

COMP417 Intro to Robotics

Final Lecture: Levels of Autonomy and Overview

Fall 2019
David Meger

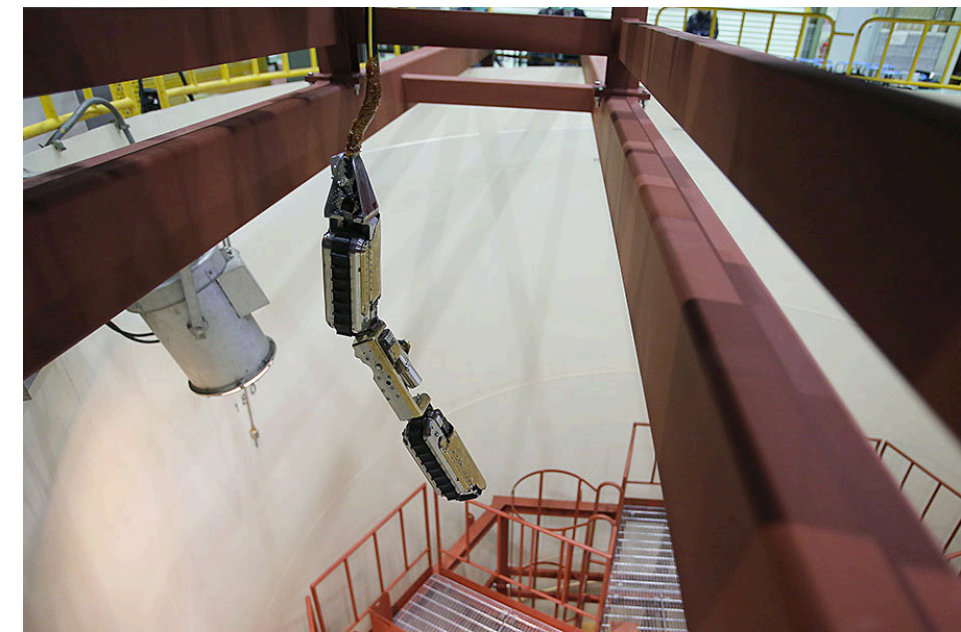
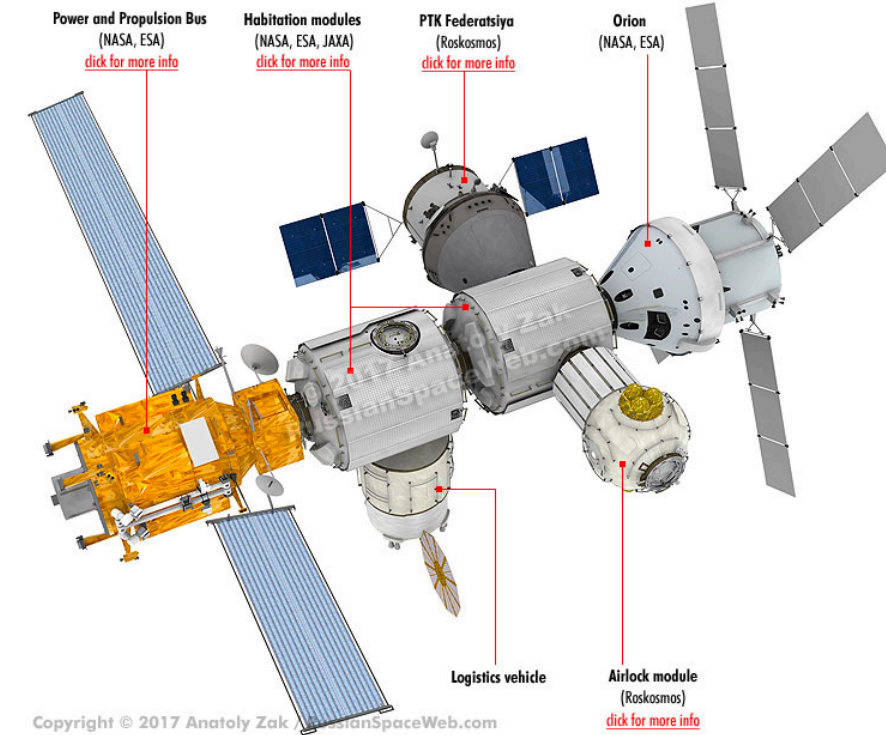


McGill

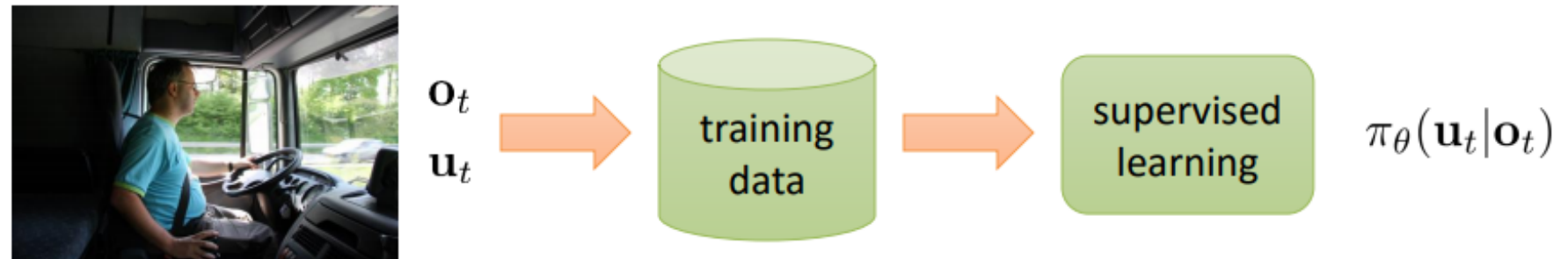
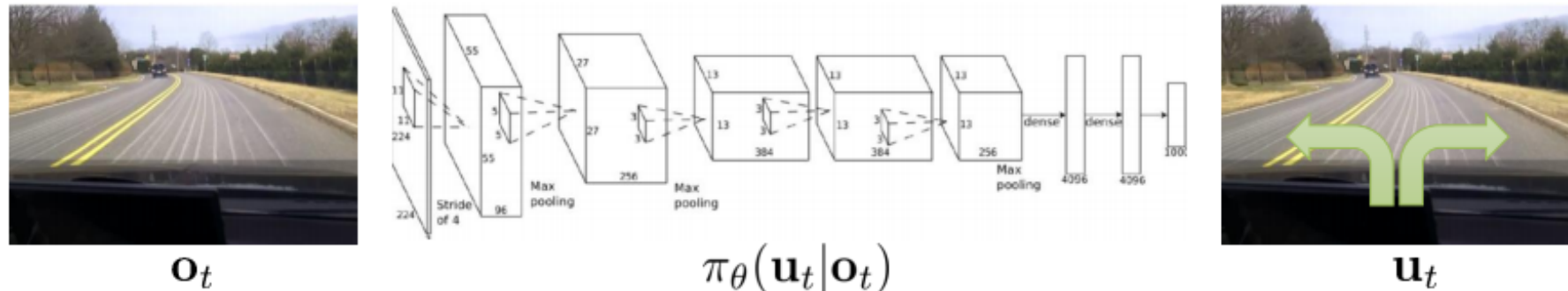
MRL Mobile Robotics Lab
at **McGill University**

How do autonomous robots affect our lives?

- They are our only feasible way to accomplish some tasks today:
 - Space
 - Damaged nuclear sites
- We envision they'll take over on problems we struggle with today:
 - "OK boomer" robot
 - Taking concrete actions to help the climate
- For many tasks right now, we often actually want a collaborative robot (cobot):
 - "Fits in" with human situations
 - Easy to program by those weaklings who have not completed 417...



Last research highlight: imitation learning

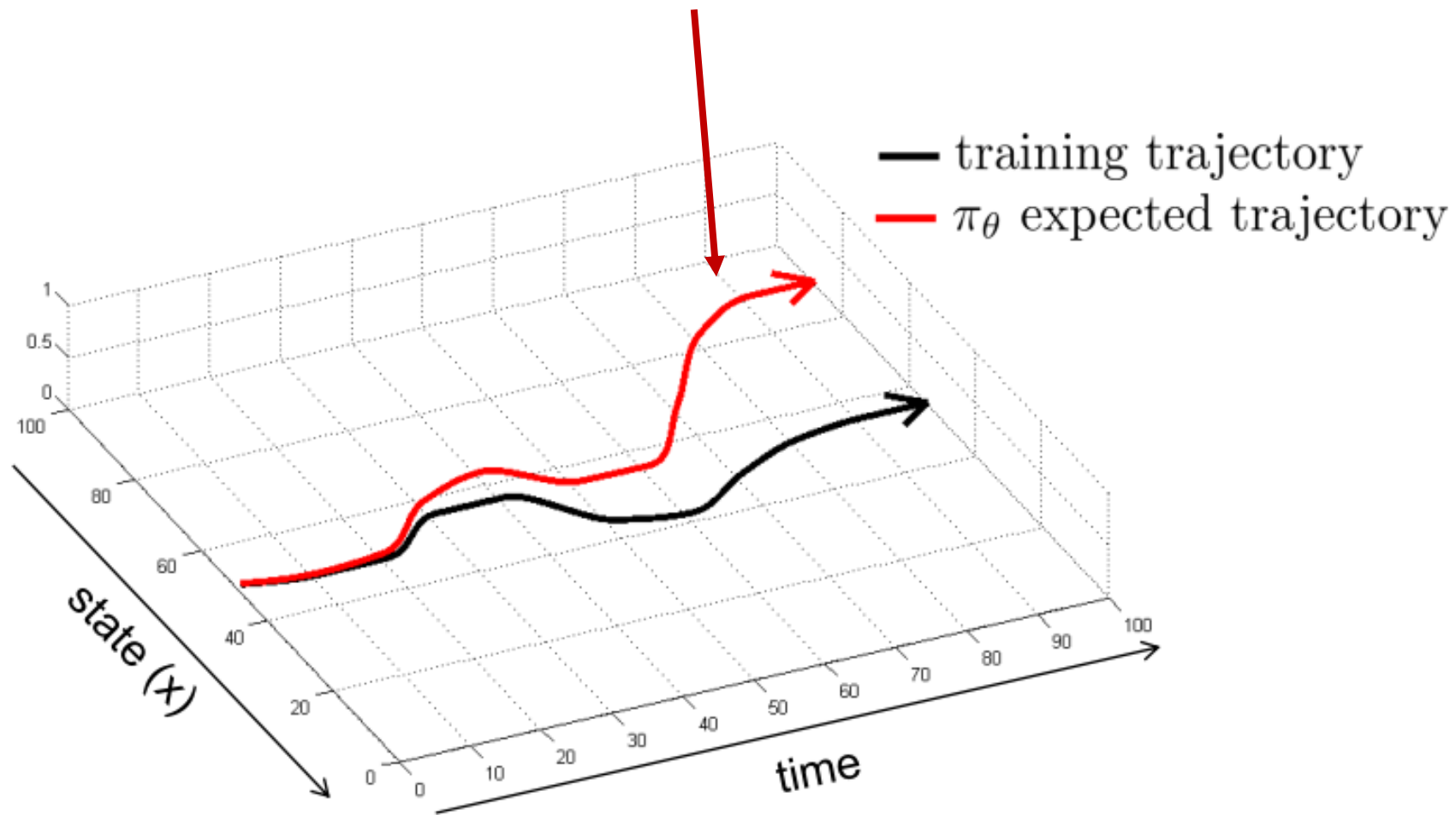


Behavior Cloning

- Given demonstrations with full state-action (x,a) pairs:
 - Treat x as input and a as learning target
 - This forms a ***supervised learning*** problem.
 - Train the network with a loss that encourages “imitating” the action that the user made for each state
- Note: implicit assumption of a perfect user
- At test time, pass the system state to the network as input. Apply the action generated. Hope to “clone” the expert.

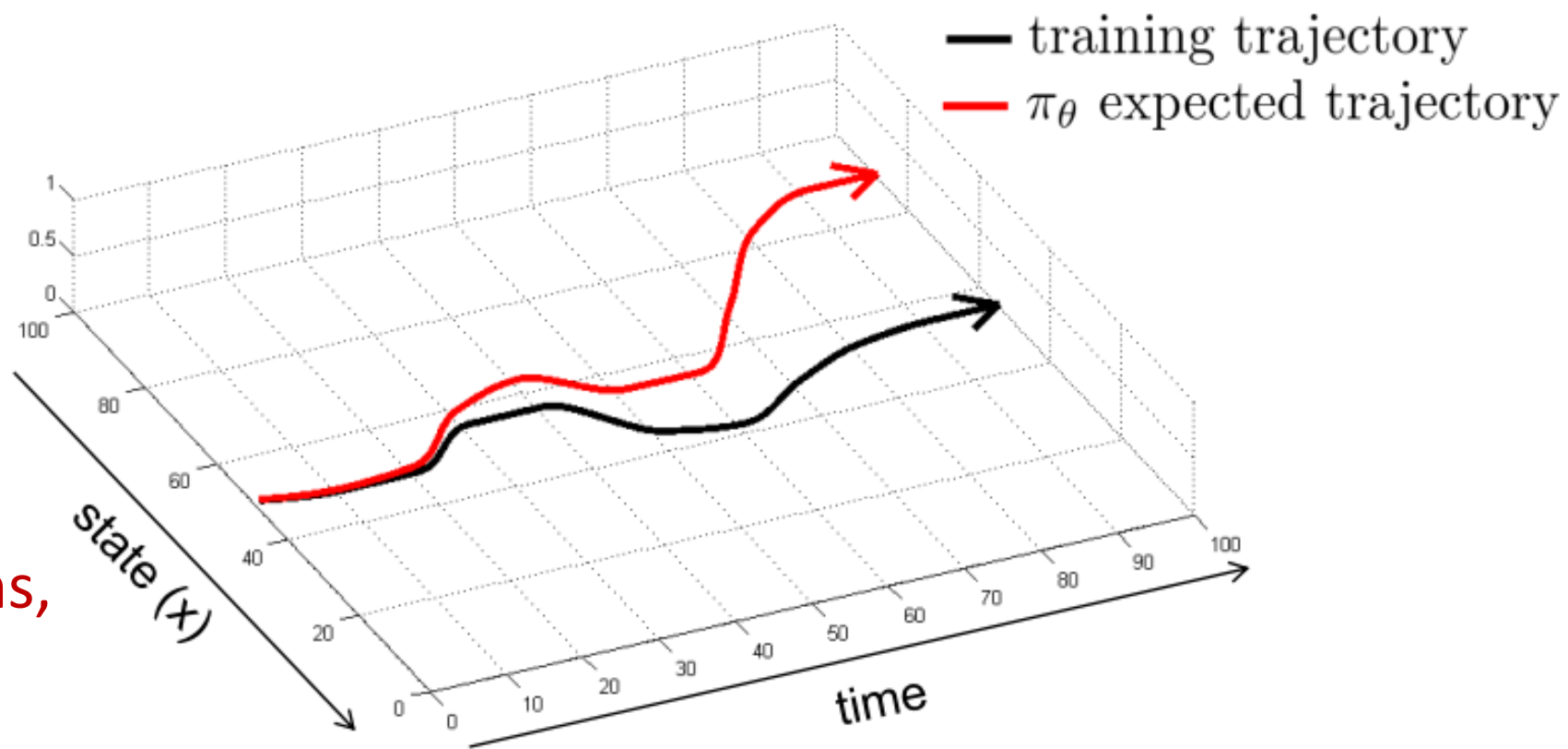
Does it work?

“Work” means that the learned policy matches expert performance



Does it work?

No!



Once the agent moves away from demonstrations, performance can be arbitrarily bad!

Subtleties

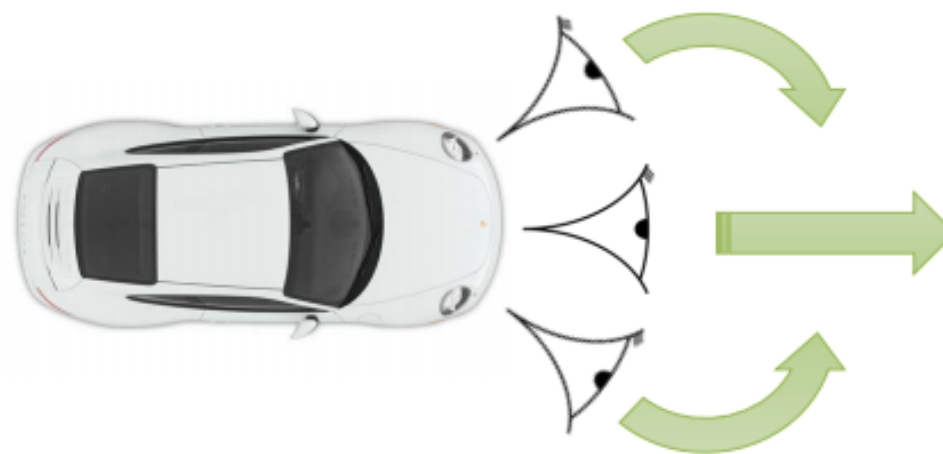
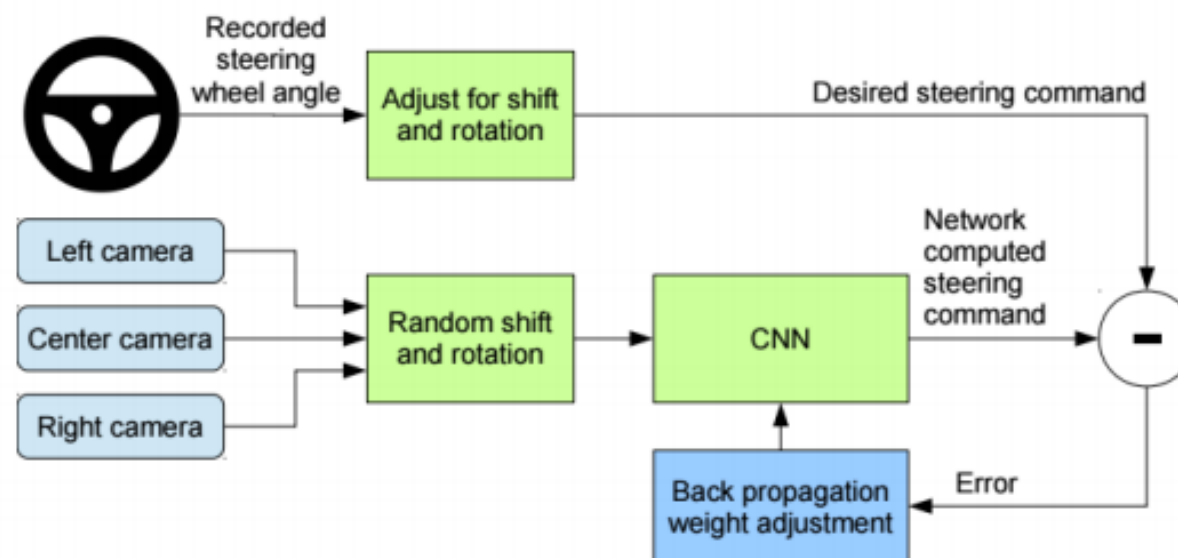
- Assumptions of supervised machine learning are violated
- Data required vs ease of providing that data
- Intelligent system's abilities may not perfectly match demonstrator
- Cascading errors over time

Does it work?

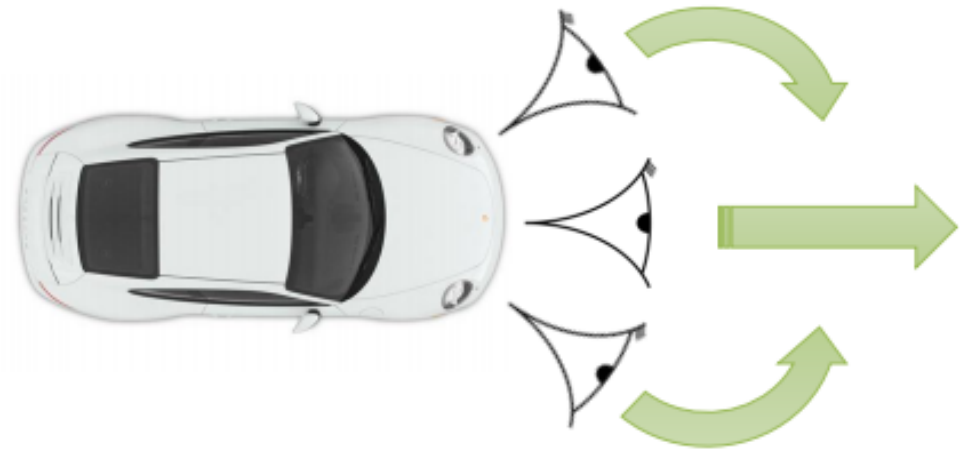
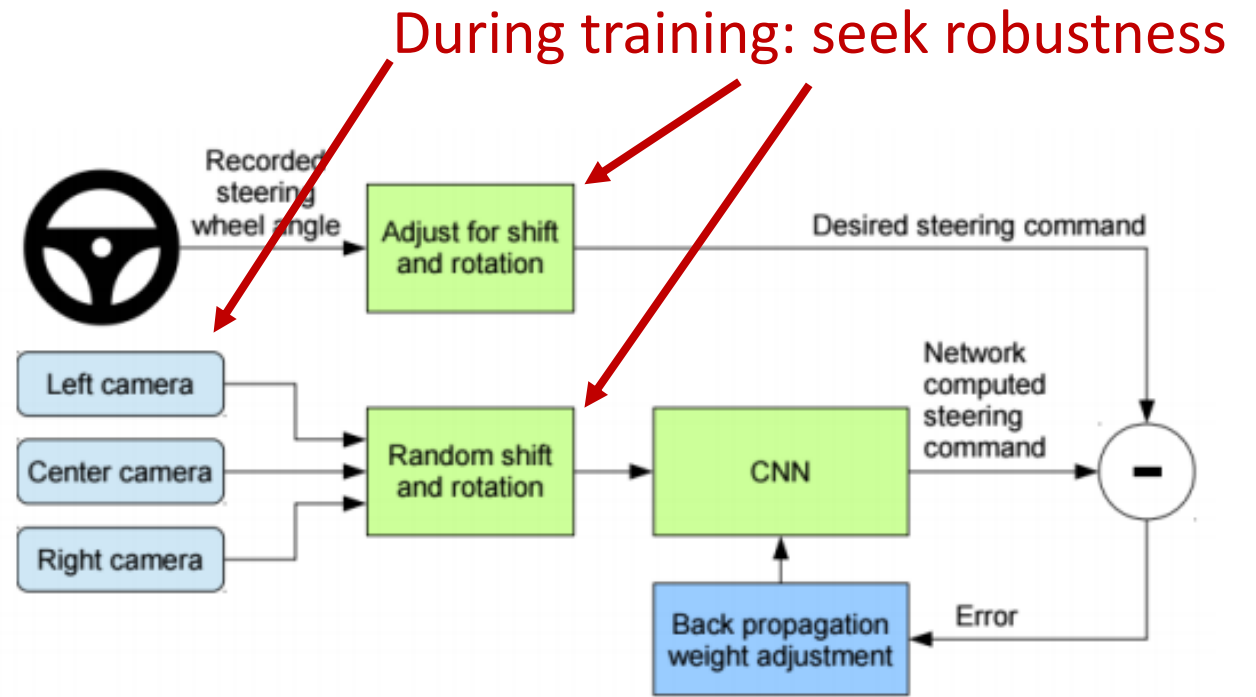
YES!



Why did that work?

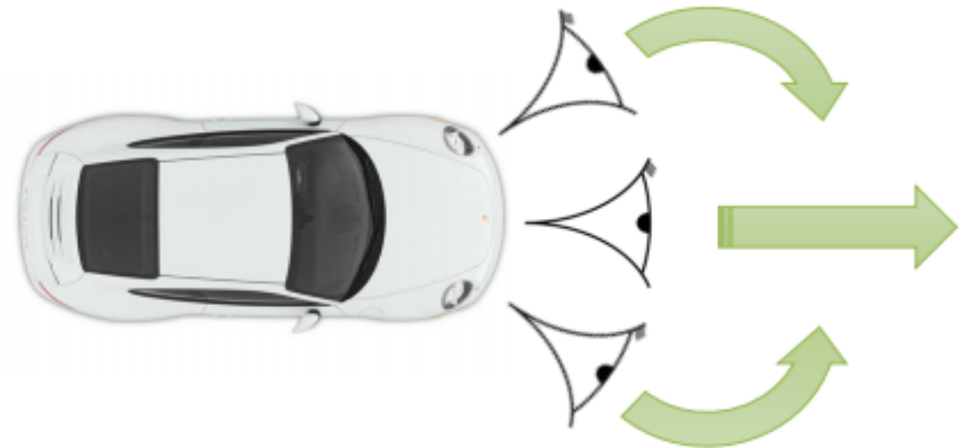
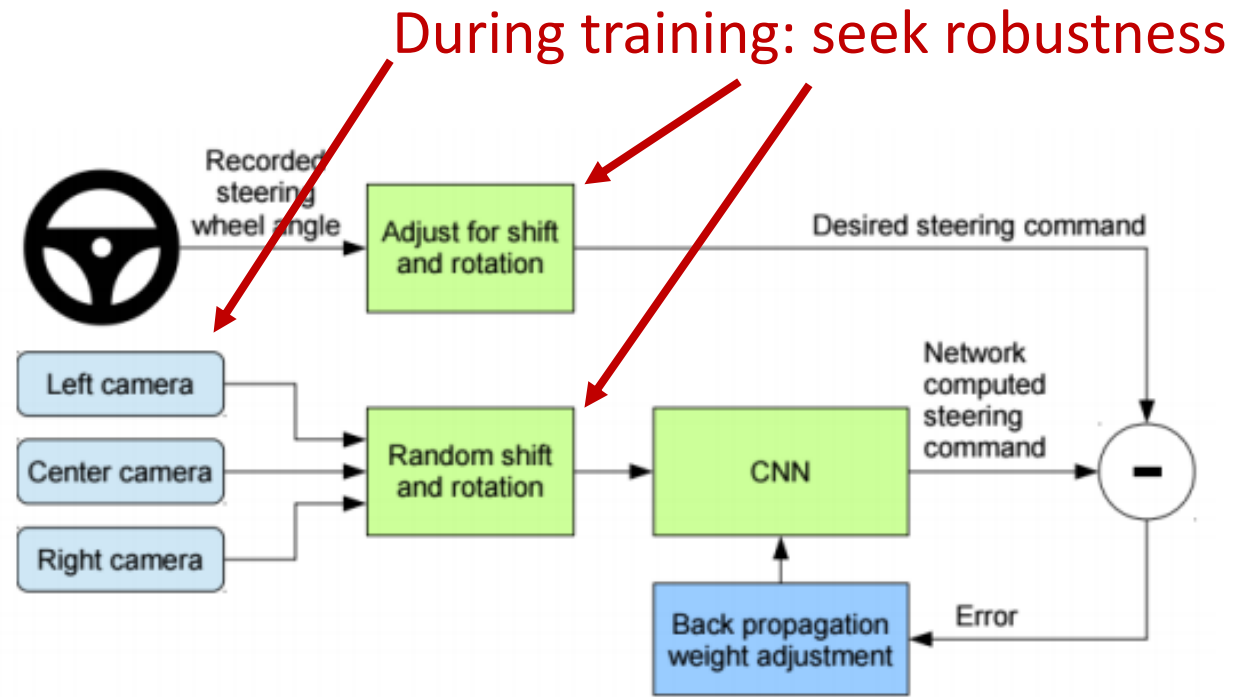
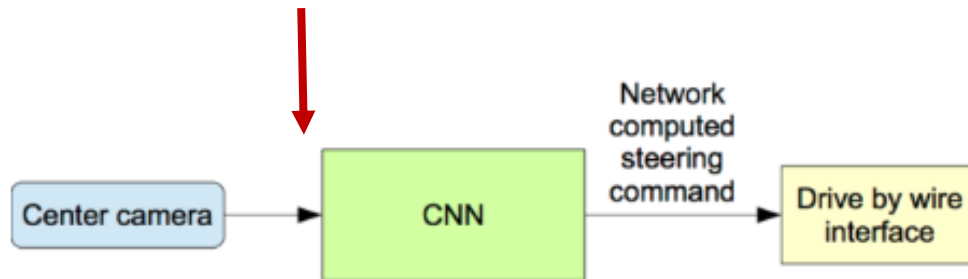


Why did that work?

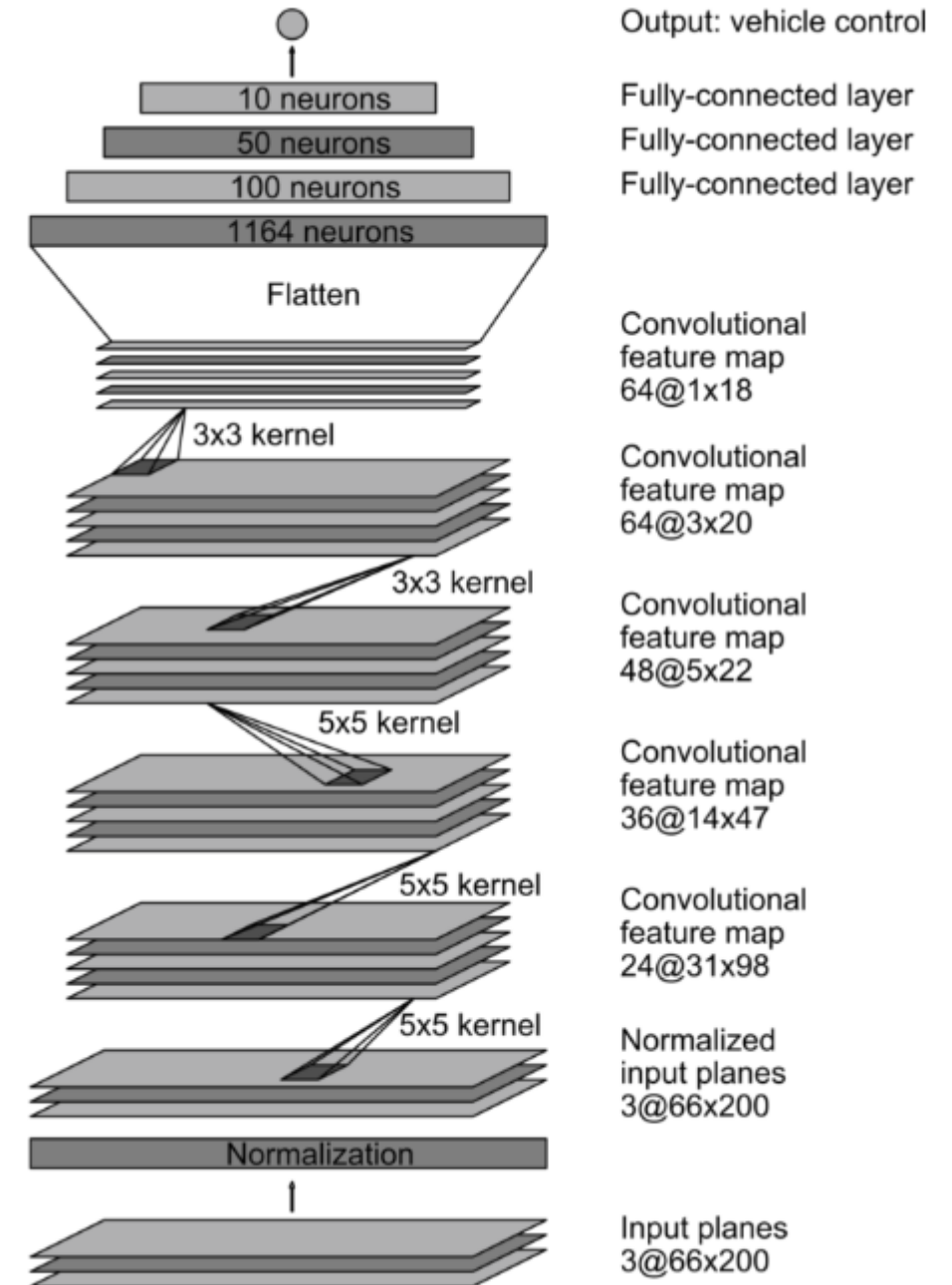


Why did that work?

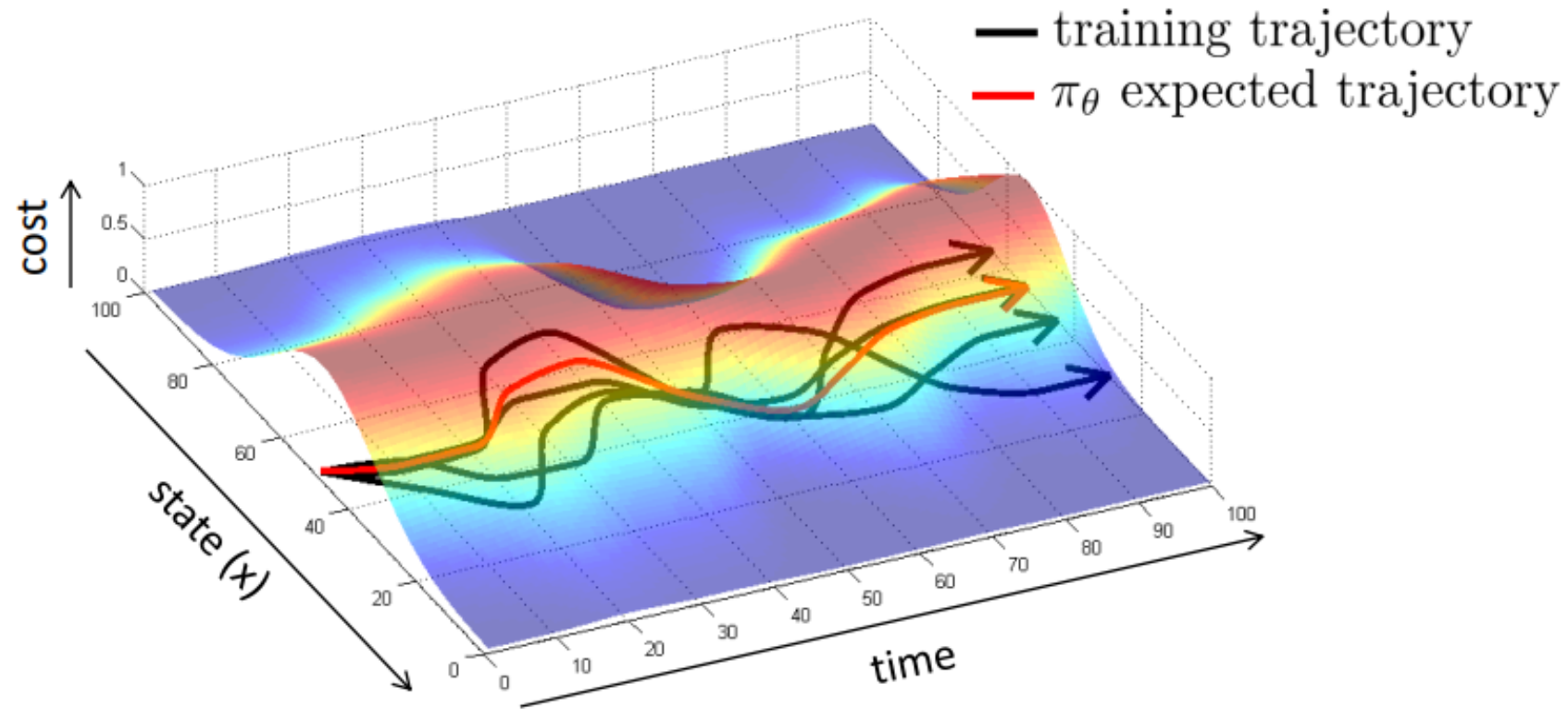
During test: choose the most robust action



Architecture: Driving from images



Can we make IL work more often?



stability

Can we make it work more often?

can we make $p_{\text{data}}(\mathbf{o}_t) = p_{\pi_\theta}(\mathbf{o}_t)$?


idea: instead of being clever about $p_{\pi_\theta}(\mathbf{o}_t)$, be clever about $p_{\text{data}}(\mathbf{o}_t)$!

DAgger: **D**ataset **A**ggregation

goal: collect training data from $p_{\pi_\theta}(\mathbf{o}_t)$ instead of $p_{\text{data}}(\mathbf{o}_t)$

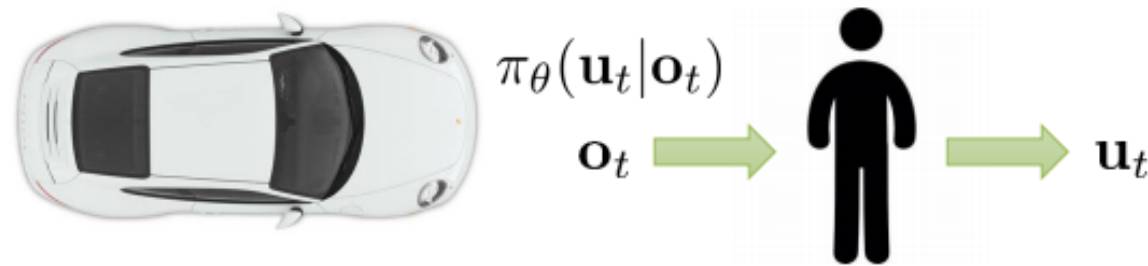
how? just run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$

but need labels \mathbf{u}_t !

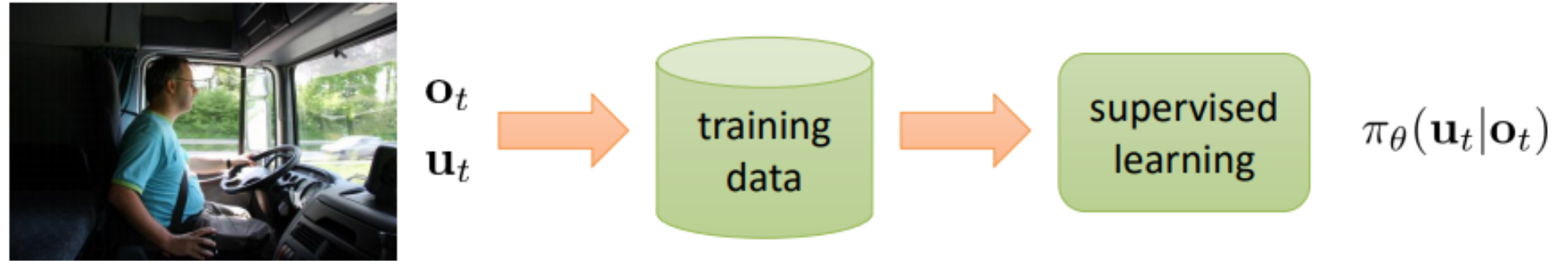
- 
1. train $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
 2. run $\pi_\theta(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
 3. Ask human to label \mathcal{D}_π with actions \mathbf{u}_t
 4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_\pi$

What's the problem?

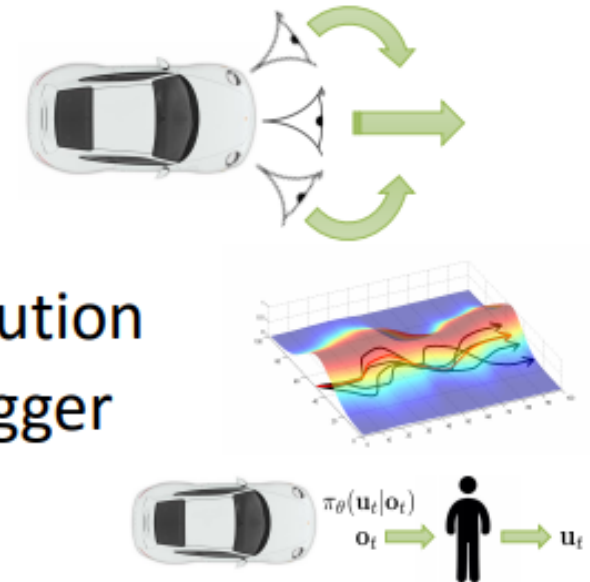
1. train $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ from human data $\mathcal{D} = \{\mathbf{o}_1, \mathbf{u}_1, \dots, \mathbf{o}_N, \mathbf{u}_N\}$
2. run $\pi_{\theta}(\mathbf{u}_t|\mathbf{o}_t)$ to get dataset $\mathcal{D}_{\pi} = \{\mathbf{o}_1, \dots, \mathbf{o}_M\}$
3. Ask human to label \mathcal{D}_{π} with actions \mathbf{u}_t
4. Aggregate: $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_{\pi}$



Recap



- Often (but not always) insufficient by itself
 - Distribution mismatch problem
- Sometimes works well
 - Hacks (e.g. left/right images)
 - Samples from a stable trajectory distribution
 - Add more **on-policy** data, e.g. using DAgger

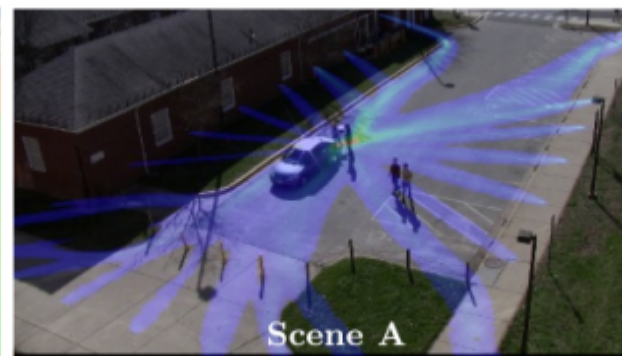


Inverse Optimal Control (overlap with IRL)

- Rather than learning $a=f(x)$ directly, instead infer the user's ***cost function***, $r(x)$
- Now, we can use the tools from optimal control, RL etc in order to come up with the policy that fits. Notable differences in learning:
 - The ability to think deliberatively (plan for the future)
 - Optimization over time rather than assuming independence
- What about robustness to missing data? What $r(x)$ do we expect for unseen x values?

Learning to infer intentions

- Suppose we have visual inputs and human trajectories. Learning $r(x)$ can capture functional dependencies:
 - Terrain type: drive on the road instead of the sidewalk
 - Rules of the road: drive only on the right or left, not both!
 - Long-term goals: exit the parking lot at one of a few locations
- Consider how a reward of this type might generalize better than the $a = f(x)$ behavior model



Case Study: Apprenticeship Learning for Helicopter Aerobatics



Big ideas and overview

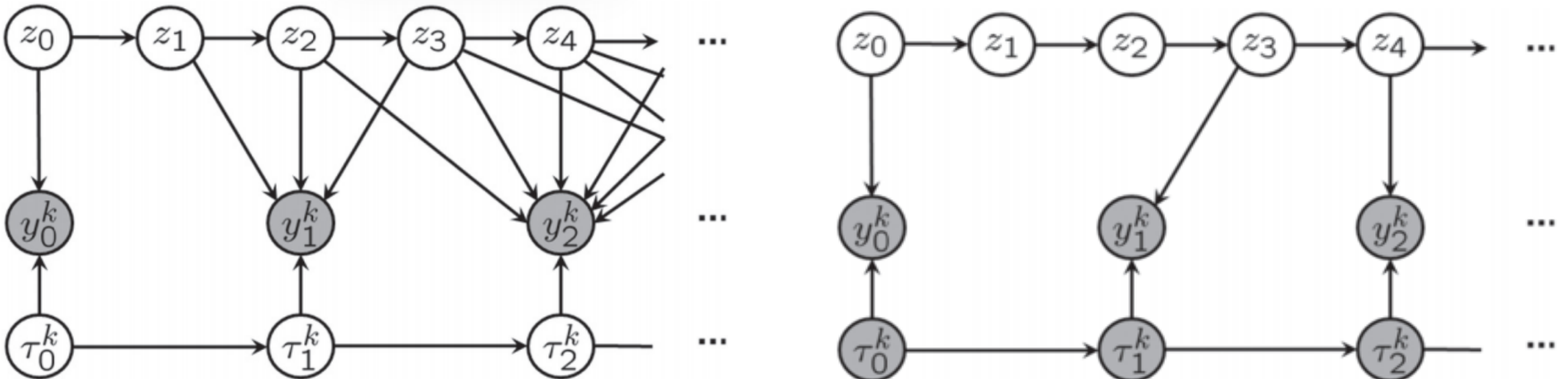
- The RL/learning-control strategies we've considered so far are not appropriate for a real physical helicopter (unless you want to support the local aviation sales industry)
- Human pilots are a great source of information to bootstrap learning into a safe regime. Multiple ways to use the "expert" humans explored in this paper and related works.
- Must still be formal with approaches to derive and update policies over time. Could we eventually outperform the "expert"?

Components of the Method

- Learn a task specification from demonstrations
- Learn a dynamics model mixing human-derived parameterizations and model fitting with demonstration/live data
- Trajectory optimization to generate autonomous control
- Experimental validation with some of the most "edge of envelope" behaviors seen to date

Inferring desired trajectory

- In this paper, we assume that the expert produces actions y , that differ from their true intention, z , with slight noise
- Different demonstrations may not align with the intentions precisely in time, so we model an offset for each observation, τ
- Inference can be derived for τ unknown (left) or known (right)



Trajectory Inference Method

- Optimization over many types of unknown parameters of the joint likelihood:

$$\max_{\tau, \Sigma^{(\cdot)}, \mathbf{d}} \log \mathbb{P}(\mathbf{y}, \rho, \tau ; \Sigma^{(\cdot)}, \mathbf{d}).$$

- Factor optimization into several tractable steps:
 - Result is an approximation algorithm

1. Initialize the parameters to hand-chosen defaults.
A typical choice: $\Sigma^{(\cdot)} = I, d_i^k = \frac{1}{3},$
 $\tau_j^k = j \frac{T-1}{N^k-1}.$
2. *E-step for latent trajectory*: for the current setting of $\tau, \Sigma^{(\cdot)}$ run a (extended) Kalman smoother to find the distributions for the latent states, $\mathcal{N}(\mu_{t|T-1}, \Sigma_{t|T-1})$.
3. *M-step for latent trajectory*: update the covariances $\Sigma^{(\cdot)}$ using the standard EM update.
4. *E-step for the time indexing (using hard assignments)*: run dynamic time warping to find τ that maximizes the joint probability $\mathbb{P}(\mathbf{z}, \mathbf{y}, \rho, \tau)$, where \mathbf{z} is fixed to $\mu_{t|T-1}$, namely the mode of the distribution obtained from the Kalman smoother.
5. *M-step for the time indexing*: estimate \mathbf{d} from τ .
6. Repeat steps 2–5 until convergence.

System Identification (Learning)

- Humans have described helicopter dynamics using non-linear differential equations. Captures drag, gravity, momentum, Coriolis etc
- For any given helicopter, determining the precise values of parameters requires fitting to data from the system. Traditionally this is known as "system identification"
- This paper learns sets of parameters and bias(error) terms locally

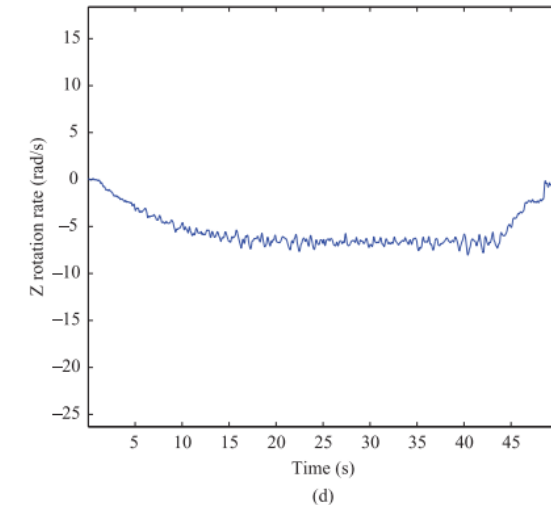
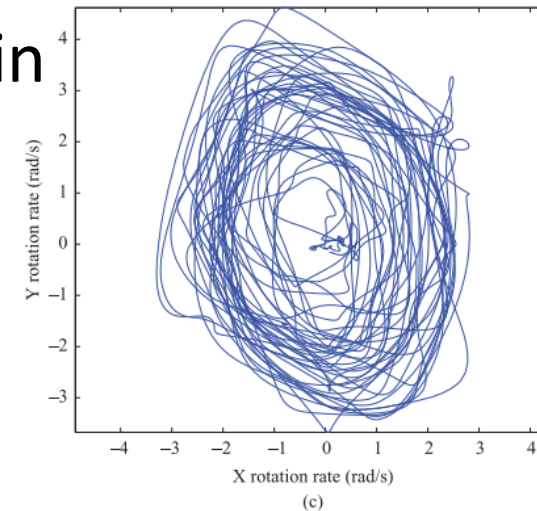
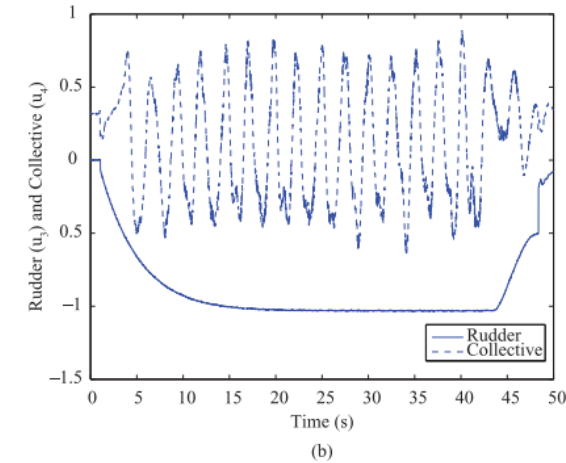
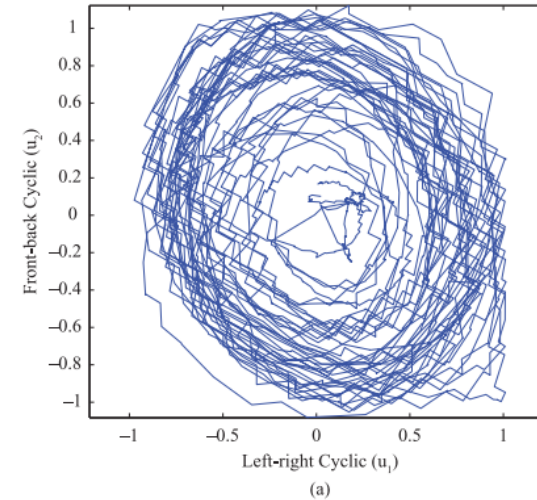
$$\begin{aligned}\dot{u} &= vr - wq + A_x u + g_x + w_u, \\ \dot{v} &= wp - ur + A_y v + g_y + D_0 + w_v, \\ \dot{w} &= uq - vp + A_z w + g_z + C_4 u_4 + D_4 + w_w, \\ \dot{p} &= qr(I_{yy} - I_{zz})/I_{xx} + B_x p + C_1 u_1 + D_1 + w_p, \\ \dot{q} &= pr(I_{zz} - I_{xx})/I_{yy} + B_y q + C_2 u_2 + D_2 + w_q, \\ \dot{r} &= pq(I_{xx} - I_{yy})/I_{zz} + B_z r + C_3 u_3 + D_3 + w_r.\end{aligned}\tag{1}$$

Control scheme

- Iterative LQR to derive optimal trajectories and controllers
 - In this context, can you think of a good way to make initial guesses?
 - Think about where our cost/reward and system dynamics are coming from
- Applied in a receding horizon fashion (Model-Predictive Control):
 - Compute optimal controls for N steps into the future
 - Execute $k < N$ steps of control, noting that the true states visited will differ from the optimal computed trajectory
 - Loop back to first step

Experimental Achievements

- Stable flight over a number of maneuvers
- Very acrobatic motions:
 - Auto-rotation
 - Flips, rolls, tic-toc, chaos!
- Better than human pilot performance in several cases





Wrap-up and Discussion

- Summary:
 - This paper demonstrates the use of our learning control tools, along with the best human expertise and engineering possible to achieve something amazing.
 - Core components are learning dynamics models robustly from data, capturing the intention behind the human's demonstrations in a sophisticated way, and computing optimal controllers in this context
- Context:
 - This is one of the most influential papers to bring learning into robotics.
 - It shows how to tackle all of the factors that make roboticists hesitant: how to use prior knowledge, how to achieve learning safely, how to keep some guarantees and control over the method
 - Today it is more fashionable to learn "end-to-end" from demonstrations, but we have not yet seen those approaches touch this level of performance for real robots

End of 417

- Thanks so much for a great term!
 - Very nice to have all of your great questions and see you mastering these topics.
- Ways to continue to explore robotics:
 - My lab: “Mobile Robotics Lab” has open meetings where grad students and current ugrad researchers present about ongoing topics. Open to the public, just ask me for the time and place.
 - Many clubs and projects around McGill. We are the top school in Canada for robotics involvement, so plenty of opportunities: robohacks, McGill Robotics, brain-powered wheelchairs, and many more
- Do complete the course evaluations please!