C18 Computer Vision

Victor Adrian Prisacariu
http://www.robots.ox.ac.uk/~victor

Michaelmas 2016
Course Content

• VP: Intro, basic image features, basic multiview geometry ...
  – Introduction: imaging geometry, camera calibration.
  – Salient feature detection: points, edges and SIFTs.
  – Recovering 3D from two images I: epipolar geometry.
  – Recovering 3D from two images II: stereo correspondences, triangulation, ego-motion.

Slides at http://www.robots.ox.ac.uk/~victor/Courses/C18
Lots borrowed from David Murray + AV C18.

• AV: Neural networks and applications ...
  – Small baseline correspondences and optic flow.
  – Large scale image retrieval.
  – Object detection and recognition.
Useful Texts

- **Multiple View Geometry in Computer Vision**  
  *Richard Hartley, Andrew Zisserman*

- **Computer Vision: A Modern Approach**  
  *David Forsyth, Jean Ponce*  

- **3-Dimensional Computer Vision: A Geometric Viewpoint**  
  *Olivier Faugeras*
Computer Vision: This time...

1. Introduction: imaging geometry, camera calibration.
   1. Introduction.
   2. The perspective camera as a geometric device.
   3. Perspective using homogeneous coordinates.
   4. Calibration the elements of the perspective model.
2. Salient feature detection: points, edges and SIFTs.
3. Recovering 3D from two images I: epipolar geometry.
4. Recovering 3D from two images II: stereo correspondences, triangulation, ego-motion.
1.1 Introduction

Aim in geometric **computational vision** is to take a number of 2D images, and obtain an understanding of the 3D environment; what is in it; and how it evolves over time.

What do we have here ...?
Wrong! It’s a very hard big data problem ...

• From the **hardware engineering** perspective...
  – The raw throughput is unsettlingly large:
  – Colour stereo pair at 30Hz -> 100s MB/s.
  – Now multiply by non-trivial processing cost per byte ...
  – Image collections are huge.
  Certainly challenging, but no longer frightening.

• From **applied maths** perspective ...
  – Scene info is made highly implicit or lost by reprojection.
  – Inverse mappings 2D -> 3D ill-posed, ill-conditioned.
  – Can only be solved by introducing **constraints**:
    • about the way the world *actually* works; or
    • about the way we *expect* it to work.
  We now know about these. No longer frightening.
Wrong! It’s a very hard big data problem ...

From the Information-Engineering / AI perspective ...

- Images have uneven information content both absolutely and contextually.
- Computational visual semantics: what does visual stuff mean exactly?
- If we are under time pressure, what is the important visual stuff right now?

Still a massive challenge – if we want genuine autonomy.
Natural vision is a hard problem ...

• But **we see effortlessly!** Yep, spot on if **one neglects:**
  – the $10^{11}$ neurons involved.
  – aeons of evolution generating hardwire priors $P(I)$.
  – that we sleep with eyes shut, and avert gaze when thinking.

• Very **hard work for our brains** – does machine vision have a hope?
  – Perhaps building a general vision system is a flawed concept.
  – Evidence that the human visual system is a bag of tricks – specialized processing for specialized tasks.
  – Perhaps we should expect no more of computer vision?

• However, the **h.v.s. does give us a convincing magic show.**
  – Each trick flows seamlessly into the next. Do we have an inkling how that will be achieved?
So why bother? What are the advantages?

• From a **sensor standpoint** ...
  
  – Vision is a passive sensor (unlike sonar, radar, lasers).
  – Wide range of depths, overlaps and complements other sensors.
  – Wide diversity of methods to recover 3D information: so vision has process redundancy.
  – Images provide data redundancy.

• From a **natural communications standpoint** ...
  
  – The world is awash with information-rich photons.
  – Because we have eyes, vision provides a natural language of communication. If we want robots/man-machine-interfaces to act and interact in our environments, they have to speak that language.
Although human and computer vision might be bags of tricks, it is useful to place the tricks **within larger processing paradigms**.

For example:

a) Data-driven, bottom-up processing.

b) Model-driven, top-down, generative processing.

c) Dynamic Vision (mixes bottom-up with top-down feedback).

d) Active Vision (task oriented).

e) Data-driven discriminative approach (machine learning).

These are neither all-embracing nor exclusive.
(a) Data-driven, bottom-up processing

- Image processing produces map of salient 2D features.

- Features input into a range of shape from X processes whose output was the 2.5D sketch.

- Only in the last stage we get a fully 3D object-centered description.
(b) Model-driven, and (c) Dynamic vision

• Model-driven, top-down, generative processing:
  – a model of the scene is assumed known.
  – Supply a pose for the object relative to the camera, and use projection to predict where salient features should be found in the image space.
  – Search for the features, and refine the pose by minimizing the observed deviation.

• Dynamic vision: mixes bottom-up/top-down by introducing feedback.
(d) Active Vision

- Introduces **task-oriented sensing-perception-action loops**:  
  - Visual data needs only be “good enough” to drive the particular action.

- No need to build and maintain an overarching representation of the surroundings.

- Computational resources focused where they are needed.
(e) Data-driven approach

• The aim is to learn a description of the transformation between input and output using exemplars.

• Geometry is not forgotten, but implicit learned representation are favored.
1.2 The perspective camera as a geometric device
This is (a picture of) my cat

\[ x = \begin{bmatrix} 295 \\ 308 \end{bmatrix} \]

Cat nose
My cat lives in a 3D world

\[
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \quad \quad \quad \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}
\]
Going from $X$ in 3D to $x$ in 2D

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$X = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}$$

film/sensor  cat

blurry 😞
Going from $X$ in 3D to $x$ in 2D

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Blur reduced, looks good 😊
Pinhole Camera

All rays pass through the **center of projection** (a single point). Image forms on the image plane.
Pinhole Camera

The 3D point \( \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \) is imaged into \( \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \) as:

\[
\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} f \frac{X_1}{X_3} \\ f \frac{X_2}{X_3} \end{bmatrix}
\]

Image plane

- \( f \) – focal length
- \( o \) – camera origin
- \( p \) – principal point
1.2 The perspective camera as a geometric device
Homogeneous coordinates

• The projection $\mathbf{x} = \frac{f \mathbf{X}}{X_3}$ is non-linear 😞.

• Can be made linear using **homogeneous coordinates**
  – involves representing the image and scene in higher dimensional space.

• Limiting cases – e.g. vanishing points – are handled better.

• Homogeneous coordinates allow for transformations to be concatenated more easily.
3D Euclidean transforms: inhomogeneous coordinates

- My cat moves through 3D space.

- The movement of the tip of the nose can be described using an Euclidean transform:

\[
X_{3\times1}' = R_{3\times3}X_{3\times1} + t_{3\times1}
\]

\[
\begin{cases}
\text{rotation} \\
\text{translation}
\end{cases}
\]
3D Euclidean transforms: **inhomogeneous** coordinates

- Euclidean transform: $X'_{3\times 1} = R_{3\times 3}X_{3\times 1} + t_{3\times 1}$

- Concatenation of successive transform is a mess!

$$
X_1 = R_1X + t_1 \\
X_2 = R_2X_1 + t_2 \\
= R_2(R_1X + t_1) + t_2 = (R_2R_1)X + (R_2t_1 + t_2).
$$
3D Euclidean transforms: **homogeneous coordinates**

- We replace the 3D points \( \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \) with a **four vector** \( \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \).

- The Euclidean transform becomes:

\[
\begin{bmatrix} X' \\ Y' \\ Z' \\ 1 \end{bmatrix} = E \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R \\ 0^T \\ t \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}
\]

- Transformations can now be concatenated by matrix multiplication:

\[
\begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} = E_{10} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \quad \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = E_{21} \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ 1 \end{bmatrix} \rightarrow \begin{bmatrix} X_2 \\ Y_2 \\ Z_2 \\ 1 \end{bmatrix} = E_{21} E_{10} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}
\]
Homogeneous coordinates – definition in $R^3$

• $\mathbf{X} = (X, Y, Z)^T$ is represented in homogeneous coordinates by any 4-vector

\[
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix}
\]

such that $X = X_1/X_4, Y = X_2/X_4,$ and $Z = X_3/X_4$.

• So the following homogeneous vectors represent the same point, for any $\lambda \neq 0$:

\[
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix} \quad \text{and} \quad \lambda \begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix}
\]

• E.g. $(2,3,5,1)^T$ is the same as $(-3,-4.5,-7.5,-1.5)^T$ and both represent the same inhomogeneous point $(2,3,5)^T$.
Homogeneous coordinates – definition in $R^2$

- $\mathbf{x} = (x, y)^T$ is represented in homogeneous coordinates by any 3-vector
  \[
  \begin{bmatrix}
  x_1 \\
  x_2 \\
  x_3
  \end{bmatrix}
  \]
  such that $x = x_1/x_3$, $y = x_2/x_3$.

- E.g. $(1,2,3)^T$ is the same as $(3,6,9)^T$ and both represent the same inhomogeneous point $(0.33, 0.66)^T$
1. Convert the inhomogeneous point to an homogeneous vector:

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} \rightarrow \begin{bmatrix}
X \\
Y \\
Z \\
1
\end{bmatrix}
\]

2. Apply a $4 \times 4$ transform.

3. Dehomogenize the resulting vector:

\[
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix} \rightarrow \begin{bmatrix}
X_1/X_4 \\
X_2/X_4 \\
X_3/X_4
\end{bmatrix}
\]
A projective transformation is a linear transformation on homogeneous 4-vectors represented by a non-singular 4x4 matrix.

\[
\begin{bmatrix}
X'_1 \\
X'_2 \\
X'_3 \\
X'_4
\end{bmatrix} =
\begin{bmatrix}
p_{11} & p_{12} & p_{12} & p_{14} \\
p_{21} & p_{22} & p_{23} & p_{24} \\
p_{31} & p_{32} & p_{33} & p_{34} \\
p_{41} & p_{42} & p_{43} & p_{44}
\end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix}
\]

The effect on the homogenous points is that the original and transformed points are linked through a projection center.

The 4x4 matrix is defined up to scale, and so has 15 degrees of freedom.
More 3D-3D and 2D-2D Transforms

Projective (15 dof):
\[
\begin{bmatrix}
X'_1 \\
X'_2 \\
X'_3 \\
X'_4
\end{bmatrix} = \begin{bmatrix} P_{4\times4} \end{bmatrix}
\begin{bmatrix}
X_1 \\
X_2 \\
X_3 \\
X_4
\end{bmatrix}
\]

Affine (12 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2 \\
x'_3
\end{bmatrix} = \begin{bmatrix} A_{3\times3} & t_3 \\ 0^T & 1 \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\]

Similarity (7 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2 \\
x'_3
\end{bmatrix} = \begin{bmatrix} SR_{3\times3} & t_3 \\ 0^T & 1 \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\]

Euclidean (6 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2 \\
x'_3
\end{bmatrix} = \begin{bmatrix} R_{3\times3} & t_3 \\ 0^T & 1 \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\]

Projective (aka Homography, 8 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2 \\
x'_3
\end{bmatrix} = \begin{bmatrix} H_{3\times3} \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3
\end{bmatrix}
\]

Affine (6 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2 \\
x'_3
\end{bmatrix} = \begin{bmatrix} A_{2\times2} & t_2 \\ 0^T & 1 \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
\]

Similarity (5 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2 \\
x'_3
\end{bmatrix} = \begin{bmatrix} SR_{2\times2} & t_2 \\ 0^T & 1 \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
\]

Euclidean (4 dof):
\[
\begin{bmatrix}
x'_1 \\
x'_2
\end{bmatrix} = \begin{bmatrix} R_{2\times2} & t_2 \\ 0^T & 1 \end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2
\end{bmatrix}
\]
2D-2D Transform Examples

\[
\begin{bmatrix}
\cos \theta & -\sin \theta & t_x \\
\sin \theta & \cos \theta & t_y \\
0 & 0 & 1
\end{bmatrix} \quad \begin{bmatrix}
\cos \theta & -\sin \theta & t_x \\
\sin \theta & \cos \theta & t_y \\
0 & 0 & 1
\end{bmatrix} \quad \begin{bmatrix}
a_{11} & a_{12} & t_x \\
a_{21} & a_{22} & t_y \\
0 & 0 & 1
\end{bmatrix} \quad \begin{bmatrix}
h_{11} & h_{12} & h_{12} \\
h_{21} & h_{22} & h_{23} \\
h_{31} & h_{32} & h_{33}
\end{bmatrix}
\]

Euclidean
3 DoF

Similarity
4 DoF

Affine
6 DoF

Projective
8 DoF
Perspective 3D-2D Transforms

• Similar to a 3D-3D projective transform, but **constrain the transformed point to a plane** \( z = f \).

\[
z = f \rightarrow \mathbf{X}_{\text{image}} = \begin{bmatrix} x_1 \\ x_2 \\ f \\ 1 \end{bmatrix}
\]

• Because \( z = f \) is fixed, we can write:

\[
\lambda \begin{bmatrix} x_1 \\ x_2 \\ f \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ fp_{31} & fp_{32} & fp_{33} & fp_{34} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}
\]

• The 3\(^{rd}\) row is redundant, so:

\[
\lambda \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{12} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix} = P_{3\times4} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}
\]

\(P_{3\times4}\) is the **projection matrix** and this is a **perspective transform**
1.3 Perspective using homogeneous coordinates

\[
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \quad \overset{\lambda}{\rightarrow} \quad \mathbf{X} = \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix}
\]

\[
[ \begin{array}{c} x_1 \\ x_2 \\ 1 \end{array} ] = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}
\]

\[
\begin{align*}
\lambda x_1 &= f X_1 \\
\lambda x_2 &= f X_2 \\
\lambda &= X_3
\end{align*}
\]

\[
x_1 = f \frac{X_1}{X_3} \\
x_2 = f \frac{X_2}{X_3}
\]

\[
\mathbf{x} \quad \mathbf{X}
\]
Perspective using homogeneous coordinates

\[
\lambda \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{bmatrix}
\]
Perspective using homogeneous coordinates

• It is useful to split up the overall projection matrix into three parts:
  1. a part that depends on the internals of the camera
  2. a vanilla projection matrix
  3. a Euclidean transformation between the world and camera frames.

• We first assume the scene and world are aligned with the camera coords, so that the extrinsic camera matrix is identity and get:

<table>
<thead>
<tr>
<th>Image Point</th>
<th>Camera’s Intrinsic Calibration</th>
<th>Projection matrix (vanilla)</th>
<th>Camera’s Extrinsic Calibration</th>
<th>World Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \lambda \begin{bmatrix} x \ y \ 1 \end{bmatrix} )</td>
<td>[ \begin{bmatrix} f &amp; 0 &amp; 0 \ 0 &amp; f &amp; 0 \ 0 &amp; 0 &amp; 1 \end{bmatrix} ]</td>
<td>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix} ]</td>
<td>[ \begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix} ]</td>
<td>( x )</td>
</tr>
</tbody>
</table>
Perspective using homogeneous coordinates

- Now let’s make things more general:
  - Insert a rotation $R$ and translation $t$ between world and camera coordinates.
  - Insert some extra term in the intrinsic calibration matrix.

<table>
<thead>
<tr>
<th>Image Point</th>
<th>Camera’s Intrinsic Calibration</th>
<th>Projection matrix (vanilla)</th>
<th>Camera’s Extrinsic Calibration</th>
<th>World Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda \begin{bmatrix} x \ y \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} f &amp; sf &amp; u_0 \ 0 &amp; yf &amp; v_0 \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$\begin{bmatrix} r_{11} &amp; r_{12} &amp; r_{13} &amp; t_1 \ r_{21} &amp; r_{22} &amp; r_{23} &amp; t_2 \ r_{31} &amp; r_{32} &amp; r_{33} &amp; t_3 \ 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$X$</td>
</tr>
</tbody>
</table>
The **camera pose** (extrinsic parameters)

The camera’s extrinsic calibration is just the rotation $R$ and translation $t$ that take points from the world frame to the camera frame.

$$
\begin{bmatrix}
X_c^1
\end{bmatrix} =
\begin{bmatrix}
R & t
\end{bmatrix}
\begin{bmatrix}
X_w^1
\end{bmatrix}
$$
Building $R$

- $R$ captures rotation and can be built from various types of rotation representations (Euler angles, quaternions, etc.).

- **Euler angles** capture the angles of rotation axis using 3 parameters, one for each axis.

\[
X' = R_z X^W = \begin{bmatrix} \cos \theta_z & \sin \theta_z & 0 \\ -\sin \theta_z & \cos \theta_z & 0 \\ 0 & 0 & 1 \end{bmatrix} X^W
\]

\[
X'' = R_y X' = \begin{bmatrix} \cos \theta_y & 0 & -\sin \theta_y \\ 0 & 1 & 0 \\ \sin \theta_y & 0 & \cos \theta_y \end{bmatrix} X'
\]

\[
X^A = R_x X'' = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta_x & \pm \sin \theta_x \\ 0 & \mp \sin \theta_x & \cos \theta_x \end{bmatrix} X''
\]

\[
R^{CW} = R_x R_y R_z
\]

Order matters!
Inverting the transform

\[
\begin{bmatrix}
R^{CW} & t^{CW} \\
0^T & 1
\end{bmatrix}^{-1} = \begin{bmatrix}
R^{WC} & t^{WC} \\
0^T & 1
\end{bmatrix}
\]

For rotation:
\[
R^{WC} = [R^{CW}]^{-1} = [R^{CW}]^T
\]

For translation:
\[
t^{WC} = -t^{CW} = -R^{WC}t^{CW}
\]
The **intrinsic** calibration parameters

Describe **hardware properties** of real cameras:

- The image plane might be skewed.
- The central axis of the lens might not line up with the optical axis.
- The light gathering elements might not be square.
- Lens distortion.

\[
K = \begin{bmatrix}
    f & 0 & 0 \\
    0 & \gamma f & 0 \\
    0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & \frac{u_0}{f} \\
    0 & 1 & \frac{v_0}{\gamma f} \\
    0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    1 & s & 0 \\
    0 & 1 & 0 \\
    0 & 0 & 1 \\
\end{bmatrix}
= \begin{bmatrix}
    f & sf & u_0 \\
    0 & \gamma f & v_0 \\
    0 & 0 & 1 \\
\end{bmatrix}
\]

- different scaling on x and y
- \(\gamma\) is the aspect ratio.
- Origin offset, \((u_0, v_0)\) is the principal point.
- \(s\) accounts for skew
Summary of steps from Scene to Image

1. Move scene point \((X^W, 1)^T\) into camera coordinate by \(4 \times 4\) extrinsic Euclidean transformation:

\[
\begin{bmatrix}
X^C \\
1
\end{bmatrix} = 
\begin{bmatrix}
R & t \\
0^T & 1
\end{bmatrix}
\begin{bmatrix}
X^W \\
1
\end{bmatrix}
\]

2. Project into ideal camera via a vanilla perspective transformation:

\[
\begin{bmatrix}
x' \\
1
\end{bmatrix} =
\begin{bmatrix}
I & 0
\end{bmatrix}
\begin{bmatrix}
X^C \\
1
\end{bmatrix}
\]

3. Map the ideal image into the real image using intrinsic matrix:

\[
\begin{bmatrix}
x \\
1
\end{bmatrix} = K
\begin{bmatrix}
x' \\
1
\end{bmatrix}
\]
1.4 Camera Calibration

• The process that finds $K$, and accounts for the internal physical characteristics of the camera.

• (Usually) done once per camera.

• There are a variety of method for self-calibration, auto-calibration or pre-calibration.

• We will gloss over pre-calibration, using a specially made “known” visual scene.
What is camera calibration?

<table>
<thead>
<tr>
<th>Image Point</th>
<th>Camera’s Intrinsic Calibration</th>
<th>Projection matrix (vanilla)</th>
<th>Camera’s Extrinsic Calibration</th>
<th>World Point</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda \begin{bmatrix} x \ y \ 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} f &amp; sf &amp; u_0 \ 0 &amp; yf &amp; v_0 \ 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$\begin{bmatrix} 1 &amp; 0 &amp; 0 &amp; 0 \ 0 &amp; 1 &amp; 0 &amp; 0 \ 0 &amp; 0 &amp; 1 &amp; 0 \end{bmatrix}$</td>
<td>$\begin{bmatrix} r_{11} &amp; r_{12} &amp; r_{13} &amp; t_1 \ r_{21} &amp; r_{22} &amp; r_{23} &amp; t_2 \ r_{31} &amp; r_{32} &amp; r_{33} &amp; t_3 \ 0 &amp; 0 &amp; 0 &amp; 1 \end{bmatrix}$</td>
<td>$X$</td>
</tr>
</tbody>
</table>

$$P_{3\times4} = K[R|t]$$

Camera calibration: recover $K$
1. Recover overall projection matrix $P_{3 \times 4}$.
   - Assume target with at least 6 known scene points.
   - Build and solve system of (at least) 12 equations.

\[
\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}
\]

\[
\lambda_i = p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34}
\]
\[
(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34})x_i = p_{11}X_i + p_{12}Y_i + p_{13}Z_i + p_{14}
\]
\[
(p_{31}X_i + p_{32}Y_i + p_{33}Z_i + p_{34})y_i = p_{21}X_i + p_{22}Y_i + p_{23}Z_i + p_{24}
\]

\[
\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & 0 & -X_ix_i & -Y_ix_i & -Z_ix_i & -x_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -X_iy_i & -Y_iy_i & -Z_iy_i & -y_i \end{bmatrix} p = \begin{bmatrix} 0 \\ 0 \end{bmatrix}
\]

where $p$ contains the unknowns.
1. Recover overall projection matrix \( P_{3\times4} \).
   • Assume target with at least 6 known scene points.
   • Build and solve system of (at least) 12 equations.

2. Construct \( P_{\text{LEFT}} = KR \) from leftmost 3x3 block of \( P = K[R|t] \).
1. Recover overall projection matrix $P_{3\times4}$.
   - Assume target with at least 6 known scene points.
   - Build and solve system of (at least) 12 equations.

2. Construct $P_{\text{LEFT}} = KR$ from leftmost 3x3 block of $P = K[R|t]$.

3. Invert $P_{\text{LEFT}}$ so $P_{\text{LEFT}}^{-1} = R^{-1}K^{-1}$.

4. Decompose $P_{\text{LEFT}}^{-1}$ using QR decomposition into $R$ and $K$. 
Camera Calibration – Math Part

1. Recover overall projection matrix $P_{3\times4}$.
   • Assume target with at least 6 known scene points.
   • Build and solve system of (at least) 12 equations.

2. Construct $P_{\text{LEFT}} = KR$ from leftmost 3x3 block of $P = K[R|t]$.

3. Invert $P_{\text{LEFT}}$ so $P_{\text{LEFT}}^{-1} = R^{-1}K^{-1}$.

4. Decompose $P_{\text{LEFT}}^{-1}$ using QR decomposition into $R$ and $K$.

5. Normalise $K$ (as scale of $P$ was unknown).

6. Recover $t = K^{-1}[p_{14} \quad p_{24} \quad p_{34}]^T$. 
Camera Calibration – Example Algorithm

1. Compute corner features
2. Hand match 6 (or more) image points to world points
3. Compute Calibration K, R, t
4. While unmatched points exist
   4.1 Project all known world points into image using current calibration
   4.2 Match to measurements using threshold on image distance
   4.3 Recompute calibration
   4.4 Relax threshold if match confusion unlikely
Radial Distortion

• So far, we have figured out the transformations that turn our camera into a notional camera with the world and the camera coordinates aligned and an “ideal” image plane.

• One often has to correct for the other optical distortions and aberrations. Radial distortion is the most common – see the Q sheet.

• Correction for this distortion is applied before carrying out calibration.
Practical Camera Calibration

Matlab Calibration Toolkit

http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/example.html
Summary of Lecture 1

In this lecture we have:

• Introduced the aims of computer vision, and some paradigms.
• Explored linear transformations and introduced homogeneous coordinates.
• Defined perspective projection from a scene, and saw that it could be made linear using homogeneous coordinates.
• Discussed how to pre-calibrate a camera using image of six or more known scene points.