CAMERA CALIBRATION : INTRINSIC & EXTRINSIC

THE FOLLOWING NOTES GO OVER

- The basic setup

Simple Intrinsic + Extrinsic Calibration

Using the constant calibration matrix, we know that

$$r = \Psi \cdot q^c / z^c = \Psi \begin{Bmatrix} x^c/z^c \\ y^c/z^c \\ 1 \end{Bmatrix} = \begin{bmatrix} f & 0 & -r_0^1 \\ 0 & f & -r_0^2 \end{bmatrix} \begin{Bmatrix} x^c/z^c \\ y^c/z^c \\ 1 \end{Bmatrix}$$

$\Rightarrow$

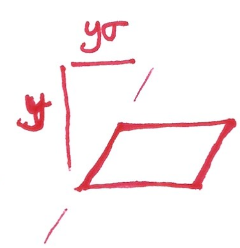
$$z^c r = \Psi \, \mathbf{p}^c$$

Now, Ψ can be made more complicated

$$\Psi = \begin{bmatrix} f/dr^1 & \sigma & -r_0^1 \\ 0 & f/dr^2 & -r_0^2 \end{bmatrix}$$

σ - skew

dr^1, dr^2 - in case not square



$\Rightarrow$

$$z^c \begin{Bmatrix} r^1 \\ r^2 \\ 1 \end{Bmatrix} = \Psi_0 \, \mathbf{p}^c = \begin{bmatrix} f/dr^1 & \sigma & -r_0^1 \\ 0 & f/dr^2 & -r_0^2 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{p}^c$$

homogeneous form

$\Rightarrow$

$$z^c \begin{Bmatrix} r^1 \\ r^2 \\ 1 \end{Bmatrix} = \Psi_0 R^c_w p^w + \Psi_0 d^c_w$$

$$= \left[\Psi_0 R^c_w \;\middle|\; \Psi_0 d^c_w \right] q^w$$

$$= [M \mid v] \, q^w = D q^w$$

called projection matrix.

Extrinsic Parameter Calibration

Well, let's recall the equations we had in the stereo depth case:

$$\Psi(r)(R_c^w)^T p^w = \Psi(r)(R_c^w)^T d_c^w$$

$\Rightarrow$

$$\Psi(r)(R_c^w)^T p^w - \Psi(r)(R_c^w)^T d_c^w = 0$$

$\Rightarrow$

$$\begin{bmatrix} (R_c^w)^T & -(R_c^w)^T d_c^w \\ 0 & 1 \end{bmatrix} \begin{bmatrix} p^w \\ 1 \end{bmatrix} = 0$$

$\Rightarrow$

$$[\Psi(r) \mid 0]\,(g_c^w)^{-1}\, q^w = 0$$

if f is known then given a point in space q^w and its projection the equation turns out to be linear in R_w^c and d_w^c

$$\underbrace{[\Psi(r) \mid 0]}_{\text{known}}\, g_w^c\, \underbrace{q^w}_{\text{known}} = 0$$

let's work this out

$$\begin{bmatrix} f & 0 & -r^1 & 0 \\ 0 & f & -r^2 & 0 \end{bmatrix} \begin{bmatrix} R_{11} & R_{12} & R_{13} & d_1 \\ R_{21} & R_{22} & R_{23} & d_2 \\ R_{31} & R_{32} & R_{33} & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} = 0$$

$\Rightarrow$

$$\begin{bmatrix} fR_{11} - r^1 R_{31} & fR_{12} - r^1 R_{32} & fR_{13} - r^1 R_{33} & fd_1 - r^1 d_3 \\ fR_{21} - r^2 R_{31} & fR_{22} - r^2 R_{32} & fR_{23} - r^2 R_{33} & fd_2 - r^2 d_3 \end{bmatrix} \begin{Bmatrix} x \\ y \\ z \\ 1 \end{Bmatrix} = 0$$

$\Rightarrow$

$$fxR_{11} - r^1 x R_{31} + fy R_{12} - r^1 y R_{32} + fzR_{13} - r^1 z R_{33} + fd_1 - r^1 d_3 = 0$$

$$fx R_{21} - r^2 x R_{31} + fy R_{22} - r^2 y R_{32} + fzR_{23} - r^2 z R_{33} + fd_2 - r^2 d_3 = 0$$

$\Rightarrow$

$$\begin{bmatrix} fx & fy & fz & f & 0 & 0 & 0 & 0 & -r^1 x & -r^1 y & -r^1 z & -r^1 \\ 0 & 0 & 0 & 0 & fx & fy & fz & f & -r^2 x & -r^2 y & -r^2 z & -r^2 \end{bmatrix} \begin{Bmatrix} R_{11} \\ R_{12} \\ R_{13} \\ d_1 \\ R_{21} \\ R_{22} \\ R_{23} \\ d_2 \\ R_{31} \\ R_{32} \\ R_{33} \\ d_3 \end{Bmatrix} = 0$$

$\Rightarrow$

$$\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -\frac{r^1 x}{f} & -\frac{r^1 y}{f} & -\frac{r^1 z}{f} & -\frac{r^1}{f} \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -\frac{r^2 x}{f} & -\frac{r^2 y}{f} & -\frac{r^2 z}{f} & -\frac{r^2}{f} \end{bmatrix} \begin{Bmatrix} R_{11} \\ R_{12} \\ R_{13} \\ d_1 \\ R_{21} \\ R_{22} \\ R_{23} \\ d_2 \\ R_{31} \\ R_{32} \\ R_{33} \\ d_3 \end{Bmatrix} = 0$$

2 equations, 12 unknowns.

How do we resolve the undetermined system (less equations, than unknowns)? We add more points & their projections!

define $\Phi(q,r) = \begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -r^1x/f & -r^1y/f & -r^1z/f & -r^1/f \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -r^2x/f & -r^2y/f & -r^2z/f & -r^2/f \end{bmatrix}$

then

$$\begin{bmatrix} \Phi(q_1,r_1) \\ \vdots \\ \Phi(q_n,r_n) \end{bmatrix} \vec{g} = 0 \qquad \text{where } \vec{g} = \begin{Bmatrix} R_{11} \\ R_{12} \\ R_{13} \\ d_1 \\ R_{21} \\ R_{22} \\ R_{23} \\ d_2 \\ R_{31} \\ R_{32} \\ R_{33} \\ d_3 \end{Bmatrix}$$

leads to $2n$ equations, 12 unknowns.

to solve, need at least $n = 6$. * need to know f.

using $n > 6$ leads to least squares solution.

[SOLVE USING SVD APPROACH]

* there are lots of other methods. what distinguishes them is the information needed for calibration. Here we asked for the 3D point coordinates plus the image coordinates.

some methods remove need for 3D point coordinates & use a flat checkerboard w/ known square lengths. More than 6 points will be needed then.

Camera Calibration

· Wait, but how do we know f & g_C^W for a single camera?

- or f_L, f_R, & g_R^L for a stereo rig?

finding these quantities is known as camera calibration.

~~these are actually~~

the camera parameters are broken up into two portions

1. intrinsic parameters - parameters needed to project a point in camera frame to image plane. (f)

2. extrinsic parameters - parameters needed to know camera frame relative to some world frame. (R & d)

- for equations I've defined only intrinsic parameter is f, but there are way more complicated models that have way more. for example, all of the important parameters that I gave you in HW 1 Problem 1. Imagine you ~~didn't know~~ didn't know those & only knew the final image resolution.

note that D is 3×4 $\Rightarrow$ 12 elements

but

$R \rightarrow$ 3 unique ~~elements~~ variables

$d \rightarrow$ 3 unique elements

$\Phi_0 \rightarrow$ 5 unique ~~elements~~

11 unique variables

$\Rightarrow$

we need at least 11 ~~pts~~ equations

$$\frac{z^c r^1}{z^c} = r^1 = \frac{[Dq^w]_1}{[Dq^w]_3} \qquad \frac{z^c r^2}{z^c} = \frac{[Dq^w]_2}{[Dq^w]_3}$$

ratio introduces a scale ambiguity.

$\Rightarrow$

$$r^1 [Dq^w]_3 = [Dq^w]_1$$

$$r^2 [Dq^w]_3 = [Dq^w]_2$$

$\Rightarrow$

$$r^1 [D_{31} x^w + D_{32} y + D_{33} z + D_{34}] = D_{11} x^w + D_{12} y^w + D_{13} z + D_{14}$$

$$r^2 [D_{31} x^w + D_{32} y^w + D_{33} z^w + D_{34}] = D_{21} x^w + D_{22} y^w + D_{23} z^w + D_{24}$$

$\Rightarrow$ factor elements of D

$$\begin{bmatrix} x^W & y^W & z^W & 1 & 0 & 0 & 0 & 0 & -r^1x^W & -r^1y^W & -r^1z^W & -r^1 \\ 0 & 0 & 0 & 0 & x^W & y^W & z^W & 1 & -r^2x^W & -r^2y^W & -r^2z^W & -r^2 \end{bmatrix} \begin{bmatrix} D_{11} \\ D_{12} \\ D_{13} \\ D_{14} \\ D_{21} \\ D_{22} \\ D_{23} \\ D_{24} \\ D_{31} \\ D_{32} \\ D_{33} \\ D_{34} \end{bmatrix} = 0$$

2 equations & 12 unknowns

(for 11 unique variables)

each q^W, r pairing gives two equations, so we'll need 6 points in the world matched to the corresponding image coordinates.

with these points, one can solve the linear system of equations, to obtain the projection matrix D up to scale. (this is where scale ambiguity pops up!)

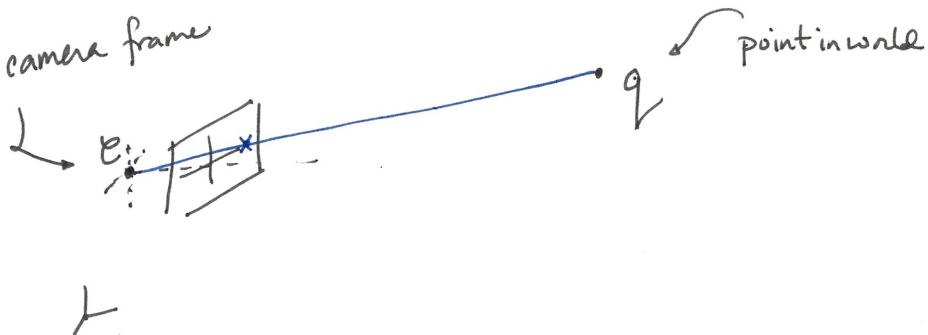
Recall $D = [M_0 \mid v_0] = [\Phi_0 R^c_W \mid \Phi_0 d^c_W]$

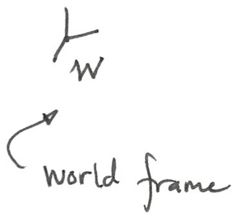
Note that $M_0 = \Phi_0 R^c_W$

orthogonal matrix (or orthonormal matrix)

upper triangular

Camera Calibration via Cross-Product and SVD





We have that

$$\vec{r}^{\,c} \approx \Psi \left[R^c_W \middle| T^c_W \right] q^W$$

or, when consolidated

$$\vec{r}^{\,c} \approx M q^W$$

For a given (static) camera setup, how does one recover M? What about Ψ and $g^c_W = \left[\begin{array}{c|c} R^c_W & T^c_W \\ \hline 0 & 1 \end{array}\right]$?

This estimation or recovery process is called "camera calibration". One way to perform camera calibration is to use some cross-product identities to create a system of equations solveable via the singular value decomposition.

The cross-product trick is clever. Since $\vec{r}^{\,c}$ and Mq^W are equivalent rays, as vectors they are parallel. Parallel vectors have a vanishing cross-product:

$$\vec{r}^{\,c} \times M q^W = \vec{0} \qquad (1)$$

Another example is the matrix times vector situation

$$A\vec{b}$$

which is linear in A,

$$(A_1 + A_2)\vec{b} = A_1 b + A_2 b$$

$$(\alpha A)\vec{b} = \alpha(A\vec{b})$$

Now, to transform it lets write A as a set of row vectors

$$A = \underbrace{\begin{bmatrix} — & \vec{a}_1^T & — \\ — & \vec{a}_2^T & — \\ — & \vec{a}_3^T & — \end{bmatrix}}_{3\times 3}$$

← here let A be 3×3, but can work for any size A. Each $\vec{a}_i$ is a 3×1 vector.

Then,

$$\underset{3\times3}{A}\,\underset{3\times1}{\vec{b}} = \underset{3\times1}{\begin{bmatrix} \vec{a}_1^T \cdot \vec{b} \\ \vec{a}_2^T \cdot \vec{b} \\ \vec{a}_3^T \cdot b \end{bmatrix}} = \underset{\text{dot product}}{\begin{bmatrix} \vec{a}_1 \cdot \vec{b} \\ \vec{a}_2 \cdot \vec{b} \\ \vec{a}_3 \cdot b \end{bmatrix}} = \underset{\text{dot product is symmetric}}{\begin{bmatrix} \vec{b} \cdot \vec{a}_1 \\ \vec{b} \cdot \vec{a}_2 \\ \vec{b} \cdot \vec{a}_3 \end{bmatrix}} = \begin{bmatrix} \vec{b}^T \vec{a}_1 \\ \vec{b}^T \vec{a}_2 \\ \vec{b}^T a_3 \end{bmatrix}$$

so

$$A\vec{b} = \underset{3\times1}{\begin{bmatrix} \vec{b}^T \vec{a}_1 \\ \vec{b}^T \vec{a}_2 \\ \vec{b}^T \vec{a}_3 \end{bmatrix}} = \underset{3\times9}{\begin{bmatrix} \vec{b}^T & 0 & 0 \\ 0 & \vec{b}^T & 0 \\ 0 & 0 & \vec{b}^T \end{bmatrix}} \underset{9\times1}{\begin{bmatrix} \vec{a}_1 \\ \vec{a}_2 \\ \vec{a}_3 \end{bmatrix}}$$

As an equation, (1) is linear in each of the entries.
This is good because a system of equations linear in one of the quantities can always be written as a matrix time the vector form of the given quantity.

For example, the cross-product is linear.

$$\vec{a} \times \vec{b} \qquad \left[\begin{array}{l} (\vec{a}_1 + \vec{a}_2) \times \vec{b} = \vec{a}_1 \times b + \vec{a}_2 \times b \\ (\alpha \vec{a}) \times \vec{b} = \alpha (\vec{a} \times \vec{b}) \end{array} \right.$$

That means it can be written as

$$A(a) \cdot b$$

where $A(a)$ is a special matrix equivalent to the vector cross product in its effect, i.e.,

- There exists a matrix $A(a)$ such that $A(\vec{a}) \cdot \vec{b} = \vec{a} \times \vec{b}$

That matrix is called the cross-product matrix or the hatted matrix form of $\vec{a}$. For $\vec{a} \in \mathbb{R}^3$, we have

$$\hat{a} = \begin{bmatrix} 0 & a^3 & -a^2 \\ -a^3 & 0 & a^1 \\ a^2 & -a^1 & 0 \end{bmatrix} \qquad \text{where } \vec{a} = \begin{bmatrix} a^1 \\ a^2 \\ a^3 \end{bmatrix}$$

Some people write $[\vec{a}]_\times$ or $\vec{a}_\times$ instead of $\hat{a}$.

Cross-product symbol.

which means that there exists a matrix $B(\vec{b})$ depending on the entries of $\vec{b}$ such that

$$A\vec{b} = B(\vec{b})\,\vec{a}$$

where $\vec{a}$ is the (row-wise) vectorization of A. $\quad \vec{a} \in \mathbb{R}^9$

From the last page,

$$B(\vec{b}) = \underbrace{\begin{bmatrix} \vec{b}^T & 0 & 0 \\ 0 & \vec{b}^T & 0 \\ 0 & 0 & \vec{b}^T \end{bmatrix}}_{3\times 9}$$

Going back to our problem,

$$\vec{r}^{\,c} \times M q^w = \vec{0}$$

$\Rightarrow$ use cross-product hat matrix

$$\underbrace{\hat{r}^c}_{3\times 3}\ \underbrace{M}_{3\times 4}\ \underbrace{q^w}_{4\times 1} = \underbrace{\vec{0}}_{3\times 1}$$

$\Rightarrow$ use matrix vectorization trick

$$\hat{r}^c\, Q(q^w)\,\vec{m} = \vec{0}$$

$$Q(\vec{q}) = \underbrace{\begin{bmatrix} \vec{q}^T & 0 & 0 \\ 0 & \vec{q}^T & 0 \\ 0 & 0 & \vec{q}^T \end{bmatrix}}_{3\times 12} \qquad \vec{m} \in \mathbb{R}^{12}$$

$\Rightarrow$ Combine $\hat{r}$ and Q

$$Q(\vec{r}^{\,c}, q^w)\,\vec{m} = \vec{0}$$

$$\longrightarrow \quad Q(\vec{r},\vec{q}) = \hat{r}\cdot Q(\vec{q}) = \begin{bmatrix} 0 & r^3 & -r^2 \\ -r^3 & 0 & r^1 \\ r^2 & -r^1 & 0 \end{bmatrix} \begin{bmatrix} \vec{q}^T & 0 & 0 \\ 0 & \vec{q}^T & 0 \\ 0 & 0 & \vec{q}^T \end{bmatrix}$$

What we get is a system of 3 equations for 12 unknowns. Sadly, the cross-product and also the ray nature of $\vec{r}$ mean that there are really 2 independent equations, with 1 dependent equation.

Thus we have 2 independent equations for 12 unknowns. How do we solve?

Take 6 unique q^W points with 6 unique image projections $\vec{r}^{\,c}$. They will give $6 \times 2 = 12$ independent equations.

$$Q(\vec{r}_1^{\,c}, q_1^W)\vec{M} = \vec{0}$$

$$Q(\vec{r}_2^{\,c}, \vec{q}_2^{\,W})\vec{M} = \vec{0}$$

$$Q(\vec{r}_3^{\,c}, \vec{q}_3^{\,W})\vec{M} = \vec{0}$$

$$\vdots$$

$$Q(\vec{r}_6^{\,c}, \vec{q}_6^{\,W})\vec{M} = \vec{0}$$

TOTAL OF 18 EQUATIONS (ONLY 12 ARE INDEPENDENT)

NOT NECESSARY ACTUALLY

but since only first two rows of each Q are needed, let $Q_{12}(\vec{r}, \vec{q})$ be the first two rows of $Q(\vec{r}, \vec{q})$ with the third row dropped. Ignoring the last row of each $Q(\vec{r}, \vec{q})$ gives 12 equations for 12 unknowns.

So,

$$\begin{bmatrix} Q_{12}(\vec{r}_1^{\,c},\, q_1^w) \\ \vdots \\ Q_{12}(\vec{r}_6,\, q_6^w) \end{bmatrix} \vec{m} = \vec{0}$$

$$12 \times 12 \qquad 12\times 1 = 12 \times 1$$

is a complete set of equations. They can be solved by using the singular value decomposition. The last, right singular vector gives $\vec{m}$ up to scale.

If desired one could use all of $Q(\vec{r}, q)$ to solve for $\vec{m}$

$$\begin{bmatrix} Q(\vec{r}_1^{\,c},\, q_1^w) \\ \vdots \\ Q(\vec{r}_6^{\,c},\, q_6^w) \end{bmatrix} \vec{M} = \vec{0}$$

$$18 \times 12 \qquad 12\times 1 = 18\times 1$$

The answer is still the last, right singular vector of the big Q matrix (up to scale).

Then

$$M = \begin{bmatrix} - & m_{1\ldots4}^T & - \\ - & m_{5\ldots8}^T & - \\ - & m_{9\ldots12}^T & - \end{bmatrix} = \begin{bmatrix} m_1 & m_2 & m_3 & m_4 \\ m_5 & m_6 & m_7 & m_8 \\ m_9 & m_{10} & m_{11} & m_{12} \end{bmatrix}$$

But what about the scale?

It's not *that* important for M. The reason is that the scale cancels out during the projection part.

Consider the two cases

$$\vec{r}_1 \approx M q^W \quad \& \quad \vec{r}_2 \approx \alpha M q^W$$

if $M = \begin{bmatrix} - \vec{m}_1^T - \\ - \vec{m}_2^T - \\ - \vec{m}_3^T - \end{bmatrix}$ then

$$\vec{r}_1 \approx \begin{bmatrix} \vec{m}_1^T q^W \\ \vec{m}_2^T q^W \\ \vec{m}_3^T q^W \end{bmatrix} \qquad \vec{r}_2 \approx \begin{bmatrix} \alpha \vec{m}_1^T q^W \\ \alpha \vec{m}_2^T q^W \\ \alpha \vec{m}_3^T q^W \end{bmatrix}$$

$\Rightarrow$ normalizing the last coordinate as per true projection

The α terms cancel.

$$\vec{r}_1 \approx \begin{bmatrix} \vec{m}_1^T q^W / \vec{m}_3^T q^W \\ \vec{m}_2^T q^W / \vec{m}_3^T q^W \\ \hline 1 \end{bmatrix} \qquad \vec{r}_2 \approx \begin{bmatrix} \alpha \vec{m}_1^T q^W / \alpha \vec{m}_3^T q^W \\ \alpha \vec{m}_2^T q^W / \alpha \vec{m}_3^T q^W \\ \hline 1 \end{bmatrix}$$

$\Rightarrow$ only take top 2 coordinates to get a 2x1 vector

$$\vec{r}_1 = \begin{bmatrix} \vec{m}_1^T q^W / \vec{m}_3^T q^W \\ \vec{m}_2^T q^W / \vec{m}_3^T q^W \end{bmatrix} = \vec{r}_2$$

They project to the same point! The unknown scale is not important!

SIDE NOTE : QR FACTORIZATION

WANT TO FIND $\underbrace{\Phi_0}_{\text{upper triangular}}$ & $\underbrace{R}_{\text{orthonormal.}}$ SO THAT $M_0 = \Phi_0 R$

TURNS OUT THAT THERE IS A WAY TO FACTOR M_0 INTO TWO PIECES. IT'S CALLED QR FACTORIZATION.

IF A IS A SQUARE, INVERTIBLE MATRIX, THEN THERE IS A UNIQUE DECOMPOSITION OF A INTO

$$A = \underbrace{Q}_{\text{orthonormal}} \cdot \underbrace{U}_{\text{upper triangular.}}$$

OUCH! IT'S BACKWARDS FROM WHAT WE NEED.

WE HAVE

$$B = \bar{U}\,\bar{Q}$$

FIRST, NOTE

$$B^T = \underbrace{\bar{Q}^T}_{\text{orthonormal still}}\ \underbrace{\bar{U}^T}_{\text{now lower triangular.}}$$

WE NEED TO GET $\bar{U}^T$ TO BE UPPER TRIANGULAR.
THIS CAN BE DONE BY FLIPPING ROWS & FLIPPING COLUMNS

THE FLIP OPERATION CAN BE GIVEN BY F, A MATRIX OF ONES ALONG THE ANTI-DIAGONAL.

$$F = \begin{bmatrix} 0 & & 1 \\ & ⋰ & \\ 1 & & 0 \end{bmatrix}$$

TO FLIP COLUMNS $\rightarrow$ $B \cdot F$

TO FLIP ROWS $\rightarrow$ $F \cdot B$

TO FLIP ROWS & COLUMNS $\rightarrow$ $F \cdot B \cdot F$

NOTE $F \cdot F = \mathbb{1}$ AND TRANSPOSE $(F) = F^T = F$.

$\Rightarrow$

$$F B^T F = F \bar{Q}^T \bar{U}^T F$$

$$= \underbrace{F \bar{Q}^T F}\ \underbrace{F \bar{U}^T F}$$

upper-triangular

$\Rightarrow$

$$F B^T F = Q \quad U$$

still orthonormal.

$$(F\bar{Q}^T F)(F\bar{Q}^T F)^T = F\bar{Q}^T F F^T (\bar{Q}^T)^T F^T$$

$$= F \bar{Q}^T \bar{Q} F^T$$

$$= F F^T$$

$$= \mathbb{1} \quad \checkmark$$

so, to get $\bar{Q}$ & $\bar{U}$, we do the following

$$[Q, U] = \text{qr factorize}\,(F B^T F)$$

this gives Q & U, but sign of elements in U may be off.

Let $S =$ sign of diagonal elements of U

in Matlab `S = diag(sign(diag(U)));`

Then, $\quad FB^T F = QS \cdot SU$

$\Rightarrow$

$$F\bar{Q}^T F = QS$$
$$F\bar{U}^T F = SU$$

$\Rightarrow$

$$\bar{Q}^T = FQSF$$
$$\bar{U}^T = FSUF$$

$\Rightarrow$

$$\bar{Q} = (FQSF)^T$$
$$\bar{U} = (FSUF)^T$$

so, given the matrix $D_0 = [M_0 | v_0]$, extract M_0 ← (3x3 submatrix of D)

then solve

$$[Q, U] = qr(F M_0^T F)$$

to get

$$\Phi_0 = (FSUF)^T$$
$$R = (FQSF)^T$$

where

S = sign of diagonal ~~elements~~ of U. (matrix, portion) but recall that, if put back in the frames,

$$D_0 = \left[\, \Phi_0 R_W^C \mid \Phi_0 d_W^C \,\right]$$

$\Rightarrow$

$$R_W^C = \text{transpose}(FSUF)^*$$

$\Rightarrow$

$$R_C^W = (R_W^C)^T = FSUF$$

don't need the final transpose.

Secondly, note that $v_0 = \Phi_0 d_W^C$ & $M_0 = \Phi_0 R_W^C$

$\Rightarrow$

$$M_0^{-1} v_0 = (R_W^C)^{-1} \Phi_0^{-1} \Phi_0 d_W^C = (R_W^C)^{-1} d_W^C$$

$\Rightarrow$

$$d_C^W = (R_W^C)^{-1} d_W^C = M_0^{-1} v_0$$

and we've successfully calibrated the extrinsic parameters. what about the intrinsic?

now, since D is known up to scale, the true Ψ_o is

$$\Psi_o = \alpha (\bar{U}P)^T$$

to figure out that scale will require solving as best as possible,

$$r = \alpha \bar{\Psi} p^c / z^c \qquad \cancel{\bar{\Psi} = (\bar{U}P) \to \text{transpose}(\text{flipud}(\bar{U}))}$$

$$\Rightarrow r = \alpha \bar{\Psi} \cancel{R^c_w p^w + d^c_w} \quad \cancel{g^c_w q^w} \quad \frac{R^c_w p^w + d^c_w}{[R^c_w p^w + d^c_w]_3}$$

it is two equations, one unknown.

so, for this we'll need a point / image coordinate pair.

can take one of the six or a new one, or take many and use least squares.

$$\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ \vdots \\ r_n \end{bmatrix} = \alpha \begin{bmatrix} \bar{\Psi}_o P^c_1 / z^c_1 \\ \bar{\Psi}_o P^c_2 / z^c_2 \\ \vdots \\ \bar{\Psi}_o P^c_n / z^c_n \end{bmatrix} \qquad \text{where } P^c_i = R^c_w p^w_i + d^c_w$$

$$b = A\alpha$$

$$A^+ b = \alpha$$

$$\alpha = A^+ b$$

then $\Psi_o = \alpha \bar{\Psi}_o$

* OR JUST USE FACT THAT LAST ROW & LAST COLUMN ENTRY MUST BE 1! MUCH EASIER!

- some camera's have distortion which is nonlinear.

  barrel or radial distortion is one such form.

  this is when a straight line looks curved. happens away (more pronounced) from the center of the image

  also, ~~sometimes~~ the ~~CCD~~ sensor center may not agree with the true focal center (the true camera axis)

- requires fundamentally nonlinear techniques. ~~But~~ ~~A~~ good start is with the linear solution, then see what kinds of problems ensue.

- The nonlinear solution involves setting up a least squares optimization problem, then solving using nonlinear techniques.

  The most popular is the Levenberg-Marquiart method. It is an iterative solver (spelling?) using linear estimate approximate solutions that converge to the true solution (hopefully!).