STEREO CAMERAS.

3D POINT RECOVERY AND DEPTH RECOVERY.

WHAT FOLLOWS ARE PDF SCANS OF DIFFERENT SOLUTIONS. THEY VARY ON THE EQUATIONS GRABBED AND THE LINEAR SYSTEM SETUP.

SOME OUTLINE OF THE APPROACHES GOES AS FOLLOWS

I] A version that uses the simplest pinhole projection model.

- Has notes for how to make more complex (by a little).
- solves in world frame.
- covers g^W_L & g^W_R or g^L_W & g^R_W known.

II] A version that takes solution I and specializes to solve for left camera frame only.

- called relative depth when done w.r.t. a camera frame

III] A version that first solves for depths (left & right cameras), then solves for actual world frame 3D point location.

I] Computing Depth

ok, so now we finally know what's going on w/(R, p).
therefore it should be possible to compute depth given
r_L and r_R.

from our previous equation,

$$\begin{bmatrix} r_R^1 \\ r_R^2 \\ f \end{bmatrix} \frac{z_R}{z_L} = R \begin{bmatrix} r_L^1 \\ r_L^2 \\ f \end{bmatrix} + \frac{f}{z_L} p$$

$\Rightarrow$

$$\begin{bmatrix} r_R^1 \\ r_R^2 \\ f \end{bmatrix} z_R = R \begin{bmatrix} r_2^1 \\ r_2^2 \\ f \end{bmatrix} z_L + f p$$

3 linear equations, 2 unknowns

$\rightarrow$ solve any way desired for z_R & z_L.

so, easy! where's the trick?

– Well, it's in knowing the ~~corresponding points~~ conjugate pair r_L & r_R ~~for a conjugate point~~.

Q: what if we are given the right and left image coordinates r_R & r_L, then what is the true world coordinate q^W? (inverse case: "depth" finding)

- this is a trickier problem since it requires inversion of the imaging process. let's examine more deeply the projection equations to see how they'd be inverted.

$$\begin{Bmatrix} r_L^1 \\ r_L^2 \end{Bmatrix} = \frac{f_L}{z^L} \begin{Bmatrix} x^L \\ y^L \end{Bmatrix}$$

← note that center of image is at (0,0)

$$\begin{Bmatrix} r_R^1 \\ r_R^2 \end{Bmatrix} = \frac{f_R}{z^R} \begin{Bmatrix} x^R \\ y^R \end{Bmatrix}$$

← SIMPLEST MODEL POSSIBLE!

recall

$$p^L = R_W^L p^W + d_W^L = \begin{Bmatrix} x^L \\ y^L \\ z^L \end{Bmatrix}$$

$$p^R = R_W^R p^W + d_W^R = \begin{Bmatrix} x^R \\ y^R \\ z^R \end{Bmatrix}$$

⇒

$$r_L^1 = f_L \left([R_W^L]_1 p^W + [d_W^L]_1 \right) / \left([R_W^L]_3 p^W + [d_W^L]_3 \right)$$

$$r_L^2 = f_L \left([R_W^L]_2 p^W + [d_W^L]_2 \right) / \left([R_W^L]_3 p^W + [d_W^L]_3 \right)$$

& similarly for r_R^1, r_R^2.

- remember r_L and r_R are known.

where $[A]_i$ takes the i^{th} row of matrix A.
and $[V]_i$ takes i^{th} coordinate of vector V.

next, let's manipulate the equations to get a linear system of equations for p^W. then we'll be able to solve for p^W.

$\Rightarrow$ multiply by z^L (or z^R) as needed

$$r_L^1\left([R_W^L]_3 p^W + [d_W^L]_3\right) - f_L\left([R_W^L]_1 p^W + [d_W^L]_1\right) = 0 \tag{1a}$$

$$r_L^2\left([R_W^L]_3 p^W + [d_W^L]_3\right) - f_L\left([R_W^L]_2 p^W + [d_W^L]_2\right) = 0 \tag{1b}$$

$$r_R^1\left([R_W^R]_3 p^W + [d_W^R]_3\right) - f_R\left([R_W^R]_1 p^W + [d_W^R]_1\right) = 0 \tag{1c}$$

$$r_R^2\left([R_W^R]_3 p^W + [d_W^R]_3\right) - f_R\left([R_W^R]_2 p^W + [d_W^R]_2\right) = 0 \tag{1d}$$

- we have a linear system of 4 equations & 3 unknowns.
- the reason for keeping all 4 equations is that 1 is redundant, but we can't be sure which. if we had knowledge of R_W^L, R_W^R, d_W^L, & d_W^R then we could specialize things. since we don't, we keep it generic.
- the next step is to figure out the algebraic and matrix manipulations to get the system to look as much like

$$A p^W = b$$

as possible to solve for p^W.

$\Rightarrow$ factor out matrix multiplier

$$\begin{bmatrix} -f_L & 0 & r_L^1 \\ 0 & -f_L & r_L^2 \end{bmatrix} (R_W^L p^W + d_W^L) = 0$$

$$\begin{bmatrix} -f_R & 0 & r_R^1 \\ 0 & -f_R & r_R^2 \end{bmatrix} (R_W^R p^W + d_W^R) = 0$$

$\Rightarrow$

$$\begin{bmatrix} f_L & 0 & -r_L^1 \\ 0 & f_L & -r_L^2 \end{bmatrix} R_W^L p^W = -\begin{bmatrix} f_L & 0 & -r_L^1 \\ 0 & f_L & -r_L^2 \end{bmatrix} d_W^L$$

$$\begin{bmatrix} f_R & 0 & -r_R^1 \\ 0 & f_R & -r_R^2 \end{bmatrix} R_W^R p^W = -\begin{bmatrix} f_R & 0 & -r_R^1 \\ 0 & f_R & -r_R^2 \end{bmatrix} d_W^R$$

$\Rightarrow$

let's define the following

$$\Psi_L(r_L) = \begin{bmatrix} f_L & 0 & -r_L^1 \\ 0 & f_L & -r_L^2 \end{bmatrix} = [\, f_L \mathbb{1} \mid r_L \,]$$

2x2 identity

column vector giving image coord.

$$\Psi_R(r_R) = [\, f_R \mathbb{1} \mid r_R \,]$$

if (0,0) is not center, then need to adjust

$$\Psi_L(r_L) = \begin{bmatrix} f_L & 0 & -r_L^1 + r_0^1 \\ 0 & f_L & -r_L^1 + r_0^2 \end{bmatrix}$$

where (r_0^1, r_0^2) are center coordinates of the image.

$\Rightarrow$

$$\Psi_L(r_L) R_W^L p^W = -\Psi_L(r_L) d_W^L$$

$$\Psi_R(r_R) R_W^R p^W = -\Psi_R(r_R) d_W^R$$

$\Rightarrow$ collect into one big matrix

$$\underbrace{\begin{bmatrix} \Psi_L(r_L) R^L_W \\ \Psi_R(r_R) R^R_W \end{bmatrix}}_{4\times 3 \text{ matrix}} p^W = - \underbrace{\begin{bmatrix} \Psi_L(r_L) d^L_W \\ \Psi_R(r_R) d^R_W \end{bmatrix}}_{4\times 1 \text{ matrix}}$$

$\Rightarrow$

$$p^W = - \begin{bmatrix} \Psi_L(r_L) R^L_W \\ \Psi_R(r_R) R^R_W \end{bmatrix}^{\dagger} \begin{bmatrix} \Psi_L(r_L) d^L_W \\ \Psi_R(r_R) d^R_W \end{bmatrix}$$

where $A^{\dagger}$ is the pseudo-inverse of A.

$$A^{\dagger} = (A^T A)^{-1} A^T$$

in Matlab, there is the pinv function. $A^{\dagger}$=pinv(A);

or, one can do $x = \cancel{b/A}$ A\b to solve $Ax=b$ for x. (I think)

gives least squares solution when measurements (r_L & r_R) are noisy.

Thus, we get the solution for the point in world coordinates given their projections onto image coordinates, and the camera configurations + projection parameters.

$\downarrow$

really we used $g^L_W \leftrightarrow$ world frame relative to left camera

$g^R_W \leftrightarrow$ world frame relative to right camera often we have the inverse knowledge, g^W_L & g^W_R.

what do we do then?

well...
$$g^L_W = (g^W_L)^{-1} = \left[\begin{array}{c|c} R^W_L & d^W_L \\ \hline 0 & 1 \end{array}\right]^{-1}$$

it is known that
$$g^{-1} = \left[\begin{array}{c|c} R & d \\ \hline 0 & 1 \end{array}\right]^{-1} = \left[\begin{array}{c|c} R^T & -R^T d \\ \hline 0 & 1 \end{array}\right]$$

$\Rightarrow$

$$g^L_W = \left[\begin{array}{c|c} (R^W_L)^T & -(R^W_L)^T d^W_L \\ \hline 0 & 1 \end{array}\right]$$

$$g^R_W = \left[\begin{array}{c|c} (R^W_R)^T & -(R^W_R)^T d^W_R \\ \hline 0 & 1 \end{array}\right]$$

$\Rightarrow$

$$\begin{bmatrix} \Phi_L(r_L)(R^W_L)^T \\ \Phi_R(r_R)(R^W_R)^T \end{bmatrix} p^W = - \begin{bmatrix} \Phi_L(r_L)\left[-(R^W_L)^T d^W_L\right] \\ \Phi_R(r_R)\left[-(R^W_R)^T d^W_R\right] \end{bmatrix}$$

CANCEL.

$\Rightarrow$

$$\begin{bmatrix} \Phi_L(r_L)(R^W_L)^T \\ \Phi_R(r_R)(R^W_R)^T \end{bmatrix} p^W = \begin{bmatrix} \Phi_L(r_L)(R^W_L)^T d^W_L \\ \Phi_R(r_R)(R^W_R)^T d^W_R \end{bmatrix}$$

looks similar!!!

let $M_L(r_L) = \Psi_L(r_L)(R^W_L)^T$

$M_R(r_R) = \Psi_R(r_R)(R^W_R)^T$

$\Rightarrow$

$$\begin{bmatrix} M_L(r_L) \\ M_R(r_R) \end{bmatrix} p^W = \begin{Bmatrix} M_L(r_L)\, d^W_L \\ M_R(r_R)\, d^W_R \end{Bmatrix}$$

$\Rightarrow$

$$\begin{bmatrix} M_L(r_L) \\ M_R(r_R) \end{bmatrix} p^W = \begin{bmatrix} M_L(r_L) & 0 \\ 0 & M_R(r_R) \end{bmatrix} \begin{Bmatrix} d^W_L \\ d^W_R \end{Bmatrix}$$

$\Rightarrow$

$$p^W = \begin{bmatrix} M_L(r_L) \\ M_R(r_R) \end{bmatrix}^{\dagger} \begin{bmatrix} M_L(r_L) & 0 \\ 0 & M_R(r_R) \end{bmatrix} \begin{Bmatrix} d^W_L \\ d^W_R \end{Bmatrix}$$

- and this is the general case solution given the left & right camera configurations w/respect to the world frame.

- it differs from most computer vision texts because they use mixed frame coordinates. I have no idea why. the approach uses R^L_W and d^W_L.

  I think it might be to avoid having transposes everywhere. and it makes camera calibration easier (i.e., linear)
  I usually see that the mixed coordinates confuses some people.

SIDE NOTE :

- why did we keep all 4 equations in Equation (1) when only ~~two~~ 3 are needed?

* there are two reasons

1) we don't know which 3 are important.
consider the first stereo setup w/horizontal displacement by b, and also a second setup w/ vertical displacent by b:

HORIZONTAL BASELINE :

$$r_L = \left(f \frac{x+b/2}{z}, \underbrace{f \frac{y}{z}}_{\text{SAME}} \right)$$

$$r_R = \left(f \frac{x-b/2}{z}, f \frac{y}{z} \right)$$

VERTICAL BASELINE :

$$r_u = \left(\underbrace{f \frac{x}{z}}_{\text{SAME}}, f \frac{y-b/2}{z} \right)$$

$$r_L = \left(f \frac{x}{z}, f \frac{y+b/2}{z} \right)$$

upper camera

lower camera

for horizontal baseline equations (1b) & (1d) are the same

for vertical baseline equations (1a) & (1c) are the same.

in general, by proper manipulation of world coordinate frame (or choice of stereo baseline + camera rotation) any pair of the 4 equations can be made equal!
thus it is safer to keep all 4 since one can't know which pair will be redundant.

2) the second reason is a bit practical.

in practice, there is noise in one's measurement of r_L & r_R. then error can lead to no solutions if only 3 equations (the correct ~~ones~~ ones!) are chosen.

using all 4 requires the pseudo-inverse.
the pseudo-inverse provides the least squares solution: gives best fit solution to data that minimizes

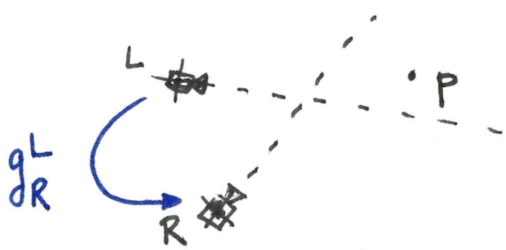
$$\| p^W - p^W_{correct} \|^2$$

so, if solution is going to be wrong due to noise, at least it should be the least wrong possible!

II] Computing Relative Depth

The previous math gave the point w/ respect to world coordinates. What if I just want the point location relative to one of the camera frames? (say the left camera)

- such a question is equivalent to placing the world frame at the same location as the left camera frame.



the point p is located at p^L relative to the left camera (or q^L in homogeneous coordinates).

and, with respect to the right camera

$$q^R = g^R_L q^L = (g^L_R)^{-1} q^L$$

so, if we want to solve for the point position relative to the left camera, we can just use the previous equations with

$$g^W_L = \mathbb{1}_{4\times4} \quad \text{and} \quad g^W_R = g^W_L g^L_R = \mathbb{1}_{4\times4}\, g^L_R = g^L_R$$

$\Rightarrow$

$$M_L(r_L) = \Psi_L(r_L) \qquad \leftarrow \text{since } R^L_L = \mathbb{1}_{3\times3}$$

$$M_R(r_R) = \Psi_R(r_R)\,(R^L_R)^T$$

since $d_L^L = 0$

$\Rightarrow$

$$\begin{bmatrix} M_L(r_L) \\ M_R(r_R) \end{bmatrix} p^L = \begin{bmatrix} M_L(r_L) & 0 \\ 0 & M_R(r_R) \end{bmatrix} \begin{Bmatrix} 0 \\ d_R^L \end{Bmatrix}$$

$\Rightarrow$ right hand-side has zero part in the vector

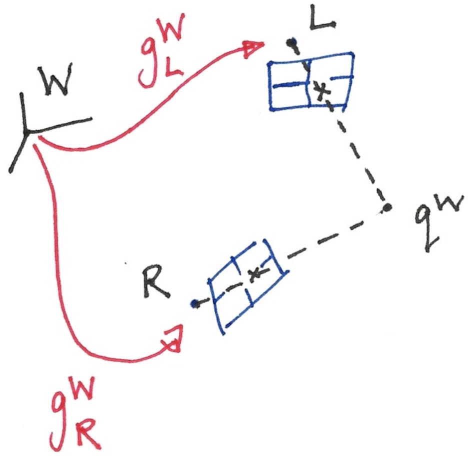
$$\begin{bmatrix} M_L(r_L) \\ M_R(r_R) \end{bmatrix} p^L = \begin{bmatrix} 0 \\ M_R(r_R) \end{bmatrix} d_R^L$$

$\Rightarrow$

$$p^L = \begin{bmatrix} M_L(r_L) \\ M_R(r_R) \end{bmatrix}^{\dagger} \begin{bmatrix} 0 \\ M_R(r_R) \end{bmatrix} d_R^L$$

- has a much simpler form since only one g was needed (g_R^L).

III] Stereo Cameras & 3D Point Recovery



Suppose that global information was known, which means all known quantities in world coordinates with projections given in proper image frame (Left or Right).

KNOWNS:

$$g_L^W = \left[\begin{array}{c|c} R_L^W & T_{WL}^W \\ \hline 0 & 1 \end{array}\right] \qquad g_R^W = \left[\begin{array}{c|c} R_R^W & T_{WR}^W \\ \hline 0 & 1 \end{array}\right]$$

CAMERA FRAMES IN WORLD FRAME

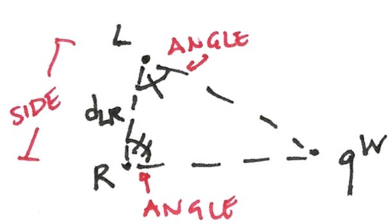
$$\vec{r}^{\,L} = \begin{bmatrix} (r^L)^1 \\ (r^L)^2 \\ 1 \end{bmatrix} \qquad \vec{r}^{\,R} = \begin{bmatrix} (r^R)^1 \\ (r^R)^2 \\ 1 \end{bmatrix}$$

LEFT & RIGHT

IMAGE PROJECTION POINTS OF WORLD POINT q^W

UNKNOWN: LOCATION OF WORLD POINT q^W.

WE KNOW THAT SOLVEABLE DUE TO ANGLE-SIDE-ANGLE,



BUT HOW TO SOLVE MORE DIRECTLY USING ALGEBRA IS THE QUESTION?

... WELL, WE HAVE TO SET UP A SYSTEM OF EQUATIONS TO RECOVER THE UNKNOWN QUANTITY AS A FUNCTION OF THE KNOWNS!

LET'S START WITH THE PROJECTION EQUATIONS ...

$$\vec{r}^{\,L} \approx \Psi_L [R^L_W \mid T^L_{LW}]\, q^W$$

$$\vec{r}^{\,R} \approx \Psi_R [R^R_W \mid T^R_{RW}]\, q^W$$

means unknown up to scale.
the scale is the unknown depth.
(normally the depths would be written as z^L & z^R since the depth is the z-component in the left/right camera frame. However, I like to use α_L & α_R as the scale factors)

ASSUME WE "KNEW" THE SCALE FACTORS α_L & α_R. THEN

$$\left.\begin{aligned} \alpha_L \vec{r}^{\,L} &= \Psi_L [R^L_W \mid T^L_{LW}]\, q^W \\ \alpha_R \vec{r}^{\,R} &= \Psi_R [R^R_W \mid T^R_{RW}]\, q^W \end{aligned}\right\} \text{units are pixels}$$

$\Rightarrow$ rearrange so that both sides in "world units".

$$\left.\begin{aligned} \alpha_L \Psi_L^{-1} \vec{r}^{\,L} &= [R^L_W \mid T^L_{LW}]\, q^W \\ \alpha_R \Psi_R^{-1} \vec{r}^{\,R} &= [R^R_W \mid T^R_{RW}]\, q^W \end{aligned}\right\} \text{units are "world units"}$$

meter
cm
feet
mm
⋮

NOW, RECALL THAT q^W IS HOMOGENEOUS FORM AND LOOKS LIKE

$$q^W = \left[\begin{array}{c} p^W \\ \hline 1 \end{array}\right] = \left[\begin{array}{c} p^1 \\ p^2 \\ p^3 \\ \hline 1 \end{array}\right]^W$$

SO, SUBSTITUTING IN THE HOMOGENEOUS PART AND MULTIPLYING THROUGH GIVES:

$$\alpha_L \Psi_L^{-1} \vec{r}^{\,L} = R_W^L p^W + T_{LW}^L$$

$$\alpha_R \Psi_R^{-1} \vec{r}^{\,R} = R_W^R p^W + T_{RW}^R$$

UNKNOWNS

UNKNOWN

SOLVING FOR p^W IN BOTH EQUATIONS GIVES

$$p^W = (R_W^L)^{-1} \left[\alpha_L \Psi_L^{-1} \vec{r}^{\,L} - T_{LW}^L \right]$$

$$p^W = (R_W^R)^{-1} \left[\alpha_R \Psi_R^{-1} \vec{r}^{\,R} - T_{RW}^R \right]$$

WE KNOW THAT

$$g_B^A = \left[\begin{array}{c|c} R_B^A & T_{AB}^A \\ \hline 0 & 1 \end{array}\right]$$

HAS INVERSE

$$(g_B^A)^{-1} = \left[\begin{array}{c|c} (R_B^A)^{-1} & -(R_B^A)^{-1} T_{AB}^A \\ \hline 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} (R_B^A)^T & -(R_B^A)^T T_{AB}^A \\ \hline 0 & 1 \end{array}\right]$$

$$= \left[\begin{array}{c|c} R_A^B & T_{BA}^B \\ \hline 0 & 1 \end{array}\right] = g_A^B$$

MEANING THAT

$$p^W = \underbrace{R^W_L}_{} \left[\alpha_L \Phi_L^{-1} \vec{r}^{\,L} - T^L_{LW} \right]$$

$$p^W = \underbrace{R^W_R}_{\text{DUE TO INVERSE}} \left[\alpha_R \Phi_R^{-1} \vec{r}^{\,R} - T^R_{RW} \right]$$

$\Rightarrow$

$$p^W = \alpha_L R^W_L \Phi_L^{-1} \vec{r}^{\,L} - \underbrace{R^W_L T^L_{LW}}_{}$$

$$p^W = \alpha_R R^W_R \Phi_R^{-1} \vec{r}^{\,R} - \underbrace{R^W_R T^R_{RW}}_{\text{AGAIN EXPLOIT INVERSE}}$$

(YES, THE SIGN CHANGE IS CORRECT BECAUSE THE BOTTOM LETTERS CHANGE ORDER)

$\Rightarrow$

$$p^W = \alpha_L R^W_L \Phi_L^{-1} \vec{r}^{\,L} + T^W_{WL}$$

$$p^W = \alpha_R R^W_R \Phi_R^{-1} \vec{r}^{\,R} + T^W_{WR}$$

NOTE! OUR g^W_L & g^W_R GIVE US EXACTLY THESE R & T MATRICES & VECTORS!

$\Rightarrow$ SET EQUAL

$$\underbrace{\alpha_L}_{\text{UNKNOWN}} R^W_L \Phi_L^{-1} \vec{r}^{\,L} + T^W_{WL} = \underbrace{\alpha_R}_{\text{UNKNOWN}} R^W_R \Phi_R^{-1} \vec{r}^{\,R} + T^W_{WR}$$

2 UNKNOWNS , 3 EQUATIONS

(ONE IS SECRETLY REDUNDANT, BUT WE KEEP ANYHOW)

α_L & α_R ARE SOLVEABLE. ONCE ISOLATED, THE SOLUTION POPS OUT. SO LET'S CREATE A MATRIX EQUATION TO SOLVE FOR α_L & α_R. FIRST ISOLATE . . .

$$\alpha_L R_L^W \Psi_L^{-1} \vec{r}^{\,L} - \alpha_R R_R^W \Psi_R^{-1} \vec{r}^{\,R} = T_{WR}^W - T_{WL}^W$$

$$\underbrace{R_L^W \Psi_L^{-1} \vec{r}^{\,L}}_{\text{VECTOR}} \cdot \underbrace{\alpha_L}_{\text{SCALAR}} - \underbrace{R_R^W \Psi_R^{-1} \vec{r}^{\,R}}_{\text{VECTOR}} \cdot \underbrace{\alpha_R}_{\text{SCALAR}} = \underbrace{T_{WR}^W - T_{WL}^W}_{\text{VECTOR}}$$

RECALL: DIFFERENCE OF TWO POINTS IS A VECTOR.

LINEAR IN α_L & α_R MEANS THAT WE CAN WRITE AS

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

TO DO SO, FACTOR OUT $\vec{\alpha} = \begin{bmatrix} \alpha_L \\ \alpha_R \end{bmatrix}$

$$\Rightarrow \begin{bmatrix} | & | \\ R_L^W \Psi_L^{-1} \vec{r}^{\,L} & -R_R^W \Psi_R^{-1} \vec{r}^{\,R} \\ | & | \end{bmatrix} \begin{bmatrix} \alpha_L \\ \alpha_R \end{bmatrix} = T_{WR}^W - T_{WL}^W$$

$$[3 \times 2] \cdot [2 \times 1] = [3 \times 1] \quad \checkmark$$

WE GET AN OVERDETERMINED SYSTEM.
MATLAB CAN SOLVE IN TWO WAYS, BUT FIRST LET'S FINISH UP THEN DISCUSS . . .

THE SOLUTION USES WHAT'S CALLED THE PSEUDO-INVERSE.

IT IS AN INVERSE FOR OVERDETERMINED SYSTEMS OF EQUATIONS.

LET

$$A = \begin{bmatrix} | & | \\ R^W_L \Psi^{-1}_L \vec{r}^{\,L} & -R^W_R \Psi^{-1}_R \vec{r}^{\,R} \\ | & | \end{bmatrix}$$

DON'T FORGET THE NEGATIVE SIGN!!!

AND

$$\vec{b} = T^W_{WR} - T^W_{WL} = \vec{V}^W_{RL}$$

VECTOR BETWEEN POINT R AND POINT L IN THE WORLD FRAME.

THIS IS THE "SIDE" IN THE ASA SOLUTION (WELL, IF WE GET ITS LENGTH).

$$d_{RL} = d_{LR} = \| \vec{V}^W_{RL} \|$$

FROM BEGINNING.

SIDE NOTE: RECALL THAT RAYS PROVIDE DIRECTION BUT NOT LENGTH.

SO THAT MEANS THE RAYS ARE IMPLICITLY ENCODING THE ANGLE/ANGLE PART OF THE ASA SOLUTION. OUR ALGEBRA BEARS IT OUT IN A WAY.

THEN

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

SO

$$\vec{\alpha} = A^{+} \vec{b}$$

PSEUDO-INVERSE.

IN MATLAB,

$$\vec{\alpha} = \mathrm{pinv}(A) \cdot \vec{b};$$

OR

$$\vec{\alpha} = A \backslash \vec{b};$$

OR

$$\vec{\alpha} = \mathrm{mldivide}(A, \vec{b});$$

GREAT, ONCE WE GET $\vec{\alpha} = \begin{bmatrix} \alpha_L \\ \alpha_R \end{bmatrix}$, WE HAVE THE DEPTH IN THE LEFT & RIGHT CAMERAS. WHAT ABOUT THE 3D COORDINATES IN THE WORLD FRAME?

GO BACK 3 PAGES TO THE EQUATIONS:

$$P^W = \alpha_L R_L^W \Psi_L^{-1} \vec{r}^L + T_{WL}^W$$

$$P^W = \alpha_R R_R^W \Psi_R^{-1} \vec{r}^R + T_{WR}^W$$

ANY OF THE ABOVE TWO WILL WORK.

OF COURSE THEY WON'T FULLY AGREE DUE TO PIXEL QUANTIZATION, BUT THEY SHOULD BE CLOSE.