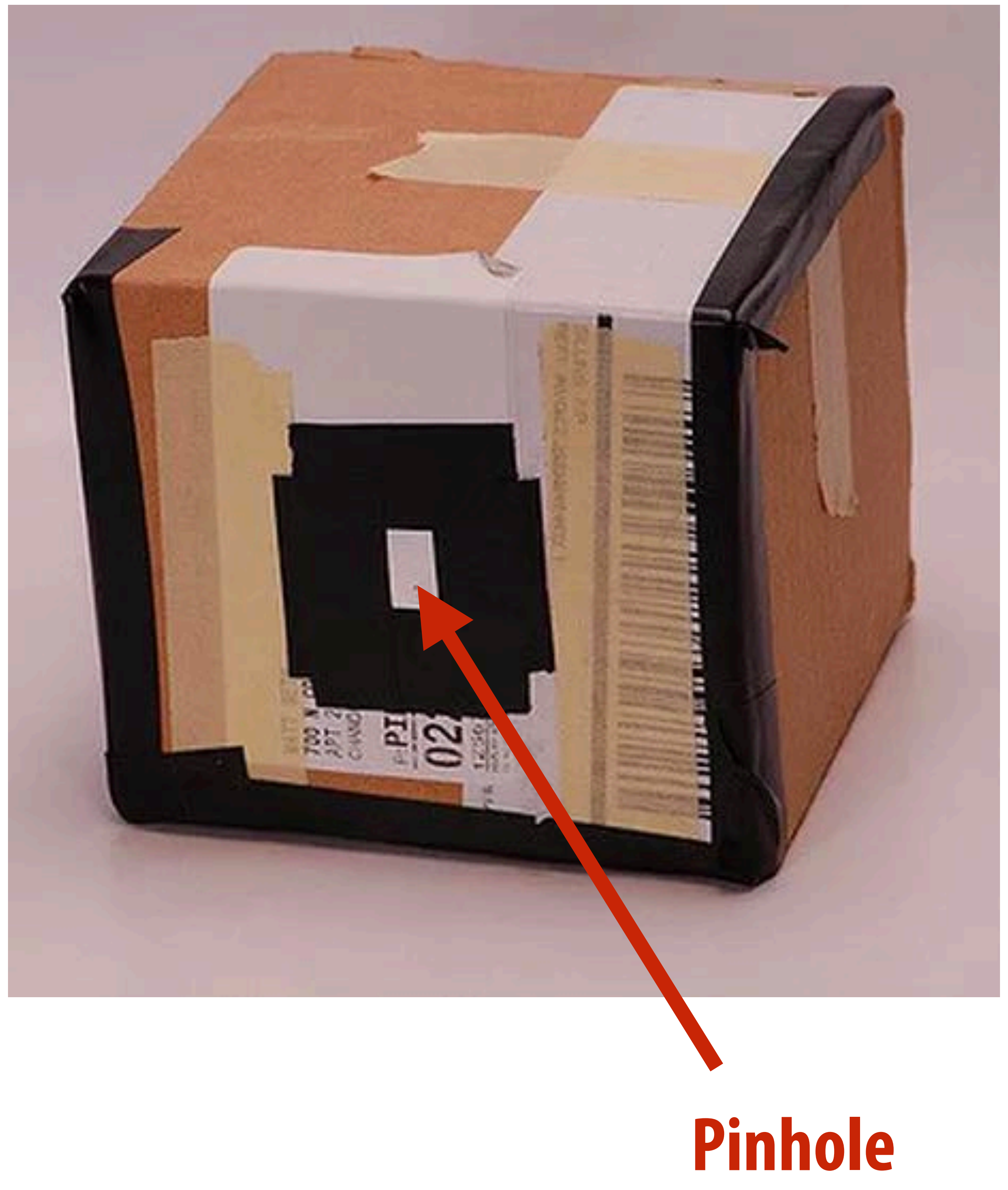
Lecture 12:

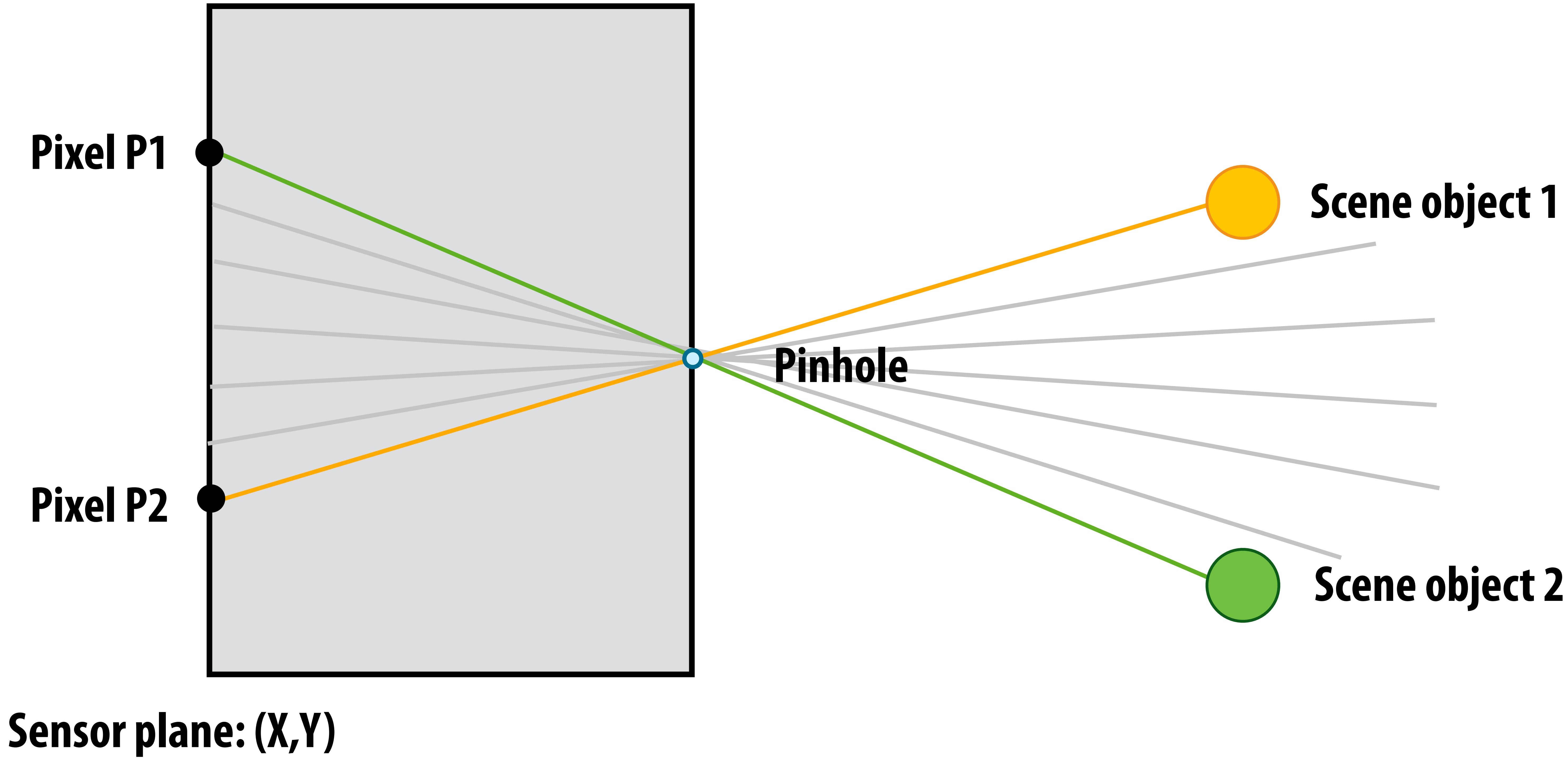
Background: the light field and rendering basics

Visual Computing Systems

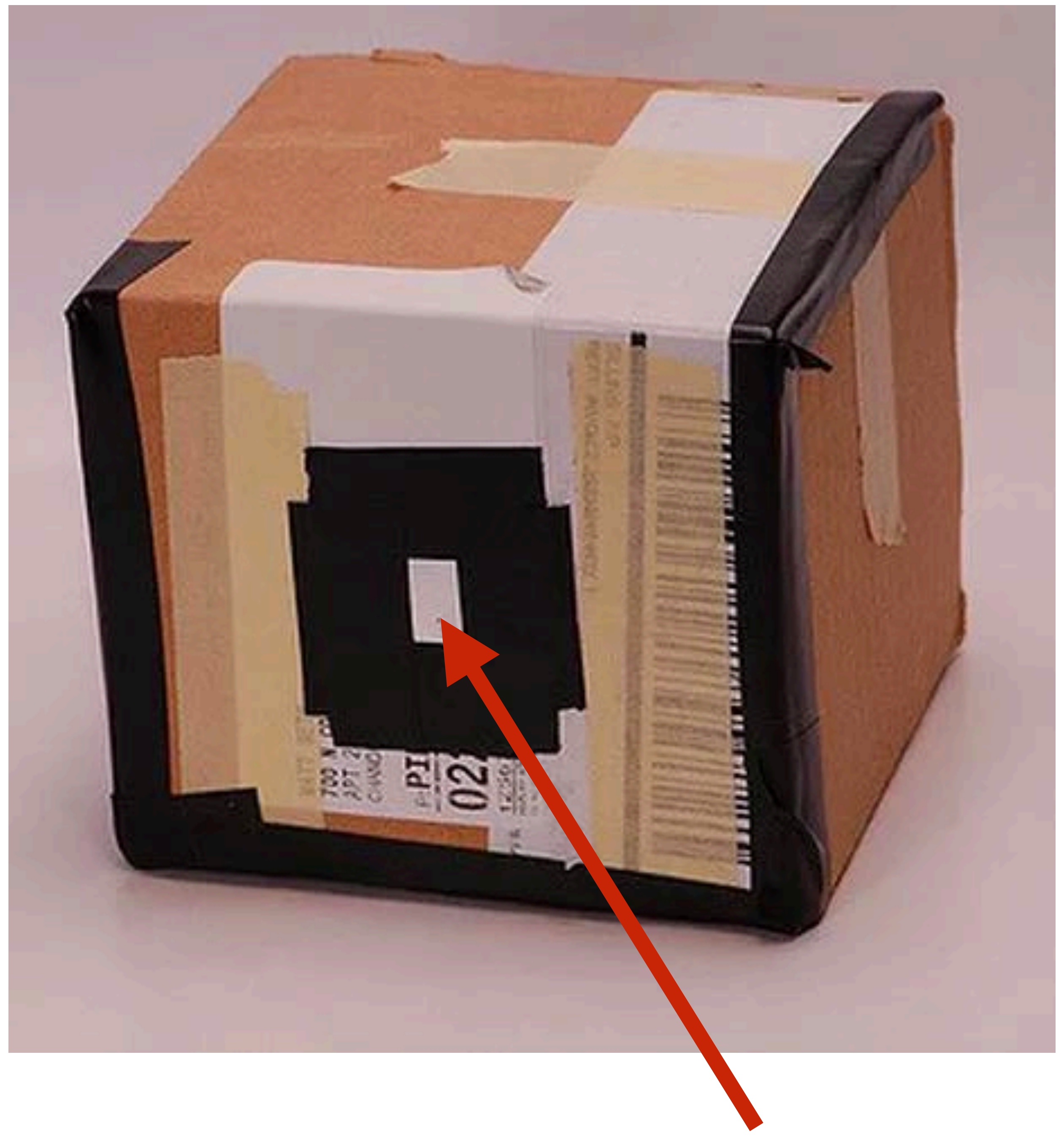
Stanford CS348K, Spring 2022

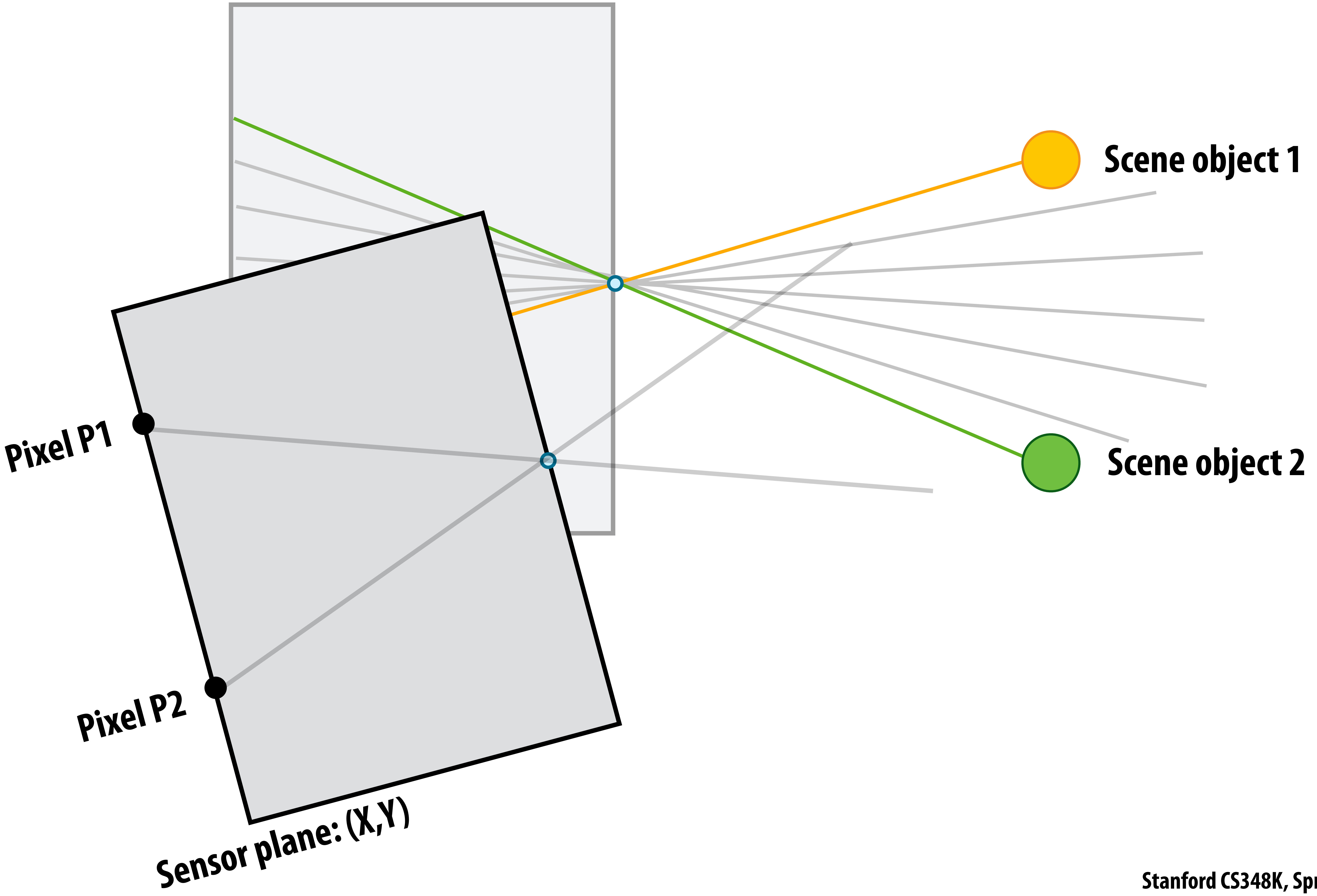
Recall basic pinhole camera





What about taking the pictures from a new viewpoint?

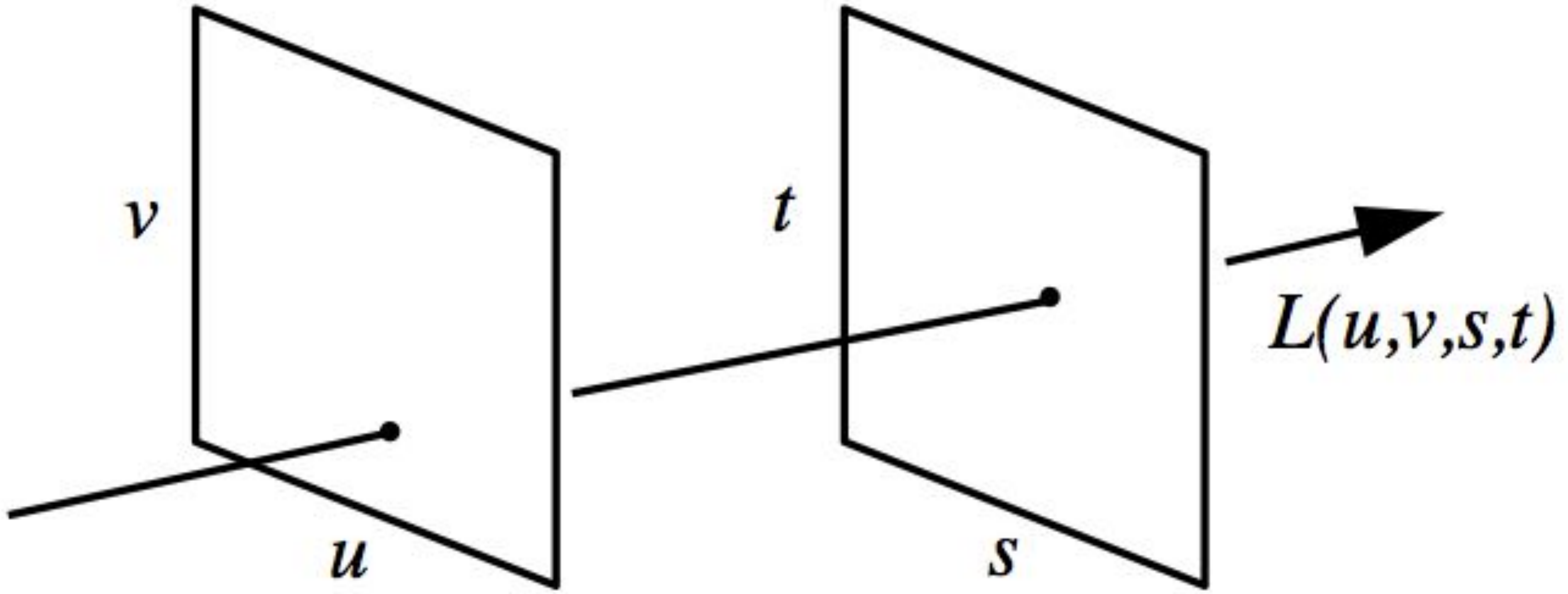




Light-field parameterization

[Levoy and Hanrahan 96]
[Gortler et al., 96]

Light field as a 4D function (represents light in free space: no occlusion)



[Image credit: Levoy and Hanrahan 96]

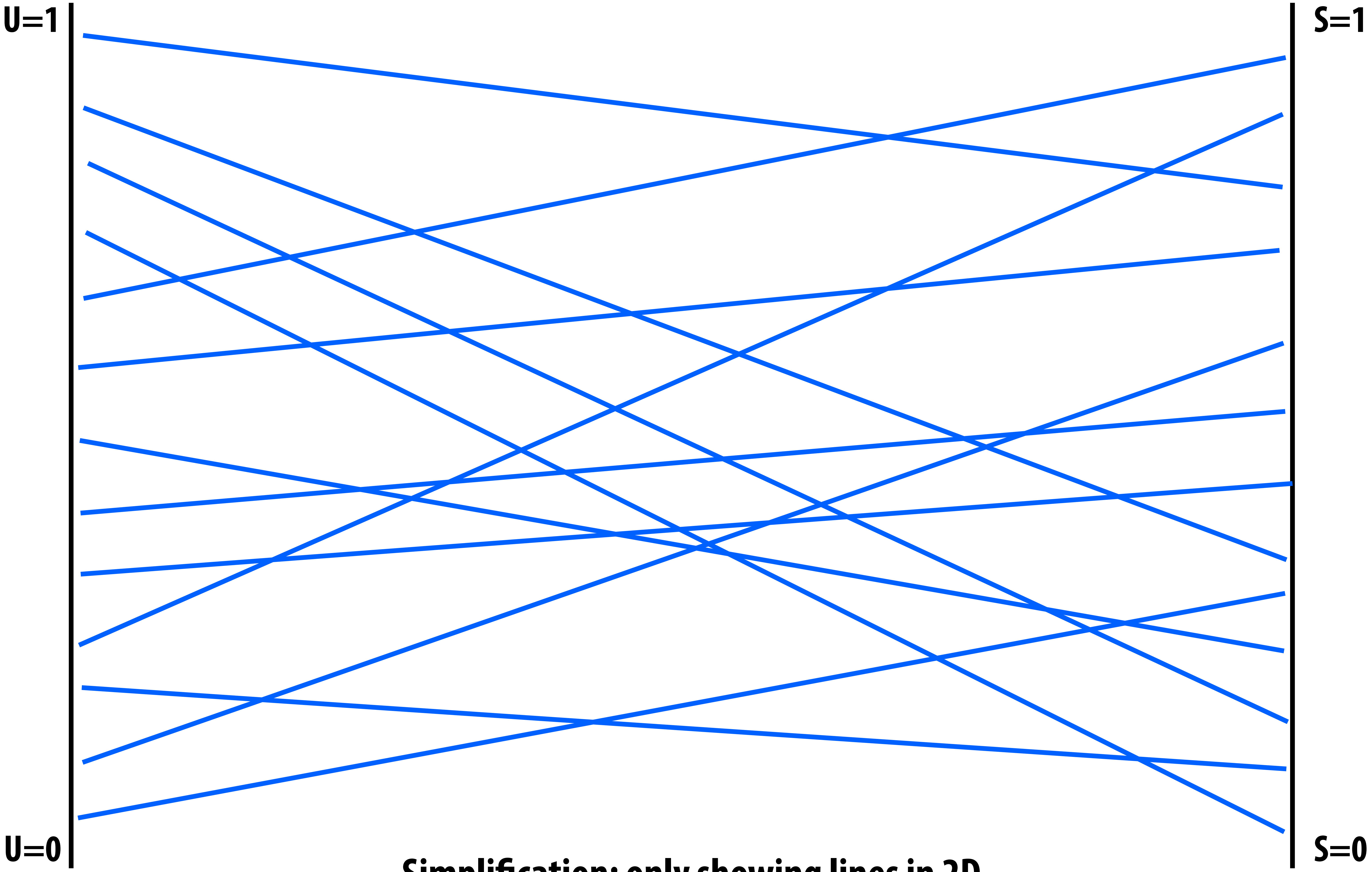
Efficient two-plane parameterization

Line described by connecting point on (u,v) plane with point on (s,t) plane

If one of the planes placed at infinity: point + direction representation

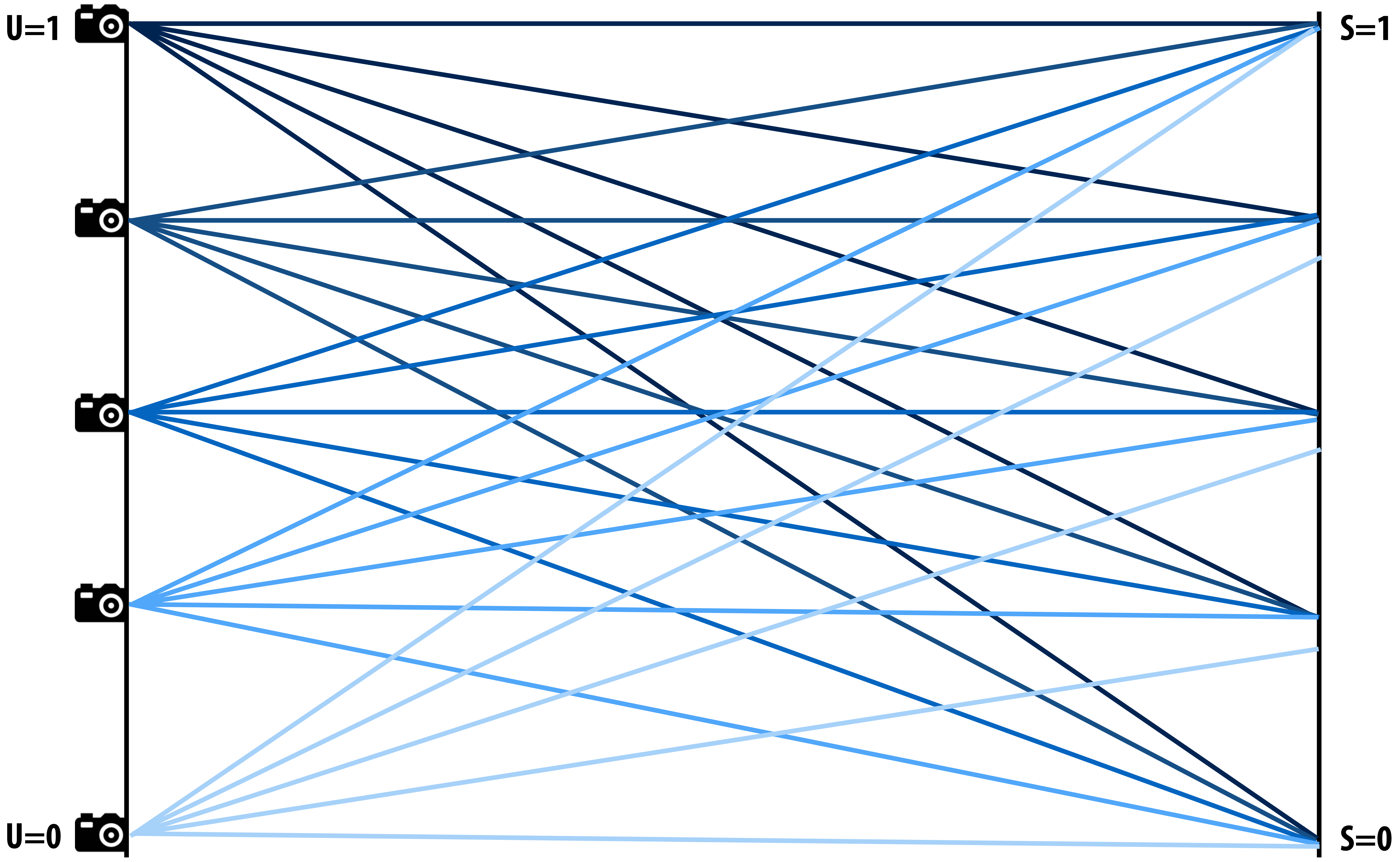
Levoy/Hanrahan refer to representation as a "light slab": beam of light entering one quadrilateral and exiting another

Sampling the light field



Simplification: only showing lines in 2D
(full light field is 4D function)

Measuring the light field by taking many pictures



[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

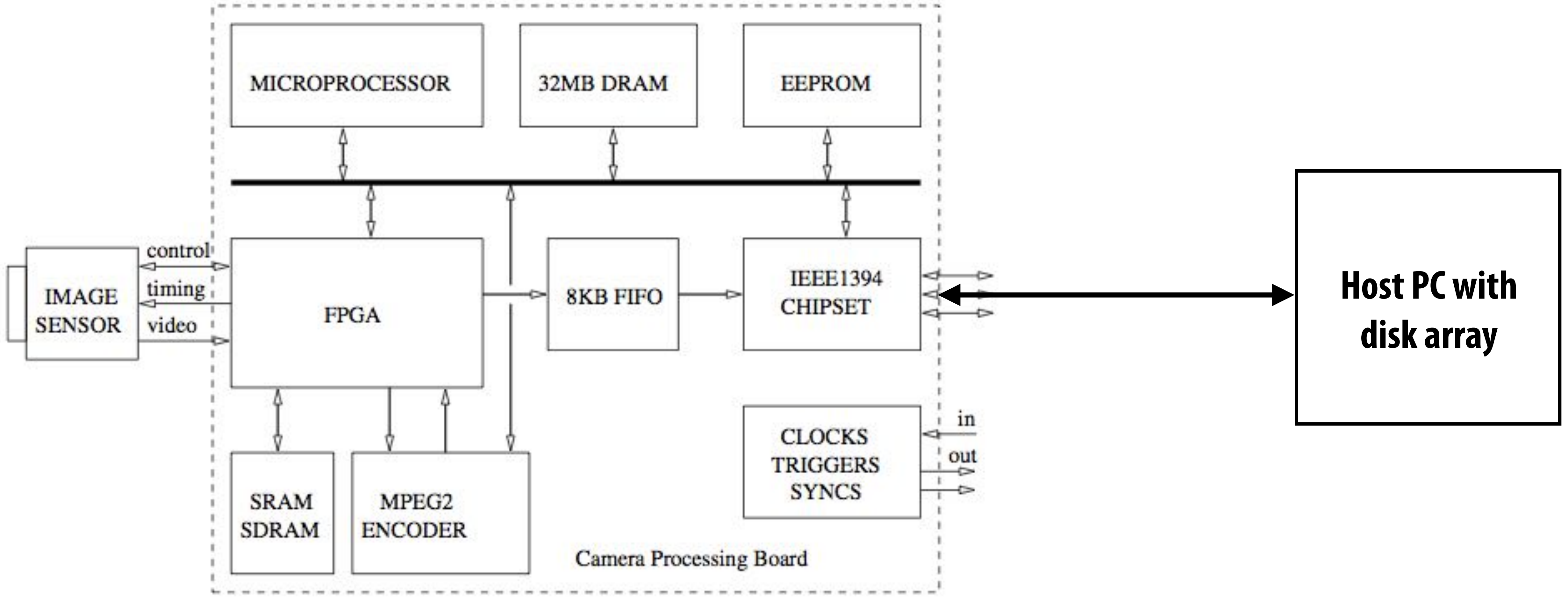
Stanford Camera Array

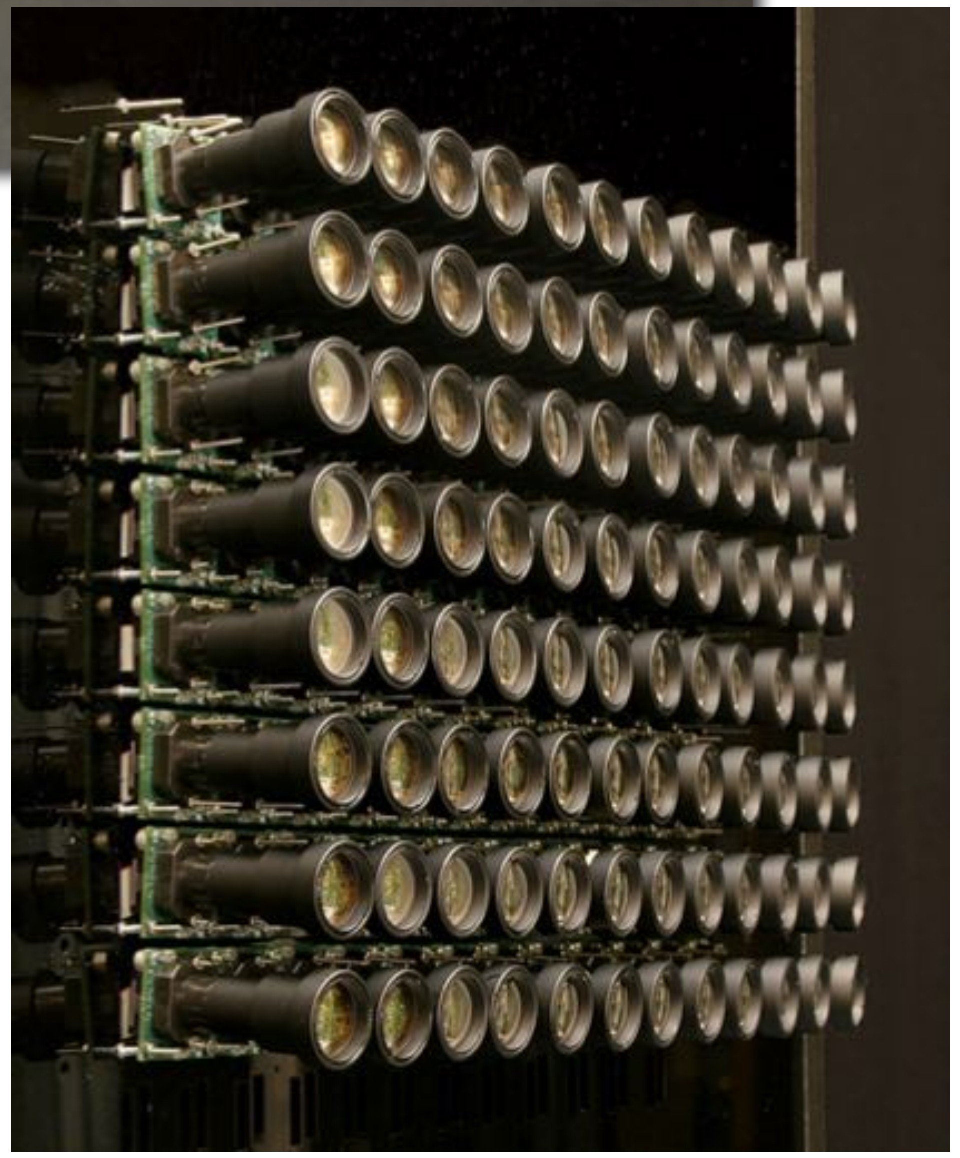
[Wilburn et al. 2005]

640 x 480 tightly synchronized, repositionable cameras

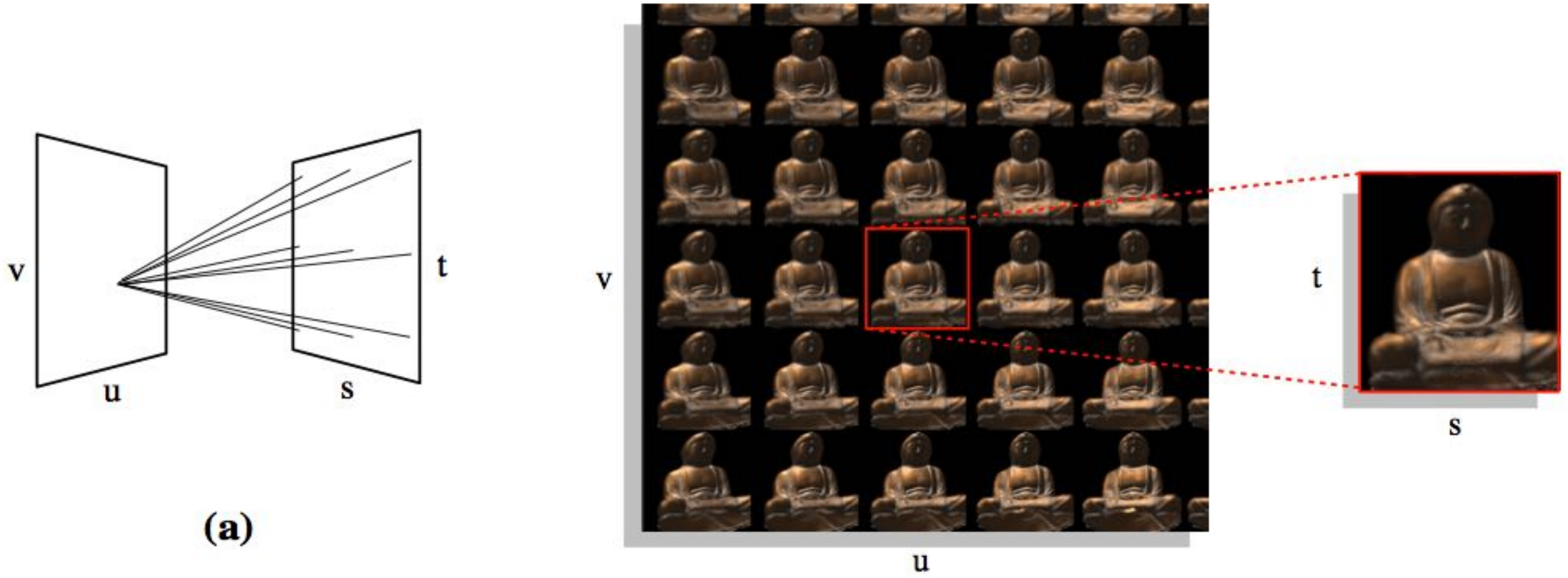
Custom processing board per camera

Tethered to PCs for additional processing/storage

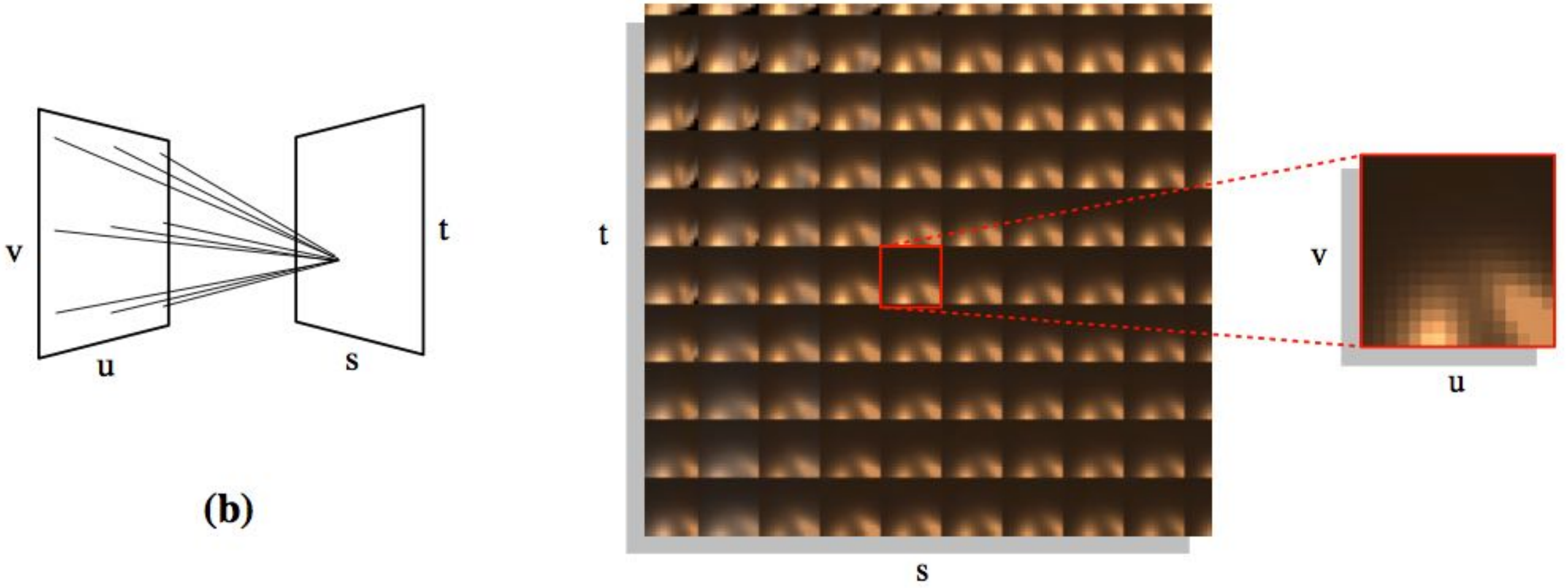




Light field storage layouts



(a)



(b)

Later light field cameras

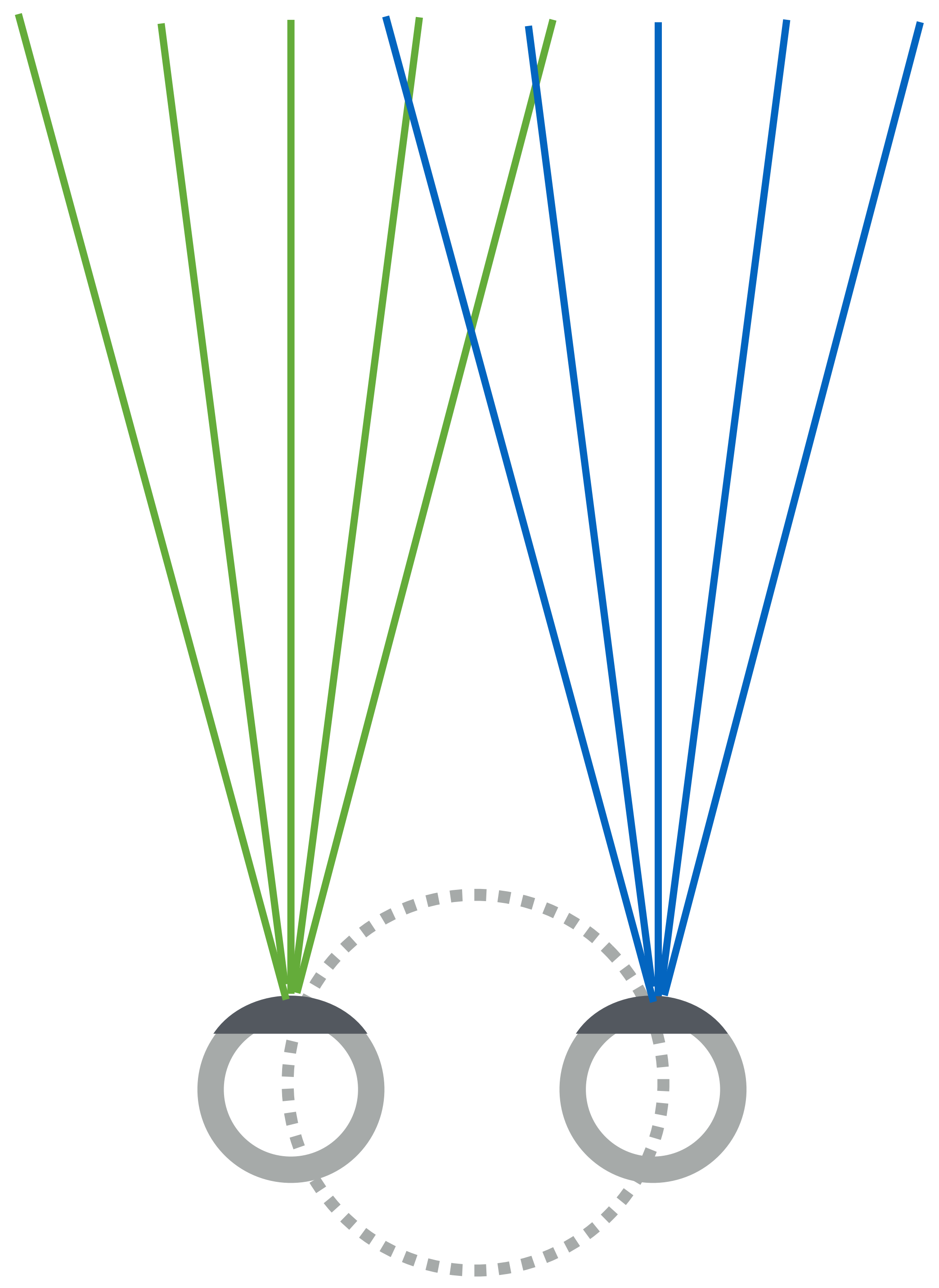
Lytro Illum

Acquiring light field content for VR

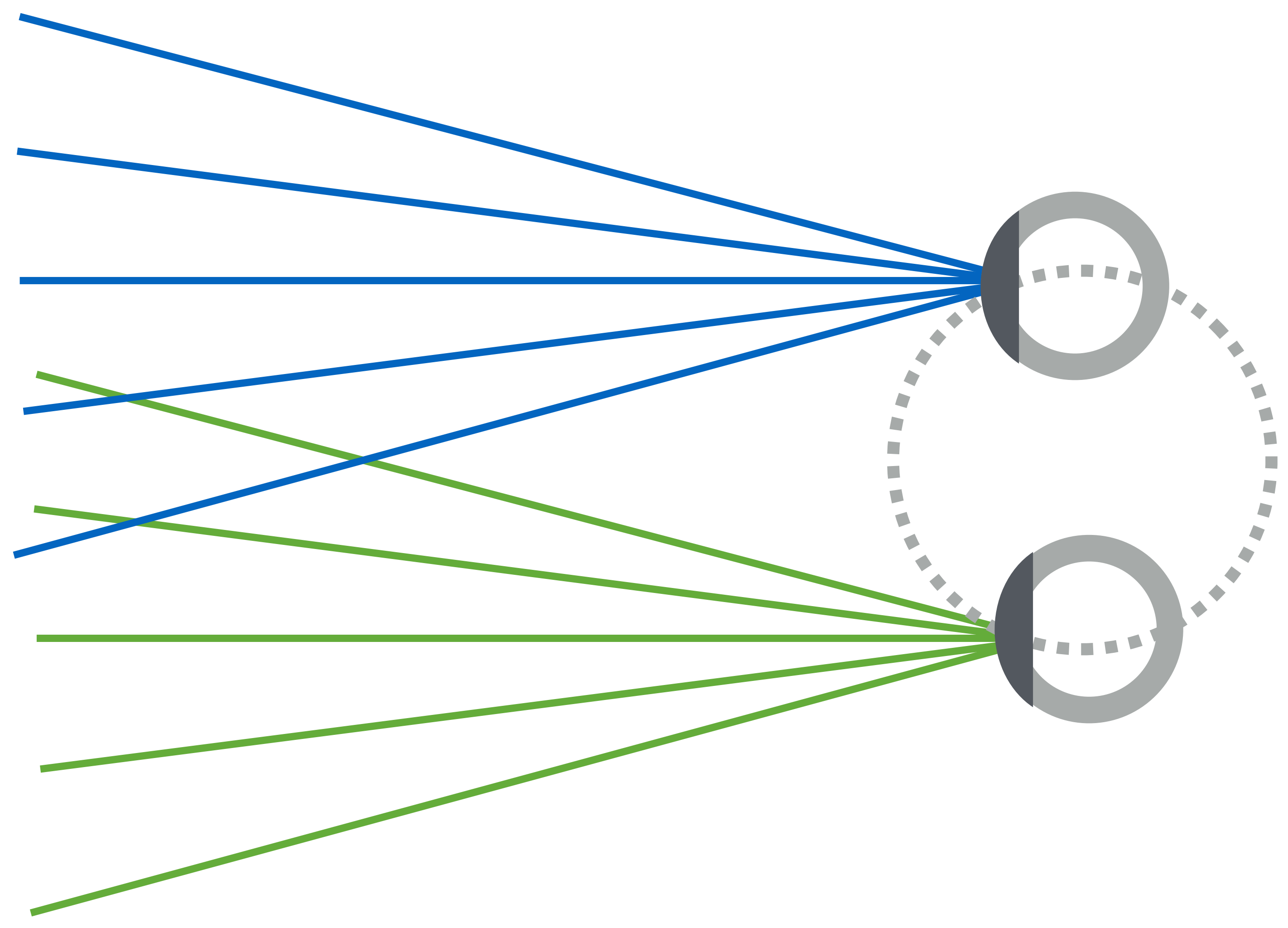
Google's Jump VR video:
Yi Halo Camera (17 cameras)

Facebook Manifold
(16 8K cameras)

Stereo, 360-degree viewing



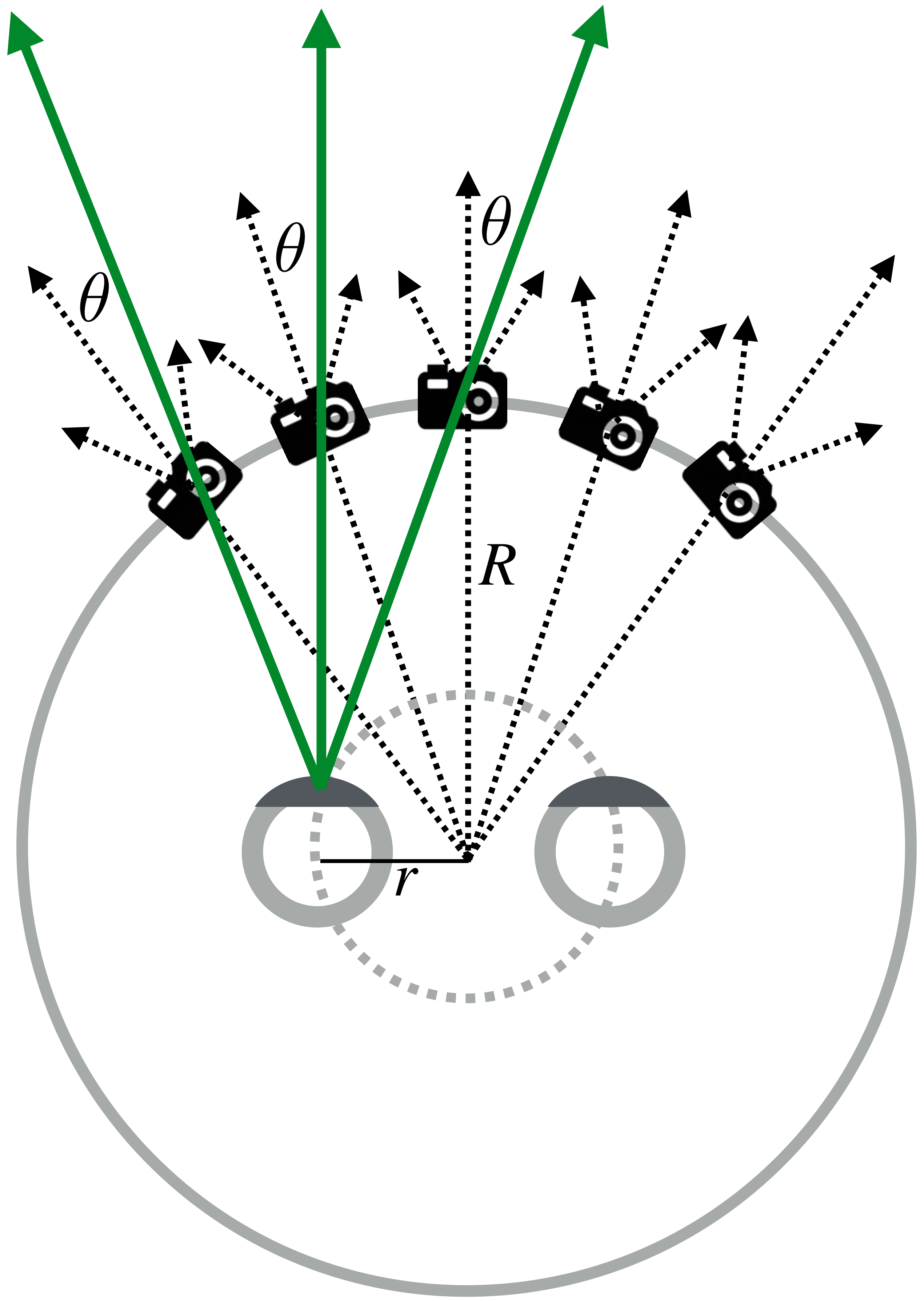
Stereo, 360-degree viewing



Measuring light arriving at left eye

Left eye

$$\sin\theta = r/R$$

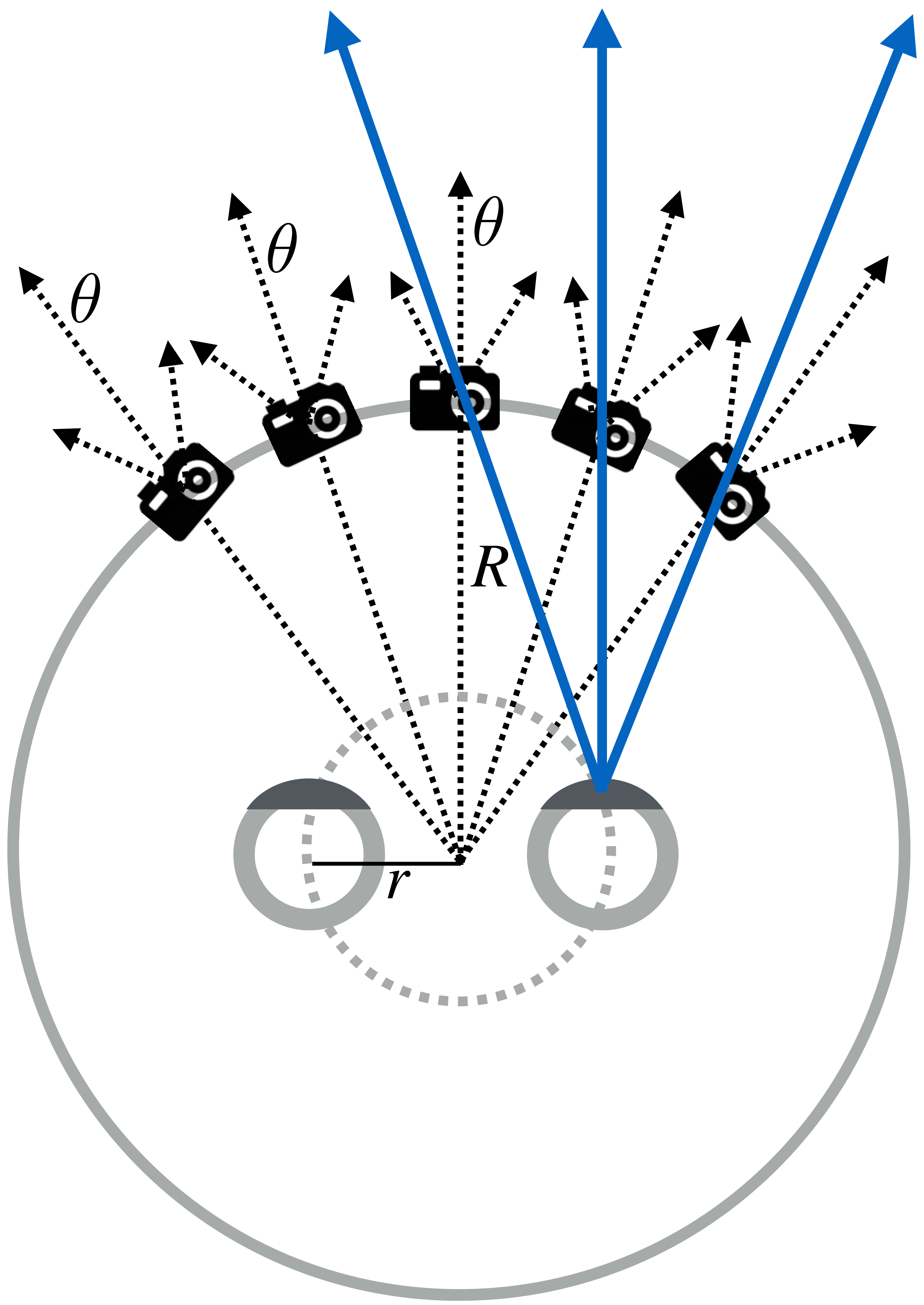


[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Measuring light arriving at right eye

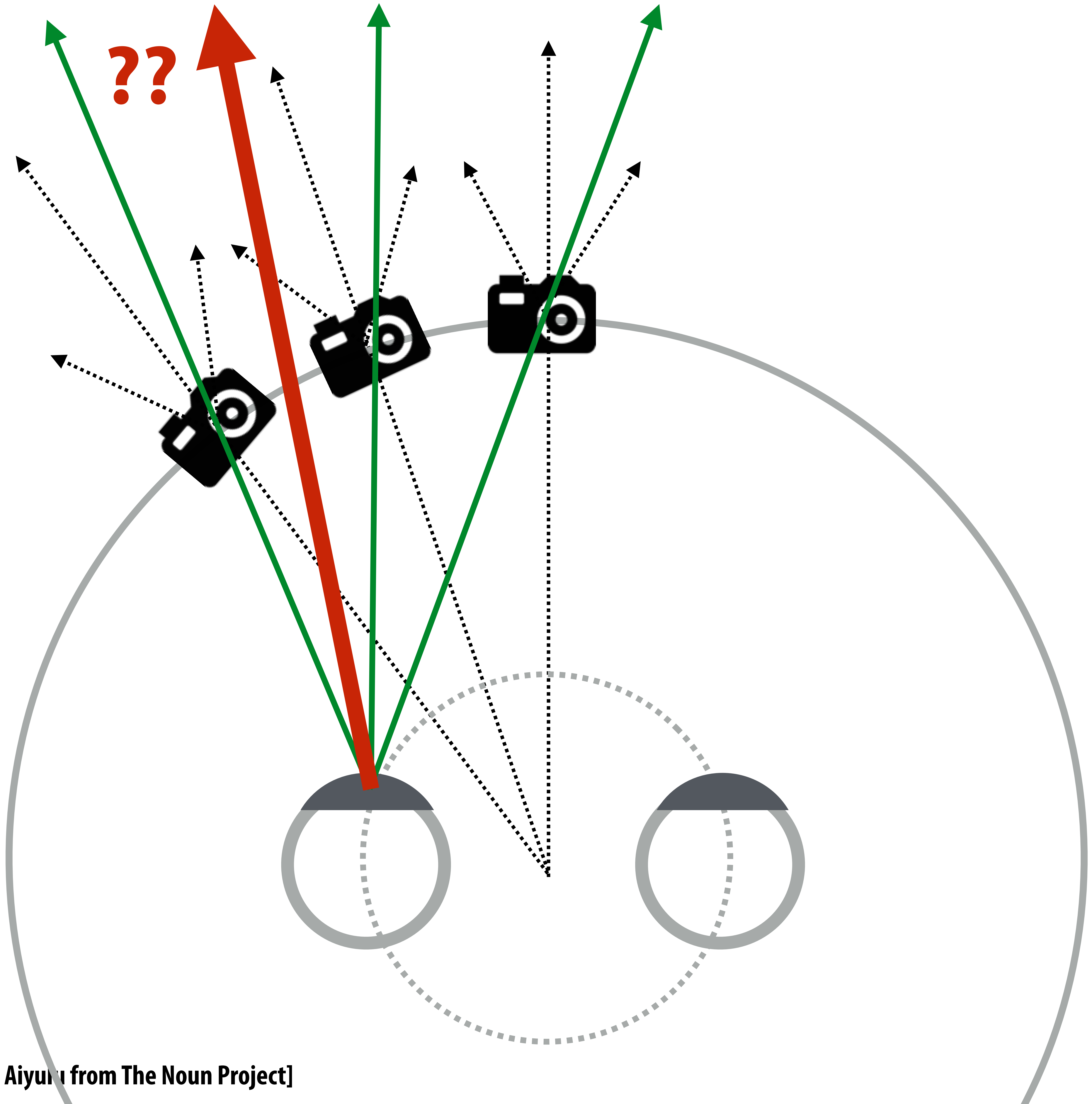
Right eye

$$\sin\theta = -r/R$$



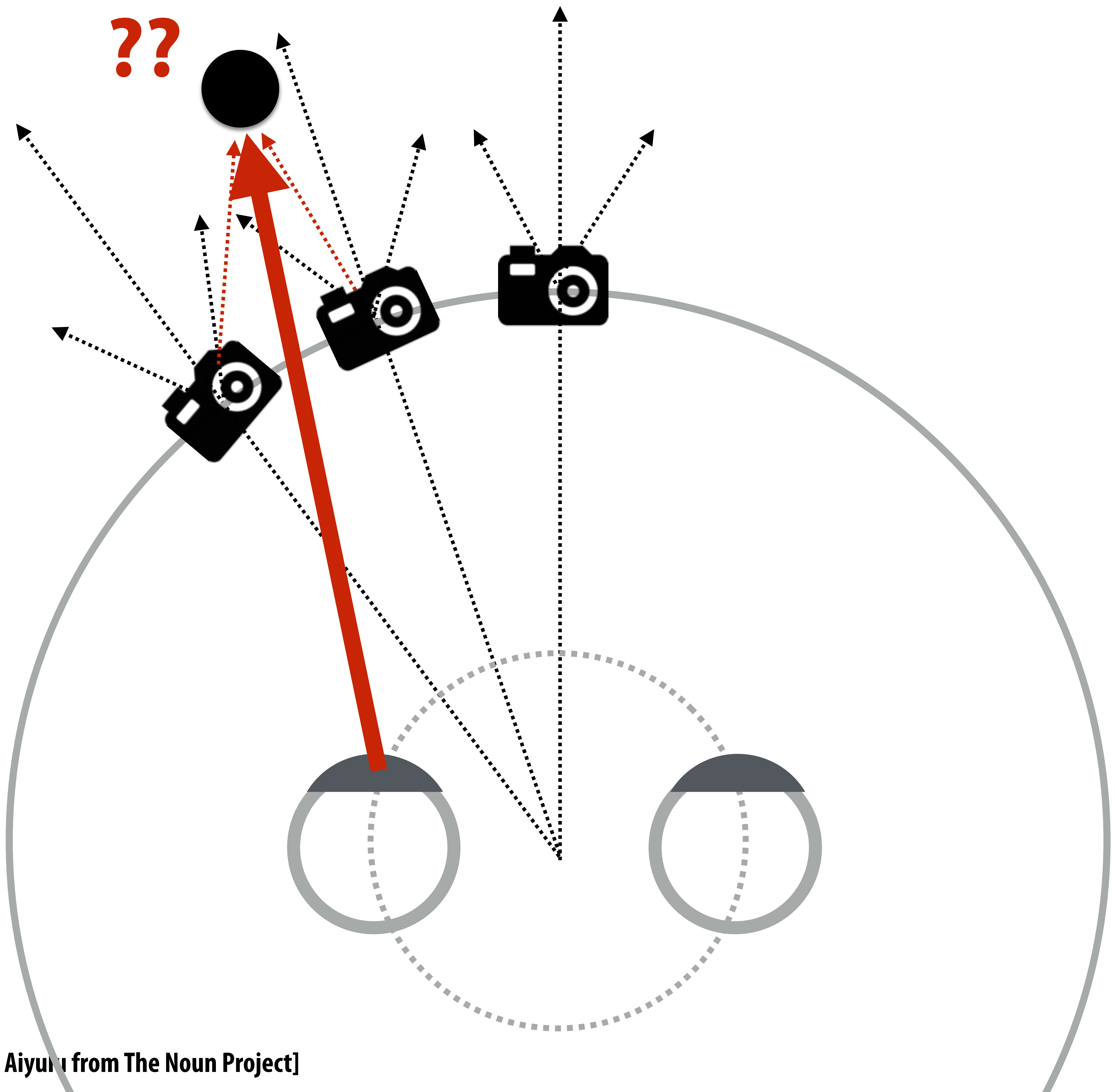
[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

How to estimate rays at "missing" views?



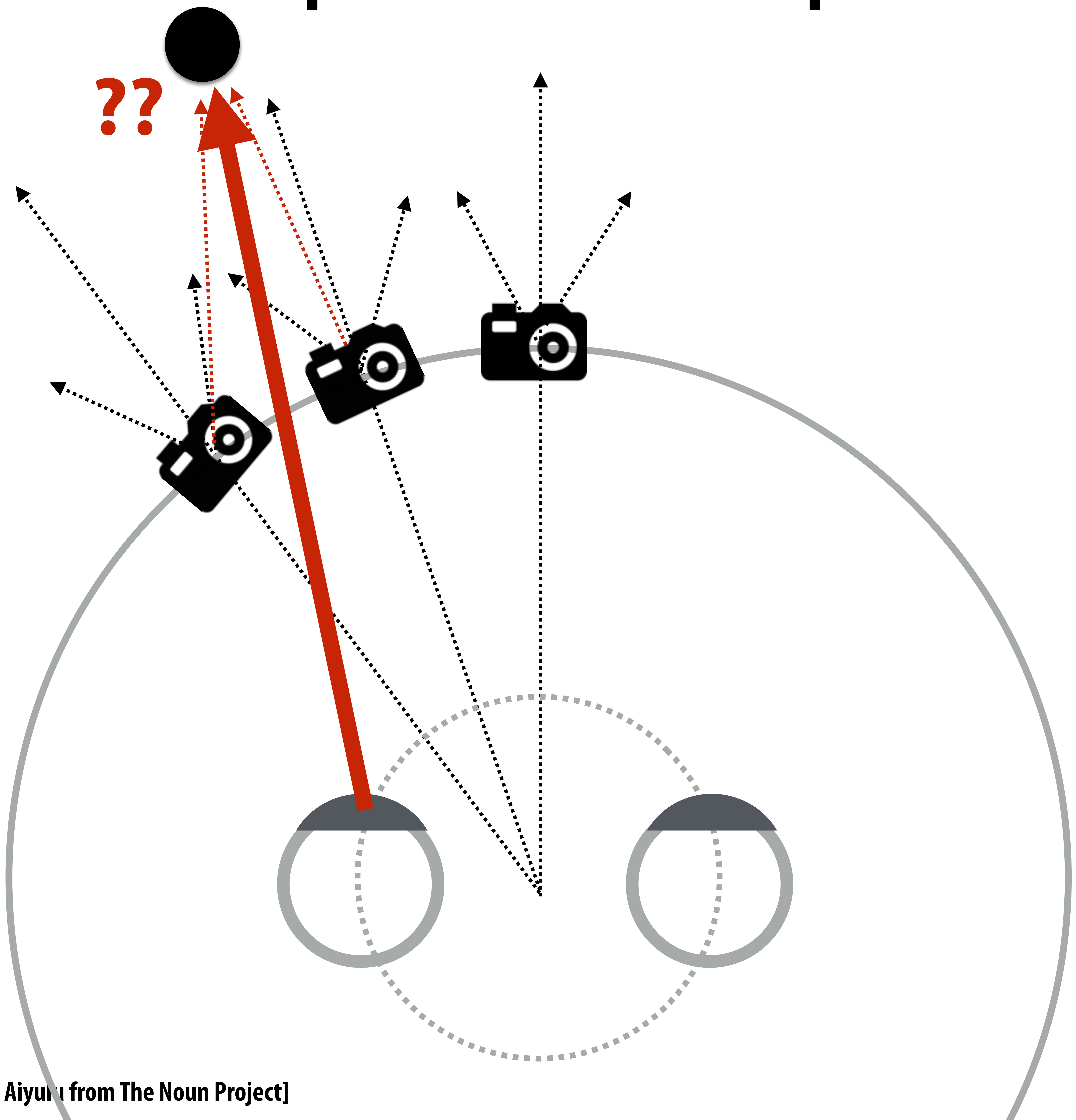
[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Interpolation to novel views depends on scene depth



[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Interpolation to novel views depends on scene depth



[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Computing depth of scene point from two images

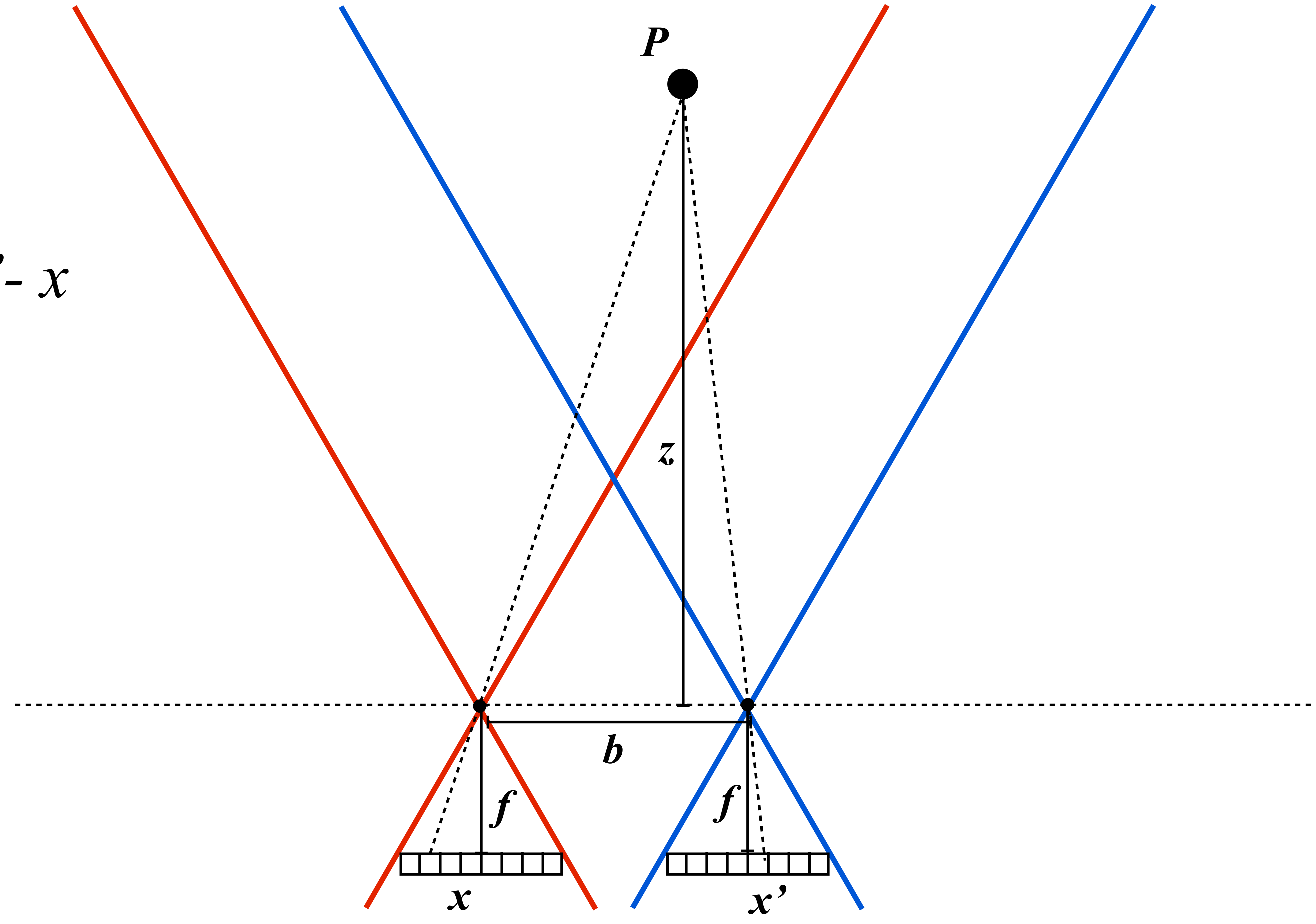
Binocular stereo 3D reconstruction of point *P*: depth from disparity

Focal length: f

Baseline: b

Disparity: $d = x' - x$

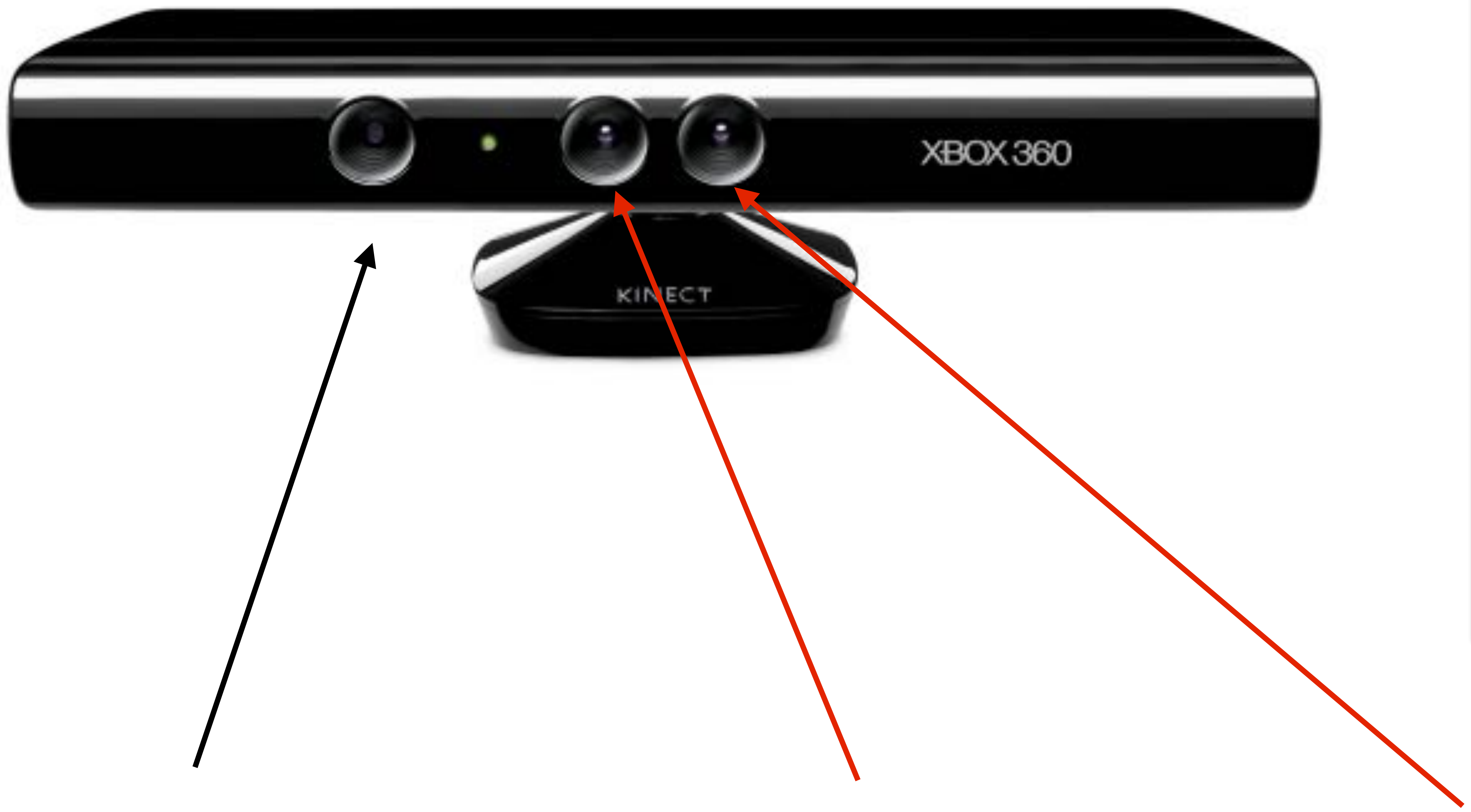
$$z = \frac{bf}{d}$$



Simple reconstruction example: cameras aligned (coplanar sensors), separated by known distance, same focal length

"Disparity" is the distance between object's projected position in the two images: x - x'

Microsoft XBox 360 Kinect



Illuminant
(Infrared Laser + diffuser)

RGB CMOS Sensor
640x480 (w/ Bayer mosaic)

Monochrome Infrared
CMOS Sensor
(Aptina MT9M001)
1280x1024 **

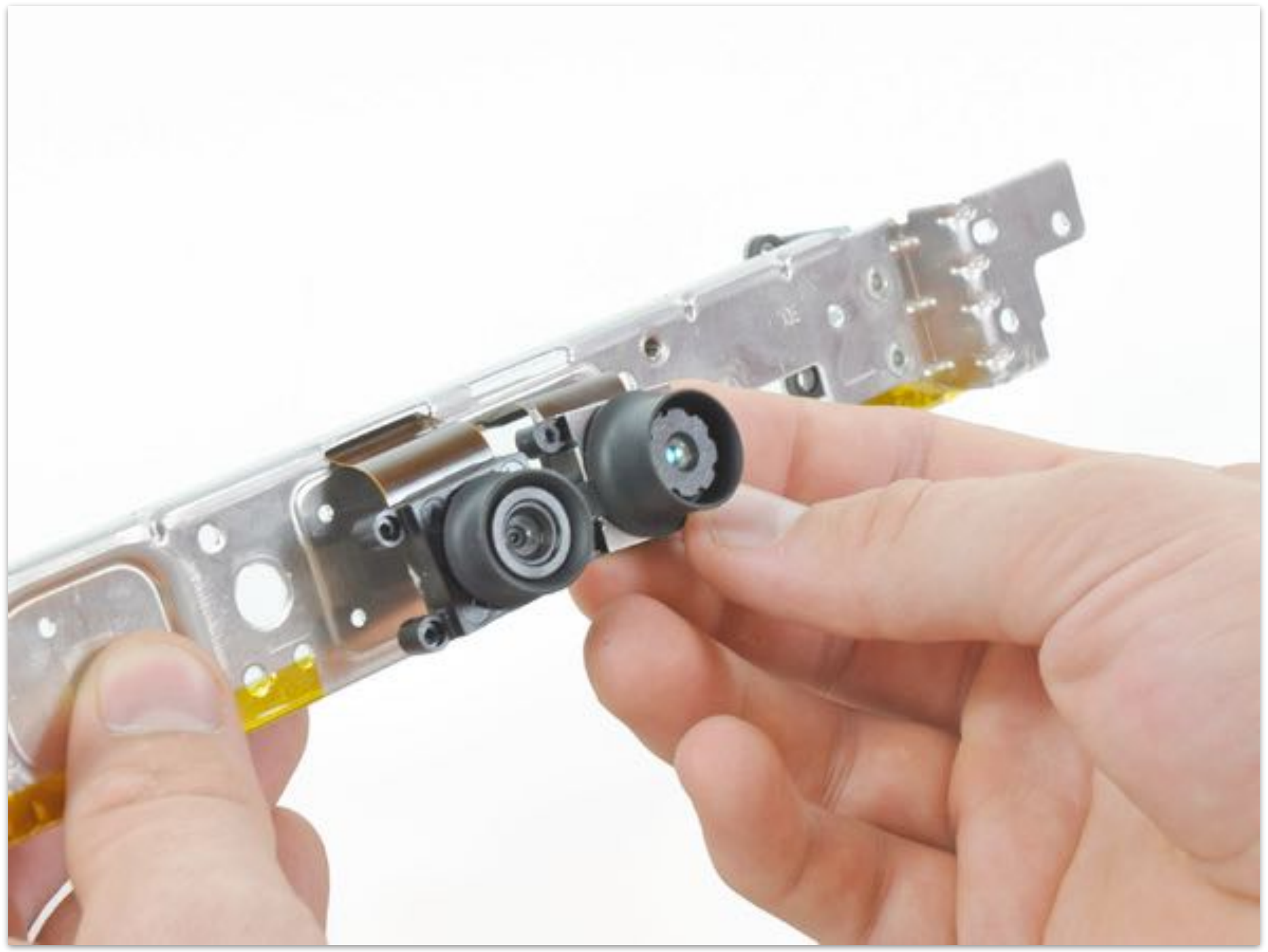
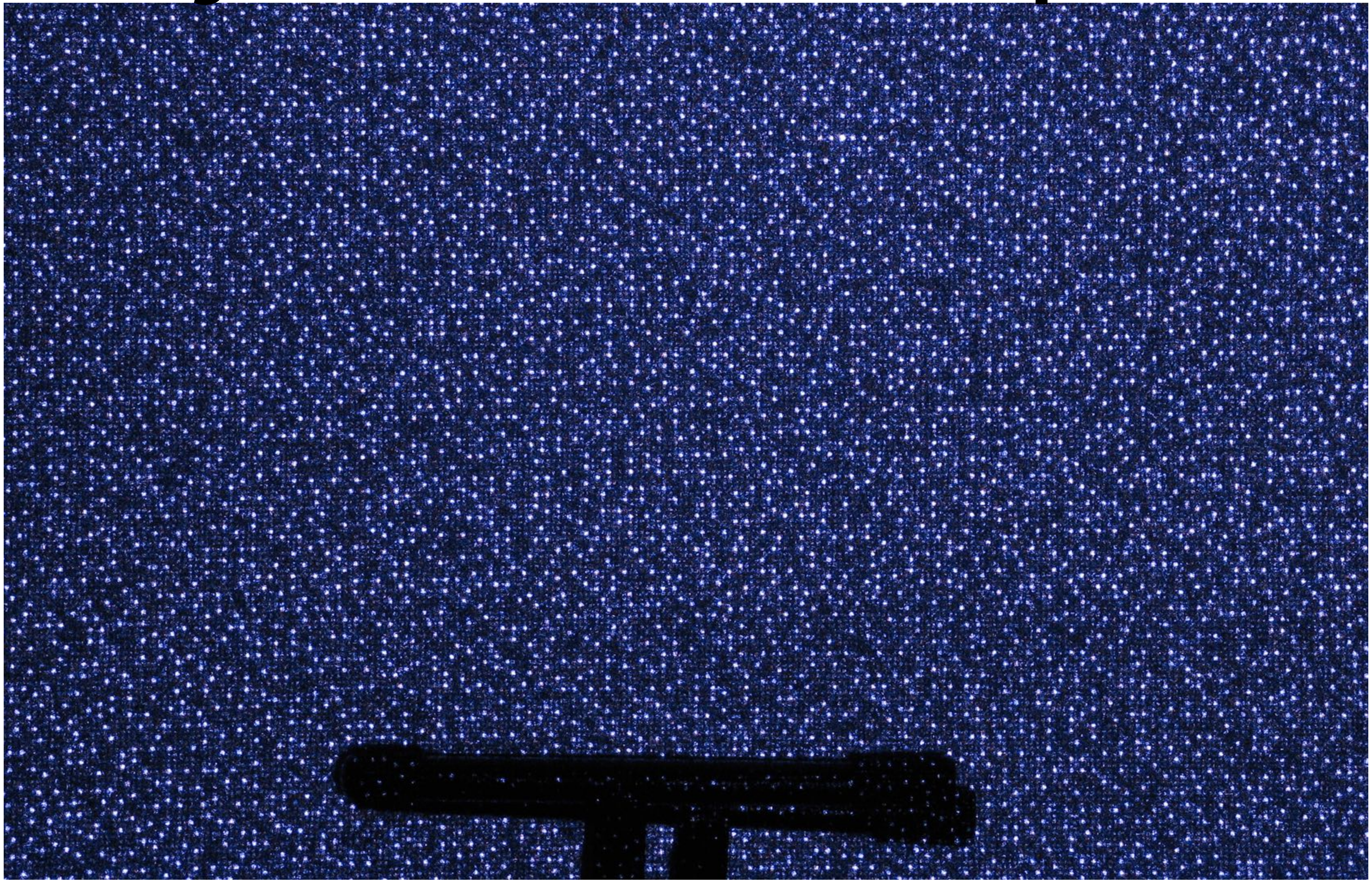


Image credit: iFixIt

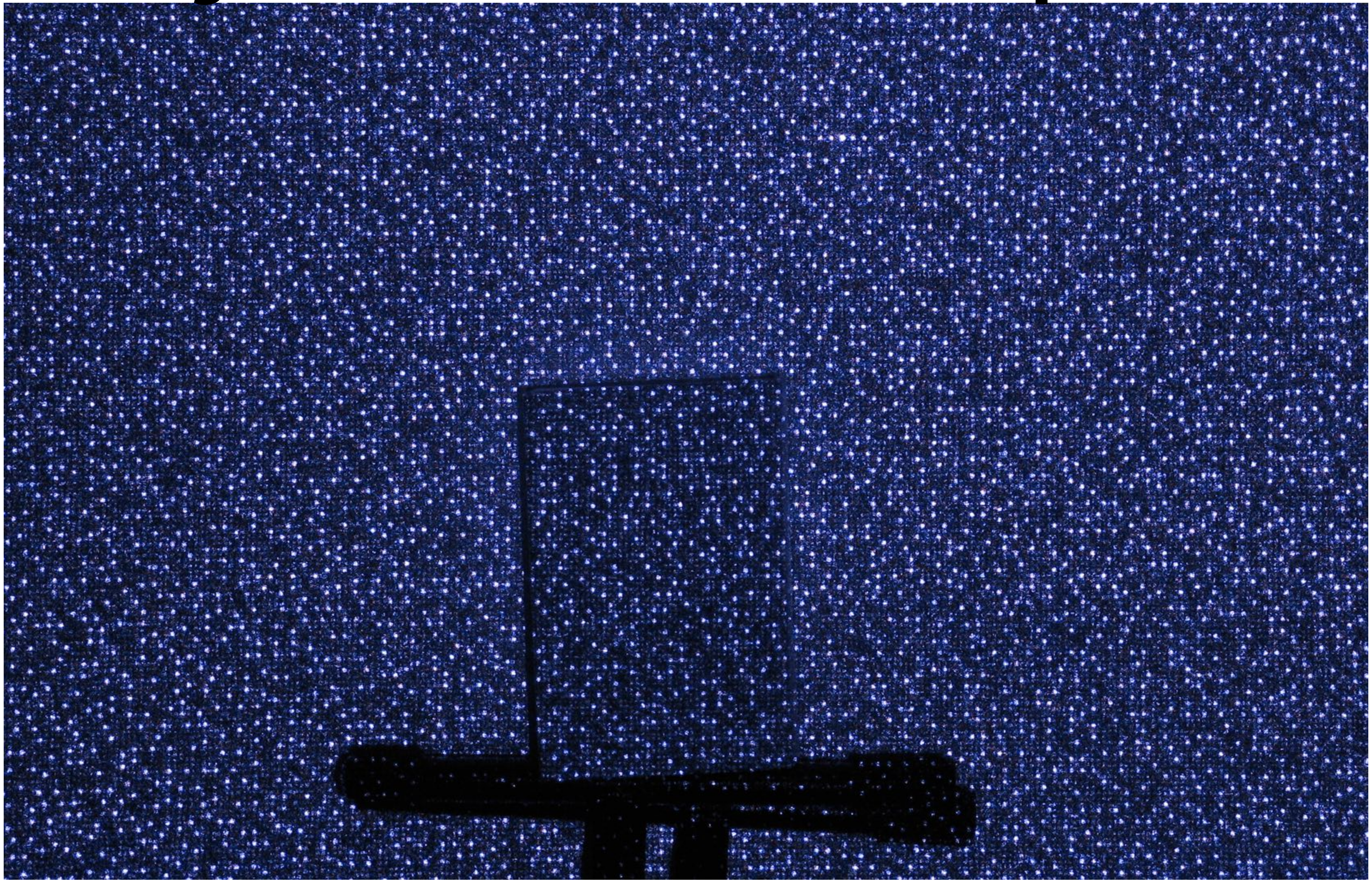
** Kinect returns 640x480 disparity image, suspect sensor is configured for 2x2 pixel binning down to 640x512, then crop

Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

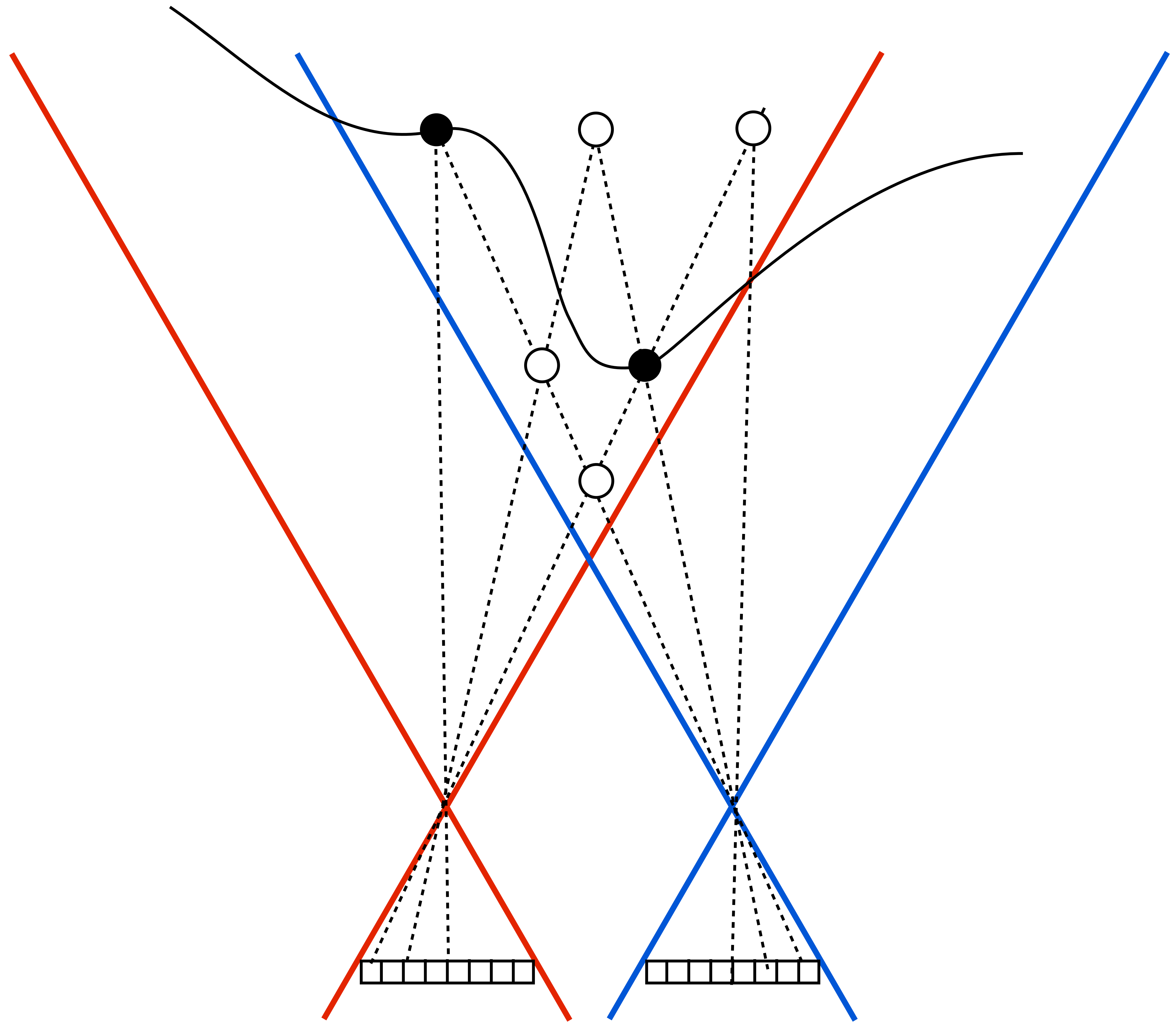
Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

Correspondence problem

How to determine which pairs of pixels in image 1 and image 2 correspond to the same scene point?

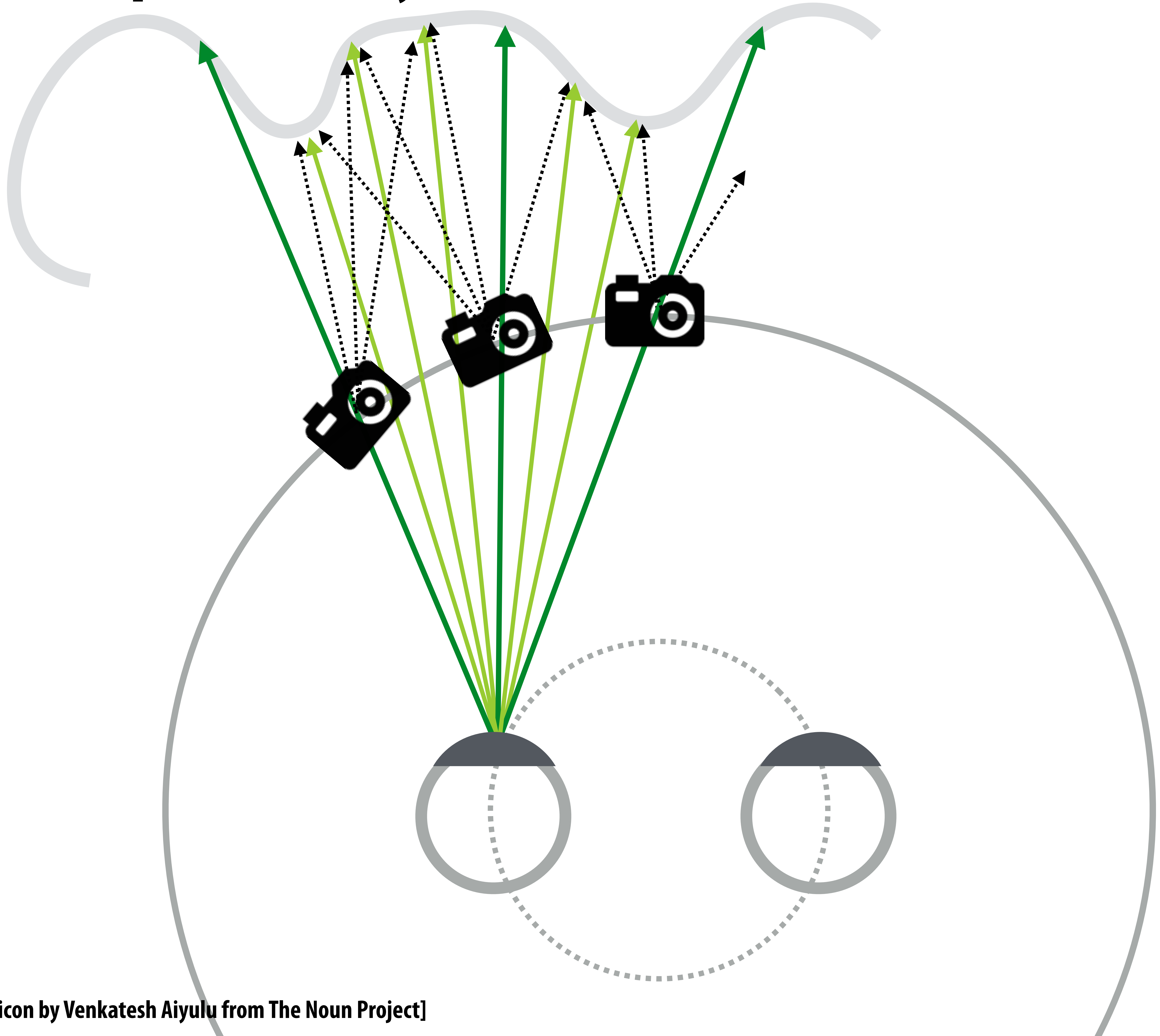


Correspondence problem = compute “flow” between adjacent cameras

- For each pixel in frame from camera *i*, find closest pixel in camera *i+1*
- Google’s Jump pipeline uses a coarse-to-fine algorithm: align 32x32 blocks by searching over local window, then perform per-pixel alignment
  - Recall: H.264 motion estimation, HDR+ burst alignment (same correspondence challenge, but here we are aligning different perspectives at the same time to estimate unknown scene depth, not estimating motion of camera or scene over time)
  - Additional tricks to ensure temporal consistency of flow over time (see papers)

Image credit: Andersen et al. 2016

Left eye: with interpolated rays



[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Omnidirectional stereo (ODS) representation

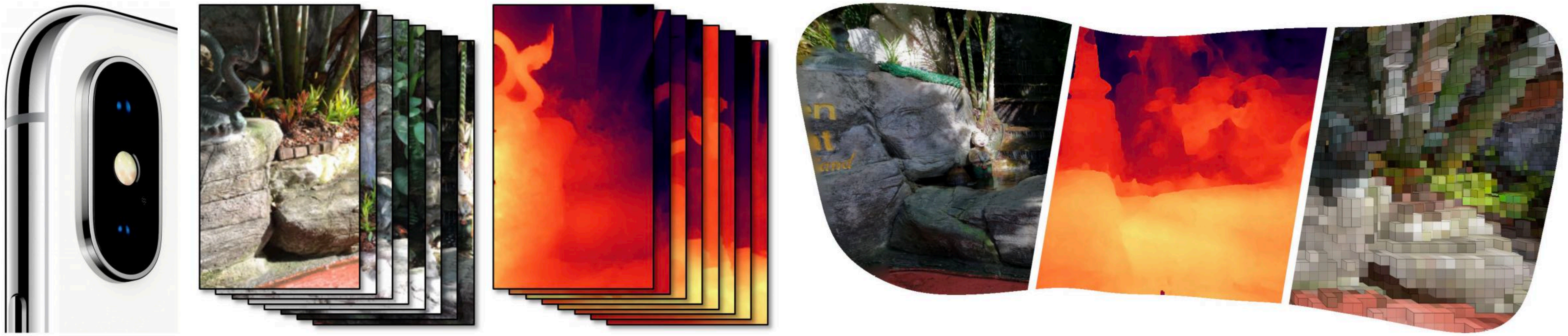
- Unique panorama of size W x H for left and right eye
- Good: can be saved, compressed, edited as normal video
- Column *j* of pixels corresponds to column from interpolated camera at ring position at angle: $\frac{2\pi j}{W}$

Overlay of Left and Right eye ODS panoramas

Image credit: Andersen et al. 2016

“Casual 3D photography”

- Acquisition: wave a smartphone camera around to acquire images of scene from multiple viewpoints
- Processing: construct 3D representation of scene from photos
  - Render a textured triangle mesh



Dual-camera Smartphone

Burst of photos + depth maps

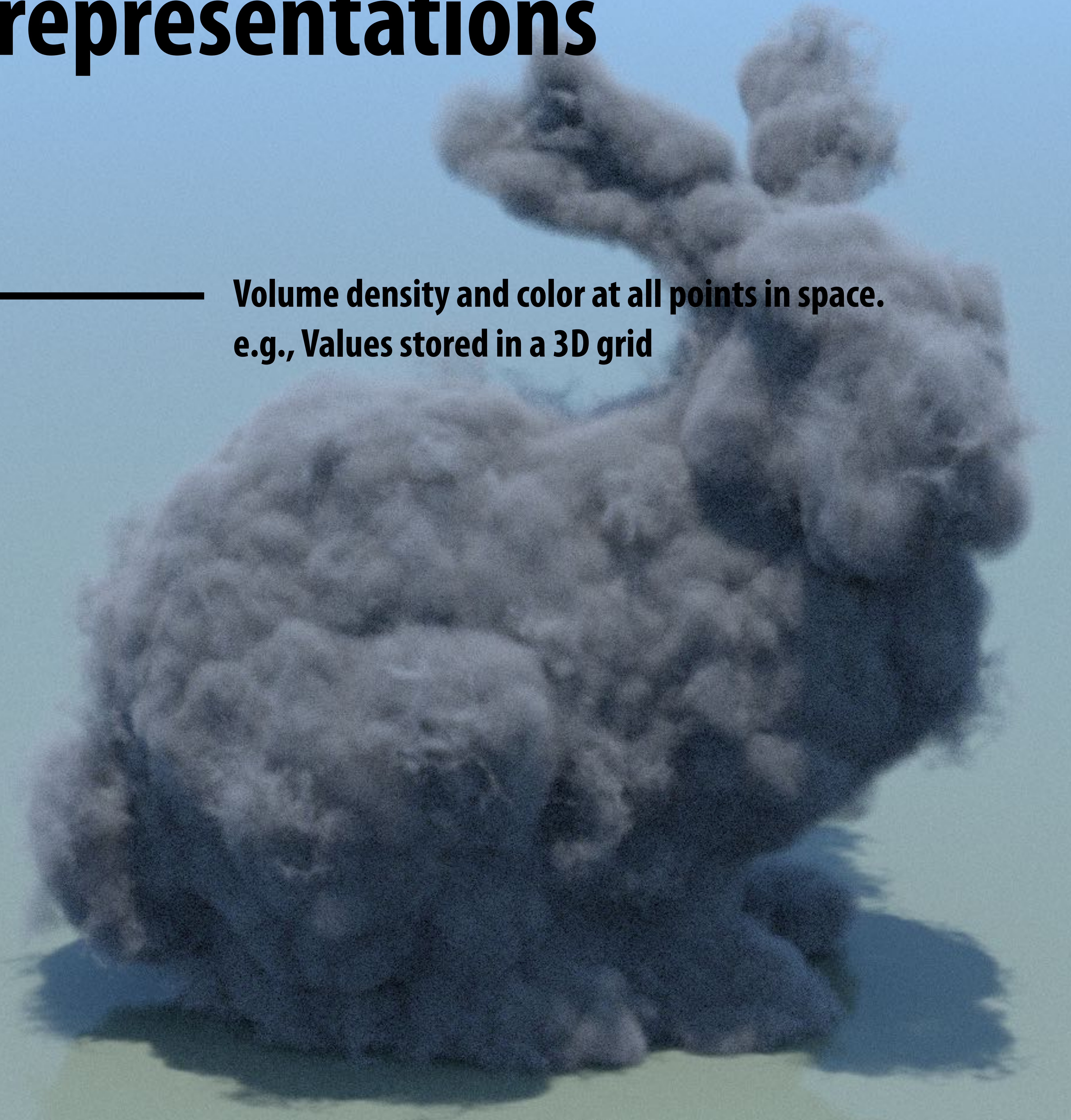
Stitch photos into depth panorama, create 3D mesh + textures, render during VR viewing

But it's hard to estimate depth and geometry

Volumetric representations

$$\sigma(\mathrm{p})$$
$$\mathrm{c}(\mathrm{p})$$

Volume density and color at all points in space.
e.g., Values stored in a 3D grid



Representing rays

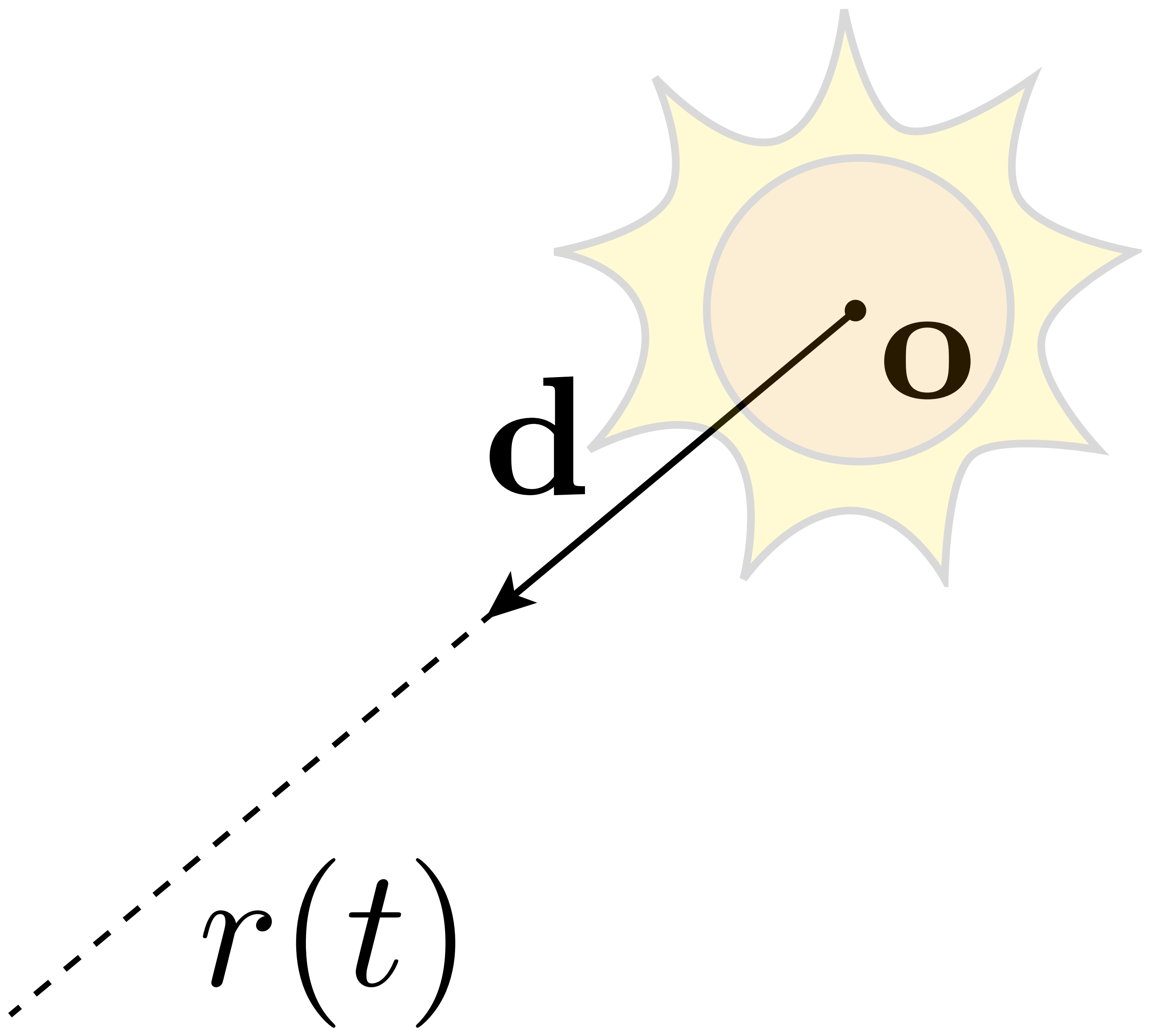
origin

unit direction

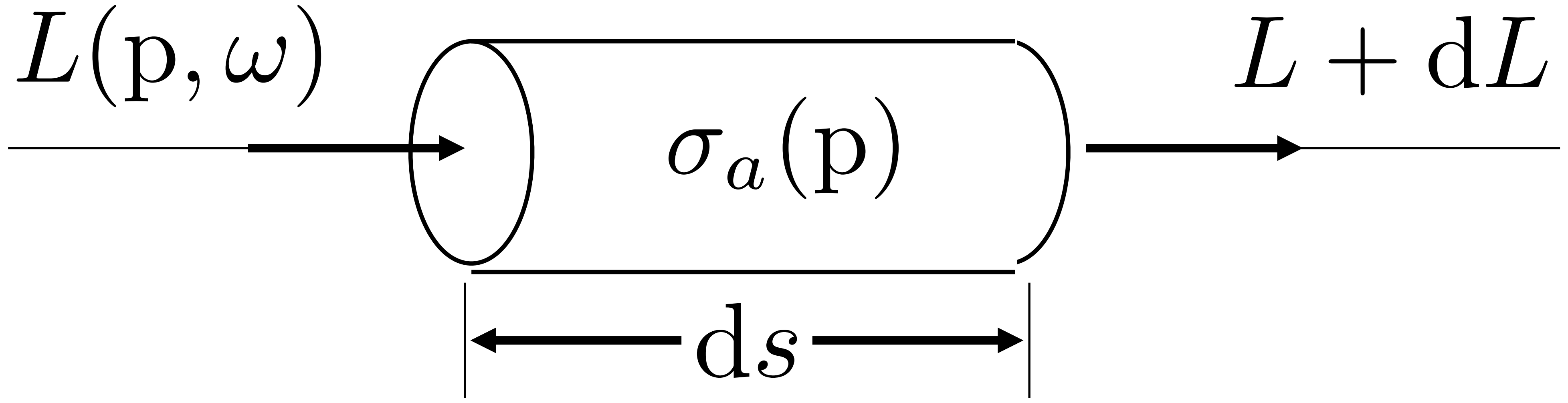
$$\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$$

point along ray

"time"



Absorption in a volume



$$\mathrm{p} = (x, y, z)$$

$$\omega = (\phi, \theta)$$

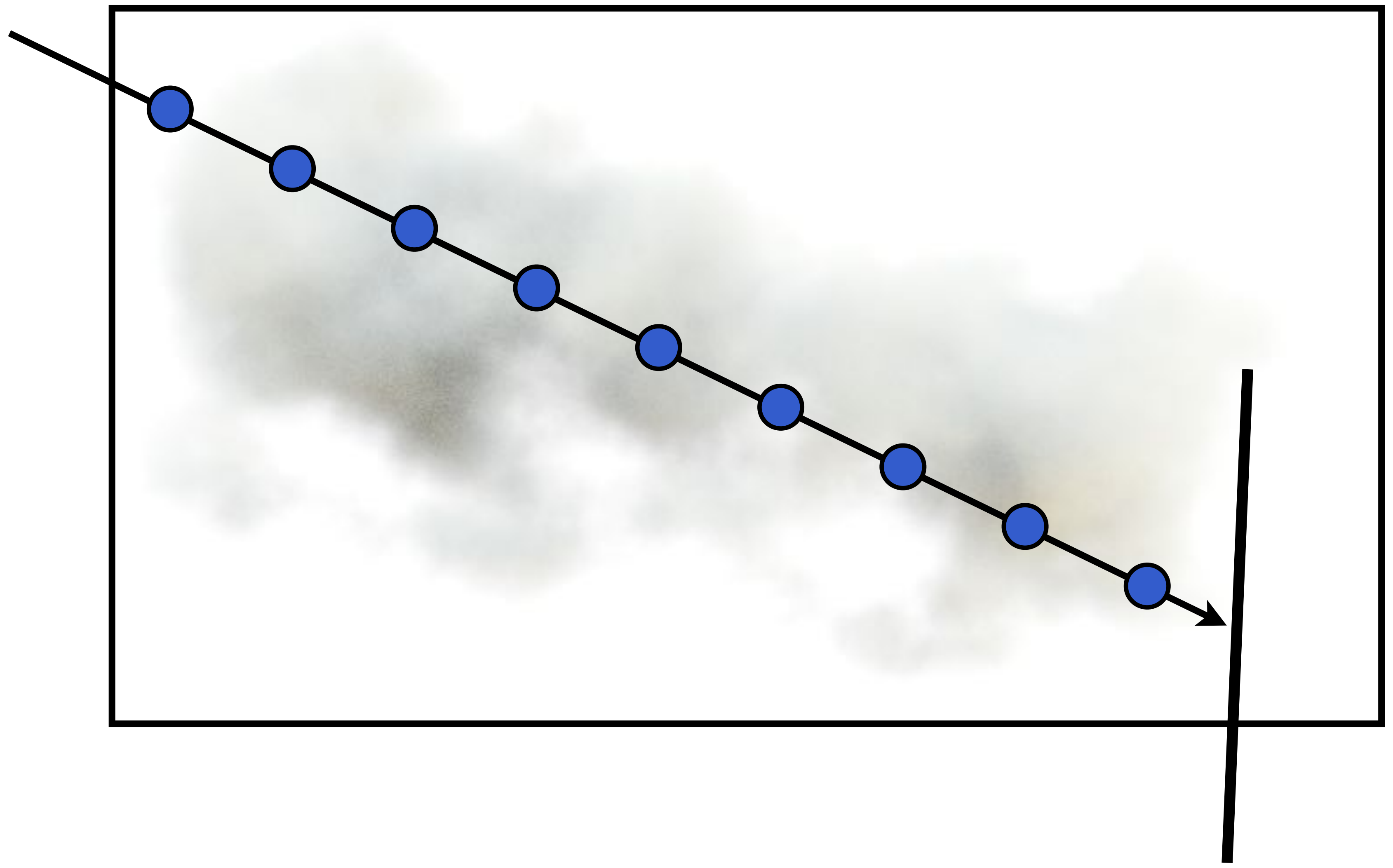
$$\mathrm{d}L(\mathrm{p}, \omega) = -\sigma_a(\mathrm{p})\, L(\mathrm{p}, \omega)\, \mathrm{d}s$$

- $L(\mathrm{p}, \omega)$ **light energy (radiance) along a ray from p in direction w**
- **Absorption cross section at point in space:** $\sigma_a(\mathrm{p})$
  - **Probability of being absorbed per unit length**
  - **Units: 1/distance**

Rendering volumes

$$\sigma(\mathrm{p})$$
$$\mathrm{c}(\mathrm{p})$$

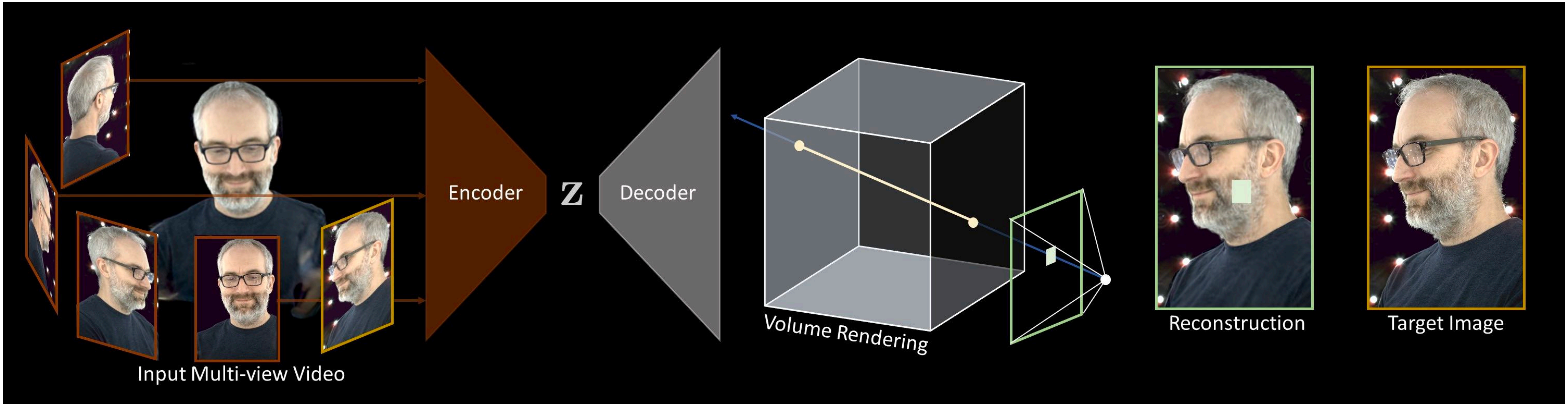
Volume density and color at all points in space.
e.g., Values stored in a 3D grid



$$C(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d})dt\,,\ \text{where } T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s))ds\right)$$

Neural volumes

- Learn to encode multiple views of a person into a latency code (z) that is decoded into a volume than can be rendered with conventional graphics techniques *from any viewpoint*



- Initially motivated by VR applications
- Want to move the view location as well as view direction

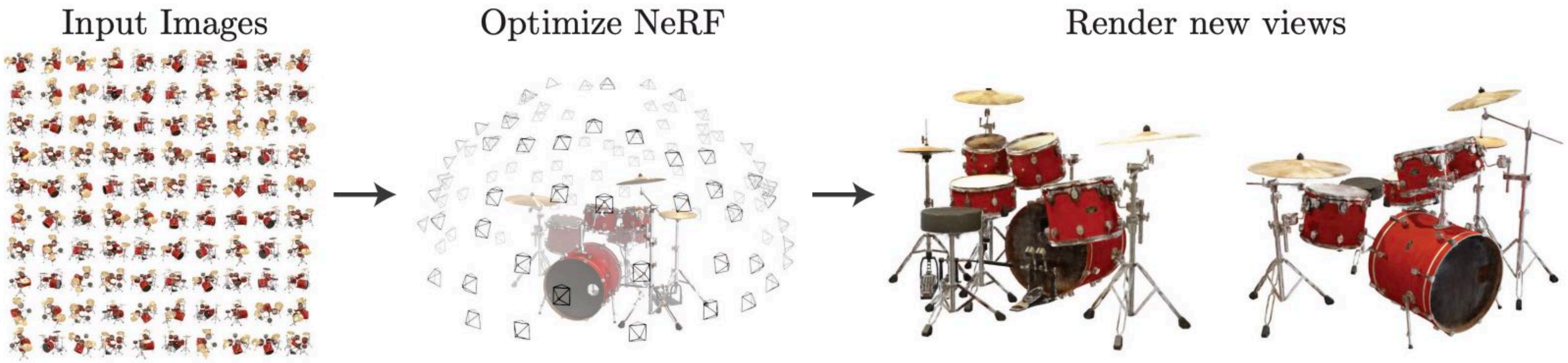
Learning better (more compressed) representations

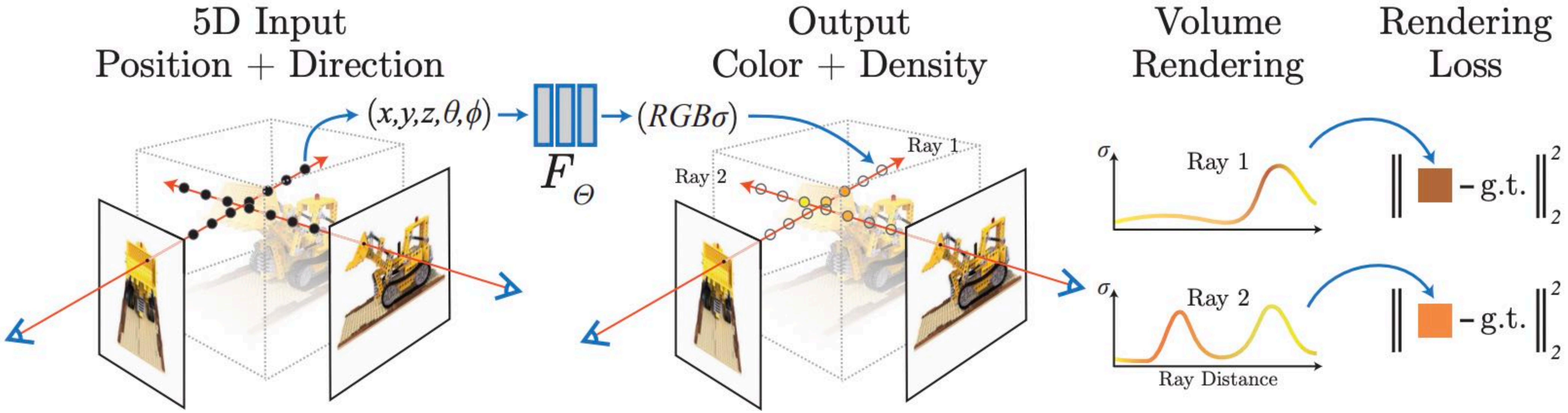
- **Why not just learn an approximation to the continuous function:**

$$(\mathrm{p}, \omega) \rightarrow \boxed{F_\theta(\mathrm{p}, \omega)} \rightarrow \begin{matrix} \sigma(\mathrm{p}) \\ \mathrm{c}(\mathrm{p}) \end{matrix}$$

- **For all photos of the scene that we have, use $F_\theta(\mathrm{p}, \omega)$ to volume render the scene from the known viewpoint.**
- **Loss is difference between rendered view and actual photo.**
- **Update θ using standard optimization techniques (SGD)**

Learning neural radiance fields (NeRF)





Key ideas of volumetric representations in this context

- Do not need to reconstruct/estimate triangle mesh surface geometry
- Volume rendering is easily differentiable, so easy to update $F_\theta(\mathrm{p}, \omega)$
- The DNN used to represent $F_\theta(\mathrm{p}, \omega)$ is a compact representation of this high-dimensional function.
  - Better representation than a dense voxel grid.

Demos