Trajectory Prediction: Learning to Map Situations to Robot Trajectories

Nikolay Jetchev and Marc Toussaint
TU Berlin
Berlin Machine Learning and Robotics Group

June 17th, 2009
ICML

Talk content

- Movement generation and motivation
- Define trajectory prediction
- Describe the prediction algorithm
- Experiments,results and conclusions

Biological Inspiration

- Humans and animals execute complex movements 'instantly'
- Don't plan a trajectory, but quickly choose a good movement
- Reactive trajectory policy
- How do people manage to do it?
  - Experience in repeating movements
  - Capabilities of our bodies (proprioception)
  - Familiar obstacles and spaces

Apply to robot movement planning

- Robots are slower to generate movement, optimize from scratch without prior knowledge
- A novel approach – **trajectory prediction**
- The essence: a mapping from a situation to a whole trajectory
  - Observe patterns in situation-movement pairs
  - Choose quickly an approximately good trajectory
- Gain for robotics: speed up movement generation by learning an approximate situation-movement mapping

Robot movement basics

- q_t is the robot posture, angles of all joints at time t
- The obstacle positions and q_t fully define the current situation
- Trajectory $\boldsymbol{q} = (q_0, .., q_T)$ for some time horizon T

Planning as cost function optimization

- A cost function characterizes good movements for the specified tasks
- Energy efficient trajectories with no collisions have low cost
- $C(x, \boldsymbol{q}) = \sum_{t=1}^{T} g(q_t) + h(q_t, q_{t\text{-}1})$
- A trajectory optimization algorithm finds the best movement for a given situation
- $x \mapsto \boldsymbol{q}^* = \operatorname*{argmin}_{\boldsymbol{q}} C(x, \boldsymbol{q})$

Trajectory prediction for faster optimization

- Approximate the mapping $x \mapsto \boldsymbol{q}^*$
- Gather data $D = \{(x_i, \boldsymbol{q}_i)_{i=1}^d\}$ of optimized movements
  - Use any classical method for movement optimization
- Use this data to find quickly good initial trajectories
  - Methods like iterated Linear Quadratic Gaussian(iLQG) and gradient descent very sensitive to initialization
  - Good starting values can improve them drastically

Prediction algorithm overview

- Train: gather data D and learn to predict
- Test input: situation x
  - Predict trajectory $\boldsymbol{q}_i$ from D
  - Transfer $\boldsymbol{q}_i$ to the current situation x
- Test output: transferred trajectory $\boldsymbol{q}^*$

Situation descriptor

- The world situation is specified by robot posture and object positions
- Model situation representation x as a high-dimensional feature vector $x = (q_0, p, o) \in \Re^{791}$:
  - q_0 is initial posture vector
  - p is vector of pairwise distances btw centers of 20 objects
  - o is vector of z axis cosine between 20 objects in scene

Feature selection

- Why these features?
  - Much information about world situation
  - Highly redundant, a lot of coordinate systems and measurements
- Feature selection can refine the descriptor
  - sparse 50 features

Feature	w_i^2
$p^y_{waist-table}$	20.7
$p^x_{chest-table}$	17.3
$p^x_{target-table}$	16.8
$p^y_{back-table}$	14.2
$p^z_{neck-target}$	11.4
$p^z_{wristR-target}$	8.9

$$P(f(x) = \mathcal{T}_{x_i x}\boldsymbol{q}_i) = \frac{1}{Z}\exp\{-\frac{1}{2}(x - x_i)^T W (x - x_i)\}$$

$$\mathrm{E}\{C(x, f(x))\} = \sum_{i=1}^{d} P(f(x) = \mathcal{T}_{x_i x}\boldsymbol{q}_i) C(x, \mathcal{T}_{x_i x}\boldsymbol{q}_i)$$

$$\min_w \sum_{x \in D} \mathrm{E}\{C(x, f(x))\} + \lambda |w|_1$$

NN for prediction

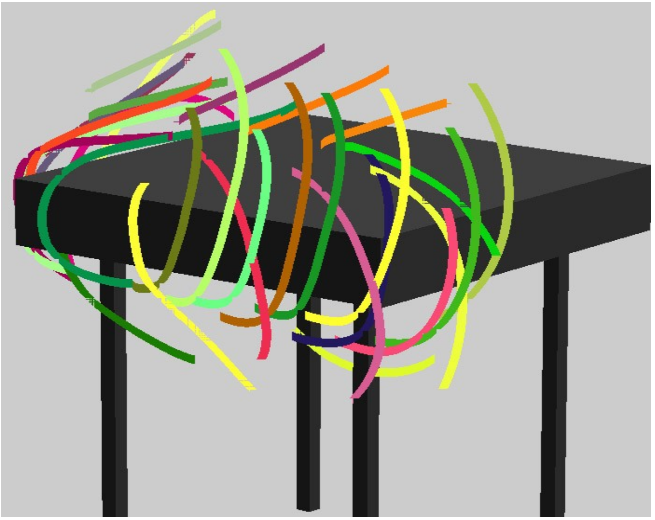
- Nearest neighbor over database situations – NN
  - $k(x^*, x_i) = \exp\{-\frac{1}{2}(x^* - x_i)^\top W (x^* - x_i)\}$ as a diagonal Gaussian similarity metric
  - $\hat{i} = \operatorname{argmax}_i k(x_i, x)$
- The predicted movement is $q_{\hat{i}}$, the trajectory of the most similar previous situation

Classification for trajectory prediction

- Classify situations according to movement type – Cluster
  - Select a smaller representative movement subset by clustering D
  - Take cluster average movements as new trajectory set
  - Gather dataset with lowest cost prototypes for each movement
  - Train a SVM on this data

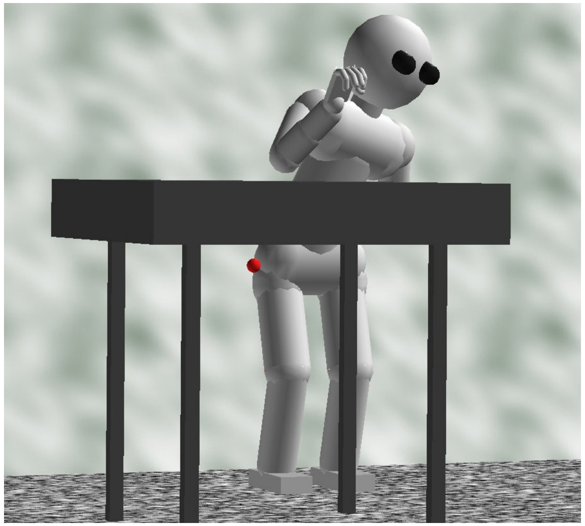
Situation transfer

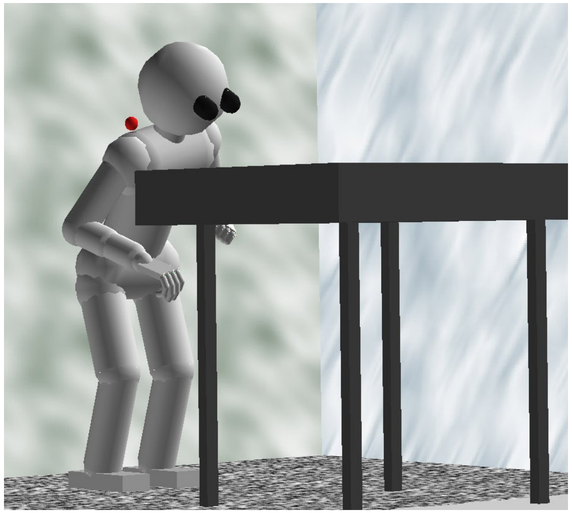
- Repeating a movement in joint space is not likely to be good
  - Different object positions between situations
- We need to transfer to the new situation x
  - Task space -coordinates of finger relative to obstacle
  - Project from joint space to task space and then back project via inverse kinematics (IK)
  - Prioritized IK avoid collisions
  - Call this the transfer operator $T_{x_i x}$



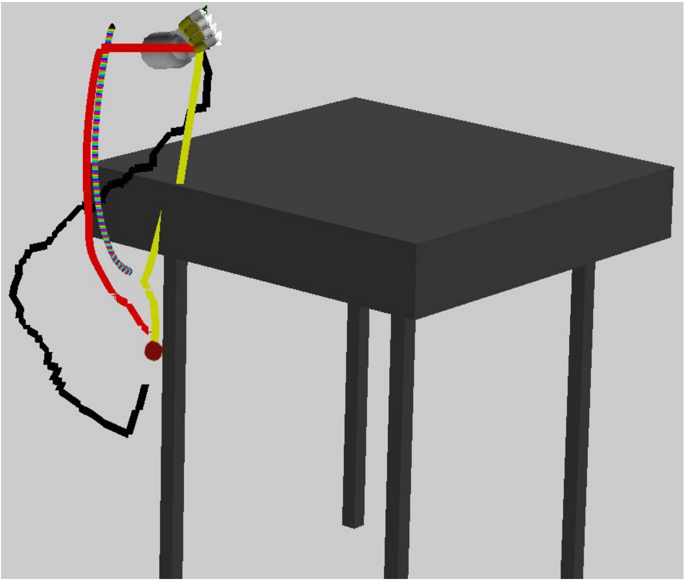
Experiments

- Use in simulation a humanoid torso with 31 joints
- Reach red point target with finger without colliding with the table
- Generate world scenarios by randomly sampling robot posture, target and obstacle positions





Compared methods



- iLQG with different initializations
  - NN prediction
  - Cluster prediction
  - Linear interpolated path
  - Rapidly Exploring Random Tree (RRT) path
- Good initial trajectory will speed-up iLQG convergence
  - It corresponds to a reasonable table avoidance path
- A single iLQG iteration: 0.065 s, prediction + transfer – 0.1s, RRT – more than 1.5s for 2000 nodes

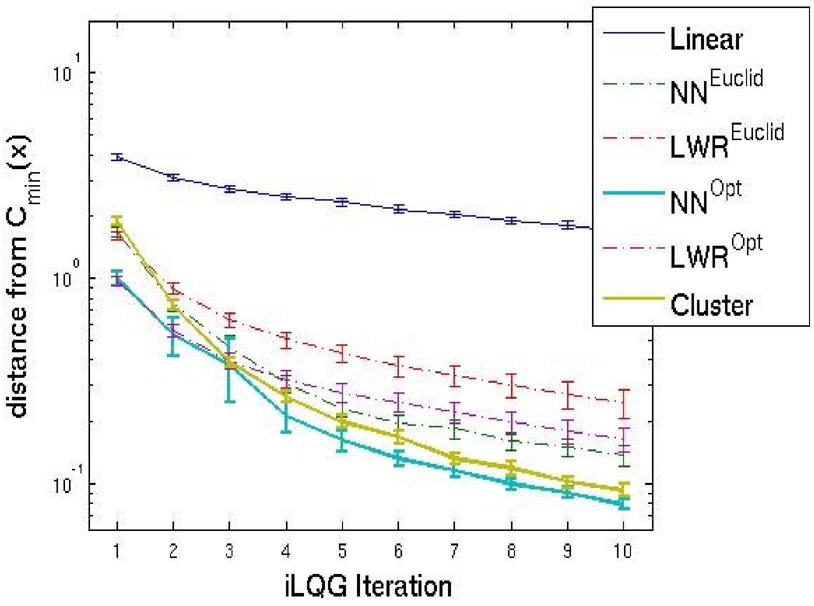
Results: time until convergence

- Cluster converges in 98.4% of scenarios to correct results (linear 85%)
- NN converges much faster
  - 0.32s for first feasible solution (linear 1.3s)
  - 0.9s for convergence (linear 1.96s)
- Sparse feature selection improves results

Method		$\epsilon = 0.2$	$\epsilon = 0.15$	$\epsilon = 0.1$	$\epsilon = 0.05$
Linear	#	31	41	63	155
	μ	1.26 ± 0.04	1.29 ± 0.05	1.4 ± 0.05	1.96 ± 0.07
NN^{Euclid}	#	4	6	9	52
	μ	0.4 ± 0.01	0.47 ± 0.01	0.62 ± 0.02	1.14 ± 0.04
LWR^{Euclid}	#	4	9	19	58
	μ	0.39 ± 0.01	0.45 ± 0.01	0.59 ± 0.02	1.05 ± 0.04
NN^{Opt}	#	2	3	4	33
	μ	$\mathbf{0.32} \pm 0.01$	$\mathbf{0.36} \pm 0.01$	0.47 ± 0.01	0.83 ± 0.02
LWR^{Opt}	#	3	4	16	49
	μ	$\mathbf{0.31} \pm 0.01$	$\mathbf{0.35} \pm 0.01$	$\mathbf{0.45} \pm 0.01$	$\mathbf{0.76} \pm 0.03$
Cluster	#	**1**	**1**	**2**	**16**
	μ	0.39 ± 0.01	0.45 ± 0.01	0.57 ± 0.01	1.02 ± 0.03

Results: iLQG iteration analysis

- Another view: cost convergence per iLQG iteration
- The prediction methods achieve very low costs in few iterations
- RRT not competitive: much slower than linear



Conclusions and future work

- Trajectory prediction can speed-up computation drastically
- Data representation should be carefully designed to transfer knowledge
  - Information in situation descriptor and transfer task space
- Future directions: more challenging movement problems
  - Dynamic worlds
  - Cluttered scenes
  - More complex manipulations - grasping

The End

- Thank you for your time.
- More info at http://user.cs.tu-berlin.de/~jetchev/