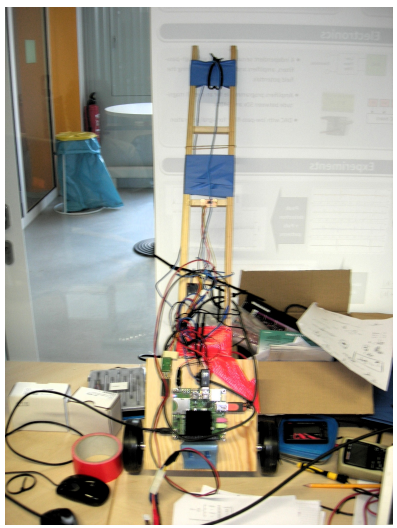
Robotics

Practical: A 2-wheeled *Racer*

Marc Toussaint
U Stuttgart

A 2-wheeled racer



- Educational ideas:
  - have a really **dynamic** system
  - have a system which, in the “racing” limit, is hard to control
  - learn about hardware, communication, etc
  - challenges connecting theory with practise:
- Real world issues:
  - control interface (“setting velocities”) is adventurous
  - PARTIAL OBSERVABILITY: we only have a noisy accelerometer & gyroscope
  - unknown time delays
  - unknown system parameters (masses, geometry, etc)

Intro

[demo]

Components

- Odroid: on-board PC running xubuntu
- Motor unit: motors, motor driver, motor controller, Hall sensor
- IMU (inertial measurement unit): 3D accelerometer, 3D gyroscope, (magnetic)
- Communication: USB-to-I2C communicates with both, motors and IMU

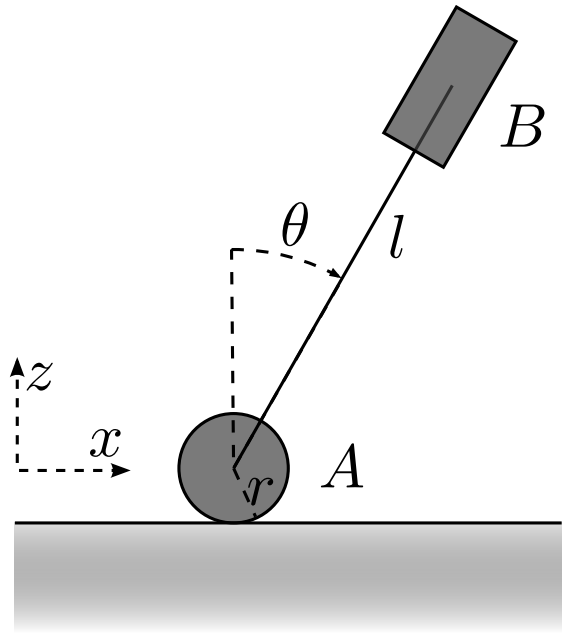
- See Marcel's thesis

Code

- From Marcel's thesis:
  - Control loop (around 36 msec)

- Kalman filter tests on the accelerometer:
  - Caroline

2D Modelling



- See theoretical modelling notes

3D Modelling

- Account for centrifugal forces in a curve
- Generalized coordinates $q = (x, y, \phi, \theta)$, with steering angle ϕ

- Exercise: Derive general Euler-Lagrange equations

Clash of theory and real world

The control interface

- Theory assumed torque control
- In real, the motor controller “does things somehow”. We can set:
  - a target velocities $v^*_{l,r}$
  - desired acceleration level $a^*_{l,r} \in \{-10, .., -1, 1, .., 10\}$
- The controller will then ramp velocity in 25msec steps depending on a^* until target v^* is reached
- Unknown: time delays, scaling of a^*?

- Potential approach:
  - Assume acceleration control interface
  - Consider constrained Euler-Lagrange equations

Coping with the partial observability

- Theoretical view: In LQG systems it is known that optimal control under partial observability is the same as optimal control assuming the Bayes estimated state as true current state. *Uncertainty principle.*
- Use a Bayes filter to estimate the state $(q, \dot{q})$ from all sensor information we have
- Sensor information:
  - Accelerometer readings $\tilde{a}_{x,y,z}$
  - Gyro readings $\tilde{g}_{x,y,z}$
  - Motor positions $\tilde{\theta}_{l,r}$. Note that $\tilde{\theta} \propto x/r - \theta$ desribes the relative angle between the pole and the wheels
- Open issue: time delays – relevant?

Coping with unknown system parameters

- **System identification**
- We derived the eqs of motion $Bu = M\ddot{q} + F$ (for 2D) – but don't know the parameters
  - m_A, I_A, m_B, I_B: masses and inertias of bodies A (=wheel) and B (=pendulum)
  - r: radius of the wheel
  - l: length of the pendulum (height of its COM)

- Focus on the local linearization around $(q, \dot{q}) = 0$
- OR: Use blackbox optimization to fit parameters to data

Data

- We need data to understand better what's going on!
- Lot's of data of full control cycles around $(q, \dot{q}) = 0$
  (sensor reading, control signals, cycle time)
- Data specifically on how motors accelerate when setting a desired acceleration level

Or completely different: Reinforcement Learning

- Alternatively one fully avoids modelling $\rightarrow$ Reinforcement Learning
- Roughly: blackbox optimization (e.g., EA) of PD parameters

Modelling

Modelling overview I

We have exact analytical models (and implemented) for the following:

- Euler-Lagange equations

$$M(q)\,\ddot{q} + F(q,\dot{q}) = B(q)\,u$$
$$\ddot{q} = M^{-1}(Bu - F)$$

  → **energy check**
  → **physical simulation**

- Local linearization $(x = (q, \dot{q}))$

$$\ddot{q} = Ax + a + \bar{B}u$$
$$A = \frac{\partial}{\partial x} M^{-1}(Bu - F)\ , \quad \bar{B} = M^{-1}B$$

  → **gradient check**
  → **Riccati eqn** → nice controller [demo]

Modelling overview II

- Sensor model

$$y^{\text{acc}} = c_1 \; R \left[\ddot{p}_B - (0, g)^\top \right], \quad R = \begin{pmatrix} \cos(\theta + c_2) & -\sin(\theta + c_2) \\ \sin(\theta + c_2) & \cos(\theta + c_2) \end{pmatrix}$$

$$y^{\text{gyro}} = c_3(\dot{\theta} + c_4)$$

$$y^{\text{enc}} = c_5(x/r - \theta)$$

$$y = (y^{\text{acc}}, y^{\text{gyro}}, y^{\text{enc}}) \in \mathbb{R}^4$$

- Local linearization

$$C = \frac{\partial y}{\partial (q, \dot{q})} = \begin{pmatrix} \frac{\partial y}{\partial q} & \frac{\partial y}{\partial \dot{q}} \end{pmatrix} + \frac{\partial y}{\partial \ddot{q}} \; \frac{\partial \ddot{q}}{\partial (q, \dot{q})}$$

$\rightarrow$ **gradient check**

$\rightarrow$ **Kalman filtering** [demo]

Modelling overview III

- Constrained Euler-Lagange equations for acceleration control
  - Our motors actually don't allow to set torques – but rather set accelerations. Setting accelerations implies the constraint

$$B' \ddot{q} = u'$$

  - Using $\ddot{q} = M^{-1}(Bu - F)$ we can retrieve the torque

$$u = (B' M^{-1} B)^{-1} [u' + B' M^{-1} F]$$

    that exactly generates this acceleration
  - Plugging this back into $\ddot{q} = M^{-1}(Bu - F)$ we get

$$\ddot{q} = B'^{\#} u' - (\mathbf{I} - B'^{\#} B') M^{-1} F \;, \quad B'^{\#} = M^{-1} B (B' M^{-1} B)^{-1}$$

Modelling summary

- We now have all analytic models we need
- **In simulation** we have no problem to apply
  - Riccati to retrieve a (locally) optimal linear regulator
  - Kalman to optimally (subject to linearizations) estimate the state

- The crux: we have 12 unknown parameters

$$m_A, I_A, m_B, I_B, \quad r, l, l_C, \quad c_1, .., c_5$$

(plus sensor noise parameters $\sigma_a, \sigma_g, \sigma_e$)

System Identification

System Identification

- Given data $D = \{(x, u, y)_t\}_{t=1}^{T}$, learn

$$(x, u) \mapsto x' \qquad \text{or} \qquad P(x'|x, u)$$
$$(x, u) \mapsto y \qquad \text{or} \qquad P(y|x, u)$$

Regression options for system identification

- Linear: (linear in finite number of parameters)

$$f(x;\theta) = \phi(x)^\top \theta$$

- Blackbox parameteric:
  - Given some blackbox parameteric model $f(x;\theta)$ with finite parameters θ; use blackbox optimization
- Non-parameteric:
  - Kernel methods
  - Gaussian processes
  - Are closely related to linear models

- In all cases one typically minimizes the squared error

$$L^{\text{ls}}(\theta) = \textstyle\sum_{i=1}^{n} (y_i - f(x_i;\theta))^2$$

- We can use the mean $\frac{1}{n} L^{\text{ls}}(\theta)$ as estimate of the output variance σ^2 to define

$$P(y|x;\theta) = \mathcal{N}(y|f(x;\theta), \sigma^2)$$

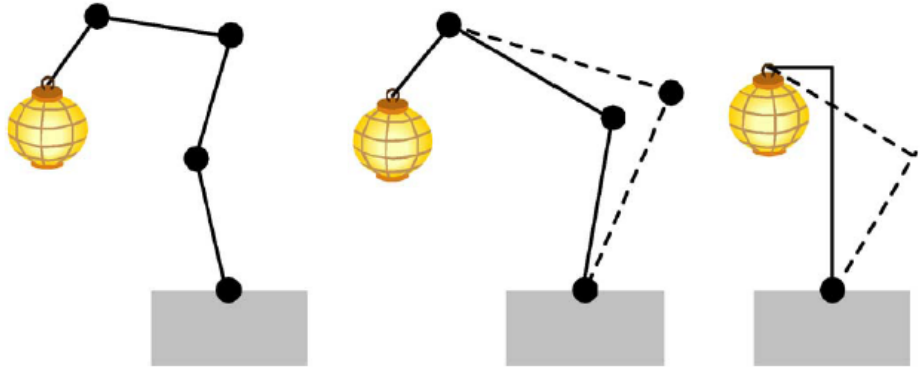
System Id examples: Kinematics

- If the kinematics ϕ are unknown, learn them from data!

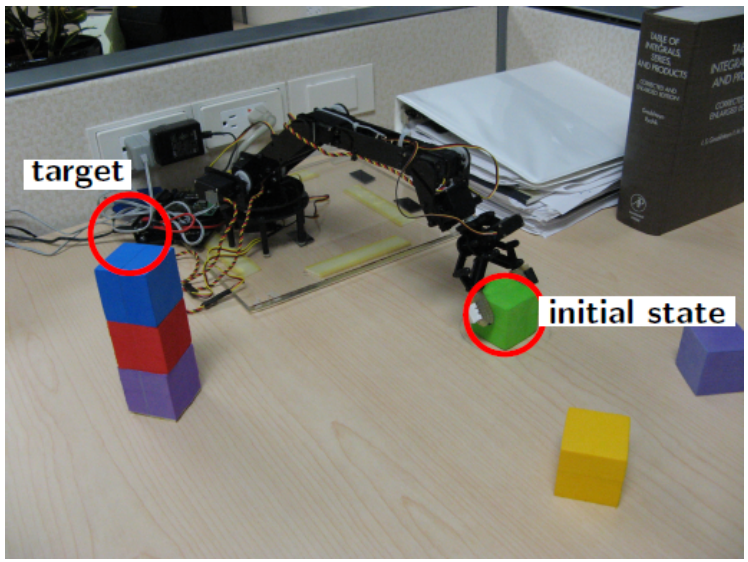
  Literature:

  Todorov: Probabilistic inference of multi-joint movements, skeletal parameters and marker attachments from diverse sensor data. (IEEE Transactions on Biomedical Engineering 2007)

  Deisenroth, Rasmussen & Fox: Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning (RSS 2011)



Todorov: Probabilistic inference of multi-joint movements, skeletal parameters and marker attachments from diverse sensor data. (IEEE Transactions on Biomedical Engineering 2007)



Deisenroth, Rasmussen & Fox: Learning to Control a Low-Cost Manipulator using Data-Efficient Reinforcement Learning (RSS 2011)

System Id examples: Dynamics

- If the dynamics $\dot{x} = f(x, u)$ are unknown, learn them from data!

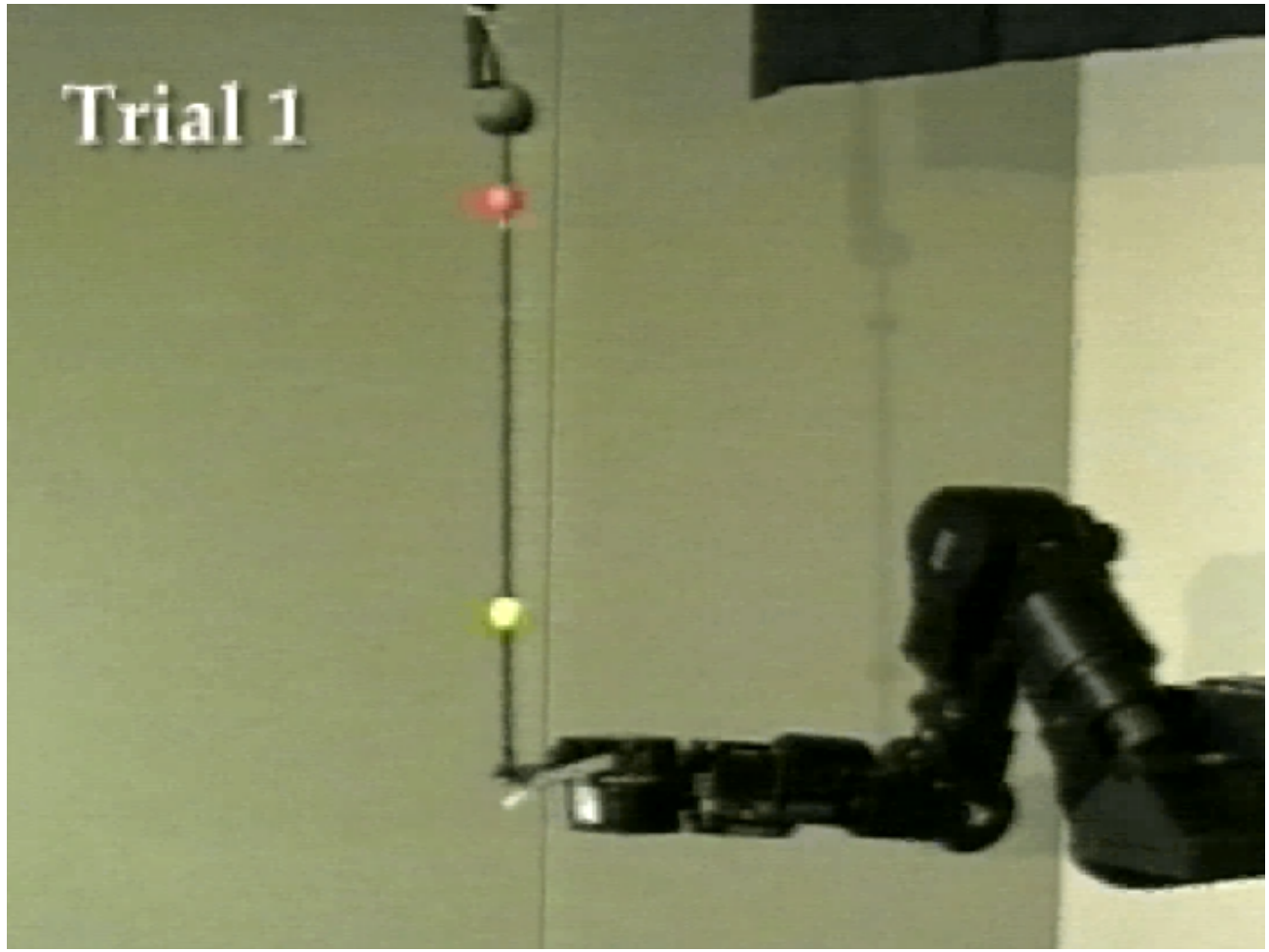
  Literature:

  Moore: Acquisition of Dynamic Control Knowledge for a Robotic Manipulator (ICML 1990)

  Atkeson, Moore & Schaal: Locally weighted learning for control. Artificial Intelligence Review, 1997.

  Schaal, Atkeson & Vijayakumar: Real-Time Robot Learning with Locally Weighted Statistical Learning. (ICRA 2000)

  Vijayakumar et al: Statistical learning for humanoid robots, Autonomous Robots, 2002.



(Schaal, Atkeson, Vijayakumar)

- Use a simple regression method (locally weighted Linear Regression) to estimate $\dot{x} = f(x, u)$

Regression basics

[ML slides]

Applying System Id to the racer?

- Core problem:

  We have no ground truth data!

- We can record data (u, y) (controls & observations), but not x!

- Try an EM like approach:
  - Hand-estimate the parameters as good as possible
  - Use a Kalman filter (better: smoother!!) to estimate the unobserved x during
  - Option (a): Learn local linear models $\ddot{q} = Ax + a + Bu$ and $y = Cx + c + Du$
    Option (b): Improve the parameters $\theta = (m_A, I_A, m_B, I_B, r, l, l_C, c_1, .., c_5)$
  - Repeat with Kalman smoothing

- I have no idea whether/how well this'll work

Data

We've collected data

- Motor responses
  - Free running wheels (no load..)
  - Setting extreme target velocities v^* and different acceleration levels $a^* \in \{-10, .., -1, 1, .., 10\}$ we can generate well-defined accelerations

- Balancing trials
  - the gyroscope picks up some oscillations
  - the accelerometer is very noisy, perhaps correlated with jerky controls
  - only 30Hz!