A Probabilistic Framework for Learning Kinematic Models of Articulated Objects

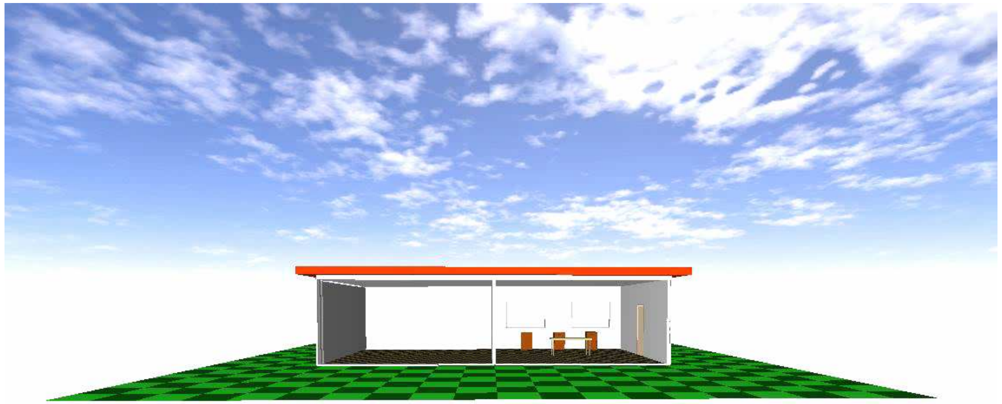
Jürgen Sturm

University of Freiburg, Germany

Motivation

Service robots in domestic environments need the capability to deal with articulated objects

- Cabinets
- Drawers
- Doors
- Windows
- Fridge
- Table
- Garage door



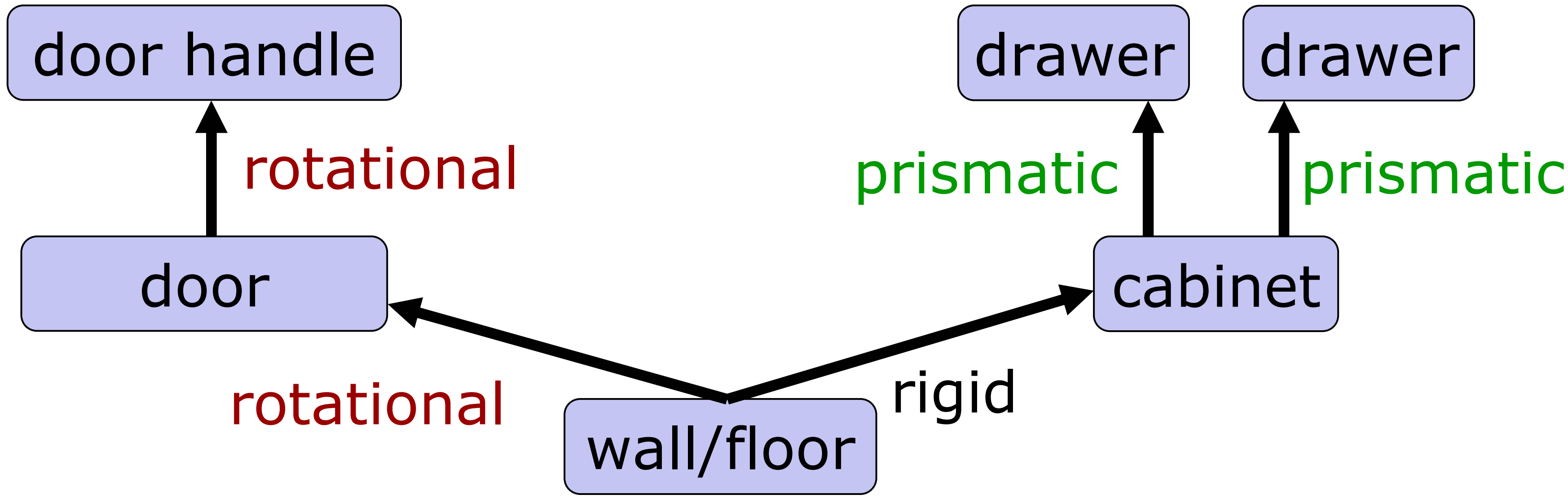
Problem: Furniture is different in each home

Motivation

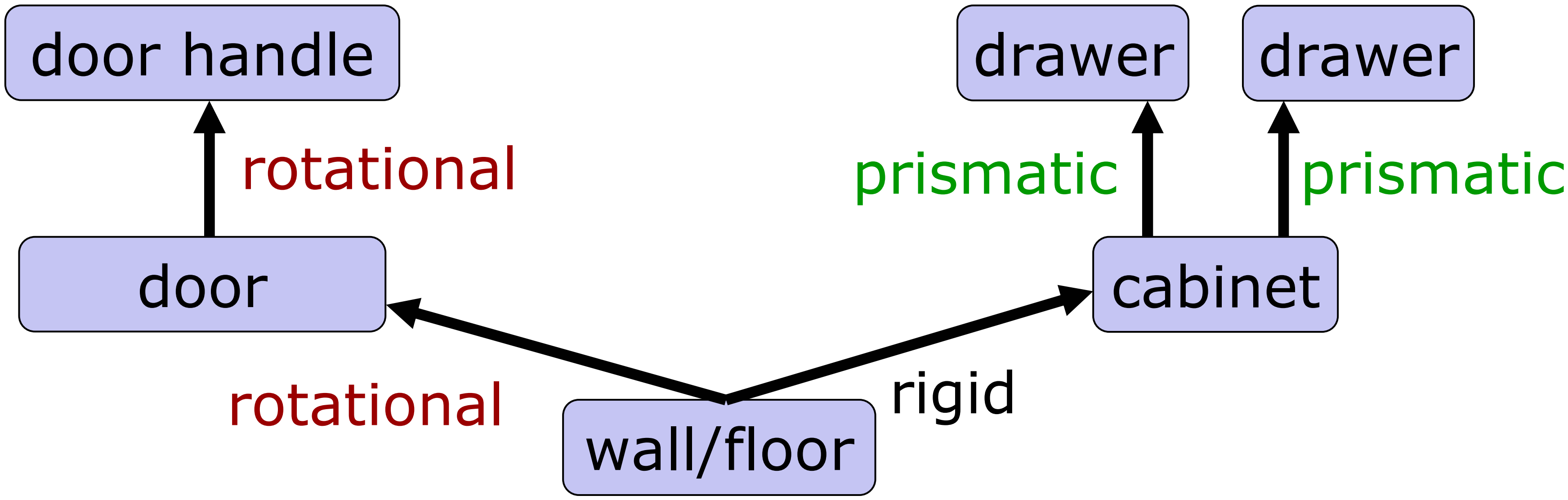
- Why learn a kinematic model?

- Improve interaction skills over time
- Generalize to unseen objects
- Allows robot to answer questions, such as:
  - Is this a door?
  - Did I succeed in opening the door?
  - In what state is the door?
  - In which other states can the door be?
  - How far can I open this door?

Goal of our Approach: Learn a articulated scene model



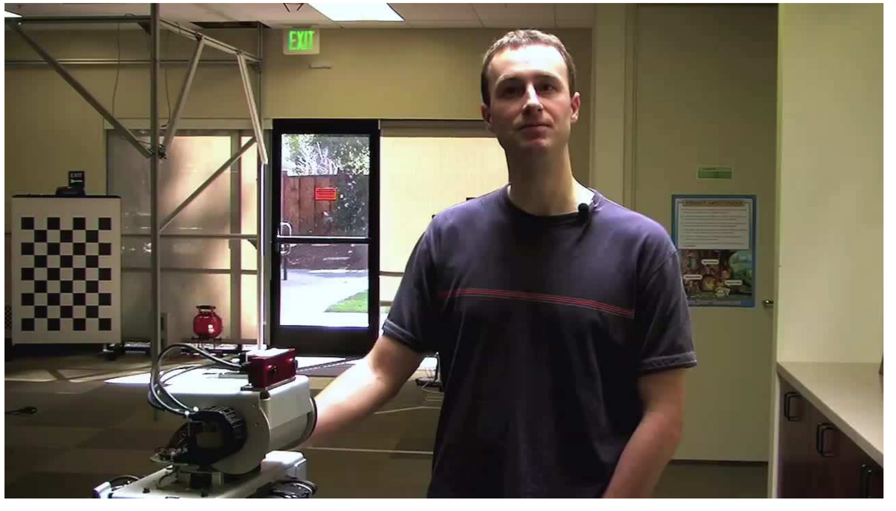
Goal of our Approach: Learn a kinematic scene model

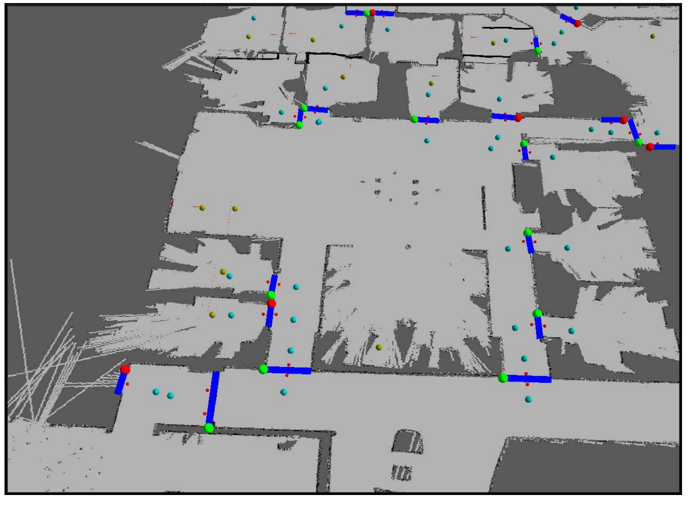


1. learn models describing the relationship between two object parts
2. infer the kinematic topology of the scene (which object parts are connected in which way)

Related Work (1)

- Door and door handle detection
- Robust control
- Door locations specified in map
- Scripted turn and push motion

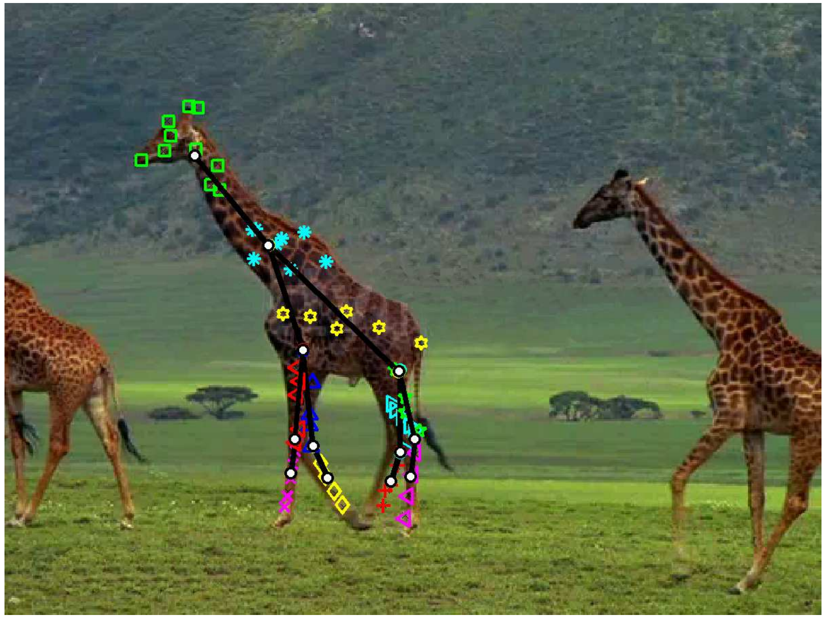




[Meeussen, Wise, Glaser, Chitta, McGann, Mihelich, Marder-Eppstein, Muja, Eruhimov, Foote, Hsu, Rusu, Marthi, Bradski, Konolige, Gerkey, Berger, ICRA 2009]

Related Work (2)

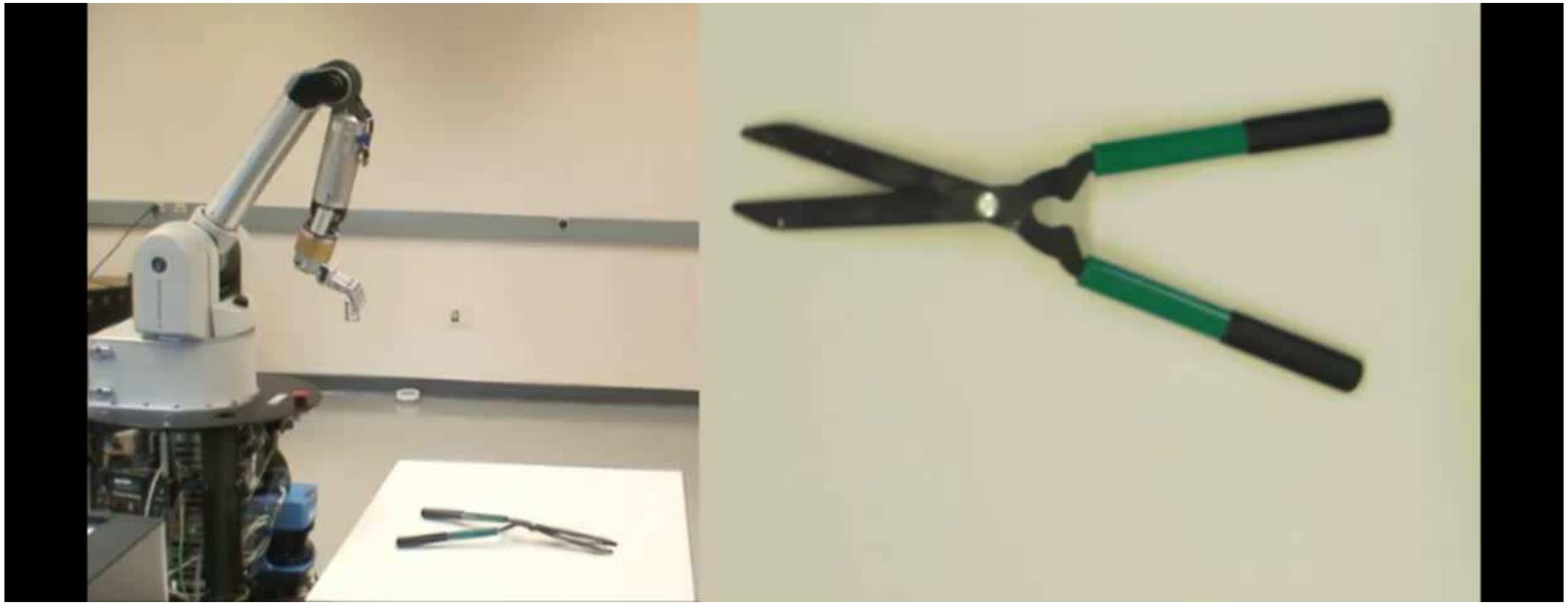
- Motion Capture and Video
- 2D/3D Feature Tracks
- Recover stick figures
- Learns graphical model

[Ross, Tarlow and Zemel, IJCV 2010]

Related Work (3)

- Manipulator + Camera
- Interactive Perception
- Tracks KLT-Features
- Min-cut algorithm on feature graph



[Katz and Brock, RSS 2008]

Features of our approach

- Fully 3D
- Accurate kinematic models
- Recover structure
- Control object with a manipulator
- Open-source, well-documented, ..

Topics covered in this talk

Bayesian learning of kinematic models for:

1. Articulated links
   - Accurate model fitting for articulated links
   - Bayesian model comparison
2. Articulated objects
   (Consisting of multiple articulated links)
   - Structure selection
   - Estimating the effective DOFs
3. Integration in ROS

Part 1: Problem Definition

- Given a sequence of pose observations of an articulated link …

$$\mathcal{D}_{\mathbf{z}} = (\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^t)$$

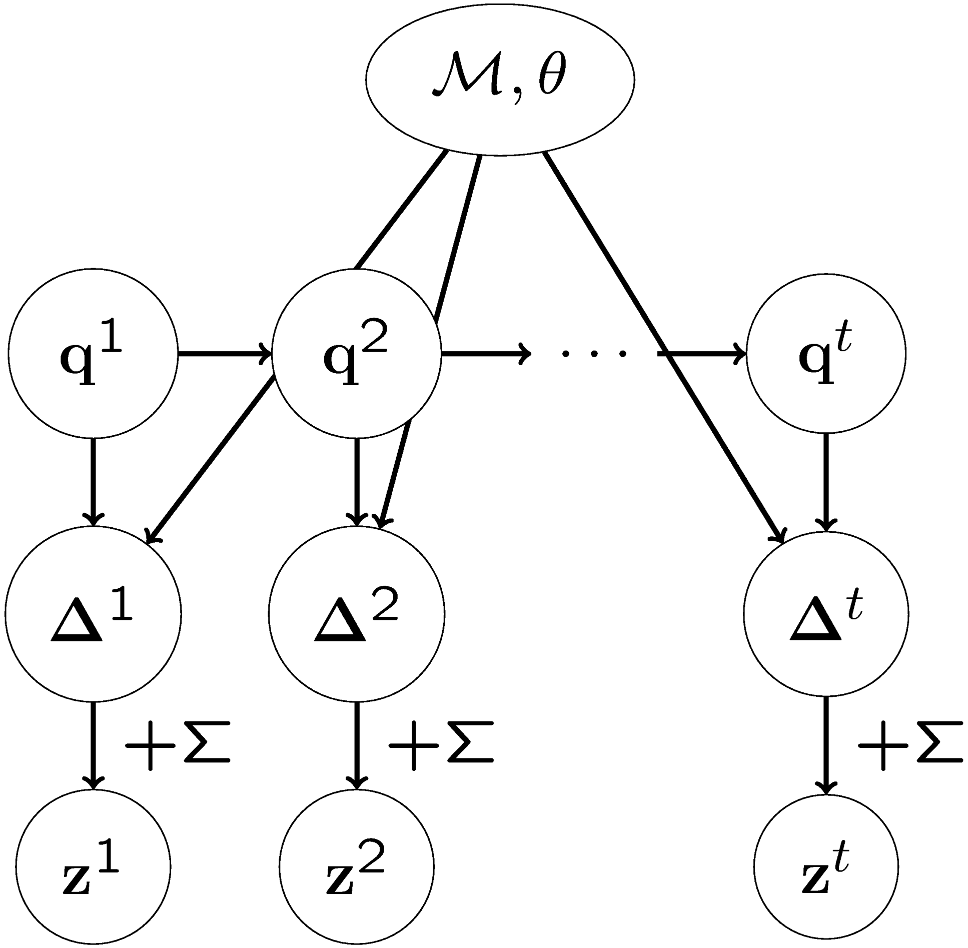
  where $\mathbf{z}^t \in SE(3)$ is a 3D pose including position and orientation

- … estimate the most likely model and parameter vector

$$\hat{\mathcal{M}}, \hat{\theta} = \underset{\mathcal{M},\theta}{\arg\max}\, p(\mathcal{M}, \theta \mid \mathcal{D}_{\mathbf{z}})$$

Process Model

- Kinematic model
- Configuration
- True pose
- Observed pose



Bayesian Model Inference

Solving

$$\hat{\mathcal{M}}, \hat{\theta} = \operatorname*{arg\,max}_{\mathcal{M},\theta} p(\mathcal{M}, \theta \mid \mathcal{D}_{\mathbf{z}})$$

can be split into two steps of inference:

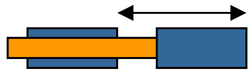
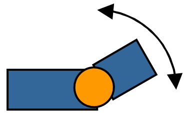
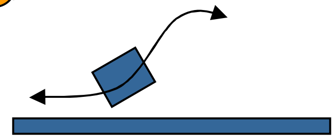
1. Model Fitting

$$\hat{\theta} = \operatorname*{arg\,max}_{\theta} p(\theta \mid \mathcal{D}_{\mathbf{z}}, \mathcal{M})$$

2. Model Comparison

$$\hat{\mathcal{M}} = \operatorname*{arg\,max}_{\mathcal{M}} \int p(\mathcal{M}, \theta \mid \mathcal{D}_{\mathbf{z}}) d\theta$$

Model Fitting (1)

- Fit different model classes:
  - Rigid Model 
  - Prismatic Model 
  - Rotational Model 
  - Gaussian Process Model 

- Each model has a
  - Forward kinematics function $\boldsymbol{\Delta} = f_{\mathcal{M},\theta}(\mathbf{q})$
  - Inverse kinematics function $\mathbf{q} = f^{-1}_{\mathcal{M},\theta}(\boldsymbol{\Delta})$

Model Fitting (2)

- Maximum-likelihood estimator for each model (MLESAC)

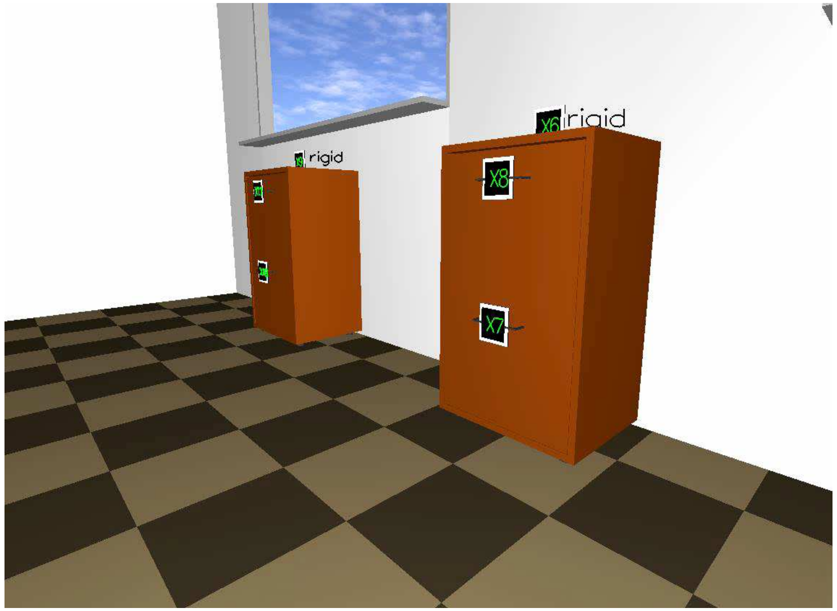
$$\hat{\theta} = \arg\max_{\theta} p(\mathcal{D}_{\mathbf{z}} \mid \mathcal{M}, \theta)$$

- Robust data likelihood
  - Assume that the process noise is sampled from a mixture of a uniform distribution and a Gaussian distribution

$$\mathbf{z} \sim \begin{cases} \boldsymbol{\Delta} + \mathcal{N}(0, \Sigma_{\mathbf{z}}) & \text{if inlier} \\ \mathcal{U}(W) & \text{if outlier} \end{cases}$$

Prismatic Model

- Parameters:
  - origin a
  - axis *e* of movement



- Forward kinematics function

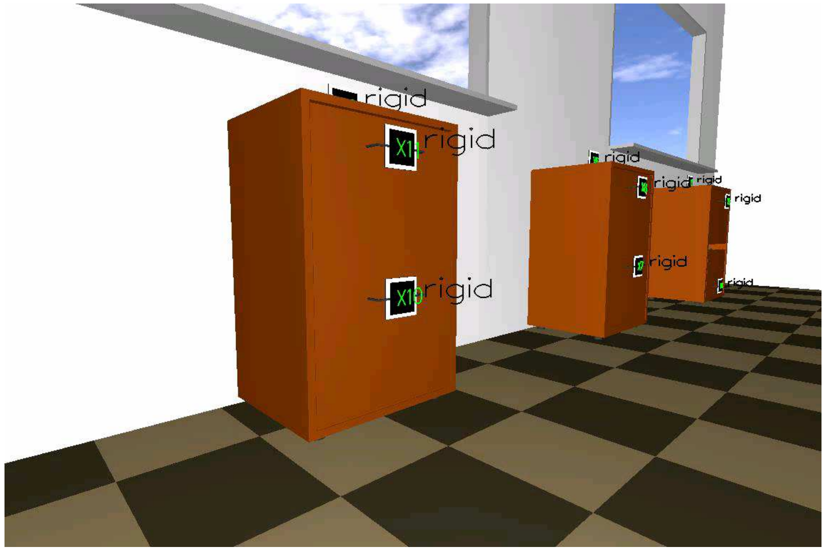
$$f_{\mathcal{M}^{\text{prismatic}},\theta}(q) = \mathbf{a} \oplus \mathbf{e}q$$

- Inverse kinematics function

$$f^{-1}_{\mathcal{M}^{\text{prismatic}},\theta}(\mathbf{z}) = \mathbf{e}^T trans(\mathbf{a} \ominus \mathbf{z})$$

Rotational Model

- Parameters
  - center of rotation and rotation axis *c*
  - rigid transform *r*



- Forward kinematics function

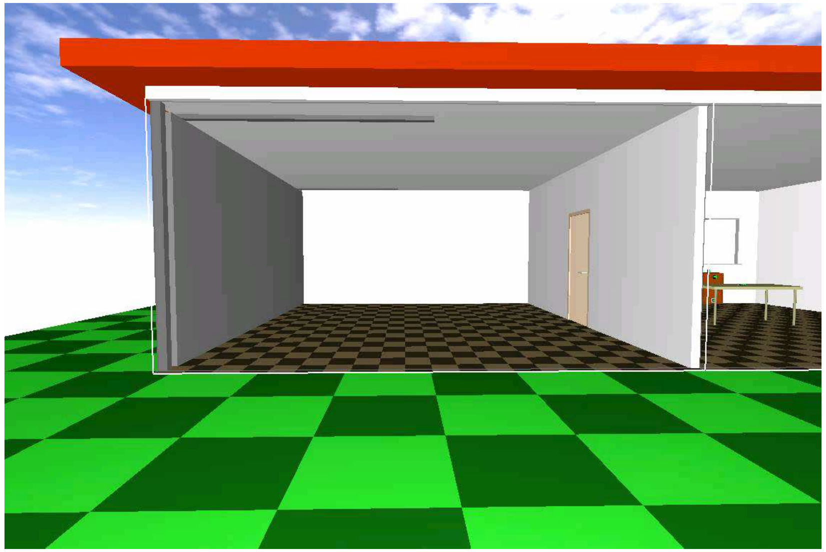
$$f_{\mathcal{M}^{\text{rotational}},\theta}(q) = \mathbf{c} \oplus Rot_Z(q) \oplus \mathbf{r}$$

- Inverse kinematics function

$$f^{-1}_{\mathcal{M}^{\text{rotational}},\theta}(\mathbf{z}) = Rot_Z^{-1}(\mathbf{c} \ominus (\mathbf{z} \ominus \mathbf{r}))$$

Garage Door: A Two-bar Link

- Garage door runs in a vertical and a horizontal slider
- Neither rotational, nor prismatic motion
- There are objects which cannot be explained well by “standard” models



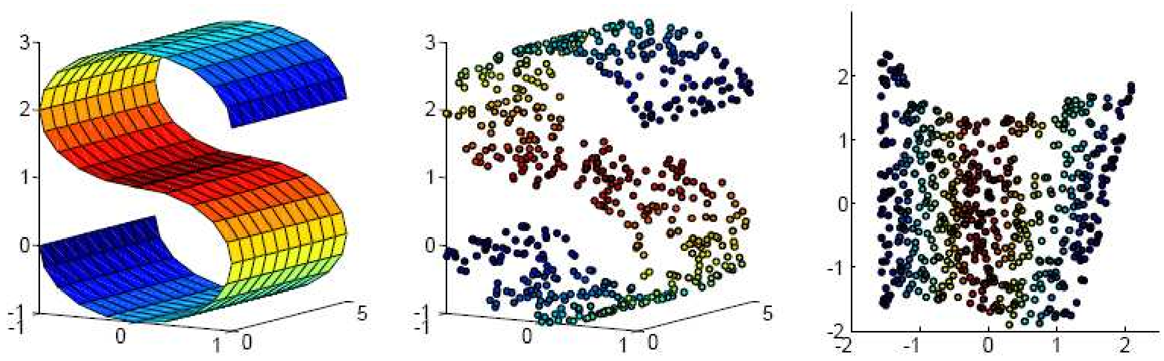
A Non-parametric Model (1)

- For a articulation model, we need to define
  - A forward kinematics function
  - An inverse kinematics function

- Assume that the data lies on
  (or close to) a low dimensional manifold in $\mathbb{R}^6$

A Non-parametric Model (1)

- Non-linear dimensionality reduction technique
- Locally Linear Embedding (LLE; other alternatives: PCA, ISOMAP, t-SNE, ..)
- Example: 2D manifold embedded in 3D space

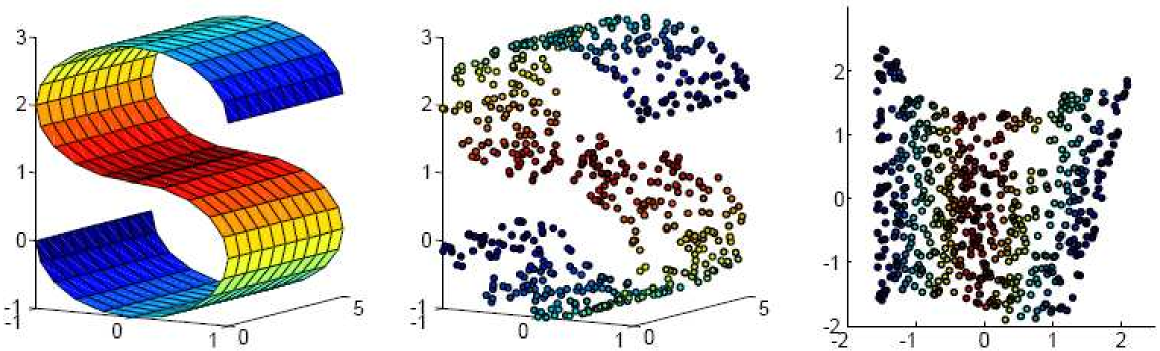


[Roweis, 2000]

A Non-parametric Model (3)

- Find latent low dimensional coordinates on the manifold → provides configurations of the object

$$f^{-1}_{\mathcal{M}^{\mathsf{GP}},\theta}(\mathbf{z}) = \mathbf{q} + \delta$$

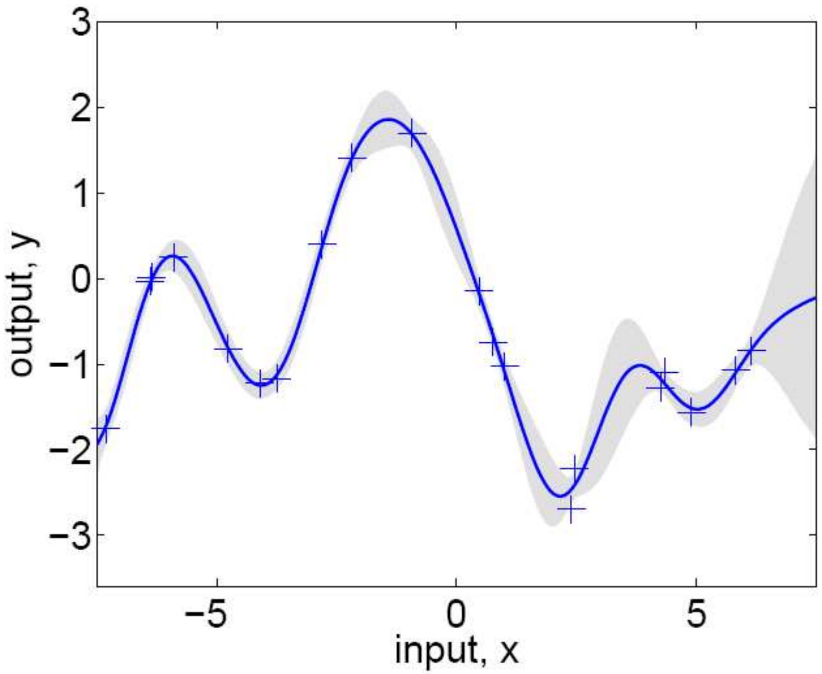


[Roweis, 2000]

A Non-parametric Model (4)

- Then learn a Gaussian process regression modeling the forward kinematics

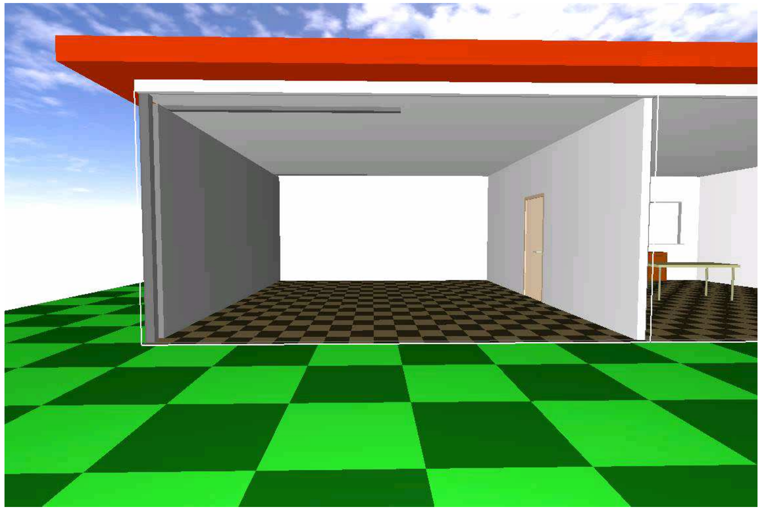
$$f_{\mathcal{M}^{\mathrm{GP}},\theta}(\mathbf{q}) = \mathbf{z} + \epsilon$$



+ data points
— mean
variance

[Rasmussen, 2006]

A Non-parametric Model (5)



- Find latent low dimensional coordinates on the manifold → dimensionality reduction using locally linear embedding (LLE) provides inverse kinematics

$$f^{-1}_{\mathcal{M}^{\mathsf{GP}},\theta}(\mathbf{z}) = \mathbf{q} + \delta$$

- Then learn a Gaussian process regression modeling the forward kinematics

$$f_{\mathcal{M}^{\mathsf{GP}},\theta}(\mathbf{q}) = \mathbf{z} + \epsilon$$

Model Evaluation (1)

- How to evaluate the data likelihood?

$$p(\mathbf{z} \mid \mathcal{M}, \theta) = ?$$

- Configuration is latent → integrate over all possible configurations

$$p(\mathbf{z} \mid \mathcal{M}, \theta) = \int p(\mathbf{z} \mid \mathbf{q}, \mathcal{M}, \theta) p(\mathbf{q}) d\mathbf{q}$$

- Approximate integral by evaluating at most likely configuration

Model Evaluation (2)

- Estimate configuration

$$\hat{\mathbf{q}} = f^{-1}_{\mathcal{M},\theta}(\mathbf{z})$$

- Predict expected pose

$$\hat{\boldsymbol{\Delta}} = f_{\mathcal{M},\theta}(\hat{\mathbf{q}})$$

- Compare prediction with observation

$$p(\mathbf{z} \mid \hat{\boldsymbol{\Delta}}) \propto \exp\left(-\|\hat{\boldsymbol{\Delta}} - \mathbf{z}\|^2/\sigma^2\right) + c$$

- Approximate data likelihood

$$p(\mathbf{z} \mid \mathcal{M}, \theta) \approx p(\mathbf{z} \mid \hat{\boldsymbol{\Delta}})p(\hat{\mathbf{q}})$$

Model Selection

- Select the model that maximizes the posterior probability

$$\hat{\mathcal{M}} = \arg\max_{\mathcal{M}} \int p(\mathcal{M}, \theta \mid \mathcal{D}_{\mathbf{z}}) d\theta$$

- Solve this using the Bayesian Information Criterion (BIC)

$$\mathrm{BIC}(\hat{\mathcal{M}}) = \underbrace{-2 \log p(\mathcal{D}_{\mathbf{z}} \mid \mathcal{M}, \theta)}_{\text{Neg. data likelihood}} + \underbrace{k \log n}_{\text{Penalty on model complexity}}$$

- Select model that minimizes the BIC

$$\hat{\mathcal{M}} = \arg\min_{\mathcal{M}} \mathrm{BIC}(\mathcal{M})$$

Examples 1/3

fridge

drawer

Examples 2/3

dishwasher

.. and tray

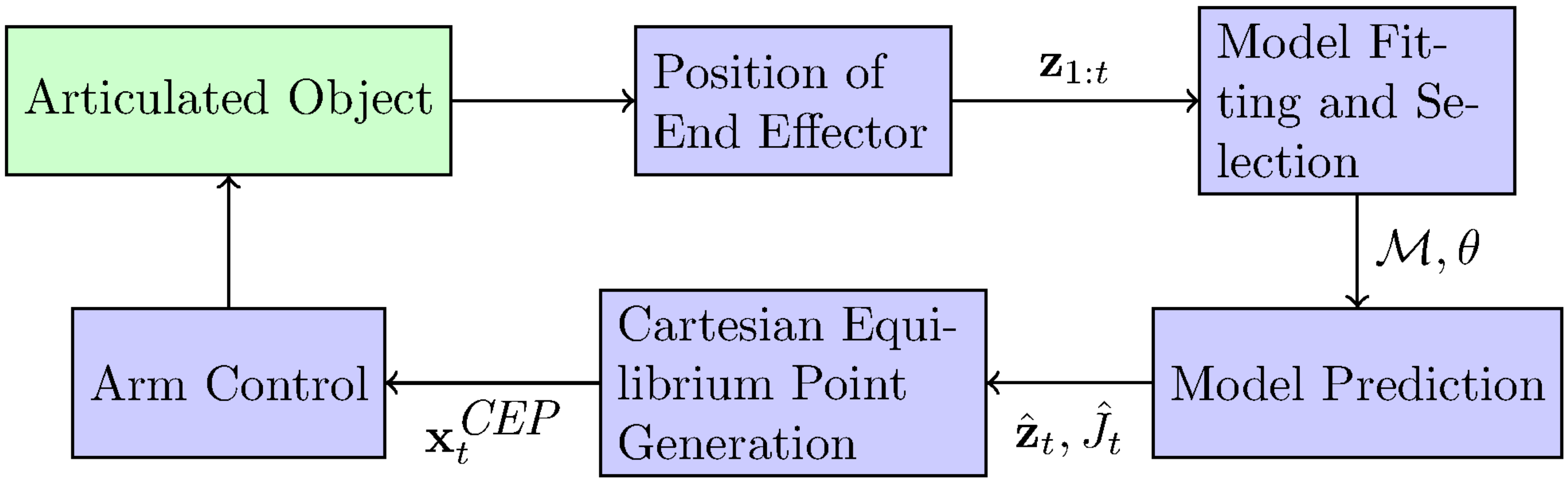
Examples 3/3

water tap

valve of a radiator

Online Estimation and Control

- Learn kinematic model while manipulating articulated object



Experimental Setup

- Experimental setup:
  - Given: 3D location of handle + initial direction
  - Robot estimates kinematic model online and in real-time
  - Robot uses estimated model for control
  - 5 different mechanisms

Experimental Results

- Video:

Joint work with Advait Jain and Charlie Kemp

- Success rate: 37 out of 40 trials (92.5%)

Exploiting Prior Information

- So far, robot learns a new model for each newly object from scratch

- However: most articulated objects in a household belong to a few different classes
  - Doors are of same/similar size
  - Standardized dimensions of kitchen interior

- Idea:
  - Find small set of representative models
  - Utilize previously learned models when handling new objects

Model Clustering

- Given two observed trajectories, should we select one or two models?

- Bayesian model comparison

If $$p(\mathcal{M}_{1+2} \mid \mathcal{D}) > p(\mathcal{M}_1, \mathcal{M}_2 \mid \mathcal{D})$$

Then: Learn single model (single set of parameters but might fit data worse)

Else: Learn two models (double set of parameters but might fit data better)

Model Clustering (2)

- Incremental clustering
- Can be done online
- Estimated model benefits from larger dataset

- Bayesian model comparison:

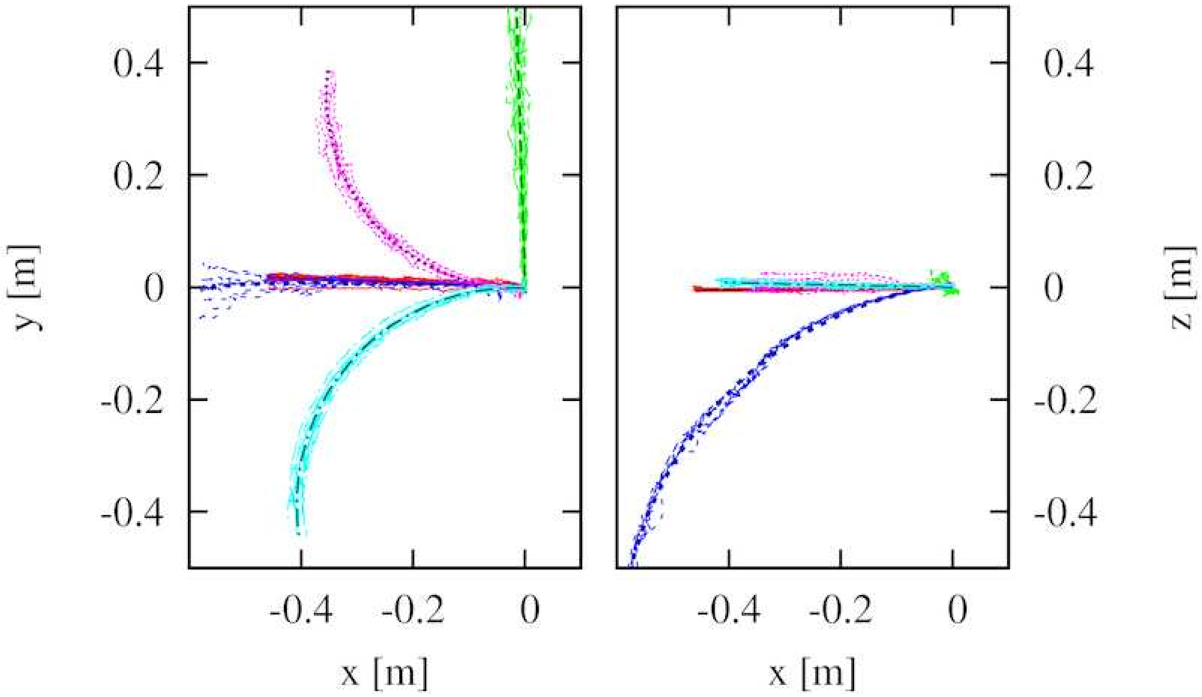
If $\max\limits_{j=1,\ldots,m} p(\mathcal{M}_1, \ldots, \mathcal{M}_{j+\mathit{new}}, \ldots, \mathcal{M}_m \mid \mathcal{D})$

$$> p(\mathcal{M}_{\mathit{new}}, \mathcal{M}_1, \ldots, \mathcal{M}_m \mid \mathcal{D})$$

Then: Merge with model j

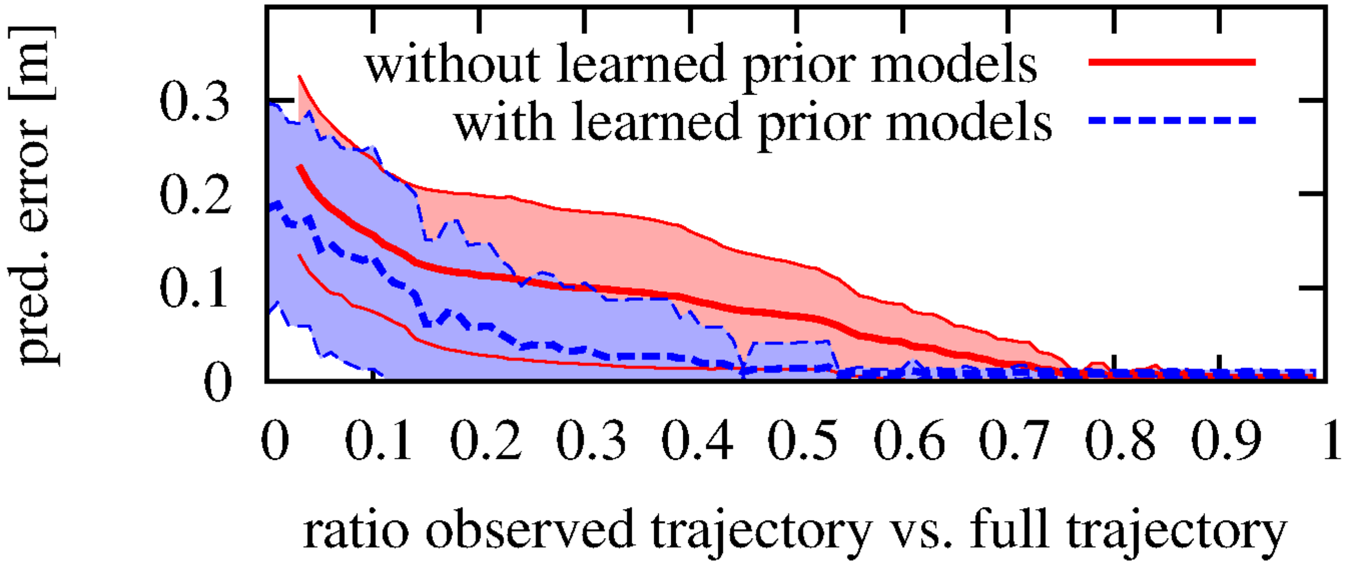
Else: Add new model

Model Clustering



- 37 trajectories
- Correctly clustered into 5 models

Exploiting Prior Information

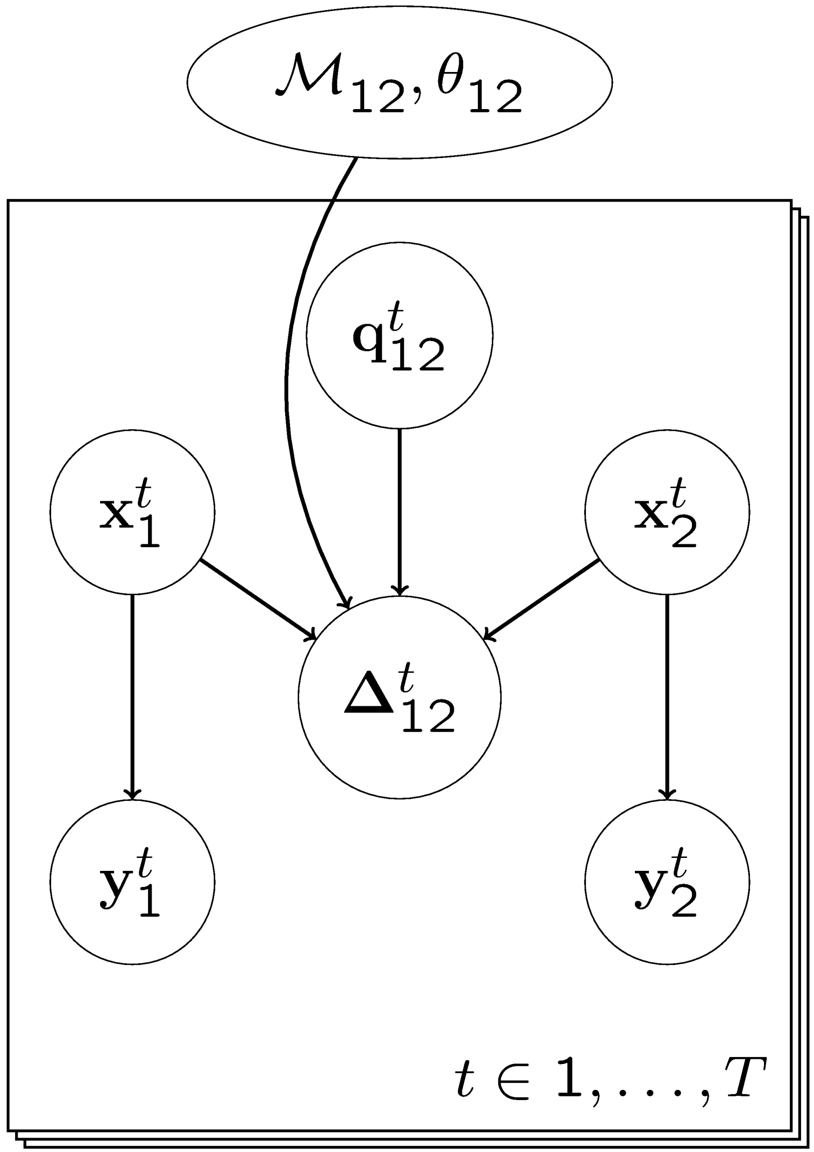


- Using prior information significantly improves prediction accuracy

Part 2: Articulated Objects

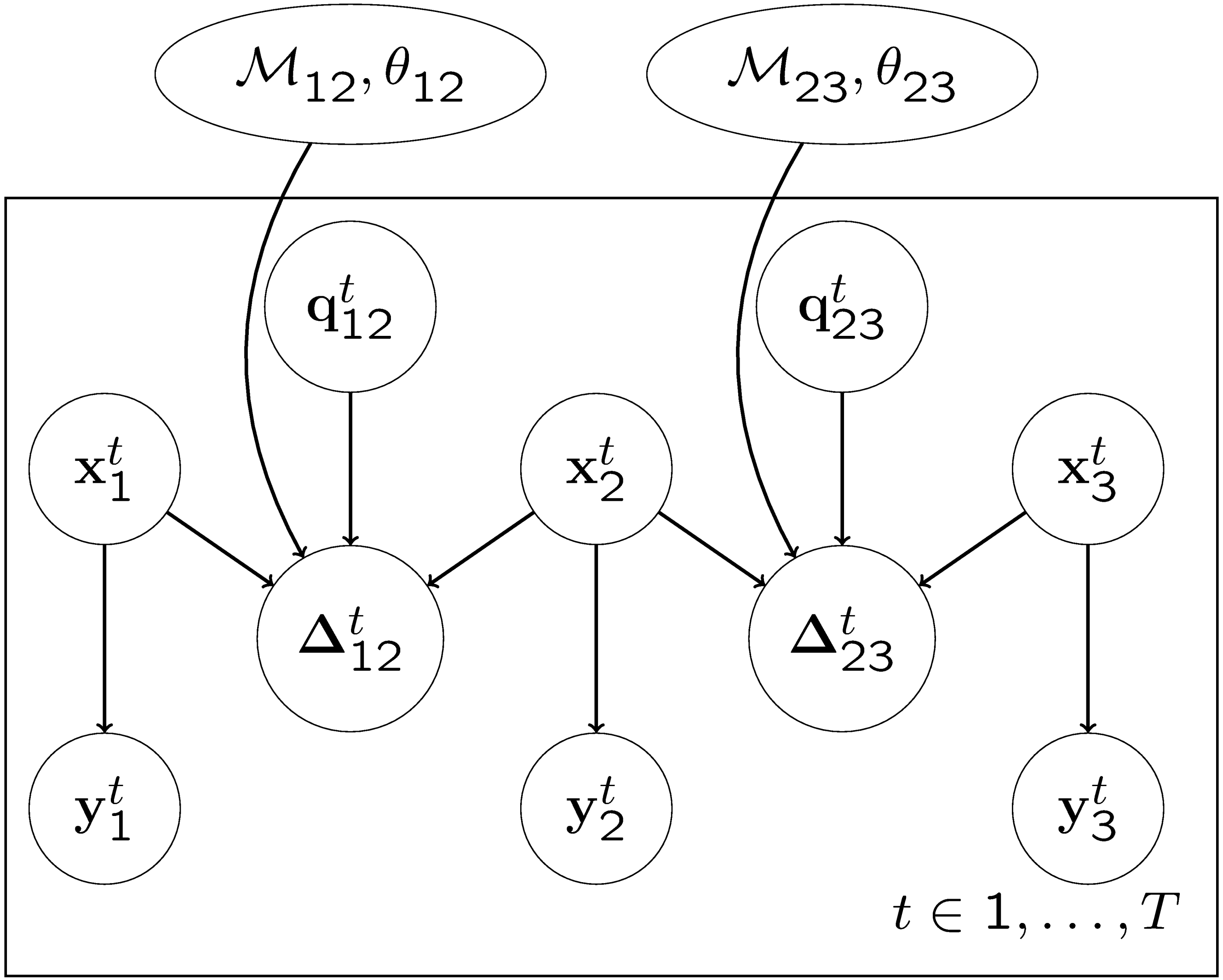
- So far, we considered only articulated objects consisting of a single link, thus of two parts
- Now, extend to p>2 parts…

Process Model for 2 parts

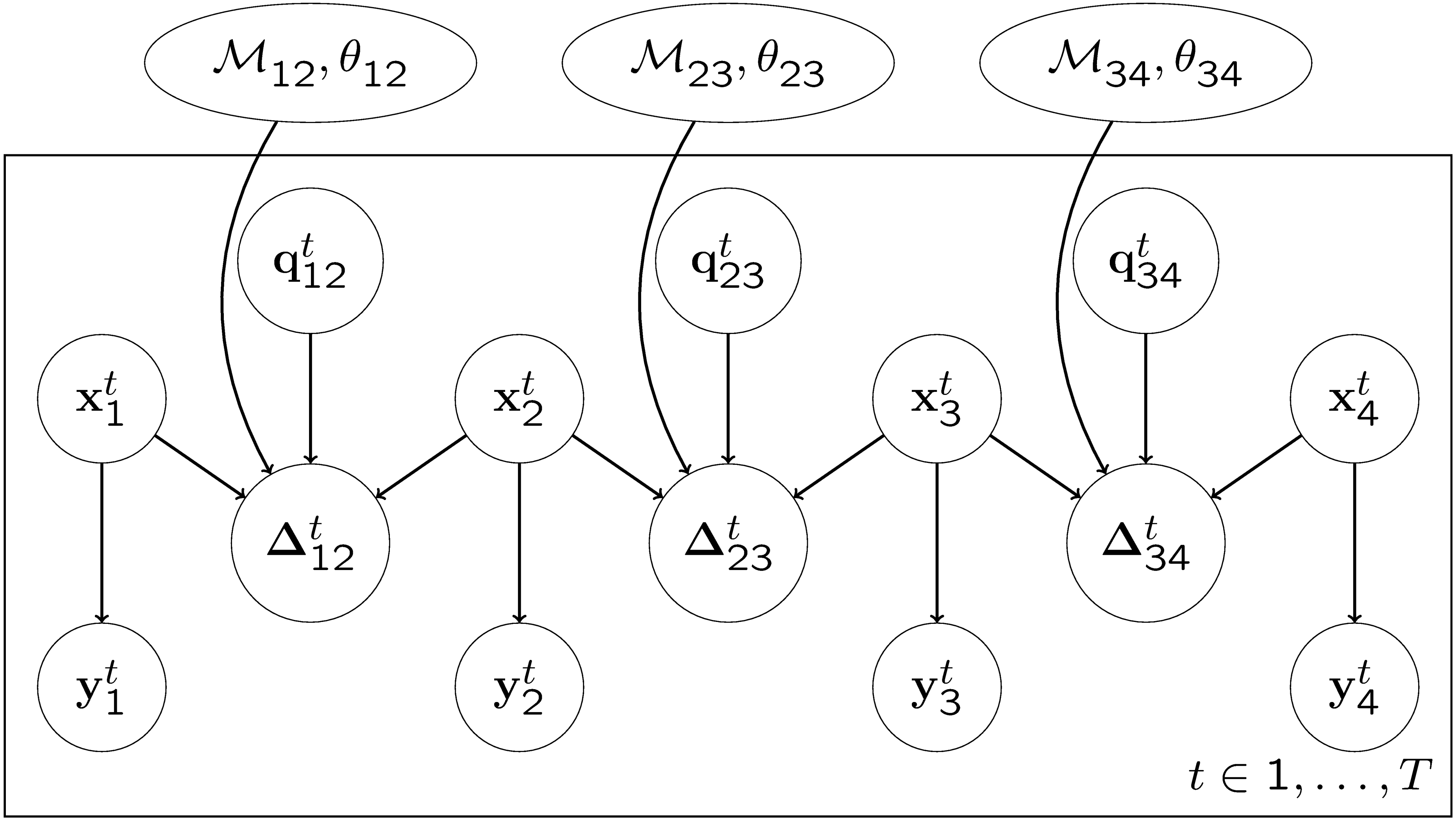


- Kinematic model
- Configuration
- True poses
- True transformation
- Observed poses

Process Model for 3-chain

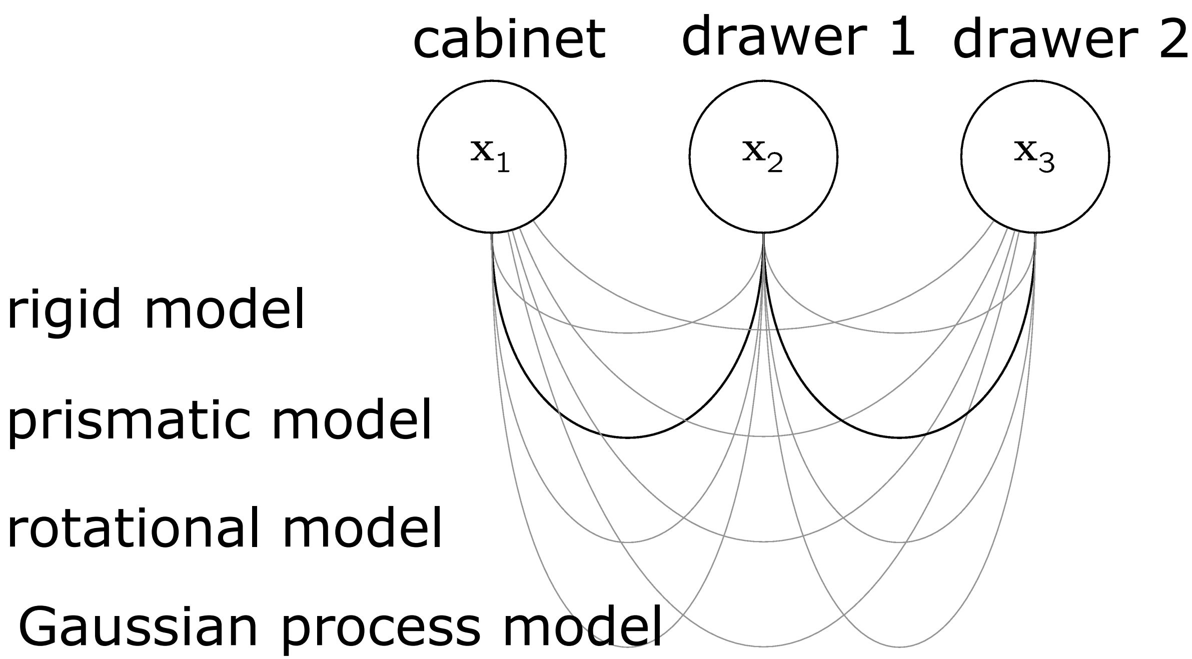


Process Model for 4-chain



Kinematic Graph (1)

- Kinematic structure is unknown → consider all possible structures, and select the best one
- Simplified graphical model (object parts and models only)



Kinematic Graph (2)

- Describe articulated objects as a kinematic graph $G = (V_G, E_G)$
  - Vertices $V_G = \{1, \ldots, p\}$ correspond to object parts
  - Edges $E_G \subset V_G \times V_G$ correspond to articulated links
  - Each edge has an associated articulated link model $\mathbb{M} = \{\mathcal{M}_{ij}, \theta_{ij} \mid (i,j) \in E_G\}$

Problem Definition

- Given a sequence of t pose observations of an articulated object consisting of p parts..

$$\mathcal{D}_{\mathbf{y}} = \begin{pmatrix} \mathbf{y}_1^1 & \mathbf{y}_1^2 & \cdots & \mathbf{y}_1^t \\ \mathbf{y}_2^1 & \mathbf{y}_2^2 & \cdots & \mathbf{y}_2^t \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{y}_p^1 & \mathbf{y}_p^2 & \cdots & \mathbf{y}_p^t \end{pmatrix}$$

- Estimate the most likely kinematic graph G

$$\widehat{G} = \arg\max_{G} p(G \mid \mathcal{D}_{\mathbf{y}})$$

Bayesian Model Inference

Solving

$$\hat{G} = \arg\max_{G} p(G \mid \mathcal{D}_{\mathbf{y}})$$

can be split into four steps of inference:

1. Link-wise model fitting (as before)
2. Link-wise model selection (as before)
3. **Object-wise structure selection**
4. **Object-wise DOF estimation**

Structure Selection (1)

- Select the graph that maximizes the posterior probability

$$\widehat{E}_G = \operatorname*{arg\,max}_{E_G} \int p(E_G, \mathbb{M} \mid \mathcal{D}_{\mathbf{y}}) d\mathbb{M}$$

➡ Select graph that minimizes the BIC

$$\widehat{E}_G = \operatorname*{arg\,min}_{E_G} \mathrm{BIC}(E_G)$$

Structure Selection (2)

- How can we find the graph that minimizes the BIC?
- Given a graph, how can we compute its data likelihood?

Structure Selection (3)

- How can we find the graph that minimizes the BIC?

➡ For kinematic trees:
  - Minimum spanning tree problem
  - Efficient and optimal solution

➡ For general kinematic graphs (including closed kinematic chains):
  - Full evaluation over all possible structures
  - Or approximation using search heuristic

Structure Selection (4)

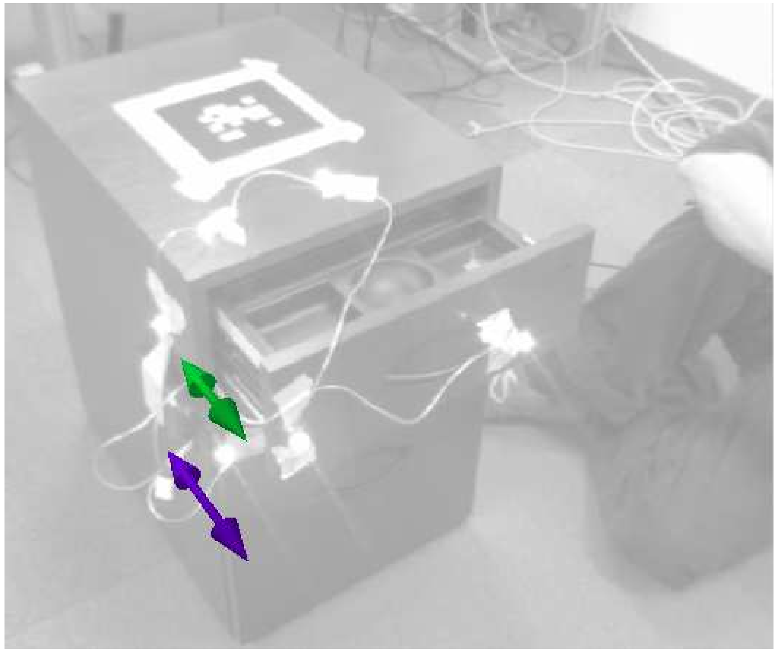
- Given a graph, how can we compute its data likelihood?
- Insight: edges of kinematic trees are mutually independent

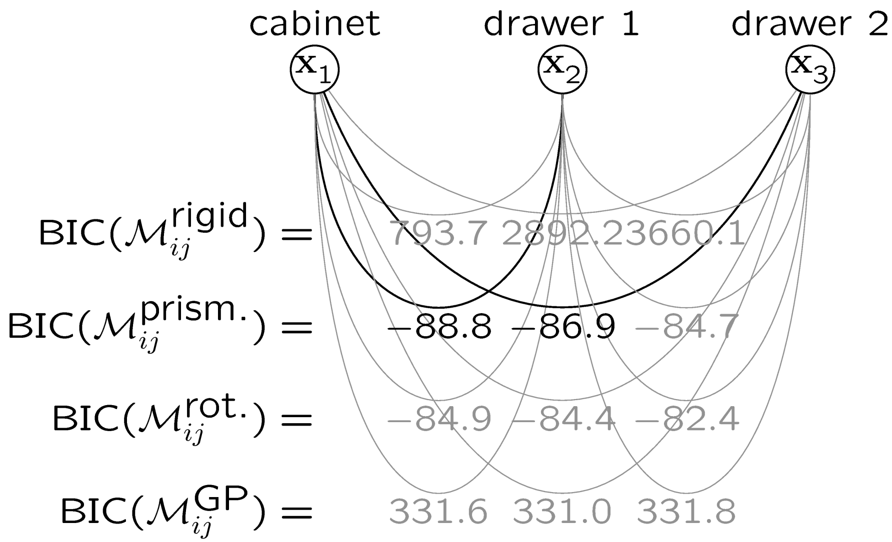
$$\hat{E}_G = \underset{E_G}{\arg\min} \sum_{(ij) \in E_G} \mathrm{BIC}(\hat{\mathcal{M}}_{ij})$$

- This corresponds to a minimum spanning tree problem
  - Fully connected graph
  - Assign edge costs

$$\mathrm{cost}_{ij} = \mathrm{BIC}(\mathcal{M}_{ij})$$

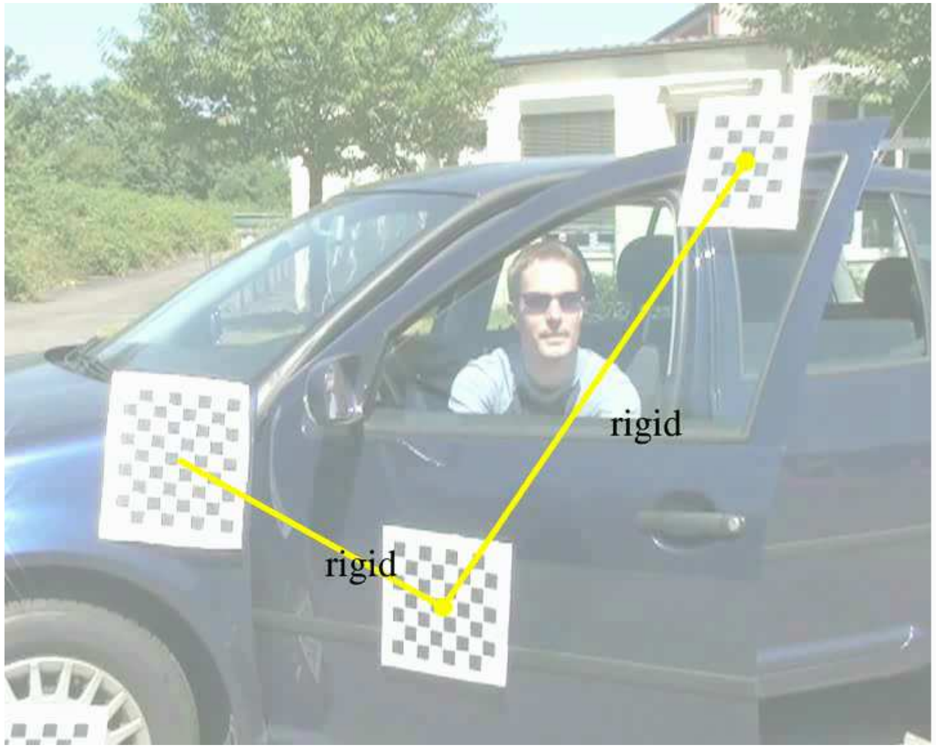
Example: Cabinet with Drawers

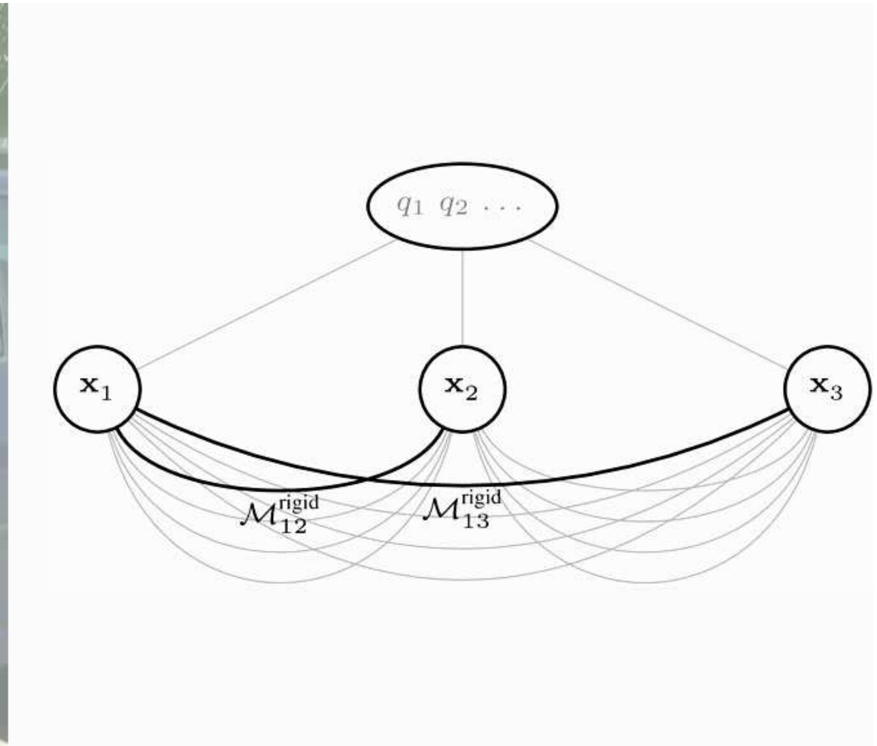




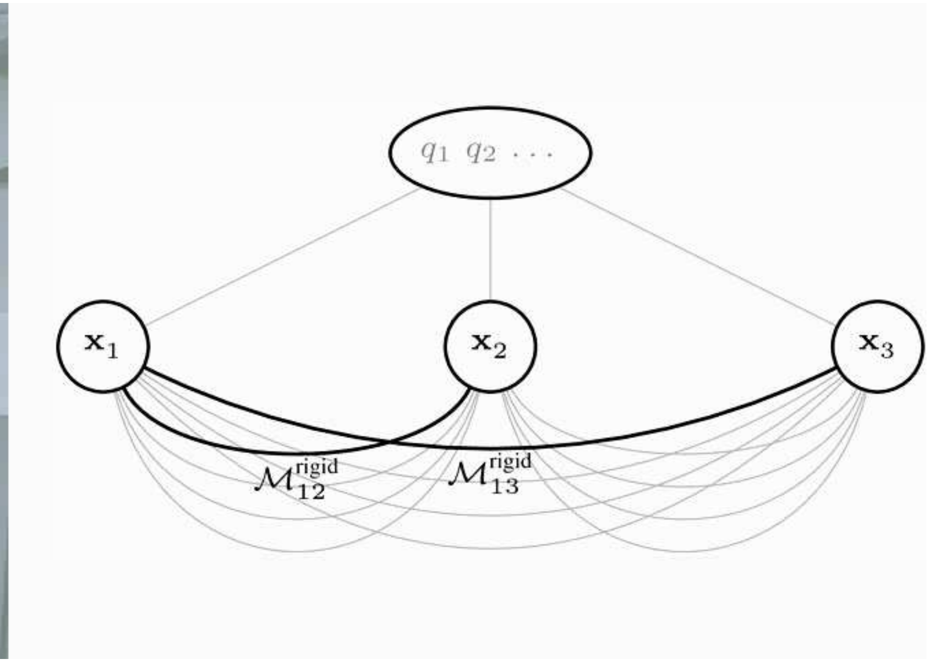
- Compute all models between all edges
- Select the minimum spanning tree

Example: Car Door

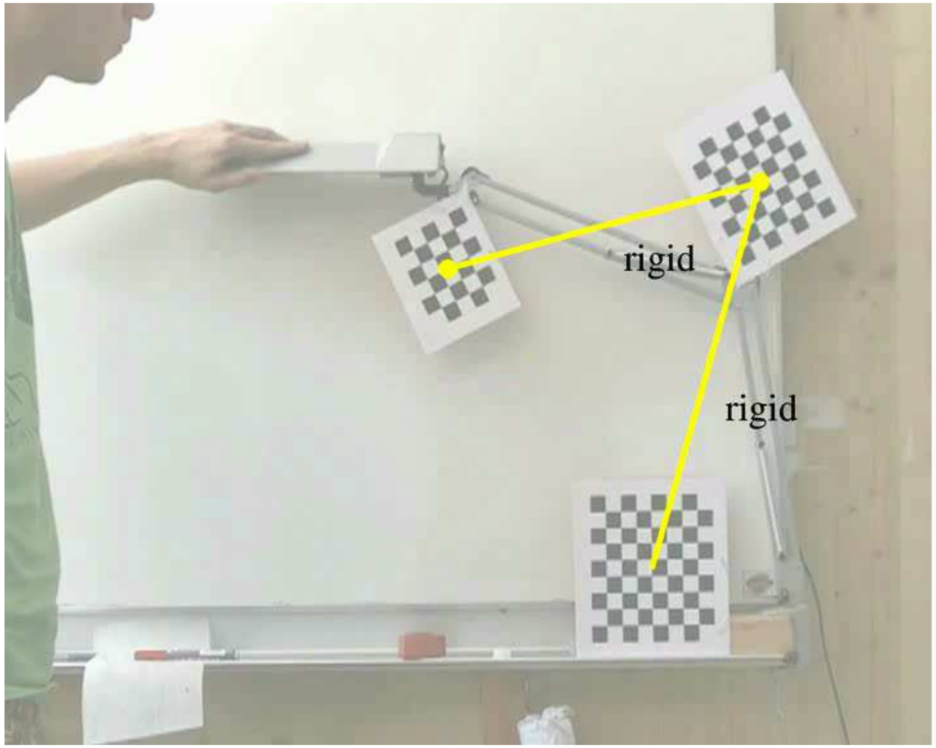


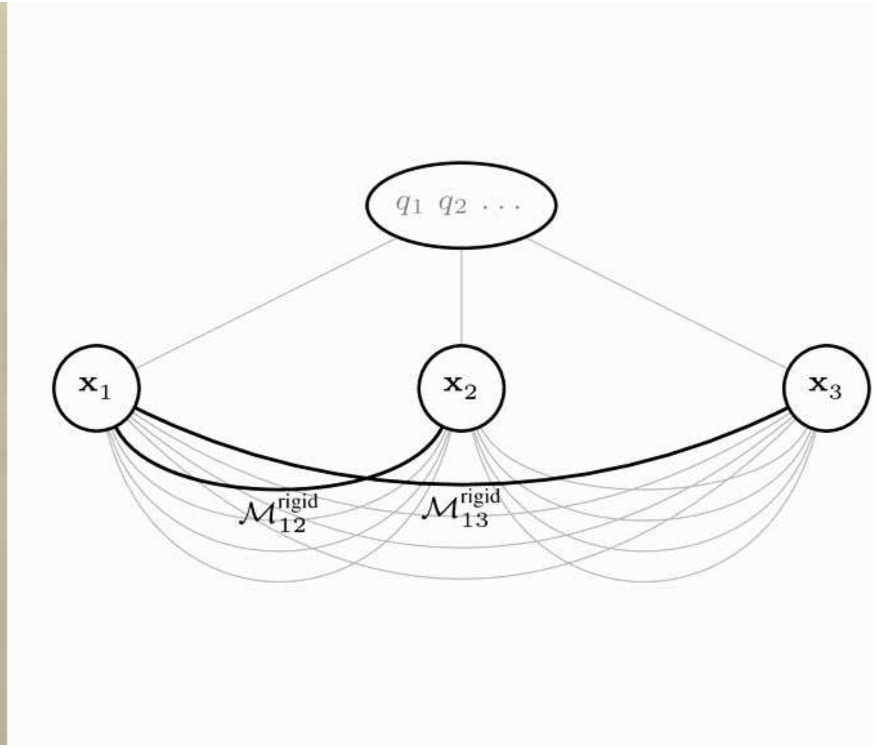


Example: Office Door



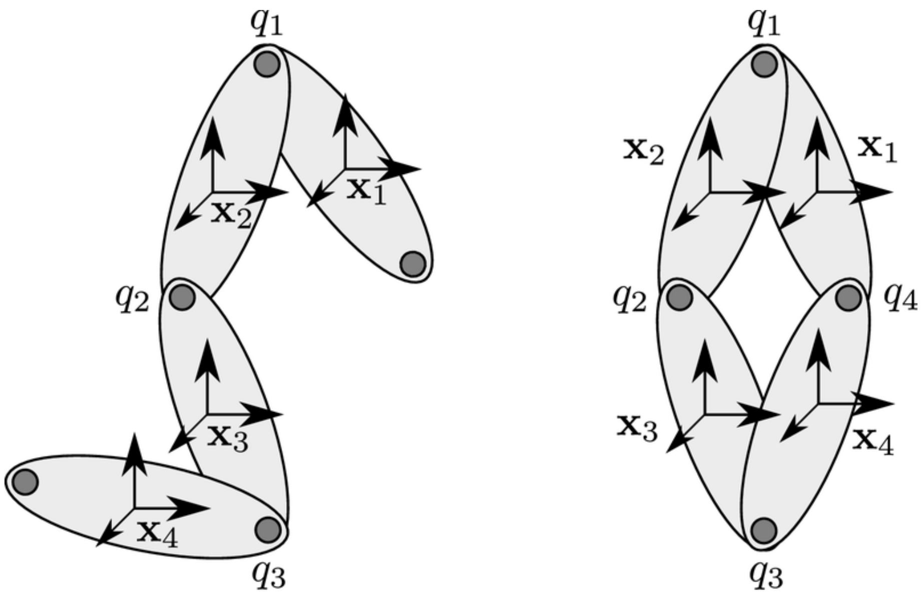
Example: Desk Lamp





Estimate effective DOFs

- Closed chain objects might have less DOFs than the sum of their links



3 links
3 DOF

4 links
1 DOF

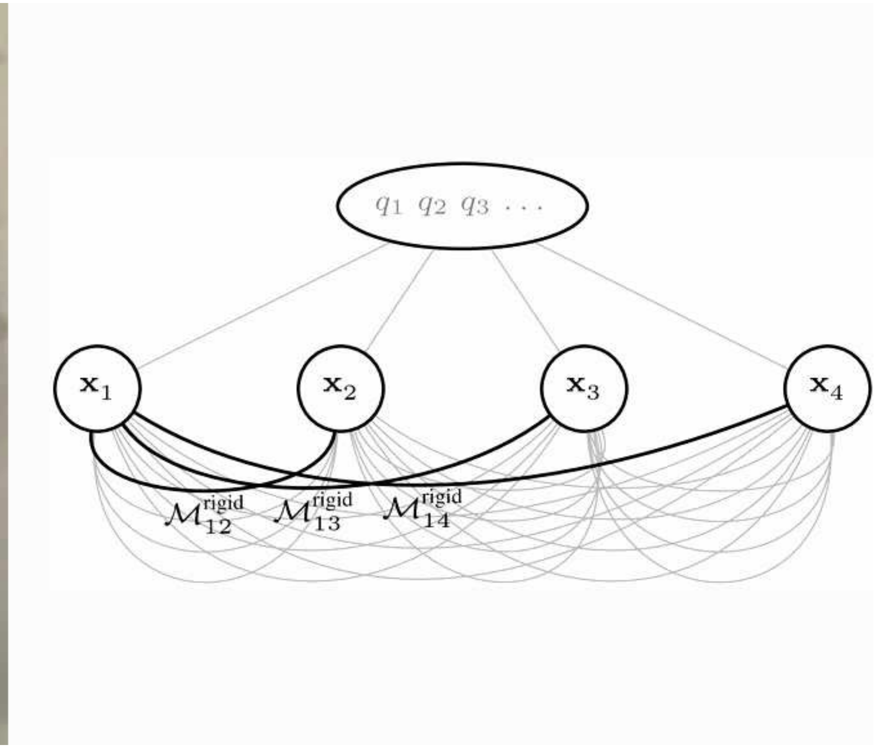
Estimate effective DOFs (2)

- Closed chain objects might have less DOFs than sum of their links
- Lower dimensional configuration space increases likelihood of a single configuration

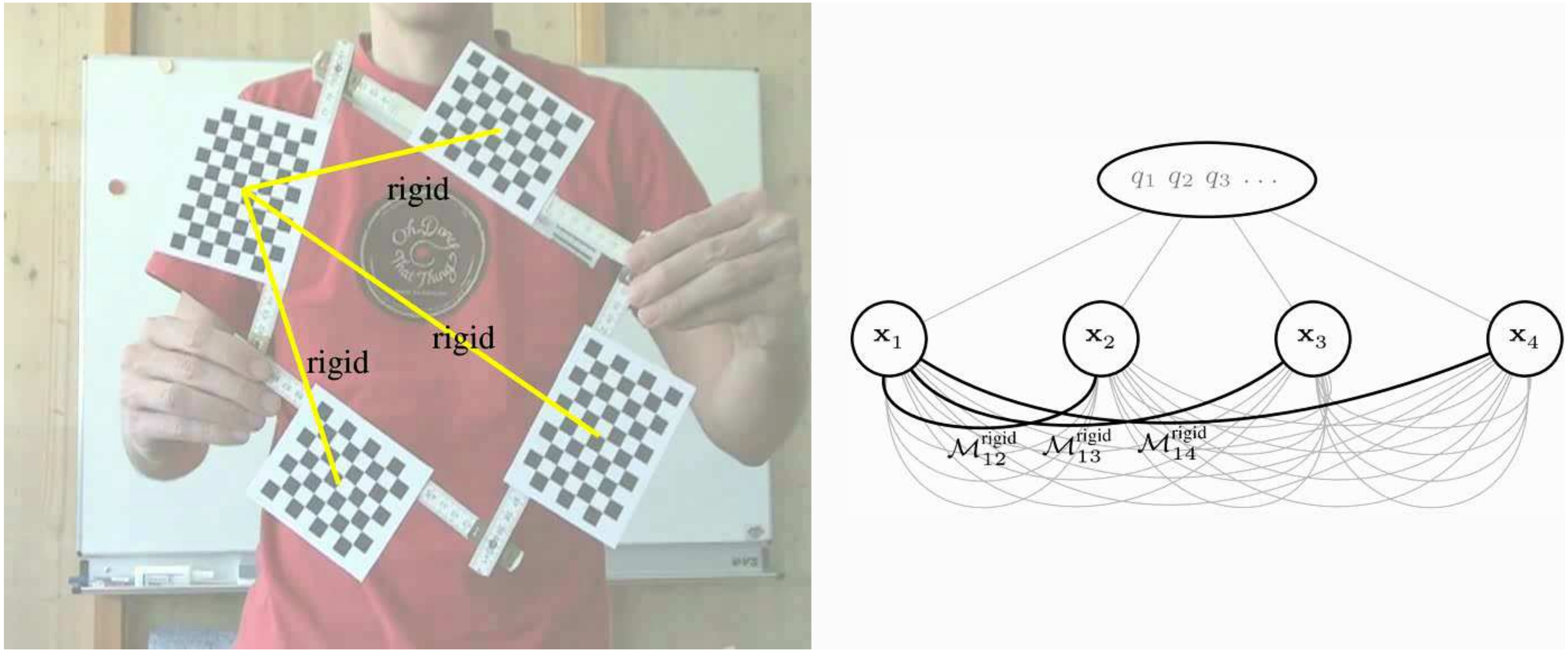
$$p(\mathbf{y} \mid \mathcal{M}, \theta) = \int p(\mathbf{y} \mid \mathbf{q}, \mathcal{M}, \theta) p(\mathbf{q}) d\mathbf{q}$$

➡ Additionally optimize number of DOFs during structure selection

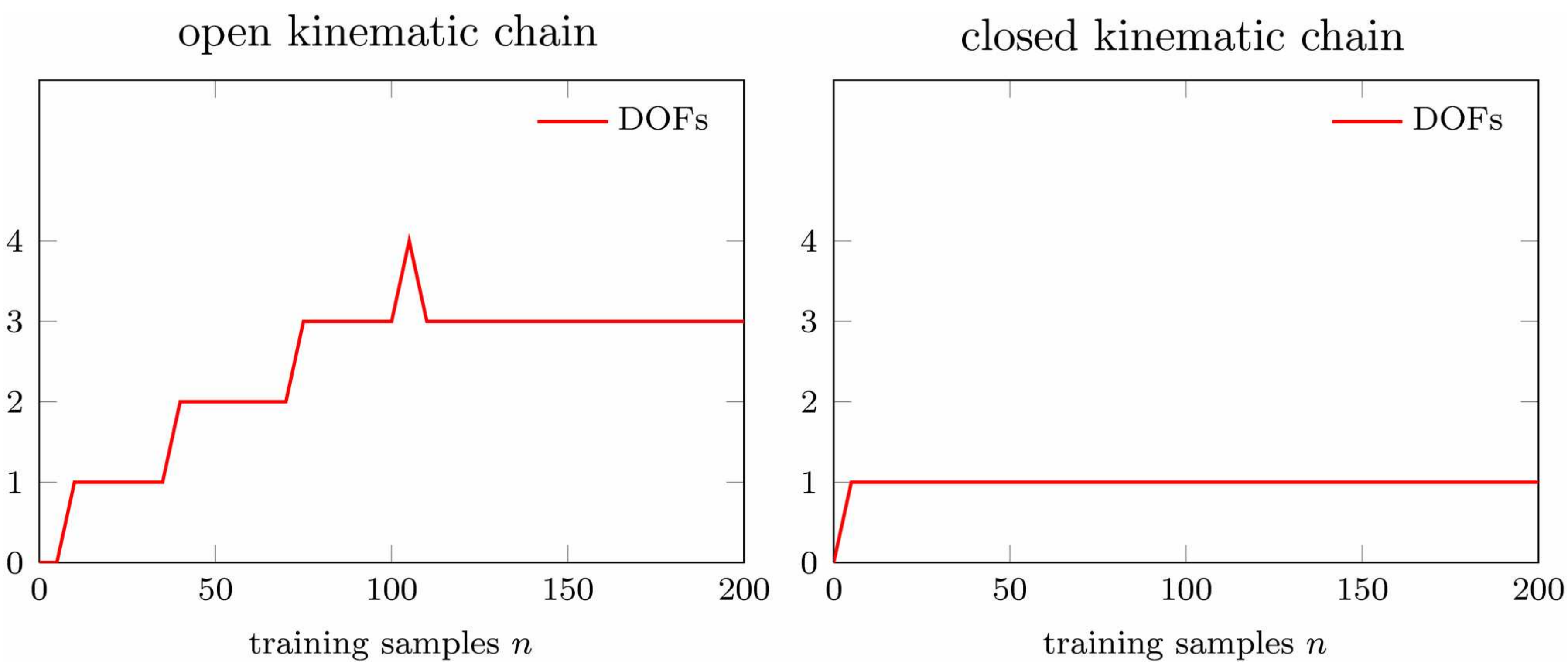
Example: Open Kinematic Chain



Example: Closed Kinematic Chain



Evaluation of DOFs



Articulated Objects in ROS

- Stacks and Packages
- Messages and Services
- Nodes
- Useful Scripts
- Tutorials and Demos

➡ **`http://www.ros.org/wiki/articulation`**

The `articulation` Stack

- Packages in the **`articulation`** Stack:
  - **`articulation_msg`**
  - **`articulation_models`**
  - **`articulation_rviz_plugin`**
  - **`articulation_structure`**
  - **`articulation_tutorials`**

Observation Sequence: `TrackMsg`

- Generic message for observed track
- Track identification number
- Observed poses $\mathcal{D}_{\mathbf{z}} = (\mathbf{z}^1, \ldots, \mathbf{z}^T)$
- Additional information (configuration q, ..)

```
articulation_msgs/TrackMsg.msg:

Header header                           # Timestamp and frame

int32 id                                # user-specified track id

geometry_msgs/Pose[] pose               # observed trajectory
geometry_msgs/Pose[] pose_projected     # projected trajectory
geometry_msgs/Pose[] pose_resampled     # re-sampled trajectory (for visualization)
sensor_msgs/ChannelFloat32[] channels   # additional information
```

Kinematic Model: `ModelMsg`

- Generic message for kinematic models
- Observation sequence $\mathcal{D}_{\mathbf{y}} = (\mathbf{y}_1^{1:T}, \ldots, \mathbf{y}_n^{1:T})$
- Model class $\hat{\mathcal{M}}$
- Model parameters $\hat{\theta}$

articulation_msgs/ModelMsg.msg:

```
Header header                          # frame and timestamp

int32 id                               # user specified model id
string name                            # name of the model class (e.g. "rotational",
                                       # "prismatic", "pca_gp", "rigid")
articulation_msgs/TrackMsg track       # data trajectory underlying the model
articulation_msgs/ParamMsg[] params    # model parameters
```

Kinematic Parameters: `ParamMsg`

- Generic message for parameters
- Type (prior, estimated, posterior)
- Name (e.g., “sigma_position”, “rot_radius”)
- Value (e.g., 0.01, 0.50,..)

```
articulation_msgs/ParamMsg.msg:

uint8 PRIOR=0   # indicates a prior model parameter
                # (e.g., "sigma_position")
uint8 PARAM=1   # indicates a estimated model parameter
                # (e.g., "rot_radius", the estimated radius)
uint8 EVAL=2    # indicates a cached evaluation of the model, given
                # the current trajectory
                # (e.g., "loglikelihood", the log likelihood of the
                # data, given the model and its parameters)

string name     # name of the parameter
float64 value   # value of the parameter
uint8 type      # type of the parameter (PRIOR, PARAM, EVAL)
```

Kinematic Object: `ArticulatedObjectMessage`

- Generic message for articulated objects
- Multiple parts
- Multiple articulated links

articulation_msgs/ParamMsg.msg:

```
Header header                        # frame and timestamp

articulation_msgs/TrackMsg[] parts    # observed trajectories for each object part
articulation_msgs/ParamMsg[] params   # global parameters
articulation_msgs/ModelMsg[] models   # models, describing relationships between parts
visualization_msgs/MarkerArray markers # marker visualization of models/object
```

Message Processing

- Articulated Link: **`model_learner_msg`**
  - Subscribes to: **`/track`** (queue size 1)
  - Publishes: **`/model`**
  - Parameters:
    - sigma_position (in meter)
    - sigma_orientation (in radians)
    - filter_models ("rigid prismatic rotational pca_gp")
- What does it do?
  - Fits model parameters
  - Estimates latent configurations of observations
  - Projects observations on model
  - Computes data likelihood and BIC score
  - Selects the best model

Services (1)

- Articulated Link: **`model_learner_srv`**
  - Services:
    - **`model_fit`**
    - **`model_select`**
    - **`model_eval`**
  - Parameters:
    - sigma_position (in meter)
    - sigma_orientation (in radians)
    - filter_models (“rigid prismatic rotational pca_gp”)
- What does it do?
  - Same as **`model_learner_msg`**: fits models, estimates configurations, evaluates data likelihood, computes BIC score, selects best model

Services (2)

- Articulated Object: **`structure_learner`**
  - Services:
    - **`fit_models`**
    - **`get_spanning_tree`**
    - **`get_fast_graph`**
    - **`get_graph`**
  - Parameters:
    - **`sigma_position`** (in meter)
    - **`sigma_orientation`** (in radians)
    - **`filter_models`** (“rigid prismatic rotational pca_gp”)
- What does it do?
  - Fits models to all possible links, estimates configurations, computes data likelihood, estimates DOFs, selects best kinematic graph

Visualizing Data Trajectories

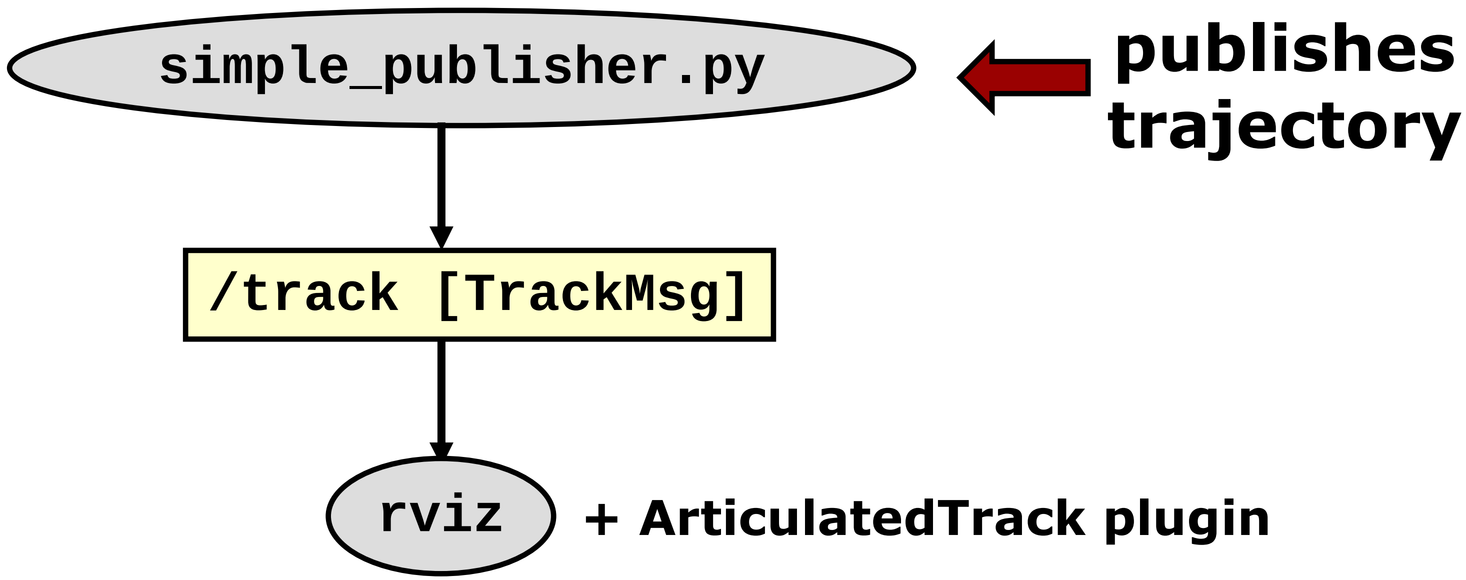
- **roslaunch articulation_tutorials visualize_tracks.launch**

articulation_tutorials/demo_fitting/data/second_set/left_door/003.log:

```
0.340249373856 -0.244008978915 -0.161821700704
0.3392470804 -0.243254309122 -0.161196313185
0.332328278289 -0.241196125961 -0.162826339162
0.331730517233 -0.240507339642 -0.162059956061
0.331761001802 -0.240391004141 -0.162215317578
0.326387758676 -0.237534821253 -0.162894338193
0.326692999754 -0.237495774032 -0.162320263447
0.326458151573 -0.236229292153 -0.161988473368
0.32643300294 -0.235474401346 -0.16226527964
0.326458151573 -0.236229292153 -0.161988473368
0.322475144873 -0.234904628909 -0.163001371167
0.322396297379 -0.234578781317 -0.162827883268
0.322338438997 -0.233729046115 -0.163104731452
0.31716772013 -0.233590696194 -0.162613189246
0.317140380085 -0.233046575927 -0.162728855923
0.317360184512 -0.232950213241 -0.162209399286
0.317291194395 -0.231950928639 -0.162542156939
0.31231370597 -0.229828694249 -0.162634424792
0.312256194198 -0.228424279769 -0.162708107274
0.312015344514 -0.227107330115 -0.162428803724
0.306629576897 -0.226190024443 -0.162693417499
0.307101002477 -0.225363586474 -0.163391447816
0.307314399481 -0.224601172119 -0.16312582421
0.307924291011 -0.223965969122 -0.162834221149
0.302440458046 -0.223043961032 -0.163117545449
0.303022295196 -0.220621222734 -0.162497012325
0.299320666256 -0.220486293623 -0.162602818856
0.299403314865 -0.219964271221 -0.163190275044
0.299065053516 -0.219226704563 -0.163054516178
[...]
```

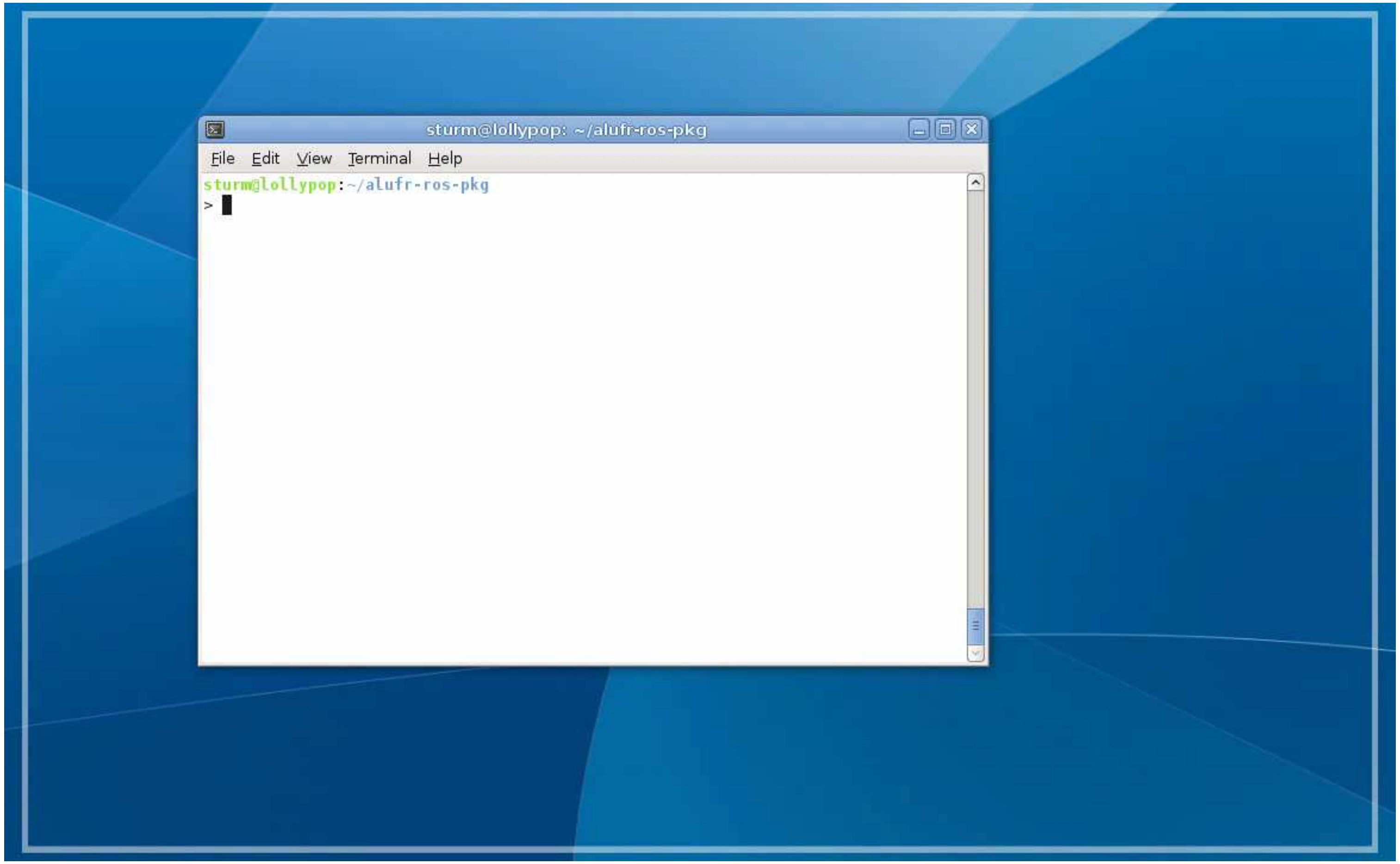
http://www.ros.org/wiki/articulation_tutorials/Tutorials/
Getting started with Articulation Models

Visualizing Data Trajectories



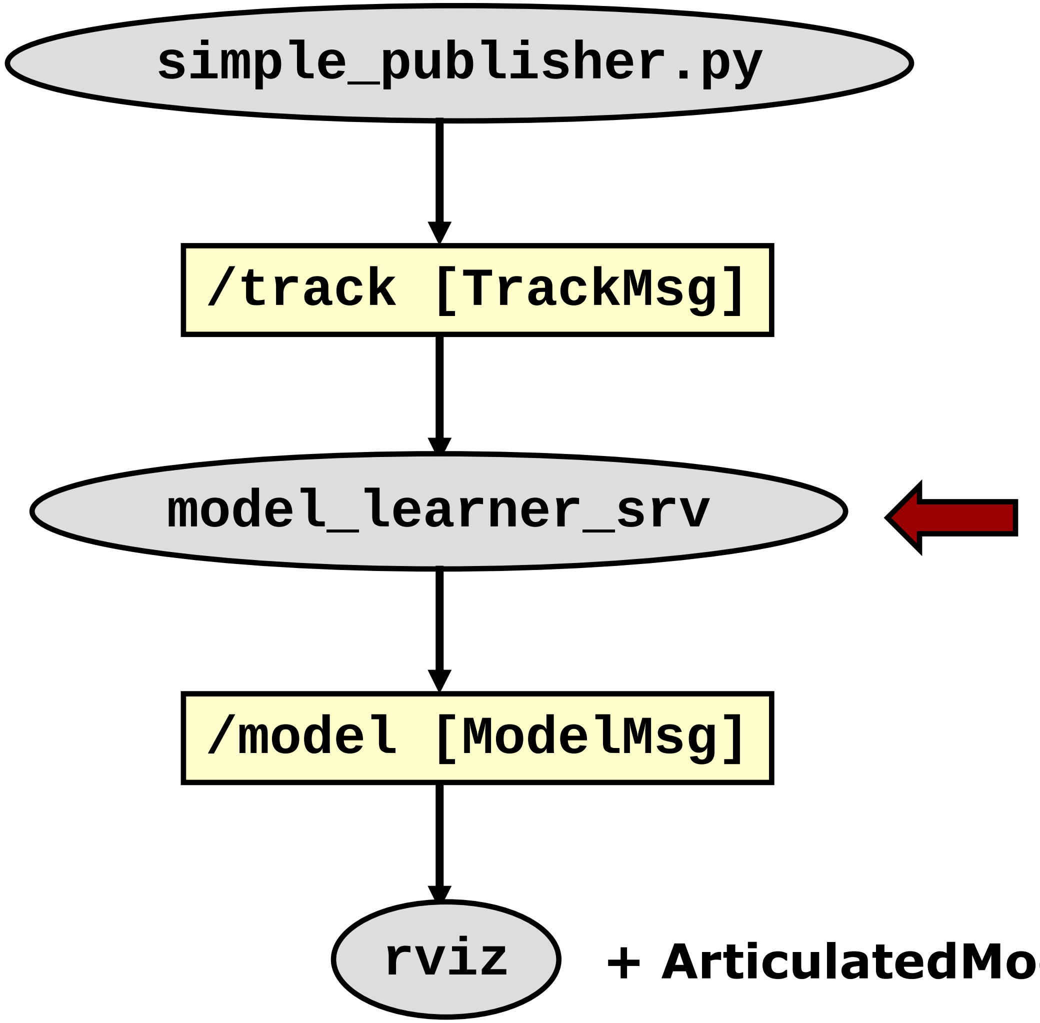
Visualizing Data Trajectories

- `roslaunch articulation_tutorials visualize_tracks.launch`



http://www.ros.org/wiki/articulation_tutorials/Tutorials/
Getting started with Articulation Models

Learning Models: Graph

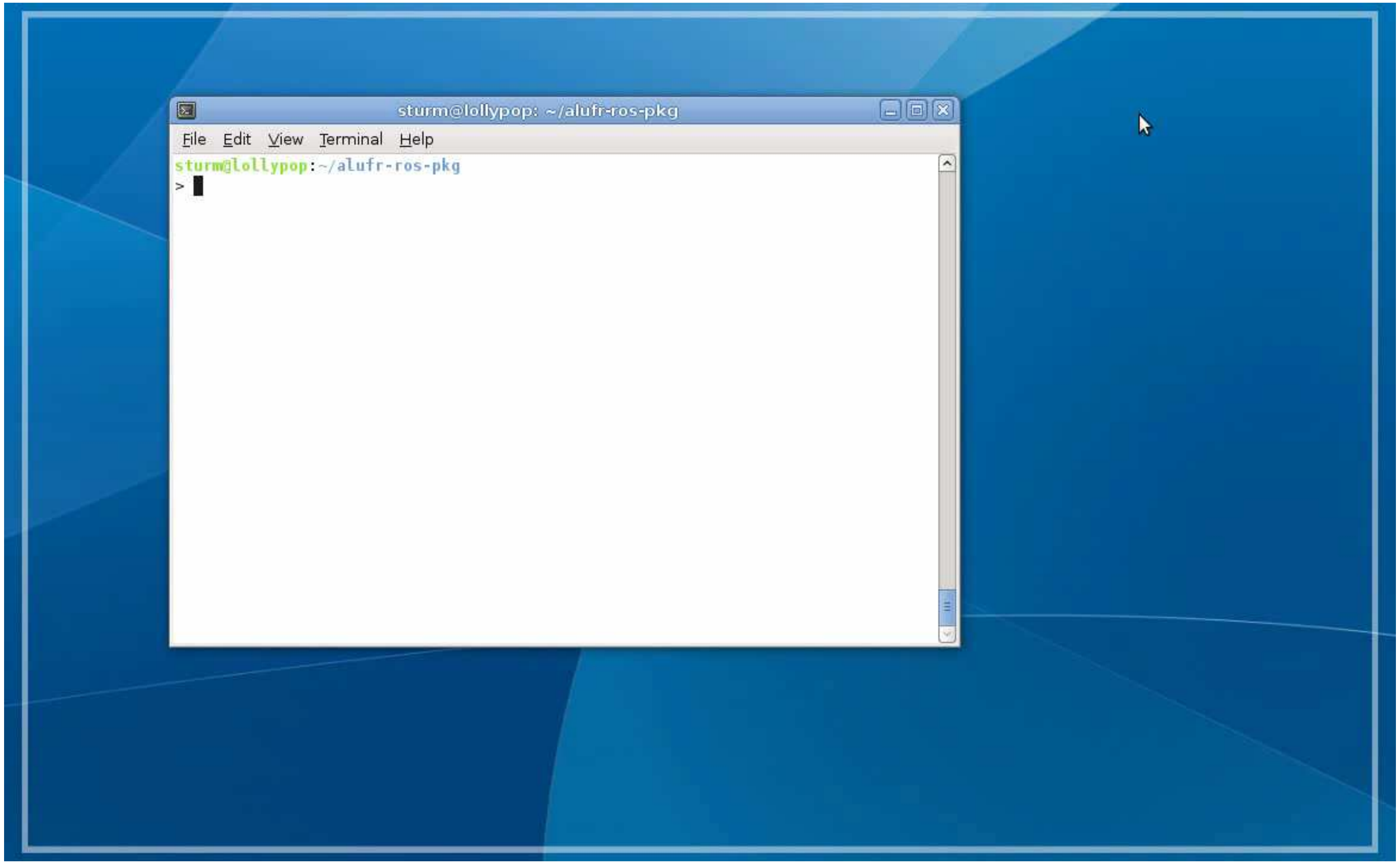


Fits all models and selects the best one

+ ArticulatedModel plugin

Learning Models: Video

- Using the **`articulation_rviz_plugin`**



http://www.ros.org/wiki/articulation_tutorials/Tutorials/ Getting started with Articulation Models

Learning Models: Launch File

```
<launch>
  <node pkg="articulation_models" type="simple_publisher.py" name="simple_publisher" output="screen“ args="
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/001.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/001.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/002.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/002.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/003.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/003.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/004.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/004.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/005.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/005.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/006.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/006.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/007.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/007.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/008.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/008.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/009.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/009.log
    $(find articulation_tutorials)/demo_fitting/data/drawer_one/010.log
    $(find articulation_tutorials)/demo_fitting/data/cabinet_one/010.log
    " >
  </node>

  <node pkg="articulation_models" type="model_learner_msg" name="model_learner" output="screen">
    <param name="filter_models" value="rotational prismatic"/>
    <param name="sigma_position" value="0.01"/>
    <param name="sigma_orientation" value="10.00"/>
  </node>

  <node pkg="rviz" type="rviz" output="screen" name="rviz" args="-d $(find
    articulation_tutorials)/demo_fitting/fit_models.vcg" />
</launch>
```

Simple publisher

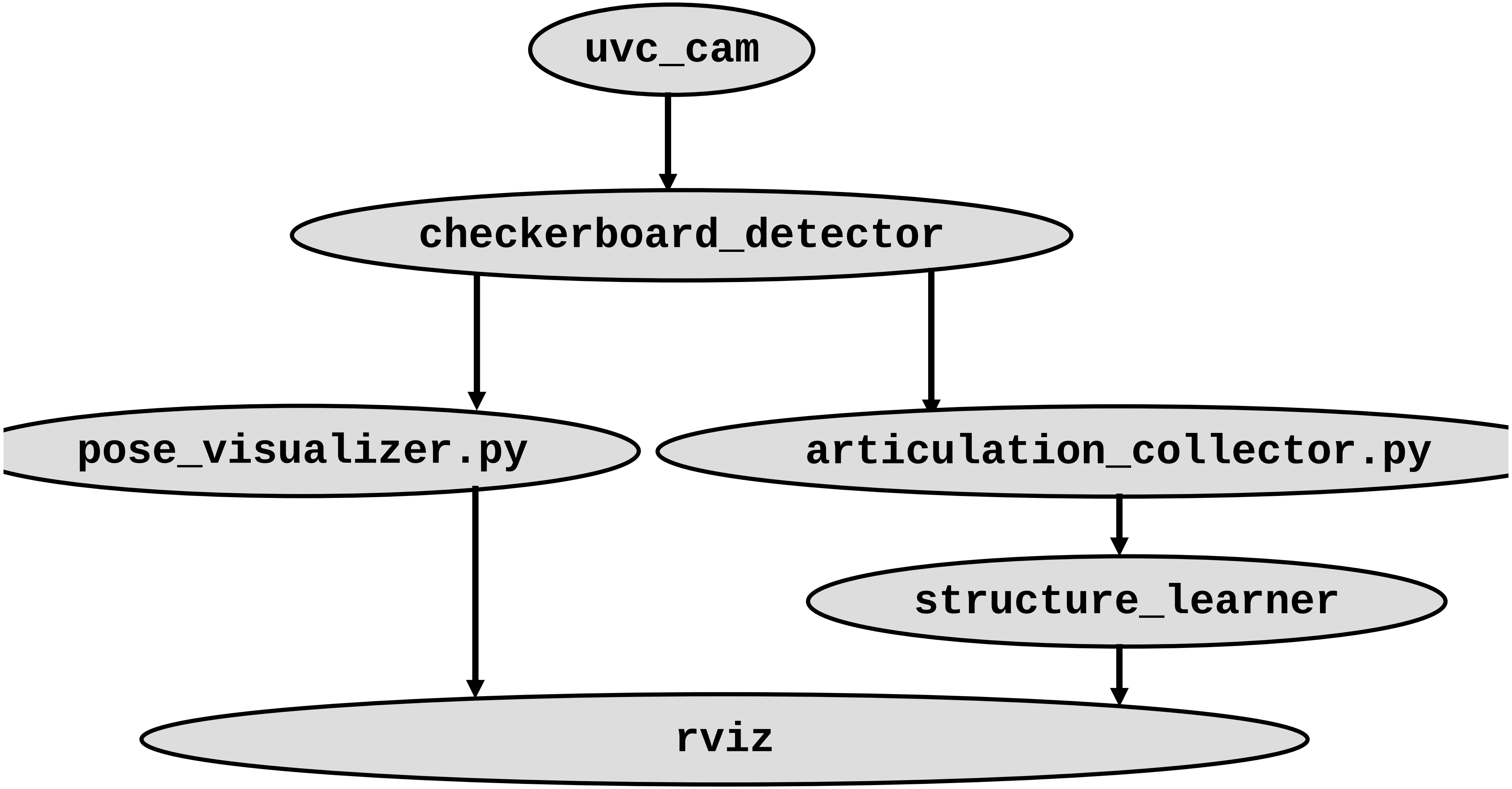
Model Learner

RVIZ

Many Interfaces

- Command-line
  - Via simple_publisher.py, process_bag.py, and others
  - Publishes trajectory from text or bag files, or directly from end-effector pose of PR2
- Python
  - Via subscriber/publisher
  - Via service calls
- C++
  - Direct library bindings
  - Fastest

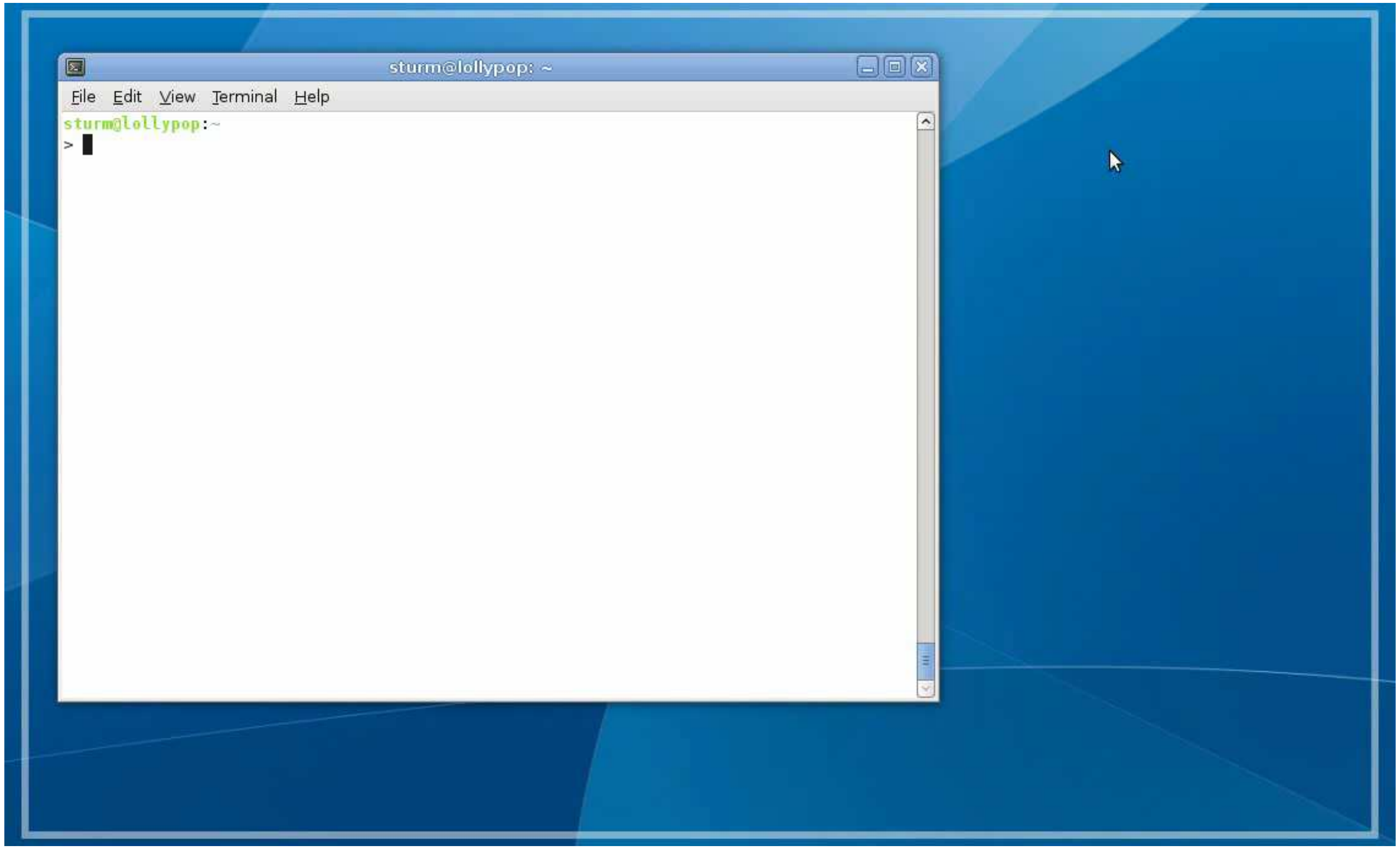
Learning Models using a Webcam



http://www.ros.org/wiki/articulation_tutorials/Tutorials/ArticulationWebcamDemo

Learning Models using a Webcam

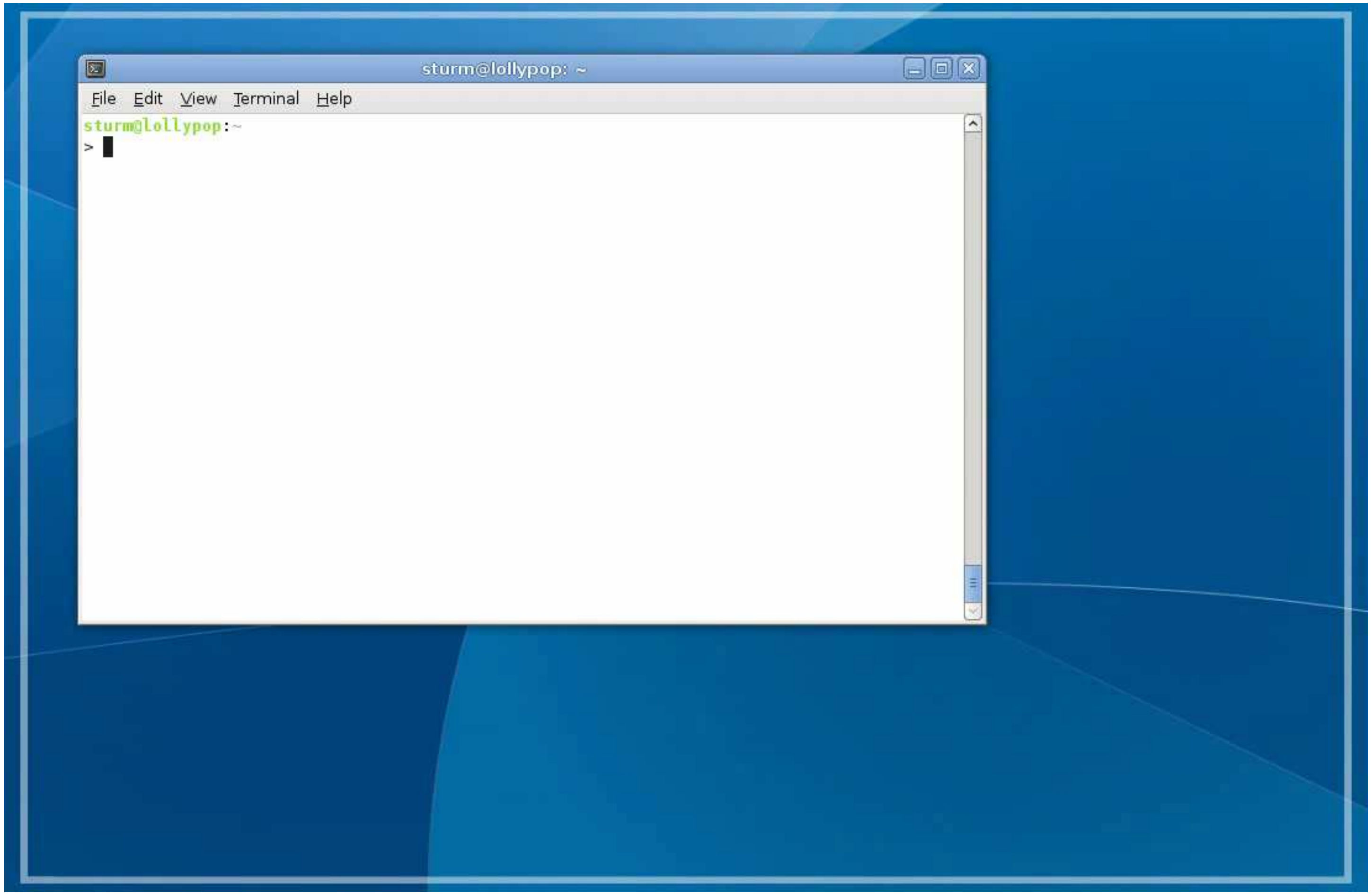
- Milka Chocolate Box



- Live demo after the talk!

Learning Models using a Webcam

- Leibniz Cookies



- Live demo after the talk!

Learning Models using a Webcam

```
articulation_tutorials/ webcam_demo/ webcam_demo-1cm-4x6-4x5.launch:

<launch>
  <node name="uvc_cam" pkg="uvc_cam2" type="sender" output="log">
    <param name="D" type="string" value="-0.0834232 0.120545 -0.0023918 0.0175383 0 "/>
    <param name="K" type="string" value="578.252 0 350.204 0 575.115 207.606 0 0 1 "/>
    <param name="R" type="string" value="1 0 0 0 1 0 0 0 1 "/>
    <param name="P" type="string" value="578.252 0 350.204 0 0 575.115 207.606 0 0 0 1 0 "/>
    <param name="device" type="string" value="/dev/video0"/>
    <param name="width" type="int" value="640"/>
    <param name="height" type="int" value="480"/>
    <param name="fps" type="int" value="2"/>
  </node>

  <node name="image_proc" pkg="image_proc" type="image_proc" output="log"/>

  <node name="pose_visualizer" pkg="checkerboard_detector2" type="pose_visualizer.py" output="screen"/>

  <node pkg="checkerboard_detector2" type="checkerboard_detector2"
    respawn="false" output="log" name="checkerboard_detector">
    <param name="display" type="int" value="0"/>


    <param name="rect0_size_x" type="double" value="0.01"/>
    <param name="rect0_size_y" type="double" value="0.01"/>
    <param name="grid0_size_x" type="int" value="4"/>
    <param name="grid0_size_y" type="int" value="6"/>

    <param name="rect1_size_x" type="double" value="0.01"/>
    <param name="rect1_size_y" type="double" value="0.01"/>
    <param name="grid1_size_x" type="int" value="4"/>
    <param name="grid1_size_y" type="int" value="5"/>
  </node>
  </group>
```

Webcam

Checkerboard Detector

Learning Models using a Webcam

```
[...]
<node name="articulation_collector" pkg="articulation_structure" type="articulation_collector.py"
    output="screen">
   <param name="samples" value="50"/>
 </node>

 <node name="structure_learner" pkg="articulation_structure" type="structure_learner_srv" output="screen">
   <param name="sigma_position" value="0.01"/>
   <param name="sigma_orientation" value="0.1"/>
   <param name="filter_models" value="rigid prismatic rotational"/>
 </node>

 <node pkg="rviz" type="rviz" output="screen" name="rviz" args="-d $(find
    articulation_tutorials)/webcam_demo/webcam_demo.vcg" />

</launch>
```

Pose Collector

Structure Learner

RVIZ

Conclusions

- Bayesian framework for learning kinematic model of articulated objects
  - Robust model fitting
  - Model comparison
  - Structure selection
  - Estimation of effective number of DOFs
- Stable code, open-source, BSD
- Fully integrated in ROS
  - Command-line
  - Python
  - C++

Future Work

- Add more model classes
- Integrate with handle detector
- Store learned articulation models in maps
- Learn force profiles

References

- J. Sturm, V. Pradeep, C. Stachniss, C. Plagemann, K. Konolige, & W. Burgard. (2009). Learning kinematic models for articulated objects. In *Proc. of the Int. Joint Conf. on Artificial Intelligence (IJCAI).*
- J. Sturm, K. Konolige, C. Stachniss, & W. Burgard. (2010). Vision-based detection for learning articulation models of cabinet doors and drawers in household environments. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA).*
- J. Sturm, A. Jain, C. Stachniss, C. Kemp, & W. Burgard. (2010). Operating articulated objects based on experience. In *Proc. of the IEEE Int. Conf. on Intelligent Robot Systems (IROS).*

Thank you!

- Any Questions?

➡ Live demo..