Estimation Module

Dan Gaines   Stergios Roumeliotis   Issa Nesnas

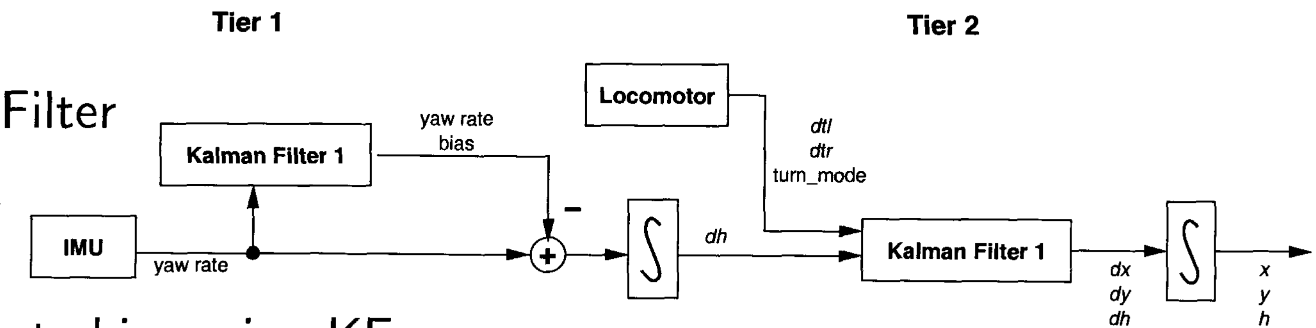
Initial estimator implementation by:
Ashitey Trebi-Ollennu   Eric Baumgartner

Overview of Current Estimator

Role: Provide 3DOF pose estimation for rover (x, y heading)

Input:

- wheel encoders
- Inertial Measuring Unit (IMU)
  - currently using only one gyro (yaw rate)



Design: Two-tier Kalman Filter

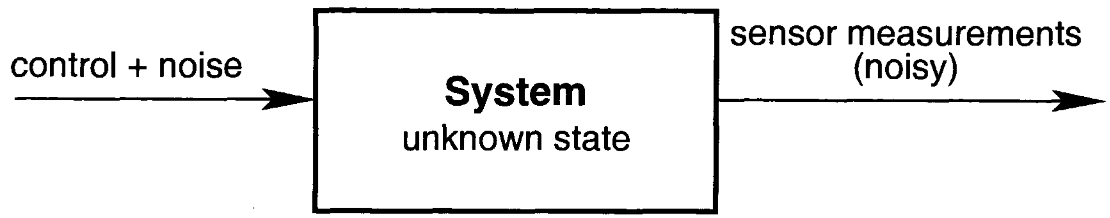
Tier 1: while stationary

- uses IMU yaw rate
- estimate IMU yaw rate bias using KF

Tier 2: while moving

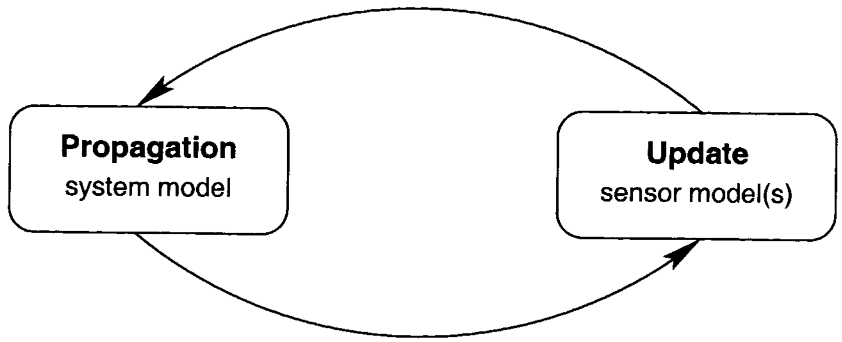
- uses corrected IMU yaw rate and wheel encoders
- fuse IMU yaw rate and encoder estimations within KF
- estimate change in rover x, y and heading
- integrate x, y and heading to keep track of 3DOF pose

Overview of Kalman Filtering



- System model predicts next state
- Measurements give evidence about state
- Both sources of information are noisy
- Kalman Filter fuses information to provide optimal state estimate
  - estimated state minimizes the mean-square estimation error
  - keeps track of uncertainty in state estimate

Kalman Filtering Algorithm



Propagation:

- using: current state, system model
- produce: estimate of next state
- system model: how state changes as a function of dynamic / kinematic model and control input

Update:

- using: measurements and sensor models
- produce: updated state based on sensor readings
- sensor model: given a state, what sensor reading(s) are expected

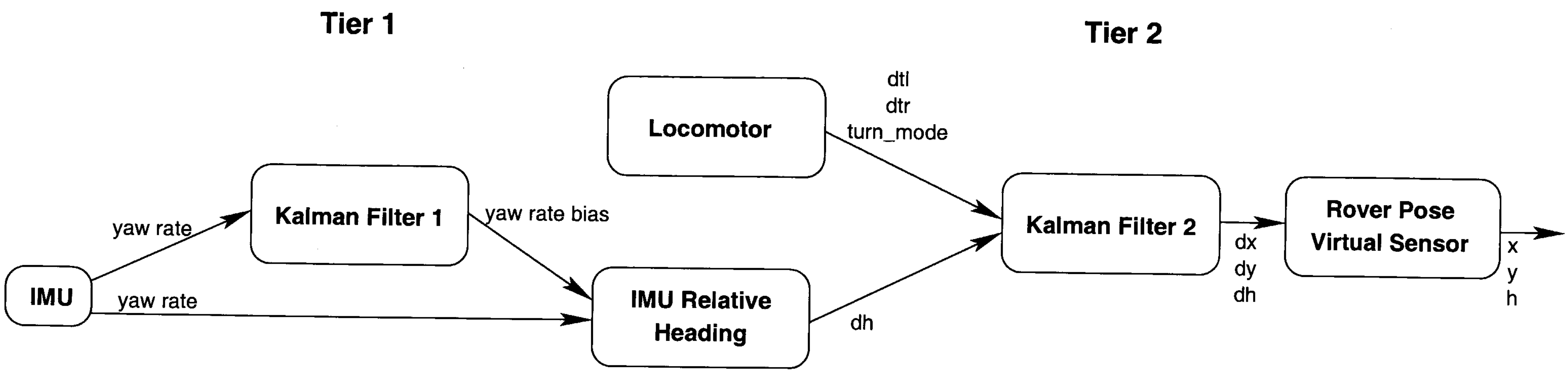
Covariance Matrix

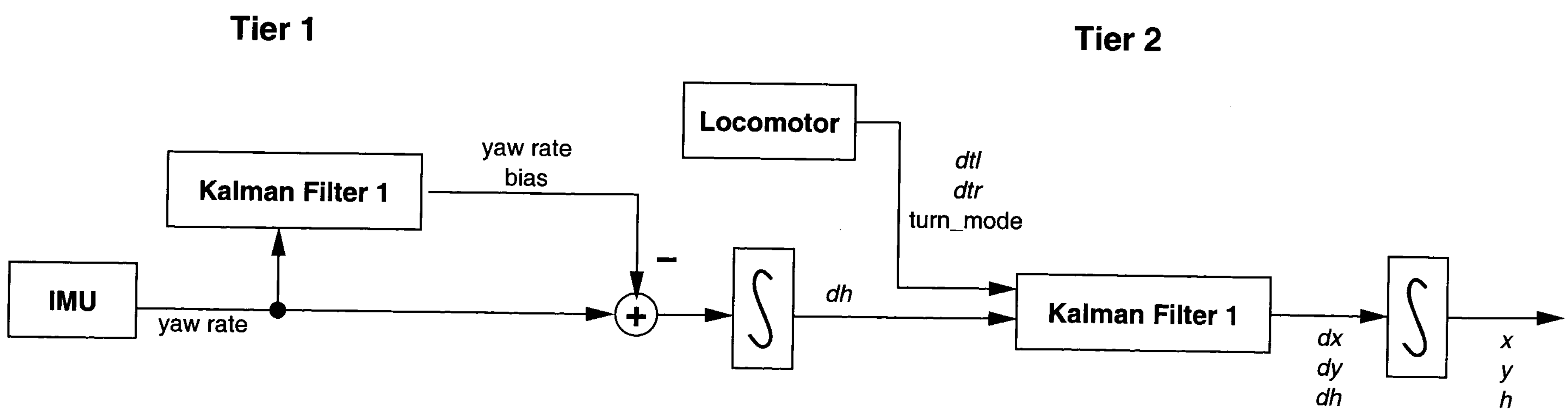
$$P = \begin{bmatrix} p_{11} & \cdots & p_{1n} \\ \vdots & \ddots & \vdots \\ p_{m1} & \cdots & p_{mn} \end{bmatrix}$$

- A.k.a. state uncertainty matrix
- Estimated by Kalman Filter
- Element p_{ii}: variance of state variable i
- Element p_{ij}: correlation between state variables i and j

Design of Current Estimator

Overview of Two Tier Design





KF Tier 1: Estimating Yaw Rate Bias

- Linear Kalman Filter
  - runs when rover is stationary

- System model when bias is stationary: next state (i.e. bias) is same as previous state

$$\frac{d\text{bias}(t)}{dt} = 0 + w_{\text{bias}}(\text{process noise})$$

- Sensor model: if yaw rate is b, expect yaw rate measurement b

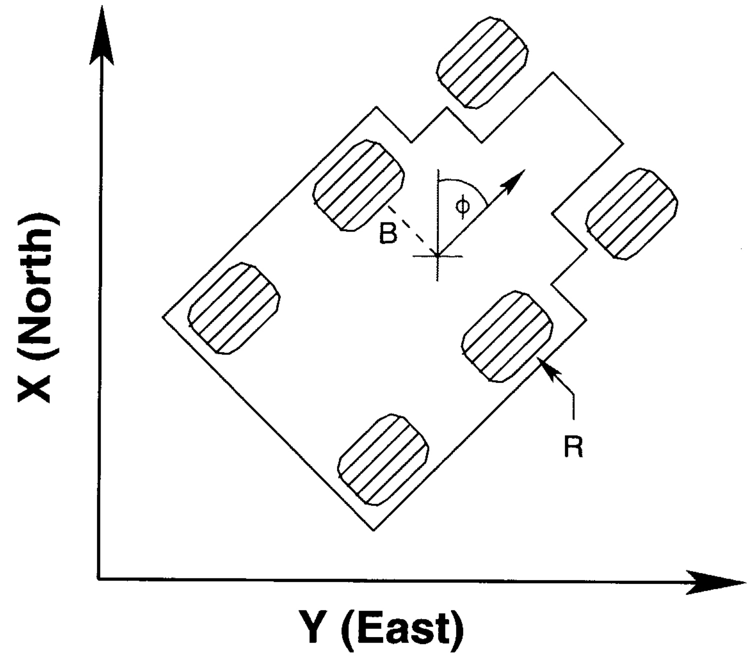
$$z_{\text{bias}} = \text{bias} + u_{\text{bias}}(\text{measurement noise})$$

- **Assumption:** rover moves in short increments so the bias estimate is valid during move

Computing IMU Relative Heading

- While rover is stationary
  - estimate yaw rate bias
- While rover is moving
  - sample yaw rate
  - subtract estimated yaw rate bias
  - heading += yaw rate * sample interval

Vehicle Kinematic Model – Arc or Straight Driving



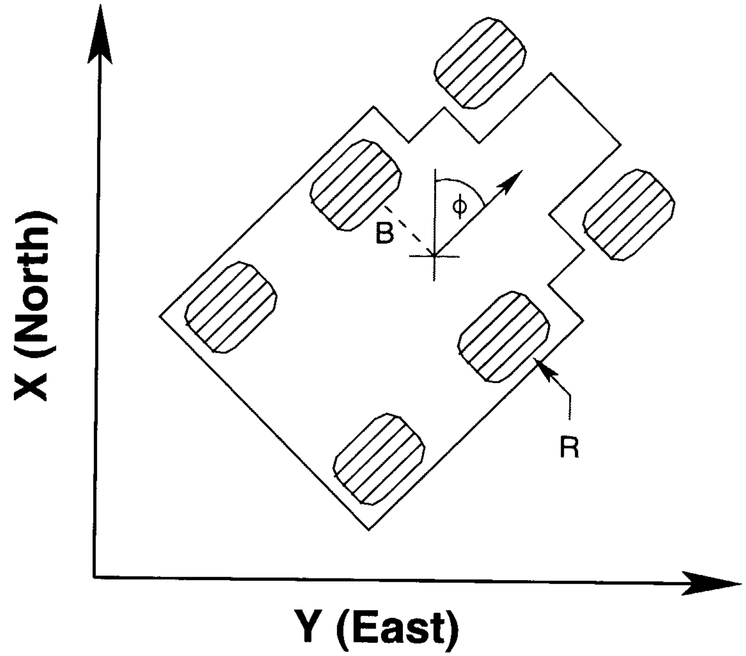
- Define virtual middle wheel with parameters:

$$
\begin{array}{lcll}
\alpha & = & \frac{d\theta_l + d\theta_r}{2} & \text{velocity} \\
u & = & \frac{d\theta_l - d\theta_r}{d\theta_l + d\theta_r} & \text{angle}
\end{array}
$$

- State equations:

$$
\frac{d\mathbf{x}(\alpha)}{d\alpha} = \begin{bmatrix} \frac{dX(\alpha)}{d\alpha} \\ \frac{dY(\alpha)}{d\alpha} \\ \frac{d\phi(\alpha)}{d\alpha} \end{bmatrix} = \begin{bmatrix} R(\cos\phi)\alpha \\ R(\sin\phi)\alpha \\ \frac{R}{B}u\alpha \end{bmatrix}
$$

Vehicle Kinematic Model – Turn In Place



- Define virtual middle wheel with parameters:

$$\alpha \quad = \quad \frac{d\theta_l - d\theta_r}{2} \quad \text{velocity}$$

- State equations:

$$\frac{d\mathbf{x}(\alpha)}{d\alpha} = \begin{bmatrix} \frac{dX(\alpha)}{d\alpha} \\ \frac{dY(\alpha)}{d\alpha} \\ \frac{d\phi(\alpha)}{d\alpha} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \frac{R}{B}\alpha \end{bmatrix}$$

KF Tier 2: Estimating Delta X, Y and Heading

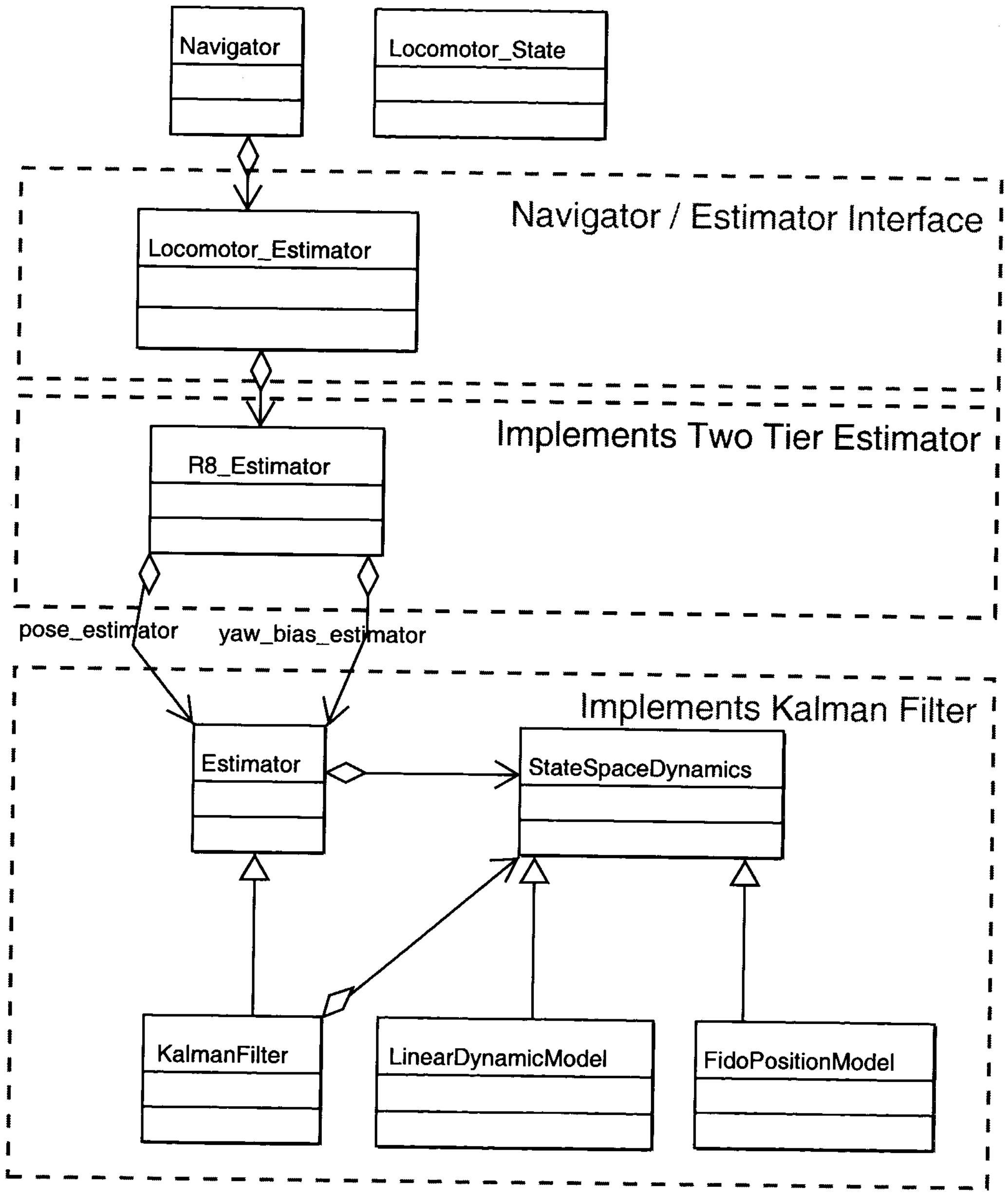
- Extended Kalman Filter
  - runs when rover is moving
- System model: see previous slides
- Sensor model: relates rover pose to IMU relative heading
  - if delta heading is θ, expect IMU relative heading to be θ

Rover Pose Virtual Sensor

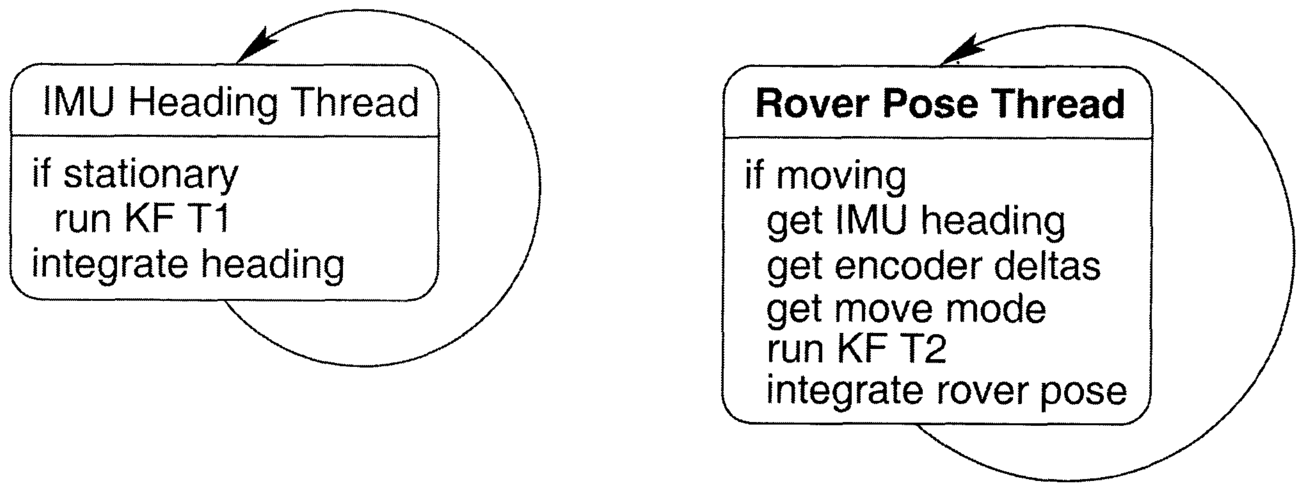
- While rover is moving
  - get IMU relative heading
  - get wheel encoder deltas and move mode from locomotor
  - propagate/update Kalman Filter
  - get delta pose (delta x, delta y, delta heading) from Kalman Filter
  - pose += delta pose * interval

Implementation of Current Estimator

Current Estimator Implementation



R8_Estimator Thread Model



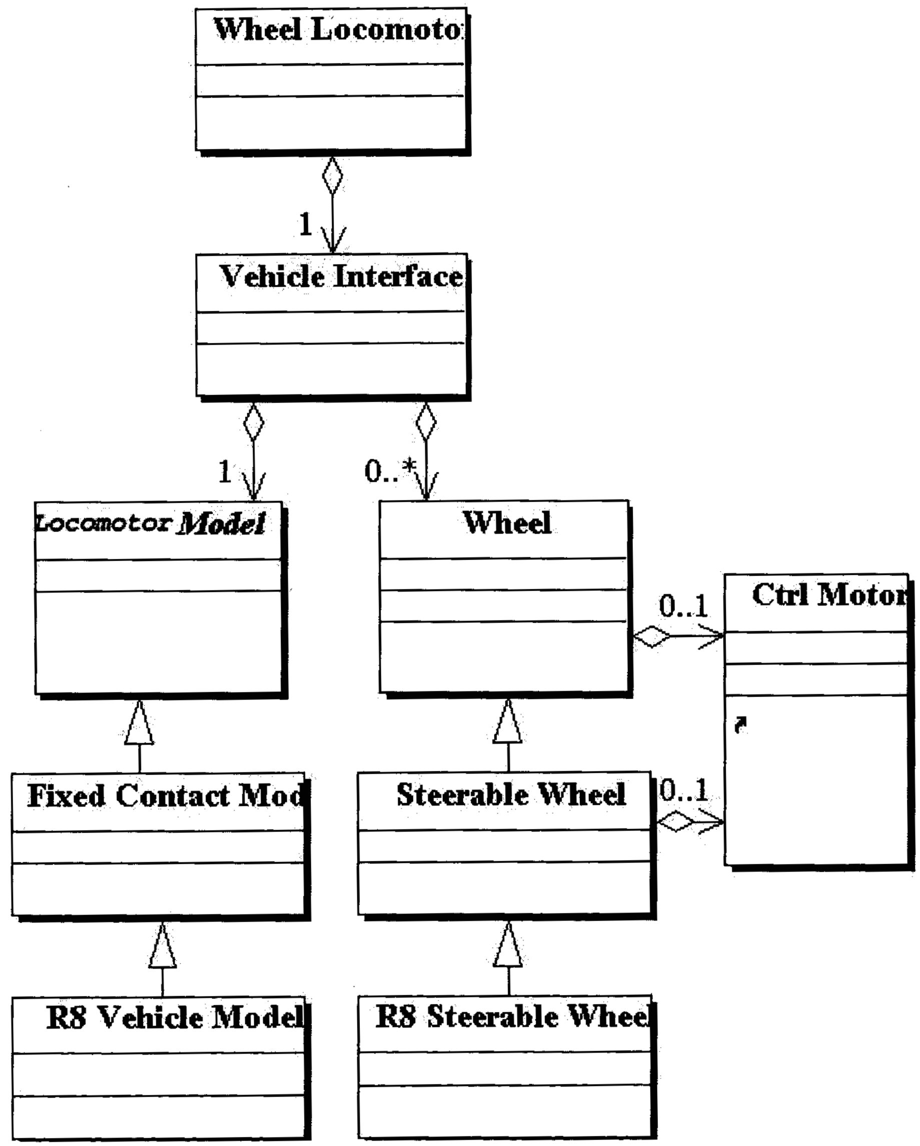
- Each thread runs at its own frequency
  - keeps track of computation time of each iteration
  - sleeps if it has extra time at end of iteration
  - issues warning message if it runs late

Estimator / Locomotor Interface

- Estimator uses the following info from locomotor
  - wheel base
  - wheel radius
  - wheel encoder delta positions
  - move mode (i.e. arc drive, straight line drive, turn in place, . . .)
  - check if locomotor is moving / driving

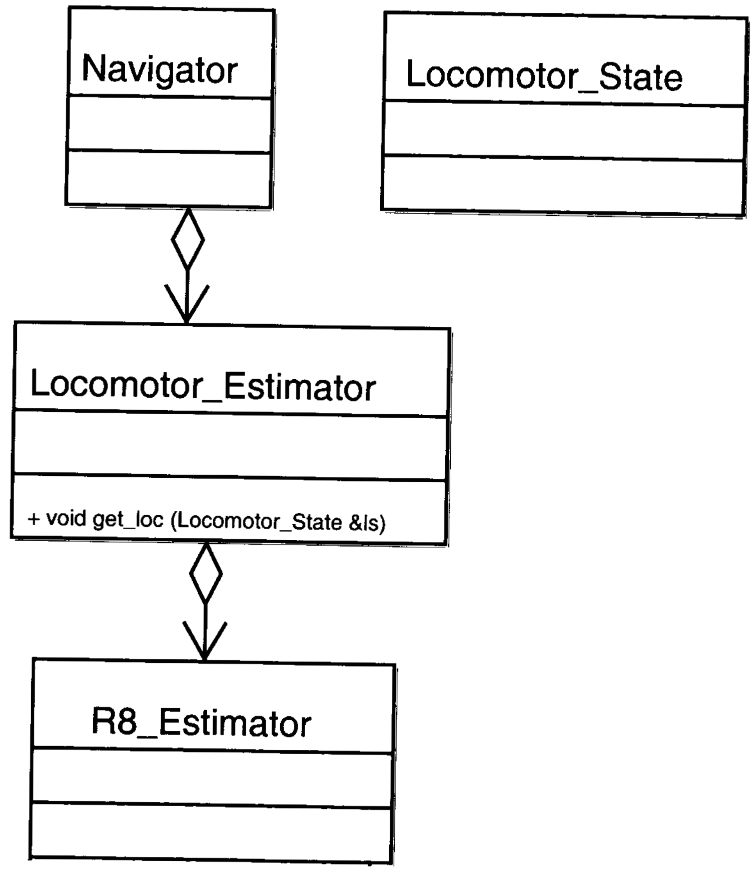
Estimator / CMU Locomotor Interface

- Locomotor Model
  - has wheel_radius
  - need to compute wheel_base
  - Wheel Locomotor has accessor to get Locomotor Model
- Wheel Locomotor
  - need accessor for motor_delta_pos
    * may be part of Wheel?
  - has is_moving() / is_driving()
  - uses DRIVE_COMMAND to talk to Vehicle Interface
  - DRIVE_COMMAND has MOVE_MODE
  - need accessor to MOVE_MODE from Wheel Locomotor



From Chris Urmson's presentation

Estimator / Navigator Interface



Level 1 Milestone Schedule

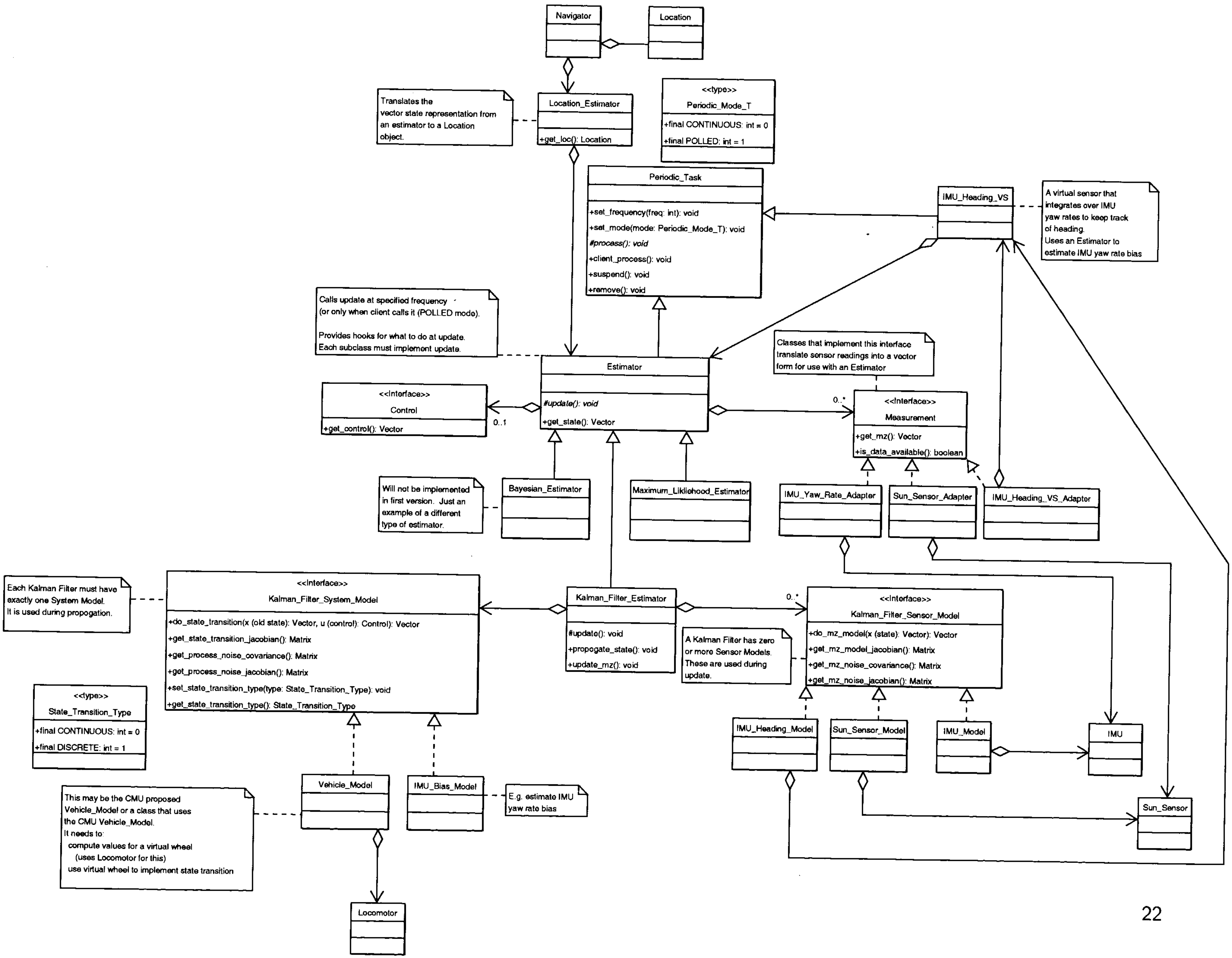
- **Sept 27: test IMU sampling with heavy CPU load**
- Sept 30: test locomotion with pose estimation
- **Oct 2: test pose estimation w/ visionless GESTALT load**
- Oct 4: test locomotion, imaging, stereo processing, pose estimation
- Oct 11: test navigator, locomotion, stereo and pose estimation
- Oct 15: test DL, communications, FL

Overview of Proposed Design (in progress)

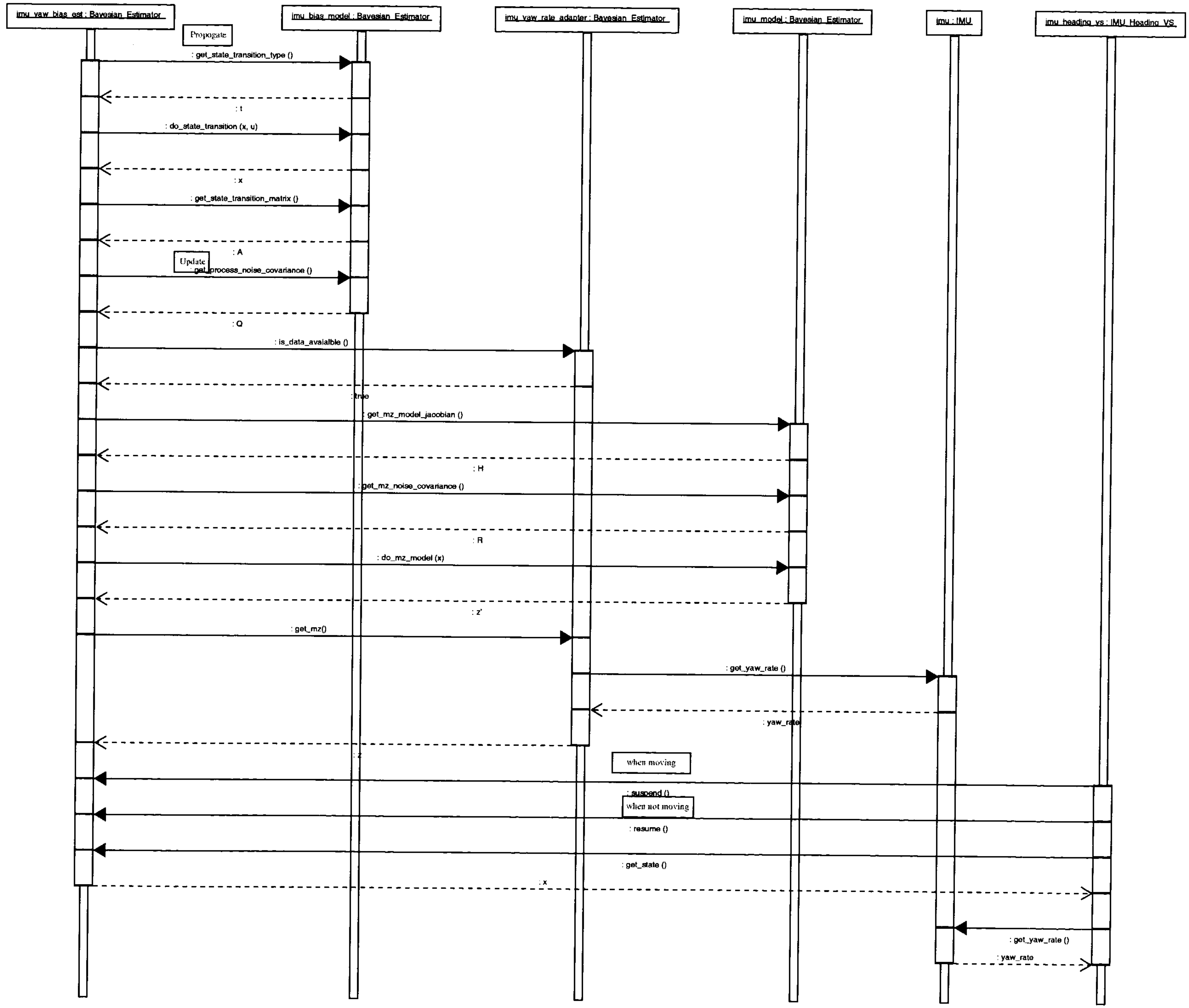
Design Goals

- Make it easier to add new estimators
- Explicitly represent System and Sensor Models and
- Explicitly represent virtual sensors
- Enforce periodic task frequencies
- Clean up inheritence structure of current estimator

Class Diagram



Sequence Diagram 1: IMU Heading



Sequence Diagram 2: Rover Pose