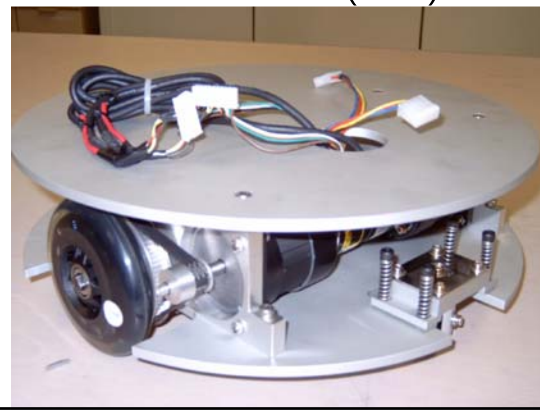
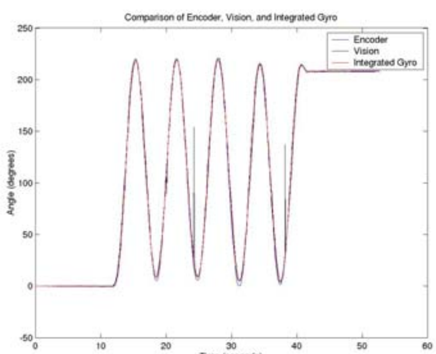
An extended Kalman filter for a mobile robot

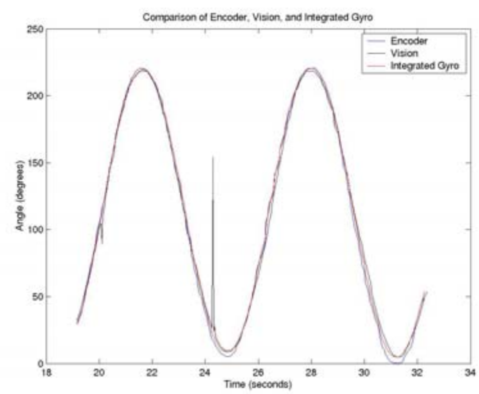
A mobile robot (base)



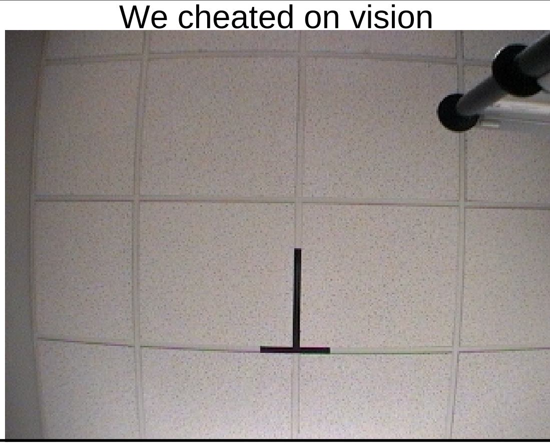
Sensors: Encoder, Gyro, Vision



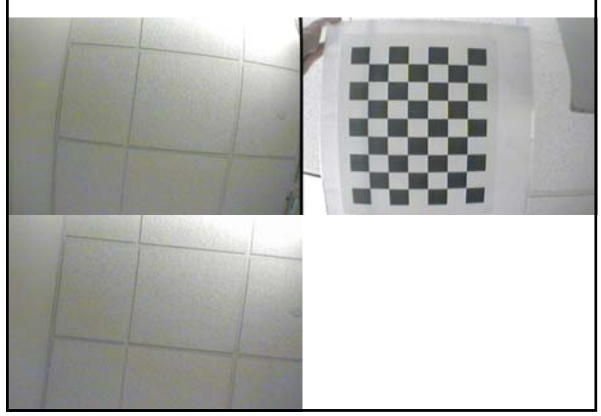

Zooming In On Previous Slide



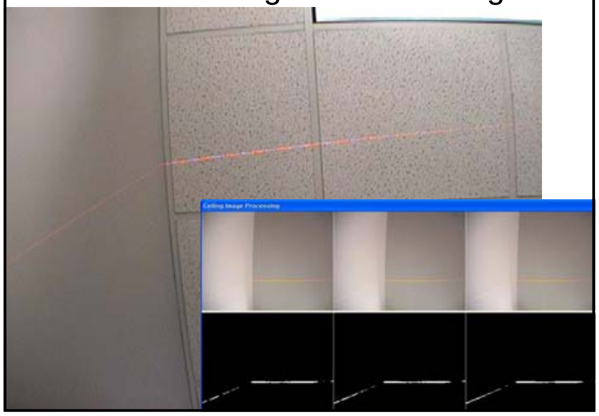

We cheated on vision



Vision calibration

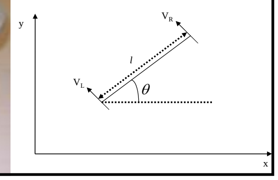


## More Cheating: Structured Light



## A mobile robot (base)

| | |
|---|---|
| $V_R$ | Right wheel velocity |
| $V_L$ | Left wheel velocity |
| $\theta$ | Angle to X axis |
| $l$ | Distance between wheels |




## Kinematics

$$x_{k+1} = x_k - 0.5(v_R + v_L)dt\sin\theta_{k+1}$$

$$y_{k+1} = y_k + 0.5(v_R + v_L)dt\cos\theta_{k+1}$$

$$\theta_{k+1} = \theta_k + \frac{v_R - v_L}{l}dt$$

$$\dot{\theta} = \frac{v_R - v_L}{l},\ v_{tot} = \frac{v_R + v_L}{2}$$

## Extended Kalman Filter (Kinematic)

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_k + \mathbf{B}\mathbf{u} + \mathbf{w}_k \qquad \mathbf{x} = [x, y, \theta, v_R, v_L]^T$$

$$c_{R_k} = e_{R_k} - e_{R_{k-1}} \qquad c_{L_k} = e_{L_k} - e_{L_{k-1}}$$

$$\mathbf{u} = [c_R, c_L]^T$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 & -0.5dt\sin\theta_k & -0.5dt\sin\theta_k \\ 0 & 1 & 0 & 0.5dt\cos\theta_k & 0.5dt\cos\theta_k \\ 0 & 0 & 1 & dt/l & -dt/l \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

## Process Noise and Initial Variance

$$\mathbf{Q} = \mathbf{E}\{\mathbf{w}\mathbf{w}^T\} = \begin{bmatrix} Q_{11} & 0 & \cdots & 0 \\ 0 & Q_{22} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & Q_{mm} \end{bmatrix}$$

$$\mathbf{P}_0 = \begin{bmatrix} \varepsilon & 0 & \cdots & 0 \\ 0 & \varepsilon & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & \varepsilon \end{bmatrix}$$

## Prediction Equations

$$\mathbf{x}_k^- = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}$$

$$\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$$

## Encoder, Gyro update

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \qquad \mathbf{z} = [e_R, e_L, g]^T$$

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1/l & -1/l \end{bmatrix}$$

## Measurement Noise

$$\mathbf{R} = \mathbf{E}\left\{\mathbf{v}\mathbf{v}^T\right\} = \begin{bmatrix} R_{11} & 0 & \cdots & 0 \\ 0 & R_{22} & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & R_{nn} \end{bmatrix}$$

## Measurement Update

$$\mathbf{K} = \mathbf{P}_k^- \mathbf{H}^T \left(\mathbf{H}\mathbf{P}_k^- \mathbf{H}^T + \mathbf{R}\right)^{-1}$$

$$\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-)$$

$$\mathbf{P}_k = \left(\mathbf{I} - \mathbf{K}\mathbf{H}\right)\mathbf{P}_k^-$$

## Vision Update (velocity)

$$\mathbf{z} = \mathbf{g}(\mathbf{x})$$

$$\mathbf{z} = \begin{bmatrix} v_v \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} 0.5(v_R + v_L) \\ (v_R - v_L)/l \end{bmatrix}$$

$$\mathbf{H} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$$

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 0 & 0.5 & 0.5 \\ 0 & 0 & 0 & 1/l & -1/l \end{bmatrix}$$

## Vision Update (landmark at ($x_L$,$y_L$))

$$\mathbf{z} = \mathbf{g}(\mathbf{x})$$

$$\mathbf{z} = \begin{bmatrix} x_v \\ y_v \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_L - x \\ y_L - y \end{bmatrix}$$

$$\mathbf{H} = \frac{\partial \mathbf{g}}{\partial \mathbf{x}}$$

$$\mathbf{H} = \begin{bmatrix} -\cos(\theta) & -\sin(\theta) & x\sin(\theta) - y\cos(\theta) & 0 & 0 \\ \sin(\theta) & -\cos(\theta) & x\cos(\theta) + y\sin(\theta) & 0 & 0 \end{bmatrix}$$

## SLAM

- SLAM (Simultaneous Localization and Mapping) puts landmark locations as part of state to be estimated in the EKF.
- Prediction step is trivial (landmark doesn't move)
- Measurement example below.
- Many landmarks means you have a very large state vector.
- Current research is addressing how to handle this well.

$$\mathbf{x} = [x, y, \theta, x_L, y_L]^T$$

$$\mathbf{H} = \begin{bmatrix} -\cos(\theta) & -\sin(\theta) & -(x_L - x)\sin(\theta) + (y_L - y)\cos(\theta) & \cos(\theta) & \sin(\theta) \\ \sin(\theta) & -\cos(\theta) & -(x_L - x)\cos(\theta) - (y_L - y)\sin(\theta) & -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

## Notes

- "Extended" KF because of angle in A matrix and full state in predicting visual observations
- A, B, H, Q, and R are sparse or diagonal, so should use special purpose coding for efficiency
- Dimensionality of inversion depends on number of sensors $\left(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R}\right)^{-1}$
- Different sampling rates can be handled with a variable length prediction and different Hs
- Need to measure gyro bias when stopped
- Need to handle slipping, vision glitches

## Particle Filtering with EKF Particles

- Each particle is EKF, with weight.
- As particles overlap, merge them and add weights.
- As particles become infeasible, kill them.
- As particles become too certain, confuse them.
- Add new particles in empty spaces according to some prior.

## What If You Took Into Account the Mobile Robot Dynamics?

- Need to change input u to be motor torques.
- This changes prediction step only.
- How do $v_R$ and $v_L$ depend on motor torques?