6DOF Vision-Aided Inertial Localization

Florian Shkurti
florian@cim.mcgill.ca
Mobile Robotics Lab

Overview

- Desired localization accuracy
- Available sensor inputs (inertial measurement unit, camera)
- Types of environments
- State propagation
- State augmentation
- Feature extraction and matching
- State update
- Some results
- Concluding remarks

Our goals

- Estimate the 3D position and 3D orientation (3D pose) of a moving robot in real-time.

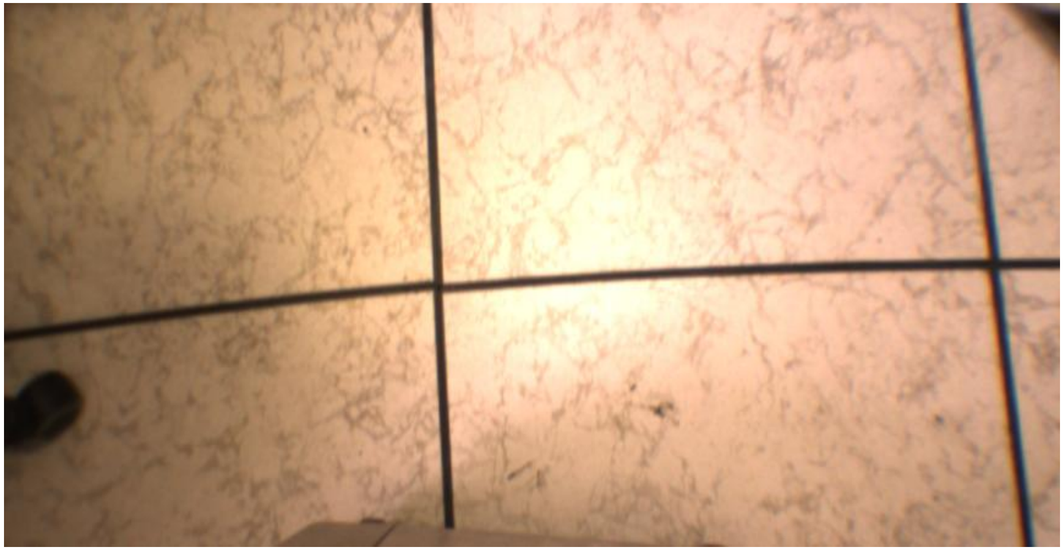
- Ideally, we'd like position error $\leq 10\%$ of the distance traveled.

- Unfortunately, this is hard to guarantee.

Visual input in the lab

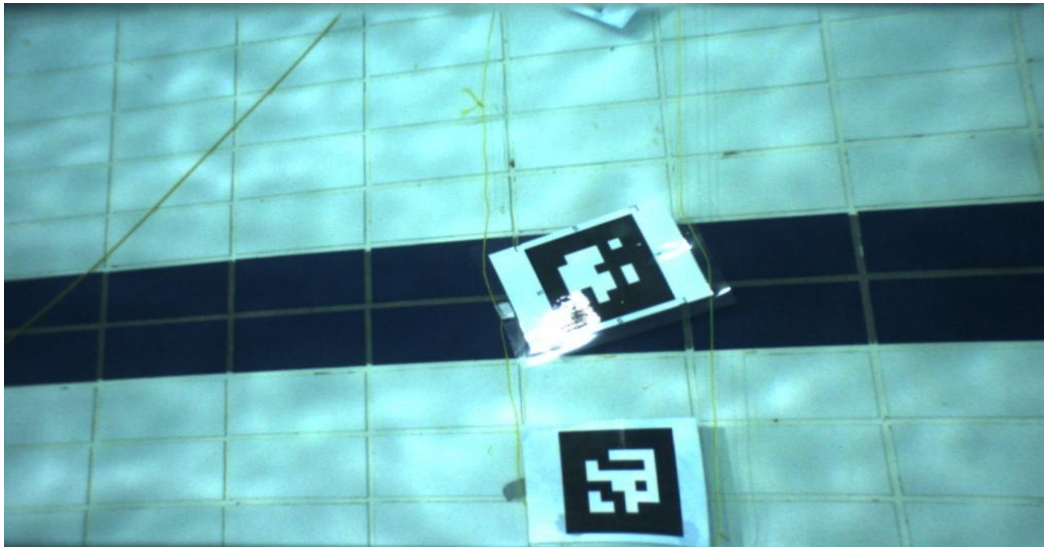
Good

Not so good

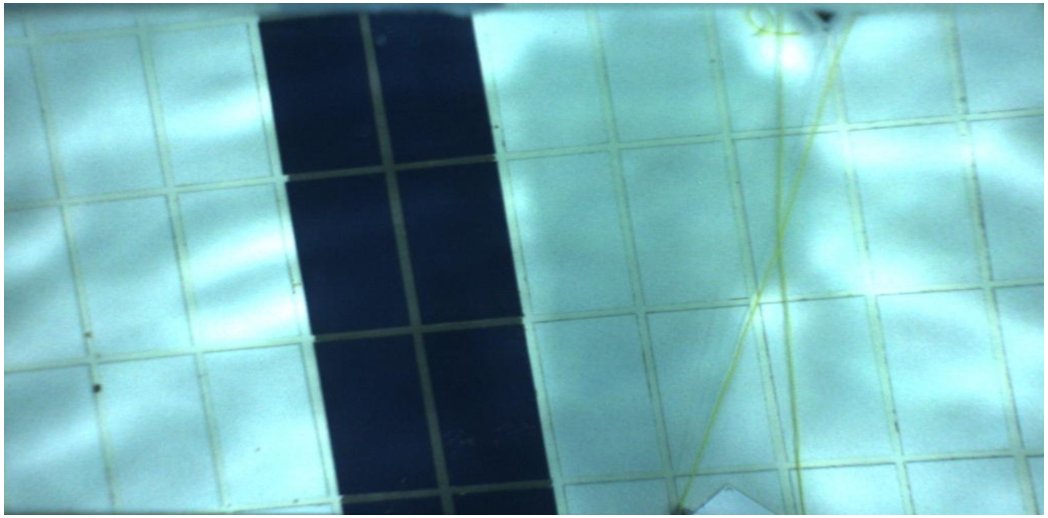


Visual input in the pool

Good

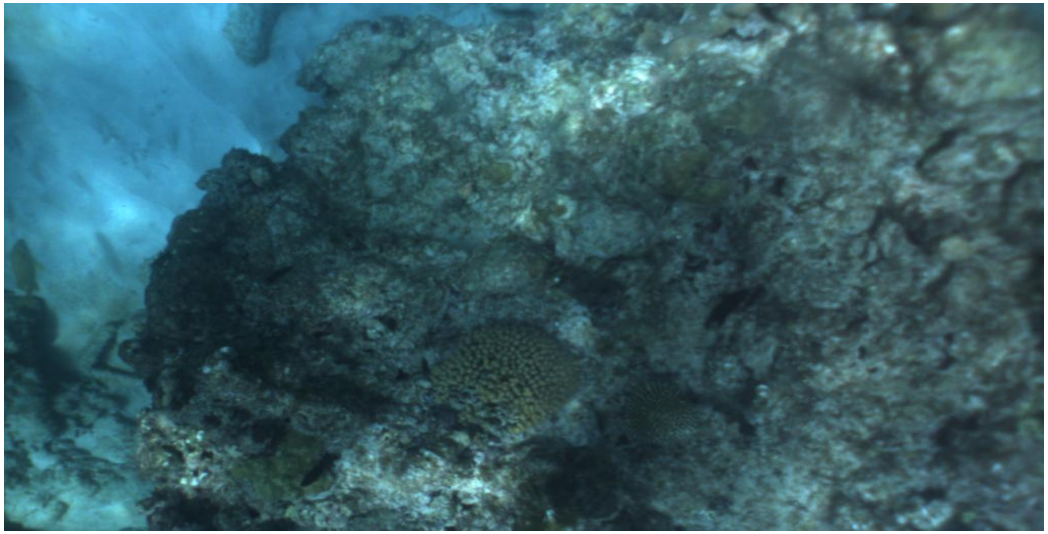


Not so good

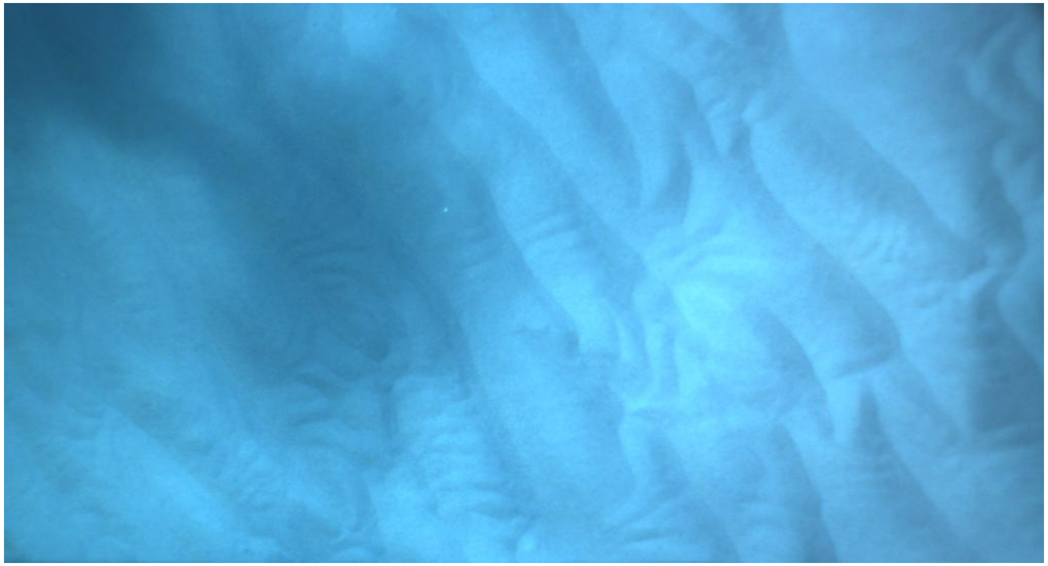


Visual input in the ocean

Good



Not so good



Typical SLAM approaches

- Try to estimate the robot's 3D pose **and** the 3D positions of the observed landmarks

$$\mathbf{x} = [\underbrace{x \;\; y \;\; z \;\; \theta \;\; \phi \;\; \psi}_{R} \;\; \underbrace{x \;\; y \;\; z}_{L_1} \;\; \ldots \;\; \underbrace{x \;\; y \;\; z}_{L_N}]$$

- Correlations between landmarks are also estimated.

- The size of the state becomes significantly big, very quickly. Each update step of EKF-SLAM takes $O(N^2)$.

- Need to remove landmarks from the state.

Proposed approach

- Heavily based on work by A. Mourikis and S. Roumeliotis [1]
- Uses input from a single camera and an inertial measurement unit (IMU).
- Combines these inputs through an Extended Kalman Filter.
- Each update step takes $O(N)$ where N is the number of landmarks.
- [1] "A multi-state constrained Kalman filter for vision-aided inertial navigation," ICRA 2007

Inertial Measurement Units

- Think of an IMU as a gyroscope and an accelerometer.
- If $[x \;\; y \;\; z \;\; \theta \;\; \phi \;\; \psi]$ is the robot's pose then the IMU gives us noisy measurements of $[\ddot{x} \;\; \ddot{y} \;\; \ddot{z} \;\; \dot{\theta} \;\; \dot{\phi} \;\; \dot{\psi}]$
- Let $\mathbf{a} = [\ddot{x} \;\; \ddot{y} \;\; \ddot{z}]$ be the true linear acceleration and $\boldsymbol{\omega} = [\dot{\theta} \;\; \dot{\phi} \;\; \dot{\psi}]$ be the true angular velocity. IMU measurements are typically modeled as:

$$\mathbf{a}_m = c\mathbf{a} + \mathbf{b}_a + \mathbf{n}_a \qquad \mathbf{n}_a \sim N(0, \boldsymbol{\sigma}_{an}) \text{ and } \dot{\mathbf{b}}_a \sim N(0, \boldsymbol{\sigma}_{ab})$$

$$\boldsymbol{\omega}_m = \boldsymbol{\omega} + \mathbf{b}_g + \mathbf{n}_g \qquad \mathbf{n}_g \sim N(0, \boldsymbol{\sigma}_{gn}) \text{ and } \dot{\mathbf{b}}_g \sim N(0, \boldsymbol{\sigma}_{gb})$$

IMU noise modeling

- The IMU measurement noise $\mathbf{n}_a \sim N(0, \boldsymbol{\sigma}_{an})$ and $\mathbf{n}_g \sim N(0, \boldsymbol{\sigma}_{gn})$ is typically assumed to be fixed. It is estimated offline while keeping the robot still.

- The accelerometer bias is initialized as the average offset from [0 0 -g] when the robot is left still with zero pitch and roll. The noise parameters of $\dot{\mathbf{b}}_a \sim N(0, \boldsymbol{\sigma}_a)$ are estimated offline. $\mathbf{b}_a$ is estimated online.

- Similarly for the gyroscope bias.

- Note: In the proposed algorithm we do not estimate the accelerometer scaling factor. We fix it to $c = 1$.

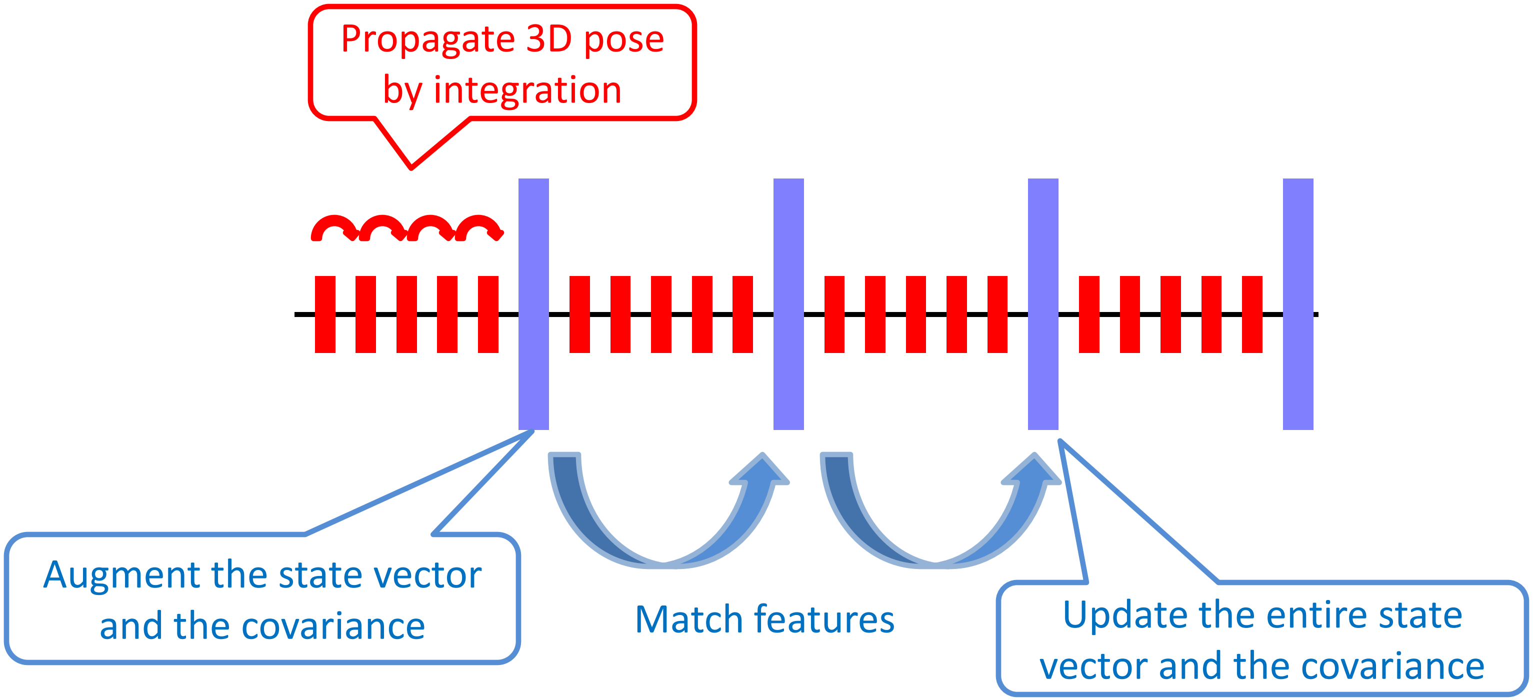
IMU integration drift

- “Why don’t we just integrate the IMU measurements to get a pose estimate?”

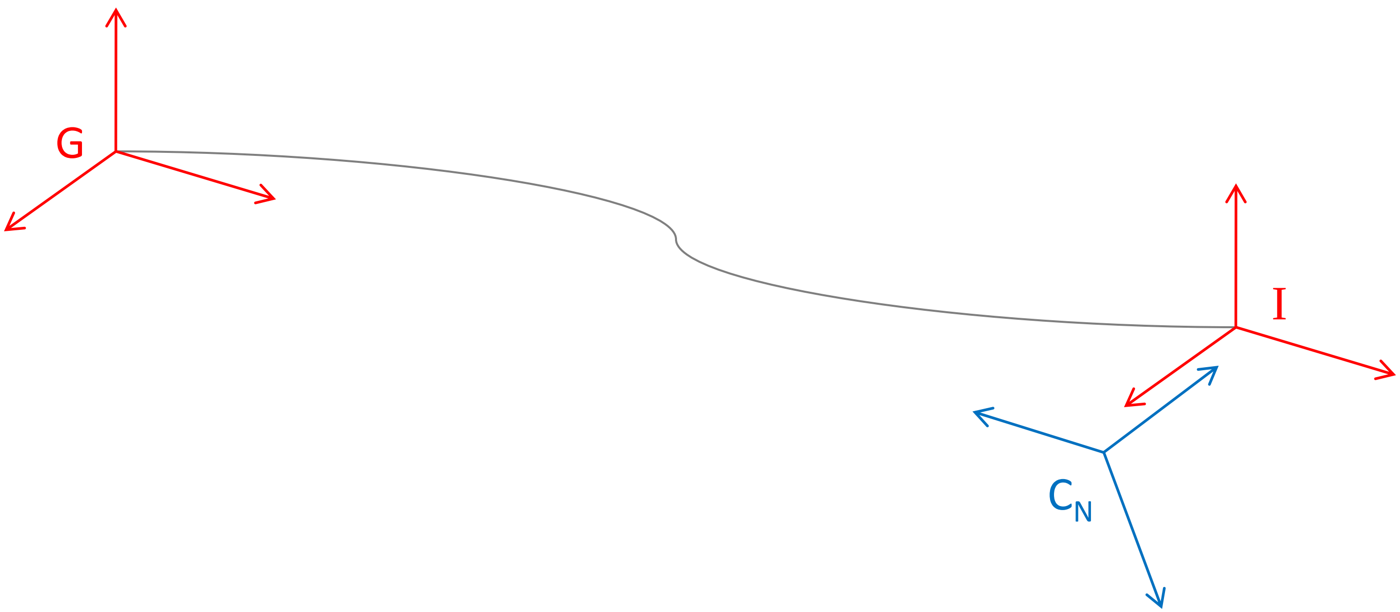
- Because errors will also be integrated.

Back to the algorithm

- Main idea: assuming feature-rich scenes we can use visual motion estimation to correct IMU drift.



The state vector



$$\mathbf{x} = [\underbrace{{}^{I}_{G}\mathbf{q}\ \ \mathbf{b}_g\ \ {}^{G}\mathbf{v}_I\ \ \mathbf{b}_a\ \ {}^{G}\mathbf{p}_I}_{\text{IMU state}}\ \ \underbrace{{}^{C_1}_{G}\mathbf{q}\ \ {}^{G}\mathbf{p}_{C_1}}_{\text{1}^{\text{st}}\text{ camera frame}}\ \cdots\ \underbrace{{}^{C_N}_{G}\mathbf{q}\ \ {}^{G}\mathbf{p}_{C_N}}_{\text{N}^{\text{th}}\text{ camera frame}}\ ]$$

The state vector

- ${}^{I}_{G}\mathbf{q} = [\mathbf{u}\sin(\theta/2) \;\; \cos(\theta/2)]$ is a quaternion that represents the rotation from the global frame G to the current IMU frame.

- ${}^{G}\mathbf{p}_{I}$ is the origin of the IMU frame in global coordinates.

- ${}^{G}\mathbf{v}_{I}$ is the velocity of the IMU frame in global coordinates.

- ${}^{C_i}_{G}\mathbf{q}$ is the rotation from the global frame to the ith camera frame.

- ${}^{G}\mathbf{p}_{C_i}$ is the origin of the ith camera frame in global coordinates.

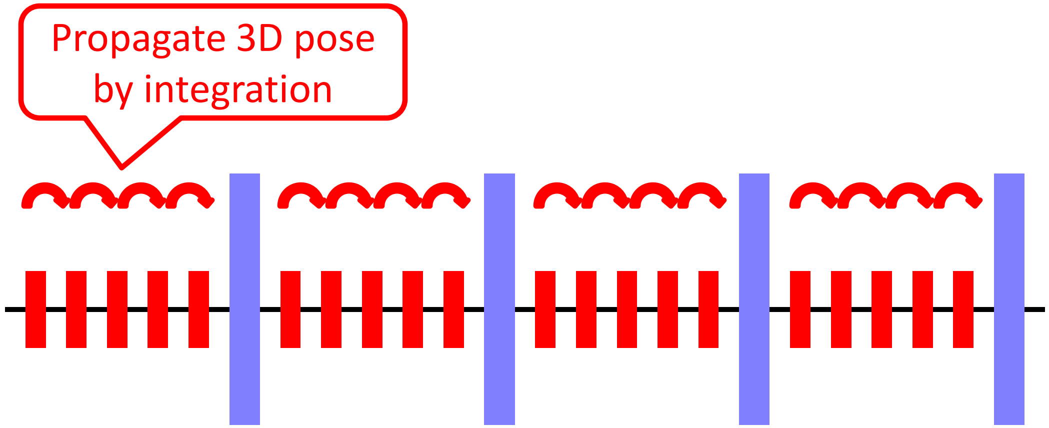
The covariance matrix

$$\mathbf{P} = \mathrm{cov}(\tilde{\mathbf{x}}) = \mathrm{cov}(\mathbf{x} - \hat{\mathbf{x}}) = \begin{bmatrix} \mathbf{P}_{II} & \mathbf{P}_{IC} \\ \mathbf{P}_{IC}^T & \mathbf{P}_{CC} \end{bmatrix}$$

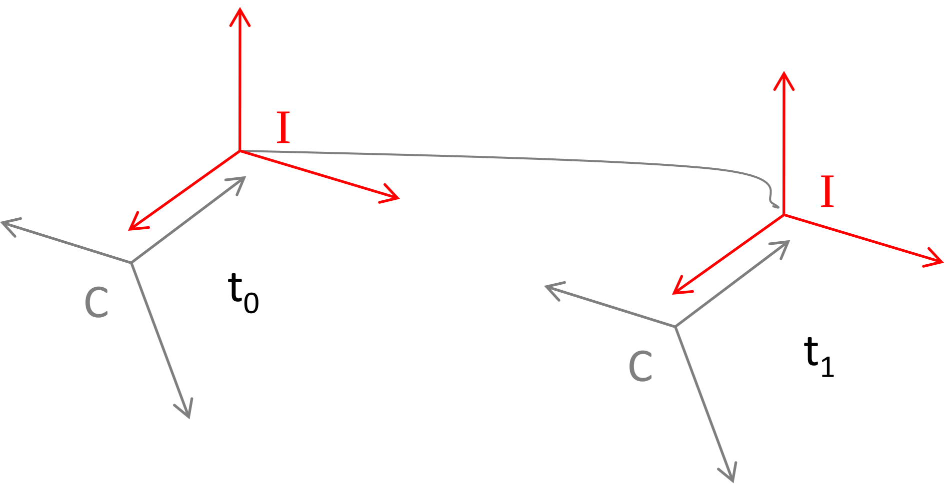
- At t=0 $\mathbf{P} = \mathbf{P}_{II} = 0$ and $\hat{\mathbf{x}} = \hat{\mathbf{x}}_{IMU}$ because no images have been recorded yet.

EKF Propagation

- Is done every time we receive an IMU measurement.



Propagation (state)



$$ {}^{G}\mathbf{p}_I(t_1) = {}^{G}\mathbf{p}_I(t_0) + {}^{G}\mathbf{v}_I(t_0)\,dt $$

$$ {}^{G}\mathbf{v}_I(t_1) = {}^{G}\mathbf{v}_I(t_0) + ({}^{G}_{I}\mathbf{R}(t_0)\,{}^{I}\hat{\mathbf{a}} + {}^{G}\mathbf{g})\,dt $$

$$ \mathbf{b}_g(t_1) = \mathbf{b}_g(t_0) $$

$$ \mathbf{b}_a(t_1) = \mathbf{b}_a(t_0) $$

$$ {}^{I}\hat{\mathbf{a}} = \mathbf{a}_m - \mathbf{b}_a(t_0) \qquad {}^{I}\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}_m - \mathbf{b}_g(t_0) \quad \Rightarrow $$

$$ {}^{I}_{G}\mathbf{q}(t_1) = \begin{bmatrix} \dfrac{{}^{I}\hat{\boldsymbol{\omega}}}{\left\| {}^{I}\hat{\boldsymbol{\omega}} \right\|} \sin\!\left(\dfrac{\left\| {}^{I}\hat{\boldsymbol{\omega}} \right\|}{2} dt\right) \\ \cos\!\left(\dfrac{\left\| {}^{I}\hat{\boldsymbol{\omega}} \right\|}{2} dt\right) \end{bmatrix} \otimes {}^{I}_{G}\mathbf{q}(t_0) $$

Note: the camera frames of the state are not propagated

Propagation (covariance)

- How do we propagate $\mathbf{P} = \mathrm{cov}(\tilde{\mathbf{x}}) = \mathrm{cov}(\mathbf{x} - \hat{\mathbf{x}}) = \begin{bmatrix} \mathbf{P}_{II} & \mathbf{P}_{IC} \\ \mathbf{P}_{IC}^{T} & \mathbf{P}_{CC} \end{bmatrix}$?

- Since we didn't modify the camera frames: $\begin{bmatrix} \mathbf{P}_{II} & \mathbf{P}_{IC} \\ \mathbf{P}_{IC}^{T} & \mathbf{P}_{CC} \end{bmatrix}$

- We can show that IMU propagation errors increase according to:

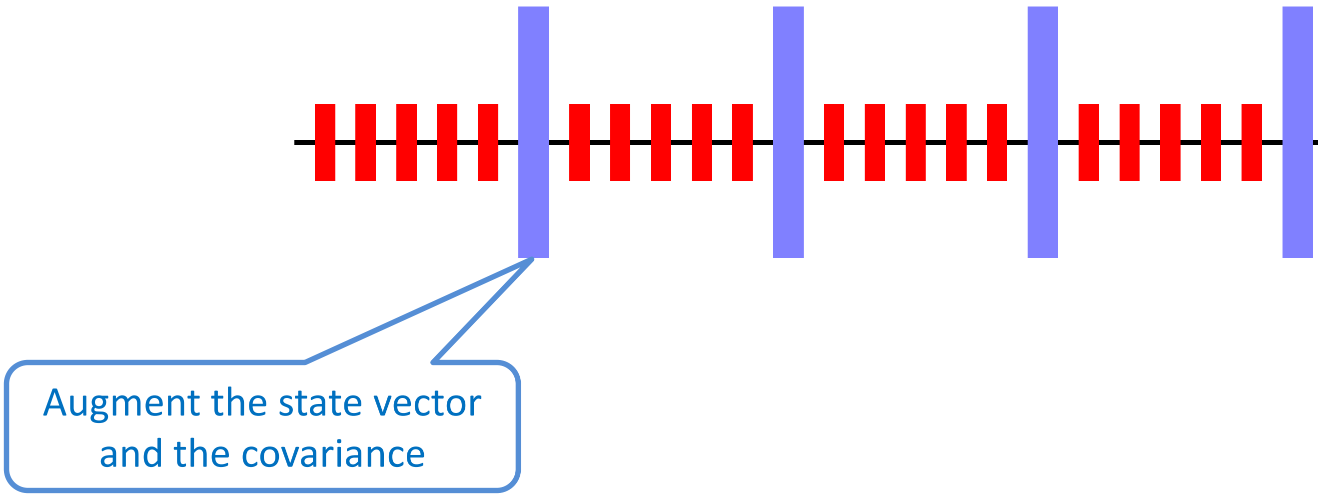
$$\frac{d}{dt}\tilde{\mathbf{x}}_{IMU}(t) = \mathbf{F}\tilde{\mathbf{x}}_{IMU}(t) + \mathbf{G}[\mathbf{n}_g \;\; \dot{\mathbf{b}}_g \;\; \mathbf{n}_a \;\; \dot{\mathbf{b}}_a]^T$$

where $\mathbf{F}$ depends on $\hat{\boldsymbol{\omega}}$, $\hat{\mathbf{a}}$, ${}^{\mathrm{I}}_{\mathrm{G}}\hat{\mathbf{q}}$ and $\mathbf{G}$ depends on ${}^{\mathrm{I}}_{\mathrm{G}}\hat{\mathbf{q}}$.

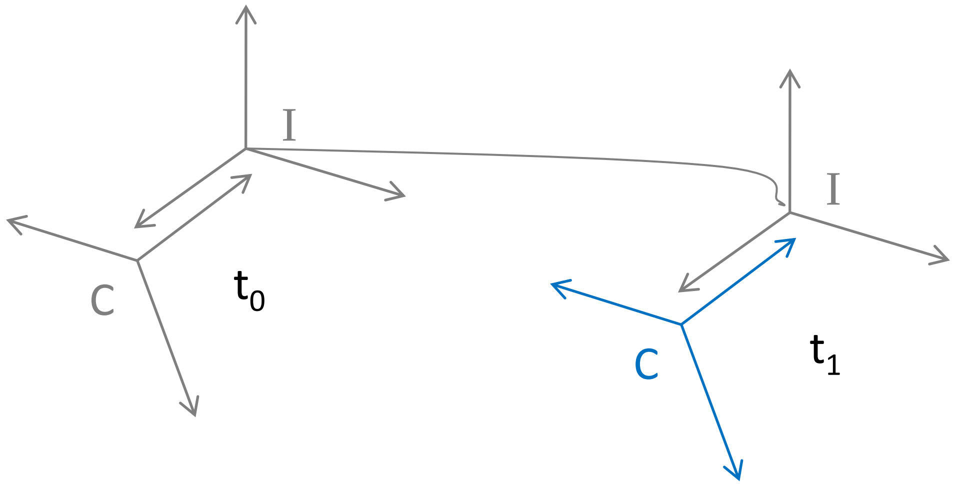
- We can get the propagated covariance by integration.

State augmentation

- Is done every time an image is recorded



State augmentation



$$\mathbf{x} = [{}^{I}_{G}\mathbf{q}\ \ \mathbf{b}_g\ \ {}^{G}\mathbf{v}_I\ \ \mathbf{b}_a\ \ {}^{G}\mathbf{p}_I]$$

$$\Downarrow$$

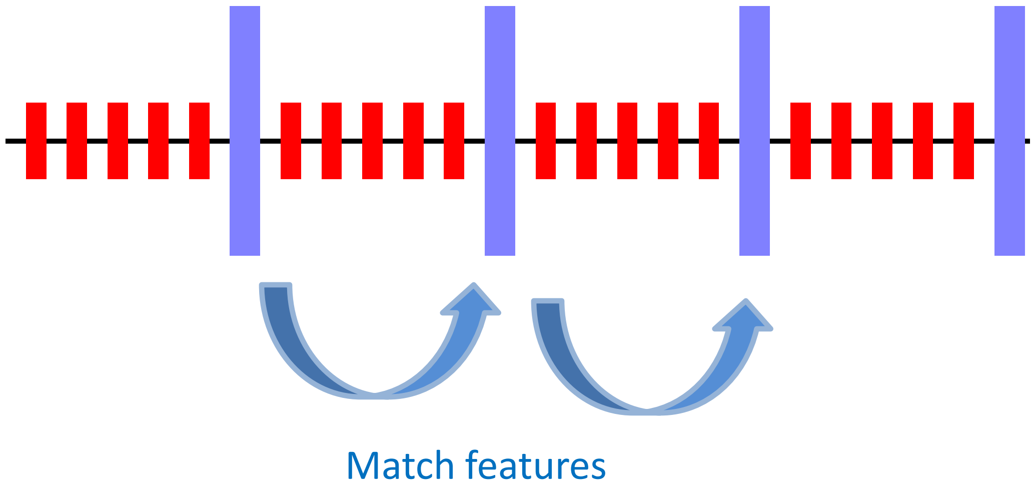
$$\mathbf{x} = [{}^{I}_{G}\mathbf{q}\ \ \mathbf{b}_g\ \ {}^{G}\mathbf{v}_I\ \ \mathbf{b}_a\ \ {}^{G}\mathbf{p}_I\ \ \boxed{{}^{C_1}_{G}\mathbf{q}\ \ {}^{G}\mathbf{p}_{C_1}}]$$

Transformation from camera to global coordinates

- The covariance is also augmented.

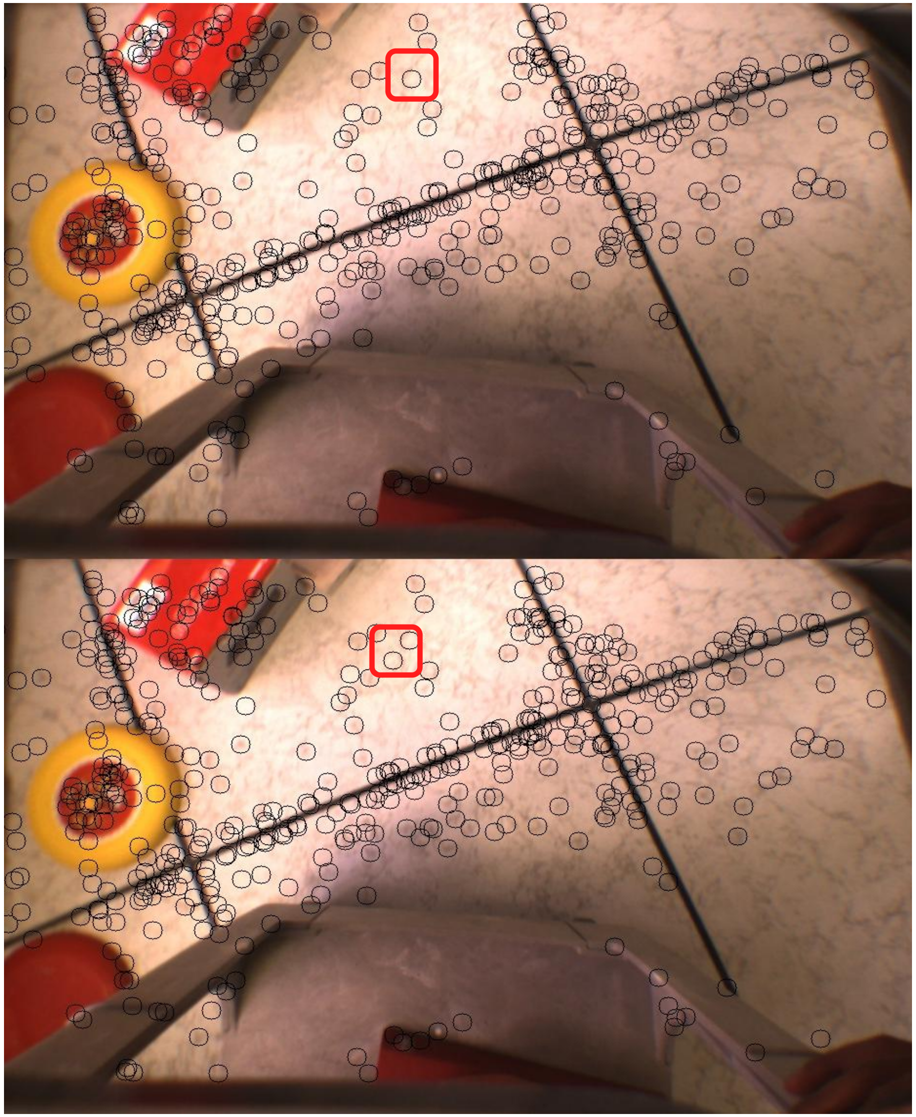
Feature matching

- Is done between every pair of consecutive images.

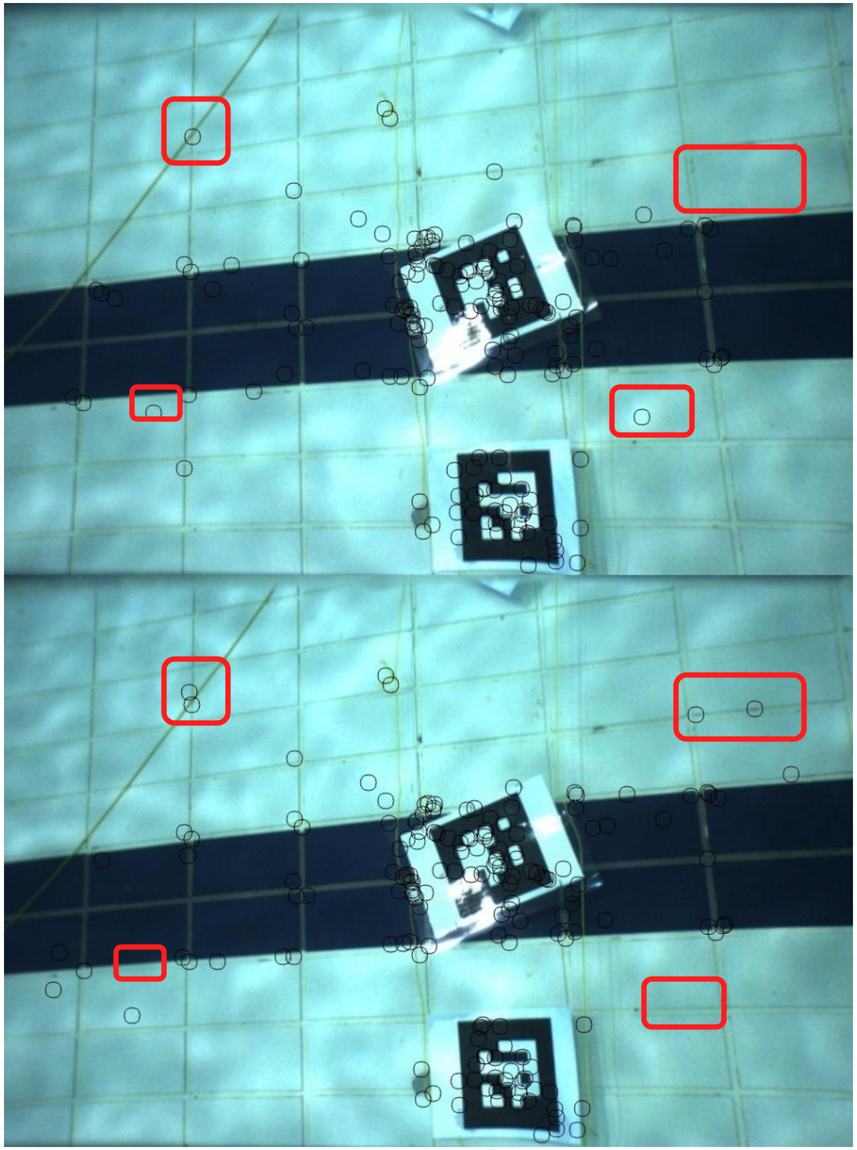


- SURF-64 features are matched using the Fast Library for Approximate Nearest Neighbors, by Muja & Lowe at UBC.
- Approximately 1000 features per frame.
- Average feature tracking length is 4 frames.

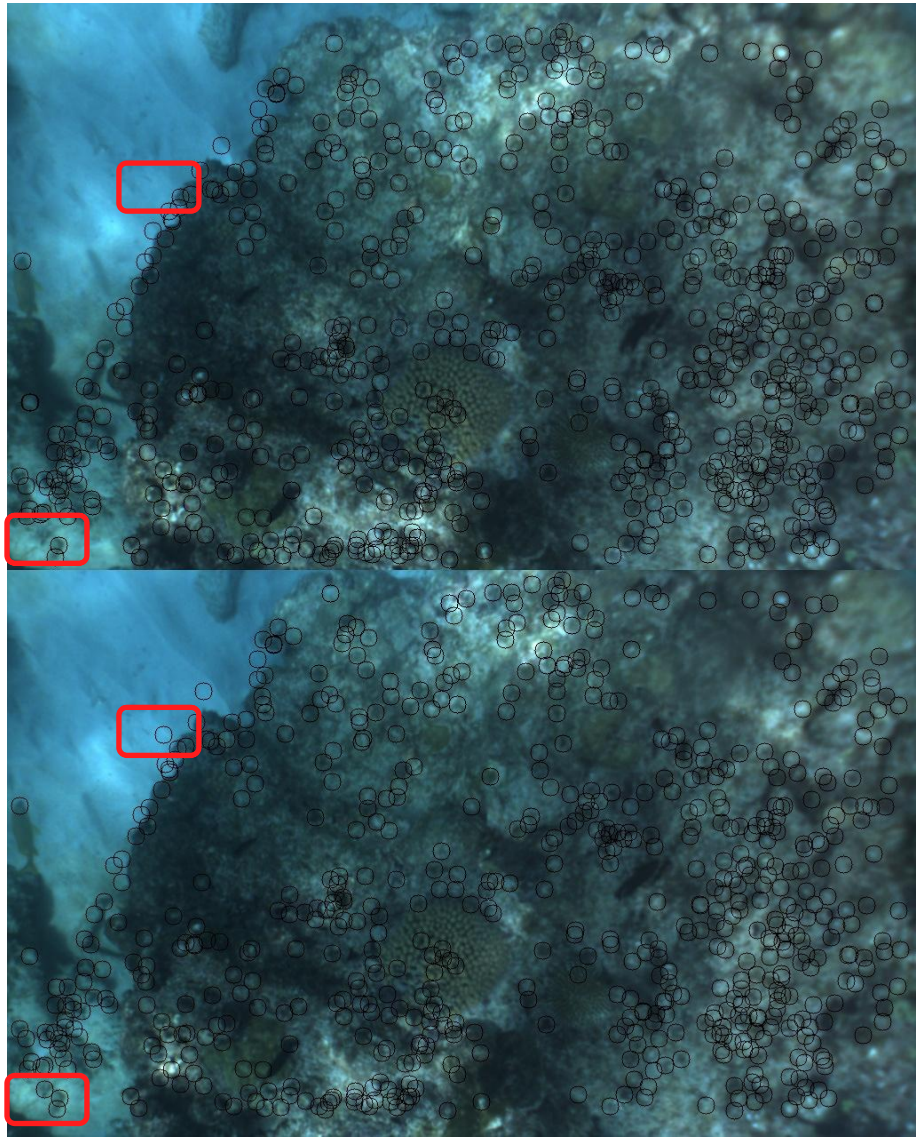
Feature matching (in the lab)



Feature matching (in the pool)

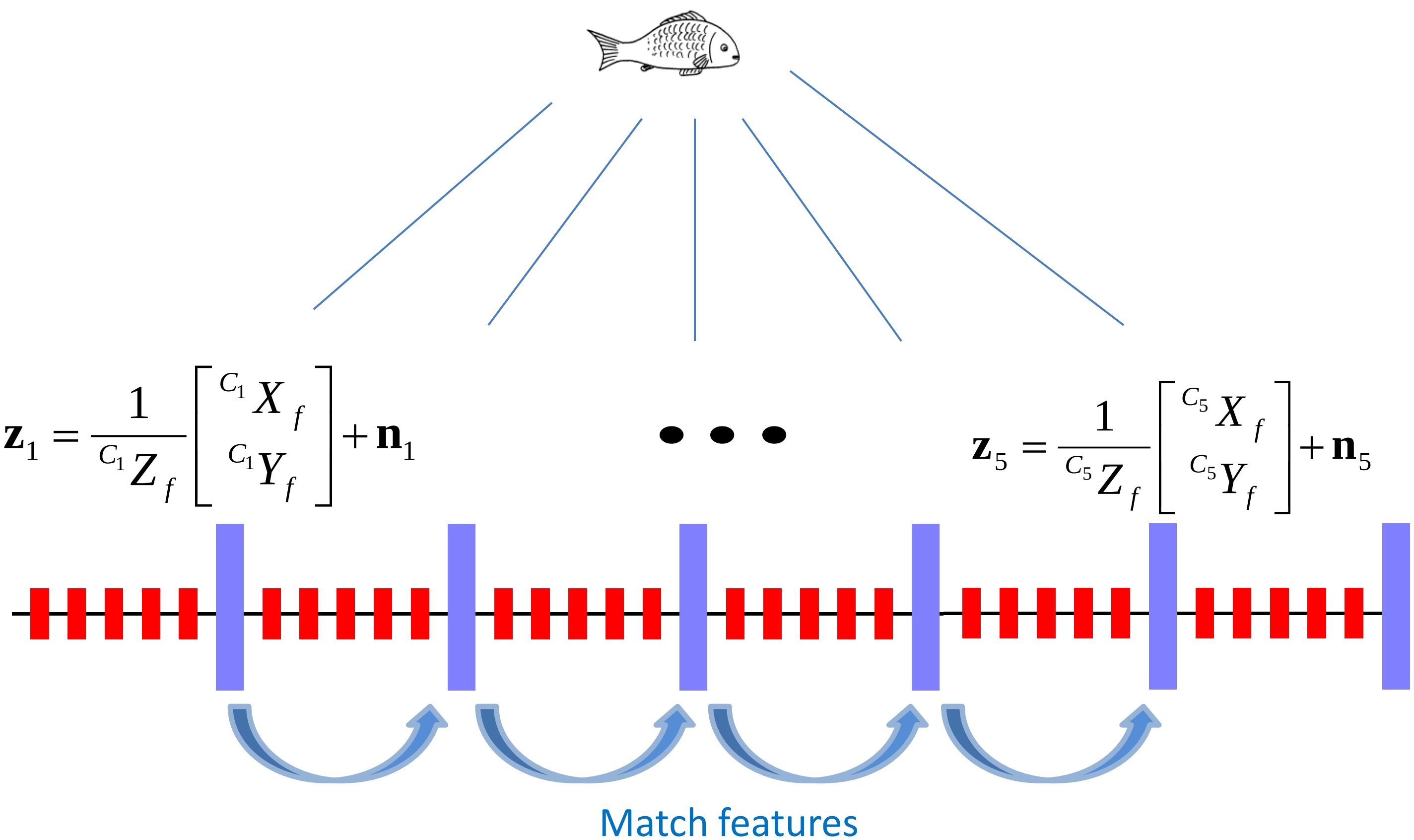


Feature matching (in the ocean)



Feature matching

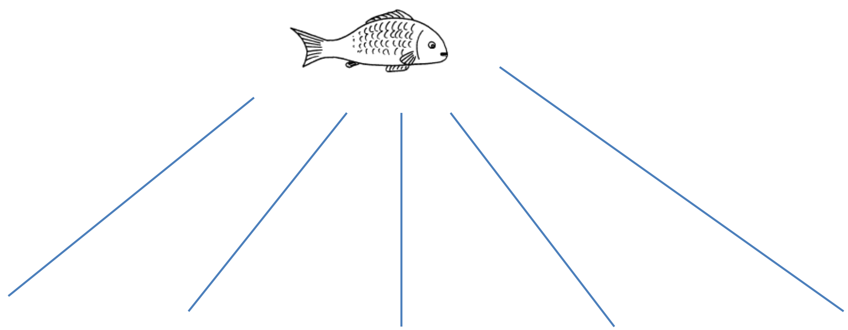
- Not very reliable, so outlier detection is necessary.
- If done carefully, it allows us to estimate the 3D position of a feature:



$$\mathbf{z}_1 = \frac{1}{{}^{C_1}Z_f}\begin{bmatrix} {}^{C_1}X_f \\ {}^{C_1}Y_f \end{bmatrix} + \mathbf{n}_1 \quad \bullet\bullet\bullet \quad \mathbf{z}_5 = \frac{1}{{}^{C_5}Z_f}\begin{bmatrix} {}^{C_5}X_f \\ {}^{C_5}Y_f \end{bmatrix} + \mathbf{n}_5$$

Structure Estimation

- We are searching for the 3D position of the fish in global coordinates, ${}^{G}\hat{\mathbf{p}}_f$

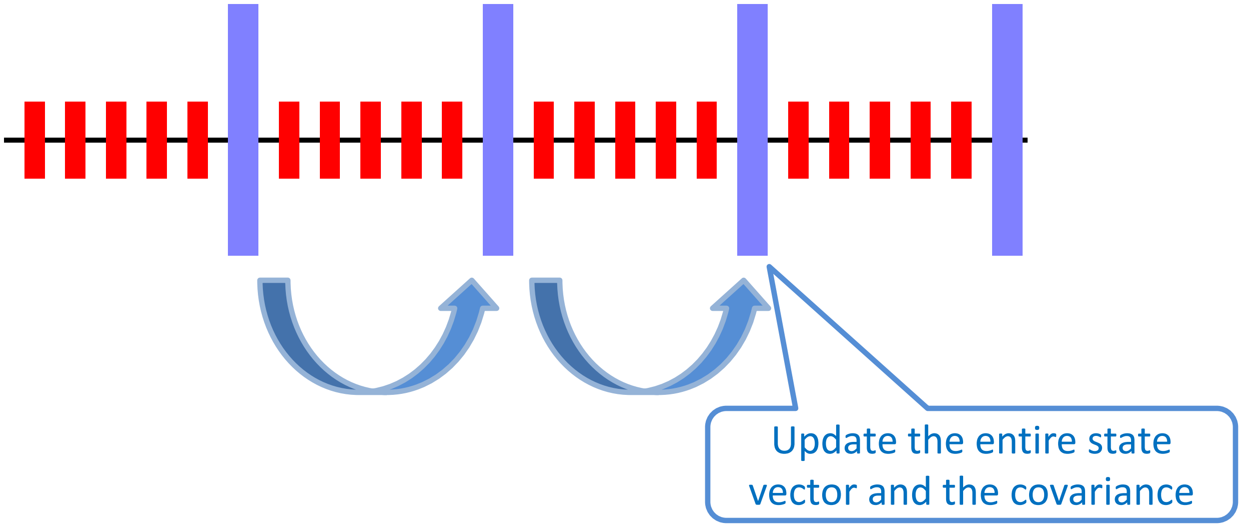


$$\mathbf{z}_1 = \frac{1}{{}^{C_1}Z_f}\begin{bmatrix} {}^{C_1}X_f \\ {}^{C_1}Y_f \end{bmatrix} + \mathbf{n}_1 \quad \bullet\bullet\bullet \quad \mathbf{z}_5 = \frac{1}{{}^{C_5}Z_f}\begin{bmatrix} {}^{C_5}X_f \\ {}^{C_5}Y_f \end{bmatrix} + \mathbf{n}_5$$

- If we work in global coordinates we have 3 unknowns ${}^{G}X_f, {}^{G}Y_f, {}^{G}Z_f$ and 5 measurements. We can solve the nonlinear problem using iterative minimization methods (e.g. Levenberg-Marquardt).

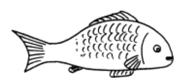
EKF Update

- Is done every time a feature stops being tracked.



EKF Update (the residual)

- The only source of correction we have is the camera.



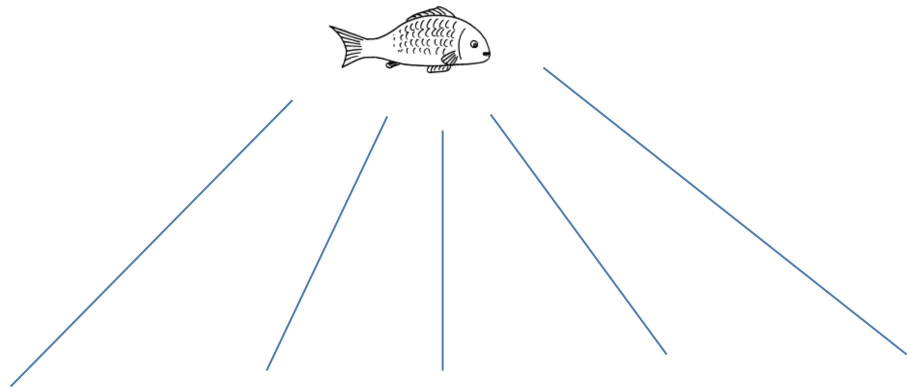
What we measured

$$\mathbf{z}_1 = \frac{1}{{}^{C_1}Z_f}\begin{bmatrix} {}^{C_1}X_f \\ {}^{C_1}Y_f \end{bmatrix} + \mathbf{n}_1 \quad \bullet\bullet\bullet \quad \mathbf{z}_5 = \frac{1}{{}^{C_5}Z_f}\begin{bmatrix} {}^{C_5}X_f \\ {}^{C_5}Y_f \end{bmatrix} + \mathbf{n}_5$$

$$\hat{\mathbf{z}}_1 = \frac{1}{{}^{C_1}\hat{Z}_f}\begin{bmatrix} {}^{C_1}\hat{X}_f \\ {}^{C_1}\hat{Y}_f \end{bmatrix} \quad \bullet\bullet\bullet \quad \hat{\mathbf{z}}_5 = \frac{1}{{}^{C_5}\hat{Z}_f}\begin{bmatrix} {}^{C_5}\hat{X}_f \\ {}^{C_5}\hat{Y}_f \end{bmatrix}$$

What we expected to measure after we estimated the 3D position of the fish

EKF Update (the residual)



$$\mathbf{r}_1 = \mathbf{z}_1 - \hat{\mathbf{z}}_1 \qquad \bullet\bullet\bullet \qquad \mathbf{r}_5 = \mathbf{z}_5 - \hat{\mathbf{z}}_5$$

$$\mathbf{r}_i = \mathbf{z}_i - \frac{1}{{}^{C_i}\hat{Z}_f} \begin{bmatrix} {}^{C_i}\hat{X}_f \\ {}^{C_i}\hat{Y}_f \end{bmatrix} \quad \text{where} \quad \begin{bmatrix} {}^{C_i}\hat{X}_f \\ {}^{C_i}\hat{Y}_f \\ {}^{C_i}\hat{Z}_f \end{bmatrix} = {}^{C_i}_{G}\hat{\mathbf{R}}({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_{C_i})$$

- Each residual is a nonlinear function of $({}^{C_i}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_i}, {}^{G}\mathbf{p}_f)$

EKF Update

- We linearize the residual $\mathbf{r}_i \approx \mathbf{H}_i \tilde{\mathbf{x}} + \mathbf{n}_i$ for each frame.
- We stack the residuals for all camera frames and for all features into one vector $\mathbf{r} \approx \mathbf{H}\tilde{\mathbf{x}} + \mathbf{n}$
- We apply the usual update equations of the Extended Kalman Filter:

$$\mathbf{K} = \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathrm{cov}(\mathbf{n}))^{-1}$$

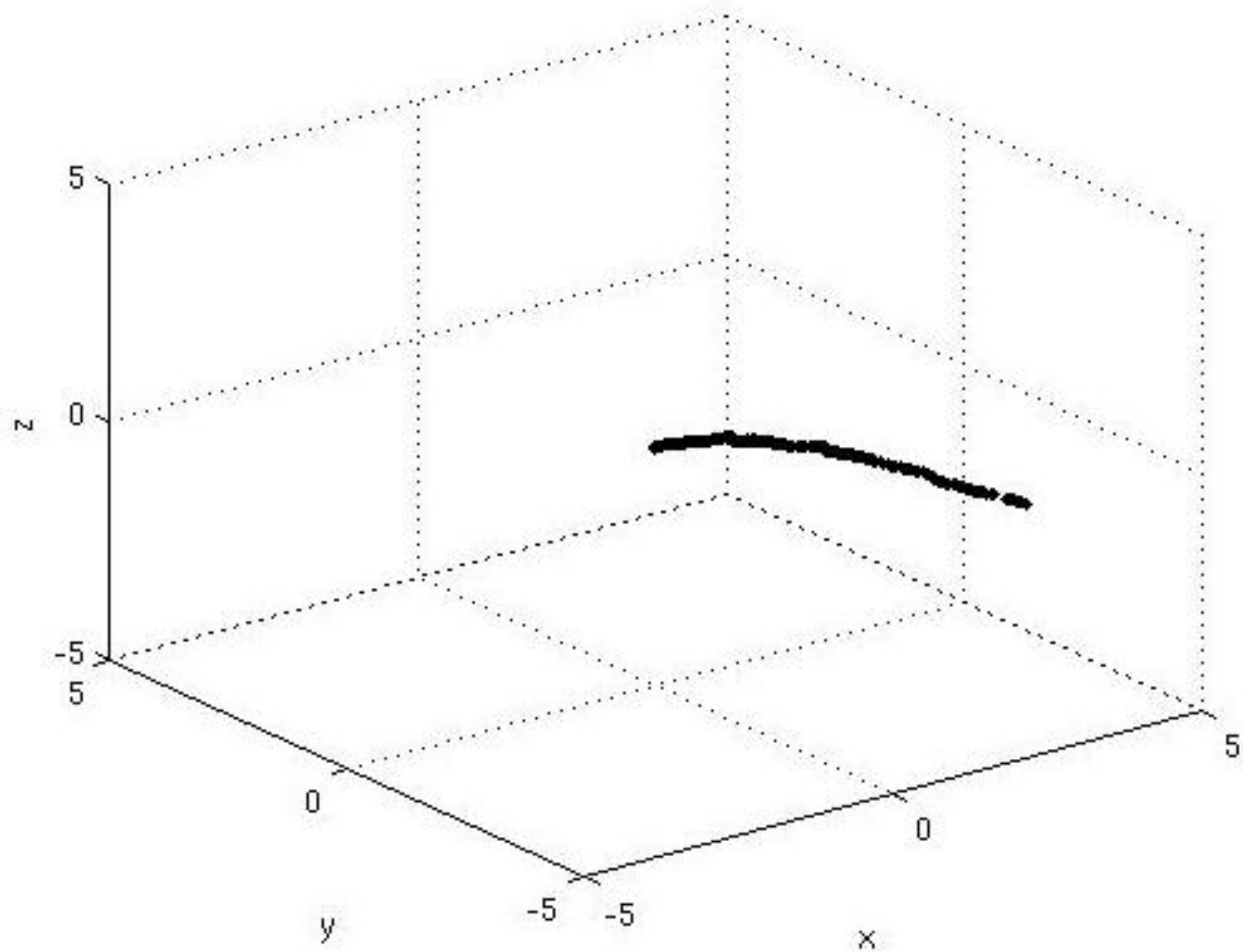
$$\Delta\mathbf{x} = \mathbf{K}\mathbf{r}$$

$$\mathbf{x}_{t+1|t+1} = \mathbf{x}_{t+1|t} + \Delta\mathbf{x}$$

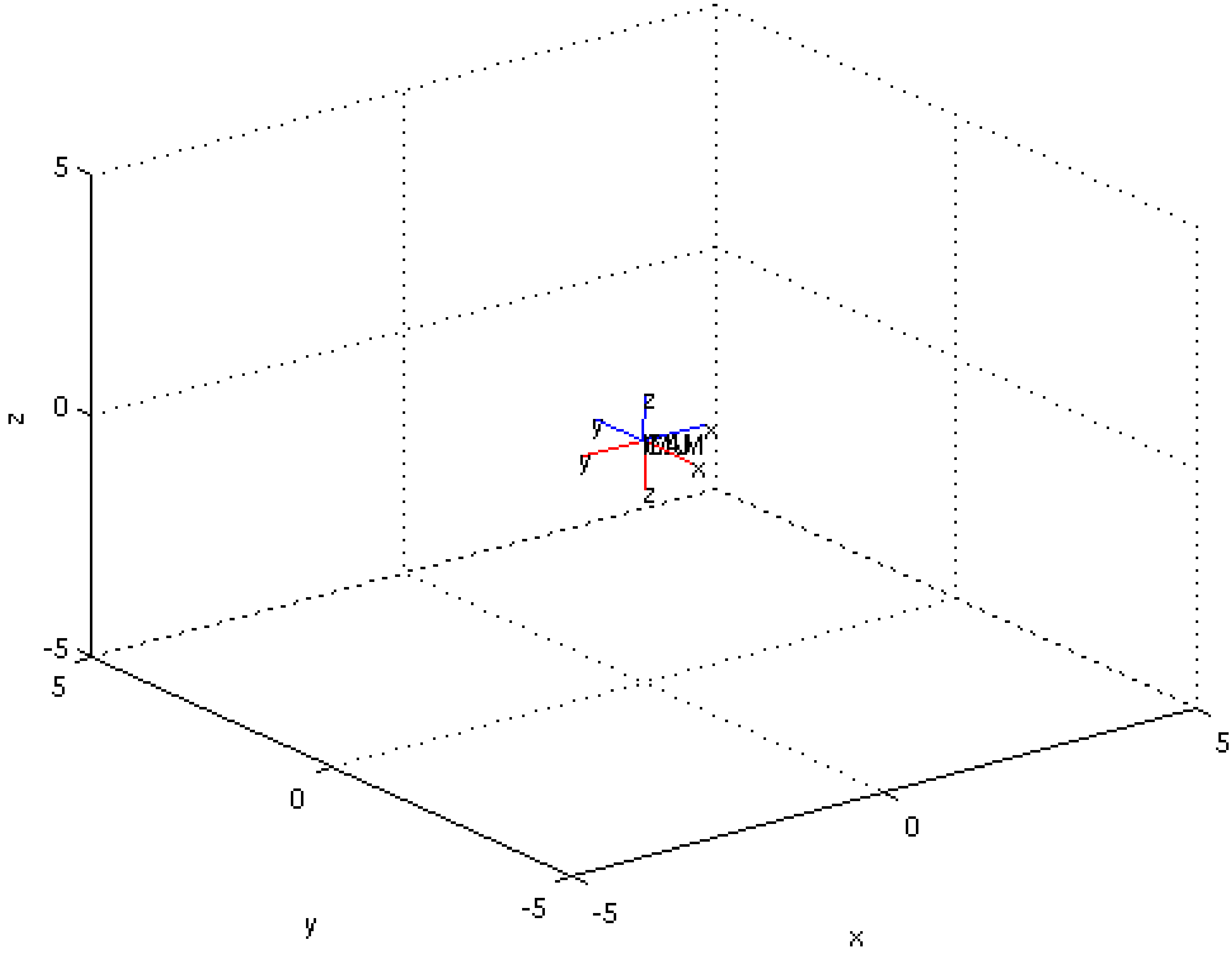
$$\mathbf{P}_{t+1|t+1} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_{t+1|t}$$

Lab experiment

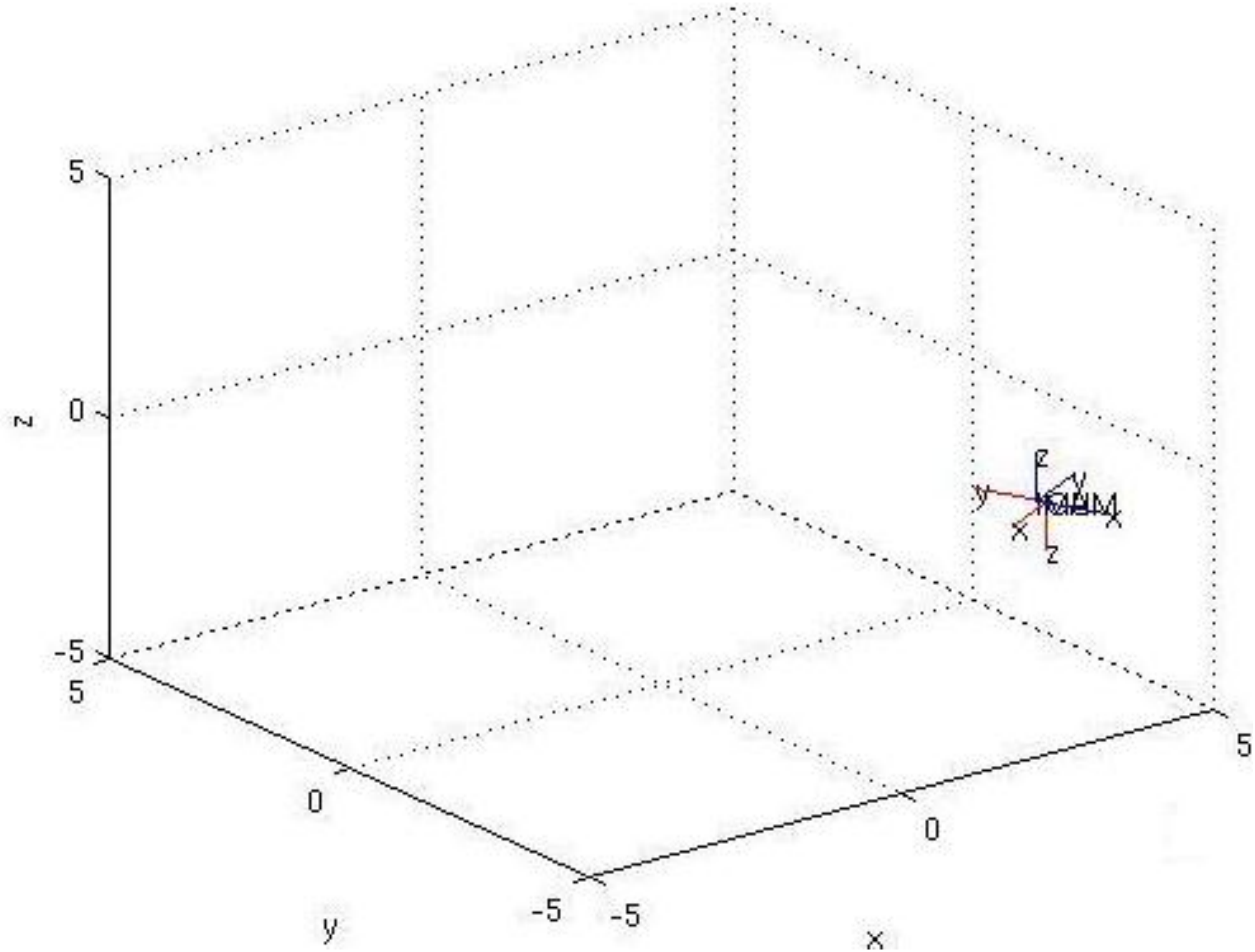
Estimated trajectory



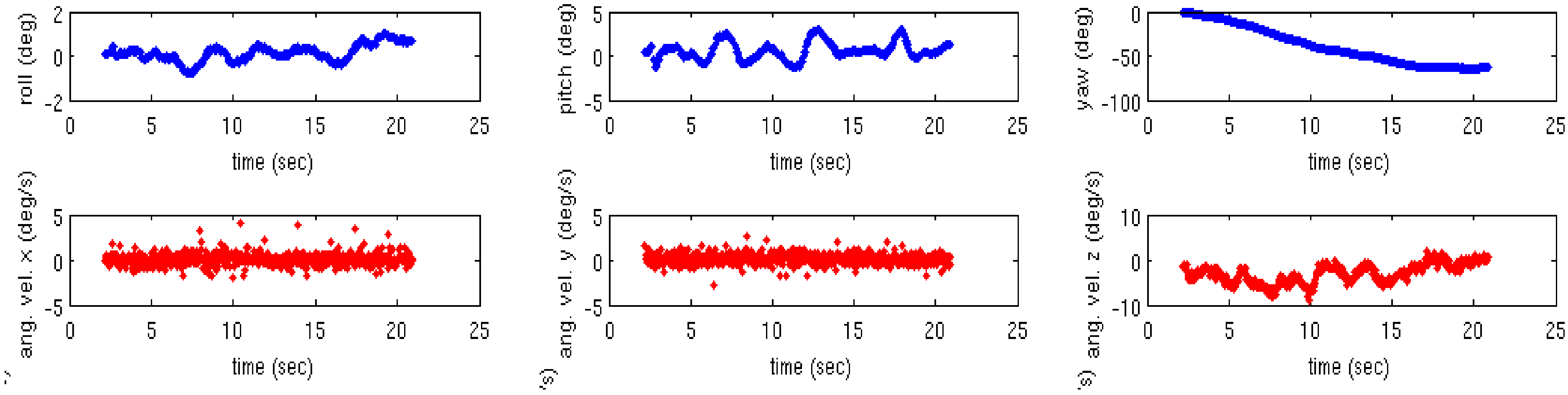
Initial pose



Estimated final pose



Orientation estimates



$\tilde{r} \approx 0^0$

$r \approx 0^0$

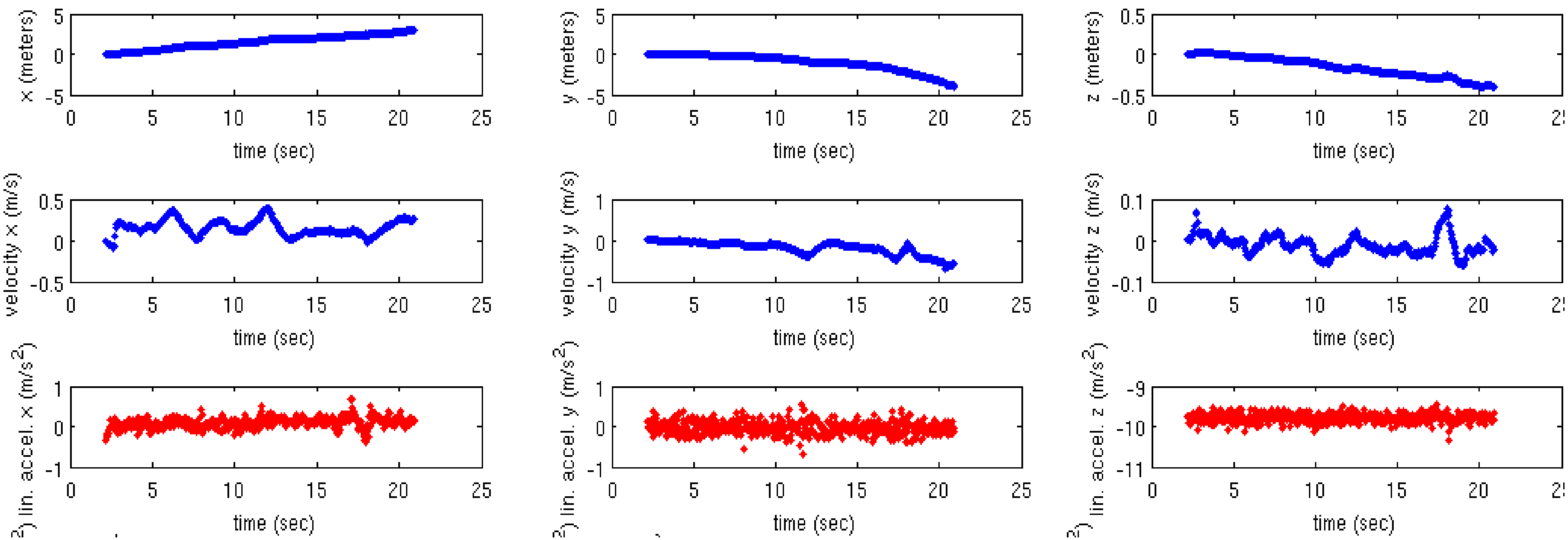
$\tilde{p} \approx 2^0$

$p \approx 0^0$

$\tilde{y} \approx 5^0$

$y \approx -70^0$

Velocity and position estimates



$\tilde{x} \approx 0.3m$

$x \approx 2.2m$

$\tilde{y} \approx 1m$

$y \approx 2m$

$\tilde{z} \approx 0.3m$

$z \approx 0m$

Concluding remarks

- The algorithm is very sensitive to the visual motion estimation.

- Current work:
- Figure out how to better estimate 3D locations of features to improve the position and velocity estimates.
- Implement the algorithm so that it runs in real time.

- Future work:
- Integrate it with the robot controller to enable the robot to perform requested trajectories
- Examine if bundle adjustment methods will make the estimate more robust .

Thank you

- Questions, feedback, and criticisms would be appreciated!

Optional: Propagation (covariance)

- From $\frac{d}{dt}\tilde{\mathbf{x}}_{IMU}(t) = \mathbf{F}\tilde{\mathbf{x}}_{IMU}(t) + \mathbf{G}[\mathbf{n}_g \;\; \dot{\mathbf{b}}_g \;\; \mathbf{n}_a \;\; \dot{\mathbf{b}}_a]^T$ we get

$$\frac{d}{dt}\mathbf{P}_{II}(t) = \mathbf{F}\mathbf{P}_{II}(t) + \mathbf{P}_{II}(t)\mathbf{F}^T + \mathbf{G}\,\mathrm{cov}([\mathbf{n}_g \;\; \dot{\mathbf{b}}_g \;\; \mathbf{n}_a \;\; \dot{\mathbf{b}}_a])\mathbf{G}^T$$

- We can numerically integrate $\frac{d}{dt}\mathbf{P}_{II}(t)$ in the propagation interval to obtain $\mathbf{P}_{II}$.

- And then compute the IMU-camera covariance:

$$\begin{aligned}
\mathbf{P}_{IC}(t+1\,|\,t) &= \mathrm{cov}(\tilde{\mathbf{x}}_{IMU}(t+1), \tilde{\mathbf{x}}_{CAM}(t)) \\
&= \mathrm{cov}(e^{\mathbf{F}dt}\tilde{\mathbf{x}}_{IMU}(t), \tilde{\mathbf{x}}_{CAM}(t)) \\
&= e^{\mathbf{F}dt}\mathbf{P}_{IC}(t\,|\,t)
\end{aligned}$$