Lecture 6 - Radial distortion & Bundle adjustment

DD2429

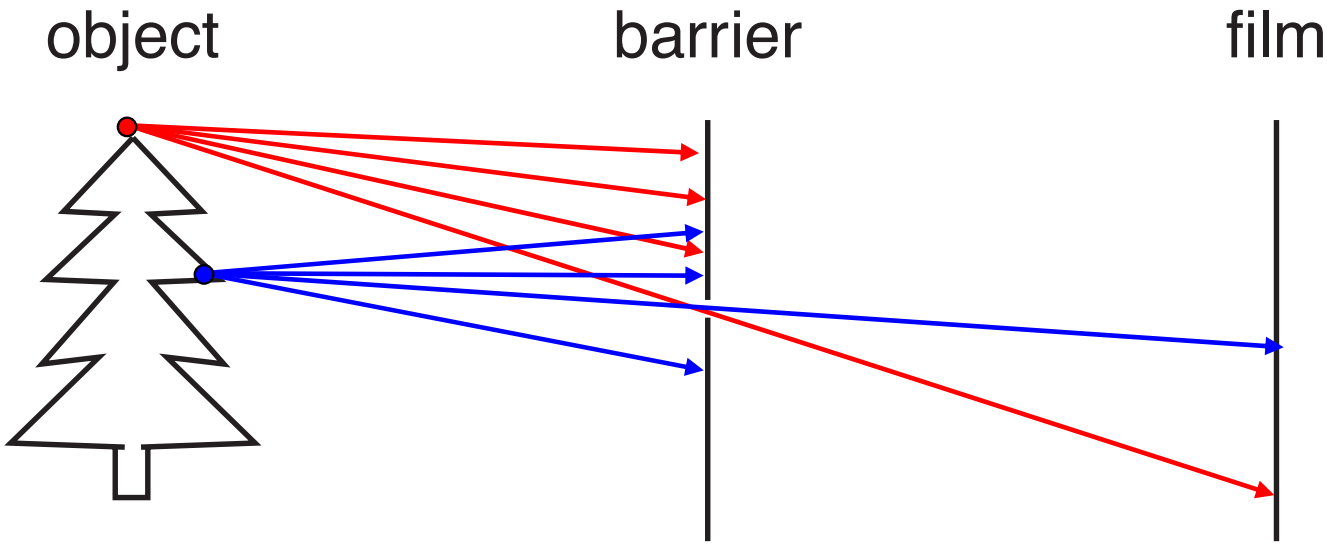
September 26, 2018

What's strange with this image?

Radial Distortion

Review of the imaging process

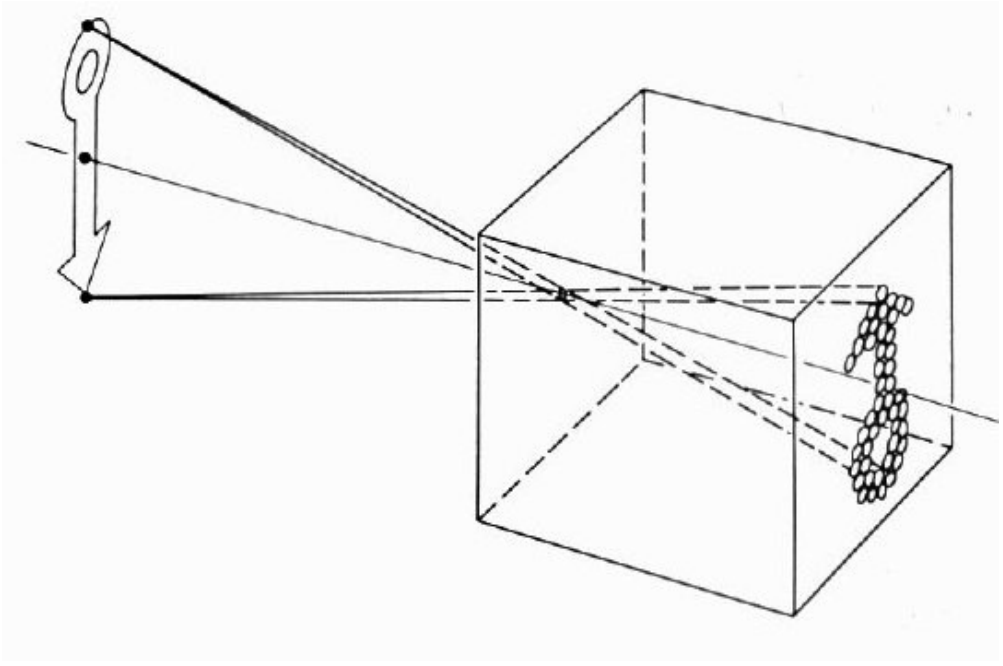
Pinhole Camera



Add a barrier to block off most of the rays coming from a point.

- The opening is known as the **aperture**.
- How does this imaging device transform the image?

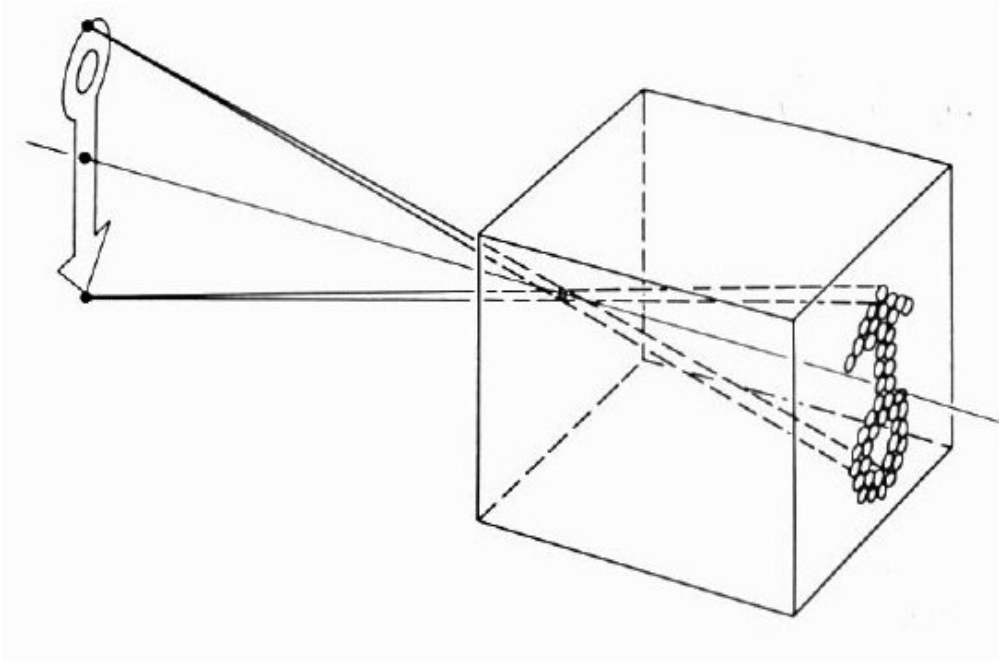
Camera Obscura



The first camera!

- Known to Aristotle.
- Analyzed by *Ibn al-Haytham* (Alhazen, 965-1039 AD) in Iraq

Camera Obscura

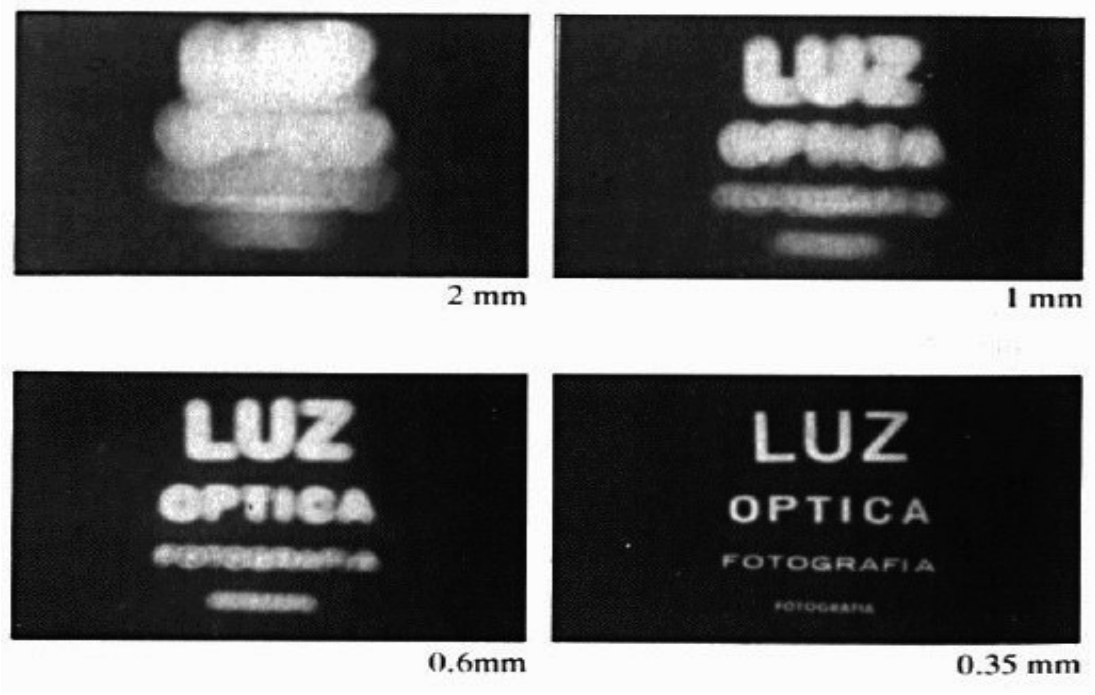


The first camera!

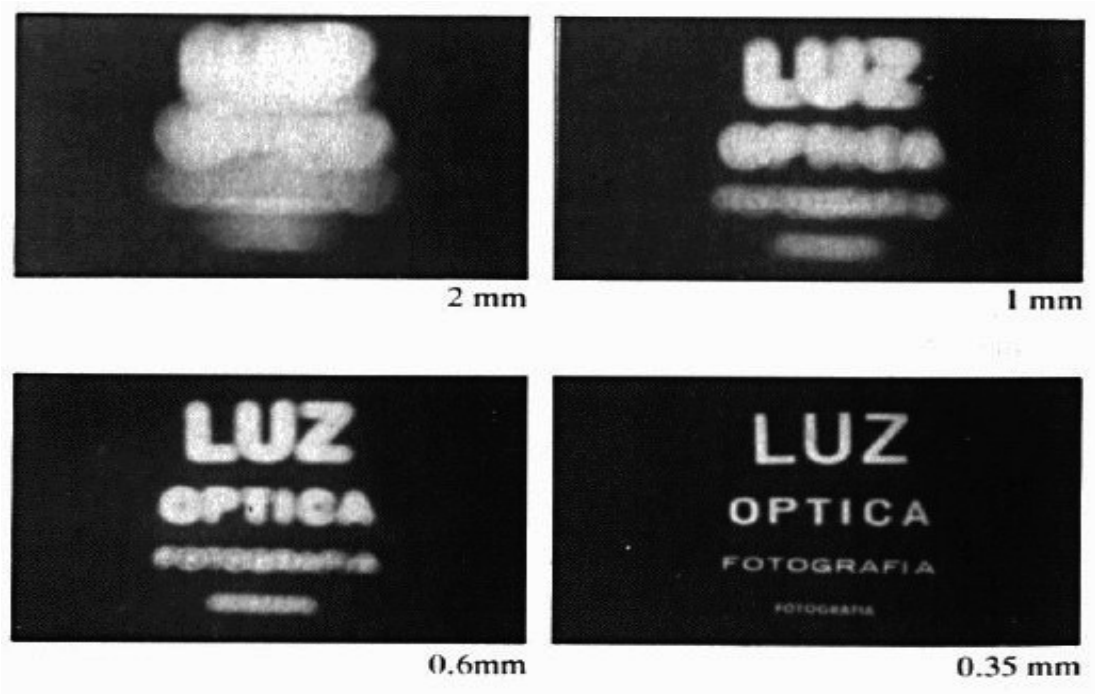
- Known to Aristotle.
- Analyzed by *Ibn al-Haytham* (Alhazen, 965-1039 AD) in Iraq

How does the aperture size affect the image?

Shrinking the aperture

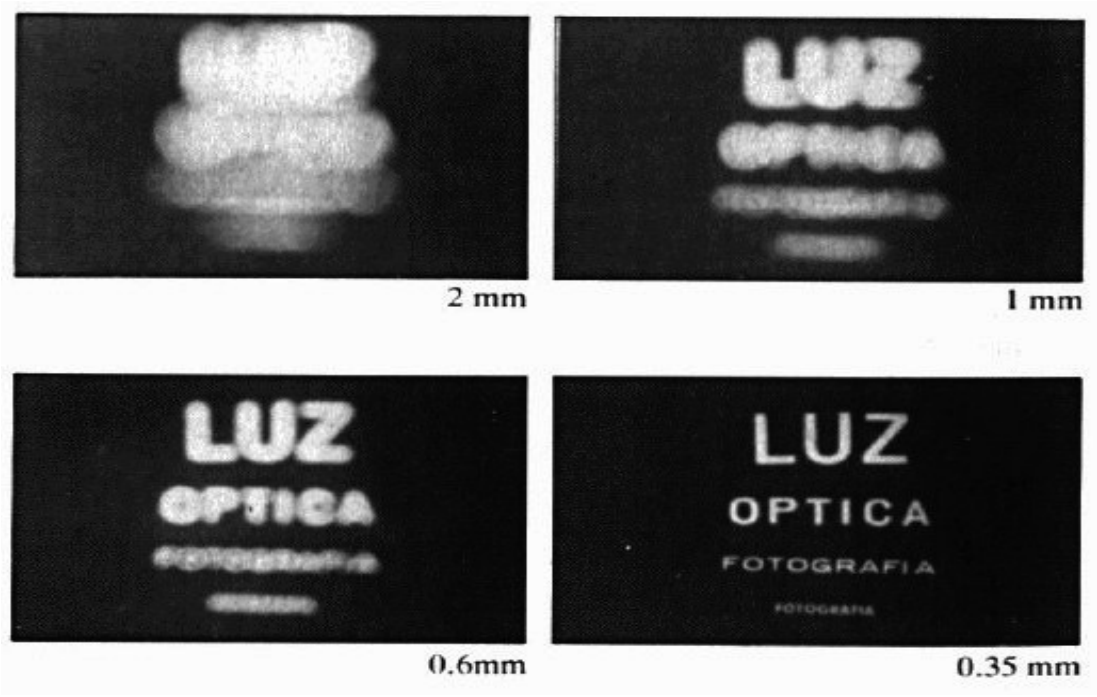


Shrinking the aperture



Why not make the aperture as small as possible?

Shrinking the aperture

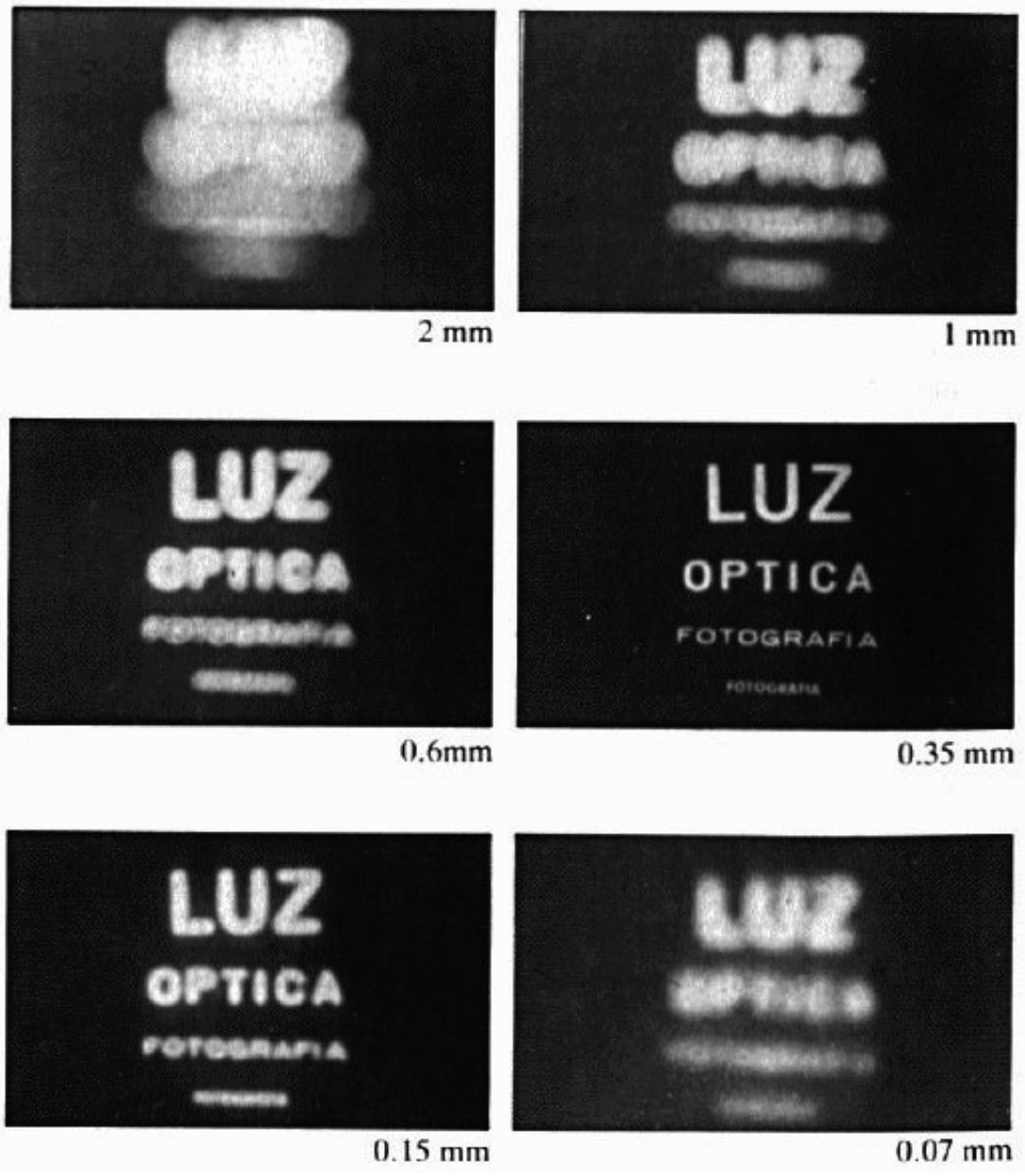


Why not make the aperture as small as possible?

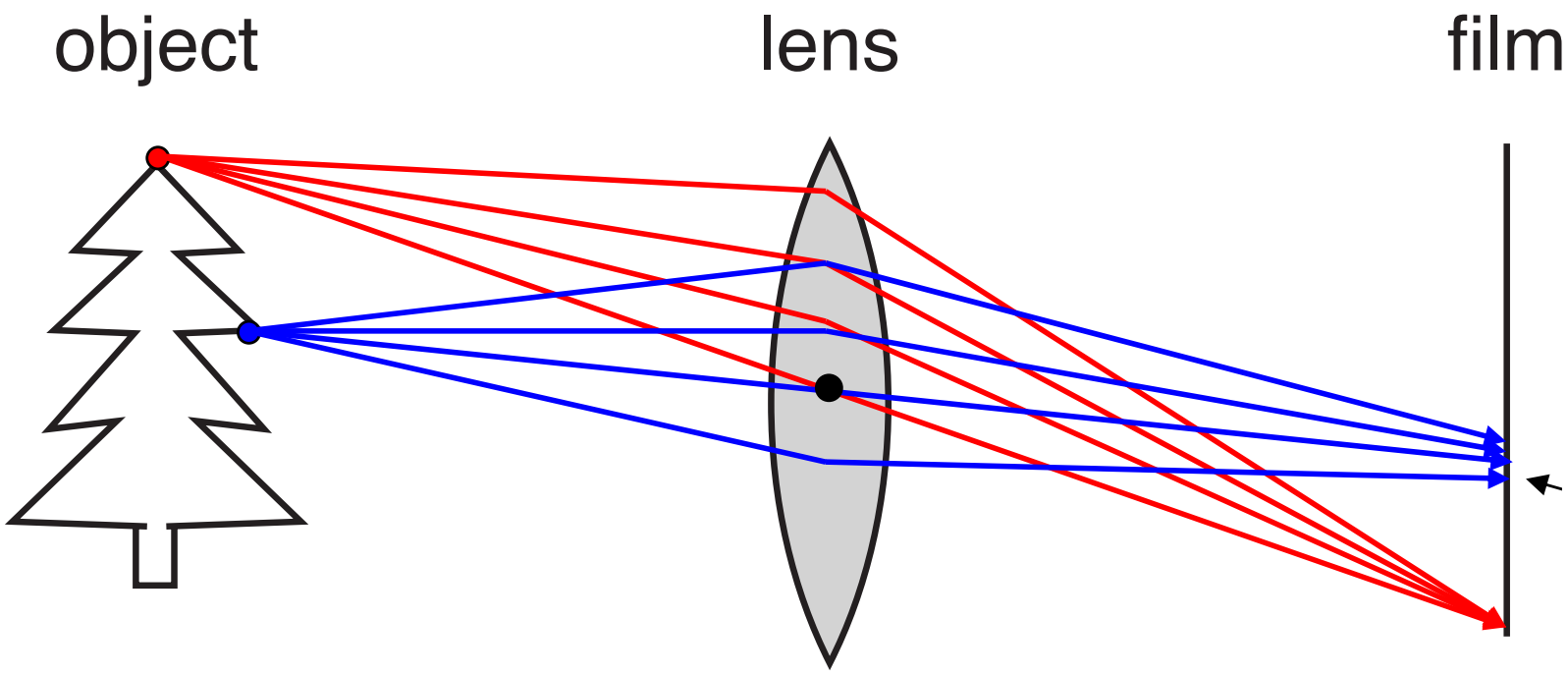
No

- Less light gets through.
- Diffraction effects . . .

Shrinking the aperture



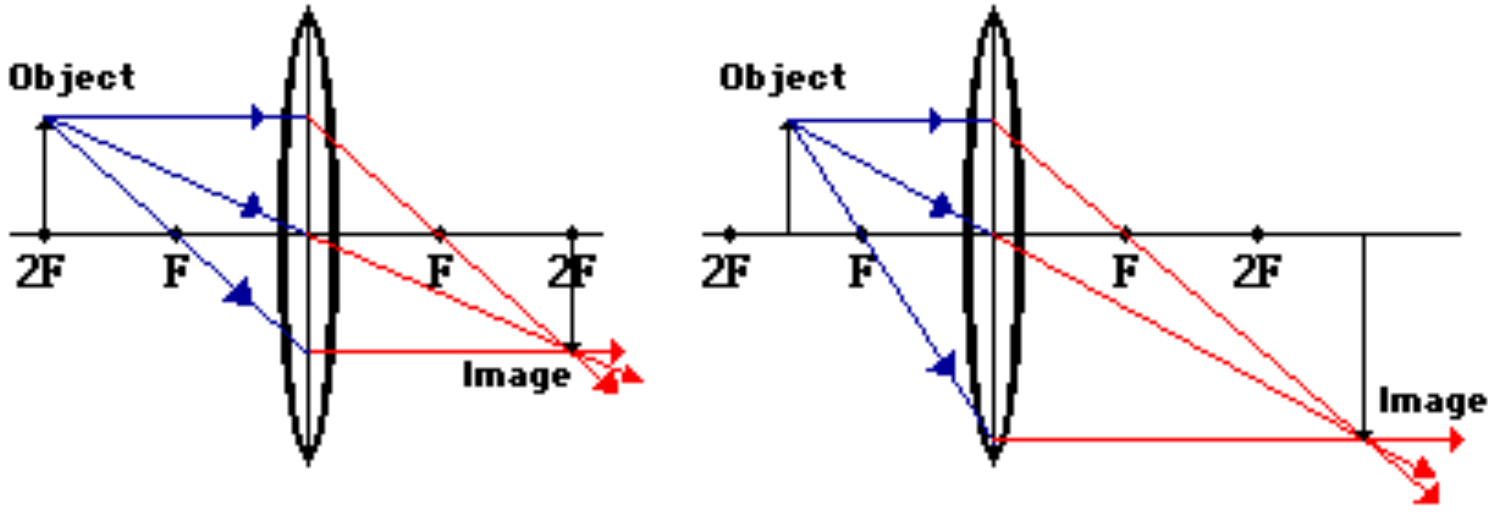
Add a lens to the imaging process



A lens focuses light onto the film.

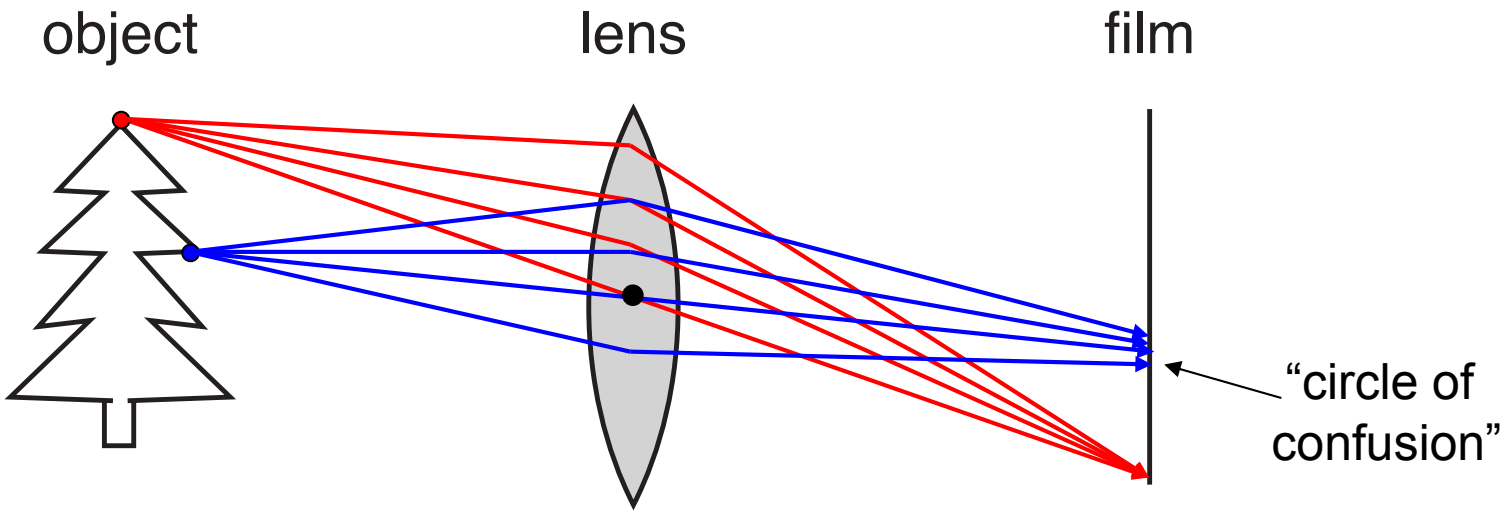
- Gather more light from each scene point.
- Lens improve image quality, leading to sharper images.

Quick review of converging lens



1. Ray parallel to the principal axis refracts through the lens and passes through the **focal point** on opposite side of lens.
2. Ray traveling through the **focal point** on way to lens refracts through the lens and travels parallel to the principal axis.
3. Ray passing through center of the lens continues in the same direction that it had when it entered the lens.

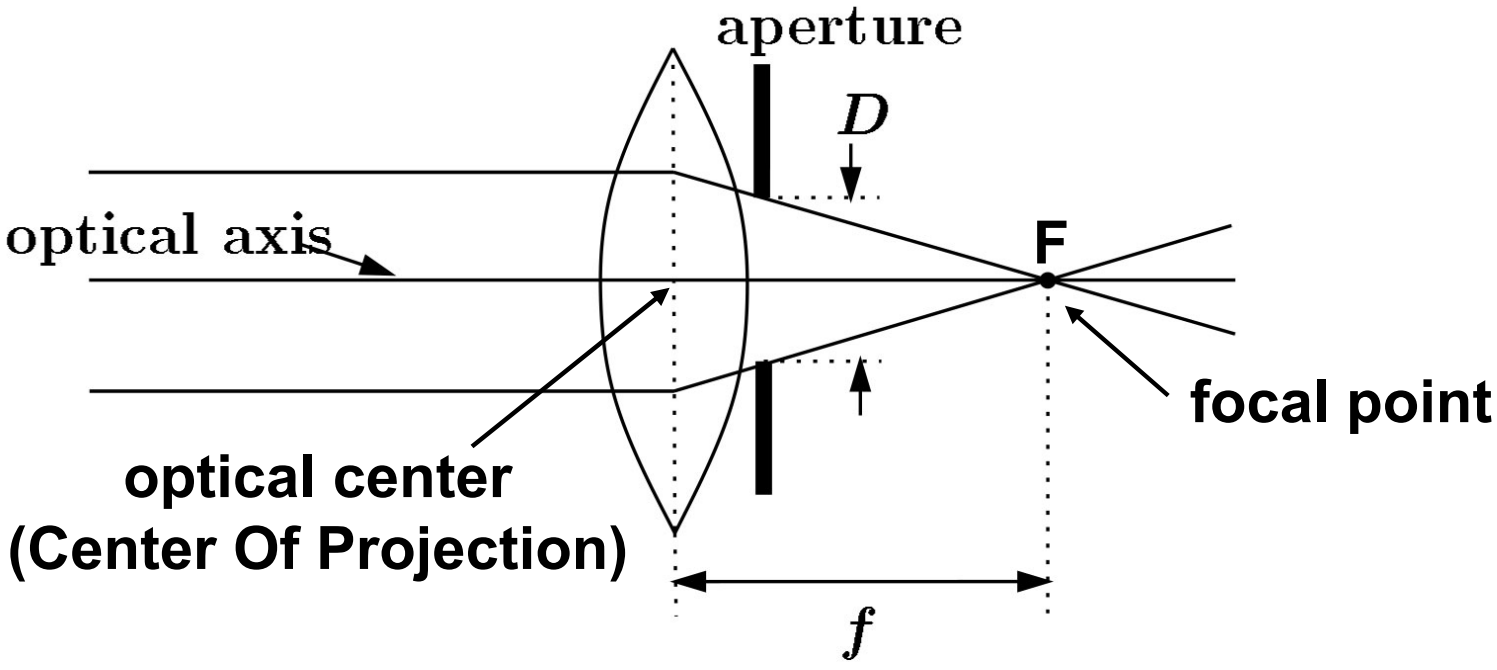
Adding a lens



A lens focuses light onto the film.

- There is a specific distance at which objects are **in focus** (i.e. when the film is aligned with where rays from a point intersect.)
- Points, not at this distance, project to a *circle of confusion* in the image.
- Changing the shape of the lens changes this distance.

Add an aperture to the lens

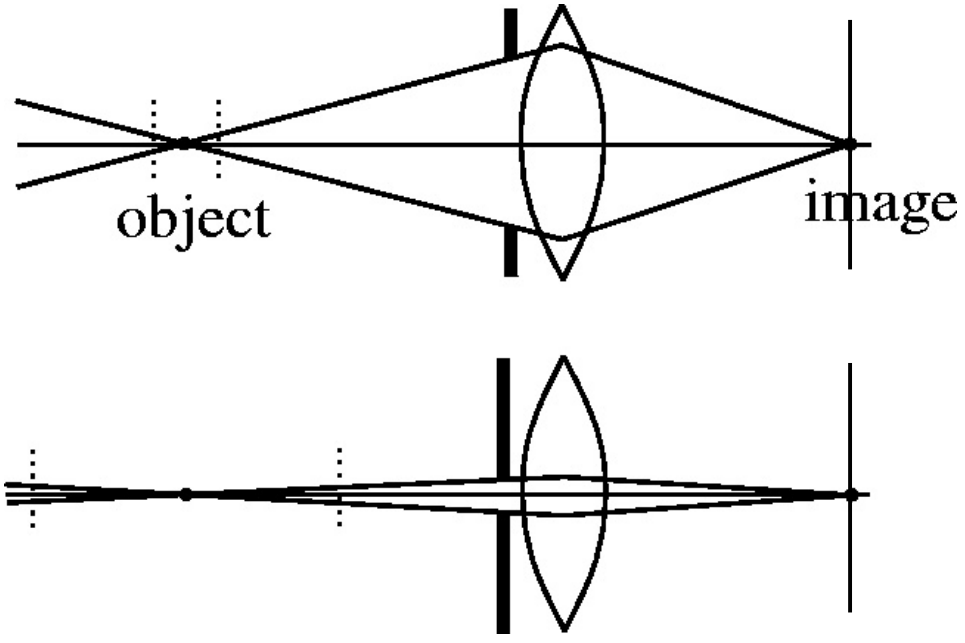


- Aperture of diameter D restricts the range of rays
- Aperture may be on either side of the lens.

Depth of field

The range of depths over which the world is approximately sharp (in focus).

Depth of field



f / 32

f / 5.6

Changing the aperture size affects depth of field

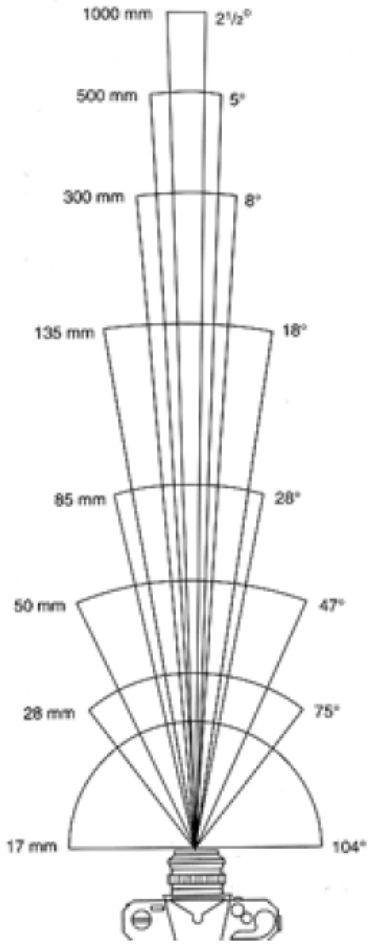
- Smaller aperture increases the range in which an object is approximately in focus. (Need to increase exposure time.)
- Larger aperture decreases the depth of field.(Can decrease exposure time.)

Varying the aperture

Large aperture = small DOF

Small aperture = large DOF

Field of View (Zoom)



17mm

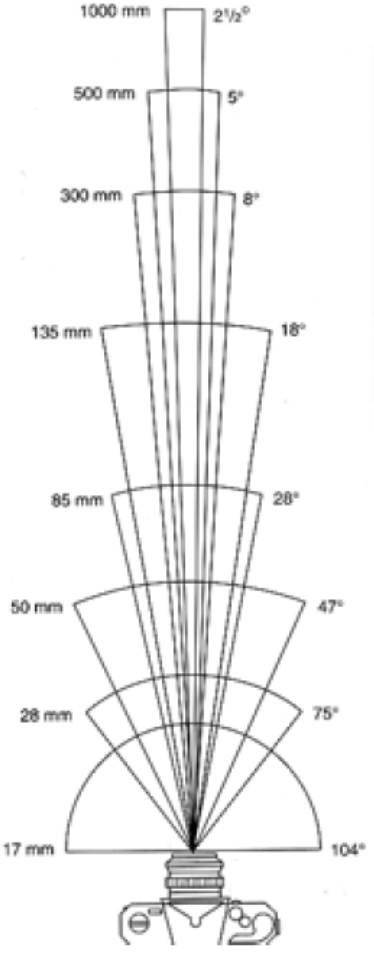
28mm

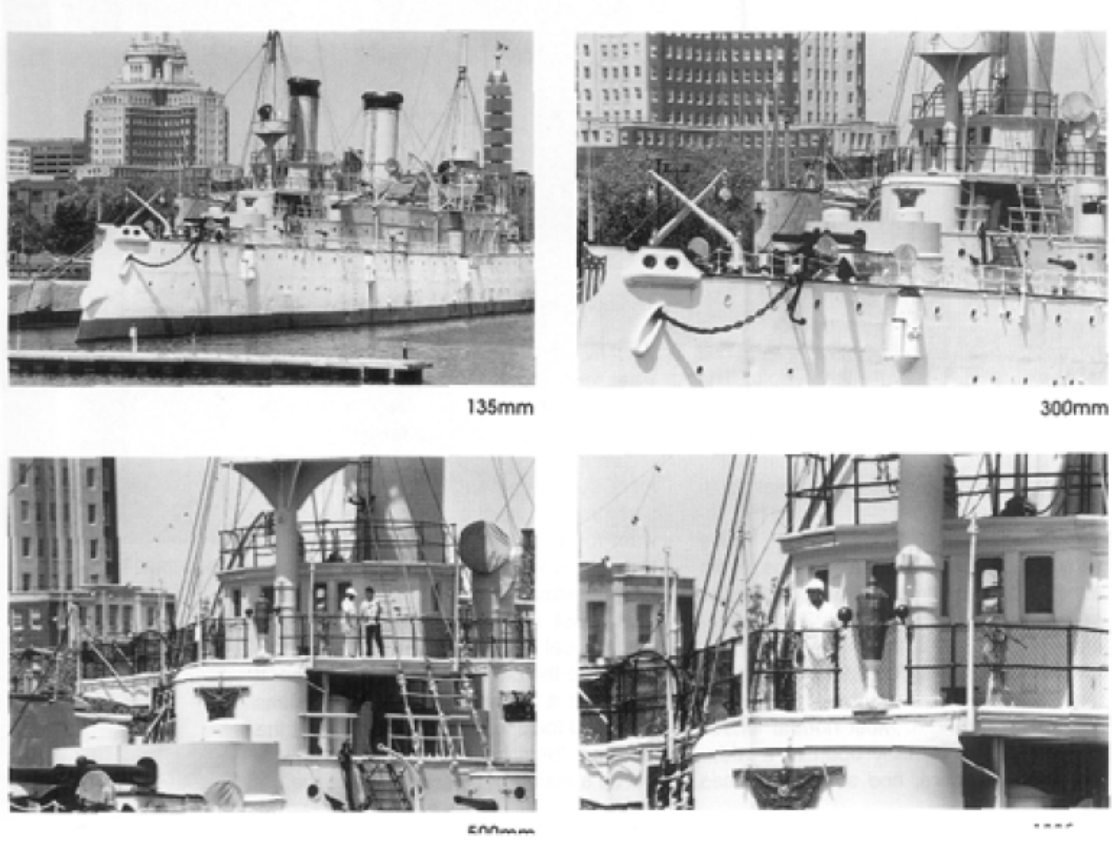
50mm

85mm

From London and Upton

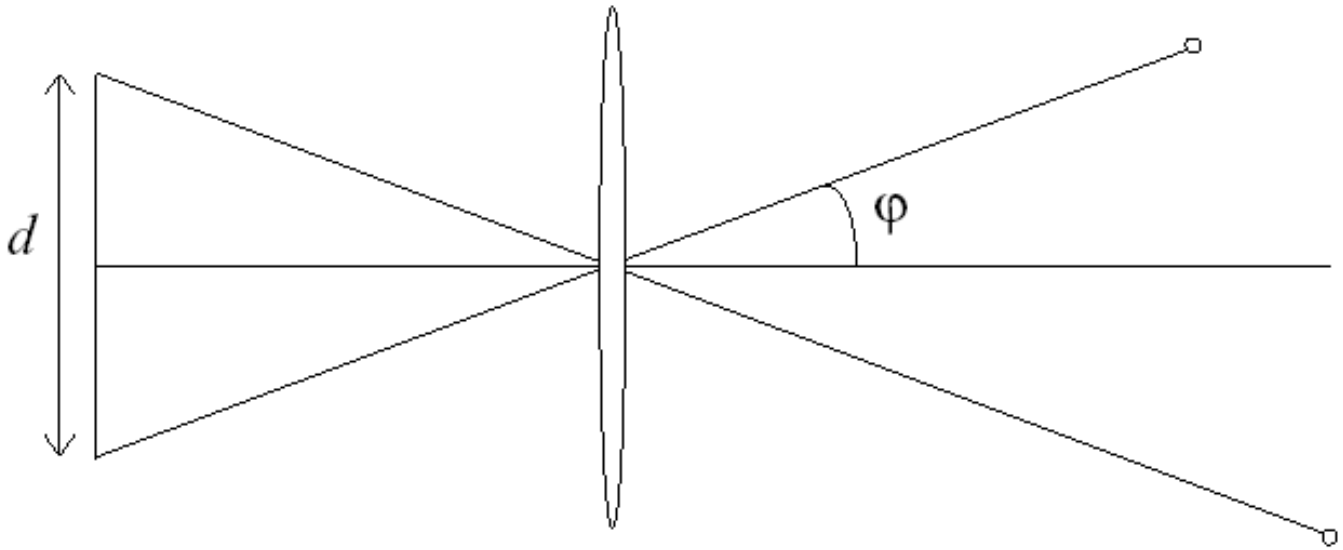
Field of View (Zoom)





From London and Upton

Field of View depends on focal length



Size of the field of view governed by size of the camera retina and focal length:

$$\phi = \tan^{-1}\left(\frac{d}{2f}\right)$$

Zooming and moving are not the same...

Large FOV, small f
Camera close to car

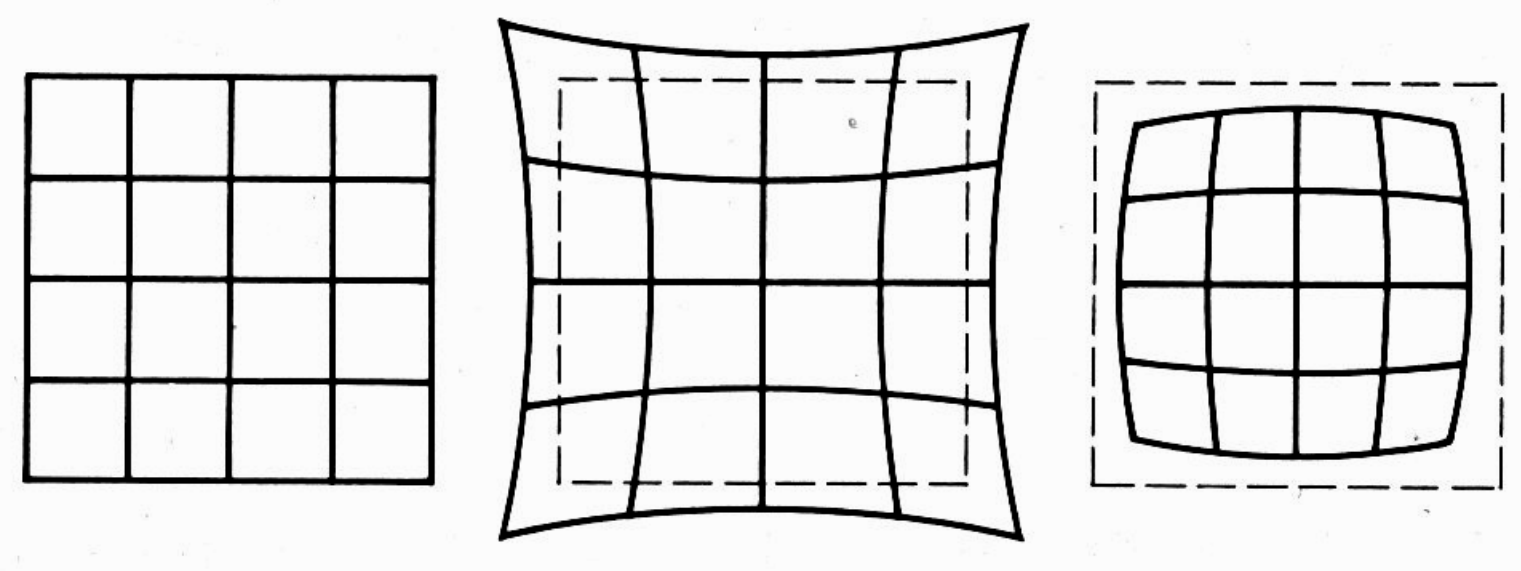
Small FOV, large f
Camera far from the car

Approximating a lens camera with a pinhole camera

- Most of the time it is fine.
- But there are some distortions due to the lens camera that we somtimes have to incorporate into the pinhole camera model.

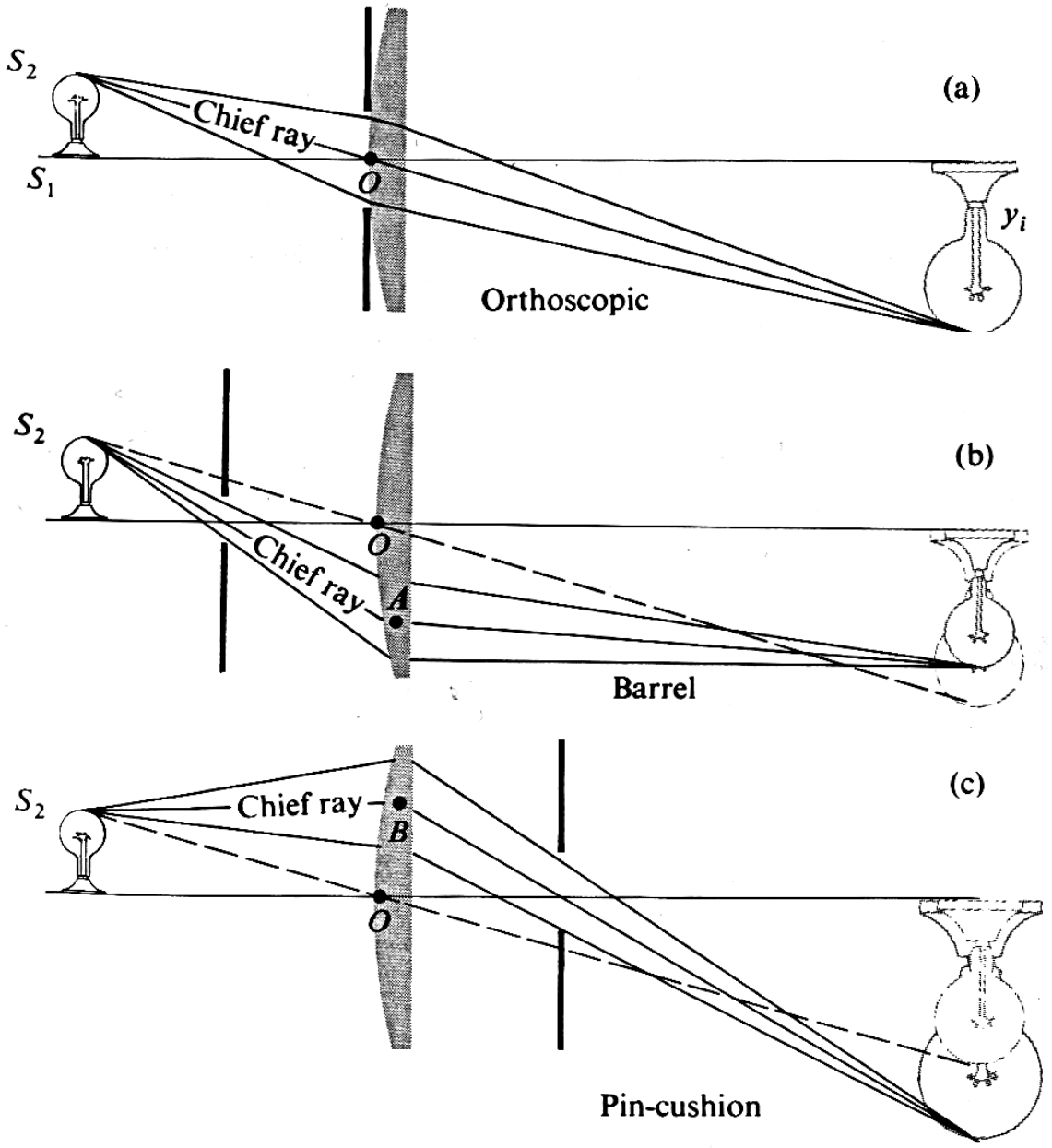
Radial Distortions

Radial distortion of the image



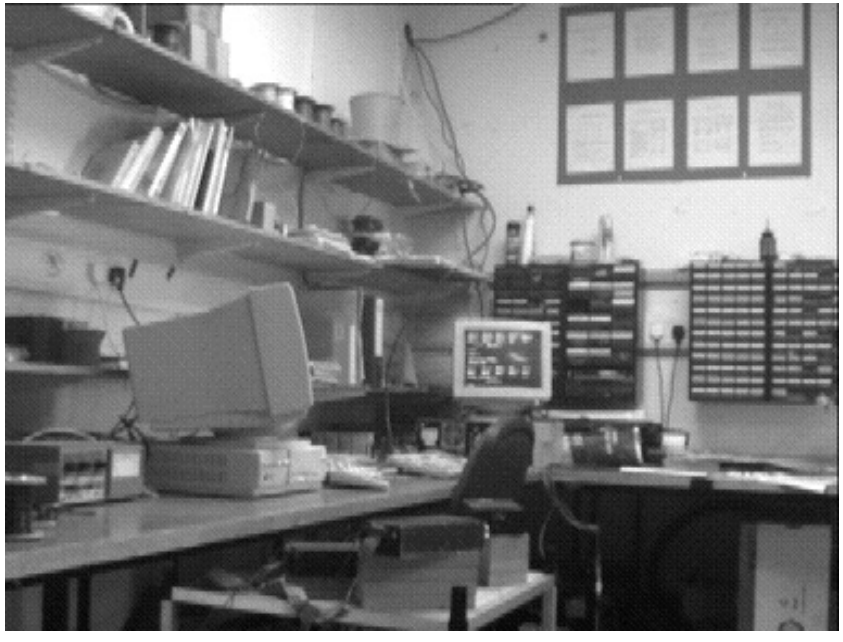
- Caused by imperfect lenses - its focal length is not constant across the lens.
- Deviations are most noticeable for rays that pass through the edge of the lens. (At image periphery.)

Distortion


S_2
Chief ray
S_1
O
(a)
y_i
Orthoscopic
S_2
O
Chief ray
A
(b)
Barrel
S_2
Chief ray
B
O
(c)
Pin-cushion

Radial distortion increases as focal length decreases

Short focal length



Long focal length

Modelling radial distortion

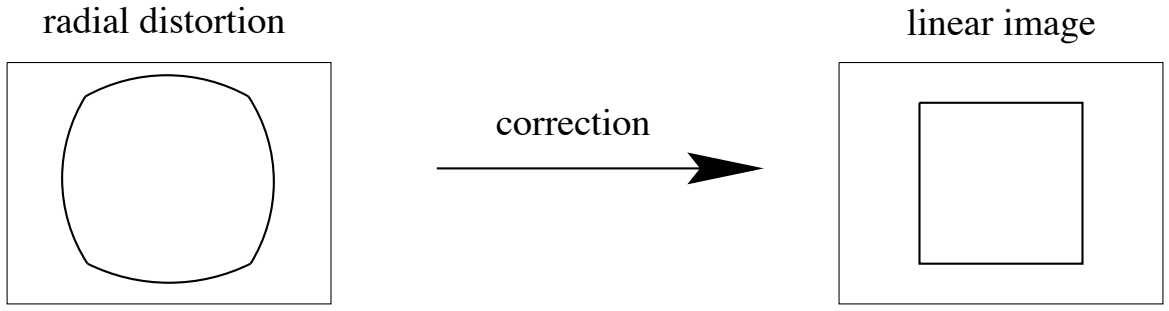
Radial distortion is modelled as

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = L(\tilde{r}) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

where

- $(\tilde{x}, \tilde{y})$ - the non-distorted image position
- (x_d, y_d) - actual image position after distortion
- $\tilde{r}$ - radial distance $\sqrt{(\tilde{x} - x_0)^2 + (\tilde{y} - y_0)^2}$ from the centre of radial distortion.
- $L(\tilde{r})$ - the distortion factor.

Correction of distortion



- Correction in pixel coordinates

$$\hat{x} = x_c + L(r)(x - x_c), \quad \hat{y} = y_c + L(r)(y - y_c)$$

where

- (x, y) - the measured coordinates (in the distorted image)
- $(\hat{x}, \hat{y})$ - corrected coordinates
- (x_c, y_c) - centre of radial distortion
- r - radial distance from the centre of radial distortion, $r^2 = (x - x_c)^2 + (y - y_c)^2$

Choice of the distortion function and centre

- Typically choose the distortion function to be

$$L(r) = 1 + \kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + \dots$$

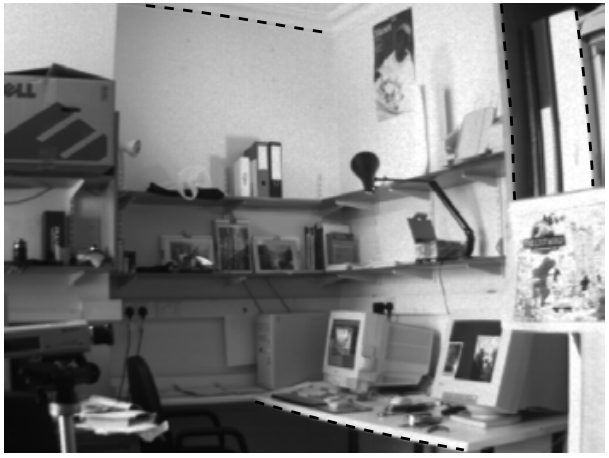
- The parameters needed for radial correction are

$$\kappa_1, \kappa_2, \kappa_3, \dots, x_c, y_c$$

  and are considered part of the camera internal calibration.

- **Principal point** is often used as the **centre for radial distortion**, though these need not coincide exactly.

Computing the distortion function I



- Enforce images of straight scene lines to be straight.
- Define a cost function on the imaged lines after apply radial distortion correction.
  - distance between the line joining the end-points and its midpoint
- Iteratively minimize cost function w.r.t. κ_i's and centre of the radial distortion.

Correcting a radially distorted image

Barrel distortion

Corrected

Projection matrix in the presence of radial distortion

- Can include radial distortion into the projection equation.
- Given 3D point $\mathbf{X}$ and camera projection matrix M:

$$\mathbf{x} \simeq \begin{pmatrix} \frac{1}{\lambda} & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{pmatrix} M\mathbf{X}$$

where

- If $u = x/z$ and $v = y/z$ with $\mathbf{x} = (x, y, z)^T$ then

$$r^2 = au^2 + bv^2 + cuv \quad \leftarrow \text{models the radial behaviour}$$

a, b, c depend on the centre of distortion and the camera internals.

- the **distortion factor** is modelled as

$$\lambda = 1 + \sum_{p=1}^{3} \kappa_p r^{2p}$$

Projection matrix in the presence of radial distortion

- Given 3D point $\mathbf{X}$ and camera projection matrix M:

$$\mathbf{x} \simeq \begin{pmatrix} \frac{1}{\lambda} & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{pmatrix} M\mathbf{X} = Q\mathbf{X}$$

  where

$$Q = \begin{pmatrix} \frac{1}{\lambda} & 0 & 0 \\ 0 & \frac{1}{\lambda} & 0 \\ 0 & 0 & 1 \end{pmatrix} M = \begin{pmatrix} \mathbf{m}_1^T/\lambda \\ \mathbf{m}_2^T/\lambda \\ \mathbf{m}_3^T \end{pmatrix}$$

- Thus

$$\mathbf{x} \simeq Q\mathbf{X} = \begin{pmatrix} \mathbf{m}_1^T\mathbf{X}/\lambda \\ \mathbf{m}_2^T\mathbf{X}/\lambda \\ \mathbf{m}_3^T\mathbf{X} \end{pmatrix}$$

Projection matrix in the presence of radial distortion

- Have

$$\mathbf{x} \simeq Q\mathbf{X} = \begin{pmatrix} \mathbf{m}_1^T\mathbf{X}/\lambda \\ \mathbf{m}_2^T\mathbf{X}/\lambda \\ \mathbf{m}_3^T\mathbf{X} \end{pmatrix}$$

$$\Longrightarrow$$

$$u = \frac{\mathbf{m}_1^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}} \qquad v = \frac{\mathbf{m}_2^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}$$

$$\Longrightarrow$$

$$u\lambda\mathbf{m}_3^T\mathbf{X} - \mathbf{m}_1^T\mathbf{X} = 0$$
$$v\lambda\mathbf{m}_3^T\mathbf{X} - \mathbf{m}_2^T\mathbf{X} = 0$$

Projection matrix in the presence of radial distortion

- Have

$$\mathbf{x} \simeq Q\mathbf{X} = \begin{pmatrix} \mathbf{m}_1^T\mathbf{X}/\lambda \\ \mathbf{m}_2^T\mathbf{X}/\lambda \\ \mathbf{m}_3^T\mathbf{X} \end{pmatrix}$$

$$\Longrightarrow$$

$$u = \frac{\mathbf{m}_1^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}} \qquad v = \frac{\mathbf{m}_2^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}$$

$$\Longrightarrow$$

$$u\lambda\mathbf{m}_3^T\mathbf{X} - \mathbf{m}_1^T\mathbf{X} = 0$$
$$v\lambda\mathbf{m}_3^T\mathbf{X} - \mathbf{m}_2^T\mathbf{X} = 0$$

Are these two equations linear w.r.t. projection matrix parameters $\mathbf{m} = (\mathbf{m}_1^T, \mathbf{m}_2^T, \mathbf{m}_3^T)^T$ and distortion parameters $\kappa_1, \kappa_2, \kappa_3$?

Projection matrix in the presence of radial distortion

- Have

$$\mathbf{x} \simeq Q\mathbf{X} = \begin{pmatrix} \mathbf{m}_1^T\mathbf{X}/\lambda \\ \mathbf{m}_2^T\mathbf{X}/\lambda \\ \mathbf{m}_3^T\mathbf{X} \end{pmatrix}$$

$$\Longrightarrow$$

$$u = \frac{\mathbf{m}_1^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}} \qquad v = \frac{\mathbf{m}_2^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}$$

$$\Longrightarrow$$

$$u\lambda\mathbf{m}_3^T\mathbf{X} - \mathbf{m}_1^T\mathbf{X} = 0$$
$$v\lambda\mathbf{m}_3^T\mathbf{X} - \mathbf{m}_2^T\mathbf{X} = 0$$

Are these two equations linear w.r.t. projection matrix parameters $\mathbf{m} = (\mathbf{m}_1^T, \mathbf{m}_2^T, \mathbf{m}_3^T)^T$ and distortion parameters $\kappa_1, \kappa_2, \kappa_3$? **NO**

General Camera Calibration Problem with distortion

- Assume have known correspondences between 2D and 3D points that is $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ for $i = 1, \ldots, n$.

- Calibrate the camera by solving this non-linear optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} \left[(u_i - f_u(\mathbf{X}_i, \Theta))^2 + (v_i - f_v(\mathbf{X}_i, \Theta))^2 \right]$$

  where

  1. $\Theta = (M, \kappa_1, \kappa_2, \kappa_3)$ and
  2. f_u, f_v are the non-linear functions projecting a 3D point to its image coordinates.

General Camera Calibration Problem with distortion

- Calibrate the camera by solving this non-linear optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} \left[(u_i - f_u(\mathbf{X}_i, \Theta))^2 + (v_i - f_v(\mathbf{X}_i, \Theta))^2 \right]$$

- Can solve the above optimization problem with
  1. Newton Method or
  2. Levenberg-Marquardt Algorithm

General Camera Calibration Problem with distortion

- Calibrate the camera by solving this non-linear optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} \left[(u_i - f_u(\mathbf{X}_i, \Theta))^2 + (v_i - f_v(\mathbf{X}_i, \Theta))^2 \right]$$

- Can solve the above optimization problem with
  1. Newton Method or
  2. Levenberg-Marquardt Algorithm

- Pros and trade-offs of Levenberg-Marquardt approach
  - Iterative, starts from initial solution.
  - May be slow if initial solution far from real solution.
  - Estimated solution may be function of the initial solution.
  - Newton requires the computation of Jacobian and Hessian.
  - Levenberg-Marquardt doesn't require computing the Hessian.

General Camera Calibration Problem with distortion

- Calibrate the camera by solving this non-linear optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} \left[(u_i - f_u(\mathbf{X}_i, \Theta))^2 + (v_i - f_v(\mathbf{X}_i, \Theta))^2 \right]$$

- **A plausible full optimization algorithm**
  1. Solve the linear system using n correspondences and assuming no distortion.
  2. Use this solution as initial condition for the full system.
  3. Solve full system using Newton or L.M.

General Camera Calibration Problem with distortion

- Calibrate the camera by solving this non-linear optimization problem:

$$\Theta^* = \arg\min_{\Theta} \sum_{i=1}^{n} \left[(u_i - f_u(\mathbf{X}_i, \Theta))^2 + (v_i - f_v(\mathbf{X}_i, \Theta))^2 \right]$$

- Typical assumptions made to make optimization/calibration easier:
  1. zero-skew,
  2. square pixels,
  3. known image center.

Incorporate distortion but with simpler optimization

- Have

$$\mathbf{x} \simeq Q\mathbf{X} = \begin{pmatrix} \mathbf{m}_1^T\mathbf{X}/\lambda \\ \mathbf{m}_2^T\mathbf{X}/\lambda \\ \mathbf{m}_3^T\mathbf{X} \end{pmatrix}$$

$$\Longrightarrow$$

$$u = \frac{\mathbf{m}_1^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}, \qquad v = \frac{\mathbf{m}_2^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}$$

- Can we estimate $\mathbf{m}_1$ and $\mathbf{m}_2$ and ignore the radial distortion?

Incorporate distortion but with simpler optimization

- Have

$$\mathbf{x} \simeq Q\mathbf{X} = \begin{pmatrix} \mathbf{m}_1^T\mathbf{X}/\lambda \\ \mathbf{m}_2^T\mathbf{X}/\lambda \\ \mathbf{m}_3^T\mathbf{X} \end{pmatrix}$$

$$\Longrightarrow$$

$$u = \frac{\mathbf{m}_1^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}, \qquad v = \frac{\mathbf{m}_2^T\mathbf{X}}{\lambda\mathbf{m}_3^T\mathbf{X}}$$

- Can we estimate $\mathbf{m}_1$ and $\mathbf{m}_2$ and ignore the radial distortion?

- Yes! Consider for each match $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$

$$\frac{u_i}{v_i} = \frac{\mathbf{m}_1\mathbf{X}_i}{\mathbf{m}_2\mathbf{X}_i}$$

Incorporate distortion but have linear optimization

- For each match $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ have

$$\frac{u_i}{v_i} = \frac{\mathbf{m}_1 \mathbf{X}_i}{\mathbf{m}_2 \mathbf{X}_i}$$

- For all matches have

$$\begin{aligned} v_1(\mathbf{m}_1\mathbf{X}_1) - u_1(\mathbf{m}_2\mathbf{X}_1) &= 0 \\ v_2(\mathbf{m}_1\mathbf{X}_2) - u_2(\mathbf{m}_2\mathbf{X}_2) &= 0 \\ &\vdots \\ v_n(\mathbf{m}_1\mathbf{X}_n) - u_2(\mathbf{m}_2\mathbf{X}_n) &= 0 \end{aligned}$$

- Write this is matrix notation

$$L\mathbf{n} = \mathbf{0}$$

with $\mathbf{n} = (\mathbf{m}_1^T, \mathbf{m}_2^T)^T$ and $L = ...$

- Find $\mathbf{n}$ from the SVD of L which solves the least squares problem.

Incorporate distortion but have linear optimization

- For each match $\mathbf{x}_i \leftrightarrow \mathbf{X}_i$ have

$$\frac{u_i}{v_i} = \frac{\mathbf{m}_1 \mathbf{X}_i}{\mathbf{m}_2 \mathbf{X}_i}$$

- For all matches have

$$\begin{aligned} v_1(\mathbf{m}_1 \mathbf{X}_1) - u_1(\mathbf{m}_2 \mathbf{X}_1) &= 0 \\ v_2(\mathbf{m}_1 \mathbf{X}_2) - u_2(\mathbf{m}_2 \mathbf{X}_2) &= 0 \\ &\vdots \\ v_n(\mathbf{m}_1 \mathbf{X}_n) - u_2(\mathbf{m}_2 \mathbf{X}_n) &= 0 \end{aligned}$$

- Write this is matrix notation

$$L\mathbf{n} = \mathbf{0}$$

  with $\mathbf{n} = (\mathbf{m}_1^T, \mathbf{m}_2^T)^T$ and $L = ...$

- Find $\mathbf{n}$ from the SVD of L which solves the least squares problem.

Finishing off the projection matrix optimization

- Have estimated $\mathbf{m}_1$ and $\mathbf{m}_2$.
- Then $\mathbf{m}_3$ can be expressed as a non-linear function of $\mathbf{m}_1$, $\mathbf{m}_2$ and λ as

$$u_i = \frac{\mathbf{m}_1^T \mathbf{X}_i}{\lambda \mathbf{m}_3^T \mathbf{X}_i}, \qquad v_i = \frac{\mathbf{m}_2^T \mathbf{X}_i}{\lambda \mathbf{m}_3^T \mathbf{X}_i}$$

- To find $\mathbf{m}_3$ need to solve a non-linear optimization problem.
- However it is a much easier optimization problem than before.

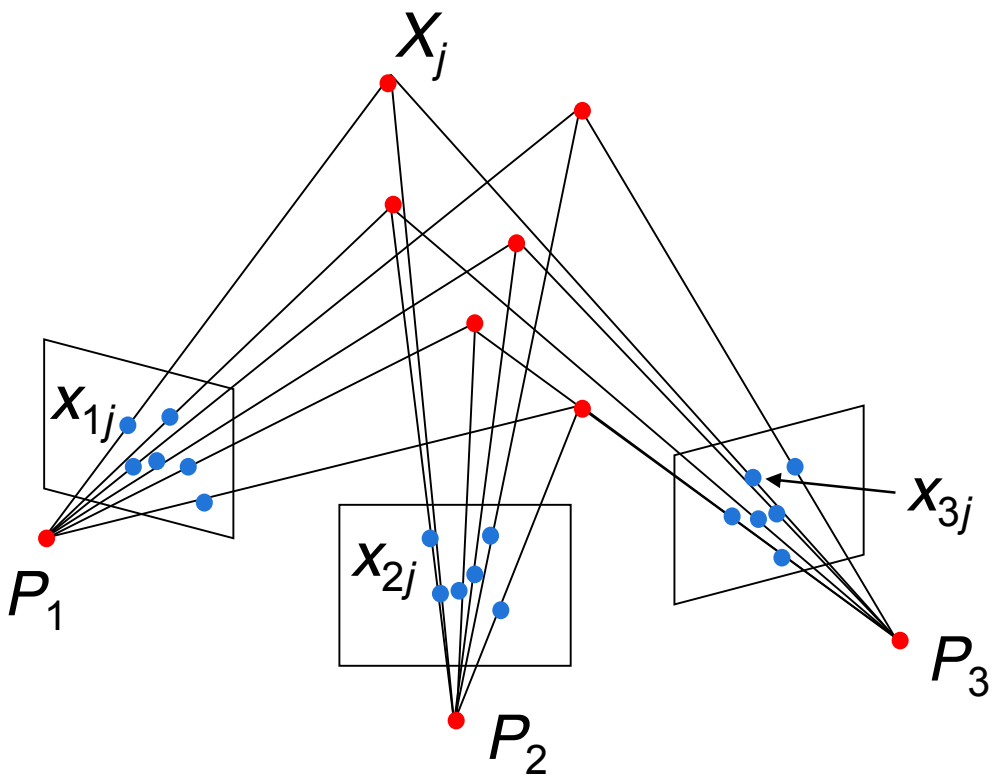
Sequential Structure from motion

Projective Structure from motion

- **Given**: m images of n fixed 3D points

$$\mathbf{x}_{ij} \simeq M_i \mathbf{X}_j \quad \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n$$

- **Problem**: Estimate m projection matrices M_i and n 3D points $\mathbf{X}_j$ from the mn correspondences $\mathbf{x}_{ij}$



Projective Structure from motion

- **Given**: m images of n fixed 3D points

$$\mathbf{x}_{ij} \simeq M_i \mathbf{X}_j \quad \text{for } i = 1, \ldots, m,\ j = 1, \ldots, n$$

- **Problem**: Estimate m projection matrices M_i and n 3D points $\mathbf{X}_j$ from the mn correspondences $\mathbf{x}_{ij}$

- **What's possible?**
  - With no calibration info, cameras and points can only be recovered up to a 4×4 projective transformation H:

$$\mathbf{X} \to H\mathbf{X}, \quad M \to MH^{-1}$$

  - Can solve for structure and motion when

$$2mn \geq 11m + 3n - 15$$

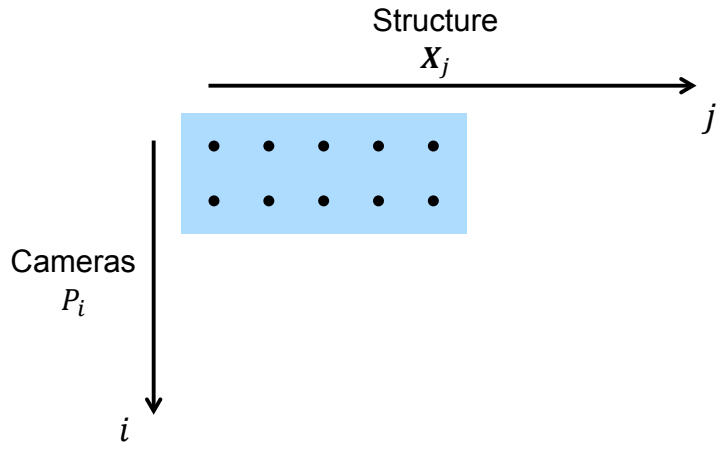
  - For two cameras, at least 7 points are needed.

Projective SfM: Two-camera case

- Compute the fundamental matrix F between the two views.
- First camera matrix: $\begin{pmatrix} I & \mathbf{0} \end{pmatrix}$
- Second camera matrix: $\begin{pmatrix} A & \mathbf{b} \end{pmatrix}$
- Then $\mathbf{b}$ is the epipole ($F^T\mathbf{b} = \mathbf{0}$) and $A = -[\mathbf{b}]_\times F$

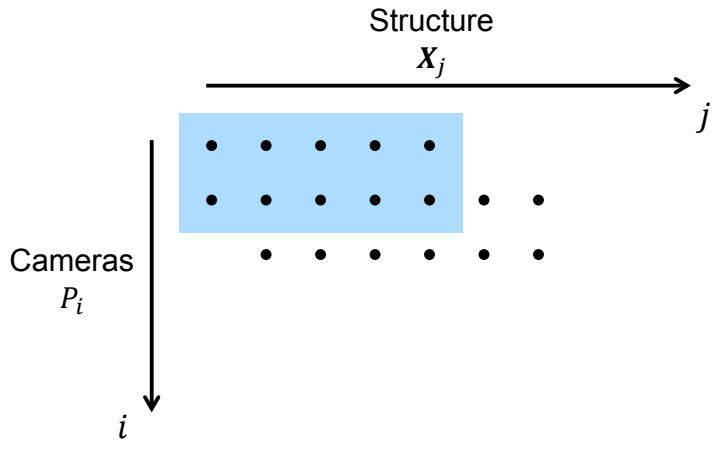
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



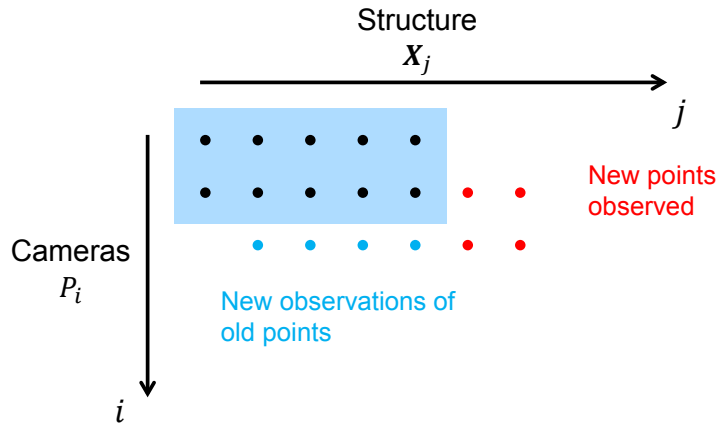
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



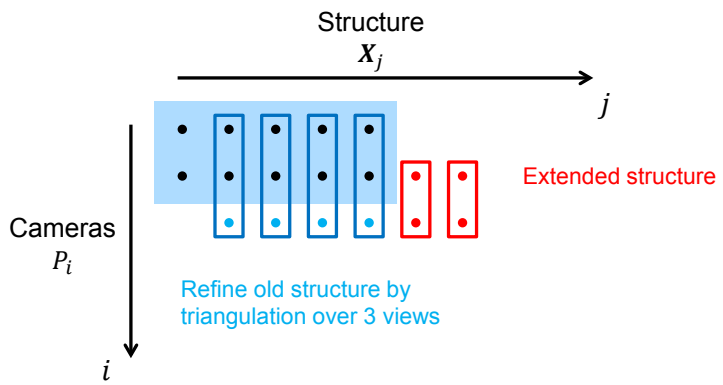
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



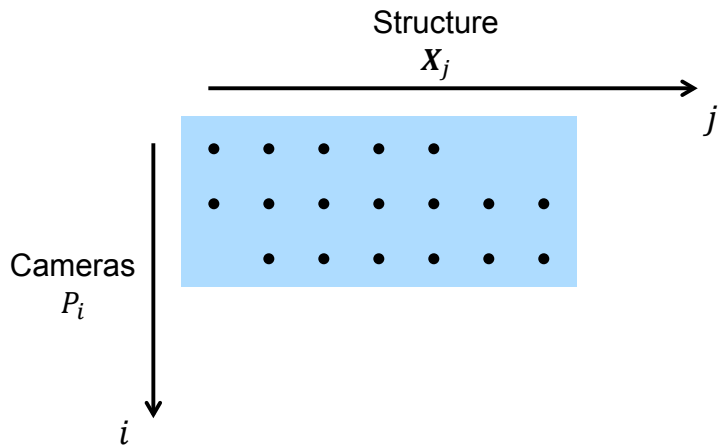
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



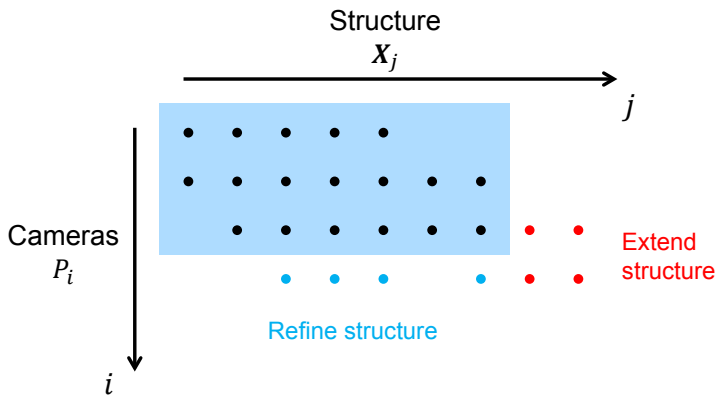
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



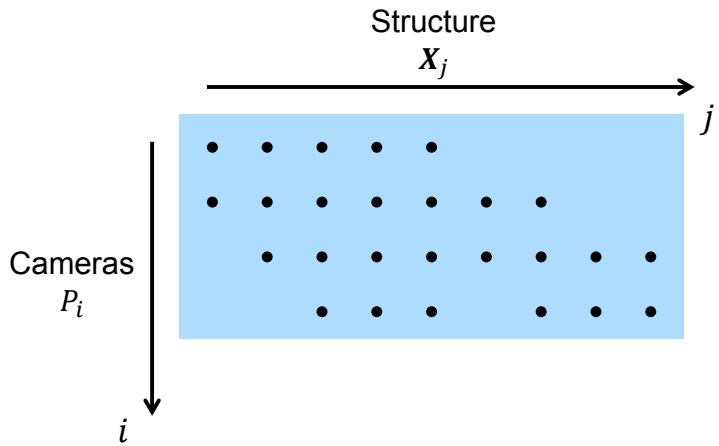
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



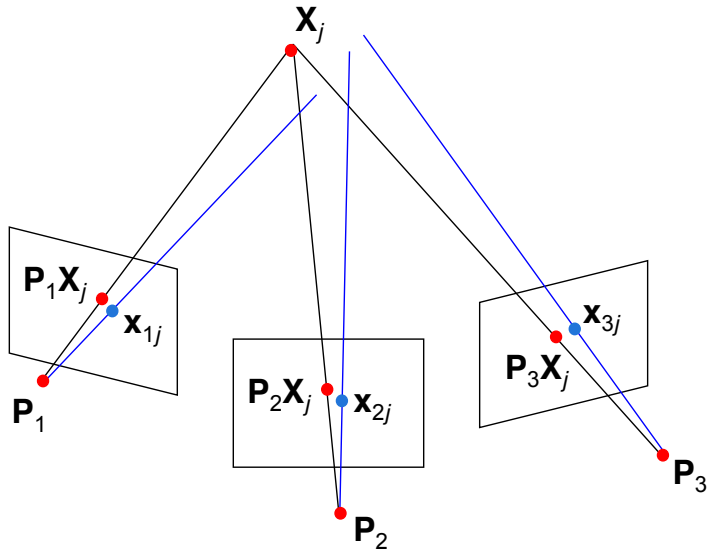
Sequential structure from motion

1. **Initialize motion**: Estimate M_1 and M_2 from the fundamental matrix $F_{1\to 2}$.
2. **Initialize structure**: Estimate 3D points of matched points in image 1 and 2 using triangulation.
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.



Sequential structure from motion

1. **Initialize motion**
2. **Initialize structure**
3. For each additional view:
   - **Calibrate**: Estimate projection matrix of new camera using all the known 3D points that are visible in the image.
   - **Extend structure**: Compute new 3D points from the matched points between current image and prior images.
   - **Refine structure**: Re-optimize existing points that are also seen by this camera.
4. Refine structure and motion:
   bundle adjustment



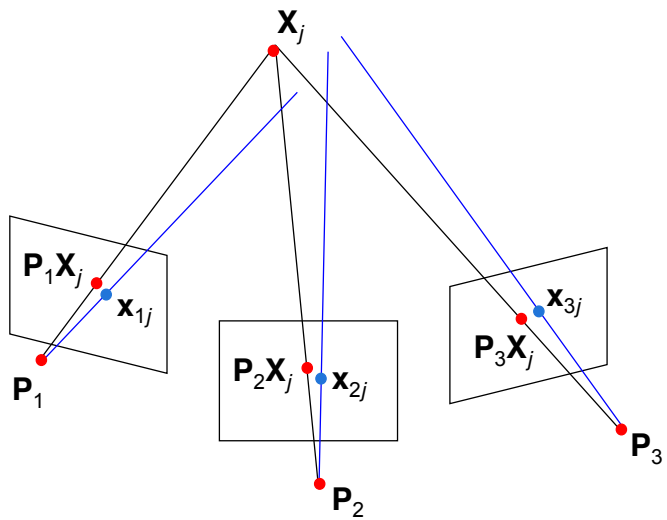
Bundle Adjustment

- Non-linear method for refining structure and motion
- Minimize reprojection error

$$\underset{\{M_i\},\{\mathbf{X}_j\}}{\arg\min} \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}\|\mathbf{x}_{ij} - M_i\mathbf{X}_j\|^2$$

where

$$w_{ij} = \begin{cases} 1 & \text{if } j\text{th point is visible in } i\text{th camera} \\ 0 & \text{otherwise} \end{cases}$$



Bundle Adjustment

- Minimize reprojection error

$$\underset{\{M_i\},\{\mathbf{X}_j\}}{\arg\min} \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}\|\mathbf{x}_{ij} - M_i\mathbf{X}_j\|^2$$

- Need initial estimates for all parameters!
- Typically use none-linear optimization methods such as a Newton method of Levenberg-Marquardt.
- Bundle adjustment is hard:
  - cost function is very non-linear w.r.t. parameters.
  - requires a good initialization.
  - can become an extremely large optimization problem because of the number of parameters involved.

    $3n + 11m$ for the uncalibrated cameras and $3n + 6m$ for calibrated cameras.

Bundle Adjustment

- Minimize reprojection error

$$\underset{\{M_i\},\{\mathbf{X}_j\}}{\arg\min} \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij}\|\mathbf{x}_{ij} - M_i\mathbf{X}_j\|^2$$

- Need initial estimates for all parameters!
- Typically use none-linear optimization methods such as a Newton method of Levenberg-Marquardt.
- Bundle adjustment is hard:
  - cost function is very non-linear w.r.t. parameters.
  - requires a good initialization.
  - can become an extremely large optimization problem because of the number of parameters involved.

    $3n + 11m$ for the uncalibrated cameras and $3n + 6m$ for calibrated cameras.

Strategies for making bundle adj. manageable

Reduce n and/or m

- Do not include all views and/or all points in the bundle adjustment. Fill in omitted views and points by resectioning or triangulation respectively **or**

- Partition data into several set. Bundle adjust independently and then merge results.

Strategies for making bundle adj. manageable

Interleaved bundle adjustment

- Alternate the following
  - Fix the cameras and minimize reprojection error by varying the 3D points.
  - Fix the 3D points and minimize reprojection error by varying the cameras.

- Computationally viable approach as each point is estimated independently given fixed cameras, and similarly each camera is estimated independently from fixed points.

- Interleaving minimizes the same cost function as bundle adjustment, so the same solution should be obtained (provided there is a unique minimum), but it may take longer to converge.

Strategies for making bundle adj. manageable

Sparse bundle adjustment

- For each iteration, iterative minimization methods determine an update vector $\boldsymbol{\delta}$ for the parameter values.

- In Levenberg Marquardt each such step is determined from:

$$(J^T J + \lambda I)\boldsymbol{\delta} = -J^T \boldsymbol{\epsilon}$$

  where

  - each row of J is the gradient of a projected point coordinate w.r.t. the parameters and
  - $\boldsymbol{\epsilon}$ is the vector of residuals between the projected point and its measured position.

- For the bundle adjustment problem J has a sparse structure that can be exploited in computations.

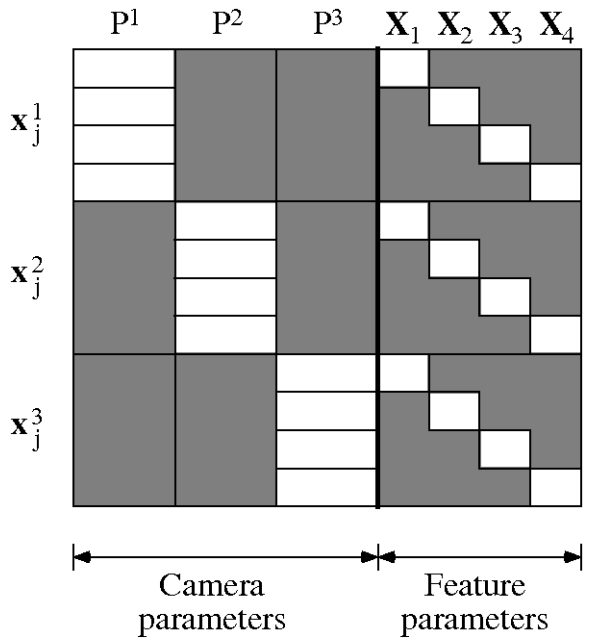
Strategies for making bundle adj. manageable

Sparse bundle adjustment

- In Levenberg Marquardt each such step is determined from:

$$(J^T J + \lambda I)\boldsymbol{\delta} = -J^T \boldsymbol{\epsilon}$$

- For the bundle adjustment problem J has a sparse structure that can be exploited in computations.



The sparse structure of the Jacobian matrix for a bundle adjustment problem

What's possible today

- Combined with parallel processing the mentioned strategies have made it possible to solve extremely large SfM problems.

- **Examples**:

  1. S. Agarwal et al, *Building Rome in a Day*, 2011
     - Cluster of 62-computers
     - 150,000 unorganized images from Rome
     - $\sim$37,000 image registered
     - Total processing time $\sim$21 hours
     - SfM time $\sim$7 hours
  2. J. Heinly et al, *Reconstructing the World in Six Days*, 2015
     - 1 dual processor PC with 5 GPUs (CUDA)
     - 96, 000, 000 unordered images spanning the globe
     - $\sim$15, 000 000 images registered
     - Total processing time 5 days
     - SfM time 17 hours