Imprecision of Calibrated Cameras

Problem Description

Given

A 2D image with

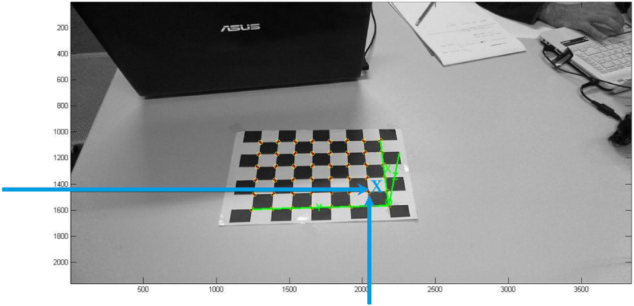
- a coordinate system
- an observed point $\mathbf{x}$ in the image

A camera modelled by a matrix P

Tasks

Determine:

- the observed point in 3D
- the precision of the estimated position



Formalisation

Given

An image I, a homogeneous image point $\mathbf{x} \in I$, $\mathbf{x} = \begin{pmatrix} \text{row} \\ \text{col} \\ 1 \end{pmatrix}$, and a camera model, which is a 3×4 matrix P that encodes:

1. the intrinsic camera parameters as a 3×3 matrix K
2. extrinsic parameters encoded by a rotation matrix R and the camera centre $\mathbf{C}$

Task

Determine a unique point

$$\mathbf{X} = \begin{pmatrix} a \\ b \\ c \\ 1 \end{pmatrix}$$

such that

$$P\mathbf{X} = \mathbf{x}$$

Observe

This task is **underdetermined**!

Camera Model Example[1]

```
import numpy as np

# camera centre in the world frame
C = np.array([1000, 2000, 1500])[np.newaxis].T

# intrinsic camera parameters
K = np.array([[468.2, 91.2, 300], [0, 427.2, 200], [0, 0, 1]])

# rotation world -> camera
R = np.array([[0.4138, 0.90915, 0.04708], \
[-0.57338, 0.22011, 0.78917], \
[0.70711, -0.35355, 0.61237]])

# M = K * R
M = np.array([[353.581904, 339.673062, 277.72616], \
[-103.525936, 23.320992, 459.607424], \
[0.70711, -0.35355, 0.61237]])

# P [M -M*C]
P = np.array([[353.553, 339.645, 277.744, -1449460], \
[-103.528, 23.3212, 459.607, -632525], \
[0.707107, -0.353553, 0.612372, -918.559]])

# sanity check: get back C from P
x = np.linalg.det(P[:,[1,2,3]])
y = -np.linalg.det(P[:,[0,2,3]])
z = np.linalg.det(P[:,[0,1,3]])
t = -np.linalg.det(P[:,[0,1,2]])

c = np.array([x / t, y / t, z / t])
print c
```

[1] Hartley/Zissermann, Chapter 6, p. 163

Calibration

Camera calibration

We can get P from K, R, and C via a process called **camera calibration**

How does the imprecision of the measured camera parameters influence the pixel coordinates in the image plane?

Calibration Strategy

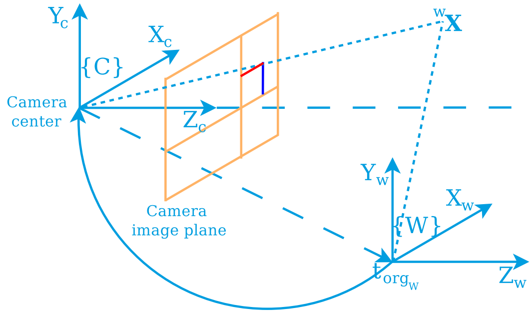
1. Start in the world frame $\{W\}$ and map to the camera frame $\{C\}$ — $\{W\} \rightarrow \{C\}$

2. Project to the image plane I (still using the camera frame coordinates) — $\{C\} \rightarrow \{I\}$

3. Map to image (pixel) coordinates — $\{I\} \rightarrow \{\mathbf{i}\}$

$1. \{W\} \rightarrow \{C\}$

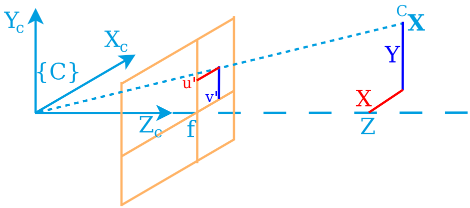
$$ {}^{C}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = {}^{C}_{W}T_{hom} \; {}^{W}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} {}^{C}_{W}\begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} & \mathbf{t}_{w_{org}} \\ 0 \quad 0 \quad 0 & 1 \end{pmatrix} {}^{W}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} $$



$$ \begin{aligned} r_{11} &= c\beta c\gamma \\ r_{12} &= s\alpha s\beta c\gamma - c\alpha s\gamma \\ r_{13} &= c\alpha s\beta c\gamma + s\alpha s\gamma \\ \\ r_{21} &= c\beta s\gamma \\ r_{22} &= s\alpha s\beta s\gamma + c\alpha s\gamma \\ r_{23} &= c\alpha s\beta s\gamma - s\alpha c\gamma \\ \\ r_{31} &= -s\beta \\ r_{32} &= s\alpha c\beta \\ r_{33} &= c\alpha c\beta \end{aligned} $$

Here, α, β, and γ are Euler angles

$\{C\} \to \{I\}$



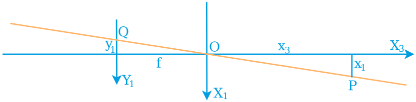
Given f (the **focal length**), we have

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

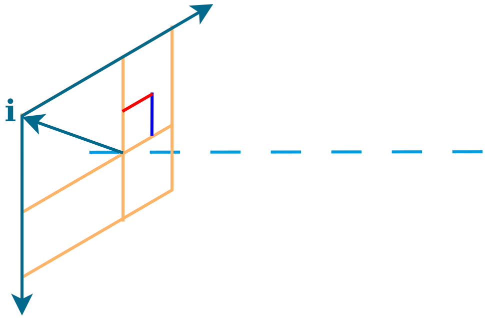
$$^{I}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix} = \frac{f}{Z}\begin{pmatrix} X \\ Y \\ \frac{Z}{f} \end{pmatrix} \sim {}^{C}\begin{pmatrix} X \\ Y \\ \frac{Z}{f} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix} {}^{C}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

where

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{f} & 0 \end{pmatrix} \sim \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} := P$$



$\{I\} \rightarrow \{\mathbf{i}\}$



$$^{\mathbf{i}}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} := \underbrace{\begin{pmatrix} fc(1) & \alpha_c \cdot fc(1) & cc(1) \\ 0 & fc(2) & cc(2) \\ 0 & 0 & 1 \end{pmatrix}}_{K} {}^{I}\begin{pmatrix} u' \\ v' \\ 1 \end{pmatrix}$$

Image to World Point Mapping Example

```
C =
 -355.4000
   92.1800
  605.5000

K =
   1.0e+03 *
   3.050393312453318                   0   1.986372170712007
                   0   3.024051420072965   0.9950097174[illegible]72
                   0                   0                   0
R =
  -0.059269656395942  -0.997265685484274   0.044139102690120
  -0.751404608620741   0.015460942212210  -0.659660574393697
   0.657174422793689  -0.072264180764005  -0.750266396824681
```

Must be

1 !!!!!

A new point is added to the image, whose world coordinates are $(30, 30, 0, 1)mm$ and whose image coordinates are $(2039, 1459, 1)$

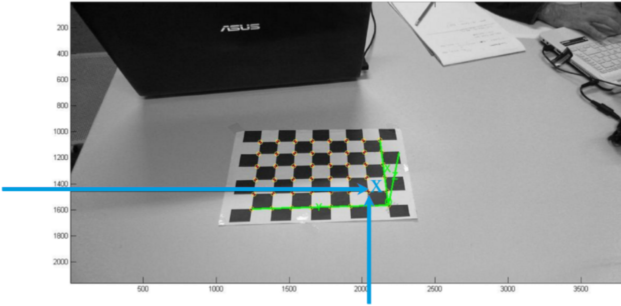
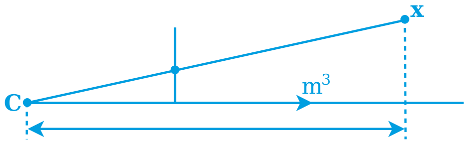


Image to World Point Mapping: Solution Idea (1/2)



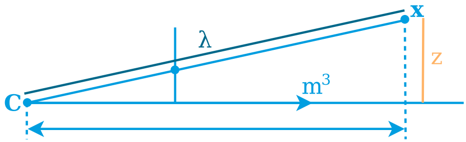
We can use

$$\mathbf{X}(\lambda) = P^{+}\mathbf{x} + \lambda\mathbf{C}$$

$\mathbf{X}(\lambda)$ is any point on the ray that connects the image point $\mathbf{x}$ and the camera centre $\mathbf{C}$

But what is λ?

Image to World Point Mapping: Solution Idea (2/2)



We can use

$$\mathbf{X}(\lambda) = P^{+}\mathbf{x} + \lambda\mathbf{C}$$

$\mathbf{X}(\lambda)$ is any point on the ray that connects the image point $\mathbf{x}$ and the camera centre $\mathbf{C}$

But what is λ?

If we know ONE component of X (in the real world), then the equation can be used to determine **one unique** λ - just based on this one known point

Then, having λ, we can determine the x and y components of $\mathbf{X}$

Image to World Point Mapping: Python Example

```
import sympy as sp

M = K.dot(R)
P = np.hstack((M, -M.dot(C)))
x = np.linalg.det(P[:,[1,2,3]])
y = -np.linalg.det(P[:,[0,2,3]])
z = np.linalg.det(P[:,[0,1,3]])
t = -np.linalg.det(P[:,[0,1,2]])
C = np.array([x / t, y / t, z / t, 1])[np.newaxis].T

P_plus = np.linalg.pinv(P)
X_image = np.array([2037, 1459, 1])[np.newaxis].T
known_z = 0. #the point is on the table

lamb = sp.Symbol('lambda')
X_lambda = P_plus.dot(X_image) + lamb * C
r = sp.solve(X_lambda[2][0] - known_z * X_lambda[3][0], lamb)[0]
Xq = X_lambda[0][0].subs(lamb, r)
Yq = X_lambda[1][0].subs(lamb, r)
Zq = X_lambda[2][0].subs(lamb, r)
Wq = X_lambda[3][0].subs(lamb, r)
Xq = Xq / Wq
Yq = Yq / Wq
Zq = Zq / Wq
print Xq, Yq, Zq
```

The result is correct up to $0.7mm$

Partial Output of the Caltech Calibration Toolbox

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.30254   657.74391 ] ± [ 0.28487   0.28937 ]
Principal point:       cc = [ 302.71656   242.33386 ] ± [ 0.59115   0.55710 ]
Skew:             alpha_c = [ 0.00042 ] ± [ 0.00019  ]   => angle of pixel axes
Distortion:            kc = [ -0.25349   0.11868   -0.00028   0.00005  0.00000
Pixel error:          err = [ 0.11743   0.11585 ]

Note: The numerical errors are approximately three times the standard deviation
```

We need to note that:

T_{hom} is undistorted

P is distorted only by the variance in $F(fc)$

K is distorted by known σ_α, σ_{cc}, $\sigma_{fc(1)}$, $\sigma_{fc(2)}$

$G(u,v)$ is distorted by a known σ_{kc}

Non-Linear Distortion $G(u,v)$

Unfortunately, u' and v' don't land where expected

$$ {}^{\mathbf{i}}\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = K \cdot {}^{I}\begin{pmatrix} u'' \\ v'' \\ 1 \end{pmatrix} = K \cdot G\left(P \cdot T_{hom} \; {}^{W}\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \right) $$

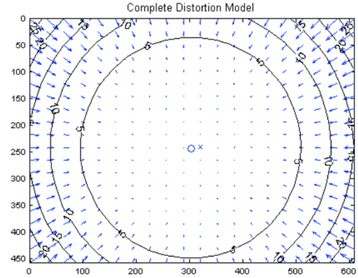
$$ \begin{pmatrix} u'' \\ v'' \end{pmatrix} = G\left(\begin{pmatrix} u' \\ v \end{pmatrix} \right) = \begin{pmatrix} G_1(u',v') \\ G_2(u',v') \end{pmatrix} $$

$$ = \begin{pmatrix} u'(1 + kc(1)r^2 + kc(2)r^4 + kc(5)r^6) \\ v'(1 + kc(1)r^2 + kc(2)r^4 + kc(5)r^6) \end{pmatrix} + \begin{pmatrix} du \\ dv \end{pmatrix} $$

$$ \begin{pmatrix} du \\ dv \end{pmatrix} = \begin{pmatrix} 2kc(3)u'v' + kc(4)(r^2 + 2u'^2) \\ kc(3)(r^2 + 2v'^2) + 2kc(4)u'v' \end{pmatrix} $$

$$ r = \sqrt{u'^2 + v'^2} $$

Lens distortion $\begin{pmatrix} u'' \\ v'' \end{pmatrix} = G(u', v')$



Pixel error	= [0.1174, 0.1159]	
Focal Length	= (657.303, 657.744)	+/- [0.2849, 0.2894]
Principal Point	= (302.717, 242.334)	+/- [0.5912, 0.5571]
Skew	= 0.0004198	+/- 0.0001905
Radial coefficients	= (-0.2535, 0.1187, 0)	+/- [0.00231, 0.009418, 0]
Tangential coefficients	= (-0.0002789, 5.174e-005)	+/- [0.0001217, 0.0001208]

Full Output of the Caltech Calibration Toolbox

T_{hom} is undistorted

P is distorted only by the variance in $F(fc)$

K is distorted by known σ_α, σ_{cc}, $\sigma_{fc(1)}$, $\sigma_{fc(2)}$

$G(u, v)$ is distorted by a known σ_{kc}

```
Calibration results after optimization (with uncertainties):

Focal Length:          fc = [ 657.30254   657.74391 ] ± [ 0.28487   0.28937 ]
Principal point:       cc = [ 302.71656   242.33386 ] ± [ 0.59115   0.55710 ]
Skew:             alpha_c = [ 0.00042 ] ± [ 0.00019  ]   => angle of pixel axes = 89.97595 ± 0.01092 degrees
Distortion:            kc = [ -0.25349   0.11868   -0.00028   0.00005  0.00000 ] ± [ 0.00231   0.00942   0.0
Pixel error:          err = [ 0.11743   0.11585 ]

Note: The numerical errors are approximately three times the standard deviations (for reference).
```

Uncertainties $S_{G(kc(1,\ldots,6))}$

If S_{z_i} are empirical variances and the z_is are independent (i.e. $S_{z_i z_j} \approx 0$ for $i \neq j$), then it holds that

$$S_G^2 = \sum_i S_{z_i}^2 \left(\frac{\partial G}{\partial z_i} \right)^2$$

In case G has imprecise coefficients and imprecise input values, take the partial derivative w.r.t. all varying inputs and neglect the second order terms

$S^2_{u''} + S^2_{v''}$

$$
\begin{aligned}
S^2_{u''} &= \sum_{i=1}^{6} S^2_{kc(i)} \left(\frac{\partial G_1}{\partial kc(i)} \right)^2 \\
&= S^2_{kc(1)}(r^4 u^2) + S^2_{kc(2)}(r^8 u^2) + S^2_{kc(3)}(4u^2 v^2) \\
&\quad + S^2_{kc(4)}(r^2 + 2u^2)^2 + S^2_{kc(5)}(r^{12} u^2)
\end{aligned}
$$

$$
\begin{aligned}
S^2_{v''} &= \sum_{i=1}^{6} S^2_{kc(i)} \left(\frac{\partial G_2}{\partial kc(i)} \right)^2 \\
&= S^2_{kc(1)}(r^4 v^2) + S^2_{kc(2)}(r^8 v^2) + S^2_{kc(4)}(4u^2 v^2) \\
&\quad + S^2_{kc(3)}(r^2 + 2v^2)^2 + S^2_{kc(5)}(r^{12} v^2)
\end{aligned}
$$

$$
\begin{aligned}
|r| &\leq \sqrt{\max v^2 + \max u^2} \\
|v| &\leq \max v \\
|u| &\leq \max u
\end{aligned}
$$

Imprecision in K

$$
\begin{aligned}
S_u^2 &= S_{fc_1}^2 \frac{\partial K_1}{\partial fc_1} + S_\alpha^2 \frac{\partial K_1}{\partial \alpha} + S_{cc_1}^2 \frac{\partial K_1}{\partial cc_1} \\
&= S_{fc_1}^2 (u + \alpha v)^2 + S_\alpha^2 fc_1^2 v^2 + S_{cc_1}^2
\end{aligned}
$$

$$
\begin{aligned}
S_v^2 &= S_{fc_2}^2 \frac{\partial K_2}{\partial fc_2} + S_{cc_2}^2 \frac{\partial K_2}{\partial cc_2} \\
&= S_{fc_2}^2 v^2 + S_{cc_2}^2
\end{aligned}
$$

Total Imprecision

Combine both imprecisions

Propagate all effects through