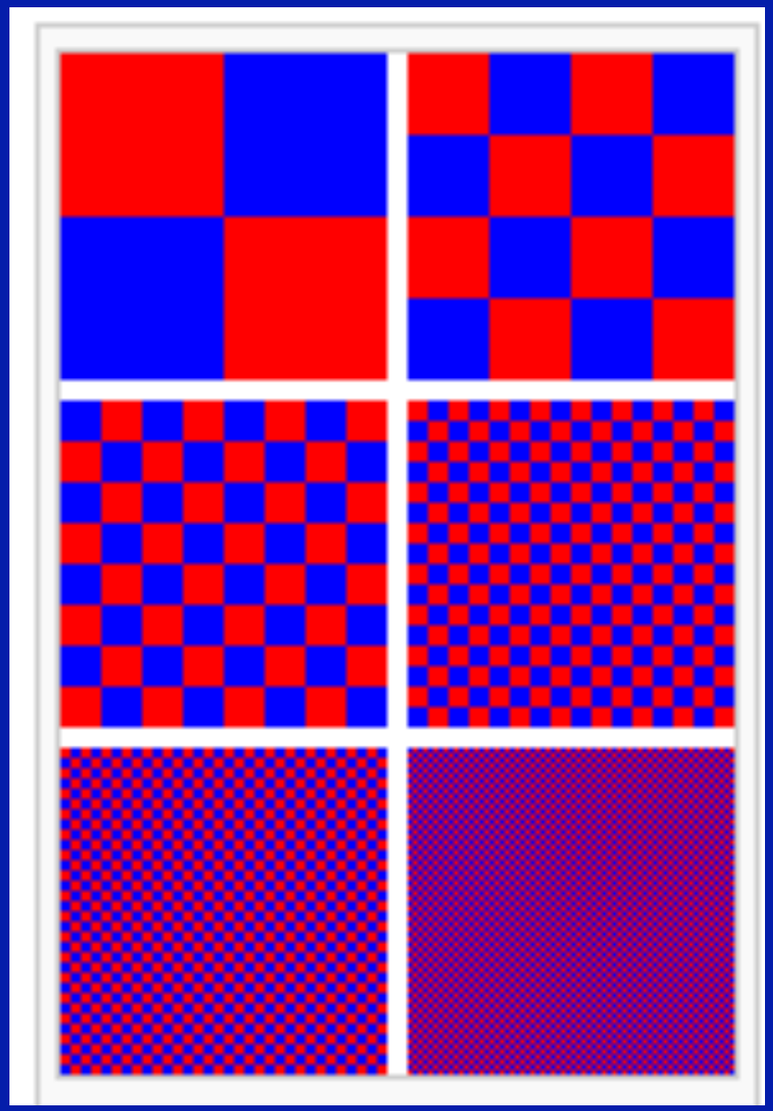
Dithering

- Compensates for lack of color resolution
- Eye does spatial averaging
- Black/white dithering to achieve gray scale
  - –Each pixel is black or white
  - –From far away, color determined by fraction of white
  - –For 3x3 block, 10 levels of gray scale



Dithering

Dithering takes advantage of the human eye's tendency to "mix" two colors in close proximity to one another.



Dithering

Dithering takes advantage of the human eye's tendency to "mix" two colors in close proximity to one another.

original

Colors = 2^{24}

no dithering

Colors = 2^{8}

with dithering

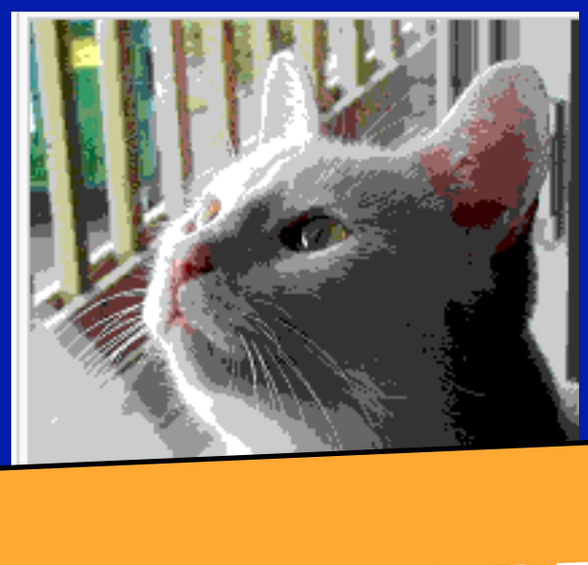
Colors = 2^{8}

Dithering

Dithering takes advantage of the human eye's tendency to "mix" two colors in close proximity to one another.

original

Colors = 2^{24}



with dithering

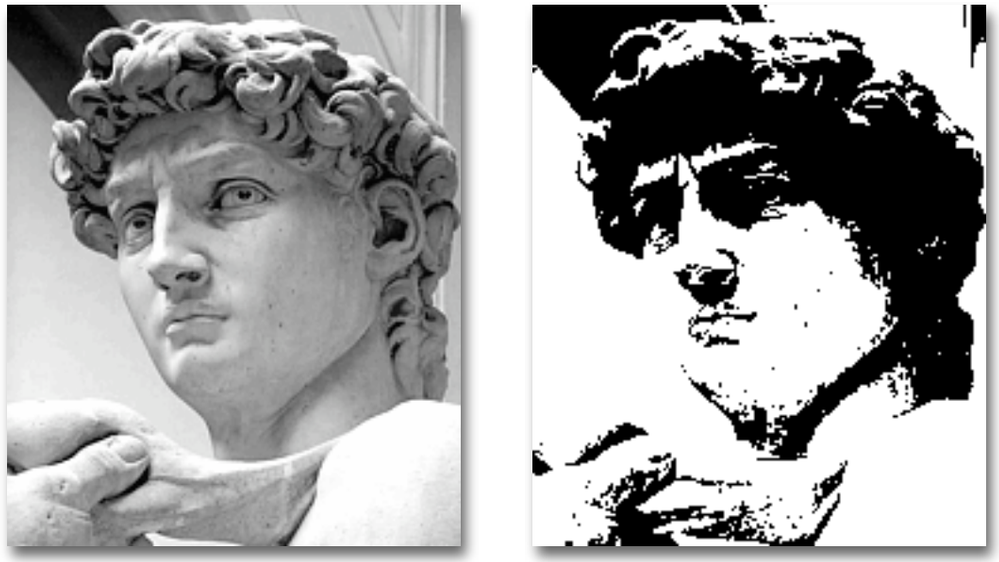
Colors = 2^{8}

How could we do this?

How Could We Do This?

- Deterministic Thresholding
- Random Thresholding
- Threshold Patterns
  - Dithering Matrices
- Diffusion

Deterministic Thresholding



source: http://en.wikipedia.org/wiki/Dither

Random Thresholding

source: http://en.wikipedia.org/wiki/Dither

Dithering Matrices

How do we select a good set of patterns?

Regular patterns create some artifacts

Example of good 3x3 dithering matrix

$$\begin{bmatrix} 6 & 8 & 4 \\ 1 & 0 & 3 \\ 5 & 2 & 7 \end{bmatrix}$$

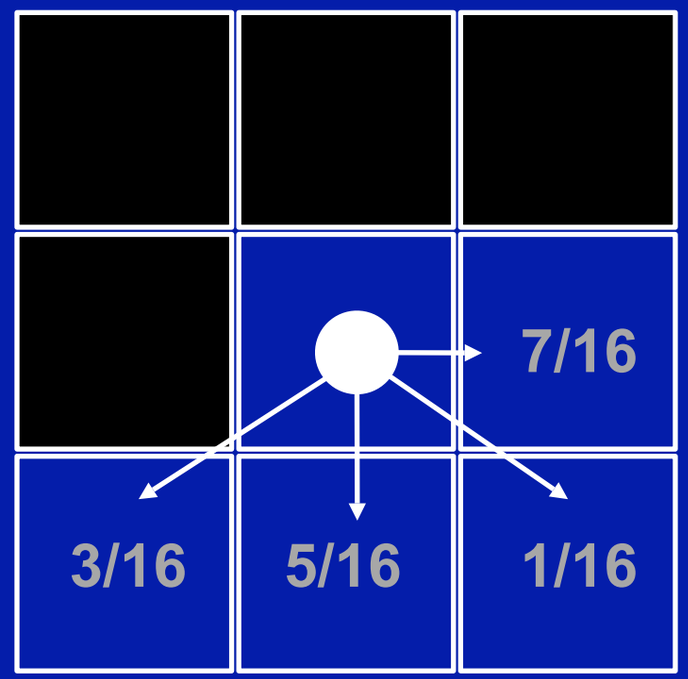
source: http://en.wikipedia.org/wiki/Dither

Floyd-Steinberg Error Diffusion

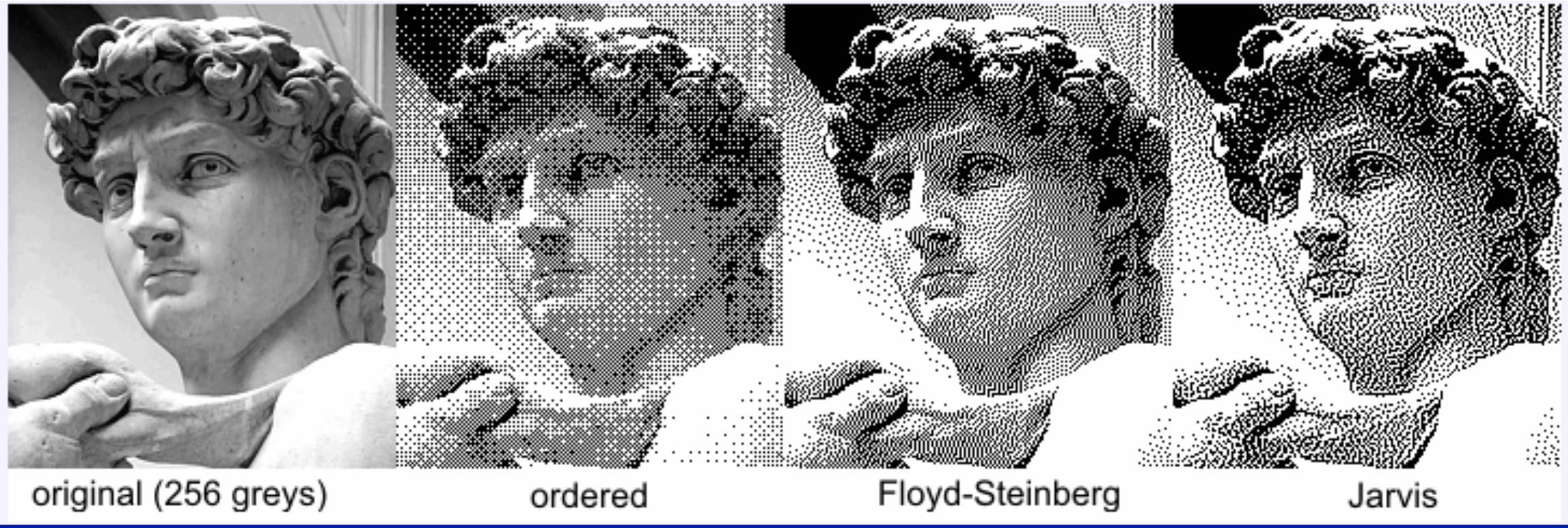
- Diffuse the quantization error of a pixel to its neighboring pixels
- Scan in raster order
- At each pixel, draw least error output value
- Add the error fractions into adjacent, unwritten pixels

- If a number of pixels have been rounded downwards, it becomes more likely that the next pixel is rounded upwards

```
for each y
    for each x
        oldpixel := pixel[x][y]
        newpixel := find_closest_palette_color(oldpixel)
        pixel[x][y] := newpixel
        quant_error := oldpixel - newpixel
        pixel[x+1][y] := pixel[x+1][y] + 7/16 * quant_error
        pixel[x-1][y+1] := pixel[x-1][y+1] + 3/16 * quant_error
        pixel[x][y+1] := pixel[x][y+1] + 5/16 * quant_error
        pixel[x+1][y+1] := pixel[x+1][y+1] + 1/16 * quant_error
```

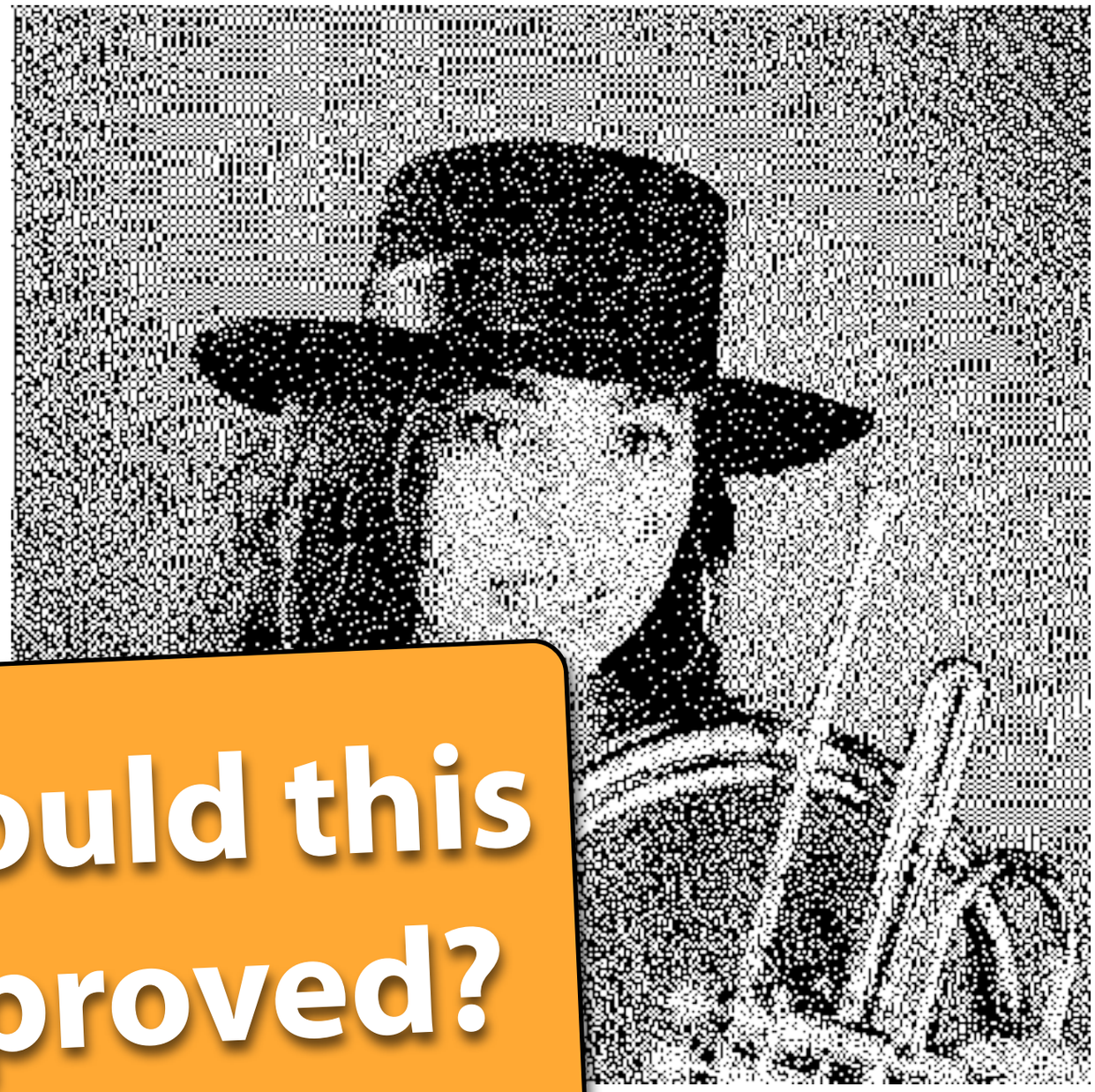


Floyd-Steinberg Error Diffusion



Floyd-Steinberg Error Diffusion

Enhances edges
Retains high frequency
Some checkerboarding



How could this be improved?

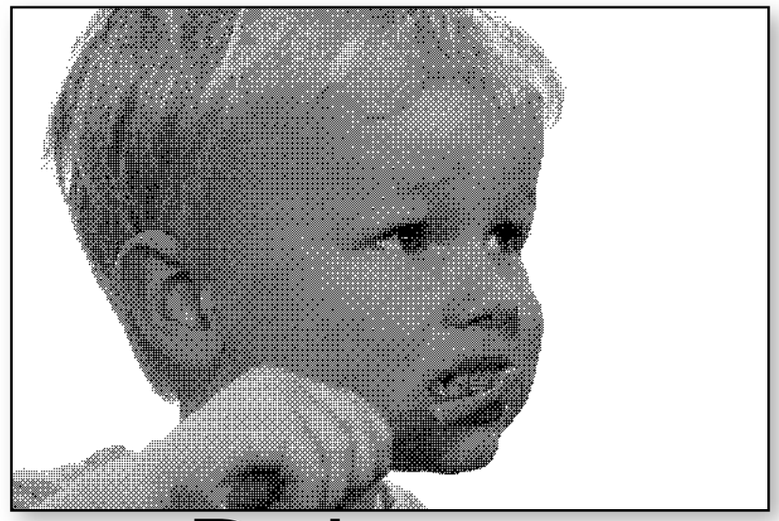
From http://www.cs

Color Dithering

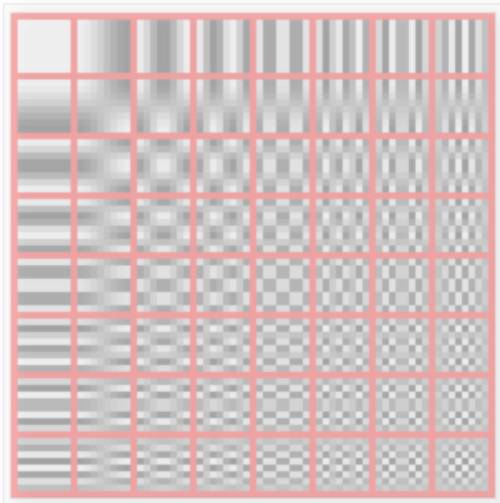
- Example: 8 bit framebuffer
  - –Set color map by dividing 8 bits into 3,3,2 for RGB
  - –Blue is deemphasized because we see it less well
- Dither RGB separately
  - –Works well with Floyd-Steinberg
- Generally looks good

Overview

Announcements



Dithering



Compression

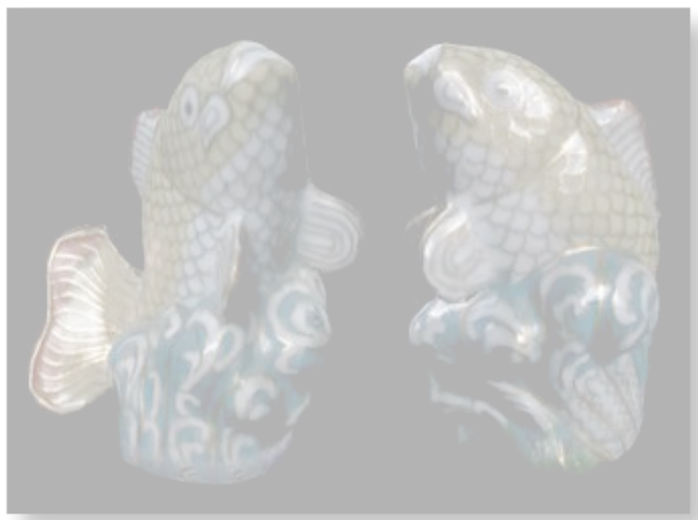
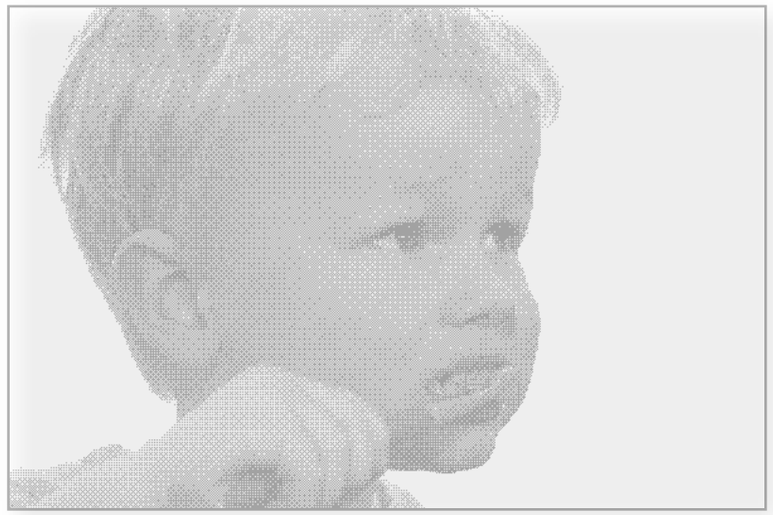


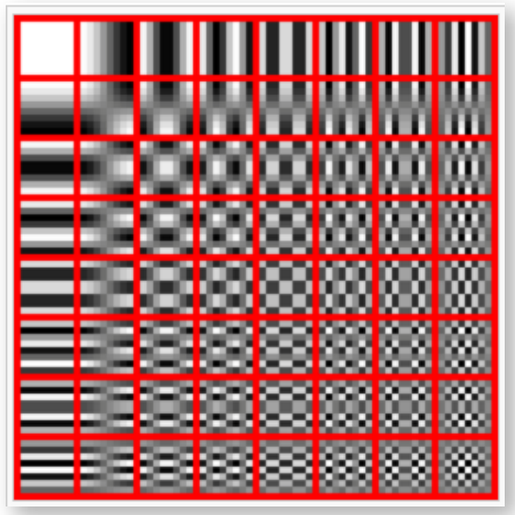
Image-based Rendering

Overview

Announcements



Dithering



Compression

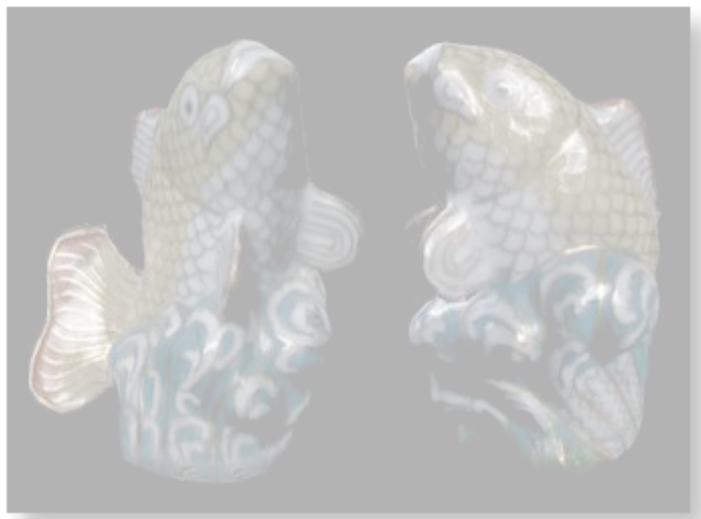


Image-based Rendering

Image Sizes

- 1024*1024 at 24 bits uses 3 MB
- Encyclopedia Britannica at 300 pixels/inch and 1 bit/pixes requires 25 gigabytes (25K pages)
- 90 minute movie at 640x480, 24 bits per pixels, 24 frames per second requires 120 gigabytes
- Applications: HDTV, DVD, satellite image transmission, medial image processing, fax, ...

Types of Compression

http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

Types of Compression

http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

Huffman Coding

Suppose we have the following 4 colors:

00 01 10 11

As used in this image:

00	00	10	10	10	10
10	10	10	10	01	10
10	10	11	10	11	00

Binary String *(36 bits)*:

000010101010101010101001101010
11101100

Switch to this encoding:

000 001 1 010

Which is equivalent to:

000	000	1	1	1	1
1	1	1	1	001	1
1	1	010	1	010	000

Binary String *(28 bits)*:

000111111111000
111101010100000

Huffman Coding

Suppose we have the following 4 colors:

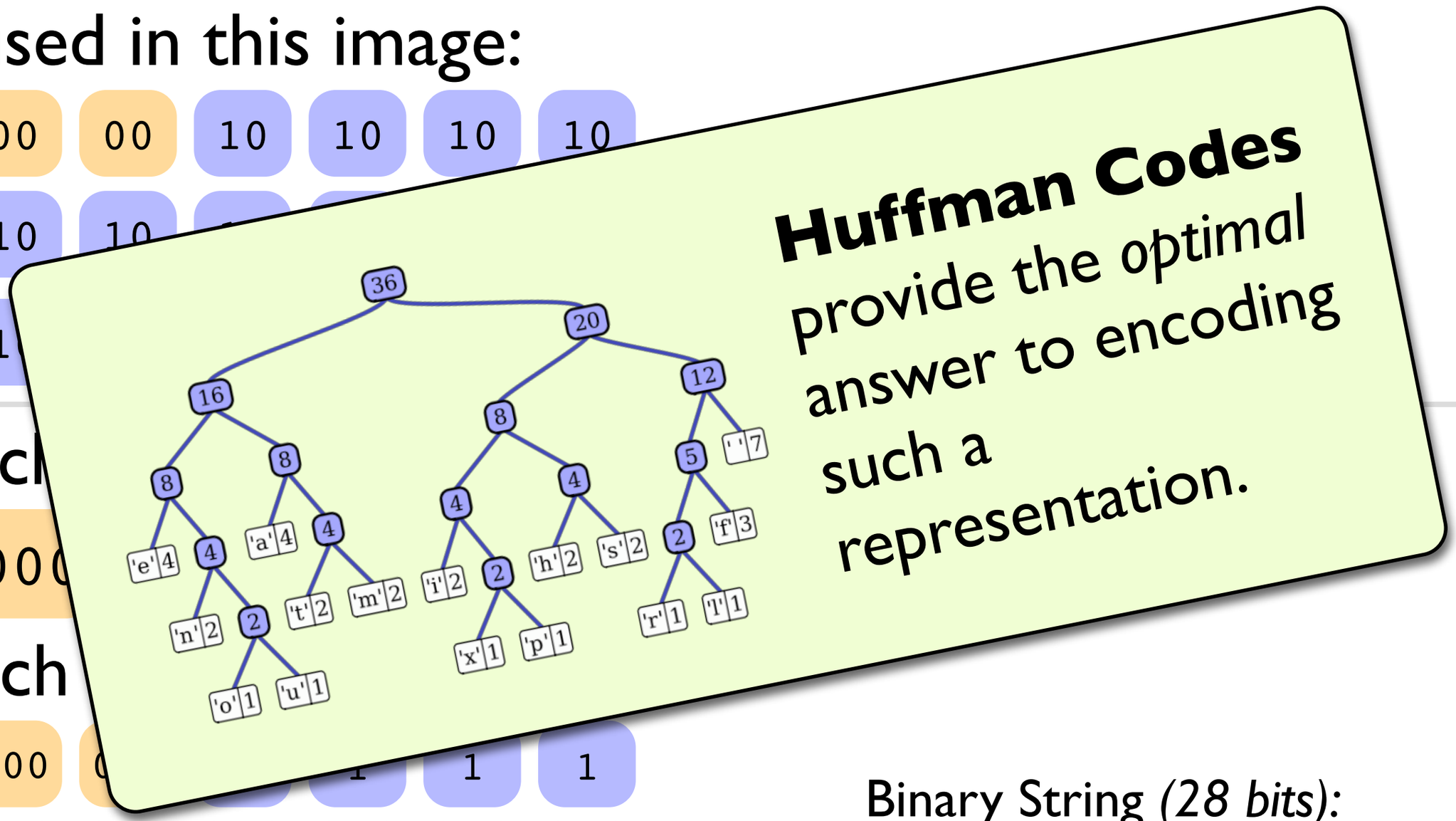
00 01 10 11

As used in this image:

00 00 10 10 10 10

10 10

1



Huffman Codes provide the optimal answer to encoding such a representation.

Switch

000

Which

000

1 1 1 1 001 1

1 1 010 1 010 000

Binary String *(28 bits)*:

0001111111111000

1111010101010000

Exploiting Coding Redundancy

- Not limited to images (text, other digital info)
- Exploit nonuniform probabilities of symbols
- Entropy as measure of information content

  – $H = -S_i \text{ Prob}(s_i) \log_2 (\text{Prob}(s_i))$

  – Low entropy → non uniform probability
  – High entropy → uniform probability

  – If source is independent random variable need H bits

Types of Compression

http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

Types of Compression

http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

Run Length Encoding

Same Image As Before:

00	00	10	10	10	10
10	10	10	10	01	10
10	10	11	10	11	00

Scan Convert:

00 00 10 10 10 10 10 10 10 10 01 10 10 10 11 10 11 00

Encode:

2× 00 8× 10 1× 01 3× 10 1× 11 1× 10 1× 11 1× 00

Run Length Encoding

Same Image As Before:

00 00 10 10 10 10

10

10

Scan

00 00

Encoc

2× 00 8× 10 1× 01 3× 10 1× 11 1× 10 1× 11 1× 00

Related Ideas

- **Quadtrees:** Recursively subdivide until cells are constant color.
- **Region Encoding:** Represent boundary curves of constant-color regions.

Types of Compression

http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

Types of Compression

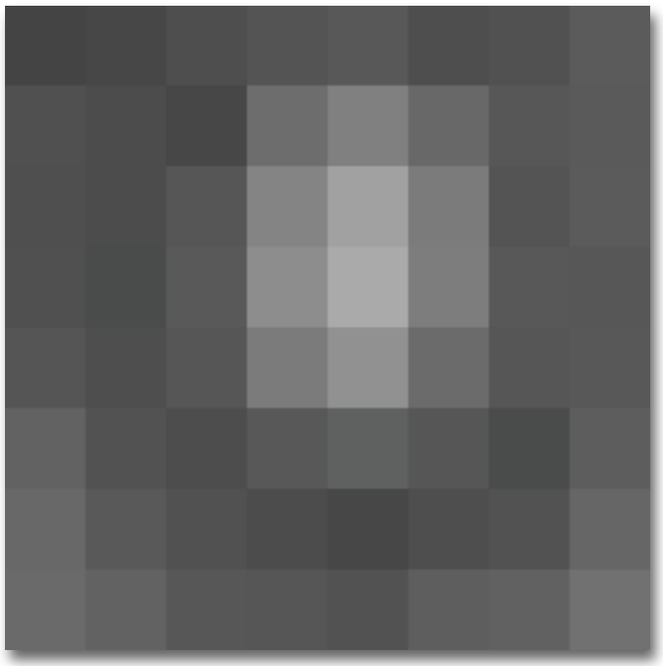
http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

JPEG Compression

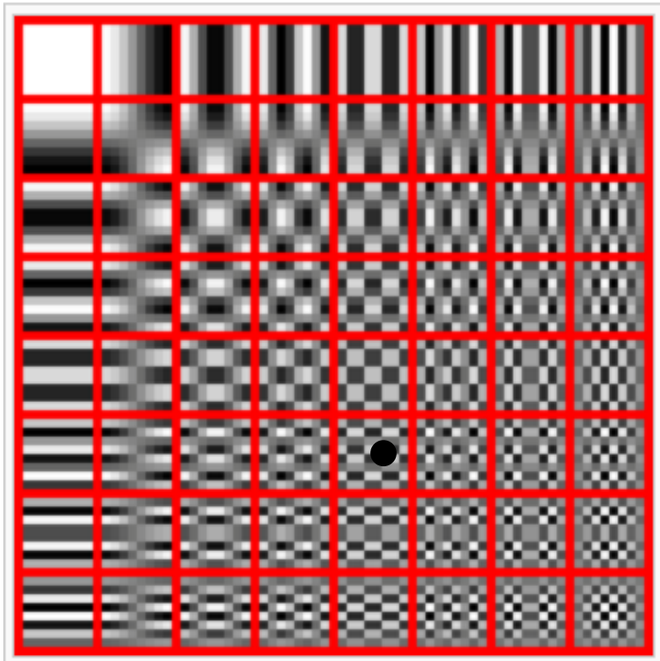
Divide image into 8x8 blocks.

JPEG Compression



(8x8 image block)

=



(discrete cosine bases)

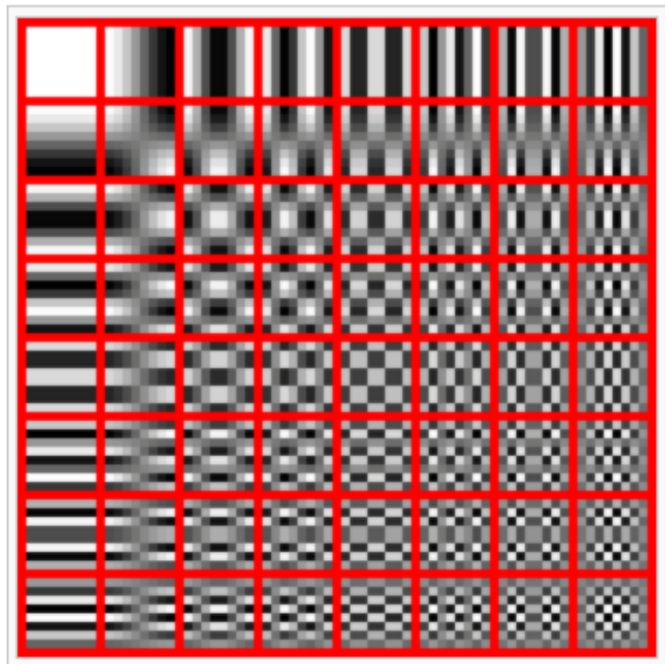
x

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

(coefficients)

- Express each block as a linear combination of 8x8 basis blocks made of cosines.
- This is called the *discrete cosine transform*.

Key Insight!



(discrete cosine bases)

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix}$$

(coefficients)

- Upper left blocks have higher values than lower right? (*They are more important.)*
- Why?

How can we exploit this insight?

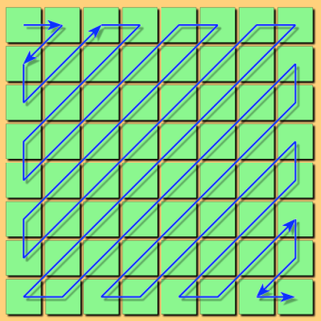
Scaled Coefficients

$$\begin{bmatrix} -415 & -30 & -61 & 27 & 56 & -20 & -2 & 0 \\ 4 & -22 & -61 & 10 & 13 & -7 & -9 & 5 \\ -47 & 7 & 77 & -25 & -29 & 10 & 5 & -6 \\ -49 & 12 & 34 & -15 & -10 & 6 & 2 & 2 \\ 12 & -7 & -13 & -4 & -2 & 2 & -3 & 3 \\ -8 & 3 & 2 & -6 & -2 & 1 & 4 & 2 \\ -1 & 0 & 0 & -2 & -1 & -3 & 4 & -1 \\ 0 & 0 & -1 & -4 & -1 & 0 & 1 & 2 \end{bmatrix} \div \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix} = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -4 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

(coefficients) (quantization matrix) (scaled coefficients)

$$\text{round}\left(\frac{-415}{16}\right) = \text{round}\left(-25.9375\right) = -26$$

- What can we see about the quantization matrix?
- How can we compress the scaled coefficients?



Answer:

Run Length + Huffman Coding

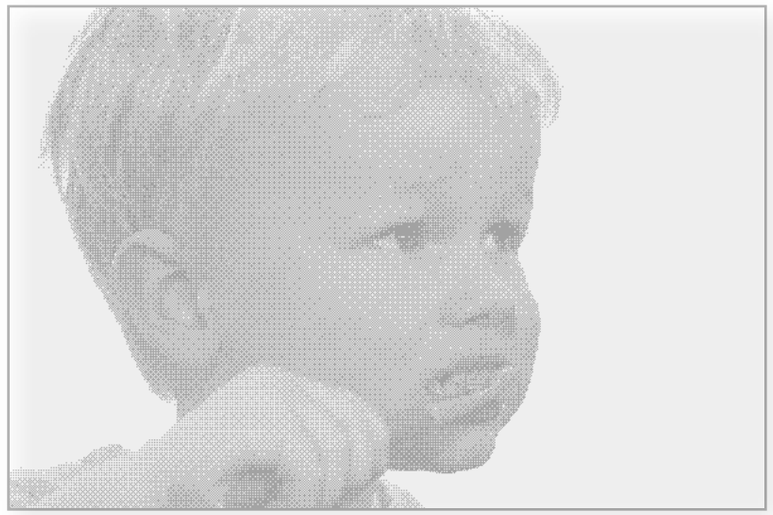
Types of Compression

http://en.wikipedia.org/wiki/File:Phalaenopsis_JPEG.png

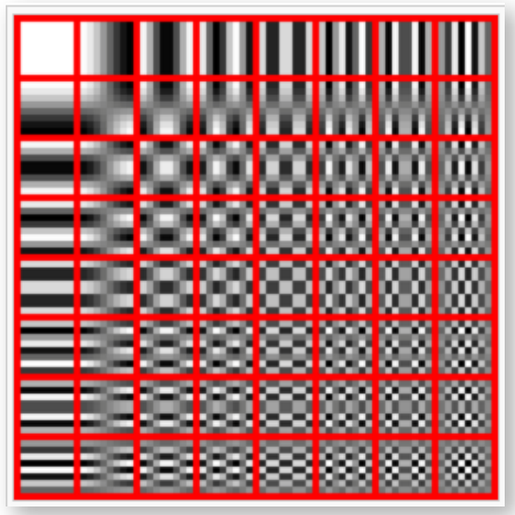
- Coding Redundancy
  - Huffman Coding (lossless)
- Spatial Coherence
  - Run Length Encoding (lossless)
- Psycho visual
  - JPEG Encoding (lossy)

Overview

Announcements



Dithering



Compression

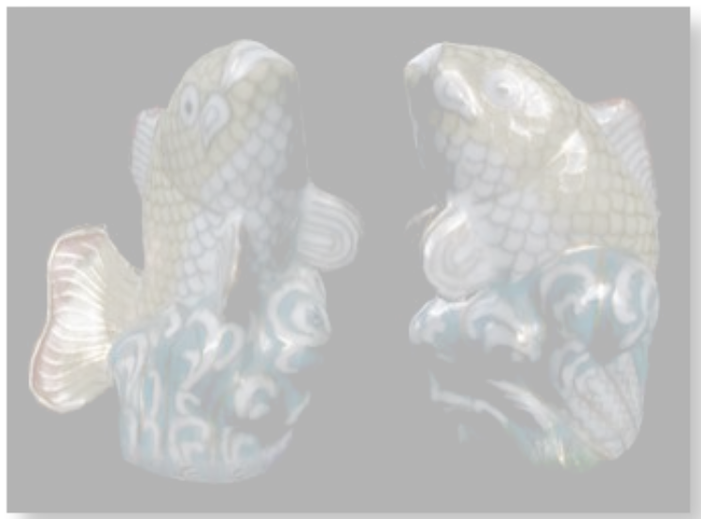
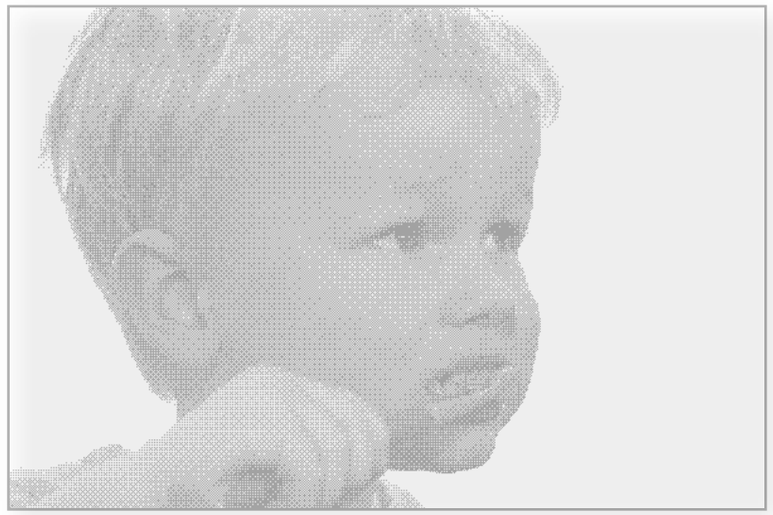


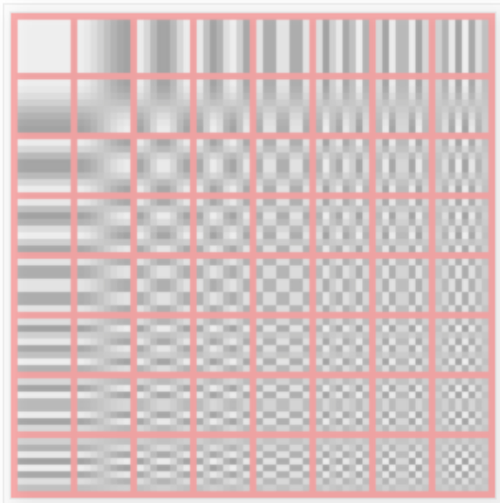
Image-based Rendering

Overview

Announcements



Dithering



Compression

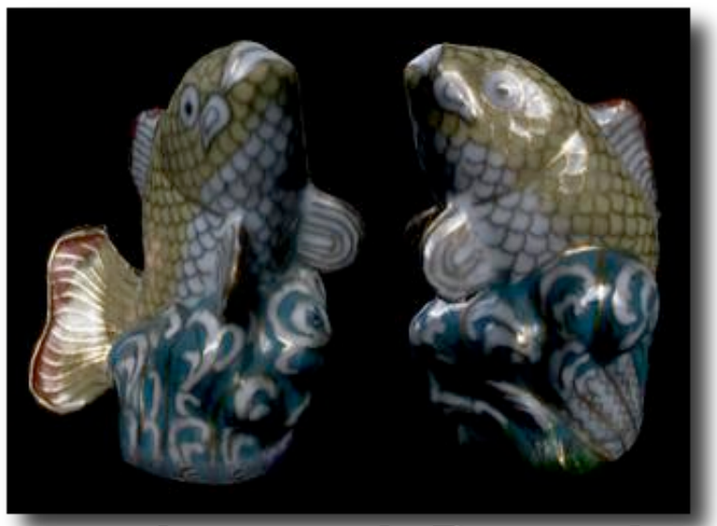
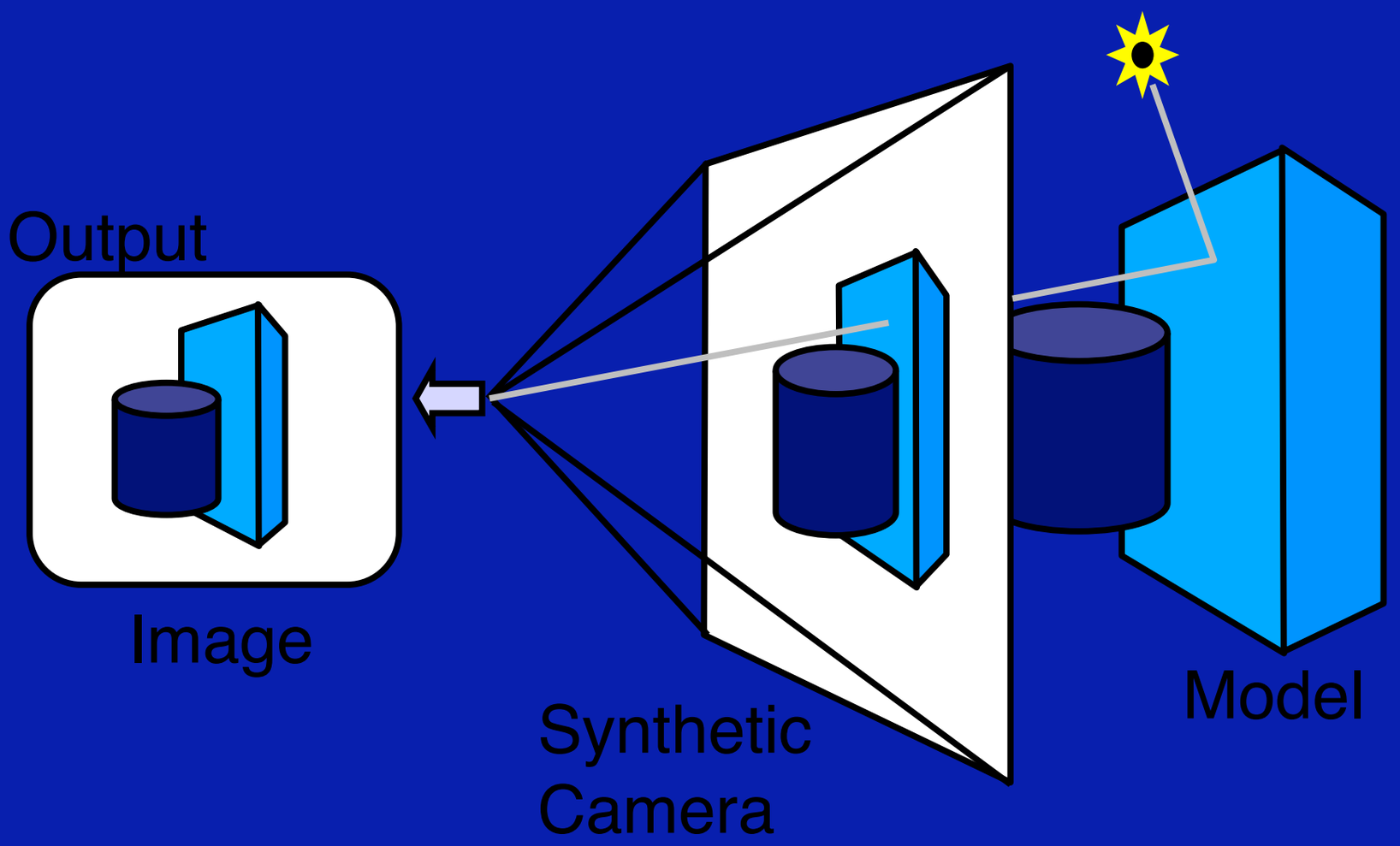


Image-based Rendering

Image-Based Rendering

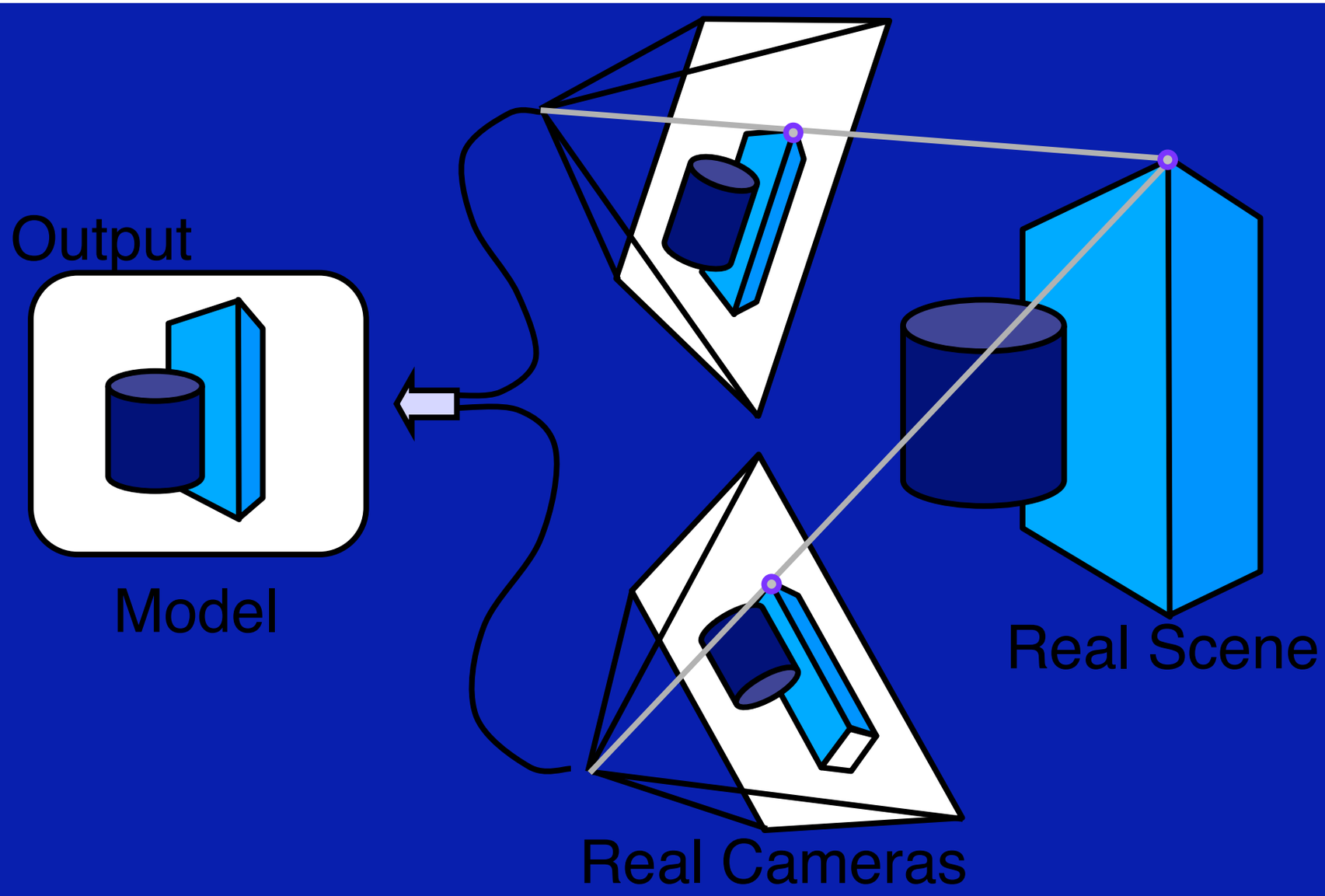
(with most of slides from Richard Szeliski and Michael Cohen)

Computer Graphics

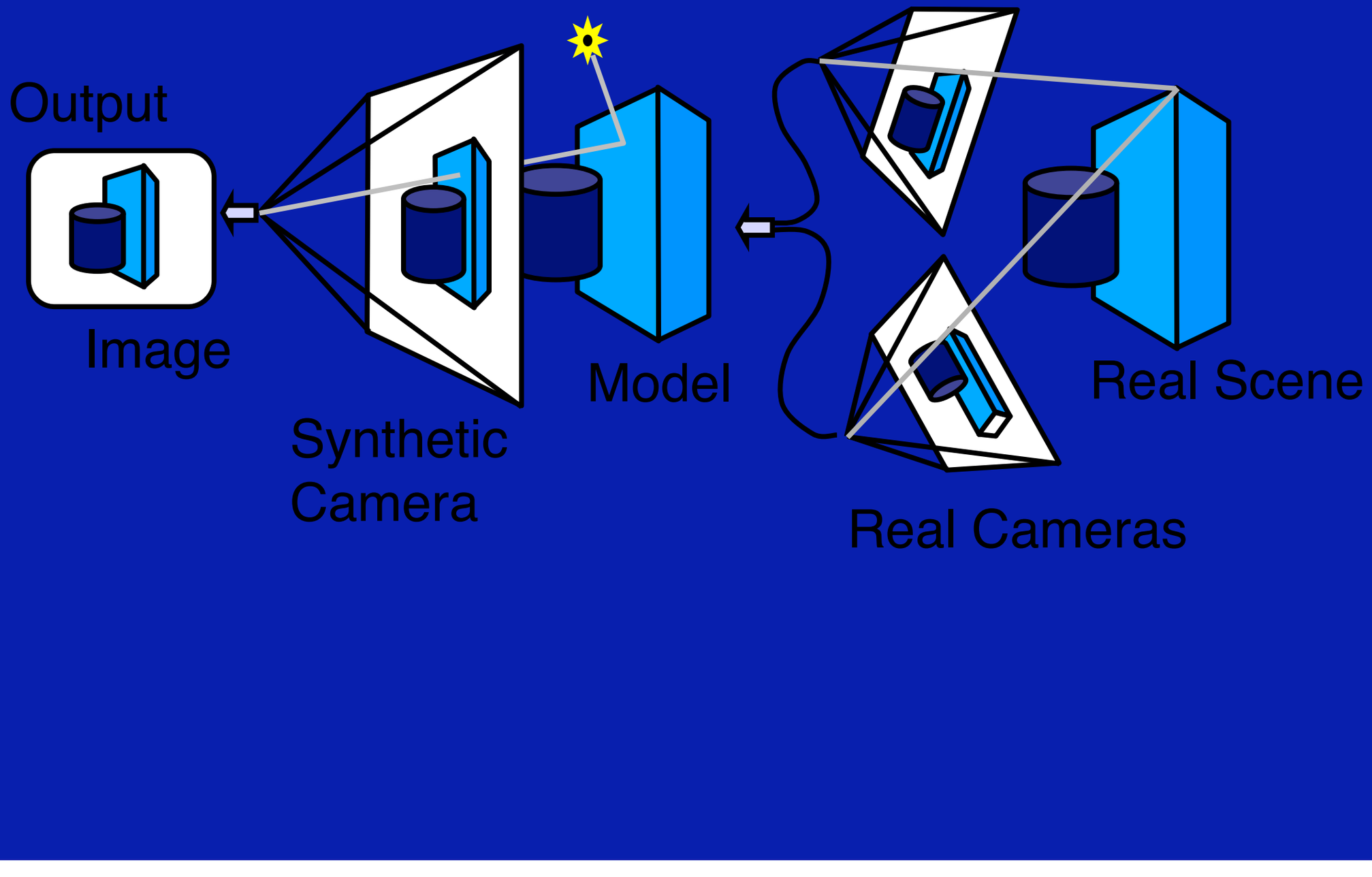


Geometry + Material attributes

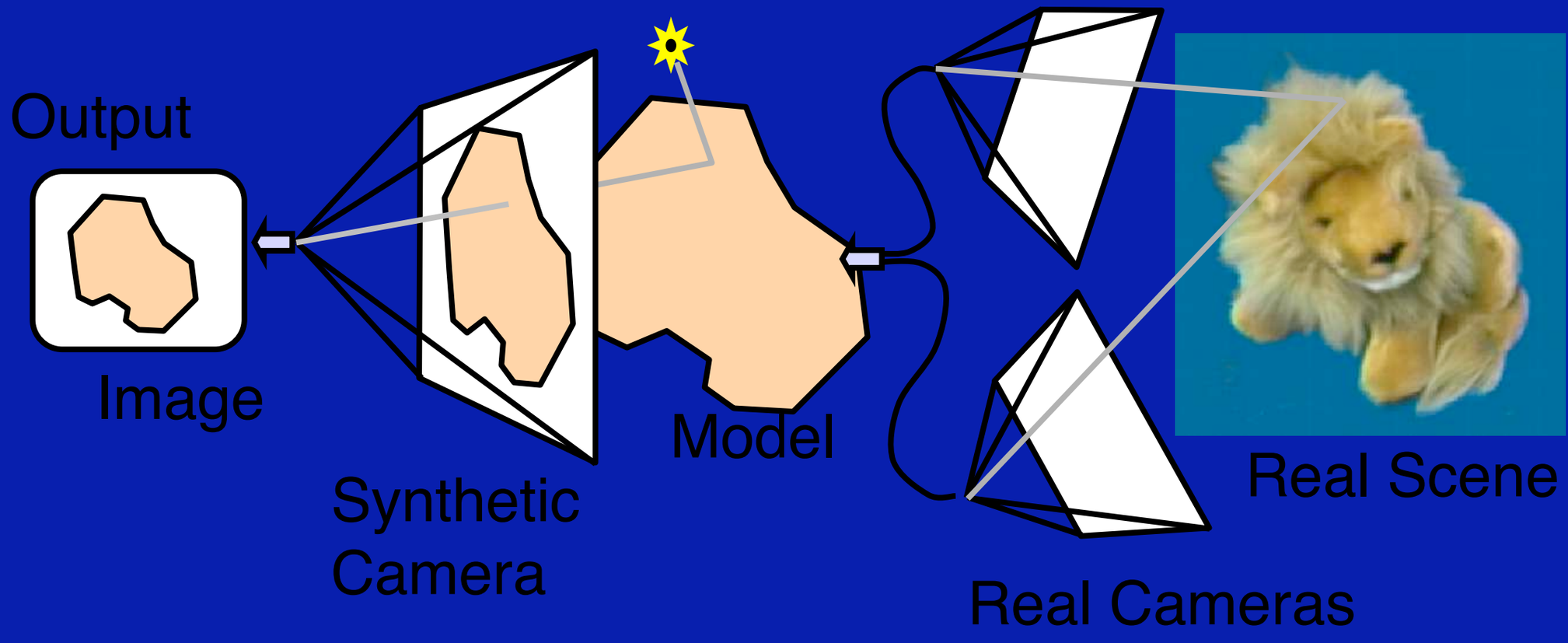
Computer Vision



Combined

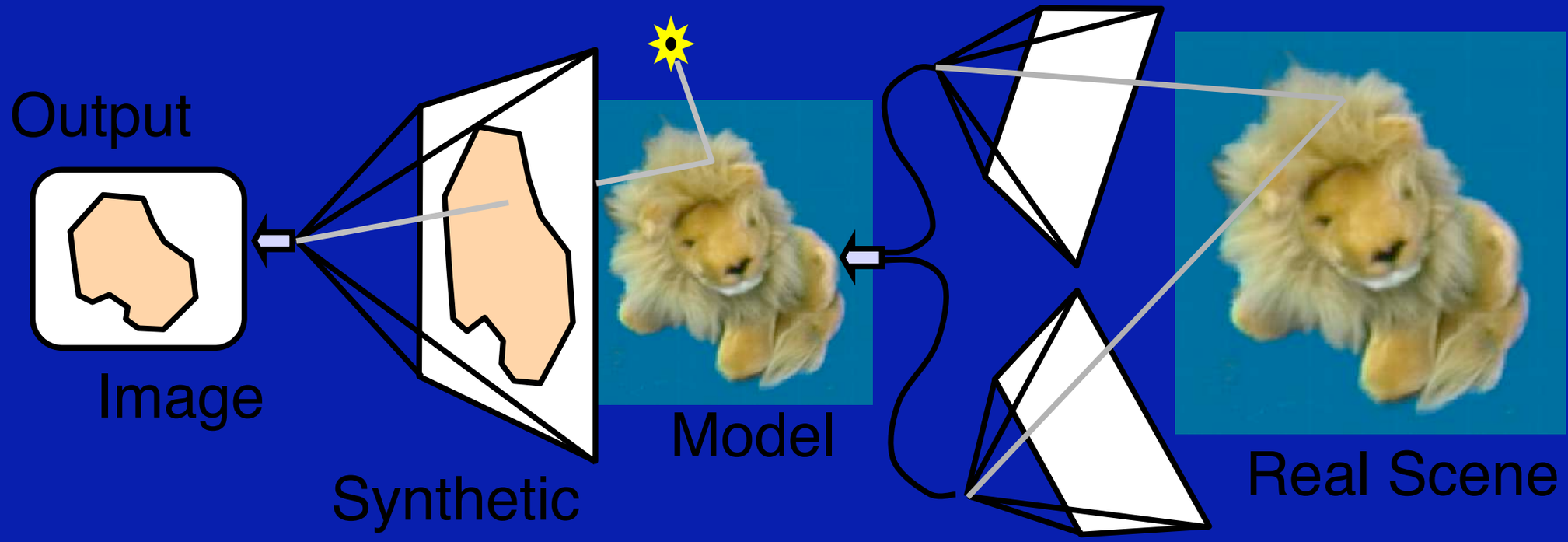

Output
Image
Synthetic
Camera
Model
Real Cameras
Real Scene

But, vision technology falls short



Hard to re-create much of the complex geometry and lighting effects found in real world

… and so does graphics.



Hard to render world illumination

What is Image-Based Rendering?

All we usually care about in rendering is generating images from new viewpoints.

Image-Based Rendering

Geometry based

- Geometry + Material attributes

Skip traditional modeling/rendering process

Image based rendering seeks to replace geometry and surface properties with images

Quicktime VR

Skip traditional modeling/rendering process

Capture environment maps from given locations

Look around from a fixed point

Show Demo

Lightfields and Lumigraphs

Modeling light

Capture flow of light in region of environment

Described by plenoptic function

Plenoptic Function

Describes the intensity of light:

- passing through a given point, ***x***
- in a given direction, (θ,ϕ)

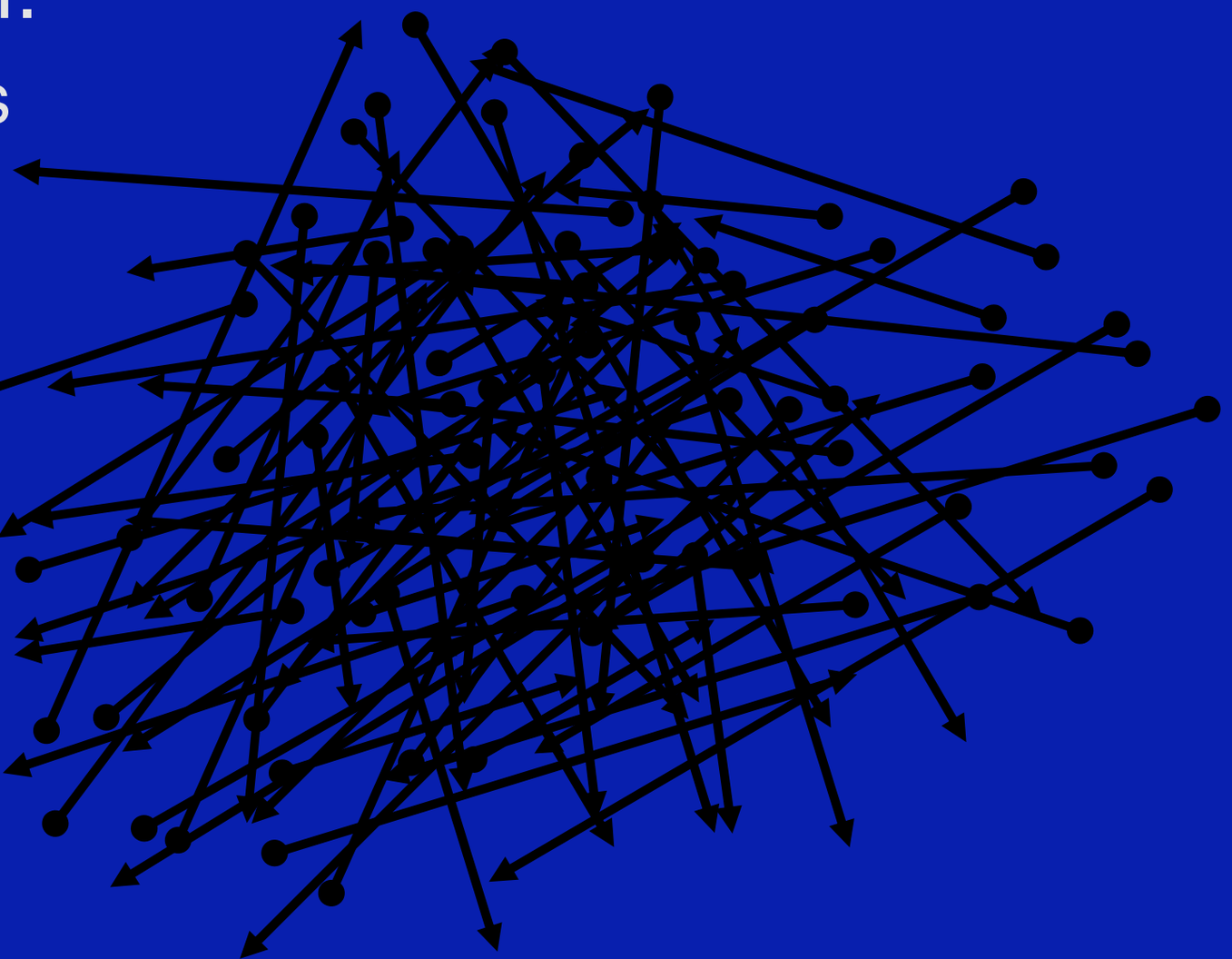
5D

- 3D position
- 2D direction

All Rays

Plenoptic Function:

- all possible rays



Plenoptic Function

Many image-based rendering approaches can be cast as sampling from and reconstructing the plenoptic function

Note, function is generally constant along segments of a line (assuming vacuum)

Line

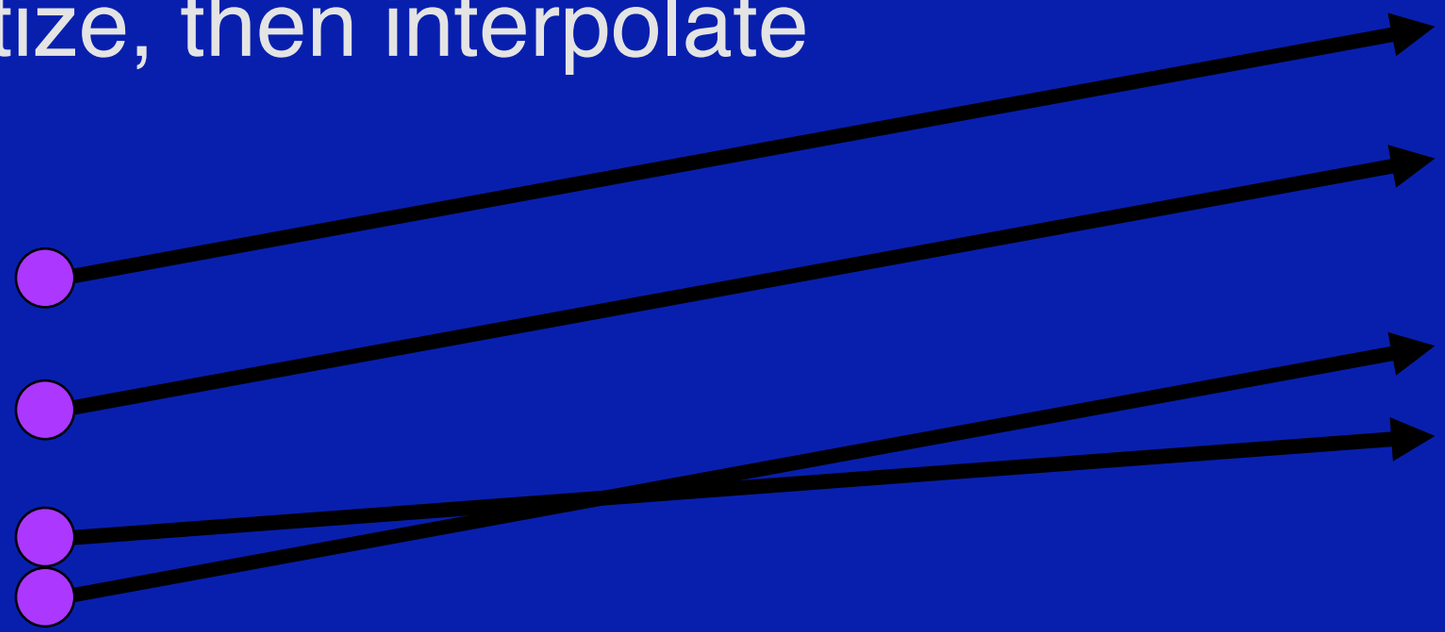
Infinite line

4D

- 2D direction
- 2D position
- Intensity does not change along the line

Ray

Discretize, then interpolate

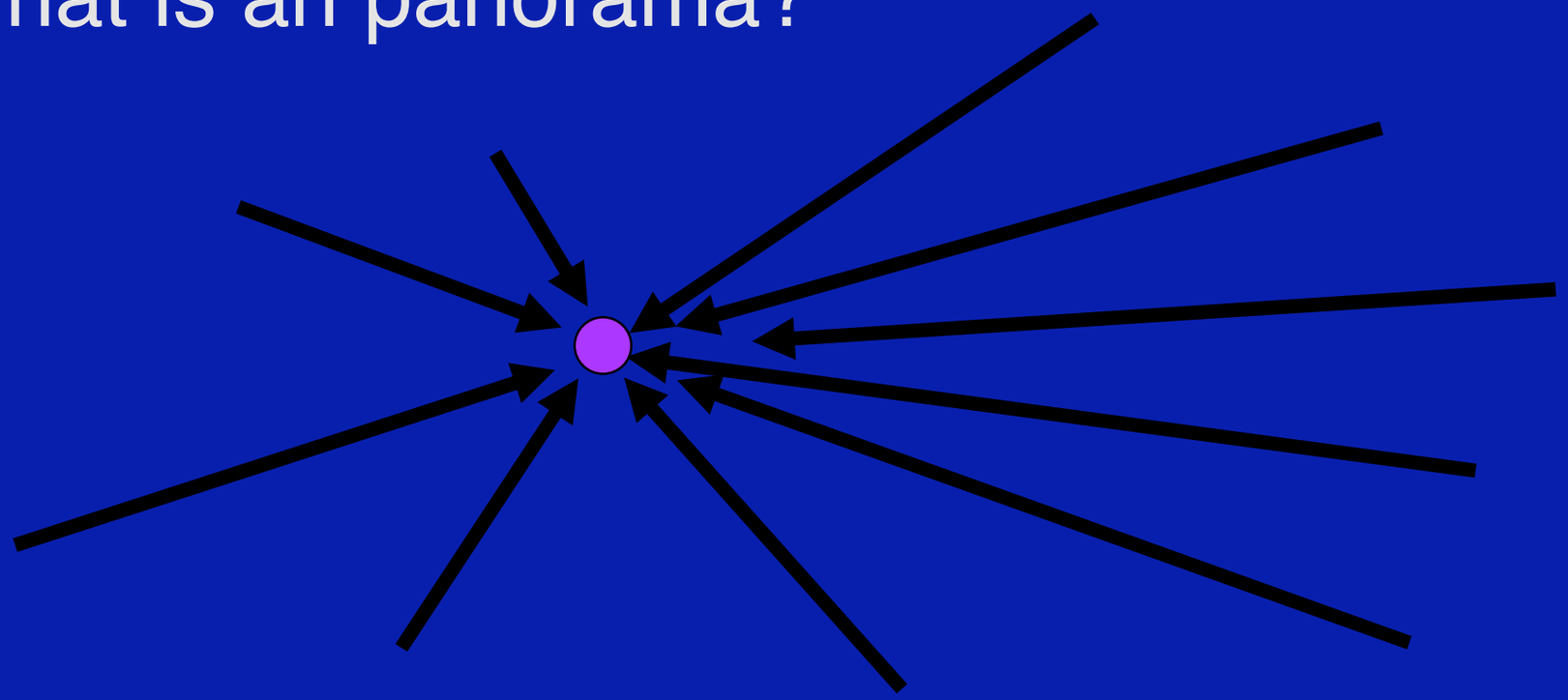


Distance between 2 rays

- Which is closer together?

Panorama

What is an panorama?



All rays through a point

Panoramic Mosaics

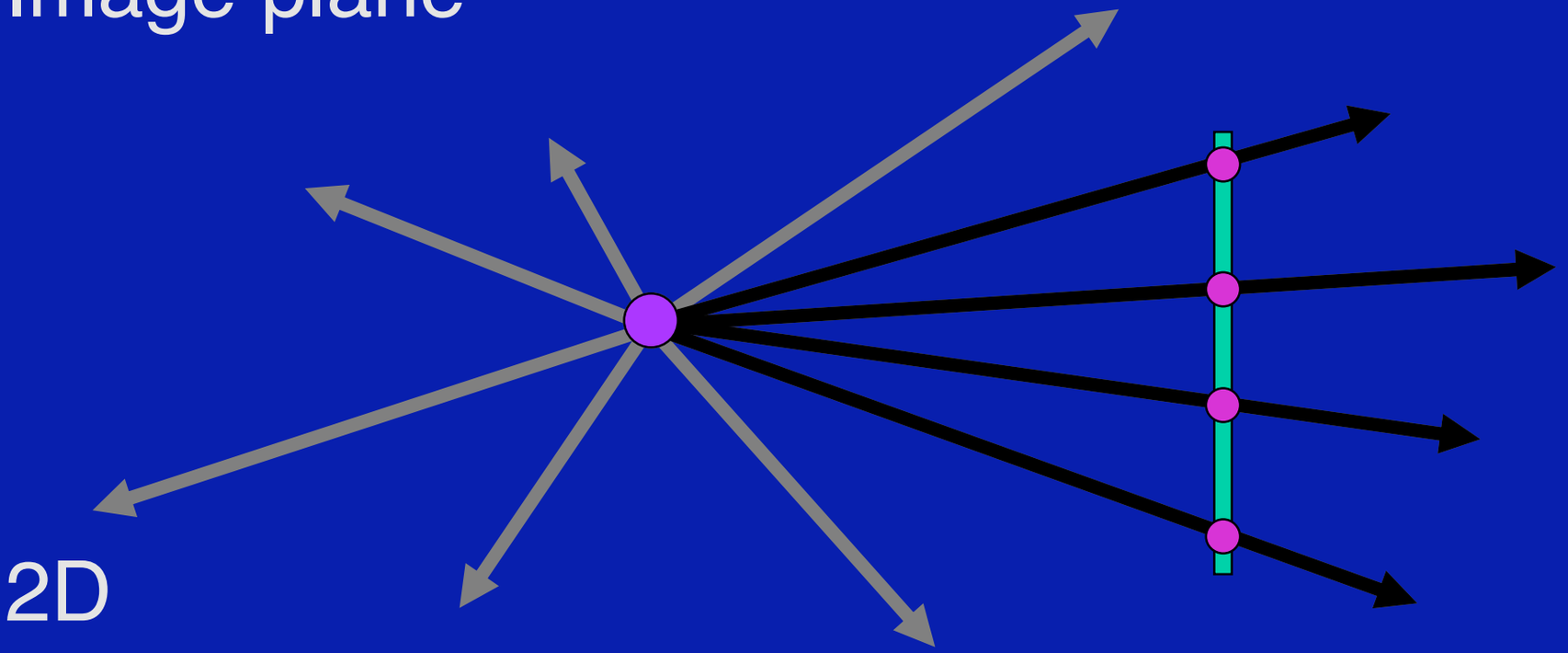
Convert panoramic image sequence into a cylindrical image



+ + + ... + =

Image

Image plane

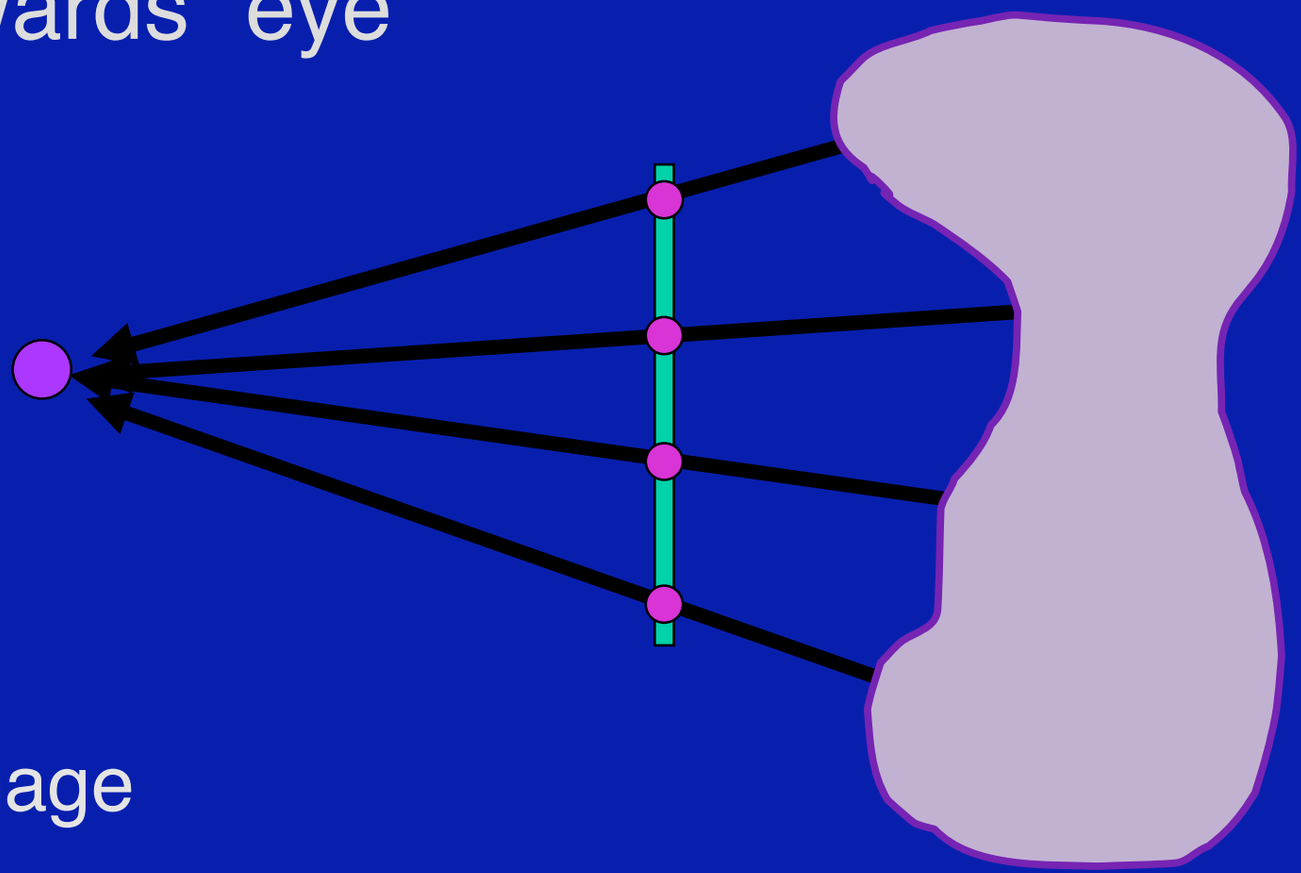


2D

- position in plane

Object

Light leaving towards “eye”

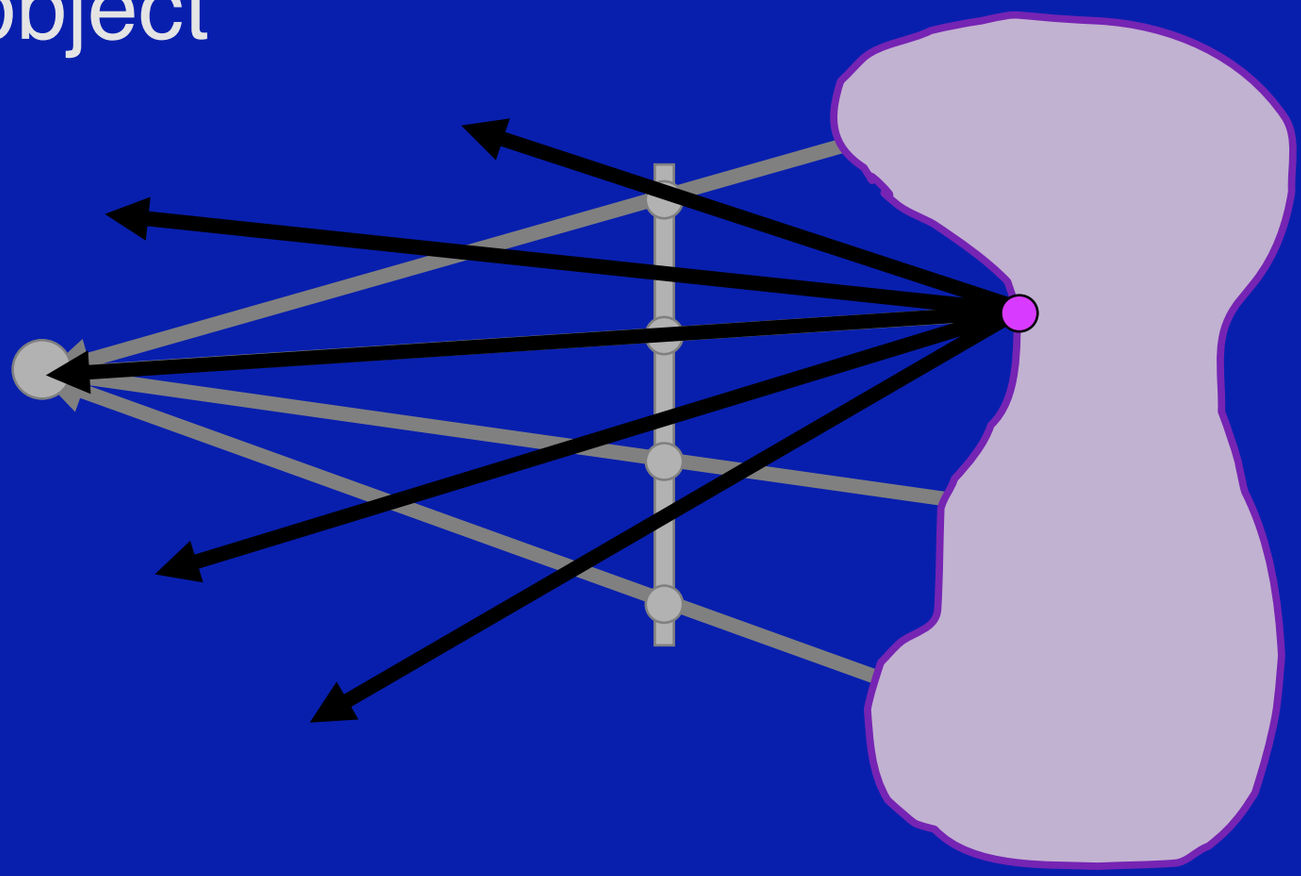


2D

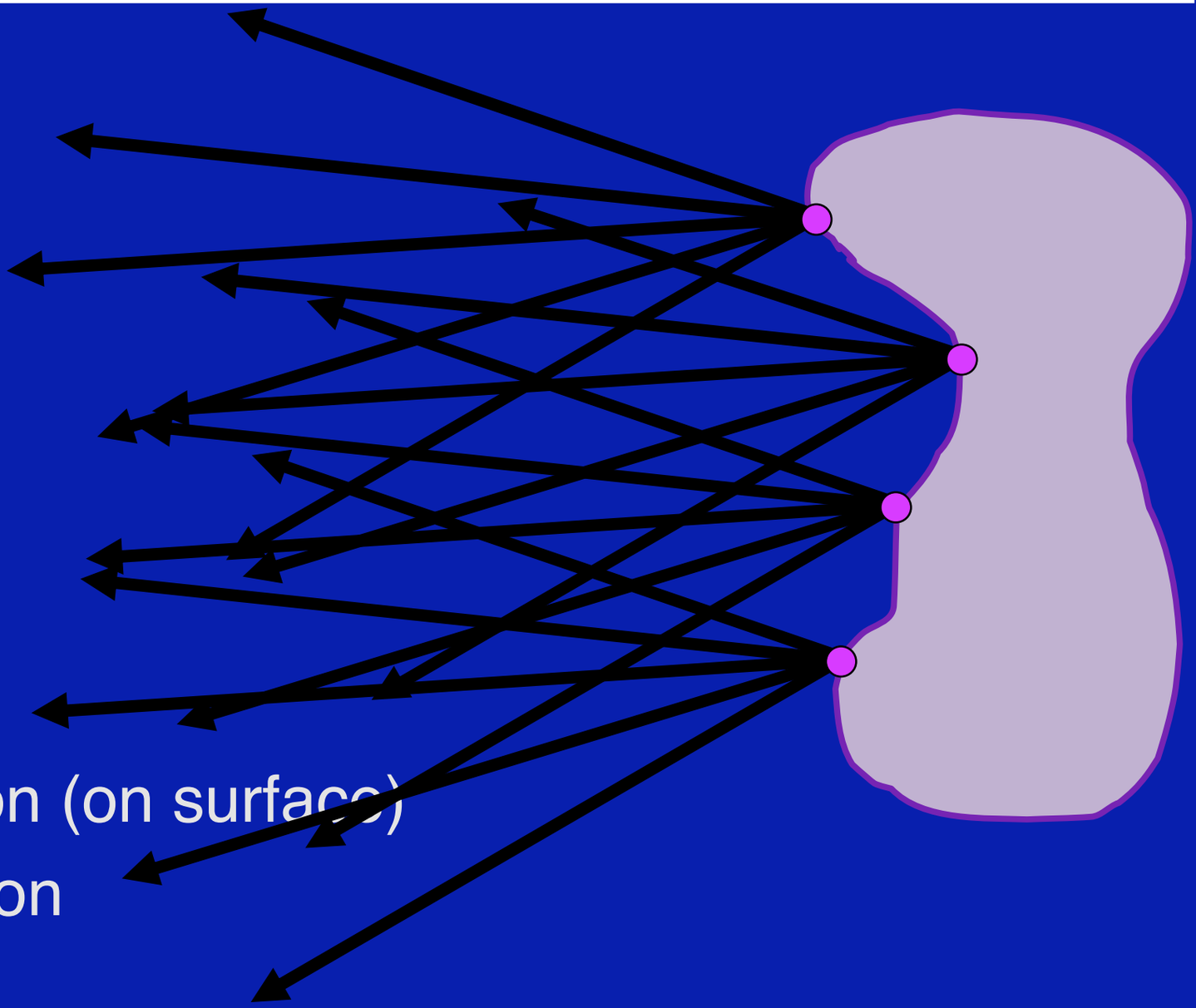
- just dual of image

Object

All light leaving object



Object

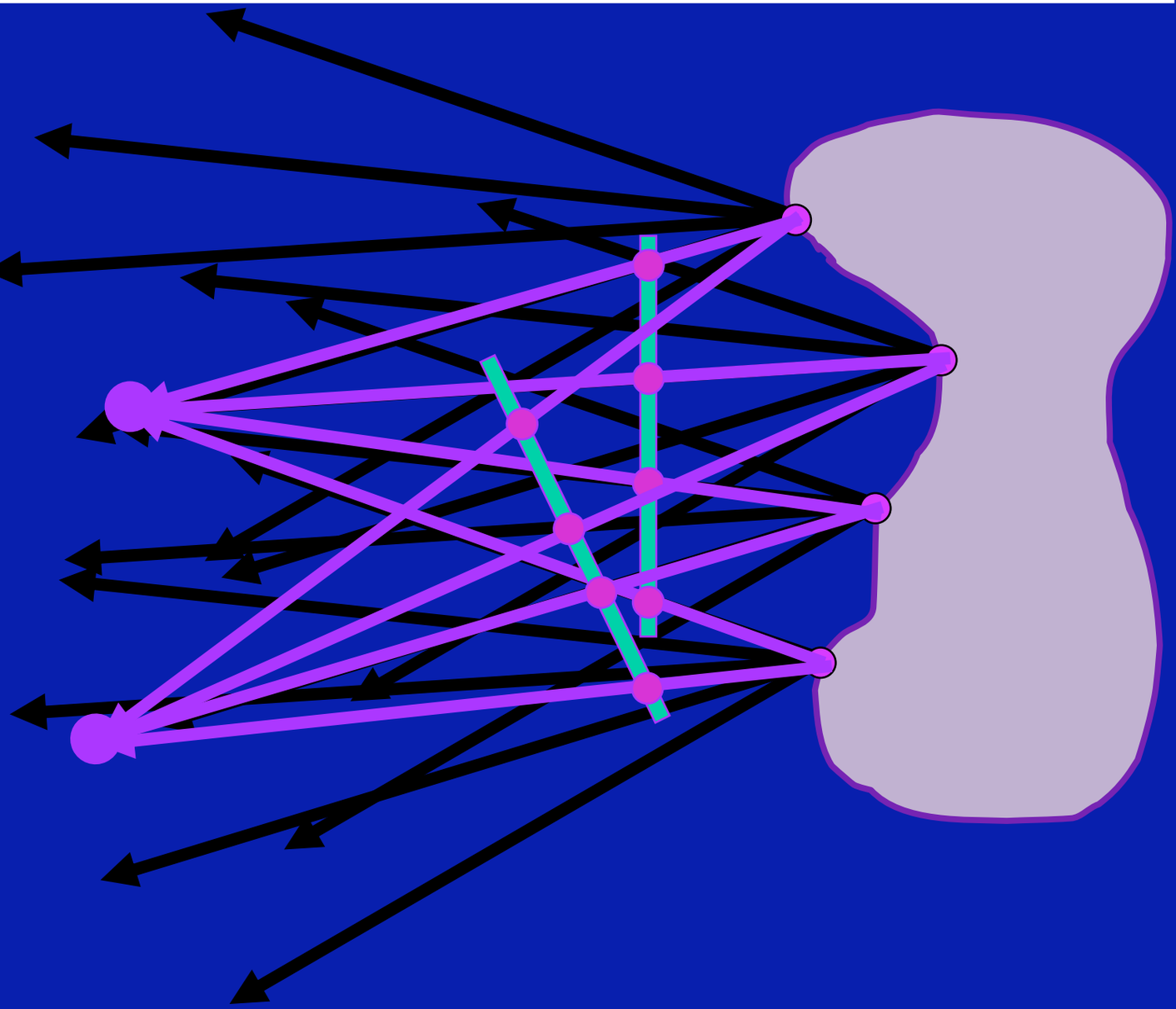


4D

- 2D position (on surface)
- 2D direction

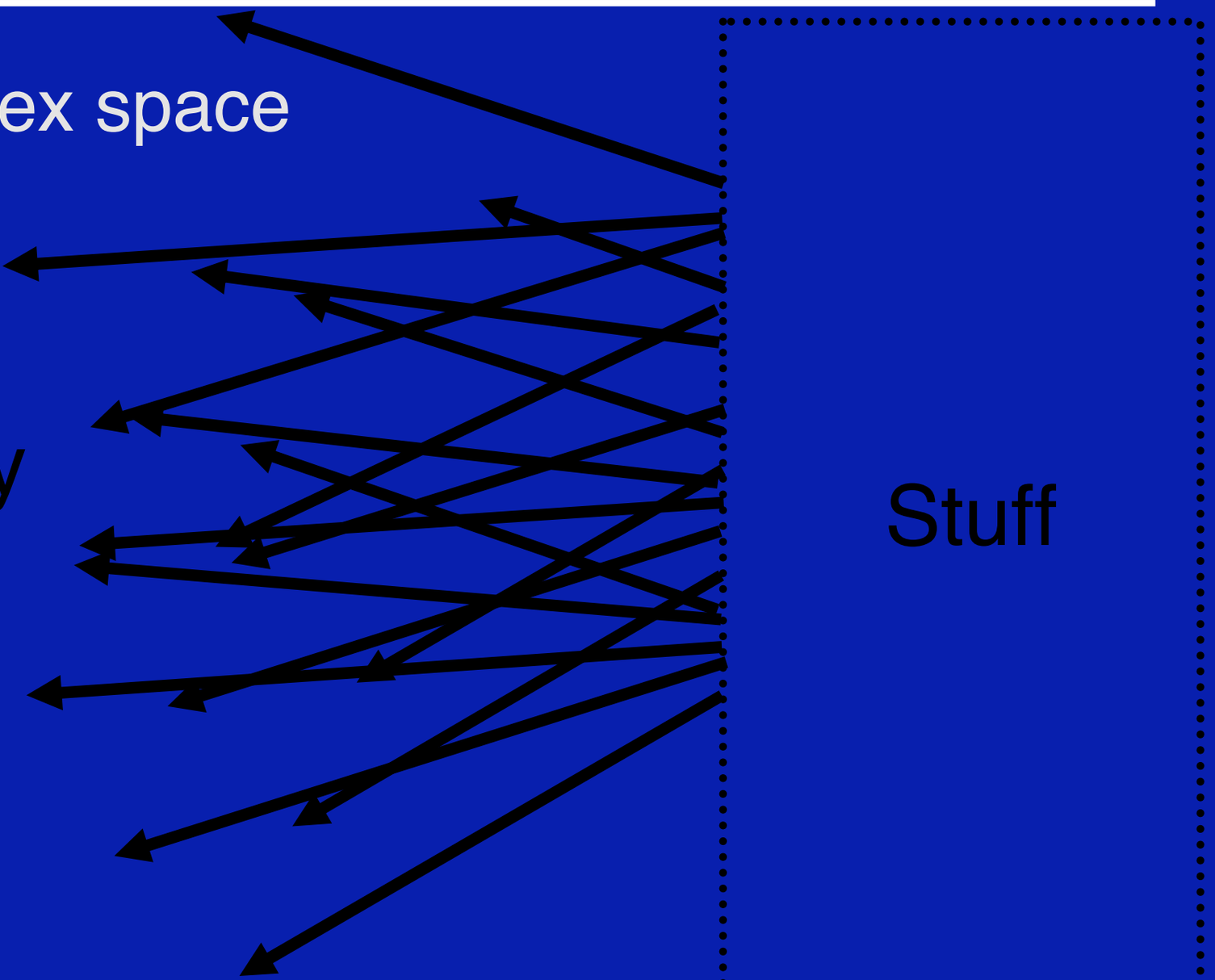
Object

All images



Lumigraph / Lightfield

Outside convex space



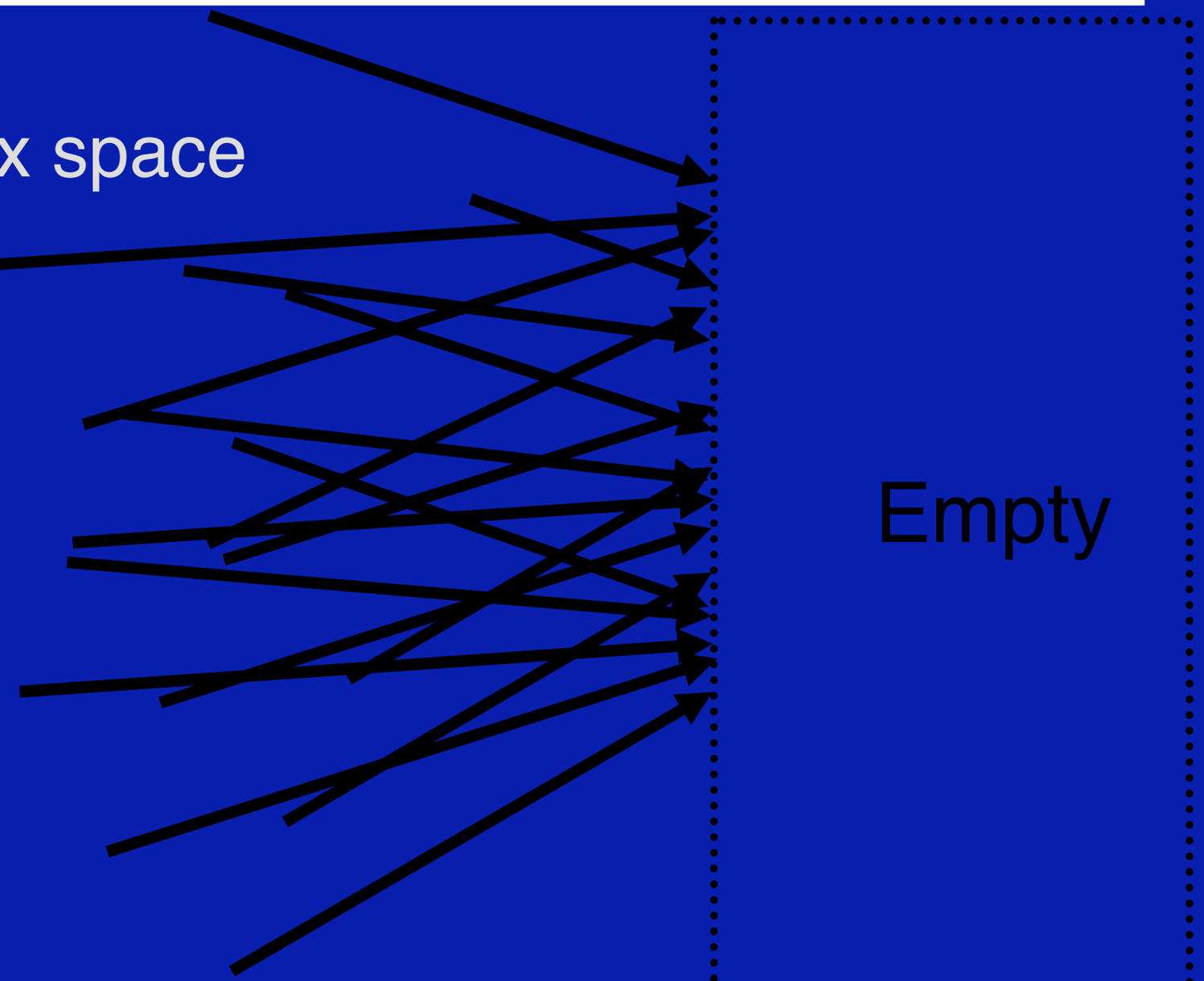
Empty

Stuff

4D

Lumigraph / Lightfield

Inside convex space



Stuff

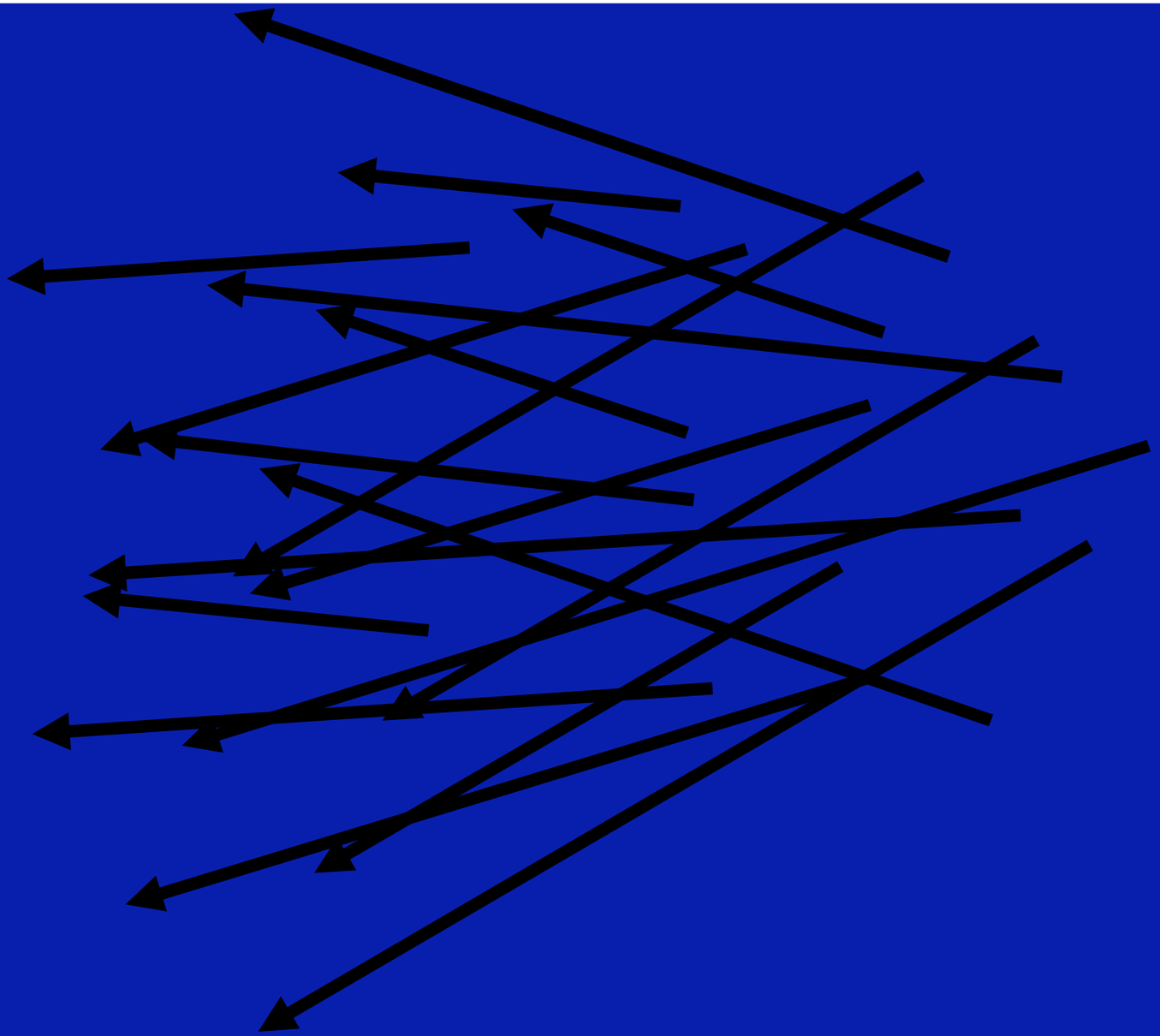
Empty

4D

Lumigraph / Lightfield

How to ?

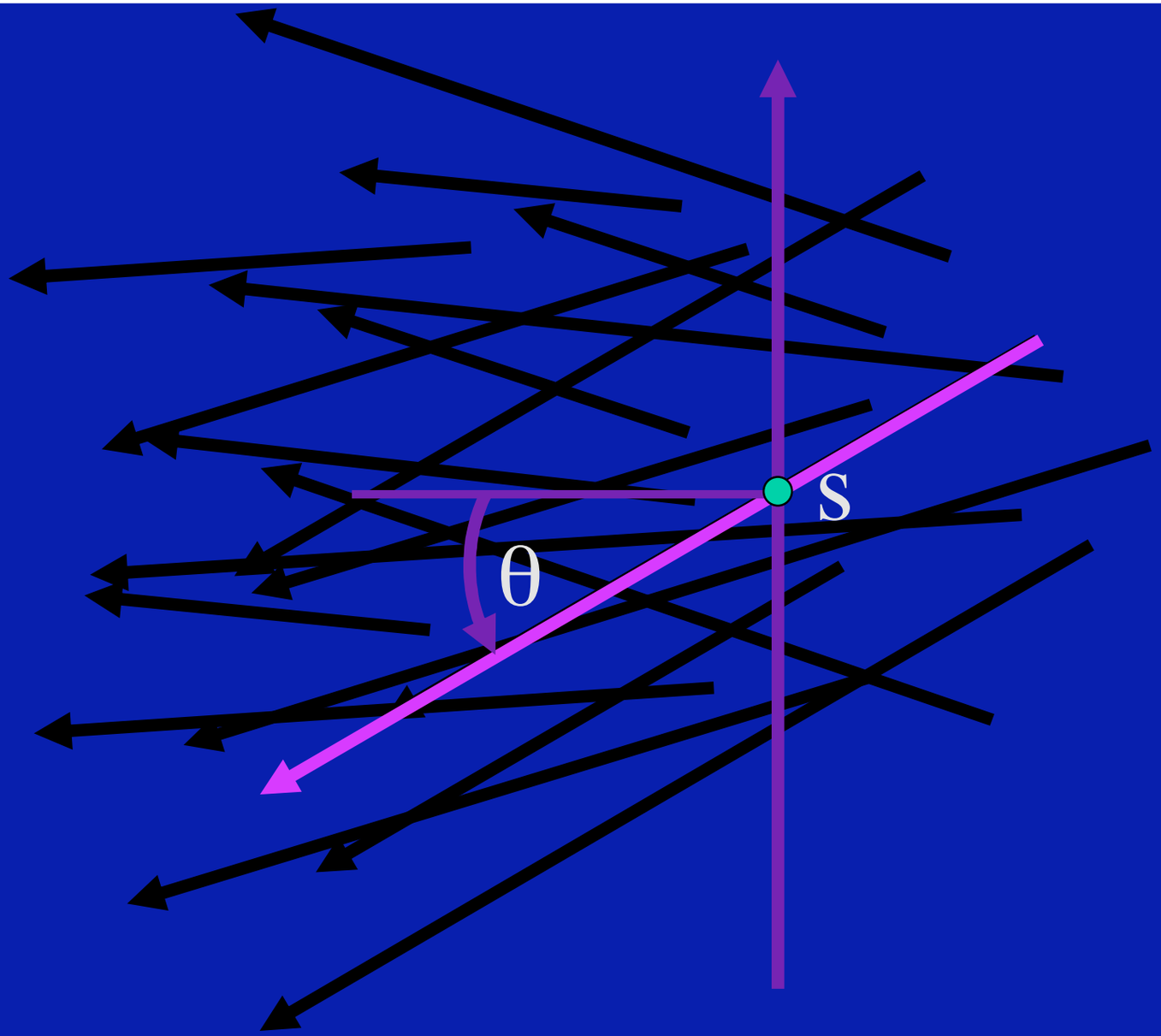
- organize
- capture
- render



Lumigraph - Organization

2D position

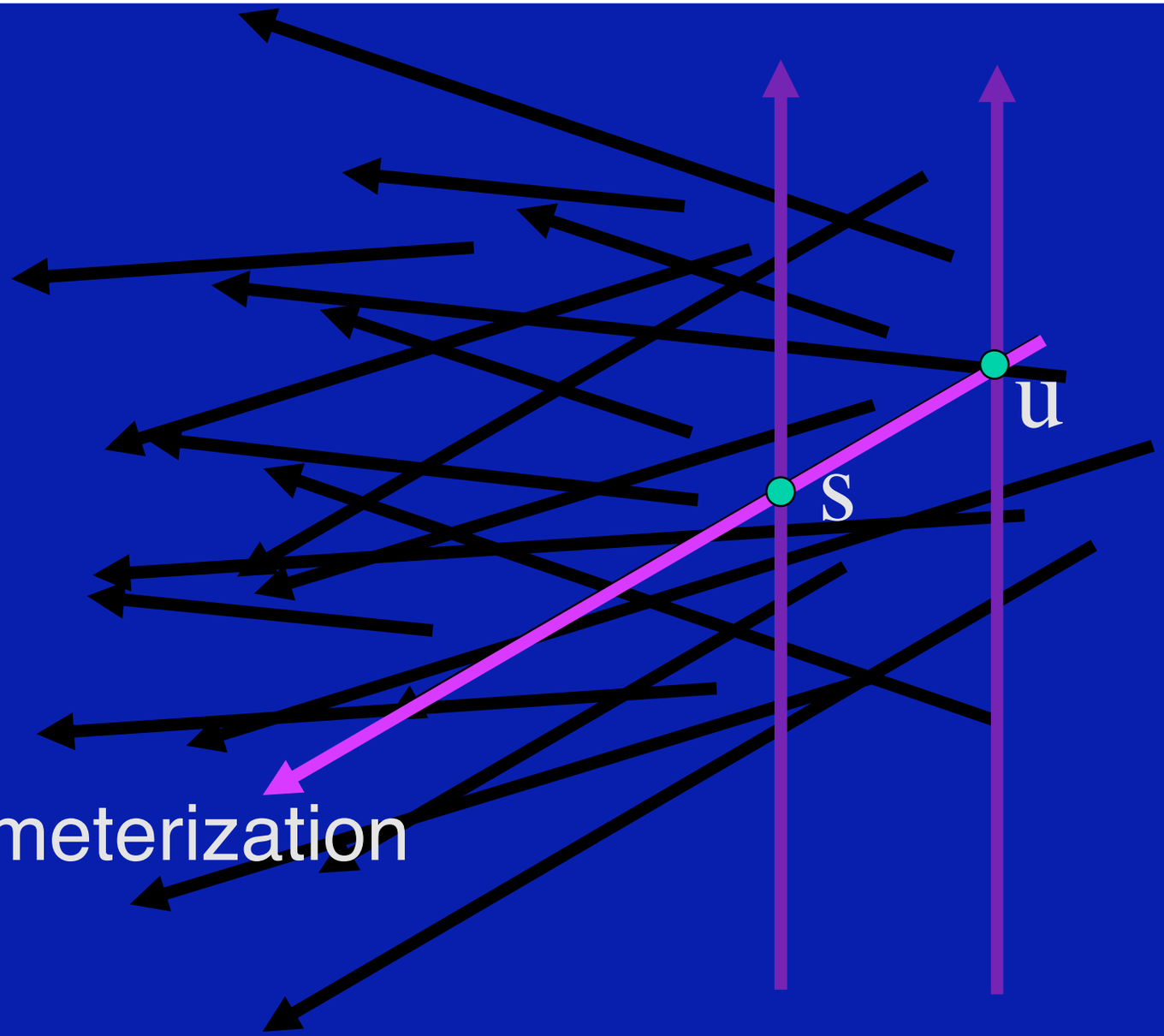
2D direction



Lumigraph - Organization

2D position

2D position

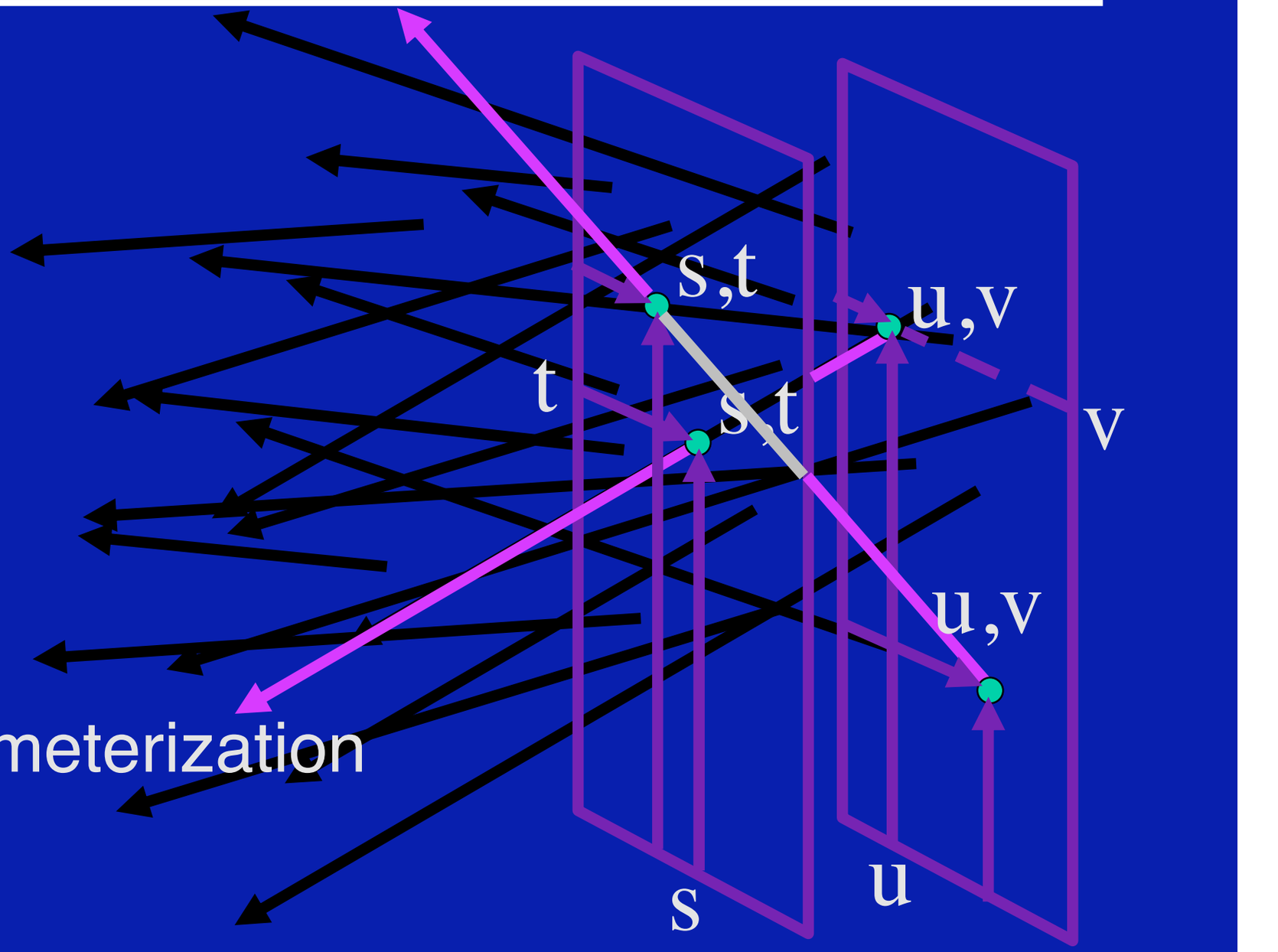


2 plane parameterization

Lumigraph - Organization

2D position

2D position



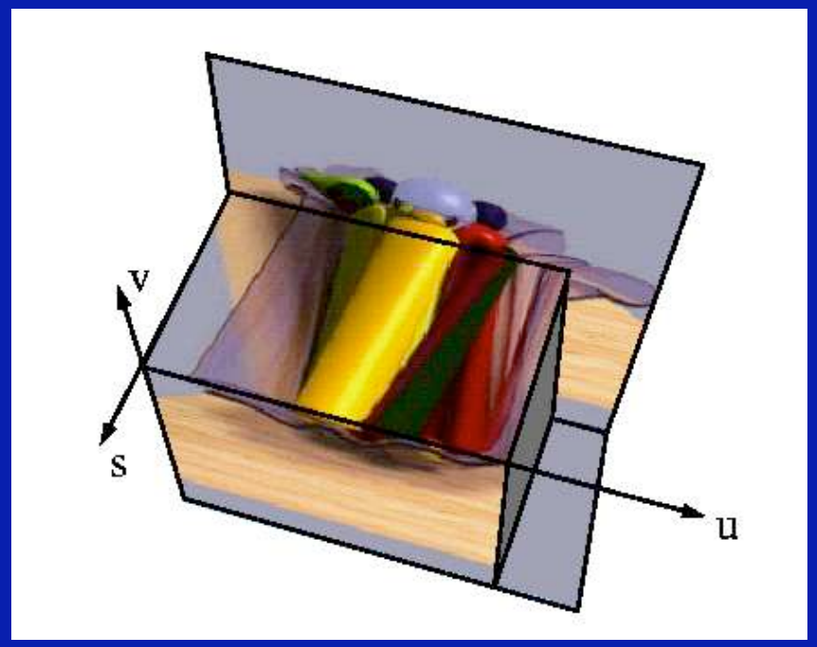
2 plane parameterization

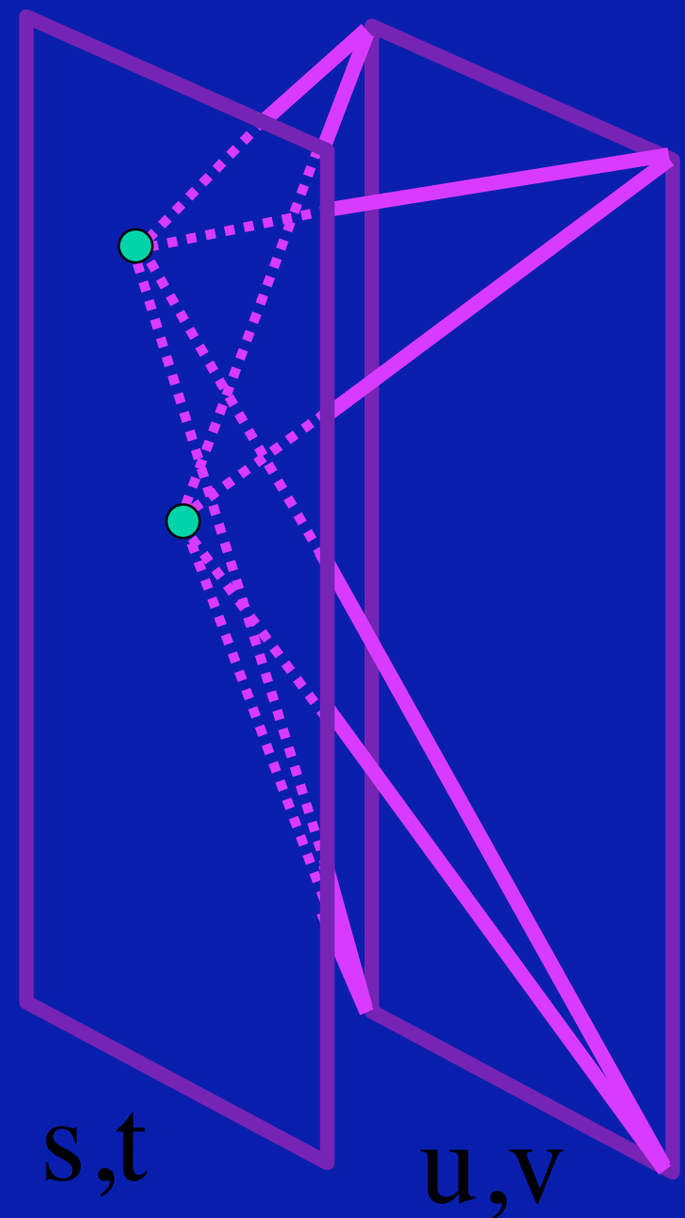
Lumigraph - Organization

Hold s,t constant

Let u,v vary

An image

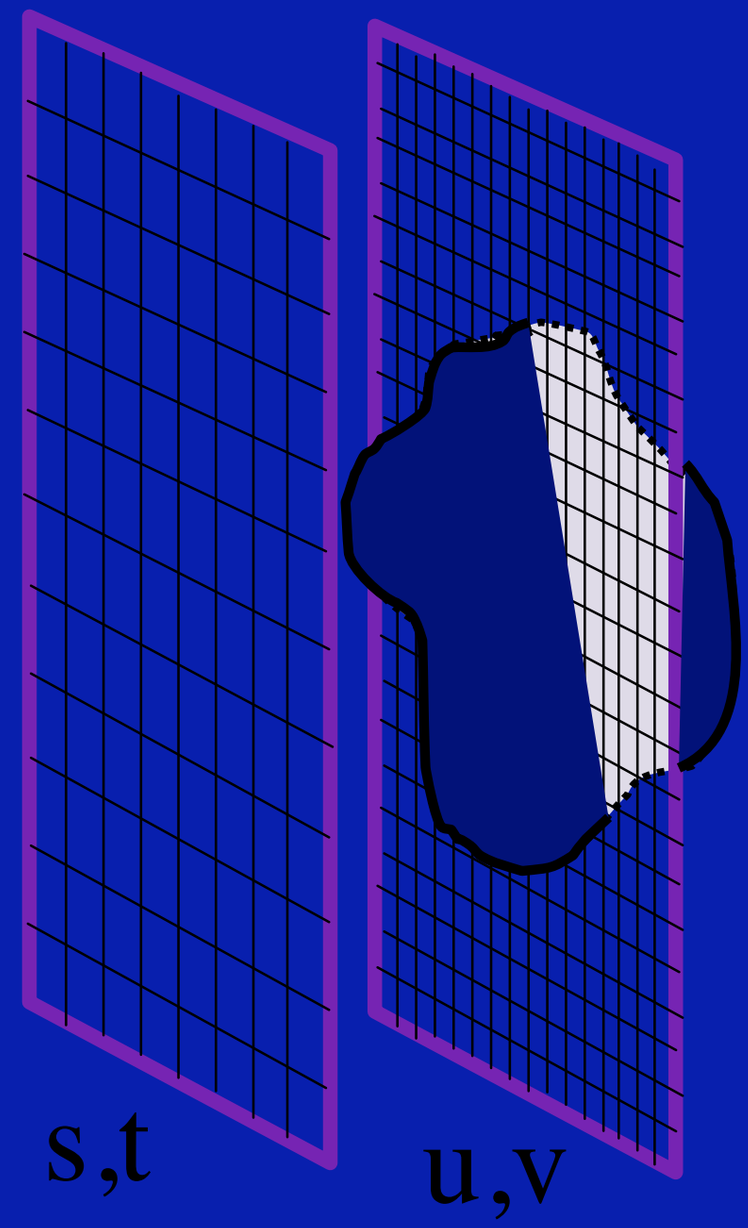




Lumigraph - Organization

Discretization

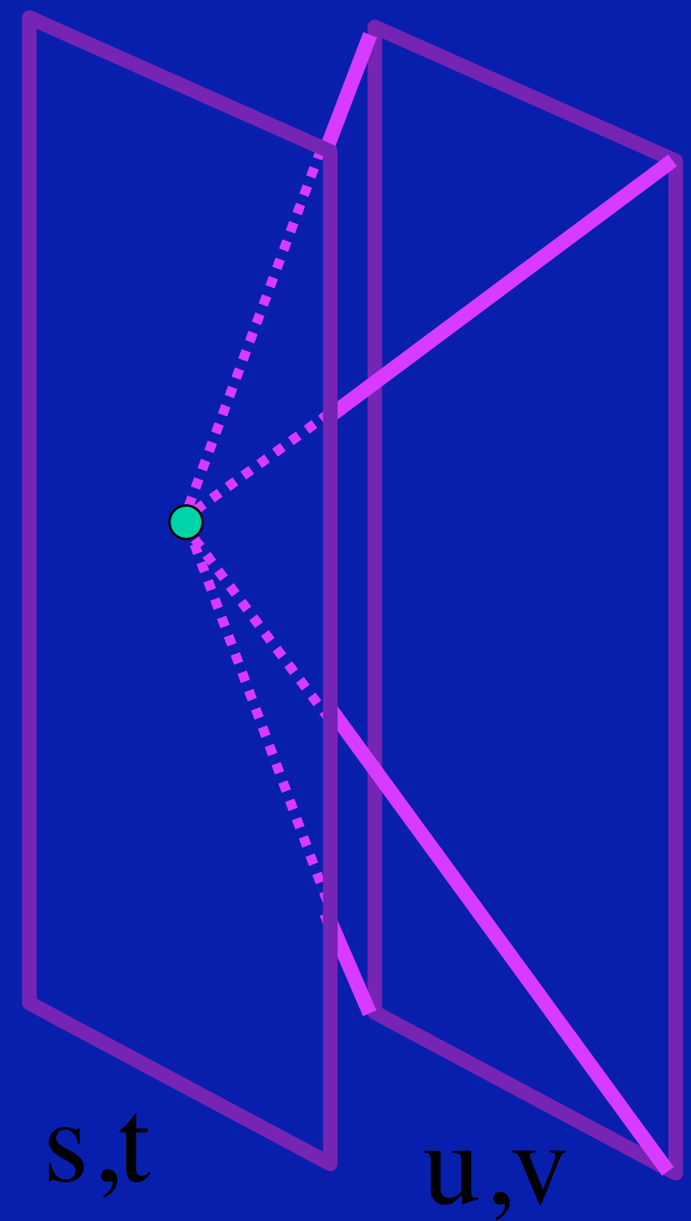
- higher res near object
  - if diffuse
  - captures texture
- lower res away
  - captures directions



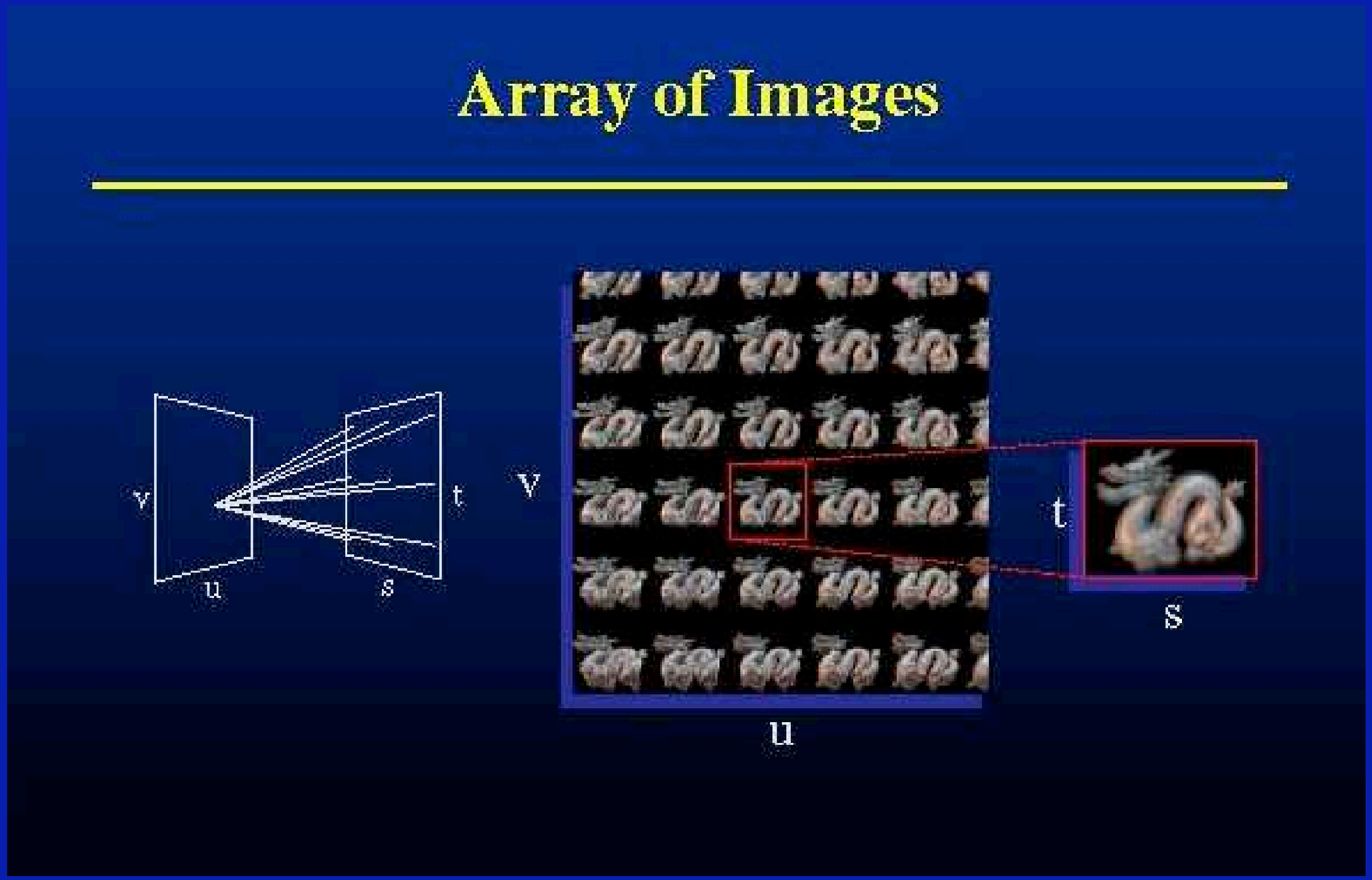
Lumigraph - Capture

Idea 1

- Move camera carefully over s,t plane
- Light Field



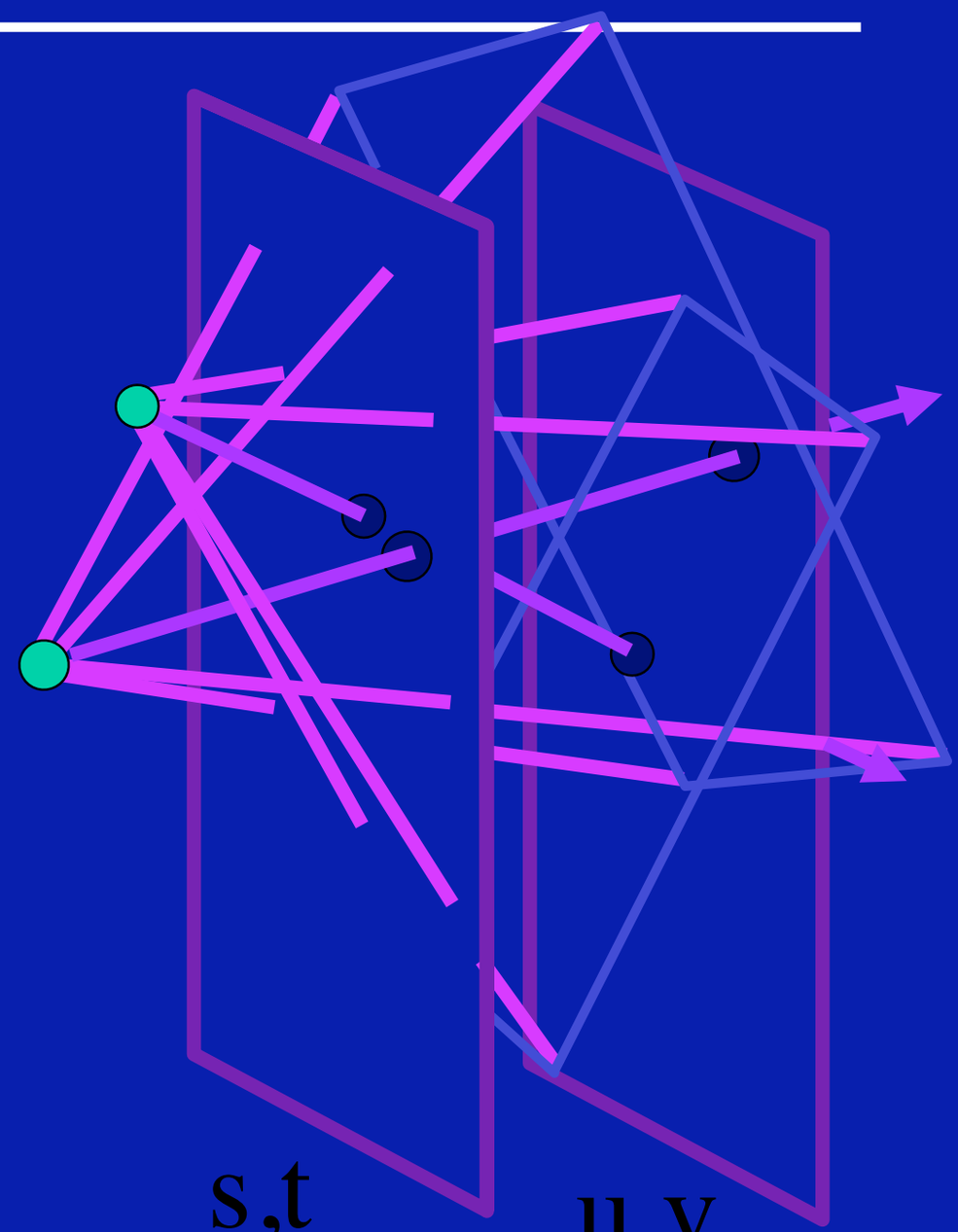
Lumigraph - Capture



Lumigraph - Capture

Idea 2

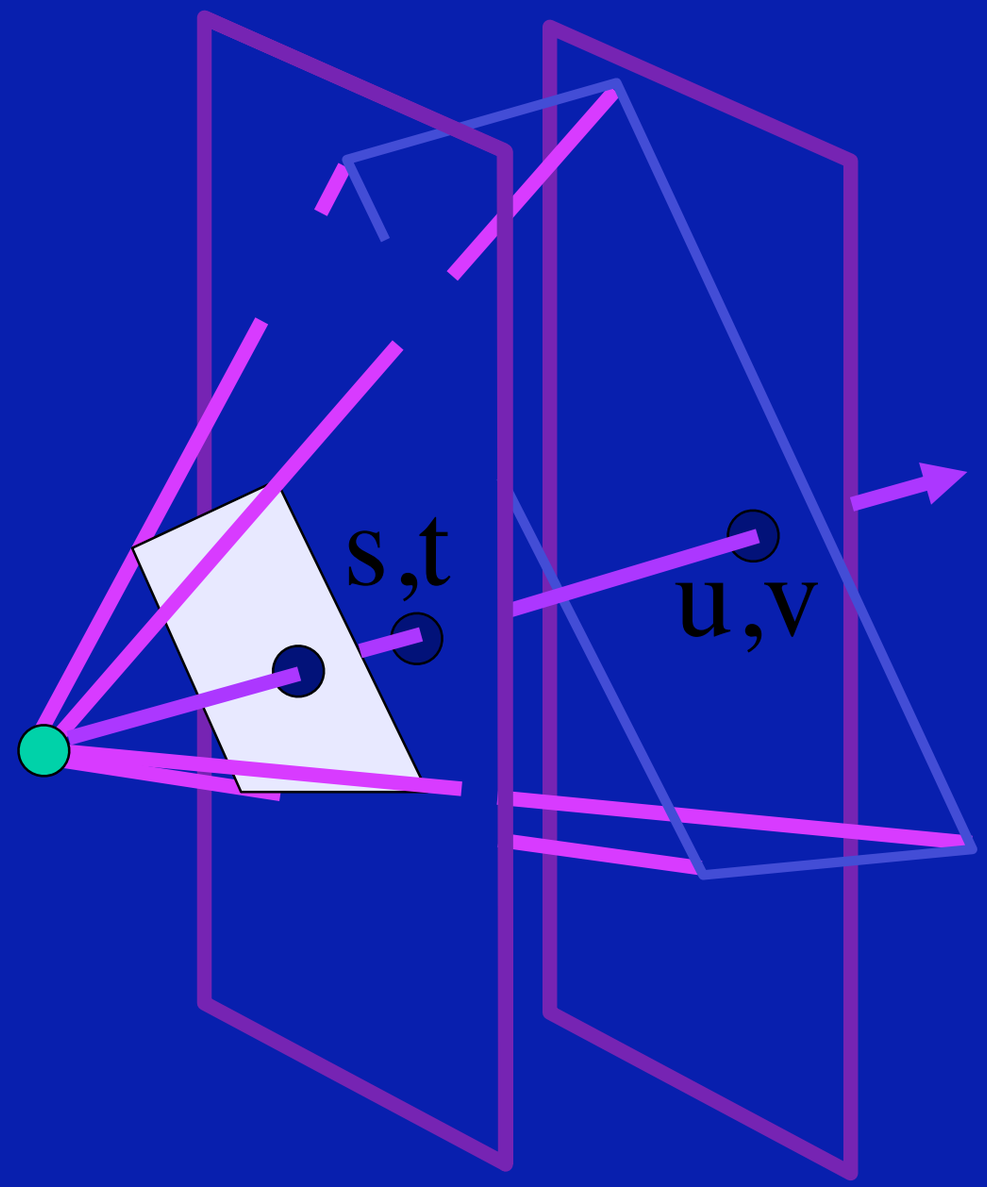
- Move camera anywhere
- Lumigraph paper



Lumigraph - Rendering

For each output pixel

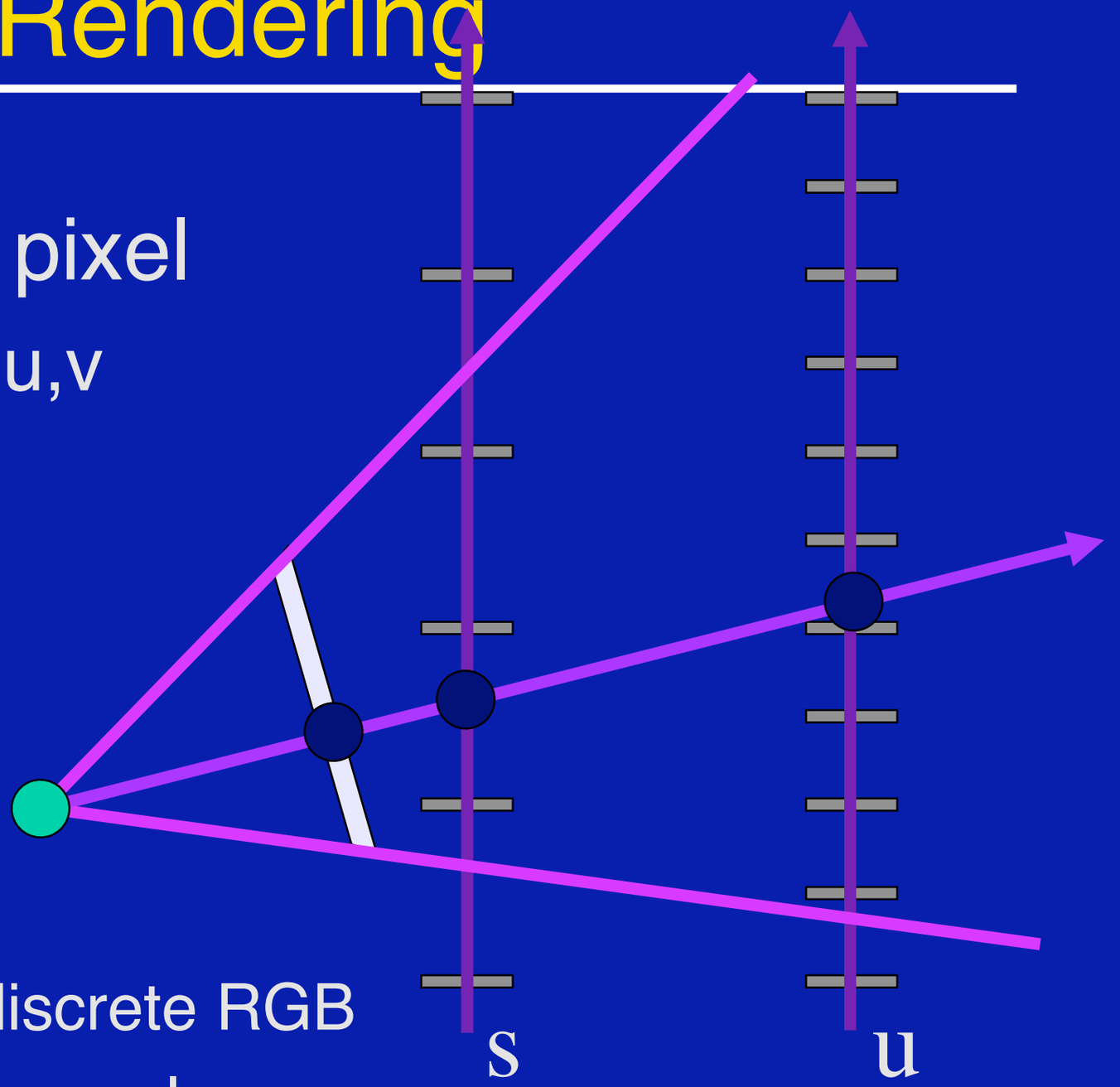
- determine s,t,u,v
- either
  - find closest discrete RGB
  - interpolate near values



Lumigraph - Rendering

For each output pixel

- determine s,t,u,v



- either
  - use closest discrete RGB
  - interpolate near values

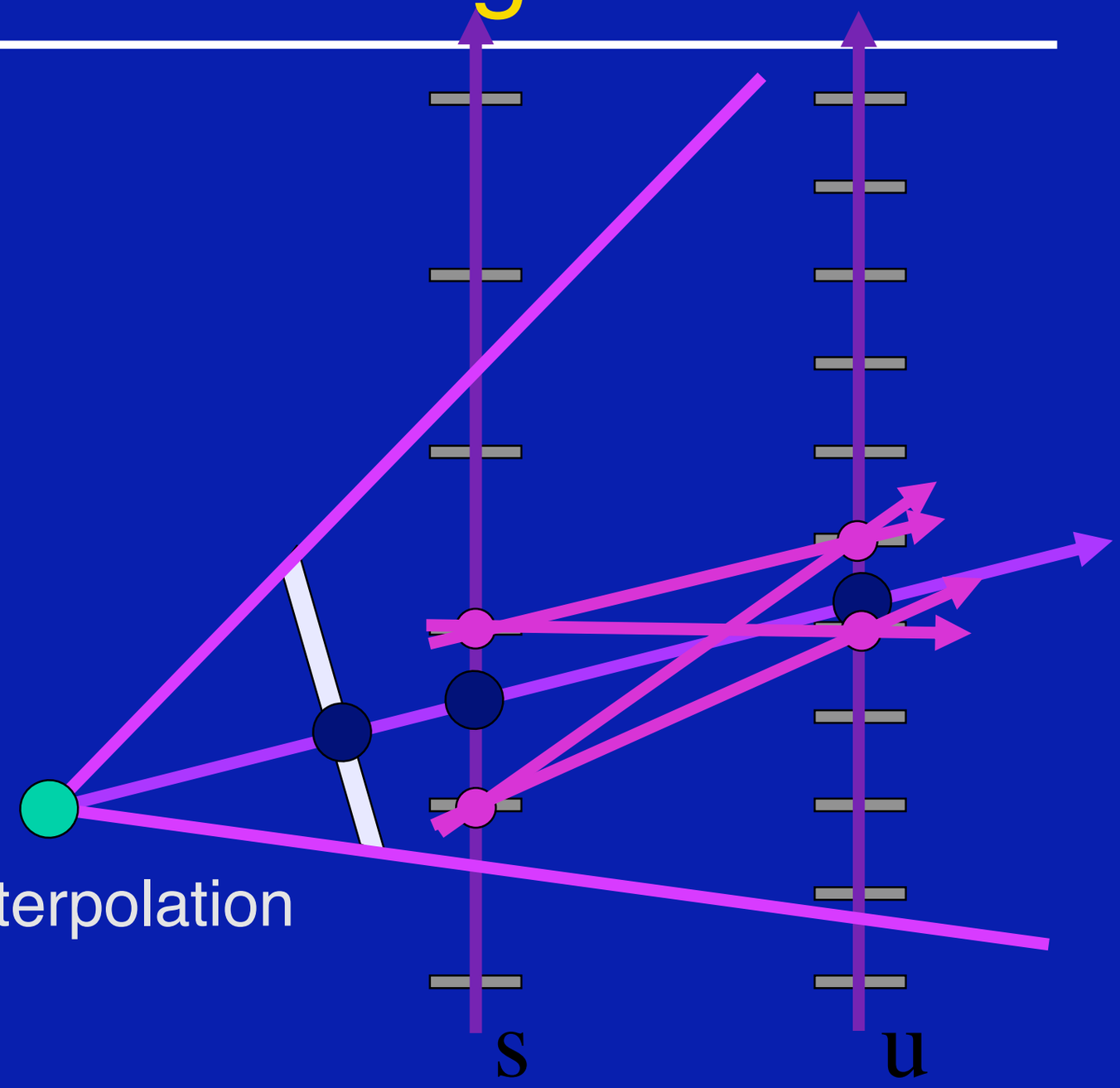
Lumigraph - Rendering

Nearest

- closest s
- closest u
- draw it

Blend 16 nearest

- quadrilinear interpolation



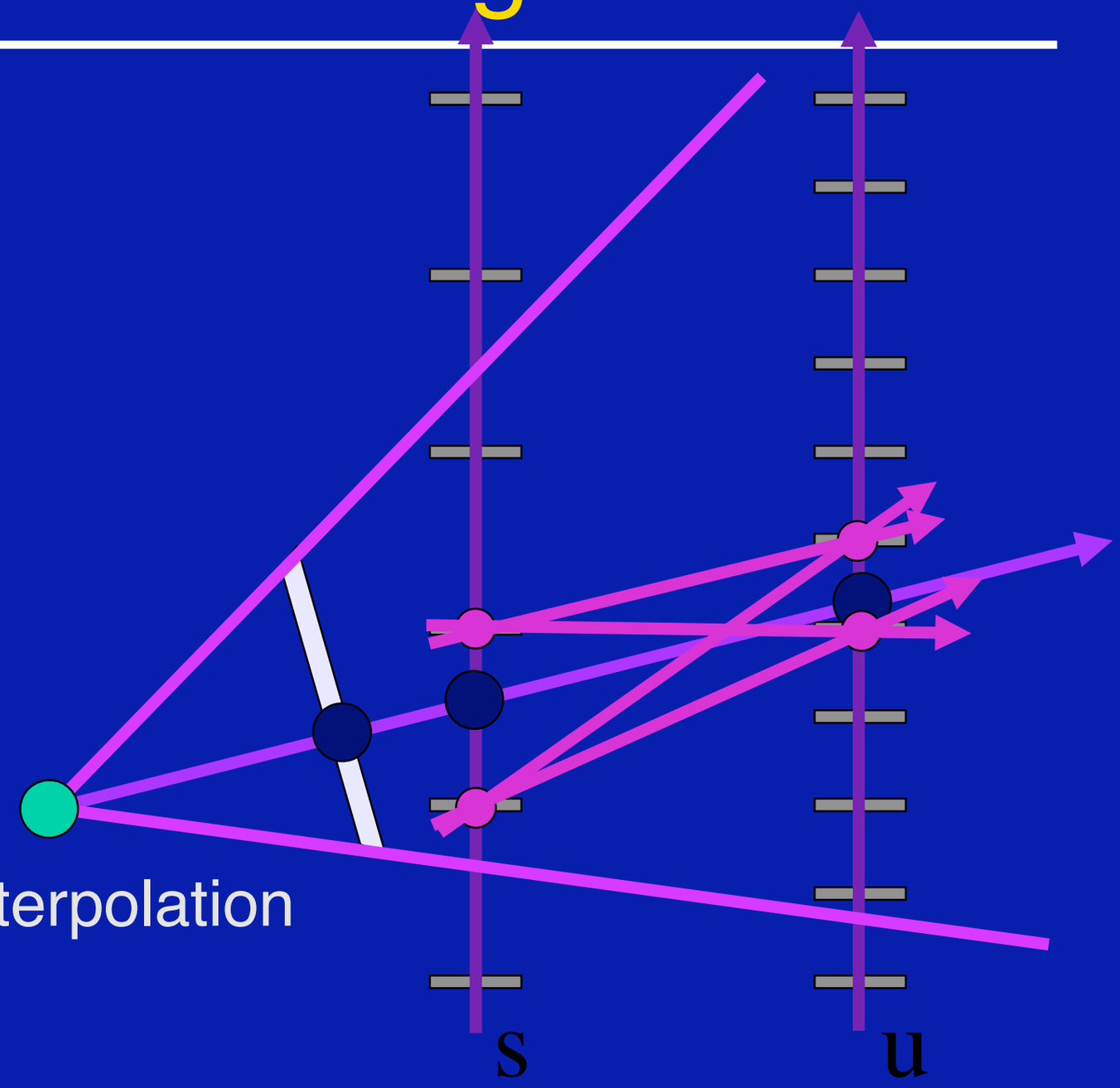
Lumigraph - Rendering

Nearest

- closest s
- closest u
- draw it

Blend 16 nearest

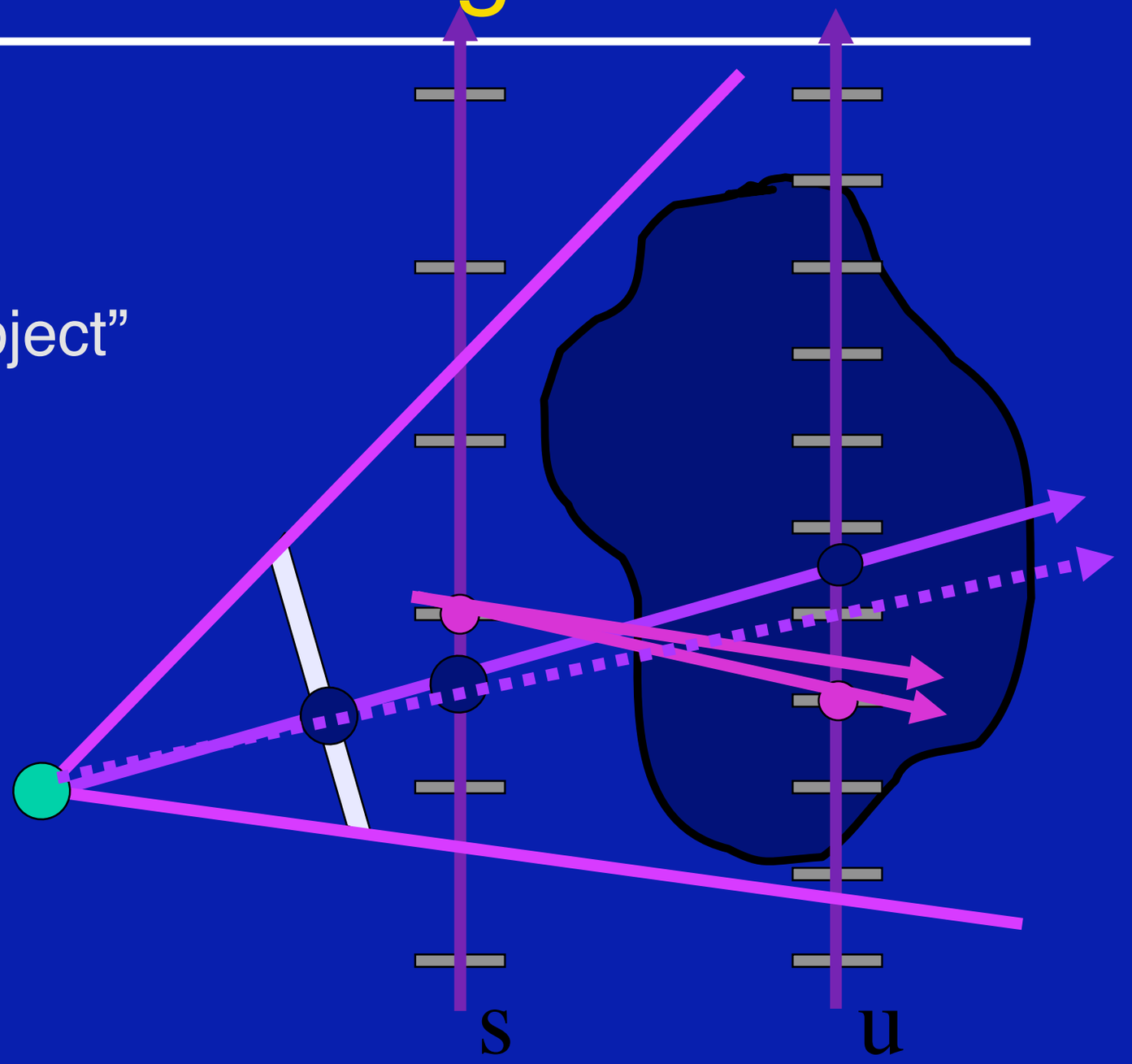
- quadrilinear interpolation



Lumigraph - Rendering

Depth Correction

- closest s
- intersection with “object”
- best u
- closest u

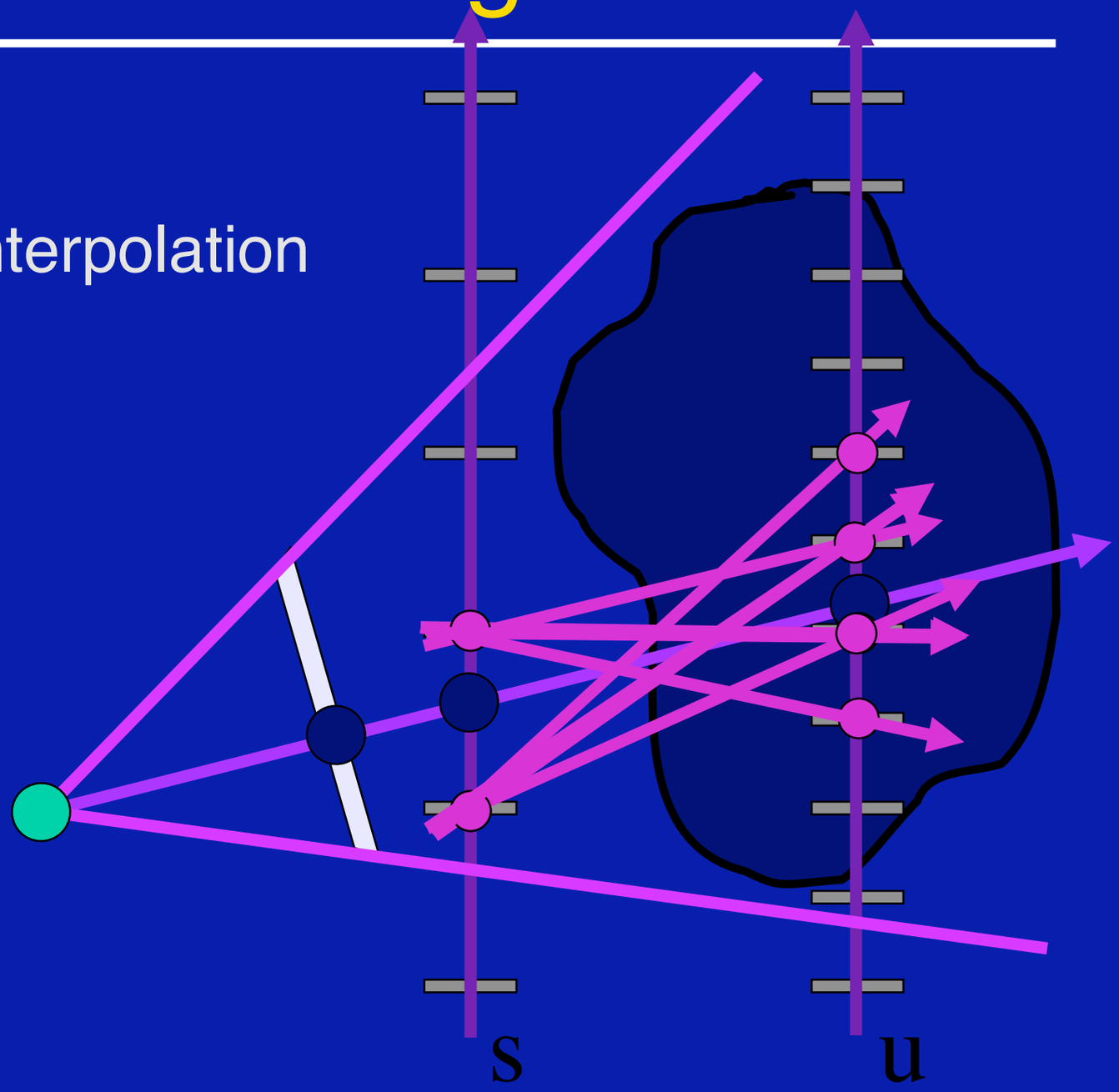


Lumigraph - Rendering

Depth Correction

- quadralinear interpolation
- new “closest”
- like focus

[Dynamically Reparameterized Light Fields, Isaksen, SG’2000]

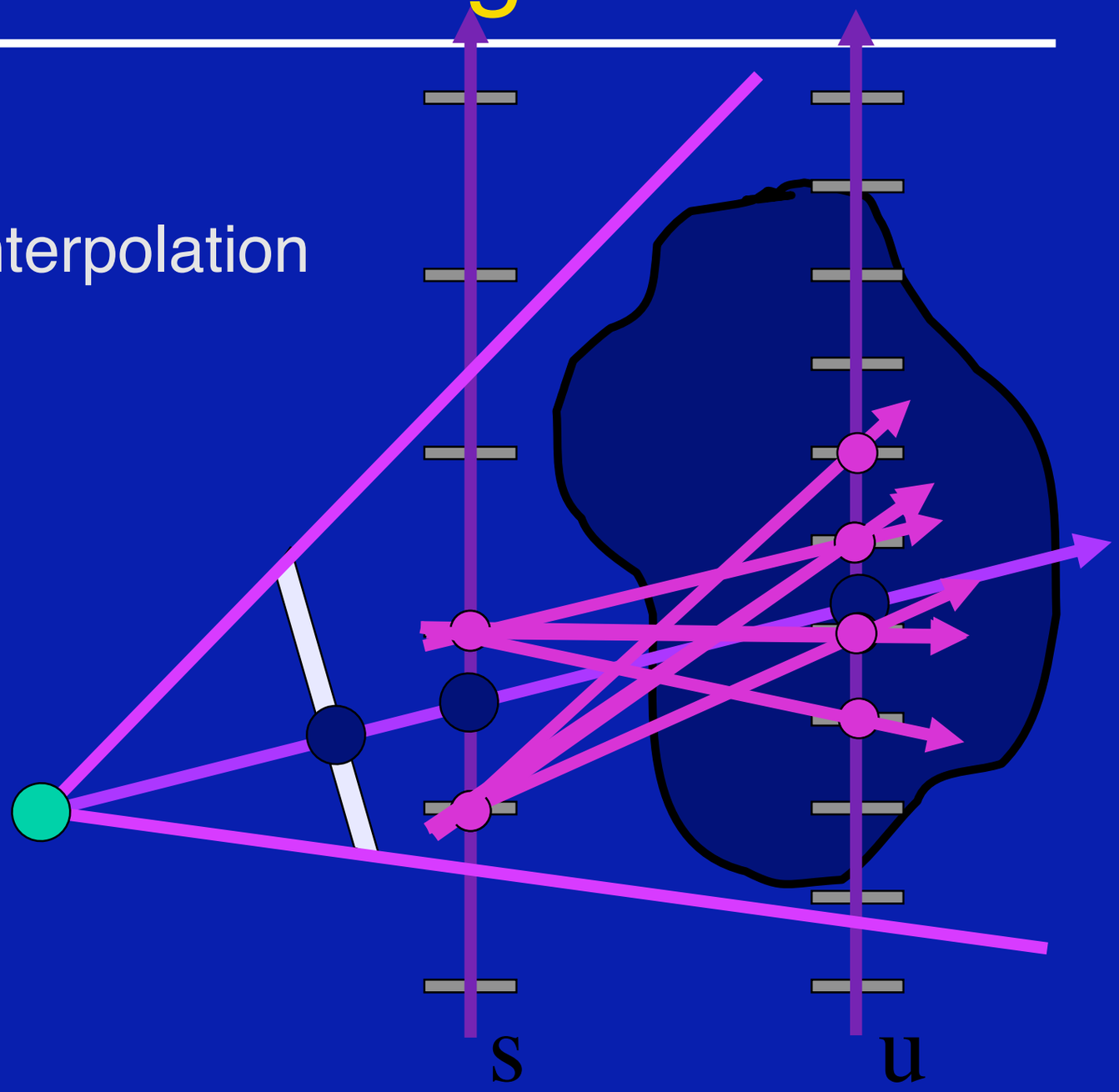


Lumigraph - Rendering

Depth Correction

- quadralinear interpolation
- new “closest”
- like focus

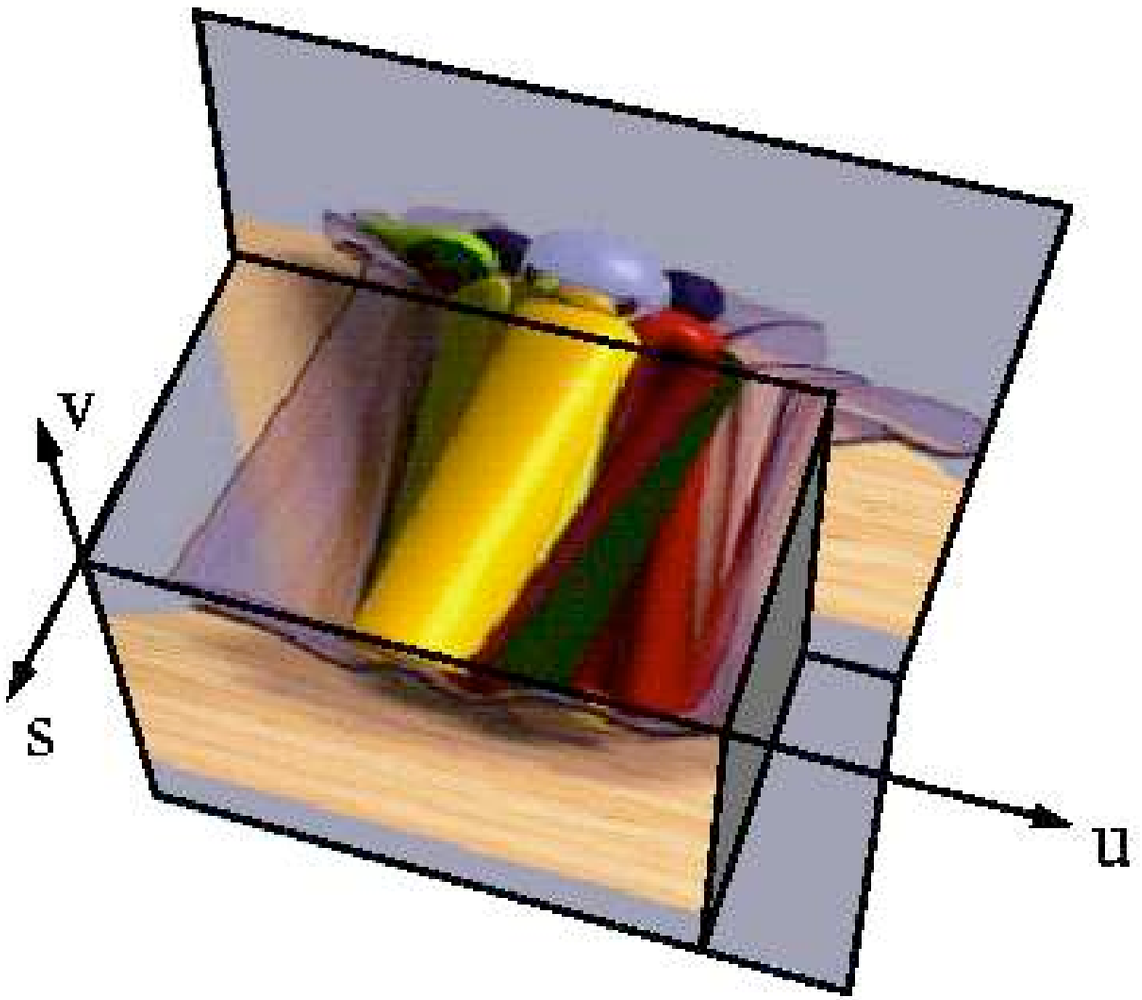
[Dynamically Reparameterized Light Fields, Isaksen, SG’2000]



Lumigraph - Ray Space

Image effects:

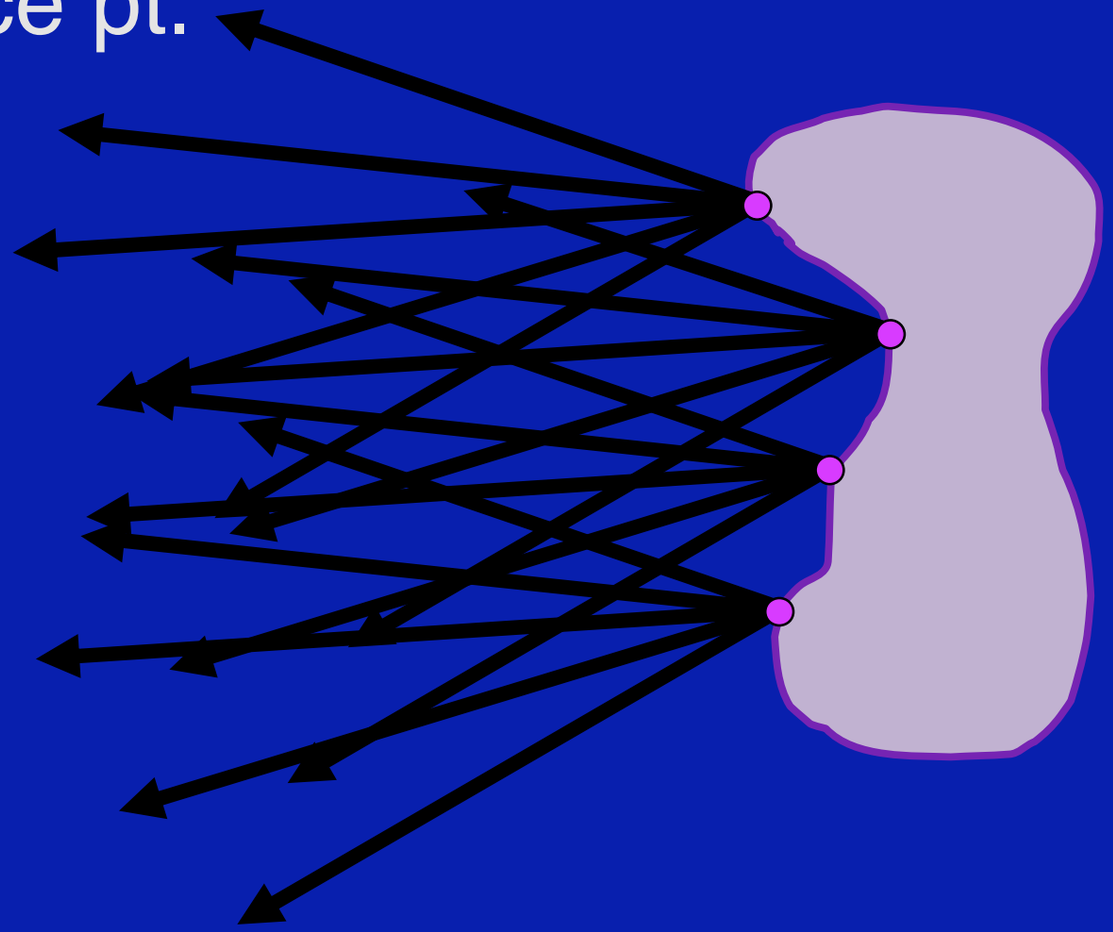
- parallax
- occlusion
- transparency
- highlights



Surface Light Fields

Turn 4D parameterization around:

image @ every surface pt.

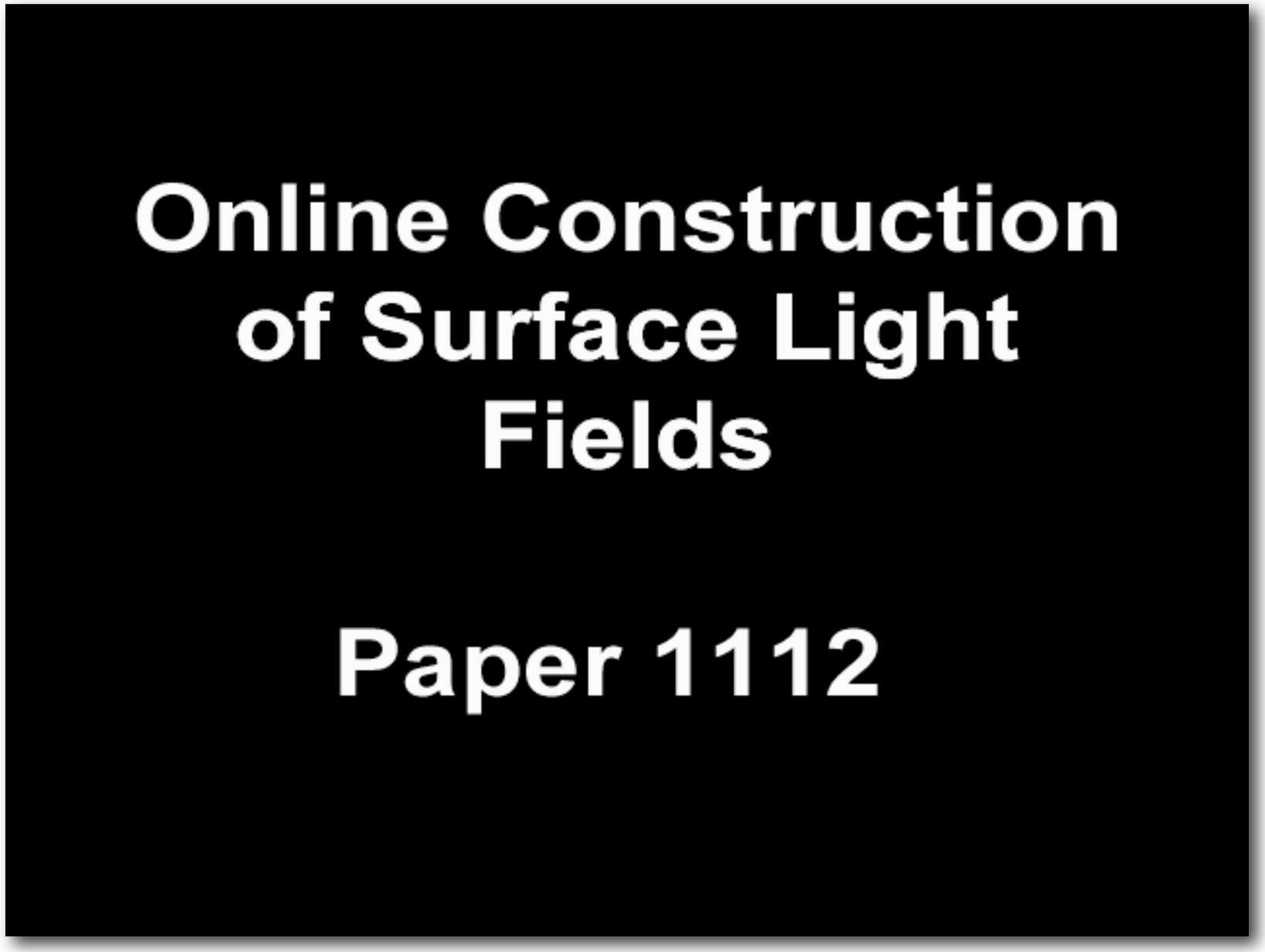


Leverage coherence:

compress radiance fn
(BRDF * illumination)
after rotation by n

Surface Light Fields

[Wood et al, SIGGRAPH 2000]



source: *Online Construction of Surface Light Fields*. Greg Coombe, Chad Hantak, Anselmo Lastra, Radek Grzeszczuk.

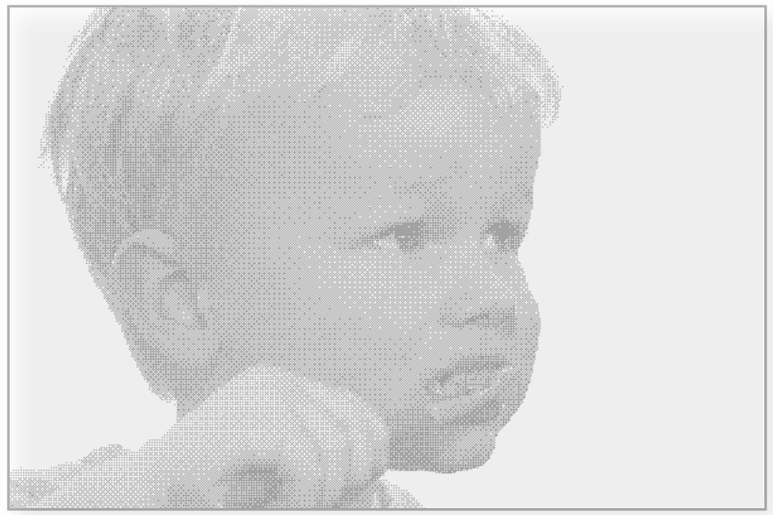
Hierarchy of Light Fields [Levoy]

5D: Plenoptic Function (Ray)
4D: Lumigraph / Lightfield
4D*: Environment Matte (single view)
3D: Lumigraph Subset
3D: Concentric Mosaics
2.5D: Layered Depth Image
2.5D: Image Based Models
2D: Images and Panoramas

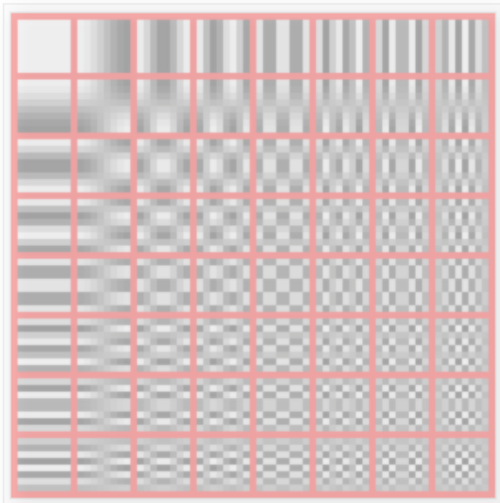
Panoramic Video Textures

Overview

Announcements



Dithering



Compression

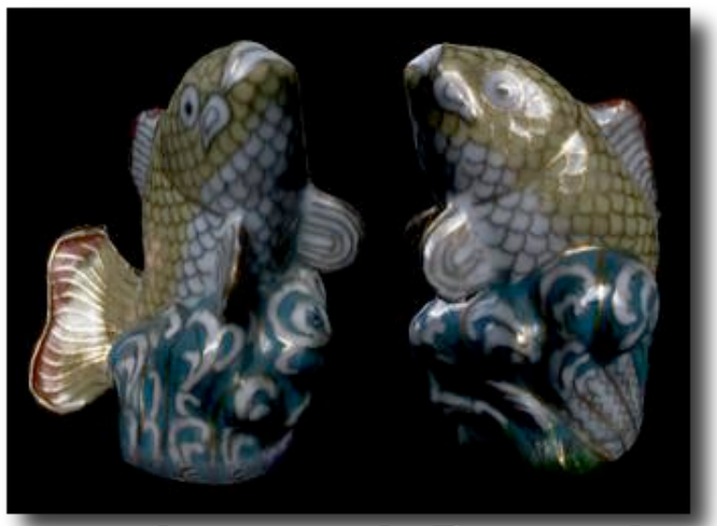
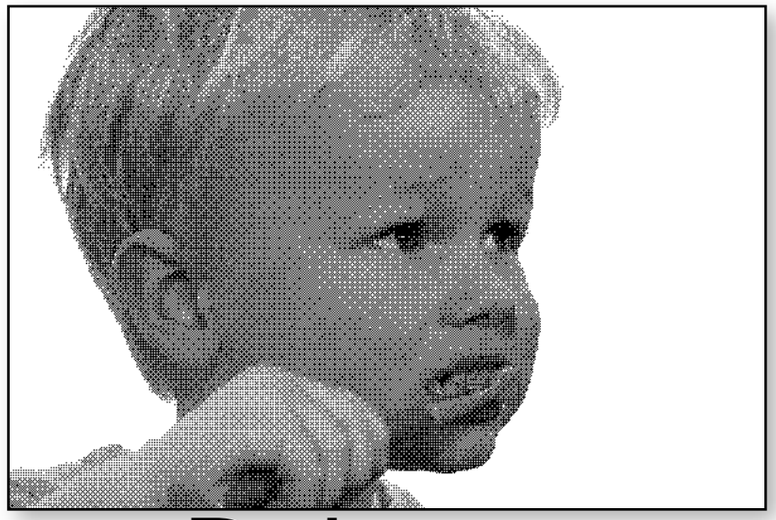


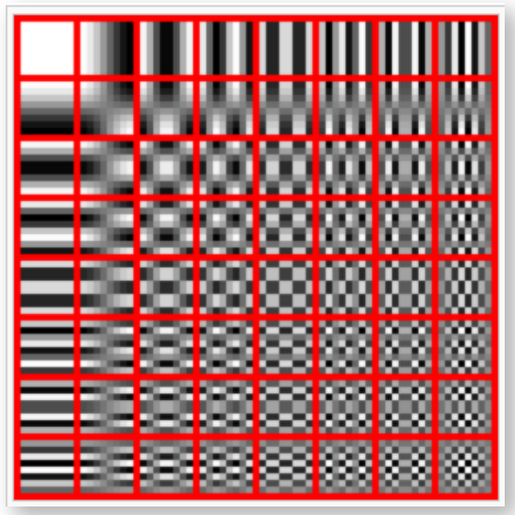
Image-based Rendering

Overview

Announcements



Dithering



Compression

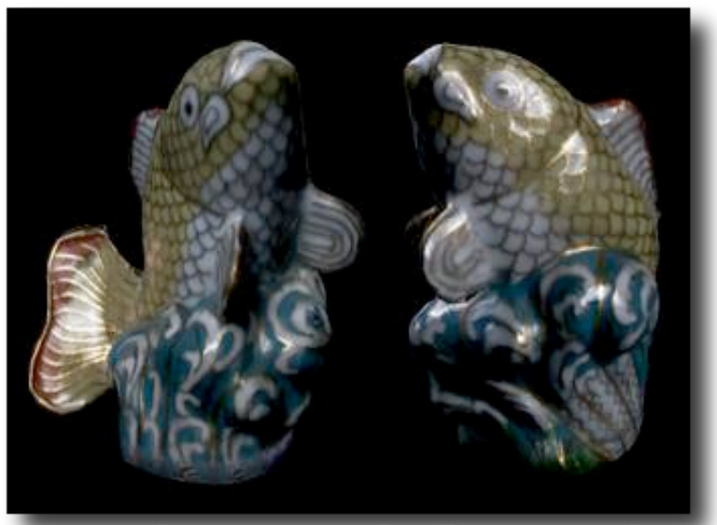


Image-based Rendering