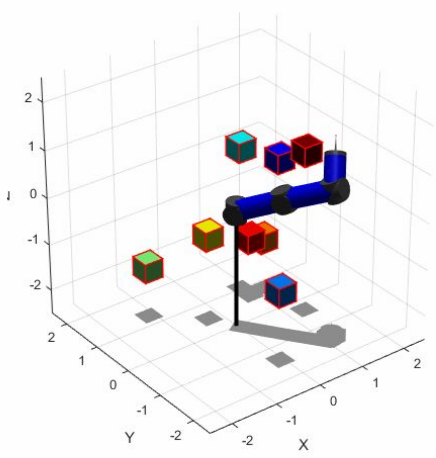
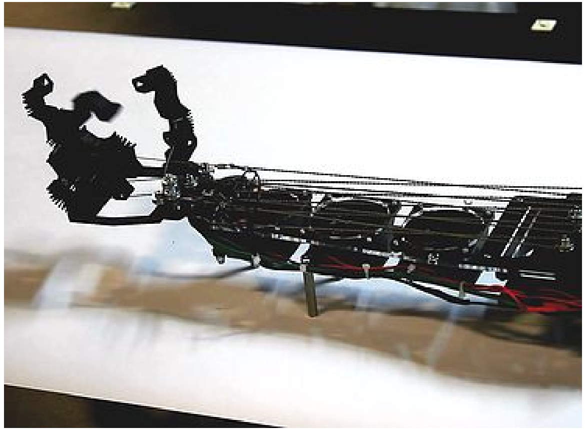
Online Model Learning and Control from Streaming Data using Gaussian Processes for Robot Teleoperation

Presentation by Brian Wilcox
2nd Year PhD in Computer Eng, UCSD
B.S. in Mechanical Eng, MIT
PhD GEM Fellow, Intel Corporation

Motivation: UCSD ARCLab

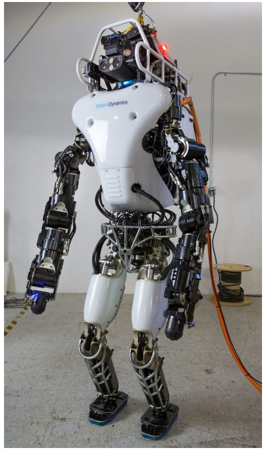
- Advanced Robotics and Control Laboratory at UC San Diego



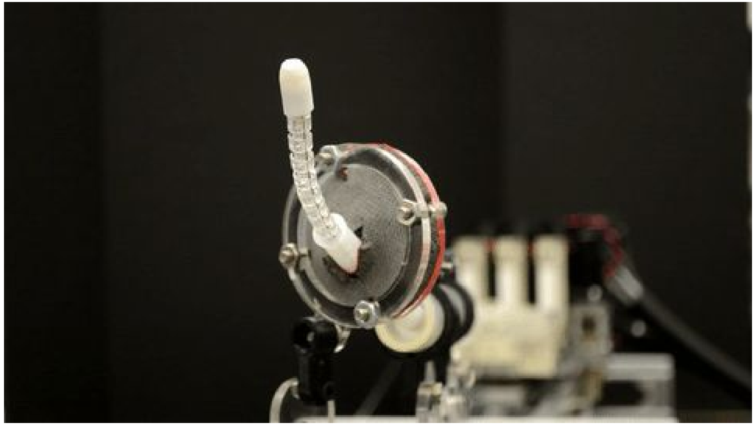


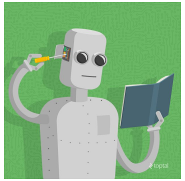
https://www.ucsdarclab.com/

Motivation: Learning to Manipulate



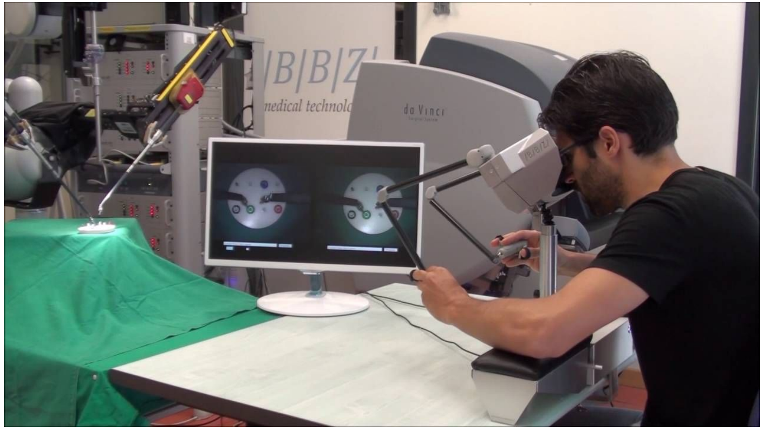
- Many complex robotic systems have:
  - Inaccurate or unknown system models
  - Unknown environment dynamics
  - Non-stationary behavior
- Through machine learning, we can *learn* a model of very complex systems, however, there remain many challenges such as:
  - Large Training Time
  - Data Availability
  - Deterministic Models
  - Appropriate model selection





Motivation: Teleoperation

- In **teleoperation tasks**, both sensor readings and motor commands come in *streams* of data
- For a data-driven controller, the robot model (mapping from state to actuator) should be *learned* and *adapt* to these streams of data, while also making accurate *predictions* about the desired actuator inputs
- In a nutshell, need a controller which is:
  - Robust
  - Adaptive
  - Real-Time
  - Data-efficient



My Research

Aim: Online model-learning and control for a teleoperation task

Plan: Sparse Online Local Gaussian Process Regression (SOLGPR) to infer a local function mapping of robot sensor states to joint states and perform a prediction of the teleoperation command for joint control.

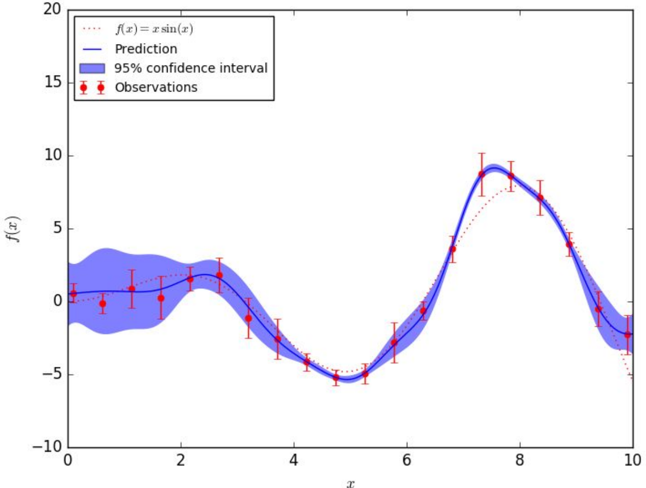
Application: Teleoperation of remote robotic catheter for catheter ablation in minimally invasive surgery

Methods: Sparse Online Local Gaussian Processes

$$f(\mathbf{x}) \sim \mathcal{GP}\left(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')\right)$$

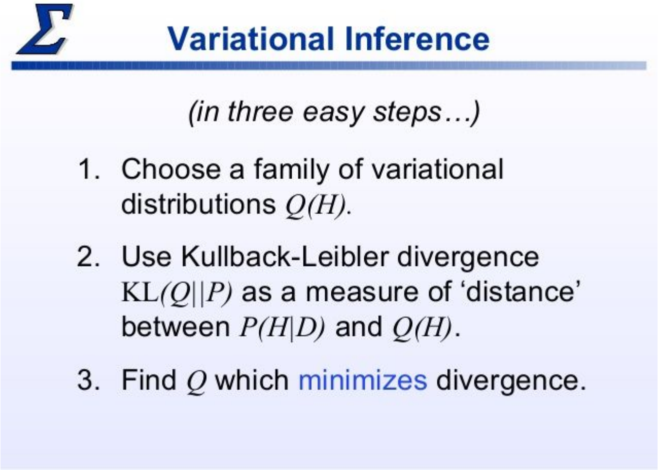
- Gaussian Process is a **distribution over functions** (Rasmussen & Williams '06)
- GP is characterized by a **mean function**, m, and **covariance function**, k
- In regression, we want to infer the function output for test inputs, f*
- We *learn* by optimizing the hyperparameters of the covariance function

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[f(\mathbf{x})], \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}[(f(\mathbf{x}) - m(\mathbf{x}))(f(\mathbf{x}') - m(\mathbf{x}'))] \end{aligned}$$



Methods: Sparse Online Local Gaussian Processes

- **Variational Inference for GP** (Titsias '09) is a method to find the optimal **M << N inducing points** which will maximize the similarity between the induced posterior distribution and the full posterior distribution via the KL divergence
- The **computational complexity can be reduced** from O(N^3) *inversion operation* to O(NM^2)
- minimize $\mathrm{KL}(q(\mathbf{f}, \mathbf{f}_m) || p(\mathbf{f}, \mathbf{f}_m | \mathbf{y}))$



Methods: Sparse Online Local Gaussian Processes

- A local model of the robot system should allow for adaptive behaviors in the region of our workspace
- In GP, can find **K local models** of our dataset, each with D local points (Nguyen-Tuong et al. '08)

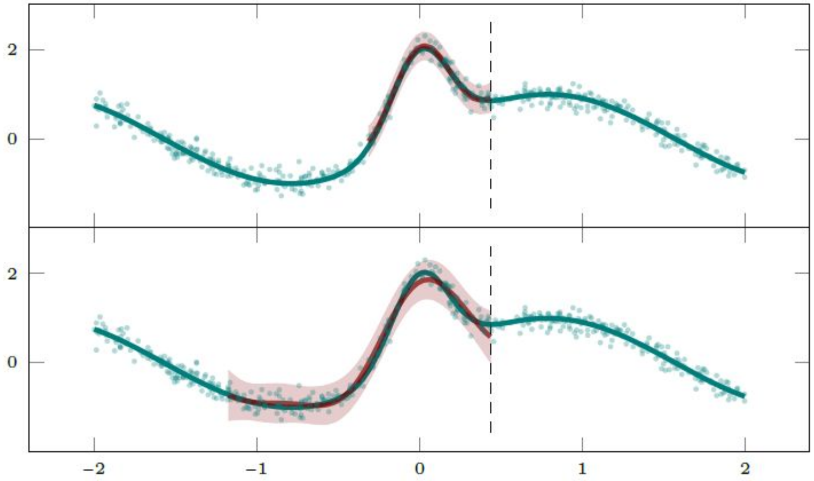
$$w_k = exp(-\frac{1}{2}(x - c_k)^T W (x - c_k))$$

- For prediction, uses a **modified* weighted prediction**

$$y_p = \frac{\sum_{k=1}^{M} w_k y_k e^{-V_k}}{\sum_{k=1}^{M} w_k e^{-V_k}}$$

Methods: Sparse Online Local Gaussian Processes

- For online, we want to **deal with minibatches** or an incremental update
- In *addition* to local models and sparsity, we use a **Drifting Gaussian Process** to speed up performance for online:
  - "Drifting" minibatches of data, where older/non-rich data points deleted as new data added (Meier & Schaal '16)
  - An optimization of the model is performed after a set interval
  - Increases adaptability to sudden changes



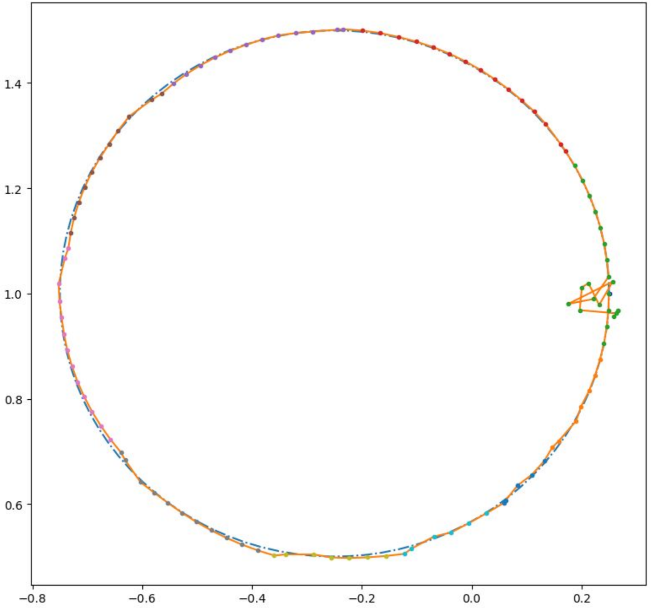
The Algorithm Summary

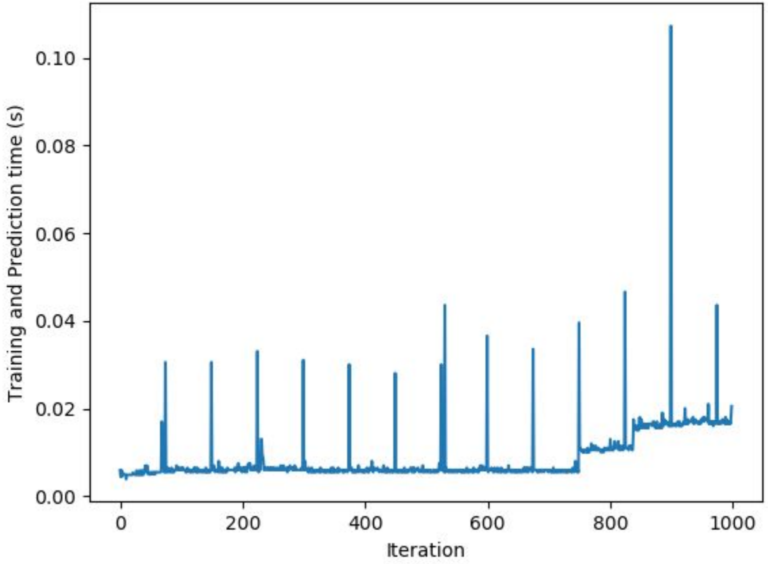
Summary of SOLGPR procedure:

1) **Initialize:** jitter robot to create initial GP models
2) **Teleoperate:** receive current command signal
3) **Control:** predict joint command and control robot
4) **Experience:** receive current robot state
5) **Partition:** Partition new experience into local regions
6) **Train:** Update and train sparse local models
7) **Learn:** Optimize drifting GP
8) **Loop:** Repeat 2-7

Results: Initial Test

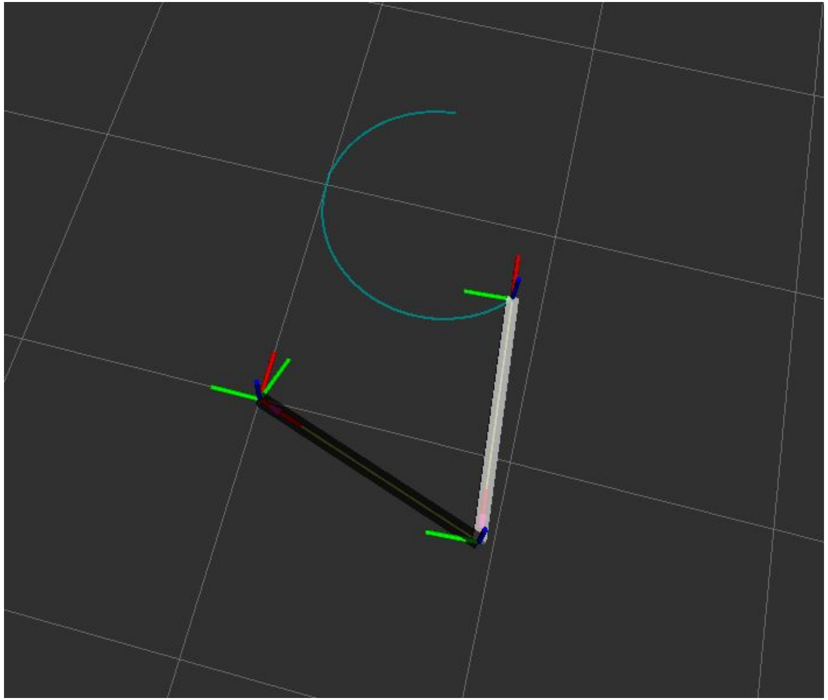
- Control of two-link arm on circular trajectory

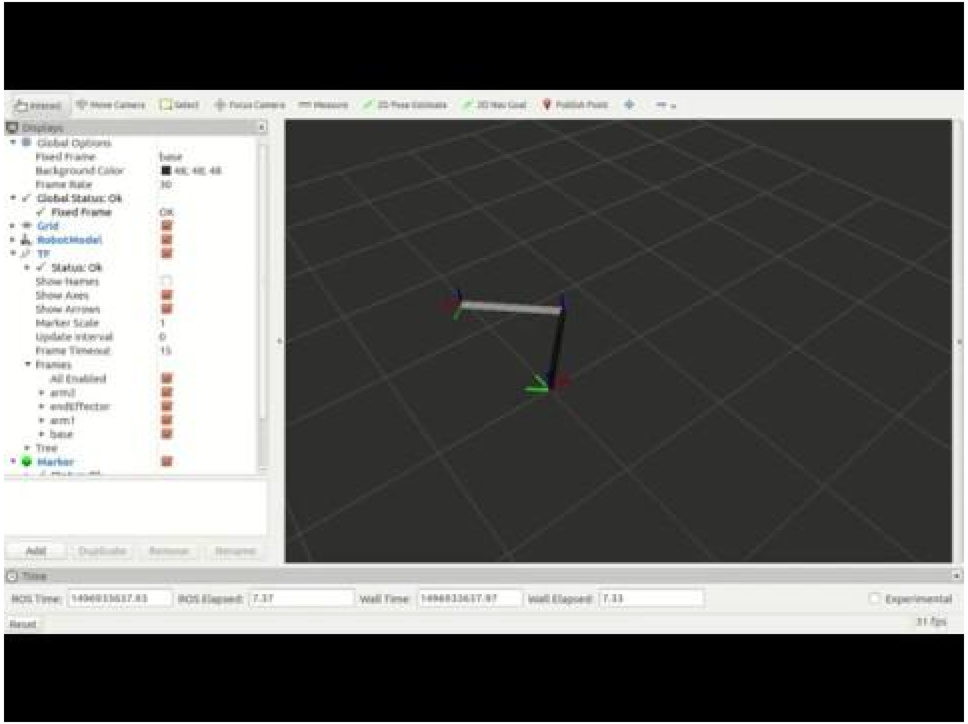




Results: ROS

- Robot Operating System (ROS) implementation of teleoperation





Looking Ahead

- Tune parameters for increased accuracy and adaptability
- *Faster, faster, faster* : take advantage of +1 data point incremental updates
- Optimize tradeoff of computational complexity and performance
- Incorporate algorithm into *model-based reinforcement learning*

THANK YOU. QUESTIONS?