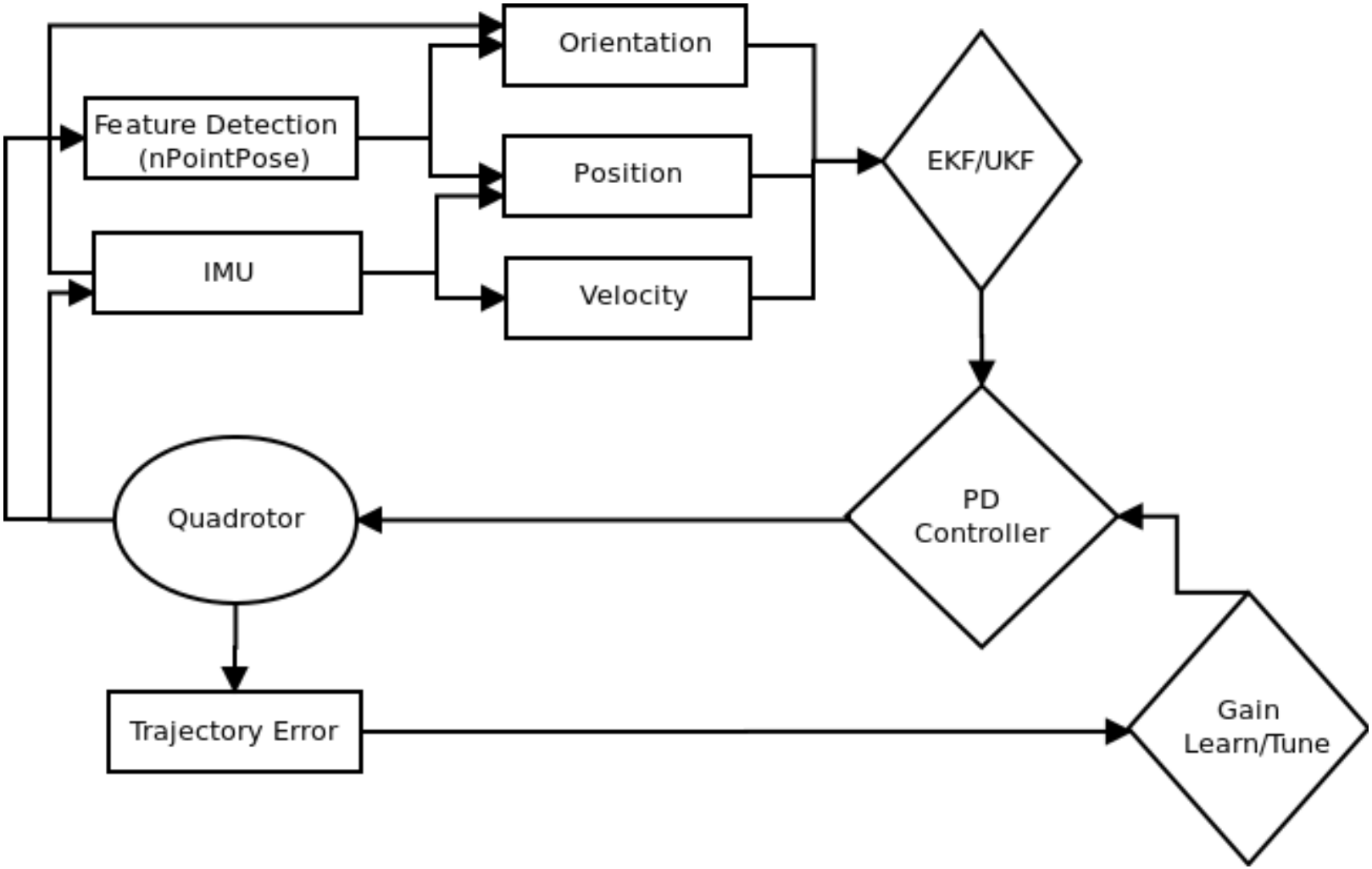
Quadrotor State Estimation and Gain Learning

Fredy Monterroza

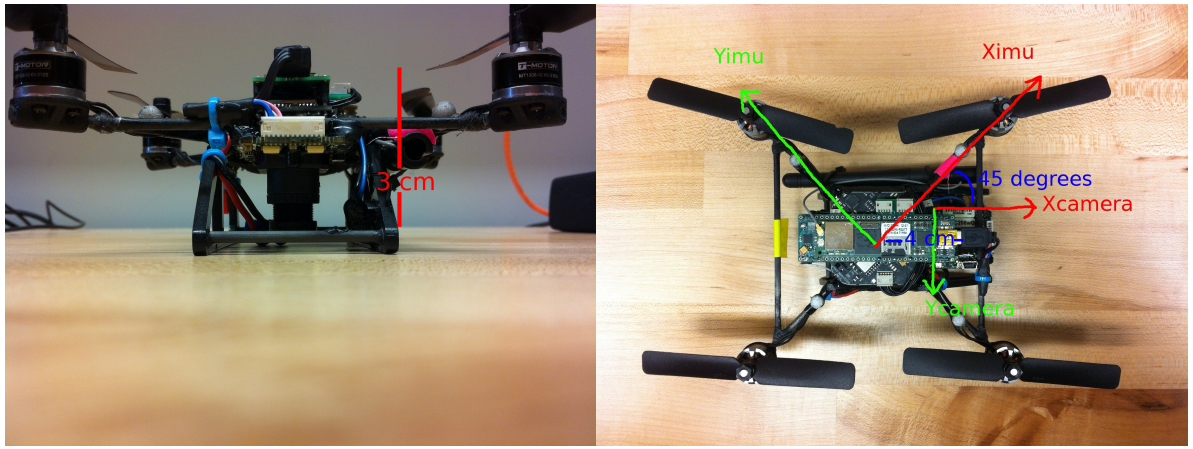
May 9, 2014

Introduction

System Diagram



Quadrotor



(a) Side (b) Top

Figure: nanoPlus Quadrotor

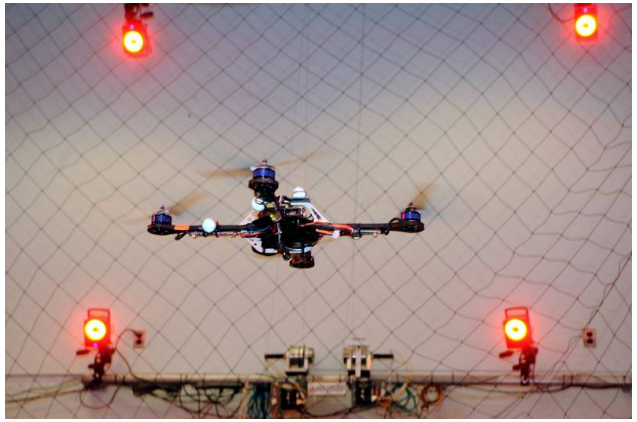


Figure: Reflective Markers on Quadrotor tracked by Vicon

N-Point Pose Algorithm

April Tags and Flight Video

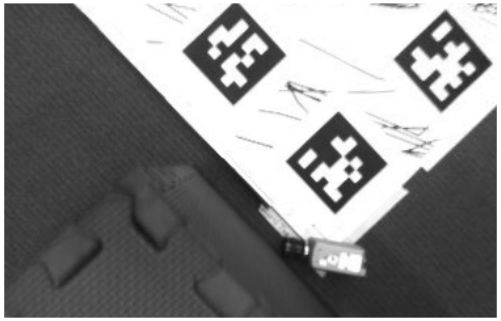


Figure: Image 536

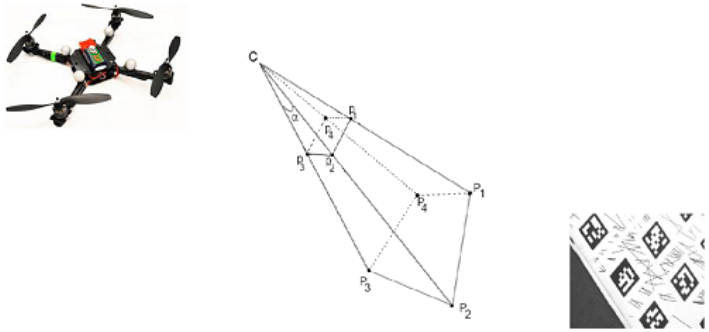


Figure: Charicature of N-Point Pose Scenario

N-Point Pose Algorithm

Linear System

Relationship between world coordinates and calibrated camera frame coordinates:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{c}X \\ {}^{c}Y \\ {}^{c}Z \end{bmatrix} = {}^{c}R_{w} \begin{bmatrix} {}^{w}X \\ {}^{w}Y \\ {}^{w}Z \end{bmatrix} + {}^{c}T_{w}$$

which becomes:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{c}R_{w} & {}^{c}T_{w} \end{bmatrix} \begin{bmatrix} {}^{w}X \\ {}^{w}Y \\ {}^{w}Z \\ 1 \end{bmatrix}$$

where x, y are calibrated camera plane coordinates obtained from the product of camera matrix inverse K^{-1} and pixels u, v:

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = K^{-1} \begin{bmatrix} u \\ u \\ 1 \end{bmatrix}$$

N-Point Pose Algorithm

2nx9 System of Equations

Since the points all lie on the April Tag Mat, ${}^{w}Z = 0$, thus we can write the world frame to calibrated camera frame relationship as:

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = A \begin{bmatrix} {}^{w}X \\ {}^{w}Y \\ 1 \end{bmatrix}$$

where $A = [r_1 r_2 {}^{c}T_w]$, $r_3 = r_1 \times r_2$. $r_{1,2}$ are the first two columns of the rotation matrix ${}^{c}R_w$. Finally, the above can be re-written as a homogeneous system of equations:

$$\begin{bmatrix} -X & 0 & xX & -Y & 0 & xY & -1 & 0 & x \\ 0 & -X & yX & 0 & -Y & yY & 0 & -1 & y \end{bmatrix} \begin{bmatrix} r_1 \\ r_2 \\ T \end{bmatrix} = 0$$

N-Point Pose Algorithm

Method of Solving Homogeneous System of Equations: SVD

The A matrix is obtained by finding a vector that lies in the null space of the 2nx9 Matrix using Singular Value Decomposition. For this purpose, the rightmost singular vector of $[\sim\sim\ V] = svd(2nx9)$ is taken and reshape(*,3,3)'ed to give the A transformation. The transformation between world and camera is then:

$$^{c}R_{w} = [r_1 r_2 r_3\, ^{c}T_{w}]$$

$$Pose of\, Quadrotor in World Frame = (Camera To Robot * {}^{c}R_{w})^{-1}$$

while not forgetting to normalize the A matrix to account for the λ scaling factor.

N-Point Pose Algorithm

Measurement Update Data: nPP Output

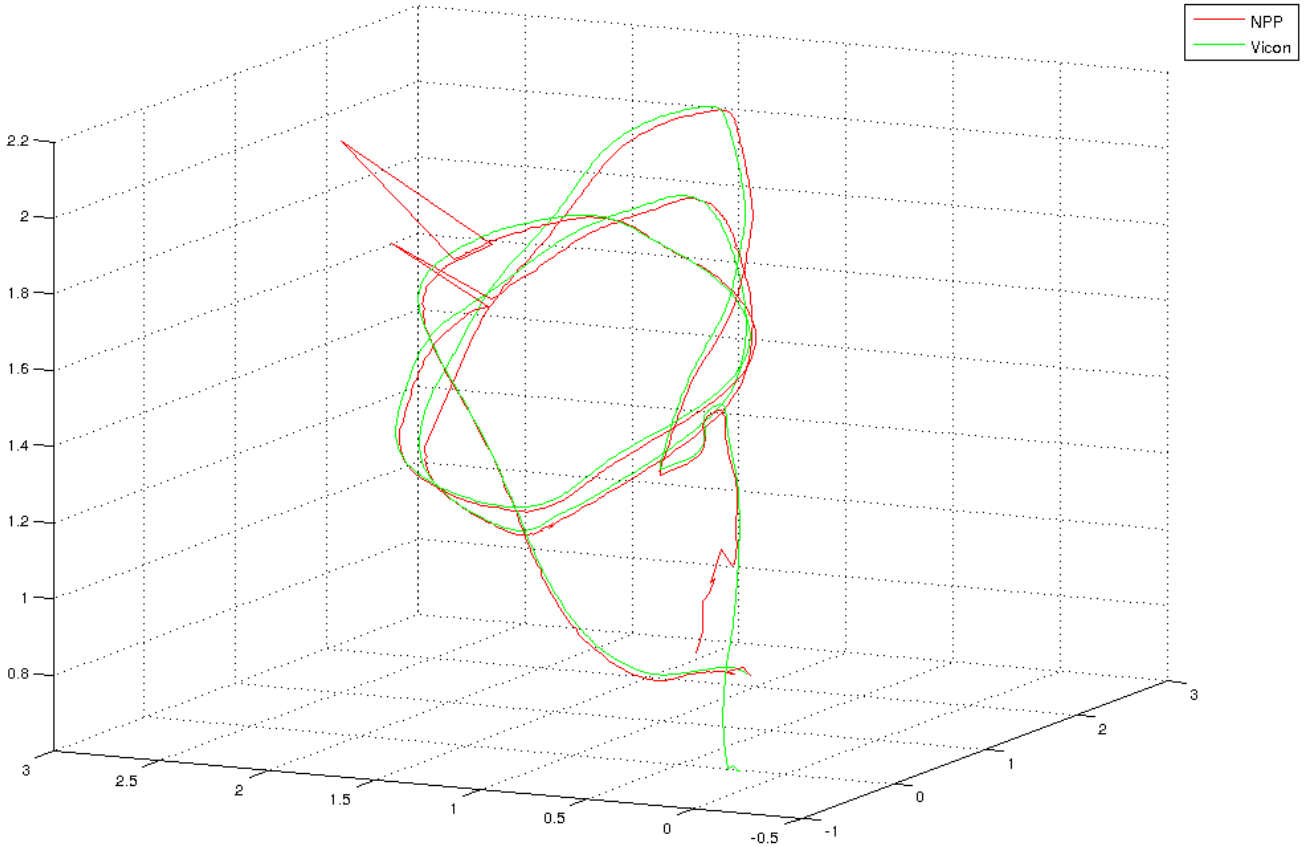


Figure: Vicon (green) and Measurement Update from nPP (red)

Kalman Filter

Motivation

To fly autonomously, without input from the Vicon System, the quadrotor would rely on the onboard IMU and Camera.

Problems:

- Measurement Noise: N-Point Pose
- Sensor Bias: IMU Heat Up Changes Characteristics (Δt^2)
- Wireless Transmission of Data: UKF/Dropped Packets

Kalman Filter

Extended and Unscented Variants

Extended Kalman Filter:

- Jacobian Computation
- Less Sensitive to Noise due to linearization.
- Approximation Correct near hover.
- Simple computation of mean, covariance, a-posteriori estimate.

Unscented Kalman Filter

- Non Linear Dynamics Preserved $\mapsto$ more accurate state.
- No Need for computation of partial derivatives.
- Potential for tracking more aggressive maneuvers (quaternions).
- Additional computation required for quaternion mean and a-posteriori state estimate.

Kalman Filter

Quadrotor Dynamics

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & ba_x & ba_y & ba_z \end{bmatrix}^T$$

The dynamics $g(X, u, \epsilon)$ take the form:

$$\begin{bmatrix} x_t \\ y_t \\ z_t \end{bmatrix} = \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ z_{t-1} \end{bmatrix} + \begin{bmatrix} \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ \dot{z}_{t-1} \end{bmatrix} * \Delta t + \begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} * \frac{\Delta t^2}{2}$$

$$\begin{bmatrix} \dot{x}_t \\ \dot{y}_t \\ \dot{z}_t \end{bmatrix} = \begin{bmatrix} \dot{x}_{t-1} \\ \dot{y}_{t-1} \\ \dot{z}_{t-1} \end{bmatrix} + \begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} * \Delta t$$

$$\begin{bmatrix} \phi_t \\ \theta_t \\ \psi_t \end{bmatrix} =$$

$$RotToRPY_ZXY(R(\phi_{t-1}, \theta_{t-1}, \psi_{t-1}) * \Delta R(\omega_x, \omega_y, \omega_z, \Delta t))$$

Kalman Filter

Quadrotor Dynamics

The acceleration from IMU is transformed into world frame and bias subtracted.

$$\begin{bmatrix} \ddot{x}_t \\ \ddot{y}_t \\ \ddot{z}_t \end{bmatrix} = R(\phi_{t-1}, \theta_{t-1}, \psi_{t-1}) * \begin{bmatrix} accel_x \\ accel_y \\ accel_z \end{bmatrix} - \begin{bmatrix} bax_t \\ bay_t \\ baz_t \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 9.81 \end{bmatrix}$$

The rotation matrix along ZXY:

$R(\phi, \theta, \psi) = RPYtoRot_ZXY(\phi, \theta, \psi)$

The incremental change in rotation returned by the gyroscope of the IMU:

$\Delta R(\omega_x, \omega_y, \omega_z, \Delta t) = RPYtoRot_ZXY(\omega_x \Delta t, \omega_y \Delta t, \omega_z \Delta t)$

Kalman Filter

Accelerometer Bias

"Random Walk" model to track the bias of the accelerometer:

$$\begin{bmatrix} bax_t \\ bay_t \\ baz_t \end{bmatrix} = \begin{bmatrix} bax_{t-1} \\ bay_{t-1} \\ baz_{t_1} \end{bmatrix} + \begin{bmatrix} \epsilon_{bax} \\ \epsilon_{bay} \\ \epsilon_{baz} \end{bmatrix} * \Delta t$$

where ϵ_{bax} ϵ_{bay} ϵ_{baz} is sampled according to a 0-mean Gaussian noise with variance determined as the IMU lay still.

Kalman Filter

Measurement and Control Input

The measurement update was taken from the N-Point Pose and is

$$Z = \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{bmatrix}$$

The expected measurement is obtained according to:
$Z = HX + noise : H = \left.\frac{\partial h}{\partial x_{t-1}}\right|_{\mu_t}$ and in this case, since the expected measurement is a subset of the apriori state estimate, H is simply a matrix that extracts the respective components.
The input to the system is $U = \begin{bmatrix} accel_x & accel_y & accel_z & w_x & w_y & w_z \end{bmatrix}$

Kalman Filter

Extended Kalman Filter

The state is a non-linear function that can be approximated as:

$$X_t = g(X_{t-1}, u_t, \epsilon_t) \approx g(\mu_{t-1}, u_t, 0) + G_t + L_t$$

where

$$G_t = \left.\frac{\partial g}{\partial x_{t-1}}\right|_{\mu_t - 1, u_t, 0} L_t = \left.\frac{\partial g}{\partial \epsilon_{t-1}}\right|_{\mu_t - 1, u_t, 0}$$

Kalman Filter

Covariance Initialization

Process Covariance (R): Variance as IMU lay-still.

Measurement Covariance (Q): Standard mean error between vicon ground truth and N-Point Pose.

... and some hand-tuning.

Kalman Filter

Extended Kalman Filter

$$\bar{X}_t = g(X_{t-1}, u_t, 0)$$

$$\bar{\Sigma}_t = G\Sigma_{t-1}G^T + LRL^T$$

$$K = \bar{\Sigma}_t H^T (H\bar{\Sigma}_t H^T + Q)^{-1}$$

$$X_t = \bar{X}_t + K(Z - H\bar{X}_t)$$

$$\Sigma_t = (I - KH)\bar{\Sigma}_t$$

Kalman Filter

Results: Main EKF Loop

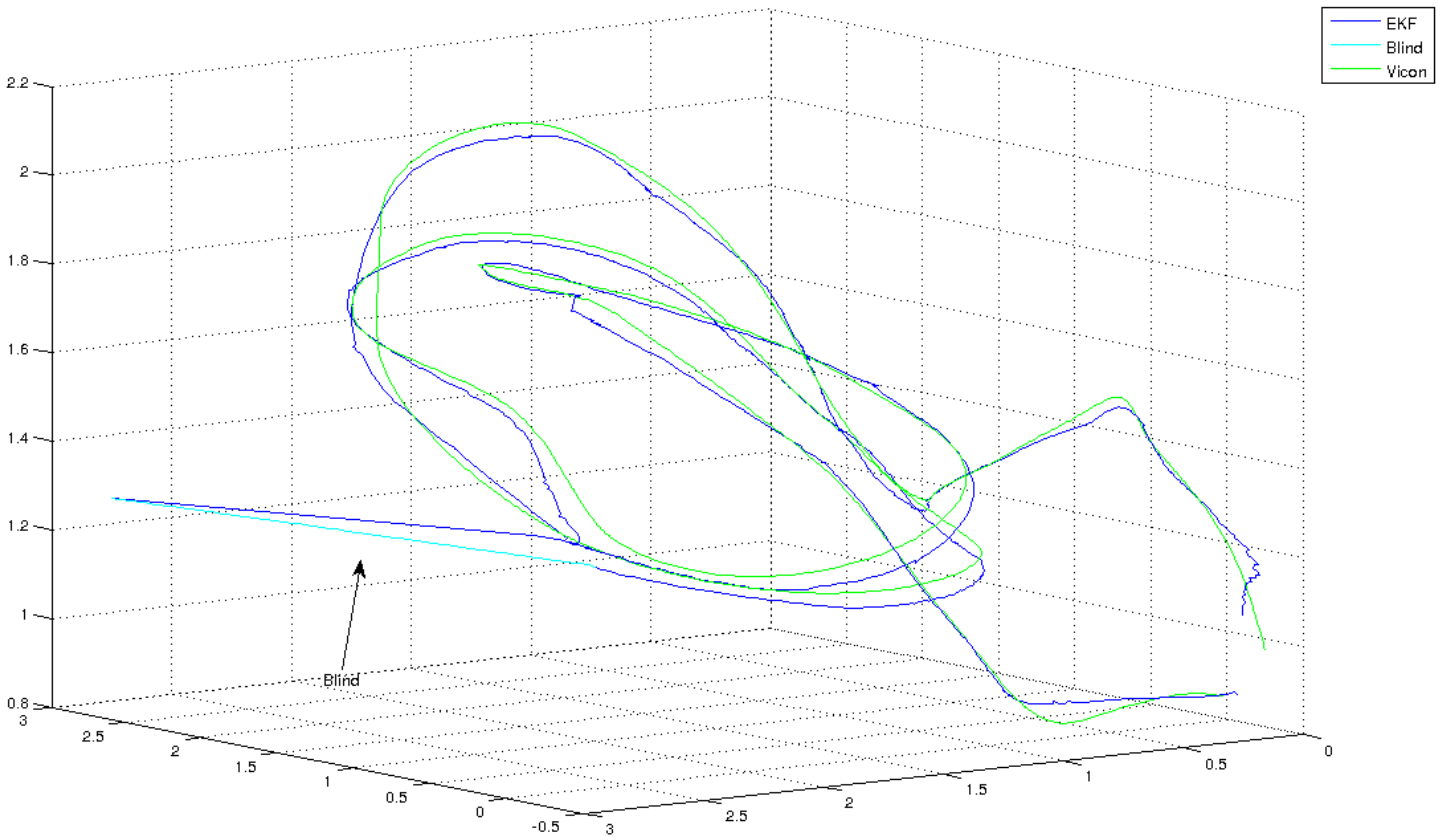


Figure: EKF (Blue) and Vicon (Green)

Kalman Filter

Results: Main EKF Loop

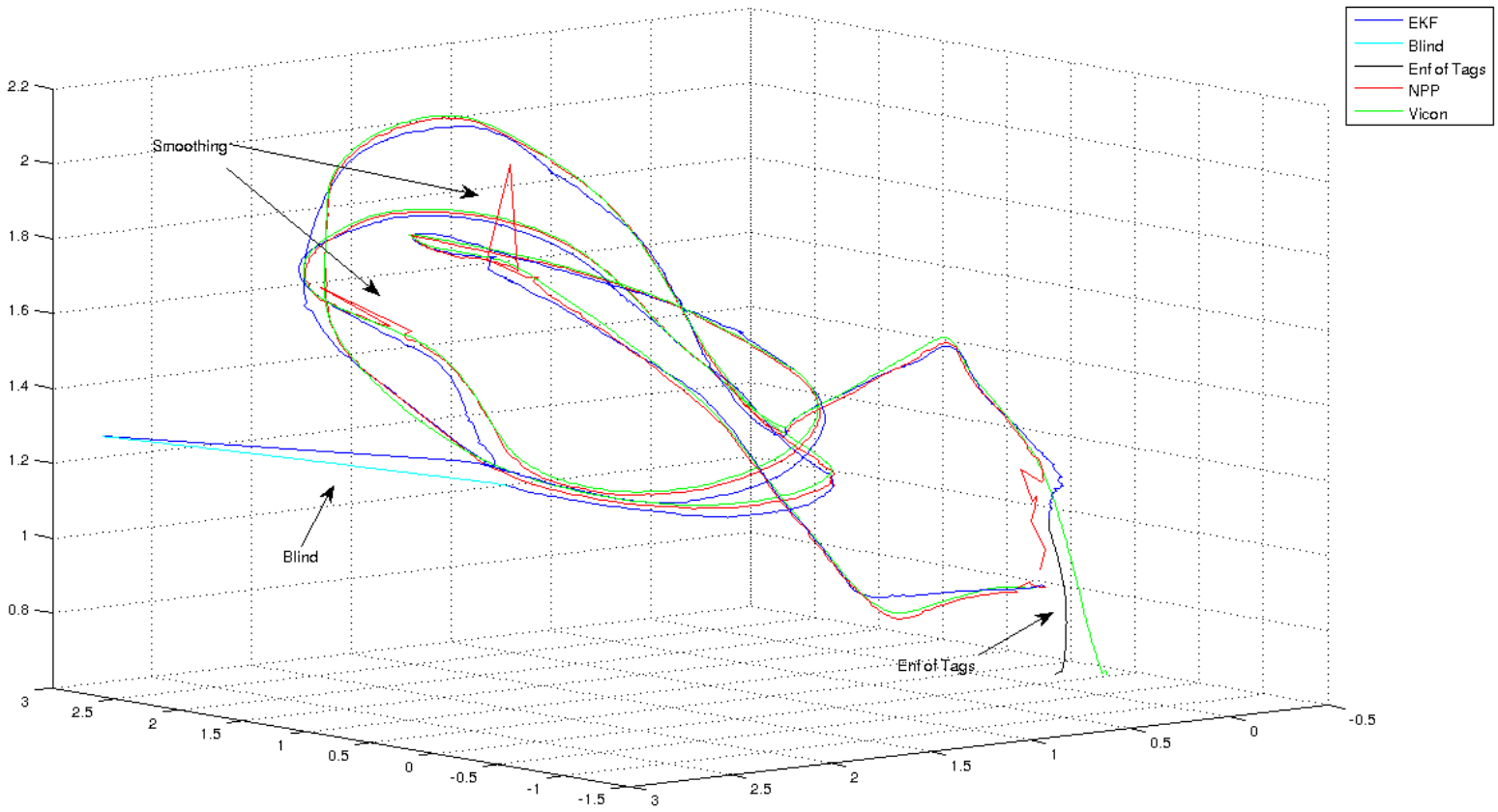


Figure: EKF (Blue), Vicon (Green) and Measurement (Red)

Kalman Filter

Results: Main EKF Loop

Using standard mean error of trajectory:

- 9 Dim EKF (No Accelerometer Bias Tracking): .0886m
- 12 Dim EKF (Full State): .0812m

Benefit: Larger controller gains to better track a trajectory.

Kalman Filter

Unscented Kalman Filter

- Maintain Non-Linear dynamics from which state is obtained.
- Would like to incorporate measurements (assumed to have Gaussian Noise) to correct the state estimate obtained from process update.
- Approximate non-linear transformation by building a Gaussian that represents it.

Kalman Filter

Unscented Kalman Filter

- Obtain Sigma Points (samples that contain a mean and covariance of the non-linear function) by:
  - Cholesky Decomposition of (Previous State Estimate Covariance + Process Covariance)
  - Form 2N set of vectors: [$\sqrt{N}$chol, -$\sqrt{N}$chol], where N is the size of state vector.
  - Add these set of points to the previous state estimate to obtain a sampling about that mean.
- Propagate these sigma points through the non-linear dynamics to obtain the set: W_i

Kalman Filter

Unscented Kalman Filter

- Build a Gaussian:
  - Compute Mean: apriori state estimate ($\bar{X}$).
  - Compute Covariance: apriori state estimate covariance ($\bar{P}$).
- Build expected measurement:
  - From the sigma points, extract measurement subset (Z_i).
  - Compute Mean.
- Compute Innovation Vector: Difference between actual and expected measurement (V_k).

Kalman Filter

Unscented Kalman Filter

- Compute P_{zz} using Z_i set.
- Compute Innovation Covariance:
  - $P_{vv} = P_{zz} +$ Measurement Cov
- Compute Cross Correlation (P_{xz}): Covariance between propagated sigma points (W_i) and measurement subset (Z_i).

- Compute Kalman Gain: $K = P_{xz} P^{-}$
- Compute updated state estimate: $X = \bar{X} + KV_k$
- Compute updated state estimate covariance: $P = \bar{P} - KP_{vv}K^{-}$

Kalman Filter

Unscented Kalman Filter Result

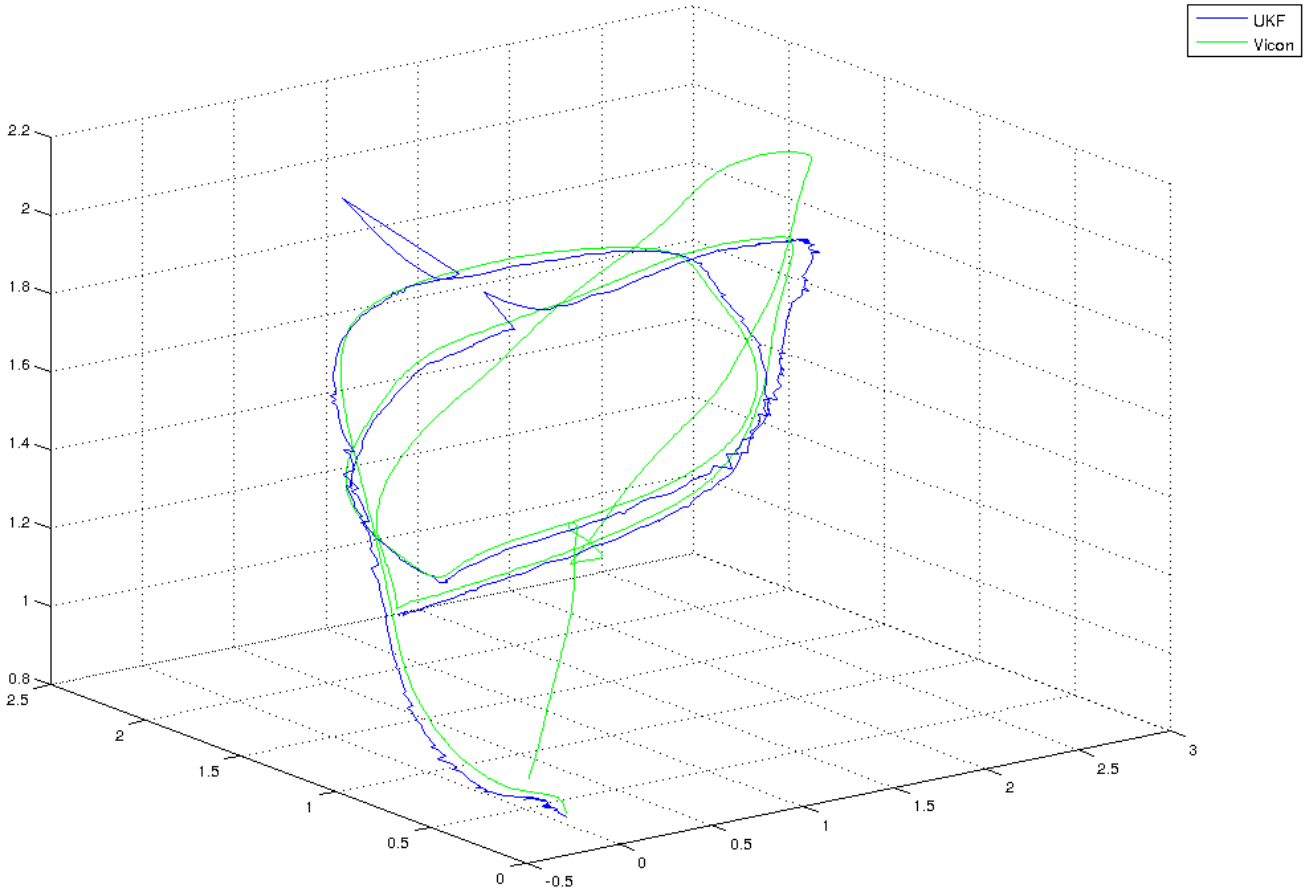


Figure: UKF (Blue) and Vicon (Green)

Kalman Filter

Unscented Kalman Filter Result

When data is available, UKF performs well. For the above, the standard mean error was .0296m (as opposed to .0364m).

Problems:

- Process and Measurement Covariance should be chosen better. (tradeoff)
- Data Transmitted wirelessly $\mapsto$ poor state estimate:
  - Loss of data packets (no measurement update or process update)
  - Norm of State Covariance varied more.
  - Poor apriori state estimate from sampled sigma points.
- UKF more susceptible to noise, since the accelerometer bias is double integrated (Δt^2).

Gain Learning

The last endeavour was an attempt at updating the PD controller gains according trajectory error. Considered:

- Path Integral Reinforcement Learning
- PD-Type Iterative Learning Control: Store Control Input/Gains along discretization of path.

Resorted to:

- $kp_i = kp_i + \rho_p errorPos$
- $kd_i = kd_i + \rho_d errorVel$

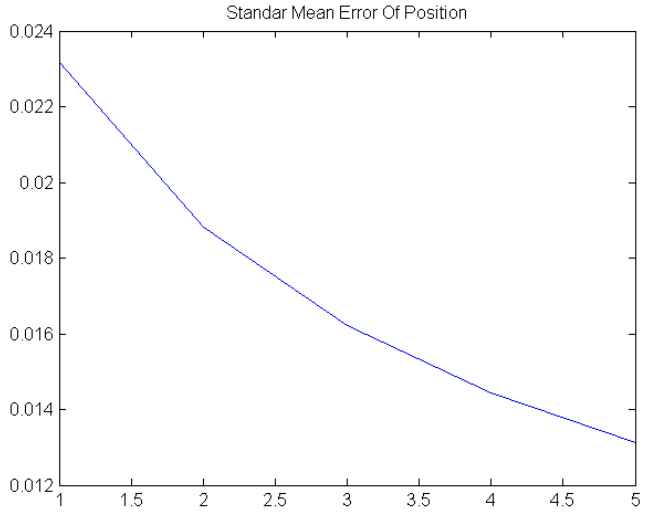


Figure: After 5 Iterations

Thank You!

References:
L.Quan, Linear N-Point Camera Pose Determination, IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 21, no.7, July 1999
Julier, SJ. and Uhlmann, J.K., Unscented Filtering and Nonlinear Estimation Proceedings of the IEEE, Vol. 92, No. 3, pp.401-422, 2004.
Kraft, Edgar, A Quaternion-based Unscented Kalman Filter for Orientation Tracking, University of Bonn.