Trinity College Dublin

The University of Dublin

V-SENSE

Augmented Reality – Guest Lecture

Valeria Olyunina, Nicolas Moreno de Palma @ Volograms

- Courtesy of Aljosa Smolic, Jan Ondrej

Fundamentals of Multi-view geometry

+ Structure from Motion (SfM)

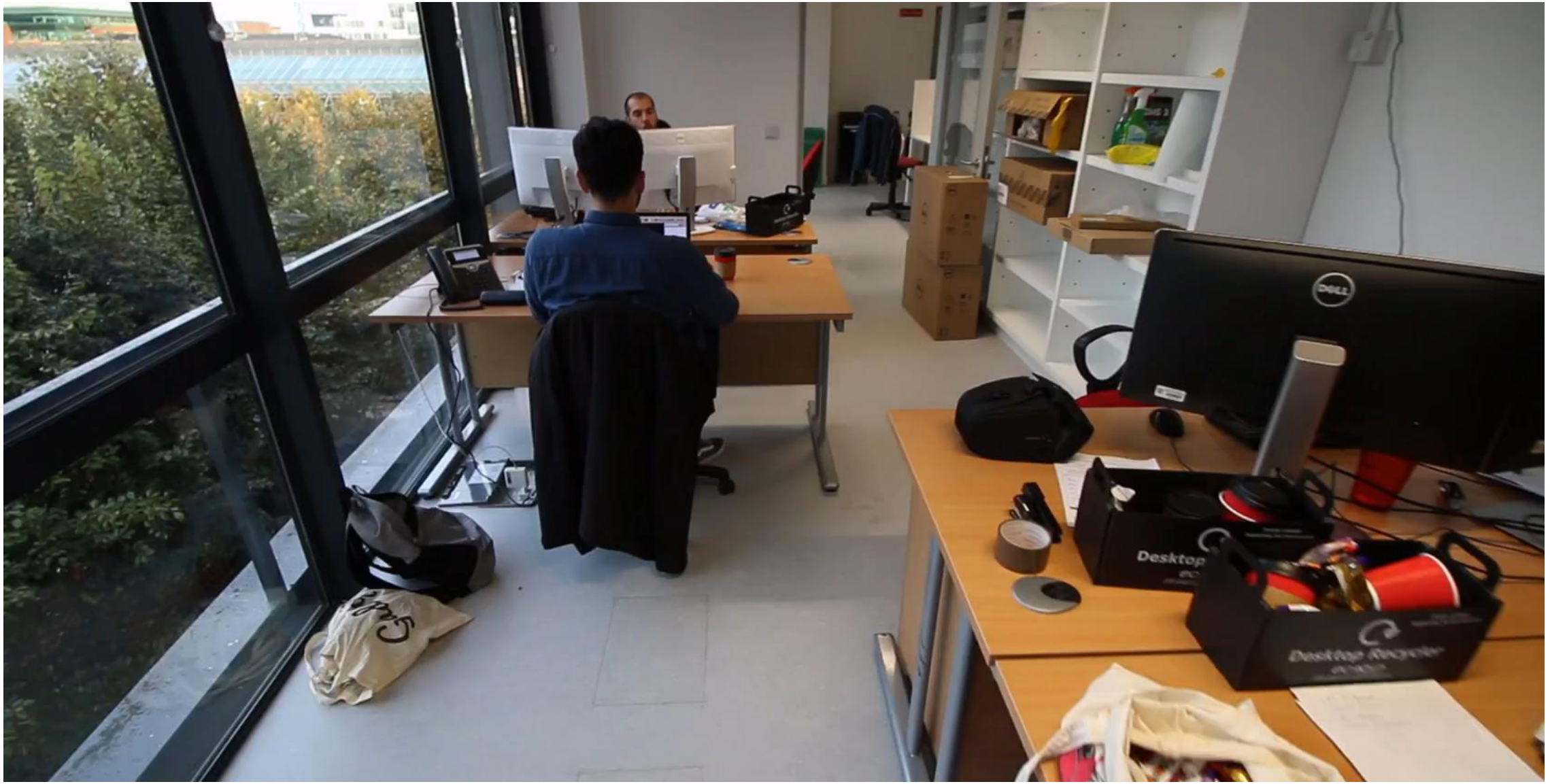
Volumetric Video

Structure From Motion: Dubrovnik

Build from 58,000 images of Dubrovnik on Flickr

Building Rome in a Day
Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski
International Conference on Computer Vision, 2009, Kyoto, Japan.

Original video of office and 3D point cloud - used every 10th frame



Reconstruction by SfM

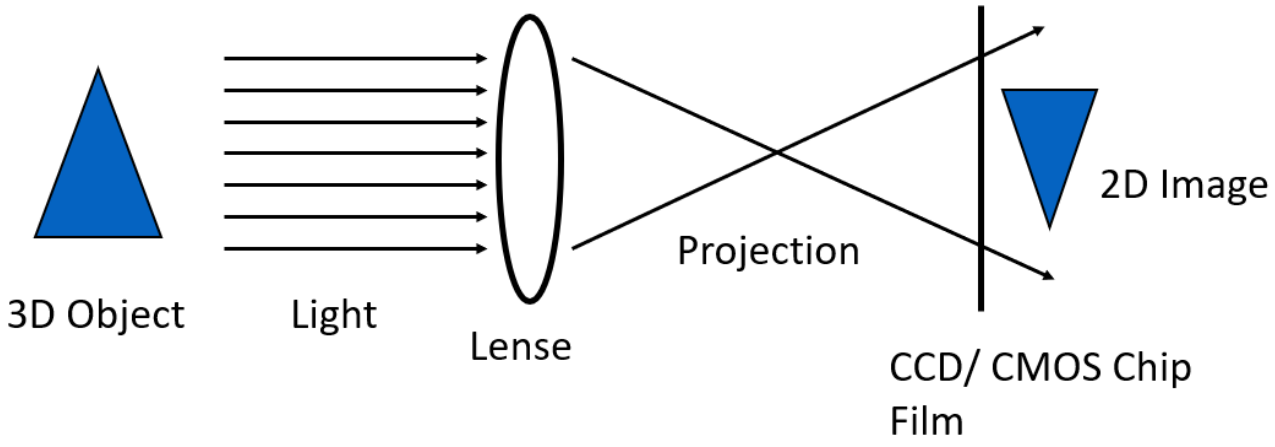
Moynihan, Matthew; Pagés, Rafael; Smolic, Aljosa
A Self-regulating Spatio-Temporal Filter for Volumetric Video Point
1182 , pp. 391-408, Springer International Publishing, 2020, ISBN: 978-3-030-41590-7.

Lecture overview:

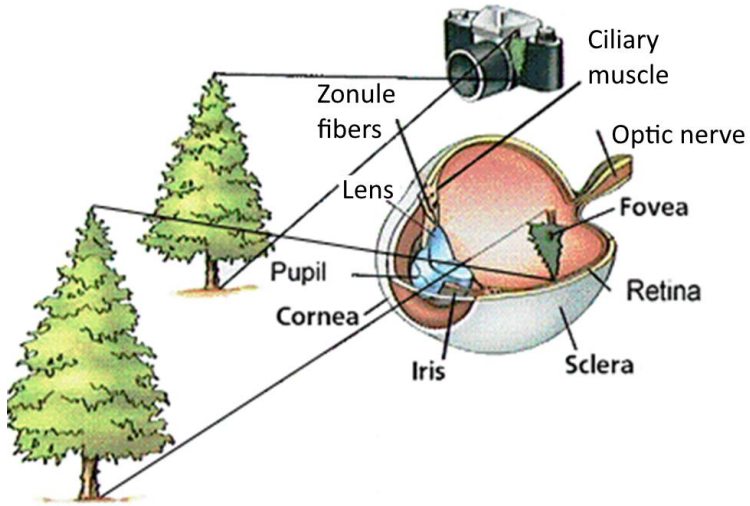
- Camera model (extrinsic and intrinsics matrices). Projection.
- Calibration and camera pose estimation
- Fundamental and Essential Matrix – relationship between 2 cameras
- Structure from Motion

Camera Model
Extrinsic and Intrinsic Camera Parameters

Pinhole Camera Model

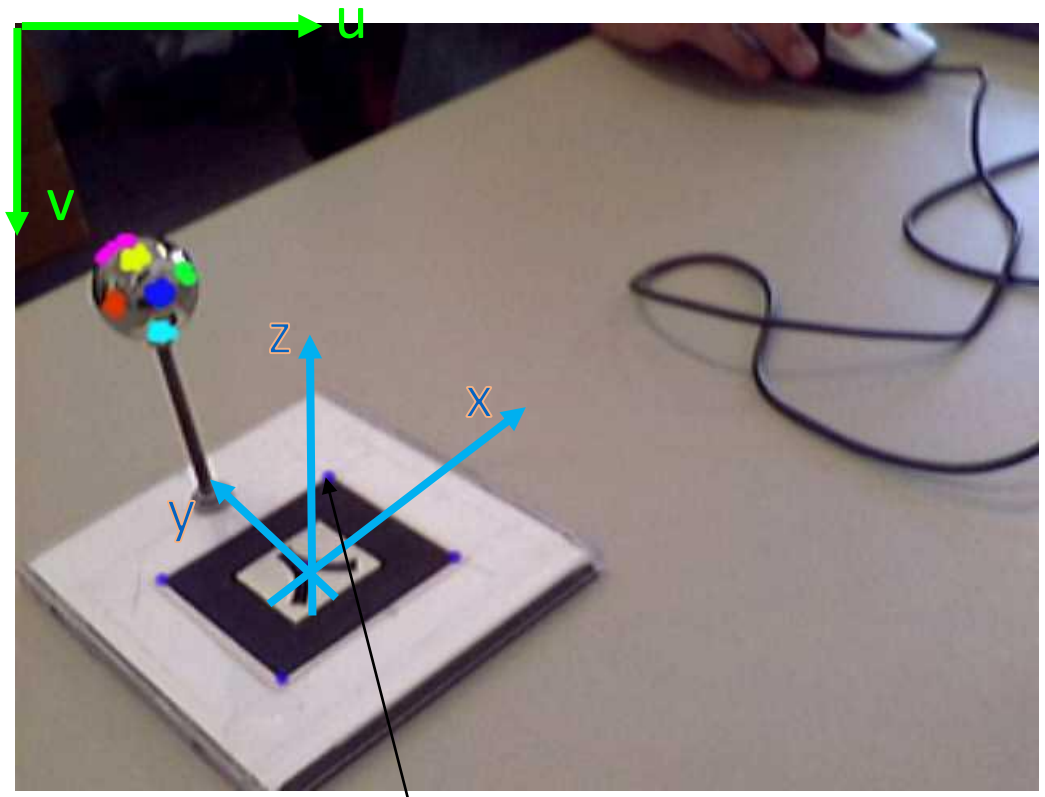


mathematical relationship between the coordinates of a point in 3D-space and its projection onto the image plane of an *ideal* pinhole camera, where camera aperture is a point and there is no lens distortion



Source: [EGBECK]

Coordinate Systems

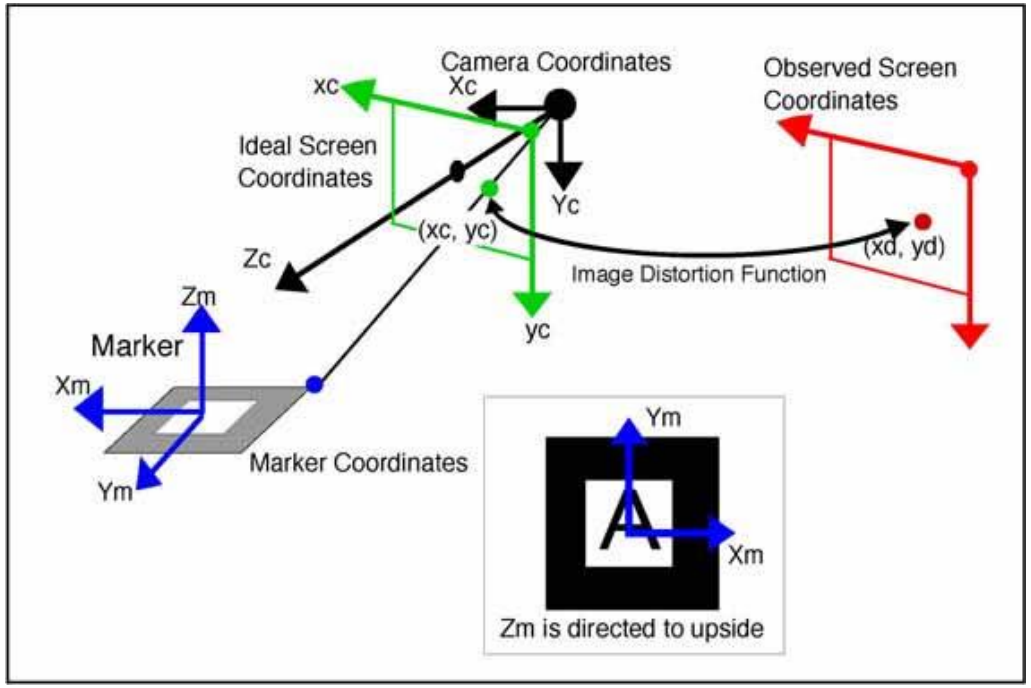


- Example:
  - $P = (40.0, 40.0, 0.0)^T \Leftrightarrow$
  - $p = (196, 284)^T$

- 3D World coordinates:
  - Origin chosen arbitrary. On a mobile phone vertical and horizontal planes are known
  - $P = (x, y, z)^T$
- 3D Camera coordinates:
  - Origin at camera position. Z-axis towards where camera is pointing
  - $P' = (x', y', z')^T$
- 2D image coordinates:
  - Metric units
  - $p' = (u', v')^T$
- 2D pixel coordinates:
  - Usually from top-left corner **in pixels**
  - $p = (u, v)^T$

Extrinsic Transformation

Transformation from *world coords (WC)* into *camera coordinate (CC) system*



$$P' = R\,P + T$$

$$\begin{pmatrix} x' \\ y' \\ z' \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_x \\ r_{31} & r_{32} & r_{33} & t_x \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{R} & \mathbf{T} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

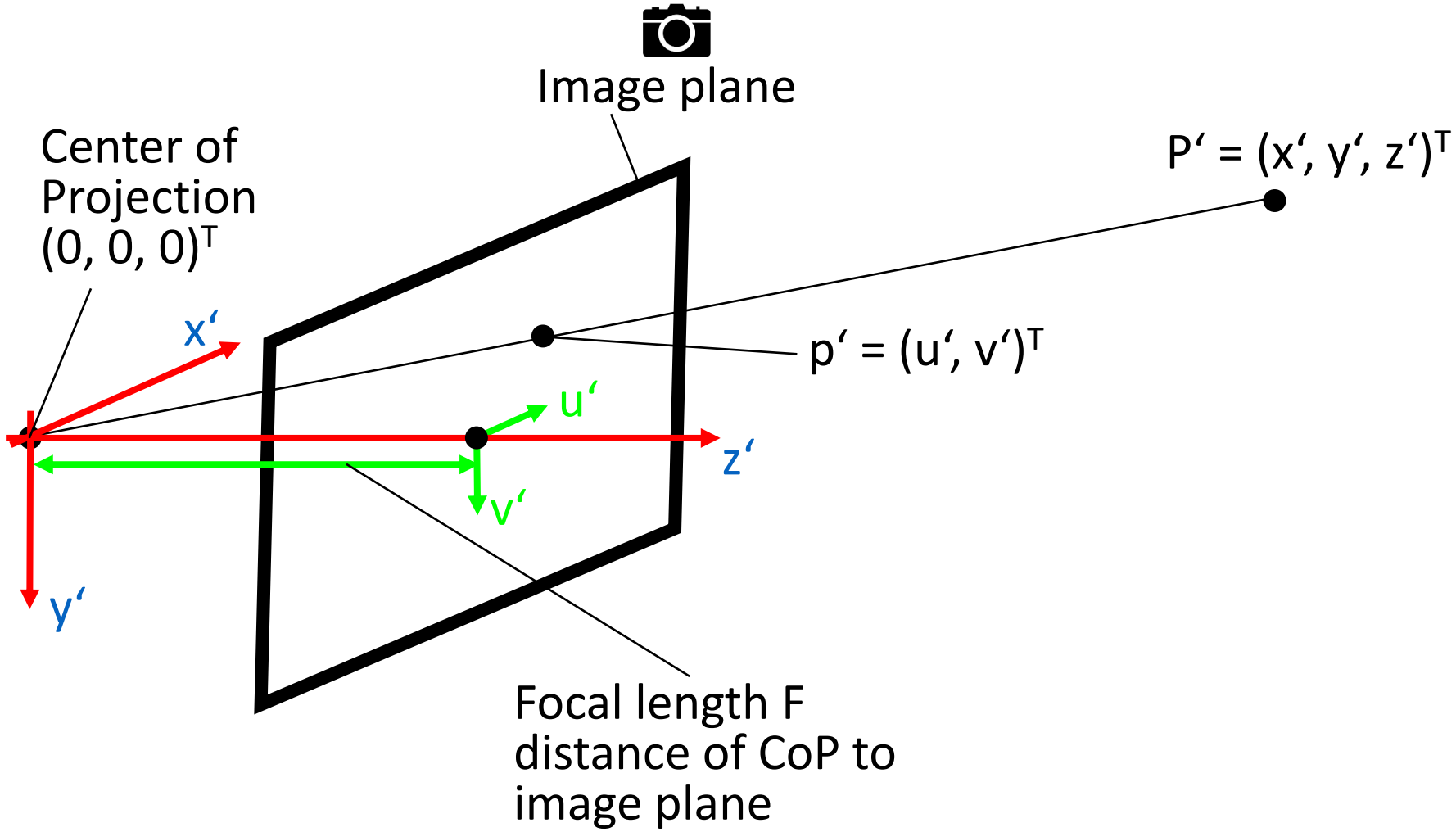
3x3 Rotation Matrix R
(WC rotation in CC)

Translation vector T
(World origin expressed in CC)

Extrinsic matrix E (3x4) $\mathbf{E} = \begin{pmatrix} \mathbf{R} & \mathbf{T} \end{pmatrix}$

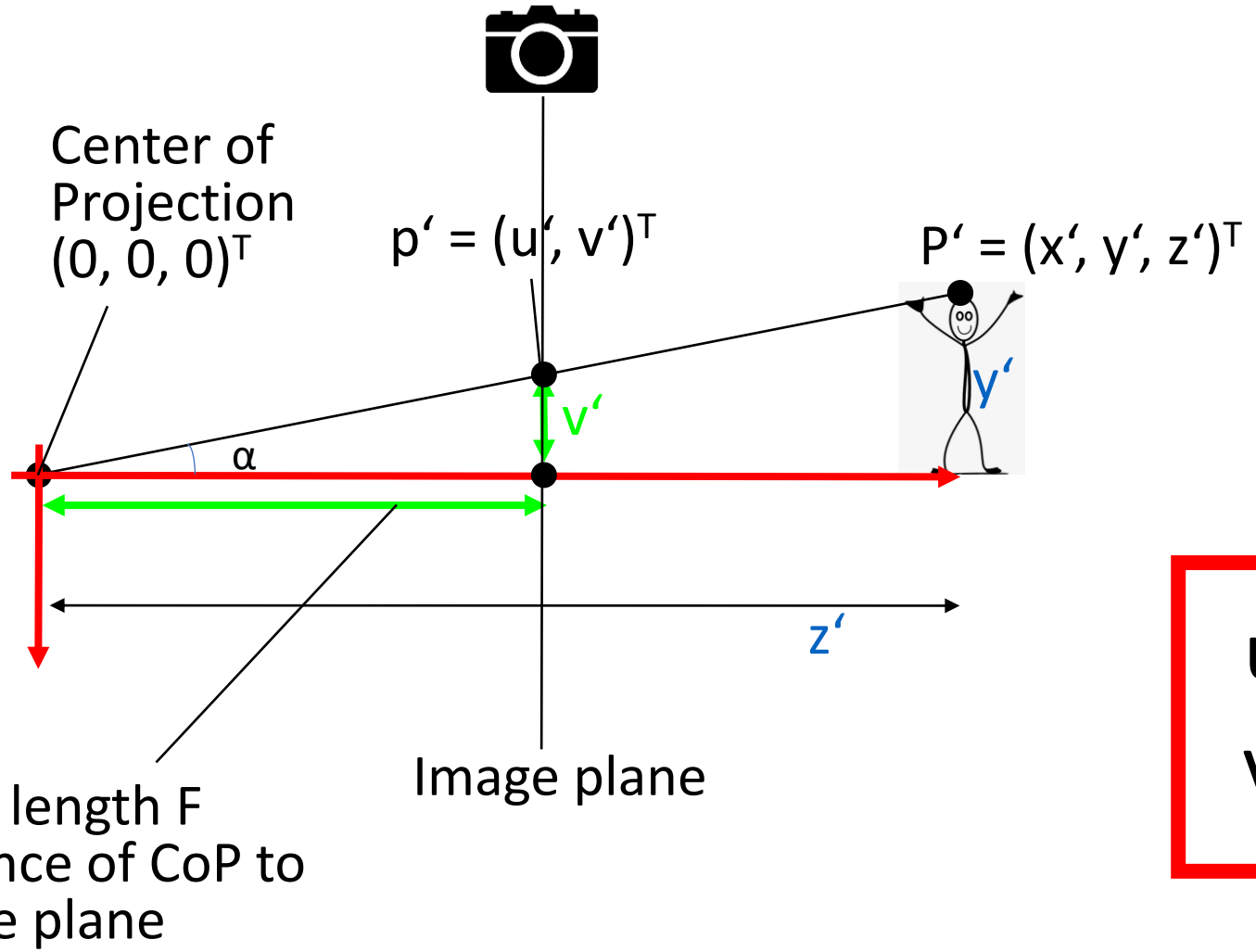
Intrinsic Transformation

Transformation to *image coordinates* – Perspective projection



Intrinsic Transformation

Transformation to *image coordinates* – Perspective projection



Tan α = y’ / z’ = v’ / F

$$u' = F/z' \cdot x'$$
$$v' = F/z' \cdot y'$$

Focal length effect

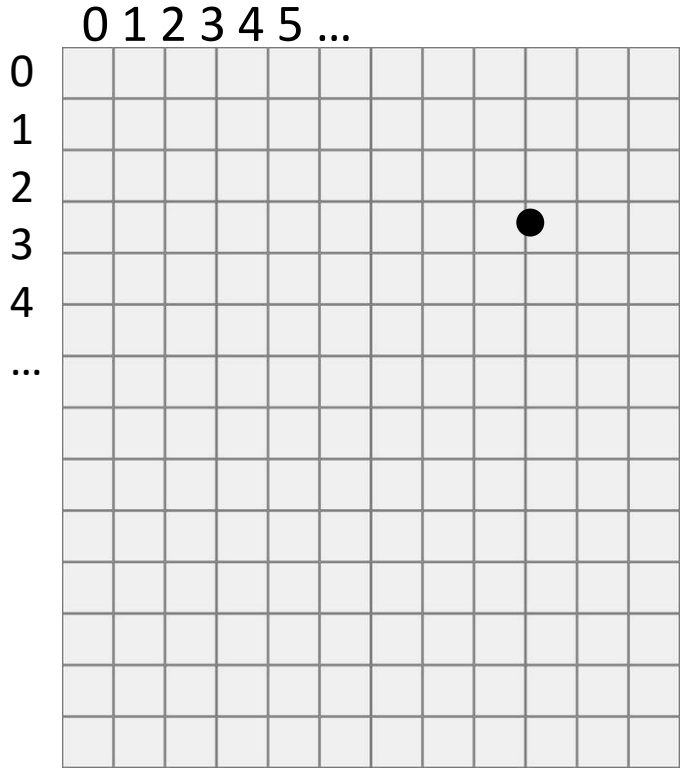
- Distance from camera to subject does not change

Source: https://www.nikonusa.com/en/learn-and-explore/a/tips-and-techniques/understanding-focal-length.html

Intrinsic Transformation

Transformation to *pixel coordinates*

- u‘ and v‘ still in metric unit same as the world 3D coordinate system
- Scale factors (related to physical sensor size on chip) d_u and d_v transform into discrete pixel positions u‘‘ and v‘‘

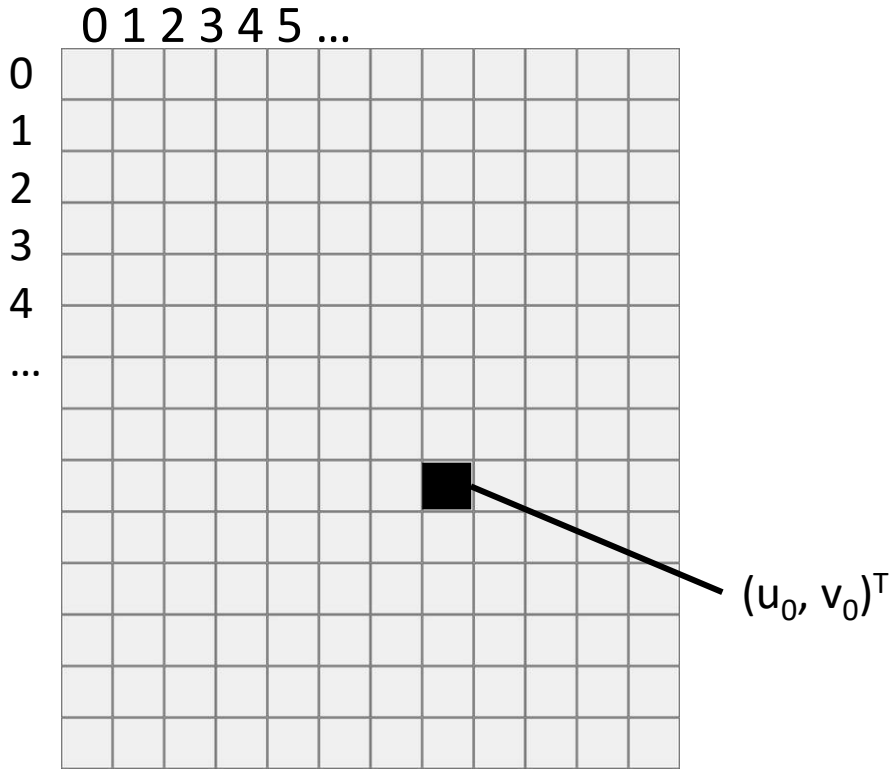


$$u'' = d_u \cdot F/z' \cdot x'$$
$$v'' = d_v \cdot F/z' \cdot y'$$

Intrinsic Transformation

Transformation to *pixel coordinates*

- Principal point u_0 and v_0 - intersection of optical axis and image plane. Relates coordinates to image corner



$$u = d_u \cdot F/z' \cdot x' + u_0$$
$$v = d_v \cdot F/z' \cdot y' + v_0$$

Intrinsic Transformation

- Multiply by z‘

$$u \cdot z' = d_u \cdot F \cdot x' + u_0 \cdot z'$$
$$v \cdot z' = d_v \cdot F \cdot y' + v_0 \cdot z'$$

Intrinsic matrix **K**

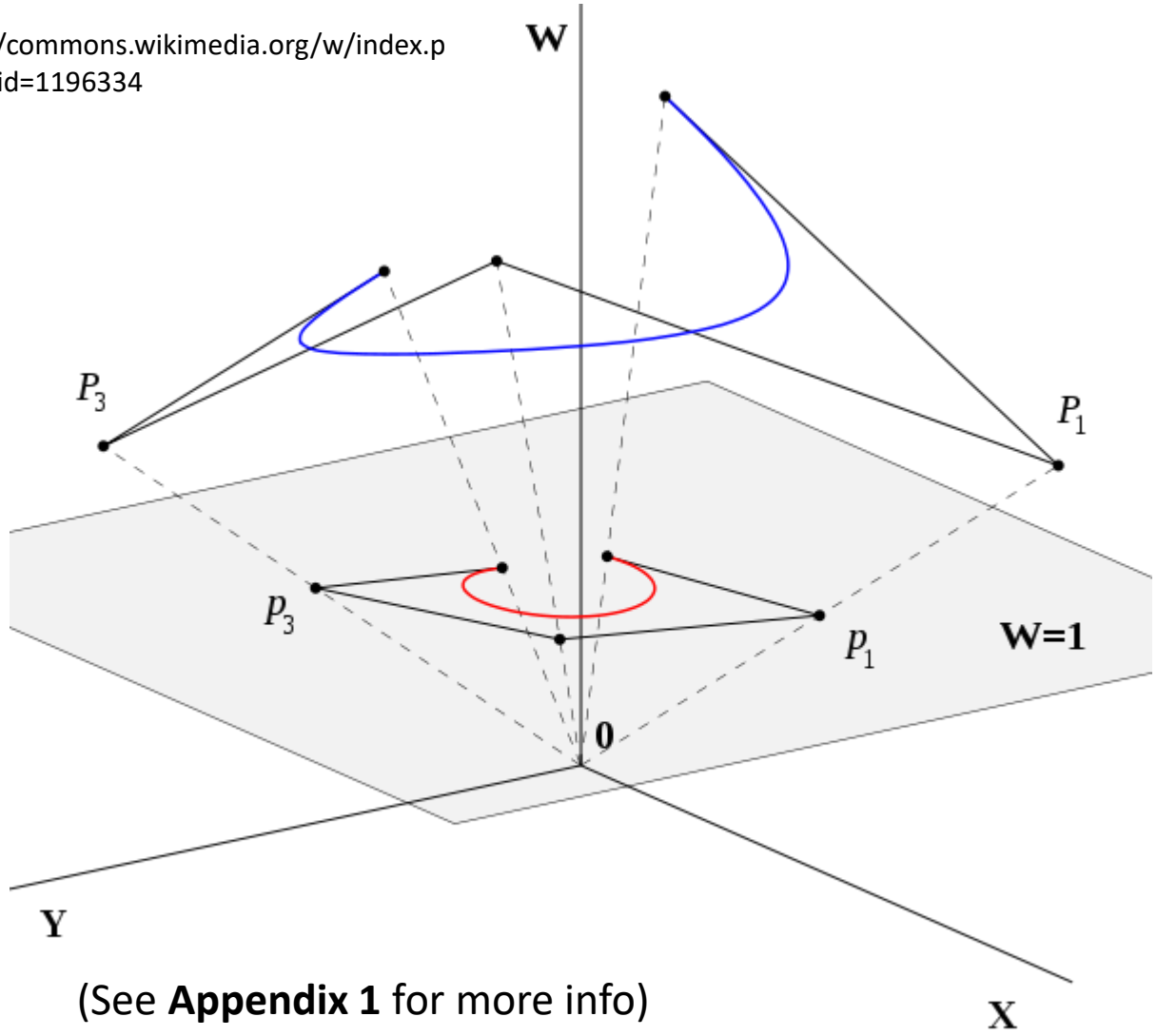
skew factor (angle between two image axes is not equal to 90 degrees)

$$z' \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} F \cdot d_u & s & u_o \\ 0 & F \cdot d_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x' \\ y' \\ z' \end{pmatrix}$$

$= K \cdot P'$

5 degrees of freedom (ommitting s)

Homogenious Coordinates



(See **Appendix 1** for more info)

Consider a Euclidian plane with Carthesian coordinates $\begin{bmatrix} X_c \\ Y_c \end{bmatrix}$

Projected to a space called projective plane $W = 1$ $\begin{bmatrix} X_c \\ Y_c \end{bmatrix} \Rightarrow \begin{bmatrix} X_h \\ Y_h \\ 1 \end{bmatrix}$

Points are projected to lines

Multiplying a point by a scalar remains equivalent with regard to Euclidian plane $\begin{bmatrix} X_h \\ Y_h \\ W \end{bmatrix} \sim \lambda \cdot \begin{bmatrix} X_h \\ Y_h \\ W \end{bmatrix} = \begin{bmatrix} \lambda \cdot X_h \\ \lambda \cdot Y_h \\ \lambda \cdot W \end{bmatrix}$

After computations in homogenious space, recover Carthesian coordinates by division by W $\begin{bmatrix} X_h / W \\ Y_h / W \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} X_c \\ Y_c \end{bmatrix}$

Recovering Carthesian Coordinates

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} a \\ b \\ c \end{pmatrix}$$

Do all calculations in homogenious space

A final step gives the image coordinates (pixel integer)

u = a / c
v = b / c

Calibration Matrix C in Projective Space

$$\begin{pmatrix} \boldsymbol{u} \\ \boldsymbol{v} \\ 1 \end{pmatrix} \sim \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} F \cdot d_u & s & u_0 \\ 0 & F \cdot d_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \boldsymbol{0}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

$p_{hom} = K \cdot A \cdot E \cdot P$

$p_{hom} = C \cdot P$

4 x 4 matrix

3 General Problems in 3D Computer Vision and AR

$$\mathbf{p} = \mathbf{C} \cdot \mathbf{P}$$

P – Real World coord (3D), p – pixel coord (projective space), C – calibration matrix

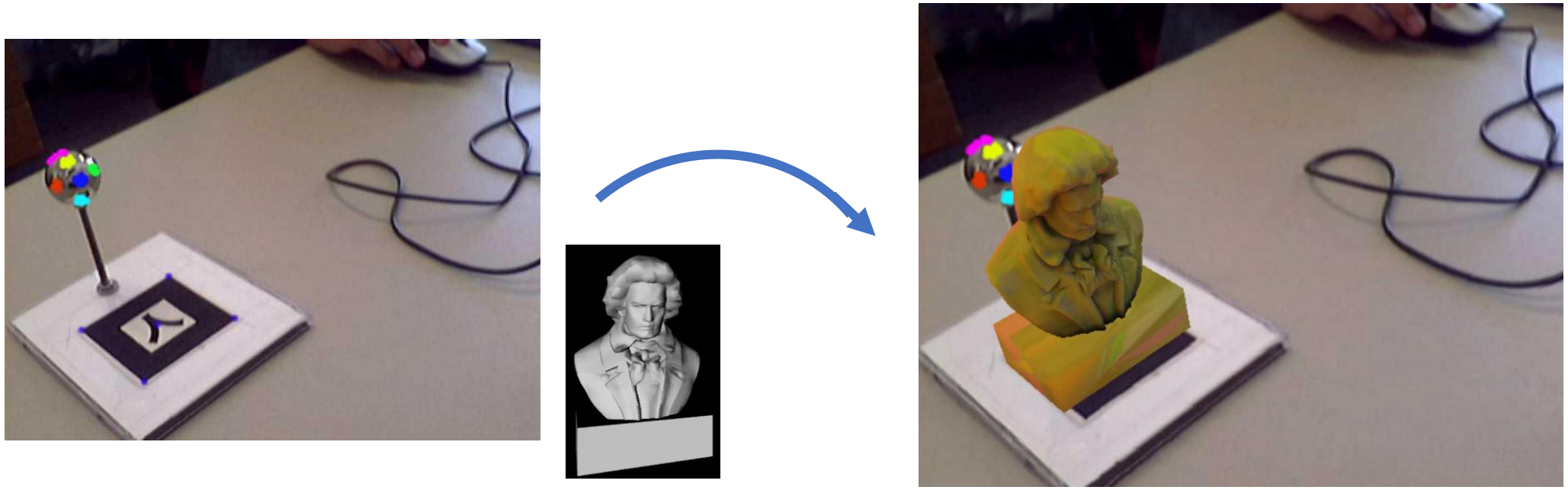
C and **P** known, solving for **p**: **Projection** -> for example, rendering in 3D CG

P and **p** known, solving for **C**: **Calibration**

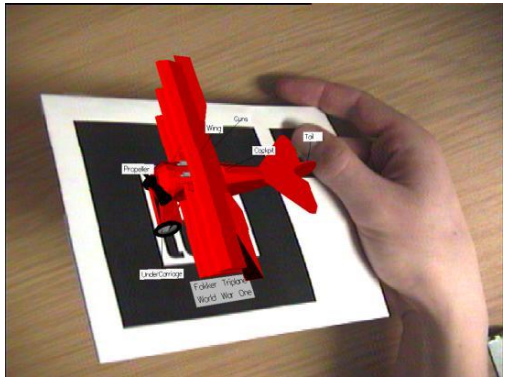
C and **p** known, solving for **P**: **3D Reconstruction**

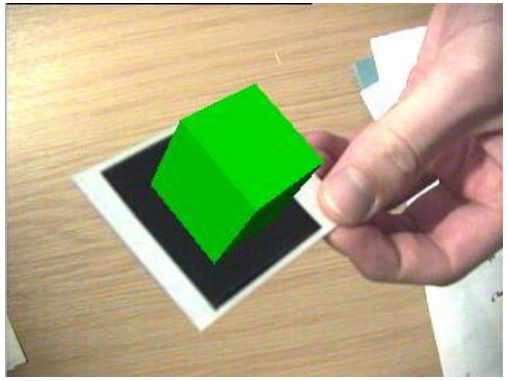
Projection -> 3D object into AR video

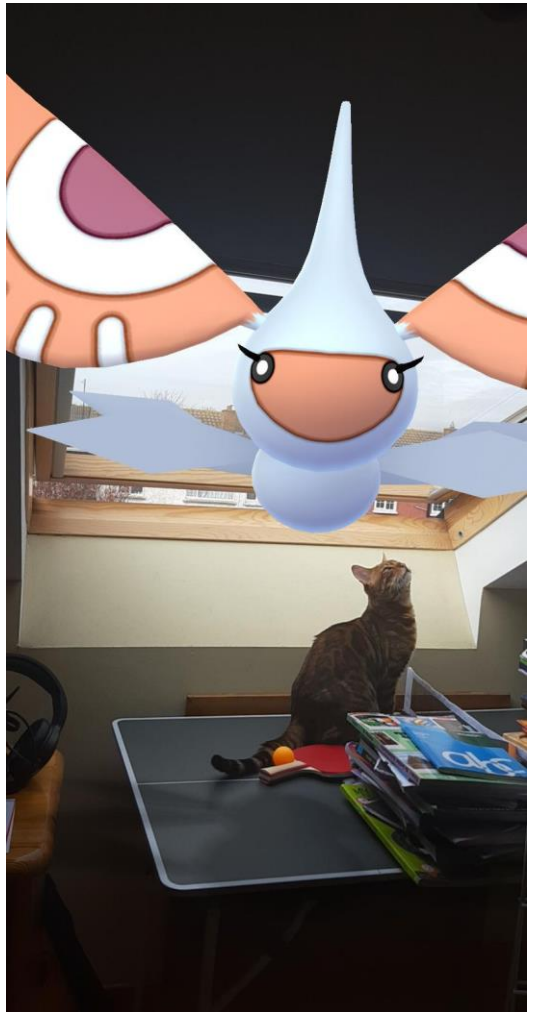
- If we know extrinsic and intrinsic matrices (E and K), can we project 3D points into an video taken by a camera?
  - Camera calibration C in this case calculated from the marker
  - For every vertex of the 3D model multiply vertex coord by calibration matrix C



Examples







Pokemon GO

3 General Problems in 3D Computer Vision and AR

$$\mathbf{p} = \mathbf{C} \cdot \mathbf{P}$$

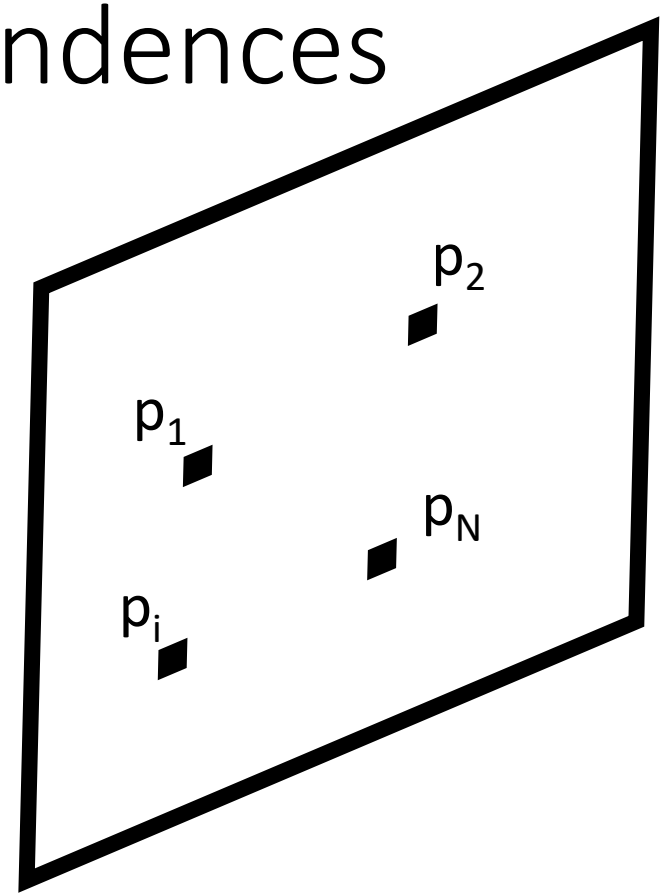
P – Real World coord (3D), p – pixel coord (2D), C – calibration matrix

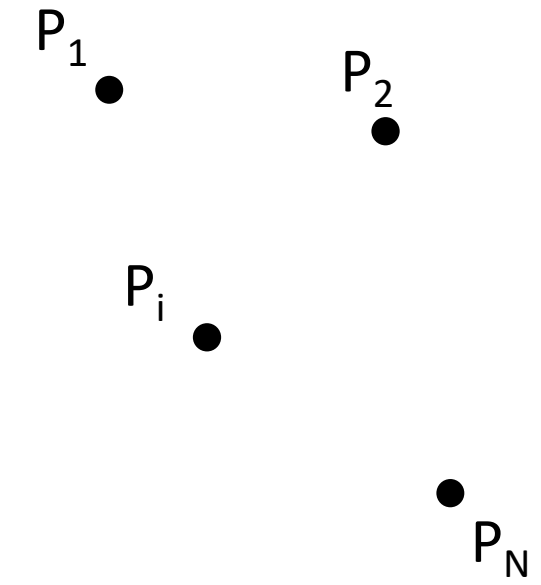
C and **P** known, solving for **p**: **Projection** -> for example, rendering in CG

P and **p** known, solving for **C**: **Calibration**

C and **p** known, solving for **P**: **3D Reconstruction**

Correspondences





We have a number N of 3D points with correspondences in the images

$P_i = (x, y, z)^T <=> p_i = (u, v)^T$

Calibration - Direct Solution

Unknown both intrinsics and extrinsics matrices

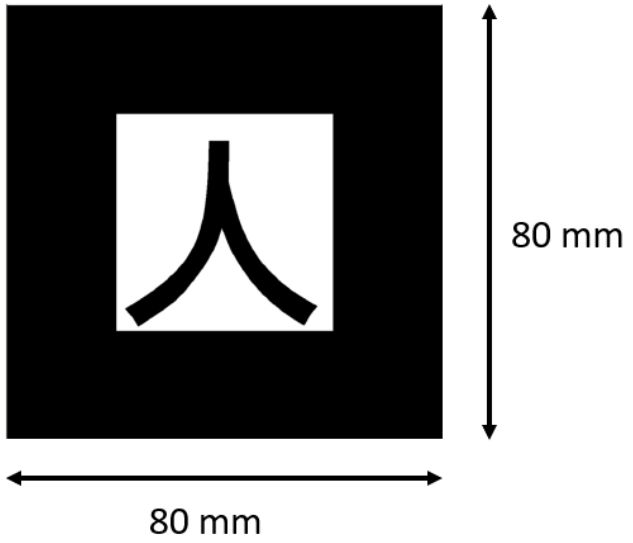
$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \sim \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} F \cdot d_u & s & u_0 \\ 0 & F \cdot d_v & v_0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{R} & \boldsymbol{T} \\ \mathbf{0}^{\mathrm{T}} & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$$

p= K ·A · E · P

Given **P**, **p** find **C:**

- **C** has 11 DoF so at least 11 eqns (5 for Intrinsic matrix, 3 for R, 3 for T)
- 5.5 correspondences (2 eqns per corr as *u* and *v*) needed for solving
- For Direct Solution derivation see **Appendix 2 -> get C (4x4 matrix)**

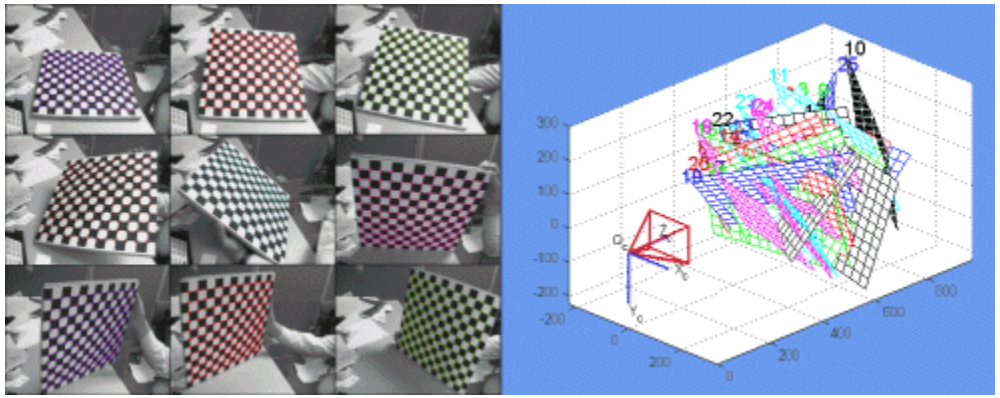
Camera Pose Estimation Using Markers



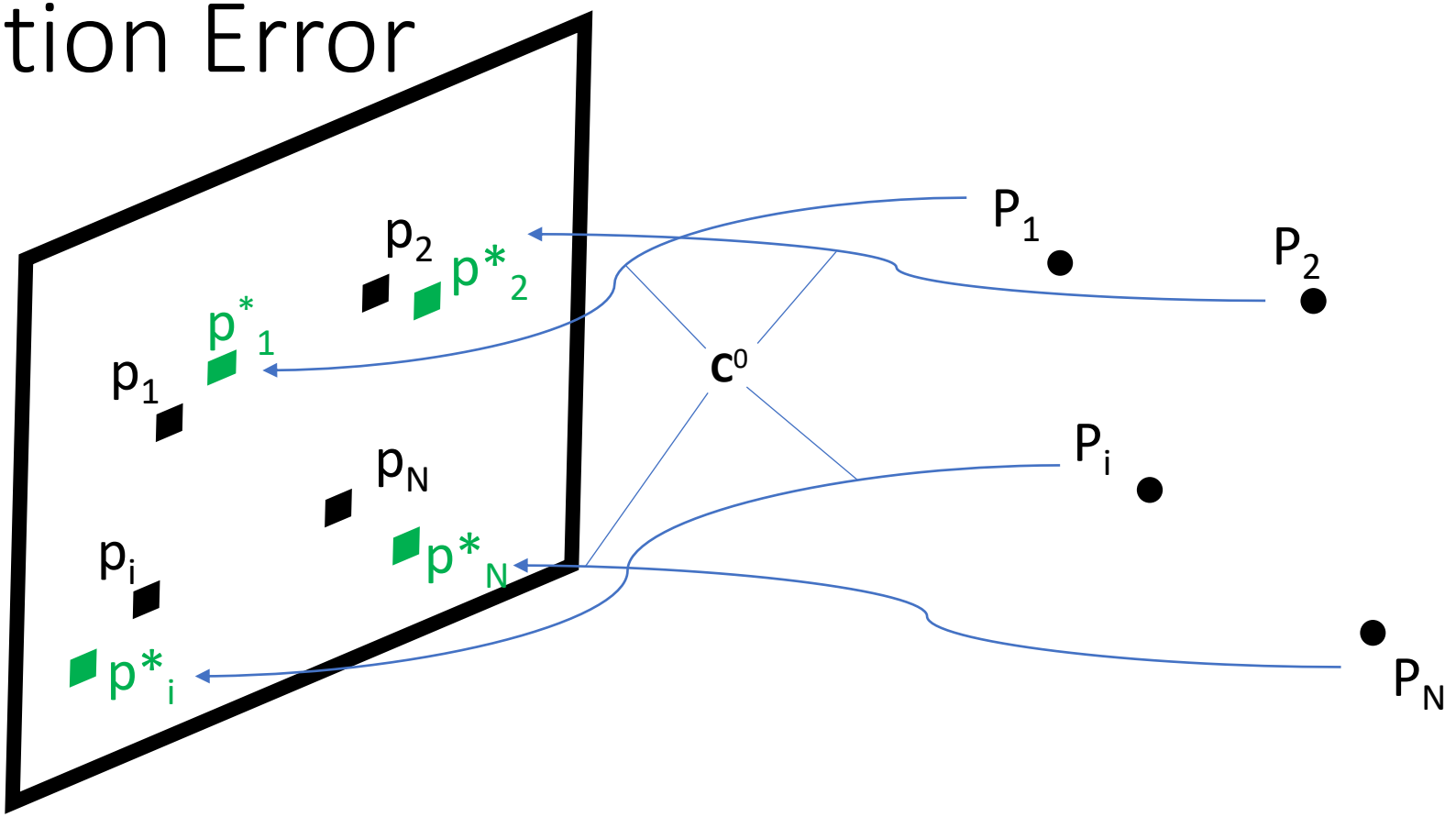
Essential problem in AR: CAMERA **POSE ESTIMATION**

- Usage of a pre-defined marker
- The computer knows how the marker looks like
- The computer knows the dimensions of the marker

(See Appendix 4 for OpenCV calibrarion toolbox)



Re-projection Error

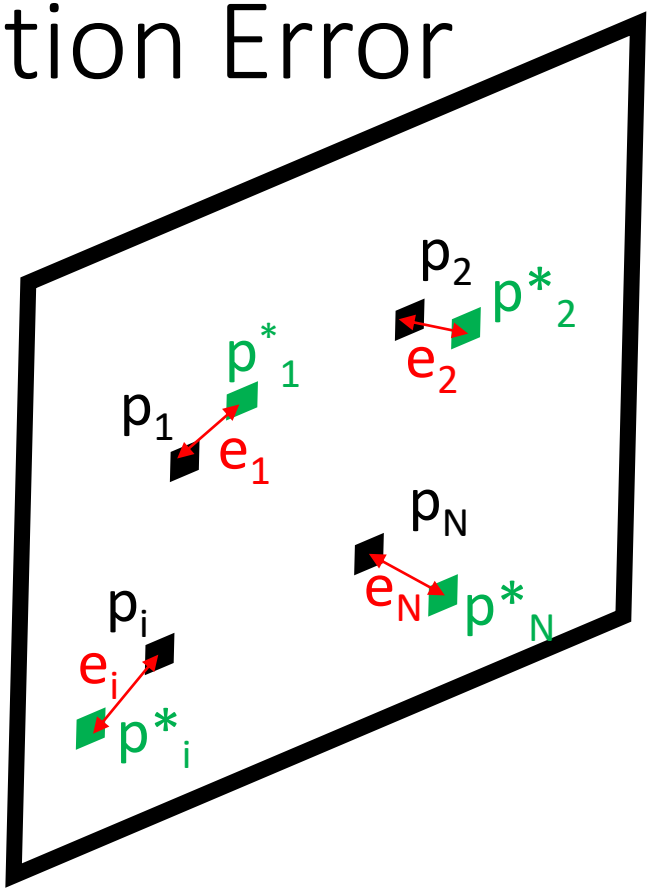


Assume we have an initial solution for the calibration matrix $\mathbf{C}^0$ computed as direct solution

We can then apply projection to map the known 3D points to the image plane

$\mathbf{p}^*_i = \mathbf{C}^0 \cdot \mathbf{P}_i$

Re-projection Error



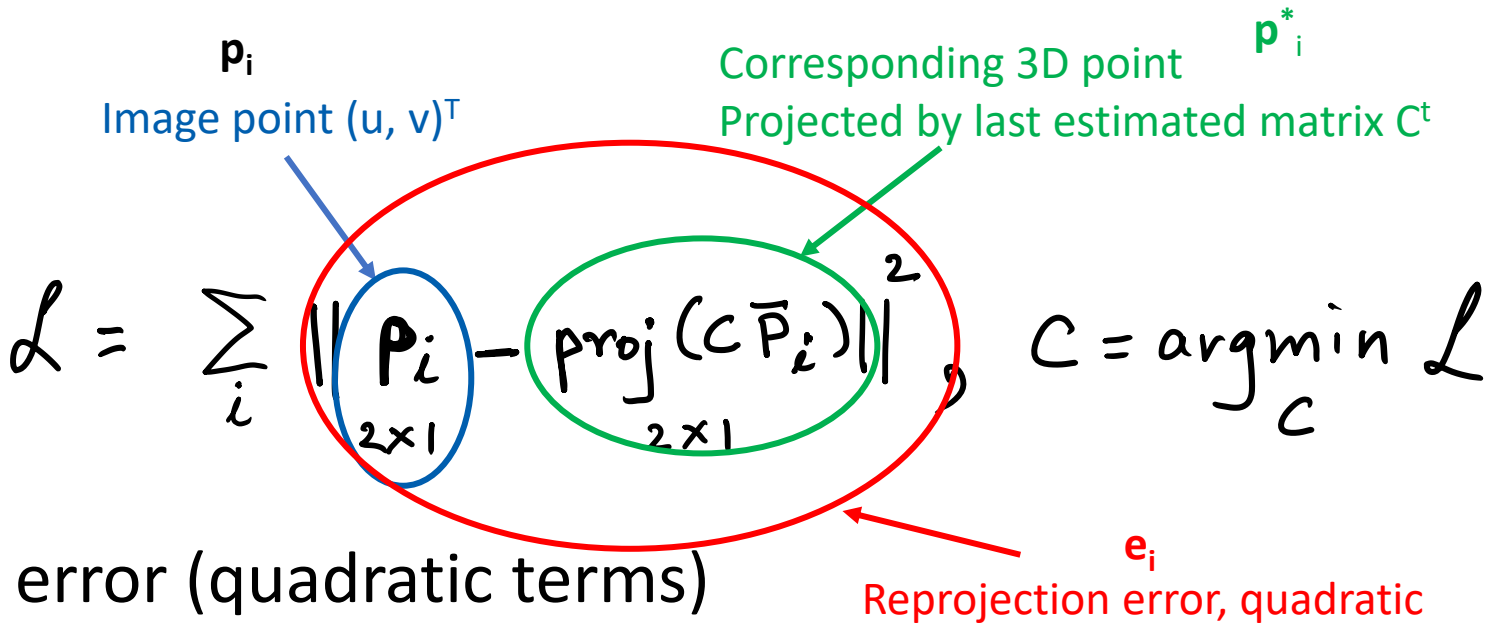
Due to inaccuracies in correspondence estimation, $\mathbf{C}^0$ is not perfect and we get a re-projection error

$\mathbf{e_i} = \mathbf{p_i} - \mathbf{p^*_i}$

We wish to minimize the re-projection error by an iterative procedure that updates from $\mathbf{C}^0$ by small steps $\mathbf{dC}^t$ towards a minimum $\mathbf{C}^{min}$

Non-linear Minimization of Re-projection Error

- Using N > 6 points makes estimation more stable and robust
- Define reprojection error



- Minimize this geometric error (quadratic terms)
  - using a non-linear iterative algorithm e.g. Levenberg-Marquardt based least squares

Problems, Use Case

- Direct Solution only gives C (4x4) calibration
- Other direct approaches e.g. Tsai (´87) able to do separation of extrinsic and intrinsic parameters

- Typical scenario in practice:
  - Pre-calibrate a camera to be used in a system/application
  - Calibration with SfM with single camera:
    - Physical properties remain unchanged (dimensions of sensor)
    - Most of the time focal length too
    - Intrinsics can be assumed constant
    - **Only extrinsics have to be estimated in real-time**
  - Calibration with SfM in multi-camera studio:
    - Pre-calibrate a given cameras, compute instrinsics & extrinsics

3 General Problems in 3D Computer Vision and AR

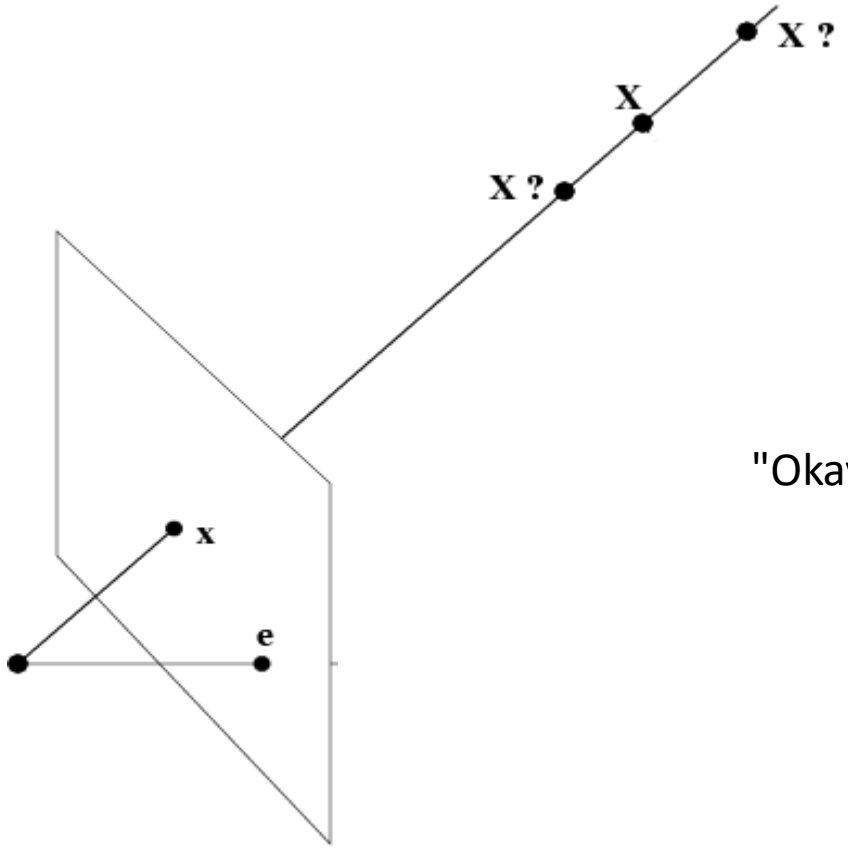
$$\mathbf{p} = \mathbf{C} \cdot \mathbf{P}$$

C and **P** known, solving for **p**: **Projection**

P and **p** known, solving for **C**: **Calibration**

C and **p** known, solving for **P**: **Reconstruction**

3D Information is lost by imaging

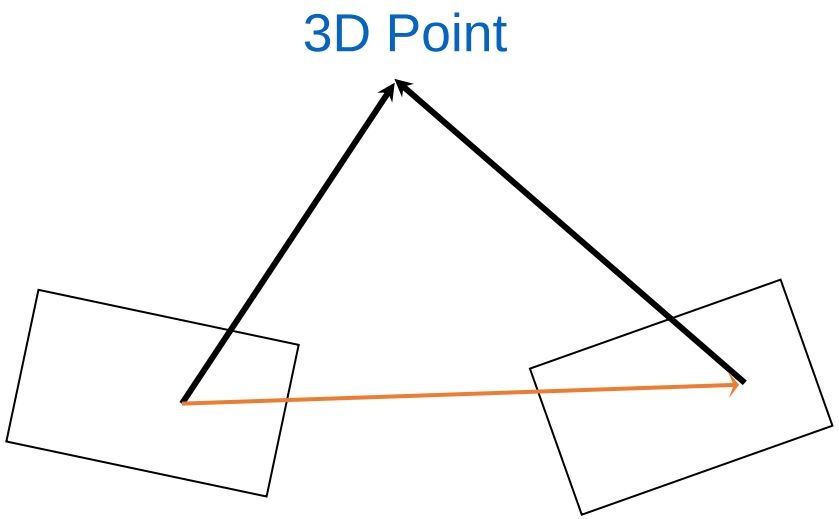


"Okay, one last time. These are small…
but those out there are far away. Small…far away. "

https://www.youtube.com/watch?v=vh5kZ4uIUC0

Triangulation

- Calculation of 3D point from point correspondences in the images (disparity)



- Computer Vision: Algorithms and Applications, Richard Szeliski, September 3, 2010 draft, 2010 Springer
- Section 7.1

3 General Problems in 3D Computer Vision and AR

$$\mathbf{p} = \mathbf{C} \cdot \mathbf{P}$$

C and **P** known, solving for **p**: **Projection**

P and **p** known, solving for **C**: **Calibration**

C and **p** known, solving for **P**: **Reconstruction**

4th General Problem in 3D Computer Vision and AR

$$\mathbf{p} = \mathbf{C} \cdot \mathbf{P}$$

- If only **p** are known between two or more images of the same 3D scene, can we recover **C** and **P**?
- From a single camera?

- **Yes, but only if camera is moving!**

- **Structure from Motion**

- **Simultaneous Locatization and Mapping (SLAM)**

Structure from Motion (SfM)

Find and match feature points

KU Leuven, Marc Pollefeys, Luc Van Gool

Find camera positions and sparse point cloud

KU Leuven, Marc Pollefeys, Luc Van Gool

Dense point cloud and 3D mesh

KU Leuven, Marc Pollefeys, Luc Van Gool

Feature extraction - Correspondence Problem

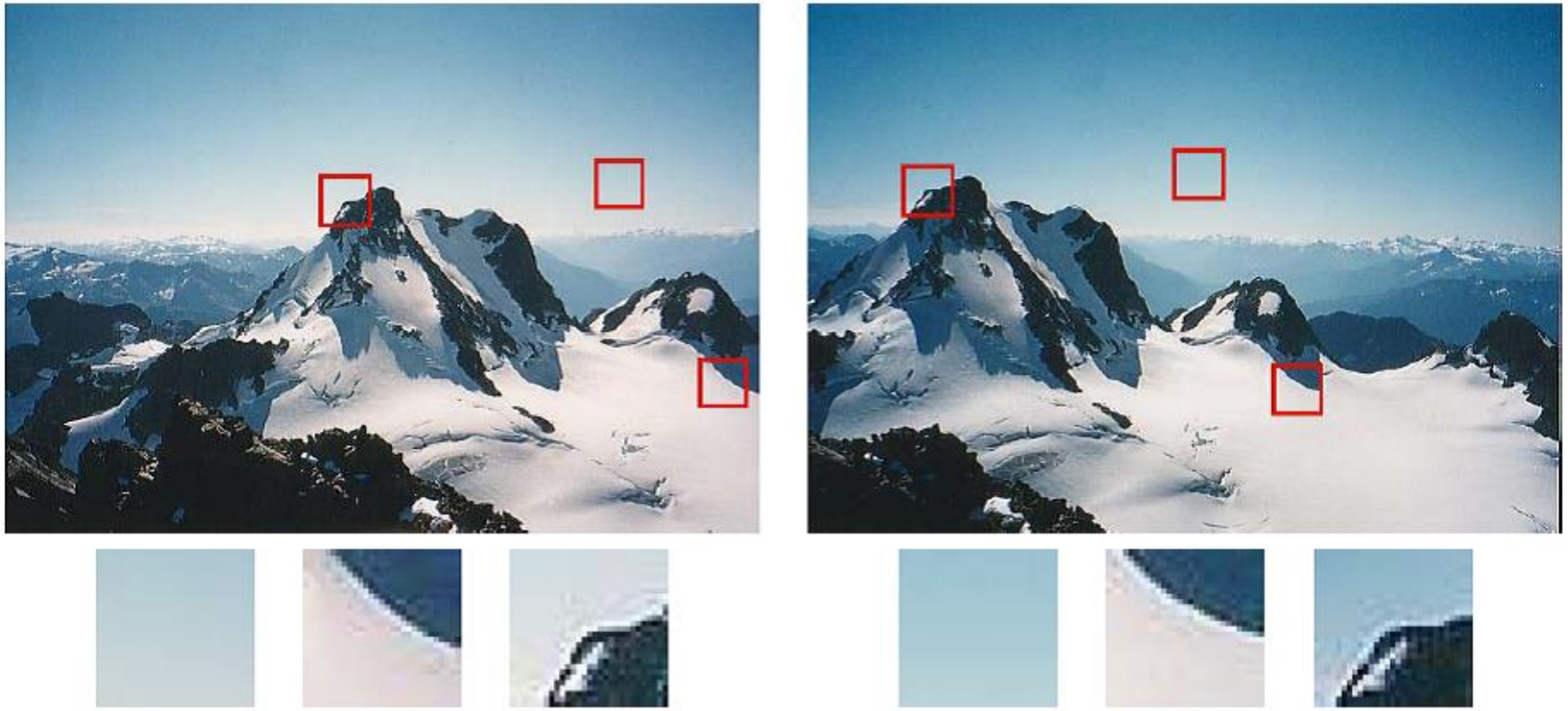


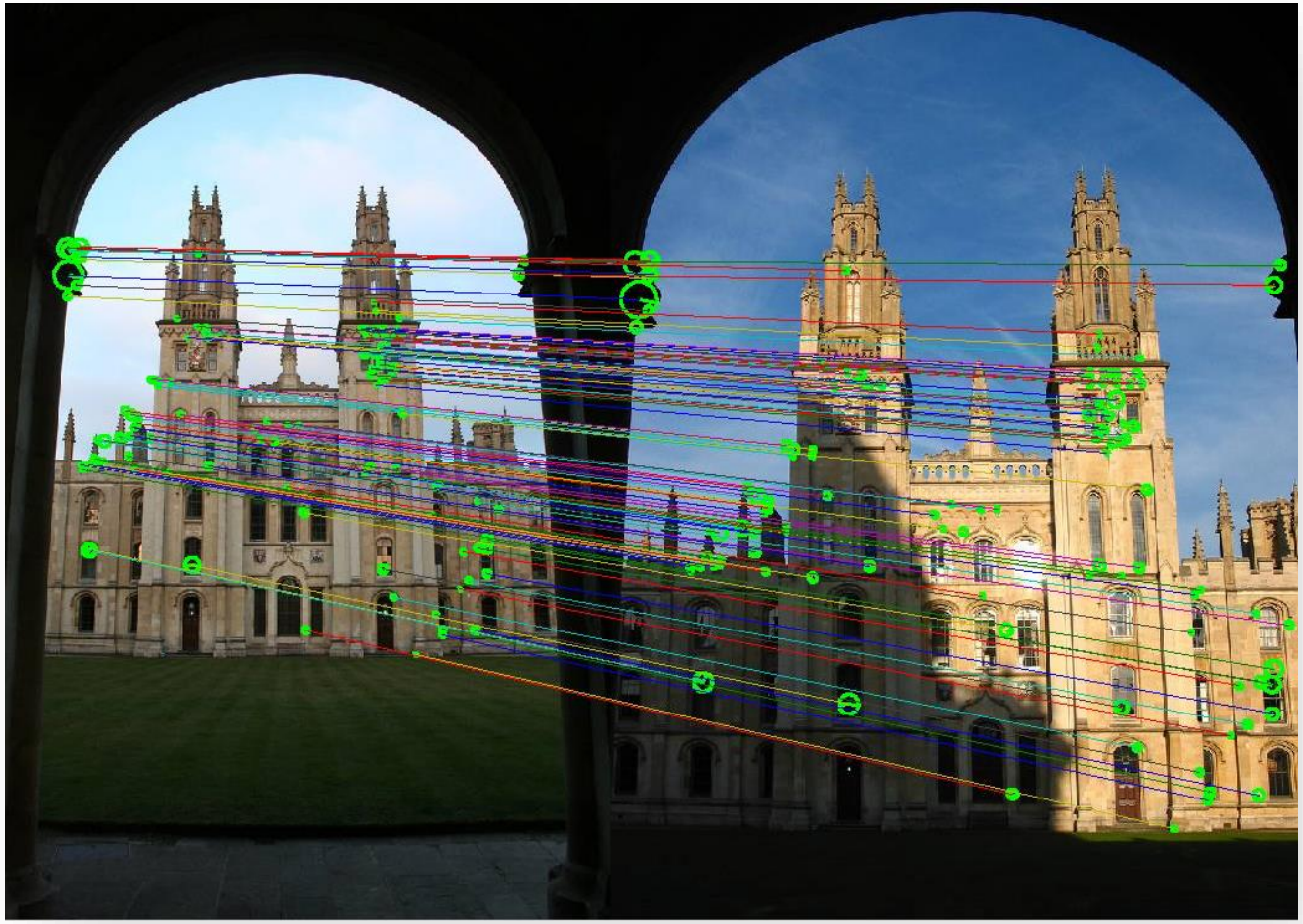
Figure 4.3 Image pairs with extracted patches below. Notice how some patches can be localized or matched with higher accuracy than others.

- http://szeliski.org/
- Computer Vision: Algorithms and Applications, Richard Szeliski, September 3, 2010 draft, 2010 Springer

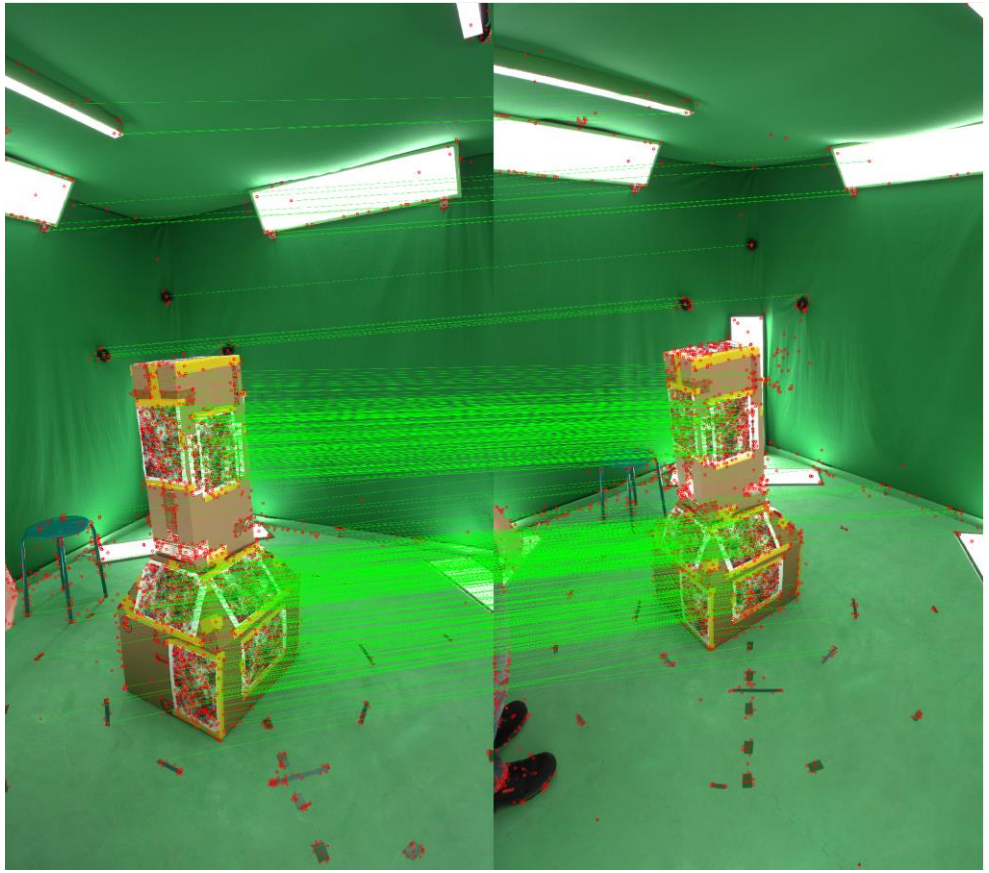
Structure from motion

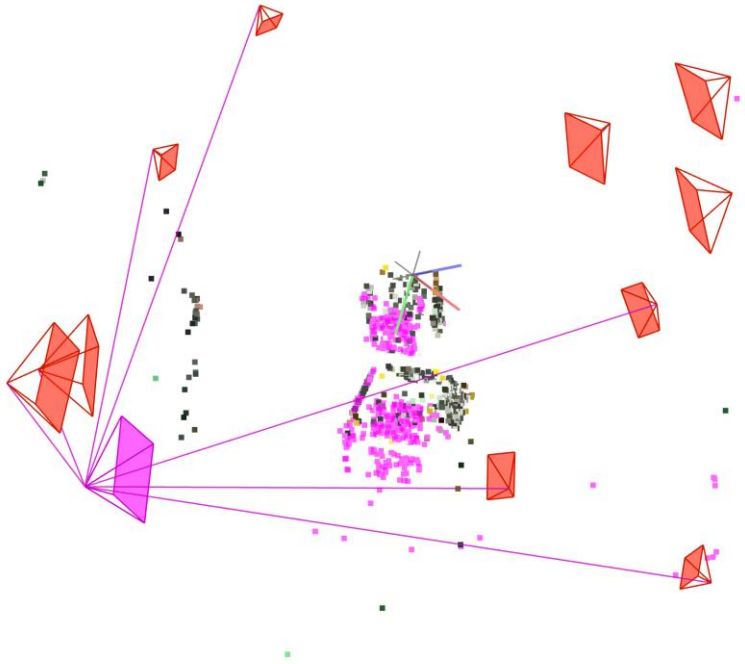
Feature detection & matching

- Features points are detected and matched across images.
- Matches are filtered using geometric and statistics constraints (e.g. RANSAC).
- SIFT features are probably the most popular, but there are alternatives (ORB, AKAZE, SURF…).



Calibration Example - Colmap

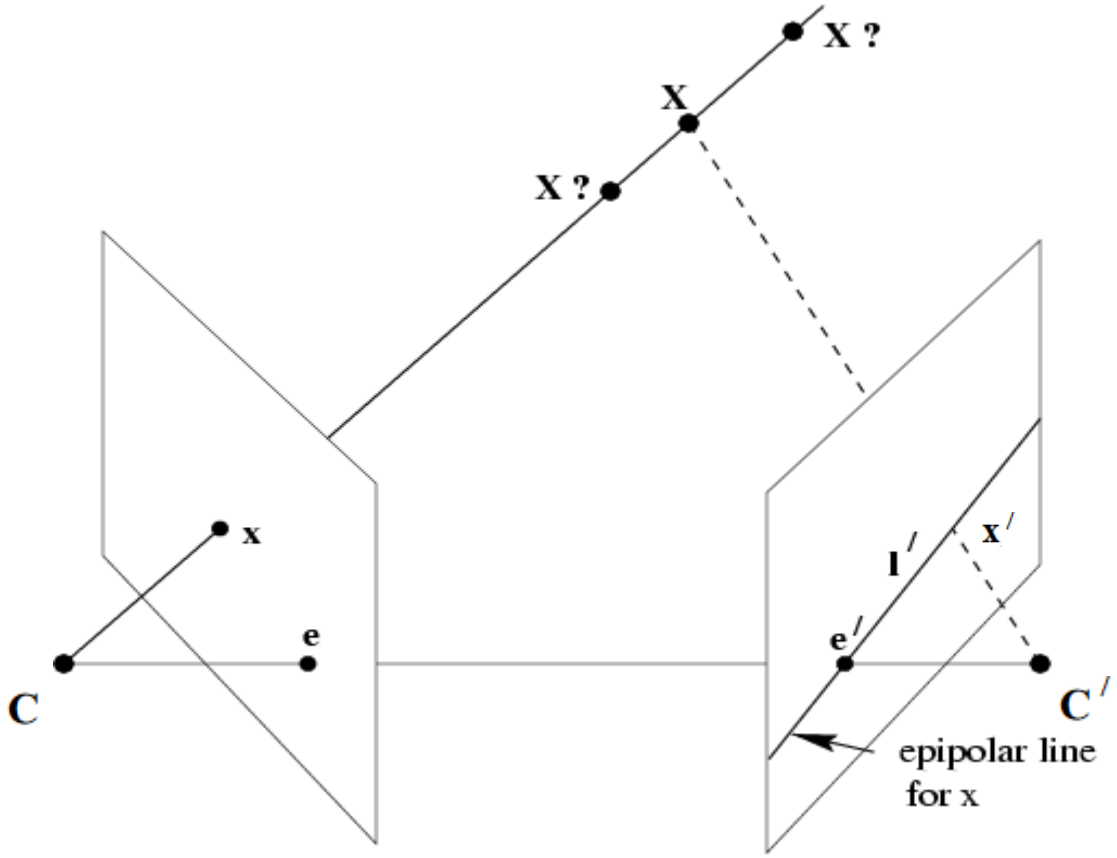


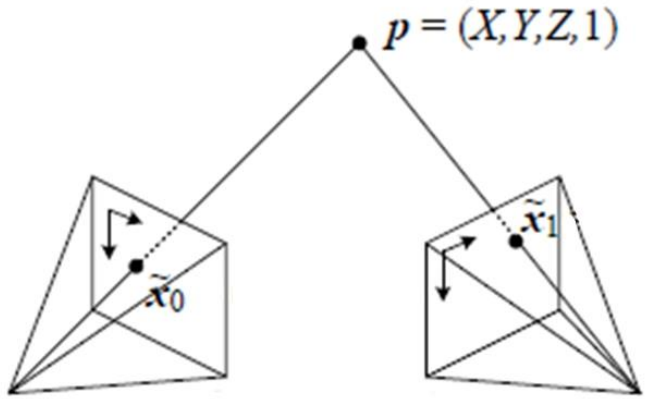


https://colmap.github.io/index.html

Epipolar Geometry
Geometry of 2 cameras

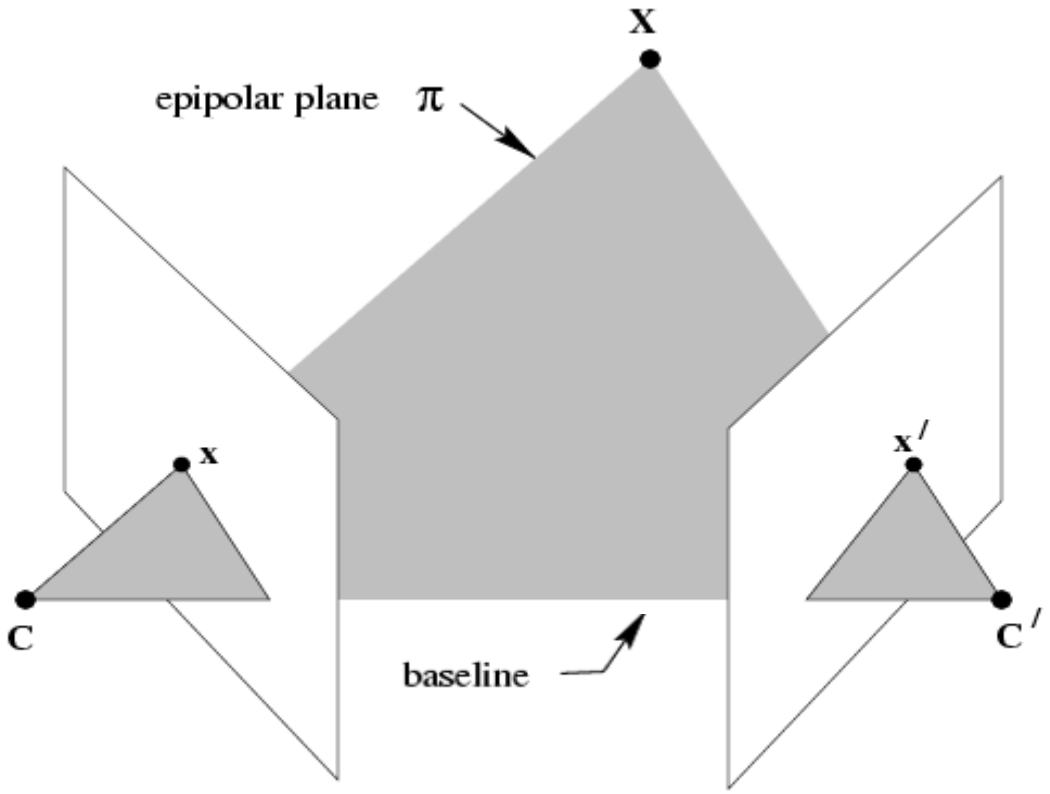
A second view is needed to recover 3D info





Epipolar line - an image point x back-projects to a ray in 3D-space defined by the first camera centre, C,
and x. This ray is imaged as a line l' in the second view

Epipolar Geometry



Points C, C', x, X, x' are co-planar

- **Epipolar plane**

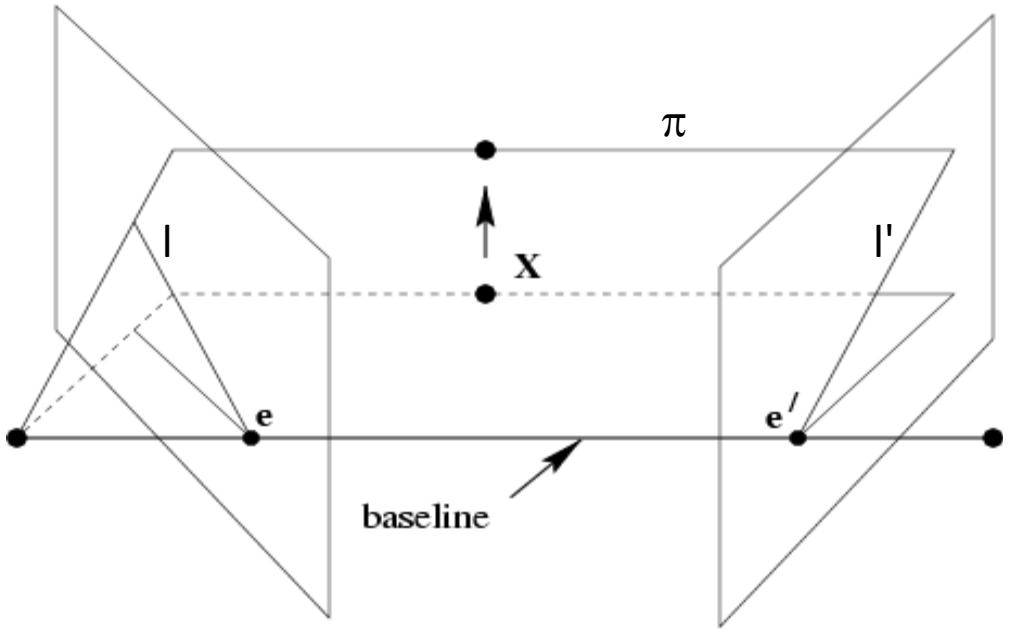
Every 3D point has it's own epipolar plane

But they all pass through one line – that connects 2 camera centers

- **baseline**

Epipolar Geometry

As the position of the 3D point X varies, the epipolar planes "rotate" about the baseline. All epipolar lines intersect at the epipoles

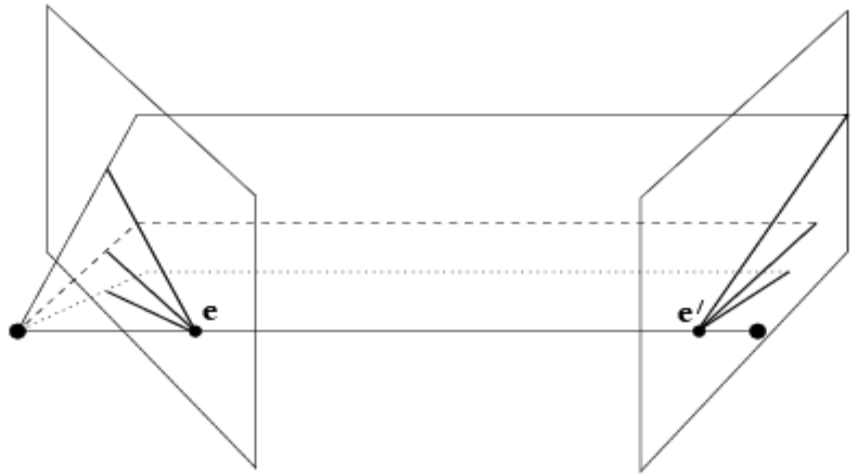


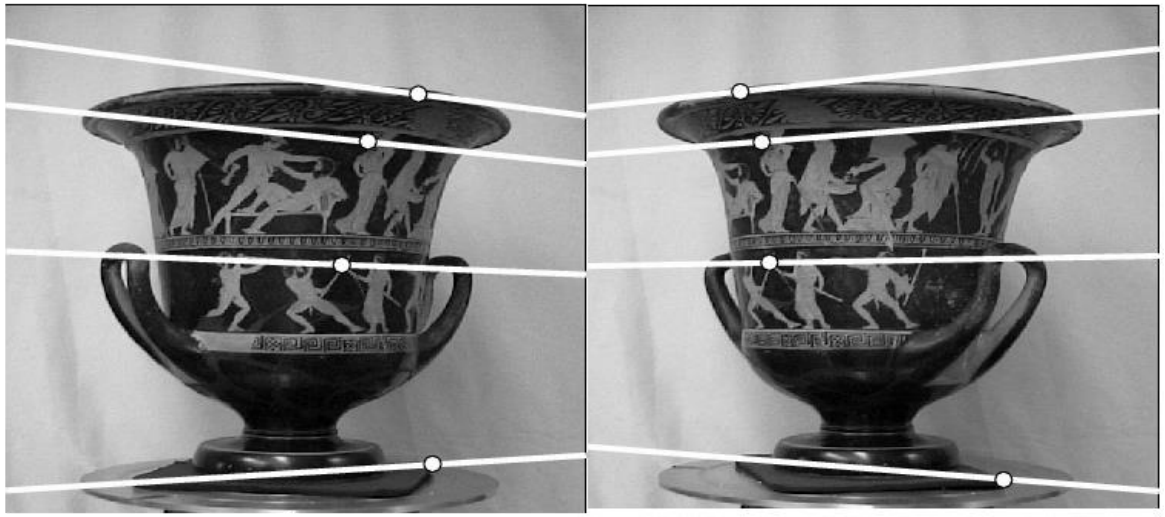
epipoles e,e' - intersection of baseline with image planes
(OR projection of camera center in the other image)

an epipolar plane π = plane containing baseline

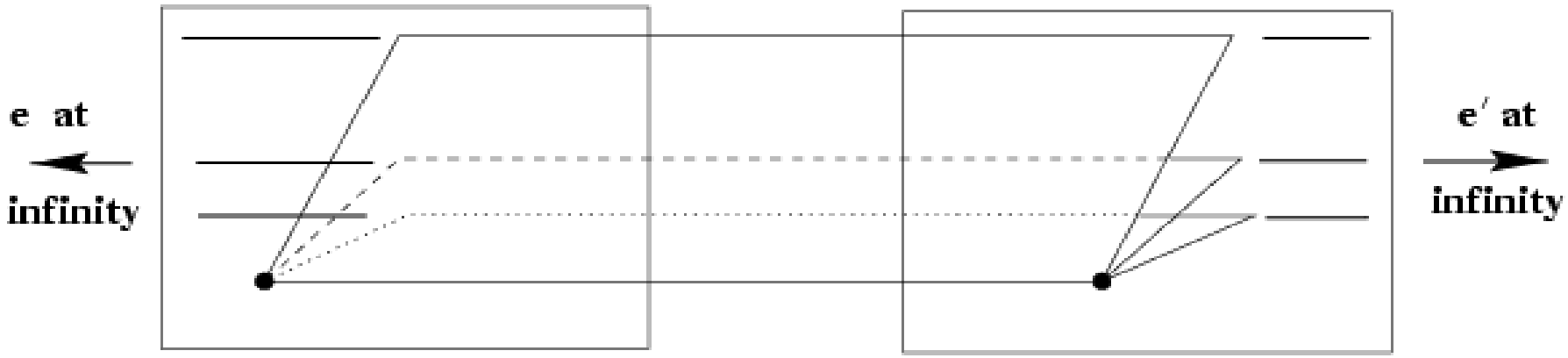
an epipolar line l, l' = intersection of epipolar plane with image
(always come in corresponding pairs)

Converging Cameras





Parallel Image Planes – special case



Calculate disparity between images $d = x_l - x_r$

Z – depth, T – distance between cameras, f – focal length, d – disparity

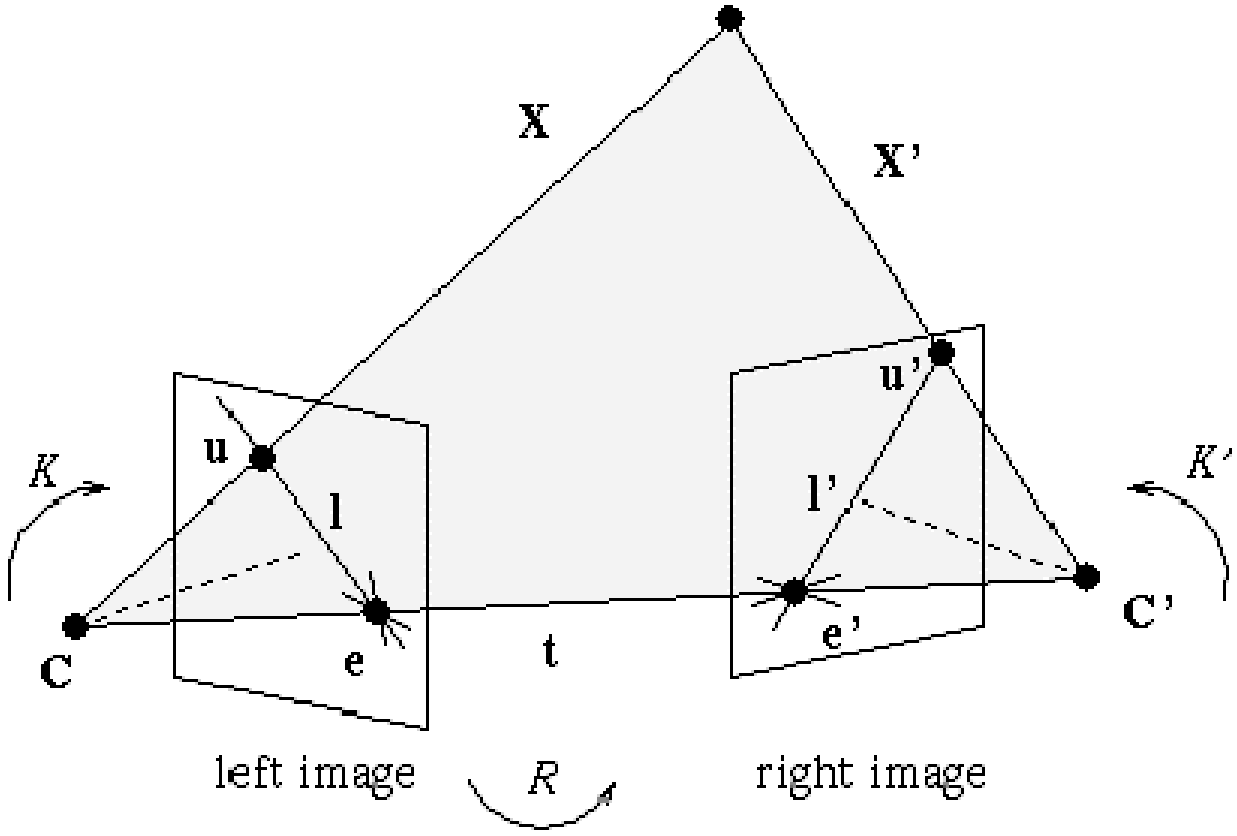
$$Z = f\frac{T}{d}$$

Use **Rectification** for solution – See **Appendix 5**

Fundamental & Essential Matrix

Fundamental Matrix

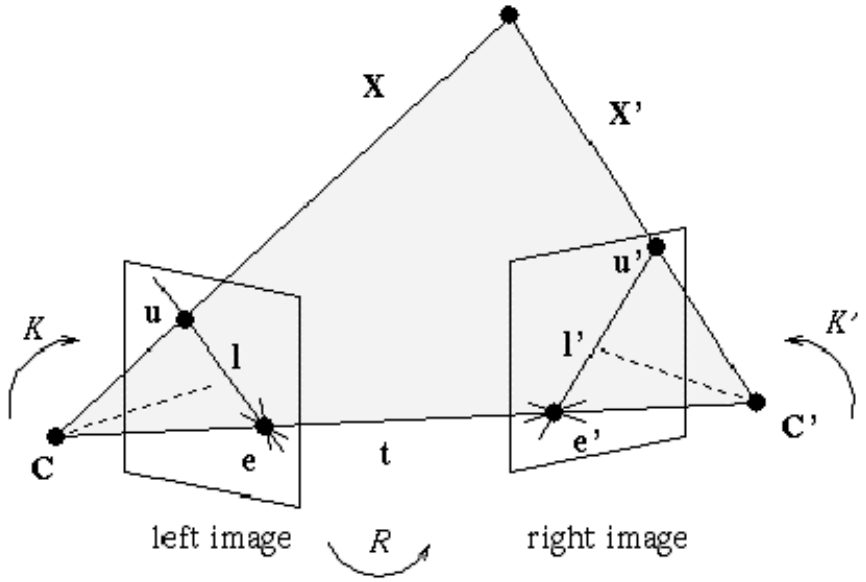
- If optical axes are not parallel
- K, K’ – intrinsic matrices (unknown)
- u, u’ – left and right projections (known)
- Left hand camera as origin:
  $R_0 = I$ (Identity), $t_0 = (0,0,0)$
- Right hand camera: R, t (unknown)
- Left Calibration as K[I | 0]
- Right Calibration as K’[R | t]
- Consider X, X’ and t as vectors



Fundamental Matrix

- X, X' and t are coplanar, hence normal to t and X' is also prependicular to X:
  - Normal to epipolar plane - $\mathbf{t} \times \mathbf{X}'$
  - Dot product of normal with X is 0:

$$\mathbf{X}^T \cdot (\mathbf{t} \times \mathbf{X}') = 0$$



- Represent X and X' as $K^{-1}\mathbf{u}$ and $R^{-1}\,K'^{-1}\mathbf{u}'$ where K^{-1} and K'^{-1} are inverse of K and K'

$$(K^{-1}\mathbf{u})^T \cdot (\mathbf{t} \times R^{-1}\,K'^{-1}\mathbf{u}') = 0$$

- Introduce S(t) – *skew symmetric matrix of* t

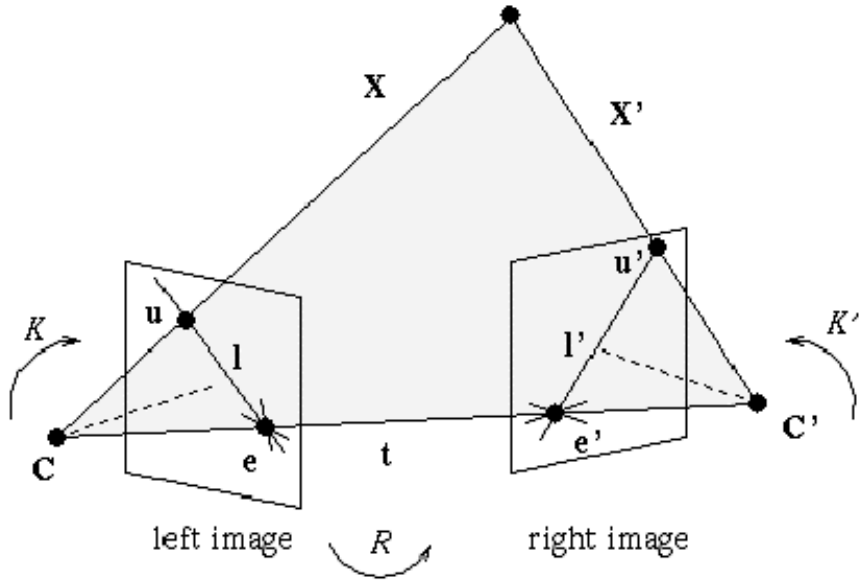
$$S(\mathbf{t}) = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

$$(K^{-1}\mathbf{u})^T\,(S(\mathbf{t})\,R^{-1}\,K'^{-1}\mathbf{u}') = 0$$

$$(K^{-1}\mathbf{u})^T\,(S(\mathbf{t})\,R^{-1}\,K'^{-1}\mathbf{u}') = 0$$

$$\mathbf{u}^T\boxed{(K^{-1})^T S(\mathbf{t})R^{-1}K'^{-1}}\mathbf{u}' = 0$$

Fundamental Matrix - $F = (K^{-1})^T S(\mathbf{t})R^{-1}K'^{-1}$



Fundamental matrix satisfies the condition that for any pair of corresponding points x ↔ x′ in the two images:

$$\mathbf{u}^T\,F\,\mathbf{u}' = 0$$

Fundamental Matrix

- F captures all the information between a pair of cameras if correspondence is solved
- Note: skew symmetric matrix can be created if t≠0, so camera origins cannot coincide

- Properties
  - Incorporates K, K′, R and t
  - F has rank 2 (3x3 matrix)
  - 7 degrees of freedom

$$F \equiv (K^{-1})^T S(t) R^{-1} K'^{-1}$$

$$\mathbf{u}^T F \mathbf{u}' = 0$$

- Matrix Estimation:
  - Need minimum 7 correspondences in the images
  - Solve equation system for components of F
  - Iterate for non-linear minimization of reprojection error

Essential Matrix

- When intrinsics are known
- Can normalise image measurements

$$\breve{\mathbf{u}} = K^{-1}\mathbf{u}, \quad \breve{\mathbf{u}}' = K'^{-1}\mathbf{u}'$$

$$(K^{-1}\mathbf{u})^T \left(S(\mathbf{t})\, R^{-1}\, K'^{-1}\mathbf{u}'\right) = 0$$

- Captures information about the relative motion of a calibrated camera.

$$\breve{\mathbf{u}}^T\, S(\mathbf{t}) R^{-1}\, \breve{\mathbf{u}}' = 0$$

- Contains information about Rotation and Translation
- Allows to recover t and R once computed

$$\breve{\mathbf{u}}^T\, E\, \breve{\mathbf{u}}' = 0$$

$$\mathrm{E} = \mathrm{K}'^{\mathsf{T}}\mathrm{F}\mathrm{K}$$

Structure from Motion

SfM and 3D Reconstruction

Photogrammetry technique for estimating 3D structures from 2D image sequences

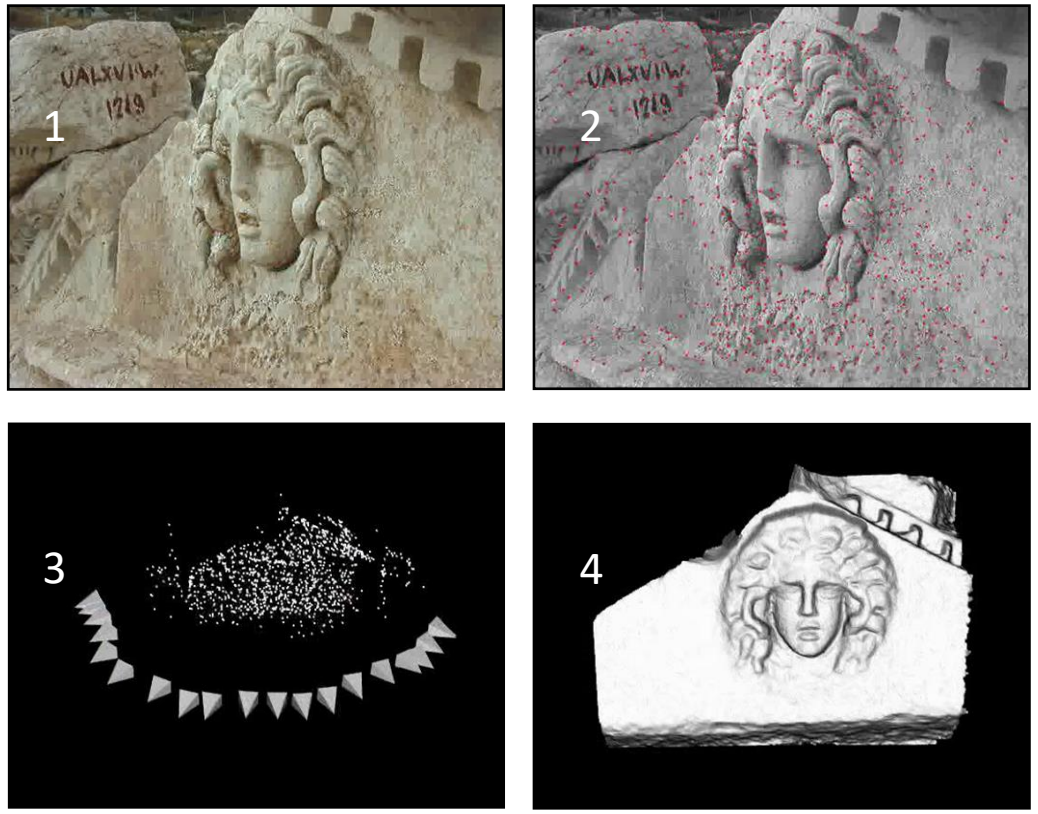
1. Monoscopic video capturing

   or Multiple cameras

2. Feature point extraction
   - Corner point detection
   - Robust feature tracking

3. Self-Calibration and 3D Points recovery
   - Extrinsic/intrinsic camera parameter
   - 3D positions of feature points

4. Dense disparity estimation
   - ``Per-pixel'' depth values



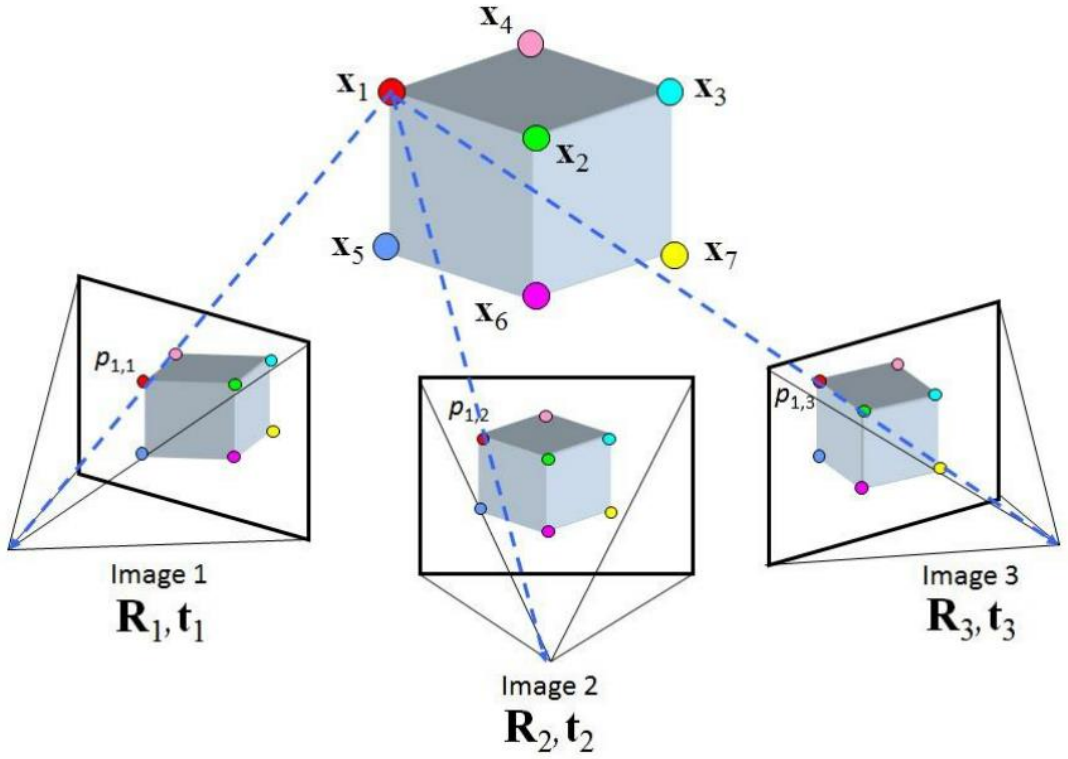
KU Leuven, Marc Pollefeys, Luc Van Gool

https://www.youtube.com/watch?v=i7ierVkXYa8

Structure from Motion

- Given N images of M **fixed** 3D points
- Estimate N calibration matrices C_i and M 3D points X_j from N x M correspondences

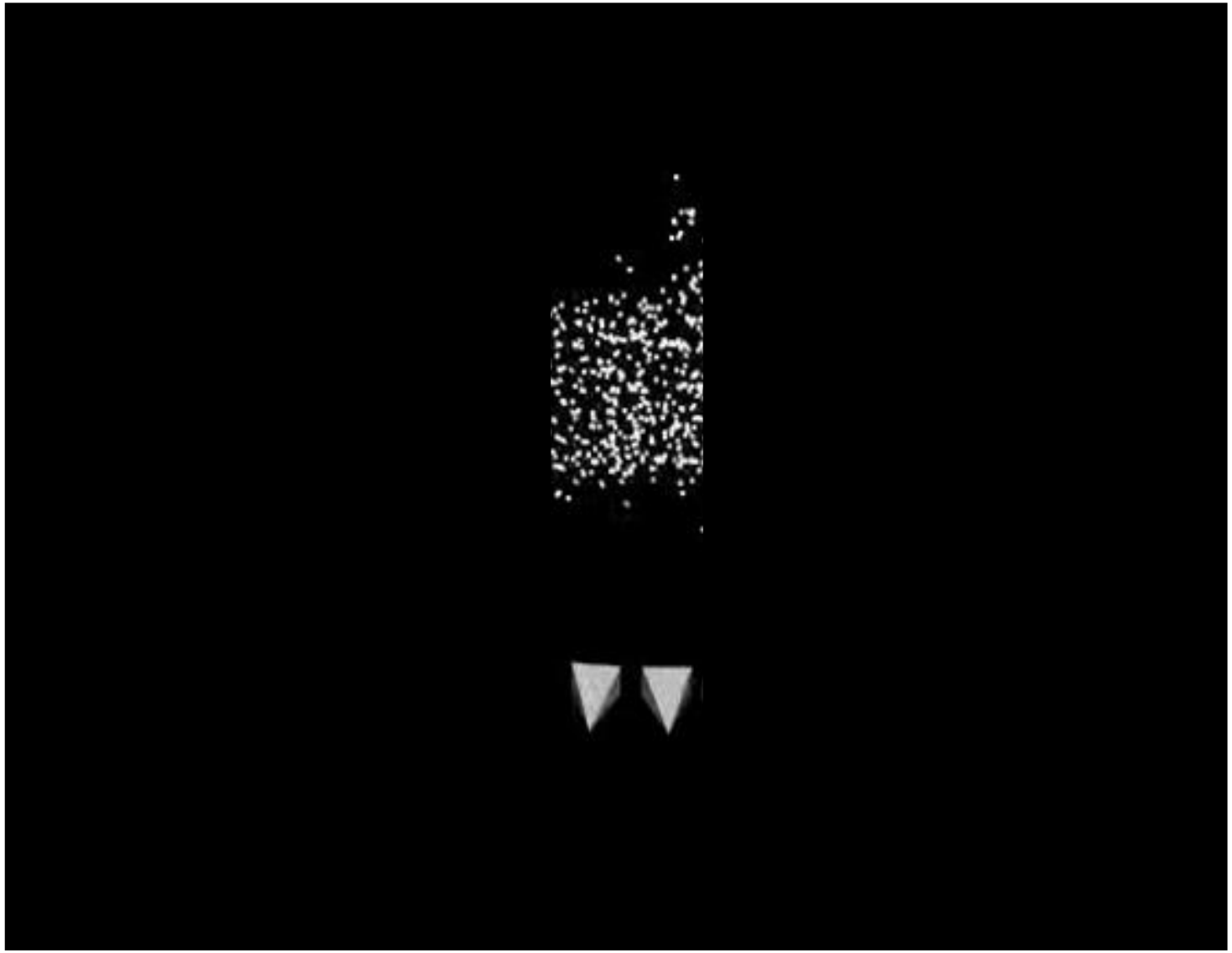
We know how to solve SfM for a pair of cameras (via F and E). How to do it for a sequence of images?



https://www.researchgate.net/figure/269327935_fig2_Figure-2-Structure-from-Motion-SfM-process-is-illustrated-The-structure-in-the

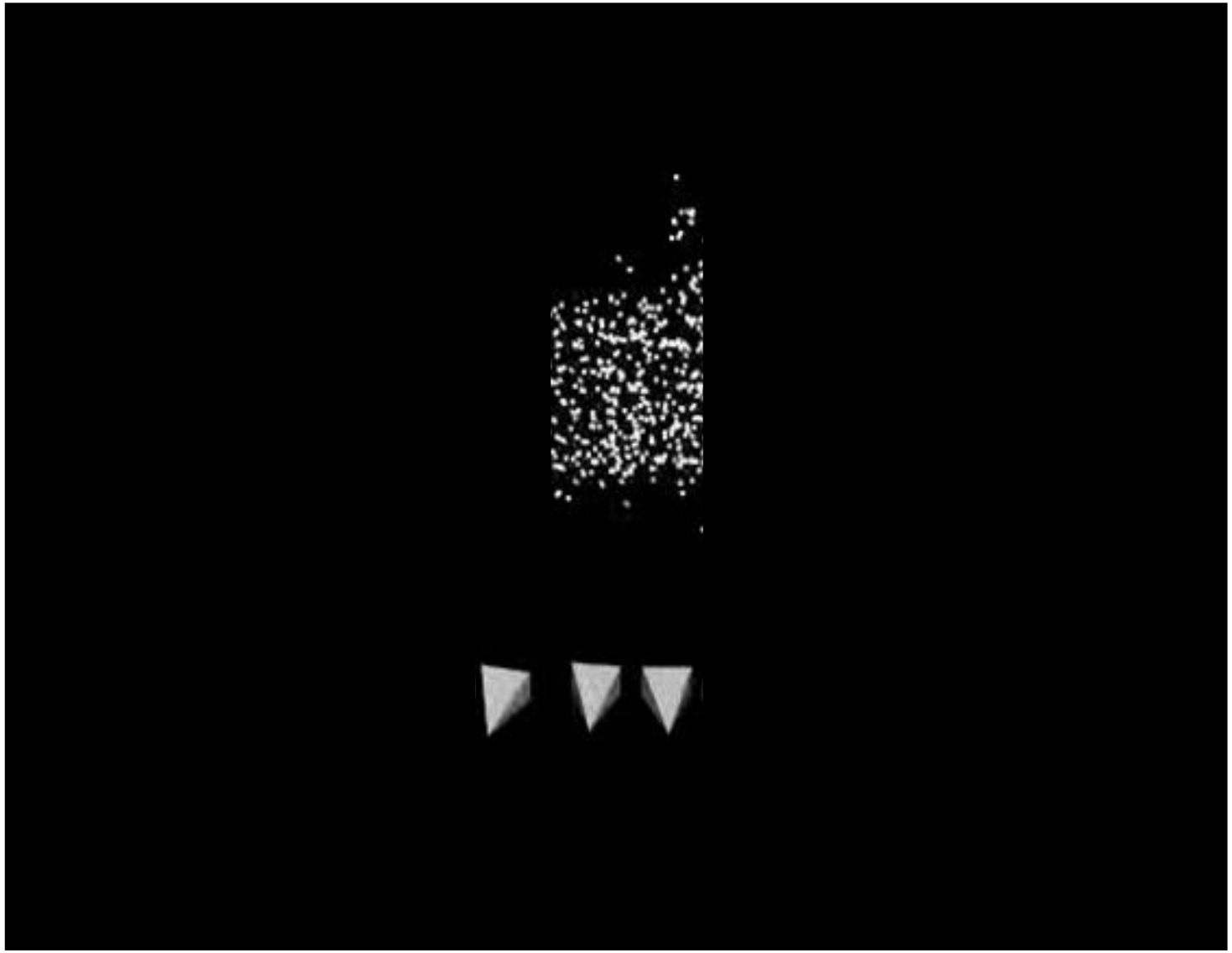
Incremental SfM

- Start with 2 views
- Take the 1st as reference
- $R_1 = I \quad t_1 = 0$
- Perform 2 view SfM as before
- Get R_2 and t_2 for 2nd view
- Also solve for structure
- Get a 1st portion of a point cloud $P_{1\text{-}2}$, as visible in views 1 and 2



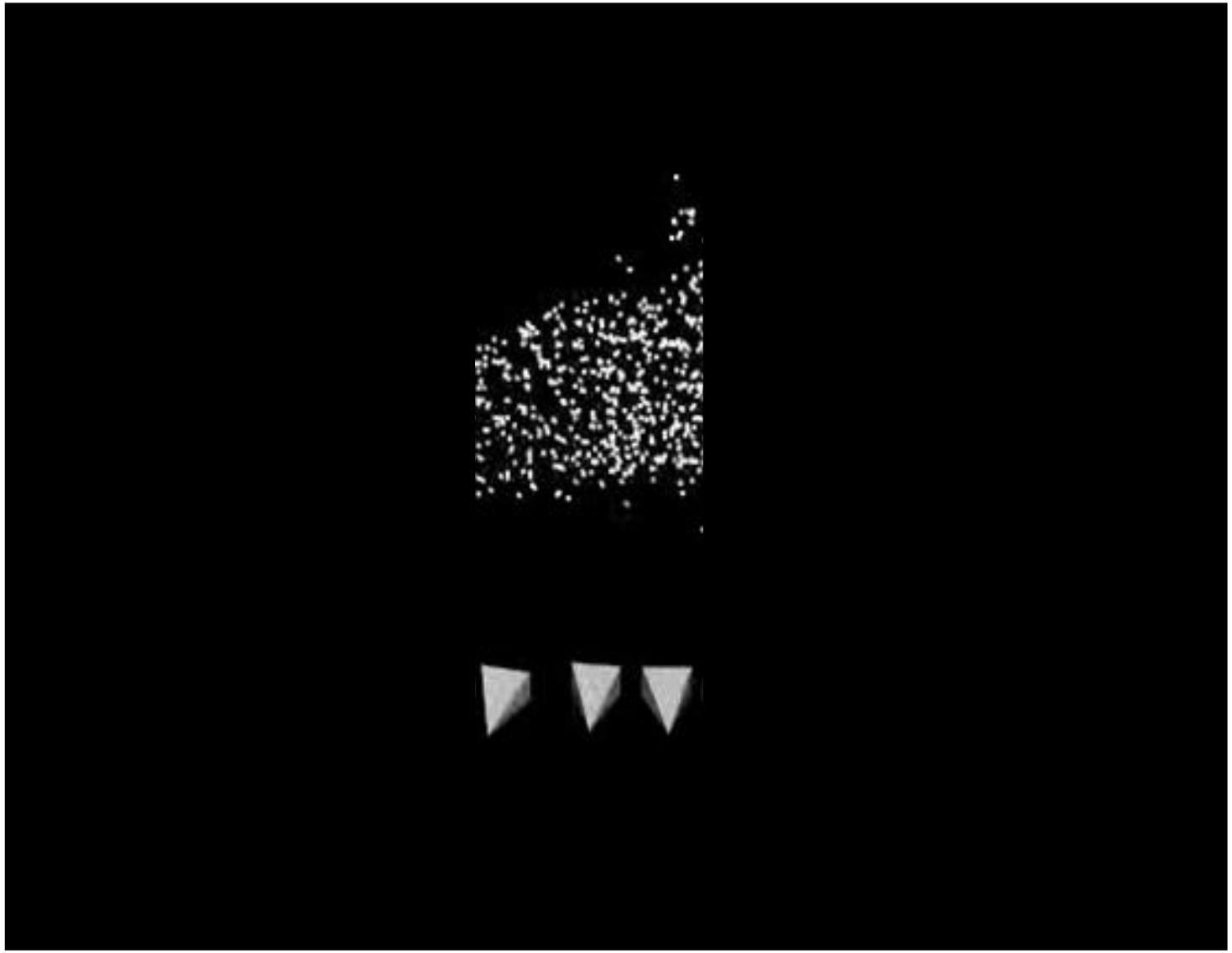
Incremental SfM

- Add the 3rd view
- Now we have a set of known 3D points from 1st iteration
- These can be regarded as “marker“ or “calibration object“
- Can compute a first estimate of R_3 and t_3 by “calibration“



Incremental SfM

- Can add new points visible in views “2 AND 3“ but not in views “1 AND 2“ ($P_{2\text{-}3} \setminus P_{1\text{-}2}$) to the overall point cloud by “triangulation“
- Also possible to refine previously calculated points $P_{2\text{-}3} \cap P_{1\text{-}2}$
- Now we have for N=3 cameras
  - N rotations (one being identity)
  - N translations (one being zero)
  - A point cloud with M 3D points



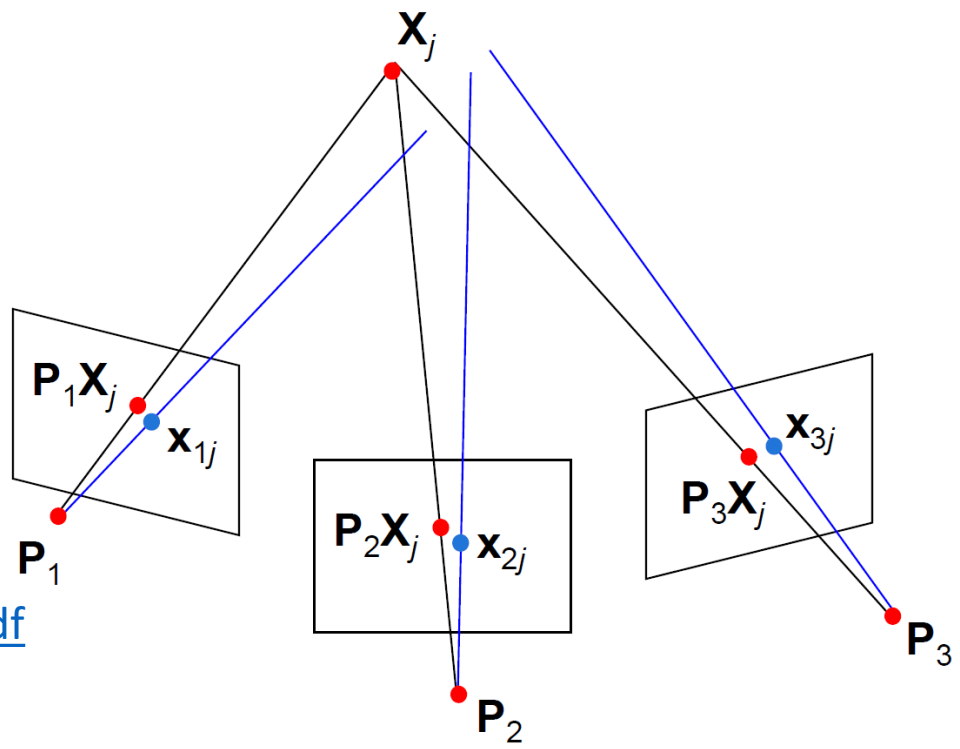
Bundle Adjustment

- Refine ALL estimates, i.e.
  - N rotations (one being identity)
  - N translations (one being zero)
  - A point cloud with M 3D points

- In a single huge non-linear minimization procedure
- Minimizing reprojection error of points
- **BUNDLE** of rays to be **ADJUSTED**
- Can be extremely complex, depending on number of images and points
- Can compute for hours or days

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

$$E(\mathbf{P},\mathbf{X}) = \sum_{i=1}^{m}\sum_{j=1}^{n} D\left(\mathbf{x}_{ij}, \mathbf{P}_i\mathbf{X}_j\right)^2$$

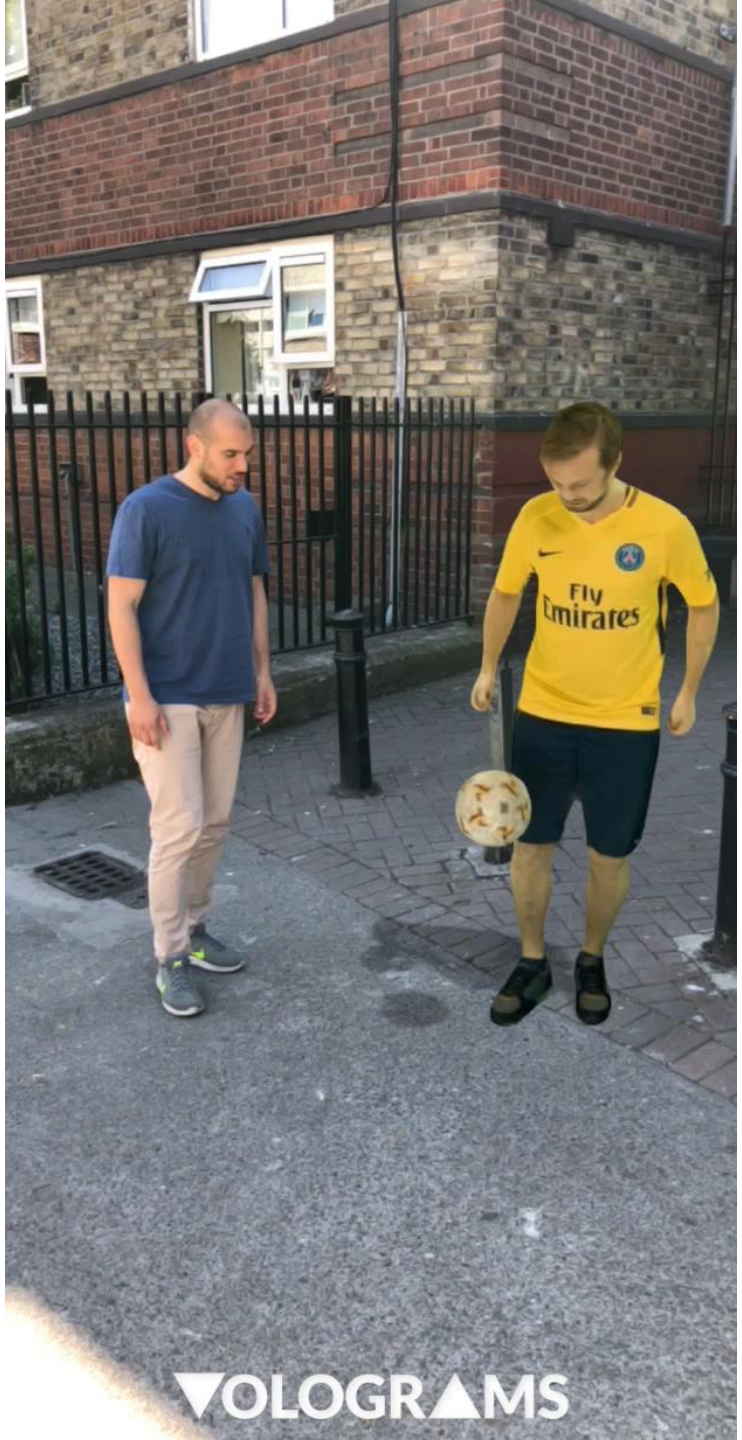


http://cs.nyu.edu/~fergus/teaching/vision_2012/6_Multiview_SfM.pdf
Lecture 6: Multi-view Stereo & Structure from Motion
Prof. Rob Fergus

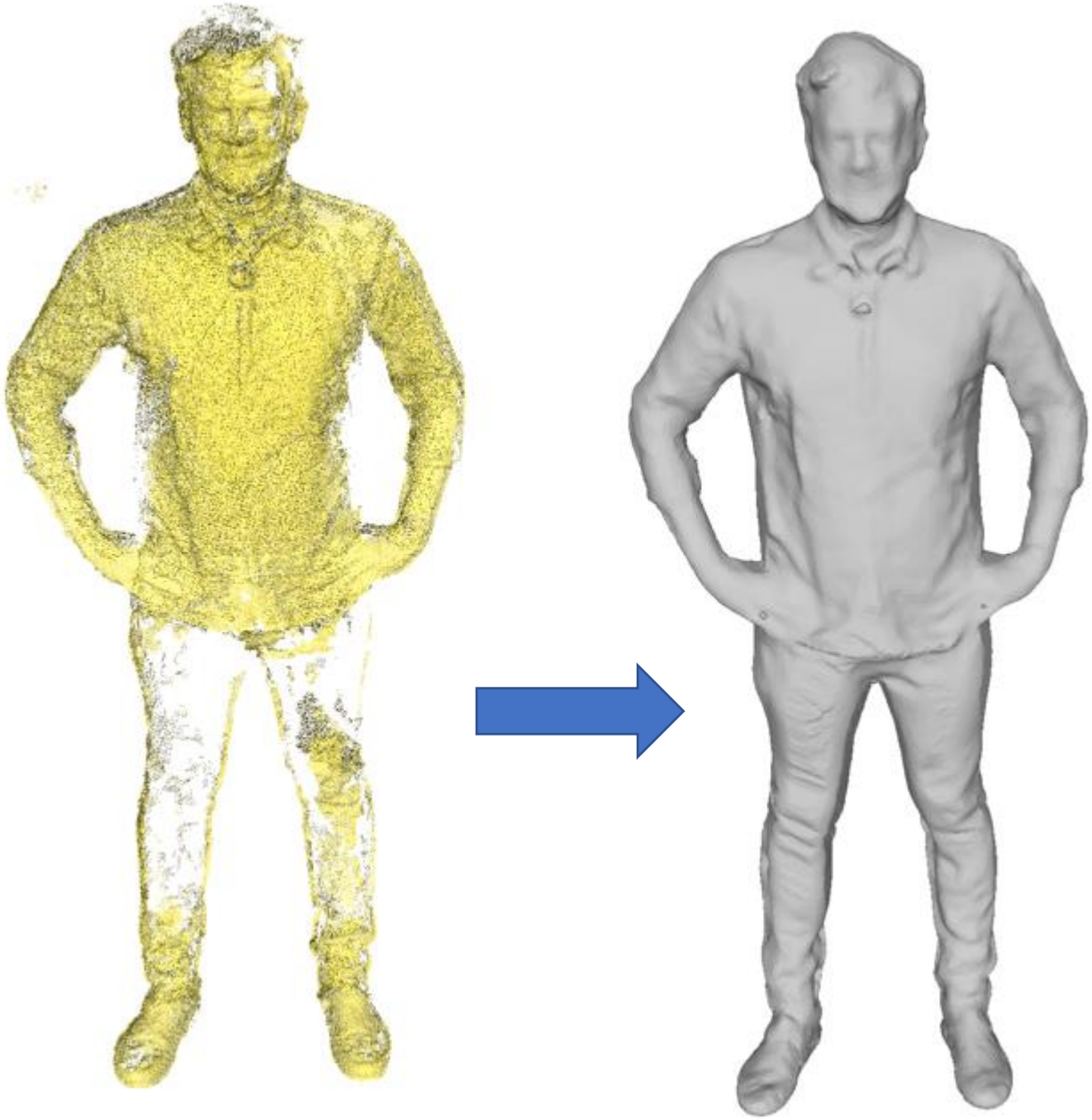
SfM - Examples

Building Rome in a Day
Sameer Agarwal, Noah Snavely, Ian Simon, Steven M. Seitz and Richard Szeliski
International Conference on Computer Vision, 2009,
Kyoto, Japan.

https://grail.cs.washington.edu/rome/


Fly
Emirates
VOLOGRAMS

More processing needed to get from SfM dense point cloud to 3D mesh – check out paper in the next slide

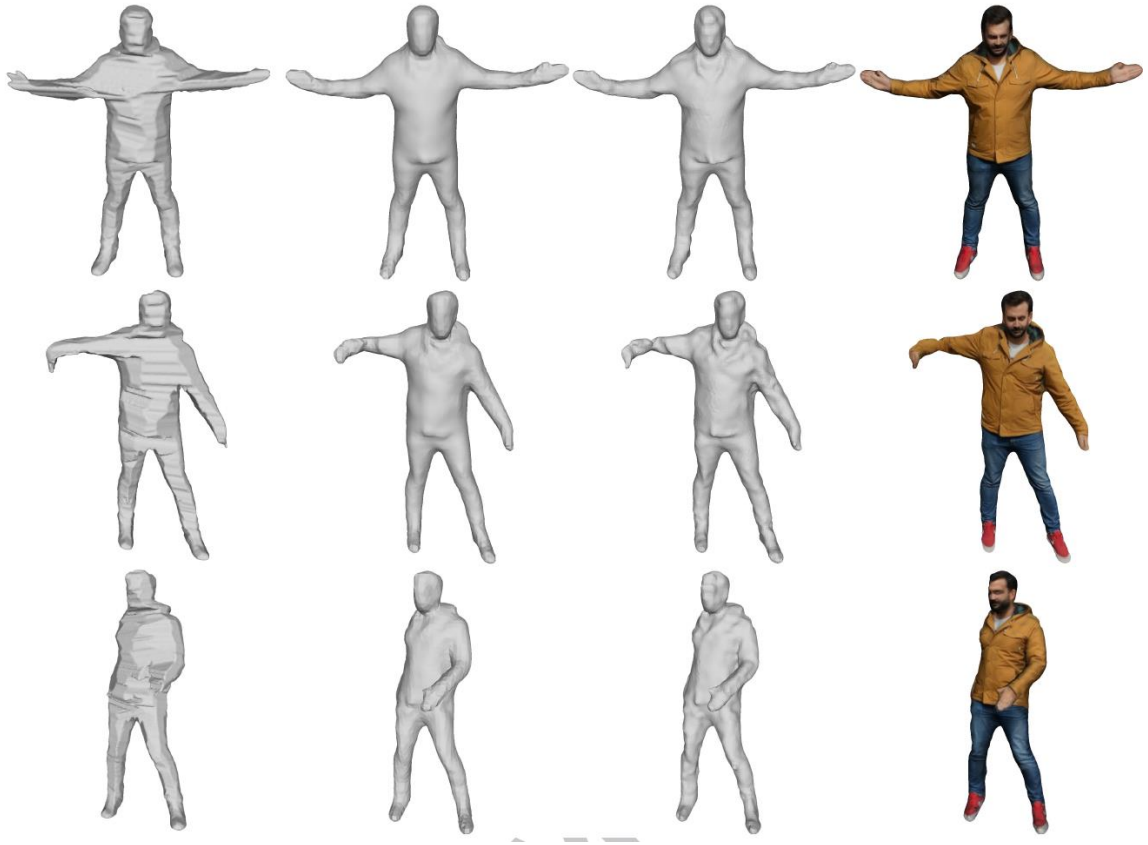


VV for VR/AR Content Creation

http://authors.elsevier.com/sd/article/S1047320318300683

R. Pagés, K. Amplianitis, D. Monaghan, J. Ondřej, A. Smolić,
Affordable Content Creation for Free-Viewpoint Video and VR/AR Applications,
J. Vis. Commun. Image R. (2018),
doi: https://doi.org/10.1016/j.jvcir.2018.03.012

2019 JVCI Best Paper Award



VOLOGRAMS

Trinity College Dublin
The University of Dublin

V-SENSE

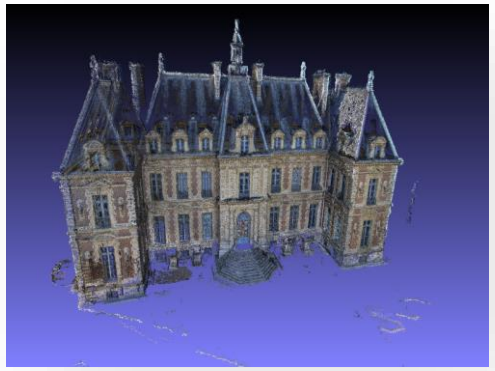

VOLOGRAMS

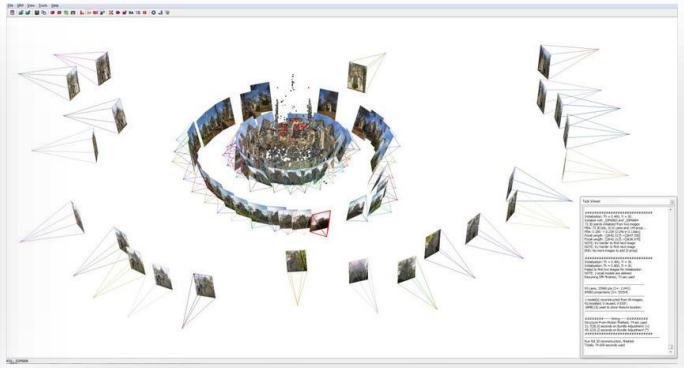
QUESTIONS?

3D reconstruction

Open Source Frameworks

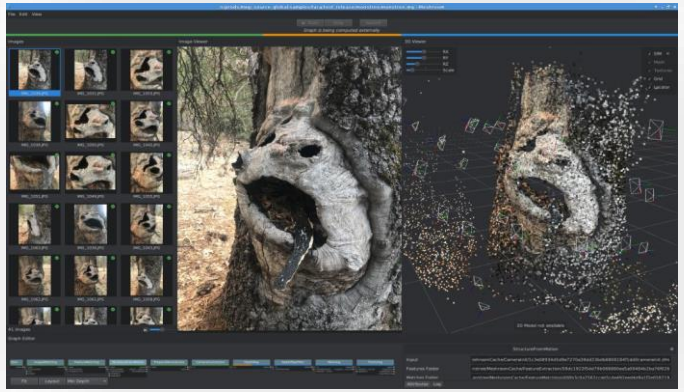
OpenMVG
https://openmvg.readthedocs.io





VisualSFM
http://ccwu.me/vsfm/

Colmap
https://colmap.github.io/



AliceVision Meshroom
https://alicevision.github.io/

VOLOGRAMS

SfM - Material

Book chapter:

Professor Roberto Chipolla, Dept. of Engineering, University of Cambridge
http://mi.eng.cam.ac.uk/~cipolla/publications/contributionToEditedBook/2008-SFM-chapters.pdf

Slides:

http://cs.nyu.edu/~fergus/teaching/vision_2012/6_Multiview_SfM.pdf
Lecture 6: Multi-view Stereo & Structure from Motion
Prof. Rob Fergus

Software:

https://alicevision.org
https://colmap.github.io/
http://www.cs.cornell.edu/~snavely/bundler/
http://ccwu.me/vsfm/
http://openmvg.readthedocs.io/en/latest/software/SfM/SfM/

Materials:

* Multiple View Geometry in Computer Vision, 2nd ed, Richard Hartley and Andrew Zisserman, Cambridge University Press, March 2004. - https://www.robots.ox.ac.uk/~vgg/hzbook/

* Computer Vision: Algorithms and Applications, Richard Szeliski, 2010

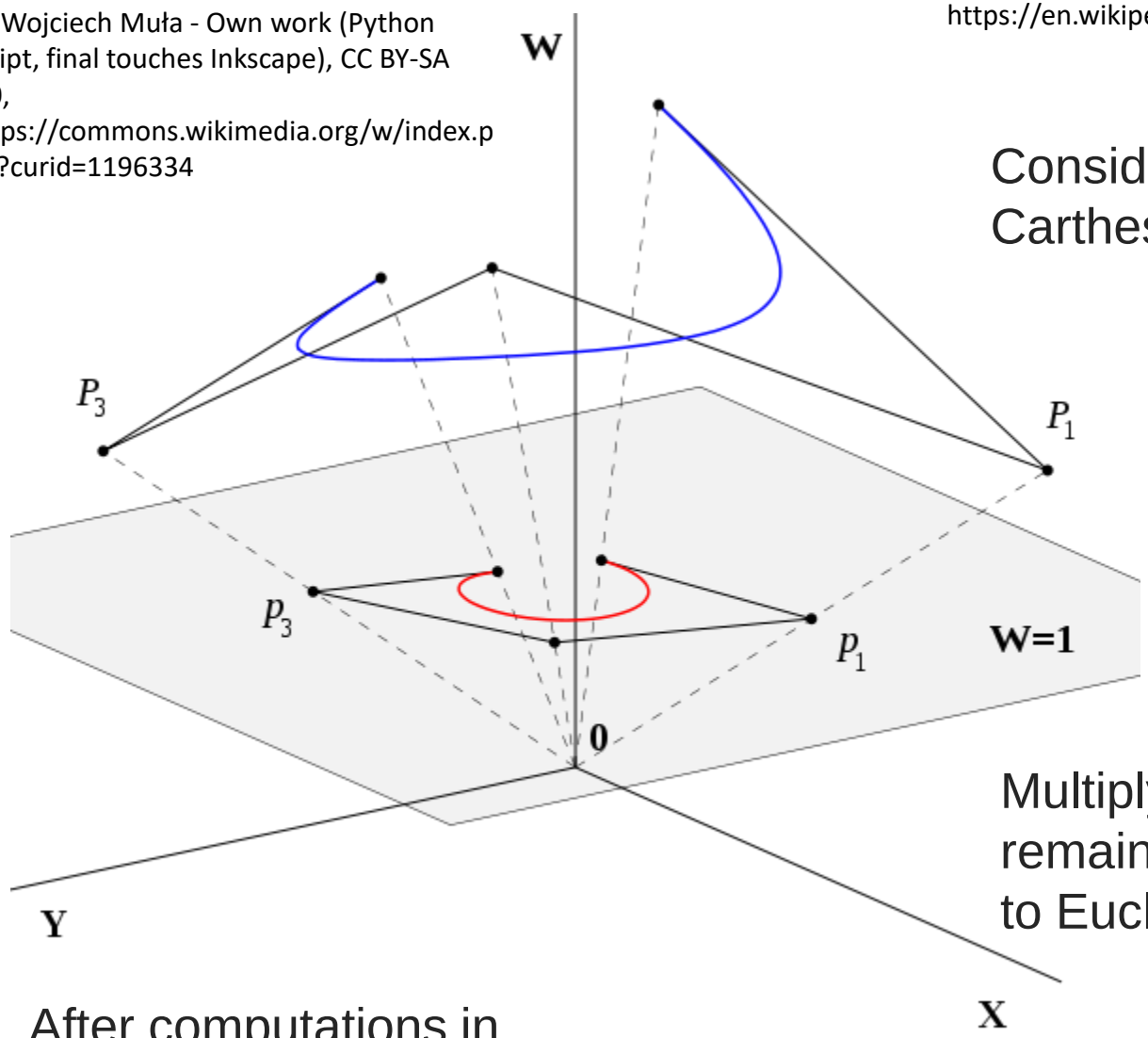
Vologrames Team via Looking Glass - https://blocks.glass/rafa/2107

Appendices

Appendix 1 – Homogenious Coordinates

Material by: Rachel McDonnell
Associate Professor in Creative Technologies

https://en.wikipedia.org/wiki/Homogeneous_coordinates



Consider a Euclidian plane with Carthesian coordinates $\begin{bmatrix} X_c \\ Y_c \end{bmatrix}$

Projected to a space called projective plane $W = 1$ $\begin{bmatrix} X_c \\ Y_c \end{bmatrix} \Rightarrow \begin{bmatrix} X_h \\ Y_h \\ 1 \end{bmatrix}$

Points are projected to lines

Multiplying a point by a scalar remains equivalent with regard to Euclidian plane

$$\begin{bmatrix} X_h \\ Y_h \\ W \end{bmatrix} \sim \lambda \cdot \begin{bmatrix} X_h \\ Y_h \\ W \end{bmatrix} = \begin{bmatrix} \lambda \cdot X_h \\ \lambda \cdot Y_h \\ \lambda \cdot W \end{bmatrix}$$

After computations in homogenious space, recover Carthesian coordinates by division by W $\begin{bmatrix} X_h / W \\ Y_h / W \\ 1 \end{bmatrix} \Rightarrow \begin{bmatrix} X_c \\ Y_c \end{bmatrix}$

Same works in higher dimensions

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \Rightarrow \begin{bmatrix} X_h \\ Y_h \\ Z_h \\ 1 \end{bmatrix}$$

https://en.wikipedia.org/wiki/Homogeneous_coordinates

•Any point in the projective plane is represented by a triple (*X*, *Y*, *W*), called the **homogeneous coordinates** or **projective coordinates** of the point, where *X*, *Y* and *W* are not all 0.

•The point represented by a given set of homogeneous coordinates is unchanged if the coordinates are multiplied by a common factor.

•Conversely, two sets of homogeneous coordinates represent the same point if and only if one is obtained from the other by multiplying all the coordinates by the same non-zero constant.

•When *W* is not 0 the point represented is the point (*X/W*, *Y/W*) in the Euclidean plane.

•When *W* is 0 the point represented is a point at infinity.

Note that the triple (0, 0, 0) is omitted and does not represent any point. The origin is represented by (0, 0, 1).[3]

https://en.wikipedia.org/wiki/Homogeneous_coordinates

Homogenous Coordinates

- Using this scheme, every rotation, translation, and scaling operation can be represented by a matrix multiplication, and any combination of the operations corresponds to the products of the corresponding matrices
- Using homogeneous co-ordinates allows us to treat translation in the same way as rotation and scaling

Appendix 2 – Direct Solution for Calibration

Material by: Aljosa Smolic
SFI Research Professor of Creative Technologies

Direct Solution

Given P, p_{hom} find C. (C has 11 dof so at least 11 eqns, 5.5 correspondences (2 eqns per corr) needed for solving)

$$\tilde{p} = C\tilde{P}$$

Source: Lectures Aljosa Smolic

Direct Solution

Homogenious multiplication

$$\tilde{p} = C\bar{P}$$

$$\Rightarrow \lambda\bar{p} = C\bar{P}$$

$$\Rightarrow \lambda\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = C\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

Source: Lectures Aljosa Smolic

Direct Solution

Reformulation

$$\tilde{p} = C\bar{P}$$

$$\Rightarrow \lambda\bar{p} = C\bar{P}$$

$$\Rightarrow \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = C \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} C^{1T}\bar{P} \\ C^{2T}\bar{P} \\ C^{3T}\bar{P} \end{bmatrix}$$

Source: Lectures Aljosa Smolic

Direct Solution

Substitution

$$\tilde{p} = C\bar{P}$$

$$\Rightarrow \lambda\bar{p} = C\bar{P}$$

$$\Rightarrow \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = C \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} C^{1T}\bar{P} \\ C^{2T}\bar{P} \\ C^{3T}\bar{P} \end{bmatrix}$$

Using row 3 in 1, 2

Source: Lectures Aljosa Smolic

Direct Solution

Reformulation

$$\tilde{p} = C\bar{P}$$

$$\Rightarrow \lambda\bar{p} = C\bar{P}$$

$$\Rightarrow \lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = C \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} \lambda x \\ \lambda y \\ \lambda \end{bmatrix} = \begin{bmatrix} C^{1T}\bar{P} \\ C^{2T}\bar{P} \\ C^{3T}\bar{P} \end{bmatrix}$$

Using row 3 in 1, 2

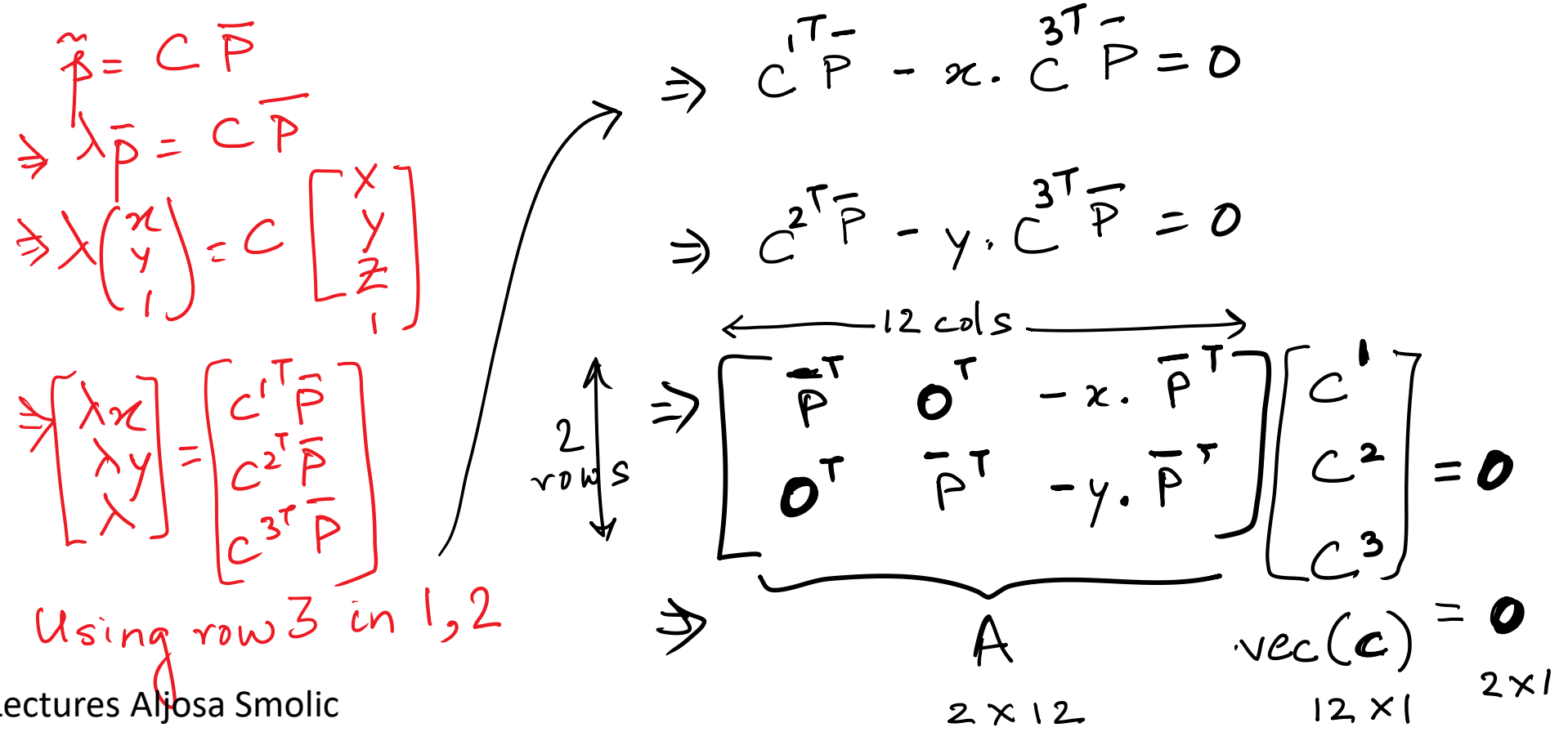
$$\Rightarrow C^{1T}\bar{P} - x \cdot C^{3T}\bar{P} = 0$$

$$\Rightarrow C^{2T}\bar{P} - y \cdot C^{3T}\bar{P} = 0$$

Source: Lectures Aljosa Smolic

Direct Solution

Reformulation as matrix equation



Source: Lectures Aljosa Smolic

Appendix 3 – Homography

Calibration to Homography Estimation

Given P, p_{hom} find C. (C has 11 dof so at least 11 eqns, 5.5 correspondences (2 eqns per corr) needed for solving)

$$\mathbf{p} = \mathbf{C} \cdot \mathbf{P}$$

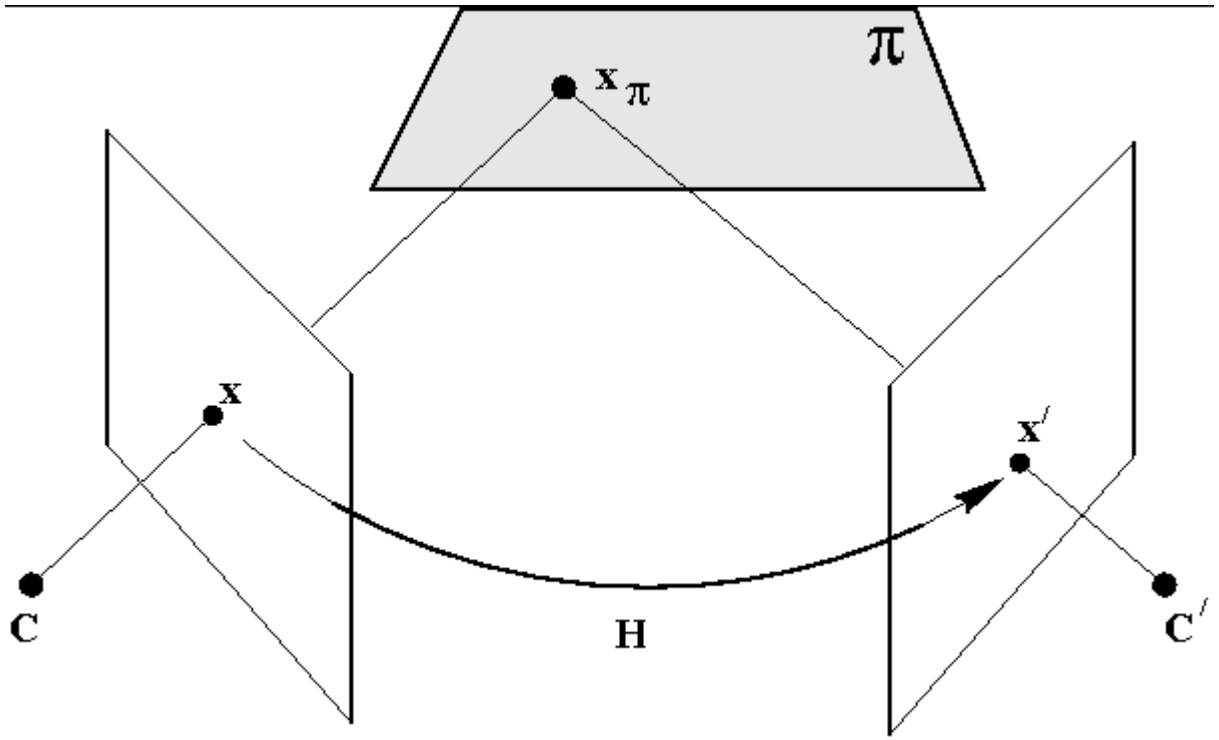
In **special cases** this is simplified to a homography estimation, relating **image pixels to image pixels**

$$\mathbf{p}_1 = \mathbf{H} \cdot \mathbf{p}_2$$

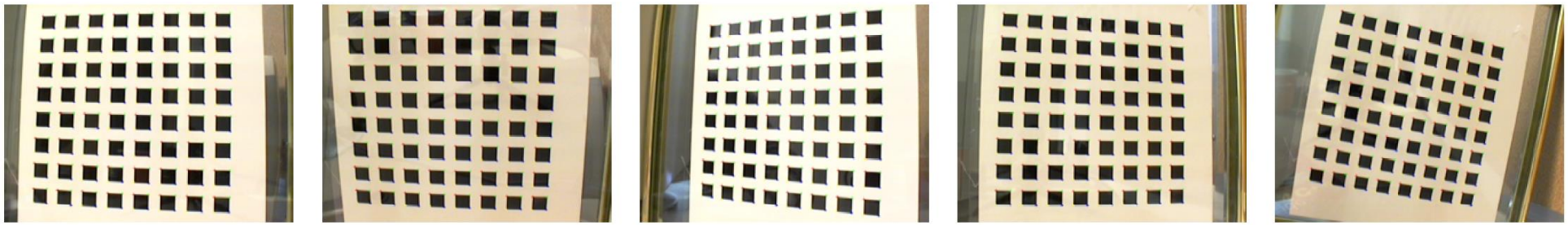
H is a 3x3 matrix with 8 degrees of freedom, i.e. defined up to an arbitrary scale factor

Homography

Given an image of the model plane, an homography can be estimated (see Appendix A). Let's denote it by $\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix}$. From (2), we have



http://www.learnopencv.com/homography-examples-using-opencv-python-c/



Homography

- Consider a point x = (u,v,1) in one image and x'=(u',v',1) in another image
- A homography is a 3 by 3 matrix M
- $$M = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix}$$

- The homography relates the pixel co-ordinates in the two images if x' = M x
- When applied to every pixel the new image is a warped version of the original image

Dr Gerhard Roth
http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf

Homography

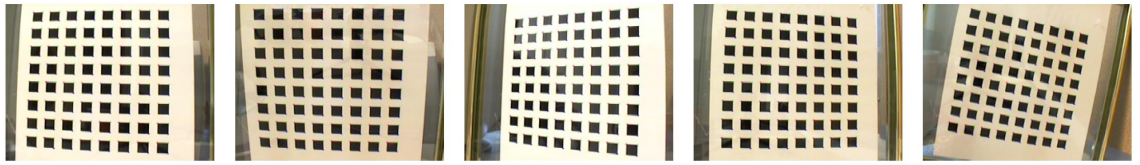
- Two images are related by a homography if and only if
- Both images are viewing the same plane from a different angle
- Both images are taken from the same camera but from a different angle
  - Camera is rotated about its center of projection without any translation
- Note that the homography relationship is independent of the scene structure
  - It does not depend on what the cameras are looking at
  - Relationship holds regardless of what is seen in the images

Dr Gerhard Roth
http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf

Appendix 4 – OpenCV Calibration Toolbox

https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#

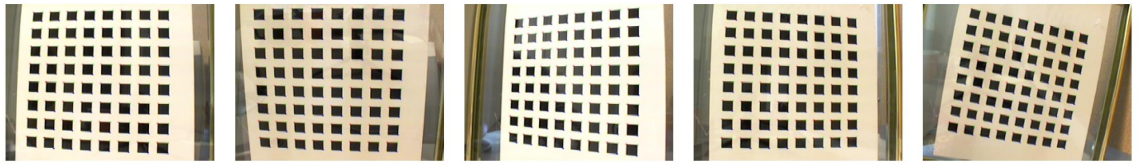
OpenCV Functions



- **CalibrateCamera2:** Finds intrinsic and extrinsic camera parameters using calibration pattern

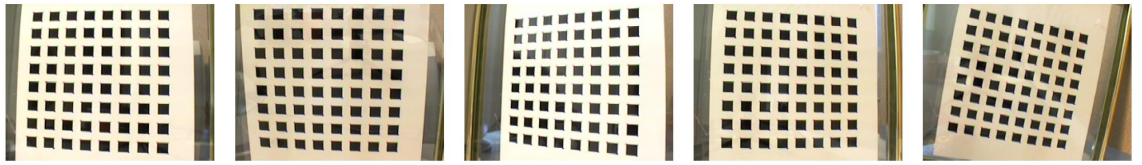
- void **cvCalibrateCamera2**(const CvMat* object_points, const CvMat* image_points, const CvMat* point_counts, CvSize image_size, CvMat* intrinsic_matrix, CvMat* distortion_coeffs, CvMat* rotation_vectors=NULL, CvMat* translation_vectors=NULL, int flags=0);

OpenCV Functions



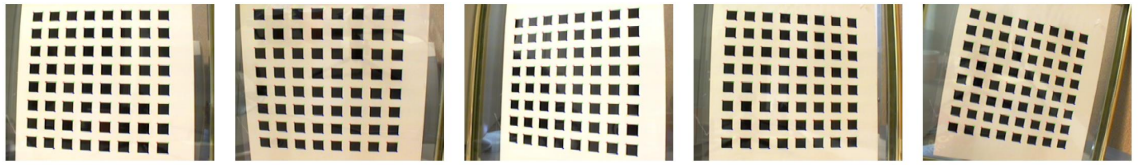
- object_points: The joint matrix of object points, 3xN or Nx3, where N is the total number of points in all views.
- image_points: The joint matrix of corresponding image points, 2xN or Nx2, where N is the total number of points in all views.
- point_counts: Vector containing numbers of points in each particular view, 1xM or Mx1, where M is the number of a scene views.
- image_size: Size of the image, used only to initialize intrinsic camera matrix.

OpenCV Functions



- intrinsic_matrix: The output camera matrix (A) [fx 0 cx; 0 fy cy; 0 0 1]. If CV_CALIB_USE_INTRINSIC_GUESS and/or CV_CALIB_FIX_ASPECT_RATION are specified, some or all of fx, fy, cx, cy must be initialized.
- distortion_coeffs: The output 4x1 or 1x4 vector of distortion coefficients [k1, k2, p1, p2].
- rotation_vectors: The output 3xM or Mx3 array of rotation vectors (compact representation of rotation matrices, see cvRodrigues2).
- translation_vectors: The output 3xM or Mx3 array of translation vectors.

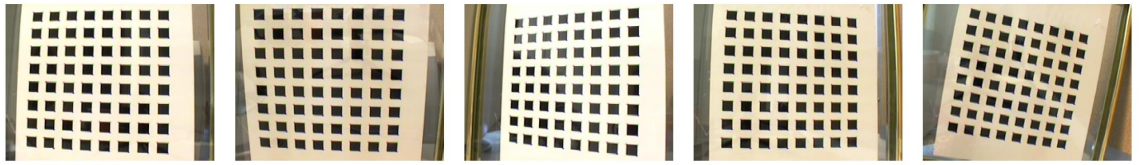
OpenCV Functions



- **FindChessboardCorners:** Finds positions of internal corners of the chessboard

- int **cvFindChessboardCorners**(const void* image, CvSize pattern_size, CvPoint2D32f* corners, int* corner_count=NULL, int flags=CV_CALIB_CB_ADAPTIVE_THRESH);

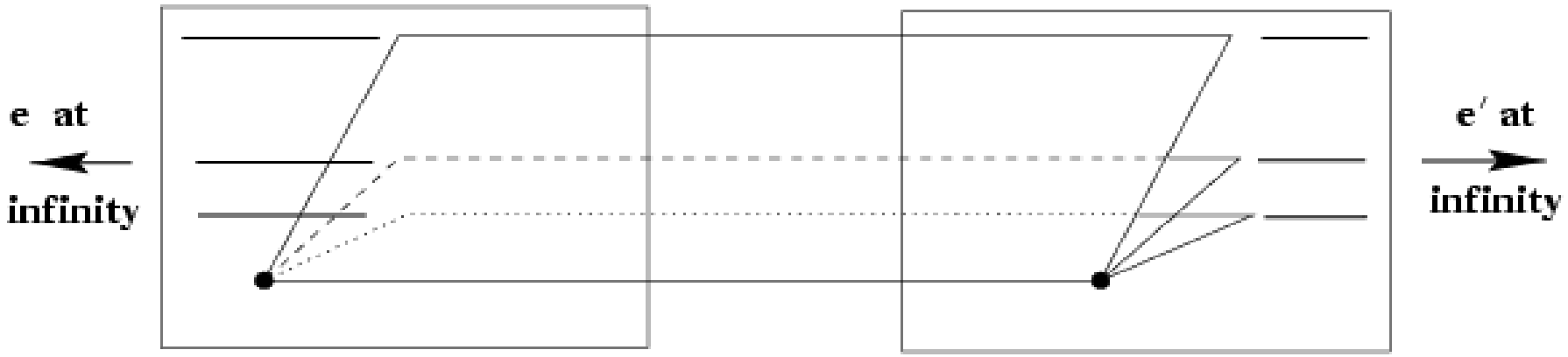
OpenCV Functions



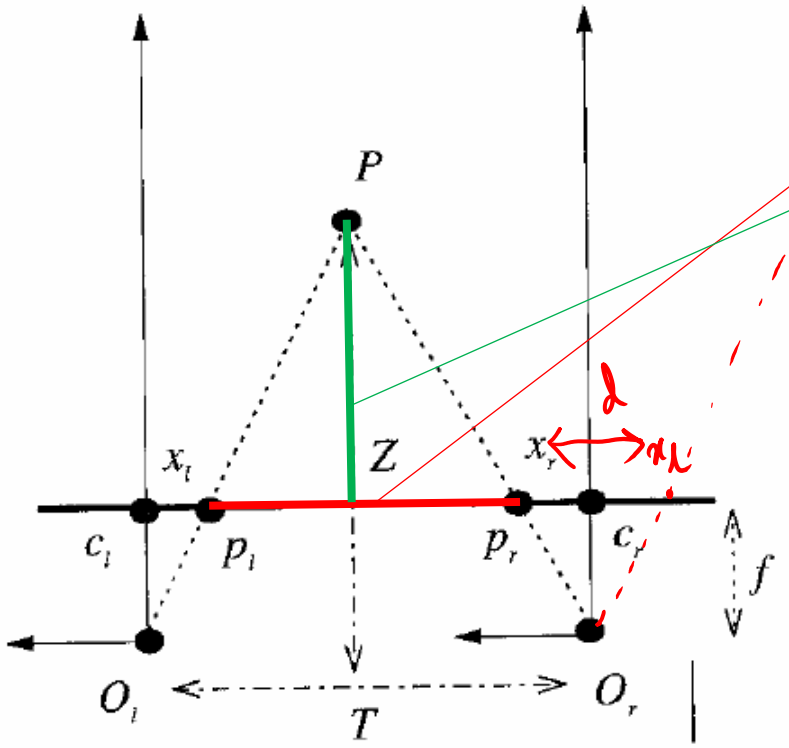
- Image: Source chessboard view; it must be 8-bit grayscale or color image.
- pattern_size: The number of inner corners per chessboard row and column.
- Corners: The output array of corners detected.
- corner_count: The output corner counter. If it is not NULL, the function stores there the number of corners found.

Appendix 5 – Rectification
The case of parallel optical axis

Parallel Image Planes



Parallel (or Rectified) Geometry



- Derive Z (depth)

$$\frac{T - x_l - x_r}{Z - f} = \frac{T}{Z}$$

- Disparity $d = x_l - x_r$
- Finally, $Z = f\frac{T}{d}$
- depth inv. prop to disparity

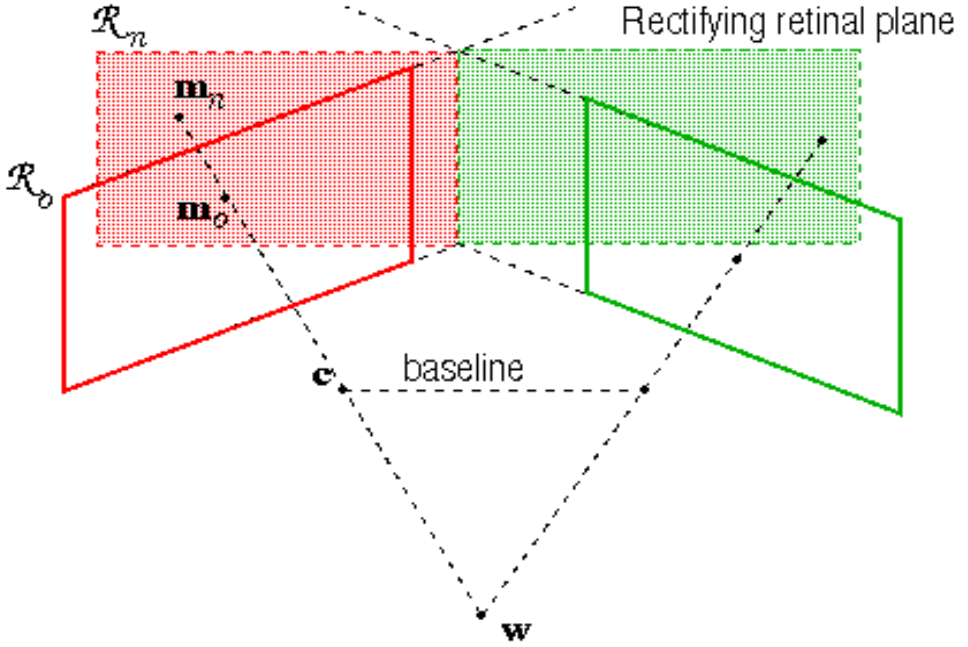
- Courtesy: Gerhard Roth

Depth Estimation from 2 Cameras

- Calibrate setup, e.g. with a calibration pattern, estimate intrinsics and extrinsics
- Rectify images, i.e. project them onto a common plane parallel to baseline
- Now epipolar lines are parallel
- Search for correspondences along image scan lines
- Estimated disparities are inversely proportional to depth
- Calculate depth by

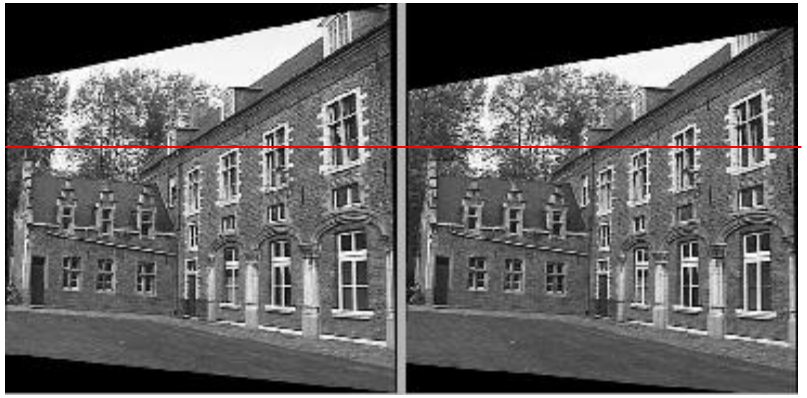
$$Z = f\frac{T}{d}$$

Planar Rectification



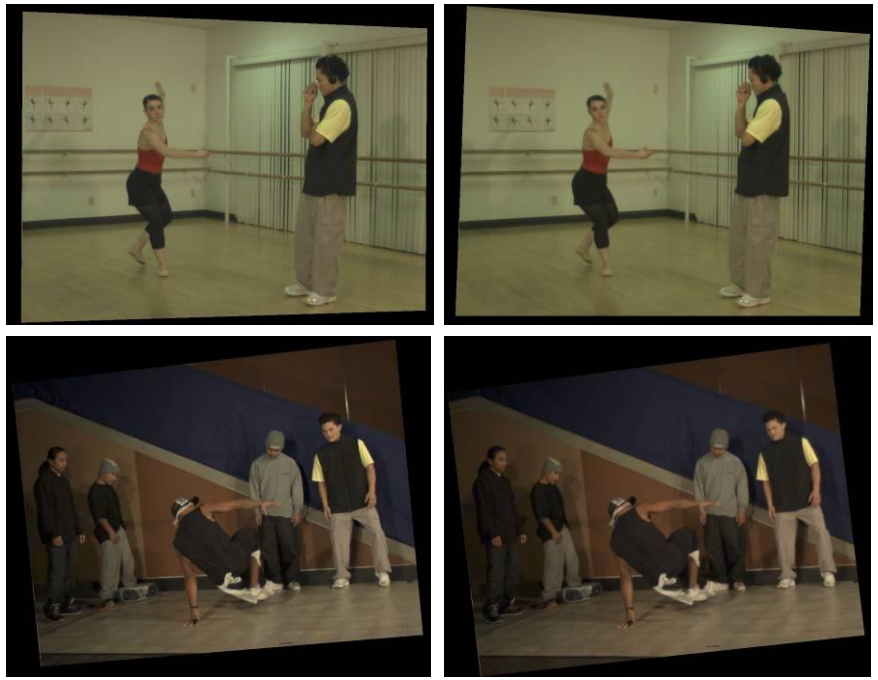
- The image pair is projected to a plane parallel to the baseline

Rectification Examples



Planar Rectification

Rectification Examples



Pablo Carballeira, Francisco Morán, Julián Cabrera, UPM

Appendix 6 – Fundamental Matrix additional material

Skew Symmetric Matrix

A skew-symmetric $\iff$ $a_{ji} = -a_{ij}$.

$A^{\mathsf{T}} = -A.$

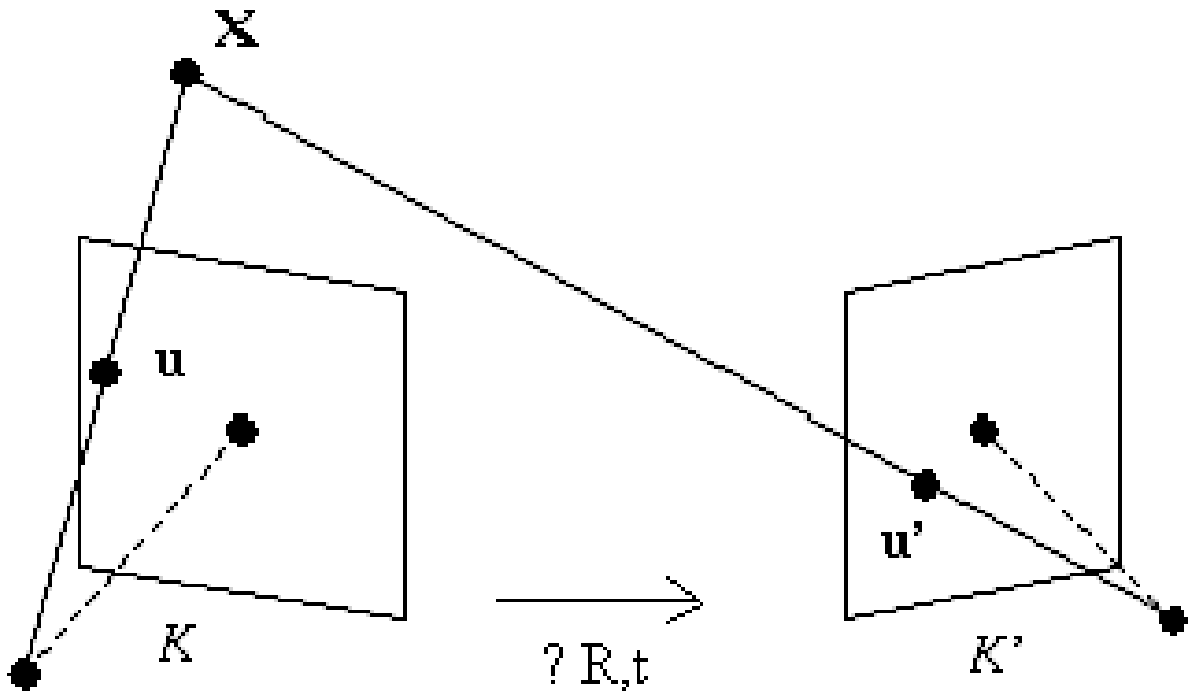
$$a \times b = [a]_{\times} b$$

where

$$[a]_{\times} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}$$

Ego Motion Estimation

- Calibrated camera(s)
- Unknown movement (R & t)
- Algorithm
  - Find correspondences (u_i & u'_i)
  - Normalise data
  - Compute the fundamental matrix
  - Compute the essential matrix
  - Determine R & t from the essential matrix (e.g. using singular value decomposition)



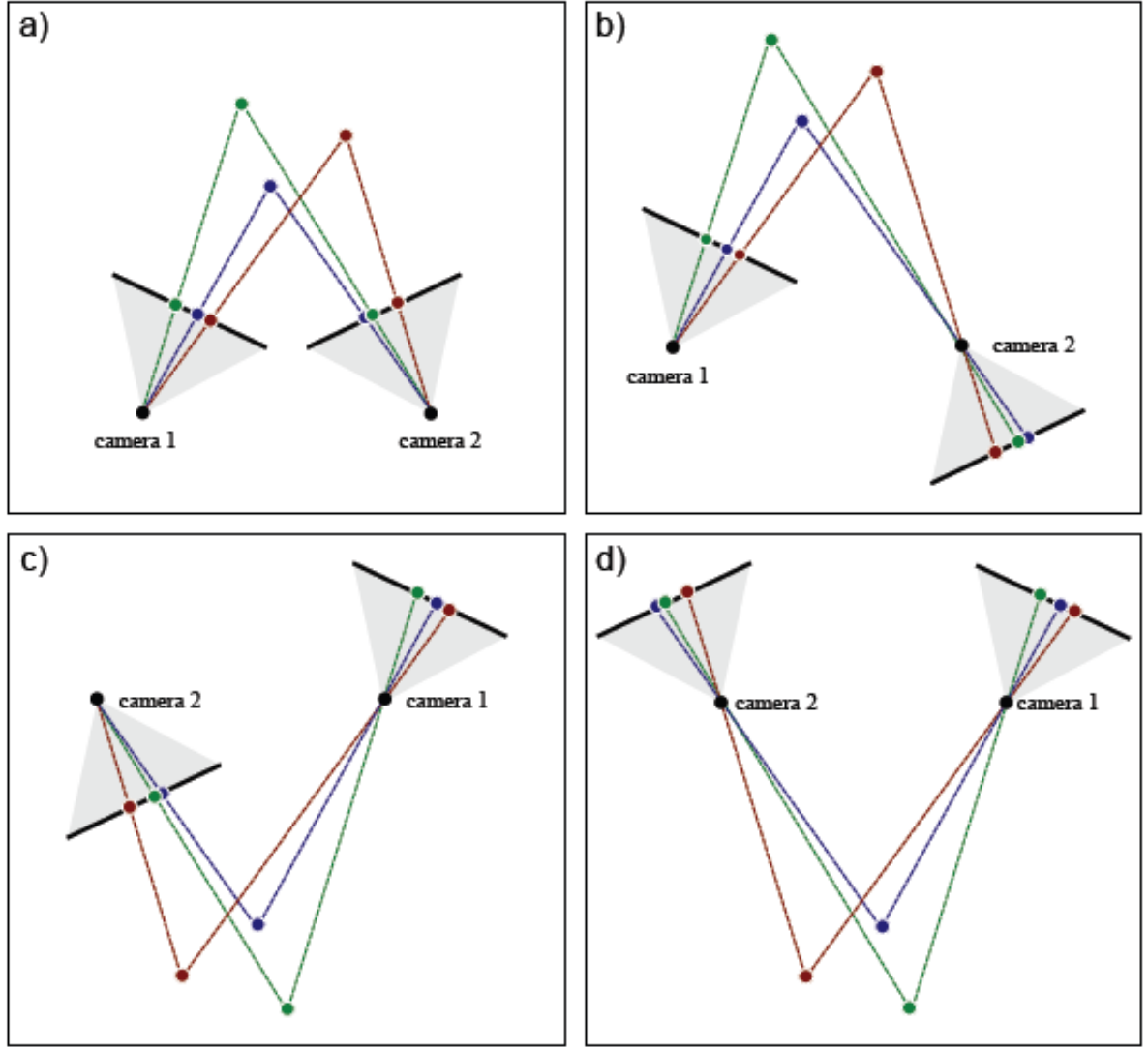


Figure 16.4 Four-fold ambiguity of reconstruction from two pinhole cameras. The mathematical model for the pinhole camera does not distinguish between points that are in front of and points that are behind the camera. This leads to a four-fold ambiguity when we extract the rotation Ω and translation τ relating the cameras from the essential matrix. a) Correct solution. Points are in front of both cameras. b) Incorrect solution. The images are identical, but with this interpretation, the points are behind camera 2. c) Incorrect solution with points behind camera 1. d) Incorrect solution with points behind both cameras.

- Must disambiguate between the four solutions so that it leads to valid camera configuration with points in front.
- We now have estimates of intrinsics and extrinsics

Fundamental Matrix, Additional Material

Multiple View Geometry in Computer Vision
Second Edition
Richard Hartley and Andrew Zisserman,
Cambridge University Press, March 2004.
http://www.robots.ox.ac.uk/~vgg/hzbook/
http://www.robots.ox.ac.uk/~vgg/hzbook/hzbook2/HZepipolar.pdf
http://www.robots.ox.ac.uk/~vgg/hzbook/code/

- http://szeliski.org/Book/drafts/SzeliskiBook_20100903_draft.pdf
- Computer Vision: Algorithms and Applications, Richard Szeliski, September 3, 2010 draft, 2010 Springer
- Section 7.2

Fundamental Matrix Estimation in OpenCV

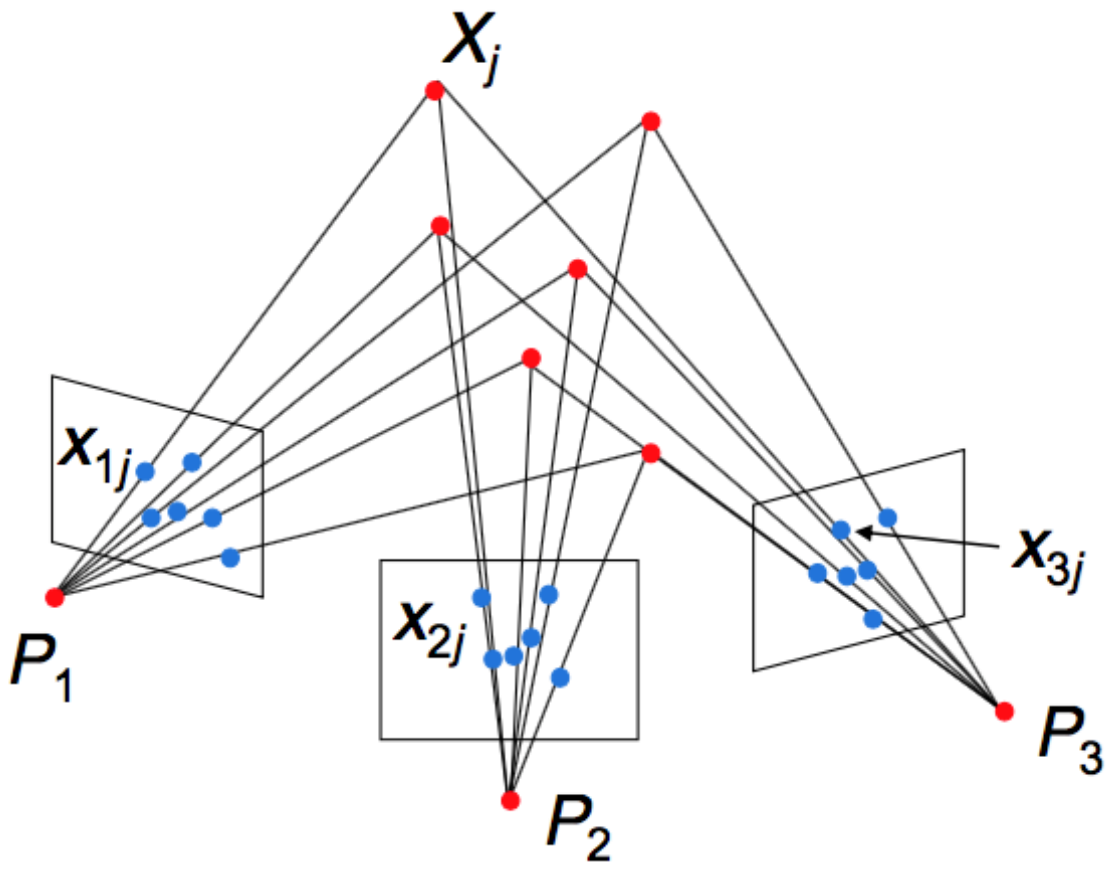
```
int cvFindFundamentalMat(
        const CvMat* points1,
        const CvMat* points2,
        CvMat* fundamental_matrix,
        int method=CV_FM_RANSAC,
        double param1=1.,
        Double param2=0.99,
        CvMat* status=NULL
);
```

http://www.robots.ox.ac.uk/~vgg/hzbook/code/

Appendix 7 – Multi-view Stereo (Getting dense point cloud)

Multi-view stereo

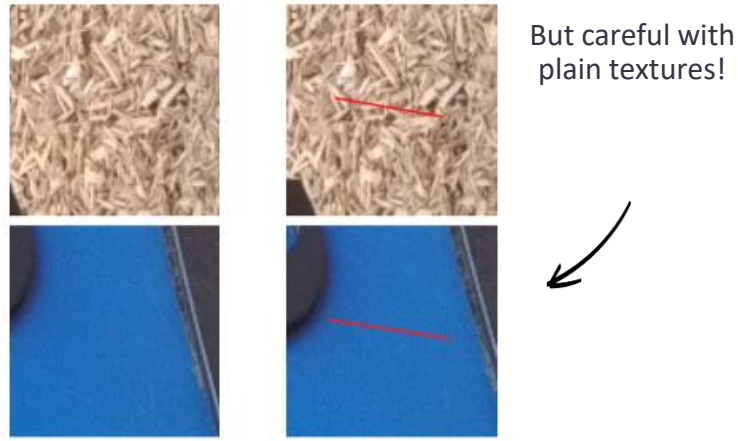
Depth map estimation



If we project the sparse 3D points back to the images that generated each point, we can search in the neighboring pixels for additional matches.

Using a photo-consistency metric (for example normalized cross-correlation) we can find additional matches.

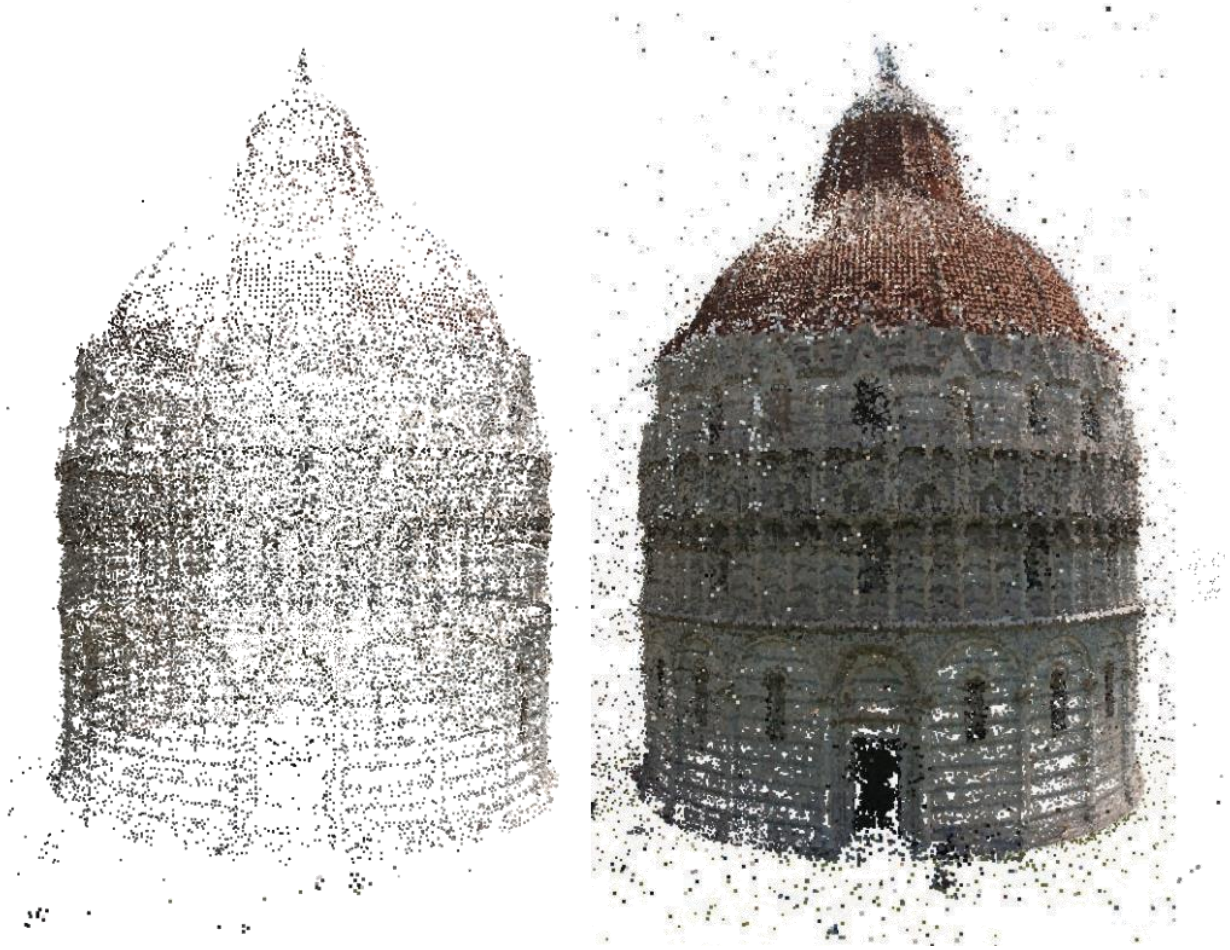
We can filter these new matches using geometric constrains.



Multi-view stereo: a tutorial, Furukawa and Hernández

Multi-view stereo

Dense point clouds



VOLOGRAMS

Furukawa and Ponce. *Accurate, Dense, and Robust Multi-View Stereopsis.* 2010.

KU Leuven, Marc Pollefeys, Luc Van Gool

artoolkitx

- Open source SW for augmented reality
- Marker detection, pose estimation, rendering

http://www.artoolkitx.org