Lecture 7:

The Light Field, Computational Cameras, and VR

Visual Computing Systems
Stanford CS348V, Winter 2018

Credit: light-field camera slides courtesy of Ren Ng

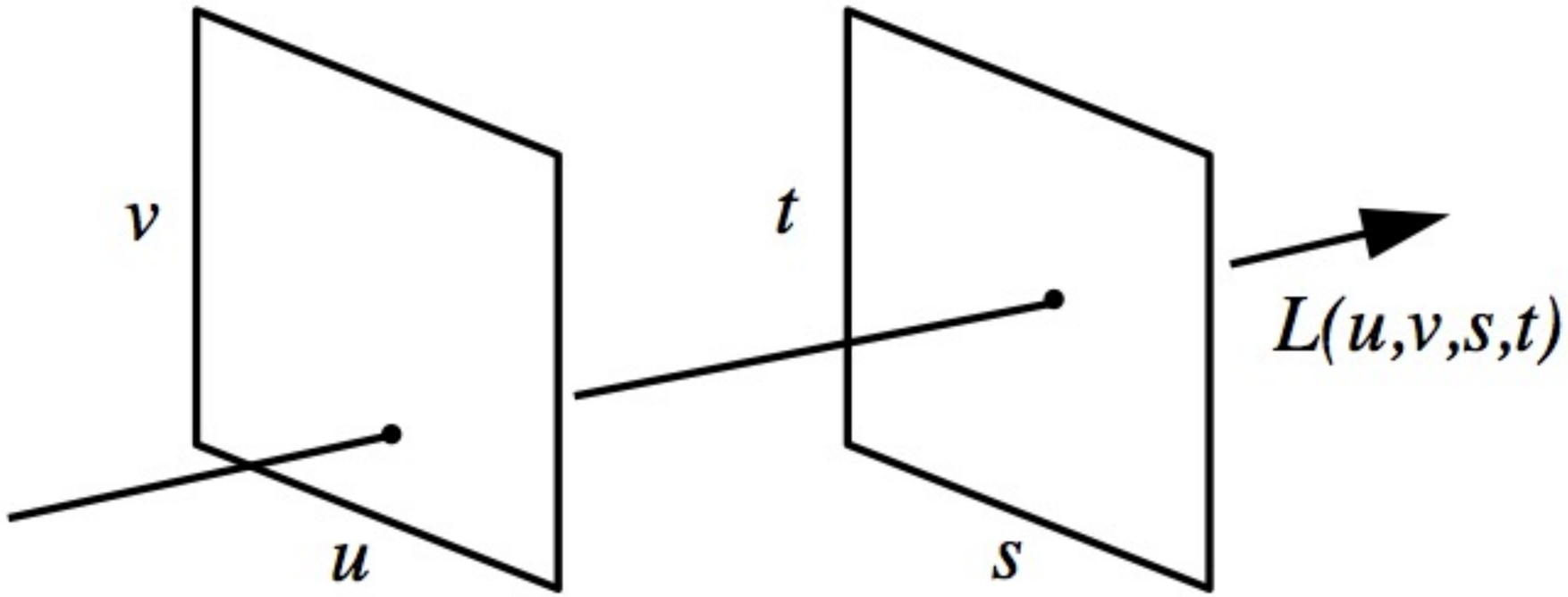
Let’s think about all the “rays of light” in this room

Light-field parameterization

[Levoy and Hanrahan 96]

[Gortler et al., 96]

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



[Image credit: Levoy and Hanrahan 96]

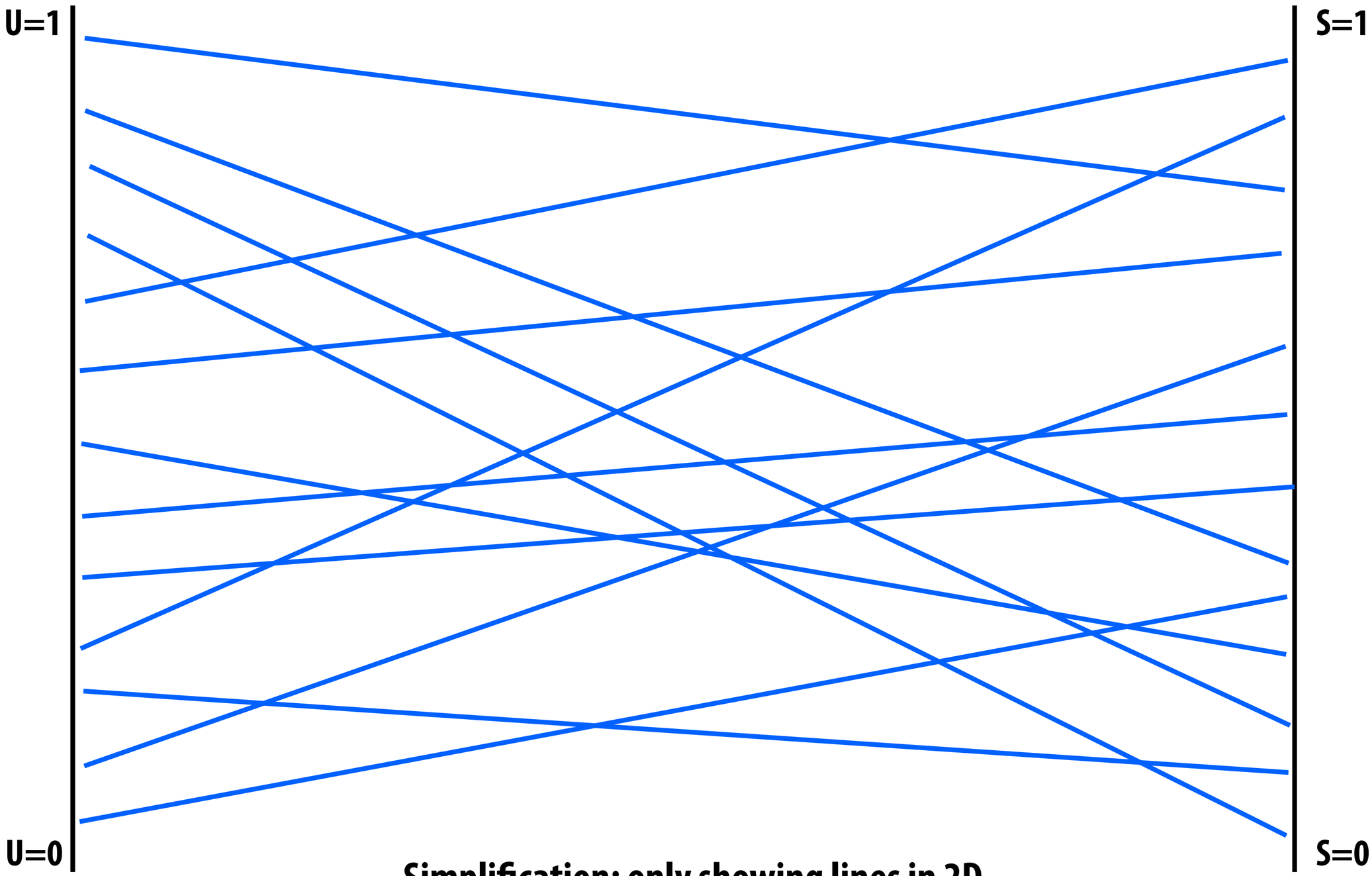
Efficient two-plane parameterization

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

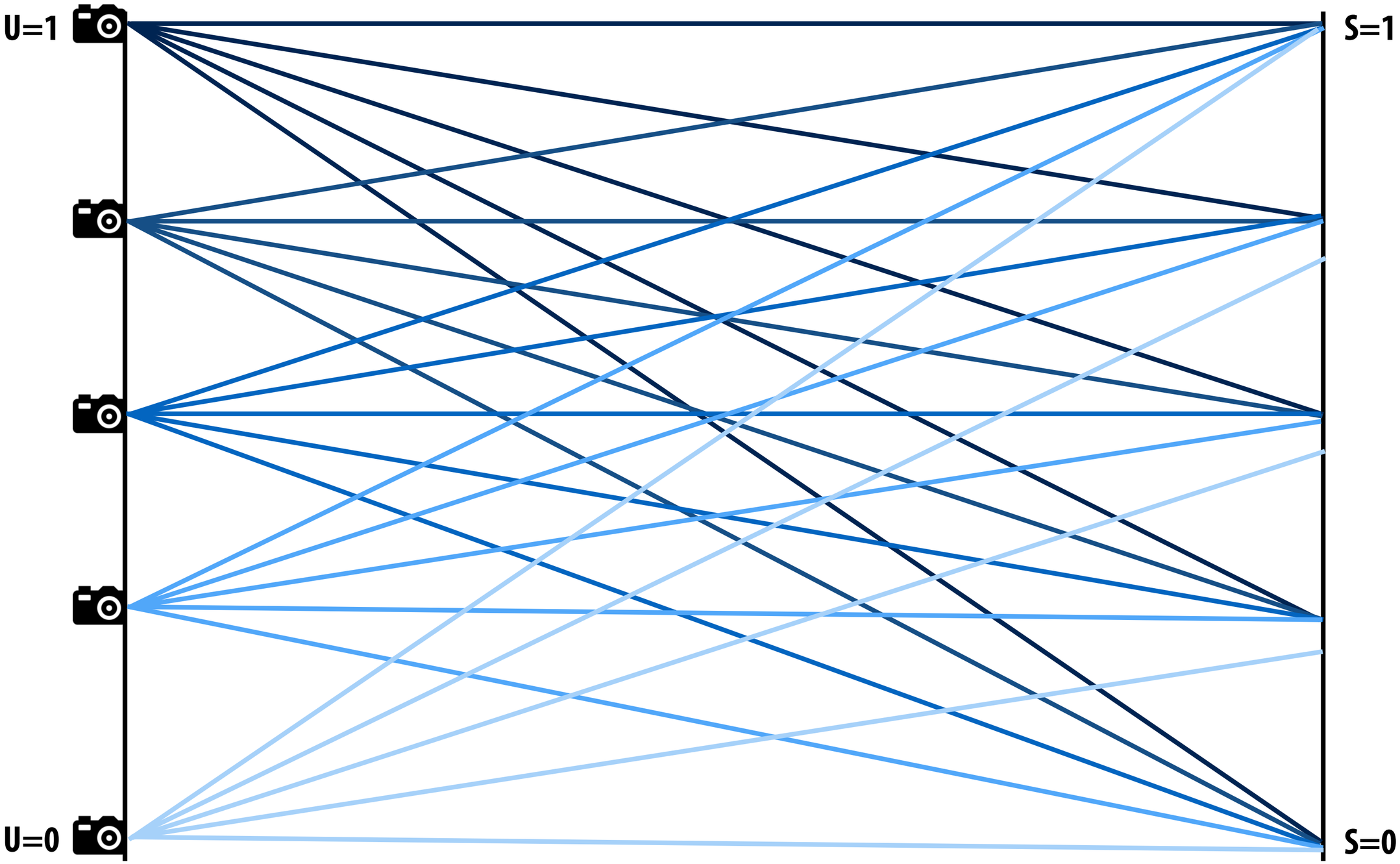
Levoy/Hanrahan refer to representation as a "light slab": beam of light entering one quadrilateral and exiting another

Sampling the light field



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

Sampling the light field by taking pictures



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

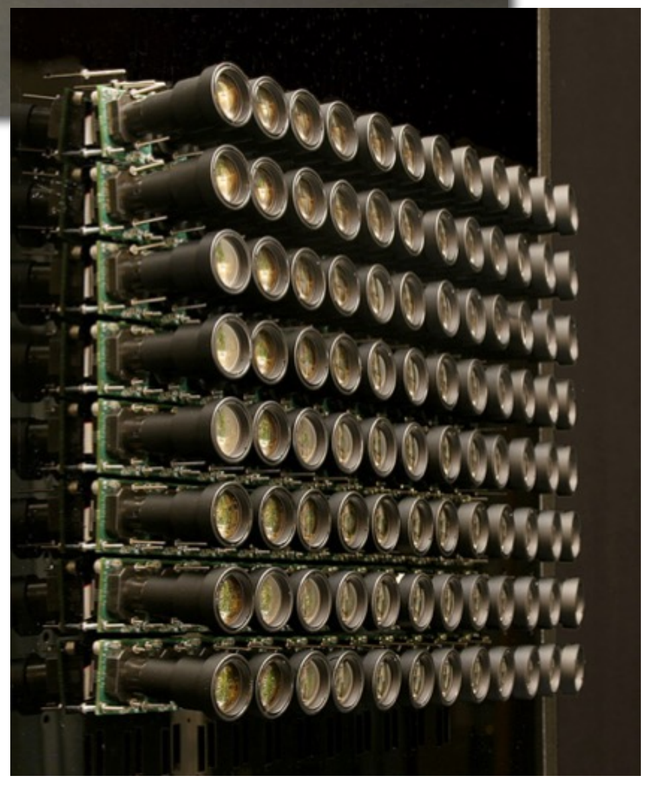
Stanford Camera Array

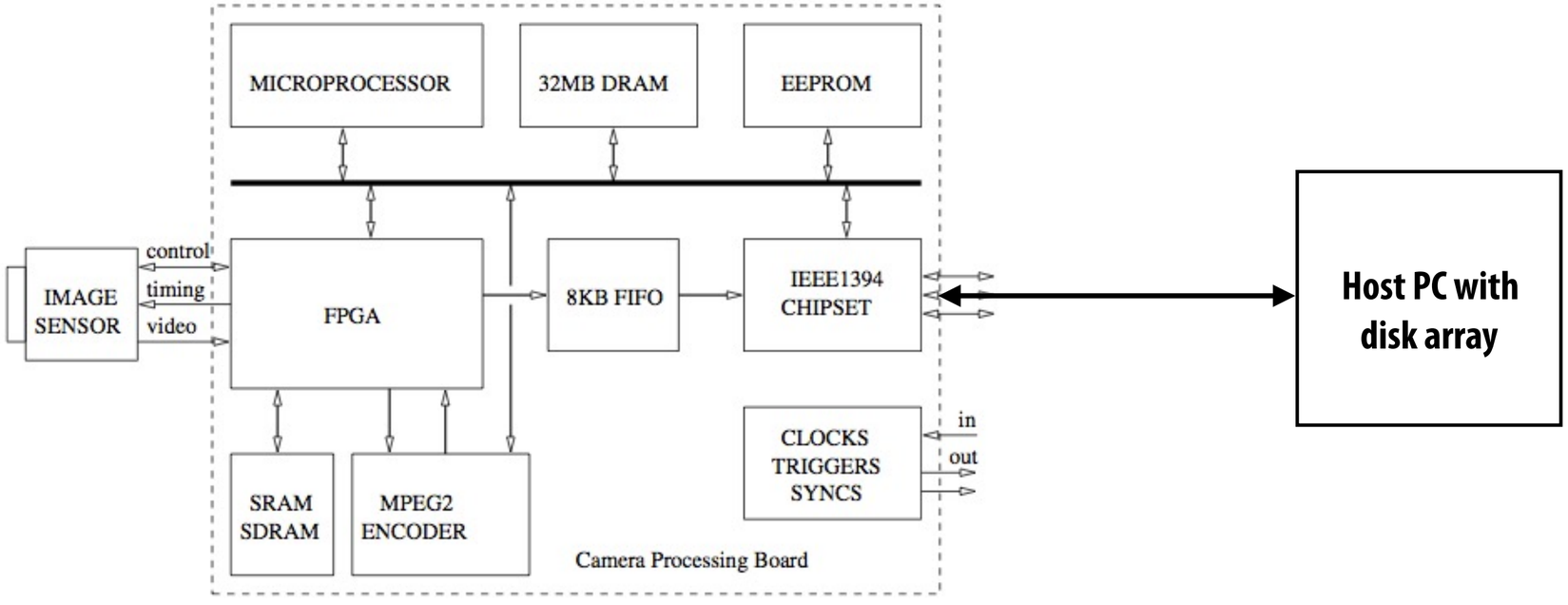
[Wilburn et al. 2005]

640 x 480 tightly synchronized, repositionable cameras

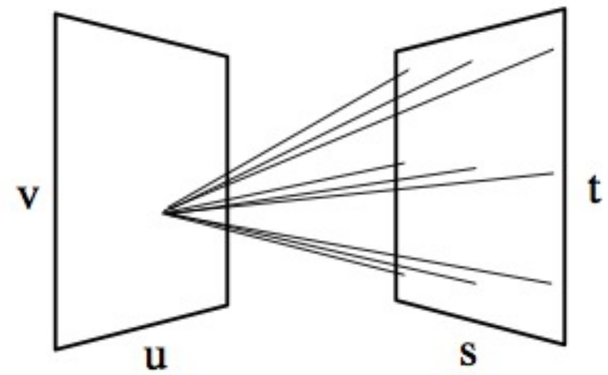
Custom processing board per camera

Tethered to PCs for additional processing/storage





Light field storage layouts



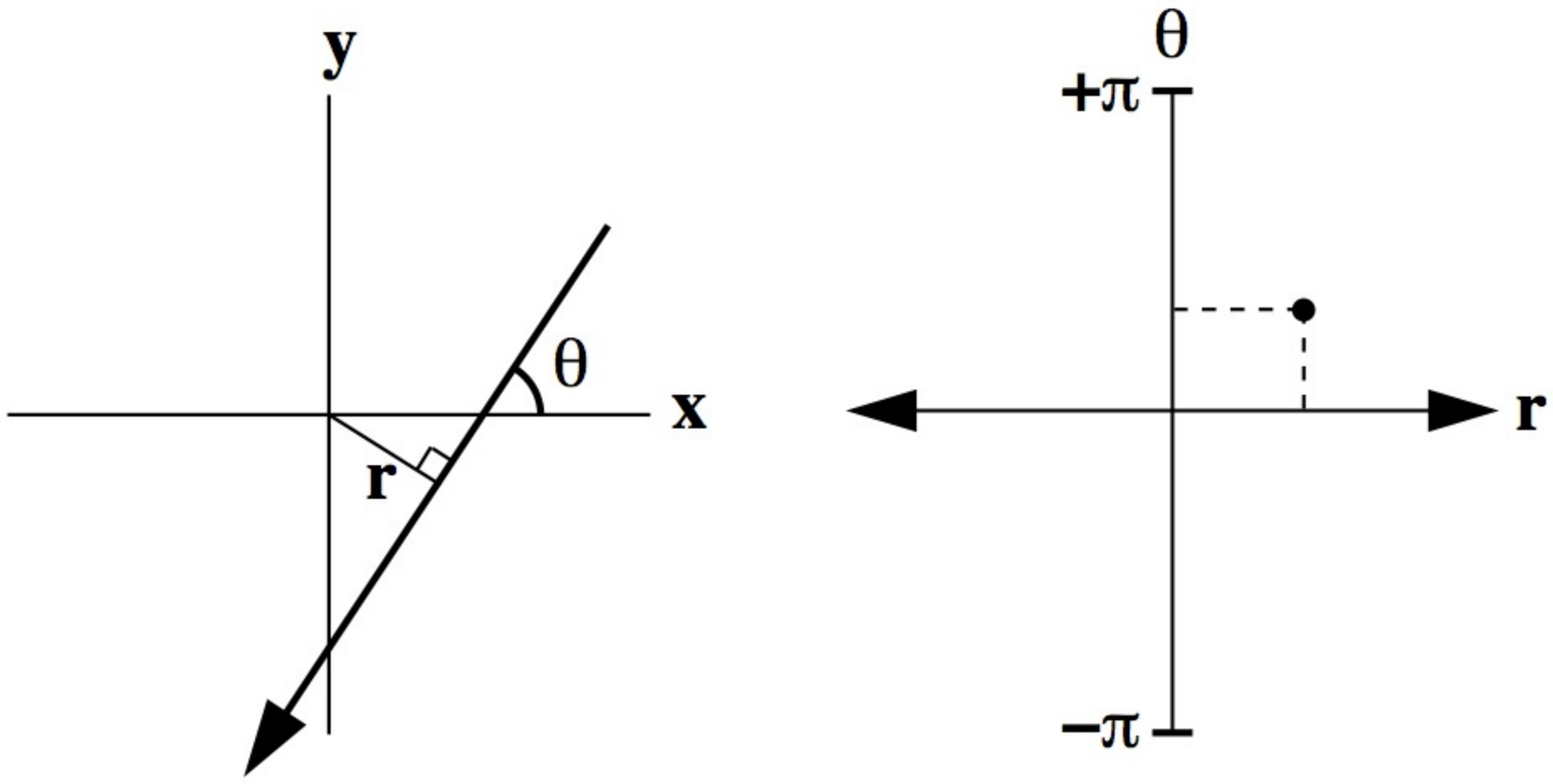
(a)

(b)

[Image credit: Levoy and Hanrahan 96]

Line-space representation

Each line in Cartesian space* is represented by a point in line space



Cartesian space **Line space**

*** Shown here in 2D, generalizes to 3D Cartesian lines**

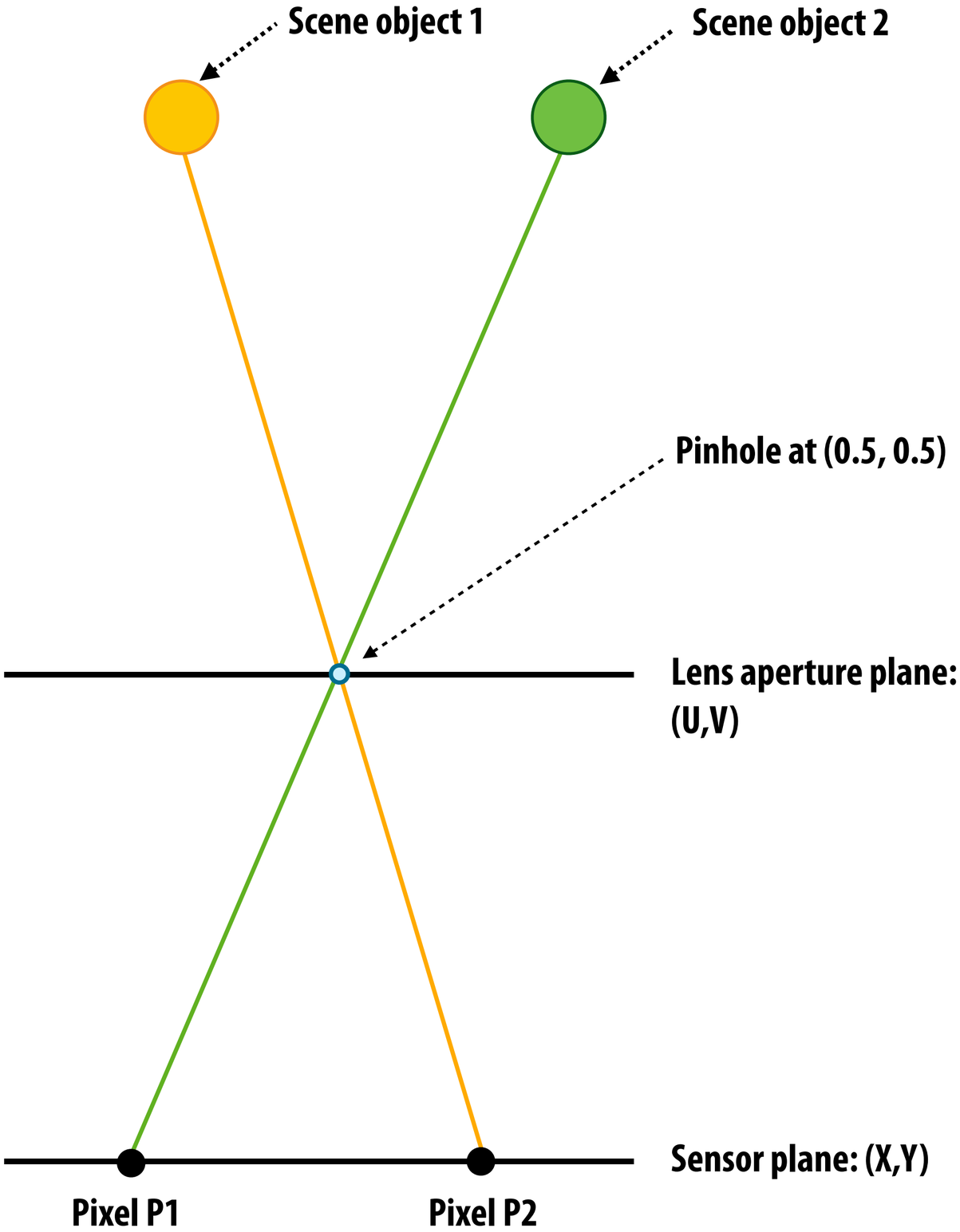
[Image credit: Levoy and Hanrahan 96]

Pinhole camera

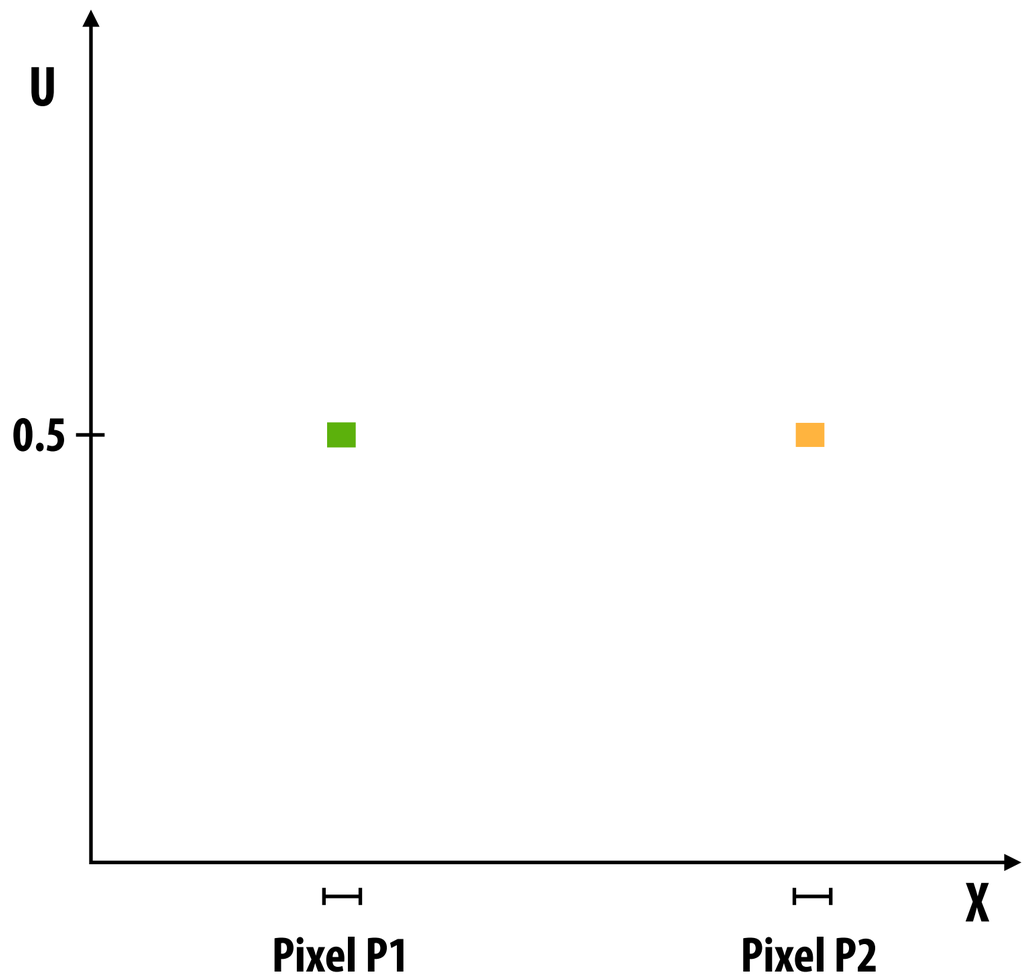
https://civilwar150pinholeproject.com/2013/04/13/pinhole-shutter/

http://brianvds.blogspot.com/2012/08/a-simple-pinhole-camera.html

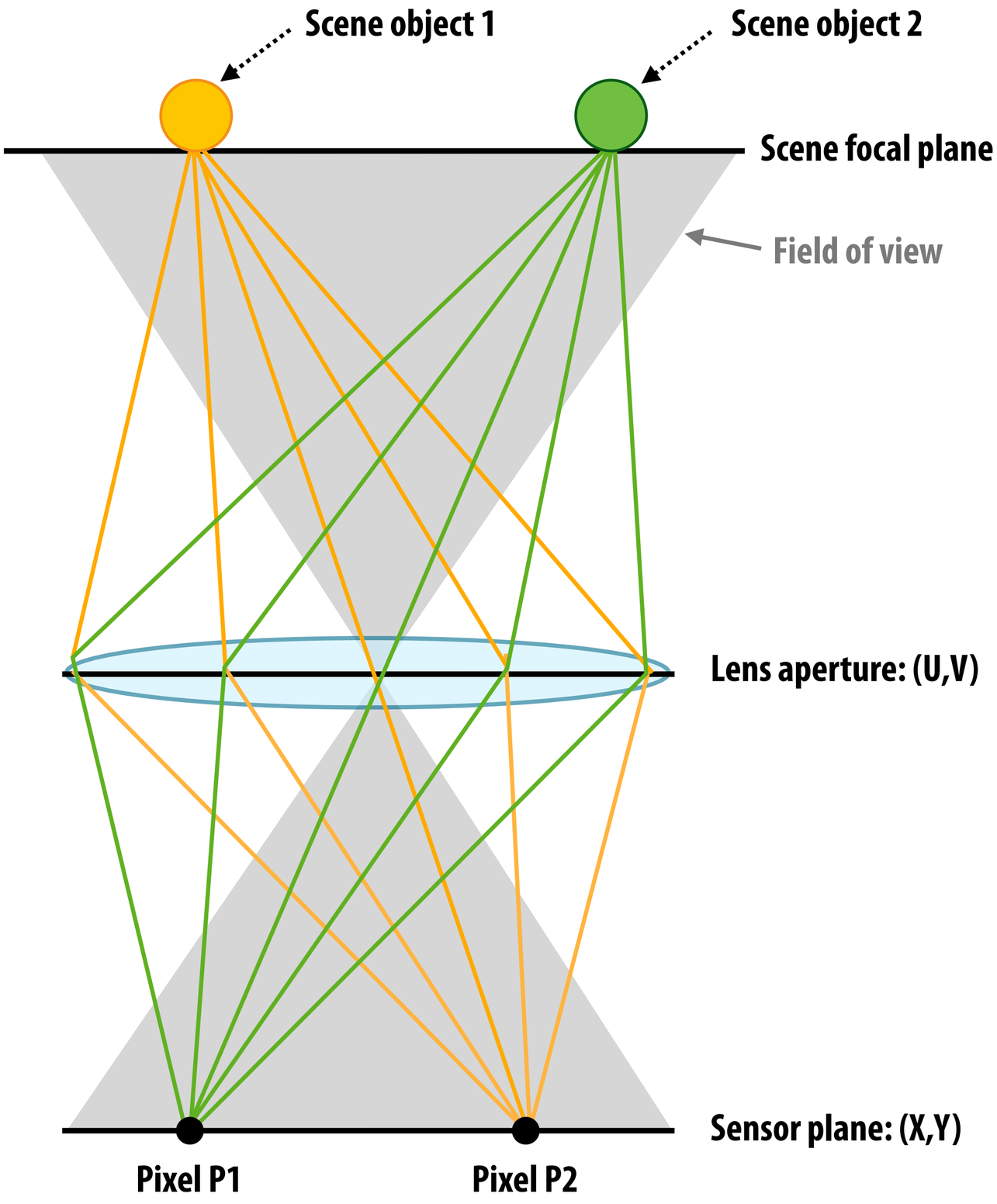
Light field inside a pinhole camera



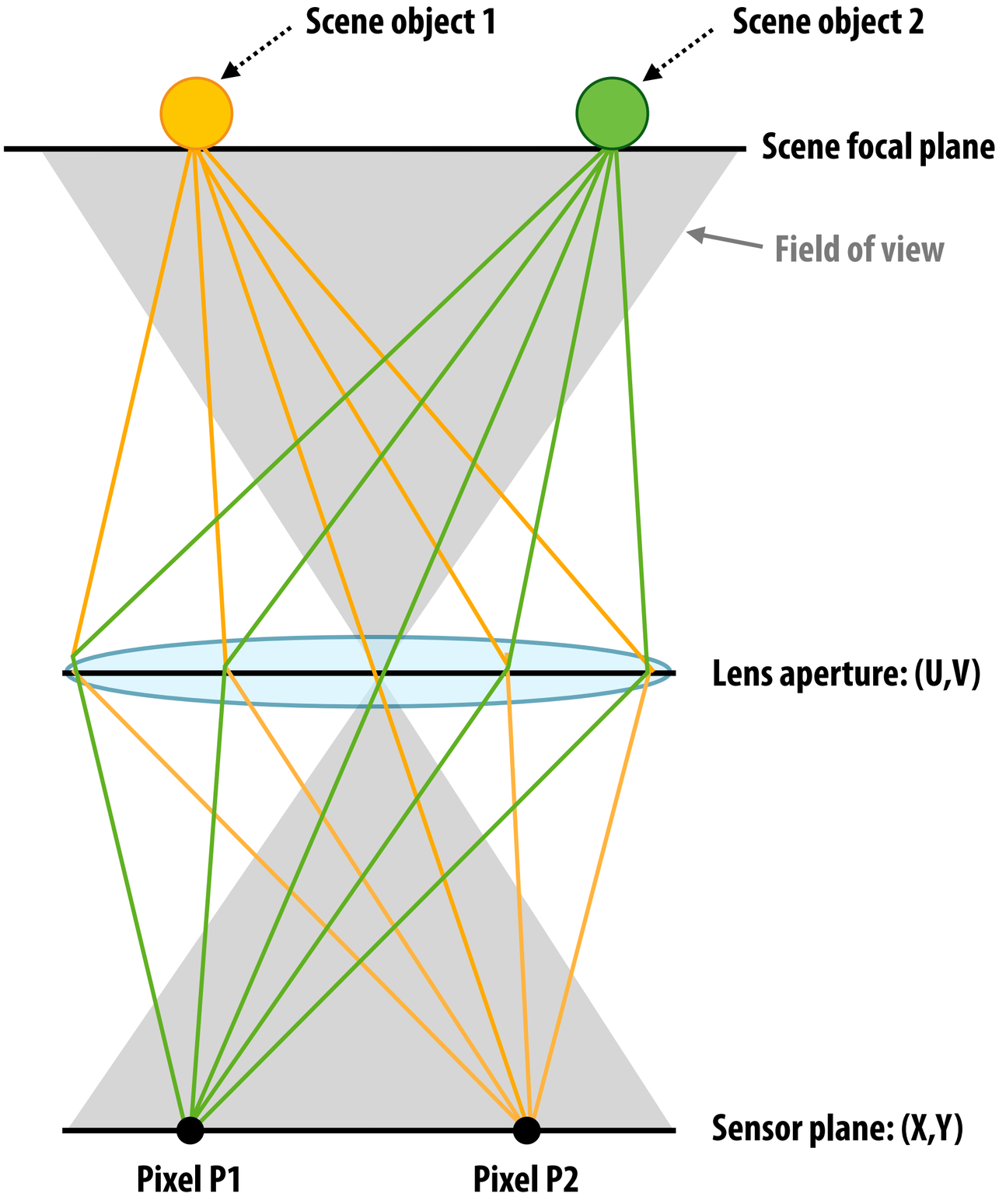
Ray space plot



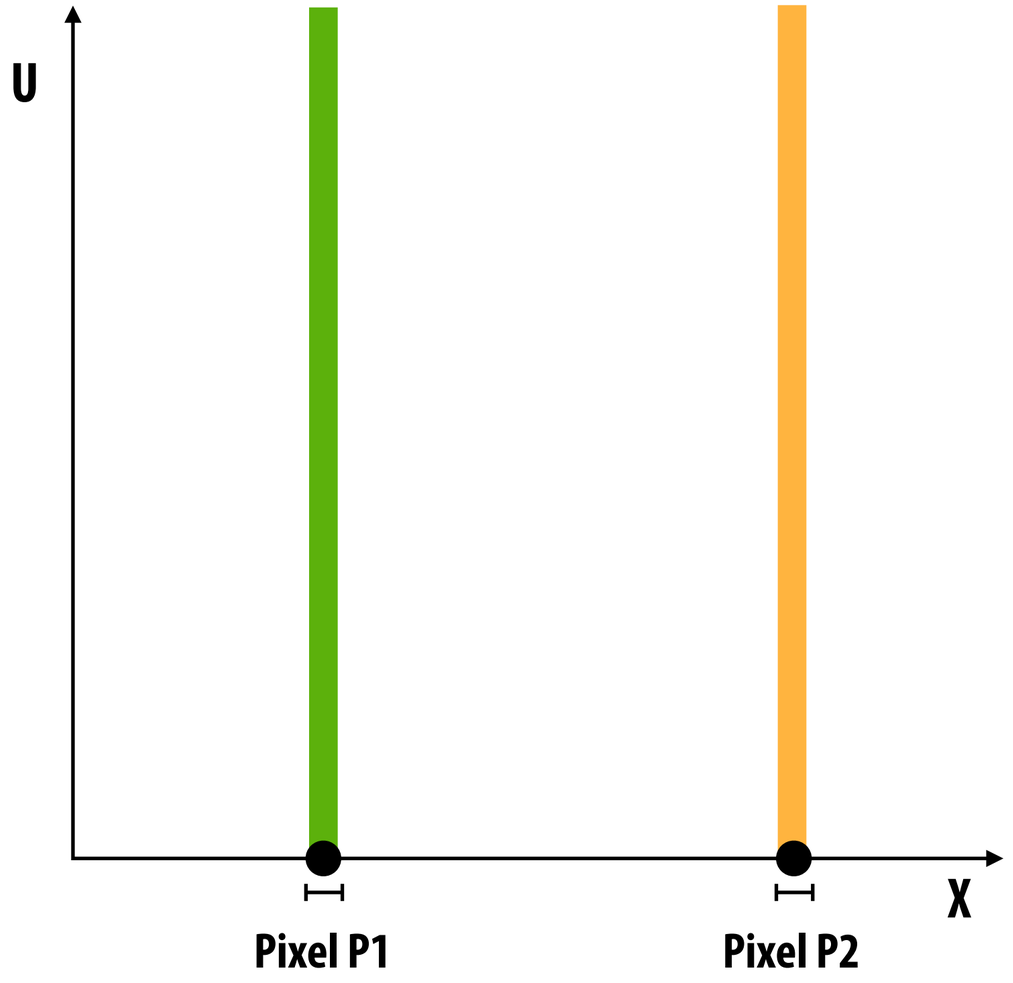
Camera with finite aperture



Light field inside a camera

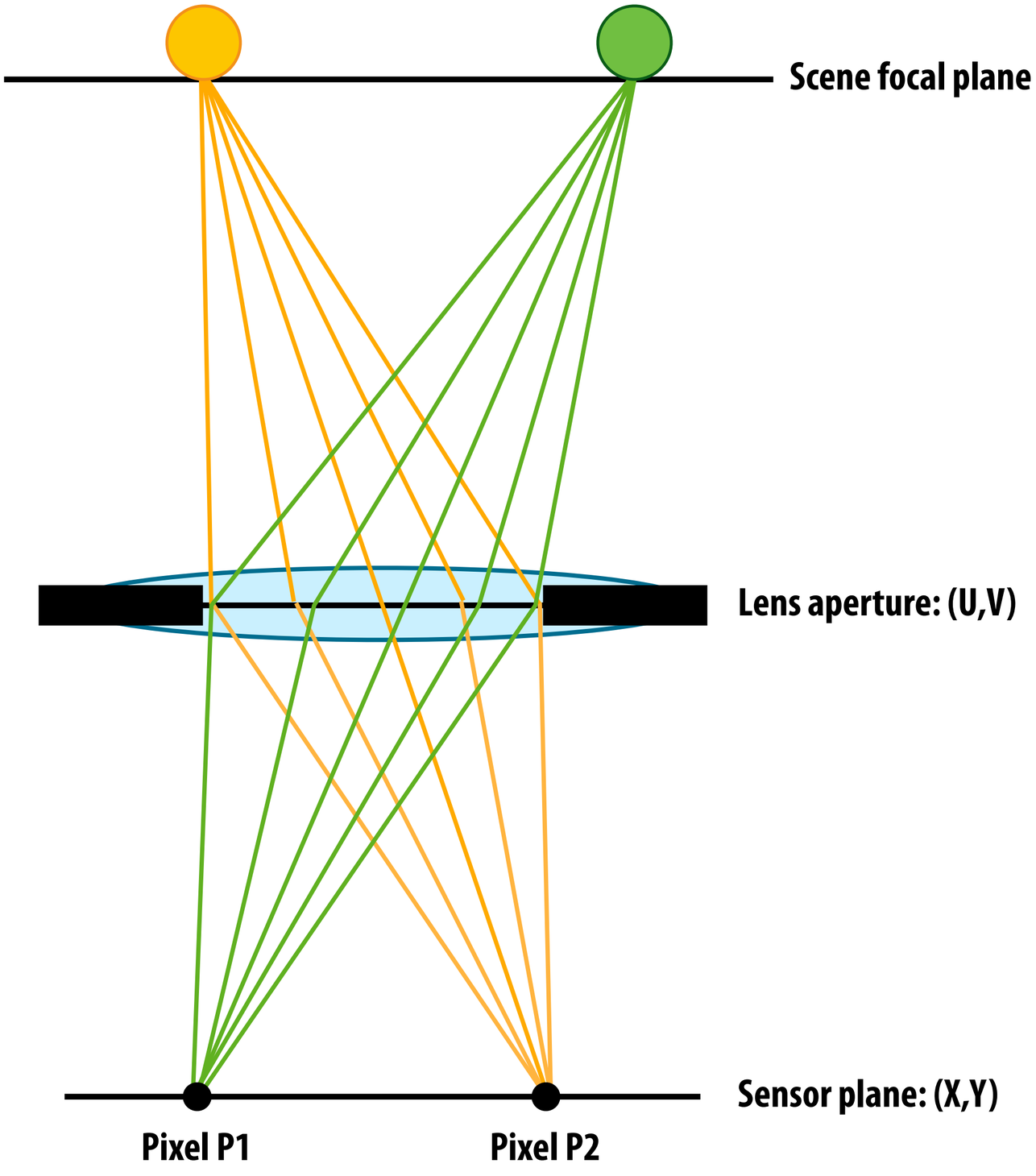


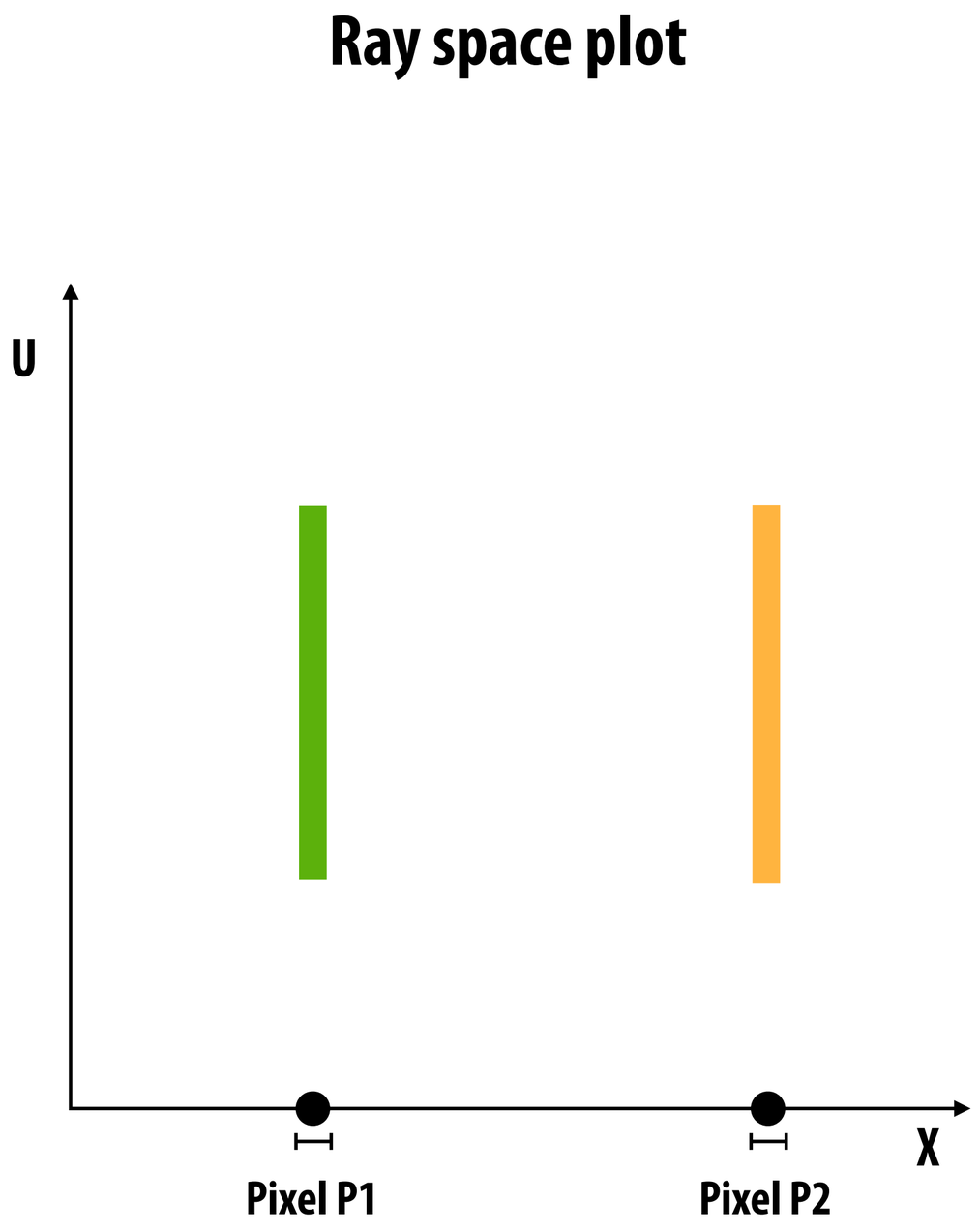
Ray space plot
(only showing X-U 2D projection)



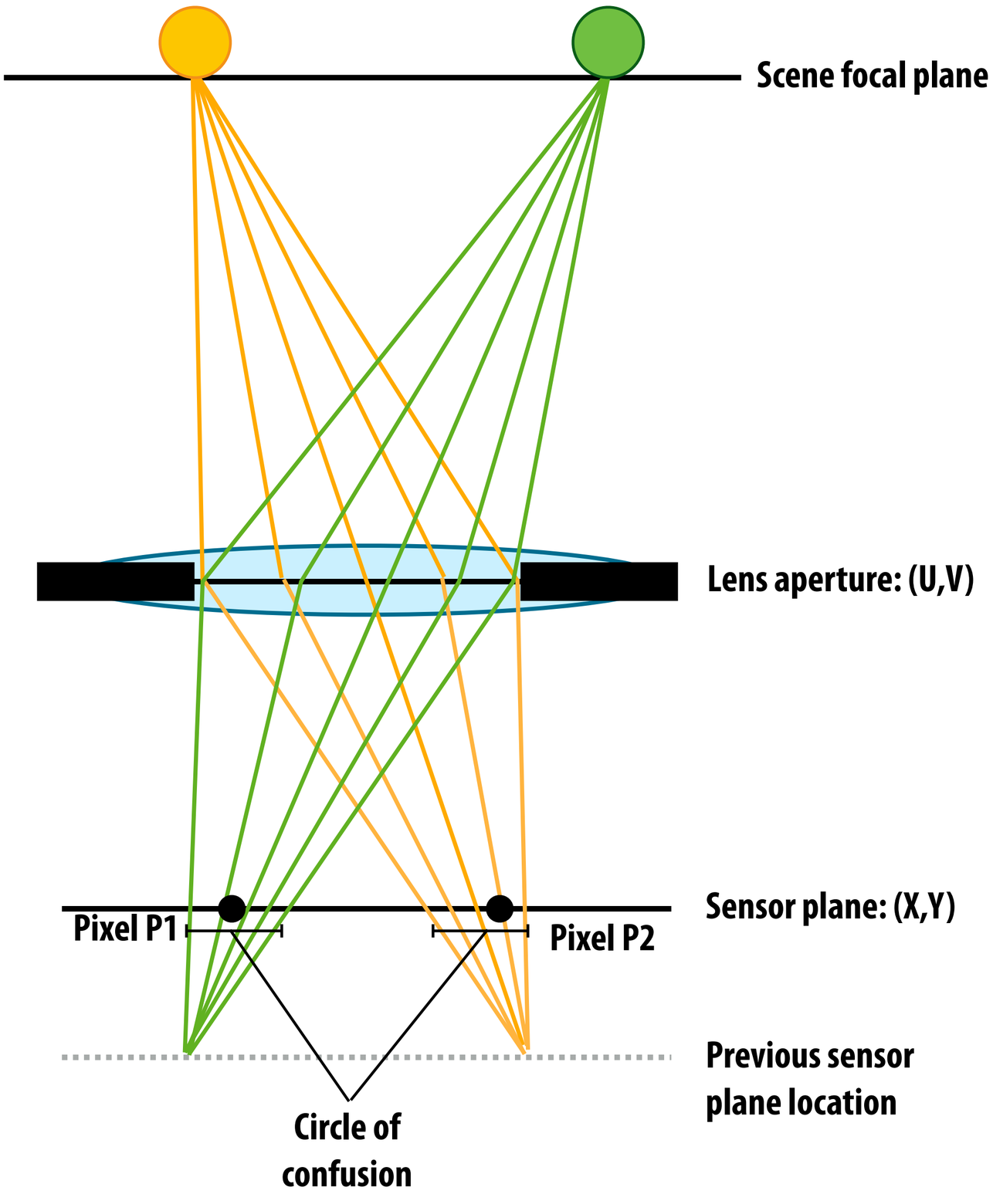
Sensor pixels measure integral of energy from all rays of light passing through points on the aperture and a pixel-sized area of the sensor.

Decrease aperture size

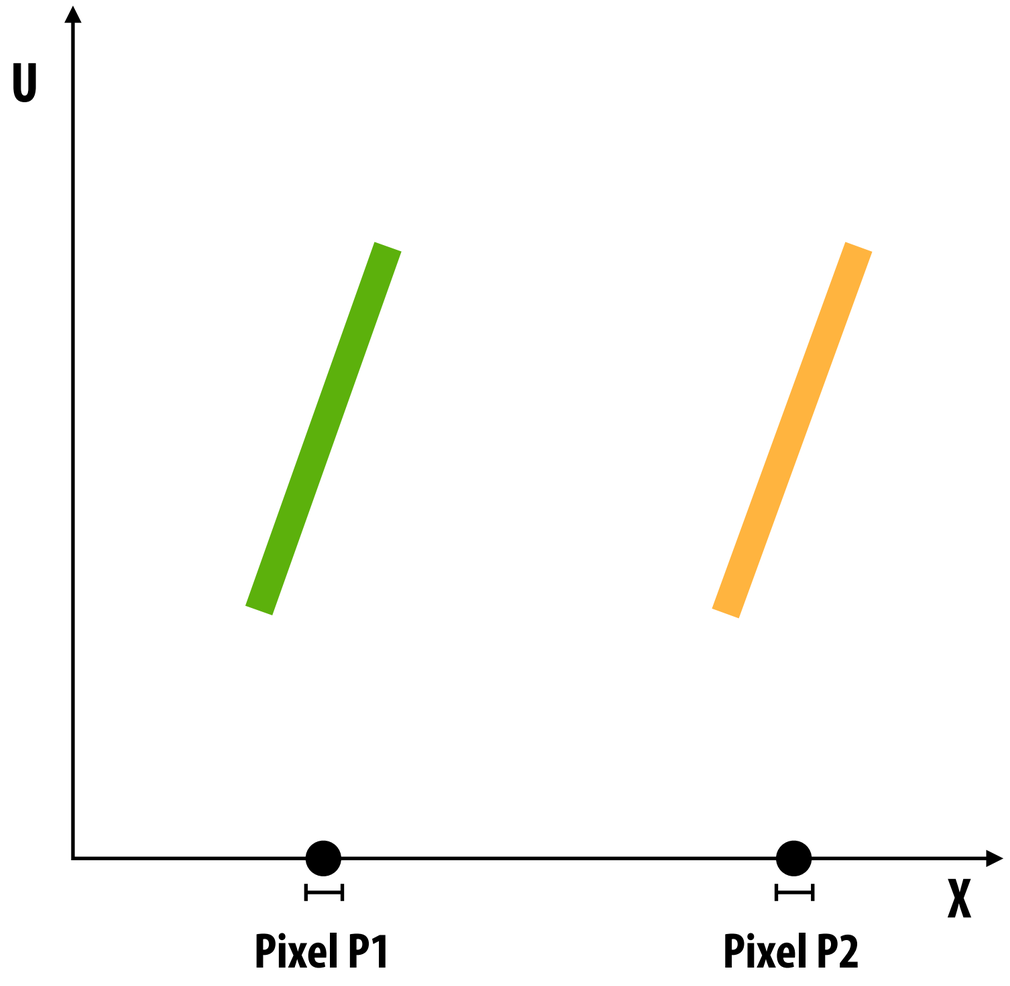




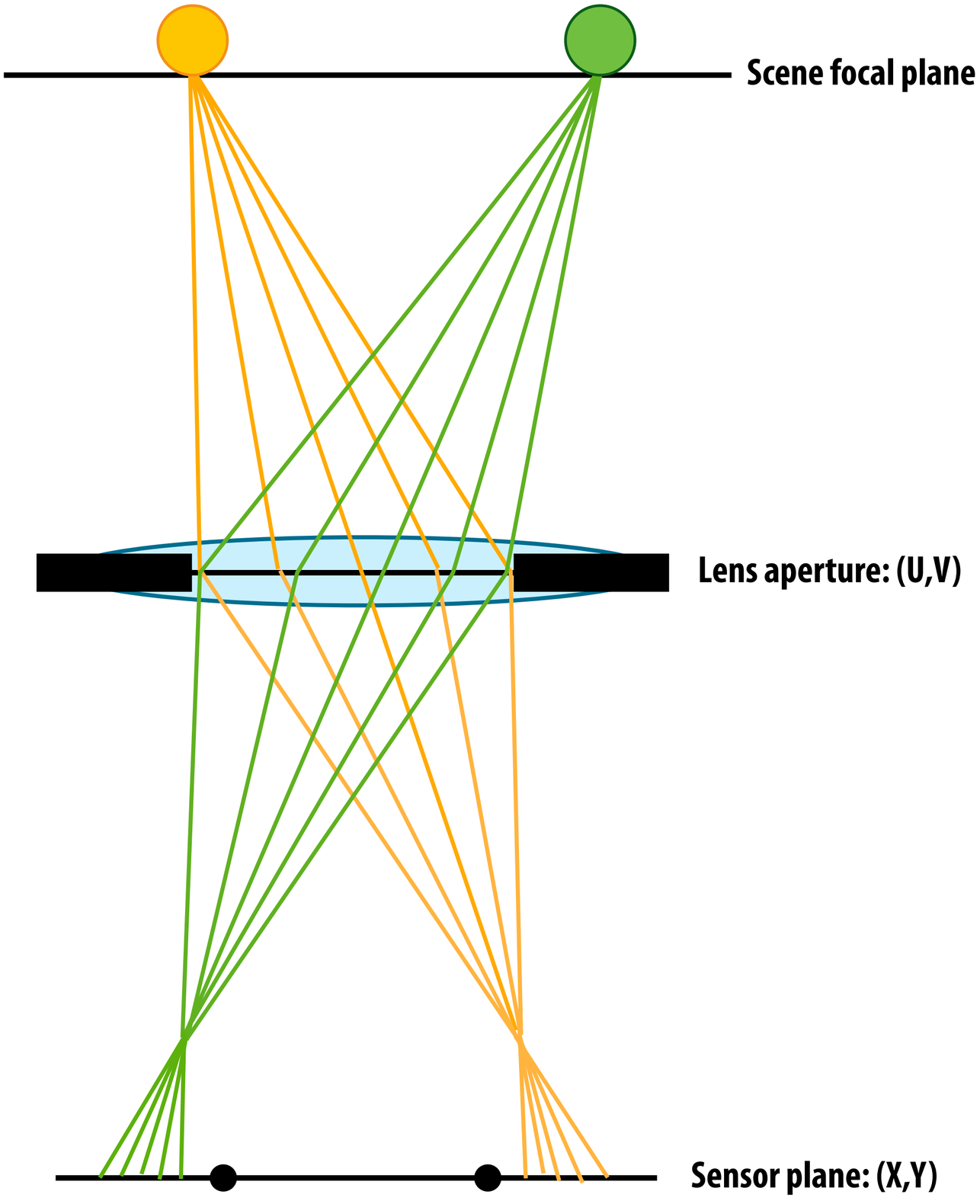
Defocus



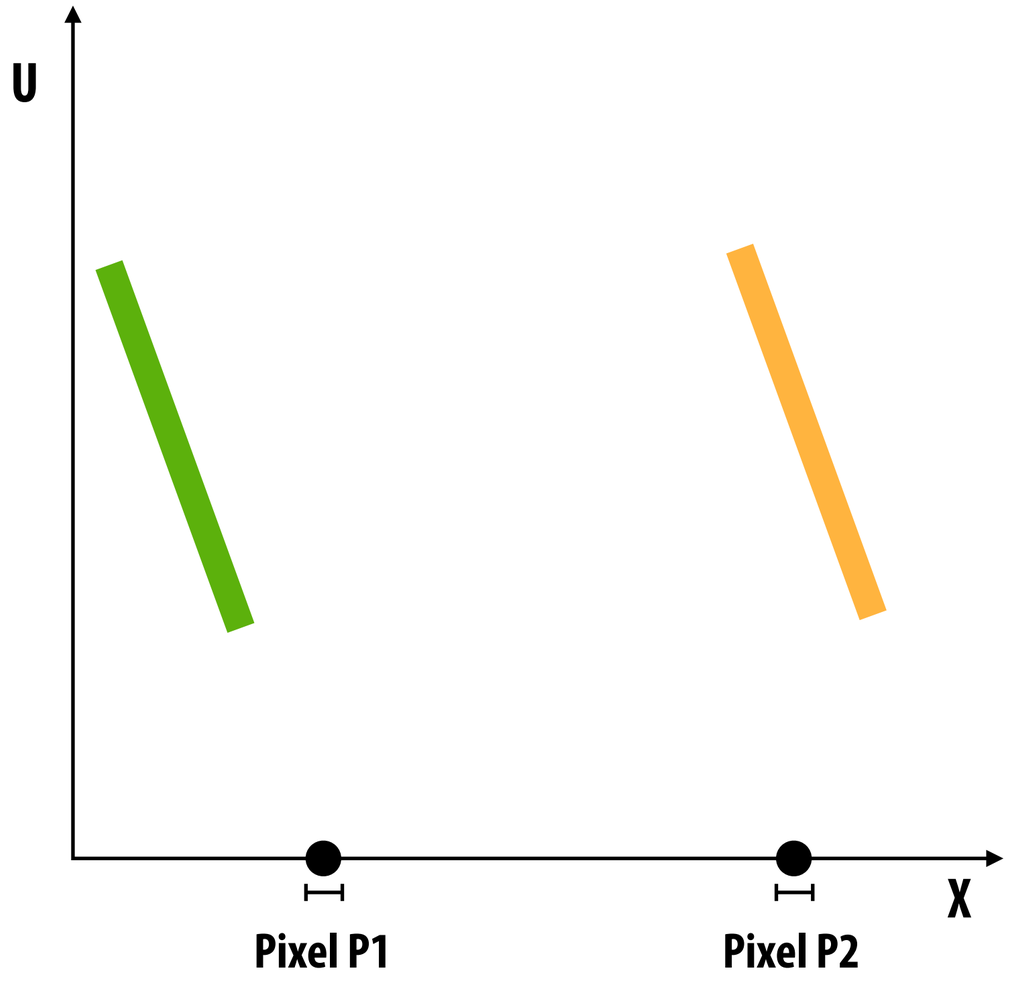
Ray space plot



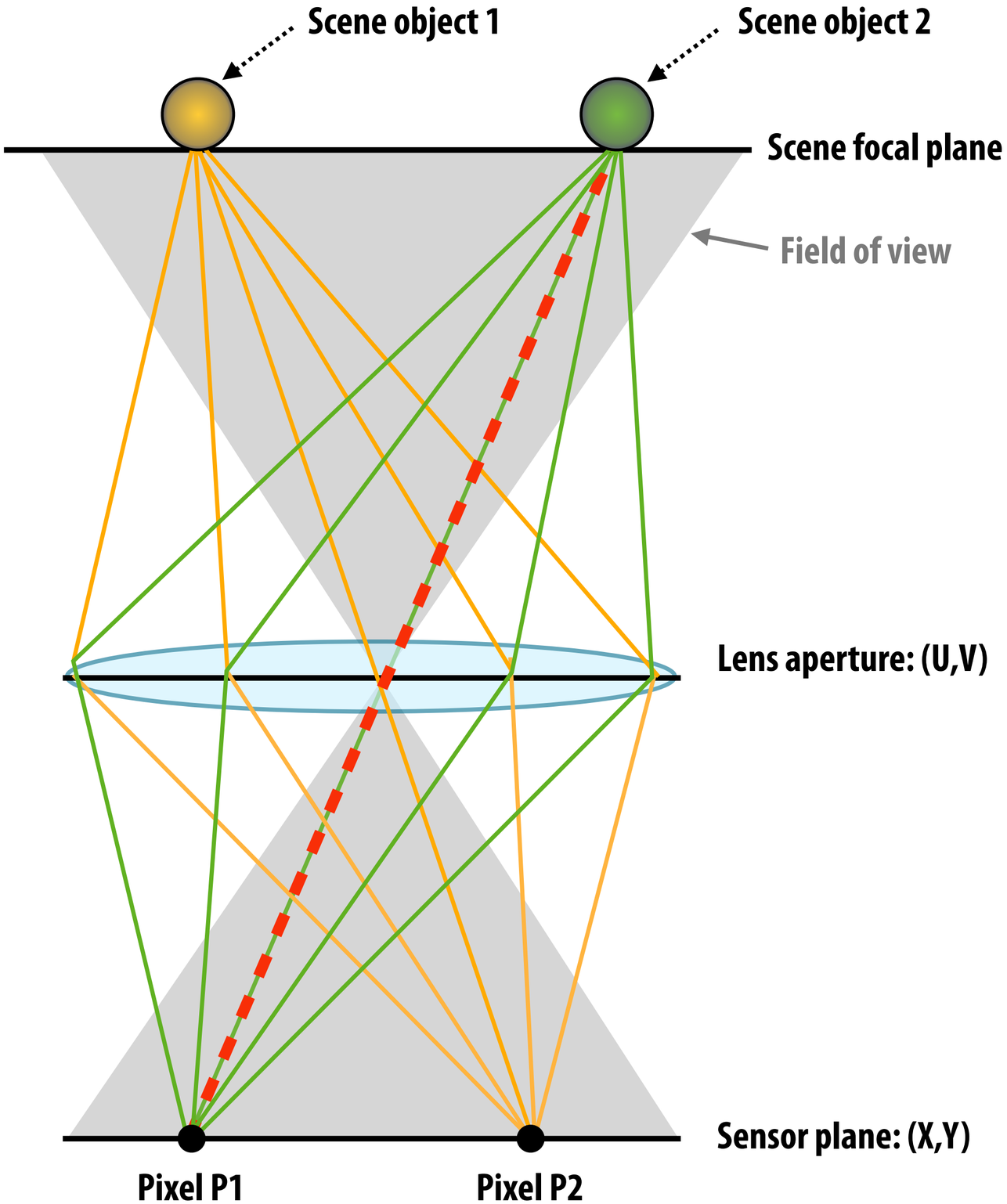
Defocus



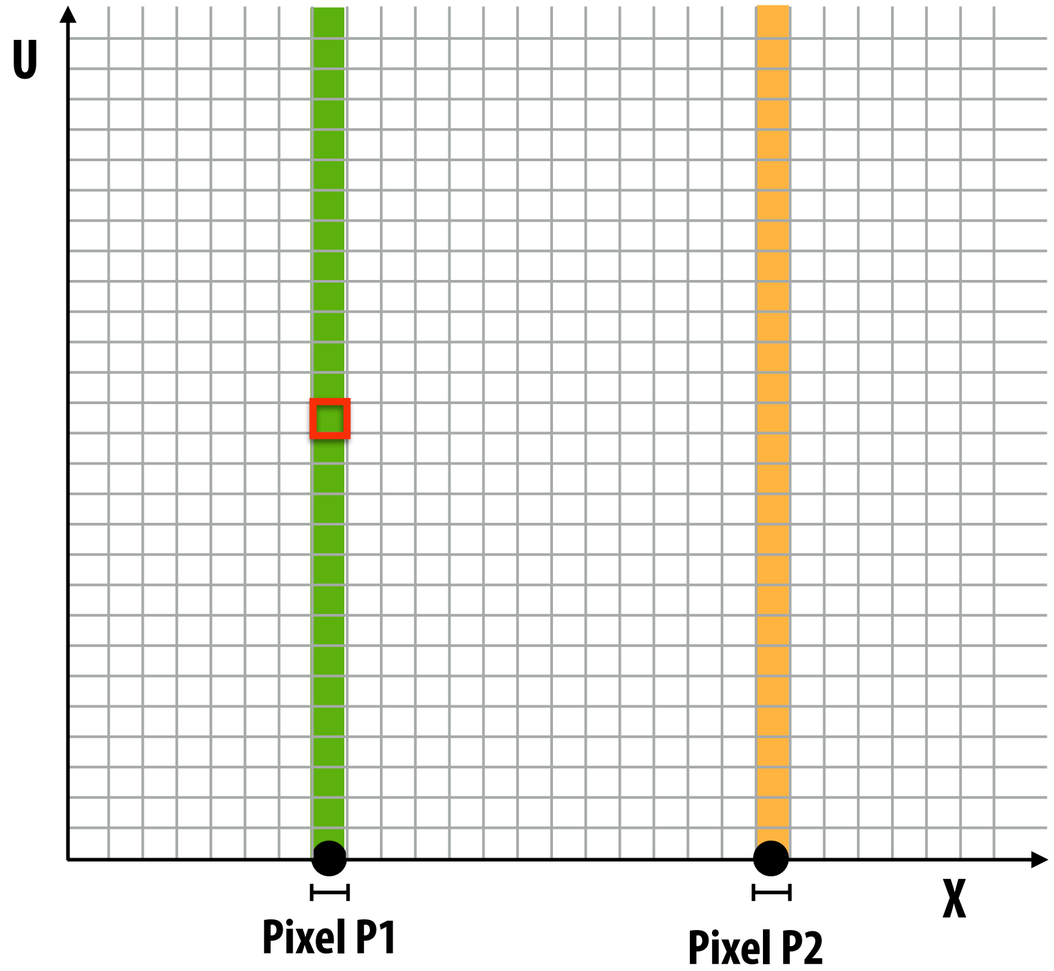
Ray space plot



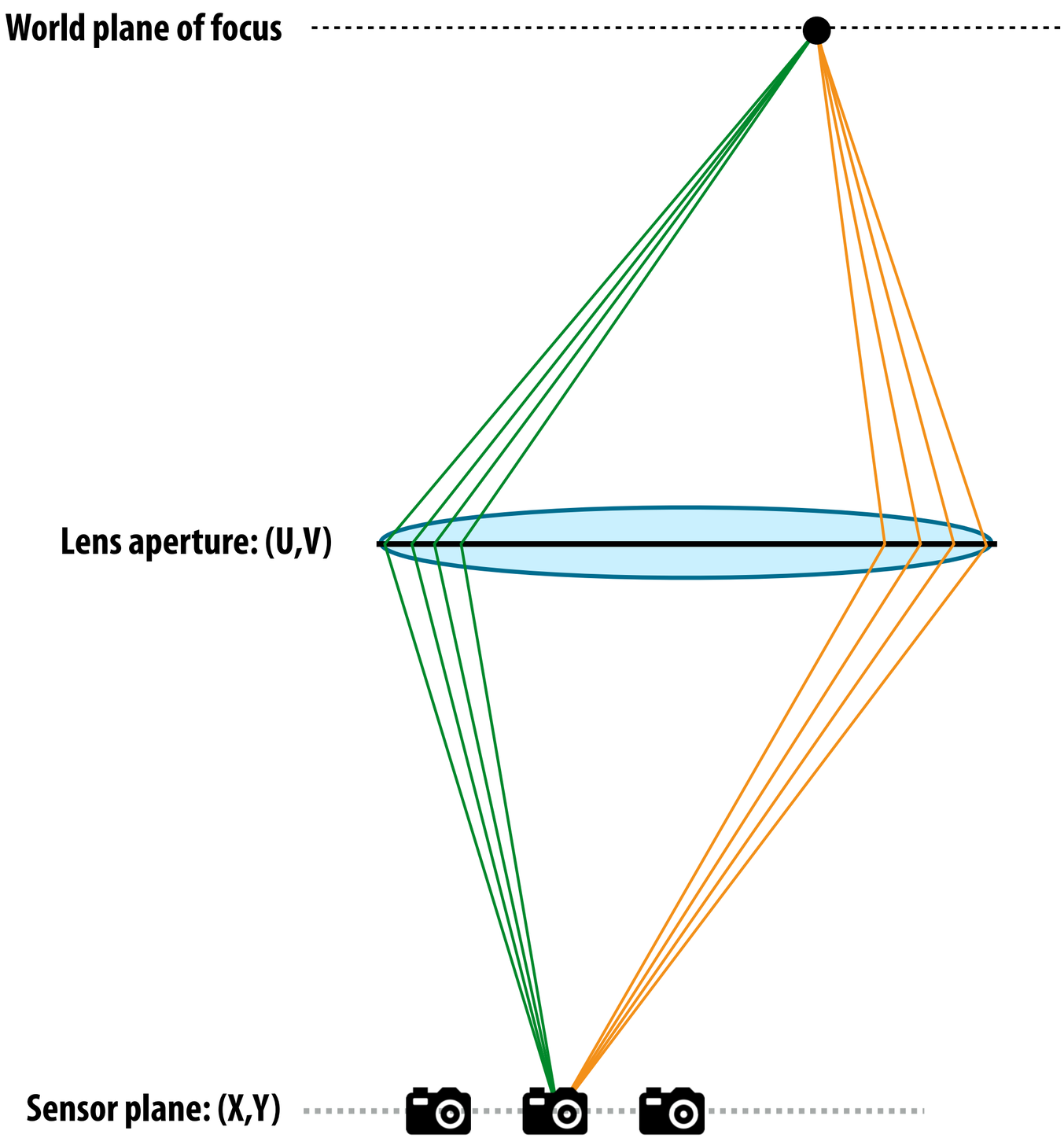
How might we measure the light field inside a camera?



Ray space plot
(only showing X-U 2D projection)



Intuition: handheld light field camera

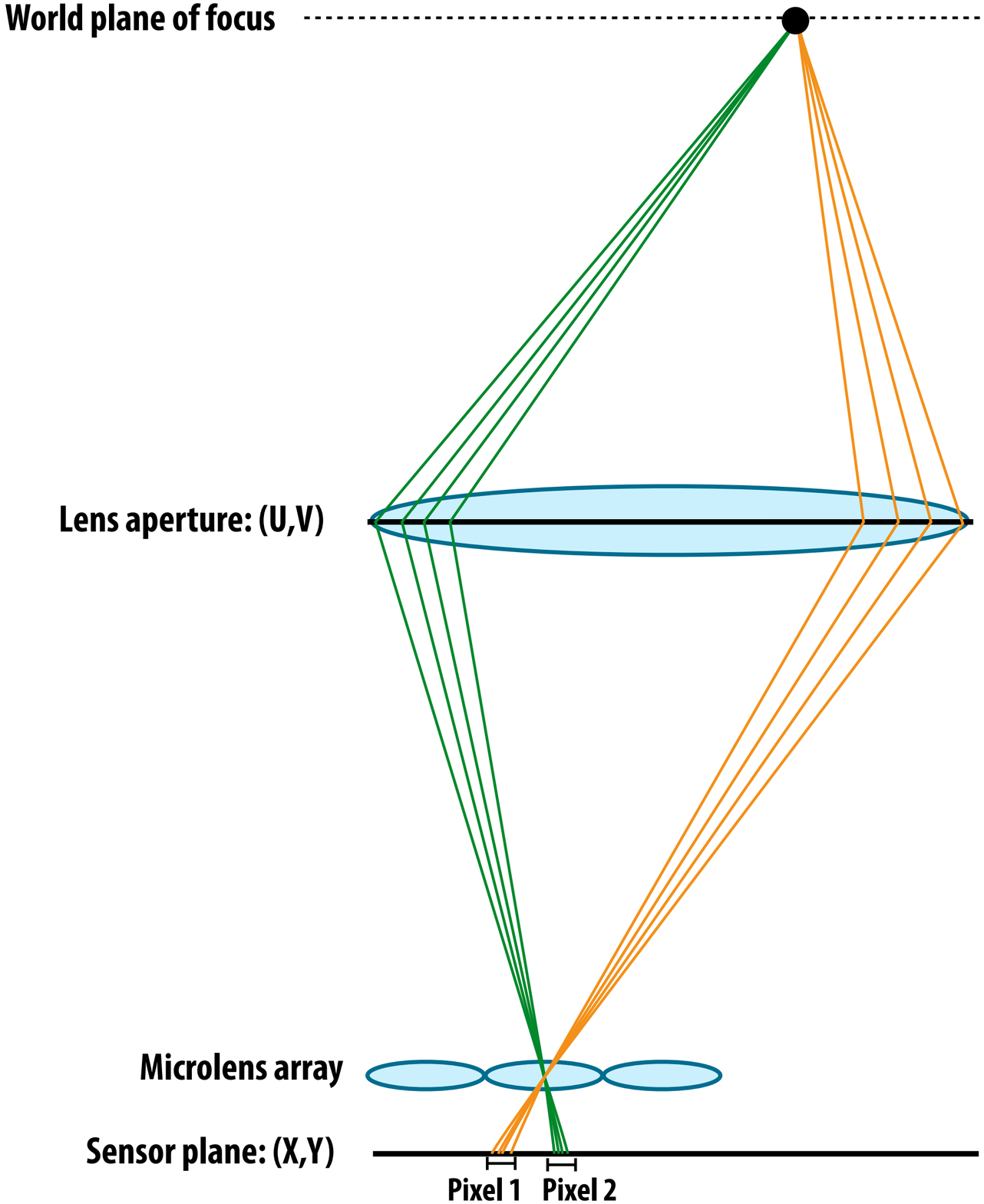


[Ng et al. 2005]

[Adelson and Wang, 1992]

Intuition: build an optical system where each region of the sensor "takes" a picture of the aperture of the main lens

Handheld light field camera

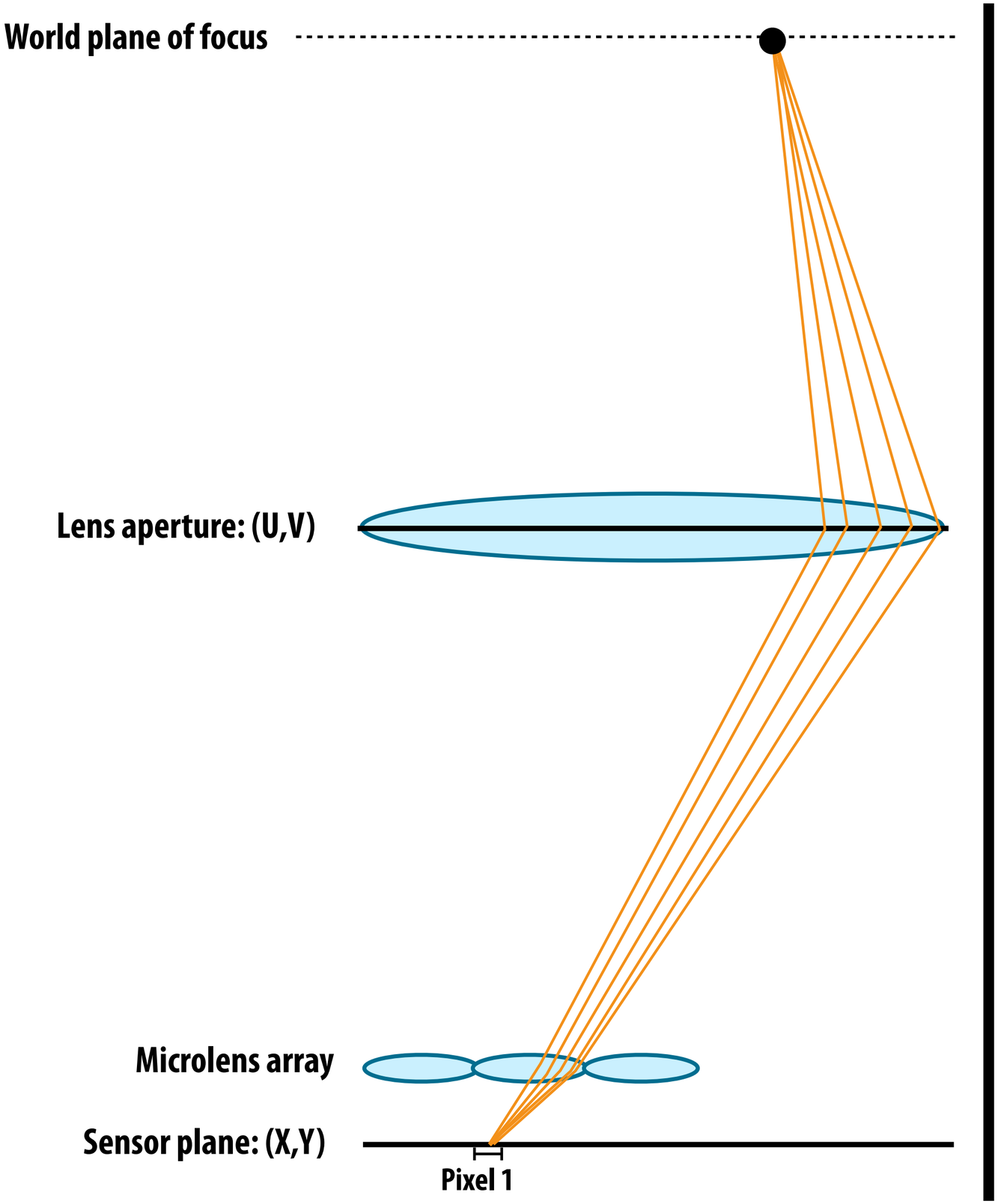


[Ng et al. 2005]

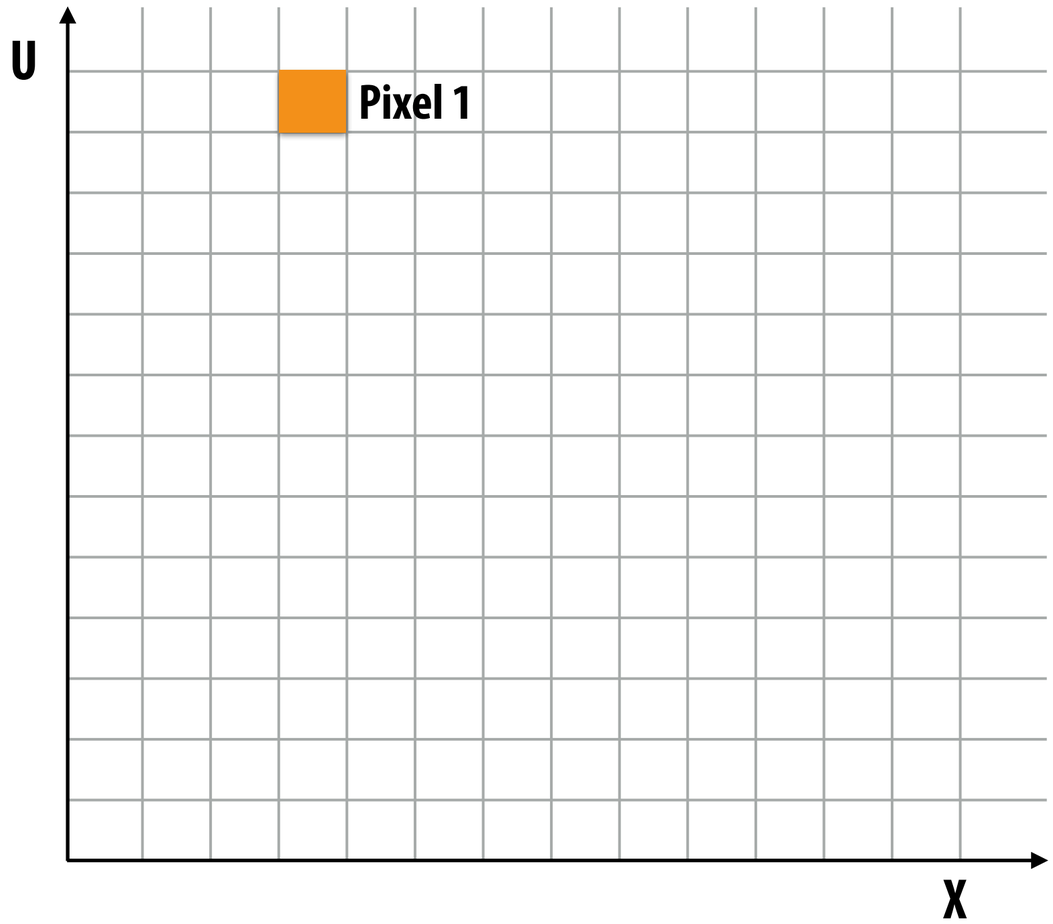
[Adelson and Wang, 1992]

Implementation: microlens array placed just on top of the sensor.

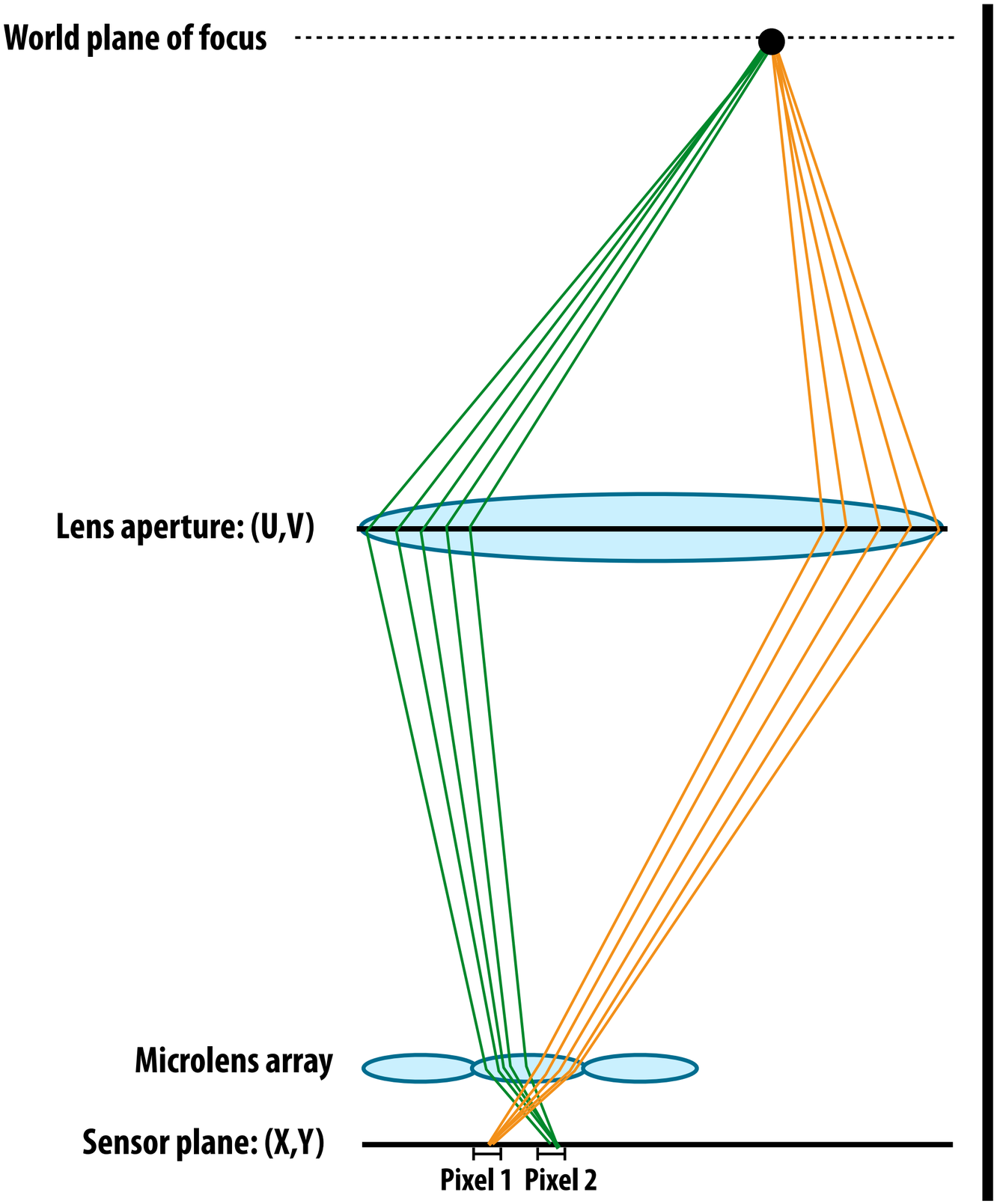
Each sensor pixel records a small beam of light inside the camera



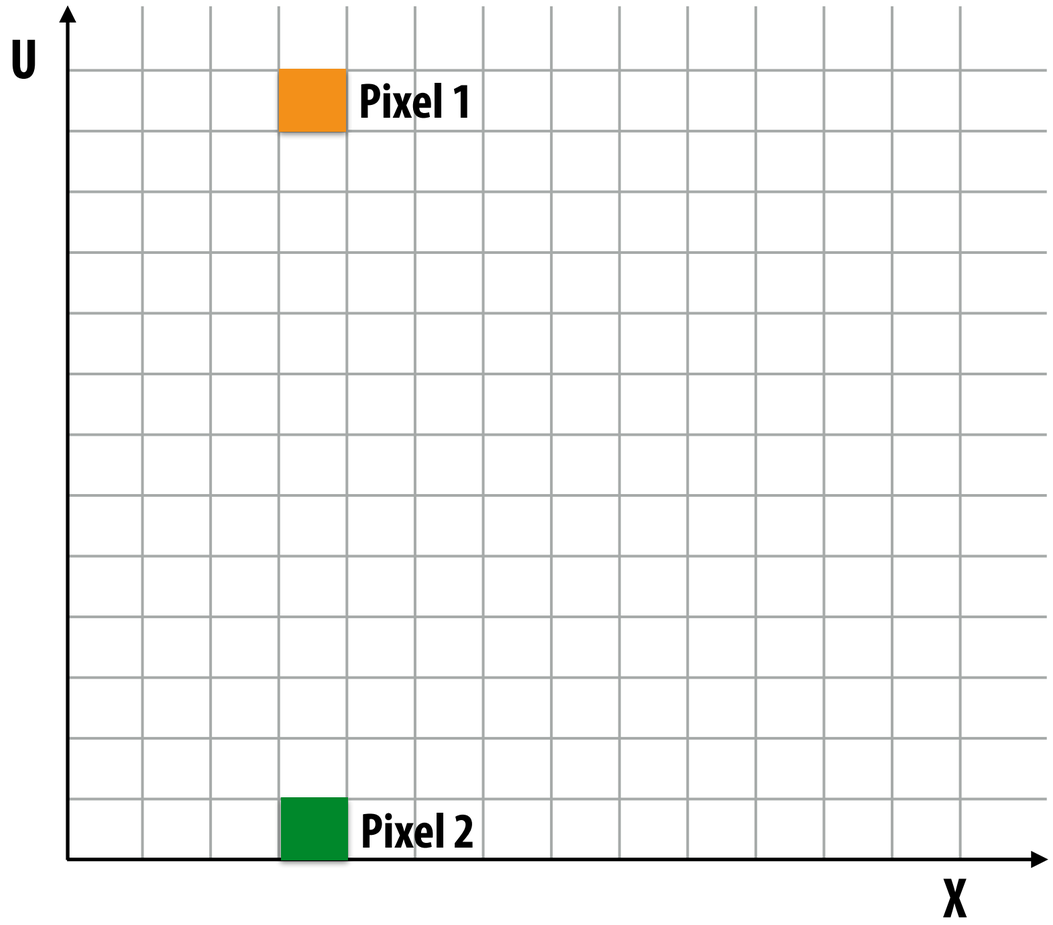
Ray space plot



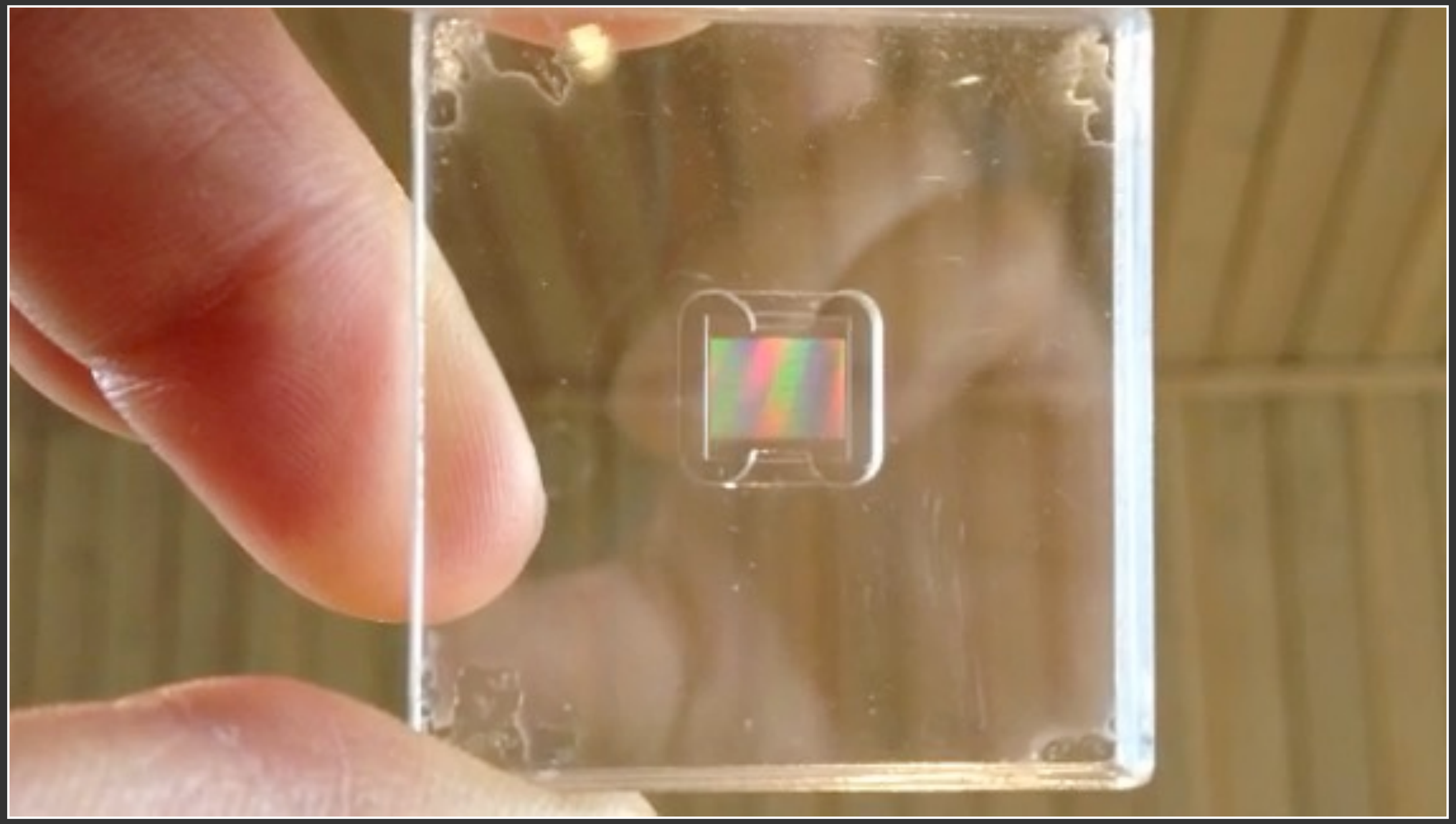
Each sensor pixel records a small beam of light inside the camera



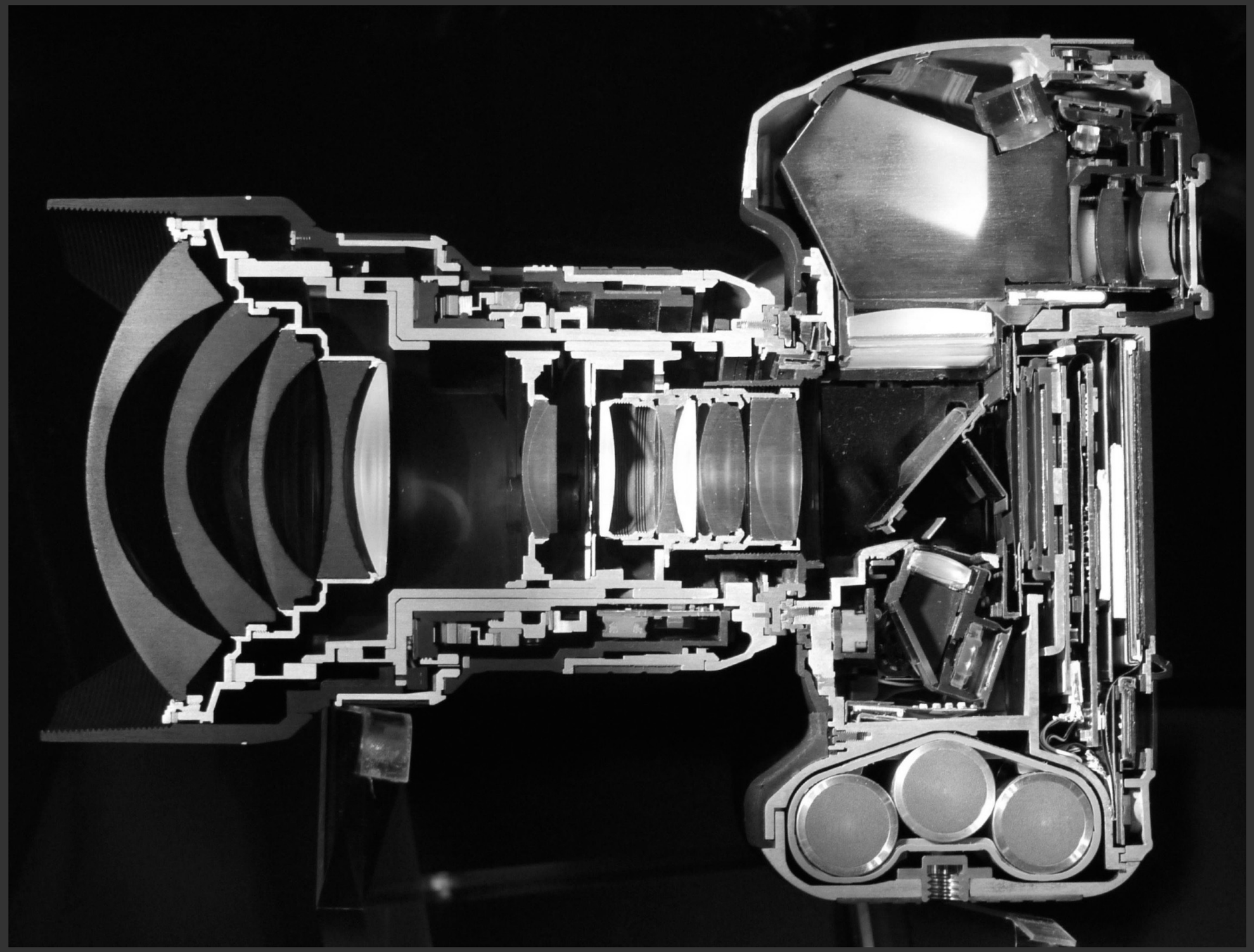
Ray space plot



Microlens Array

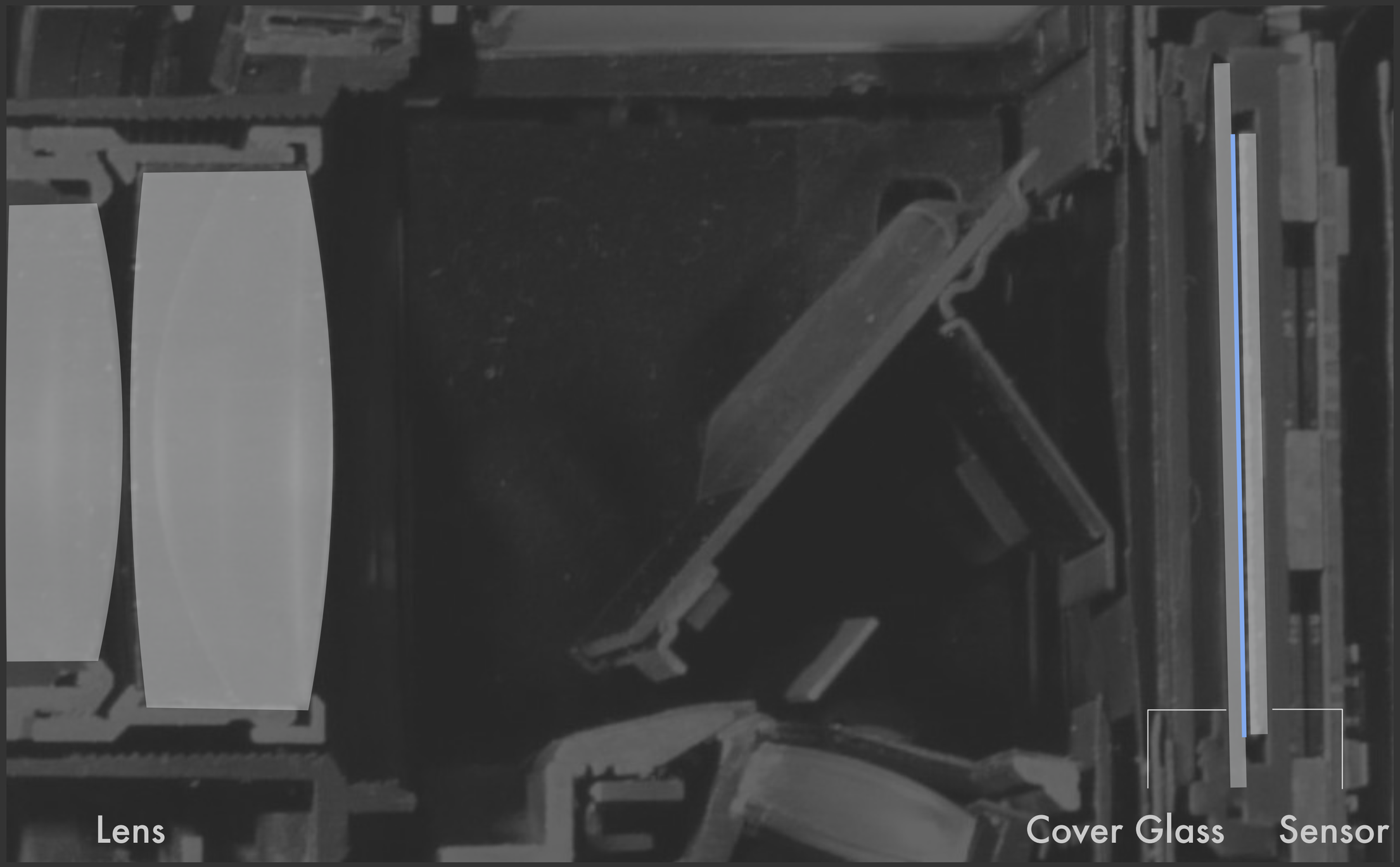


Where Microlenses Go Inside Camera



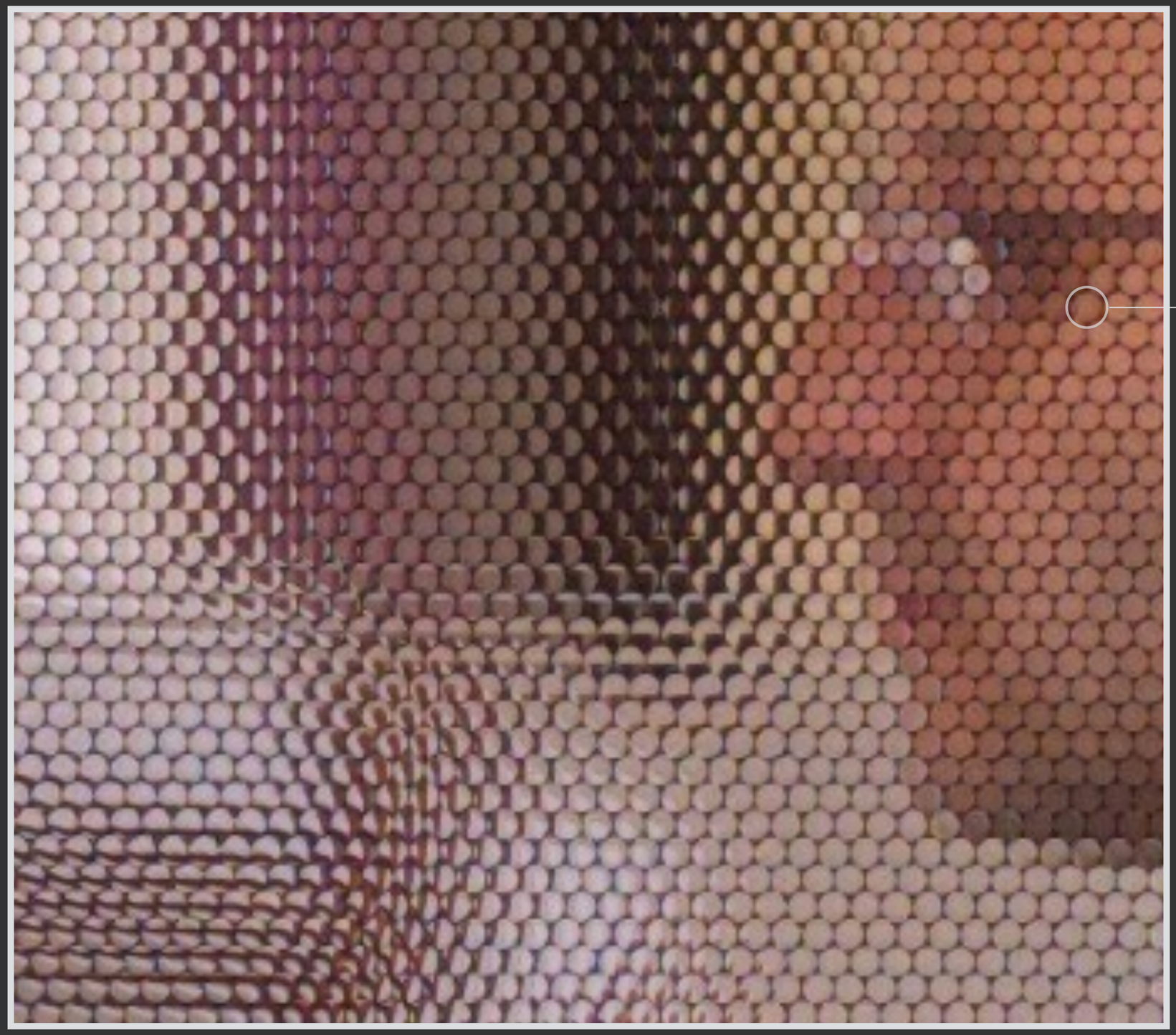
Cross-section of Nikon D3, 14-24mm F28 lens, Credit: ??

Where Microlenses Go Inside Camera

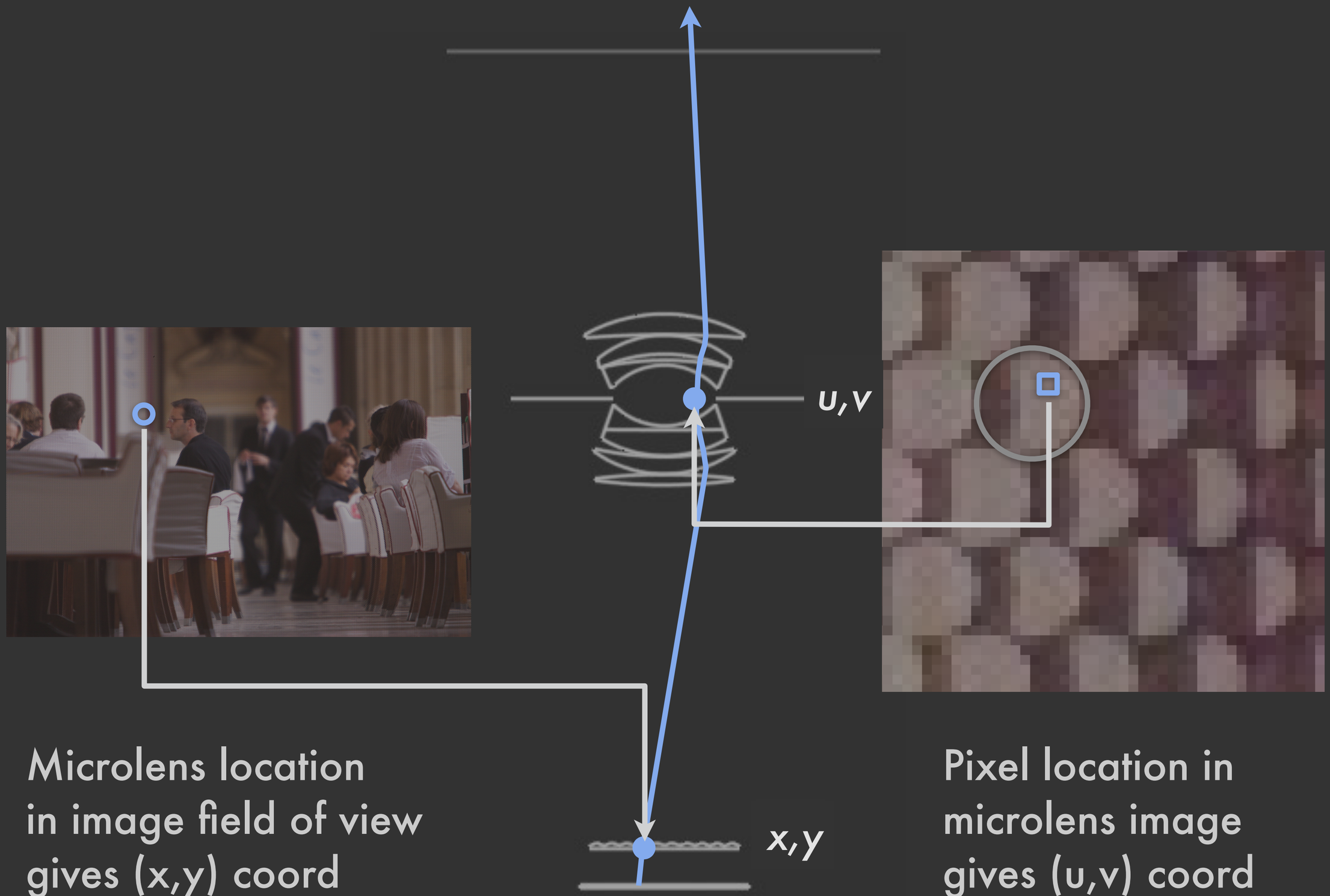


Raw Data From Light Field Sensor

Raw Data From Light Field Sensor



Mapping Sensor Pixels to (x,y,u,v) Rays



Microlens location
in image field of view
gives (x,y) coord

Pixel location in
microlens image
gives (u,v) coord

Sub-Aperture Images

Image from selecting the same pixel under every microlens

Sub-Aperture Images

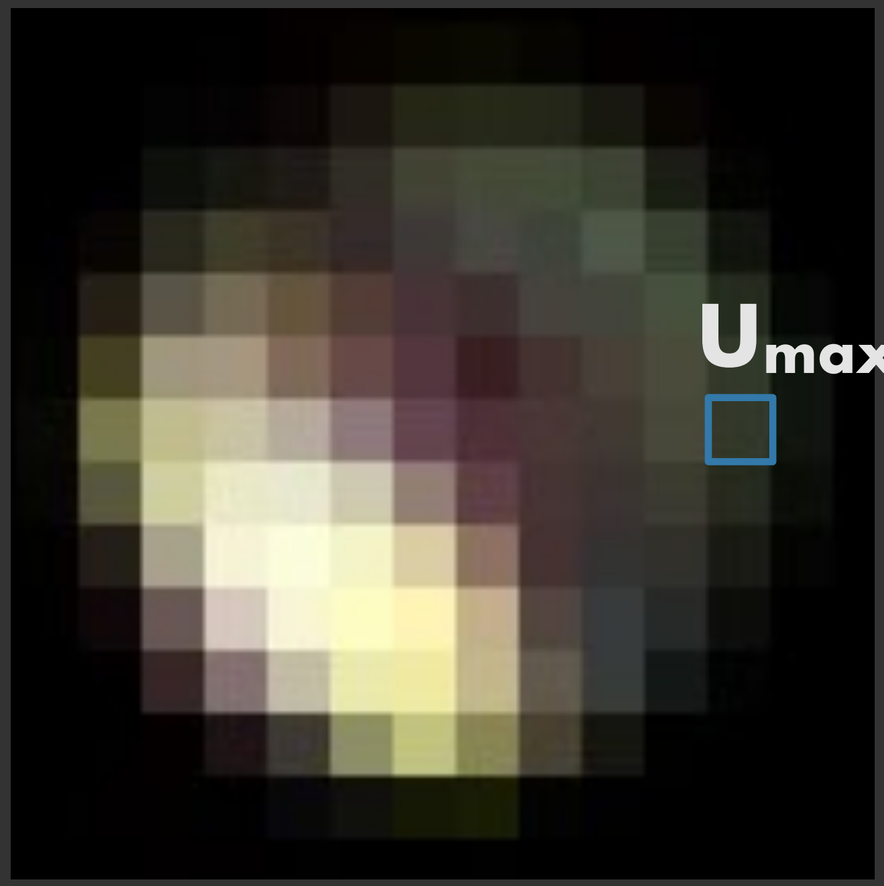
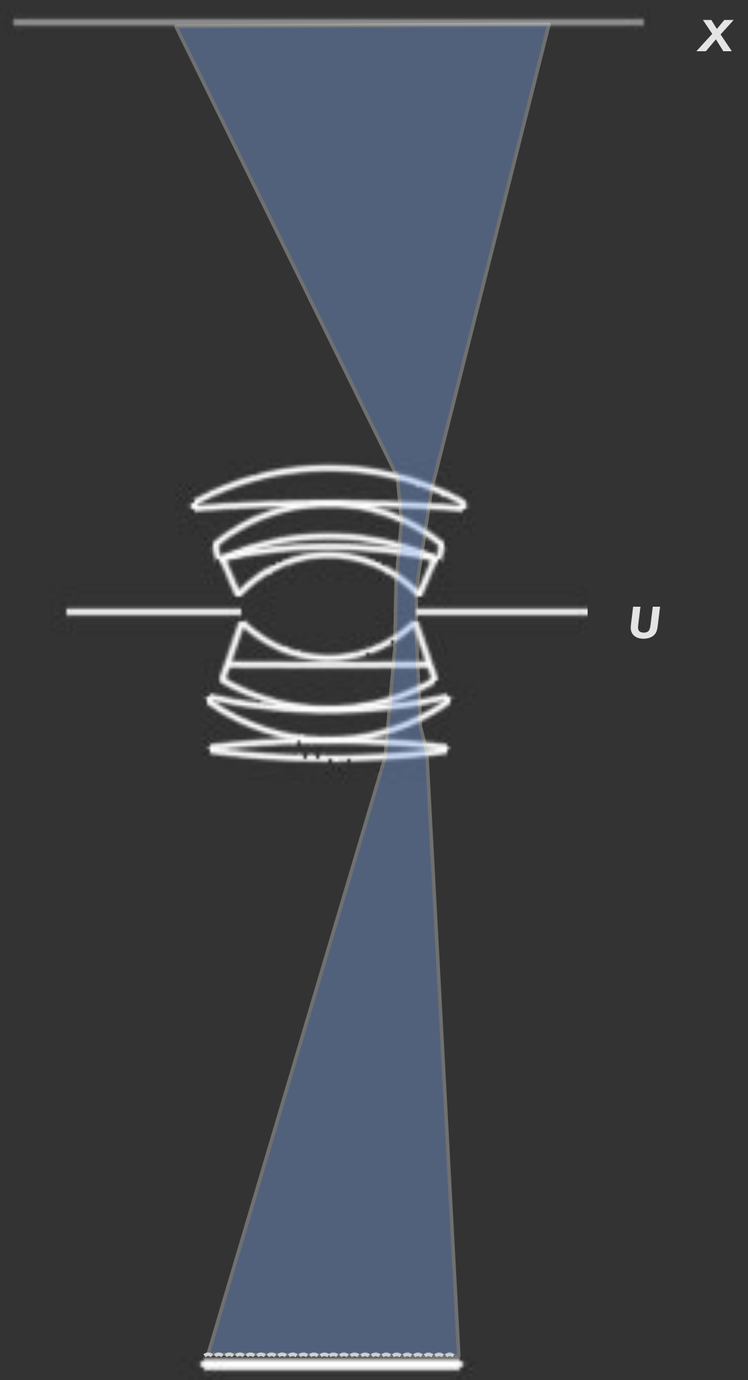
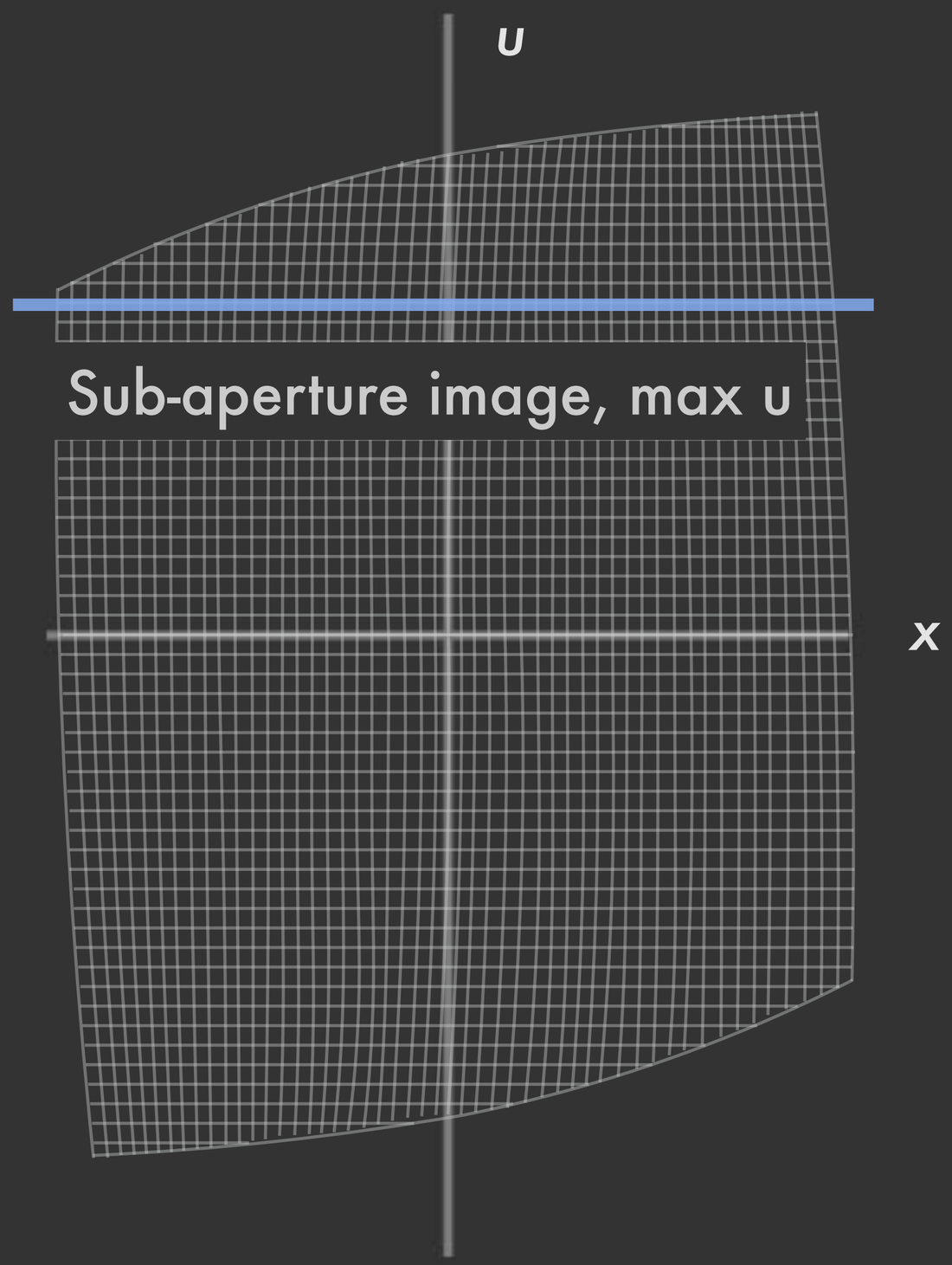


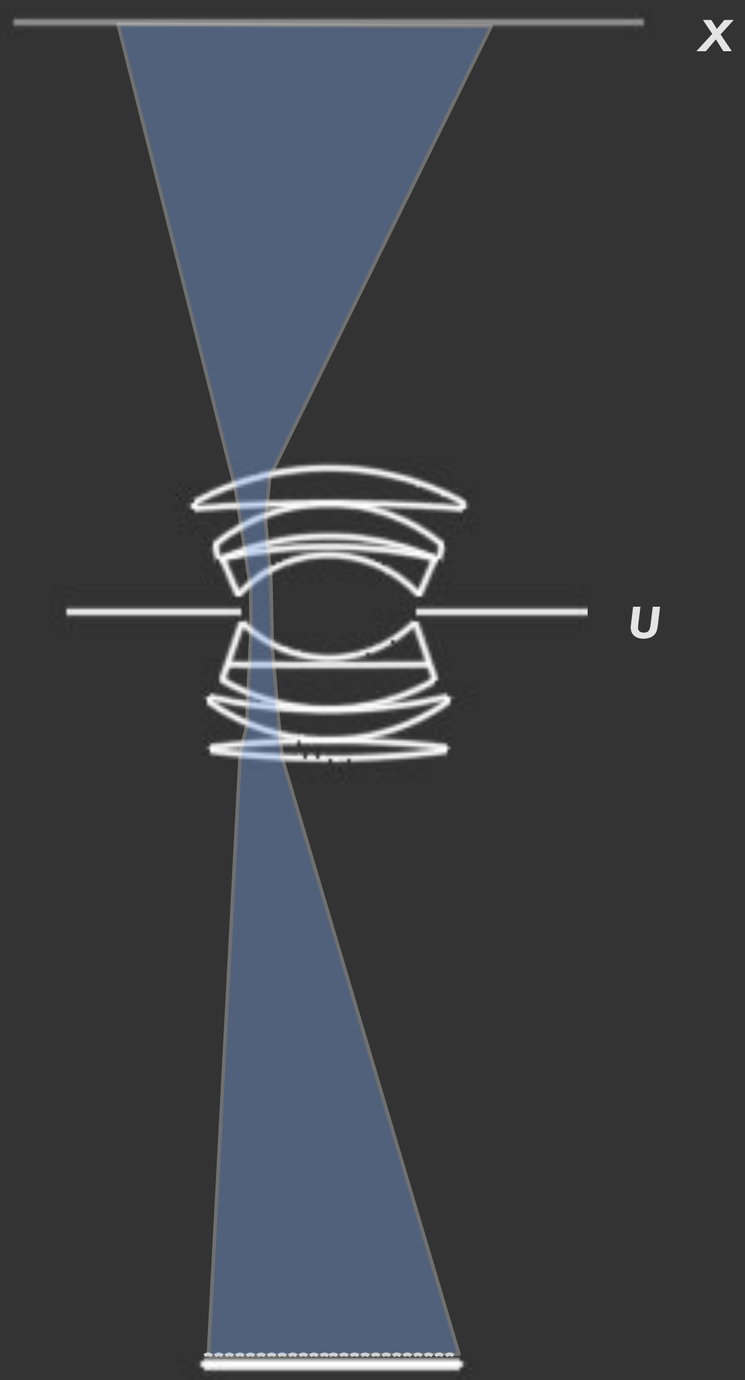
Image from selecting same pixel under every microlens

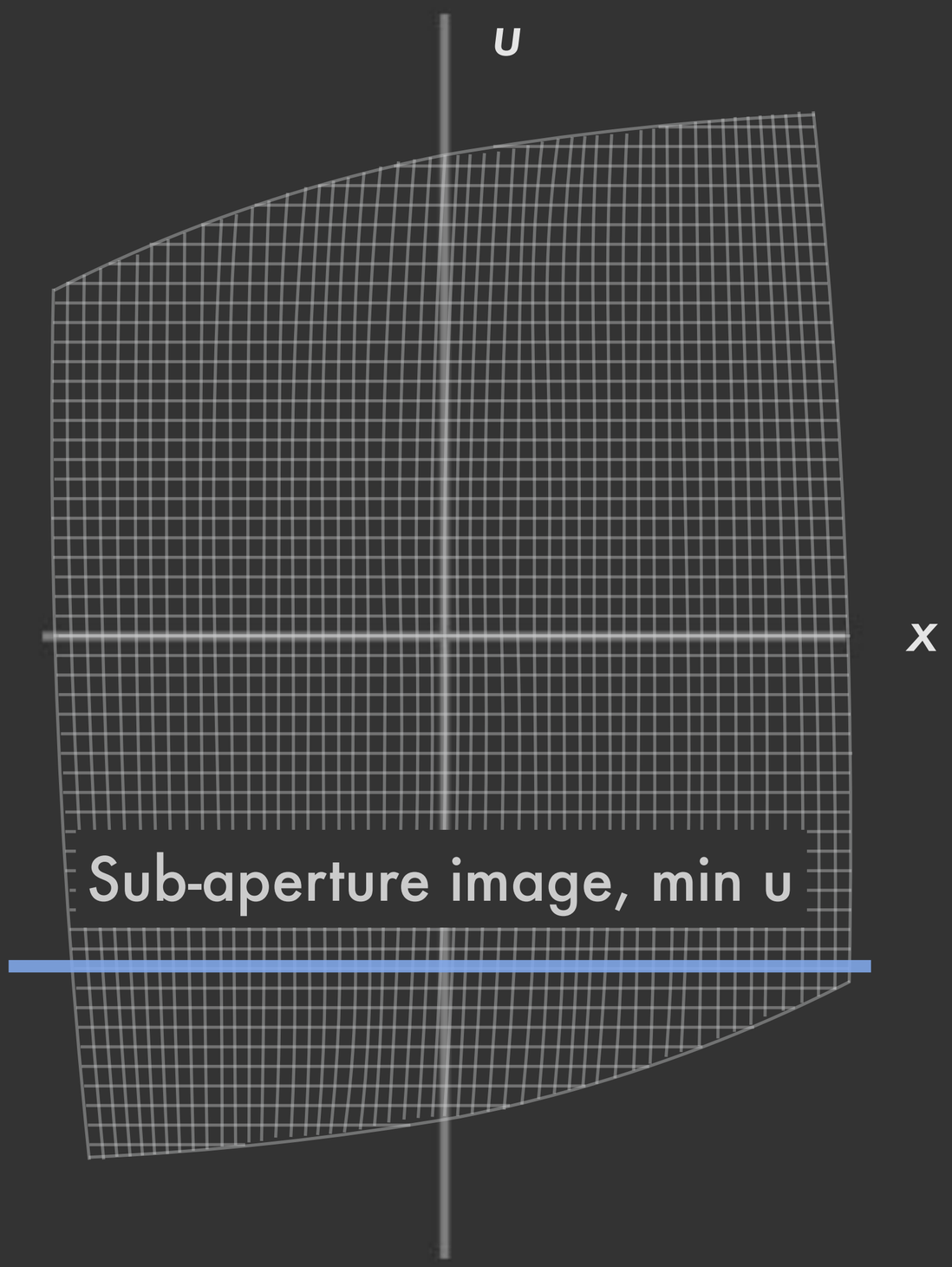
Sub-Aperture Images





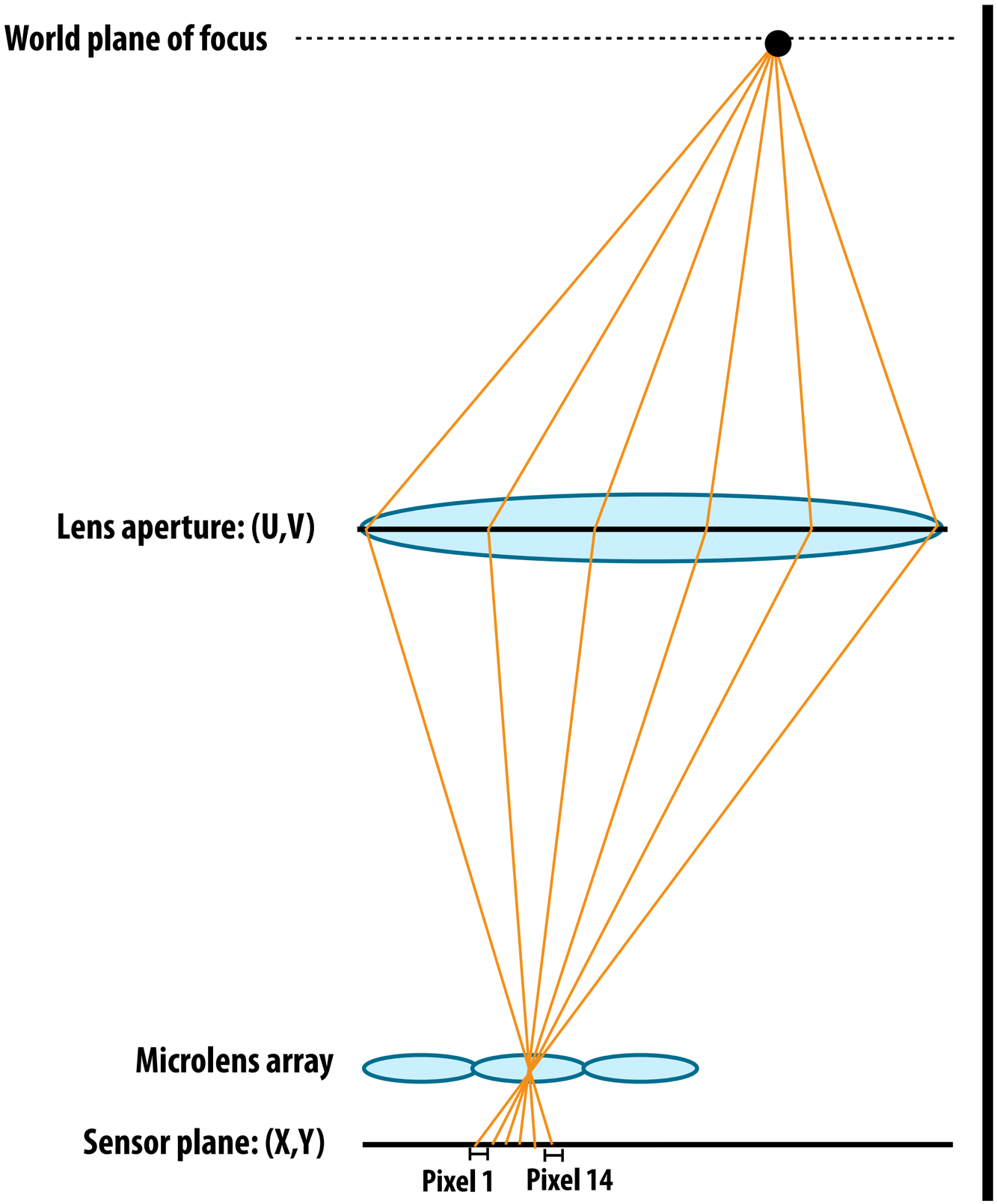
Sub-Aperture Images



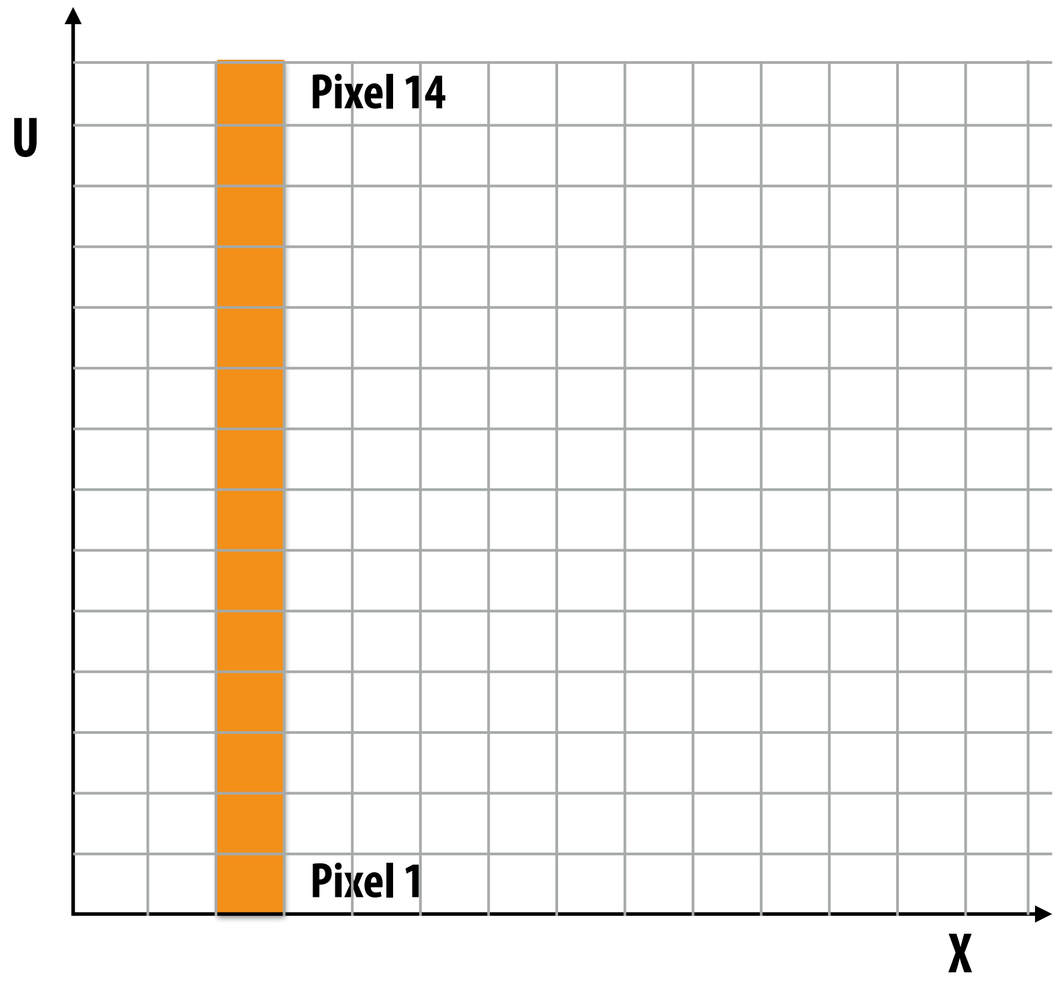


How do you compute a regular photograph from a light field?

Computing a photograph from a light field

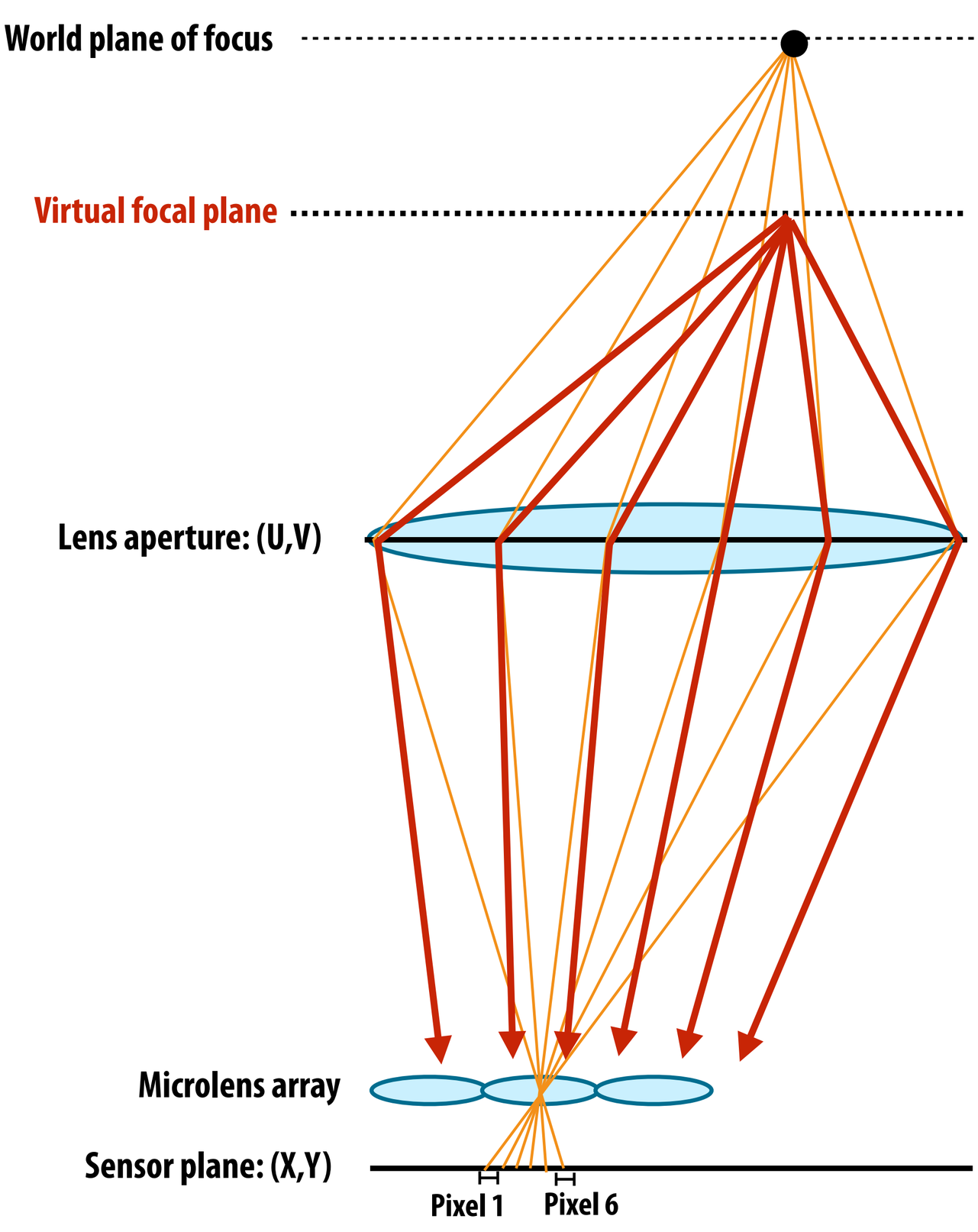


Ray space plot

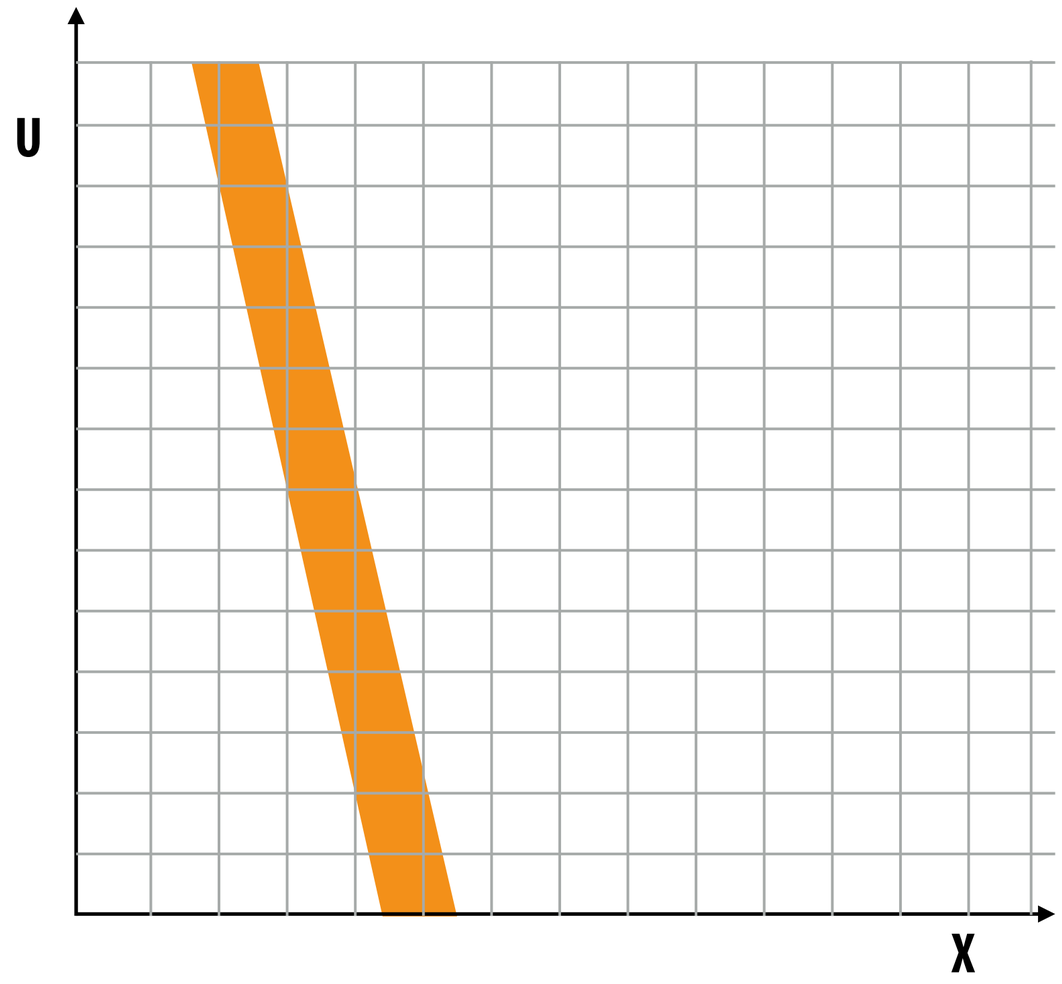


Computing photograph is integral projection (Output image pixel is sum of highlighted light-field sensor pixels)

Computing a photograph at a new focal plane

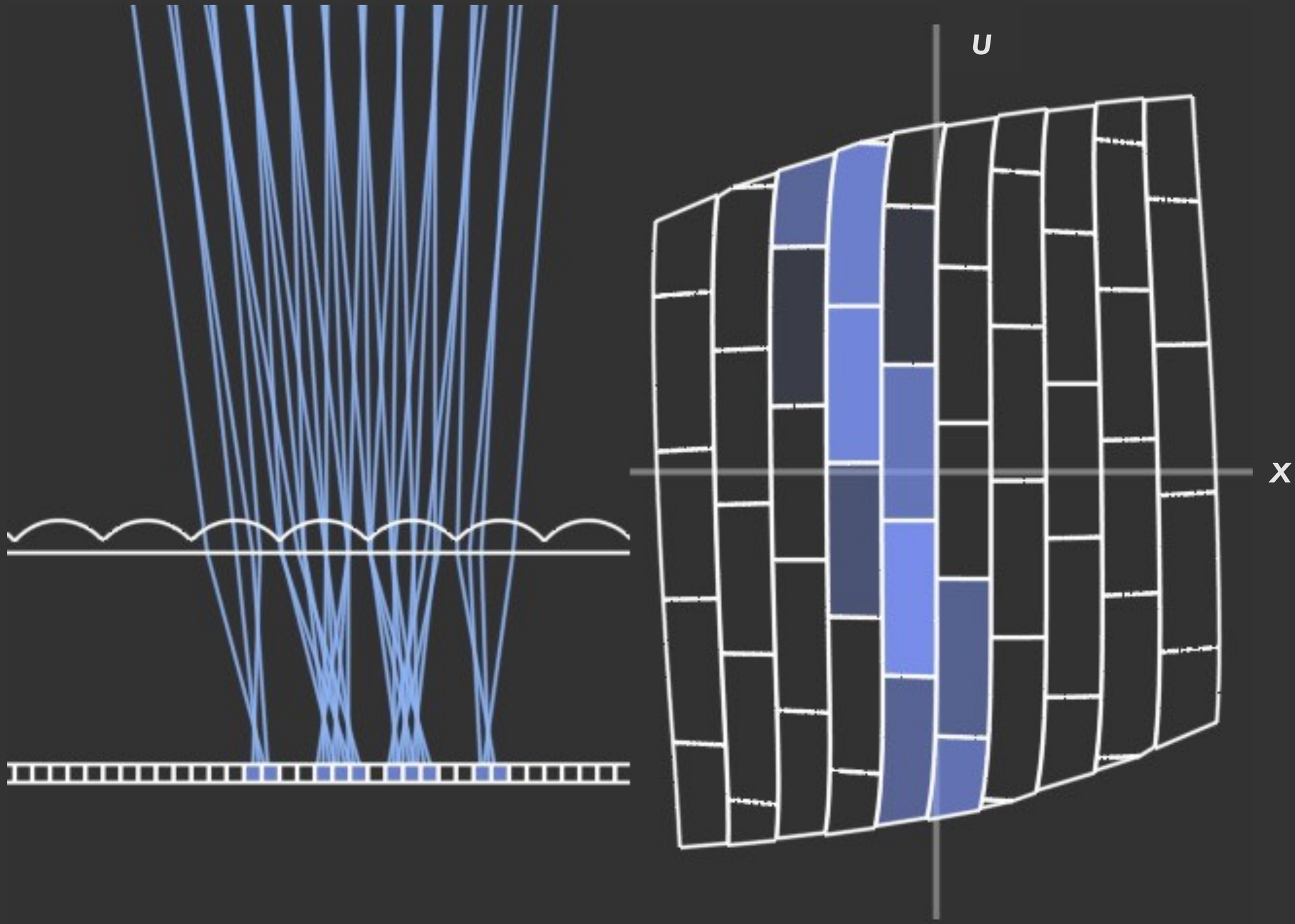


Ray space plot



Computing photograph is integral projection (Output image pixel is integral over highlighted region: resample)

Output Image Pixel is Sum of Many Sensor Pixels

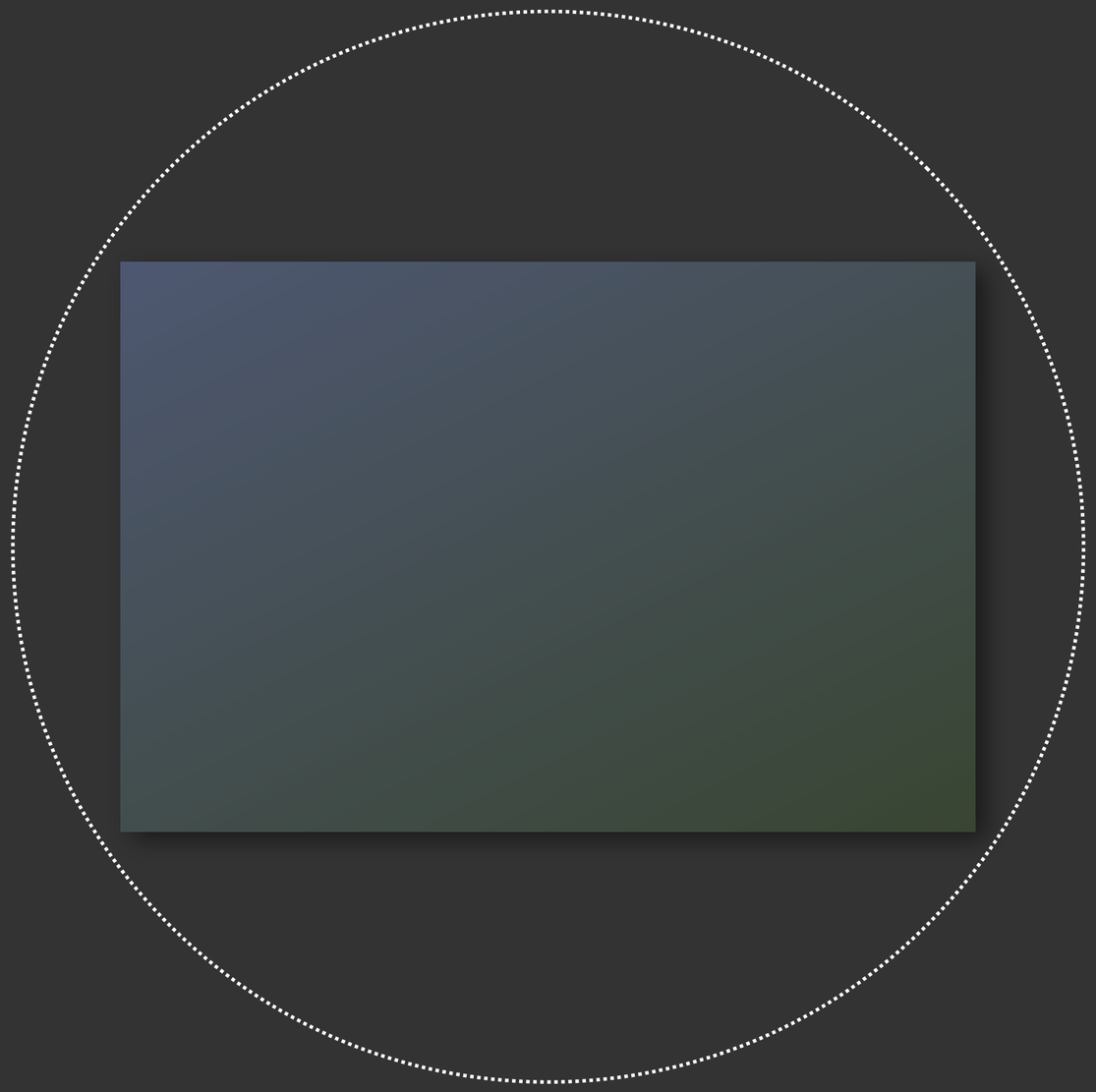


Consumer Light Field Resolutions

Lytro (2012)
10 MegaRay
~10 pixels / microlens

Lytro ILLUM (2014)
40 MegaRay
~14 pixels / microlens

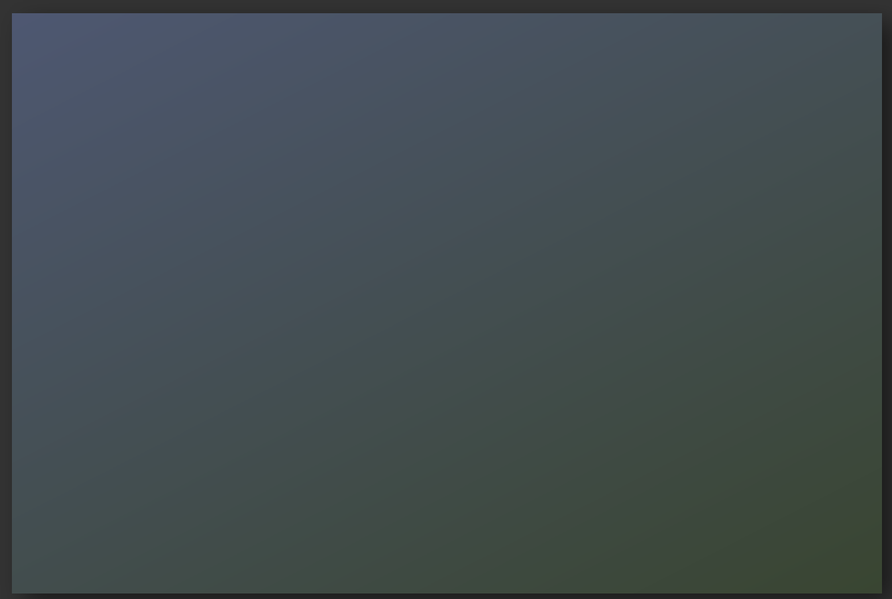
Sensor Industry Has Large Untapped Resolution



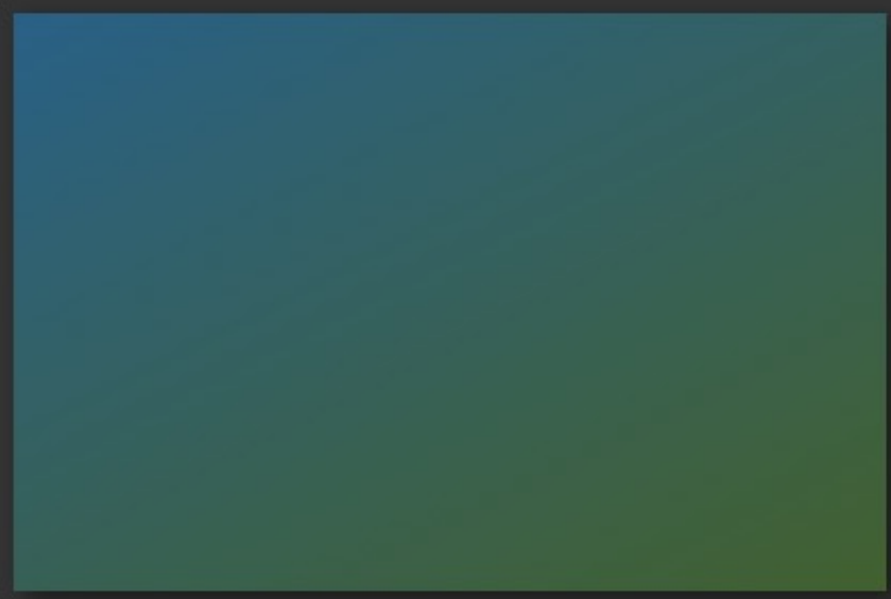
Full-Frame Sensor
36 x 24 mm
Up to 36 MP
4.9 micron pixel

1/3″ Sensor
4.8 x 3.6 mm
Up to 13 MP
1.12 micron pixel

Sensor Industry Has Large Untapped Capability



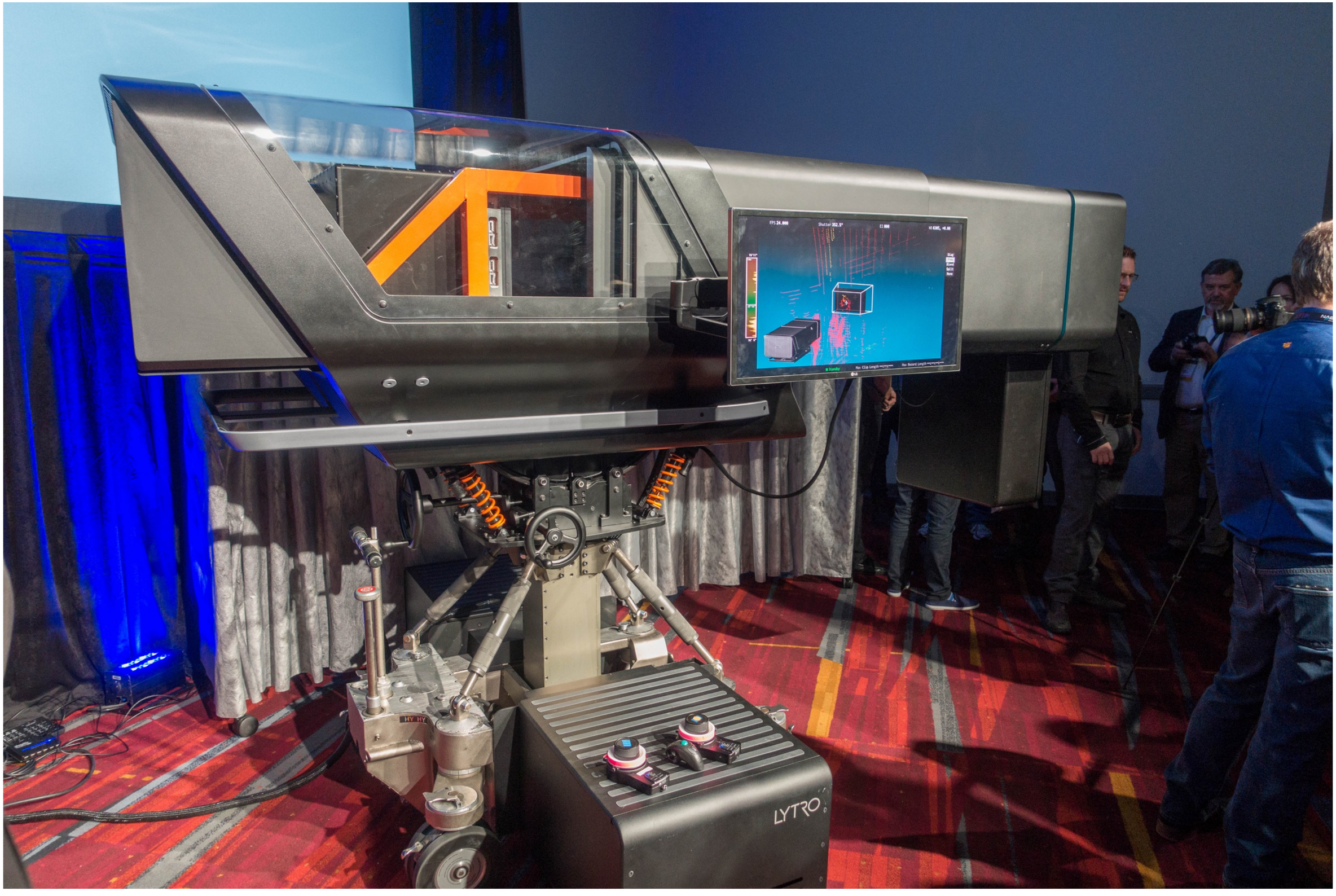
Full-Frame Sensor
36 x 24 mm
Up to 36 MP
4.9 micron pixel



Full-Frame Sensor
36 x 24 mm
688 MP
1.12 micron pixel

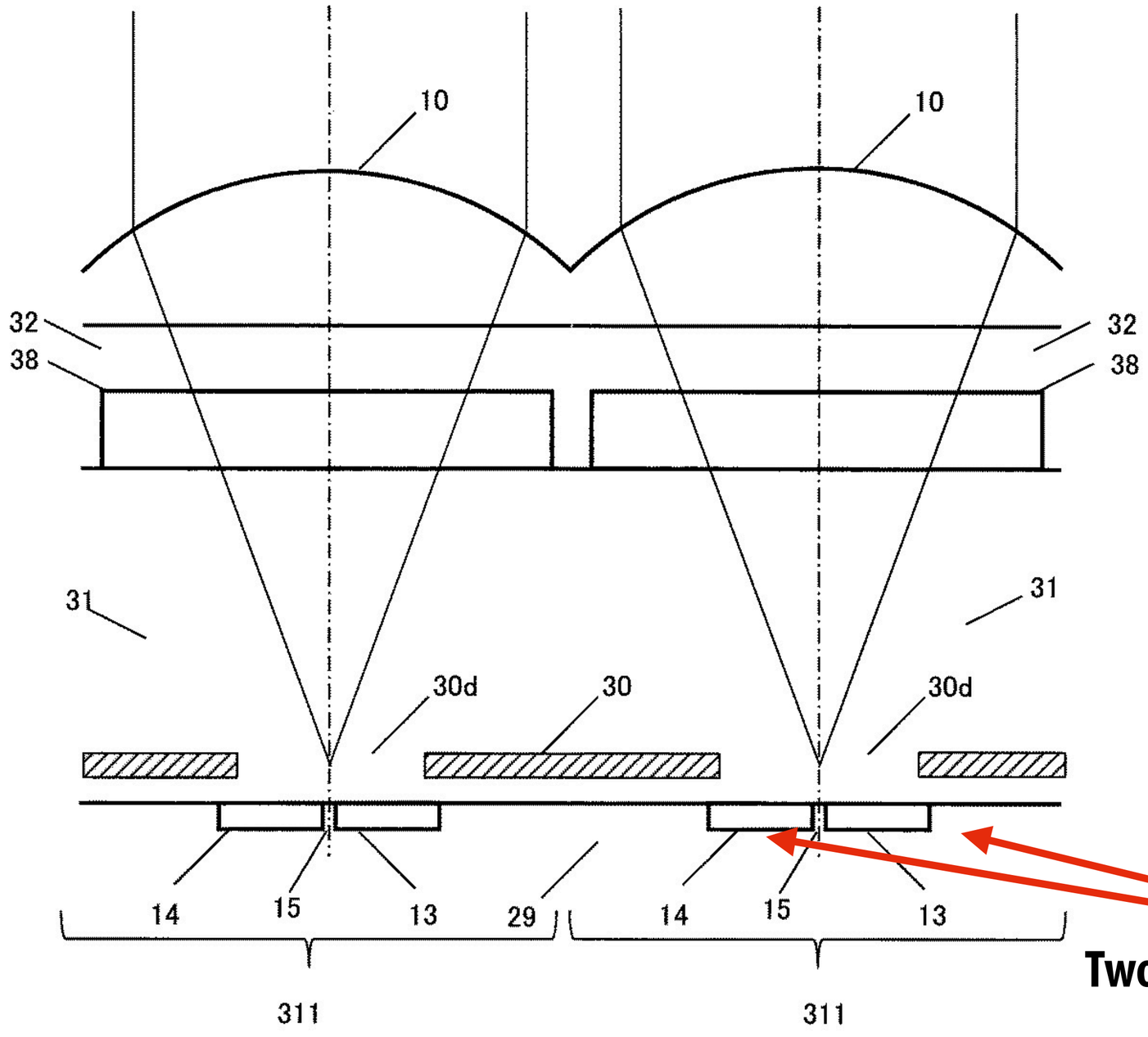
Lytro Cinema

755 Mpixel camera



Split pixel sensor

Used in cell-phone cameras today to assist with autofocus



Two pixels under microlens

Image credit: Nikon

Virtual reality displays

Virtual reality (VR) vs augmented reality (AR)

VR = virtual reality

User is completely immersed in virtual world (sees only light emitted by display

AR = augmented reality

Display is an overlay that augments user's normal view of the real world (e.g., terminator)

Image credit: Terminator 2 (naturally)

VR headsets

Oculus Rift

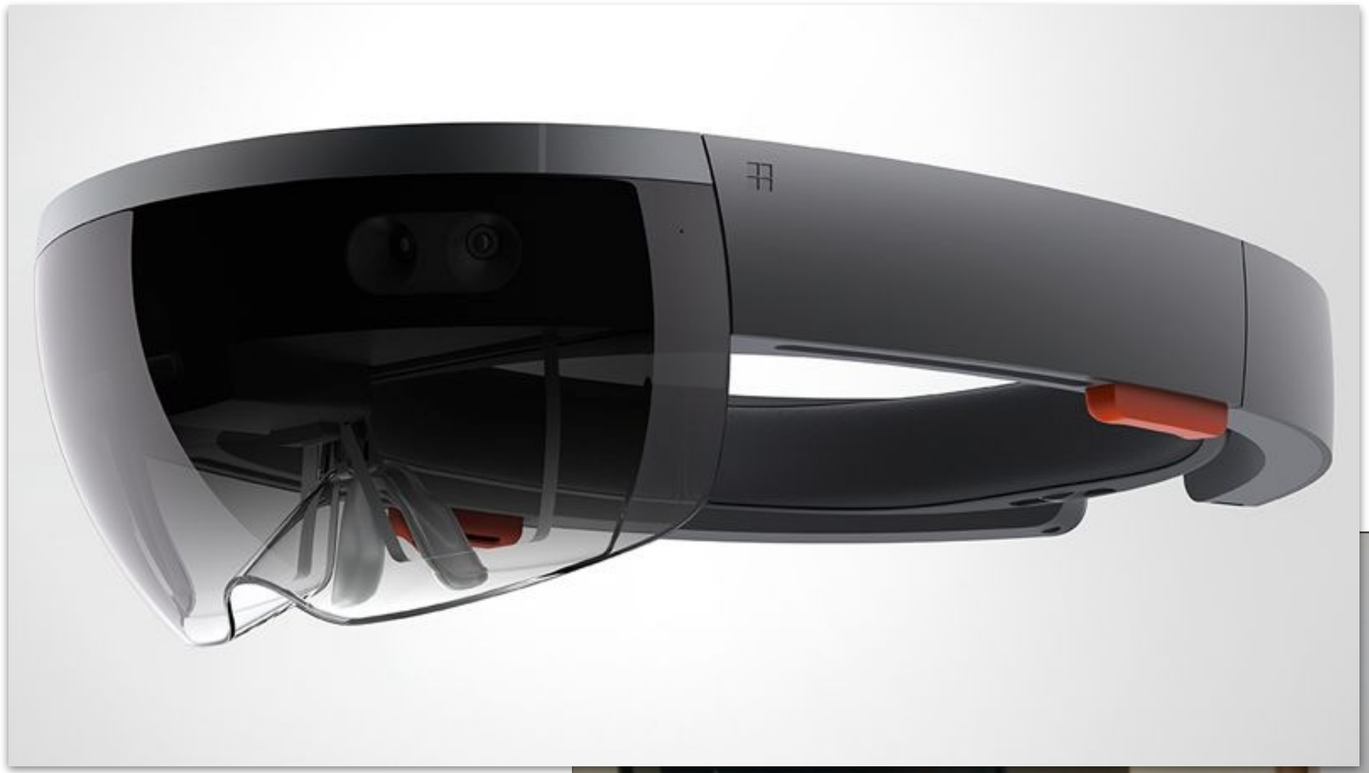
HTV Vive

Sony Morpheus

Google Daydream

Google Cardboard

AR headset: Microsoft Hololens



VR gaming

Bullet Train Demo (Epic)

VR video

VR video

VR teleconference / video chat

http://vrchat.com/

Oculus Rift DK2

Oculus Rift DK2

Oculus Rift DK2 headset

Image credit: ifixit.com

Oculus Rift DK2 headset

Image credit: ifixit.com

Oculus Rift DK2 display

5.7″ 1920 x 1080 OLED display
75 Hz refresh rate

(Same display as Galaxy Note 3)

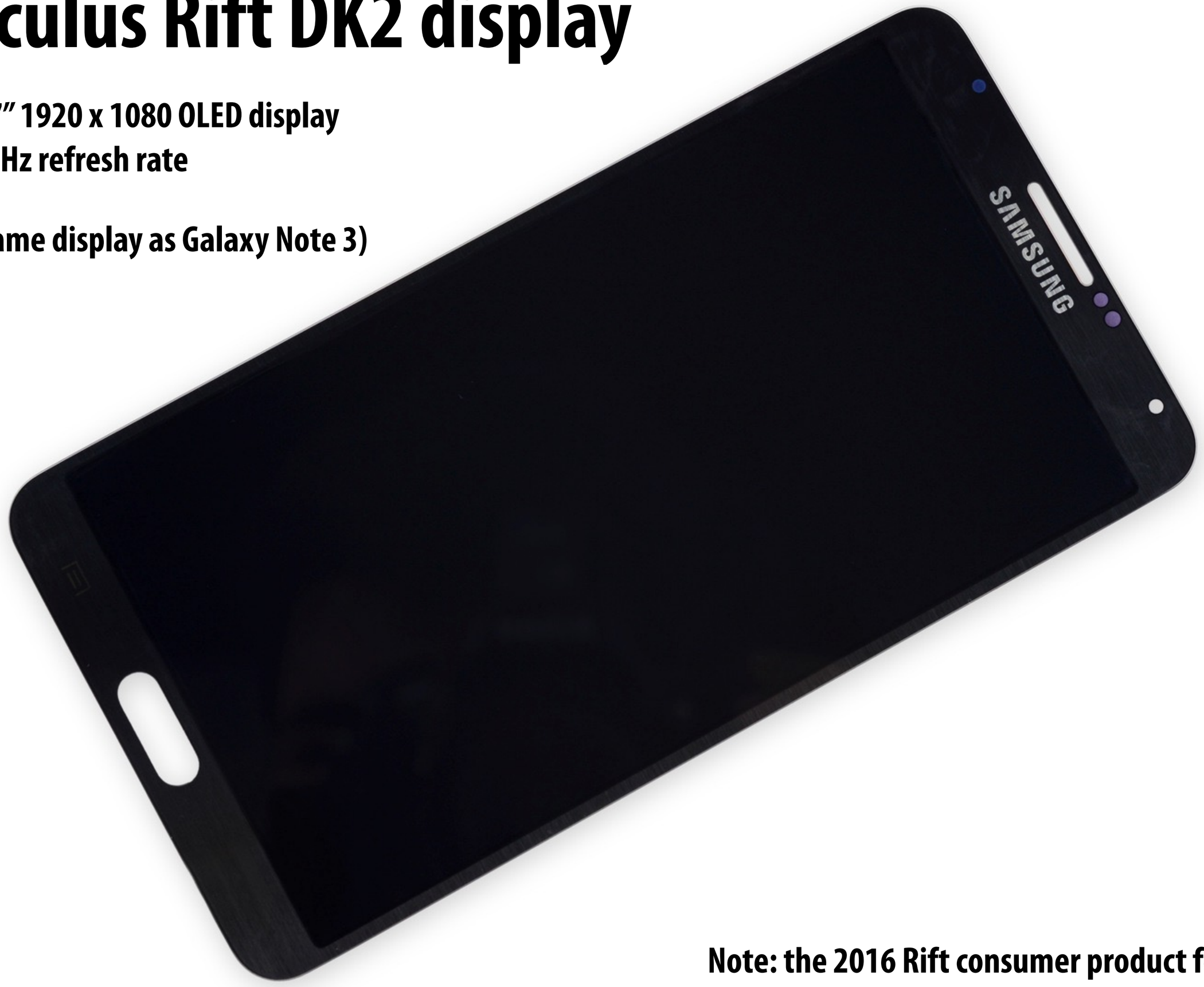
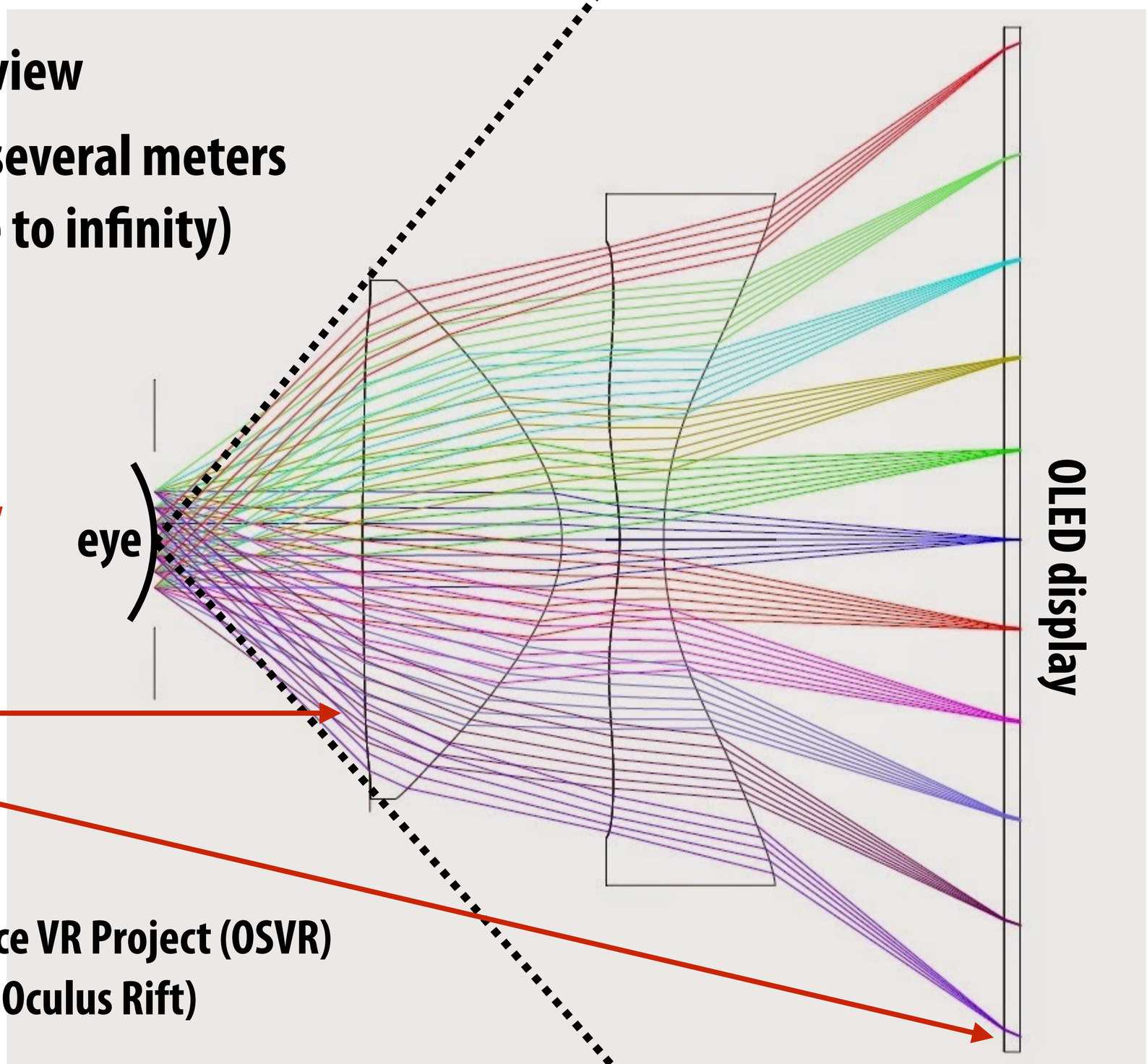


Image credit: ifixit.com

Note: the 2016 Rift consumer product features two 1080×1200 displays at 90Hz.

Role of optics in headset

1. Create wide field of view
2. Place focal plane at several meters away from eye (close to infinity)



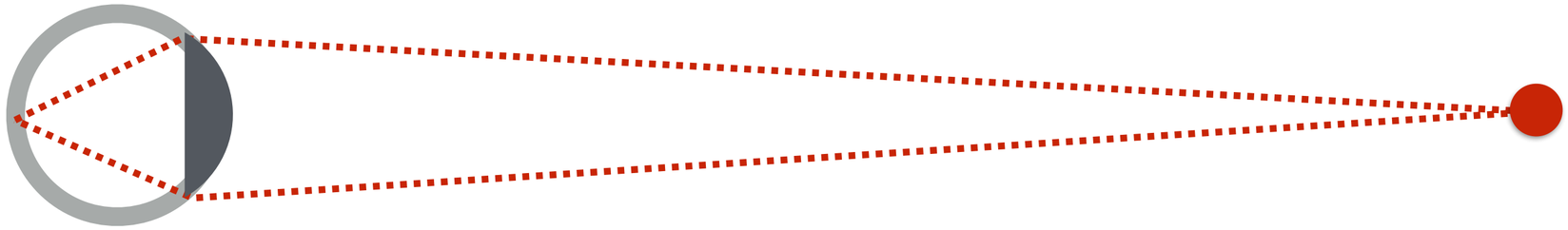
Note: parallel lines reaching eye converge to a single point on display (eye accommodates to plane near infinity)

Lens diagram from Open Source VR Project (OSVR)
(Not the lens system from the Oculus Rift)
http://www.osvr.org/

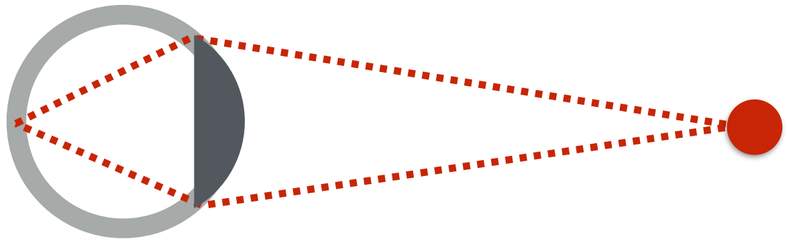
Accommodation and vergence

Accommodation: changing the optical power of the eye to focus at different distances

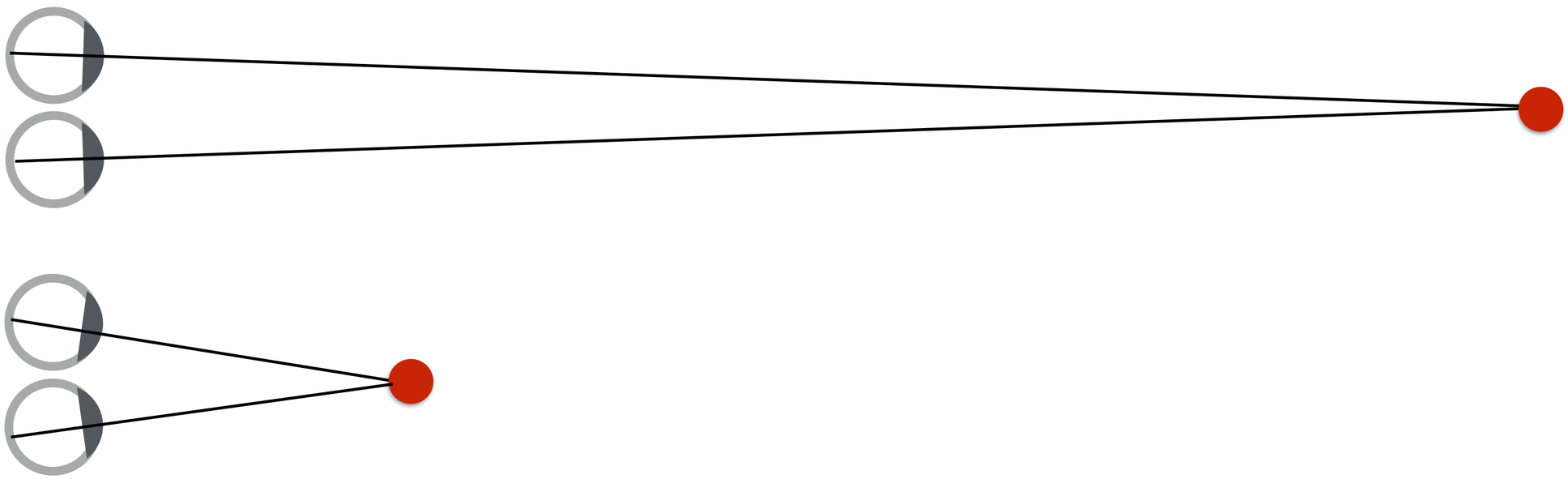
Eye accommodated at far distance



Eye accommodated at near distance

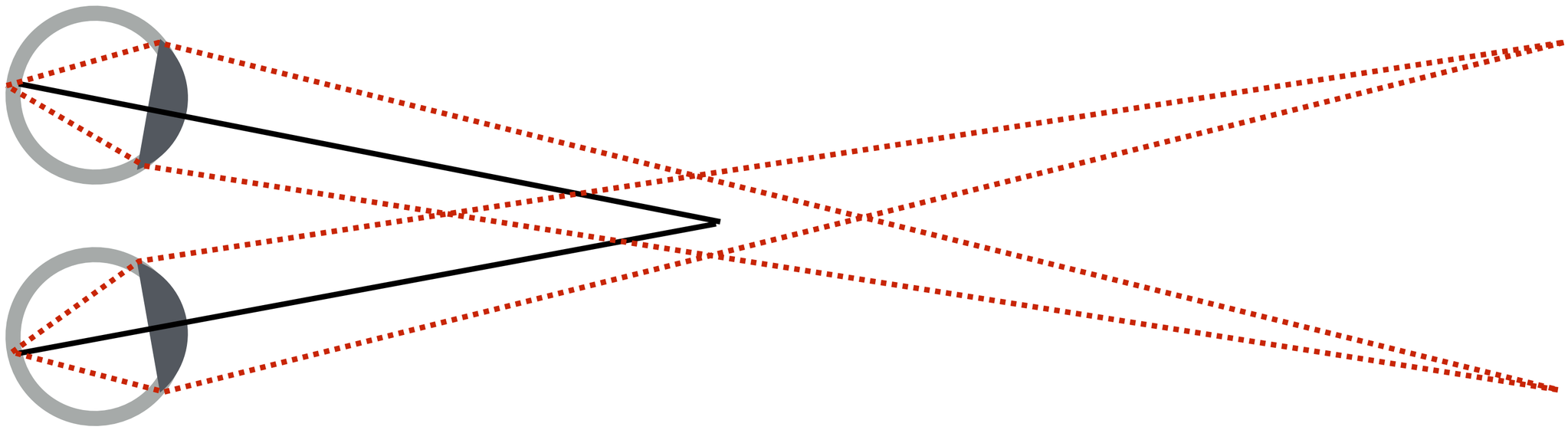


Vergence: rotation of eye to ensure projection of object falls in center of retina



Accommodation - vergence conflict

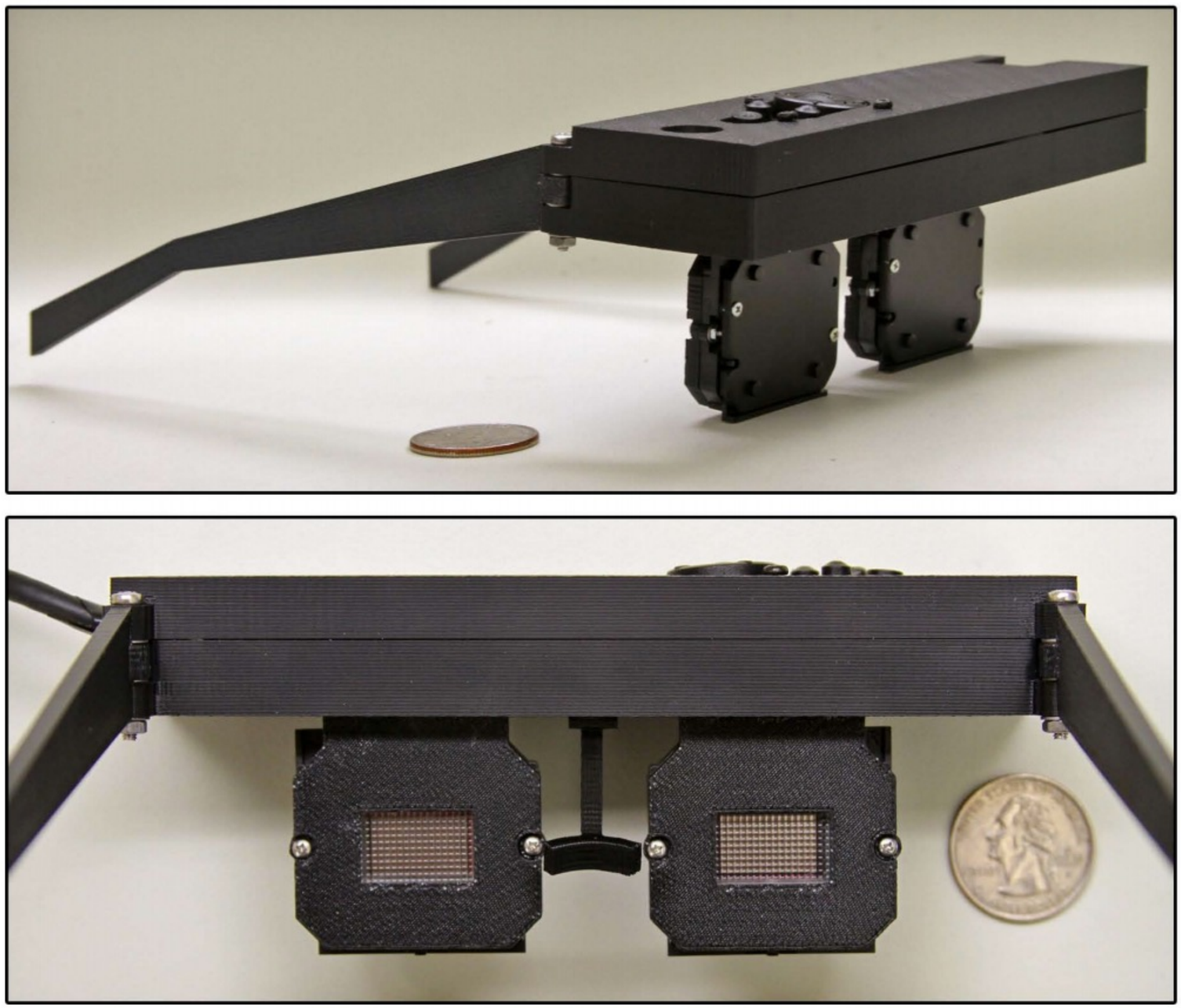
- Given design of current VR displays, consider what happens when objects are up-close to eye in virtual scene
  - Eyes must remain accommodated to near infinity (otherwise image on screen won't be in focus)
  - But eyes must converge in attempt to fuse stereoscopic images of object up close
  - Brain receives conflicting depth clues… (discomfort, fatigue, nausea)

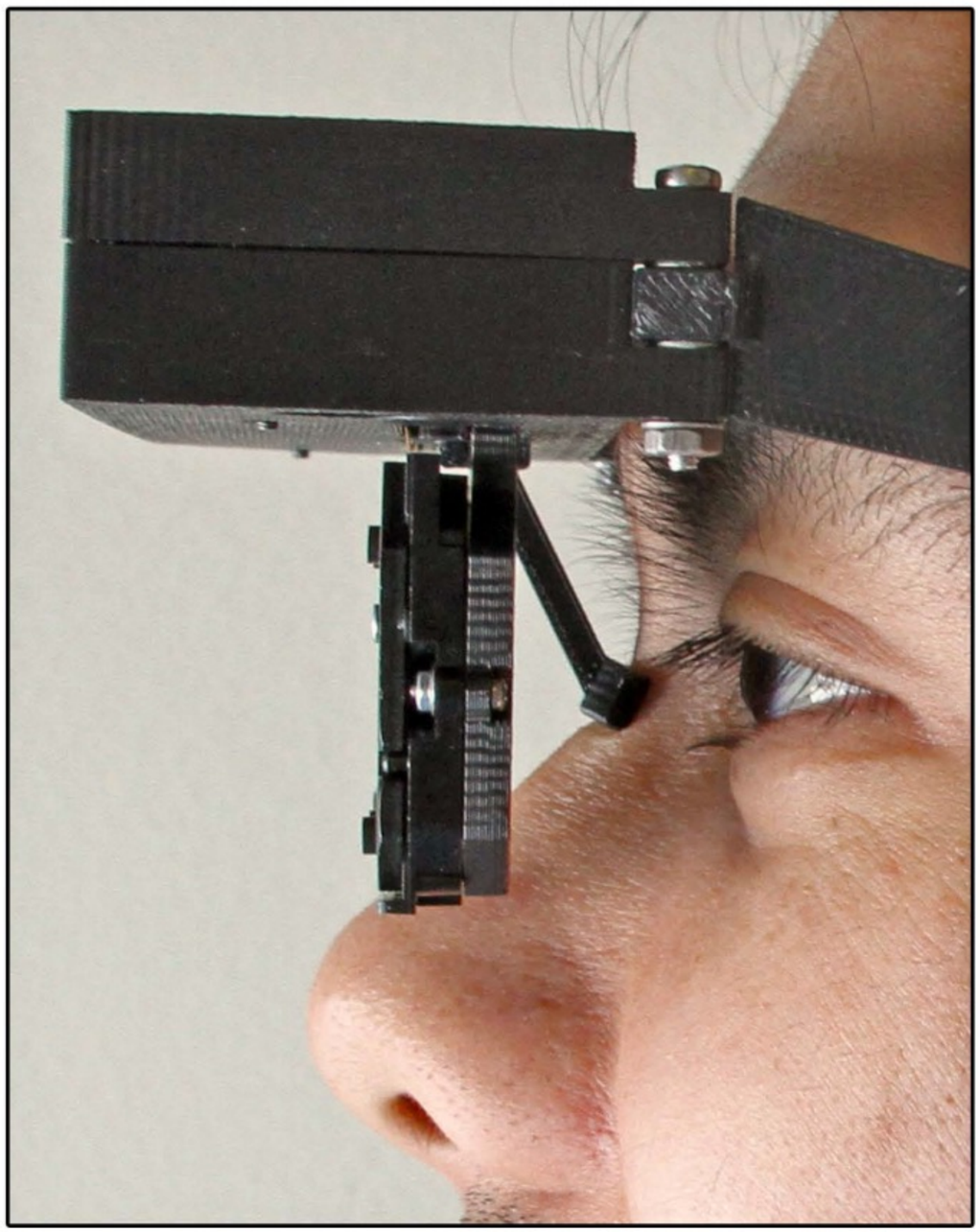


This problem stems from nature of display design. If you could just make a display that emits the light field that would be produced by a virtual scene, then you could avoid the accommodation - vergence conflict…

Aside: near-eye light field displays

Recreate light field in front of eye





Oculus DK2 IR camera and IR LEDs

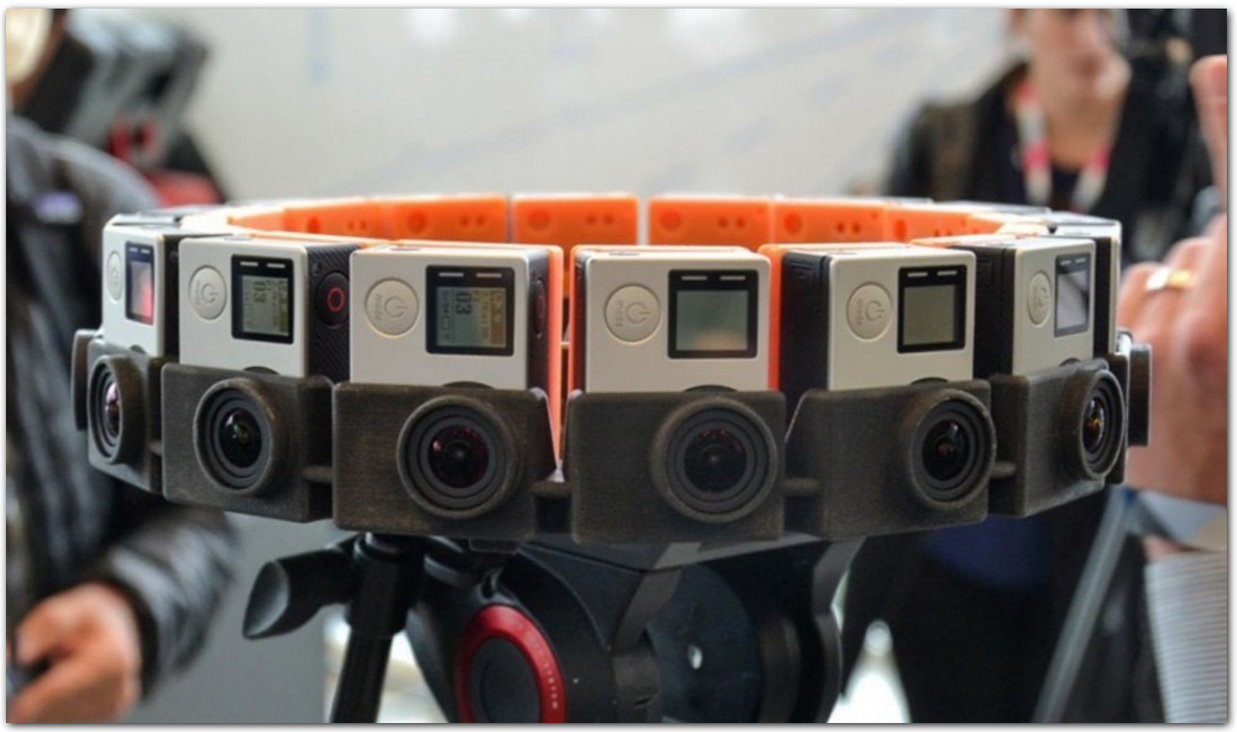
60Hz IR Camera

Headset contains:

40 IR LEDs
Gyro + accelerometer (1000Hz)

Image credit: ifixit.com

Interest in acquiring VR content

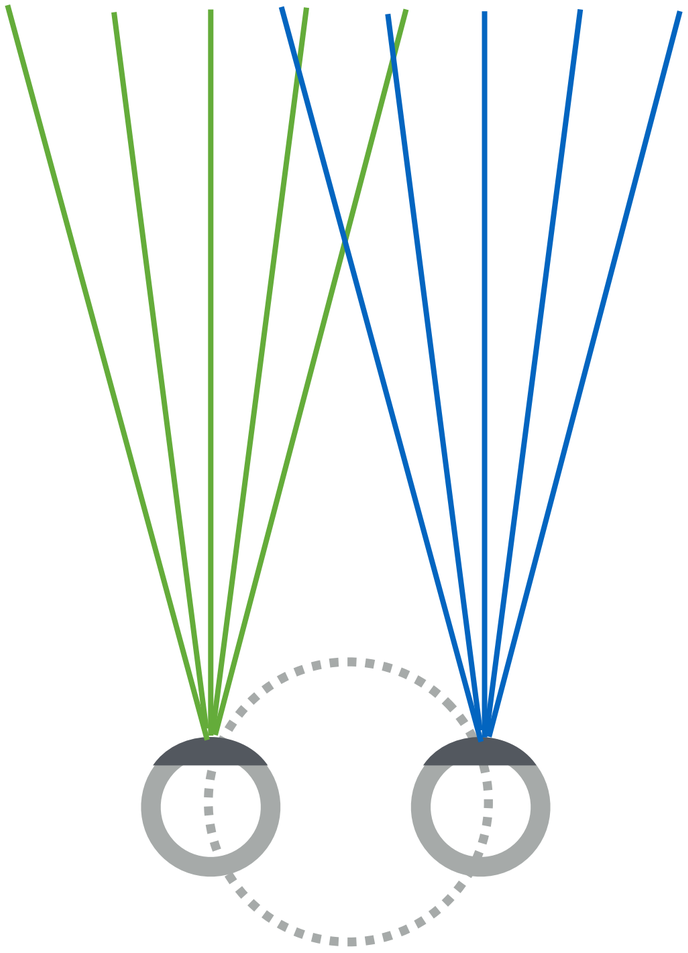


Google's Jump VR video:
16 4K GoPro cameras

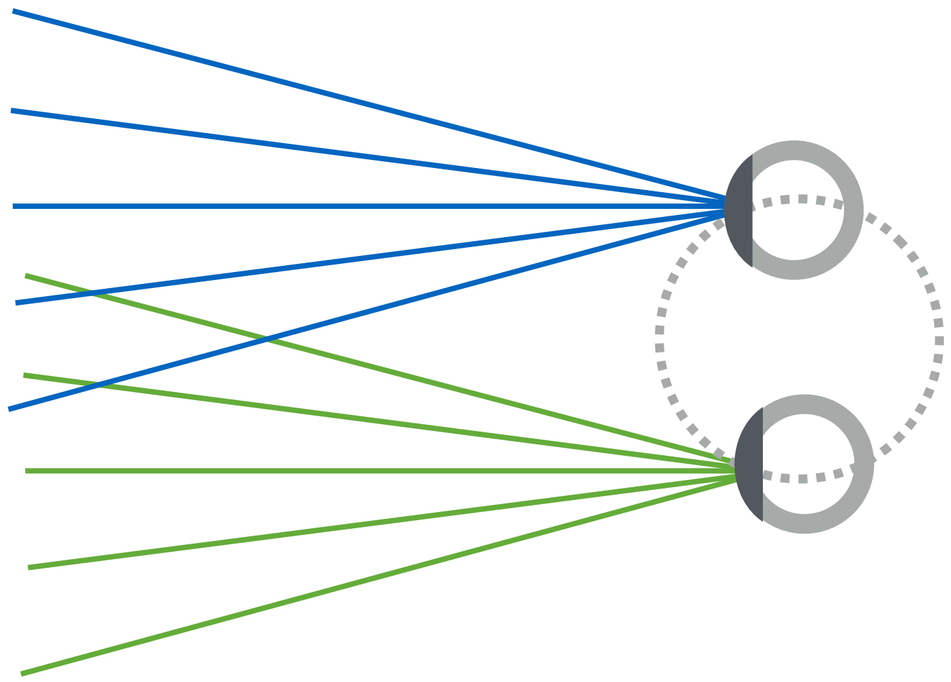
Facebook
Surround 360

Lytro Immerge
(leveraging light field camera technology to acquire VR content)

Stereo, 360-degree viewing



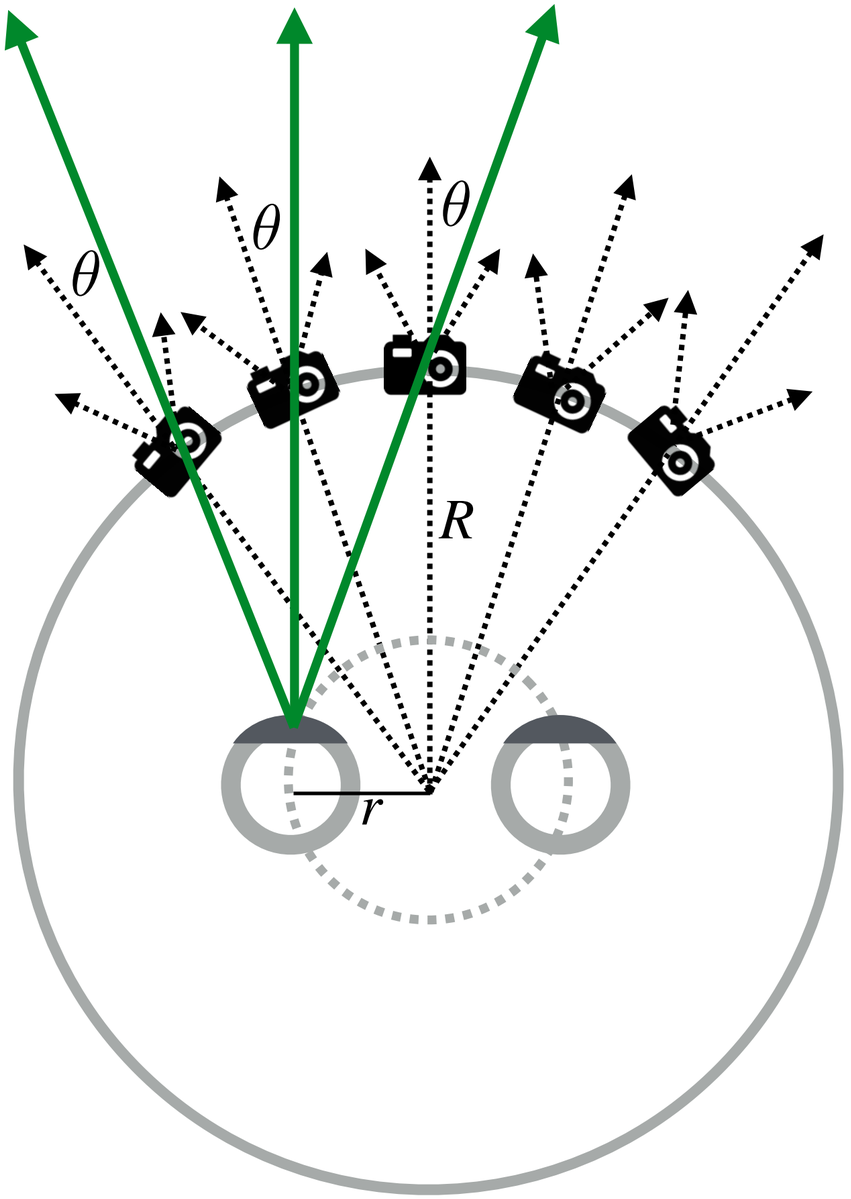
Stereo, 360-degree viewing



Measuring light arriving at left eye

Left eye

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

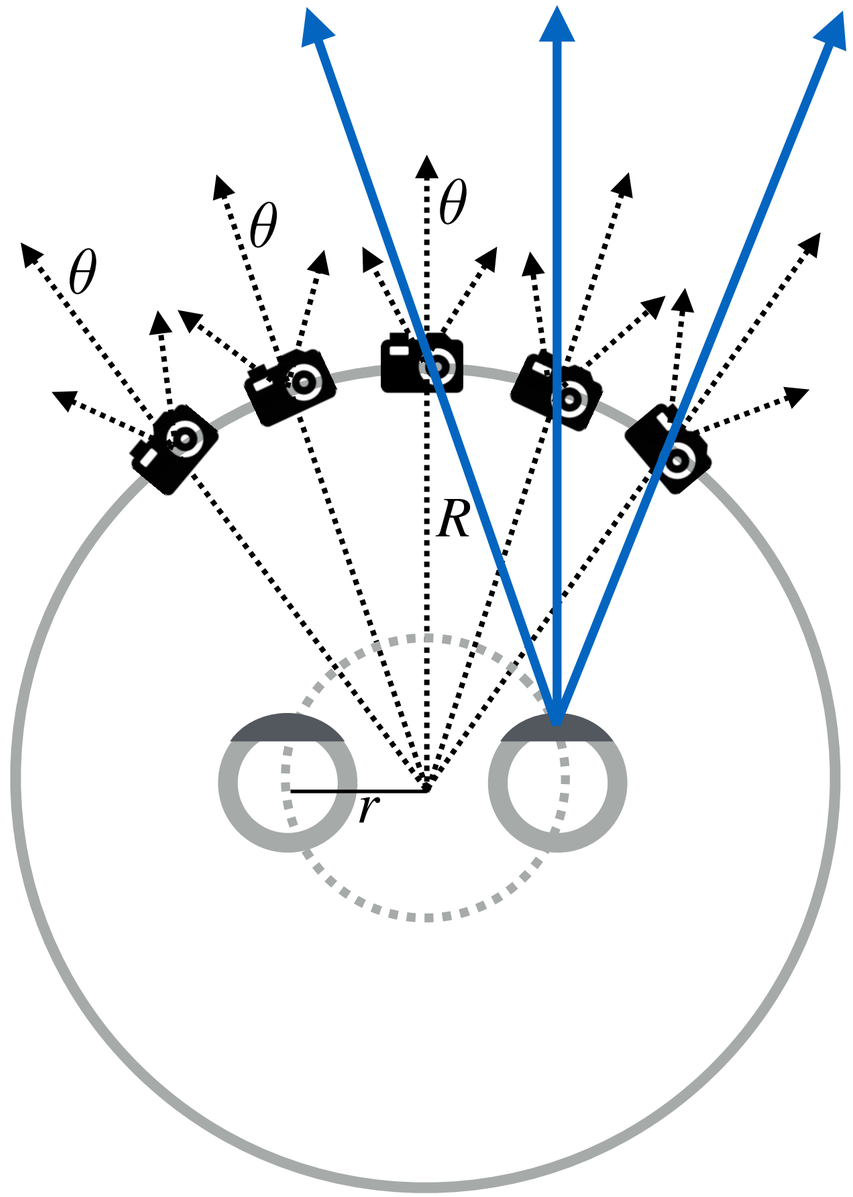


[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Measuring light arriving at right eye

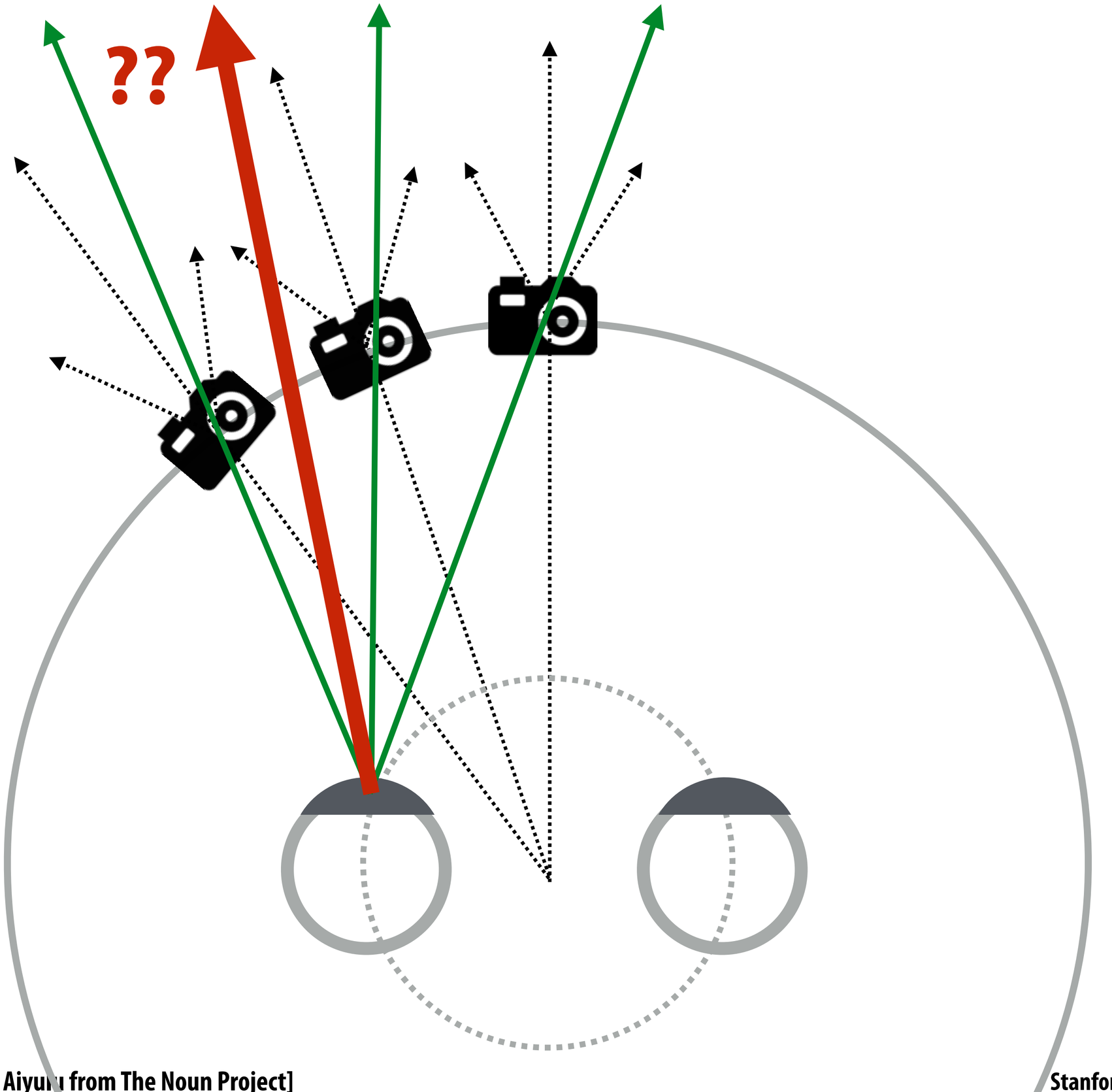
Right eye

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



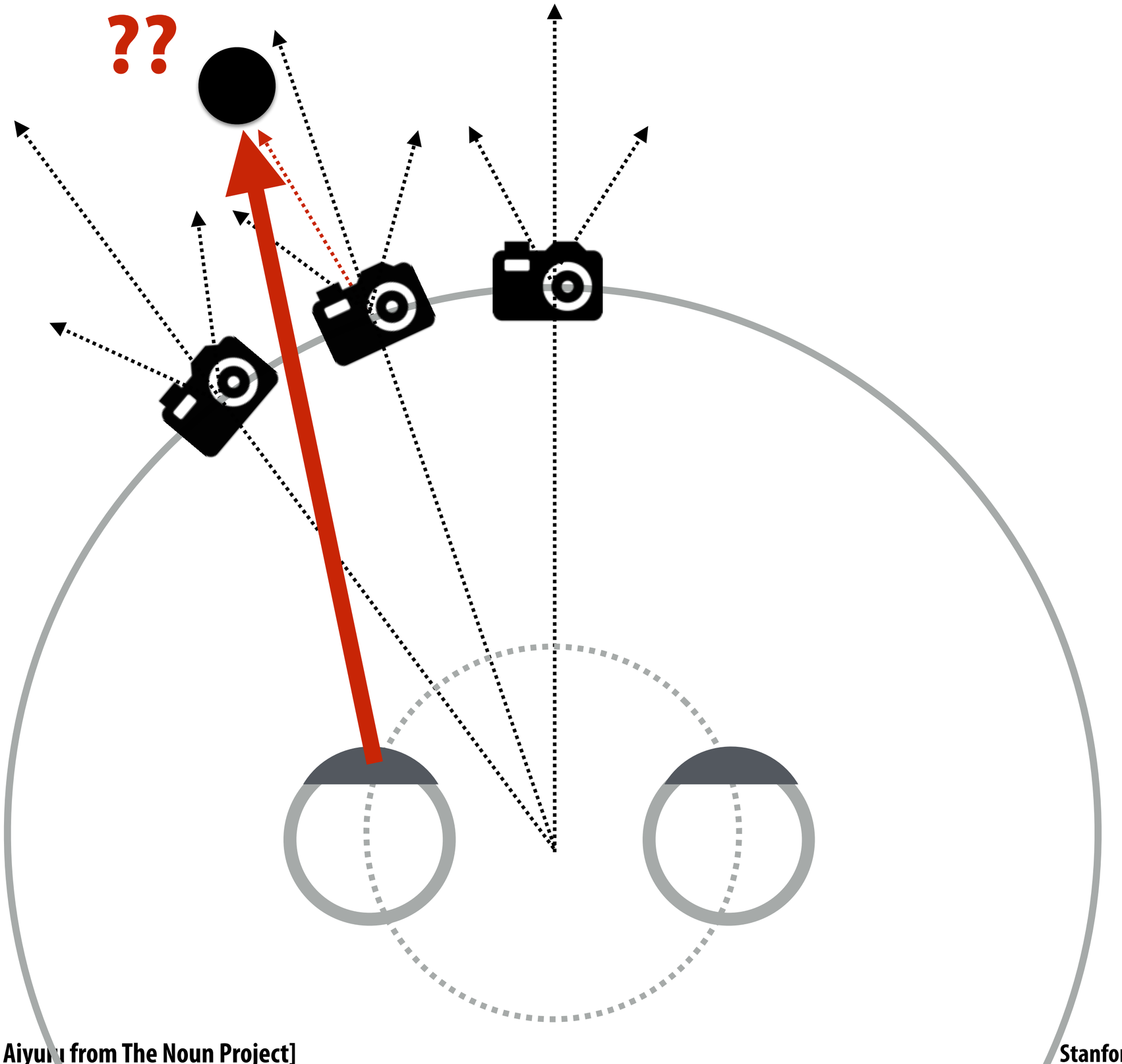
[Credit: Camera icon by Venkatesh Aiyulu from The Noun Project]

Interpolation of novel views depends on scene depth



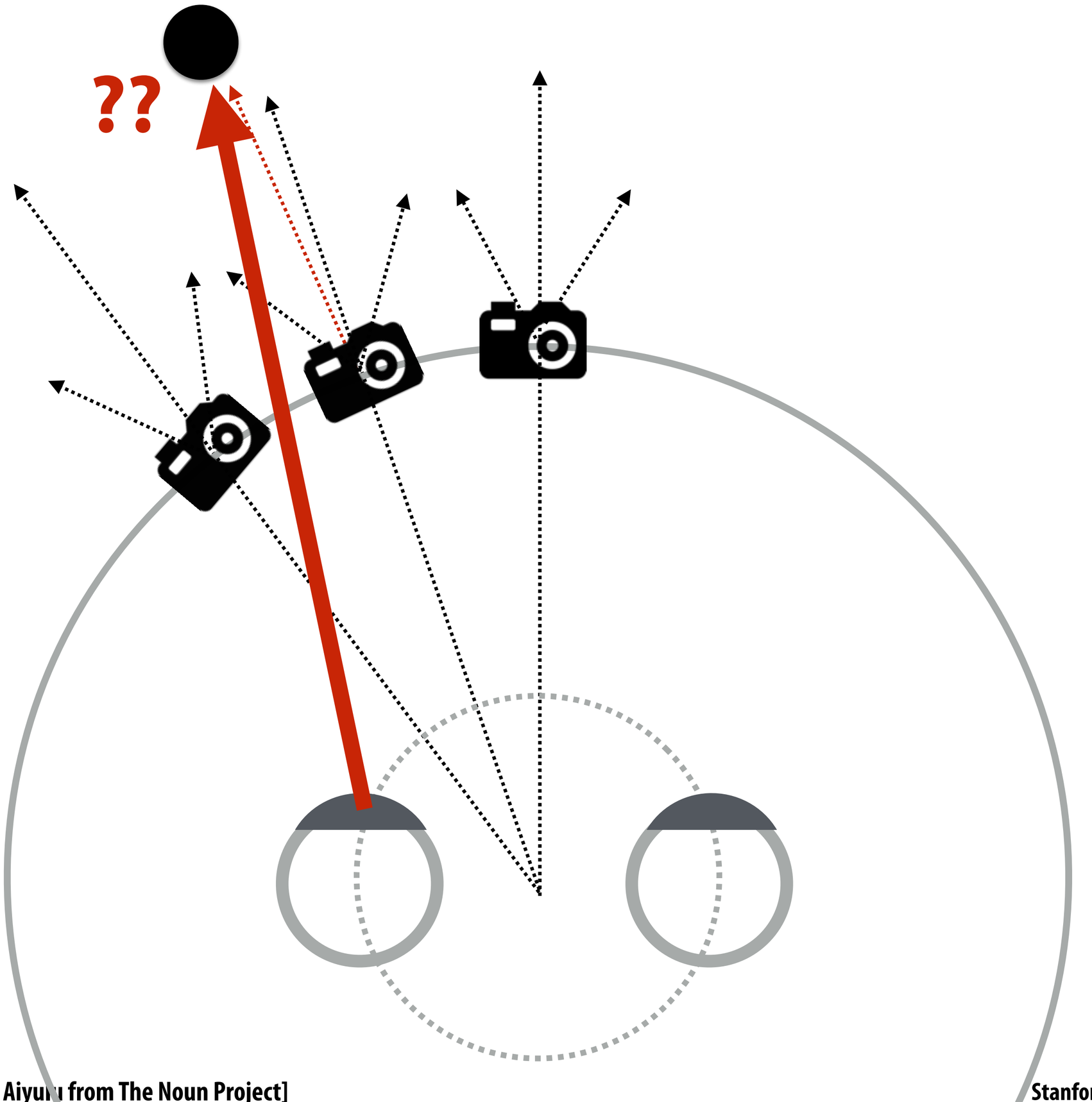
[Credit: Camera icon by Venkatesh Aiyuru from The Noun Project]

Interpolation to novel views depends on scene depth



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

Interpolation of novel views depends on scene depth



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

Computing depth of scene point from two images

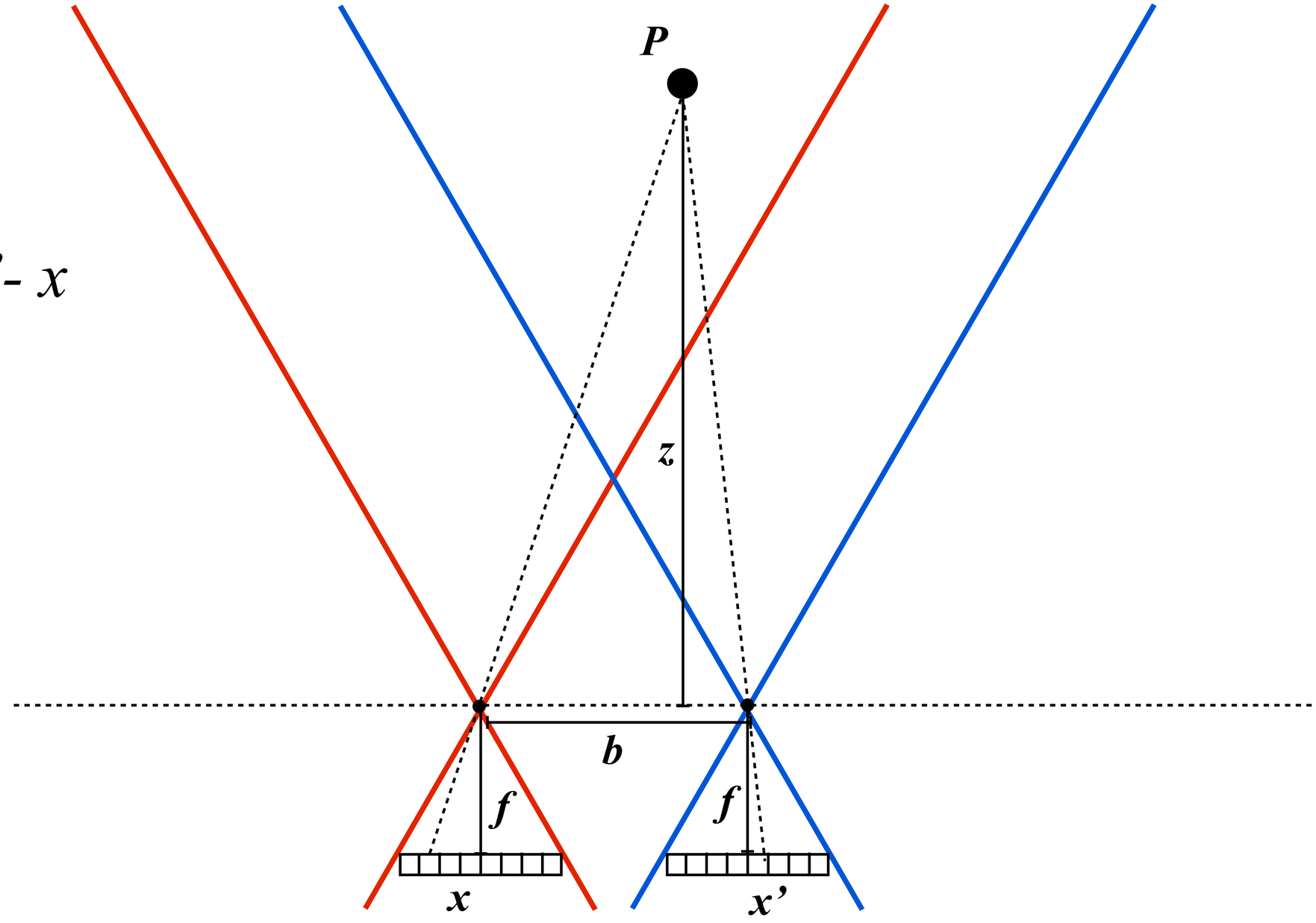
Binocular stereo 3D reconstruction of point *P*: depth from disparity

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

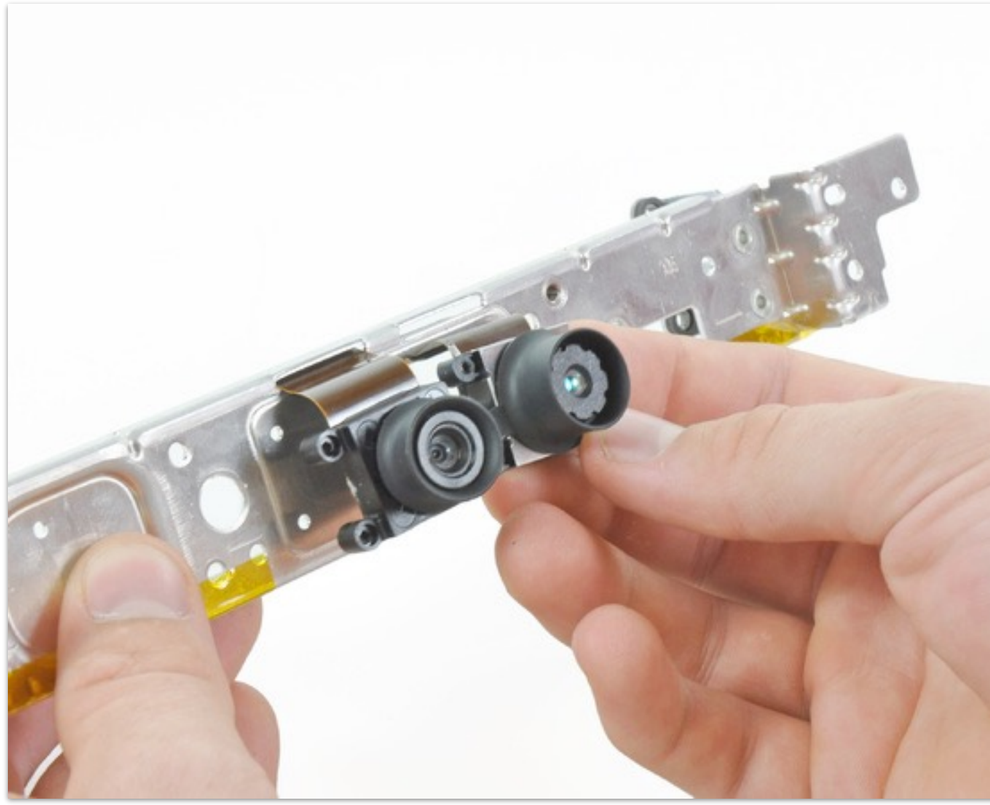


Image credit: iFixIt

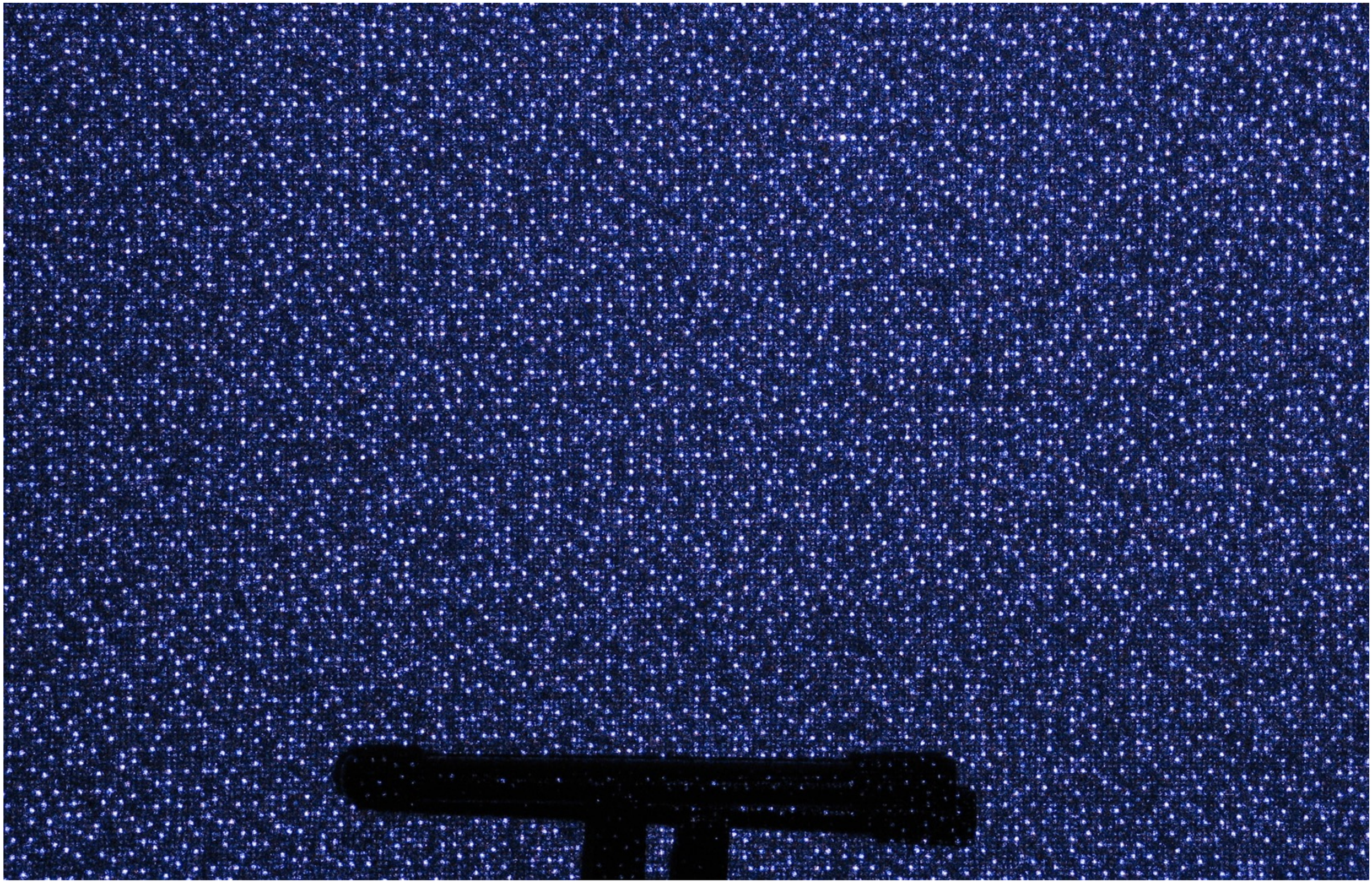
Illuminant
(Infrared Laser + diffuser)

RGB CMOS Sensor
640x480 (w/ Bayer mosaic)

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

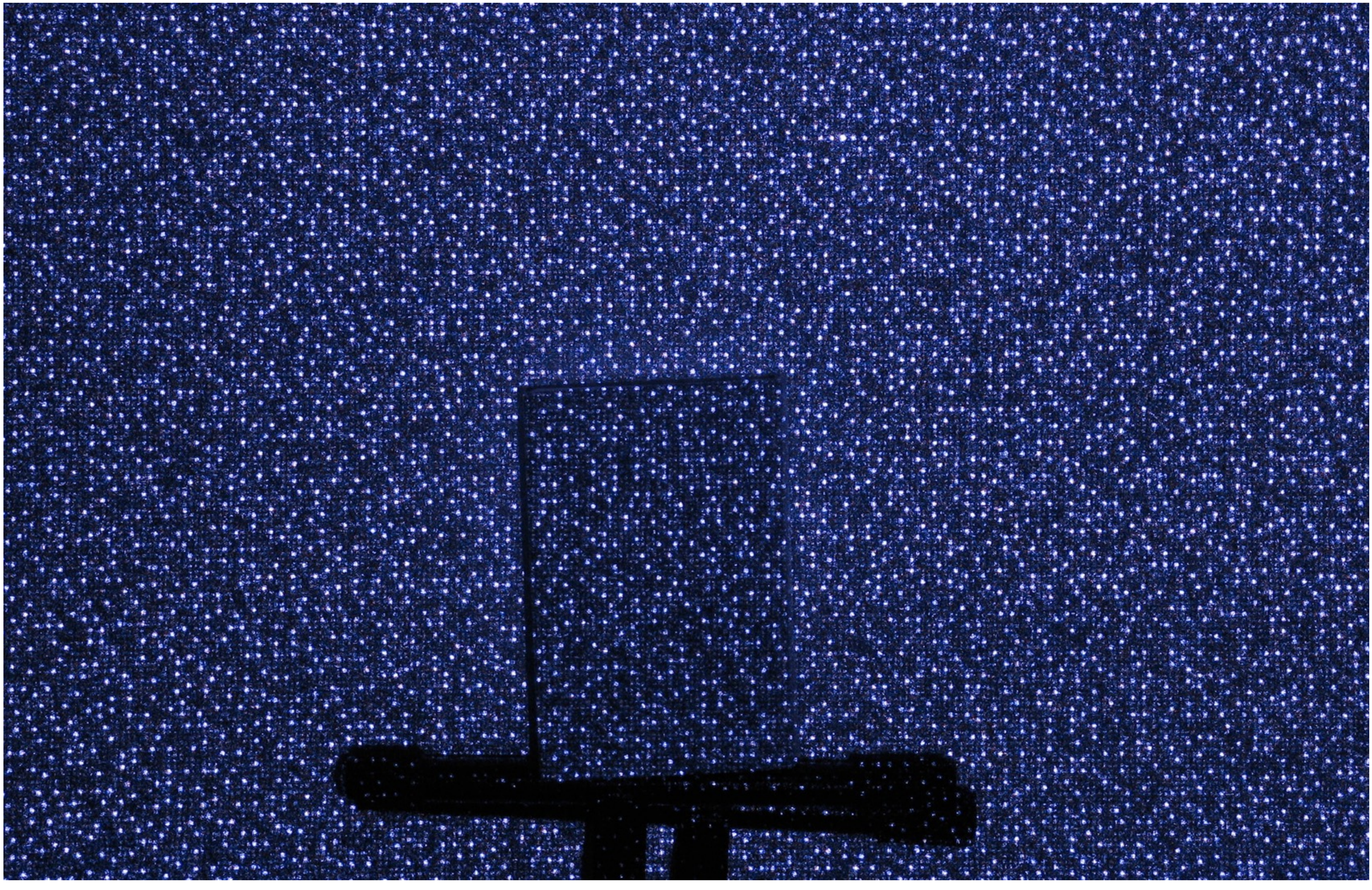
**** Kinect returns 640x480 disparity image, suspect sensor is configured for 2x2 pixel binning down to 640x512, then crop**

Infrared image of Kinect illuminant output



Credit: www.futurepicture.org

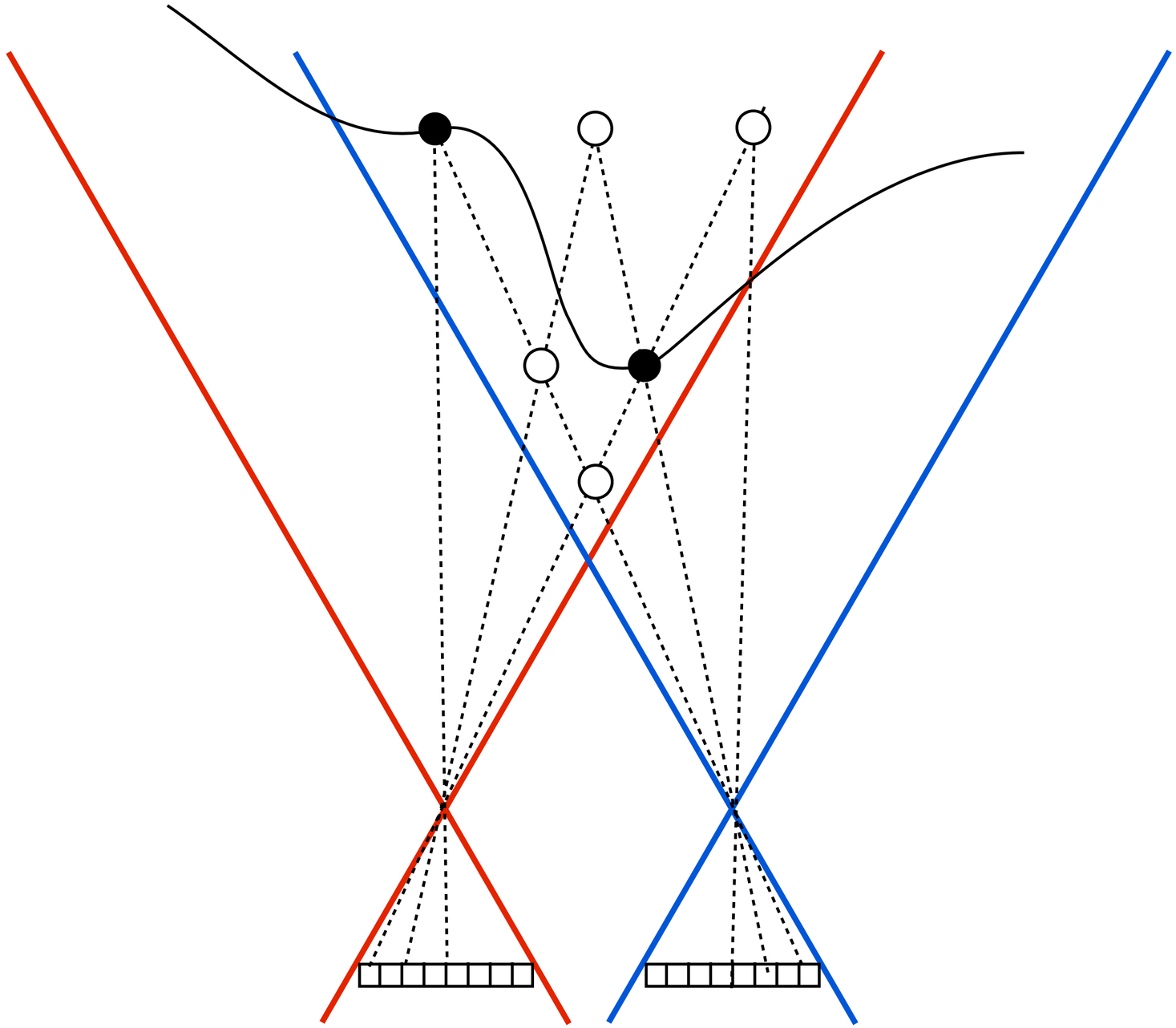
Infrared image of Kinect illuminant output



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*
- Jump 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+ frame sequence 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 paper)

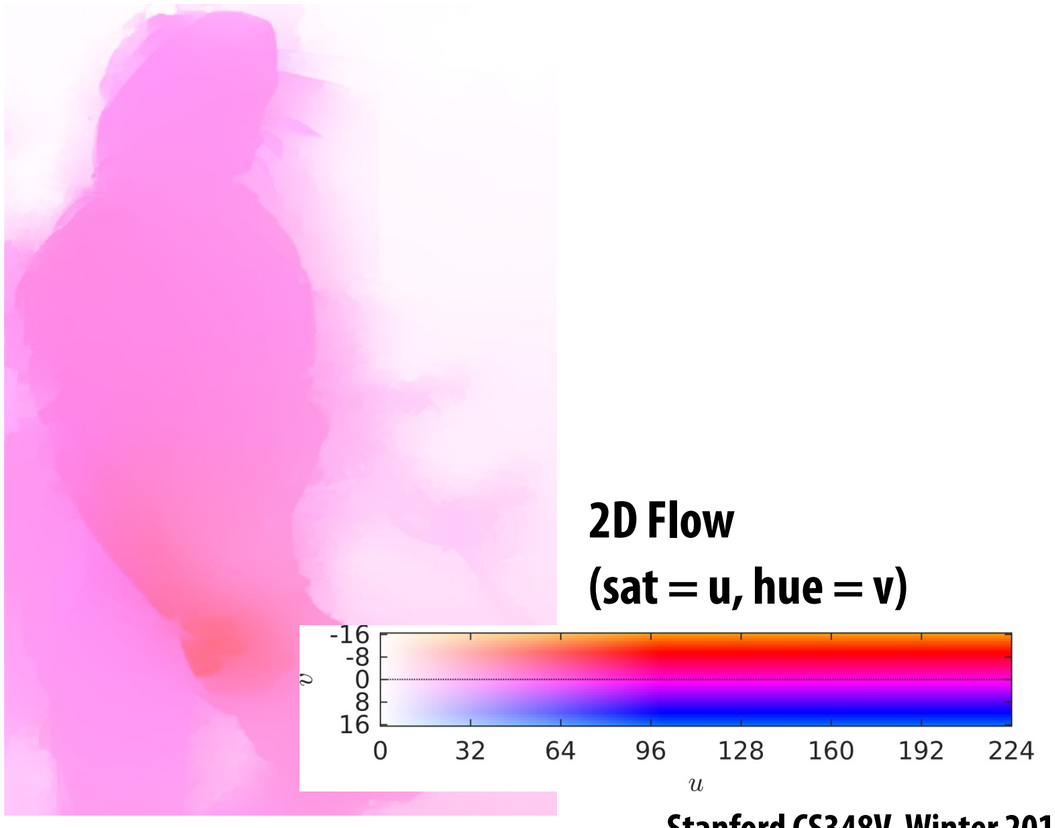
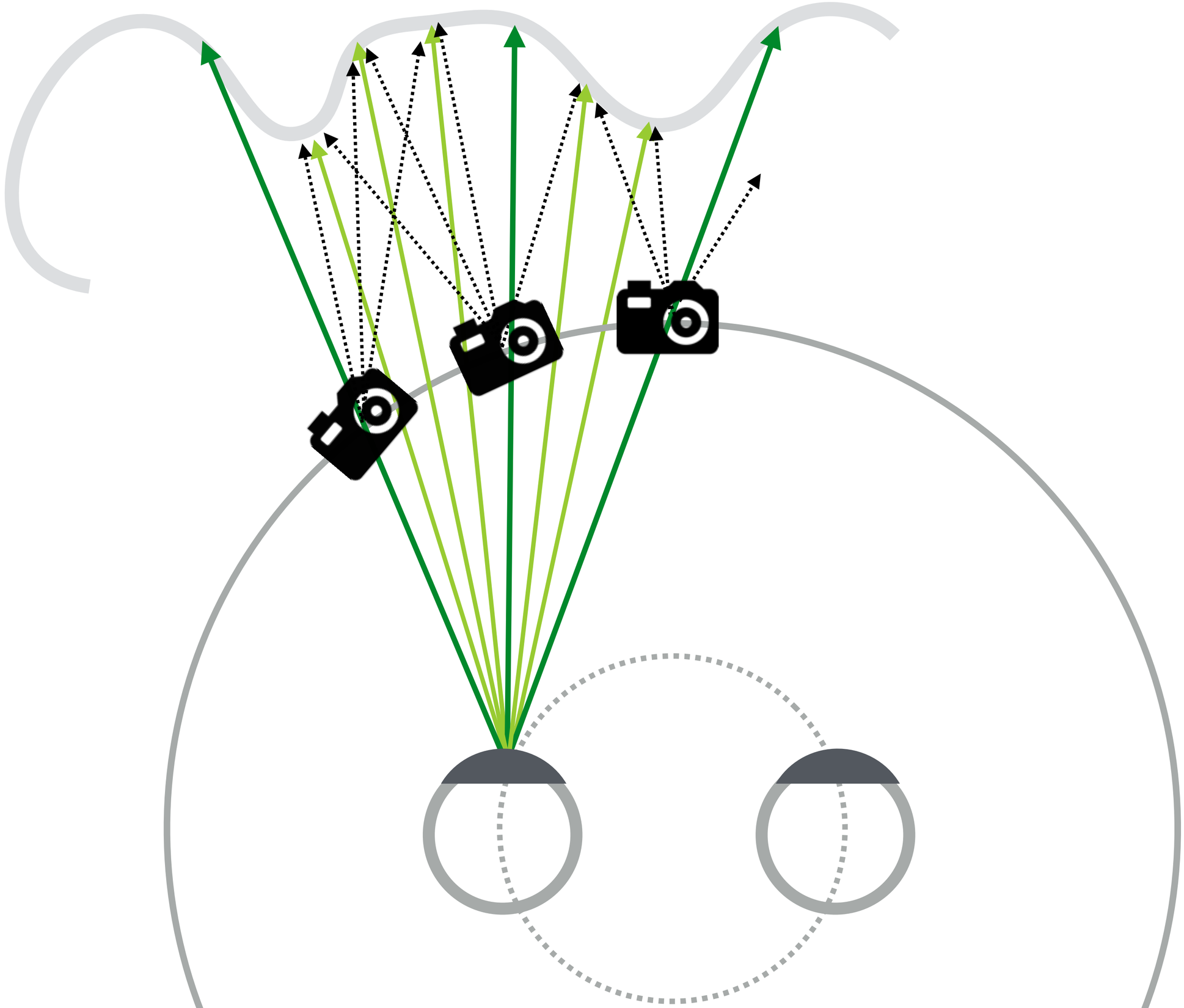


Image credit: Andersen et al. 2016

Left eye: with interpolated rays



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

Omnidirectional stereo (ODS) representation

- Unique panorama of size W x H for left and right eye
- Can be saved, editing 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

Bonus material:

challenges of VR display

Name of the game, part 1: low latency

- The goal of a VR graphics system is to achieve "presence", tricking the brain into thinking what it is seeing is real

- Achieving presence requires an exceptionally low-latency system
  - What you see must change when you move your head!
  - End-to-end latency: time from moving your head to the time new photons hit your eyes
    - Measure user's head movement
    - Update scene/camera position
    - Render new image
    - Transfer image to headset, then to transfer to display in headset
    - Actually emit light from display (photons hit user's eyes)
- Latency goal of VR: 10-25 ms
  - Requires exceptionally low-latency head tracking
  - Requires exceptionally low-latency rendering and display

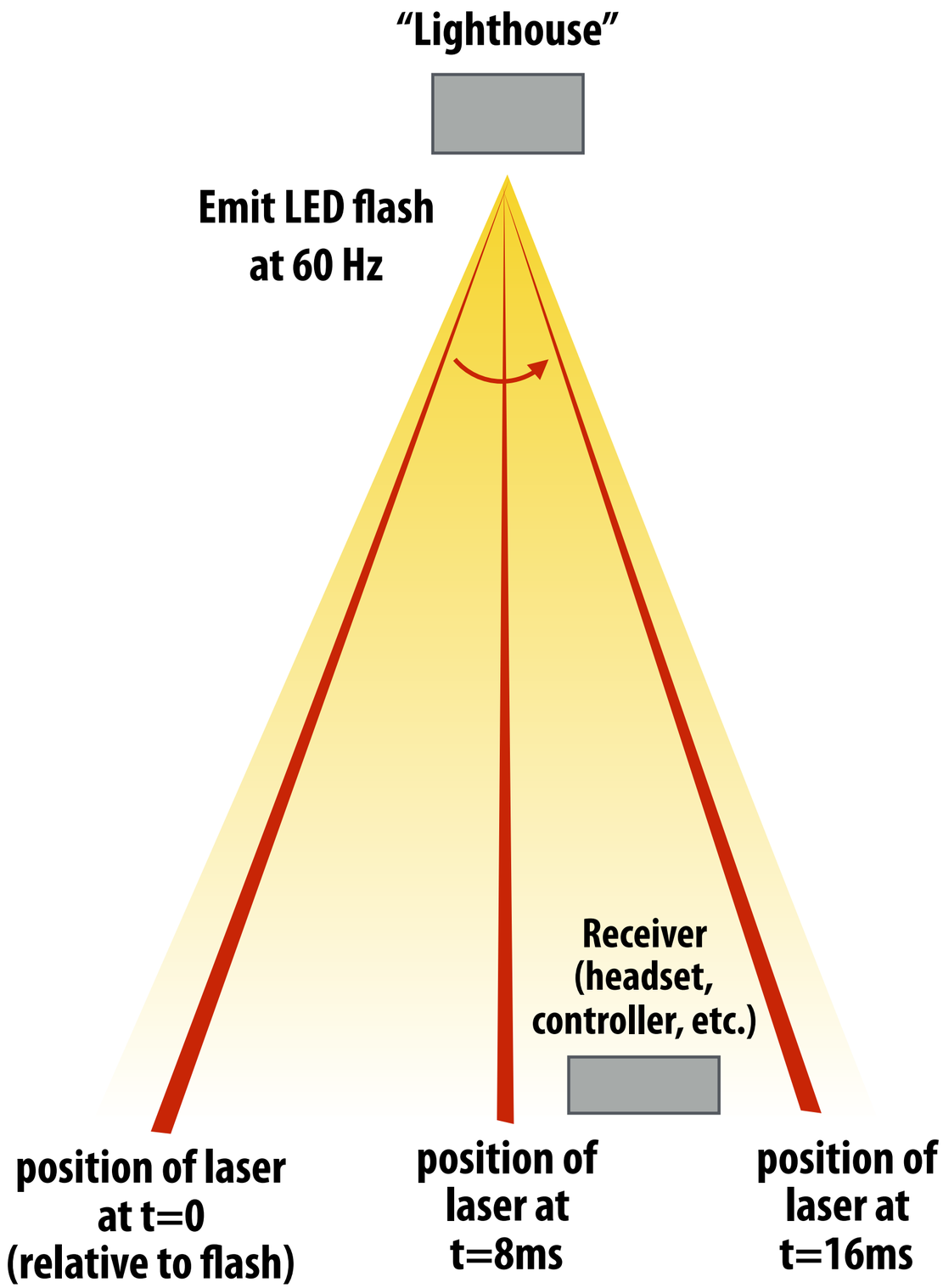
Thought experiment: effect of latency

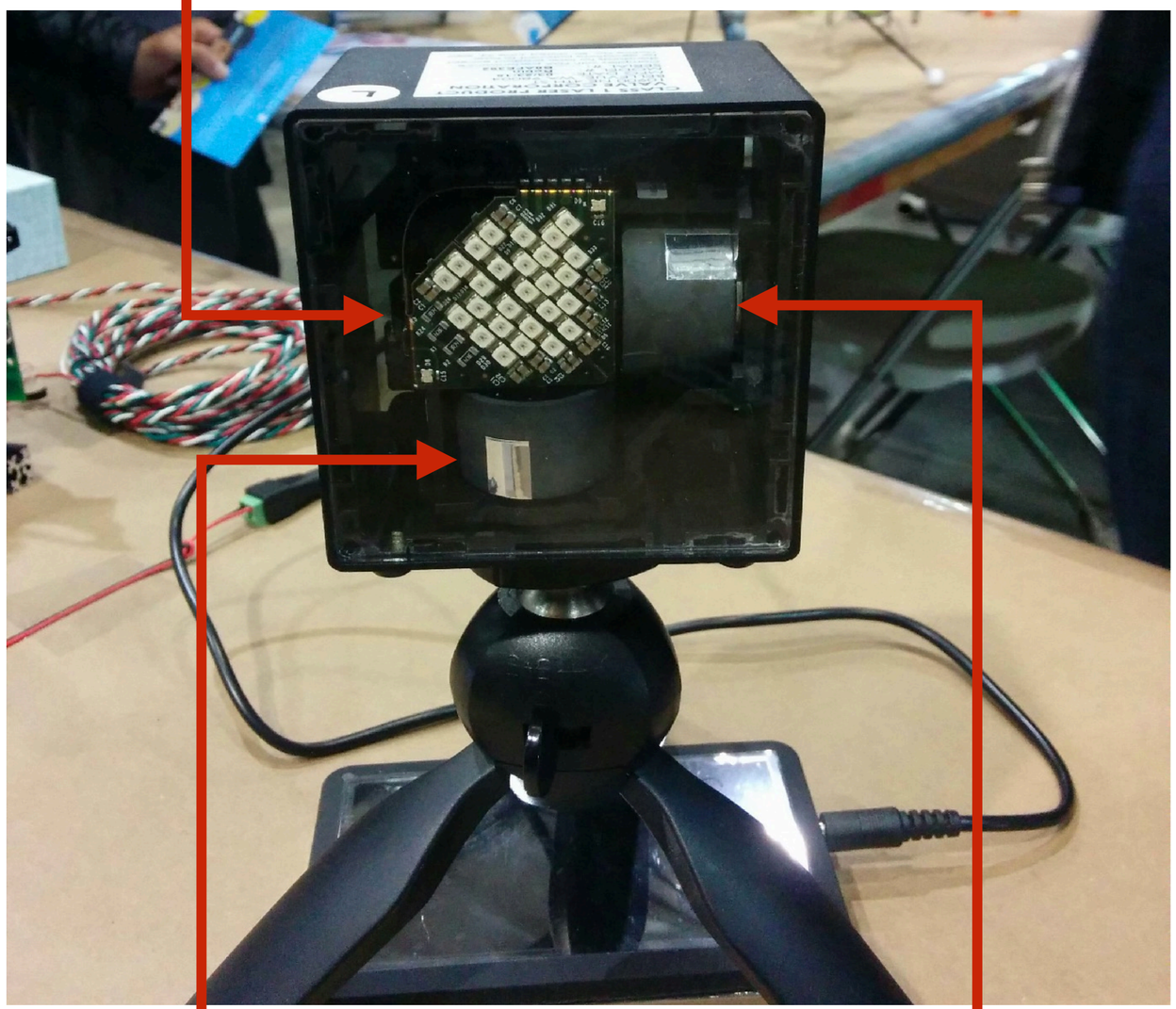
- Consider a 1,000 x 1,000 display spanning 100° field of view
  - 10 pixels per degree

- Assume:
  - You move your head 90° in 1 second (only modest speed)
  - End-to-end latency of system is 50 ms (1/20 sec)

- Therefore:
  - Displayed pixels are off by 4.5° ~ 45 pixels from where they would be in an ideal system with 0 latency

Valve's Lighthouse: cameraless position tracking

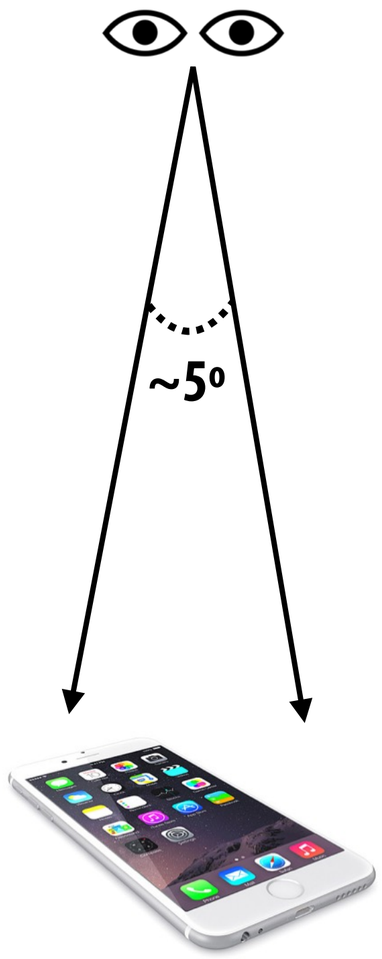




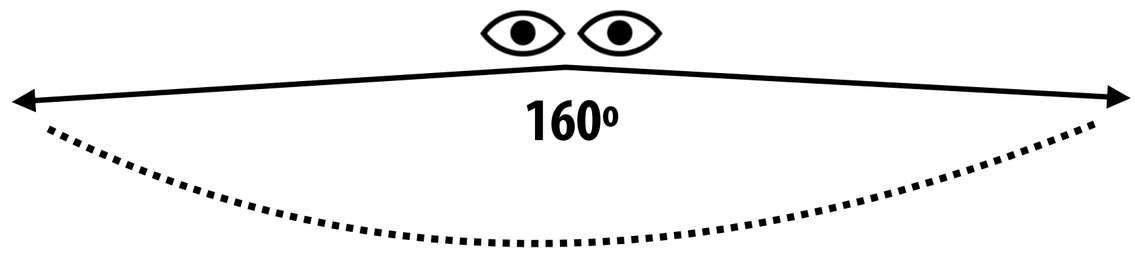
No need for computer vision processing to compute position of receiver: just a light sensor and an accurate clock!

Image credit: Travis Deyle
http://www.hizook.com/blog/2015/05/17/valves-lighthouse-tracking-system-may-be-big-news-robotics

Name of the game, part 2: high resolution



iPhone 6: 4.7 in "retina" display:
1.3 MPixel
326 ppi → 57 ppd



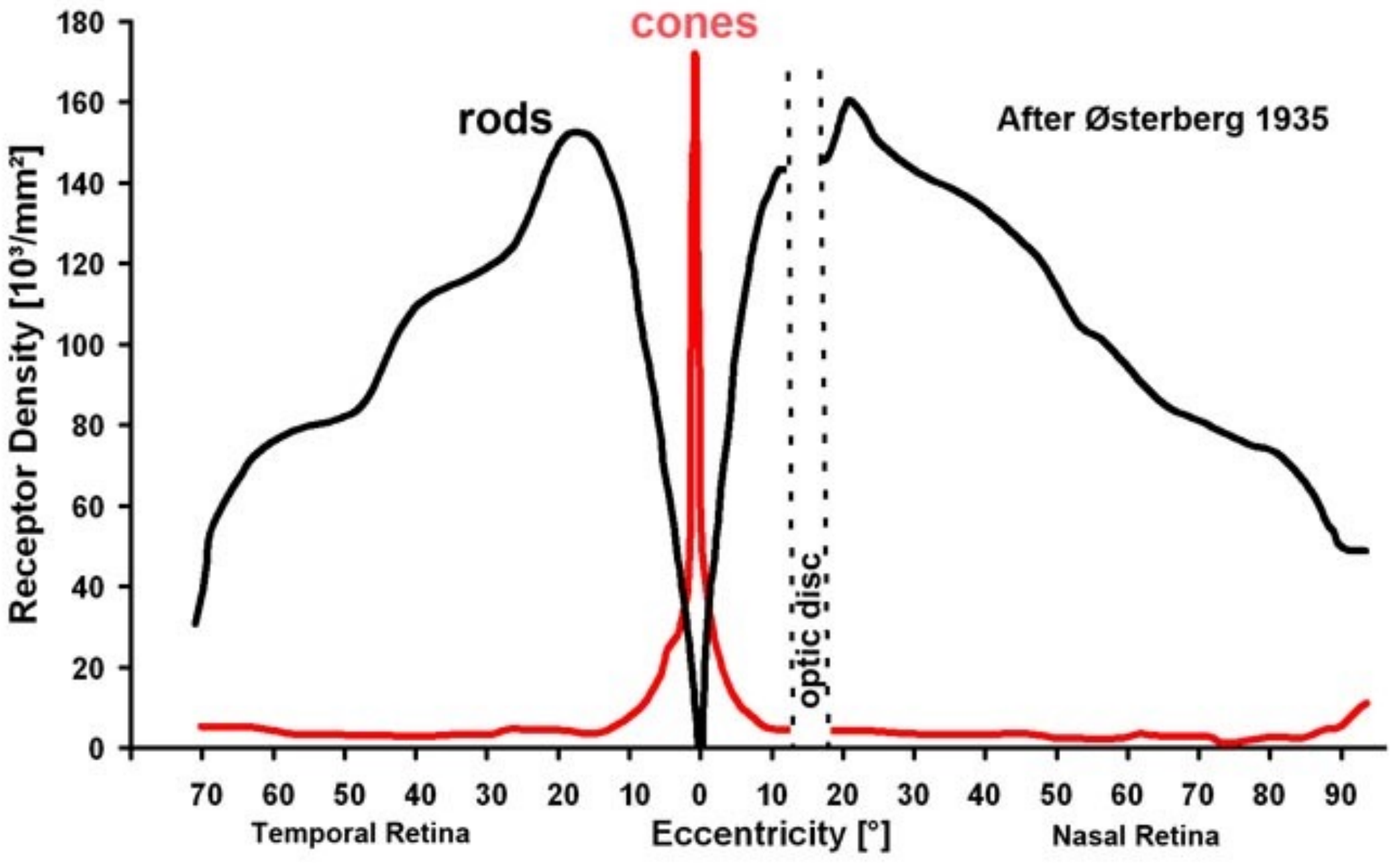
Human: ~160° view of field per eye (~200° overall)
(Note: this does not account for eye's ability to rotate in socket)

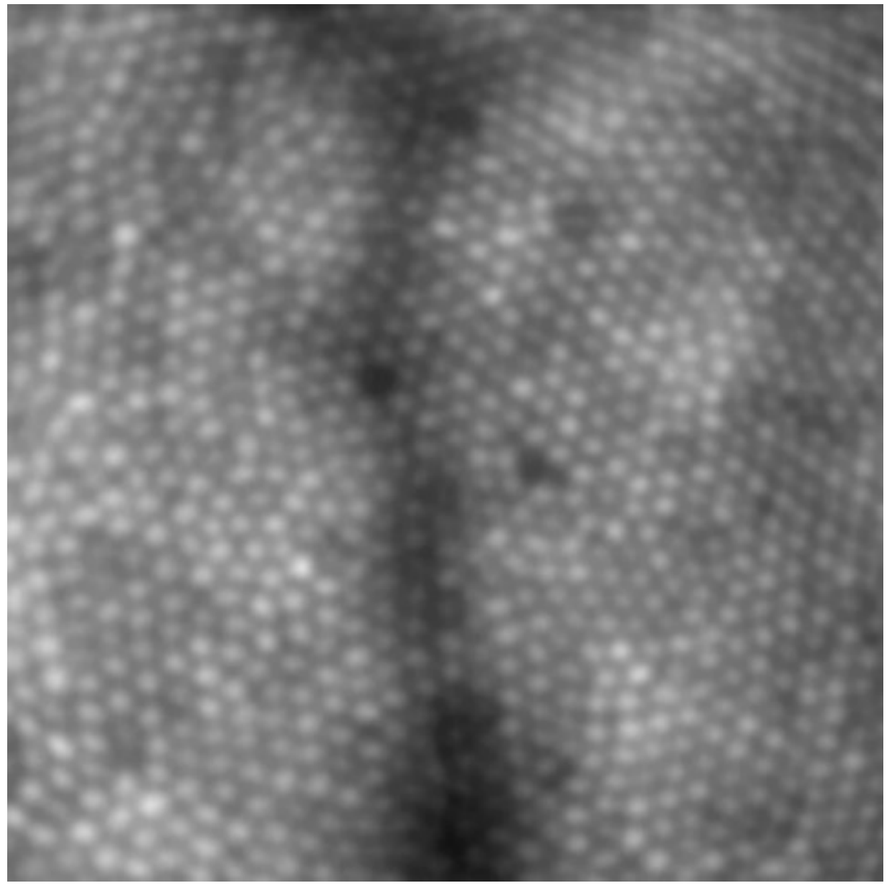
Future "retina" VR display:
57 ppd covering 200°
= 11K x 11K display per eye
= 220 MPixel

Strongly suggests need for eye tracking and foveated rendering (eye can only perceive detail in 5° region about gaze point

Eyes designed by SuperAtic LABS from the thenounproject.com

Recall: density of rods and cones in the retina



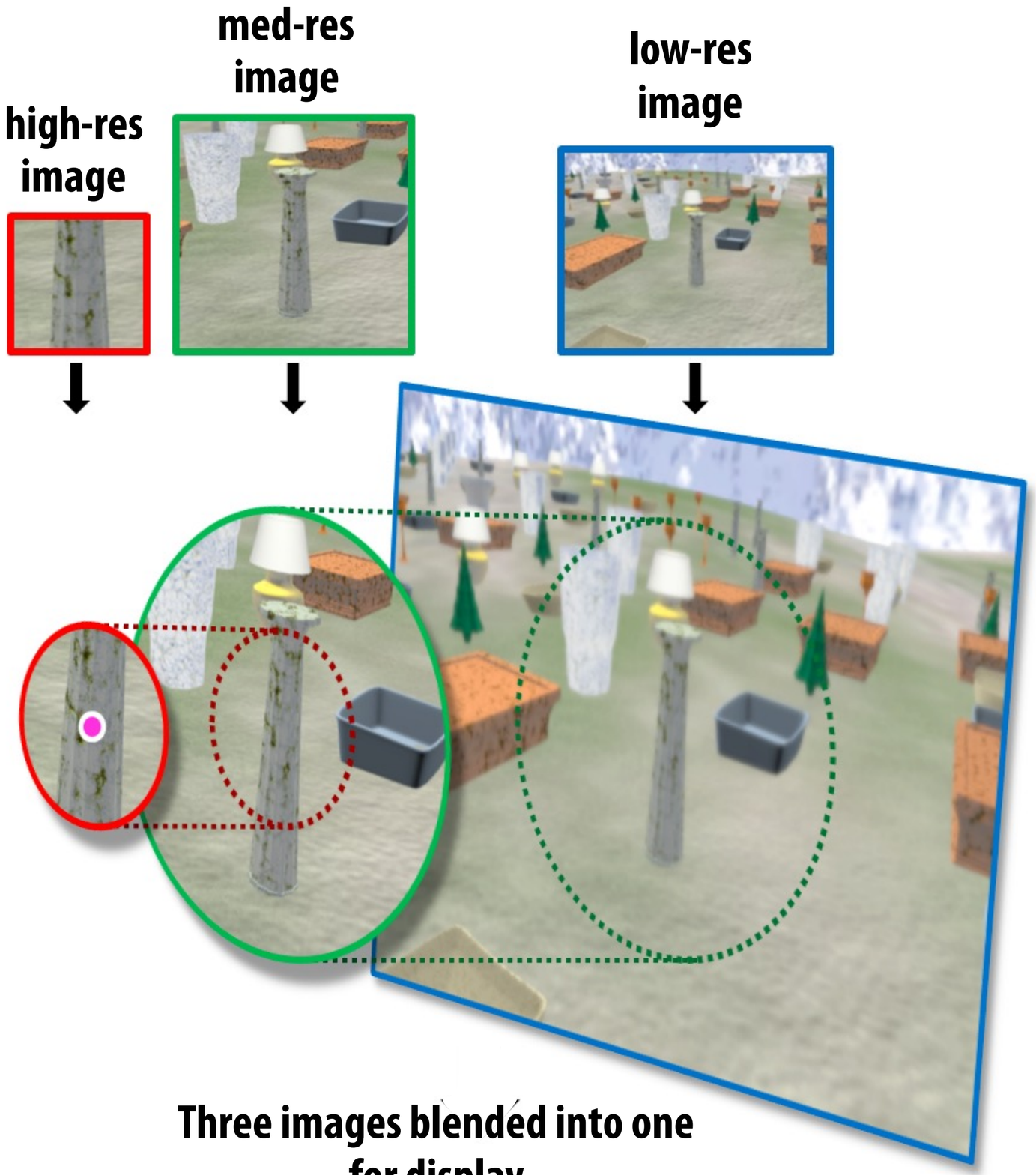


[Roorda 1999]

- Highest density of cones is in fovea
  (best color vision at center of where human is looking)
- Note "blind spot" due to optic disk

Addressing high resolution and high field of view: foveated rendering

Idea: track user's gaze, render with increasingly lower resolution farther away from gaze point



Three images blended into one for display

Traditional rendering (uniform screen sampling)

[Patney et al. 2016]

Low-pass filter away from fovea

In this image, gaussian blur with radius dependent on distance from fovea is used to remove high frequencies

[Patney et al. 2016]

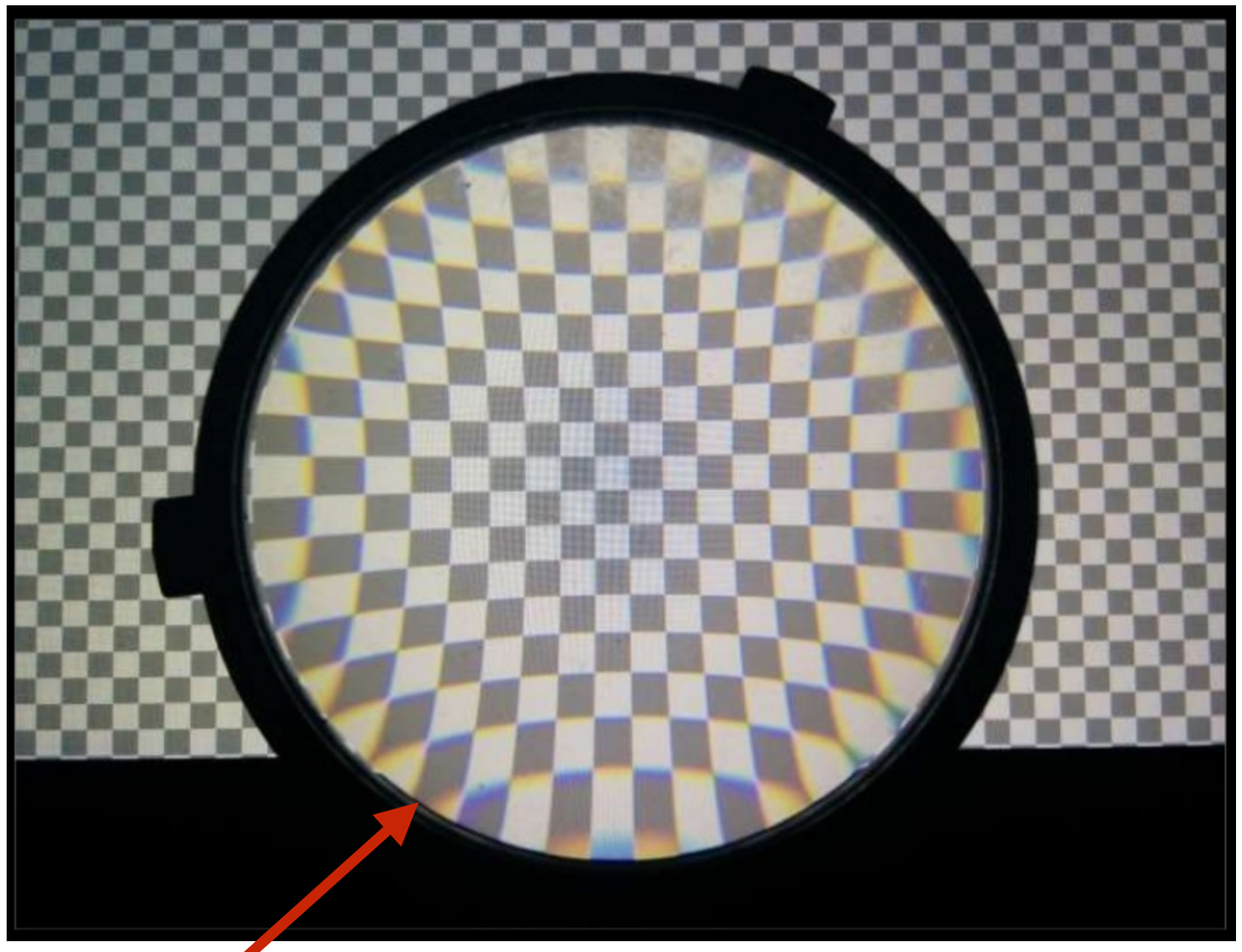
Contrast enhance periphery

Eye is receptive to contrast at periphery

[Patney et al. 2016]

Requirement: wide field of view

View of checkerboard through Oculus Rift lens

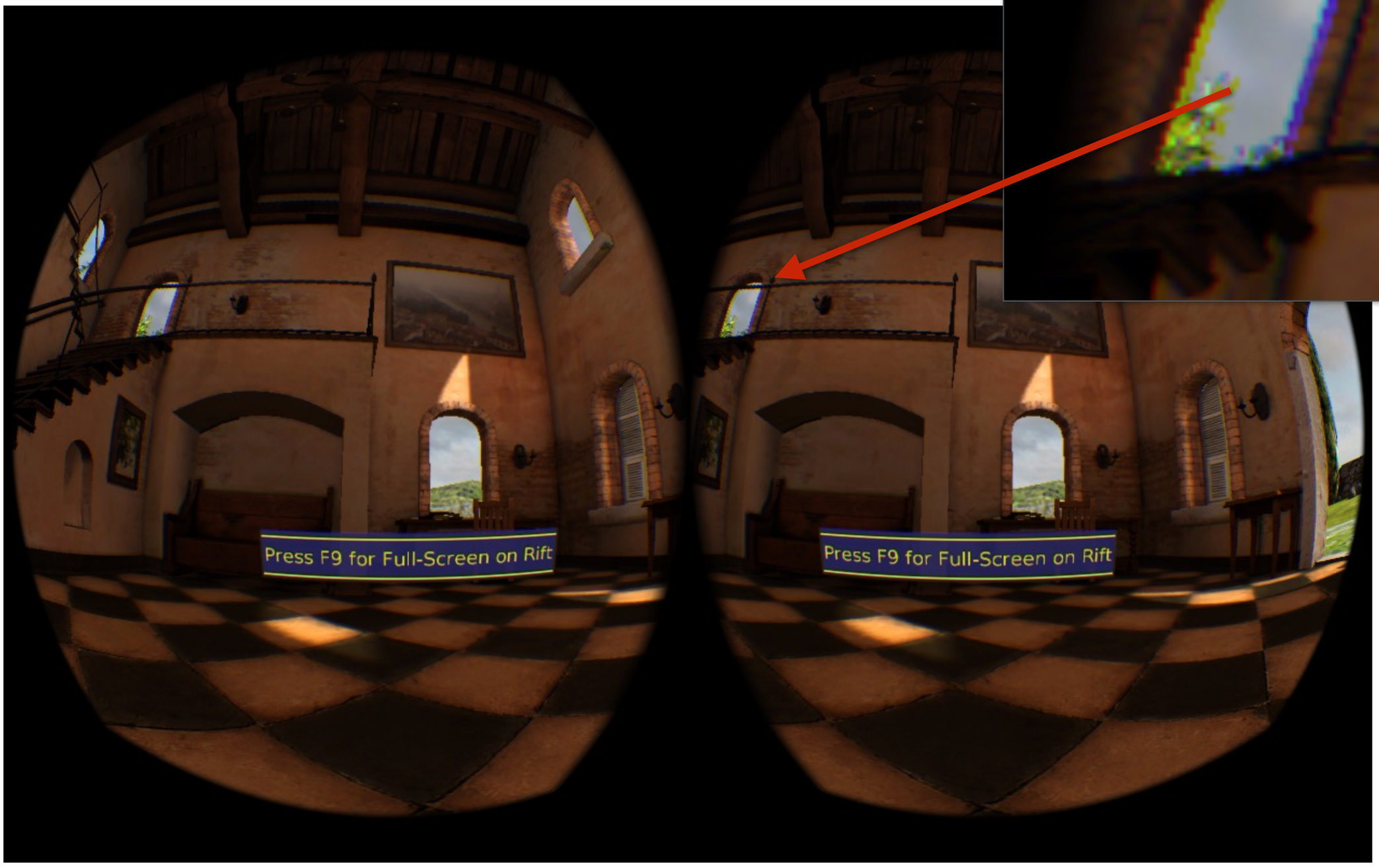


Lens introduces distortion

- Pincushion distortion
- Chromatic aberration (different wavelengths of light refract by different amount)

Icon credit: Eyes designed by SuperAtic LABS from the thenounproject.com
Image credit: Cass Everitt

Rendered output must compensate for distortion of lens in front of display



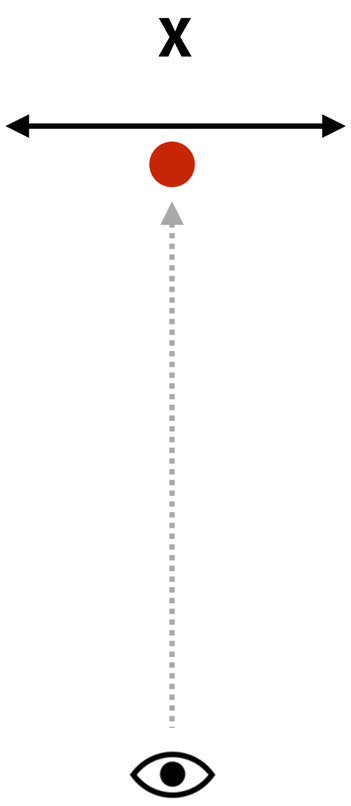
Step 1: render scene using traditional graphics pipeline at full resolution for each eye

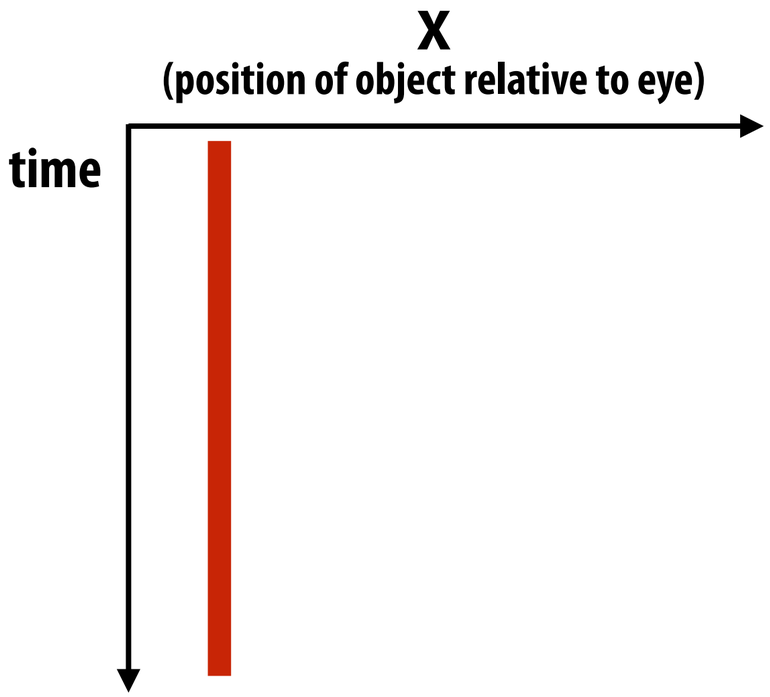
Step 2: warp images and composite into frame so rendering is viewed correctly after lens distortion

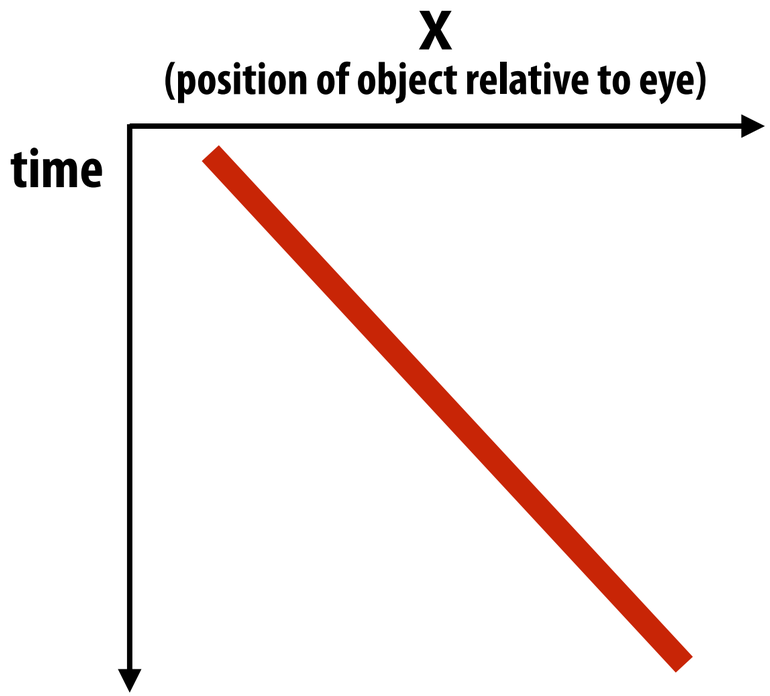
(Can apply unique distortion to R, G, B to approximate correction for chromatic aberration)

Image credit: Oculus VR developer guide

Consider object position relative to eye







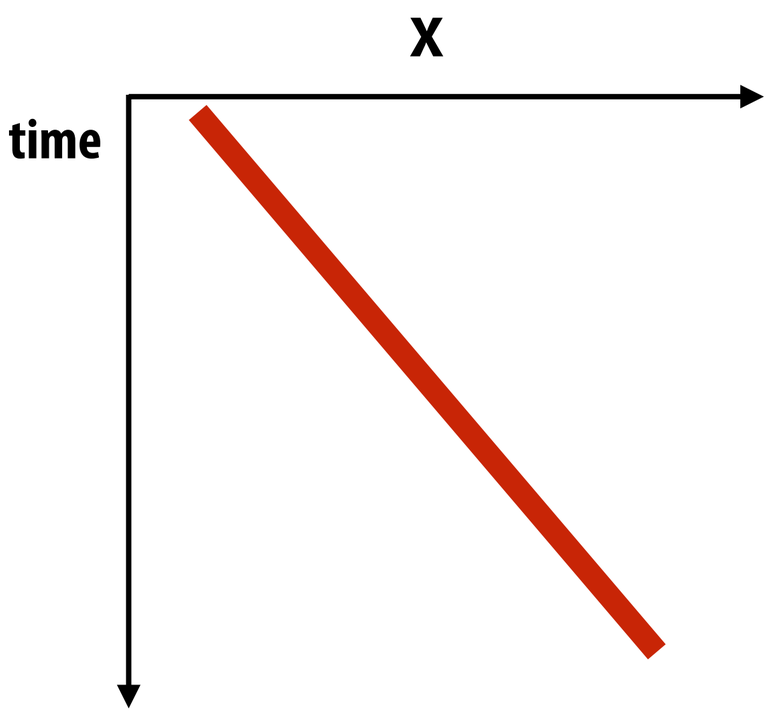
Case 1: object stationary relative to eye:
(eye still and red object still
OR
red object moving left-to-right and
eye moving to track object
OR
red object stationary in world but head moving
and eye moving to track object)

Case 2: object moving relative to eye:
(red object moving from left to right but
eye stationary, i.e., it's focused on a different
stationary point in world)

NOTE: THESE GRAPHS PLOT OBJECT POSITION RELATIVE TO EYE
RAPID HEAD MOTION WITH EYES TRACK A MOVING OBJECT IS A FORM OF CASE 1!!!

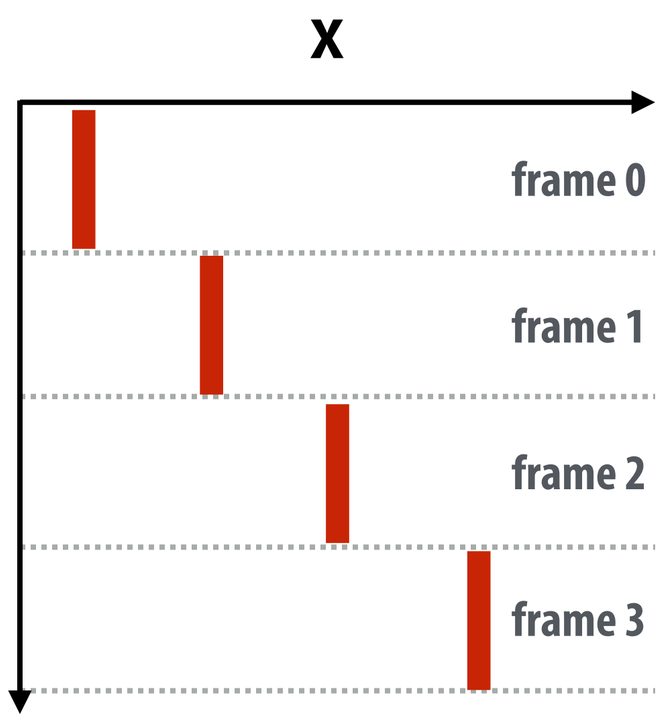
Spacetime diagrams adopted from presentations by Michael Abrash
Eyes designed by SuperAtic LABS from the thenounproject.com

Effect of latency: judder



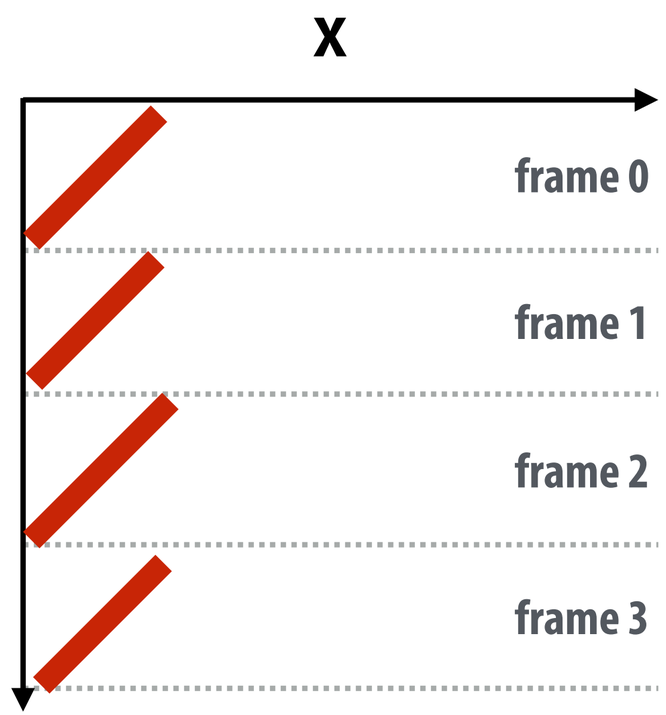
Case 2: object moving from left to right, eye stationary (eye stationary with respect to display)

Continuous representation.



Case 2: object moving from left to right, eye stationary (eye stationary with respect to display)

Light from display (image is updated each frame)



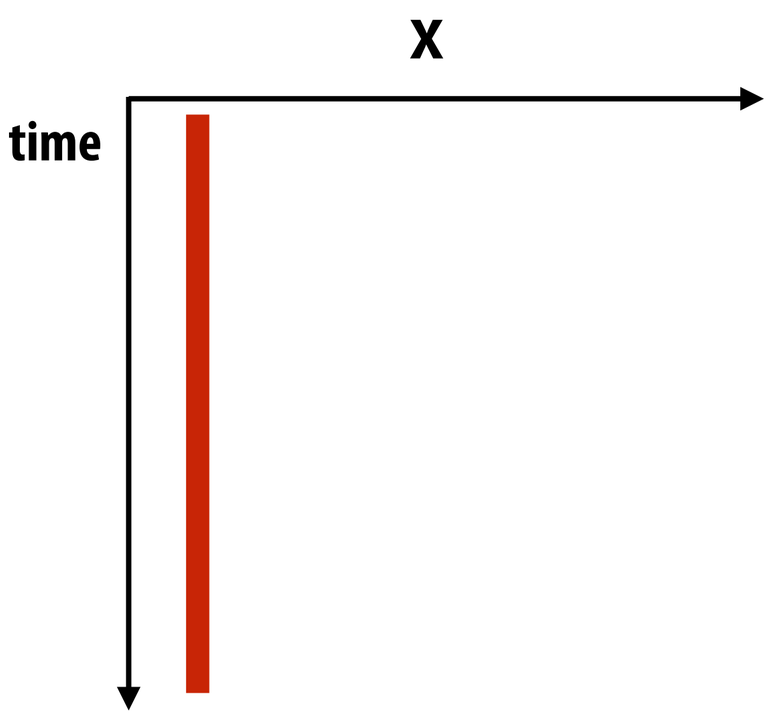
Case 1: object moving from left to right, eye moving continuously to track object (eye moving relative to display!)

Light from display (image is updated each frame)

Case 1 explanation: since eye is moving, object's position is relatively constant relative to eye (as it should be since the eye is tracking it). But due discrete frame rate, object falls behind eye, causing a smearing/strobing effect ("choppy" motion blur). Recall from earlier slide: 90 degree motion, with 50 ms latency results in 4.5 degree smear

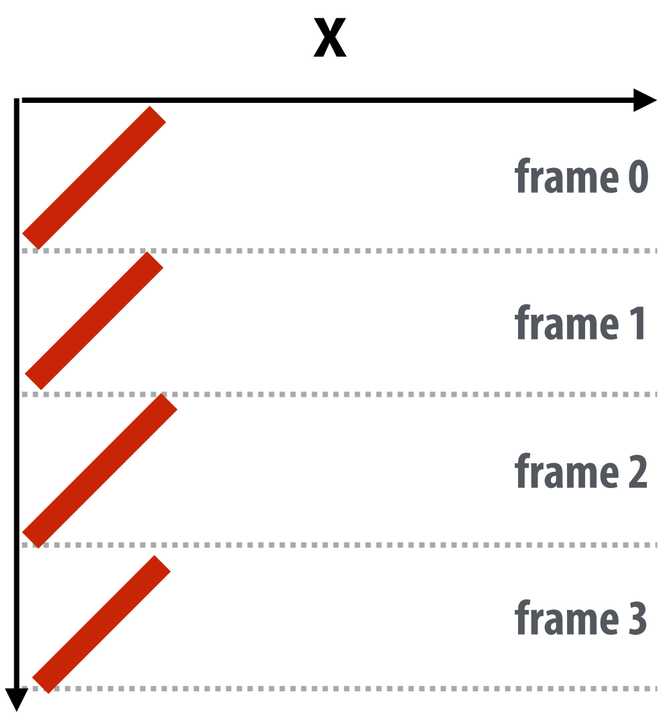
Spacetime diagrams adopted from presentations by Michael Abrash

Reducing judder: increase frame rate

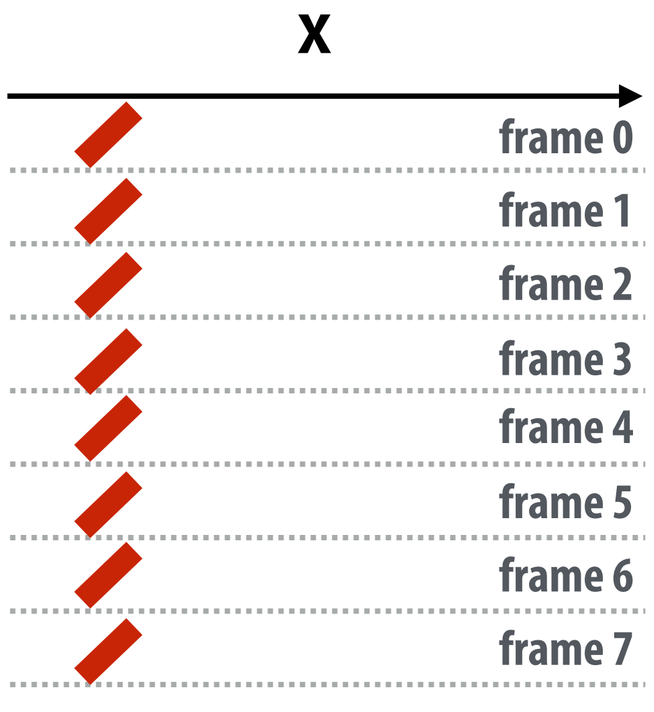


Case 1: continuous ground truth

red object moving left-to-right and
eye moving to track object
OR
red object stationary but head moving
and eye moving to track object



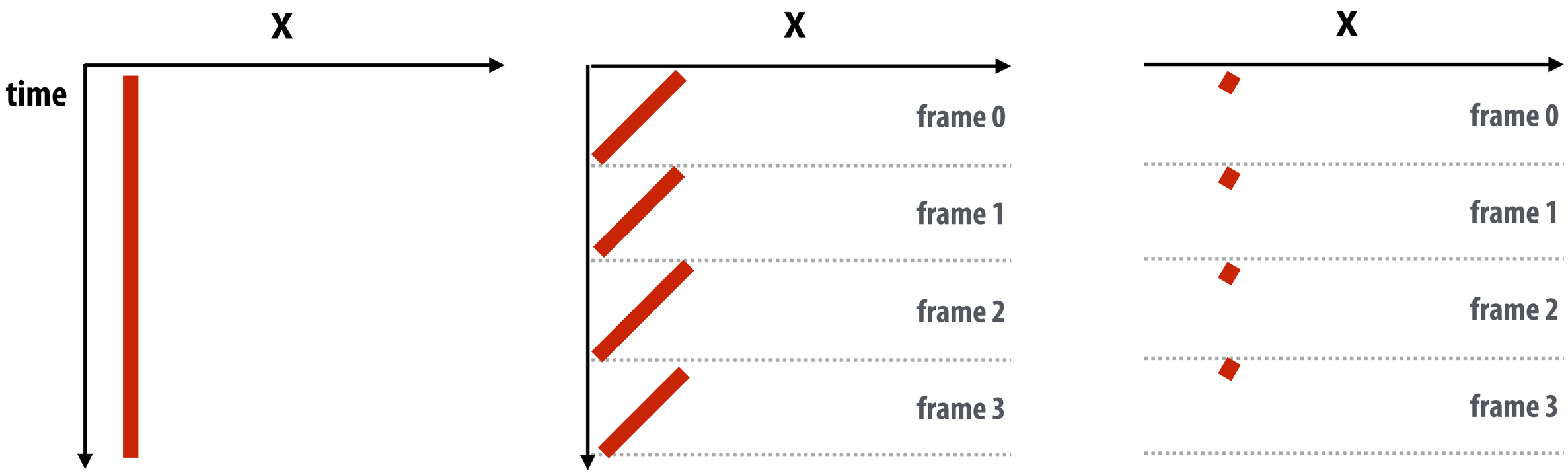
Light from display
(image is updated each frame)



Light from display
(image is updated each frame)

Higher frame rate results in closer
approximation to ground truth

Reducing judder: low persistence display



Case 1: continuous ground truth

red object moving left-to-right and
eye moving to track object
OR
red object stationary but head moving
and eye moving to track object

Light from full-persistence display

Light from low-persistence display

Full-persistence display: pixels emit light for entire frame

Low-persistence display: pixels emit light for small fraction of frame

Oculus DK2 OLED low-persistence display

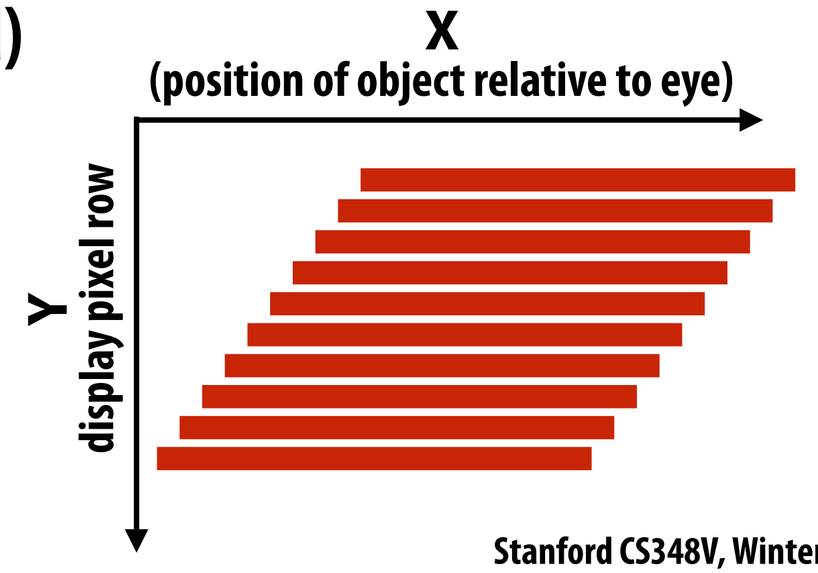
- 75 Hz frame rate (~13 ms per frame)
- Pixel persistence = 2-3ms

Artifacts due to rolling OLED backlight

- Image rendered based on scene state at time t_0
- Image sent to display, ready for output at time $t_0 + \Delta t$
- "Rolling backlight" OLED display lights up rows of pixels in sequence
  - Let *r* be amount of time to "scan out" a row
  - Row 0 photons hit eye at $t_0 + \Delta t$
  - Row 1 photos hit eye at $t_0 + \Delta t + r$
  - Row 2 photos hit eye at $t_0 + \Delta t + 2r$
- Implication: photons emitted from bottom rows of display are "more stale" than photos from the top!
- Consider <u>eye moving horizontally relative to display</u> (e.g., due to head movement while tracking square object that is stationary in world)

Result: perceived shear!

Recall rolling shutter effects on modern digital cameras.



Compensating for rolling backlight

- Perform post-process shear on rendered image
  - Similar to previously discussed barrel distortion and chromatic warps
  - Predict head motion, assume fixation on static object in scene
    - Only compensates for shear due to head motion, not object motion

- Render each row of image at a different time (the predicted time photons will hit eye)
  - Suggests exploration of different rendering algorithms that are more amenable to fine-grained temporal sampling, e.g., ray caster? (each row of camera rays samples scene at a different time)

Increasing frame rate using re-projection

- **Goal: maintain as high a frame rate as possible under challenging rendering conditions:**
  - Stereo rendering: both left and right eye views
  - High-resolution outputs
  - Must render extra pixels due to barrel distortion warp
  - Many "rendering hacks" (bump mapping, billboards, etc.) are less effective in VR so rendering must use more expensive techniques

- **Researchers experimenting with reprojection-based approaches to improve frame rate (e.g., Oculus' "Time Warp")**
  - Render using conventional techniques at 30 fps, reproject (warp) image to synthesize new frames based on predicted head movement at 75 fps
  - Potential for image processing hardware on future VR headsets to perform high frame-rate reprojection based on gyro/accelerometer

Near-future VR system components

Low-latency image processing for subject tracking

High-resolution, high-frame rate, wide-field of view display

Massive parallel computation for high-resolution rendering

Exceptionally high bandwidth connection between renderer and display:
e.g., 4K x 4K per eye at 90 fps!

In headset motion/accel sensors + eye tracker

On headset graphics processor for sensor processing and re-projection