L1b: Cameras, Sampling, Calibration

2018 IEEE SPS Summer School on Light Field Data Representation, Interpretation, and Compression

Donald G. Dansereau, May 2018

Schedule

09:30 - 10:15	Lecture 1 a D Danserau
10:15 - 10:45	Coffee break
10:45 - 11:30	Lecture 1 b D Danserau
11:30 - 11:45	Break
11:45 - 12:30	Lecture 1 c D Danserau
12:30 - 13:00	Lunch
13:00 - 14:00	

14:00 - 15:00	IEEE
15:00 - 15:30	Exercise 1 D Danserau
15:30 - 16:00	Coffee break
16:00 - 16:45	Exercise 1 D Danserau
16:45 - 17:00	
17:00 - 17:30	

Outline

Lecture 1a: Introduction to Light Fields

Lecture 1b: Cameras, Sampling, & Calibration

- Review: 2D Image Geometry
- Sampling Patterns
- Decoding, Calibration, Rectification
- An interpolating renderer

Lecture 1c: Basic Processing

Hands-on: Writing a renderer, handling light fields in matlab

Review: Image Transformations

$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 \\ \sin\theta_0 & \cos\theta_0 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \end{bmatrix} + \begin{bmatrix} \Delta x_0 \\ \Delta y_0 \end{bmatrix}$$

$$\boldsymbol{p_i} = [x_i, y_i]^\top \qquad \boldsymbol{t_i} = [\Delta x_i, \Delta y_i]^\top \qquad R_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}$$

$$\begin{aligned} \boldsymbol{p_1} &= R_0 \boldsymbol{p_0} + \boldsymbol{t_0} \\ \boldsymbol{p_2} &= R_1 \boldsymbol{p_1} + \boldsymbol{t_1} \\ &= R_1 (R_0 \boldsymbol{p_0} + \boldsymbol{t_0}) + \boldsymbol{t_1} \end{aligned}$$

… Concatenation is cumbersome

Again with Homogeneous Coordinates

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta_0 & -\sin\theta_0 & \Delta x_0 \\ \sin\theta_0 & \cos\theta_0 & \Delta y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix}$$

$$\boldsymbol{p_i} = [x_i, y_i, 1]^{\mathsf{T}} \qquad H_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i & \Delta x_i \\ \sin\theta_i & \cos\theta_i & \Delta y_i \\ 0 & 0 & 1 \end{bmatrix}$$

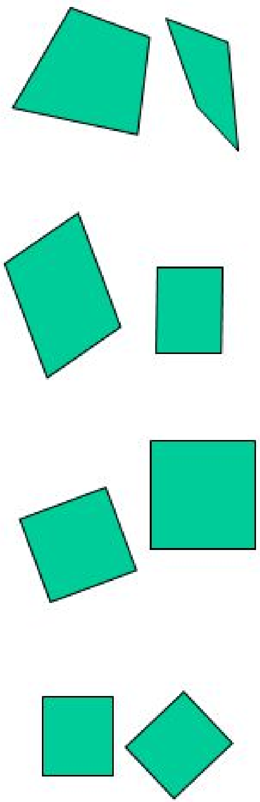
$$\boldsymbol{p_1} = H_0 \boldsymbol{p_0}$$

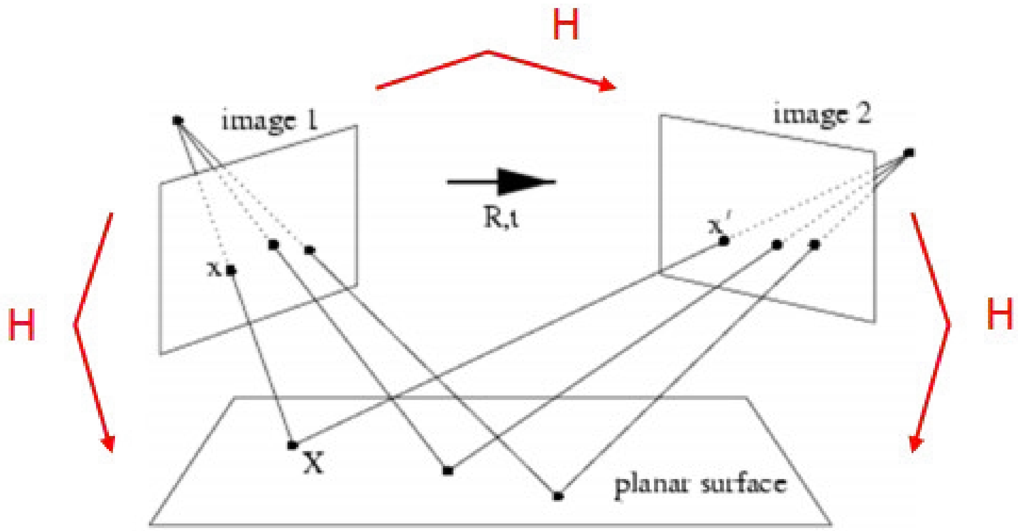
$$\boldsymbol{p}_2 = H_1 \boldsymbol{p_1} = H_1 H_0 \boldsymbol{p_0}$$

… Concatenation is matrix multiplication, much nicer

Review: Image Transformations

Projective 8dof	$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$
Affine 6dof	$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$
Similarity 4dof	$\begin{bmatrix} sr_{11} & sr_{12} & t_x \\ sr_{21} & sr_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$
Euclidean 3dof	$\begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix}$





Projective transform between two camera views

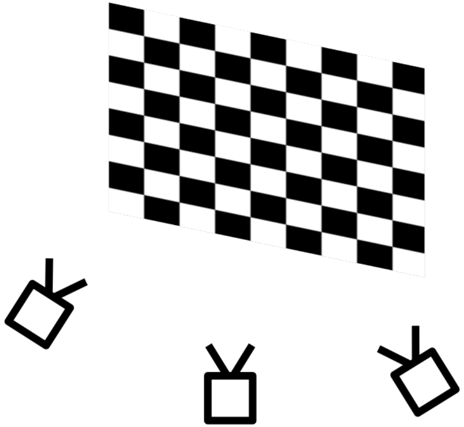
Image registration, stitching, ...

Planar scene only!

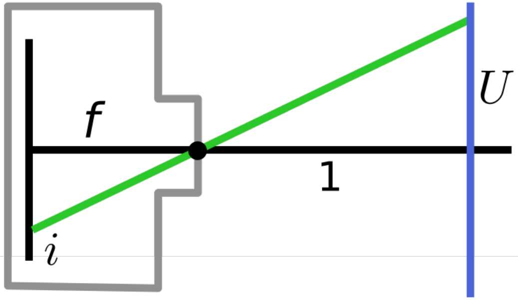
Image c/o Mark Pollefeys http://slideplayer.com/slide/5947732/

Review: Camera Calibration

Extrinsics (pose):



Intrinsics (pixel-ray mapping):



$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \begin{bmatrix} h_{1,1} & \cancel{h_{1,2}} & h_{1,3} \\ 0 & h_{2,2} & h_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ V \\ 1 \end{bmatrix}$$

Radial lens distortion:

$$\begin{bmatrix} x_{\mathrm{d}} \\ y_{\mathrm{d}} \end{bmatrix} = \left(1 + k_1^{(\mathrm{Z})} r^2 + k_2^{(\mathrm{Z})} r^4\right) \begin{bmatrix} x_{\mathrm{u}} \\ y_{\mathrm{u}} \end{bmatrix}$$

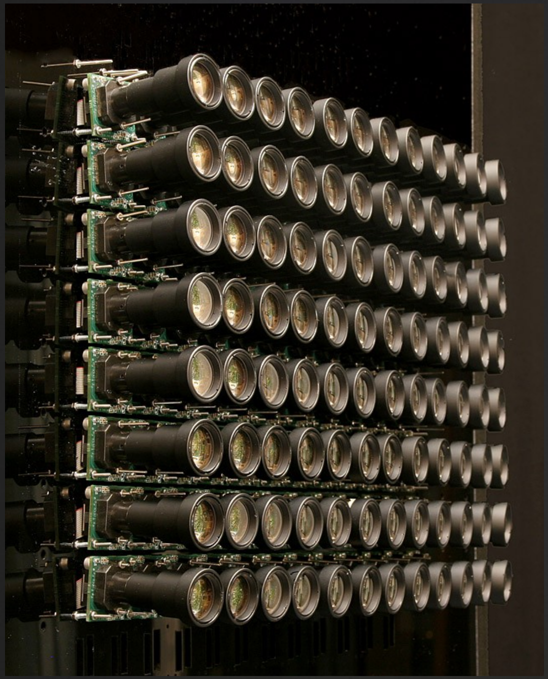
$$\text{where } r = \sqrt{{x_{\mathrm{u}}}^2 + {y_{\mathrm{u}}}^2}$$

For N_{poses} poses

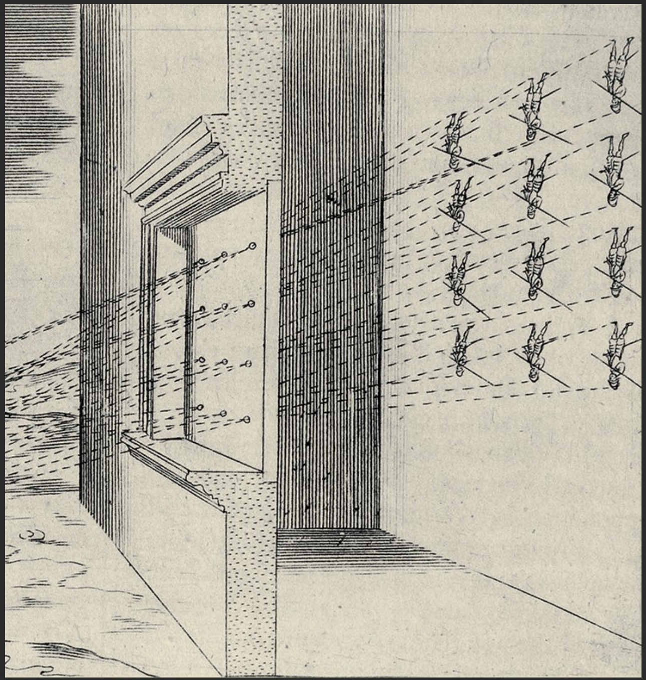
Estimate $6N_{poses}$ extrinsics
4 intrinsics
~3 radial distortion

Camera Sampling Patterns

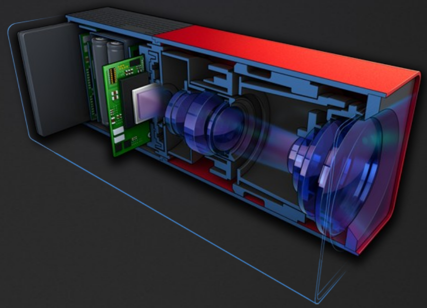
Light Field Cameras



Stanford camera array



Bettini 1642

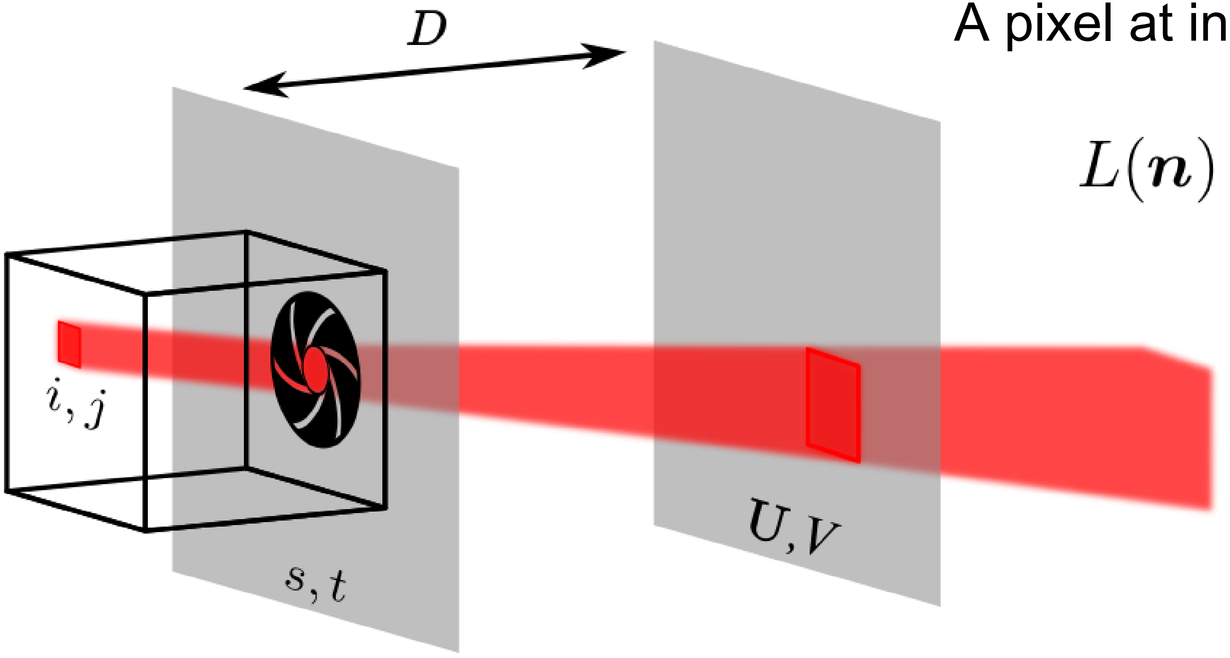


Lytro lenslet-based cameras

Raytrix

EPIImaging

Sampling Patterns



A pixel at index $\boldsymbol{n}$ integrates over rays $\boldsymbol{\Phi}$

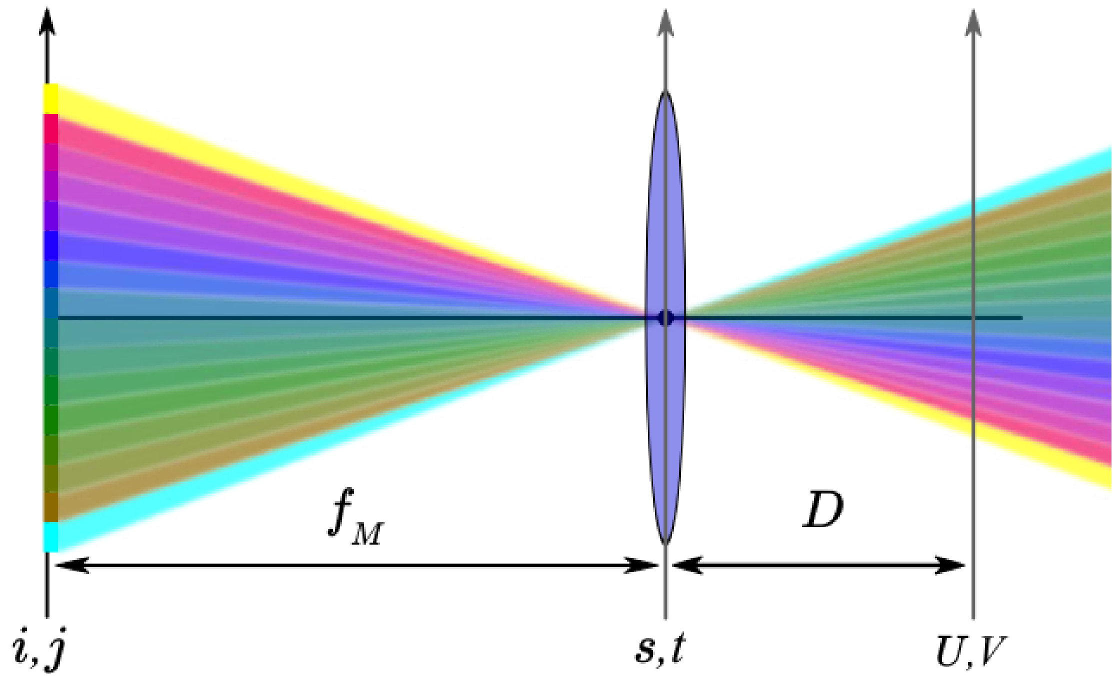
$$L(\boldsymbol{n}) = \int_{4D} w(\boldsymbol{n}, \boldsymbol{\Phi})\, \mathcal{L}(\boldsymbol{\Phi})\; d\boldsymbol{\Phi}$$

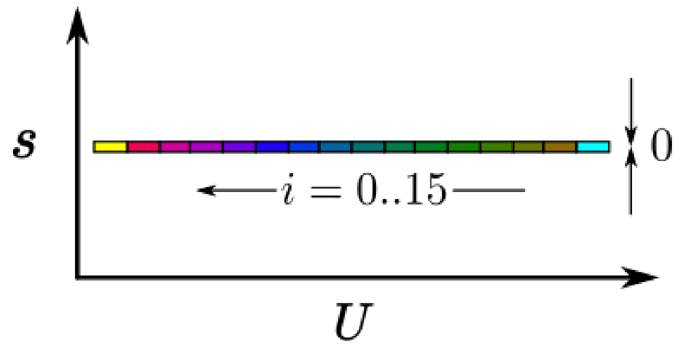
Following a per-pixel weighting patch w

For ideal cameras w are typically 4D parallelograms in ray space

Note rays are defined by *s,t,u,v* in meters; pixels by *i,j* indices (and later *i,j,k,l*)

Example 1: Pinhole Camera

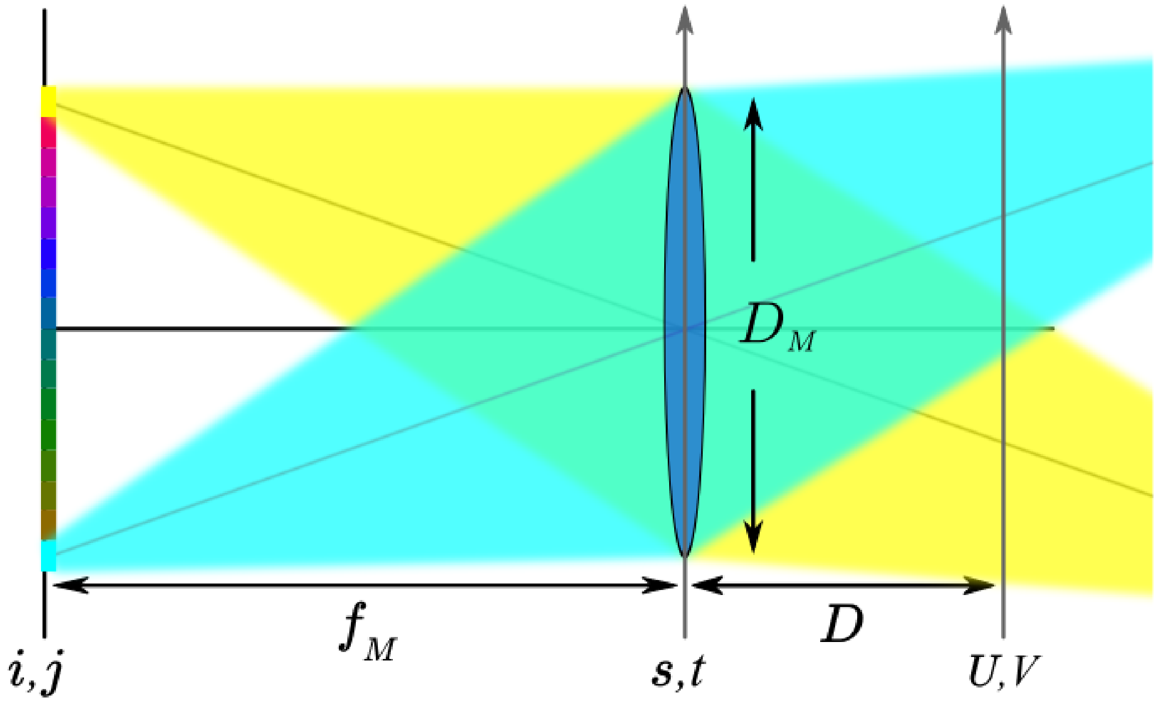


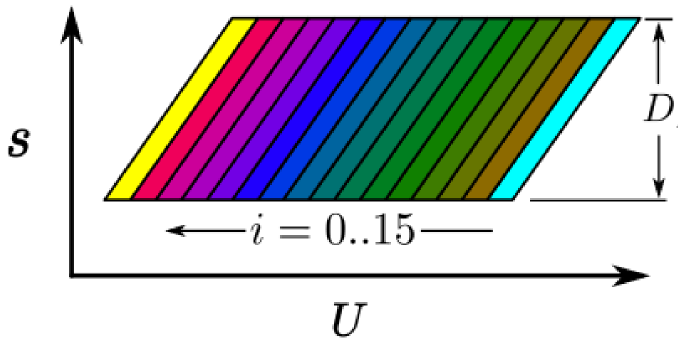


Pinhole: w are infinitesimal in *s*

Finite pixels: w have width in U

Example 2: Thin-Lens Camera





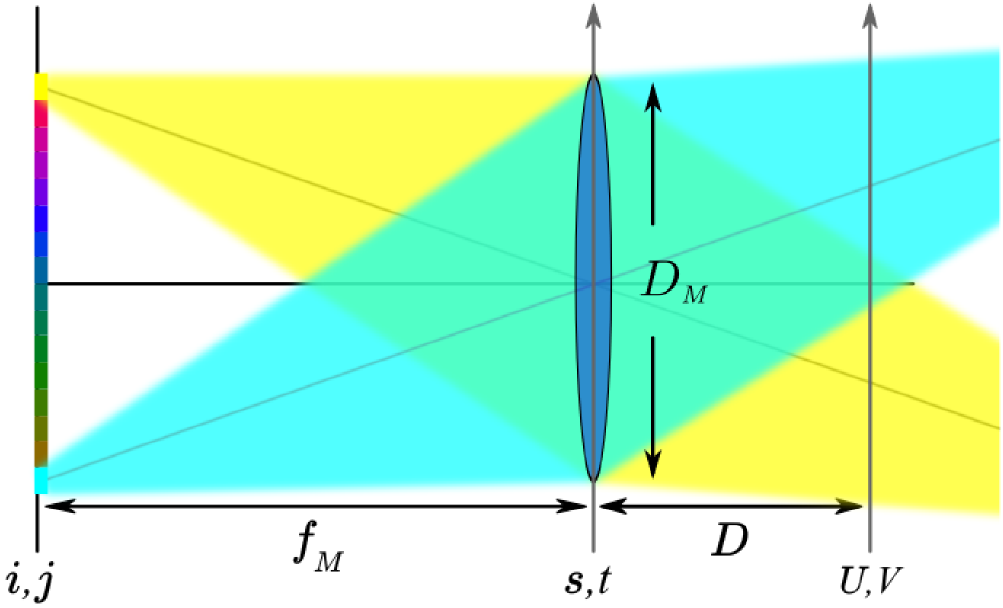
Lens: now w has width in *s* and U

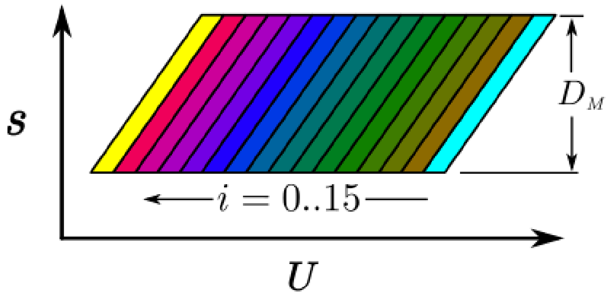
Calibration generally maps out the centroids of the *w* quads

Camera Intrinsics

We can write a simple expression relating pixel index to patch center

If we set plane separation D=1m, *H* is a conventional camera intrinsic matrix





$$\begin{bmatrix} i \\ j \\ 1 \end{bmatrix} = \begin{bmatrix} h_{1,1} & h_{1,2} & h_{1,3} \\ 0 & h_{2,2} & h_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} U \\ V \\ 1 \end{bmatrix}$$

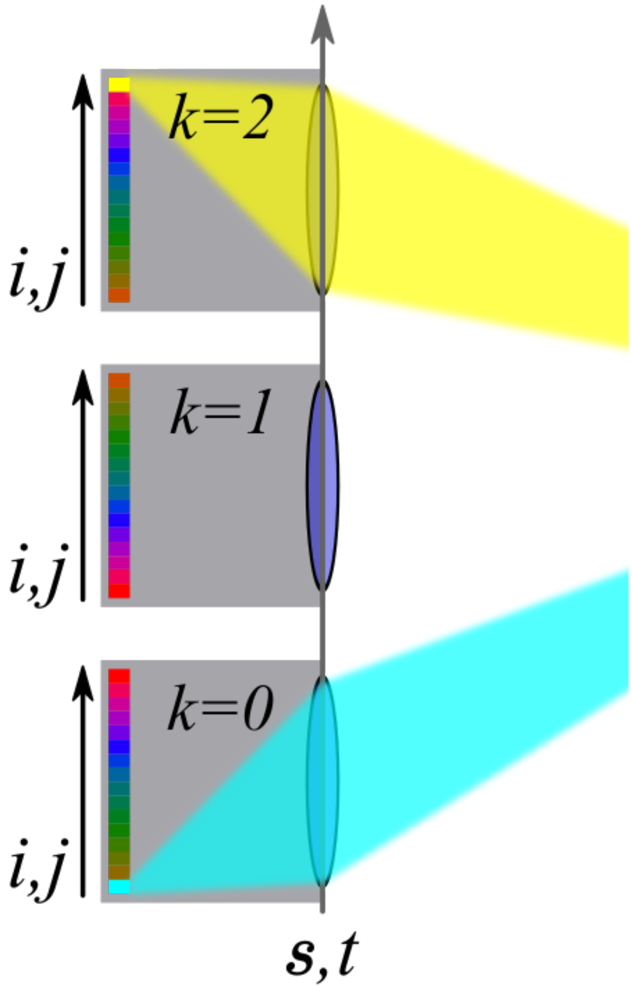
$$\boldsymbol{n} = H\boldsymbol{\Phi}$$

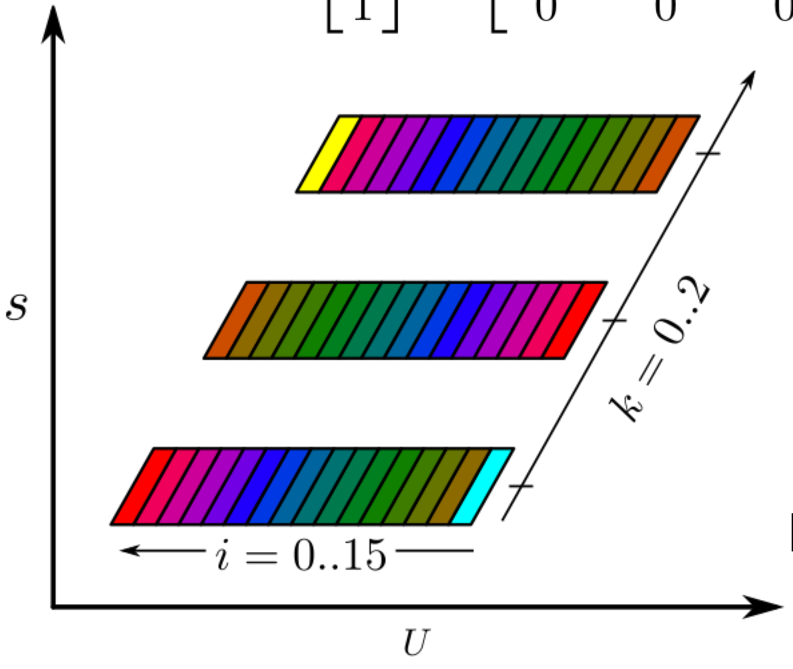
Camera Array

Plenoptic intrinsic matrix H:

$$\boldsymbol{\Phi} = \boldsymbol{H}\boldsymbol{n} \qquad \boldsymbol{n} = \boldsymbol{H}^{-1}\boldsymbol{\Phi}$$

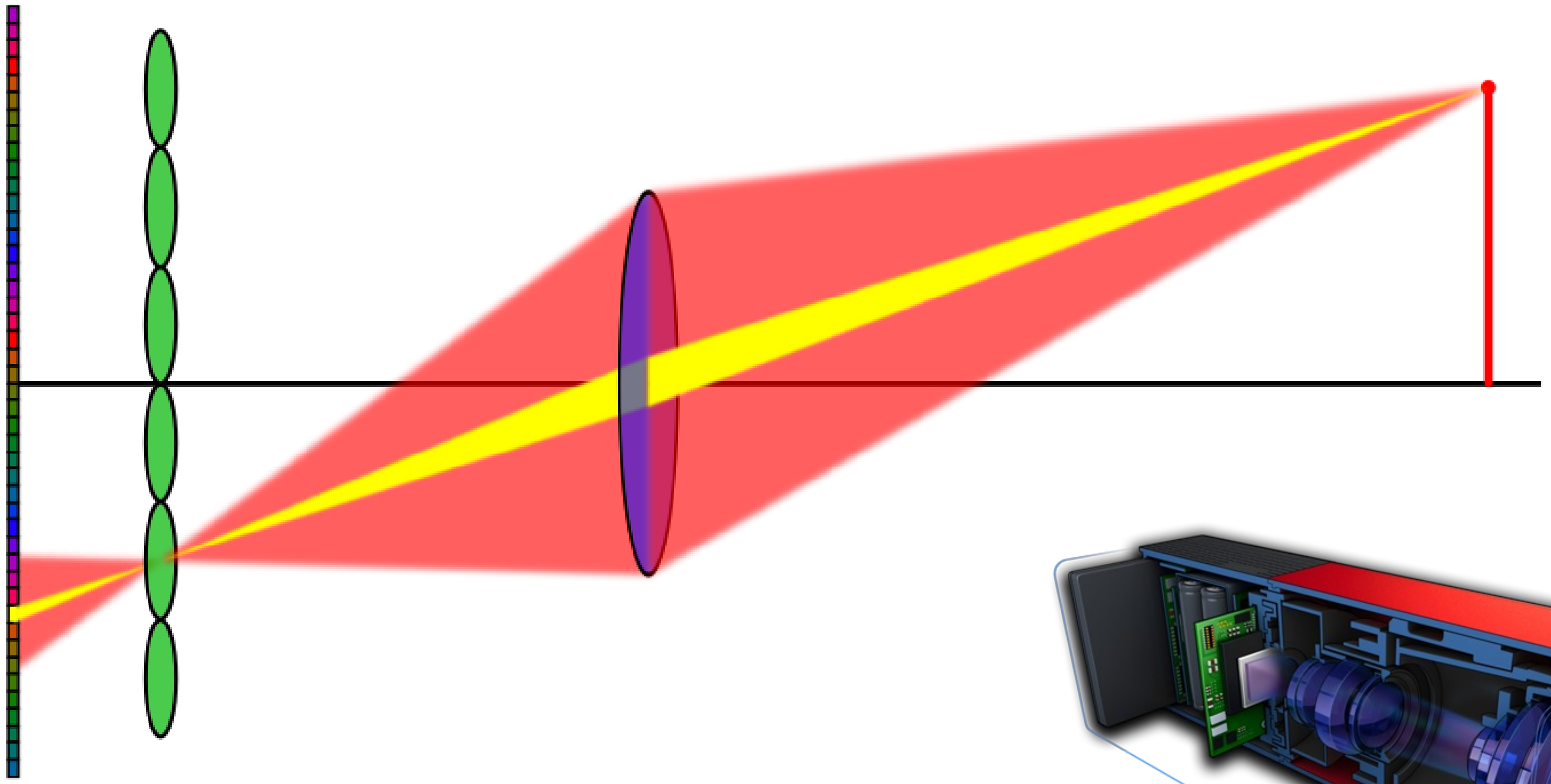
$$\begin{bmatrix} s \\ t \\ U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & h_{3,3} & 0 & h_{1,5} \\ 0 & 0 & 0 & h_{4,4} & h_{2,5} \\ h_{3,1} & 0 & h_{3,3} & 0 & h_{3,5} \\ 0 & h_{4,2} & 0 & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ l \\ 1 \end{bmatrix}$$

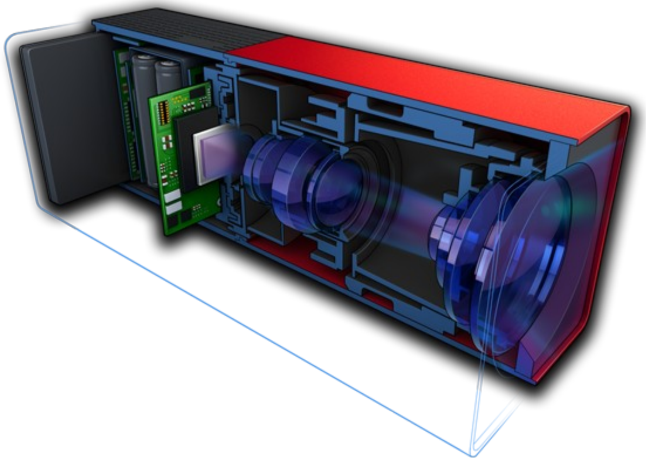




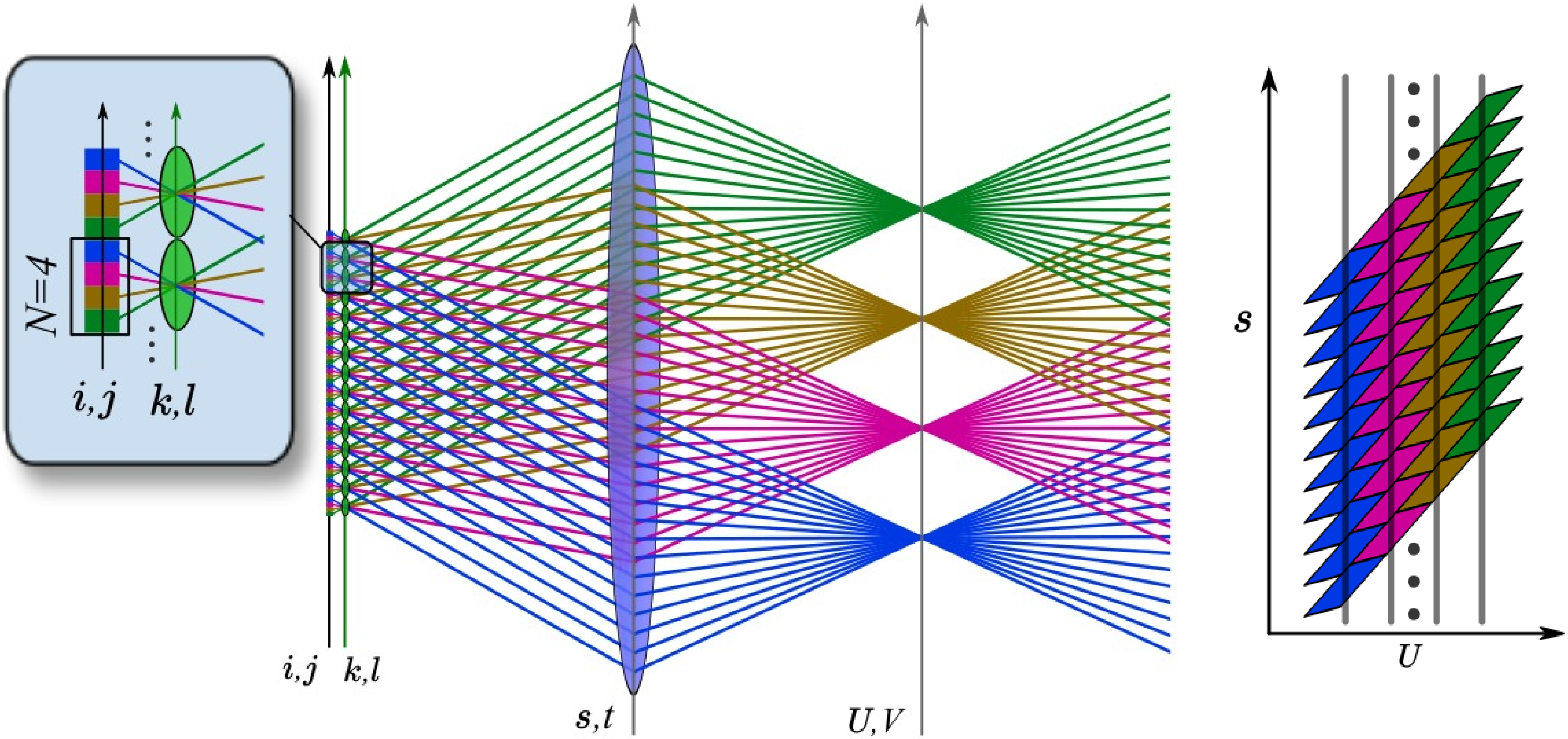
Note the aliasing!

Lenslet-Based LF Camera



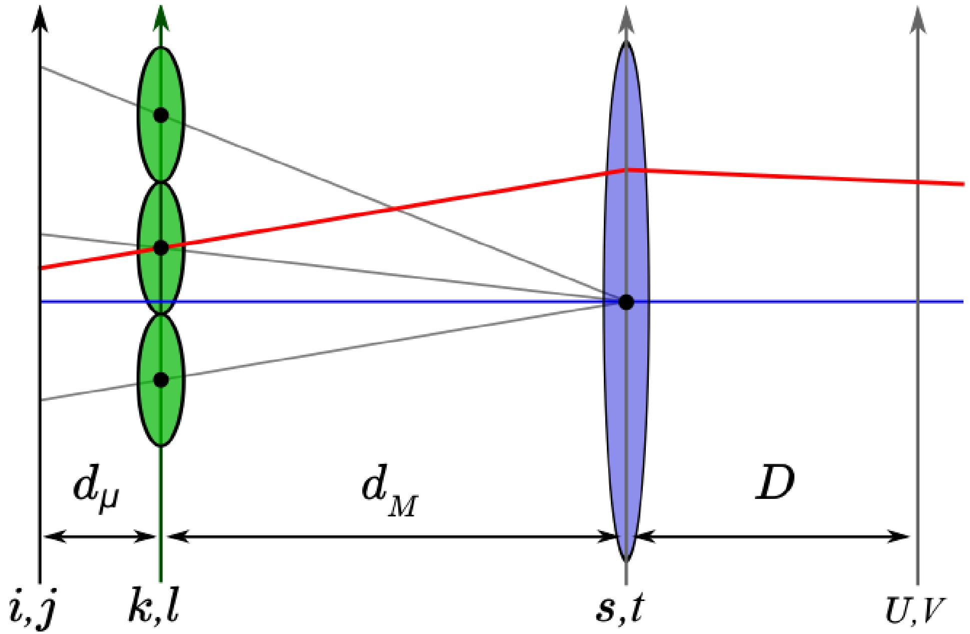


Simplified Lenslet Camera



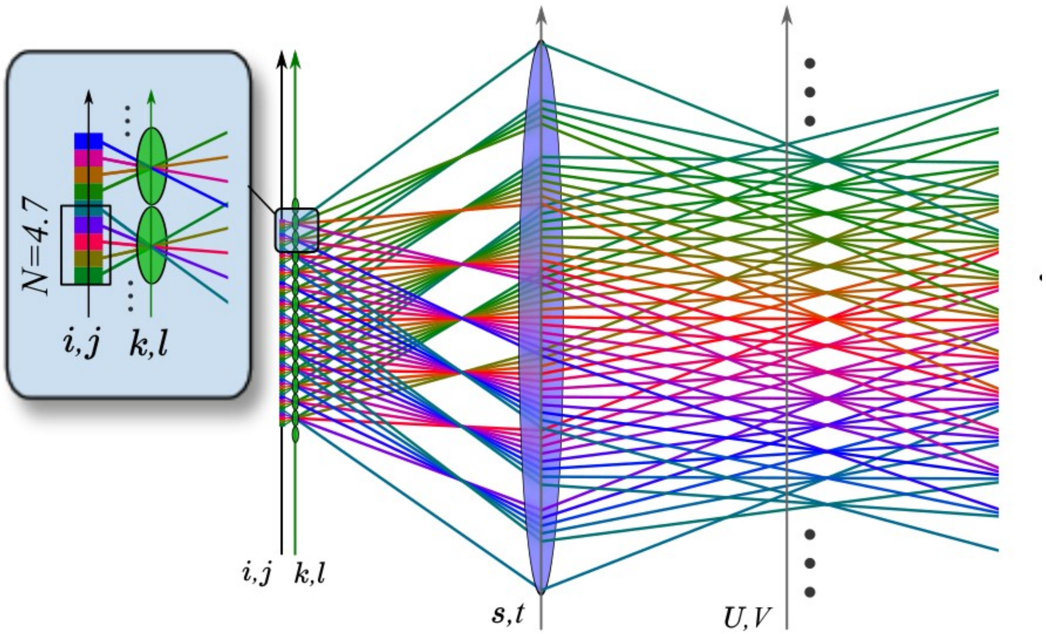
(ideally) no aliasing

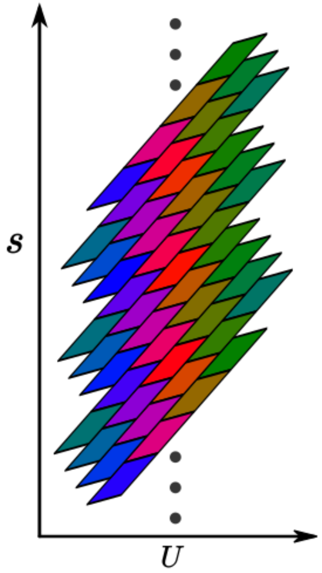
Less Simplified Lenslet Camera



$$\begin{bmatrix} s \\ t \\ U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} h_{1,1} & 0 & h_{1,3} & 0 & h_{1,5} \\ 0 & h_{2,2} & 0 & h_{2,4} & h_{2,5} \\ h_{3,1} & 0 & h_{3,3} & 0 & h_{3,5} \\ 0 & h_{4,2} & 0 & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ l \\ 1 \end{bmatrix}$$

Less Simplified Lenslet Camera



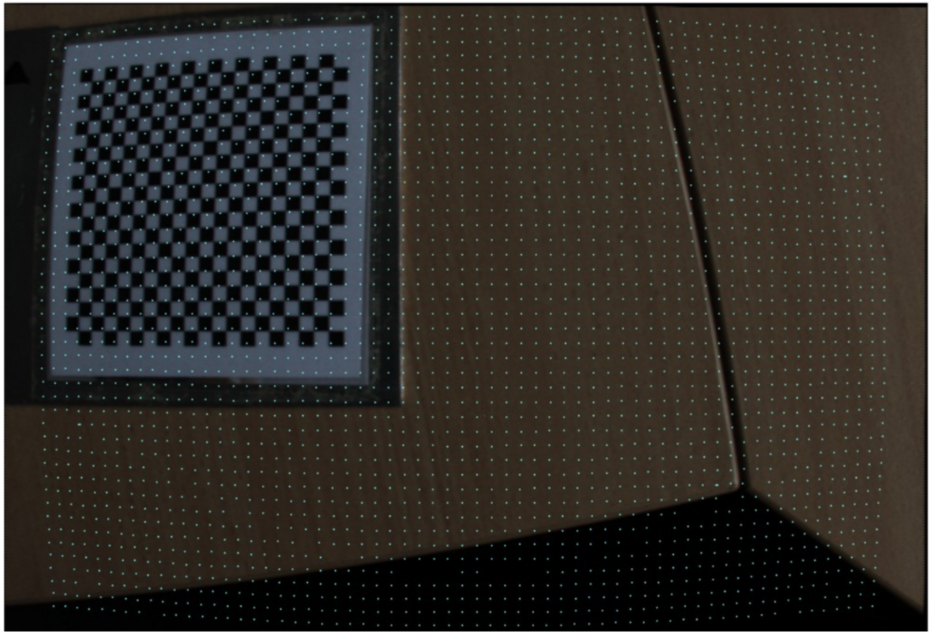


Plenoptic intrinsic matrix H:

$$\boldsymbol{\Phi} = \boldsymbol{H}\boldsymbol{n} \qquad \boldsymbol{n} = \boldsymbol{H}^{-1}\boldsymbol{\Phi}$$

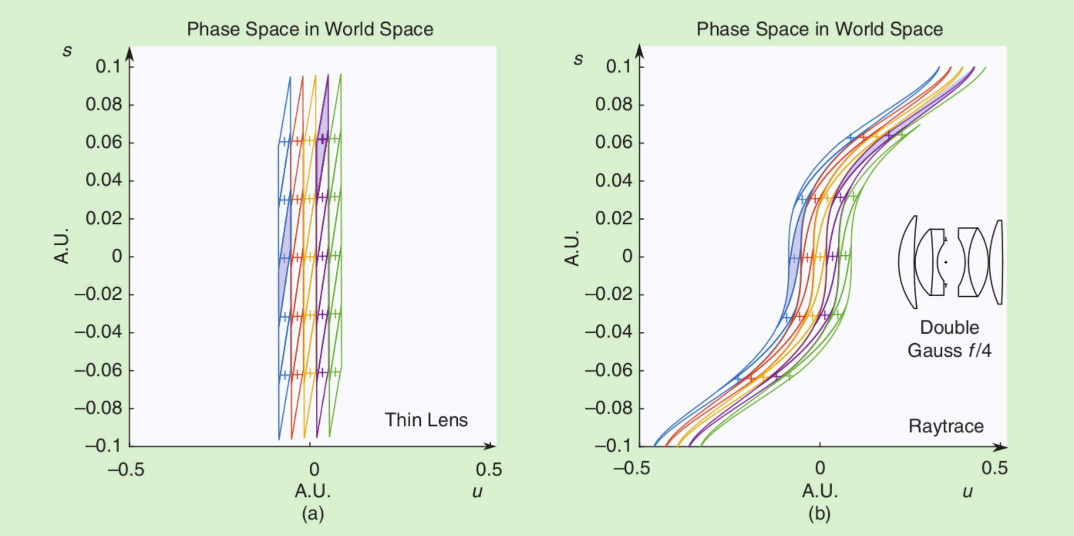
$$\begin{bmatrix} s \\ t \\ U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} h_{1,1} & 0 & h_{1,3} & 0 & h_{1,5} \\ 0 & h_{2,2} & 0 & h_{2,4} & h_{2,5} \\ h_{3,1} & 0 & h_{3,3} & 0 & h_{3,5} \\ 0 & h_{4,2} & 0 & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ l \\ 1 \end{bmatrix}$$

Even Less Simplified: Lens Distortion



Unrectified Illum image: distortion in u,v

Distortion is in 4D, not only 2D



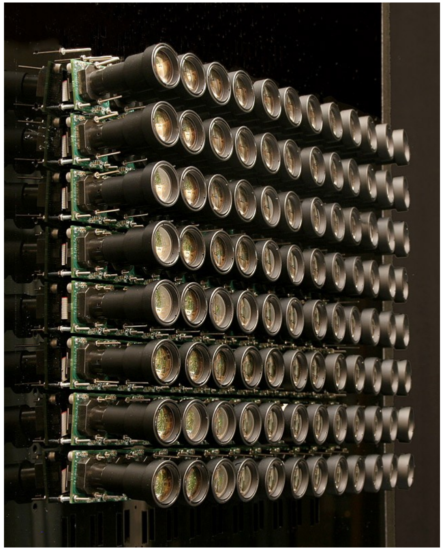
[Image c/o Ihrke et al 2016]

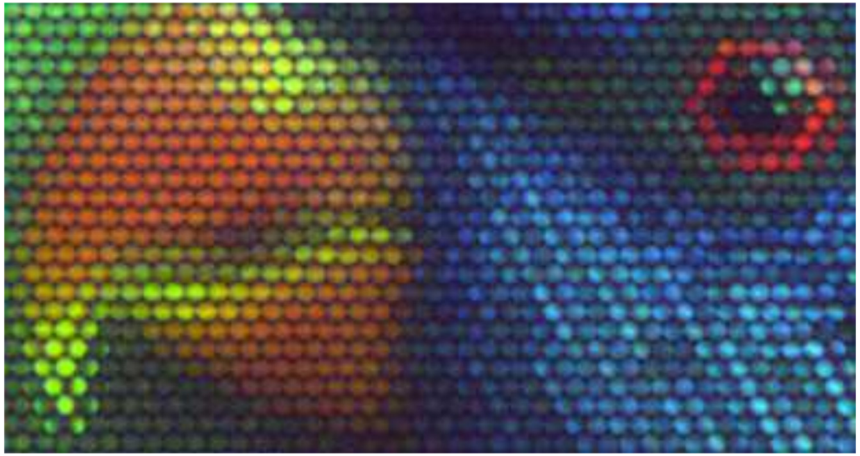
Solution: “rectify” (unwarp) images before using instrinsic model

Usually needs calibration to be accurate

Decoding, Calibration, Rectification

Decoding, Calibration, Rectification





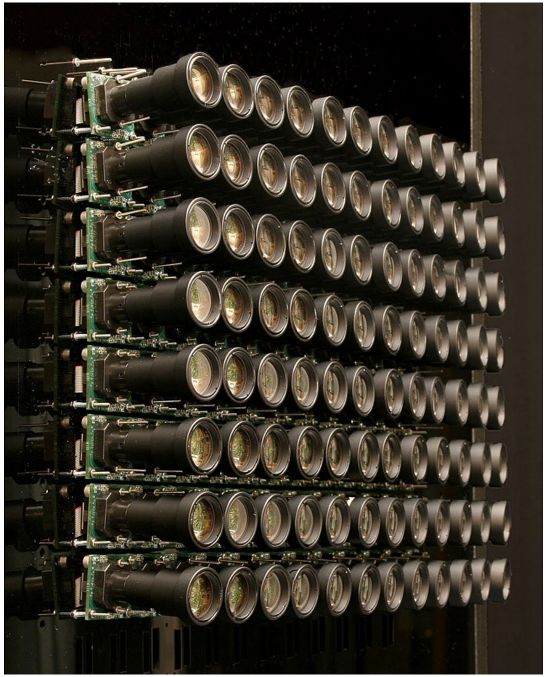
Decoding: Converting from raw input (2D in case of lenslet camera) to a 4D light field in a desired parameterization

Calibration: modelling the camera's intrinsics (and lens distortion) to map pixels to rays and vice-versa

Rectification: Undoing lens distortion

Depending on access to hardware and needs of a project, different combinations will be appropriate

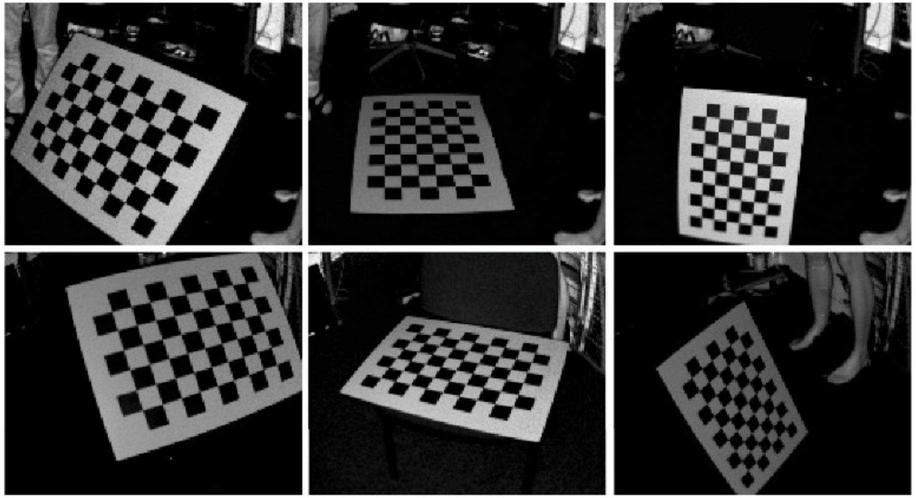
Camera Array: Full Metric Calibration



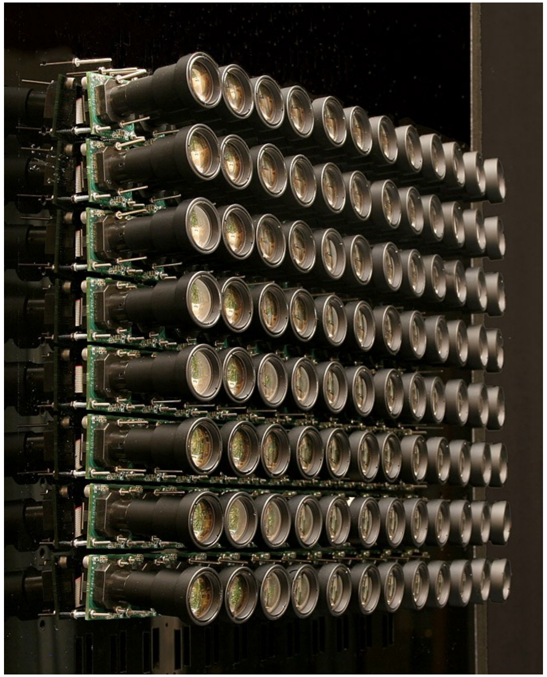
Per-camera intrinsics, extrinsics, lens distortion

Hard, but can use priors: close to grid; similar lenses

$$\begin{bmatrix} s \\ t \\ U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & h_{3,3} & 0 & h_{1,5} \\ 0 & 0 & 0 & h_{4,4} & h_{2,5} \\ h_{3,1} & 0 & h_{3,3} & 0 & h_{3,5} \\ 0 & h_{4,2} & 0 & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ l \\ 1 \end{bmatrix}$$



Camera Array: Non-Metric Calibration



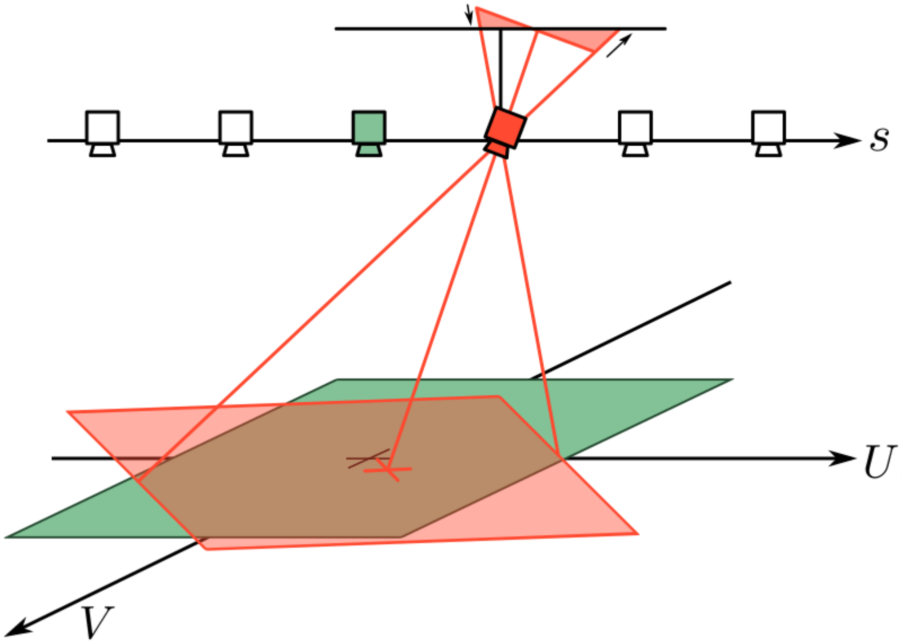
Assume rotation error dominates

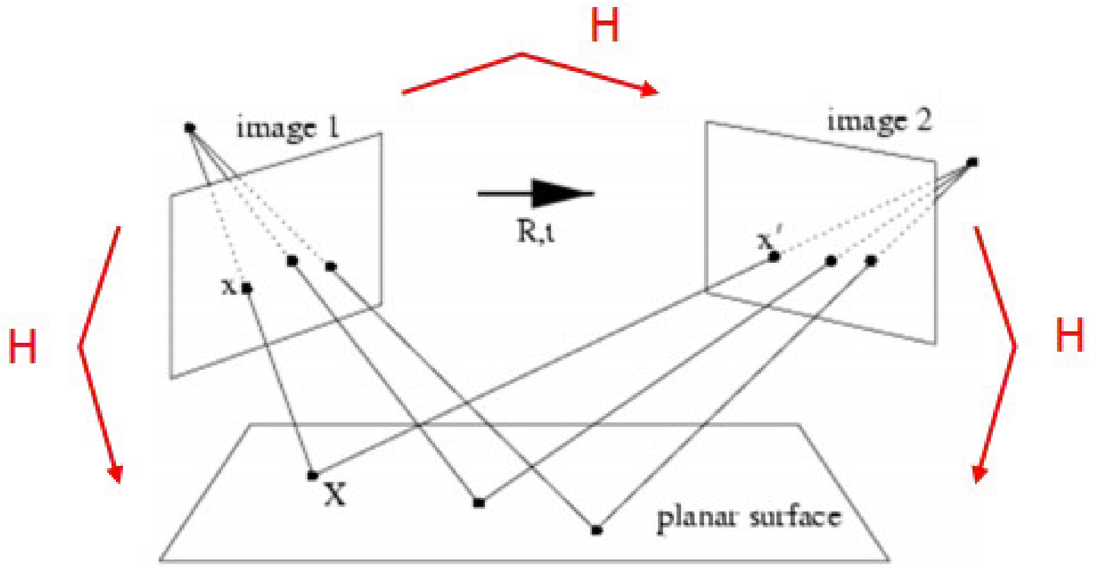
Co-register all images to a common view (e.g. center view)

At a specific depth plane: Homography

Animation: cycling through views of LF

Homographies for Array Registration

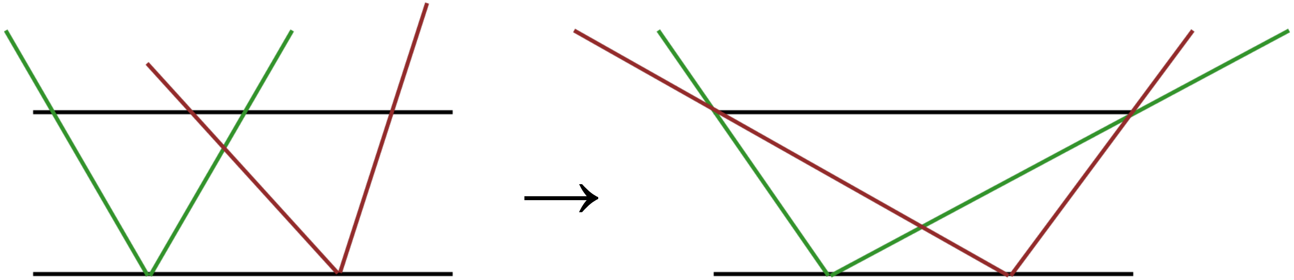




Pairwise homographies → Co-registered LF

Register each with center view

Camera Array: Non-Metric Calibration

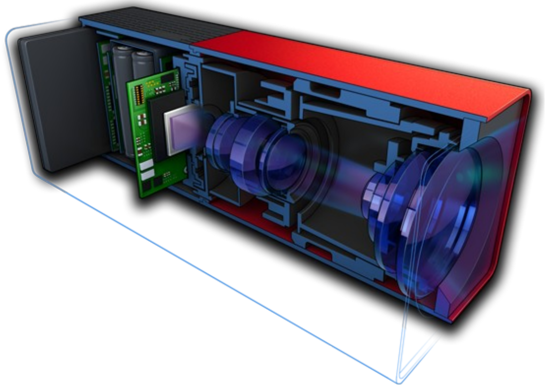


Errors in rotation get corrected

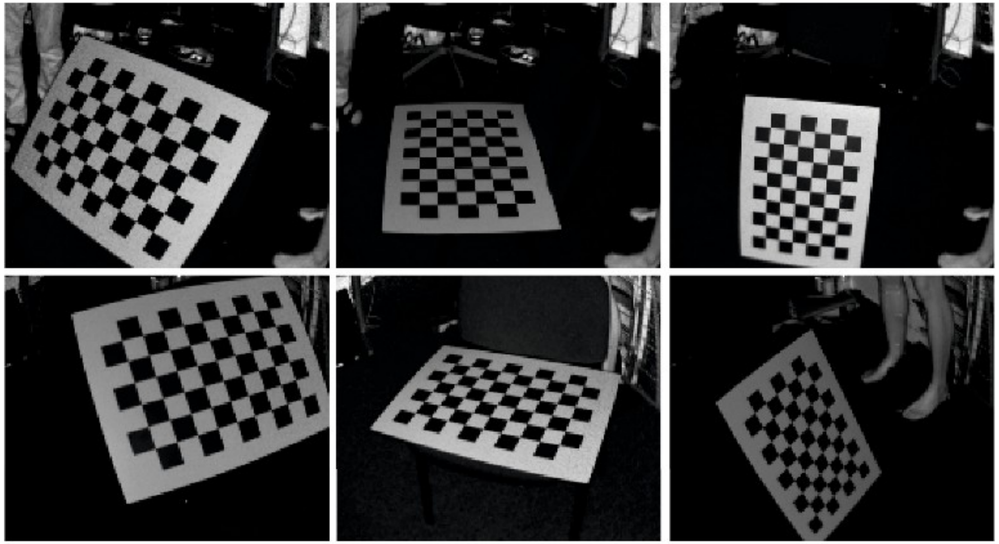
We can do a lot with the LF even without metric information

Lenslet Camera: Metric Calibration

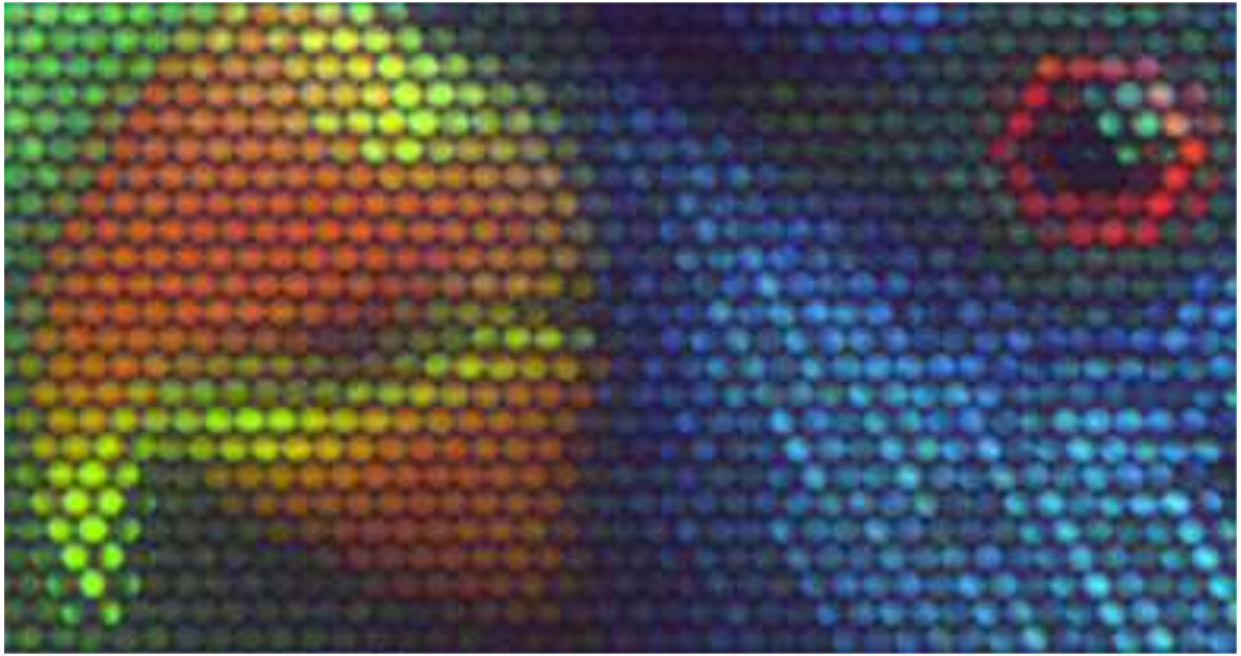
Extrinsics, intrinsics, radial lens distortion



$$\begin{bmatrix} s \\ t \\ U \\ V \\ 1 \end{bmatrix} = \begin{bmatrix} h_{1,1} & 0 & h_{1,3} & 0 & h_{1,5} \\ 0 & h_{2,2} & 0 & h_{2,4} & h_{2,5} \\ h_{3,1} & 0 & h_{3,3} & 0 & h_{3,5} \\ 0 & h_{4,2} & 0 & h_{4,4} & h_{4,5} \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} i \\ j \\ k \\ l \\ 1 \end{bmatrix}$$

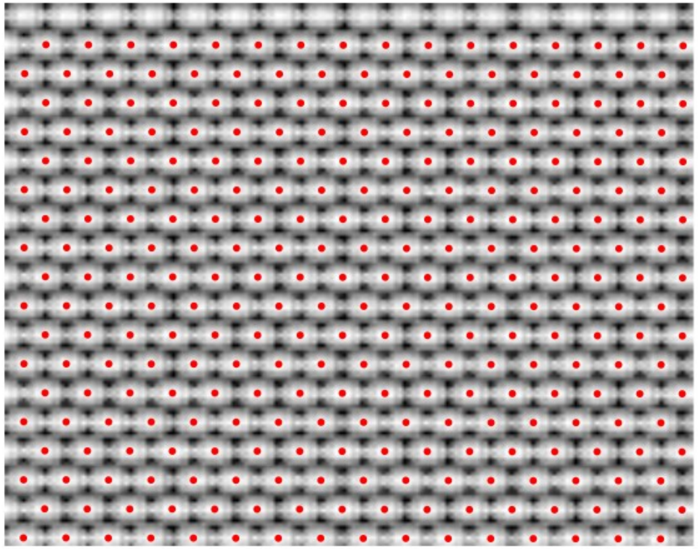


Lenslet Camera Decode Only (no calibration)

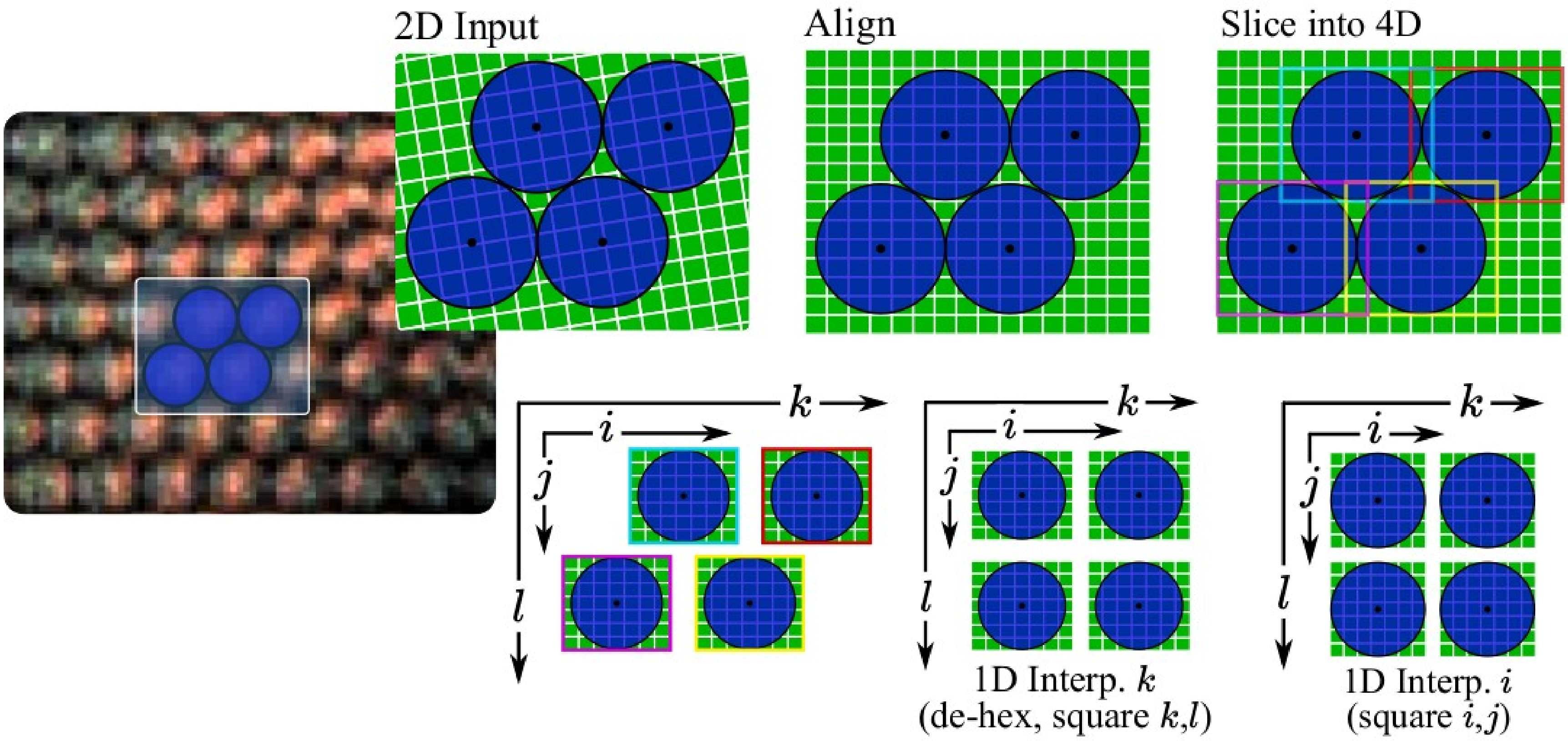


Raw lenslet image after 2D debayering and devignetting (cropped)

Flat-field white image (crop)



Decode from Lenslet Image to 4D



Example Decode (Light Field Toolbox, no rectification)

Gantry LF for comparison

[LF c/o Stanford]

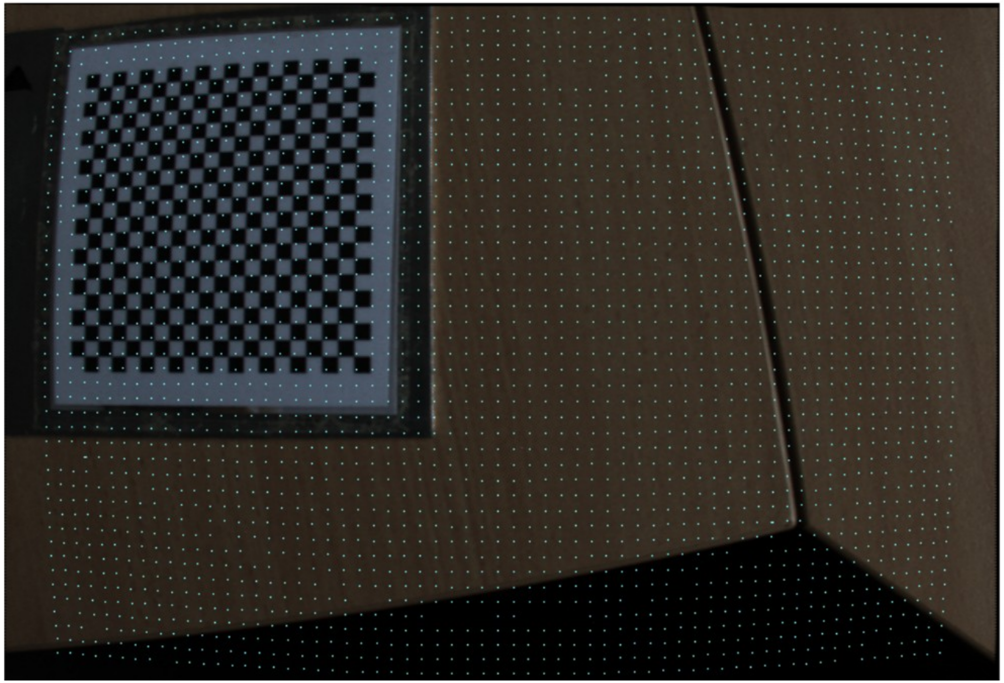
Animation: Lytro Illum image Decoded with LF Toolbox

Rectification without Calibration (Lytro Power Tools Beta, no H matrix)

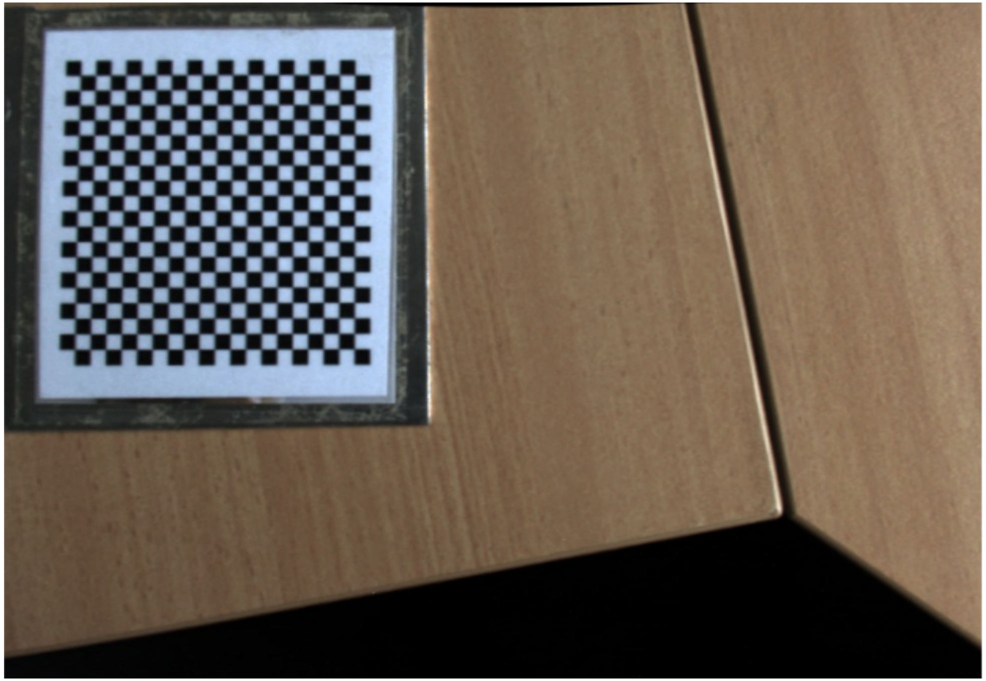
Animation: Lytro Power Tools rectified output

Uses an idealized lens model to undo lens distortion

Rectification with Calibration (Light Field Toolbox, Calibrated)



Unrectified, rectified samples overlaid



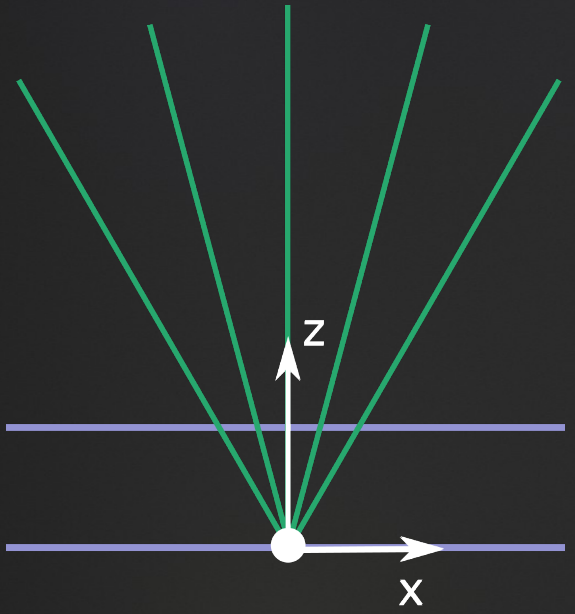
Calibrated rectification includes metric H

Lytro Illum image Decoded with LF Toolbox

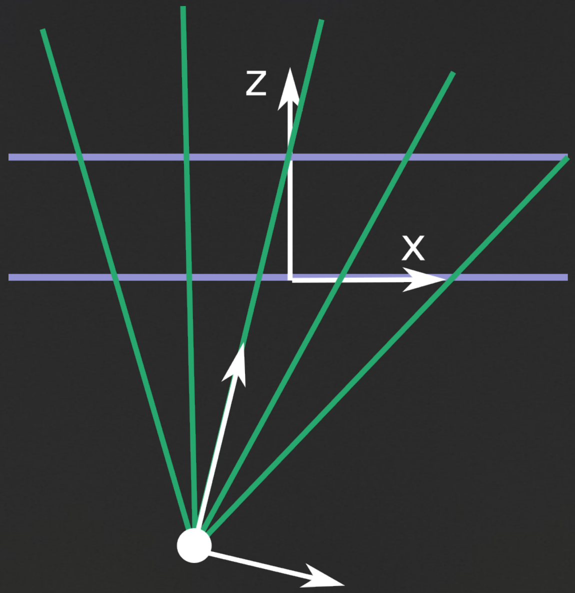
A Simple Renderer

A Simple Renderer

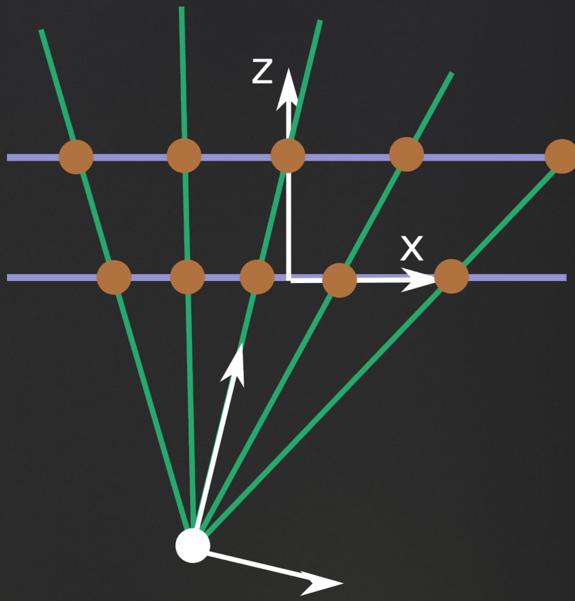
Given a light field render novel 2D perspective views



Describe camera's rays
As though at origin
Position + direction

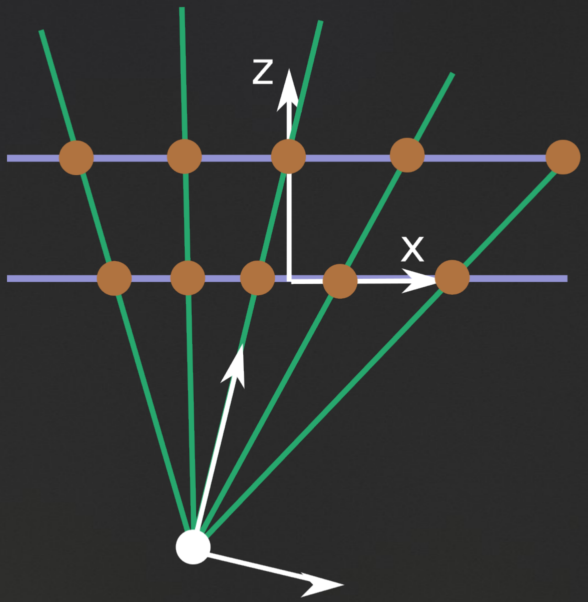


Transform each ray using camera's pose



Intersect each ray with s,t and u,v planes, yielding a list of rays Φ

A Simple Renderer



Convert the list of rays $\boldsymbol{\Phi}$ to a list of fractional indices **n**

$$\mathbf{n} = H^{-1}\,\boldsymbol{\Phi}$$

Interpolate to estimate the light field's value at ***n***

$$I = L(\boldsymbol{n})$$

Points to Ponder

When rendering, should we interpolate in u,v; in s,t; in s,u, … or … ?

How will aliasing affect the rendering?