3D Reconstruction

Part II – A deep insight…

Elmar Eisemann

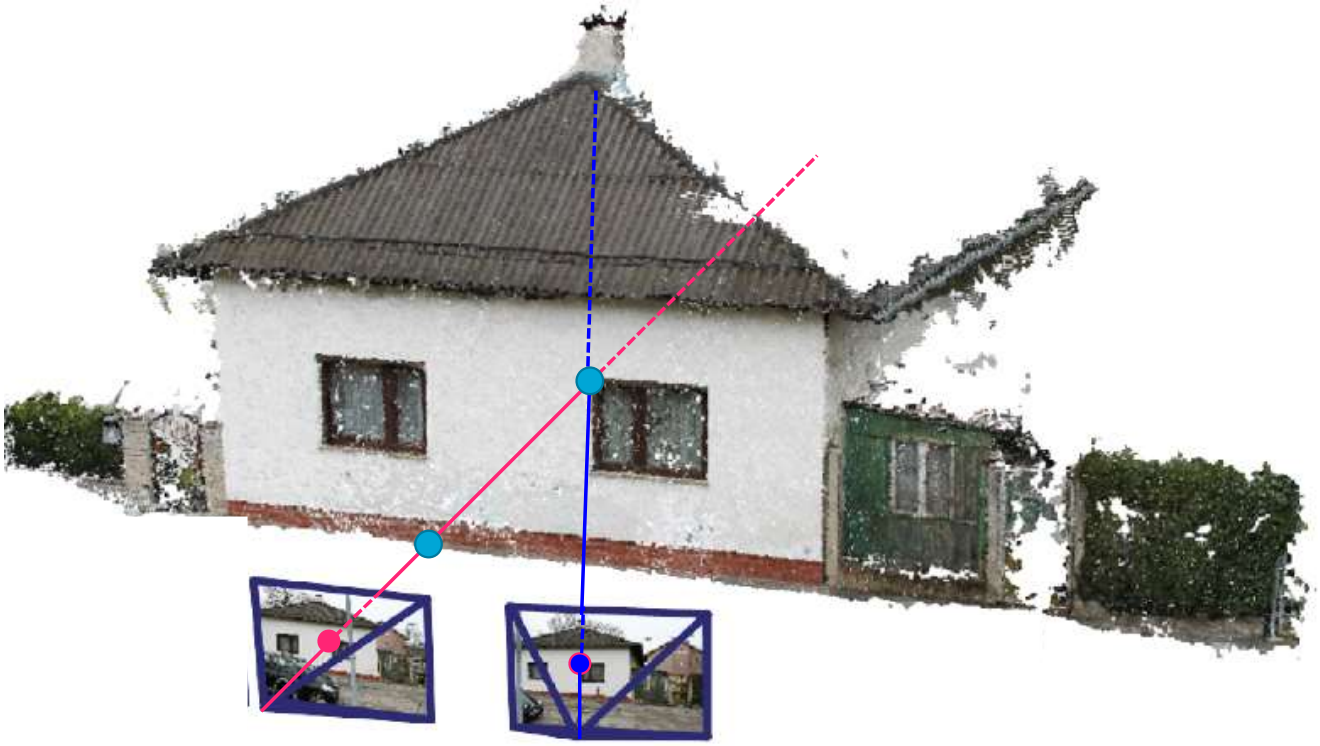
Computer Graphics and Visualization

Delft University of Technology

Stereograms

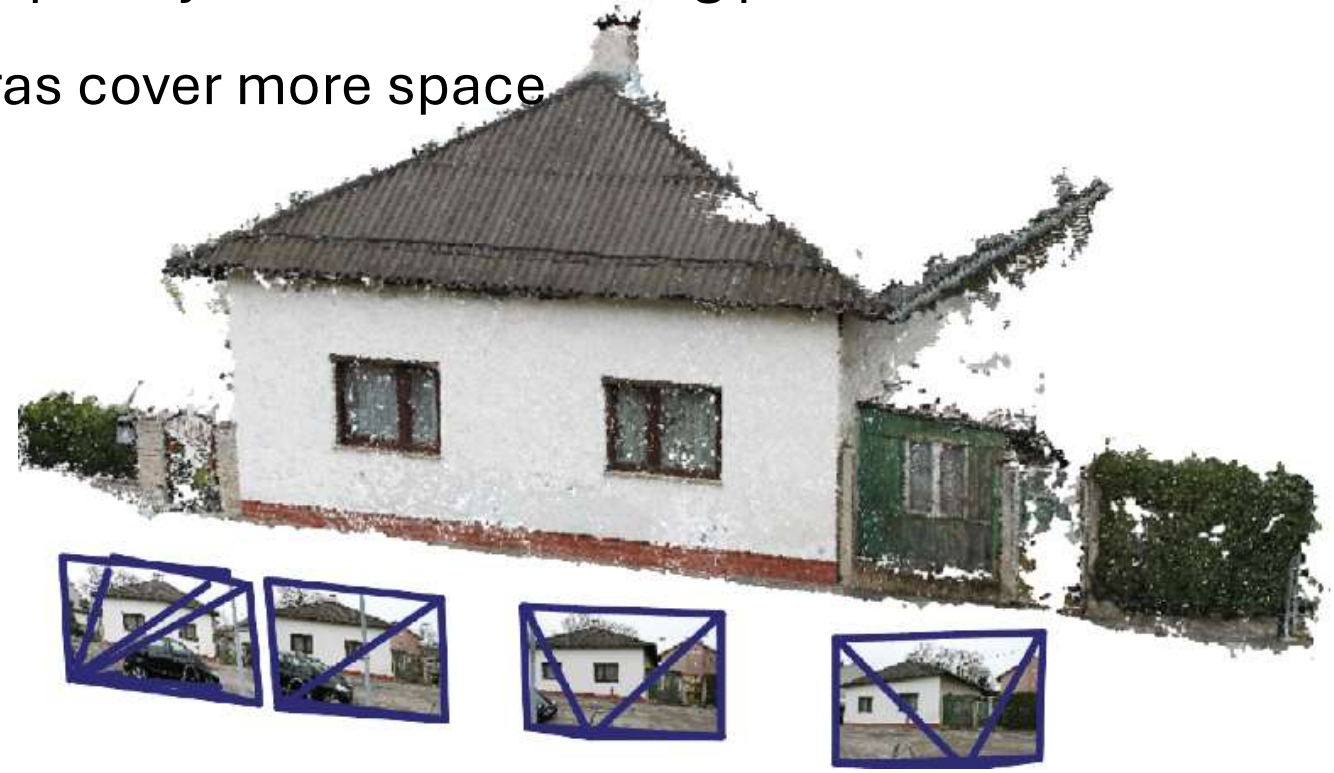
- Stereo pairs reveal 3D information

3D Reconstruction



3D Reconstruction

- Usually not perfectly aligned
- For every 3D point you need a matching pair
- More cameras cover more space



3D Reconstruction

- How do cameras work?
  - How to model cameras mathematically?
- How to find the pose and calibrate a camera?
- How to reconstruct a scene ?
- How to calibrate from photo collections?

3D Reconstruction

- How do cameras work?
  - How to model cameras mathematically?

- How to find the pose and calibrate a camera?

- How to reconstruct a scene ?

- How to calibrate from photo collections?

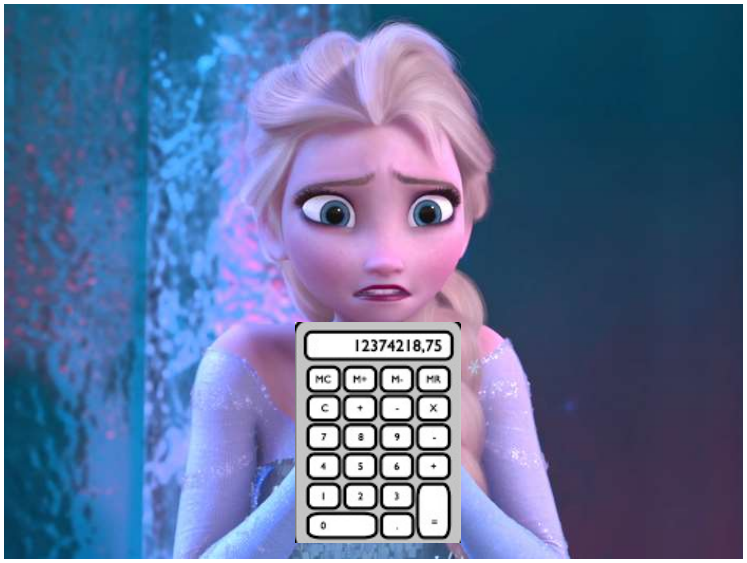
Producing Images in the Real World

Do you wanna build...

~~a snowman?~~

a camera...?

...mathematically? ☺



What does this mean?

- Given 3D point M, we want a function C such that C(M) is the point's projection in the photo.

Virtual Camera Model

Projecting a scene point with the camera:

- Apply camera position (adding an offset)
- Apply rotation (matrix multiplication)
- Apply projection (non-linear scaling)

Our camera starts to become very complicated, especially if we want to calculate with it...

There has to be a better way...

Homogenous Coordinates - Definition

- N-D *projective space* P^n is represented by N+1 coordinates, has no null vector, but a special equivalence relation:

Two points p, q are **equal**

iff (if and only if)

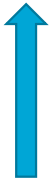
exists a!=0 such that p*a = q

Examples in a 2D projective space P^2:
(2,2,2) = (3,3,3) = (4,4,4)=(π, π, π)
(2,2,2) $\neq$ (3,1,3)
(0,1,0) = (0,2,0)
(0,0,0) not part of the space

Homogeneous Coordinates

To embed a standard vector space R^n in an n-D projective space P^n, we can map:

$(x_0,x_1,\ldots x_{n-1})$ in R^n to $(x_0,x_1,\ldots x_{n-1}, 1)$ in P^n

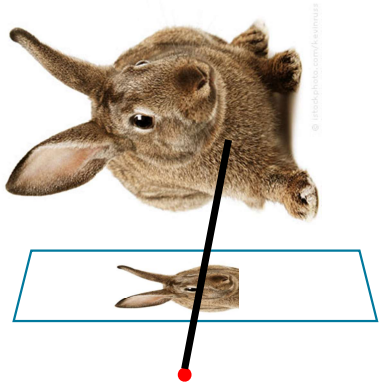


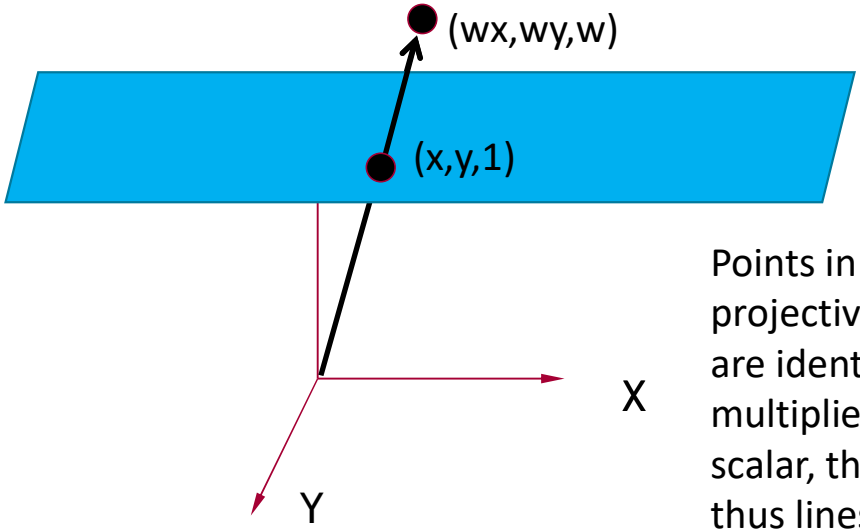
Typically, the last coordinate in a projective space is denoted with w.

Homogeneous Coordinates

- A point (x,y) in R^2 embedded in a projective space corresponds to (x,y,1). All points (x,y,1) form a plane (referred to as *affine plane*)

The points in this plane correspond to points in our standard vector space R^2

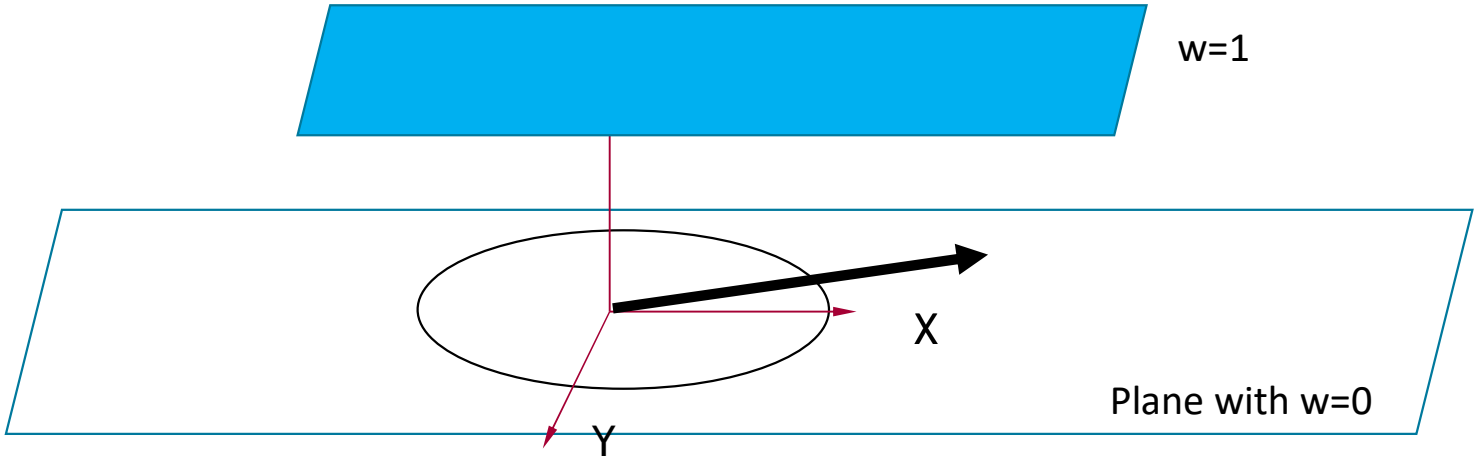




Points in a projective space are identical when multiplied by any scalar, they are thus lines!

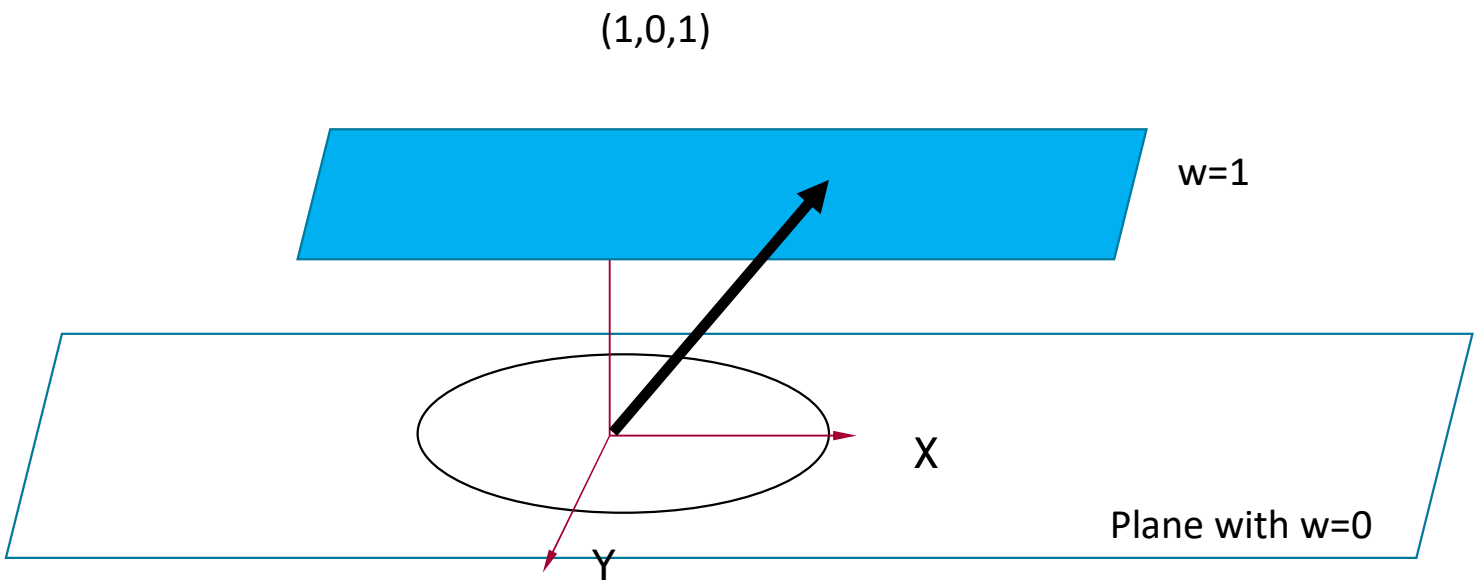
Homogeneous Coordinates

- What about the points with w=0?



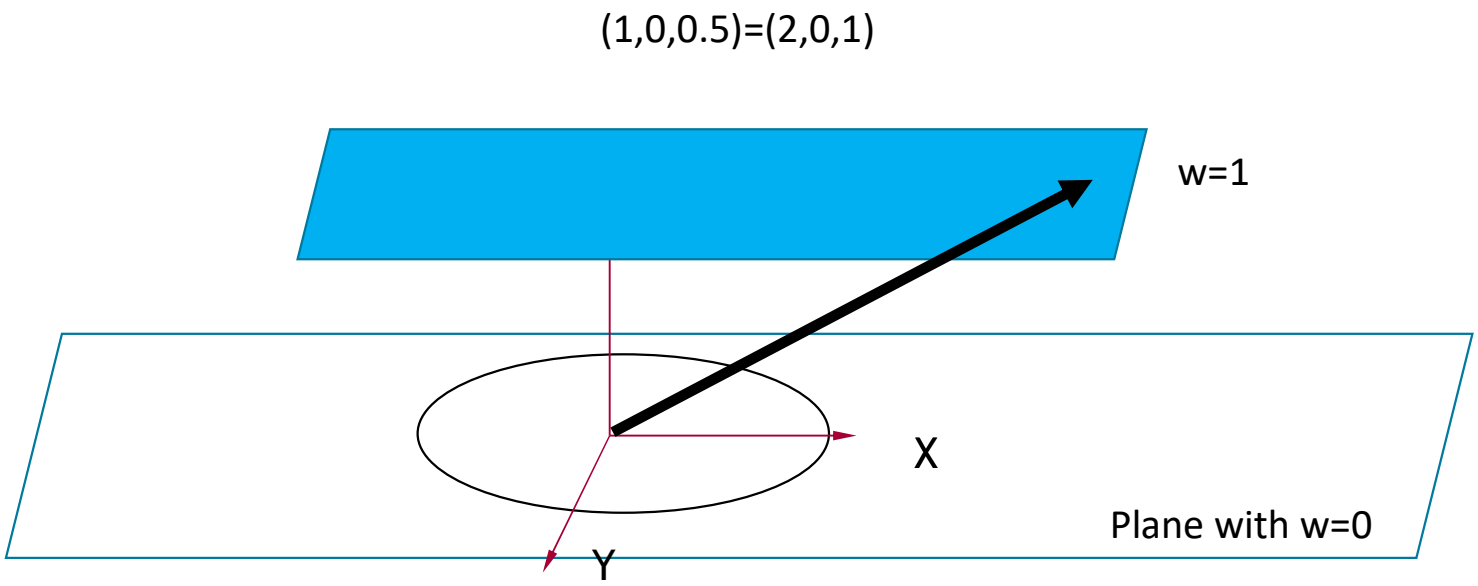
Homogeneous Coordinates

- What about the points with w=0?
- Let's try it out with a point (1,0,w) and we decrease w…



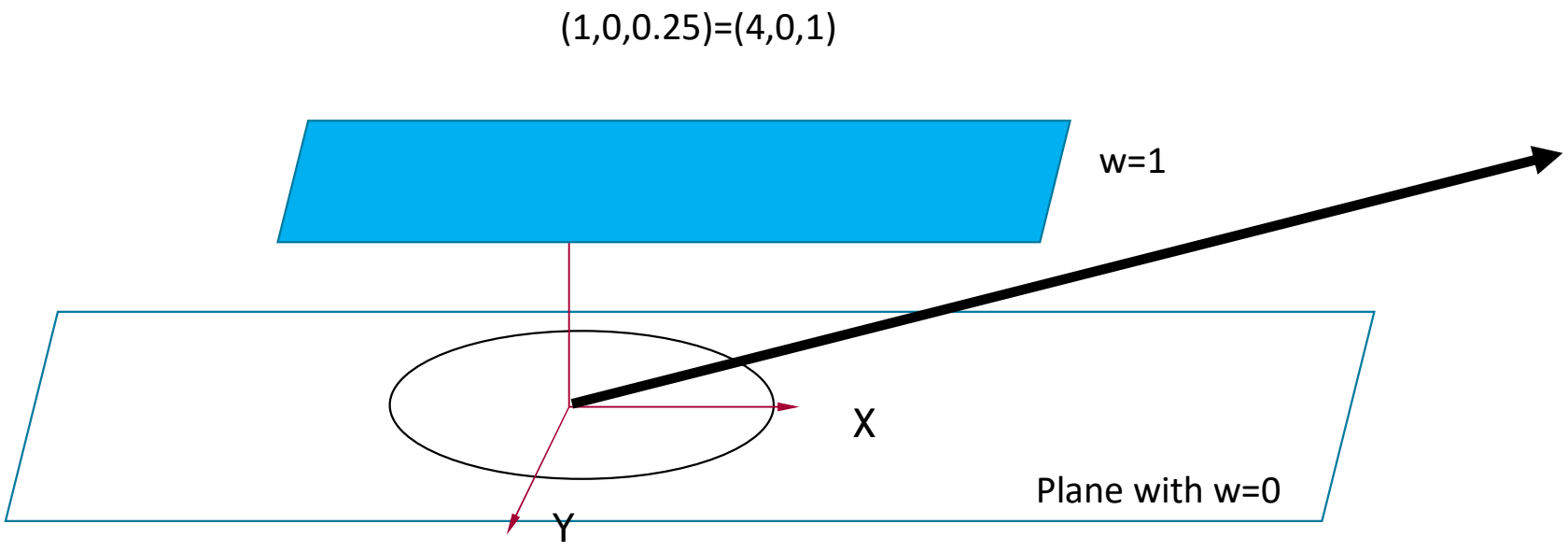
Homogeneous Coordinates

- What about the points with w=0?
- Let's try it out with a point (1,0,w) and we decrease w…



Homogeneous Coordinates

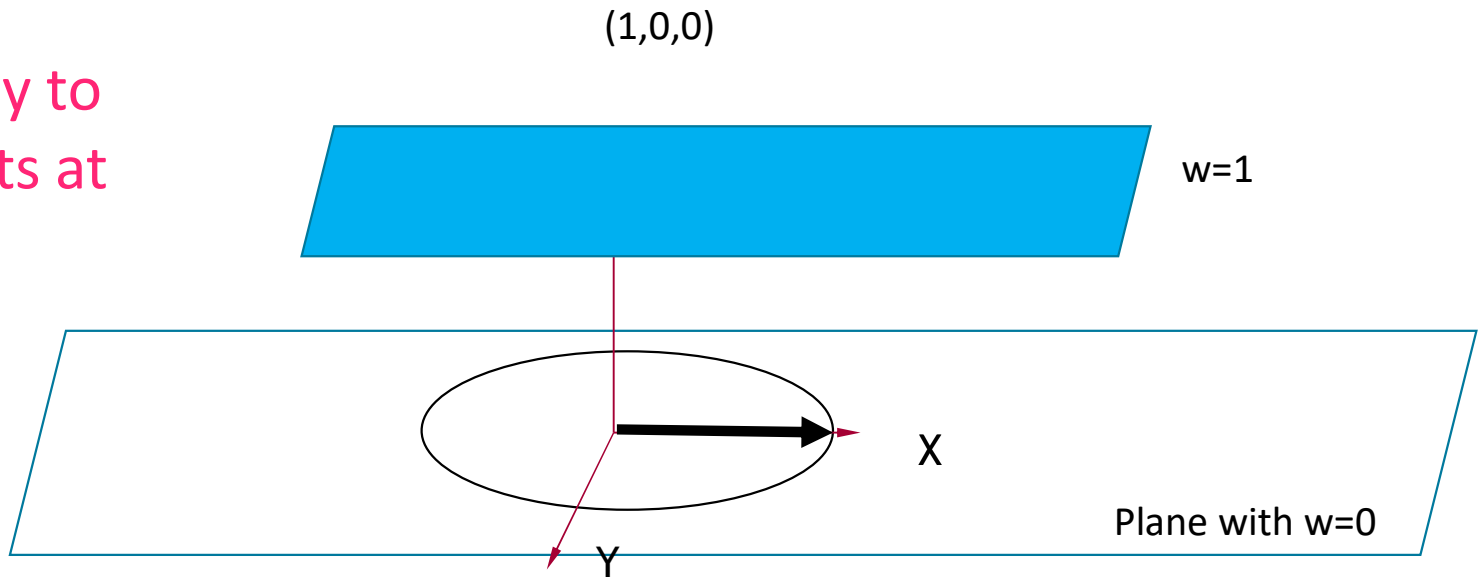
- What about the points with w=0?
- Let's try it out with a point (1,0,w) and we decrease w…



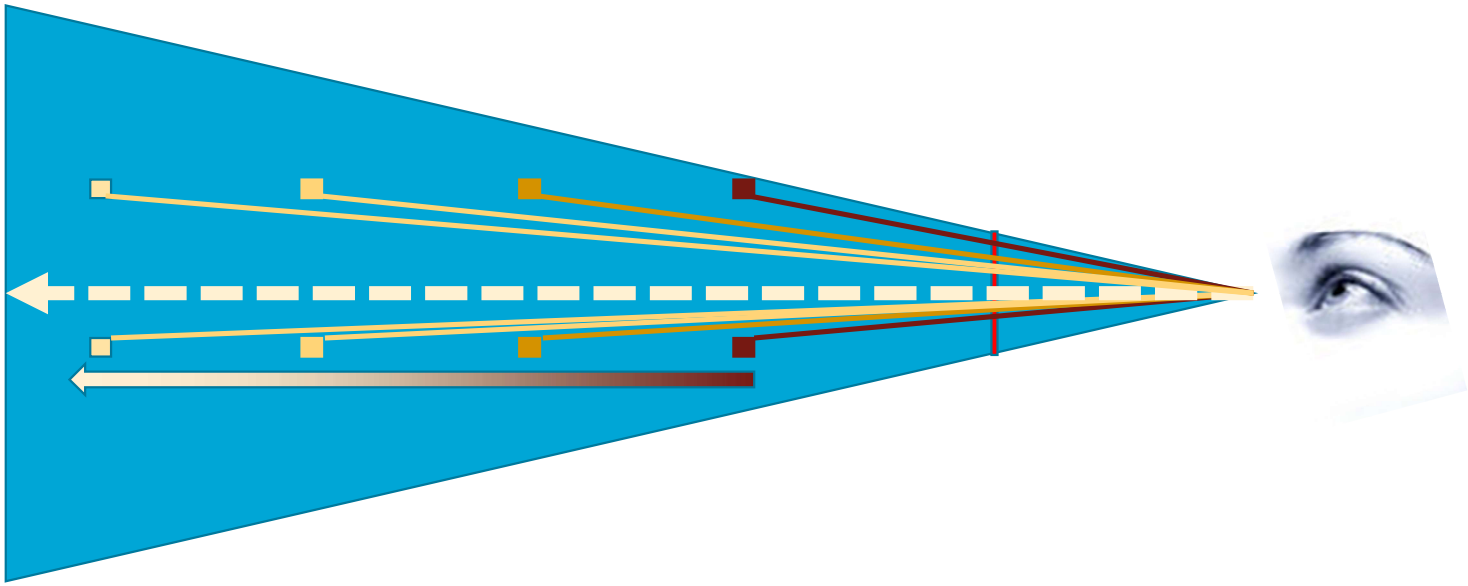
Homogeneous Coordinates

- What about the points with w=0?
- Let's try it out with a point (1,0,w) and we decrease w…

We have a way to describe points at INFINITY!



Linear Perspective



Linear Perspective

Homogeneous Coordinates

We wanted:

Easy transformations = matrices

Translations, rotations,
scaling, projection should all be matrices…

Concatenating transformations
means simply matrix multiplications!

Translations in R^2

- To translate vector (x,y),
  we add (t_x,t_y) to obtain: $(x+t_x,y+t_y)$

In a projective space,
we would like to have a matrix M, such that

$M(x,y,1) = (x+t_x,y+t_y,1)$

Translations in homog. coordinates

- R^2

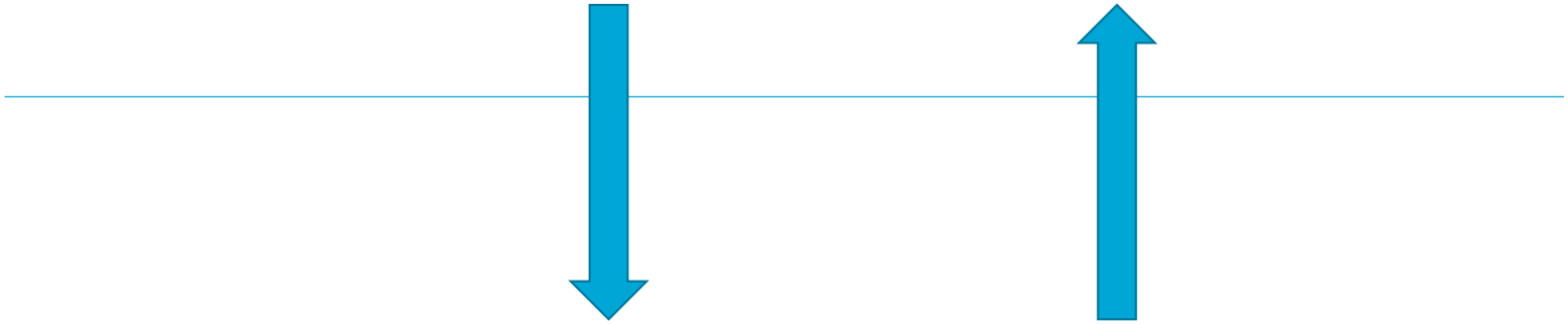
$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \end{bmatrix}$$

- P^2

Translations in homog. coordinates

- R^2

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \end{bmatrix}$$



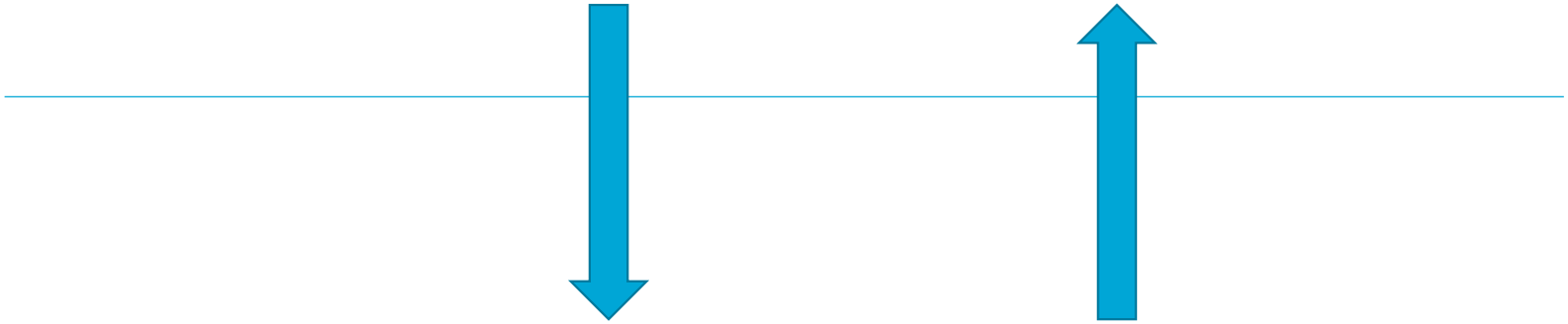
- P^2

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \qquad \begin{bmatrix} x + tx \\ y + ty \\ 1 \end{bmatrix}$$

Translations in homog. coordinates

- R^2

$$\begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \end{bmatrix}$$



- P^2

$$\begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + tx \\ y + ty \\ 1 \end{bmatrix}$$

Similar stuff for rotations...

But…

- So far, we only made things more complicated!
- Is there any benefit to all of this?
- Yes, let’s look at one example!

Rotation around point Q

- Rotation around point Q:
  - Translate Q to origin($\mathbf{T}_Q$),
  - Rotate around origin ($\mathbf{R}_\Theta$)
  - Translate back to Q ($\mathbf{T}_{-Q}$).

$$P' = (T_{-Q})\mathbf{R}_\Theta T_Q \, P$$

Camera Model

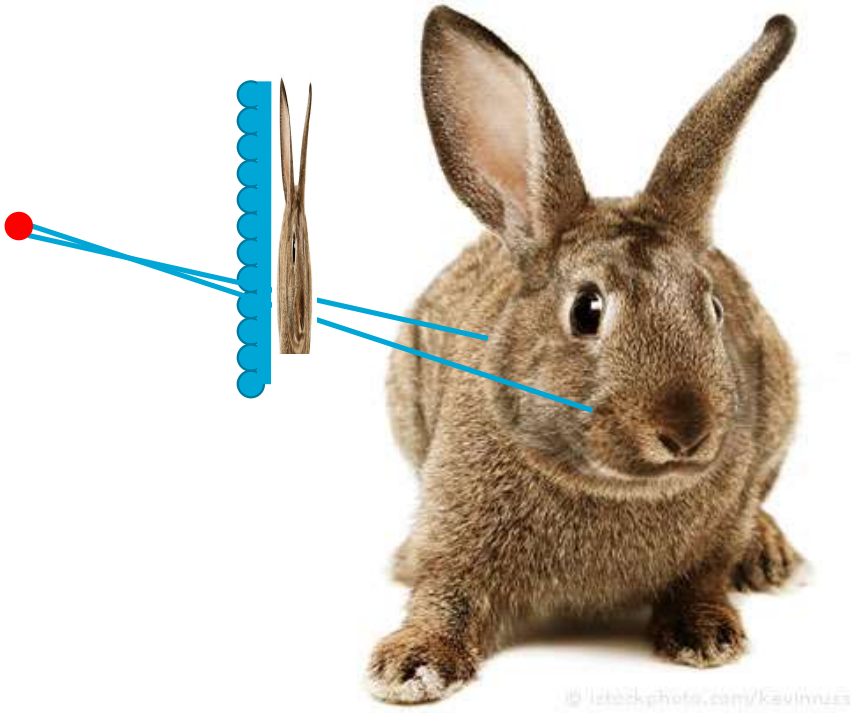
- Pinhole Perspective Projection

    using matrices

= ?

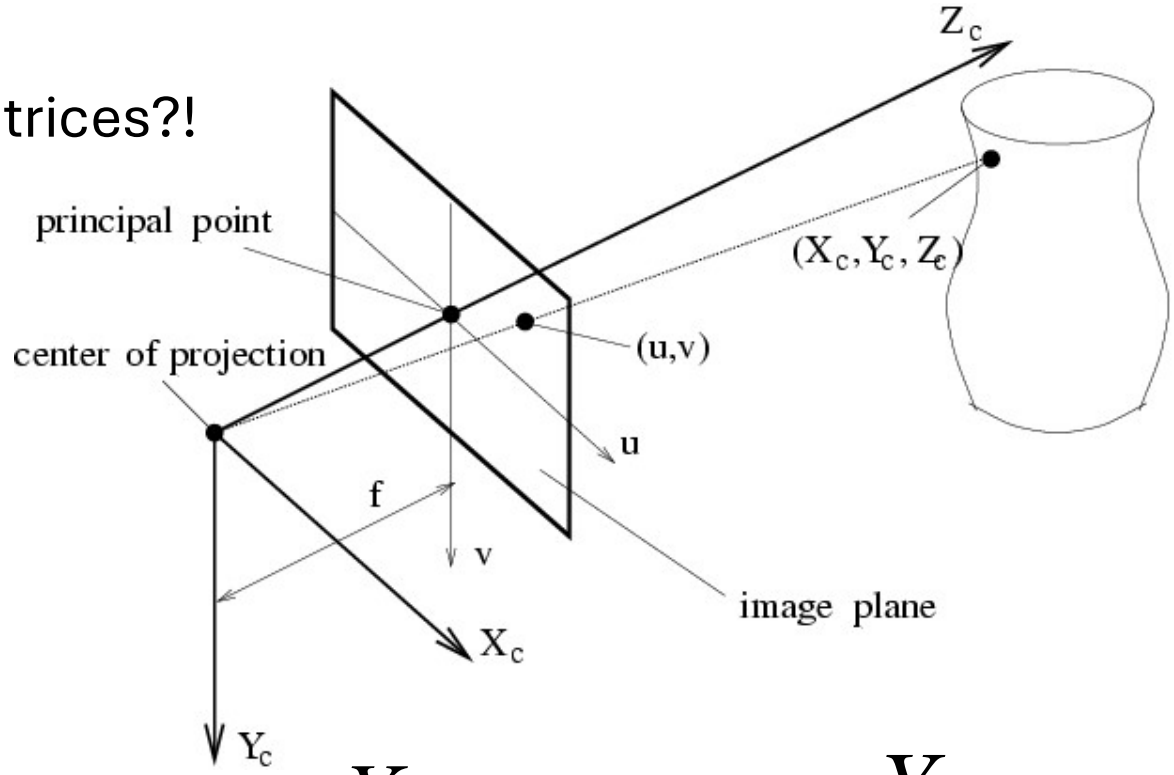
Virtual Camera

- Pinhole where real cameras have lens
- Virtual Camera Plane



Perspective Projection

- Represent with matrices?!



$$u = f\frac{X}{Z} \qquad v = f\frac{Y}{Z}$$

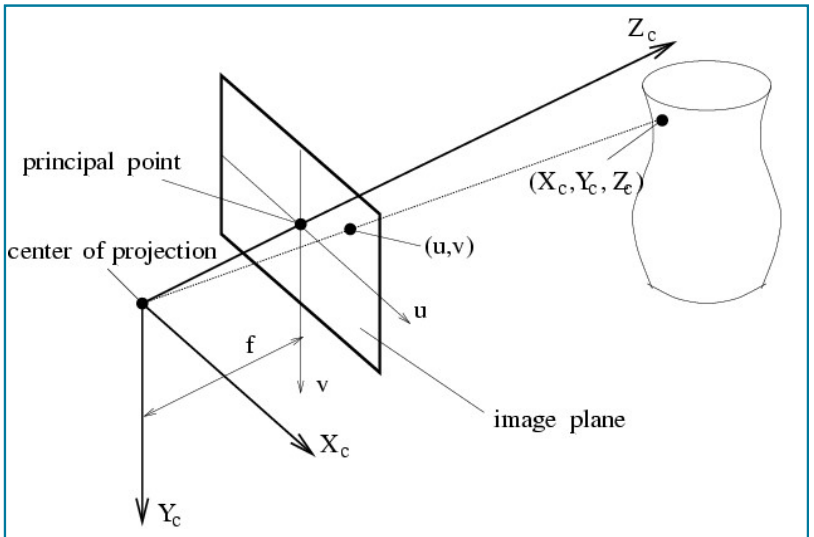
Perspective Projection

- "Homogenize" the points

$$P = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \quad \Rightarrow \quad \tilde{P} = \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- Look for M such that:

$$M\tilde{P} = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix}$$



$$u = f\frac{X}{Z} \qquad v = f\frac{Y}{Z}$$

Perspective Projection

- Hint: Think projective!

$$M\tilde{P} = \begin{pmatrix} fX/Z \\ fY/Z \\ 1 \end{pmatrix} \Rightarrow M\tilde{P} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix}$$

Perspective Projection

- Solution:

$$M = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

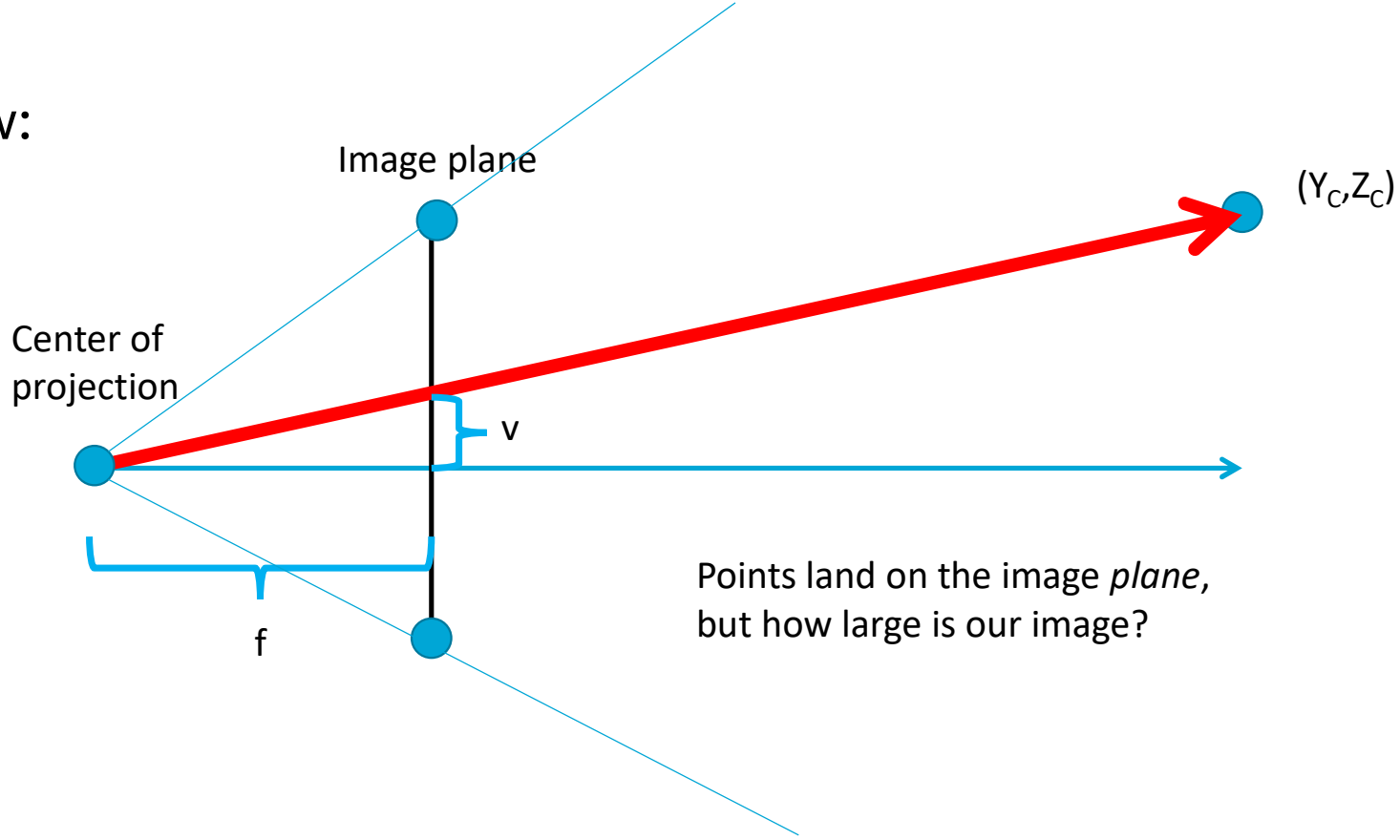
Putting things together: Camera Model

- Projection + Movement + Rotation

$$M = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

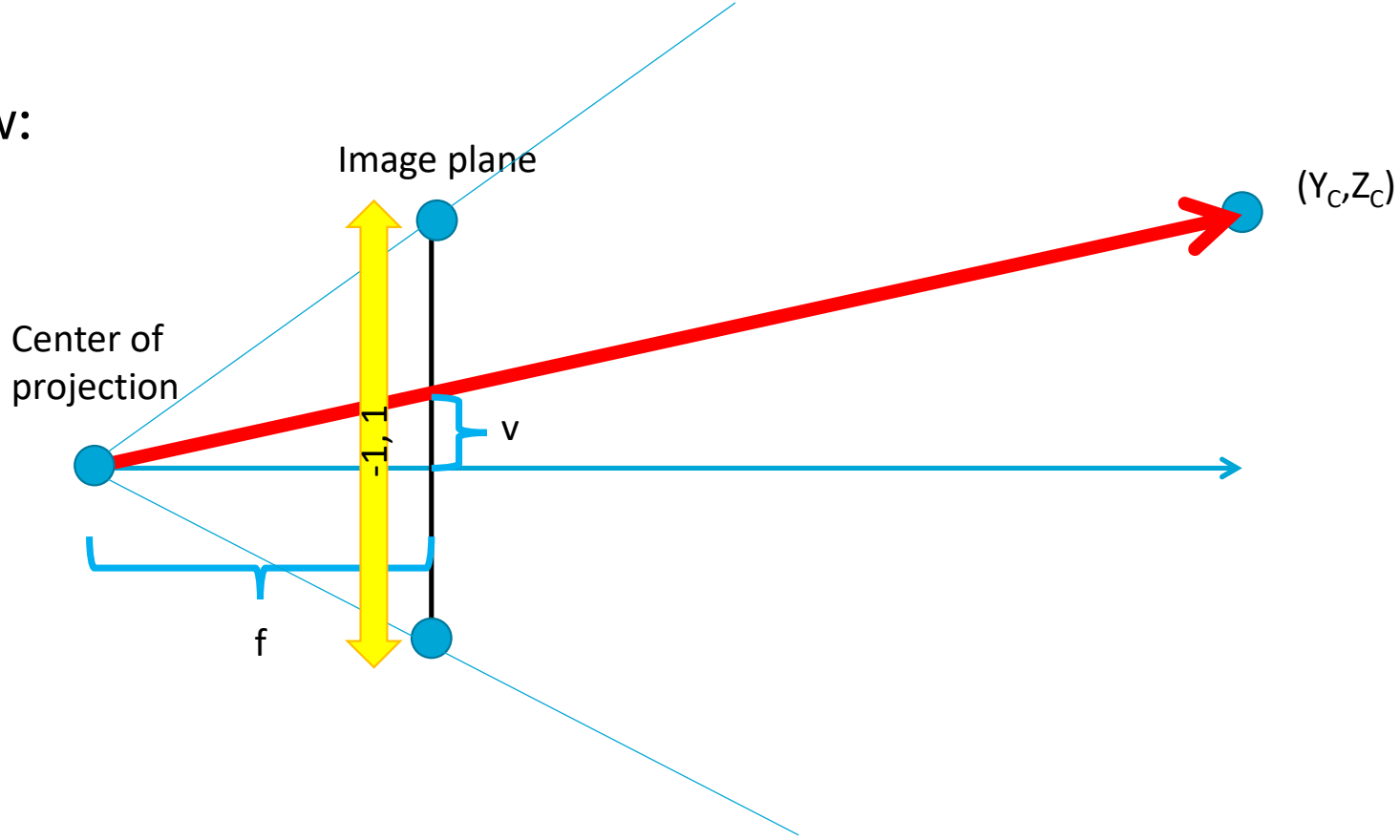
Virtual Camera Model

- sideview:



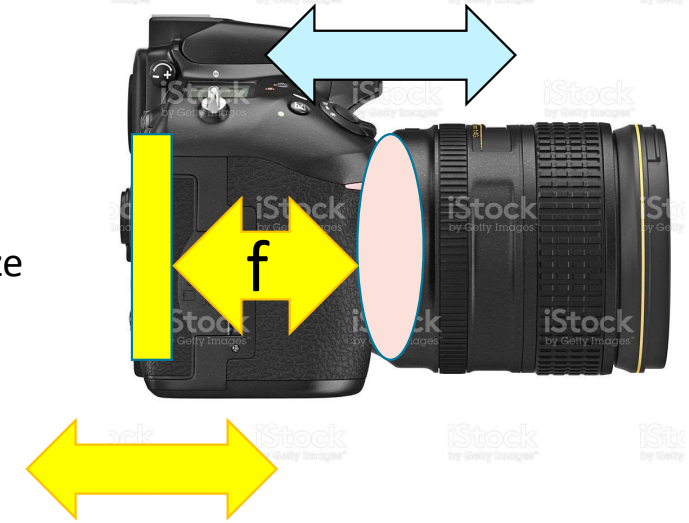
Virtual Camera Model

- sideview:



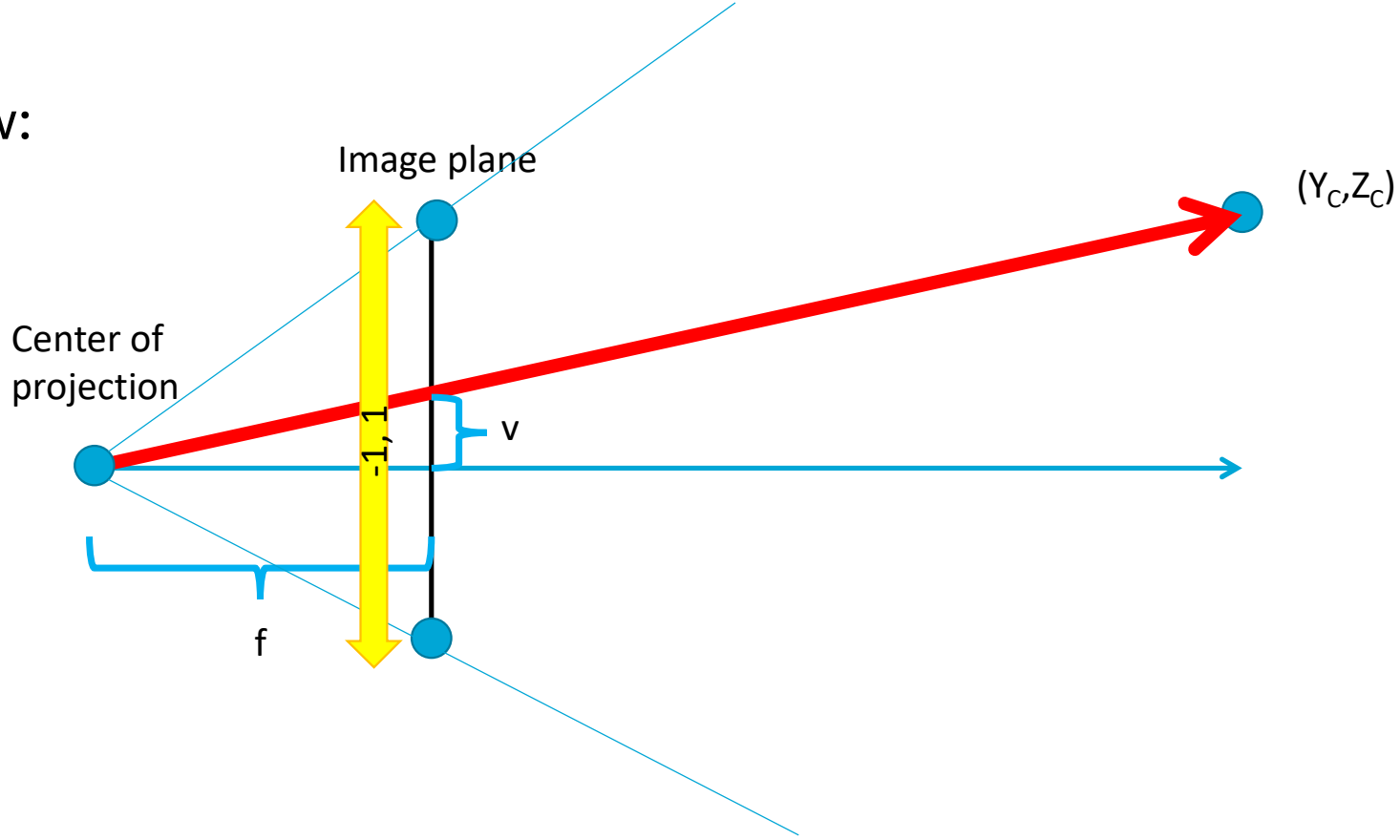
Real camera

Sensor has a fixed size
“[-1,1]”



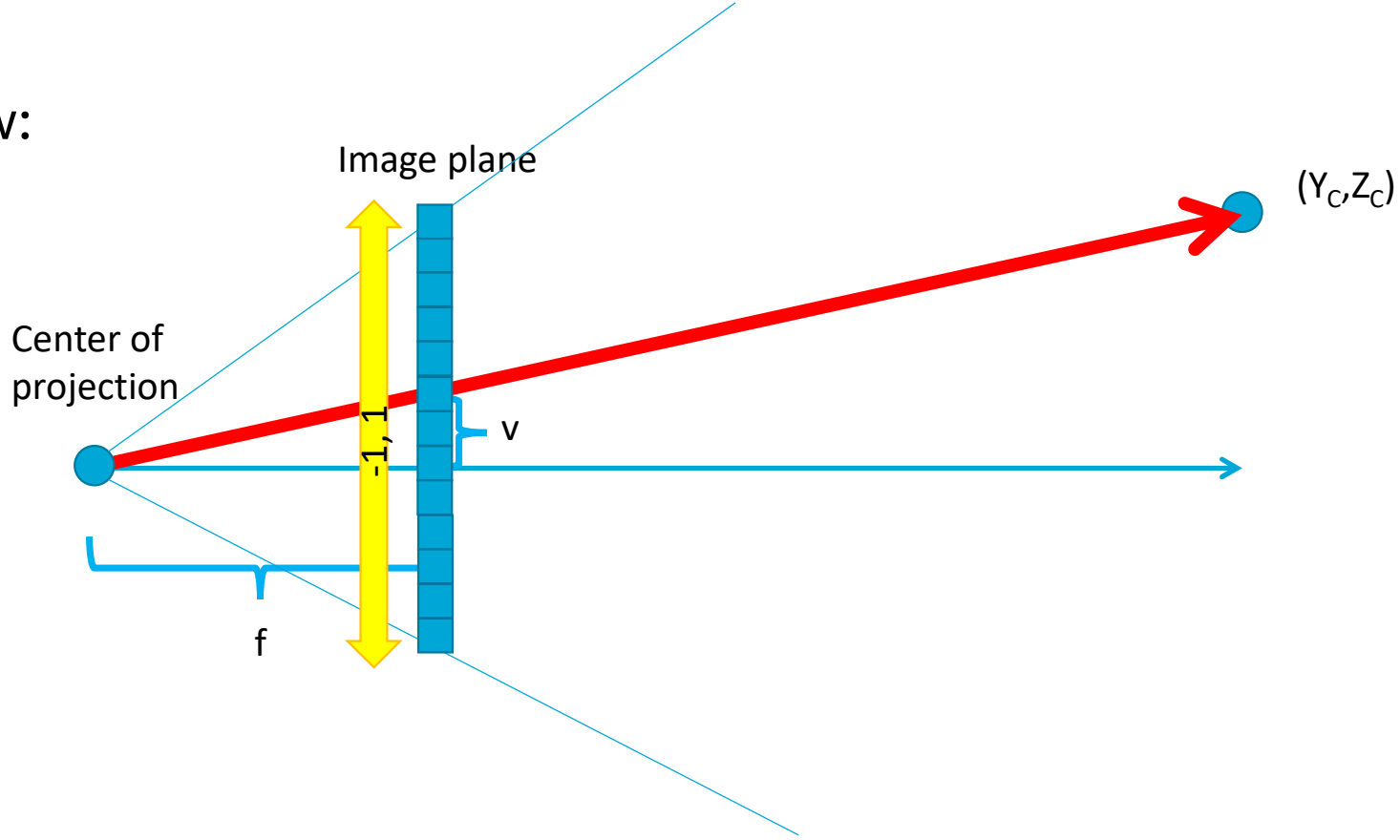
Virtual Camera Model

- sideview:



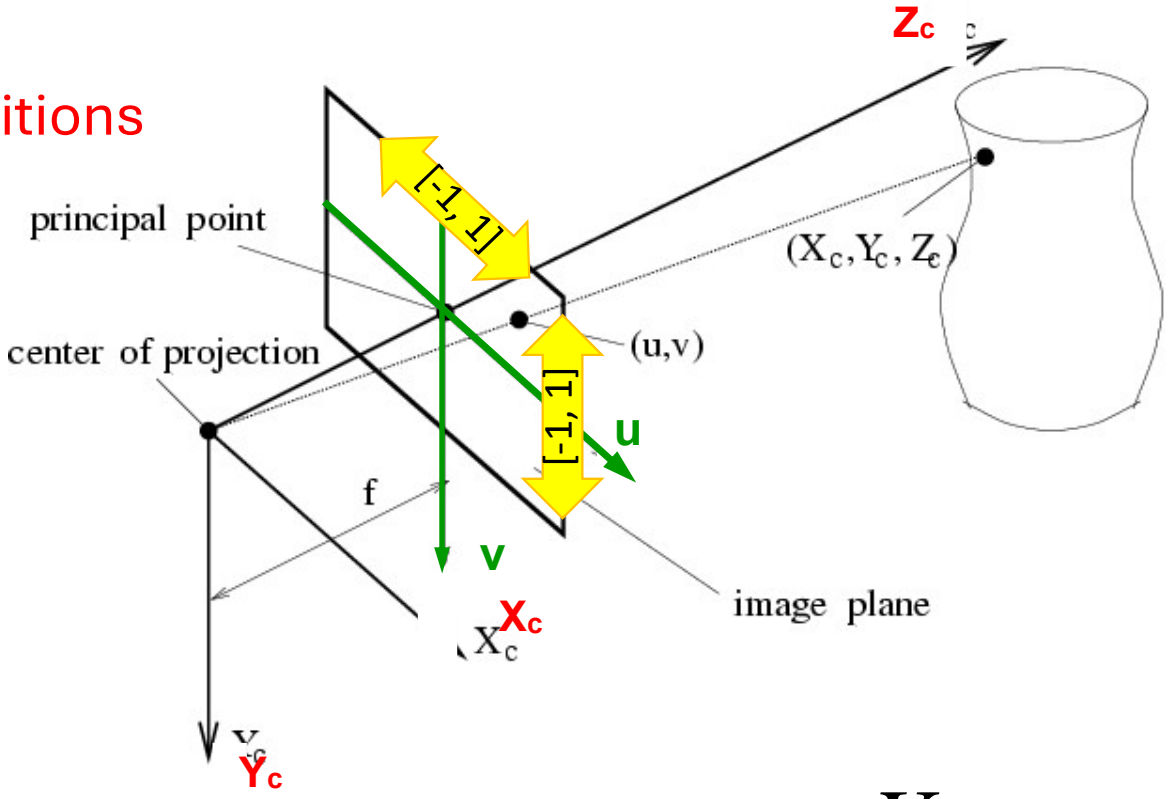
Virtual Camera Model

- sideview:



Adding Image Space to Camera Model

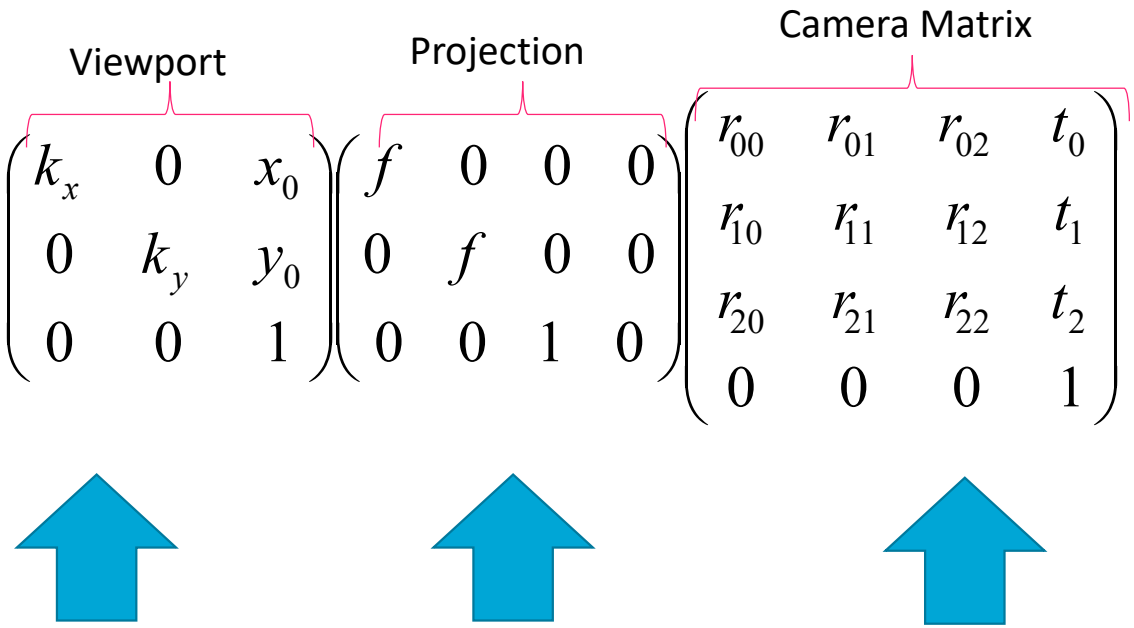
- Make u, v pixel positions



$$u = f\frac{X}{Z} \qquad v = f\frac{Y}{Z}$$

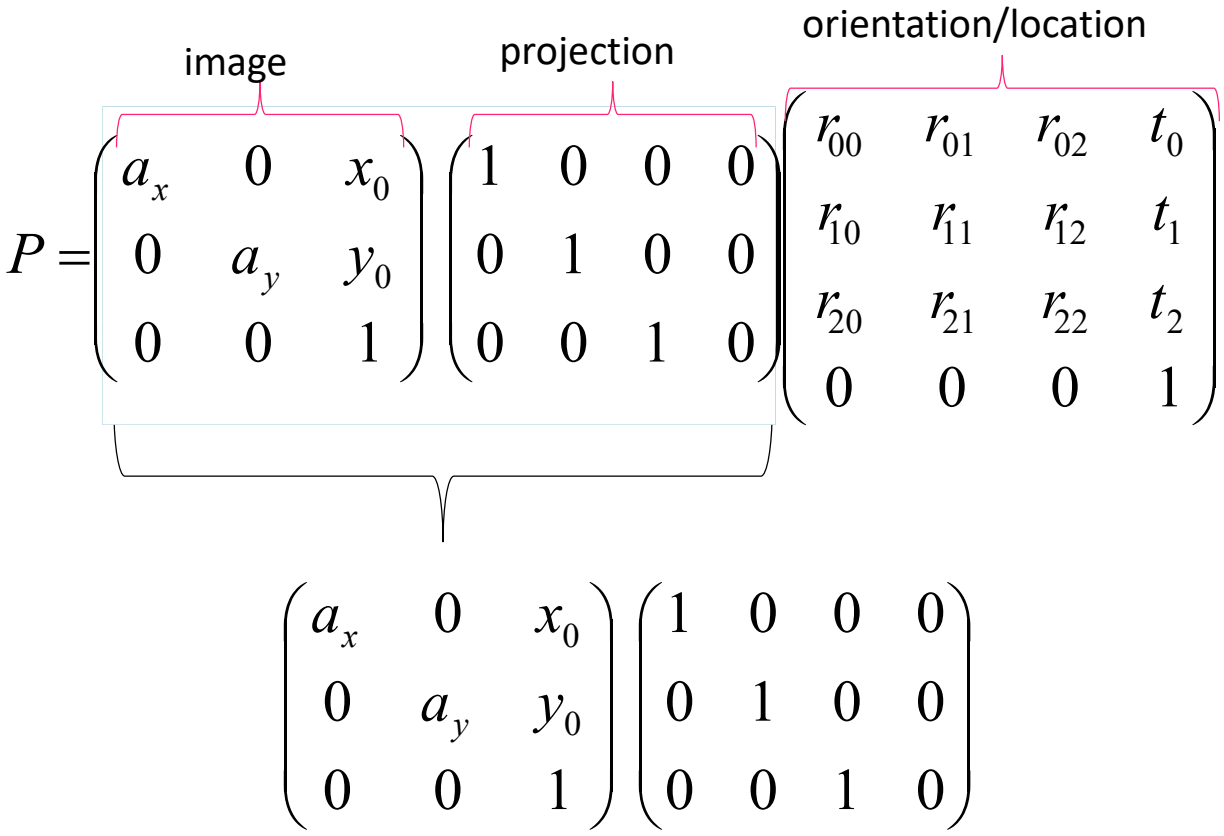
Complete Camera Model

- Finally:



Complete Camera Model

- Finally:



Complete Camera Model

- Finally:

$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$

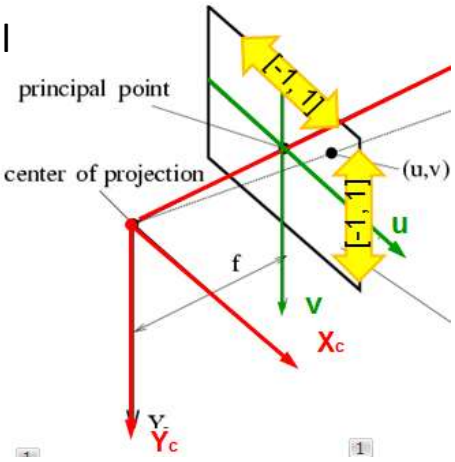
Complete Camera Model

- Finally (notation often used in vision literature):

$$P = \underbrace{\begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix}}_{\text{intrinsic/ internal parameters}} \underbrace{\begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}}_{\text{extrinsic / external parameters}}$$

Change settings

Move Camera



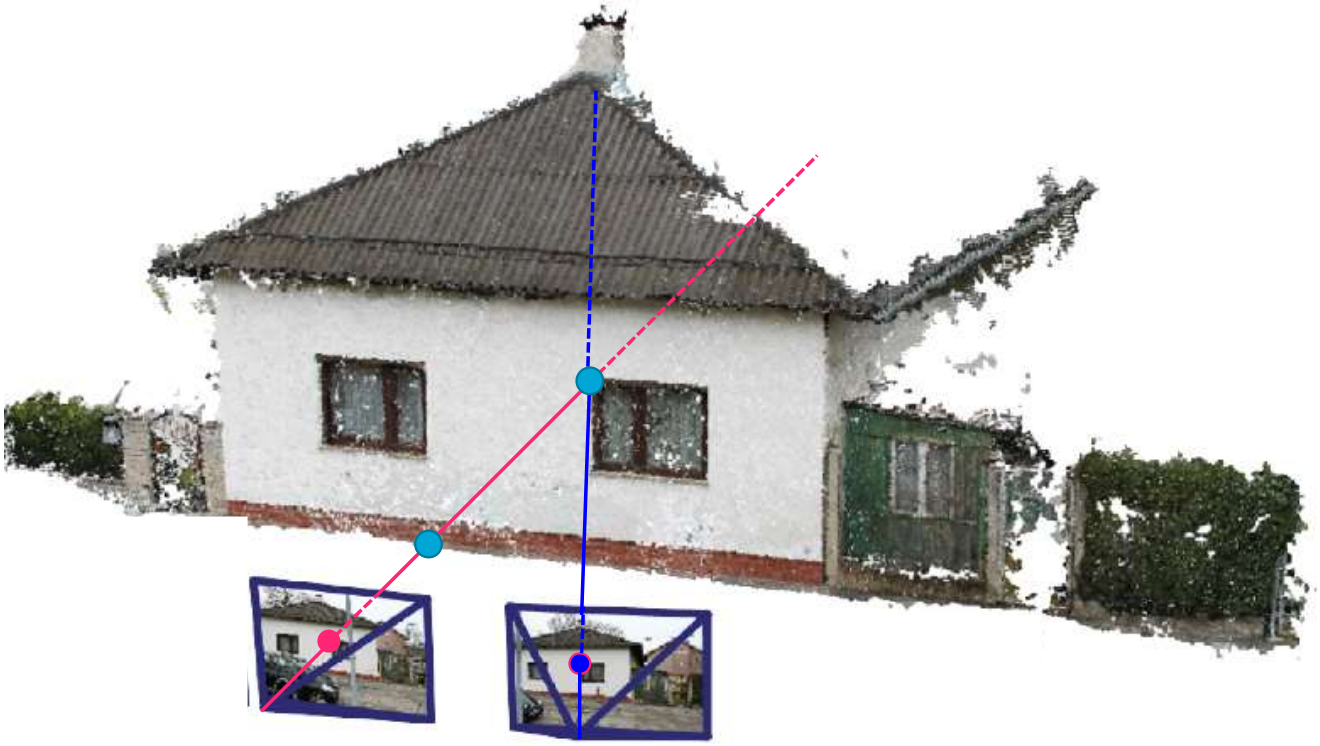
Complete Camera Model

- Finally (notation often used in vision literature):

$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$

- Realize:
  If P is known, we can project any 3D point M to its pixel position m
- Thus, if we have a matching pixel position for cameras P1, P2, then we only need to solve P1(M)=m1, P2(M)=m2 for an unknown M

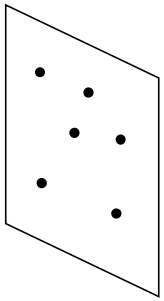
3D Reconstruction

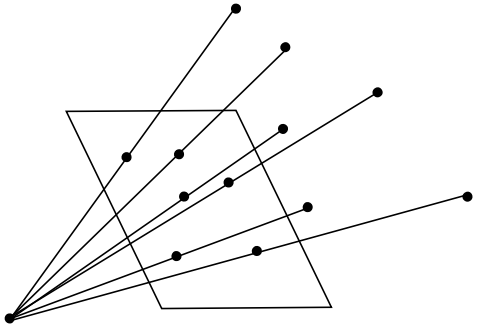


Great! But how to find the projection matrix?

Finding the projection matrix

Given enough known 3D points M_i and their projection m_i , we can compute camera matrix P





m_i (Image Pos)

M_i (World Pos)

Hence, we determine the camera matrix P in its entirety.

Finding the projection matrix

- Solve for the 4x3 projection matrix P:
  - M_i (World Pos) and m_i (Image Pos) are known, which means we can setup many equations of the form:
    $P\ M_i = m_i$
    each such expression represents 2 equations
    (2 equations= x, y position on photo)

12 entries in matrix P

in projective space, this means 11 unknowns;
one entry can be freely chosen (e.g., $P_{00}=1$)

6 points in 3D with matching pixel coordinates are enough to find P

Finding the projection matrix

- Attention:

  The expression : P $M_i = m_i$

$$\begin{pmatrix} 1 & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{kx} \\ \mathrm{ky} \\ \mathrm{k} \end{pmatrix}$$

I see three rows?! Why 2 equations per point?

We are in a projective space!

What to do to eliminate this scalar k?

Finding the projection matrix

Divide first two by the last equation to get rid of the scalar k:

$$\begin{pmatrix} 1 & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \mathrm{kx} \\ \mathrm{ky} \\ \mathrm{k} \end{pmatrix}$$

$$(X + p_{01}Y + p_{02}Z + p_{03}) / (p_{20}X + p_{21}Y + p_{22}Z + p_{23}) = \mathrm{x}$$

$$(p_{10}X + p_{11}Y + p_{12}Z + p_{13}) / (p_{20}X + p_{21}Y + p_{22}Z + p_{23}) = \mathrm{y}$$

~~$(p_{20}X + p_{21}Y + p_{22}Z + p_{23}) / (p_{20}X + p_{21}Y + p_{22}Z + p_{23}) = 1$~~

Finding the projection matrix

$$(\quad X + p_{01}Y + p_{02}Z + p_{03})/(p_{20}X + p_{21}Y + p_{22}Z + p_{23}) = \mathrm{x}$$
$$(p_{10}X + p_{11}Y + p_{12}Z + p_{13})/(p_{20}X + p_{21}Y + p_{22}Z + p_{23}) = \mathrm{y}$$

Bring denominator on other side:

$$(\quad X + p_{01}Y + p_{02}Z + p_{03}) = \mathrm{x}(p_{20}X + p_{21}Y + p_{22}Z + p_{23})$$
$$(p_{10}X + p_{11}Y + p_{12}Z + p_{13}) = \mathrm{y}(p_{20}X + p_{21}Y + p_{22}Z + p_{23})$$

These are two linear equations
with the p_{xy} being 11 unknowns to solve for.

Finding the projection matrix

- We now solve these equations to find values for projection matrix P:

$$\begin{pmatrix} 1 & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{pmatrix}$$

- In consequence, P satisfies $P(M_i)=m_i$ for all the initial points and must be the projection matrix of the camera because there is a unique solution.

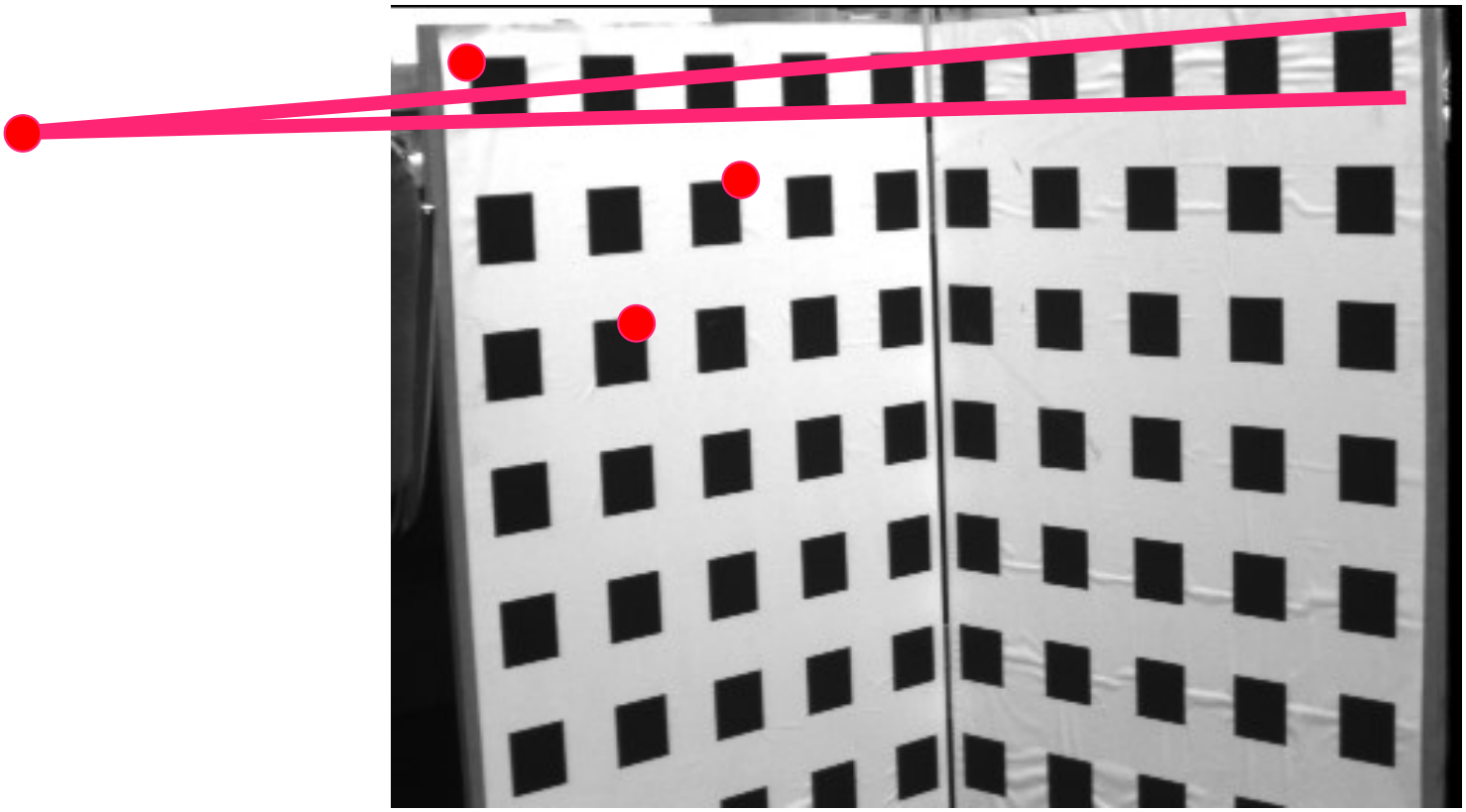
Pitstop

- We have seen, if we have 6 Points in 3D and their projected pixel position, we can find the camera matrix P.

- How do we find 6 points and their projection?

Finding the projection matrix

- Take a photo of something you know
  Identify known points in 3D in the photo



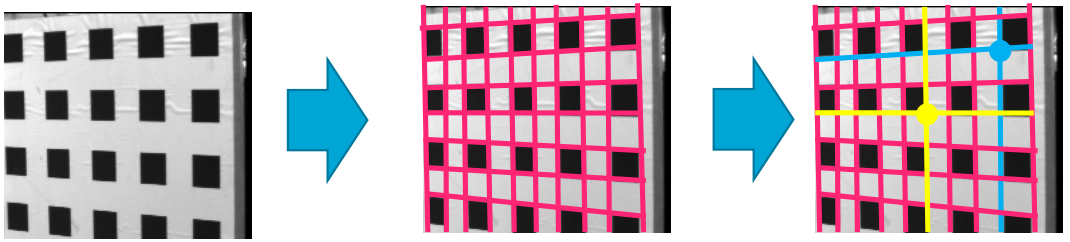
Finding the projection matrix

One way to identify points in the image:

(i) Edge detection

(ii) Straight line fitting to the detected edges

(iii) Intersect lines to find images corners



Finding the projection matrix

- Imprecision can introduce errors:
  - world positions on the object are not perfect
  - pixel locations of projections are not perfect

- How to make things more robust?
  - Use more than 6 Points!
  - solve in a least-square sense

    (rule of thumb: $5n$ constraints for n unknowns)

Pitstop

- We can find the projection matrix of a camera by
  - Taking a photo of a known object
  - Choosing known points and observe their projection on the image
  - Setup a linear system with these points - $P(M_i)=m_i$ – and solve for the entries of P.

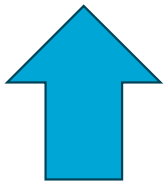
- What about the actual position/orientation of the camera?
  - Hard to determine from the matrix P alone…
  - We need to decompose P into the intrinsic and extrinsic camera, which makes it easier

3D Reconstruction

- How do cameras work?
  - How to model cameras mathematically?
- How to find the pose and calibrate a camera?
- How to reconstruct a scene ?
- How to calibrate from photo collections?

From Camera Matrix to Intrinsic/Extrinsic Matrix

$$P$$



We saw how to find this matrix

How can we find the actual orientation, position and intrinsic matrix?

Complete Camera Model

- Let's understand how orientation and camera position relate to the matrix

$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$

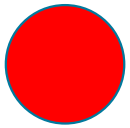
$$K \quad (R \; t)$$

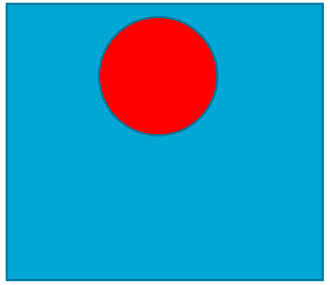
- Orientation
- Translation
- BUT: The values are not what they seem!

Calibration – Extracting Orientation

- Why is the camera orientation not directly the rotation matrix?

$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$



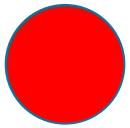


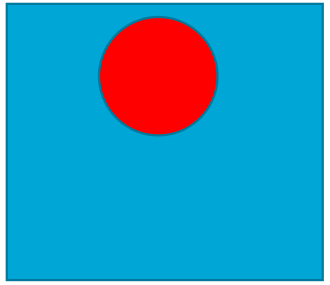
Image

Calibration – Extracting Orientation

- Why is the camera orientation not directly the rotation matrix? Matrix manipulates points in the world!

$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$





Image

Calibration – Extracting Orientation

- Why is the camera orientation not directly the rotation matrix? Matrix manipulates points in the world!

$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$

Camera orientation corresponds to the inverse

$$\begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}^{-1} = \begin{pmatrix} r_{00} & r_{10} & r_{20} \\ r_{01} & r_{11} & r_{21} \\ r_{02} & r_{12} & r_{22} \end{pmatrix}$$

For rotations the inverse is the transpose

Camera Center?

- Translation: Is this translation vector the position of the camera center?

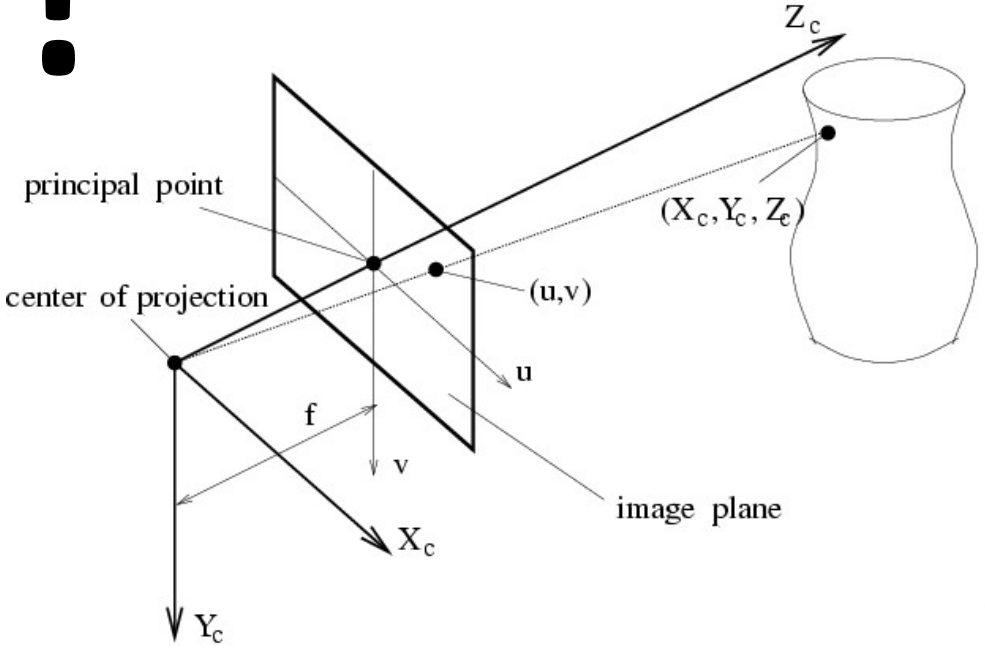
$$P = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$

- No, it is not…

Calibration – Extracting Camera Center

- If it were, then its projection should not give a finite point!

$$\begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix} \begin{pmatrix} t_0 \\ t_1 \\ t_2 \\ 1 \end{pmatrix} = ?$$



Calibration – Extracting Camera Center

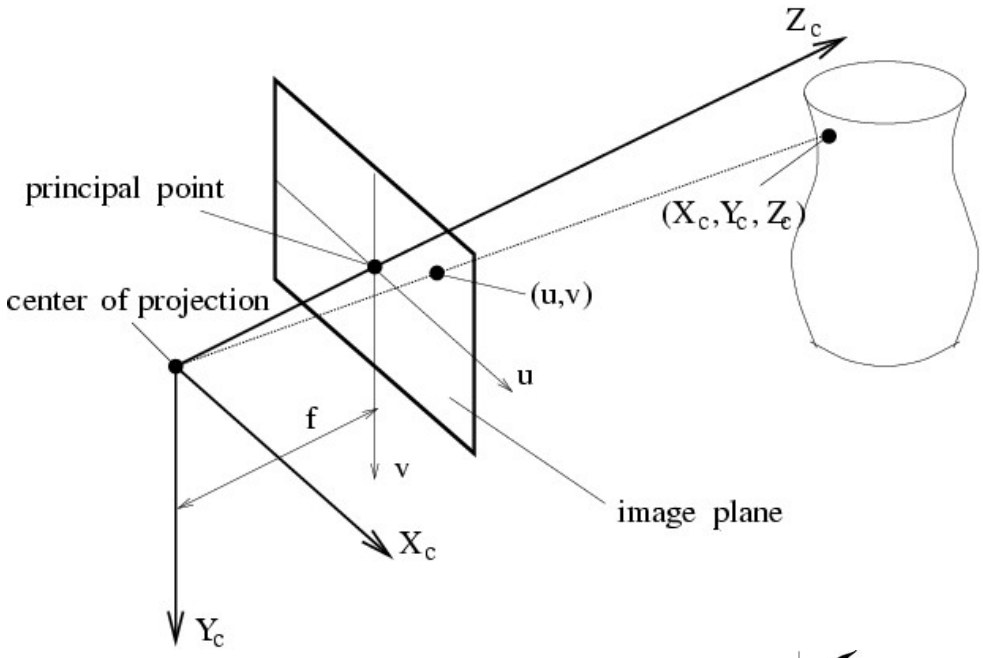
- How can we find the camera center C from the matrix ?

$$\begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix}$$

Calibration – Extracting Camera Center

- Changing C has no influence on R;
  it should be possible to derive C from t!
- It can be shown that :

$$K\begin{pmatrix} R & t \end{pmatrix}\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$



- Therefore, we have : $t := -RC$

Calibration – Extracting Camera Center

- Proof that **t = -RC**:

$$K\begin{pmatrix} R & -RC \end{pmatrix}\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ 1 \end{pmatrix} = K((RC) - RC) = 0$$

- Thus, **$C=-R^{-1}t$**, so if P=K (R t) are known, we can find the camera position

Pitstop - Calibration

- We have seen how to find the camera matrix P
- We know $P = K (R^T - R^T C)$,

  If P is in this form, we can easily find the camera's :
- Orientation R
- Center C

- BUT: How can decompose P into matrices K and $(R^T - R^T C)$?

Camera Matrix Decomposition

- Observation:

- P = $\begin{pmatrix} 1 & p_{01} & p_{02} & p_{03} \\ p_{10} & p_{11} & p_{12} & p_{13} \\ p_{20} & p_{21} & p_{22} & p_{23} \end{pmatrix}$ = K(R^T , -R^TC) = (KR^T , -KR^{T}C)

Camera Matrix Decomposition

- Observation: $P = K(R^T, -R^TC) = (KR^T, -KR^TC)$
- $KR^T = \begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{10} & r_{20} \\ r_{01} & r_{11} & r_{21} \\ r_{02} & r_{12} & r_{22} \end{pmatrix} = \begin{pmatrix} 1 & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{pmatrix}$
- A QR Decomposition decomposes any matrix UNIQUELY into: a rotation R and a triangular matrix Q!
- Hence, if we use the entries in P corresponding to KR^T, the decomposition has to yield K and R^T

Notation

- We understood the relationship:
- $P = K(R^T , -R^TC) = (KR^T , -KR^TC)$

- To reduce clutter, we might write:
- $P = K(R' , t')$
- Note this is just a renaming, here: $t' = -R^TC$ and $R' = R^T$

Complete Overview:

- Shoot image of known object
- Detect points of interest
- Solve camera matrix with these points
- Decompose into intrinsic/extrinsic matrix
- Derive position and orientation

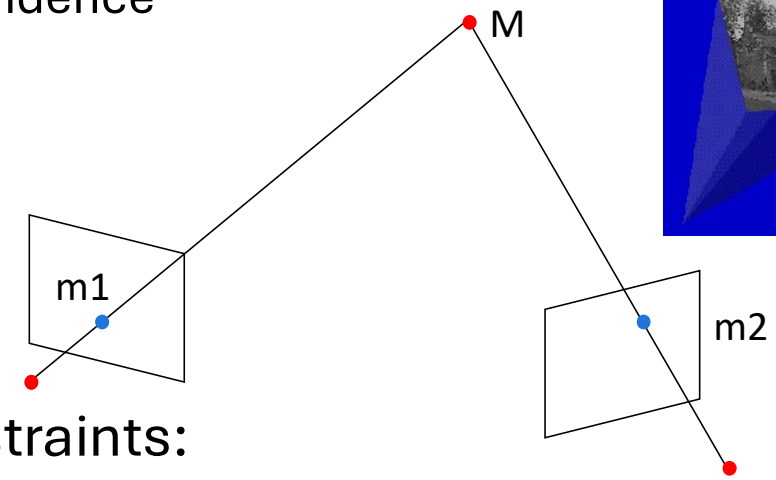
Questions???

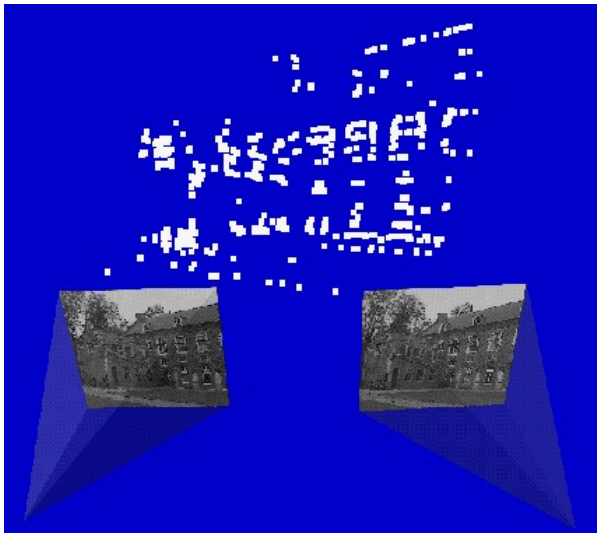
Today: Cameras

- How do cameras work?
  - How to model this mathematically?
- How to find the pose and calibrate a camera?
- How to reconstruct a scene ?
- How to calibrate from photo collections?

How to compute stereo

- Let's assume:
  - two **calibrated cameras (known matrix)**
  - Image point correspondence





- Corresponds to constraints:

  $P_1 M = m_1$ and $P_2 M = m_2$

  4 equations-> we can reconstruct M

How to find point correspondences?

- Simple Solution:
  Ask user…

- Naïve solution:

  Exhaustive search and compare neighborhoods around pixels

- Smarter solution:

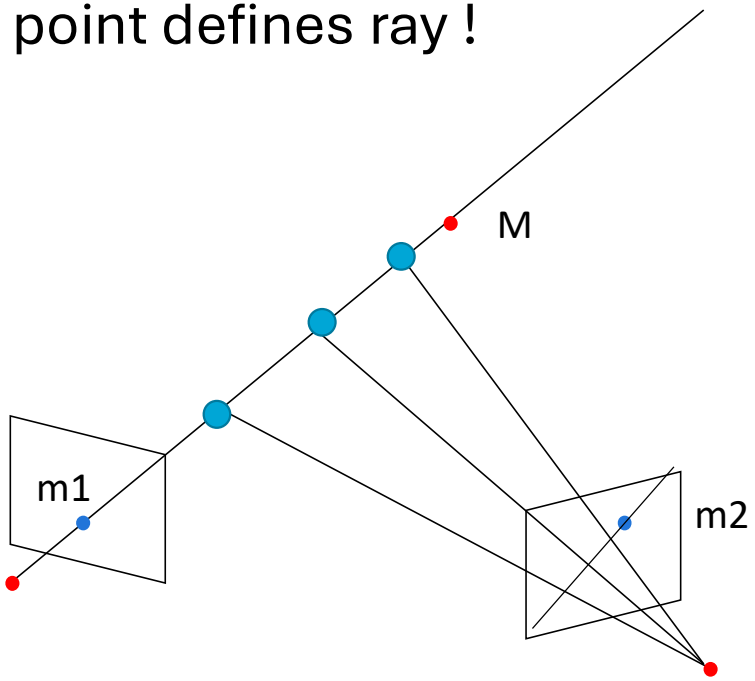
  Find feature points and match those

- “Smartest solution”:

  Find feature points, but only match what can geometrically work

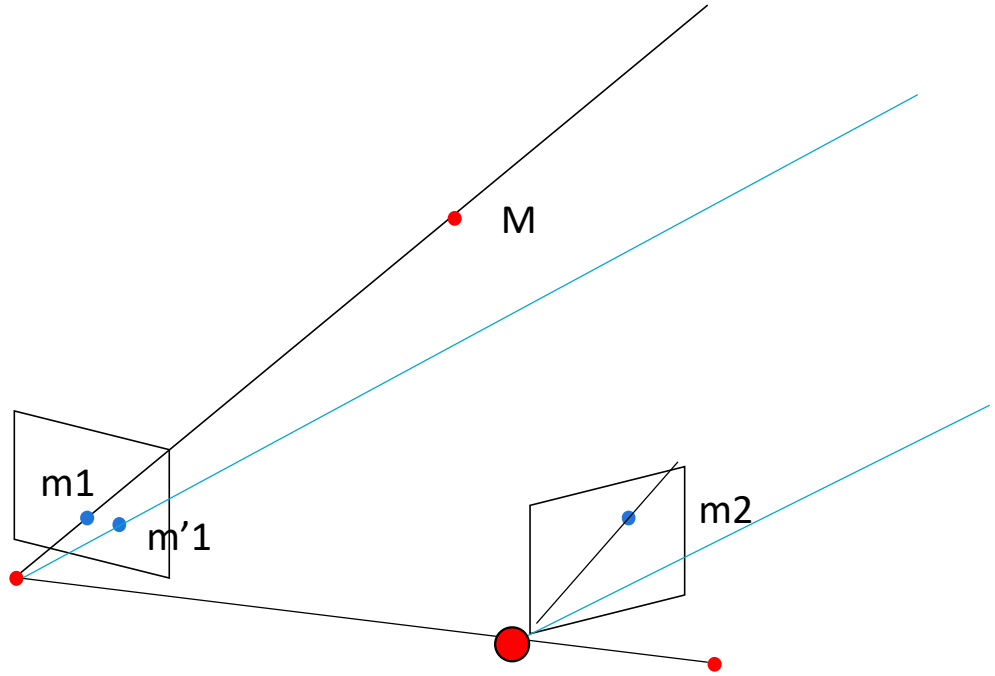
“Smartest solution”

- Single camera: image point defines ray !



“Smartest solution”

- Single camera: image point defines ray !



- Line in camera 2 is called the epipolar line
- All epipolar lines meet in an epipolar point!

“Smartest solution”

- Look along the line for matching pixel values in a neighborhood
- Image in camera 1:

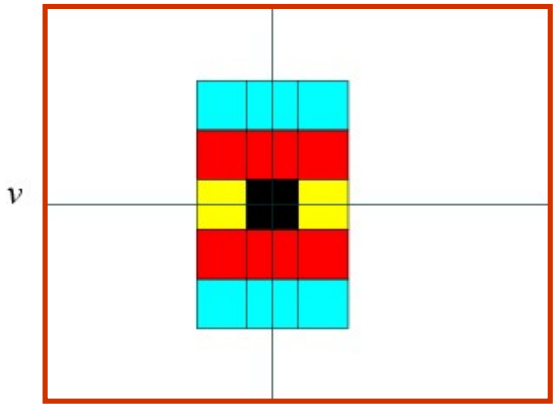
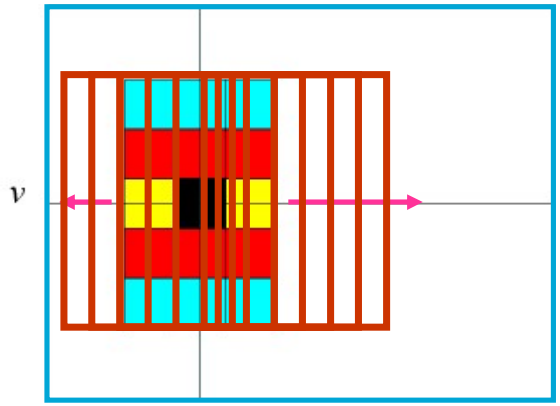


Image in camera 2:



- Try all positions until the position of the most similar pixels is found.

So far so good...

- BUT:
- What happens if the cameras are **not calibrated**?

What do 2 images tell us about cameras?

- Is it possible to reconstruct the 3D configuration from 2 images?
- Probably not because we cannot obtain:
- Absolute locations
- Absolute orientation
- Scale!
- “Rancor effect”

TU Delft

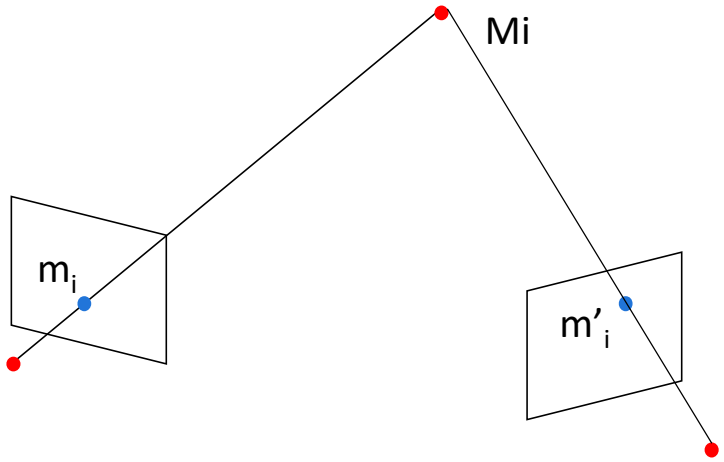
No absolute orientation or position

- Remember camera:

$$\begin{pmatrix} a_x & 0 & x_0 \\ 0 & a_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{00} & r_{01} & r_{02} & t_0 \\ r_{10} & r_{11} & r_{12} & t_1 \\ r_{20} & r_{21} & r_{22} & t_2 \end{pmatrix} =: K \begin{pmatrix} R & t \end{pmatrix} =: P$$

- Given

$$\mathrm{PM_i} = \mathrm{m_i}$$
$$\mathrm{P'M_i} = m'_{\mathrm{i}}$$



, where P, P' are the projections, M_i the world and m_i, m'_i the corresponding projected points

No absolute orientation or position

- Assume we have a solution for all M_i and m_i: $P := K(R \quad t)\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} x \\ y \\ w \end{pmatrix}$

- Affine transform leads to new solution:

$$K(R \quad t)\begin{pmatrix} R' & t' \\ 0 & 1 \end{pmatrix}\begin{pmatrix} R' & t' \\ 0 & 1 \end{pmatrix}^{-1}\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} x \\ y \\ w \end{pmatrix}$$

$$K(R^* \quad t^*) \qquad \begin{pmatrix} X' \\ Y' \\ Z' \\ W' \end{pmatrix}$$

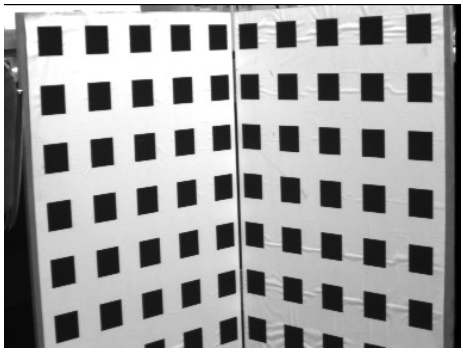
What if I know the intrinsic matrix?

- We have seen that we can find it by making a photo of a known object.

Finding the projection matrix

- Solve for the 4x3 projection matrix P:

  M_i (World Pos) and m_i (Image Pos) are known, which means we can setup many equations of the form: $P\,M_i = m_i$



What if we assume that the camera defines the center of the world?

$$K(R^T \quad -RTC) \quad = K(Id \quad 0) = (K \quad 0)$$

Does knowing K help with two photos?

What if I know the intrinsic matrix?

- If K is known:

$$K\begin{pmatrix} R & t \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

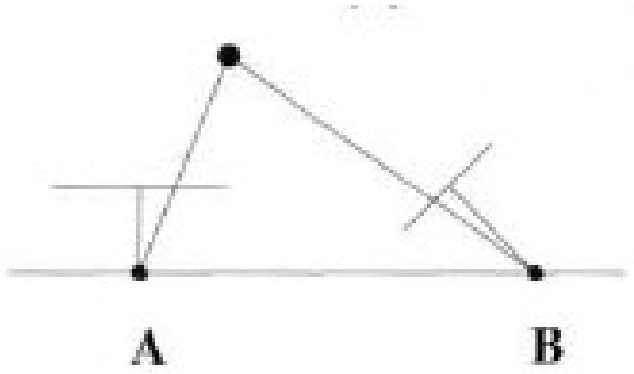
- Idea:
  Multiply all pixel correspondences with K^{-1}

$$\begin{pmatrix} R & t \end{pmatrix}\begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = K^{-1}\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix}$$

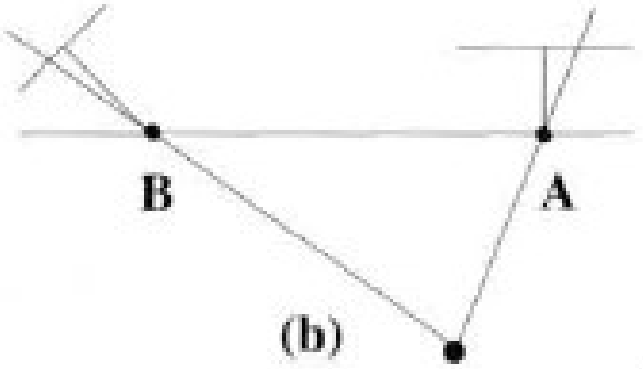
What if I know the intrinsic matrix?

$$\begin{pmatrix} R & t \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix}$$

- Problem becomes simpler, camera matrix is only a rotation and a translation.
- NOTE: One camera can be a reference (defining the origin and axes of the world)
  - Its rotation is identity, its origin is $(0,0,0,1)^T$
- In this setting, only 4 solutions exist!



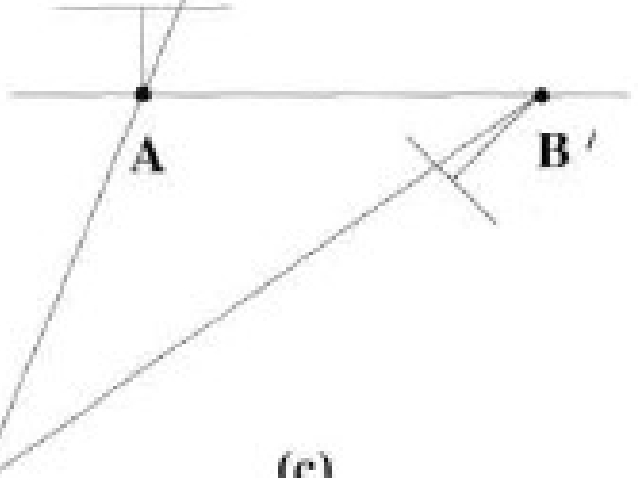
(a)



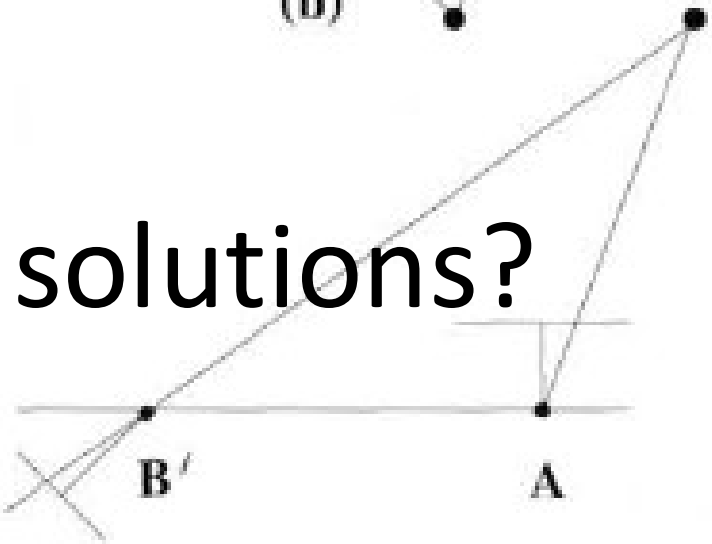
(b)

What now?

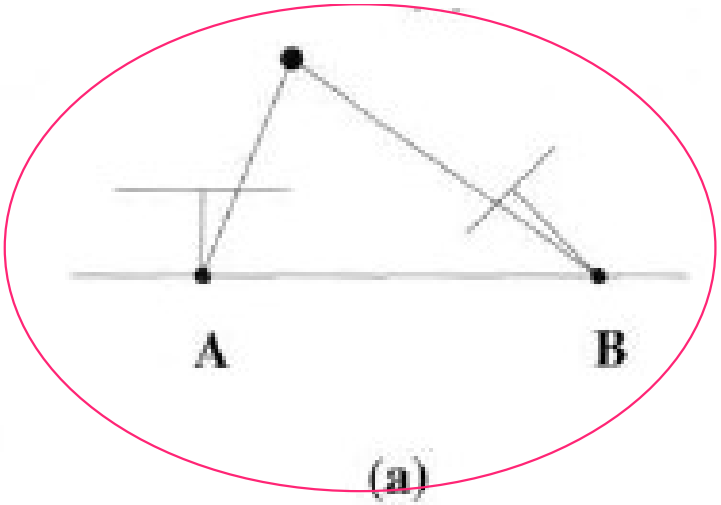
There are still 4 solutions?



(c)



(d)



What now?
There are still 4 solutions?

Only one solution has the points in front of both cameras!
All others cannot actually arise in the real world!

Pitstop

- If you have found the intrinsic matrix of a camera:

- Take camera to a new place
- take two different photos
- you can reconstruct the 3D scene
  [except for scale, absolute orientation and absolute location (Rancor Effect)]
  - One of the two camera views defines your origin and world orientation

What do 2 images tell us about cameras?

For **uncalibrated** cameras (intrinsic matrix not known!),

Scene/Cameras are only defined up to a projective transformation.

Still better than nothing!

In fact, until [Faugeras 1993] most people even doubted that anything useful can be extracted…

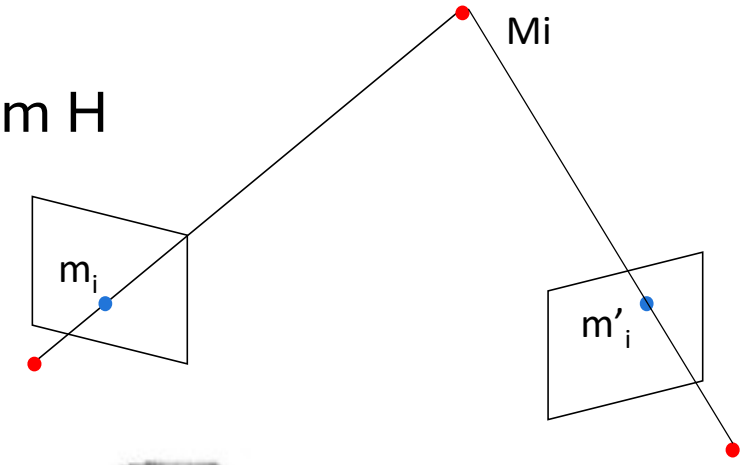
No absolute orientation or position

- Assume we have a solution

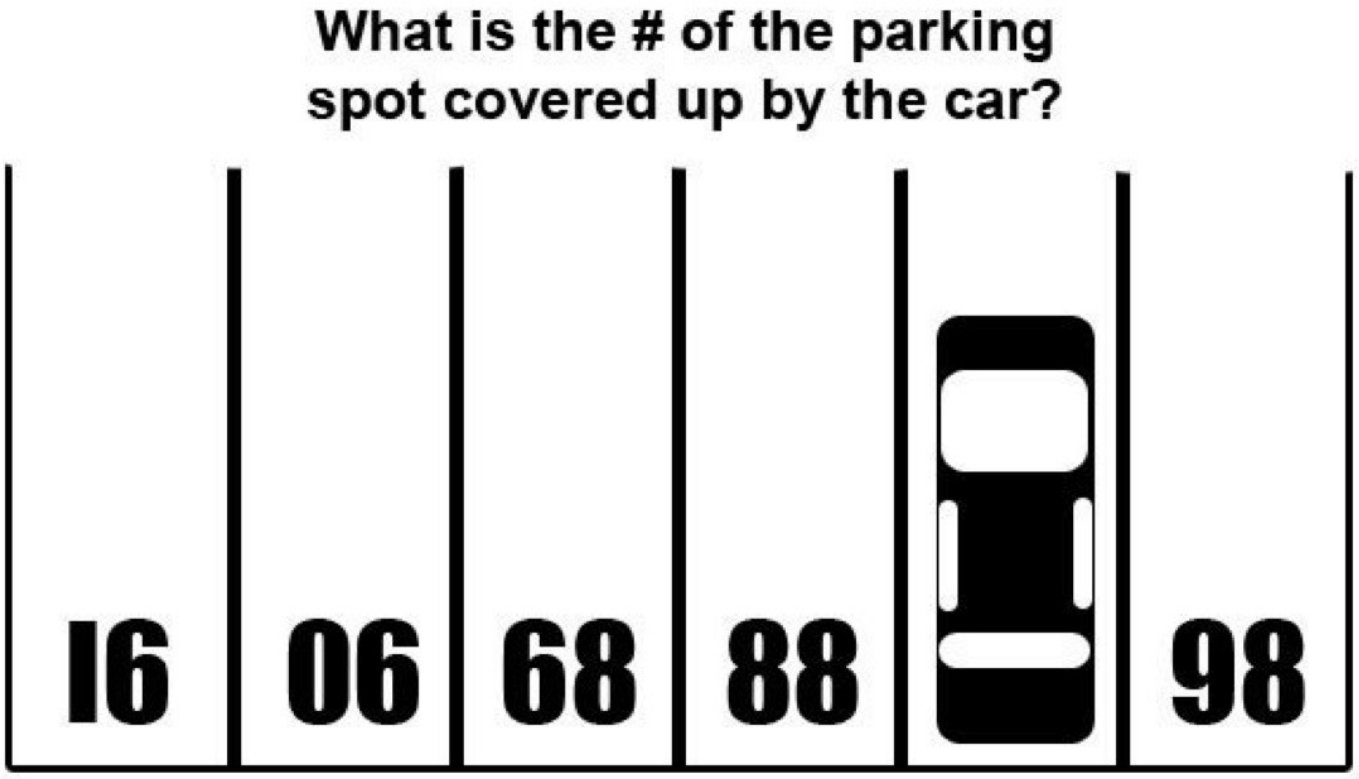
$$\mathrm{P}\mathrm{M}_\mathrm{i} = \mathrm{m}_\mathrm{i}$$
$$\mathrm{P}'\mathrm{M}_\mathrm{i} = m'_\mathrm{i}$$

Then PH, P'H , H^{-1} M_i is

also a solution, for any Projective Transform H

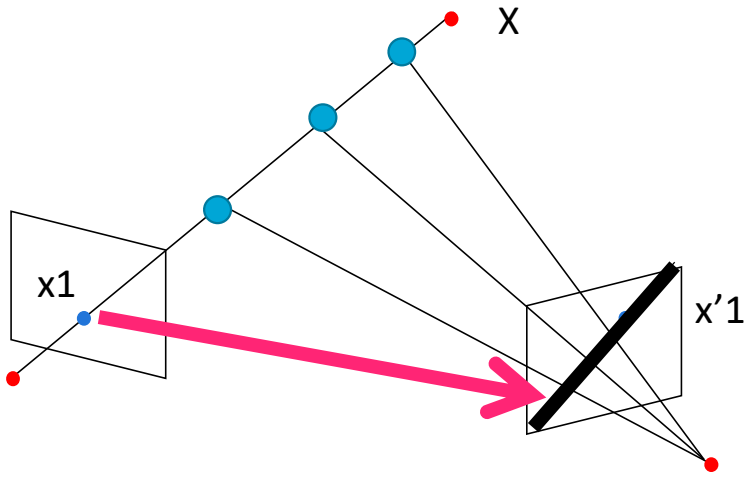


Questions?


What is the # of the parking spot covered up by the car?
16
06
68
88
98

How to calibrate cameras with 2 images

- Projection matrices do not well describe uncalibrated camera relations.



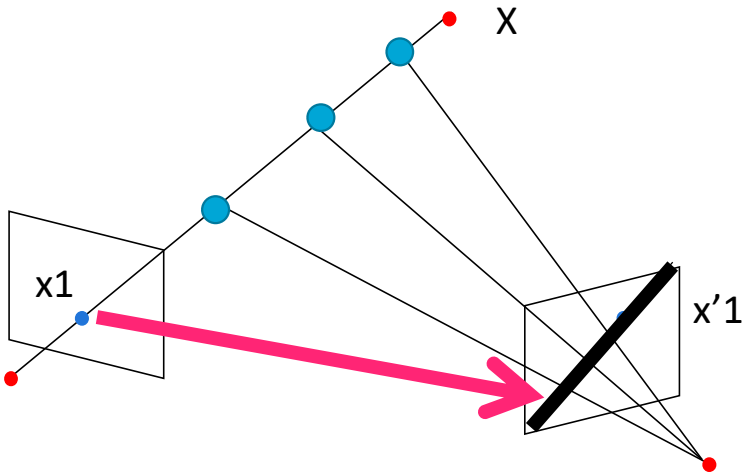
How to calibrate cameras with 2 images

- Projection matrices do not well describe uncalibrated camera relations.

- Uncalibrated camera relations are usually defined by the:

  Fundamental Matrix F

F maps a point
in one image to
the epipolar line of the other.



How to map a point to a line?

Homogeneous Coordinates

Line Point Duality in 2D

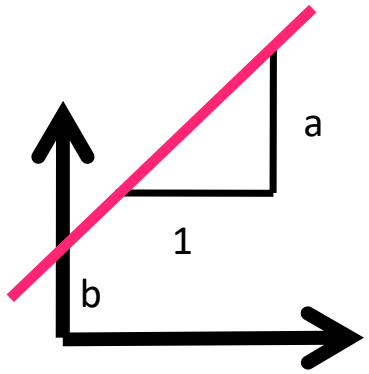
Line equation in 2D:

$y = ax+b$

$\Leftrightarrow$ $ax-y+b=0$

$\Leftrightarrow$ $(a,-1,b)(x,y,1)^t = 0$

$\Leftrightarrow$ $(sa,-s,sb)(kx,ky,k)^t = 0$, proj. equation!



In other words:

Hom. points in 2D are lines!

Dot products in 2D are tests if a point is on a line!

What lines are represented by a point?

Line l through 2 points X,X’

- If X, X’ are points on the line:

$$l^t X = 0$$

$$l^t X' = 0$$

- How can we define l ? $l := X \wedge X'$

Point X on 2 lines l,l’

- If l, l’ are lines through X:

$$l'^t X = 0$$

$$l^t X = 0$$

- How can we find X ?

$$X := l \wedge l'$$

Homogenous Coordinates simplify

- Because we have points at infinity,
  there are no case distinctions for parallel lines !

e.g., (a,-1,b) ^(a,-1,c)= (b-c, ba-ac, 0)

w=0 intersection at infinity!

Interesting side note: Two lines ALWAYS intersect ONCE in projective spaces!

Theorem of Bézout: Two curves of degree n, m generally have n*m intersections

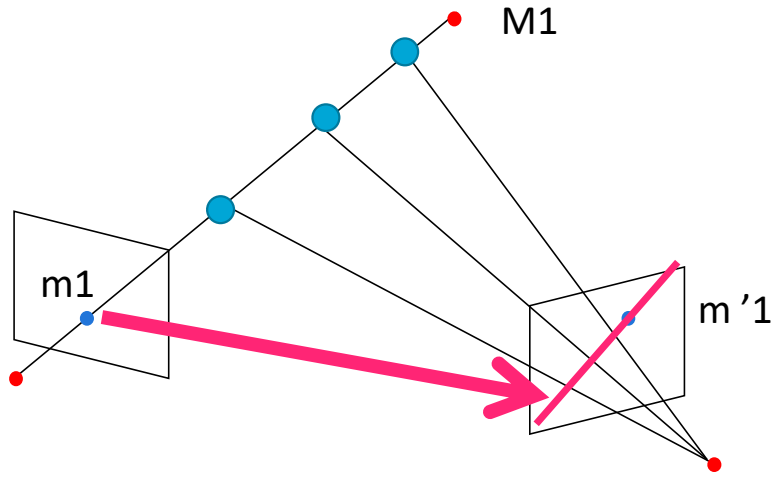
Compute F for uncalibrated cameras

Let mi, m'i be point correspondences in image plane.

Then we want F to satisfy:

$$m'_i \, F \, m_i = 0$$

($F \, m_i$) is the epipolar line for m_i



Computing the Fundamental Matrix

- Big equation system:

$$m'_i \, F \, m_i = 0$$

So we still solve a linear system?

- 8 correspondences are enough

  (smart: 7, but don't worry about that)
  Least-square solution with more correspondences possible

Pitstop – All the information between two images

- F encodes exactly ALL information that two images offer you!
  - It can be shown that we can build a solution with cameras directly from F
  - And a solution does imply exactly one F

Today:

- How do cameras work?
  - How to model this mathematically?
- How to find the pose and calibrate a camera?
- How to reconstruct a scene ?
- How to calibrate from photo collections?

Outlook

Outside scope of this lecture:

3 images from same camera are enough to calibrate

[Stephen J. Maybank and Olivier D. Faugeras. A theory of self-calibration of a moving camera. International Journal of Computer Vision, 8:2, 123-151, 1992]

Uses “magic” that is called trifocal tensor

Time for some music...

Summary: Guide for image collections

- Many photos come with tags…
  - If camera calibration is stored

    –> perfect: take two cameras and reconstruct 3D
  - Camera model and settings stored

    -> find 2 more photos with **same** camera
    with 3 photos you can calibrate the camera
  - Once you have known 3D points you can use those to calibrate other cameras!

Modern Solutions

- They still start with a sparse 3D point cloud reconstruction!
- Newest trend: Gaussian Splatting

- Start with 3D point reconstruction
- Replace points by Gaussians
- Optimize their position, size, and ratios to match the view from the cameras

Now we know it all...

Limitations of Stereo?

Conclusion

- We saw:

- How cameras work
- How to model cameras
- How to use homogenius coordinates
- How to calibrate a camera for known content
- How to calibrate with unknown content
  - How to reconstruct stereo
  - How to be more robust/faster (epipolar lines)

Thank you very much!