Incremental Learning of Robot Dynamics using Random Features

Arjan Gijsberts, Giorgio Metta

Cognitive Humanoids Laboratory
Dept. of Robotics, Brain and Cognitive Sciences
Italian Institute of Technology

Cogsys
Cognitive Systems

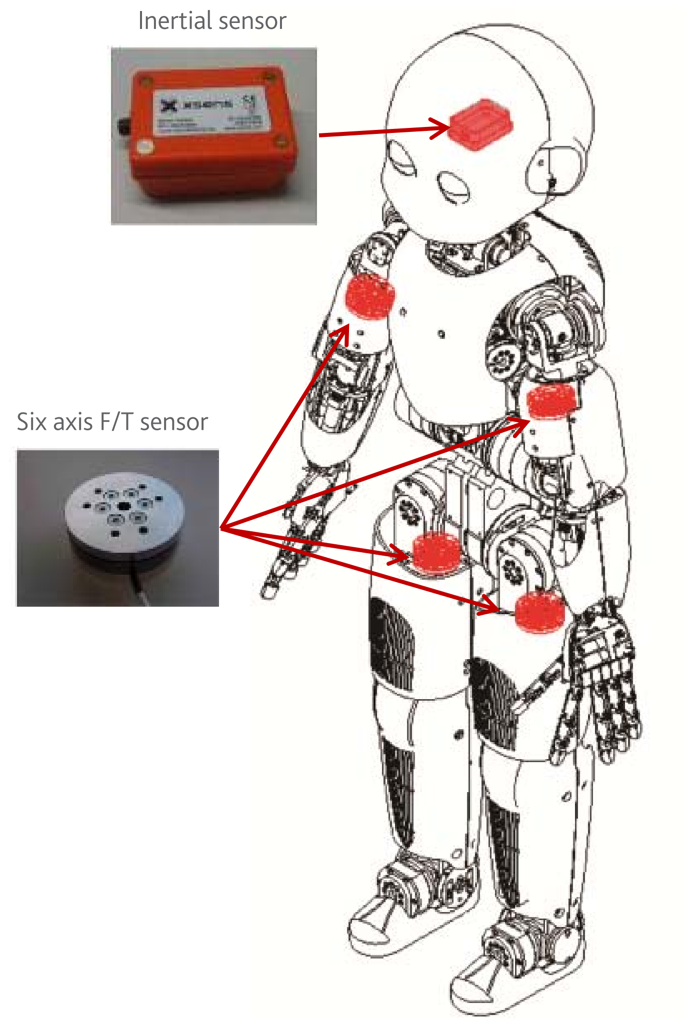
RobotCub

general setting

- learning incrementally
  - because the world is non-stationary (concept drift)
- learn efficiently
  - real-time (hard) constraints
- we'd like to learn
  - accurately (guarantees that learning learns)
  - autonomously (little prior programming)

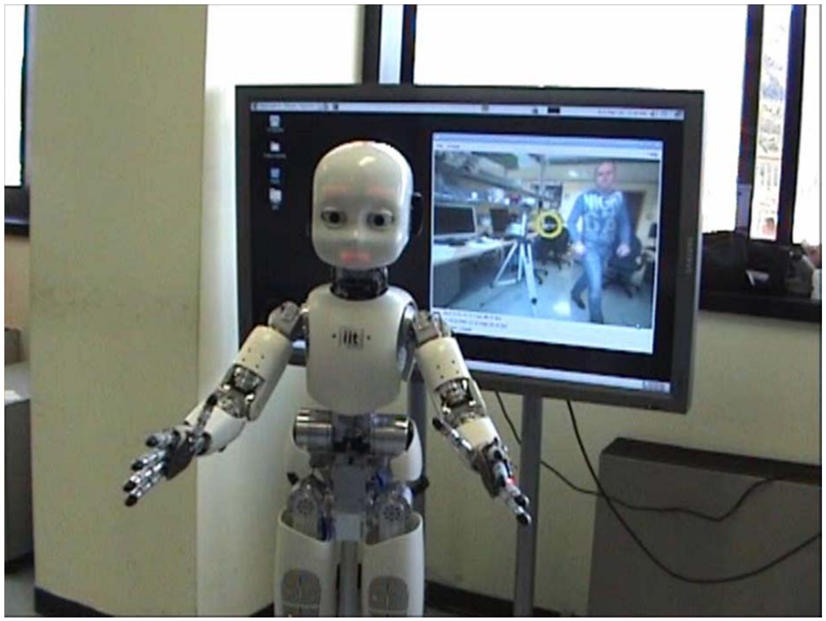
specific setting

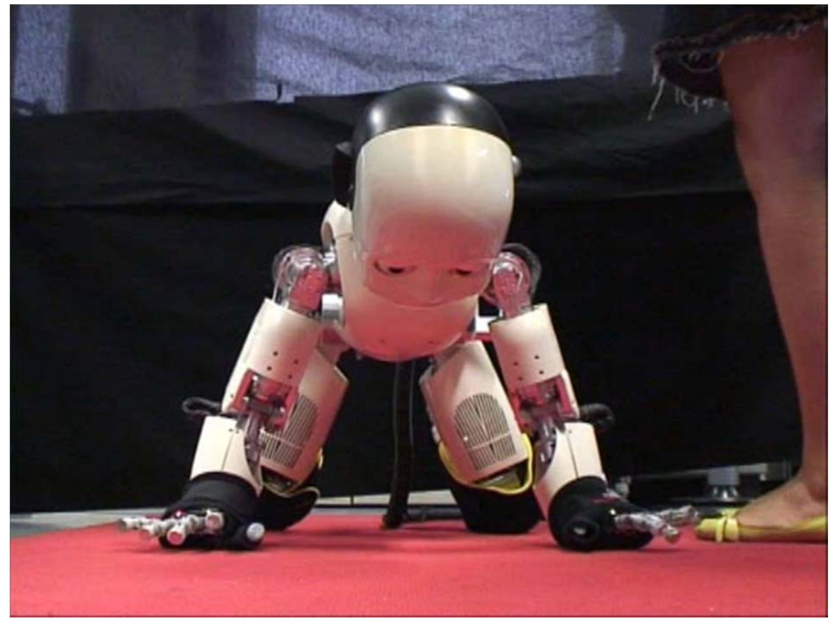
- learning body dynamics
  - compute external forces
  - implement compliant control



- so far we did it starting from e.g. the cad models
  - but we'd like to avoid it

...so





some incremental learning methods

- LWPR [Vijayakumar et al., 2005]
- Kernel Recursive Least Squares [Engel et al., 2004]
- Local Gaussian Processes [Nguyen-Tuong et al., 2009]
- Sparse Online GPR [Csató and Opper, 2002]

typical problems (not everywhere):

- high per-sample complexity (slow learning)
- increasing or unpredictable computational requirements
- limited theoretical support and understanding

our method

- linear ridge regression as base algorithm
  - efficient, elegant, effective
  - theoretically well-studied

- possible extensions for non-linear regression and incremental updates

$$f(x) = w^T x$$

$$\min_w J = \frac{\lambda}{2}\|w\|^2 + \frac{1}{2}\|y - Xw\|^2$$

$$w = \left(\lambda I + X^T X\right)^{-1} X^T y$$

our method in 3 easy steps

- kernel trick

$$f(x) = \sum_{i=1}^{m} c_i k(x, x_i)$$

$$c = (K + \lambda I)^{-1} y$$

- approximate kernel

Rahimi, A. & Recht, B. (2008)

$$k(x_i, x_j) = E\left[\frac{1}{D}\sum_{d=1}^{D} z_{w_d}(x_i)^T z_{w_d}(x_j)\right]$$

$$z_w(x) = \left[\cos(w^T x), \sin(w^T x)\right]$$

- make it incremental

$$w = (\lambda I + \Phi^T \Phi)^{-1} \Phi^T y$$

\+ Cholesky rank-1 update

features

- $O(1)$ update complexity w.r.t. # training samples
- exact batch solution after each update
- dimensionality of feature mapping trades computation for approximation accuracy
- $O(n^2)$ time and space complexity per update w.r.t. dimensionality of feature mapping
- easy to understand/implement (few lines of code)
- not exclusively for dynamics/robotics learning!

Algorithm 4.1 Incremental RFRLS with isotropic RBF kernel

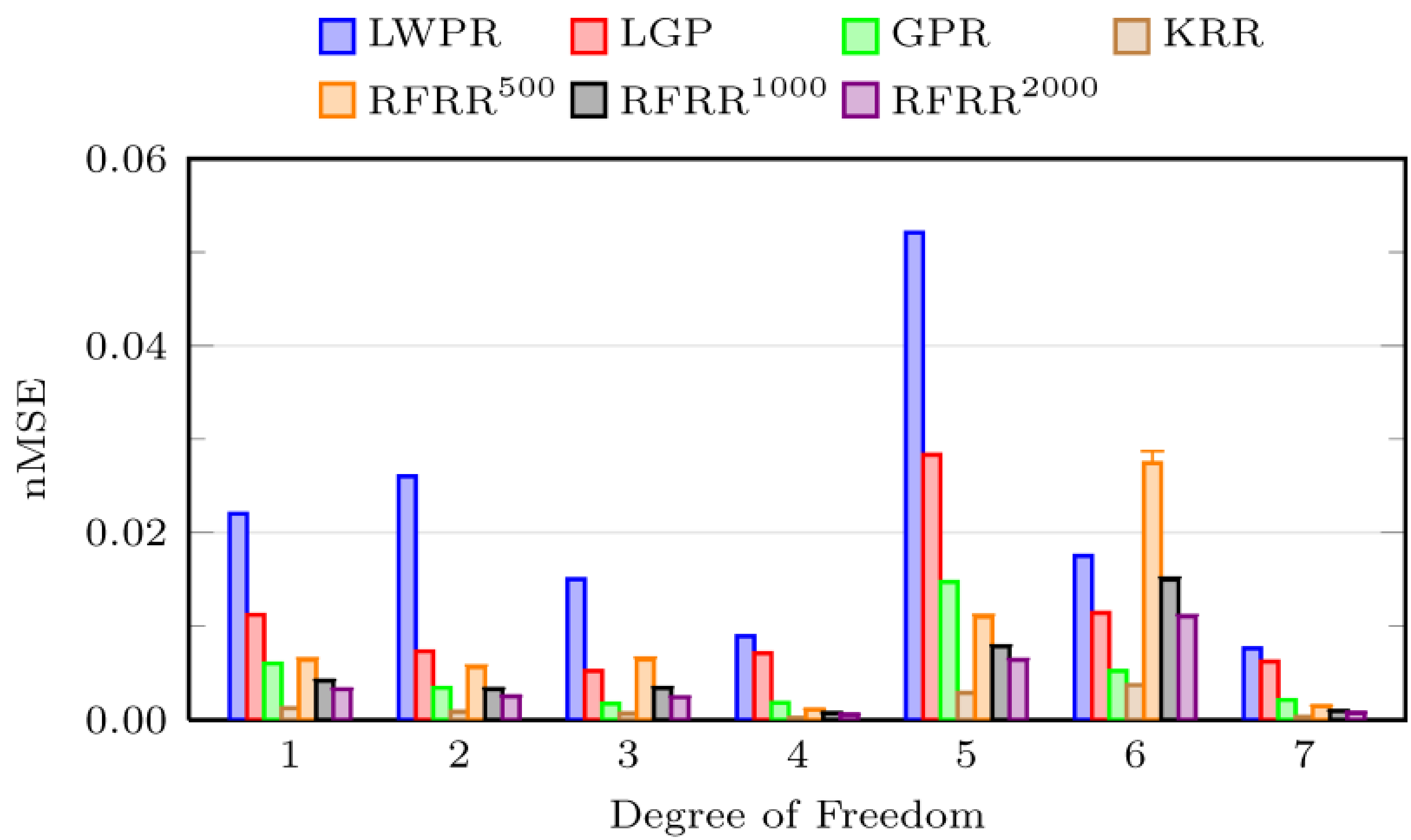
Require: $\lambda > 0, \gamma > 0, n > 0, D > 0$

1: $\boldsymbol{R} \leftarrow \sqrt{\lambda}\boldsymbol{I}_{D\times D}$
2: $\boldsymbol{w} \leftarrow \boldsymbol{0}_{D\times 1}$
3: $\boldsymbol{b} \leftarrow \boldsymbol{0}_{D\times 1}$
4: $\boldsymbol{\Omega} \sim \mathcal{N}\left(0, (2\gamma)^2\right)_{D\times n}$
5: $\boldsymbol{\beta} \sim \mathcal{U}(-\pi, \pi)_{D\times 1}$
6: **for all** $(\boldsymbol{x}, y)$ **do**
7: $\quad \boldsymbol{\phi} \leftarrow \sqrt{\frac{2}{D}} \cos\left(\boldsymbol{\Omega x} + \boldsymbol{\beta}\right)$ // *Equation* (4.4)
8: $\quad \hat{y} \leftarrow \langle \boldsymbol{w}, \boldsymbol{\phi} \rangle$
9: $\quad \boldsymbol{b} \leftarrow \boldsymbol{b} + \boldsymbol{\phi} y$
10: $\quad \boldsymbol{R} \leftarrow \textsc{CholeskyUpdate}(\boldsymbol{R}, \boldsymbol{\phi})$
11: $\quad \boldsymbol{w} \leftarrow \boldsymbol{R} \backslash \left(\boldsymbol{R}^{\mathrm{T}} \backslash \boldsymbol{b}\right)$
12: $\quad$ **yield** $\hat{y}$
13: **end for**

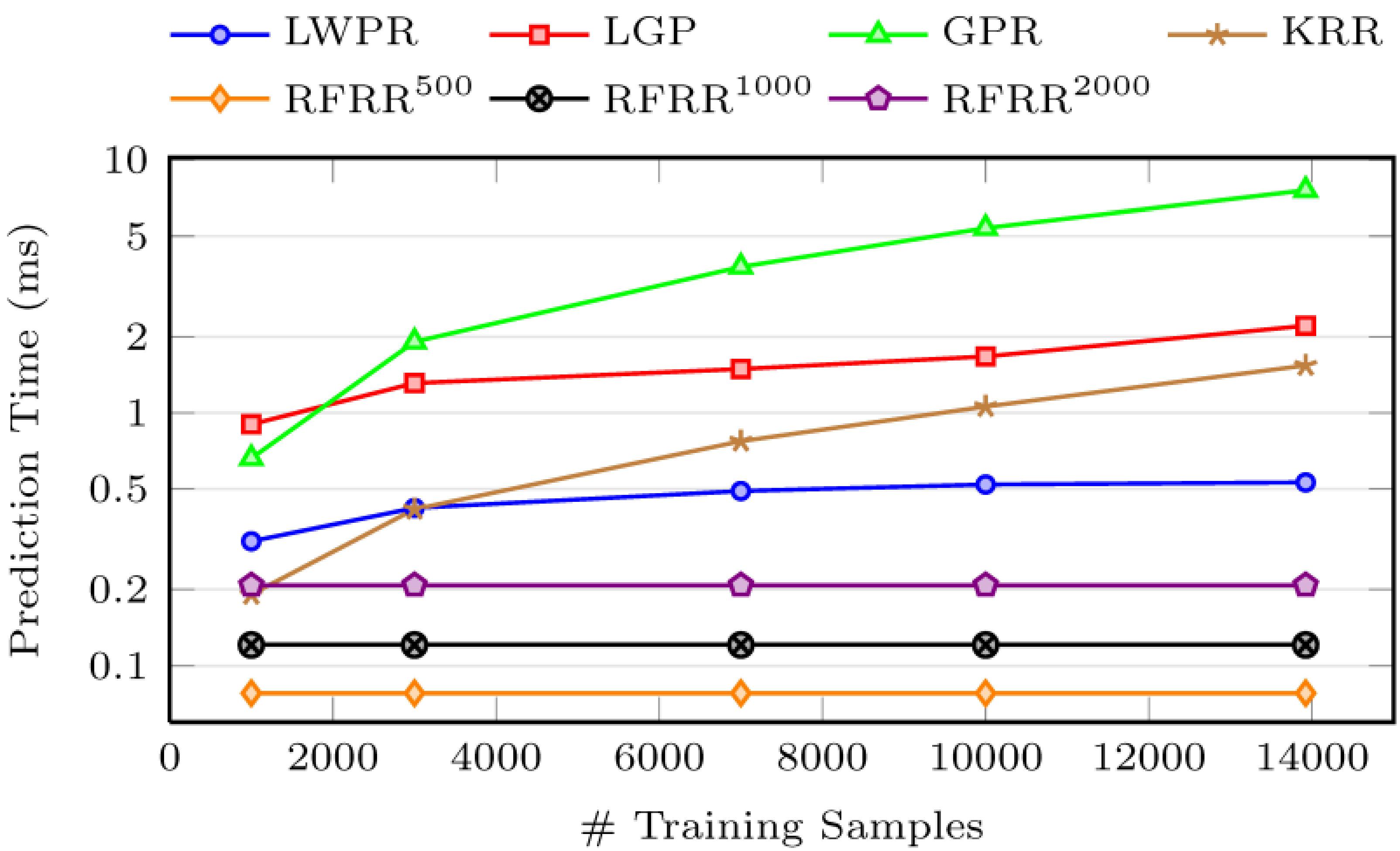
batch experiments

- 3 inverse dynamics datasets: Sarcos, Simulated Sarcos, Barrett [Nguyen-Tuong et al., 2009]
- approximately 15k training and 5k test samples
- comparison with LWPR, GPR, LGP, Kernel RR
- RFRR with 500, 1000, 2000 random features
- hyperparameter optimization by exploiting functional similarity with GPR (log marginal likelihood optimization)

batch error on 7-DOF Sarcos arm



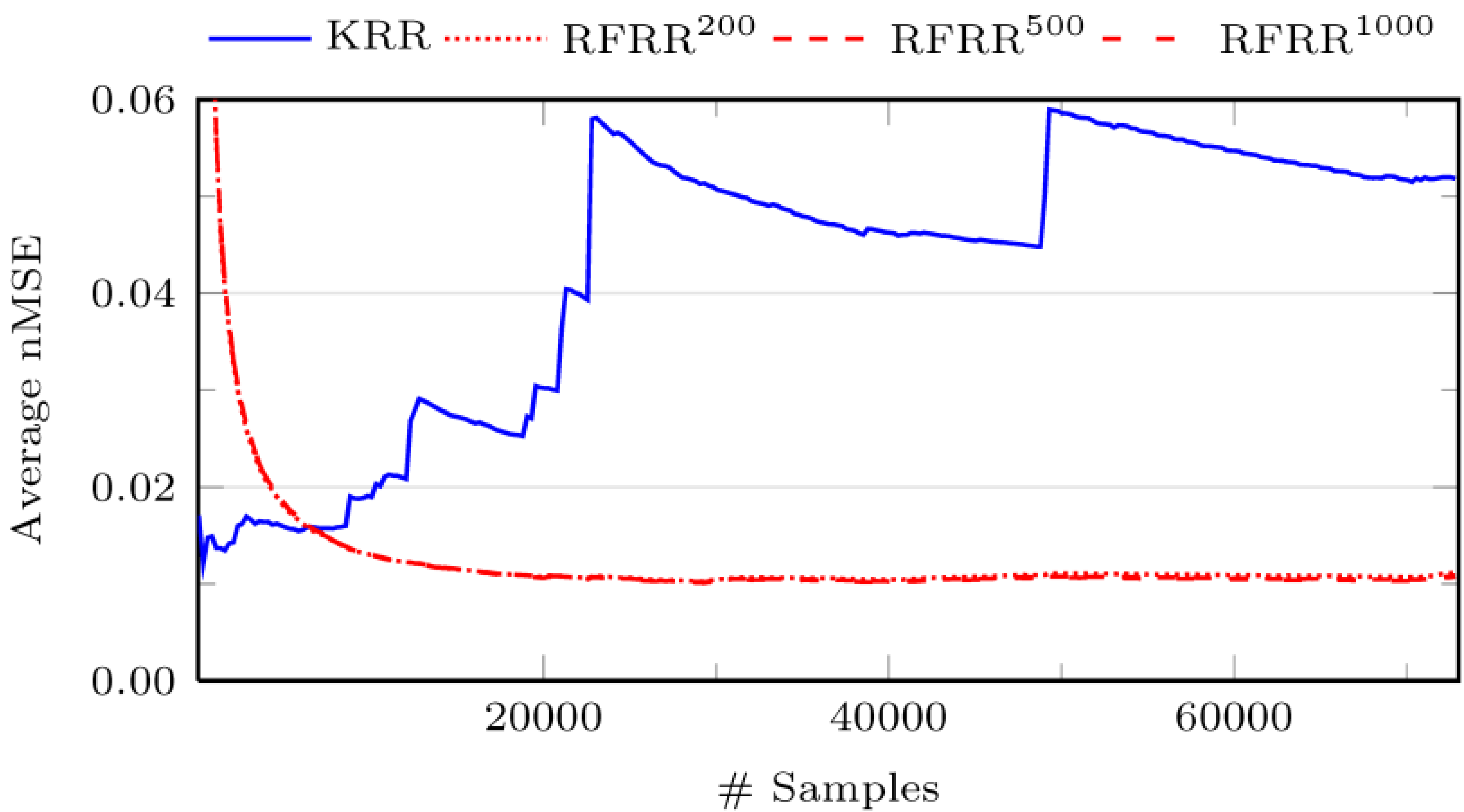
prediction time



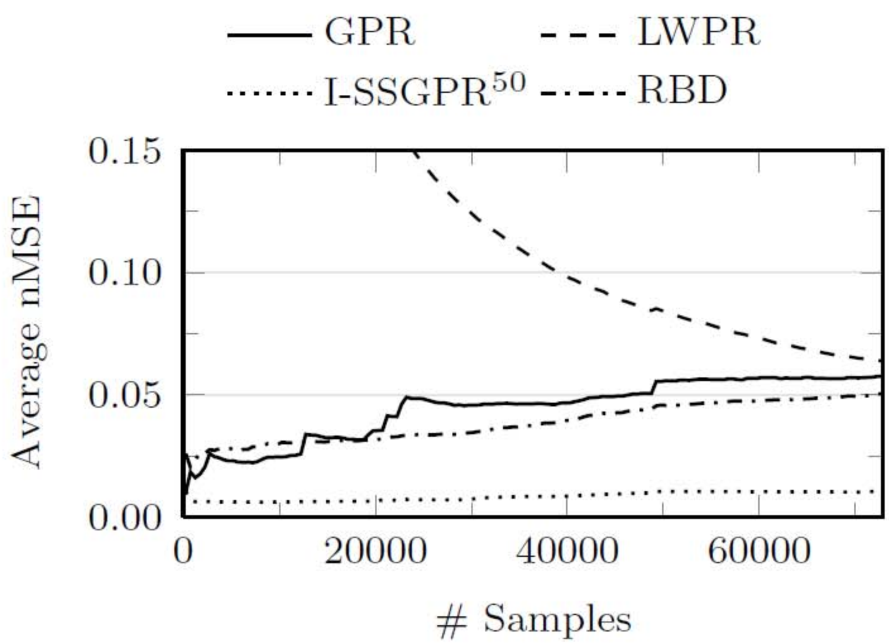
incremental experiments

- two large scale inverse dynamics datasets from "James" and iCub humanoids (4-DOF)
- realistic scenario: initial 15k training and remaining approx. 200k and 80k test samples
- RFRR with 200, 500, 1000 random features
- RFRR uses training samples only for hyperparameter optimization
- comparison with batch Kernel RR (identical hyperparameters)

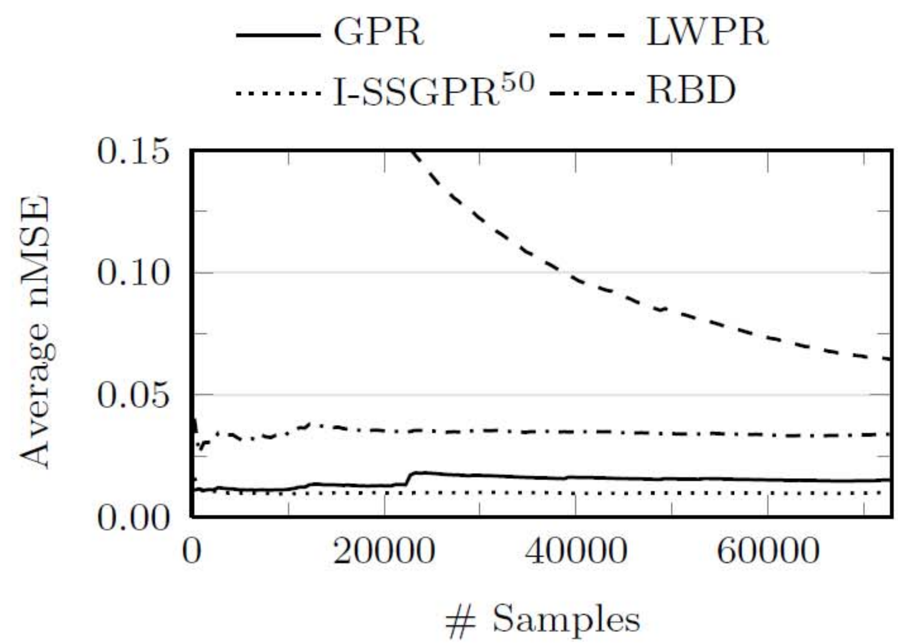
batch vs. incremental



verification (learning dynamics)

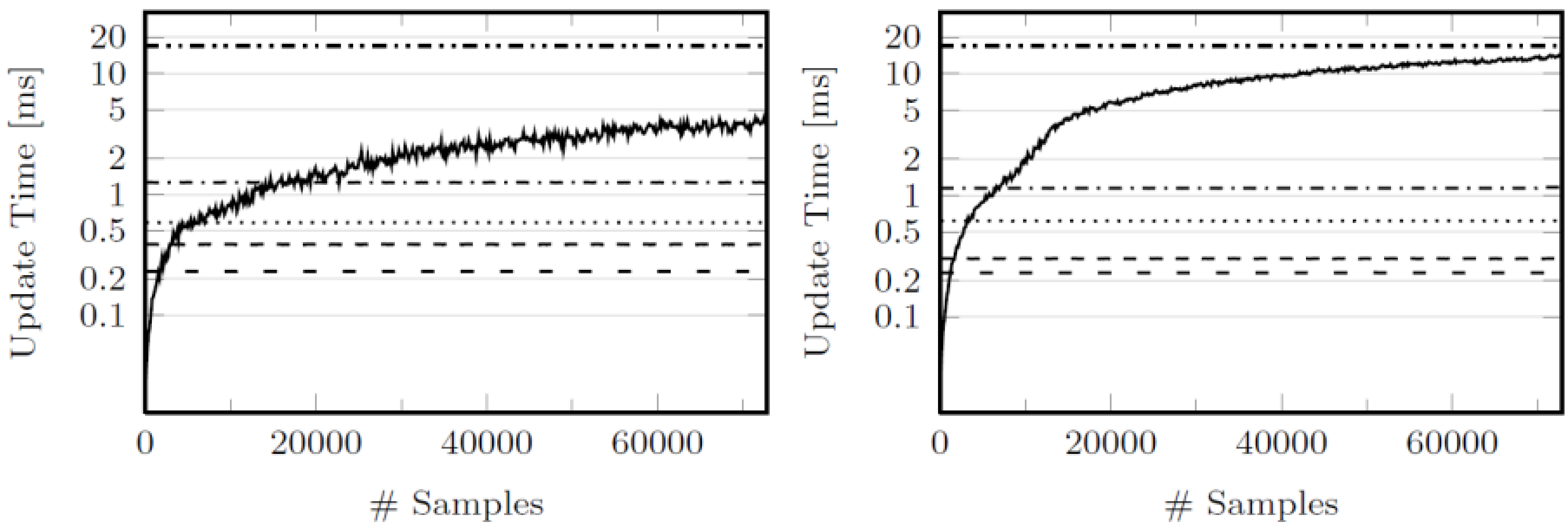


(d) iCub Forces



(e) iCub Torques

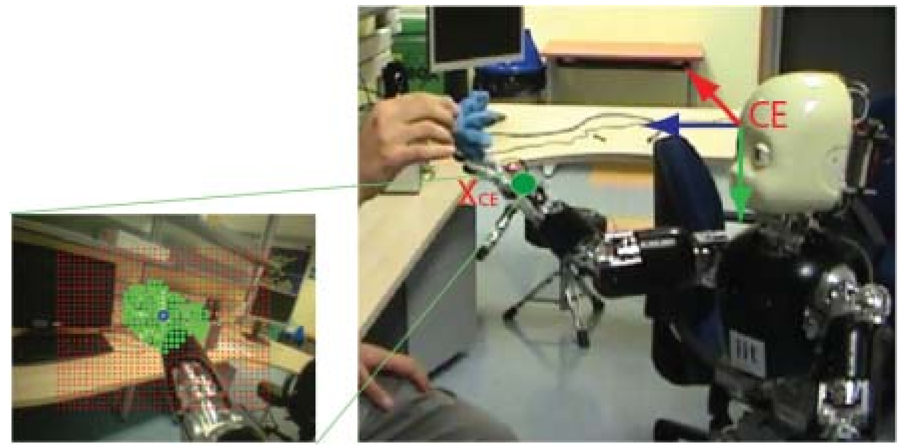
verification: time



(d) iCub Forces

(e) iCub Torques

verification: reaching



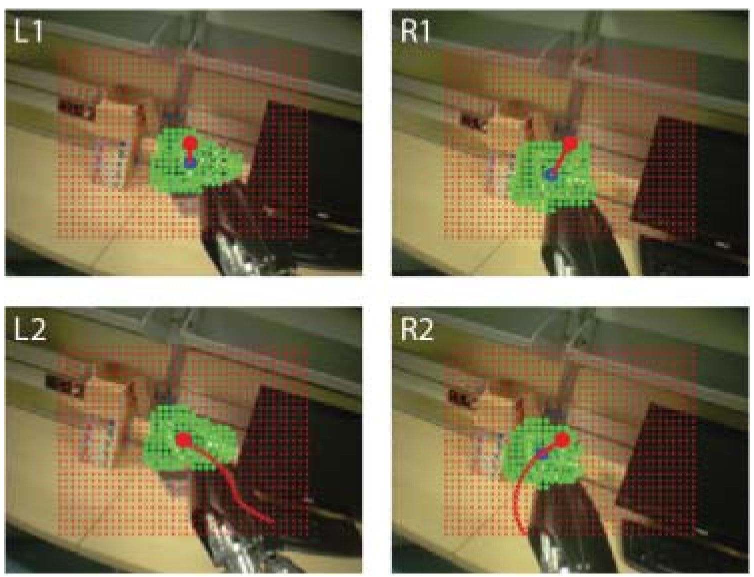
$$(x, y, z)_{CE} = M\left(\underbrace{u_l, v_l, u_r, v_r}_{\text{image}}, T, V_s, V_g\right)$$

fixation point

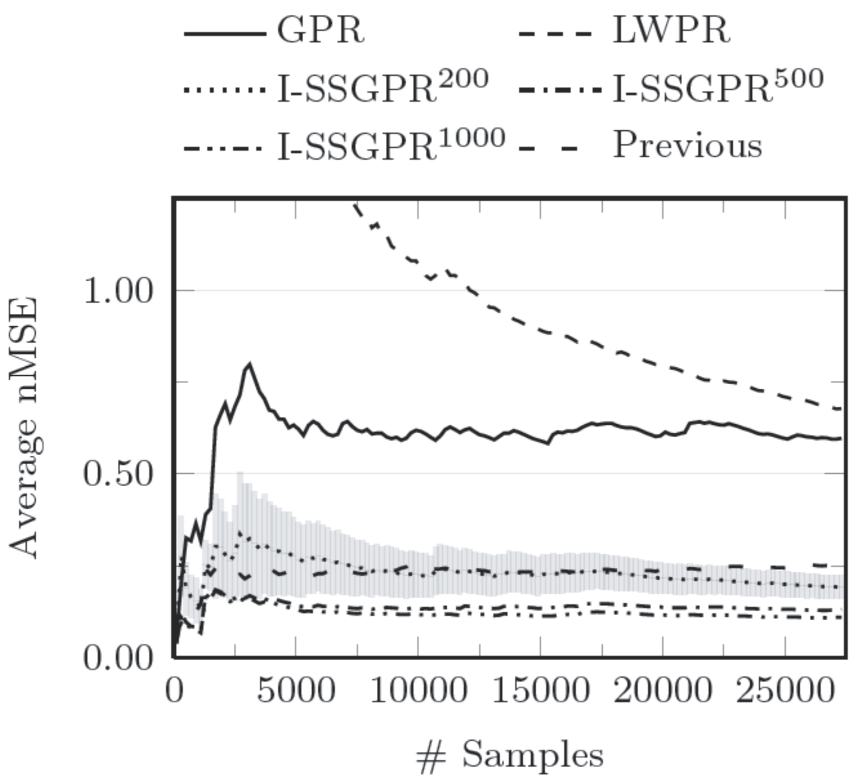
to learn

image

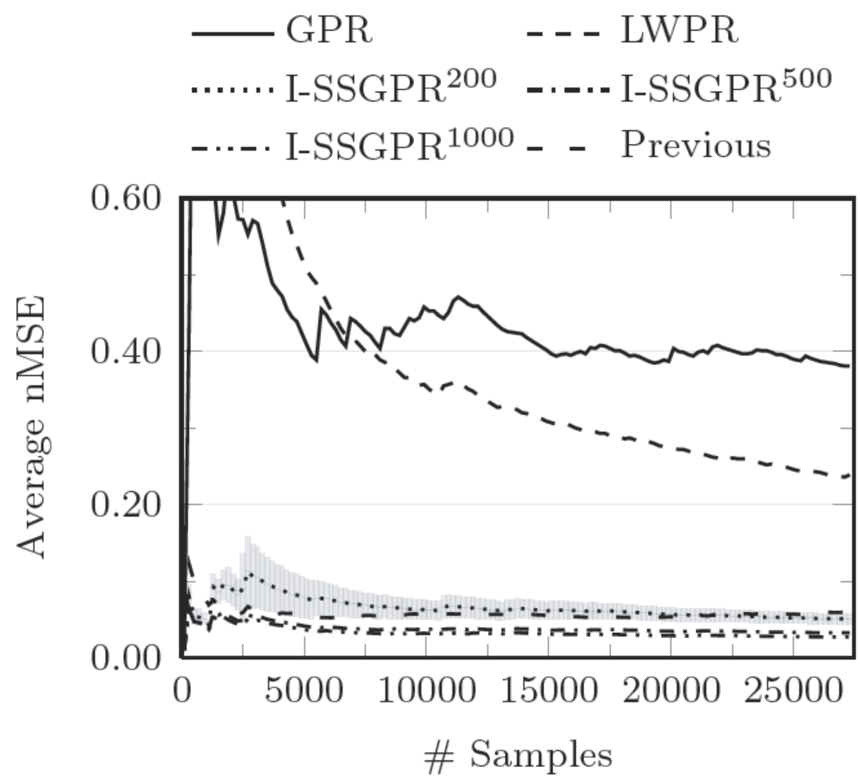
eye configuration



verification

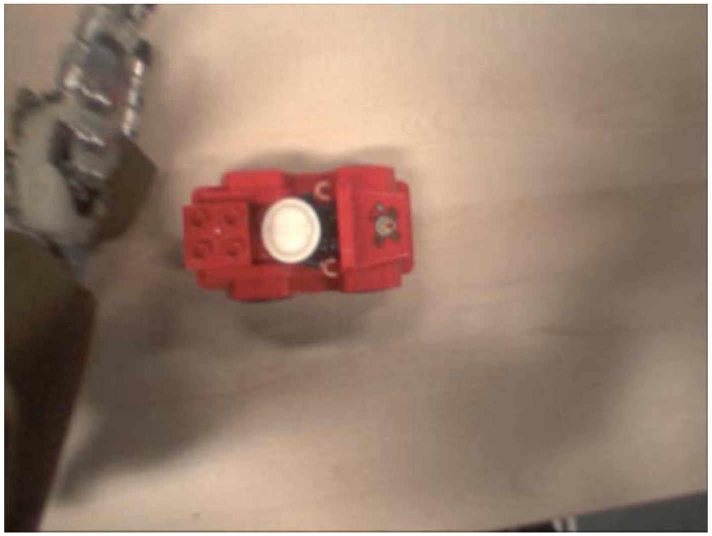


(a) u_{left}

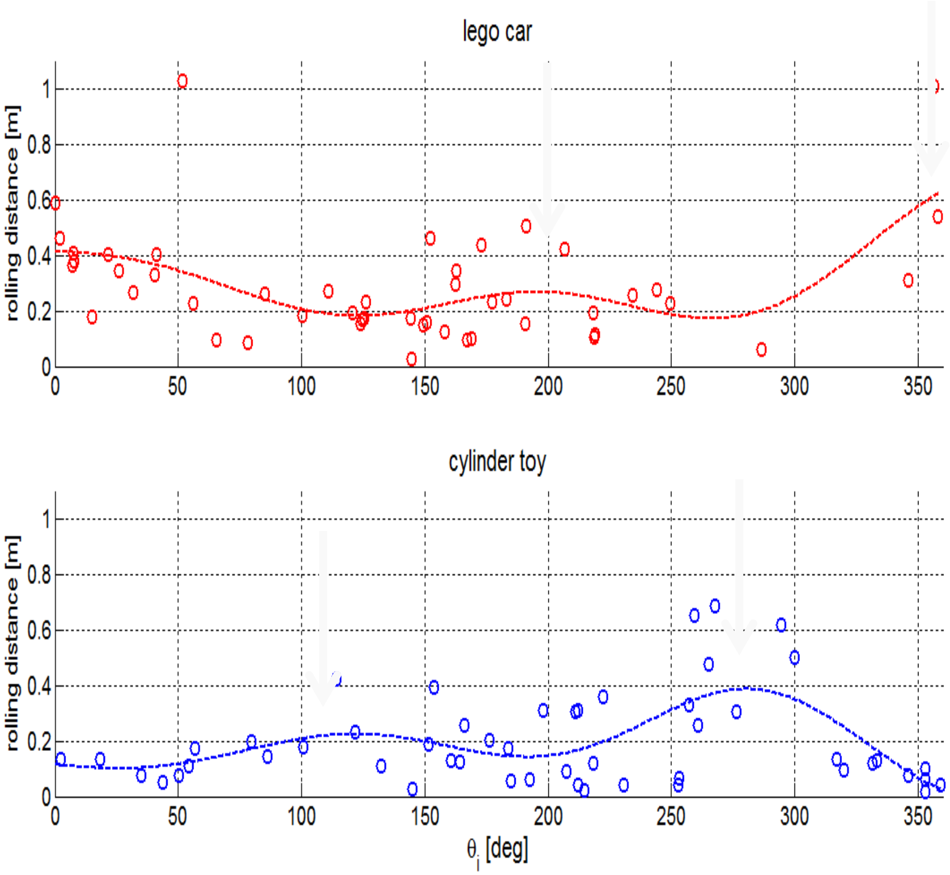


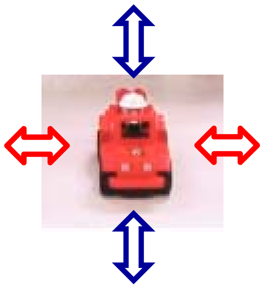
(b) u_{right}

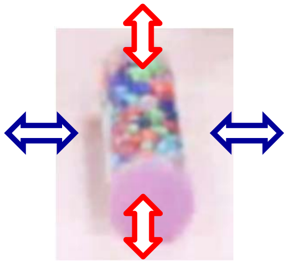
affordances (learning objects)

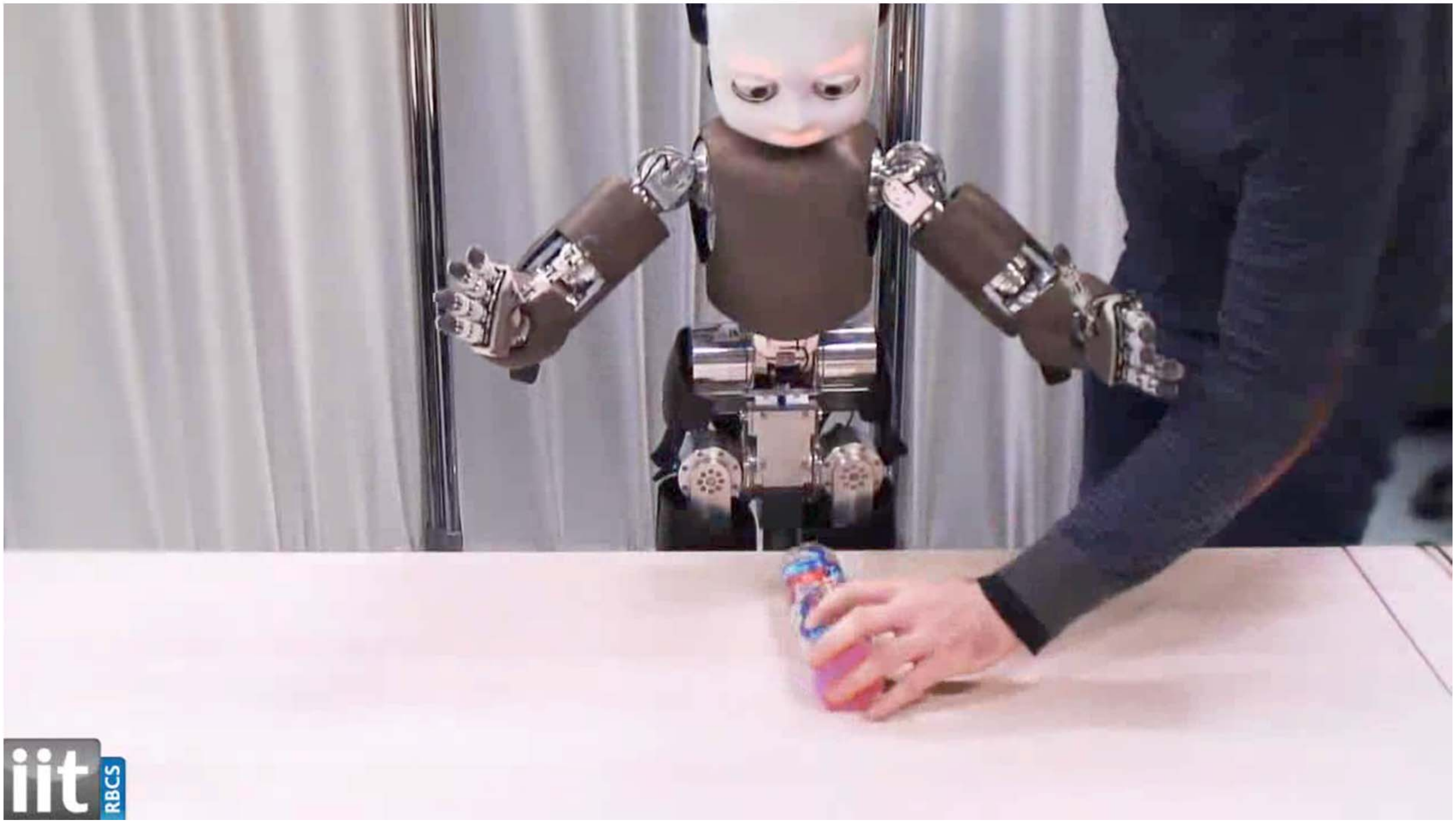


learning object behavior








iit
RBCS

conclusions

- incremental learning is advantageous when models cannot be assumed stationary
- ridge regression with kernel approximation and exact update rule for efficient incremental learning
- RFRR has an *O(1)* time and space complexity per update (suitable for hard real-time)
- number of random features regulates computation vs. accuracy tradeoff

sponsors

- EU Commission projects:
  - RobotCub, grant FP6-004370, http://www.robotcub.org
  - CHRIS, grant FP7-215805, http://www.chrisfp7.eu
  - ITALK, grant FP7-214668, http://italkproject.org
  - Poeticon, grant FP7-215843 http://www.poeticon.eu
  - Robotdoc, grant FP7-ITN-235065 http://www.robotdoc.org
  - Roboskin, grant FP7-231500 http://www.roboskin.eu
  - Xperience, grant FP7-270273 http://www.xperience.org
  - EFAA, grant FP7-270490 http://notthereyet.eu

- More information: http://www.iCub.org

Cogsys Cognitive Systems

RobotCub