

Lab Course Kinect Programming for Computer Vision

Transformations and Camera Calibration

Loren Schwarz (schwarz@in.tum.de)

May 25, 2011

Lecture Outline

Part I: Basics of Transformations

Part II: Basics of Cameras

Outline Part I

- 2D Transformations
- Homogeneous Coordinates in 2D
- Hierarchy of Transformations in 2D
- Hierarchy of Transformations in 3D

2D Transformations

The basic building blocks of transformations are:

- Scaling (isotropic, non-isotropic)
- Rotation (around one rotation axis)
- Translation (motion along the two axes)

Transformations are defined by their action on point coordinates.

Notation:

$$\mathbf{p} = (x, y)^{\top} = \begin{bmatrix} x \\ y \end{bmatrix}$$

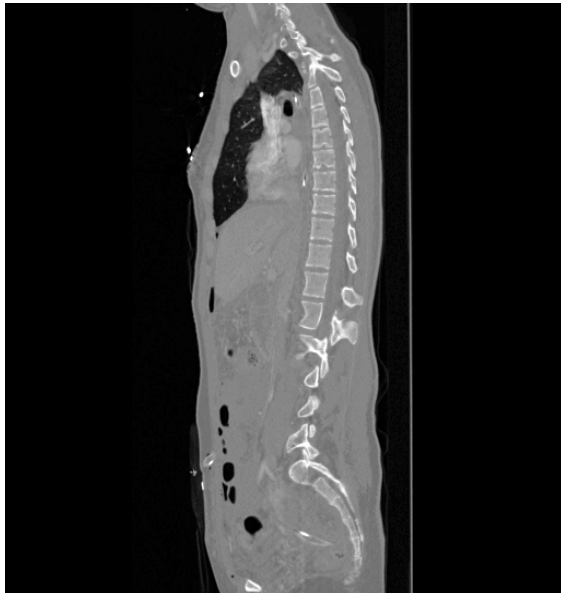
original point

$$\mathbf{p}' = (x', y')^{\top} = \begin{bmatrix} x' \\ y' \end{bmatrix}$$

transformed point

2D Transformations – Scaling

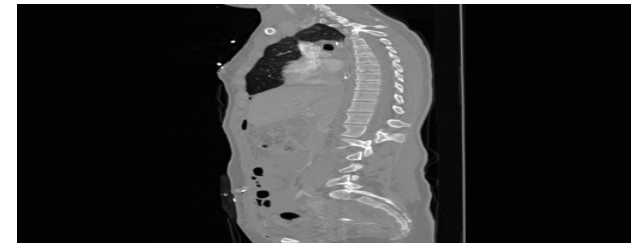
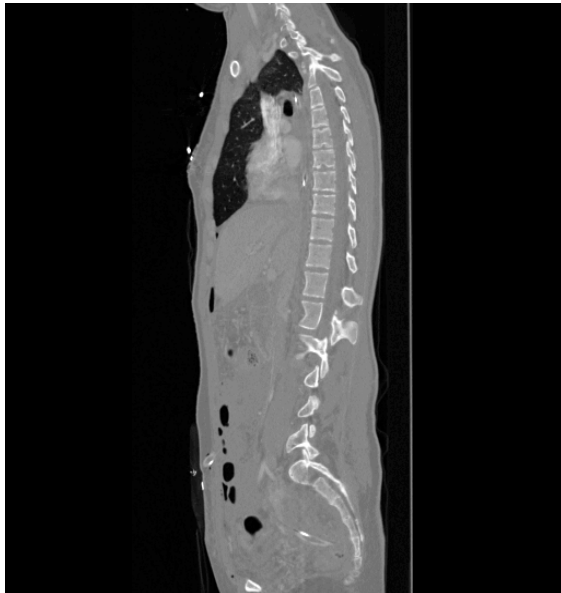
Isotropic scaling: same scaling factor in both spatial dimensions



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

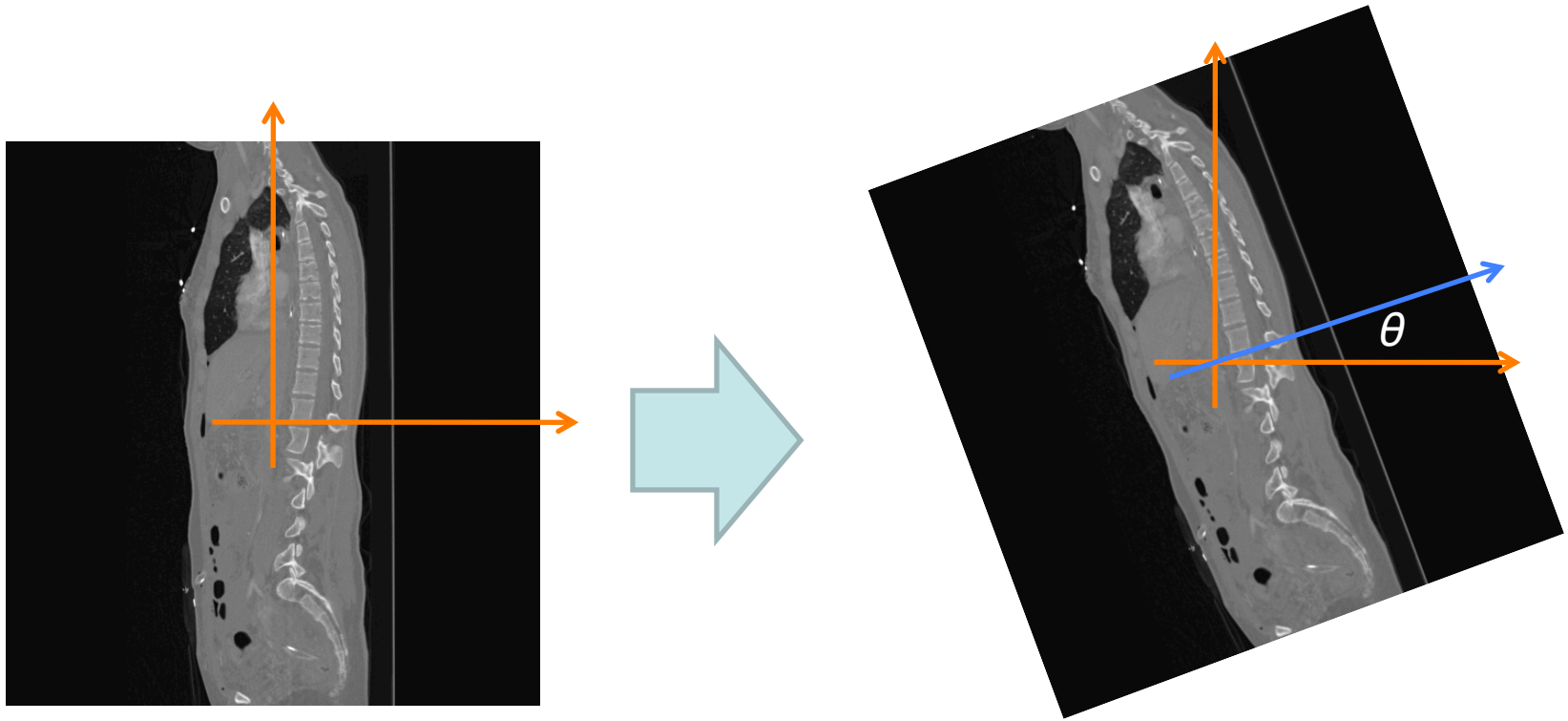
2D Transformations – Scaling

Non-isotropic scaling: different scaling factors in the two spatial dimensions



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

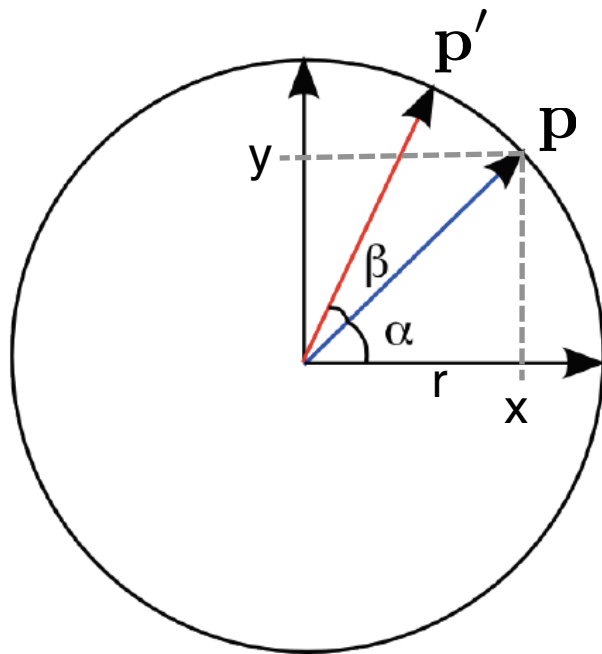
2D Transformations – Rotation



$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

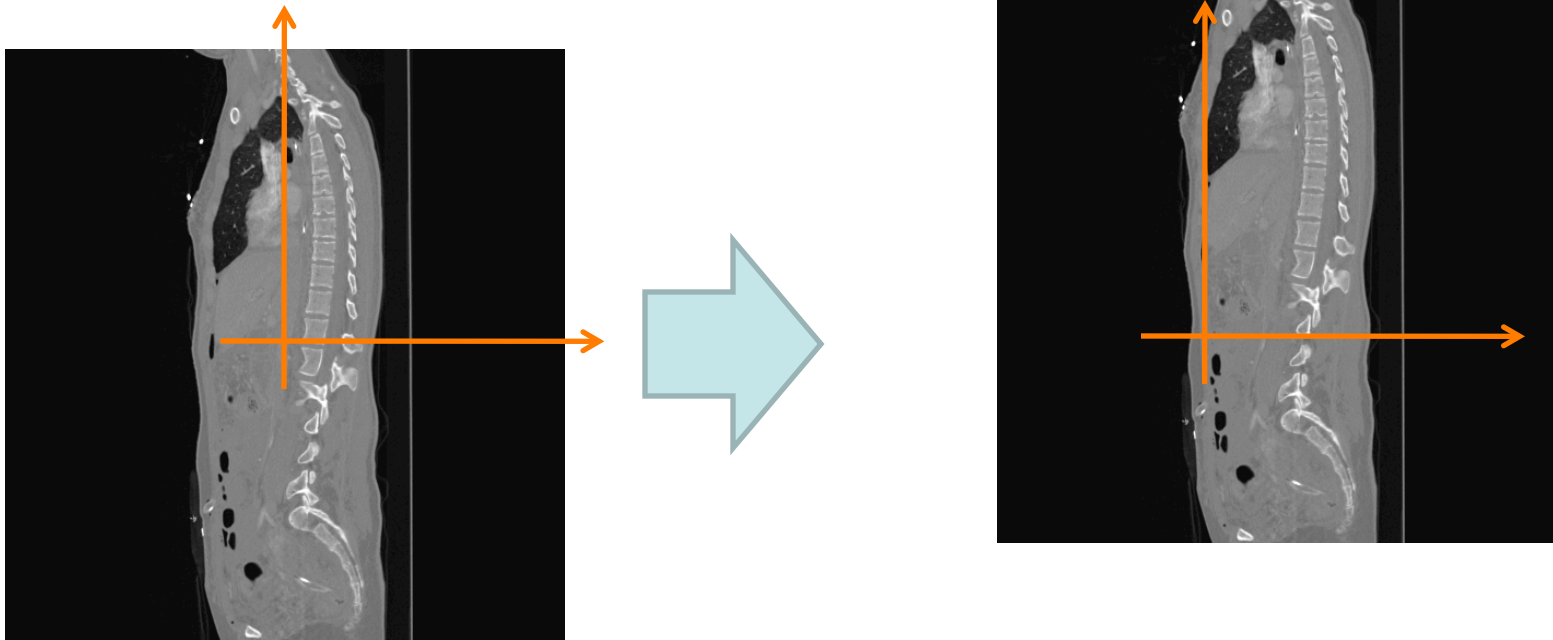
2D Transformations – Rotation

$$\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} r \cos(\alpha) \\ r \sin(\alpha) \end{pmatrix}$$



$$\begin{aligned} \mathbf{p}' &= \begin{pmatrix} r \cos(\alpha + \beta) \\ r \sin(\alpha + \beta) \end{pmatrix} \\ &= \begin{pmatrix} r \cos(\alpha) \cos(\beta) - r \sin(\alpha) \sin(\beta) \\ r \sin(\alpha) \cos(\beta) + r \cos(\alpha) \sin(\beta) \end{pmatrix} \\ &= \begin{pmatrix} x \cos(\beta) - y \sin(\beta) \\ y \cos(\beta) + x \sin(\beta) \end{pmatrix} \\ &= \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \\ &= \begin{pmatrix} \cos(\beta) & -\sin(\beta) \\ \sin(\beta) & \cos(\beta) \end{pmatrix} \mathbf{p} \end{aligned}$$

2D Transformations – Translation



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

2D Transformations

Have you noticed we have two types of representations for the basic transformations seen so far?

Scaling, Rotation: **Matrix-vector product**

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Translation: **Sum of vectors**

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

How to achieve a matrix-vector product form for all types of transformations?

Outline Part I

- 2D Transformations
- Homogeneous Coordinates in 2D
- Hierarchy of Transformations in 2D
- Hierarchy of Transformations in 3D

Homogeneous Coordinates in 2D

How to achieve a matrix-vector product form for all types of transformations?

- Write a 2D point in **homogeneous coordinates**:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \in \mathbb{R}^3$$

We will see more properties of homogeneous representation later.

- Then, translation can easily be expressed as a matrix-vector product:

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

Homogeneous Coordinates in 2D

- Rotation in homogeneous representation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Combined transformations (here: rotation and translation):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x \cos \theta - y \sin \theta + t_x \\ x \sin \theta + y \cos \theta + t_y \\ 1 \end{bmatrix}$$

2D Projective Space

When dealing with points represented in homogeneous coordinates, we are working in a Projective space.

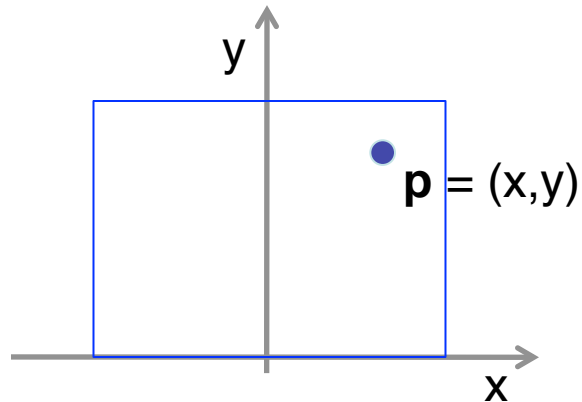
- Points in 2D **Euclidean** space are represented by **Cartesian** coordinates

$$\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$$

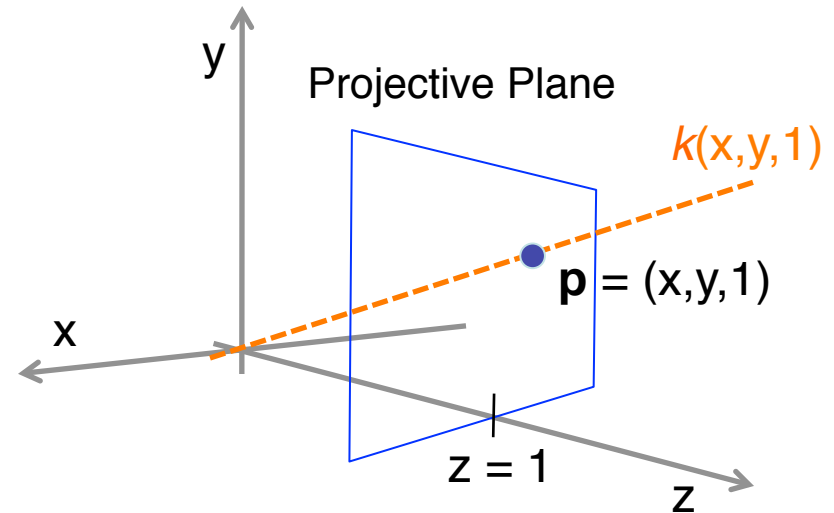
- Points in 2D **Projective** space are represented by **Homogeneous** coordinates:

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \in \mathbb{P}^2$$

2D Projective Space



Point in **Euclidean** 2D space



Point in **Projective** 2D space.

All points on the orange line represent the **same point**. A homogeneous vector in 2D projective space represents an **equivalence class** of points.

Points and Lines in 2D Projective Space

Properties of points

- A 2D point $(x, y)^\top$ has the **homogeneous** representation $(x, y, 1)^\top$.
- The homogeneous vector $(x, y, w)^\top$ with $w \neq 0$ represents the point $(\frac{x}{w}, \frac{y}{w}, 1)^\top$.
- For any non-zero k , $(x, y, w)^\top$ and $k(x, y, w)^\top$ represent the same point.

Points and Lines in 2D Projective Space

Properties of lines

- A line $ax + by + c = 0$ has the homogeneous representation $\mathbf{l} = (a, b, c)^\top$.
- A point \mathbf{x} lies on a line \mathbf{l} if and only if $\mathbf{x}^\top \mathbf{l} = 0$.
- Two lines \mathbf{l} and \mathbf{l}' intersect in the point $\mathbf{x} = \mathbf{l} \times \mathbf{l}'$.
- The line joining the two points \mathbf{x} and \mathbf{x}' is $\mathbf{l} = \mathbf{x} \times \mathbf{x}'$.

Points and Lines in 2D Projective Space

Special points and lines

- Points with 3rd coordinate zero $(x, y, 0)^\top$ are called **points at infinity**.
- All points at infinity lie on the **line at infinity** $\mathbf{l}_\infty = (0, 0, 1)^\top$.

$$\mathbf{l}_\infty^\top (x, y, 0) = (0, 0, 1)^\top (x, y, 0) = 0$$

- Parallel lines intersect at infinity.

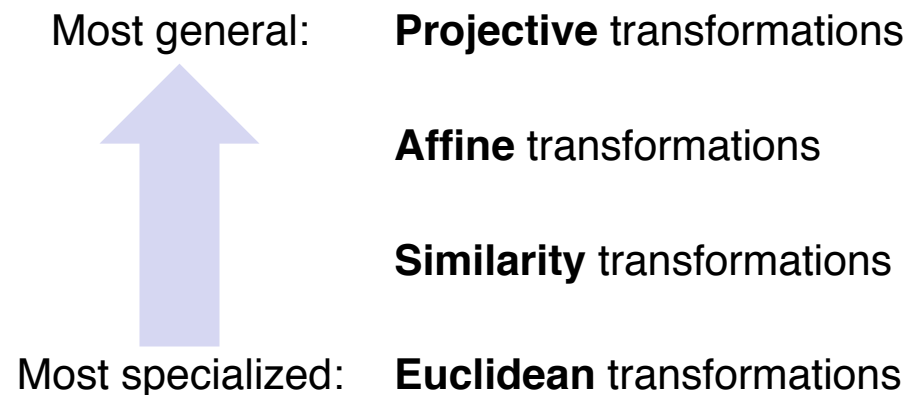
In projective space, all lines intersect in a point, even parallel lines.

Outline Part I

- 2D Transformations
- Homogeneous Coordinates in 2D
- Hierarchy of Transformations in 2D
- Hierarchy of Transformations in 3D

Hierarchy of Transformations in 2D

- Classification of transformations in terms of quantities or properties that are **invariant** (left unchanged) by the transformation
- Hierarchy: Starting from most specialized type of transformation, successively remove invariants to get more general types of transformations



- Containment relation from general to specialized, e.g.: affine transformations have all properties of projective transformations, plus additional ones.

Hierarchy of Transformations in 2D: Euclidean Transformations

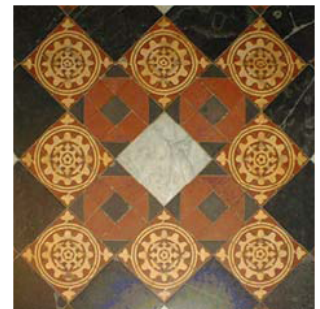
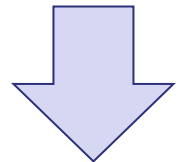
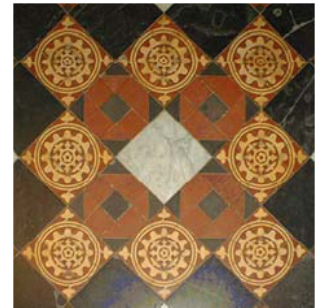
- Transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & t_x \\ \sin \theta & \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{euclidean}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \mathbf{R}^\top \mathbf{R} = \mathbf{I} \\ \det(\mathbf{R}) = 1$$

- Degrees of freedom:
 - 3 (1 rotation, 2 translation)
- Invariants: length, area, angles



Hierarchy of Transformations in 2D: Similarity Transformations

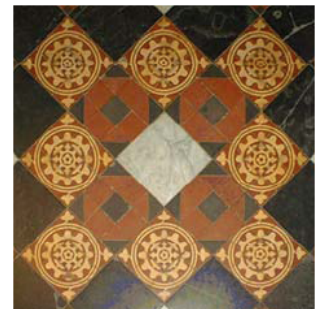
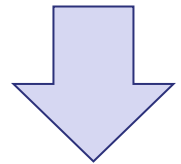
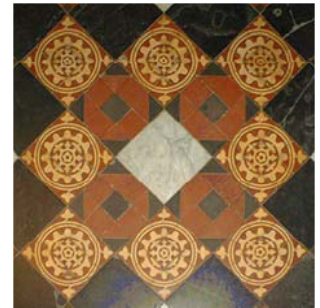
- Transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s \cos \theta & -s \sin \theta & t_x \\ s \sin \theta & s \cos \theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{similarity}} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \begin{array}{l} \mathbf{R}^\top \mathbf{R} = \mathbf{I} \\ \det(\mathbf{R}) = 1 \end{array}$$

- Degrees of freedom:
 - 4 (1 rotation, 2 translation, 1 isotropic scaling)
- Invariants: ratio of lengths, angles, parallel lines



Hierarchy of Transformations in 2D: Affine Transformations

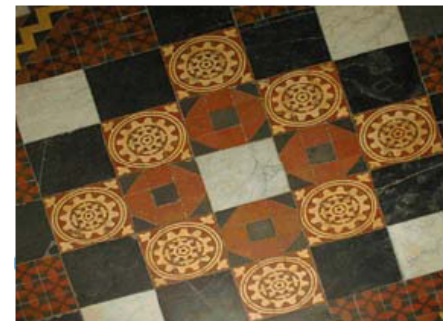
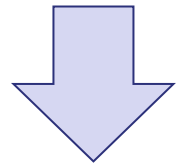
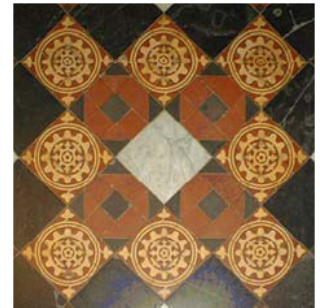
- Transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{affinity}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \mathbf{A} \text{ non-singular}$$

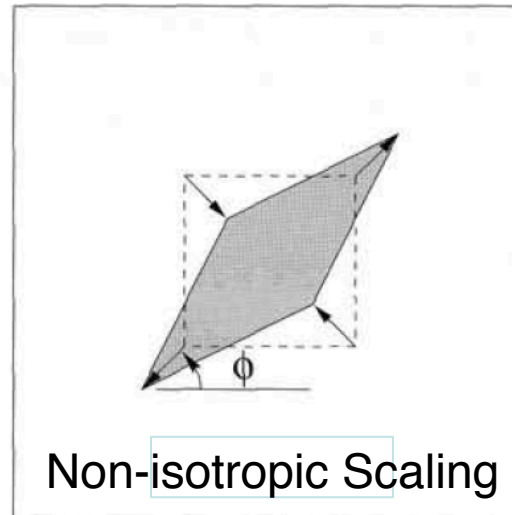
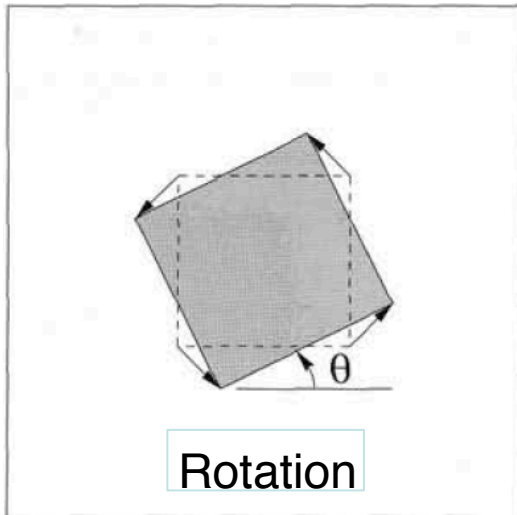
- Degrees of freedom:
 - 6 (2 translation, 4 entries of \mathbf{A})
- Invariants: parallel lines, ratios of areas, line at infinity



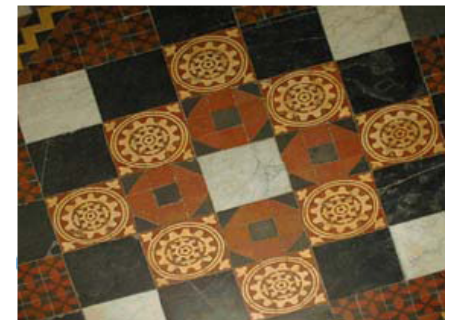
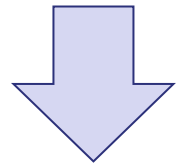
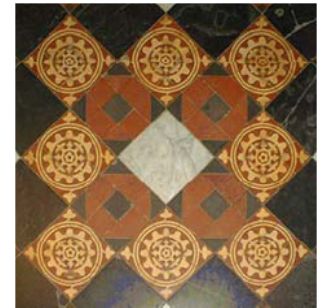
Hierarchy of Transformations in 2D:

Affine Transformations

Affine transformations geometrically consist of two components:



$$\mathbf{A} = \mathbf{R}(\theta)\mathbf{R}(-\phi) \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \mathbf{R}(\phi)$$



Hierarchy of Transformations in 2D: Projective Transformations

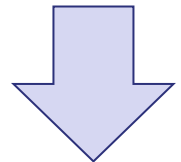
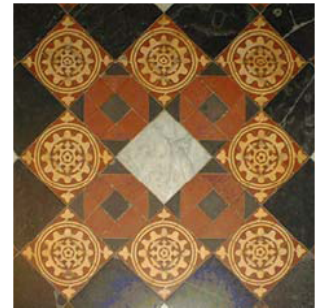
- Transformation:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & v \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

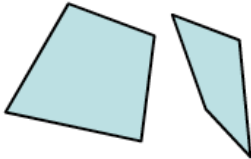
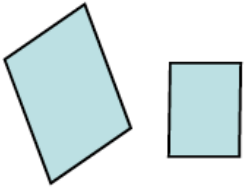
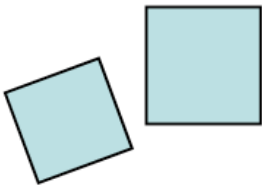
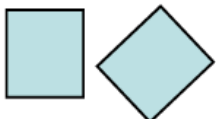
- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{projectivity}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix} \quad \mathbf{A} \text{ non-singular}$$

- Degrees of freedom:
 - 8 (9 arbitrary matrix entries, up to scale)
- Invariants: Intersection points, tangency, cross-ratios



Hierarchy of Transformations in 2D – Summary

Projective	8 DOF		$\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix}$
Affine	6 DOF		$\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Similarity	4 DOF		$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Euclidean	3 DOF		$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$

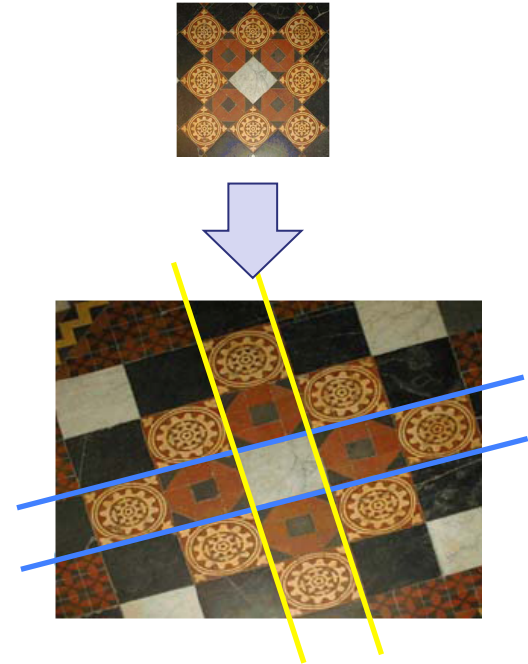
Action of Transformations in 2D on Points at Infinity

- Affine Transformations:**

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \\ 0 \end{bmatrix}$$

Point at infinity,
e.g. intersection
of blue lines

Mapped by affine
transformation to another
point at infinity



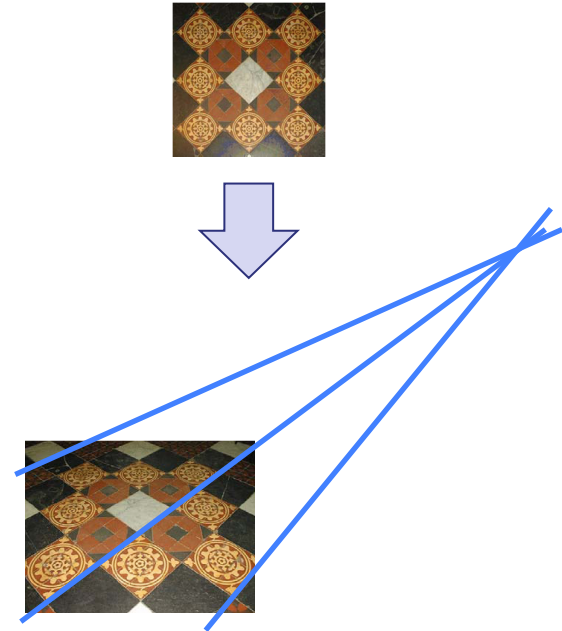
Action of Transformations in 2D on Points at Infinity

- **Projective Transformations:**

$$\begin{bmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ v_1 & v_2 & v \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \end{bmatrix} = \begin{bmatrix} a_{11}x + a_{12}y \\ a_{21}x + a_{22}y \\ v_1x + v_2y \end{bmatrix}$$

Point at infinity,
e.g. intersection
of blue lines

Mapped by projective
transformation to a finite
point (**vanishing point**)



Only projective transformations can make points at infinity finite points.

Outline Part I

- 2D Transformations
- Homogeneous Coordinates in 2D
- Hierarchy of Transformations in 2D
- Hierarchy of Transformations in 3D

Homogeneous Coordinates in 3D

How to achieve a matrix-vector product form for all types of transformations?

- Write a 3D point in **homogeneous coordinates**:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \in \mathbb{R}^4$$

- 3D projective space: properties analogous to 2D homogeneous representation
- All types of transformations (scaling, translation, rotation, etc.) can be written as a matrix-vector product
- Several transformations can be composed into one transformation matrix

Hierarchy of Transformations in 3D (1): Euclidean Transformations

- Transformation:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{euclidean}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \begin{array}{l} \mathbf{R}^\top \mathbf{R} = \mathbf{I} \\ \det(\mathbf{R}) = 1 \end{array}$$

- Degrees of freedom:
 - 6 (3 rotation, 3 translation)
- Invariants: volume, angles

Hierarchy of Transformations in 3D (2): Similarity Transformations

- Transformation:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} sr_{11} & sr_{12} & sr_{13} & t_x \\ sr_{21} & sr_{22} & sr_{23} & t_y \\ sr_{31} & sr_{32} & sr_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{similarity}} = \begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \begin{array}{l} \mathbf{R}^\top \mathbf{R} = \mathbf{I} \\ \det(\mathbf{R}) = 1 \end{array}$$

- Degrees of freedom:
 - 7 (3 rotation, 3 translation, 1 scaling)
- Invariants: angles

Hierarchy of Transformations in 3D (3): Affine Transformations

- Transformation:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{affinity}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad \mathbf{A} \text{ non-singular}$$

- Degrees of freedom:
 - 12 (3 translation, 9 entries of \mathbf{A})
- Invariants: parallel planes, ratios of volumes, plane at infinity

Hierarchy of Transformations in 3D (4)

Projective Transformations

- Transformation:


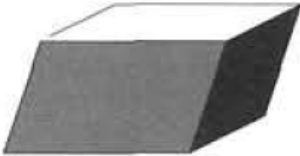


$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ v_1 & v_2 & v_3 & v \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- Concise notation for transformation matrix:

$$\mathbf{T}_{\text{projectivity}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix} \quad \mathbf{A} \text{ non-singular}$$

- Degrees of freedom:
 - 15 (16 arbitrary matrix entries, up to scale)
- Invariants: Intersection points, tangency

Hierarchy of Transformations in 3D – Summary

Projective	15 DOF		$\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{v}^\top & v \end{bmatrix}$
Affine	12 DOF		$\begin{bmatrix} \mathbf{A} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Similarity	7 DOF		$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$
Euclidean	6 DOF		$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix}$

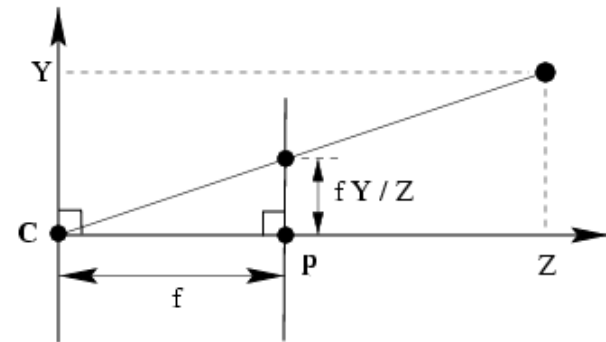
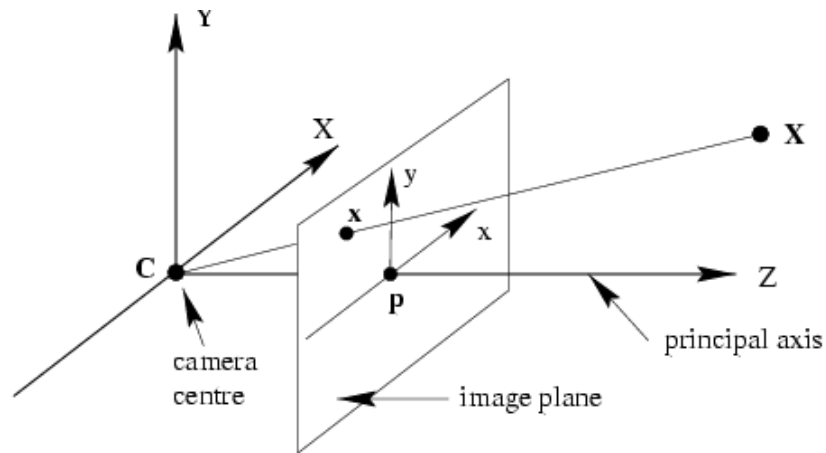
Lecture Outline

Part I: Basics of Transformations

Part II: Basics of Cameras

Pinhole Camera Model

Mapping between 3D world and 2D image by central projection

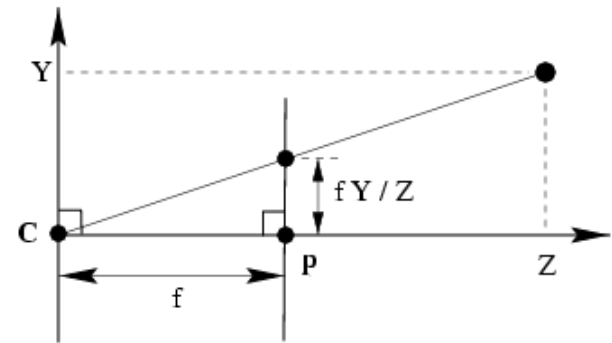
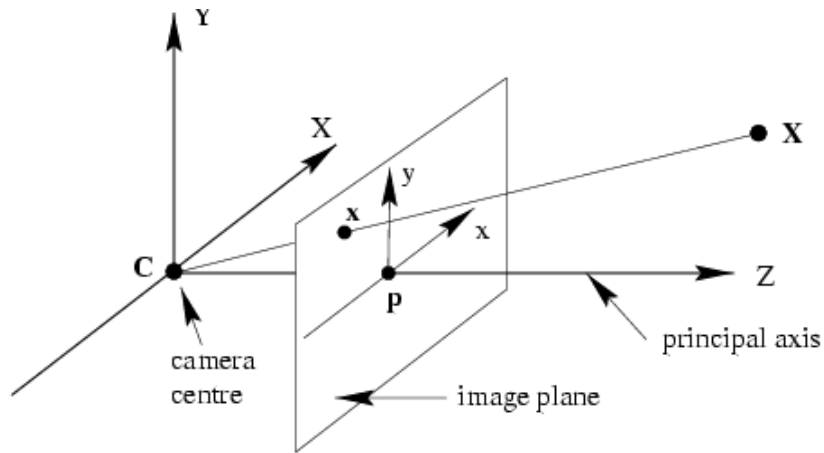


$$(X, Y, Z)^{\top} \mapsto (fX/Z, fY/Z, f)^{\top}$$

$$(X, Y, Z)^{\top} \mapsto (fX/Z, fY/Z)^{\top} = (x, y)^{\top}$$

Pinhole Camera Model

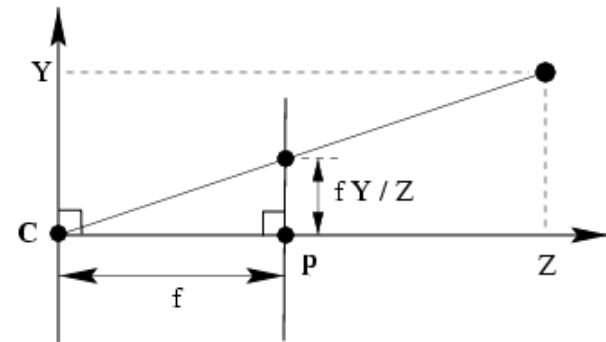
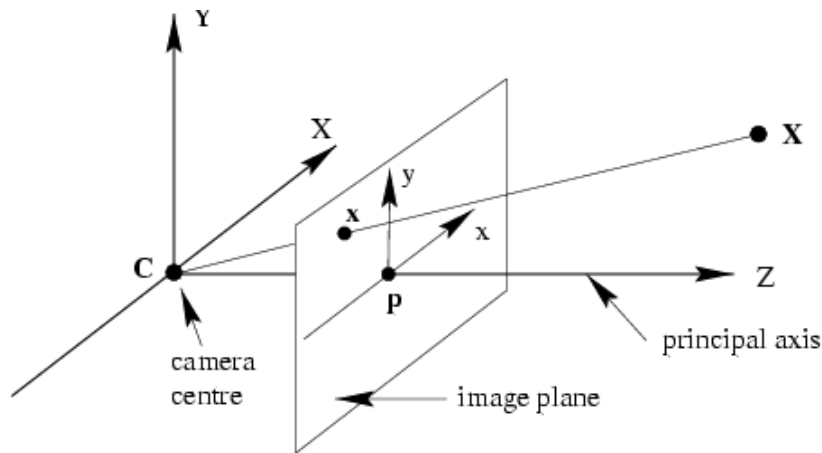
In homogeneous coordinates...



$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 \\ 0 & f \\ 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Pinhole Camera Model

Mapping between 3D world and 2D image by central projection

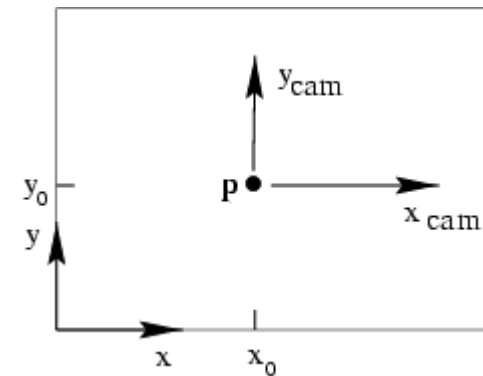
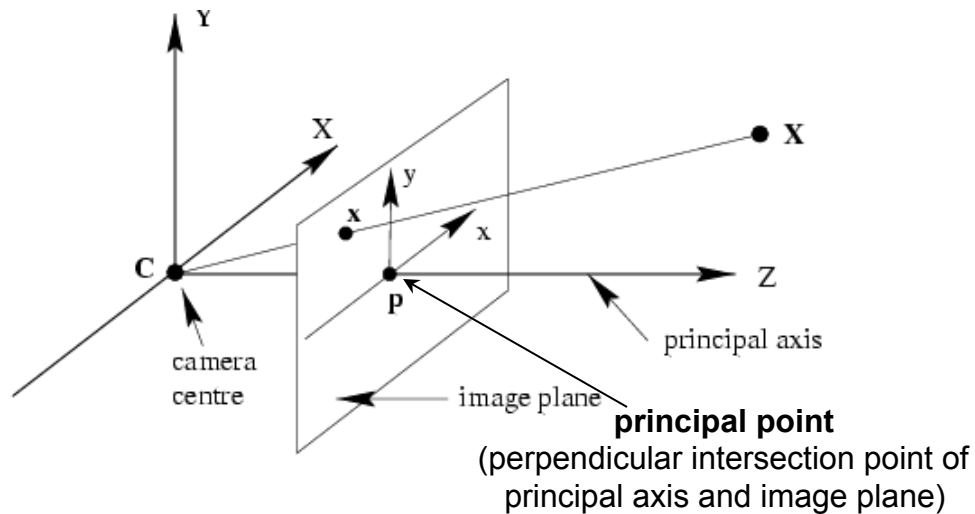


$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ & 1 & 0 \\ & & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{x} = \text{diag}(f, f, 1) [\mathbf{I} | \mathbf{0}] \mathbf{X}$$

Pinhole Camera Model

Origin of image plane is not necessarily at the camera center



$$\mathbf{p} = (p_x, p_y)^\top$$

$$(X, Y, Z) \mapsto (fX/Z + p_x, fY/Z + p_y)$$

Pinhole Camera Model

Origin of image plane is not necessarily at the camera center

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & p_x & 0 \\ & f & p_y \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

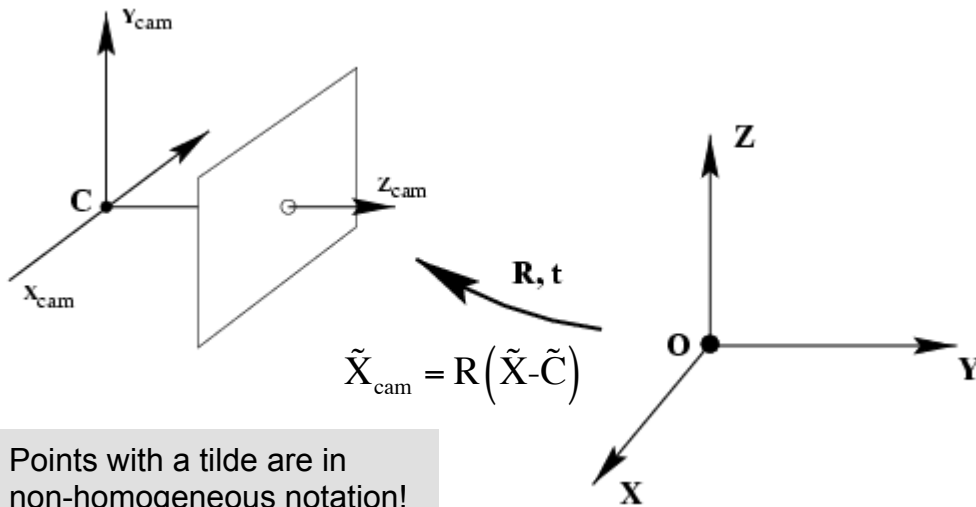
$$\mathbf{x} = \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X}$$

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

intrinsic camera
parameters matrix

Pinhole Camera Model

Camera center is not necessarily at the world coordinate system origin



Points with a tilde are in non-homogeneous notation!

\tilde{C}
camera center
in world coordinates

$$\mathbf{X}_{cam} = \begin{bmatrix} R & -R\tilde{C} \\ 0 & 1 \end{bmatrix} \mathbf{X}$$

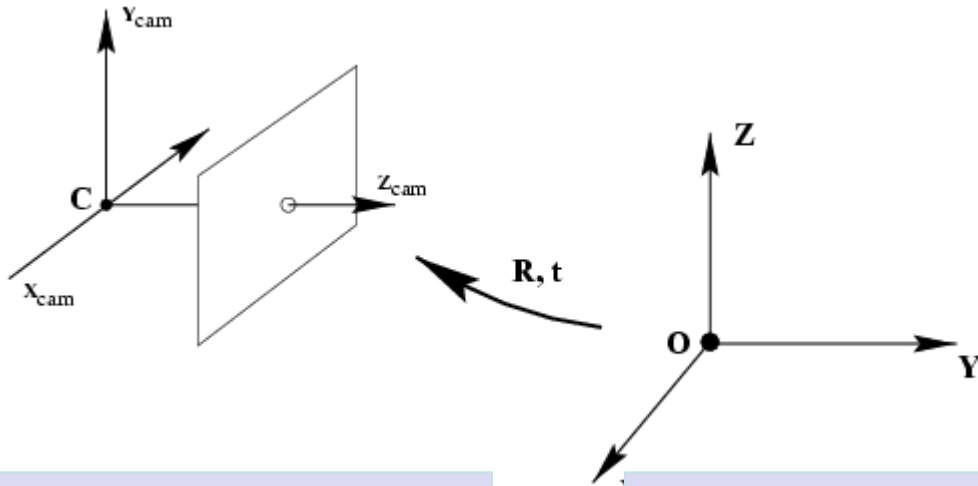
3D point in camera
coordinates

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} I & | & 0 \end{bmatrix} \mathbf{X}_{cam} \quad \text{projection to image plane from camera coordinates}$$

$$\mathbf{x} = \mathbf{K} \mathbf{R} \begin{bmatrix} I & | & -\tilde{C} \end{bmatrix} \mathbf{X} \quad \text{projection to image plane from world coordinates}$$

Pinhole Camera Model

Camera center is not necessarily at the world coordinate system origin



$$\mathbf{x} = \mathbf{KR} \left[\mathbf{I} \parallel -\tilde{\mathbf{C}} \right] \mathbf{X}$$

$$\mathbf{x} = \mathbf{PX}$$

$$\mathbf{P} = \mathbf{KR} \left[\mathbf{I} \parallel -\tilde{\mathbf{C}} \right]$$

projection matrix,
9 DOF

$$\mathbf{K} = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

intrinsic parameters
matrix, 3 DOF

$$\mathbf{R}, \tilde{\mathbf{C}}$$

extrinsic parameters,
each 3 DOF

CCD Camera Model

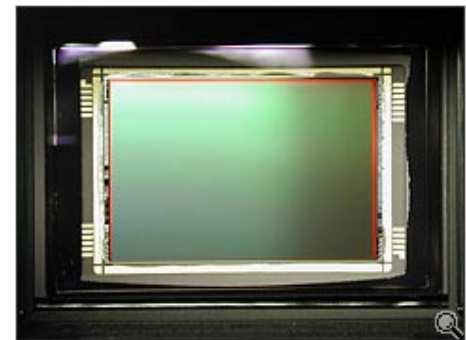
Camera coordinates are mapped to pixels of the CCD chip

m_x, m_y number of pixels per unit distance in camera coordinates

$$\mathbf{K} = \begin{bmatrix} m_x & & & \\ & m_y & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{bmatrix} f & p_x \\ f & p_y \\ & 1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} \alpha_x & & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \quad 4 \text{ DOF}$$

$$\mathbf{P} = \mathbf{KR} \begin{bmatrix} \mathbf{I} & | & -\tilde{\mathbf{C}} \end{bmatrix} \quad 10 \text{ DOF}$$



Finite Projective Camera Model

Both axes in camera coordinates need not have identical scaling

s skew parameter

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \quad 5 \text{ DOF}$$

$$\mathbf{P} = \mathbf{KR} \begin{bmatrix} \mathbf{I} & | & -\tilde{\mathbf{C}} \end{bmatrix} \quad 11 \text{ DOF}$$

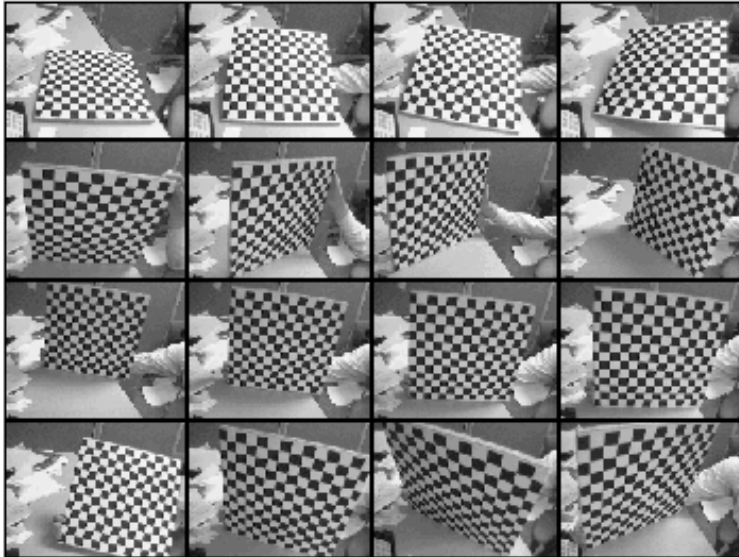
Camera Calibration

- Relating the ideal camera model to the properties of an actual physical device and determining the position and orientation of the camera w.r.t. the world coordinate frame¹.
- Estimating the entries of $\mathbf{K}, \mathbf{R}, \tilde{\mathbf{C}}$
- Camera models with increasing generality:
 - pinhole camera 9 DOF
 - CCD camera 10 DOF
 - finite projective camera 11 DOF
- In addition: lens distortion parameters

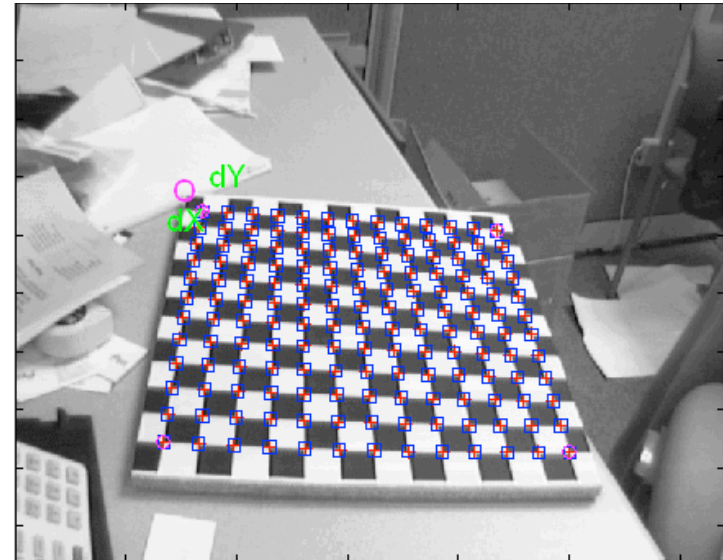
$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix}$$

$$\mathbf{P} = \mathbf{KR} \begin{bmatrix} \mathbf{I} & | & -\tilde{\mathbf{C}} \end{bmatrix}$$

Camera Calibration



Checkerboard pattern captured from multiple views (the checker sizes in the real world are known)



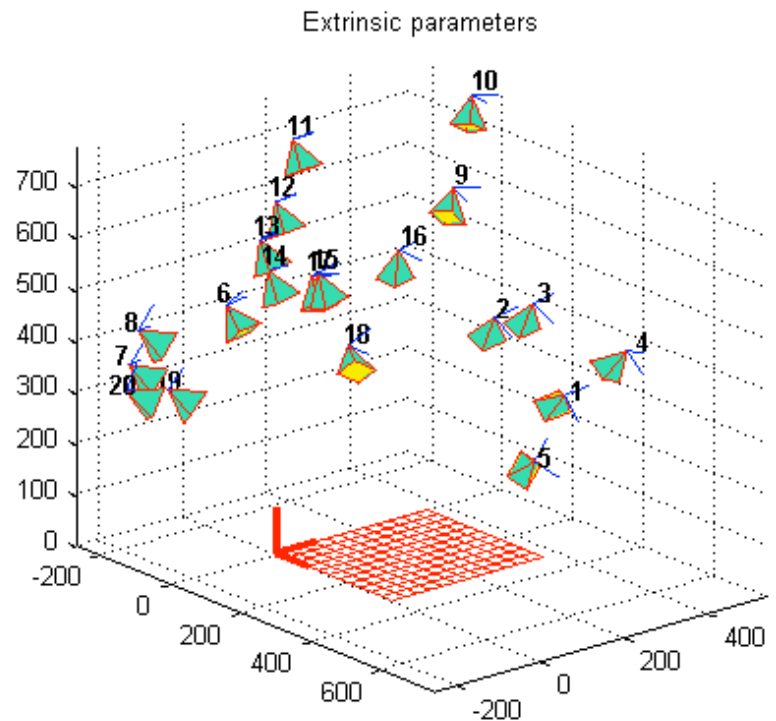
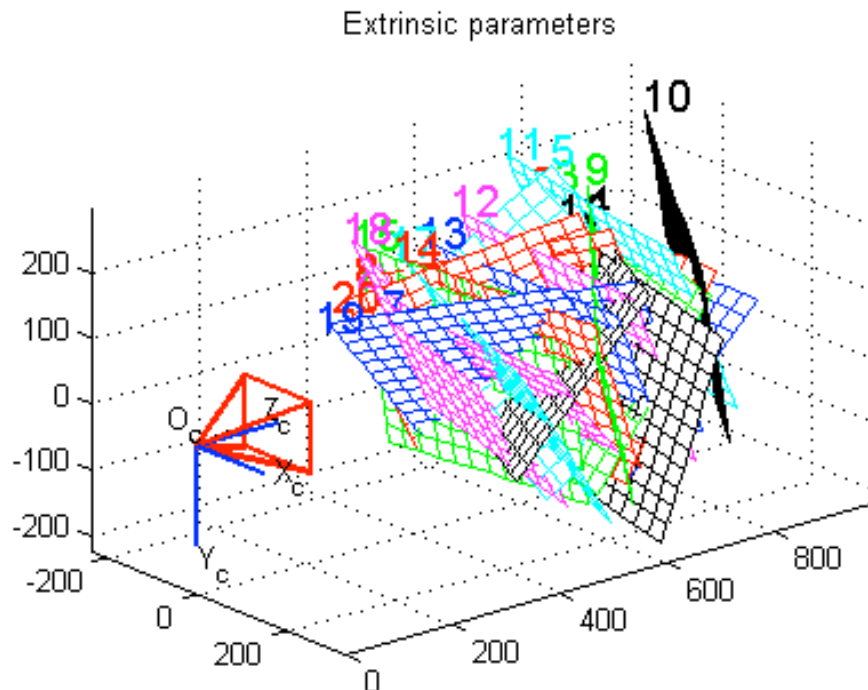
Semi-automatic measurements to find sizes in each of the images

- Linear and non-linear optimization for estimating camera parameters
- MATLAB calibration toolbox: http://www.vision.caltech.edu/bouquetj/calib_doc

Camera Calibration

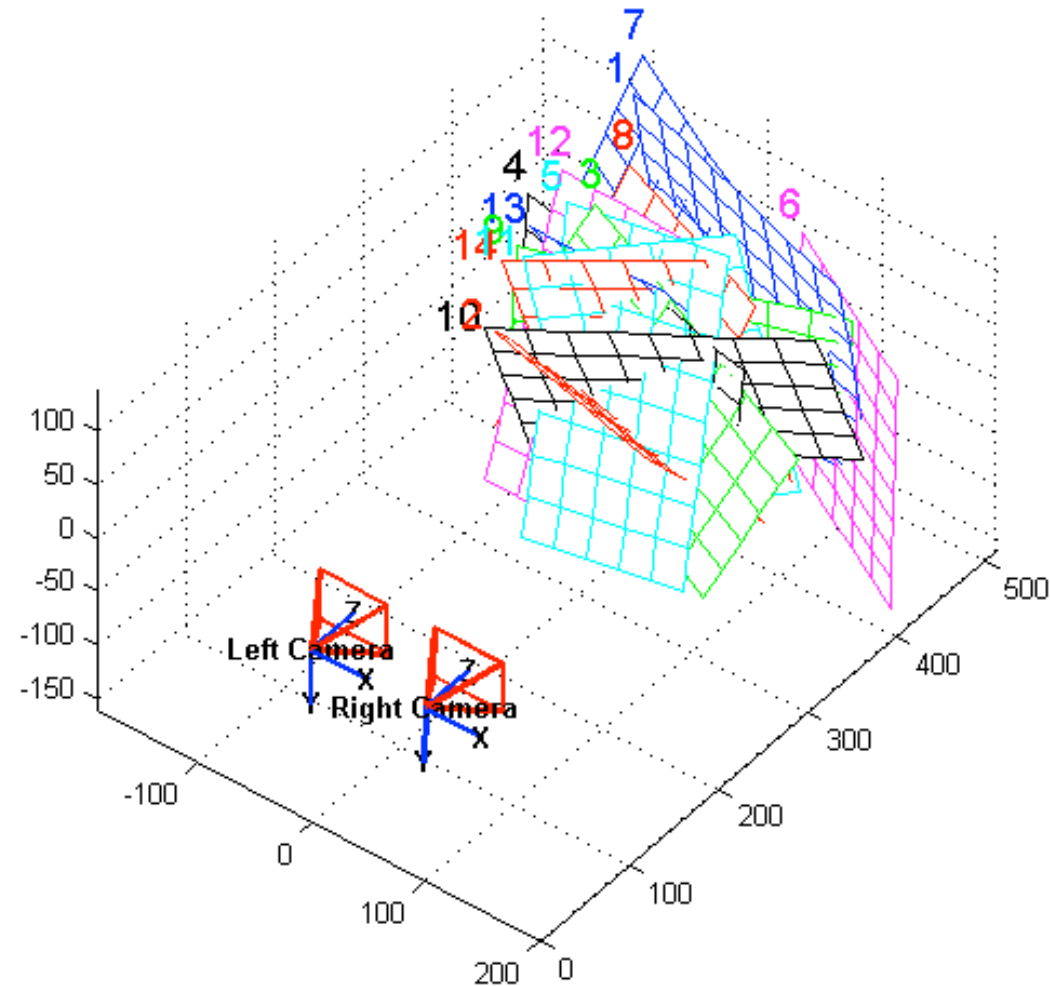
Calibration results after optimization (with uncertainties):

Focal Length: $f_c = [661.67001 \quad 662.82858] \pm [1.17913 \quad 1.26567]$
 Principal point: $cc = [306.09590 \quad 240.78987] \pm [2.38443 \quad 2.17481]$
 Skew: $\alpha_c = [0.00000] \pm [0.00000] \Rightarrow$ angle of pixel axes = 90.00000 ± 0.00000 degrees
 Distortion: $kc = [-0.26425 \quad 0.22645 \quad 0.00020 \quad 0.00023 \quad 0.00000] \pm [0.00934 \quad 0.03826 \quad 0.00052$
 Pixel error: $err = [0.45330 \quad 0.38916]$



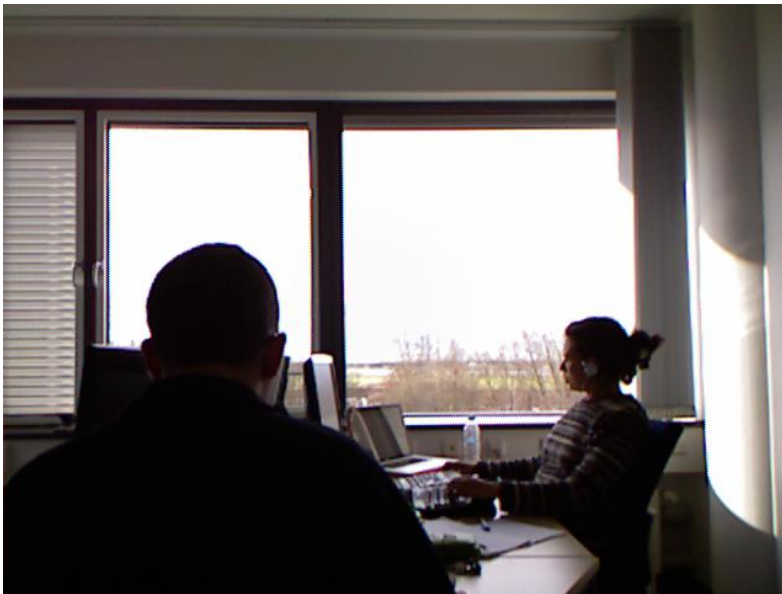
Stereo Calibration

- Find the spatial relation between two cameras observing the same scene (stereo setup)
- For each orientation of the calibration pattern, capture corresponding left and right camera images
- Calibrate each camera separately, as before
- Perform stereo calibration using the result of separate calibration as an input

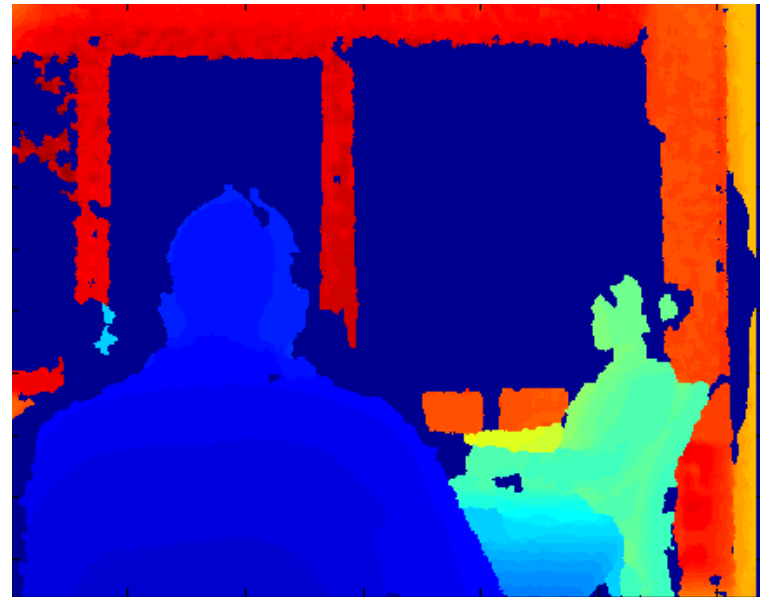


Calibrating the Kinect

- Depth and RGB images do not match exactly, there is an offset
- Stereo calibration for getting both images from the same camera perspective



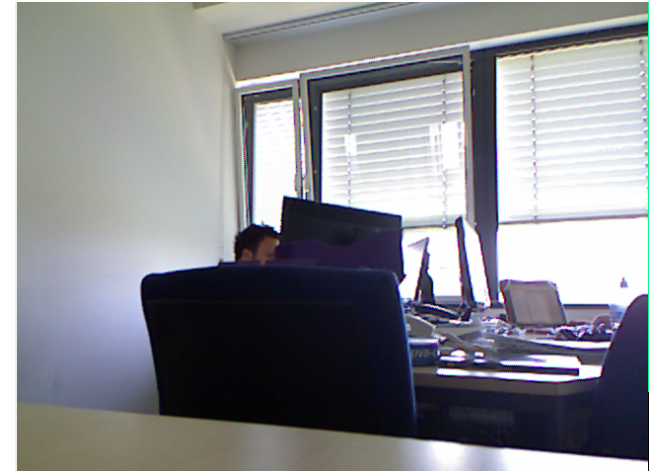
RGB Image



Depth Image

Calibrating the Kinect

- Kinect has two cameras with standard optics
 - RGB camera
 - Infrared (IR) camera
- Infrared image is used internally for computing depth image, but is accessible through the API
- Checkerboard pattern is not visible in depth images, but visible in IR images



RGB Image



Infrared Image

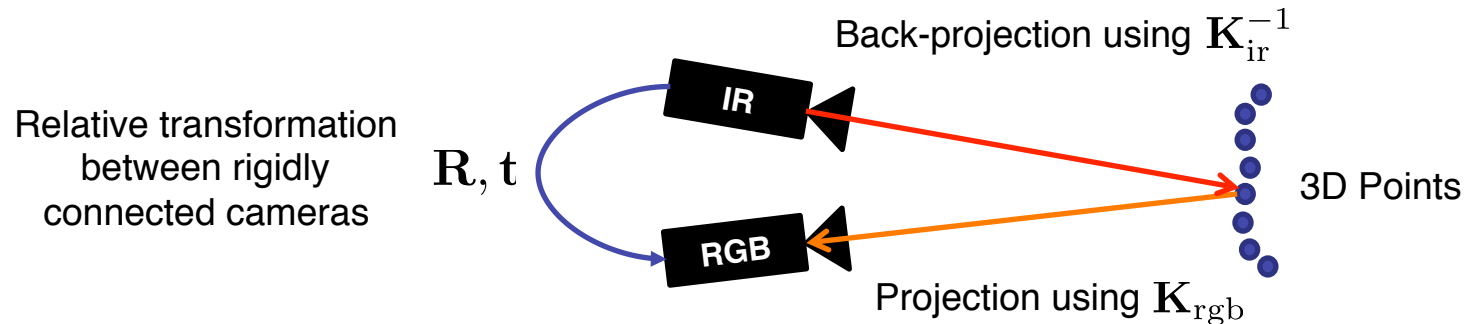
Relating RGB and Depth Images

Stereo calibration gives \mathbf{R} , \mathbf{t} and \mathbf{K}_{rgb} , \mathbf{K}_{ir} .

Step 1: Back-project every 2D point from depth image into 3D space

Step 2: Apply rigid transformation between two cameras to 3D points

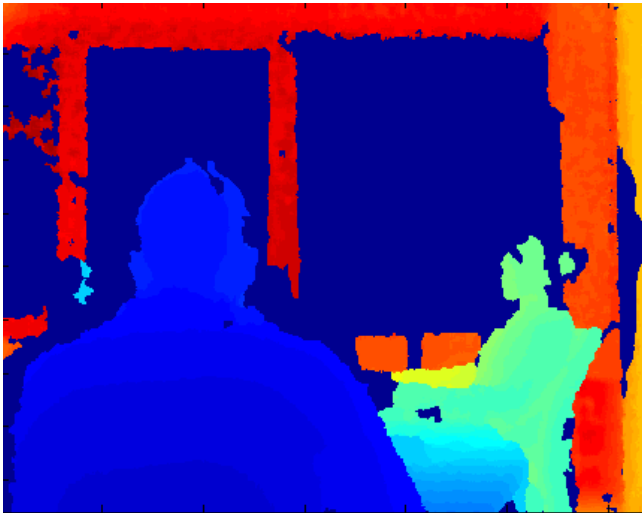
Step 3: Project every transformed 3D point into the RGB image



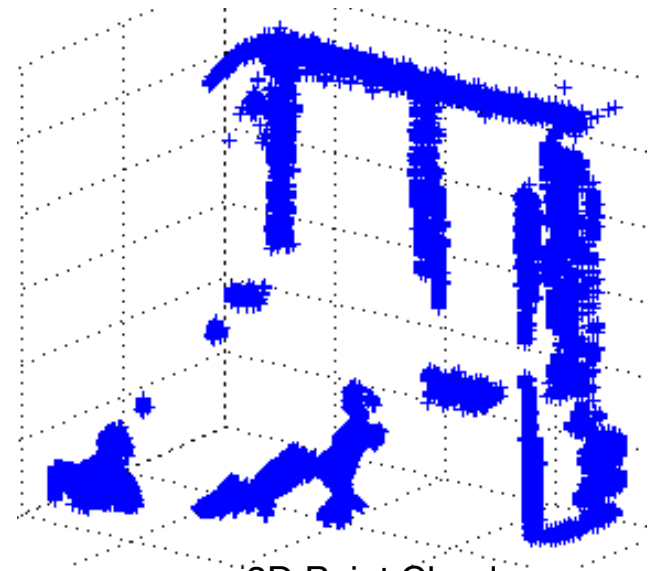
Relating RGB and Depth Images

Step 1: Back-project every 2D point from depth image into 3D space

- ToF data: depth d in meters for every pixel location $\mathbf{x} = (x, y)^\top$
- Desired data: 3D coordinates $\mathbf{X} = (X, Y, Z)^\top$ for every pixel



Depth Image



3D Point Cloud

Relating RGB and Depth Images

Step 1: Back-project every 2D point from depth image into 3D space

Apply inverse of IR camera projection to obtain 3D points:

$$\mathbf{x}_{\text{ir}} = \mathbf{P}_{\text{ir}} \mathbf{X} = \mathbf{K}_{\text{ir}} [\mathbf{I} | \mathbf{0}] \mathbf{X}$$

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \begin{pmatrix} f_x X + c_x Z \\ f_y Y + c_y Z \\ Z \end{pmatrix}$$

$$X = \frac{(x - c_x)Z}{f_x} \quad Y = \frac{(y - c_y)Z}{f_y}$$

Inverse relation for X and Y

Relating RGB and Depth Images

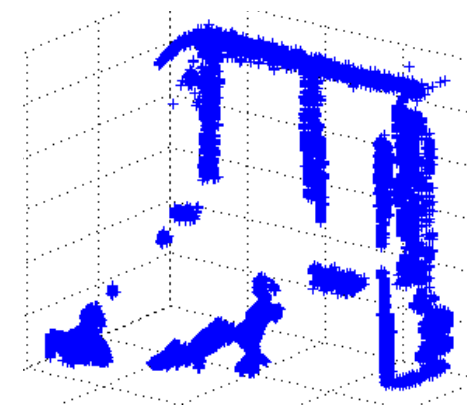
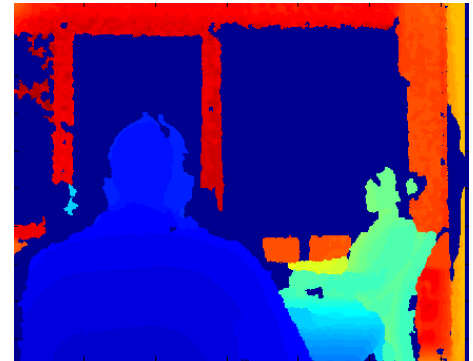
Step 1: Back-project every 2D point from depth image into 3D space

For every depth pixel at location $\mathbf{x} = (x, y)^\top$ with depth d , compute a 3D point $\mathbf{X} = (X, Y, Z)^\top$ with coordinates:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} \frac{(x - c_x)d}{f_x} \\ \frac{(y - c_y)d}{f_y} \\ d \end{pmatrix}$$

c_x, c_y, f_x, f_y

Intrinsics of the IR (depth) camera



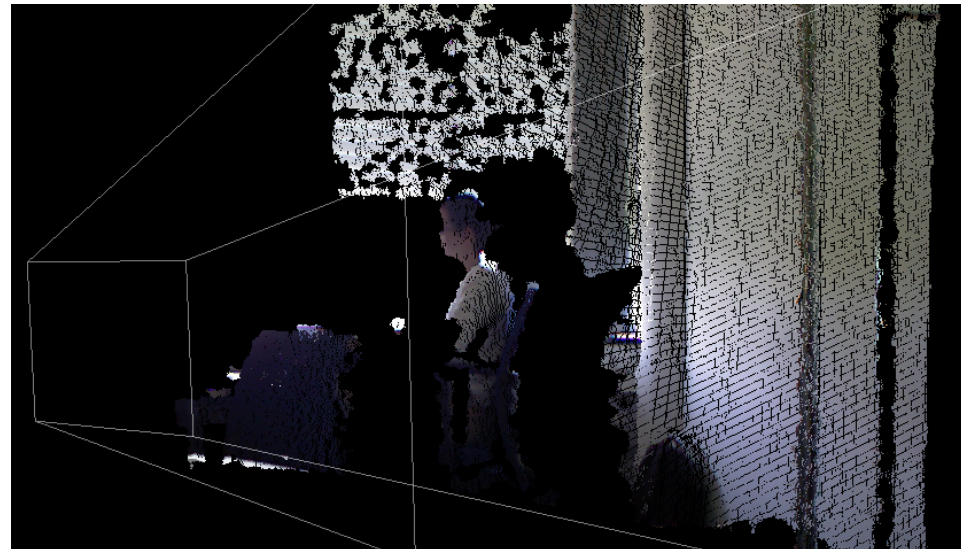
Relating RGB and Depth Images

Step 2: Apply rigid transformation between two cameras to 3D points

Step 3: Project every transformed 3D point into the RGB image

$$\mathbf{x}_{\text{rgb}} = \mathbf{P}_{\text{rgb}} \mathbf{X} = \mathbf{K}_{\text{rgb}} [\mathbf{R} | \mathbf{t}] \mathbf{X}$$

- \mathbf{X}_{rgb} are the coordinates of the projection of \mathbf{X} into the RGB image
- look up color corresponding to a given 3D point



Assignment 3

- Extend your application such that it can read out Kinect IR images
- Extend your application for saving IR and RGB images to bitmap files
- Familiarize yourself with the Matlab Camera Calibration Toolbox
http://www.vision.caltech.edu/bouguetj/calib_doc/
 - Read the examples (especially examples 1 and 5)
 - Download the toolbox
 - Try out the examples
- Perform stereo-calibration for your Kinect device
 - Note that you cannot extract IR and RGB images simultaneously...

Outlook Assignment 4:

- Implement a 3D point cloud visualization for kinect depth data
- Add a feature to color the 3D points with the colors from the RGB camera