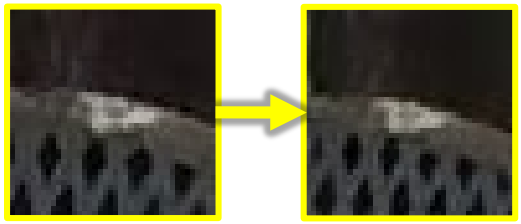
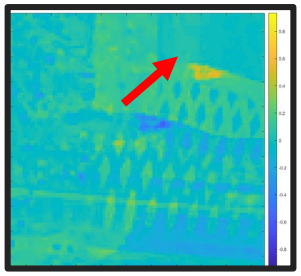
Perception:
Motion Field Equations

Last Time:

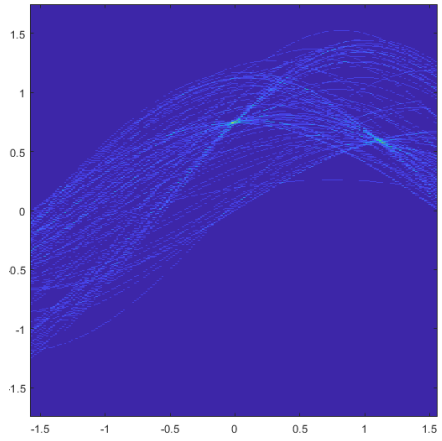
Computing optical flow

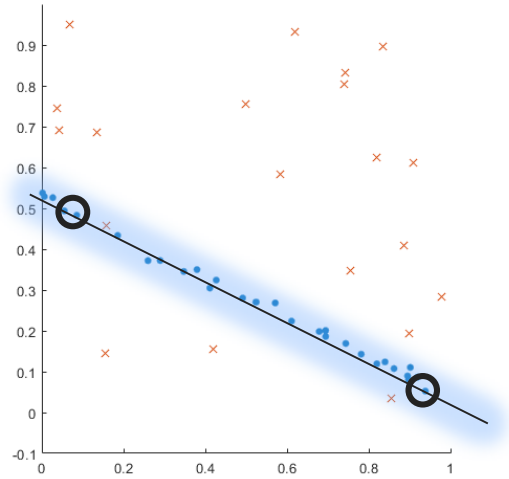
$$\begin{pmatrix}\delta x^* \\ \delta y^*\end{pmatrix} = \left(\sum_{(x,y)} \nabla I_{t+1}(x,y) \nabla I_{t+1}(x,y)^T\right)^{-1} \left(\sum_{(x,y)} \Delta I_t(x,y) \nabla I_{t+1}(x,y)\right)$$





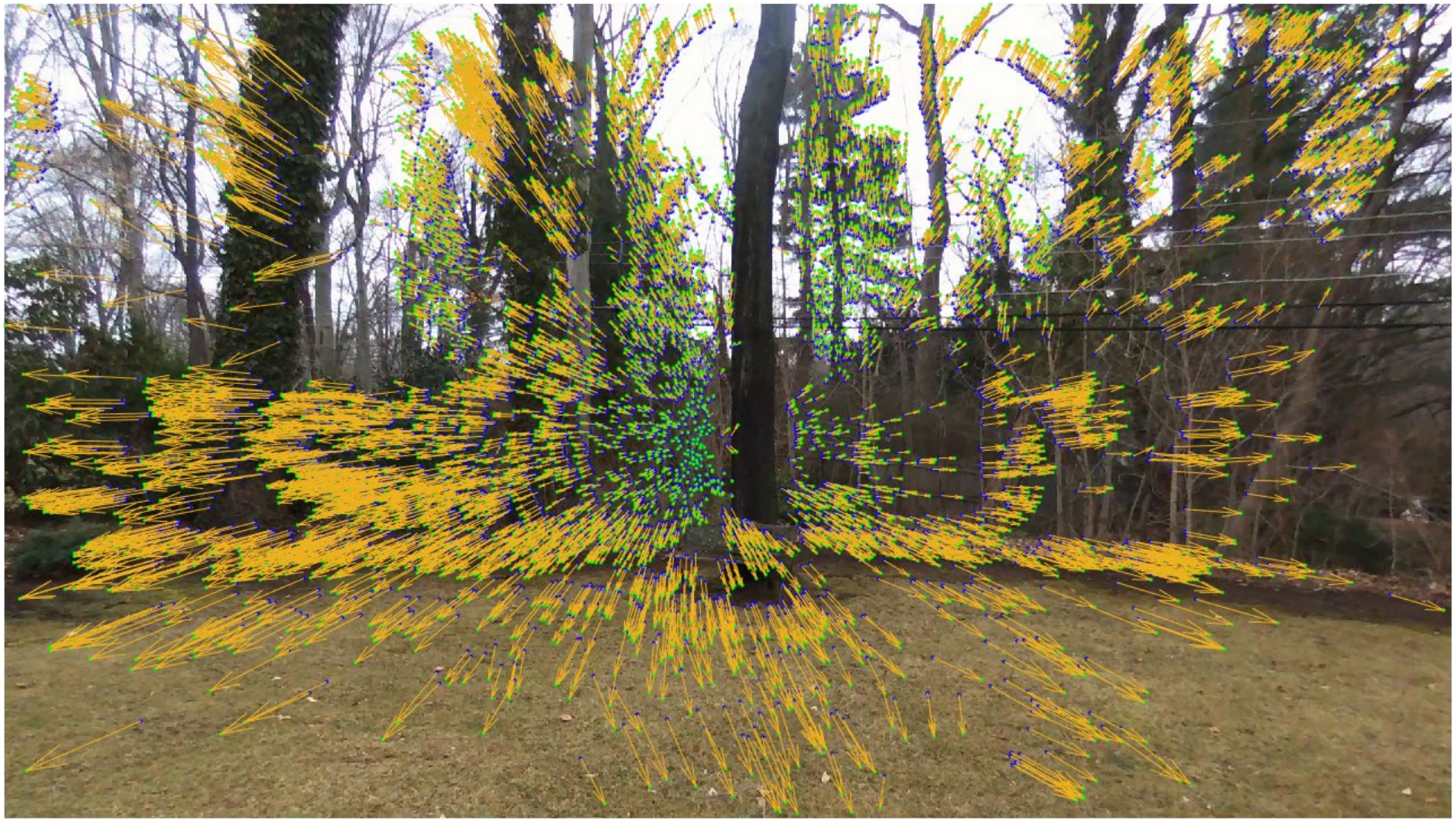
Hough Transform and RANSAC:



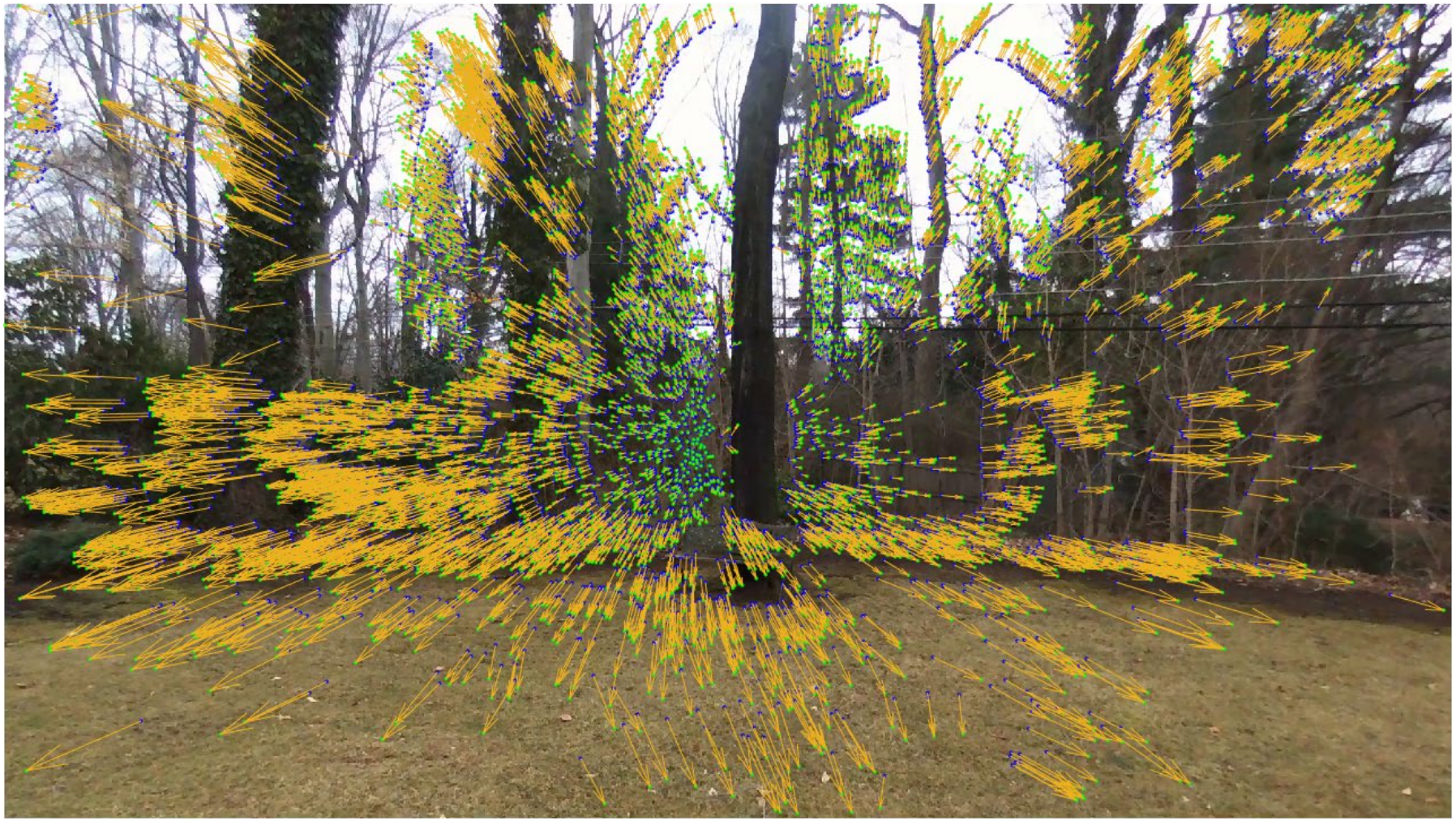


Repeat for k iterations
1. Choose a minimal sample set
2. Count the inliers for this set
3. Keep maximum, if it exceeds a desired number of inliers stop.

Now: Computing 3D Velocity from 2D



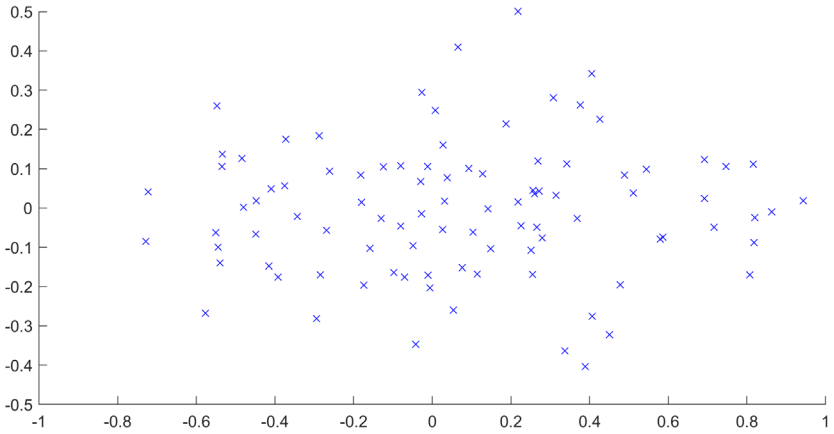
Now: Computing 3D Velocity from 2D

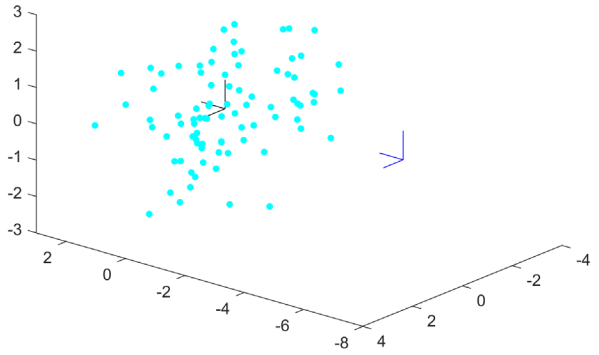


Which direction is the camera going?

Systematic Testing

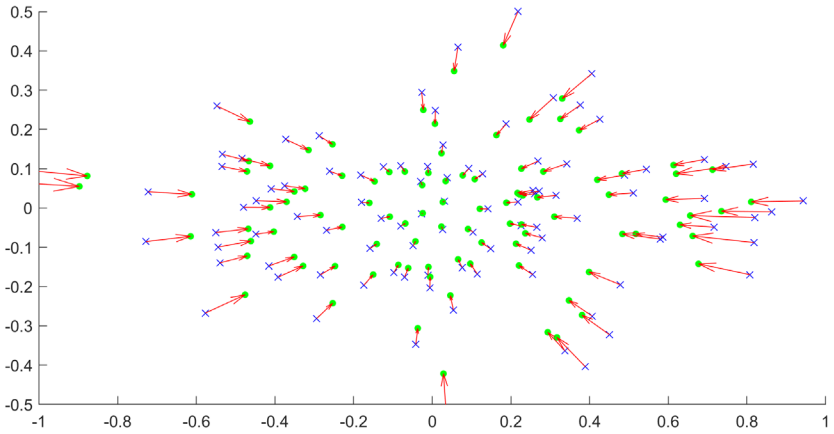
Points from the original view

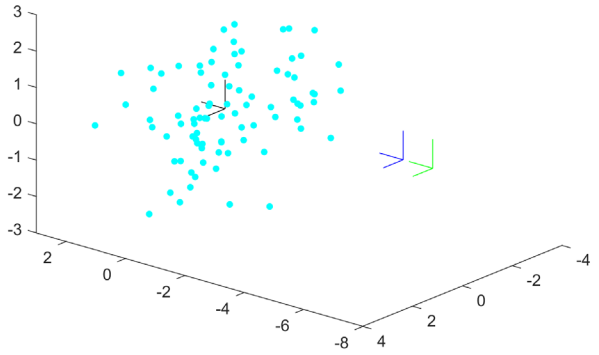




Systematic Testing

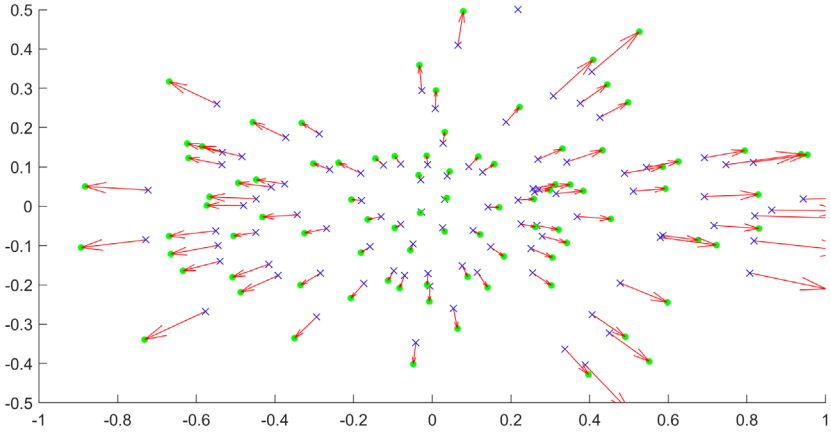
Moving
along z-axis
(optical axis)

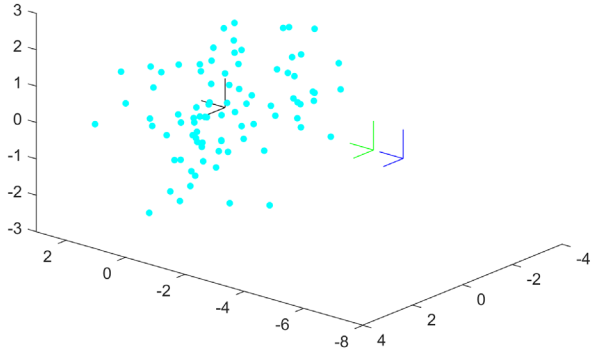




Systematic Testing

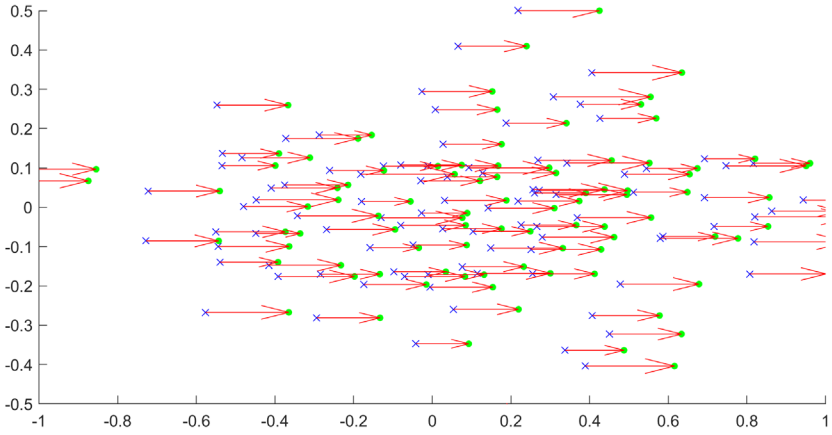
Moving
along z-axis
(optical axis)
Opposite way

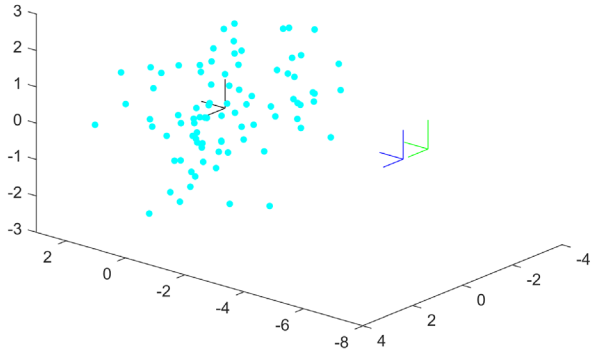




Systematic Testing

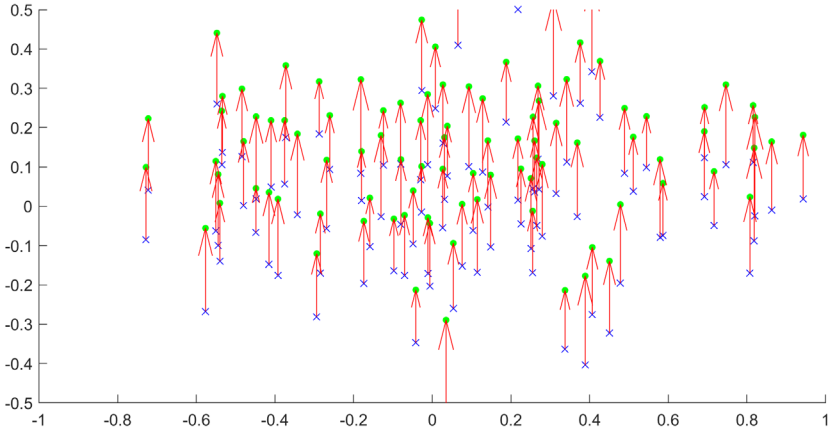
Moving
along x-axis

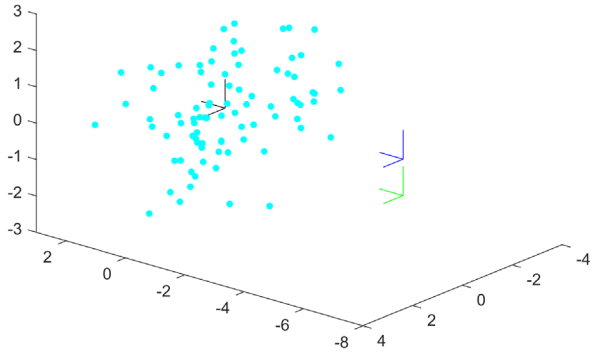




Systematic Testing

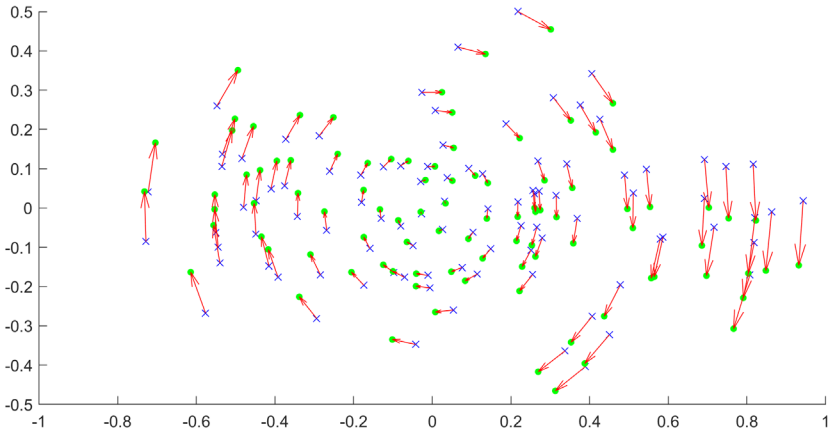
Moving
along y-axis

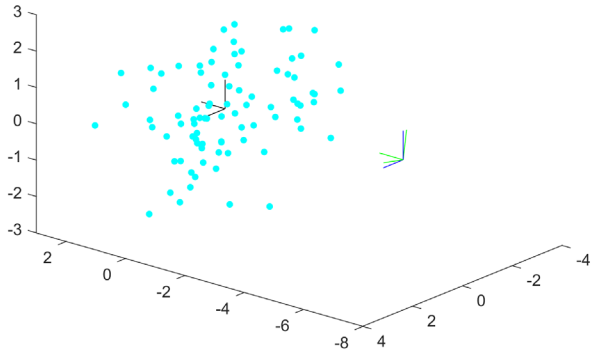




Systematic Testing

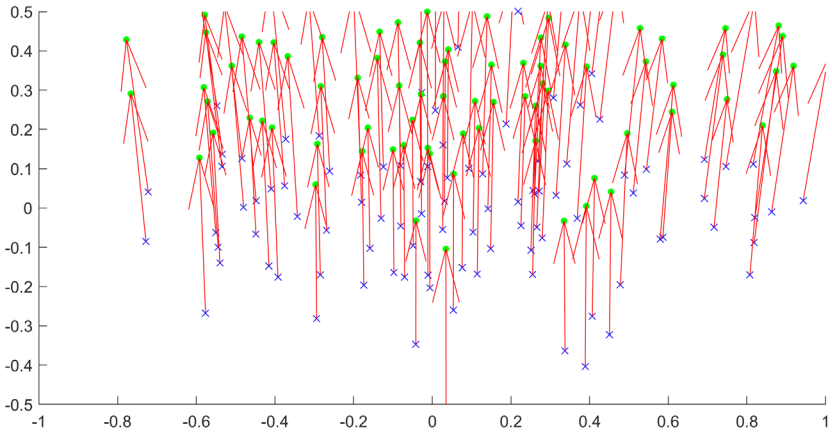
Rotating
along z-axis
(optical axis)

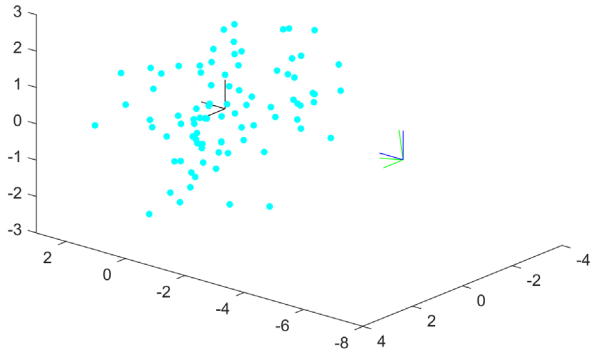




Systematic Testing

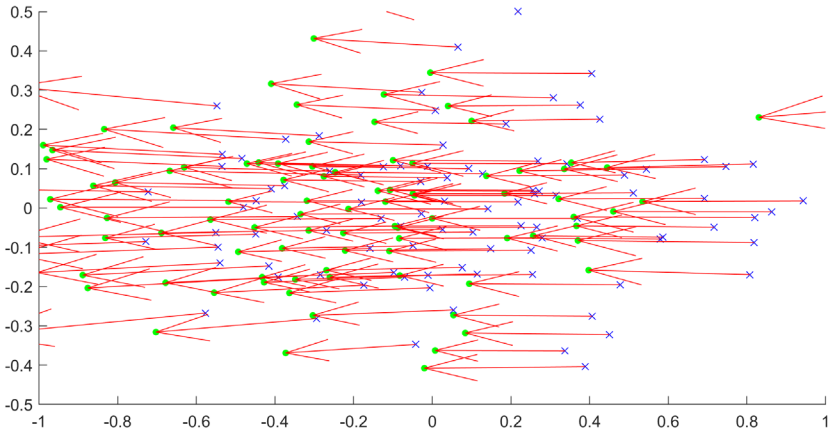
Rotating
along x-axis

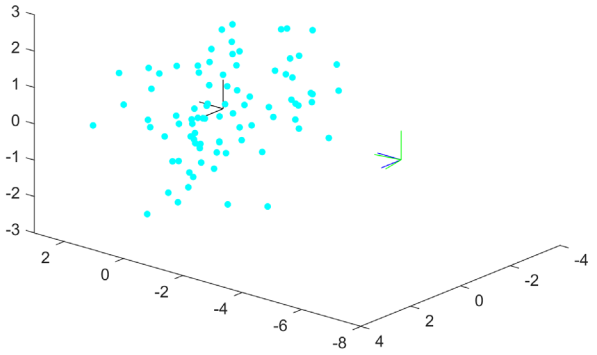




Systematic Testing

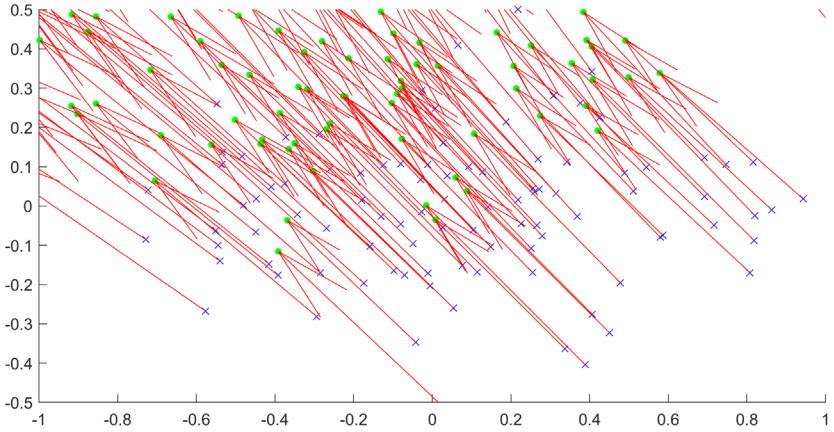
Rotating
along y-axis

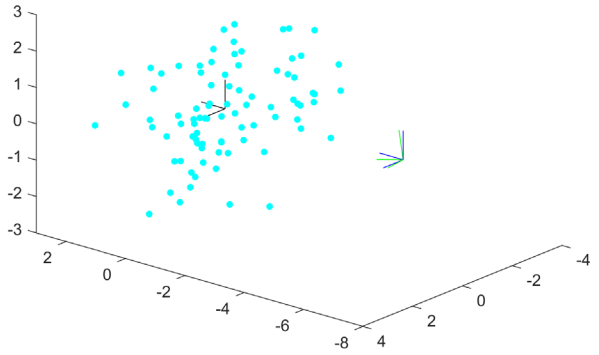




Systematic Testing

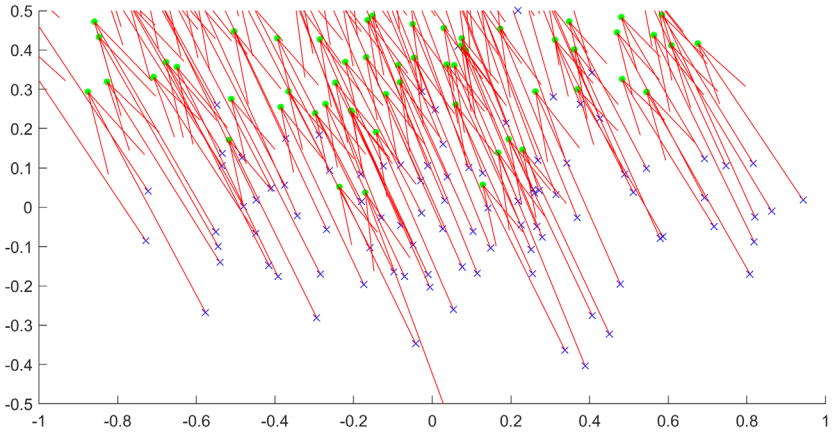
Rotating along x-axis and y-axis

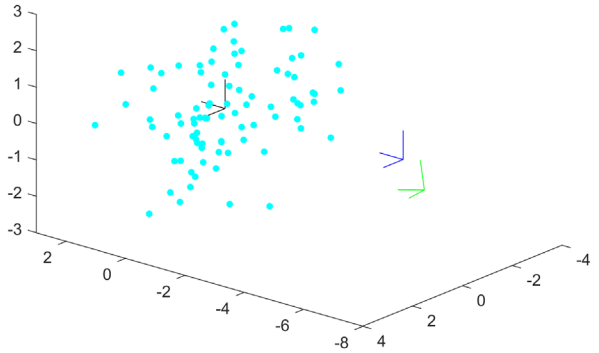




Systematic Testing

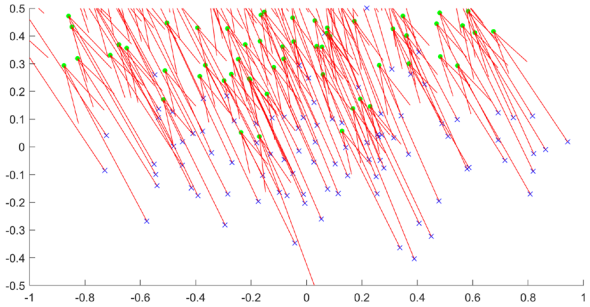
Rotating along x-axis and y-axis, and translating

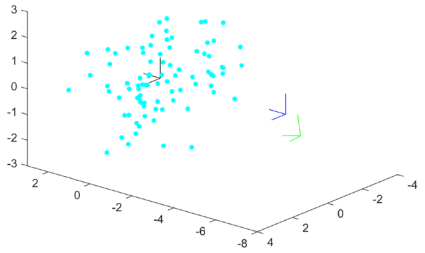




Deriving Motion Field Equations

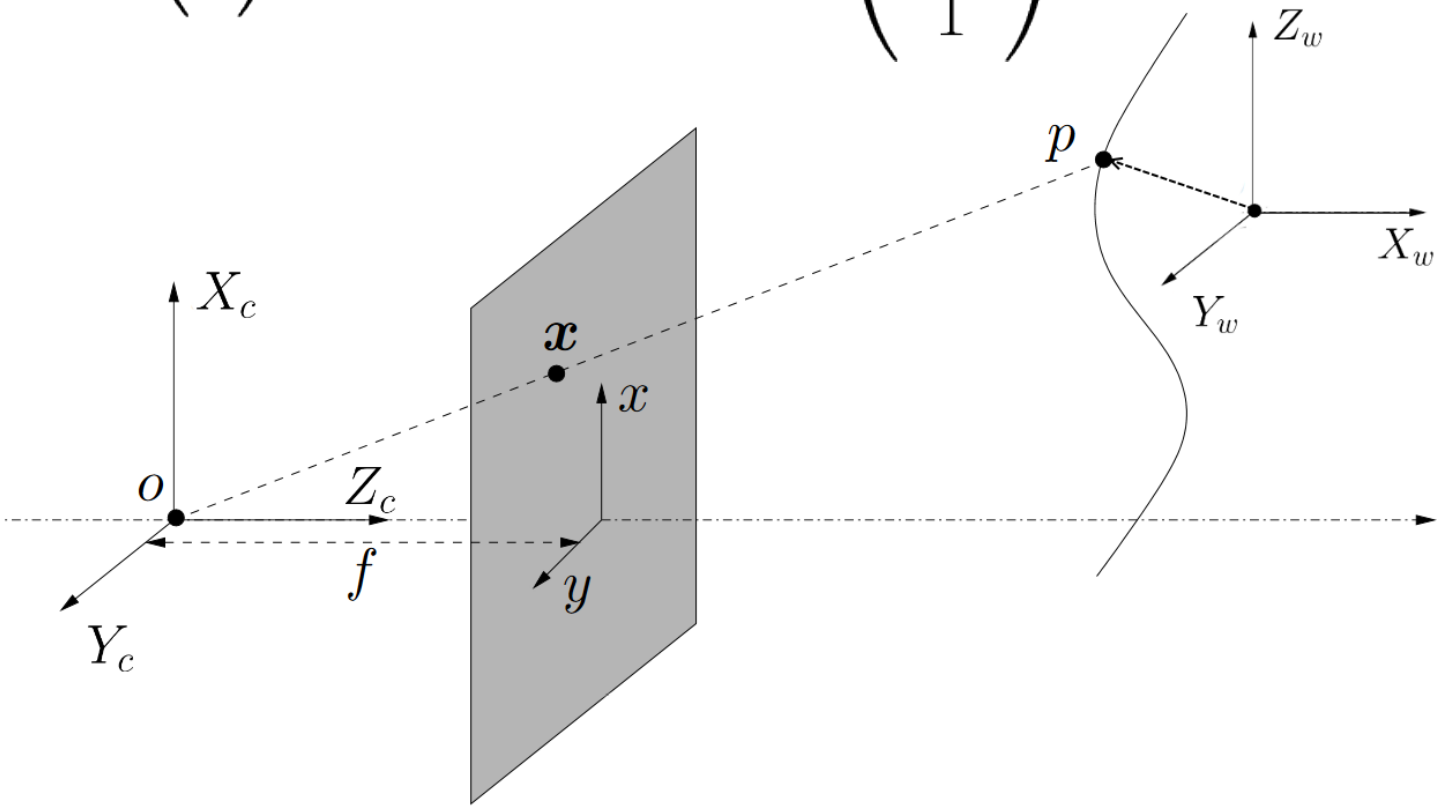
To be able to predict and extract information from these optical flow fields, we need to derive a model of the equations of motion for points on an image





Projection Equations

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



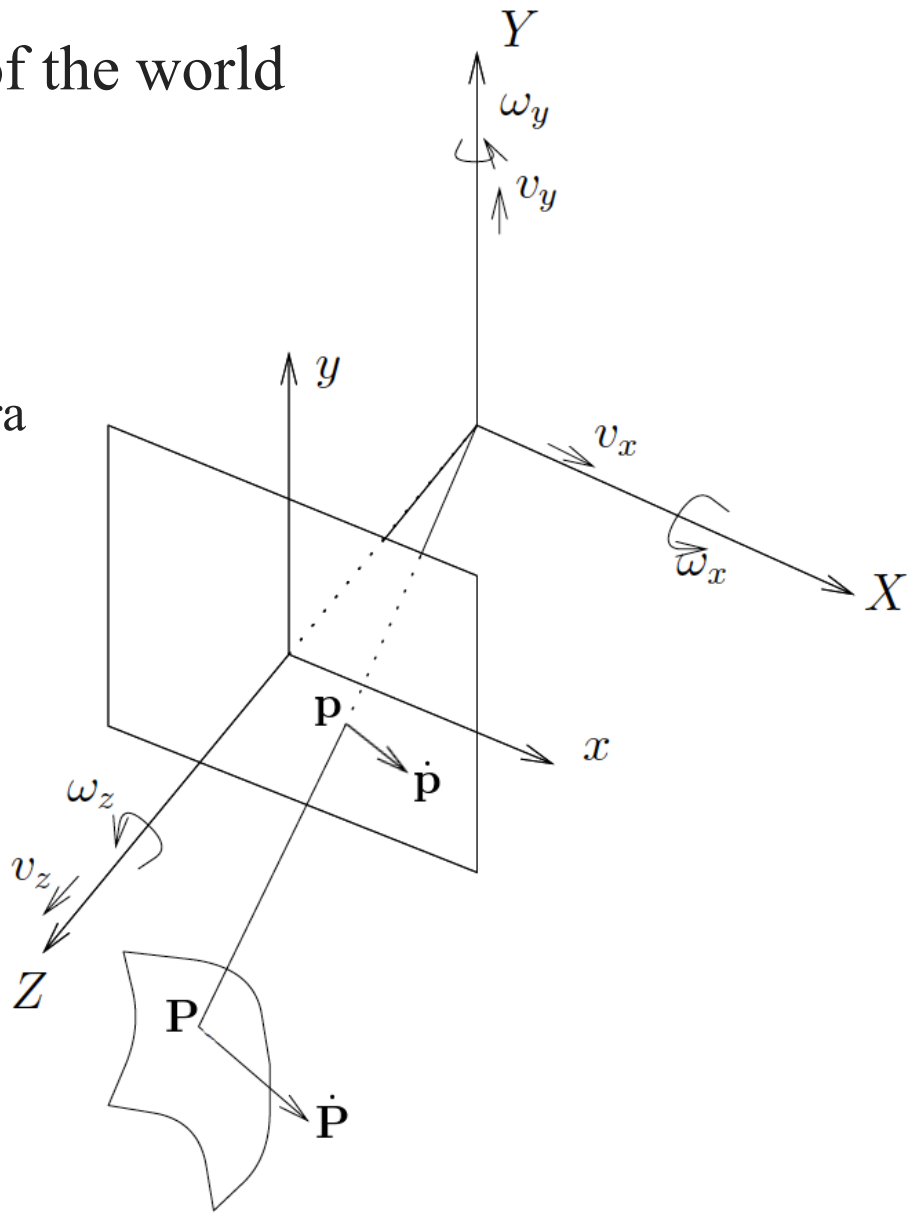
Motion Model

Camera is moving, the rest of the world is not moving i.e. static

$$\mathbf{P} = \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}$$

3D point, in the (rotating) camera frame

$\dot{\mathbf{P}}$ Velocity of the point in the camera frame



Motion Model

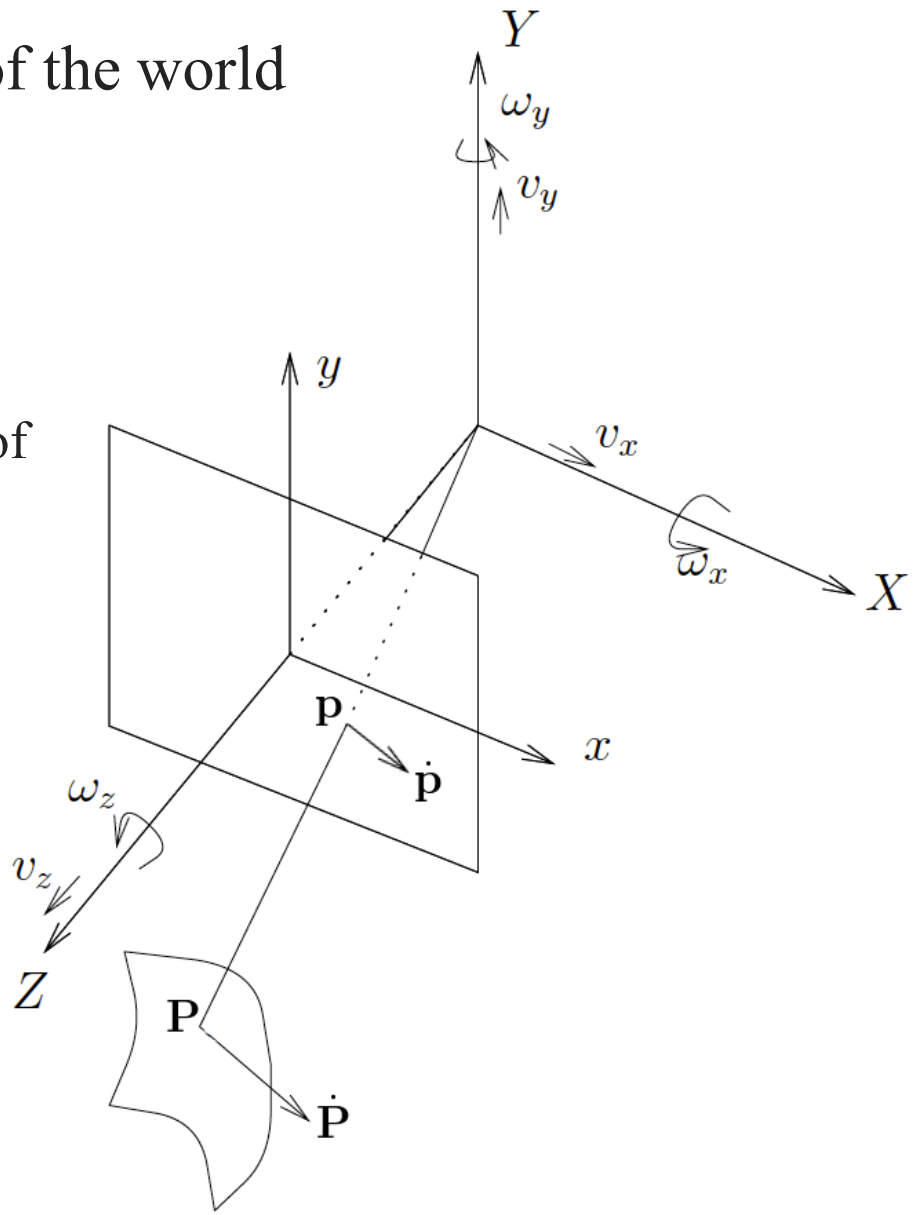
Camera is moving, the rest of the world is not moving i.e. static

$$\mathbf{V} = \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

Translational velocity in a inertial frame of reference

$$\mathbf{\Omega} = \begin{pmatrix} \omega_x \\ \omega_y \\ \omega_z \end{pmatrix}$$

Angular velocity in a inertial frame of reference



Motion Model

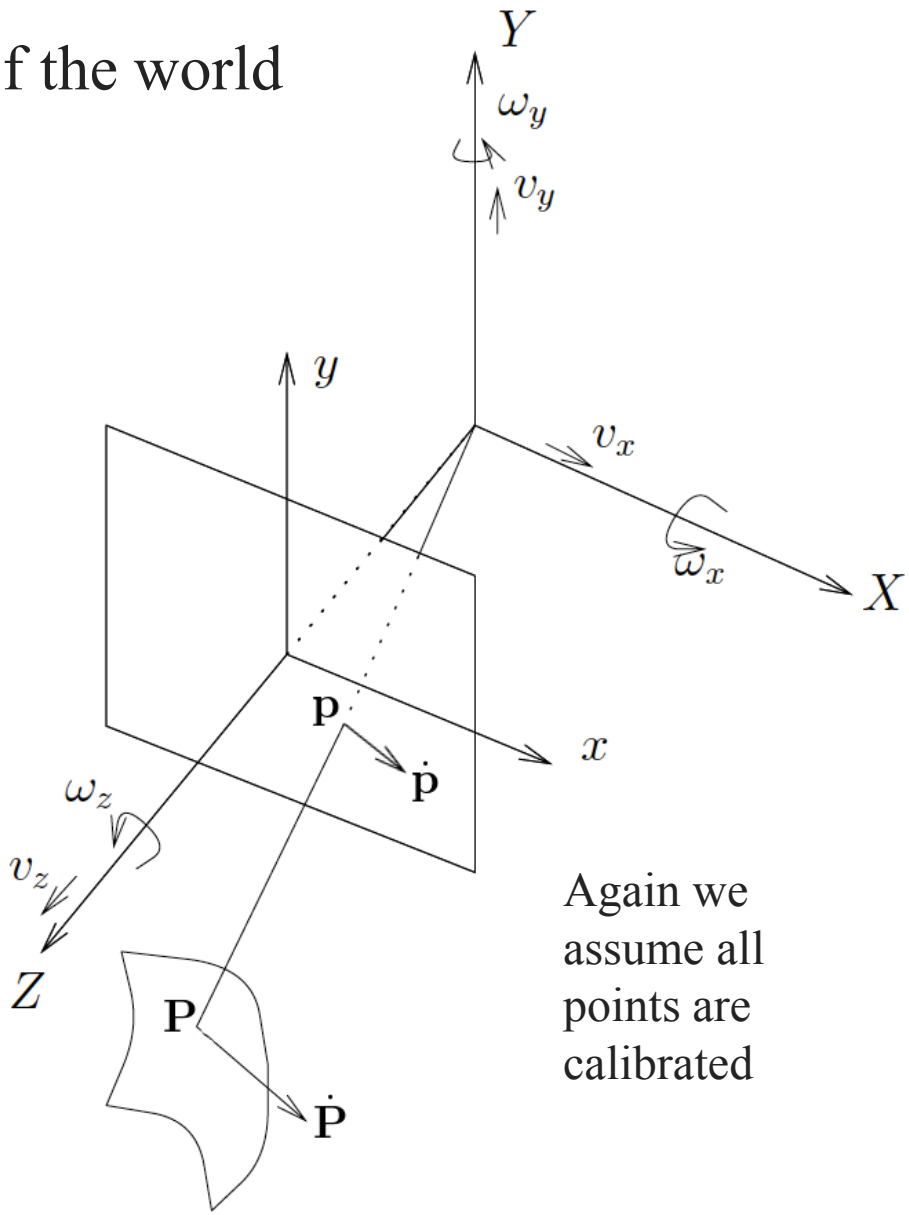
Camera is moving, the rest of the world is not moving i.e. static

$$\mathbf{p} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

2D point on the image

$\dot{\mathbf{p}}$ 2D velocity of the point

$$\mathbf{p} = \frac{\mathbf{P}}{Z}$$



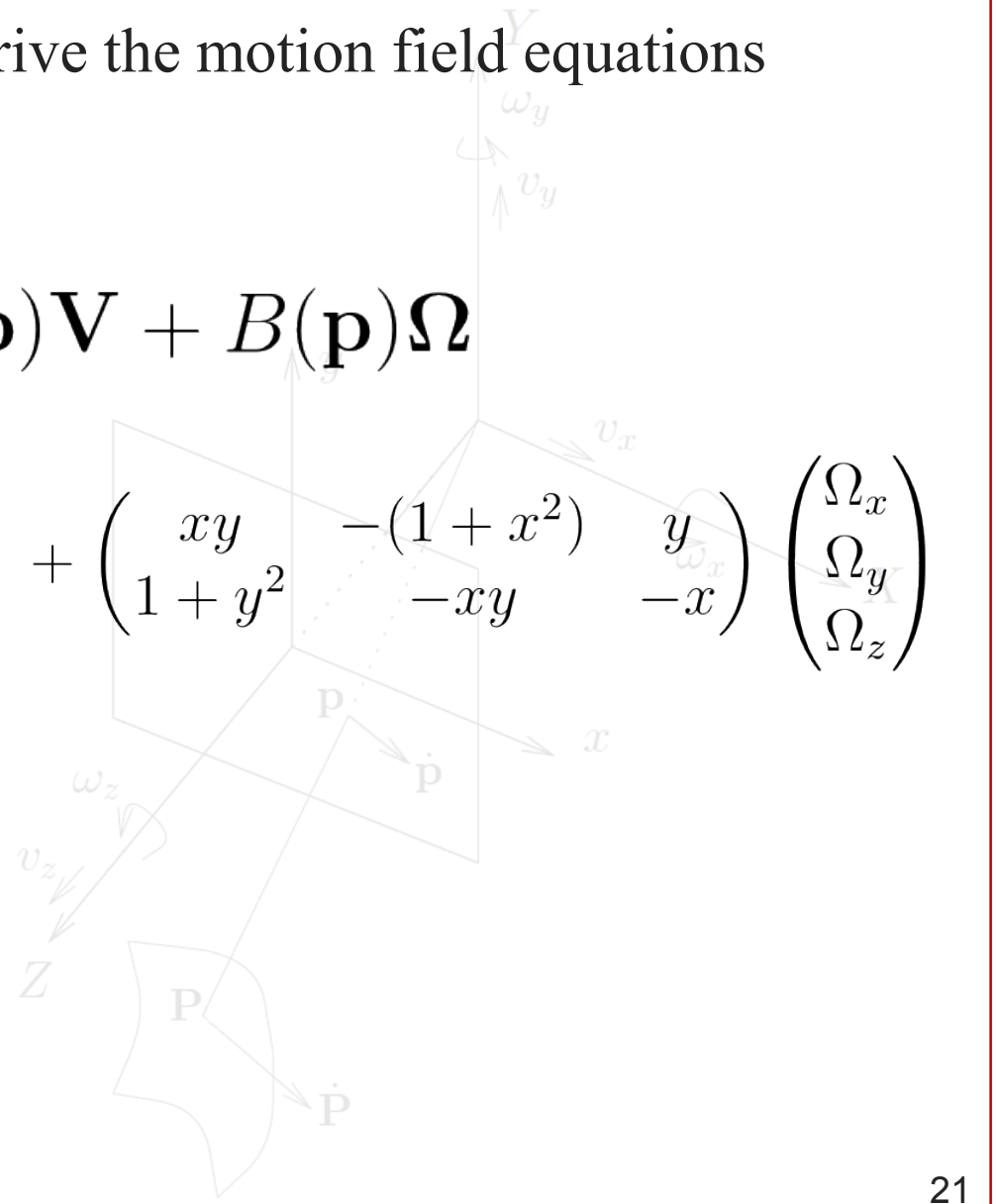
Again we assume all points are calibrated

Motion Field Equations

The punchline is we can derive the motion field equations from these:

$$\dot{\mathbf{p}} = \frac{1}{Z} A(\mathbf{p})\mathbf{V} + B(\mathbf{p})\boldsymbol{\Omega}$$

$$\begin{pmatrix} u \\ v \end{pmatrix} = \frac{1}{Z} \begin{pmatrix} -1 & 0 & x \\ 0 & -1 & y \end{pmatrix} \begin{pmatrix} V_x \\ V_y \\ V_z \end{pmatrix} + \begin{pmatrix} xy & -(1+x^2) & y \\ 1+y^2 & -xy & -x \end{pmatrix} \begin{pmatrix} \Omega_x \\ \Omega_y \\ \Omega_z \end{pmatrix}$$



Motion Field Equations - Derivation

Now to find the velocity on the image i.e. the motion field, we take the derivative of the 2D point:

$$\mathbf{p} = \frac{\mathbf{P}}{Z} \Longrightarrow \dot{\mathbf{p}} = \frac{d}{dt}\frac{\mathbf{P}}{Z}$$

$$\Longrightarrow \dot{\mathbf{p}} = \frac{\dot{\mathbf{P}}}{Z} - \frac{\mathbf{P}\dot{Z}}{Z^2} = \frac{\dot{\mathbf{P}}}{Z} - \frac{\dot{Z}}{Z}\mathbf{p}$$

We rewrite this as:

$$\dot{Z} = e_3^T \dot{\mathbf{P}}$$

With: $e_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$

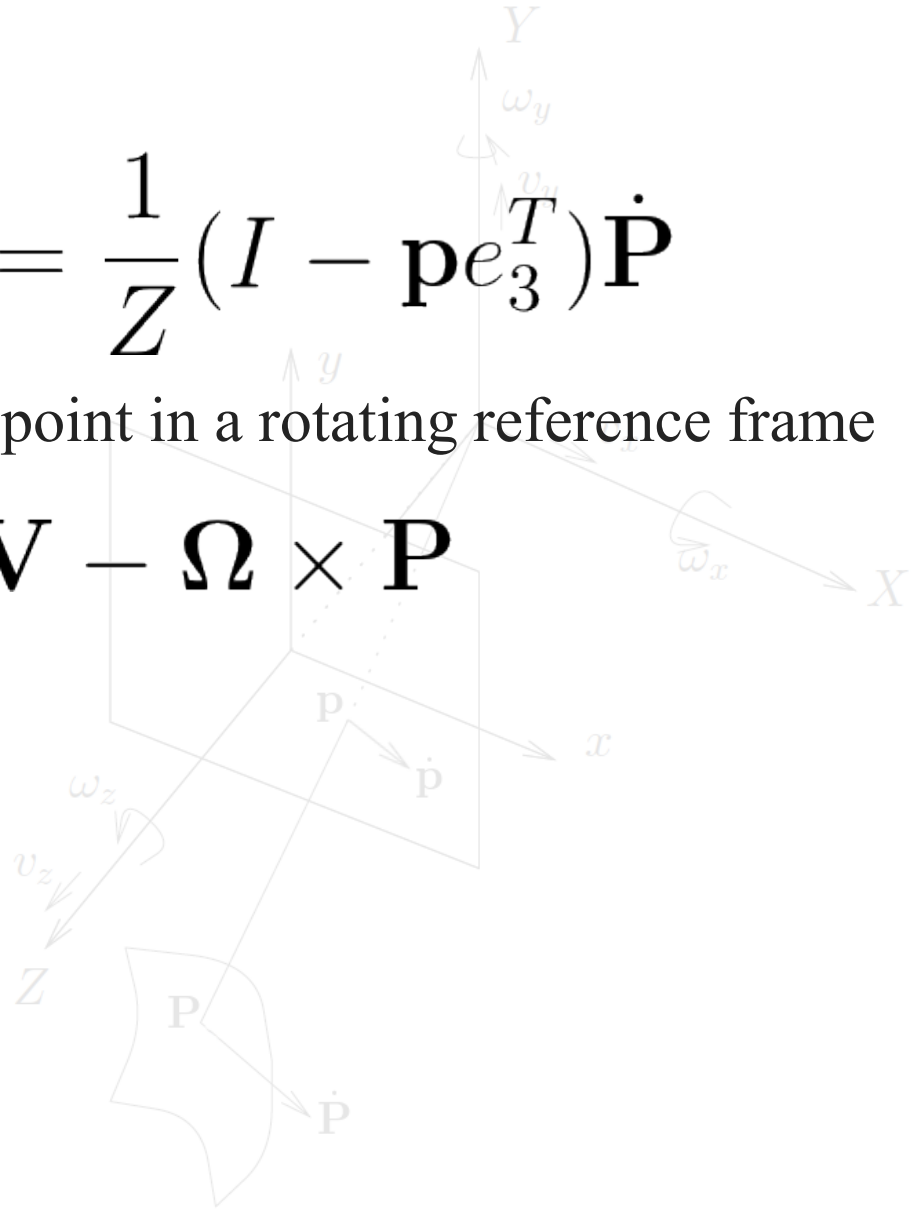
Motion Field Equations - Derivation

Substitute and rearrange:

$$\frac{\dot{\mathbf{P}}}{Z} - \frac{e_3^T \dot{\mathbf{P}}}{Z}\mathbf{p} = \frac{1}{Z}(I - \mathbf{p}e_3^T)\dot{\mathbf{P}}$$

From physics, velocity of a point in a rotating reference frame

$$\dot{\mathbf{P}} = -\mathbf{V} - \boldsymbol{\Omega} \times \mathbf{P}$$



Motion Field Equations - Derivation

Substitute and rearrange:

Rewrite this as: $-[\mathbf{P}]_\times \boldsymbol{\Omega}$

$$\frac{1}{Z}(I - \mathbf{p}e_3^T)\,(-\mathbf{V} - \boldsymbol{\Omega} \times \mathbf{P})$$

$$= -\frac{1}{Z}(I - \mathbf{p}e_3^T)\mathbf{V} + (I - \mathbf{p}e_3^T)\left[\frac{\mathbf{P}}{Z}\right]_\times \boldsymbol{\Omega}$$

$$= \frac{1}{Z}\underbrace{(\mathbf{p}e_3^T - I)}_{\mathbf{A}(\mathbf{p})}\mathbf{V} + \underbrace{(I - \mathbf{p}e_3^T)[\mathbf{p}]_\times}_{\mathbf{B}(\mathbf{p})}\boldsymbol{\Omega}$$

Solving the Motion Field Equations

We now know the form of the Motion Field Equations:

$$\dot{\mathbf{p}} = \frac{1}{Z} A(\mathbf{p})\mathbf{V} + B(\mathbf{p})\boldsymbol{\Omega}$$

How do we solve it

- Case 1: Known depth – easy, simply use least squares
- Case 2: No translational velocity – also least squares
- Case 3: No angular velocity – slightly harder, but a form of least squares
- Case 4: Both translational and angular velocity with unknown depth – fairly difficult

Case 1: Known Depth

With known depth the equations reduce to:

$$\dot{\mathbf{p}} = \frac{1}{Z} A(\mathbf{p})\mathbf{V} + B(\mathbf{p})\boldsymbol{\Omega} = \begin{pmatrix} \frac{1}{Z} A(\mathbf{p}) & B(\mathbf{p}) \end{pmatrix} \begin{pmatrix} \mathbf{V} \\ \boldsymbol{\Omega} \end{pmatrix}$$

And this can be solved with a least squares problem:

$$\mathbf{V}^*, \boldsymbol{\Omega}^* = \arg\min_{\mathbf{V},\boldsymbol{\Omega}} \sum_{i=1}^{n} \left\| \begin{pmatrix} \frac{1}{Z_i} A(\mathbf{p}_i) & B(\mathbf{p}_i) \end{pmatrix} \begin{pmatrix} \mathbf{V} \\ \boldsymbol{\Omega} \end{pmatrix} - \dot{\mathbf{p}}_i \right\|^2$$

THIS IS WHAT YOU WILL USE IN THE PROJECT
You will need to run RANSAC to make this robust.
How many points minimum do you need for this?

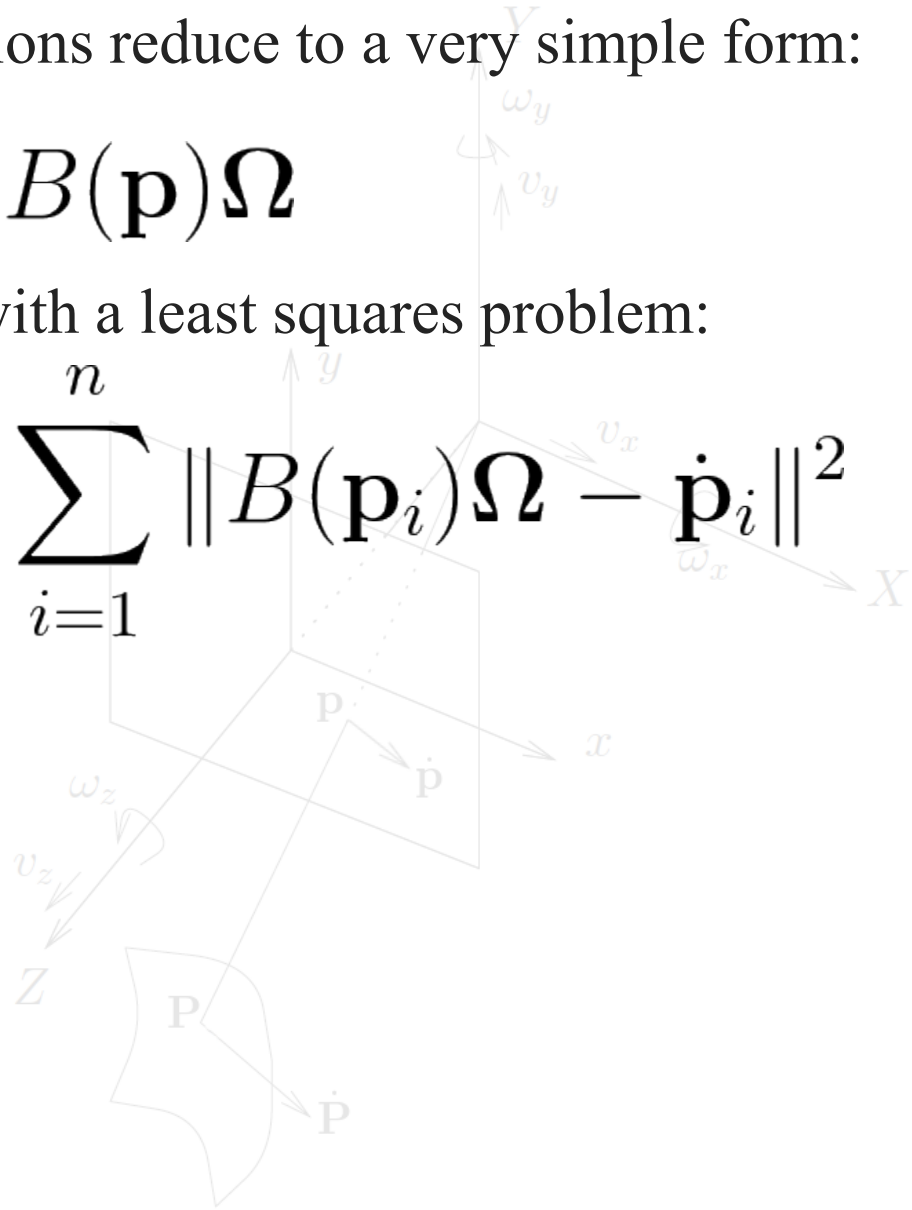
Case 2: No Translational Velocity

With no translation the equations reduce to a very simple form:

$$\dot{\mathbf{p}} = B(\mathbf{p})\boldsymbol{\Omega}$$

And this can also be solved with a least squares problem:

$$\Omega^* = \arg\min_{\Omega} \sum_{i=1}^{n} \|B(\mathbf{p}_i)\boldsymbol{\Omega} - \dot{\mathbf{p}}_i\|^2$$



Case 3: No Angular Velocity

With no rotation the equations reduce to a different form than what we are used to:

$$\dot{\mathbf{p}} = \frac{1}{Z_i} A(\mathbf{p})\mathbf{V} \iff [\dot{\mathbf{p}}]_\times A(\mathbf{p})\mathbf{V} = 0$$

This can be reformulated as a minimum eigenvalue problem:

$$\arg\min_{\mathbf{V}:\|\mathbf{V}\|=1} \sum_{i=1}^{n} \|[\dot{\mathbf{p}}_i]_\times A(\mathbf{p}_i)\mathbf{V}\|^2$$

$$= \sum_{i=1}^{n} \left([\dot{\mathbf{p}}_i]_\times A(\mathbf{p}_i)\mathbf{V}\right)^T \left([\dot{\mathbf{p}}_i]_\times A(\mathbf{p}_i)\mathbf{V}\right)$$

$$= \sum_{i=1}^{n} \mathbf{V}^T A(\mathbf{p}_i)^T [\dot{\mathbf{p}}_i]_\times^T [\dot{\mathbf{p}}_i]_\times A(\mathbf{p}_i)\mathbf{V}$$

$$= \mathbf{V}^T \left(\sum_{i=1}^{n} A(\mathbf{p}_i)^T [\dot{\mathbf{p}}_i]_\times^T [\dot{\mathbf{p}}_i]_\times A(\mathbf{p}_i) \right) \mathbf{V}$$

Case 4: Everything Unknown

With no rotation the equations reduce to a different form than what we are used to:

$$\dot{\mathbf{p}} = \frac{1}{Z} A(\mathbf{p})\mathbf{V} + B(\mathbf{p})\mathbf{\Omega}$$

There is not too much we can do here as this is bilinear in our unknowns (depth and rotation). There are a few similar methods people have used to solve this:

- Method 1: Alternating minimization between *V* and *Ω/Z*
- Method 2: Marginalize out *Ω/Z* and exhaustively search for

Case 4: Everything Unknown

When we don't have any prior knowledge we are just left with the original equation:

$$\dot{\mathbf{p}} = \frac{1}{Z} A(\mathbf{p})\mathbf{V} + B(\mathbf{p})\mathbf{\Omega}$$

There is not too much we can do here as this is bilinear in our unknowns (depth and velocity). Velocity can only be recovered up to a scale. There are a few similar methods people have used to solve this:

- Method 1: Alternating minimization between V and Ω/Z [1]
- Method 2: Marginalize out Ω/Z and exhaustively search for V [2]

[1] Optimal structure from motion: Local ambiguities and global estimates, Soatto and Brockett 1998
[2] Subspace methods for recovering rigid motion I: Algorithm and implementation, Heeger and Jepson 1992

Case 4: Everything Unknown

We can rewrite the equation as:

$$\dot{\mathbf{p}}_i = \frac{1}{Z_i} A(\mathbf{p}_i)\mathbf{V} + B(\mathbf{p}_i)\mathbf{\Omega} = \begin{pmatrix} A(\mathbf{p}_i)\mathbf{V} & B(\mathbf{p}_i) \end{pmatrix} \begin{pmatrix} \frac{1}{Z_i} \\ \mathbf{\Omega} \end{pmatrix}$$

$$\Longrightarrow \begin{pmatrix} \dot{\mathbf{p}}_1 \\ \vdots \\ \dot{\mathbf{p}}_n \end{pmatrix} = \begin{pmatrix} A(\mathbf{p}_1)\mathbf{V} & \dots & 0 & B(\mathbf{p}_1) \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & A(\mathbf{p}_n)\mathbf{V} & B(\mathbf{p}_n) \end{pmatrix} \begin{pmatrix} \frac{1}{Z_1} \\ \vdots \\ \frac{1}{Z_n} \\ \mathbf{\Omega} \end{pmatrix}$$

$$\Longrightarrow \mathbf{u} = \begin{pmatrix} \mathbf{A}(\mathbf{V}) & \mathbf{B} \end{pmatrix} \begin{pmatrix} \rho \\ \mathbf{\Omega} \end{pmatrix}$$

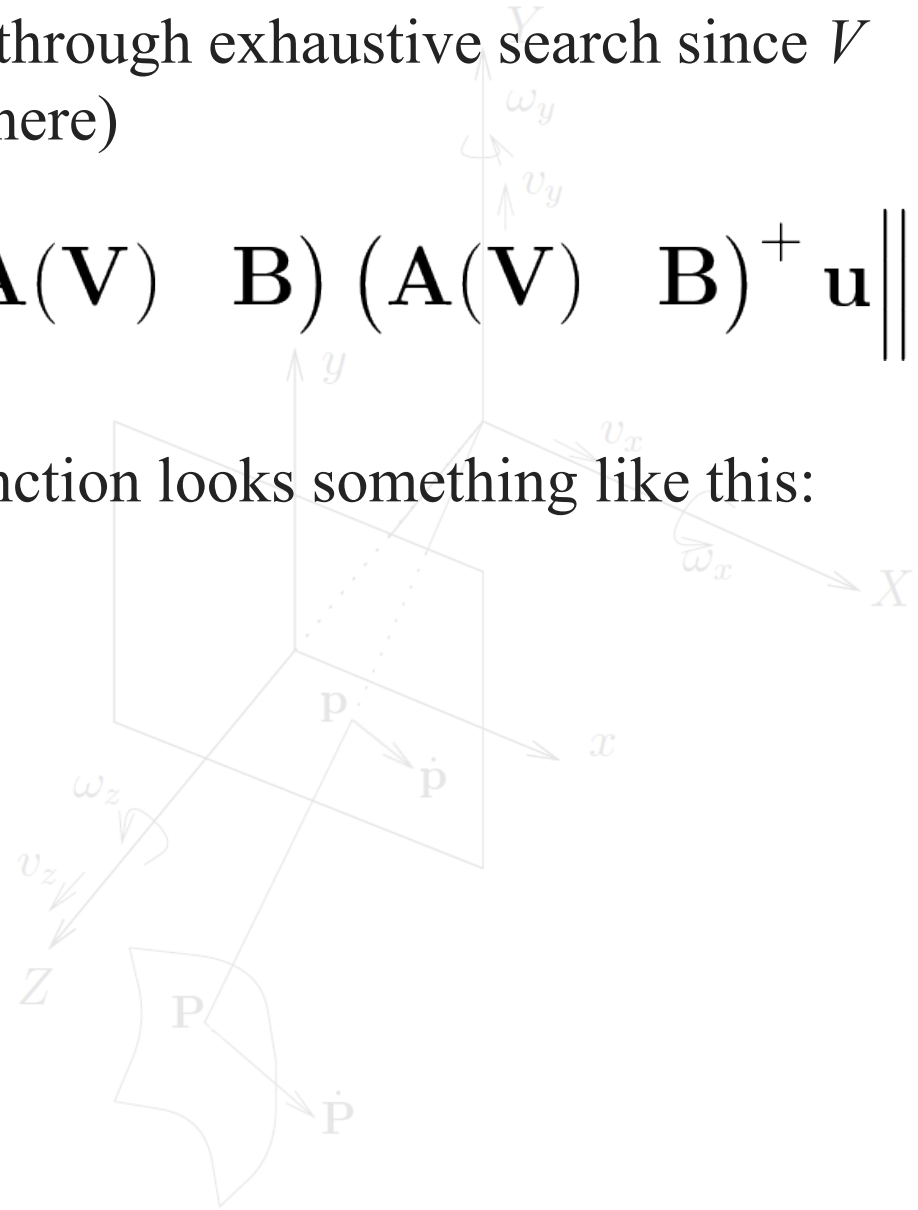
$$\Longrightarrow \begin{pmatrix} \rho^* \\ \mathbf{\Omega}^* \end{pmatrix} = \begin{pmatrix} \mathbf{A}(\mathbf{V}) & \mathbf{B} \end{pmatrix}^+ \mathbf{u}$$

Case 4: Everything Unknown

With this we can solve for V through exhaustive search since V is in a bounded space (the sphere)

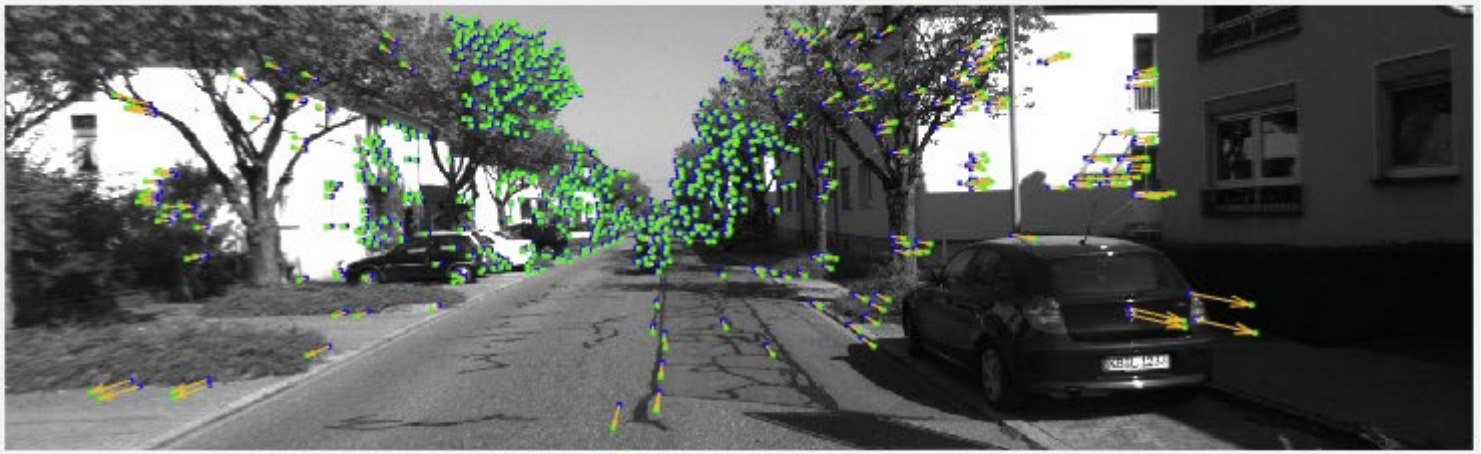
$$\arg\min_{\mathbf{V}\in S^2} \left\| \mathbf{u} - \begin{pmatrix} \mathbf{A}(\mathbf{V}) & \mathbf{B} \end{pmatrix} \begin{pmatrix} \mathbf{A}(\mathbf{V}) & \mathbf{B} \end{pmatrix}^{+} \mathbf{u} \right\|^2$$

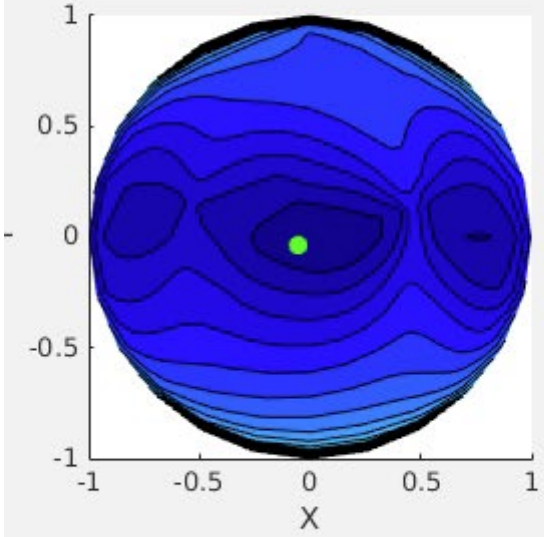
A visualization of the loss function looks something like this:



Case 4: Everything Unknown

A visualization of the loss function looks something like this:

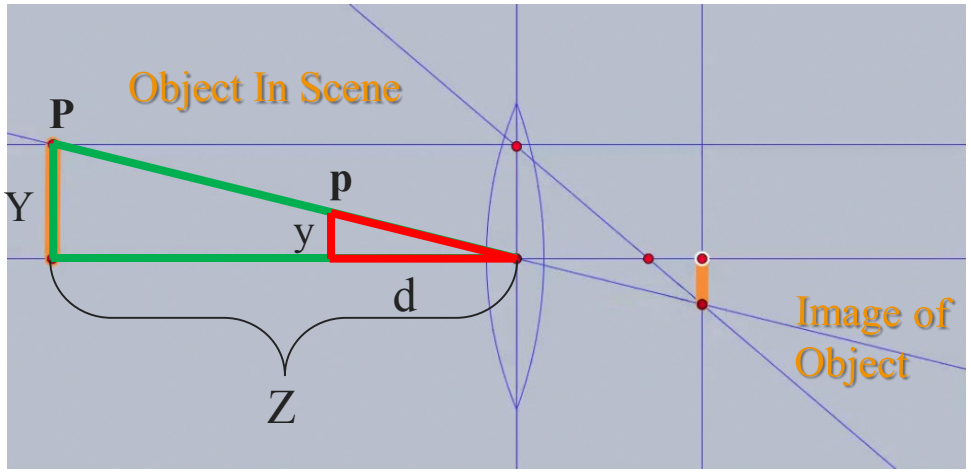




Recap of Vision Thus Far

Basic Assumptions

We take the thin lens model and approximate it as a pinhole model using calibration to make it good enough

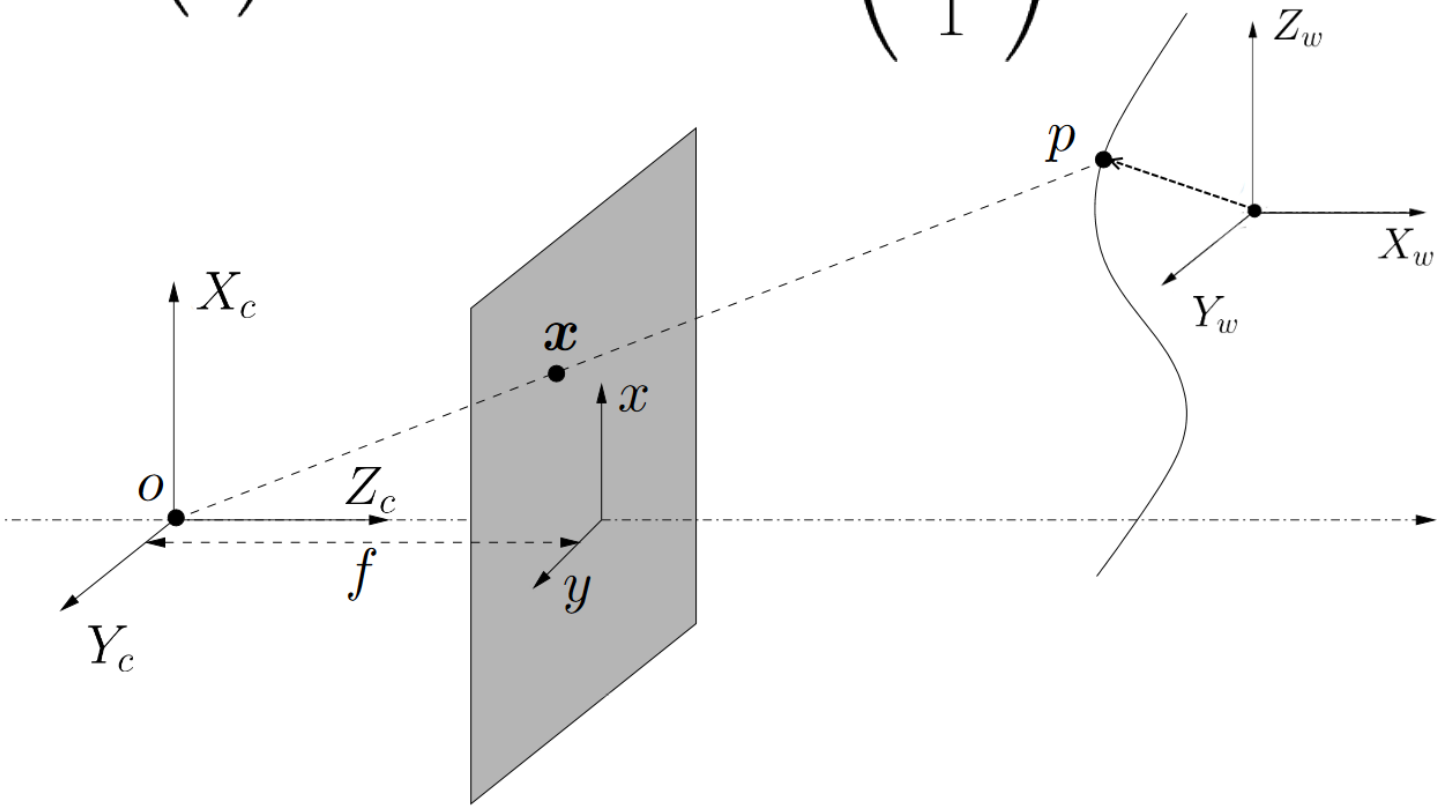


$$x' = fx(1 + k_1 r + k_2 r^2 + \ldots) + x_0$$
$$y' = fy(1 + k_1 r + k_2 r^2 + \ldots) + y_0$$

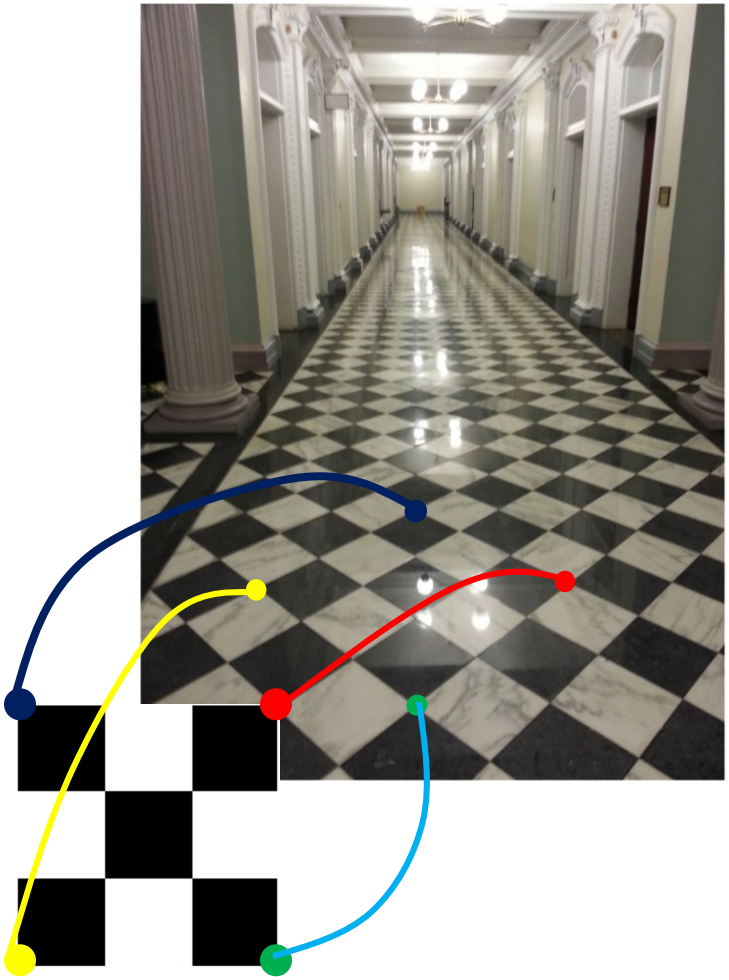
$$x = \frac{X_c}{Z_c}, \quad y = \frac{Y_c}{Z_c}$$

Projection Equations

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{pmatrix} \begin{pmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{pmatrix}$$



Projective Transformations



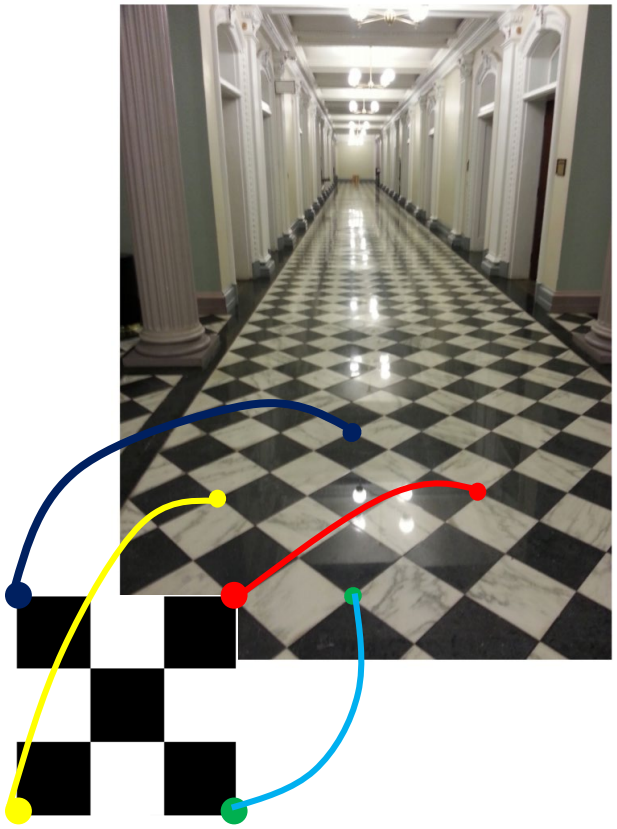
We learned about how to map points on the image to points on the image, specifically how planes transform

$$\lambda_i \begin{pmatrix} x_i' \\ y_i' \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x_i \\ y_i \\ 1 \end{pmatrix}$$

$$Ah = 0$$

$$A = USV^T \implies h = V_9$$

Pose from Projective Transformations



Assuming the points are on the plane we can solve for the exact pose from the transformation parameters and calibration

$$\begin{pmatrix} \hat{R}_1 & \hat{R}_2 & \hat{T} \end{pmatrix} = \begin{pmatrix} \hat{r}_{11} & \hat{r}_{12} & \hat{t}_1 \\ \hat{r}_{21} & \hat{r}_{22} & \hat{t}_2 \\ \hat{r}_{31} & \hat{r}_{32} & \hat{t}_3 \end{pmatrix} = \underbrace{\begin{pmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{pmatrix}^{-1} \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix}}_{K^{-1}H}$$

$$\begin{pmatrix} \hat{R}_1 & \hat{R}_2 & \hat{R}_1 \times \hat{R}_2 \end{pmatrix} = USV^T$$

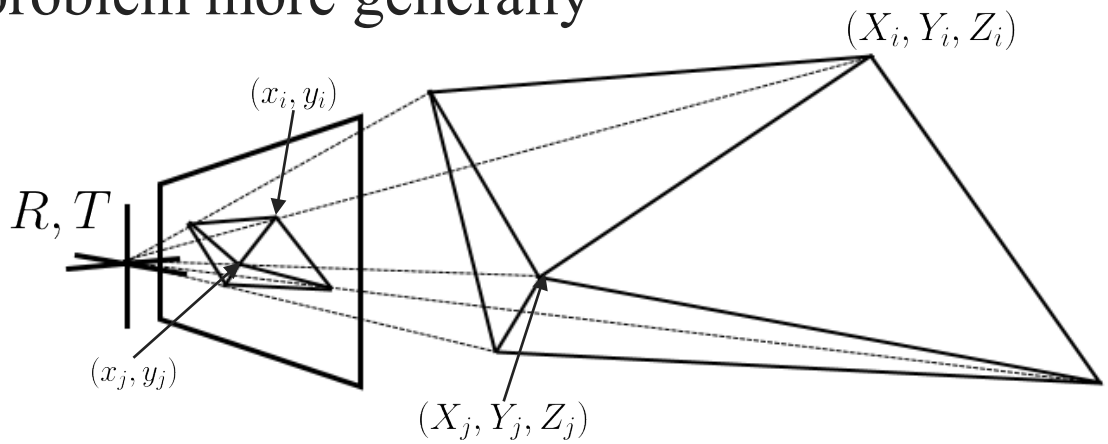
$$R = U \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(UV^T) \end{pmatrix} V^T$$

$$T = \hat{T} / \|\hat{R}_1\|$$

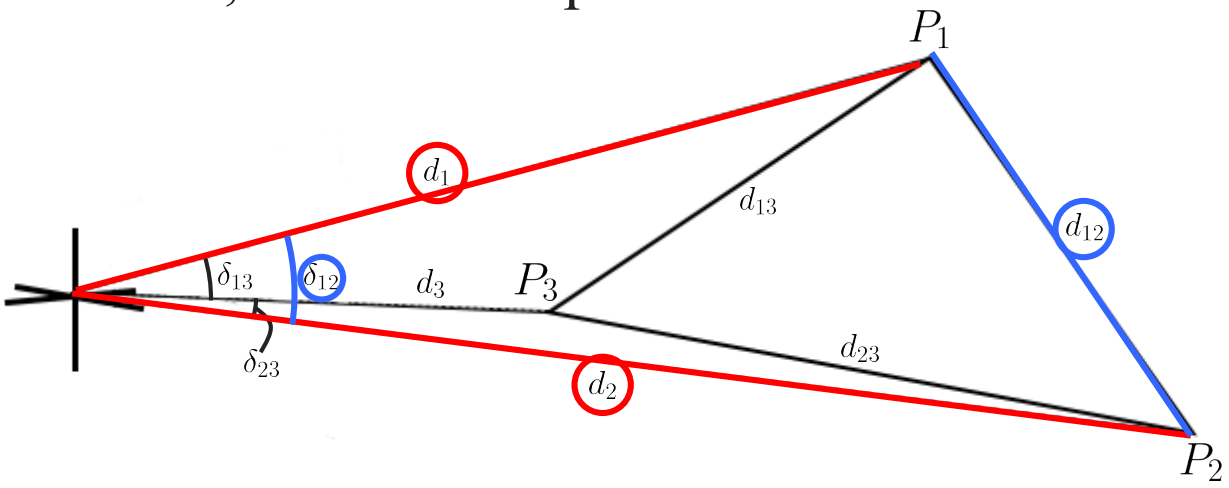
This is arbitrary, can also use second column or average

PnP and P3P

If we have more general 2D-3D correspondences, we need to solve the problem more generally



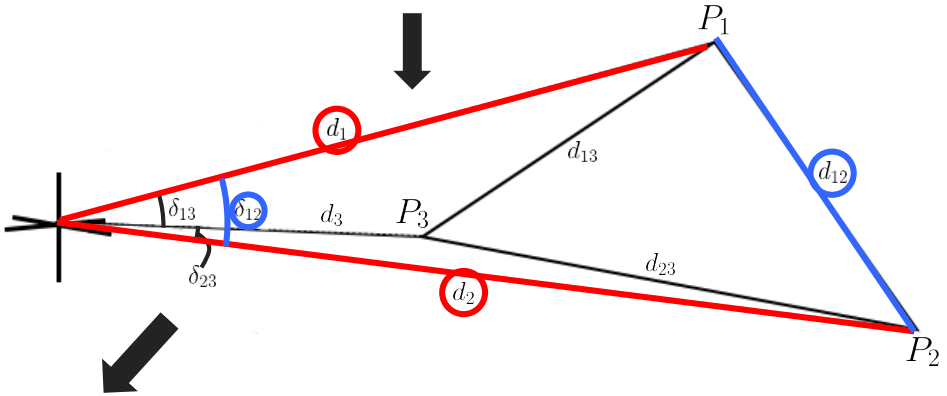
Minimal case is P3P, which is helpful if we want to do RANSAC



PnP and P3P

For P3P, we needed to solve a fairly complicated polynomial and go back to solve for the distances

$$(x_i, y_i) \leftrightarrow (X_i, Y_i, Z_i)$$



$$\begin{aligned} d_1^2 + d_2^2 - 2d_1 d_2 \cos(\delta_{12}) &= d_{12}^2 \\ d_1^2 + d_3^2 - 2d_1 d_3 \cos(\delta_{13}) &= d_{13}^2 \\ d_2^2 + d_3^2 - 2d_2 d_3 \cos(\delta_{23}) &= d_{23}^2 \end{aligned}$$

$$\begin{aligned} d_{13}^2(u^2 + v^2 - 2uv\cos(\delta_{23})) &= d_{23}^2(1 + v^2 - 2v\cos(\delta_{13})) \\ d_{12}^2(1 + v^2 - 2v\cos(\delta_{13})) &= d_{13}^2(1 + u^2 - 2u\cos(\delta_{12})) \end{aligned}$$

$$\begin{aligned} & u^2 = a_1 v^2 + a_2 v + a_3 uv + a_4 \\ \Longrightarrow\ & u = (b_1 v^2 + b_2 v + b_3)/(b_4 v + b_5) \\ \Longrightarrow\ & \alpha v^4 + \beta v^3 + \gamma v^2 + \zeta v + \eta = 0 \end{aligned}$$

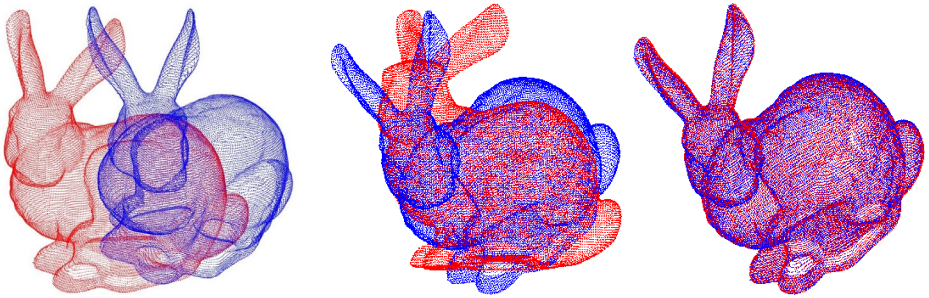
→ 4 solutions for u, v

Substitute back in for d_i to and remove physically invalid solutions

Procrustes

Once we have the distances in P3P we can use that to solve for the position and orientation of the points

$$\arg\min_{R\in \mathrm{SO}(3), T\in\mathbb{R}^3} \sum_i \|RP_i + T - P_i'\|^2$$



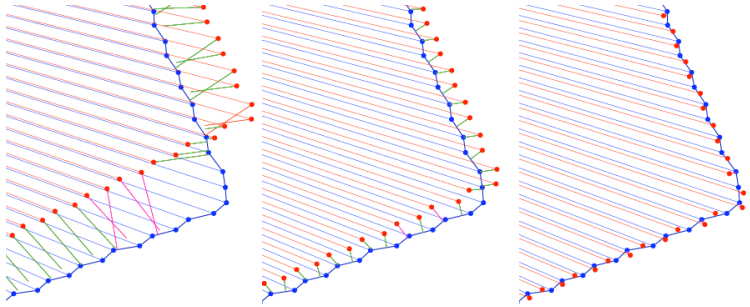
$$\overline{P} = \frac{1}{N}\sum_i P_i, \quad \overline{P'} = \frac{1}{N}\sum_i P_i$$

$$Z = \sum_i (P_i - \overline{P})(P_i' - \overline{P})^T = USV^T$$

$$R = V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{pmatrix} U^T$$

$$T = \overline{P'} - R\overline{P}$$

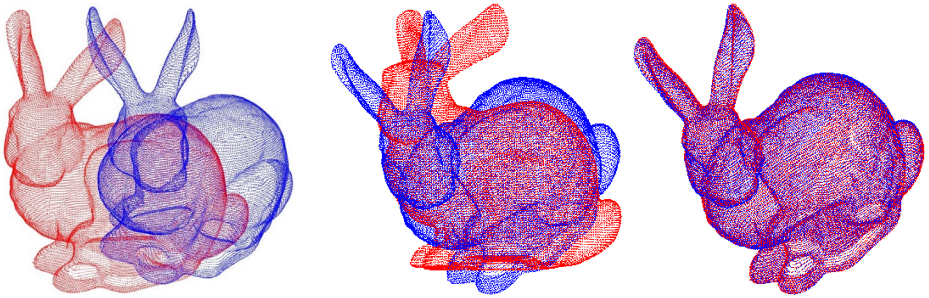
Can also use Procrustes in other contexts e.g. solving ICP



Procrustes

Once we have the distances in P3P we can use that to solve for the position and orientation of the points

$$\arg\min_{R\in \mathrm{SO}(3), T\in\mathbb{R}^3} \sum_i \|RP_i + T - P_i'\|^2$$



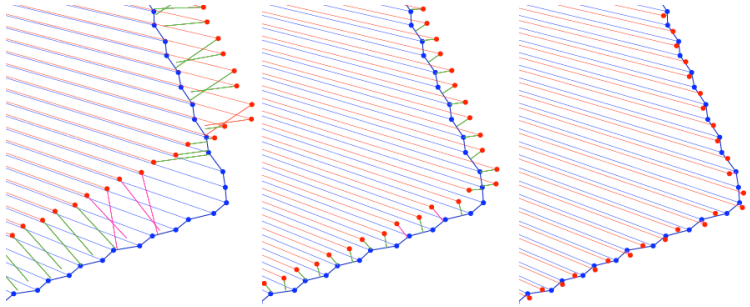
$$\overline{P} = \frac{1}{N}\sum_i P_i, \quad \overline{P'} = \frac{1}{N}\sum_i P_i$$

$$Z = \sum_i (P_i - \overline{P})(P_i' - \overline{P})^T = USV^T$$

$$R = V \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(VU^T) \end{pmatrix} U^T$$

$$T = \overline{P'} - R\overline{P}$$

Can also use Procrustes in other contexts e.g. solving ICP



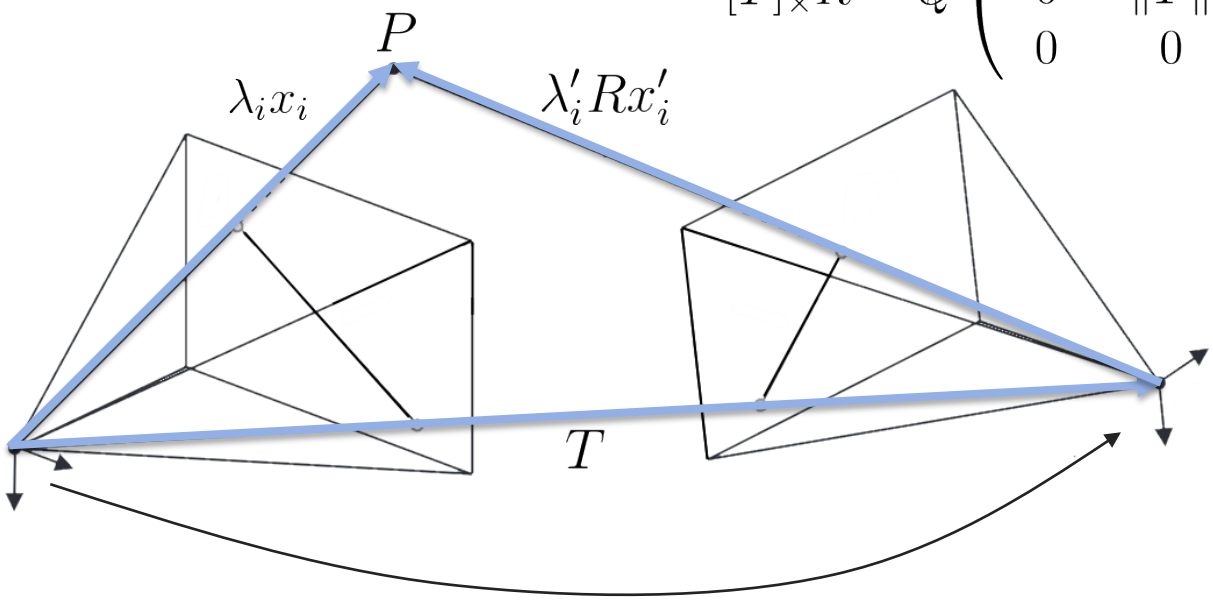
Epipolar Constraint

We also briefly went over how to get 3D pose from 2D-2D correspondences via the Essential matrix, and how to extract the pose from the essential matrix

$$\lambda_i x_i^T (T \times \lambda_i' R x_i') = 0$$
$$\implies x_i^T [T]_\times R x_i' = 0$$

$$[T]_\times = Q \begin{pmatrix} 0 & -\|T\| & 0 \\ \|T\| & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} Q^T$$

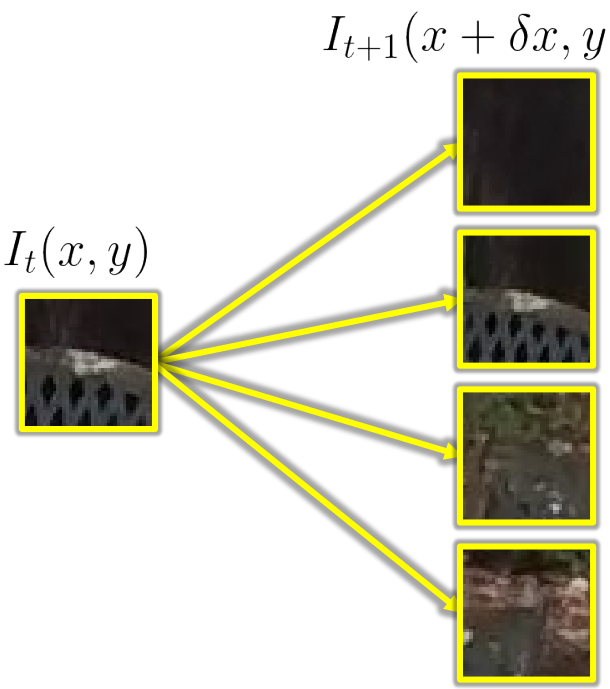
$$[T]_\times R = Q \begin{pmatrix} \|T\| & 0 & 0 \\ 0 & \|T\| & 0 \\ 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} Q^T R$$



Optical Flow

Next we stepped back a bit and looked more at the image side and how to get correspondences, and some troubles that arise with that

$$(\delta x, \delta y) = \arg\min_{\delta x, \delta y} \sum_{(x,y) \in \mathcal{N}(x_0, y_0)} \left(\Delta I_t(x,y) - \nabla I_{t+1}(x,y)^T \begin{pmatrix} \delta x \\ \delta y \end{pmatrix} \right)^2$$

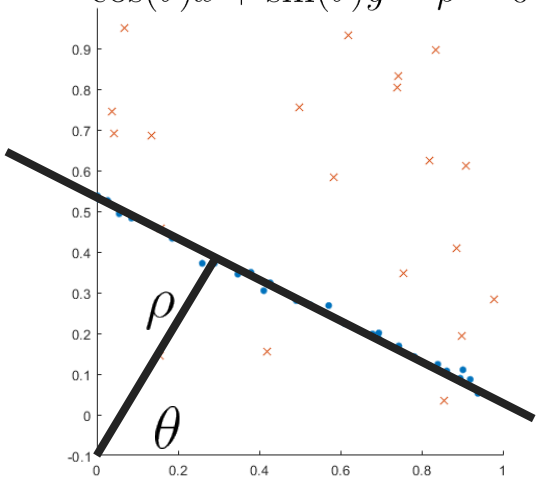


$$\begin{pmatrix} \delta x^* \\ \delta y^* \end{pmatrix} = \left(\sum_{(x,y)} \nabla I_{t+1}(x,y) \nabla I_{t+1}(x,y)^T \right)^{-1} \left(\sum_{(x,y)} \Delta I_t(x,y) \nabla I_{t+1}(x,y) \right)$$

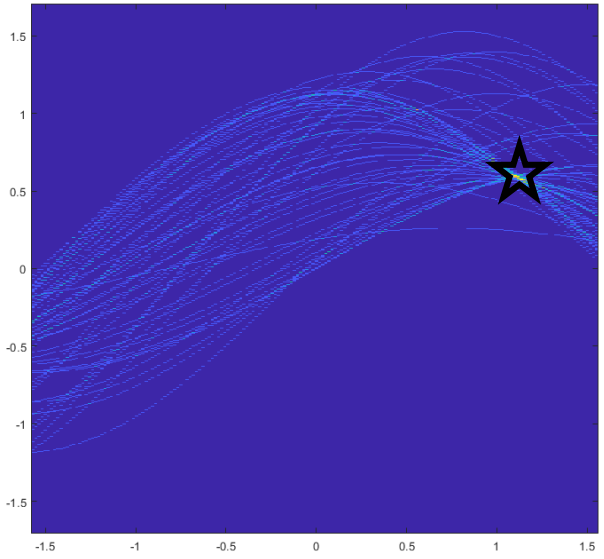
Outlier Rejection

Given the high potential for error in these methods, we learned about methods to separate ‘inliers’ from ‘outliers’ using the Hough Transform, but primarily RANSAC

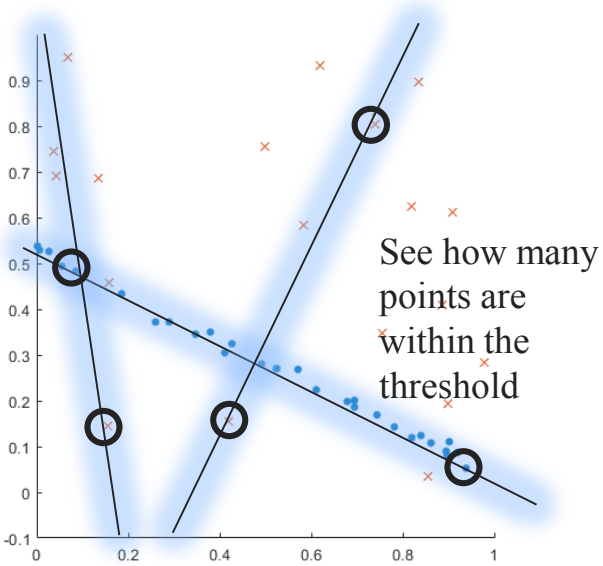
$$\cos(\theta)x + \sin(\theta)y - \rho = 0$$



Hough



RANSAC



See how many points are within the threshold

Today: Motion Field Equations

And we learned how to take these things and derive motion models as opposed to merely static models, and we can estimate position and velocity

$$\dot{\mathbf{p}} = \frac{1}{Z} A(\mathbf{p})\mathbf{V} + B(\mathbf{p})\boldsymbol{\Omega}$$

Next Time

Now that we have estimates involving time. Given measurement error (even with something like RANSAC) there can still be estimation problems. Can we use temporal information to make things better? (Spoilers: yes). More importantly, how can we use temporal information in an optimal way while still being fast enough to be practical?