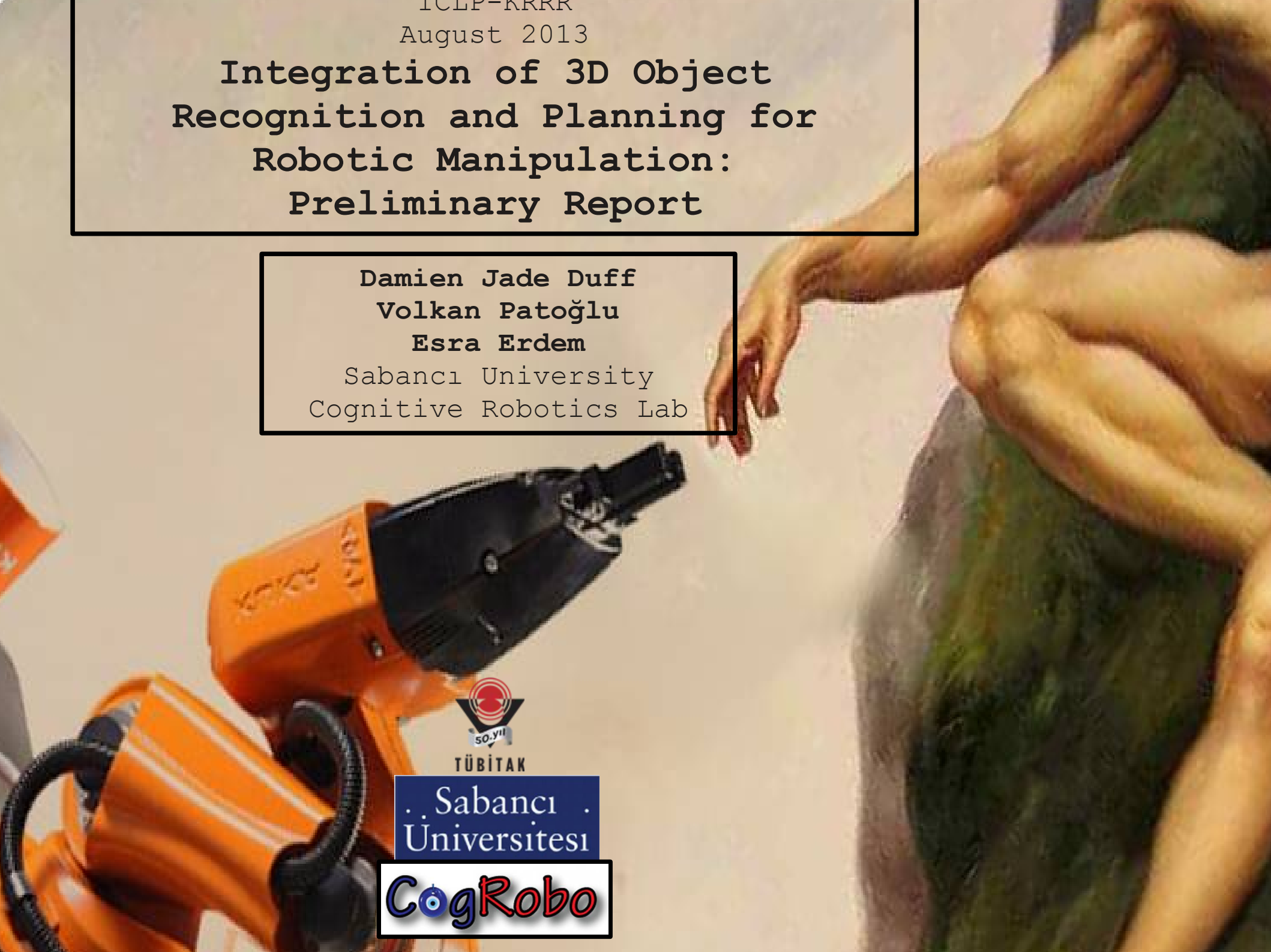
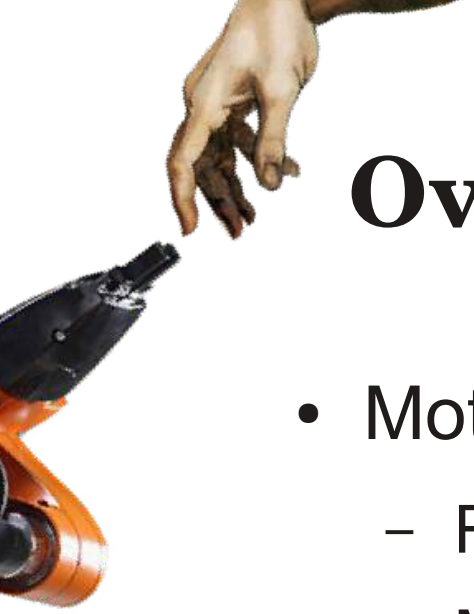
ICLP-KRRR
August 2013

Integration of 3D Object Recognition and Planning for Robotic Manipulation: Preliminary Report

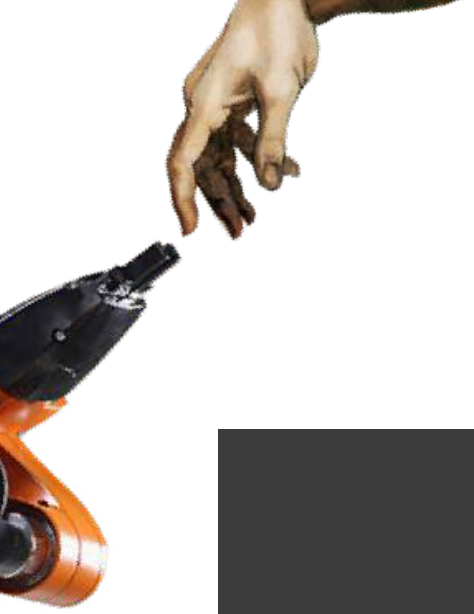
Damien Jade Duff
Volkan Patoğlu
Esra Erdem
Sabancı University
Cognitive Robotics Lab





Overview

- Motivation
  - Robocup@Work
  - Mobile manipulation & planning
- Overview of approach
- Domain description
- Incorporating perception
- Experiments and conclusions



Mobile manipulation Robocup@Work

Why mobile manipulation?

The direction of industry

Reasoning for manipulation?

- Discrete planning:
  - Placement puzzles
  - Picking puzzles
  - Sequences of manipulations (task planning)
- Continuous planning:
  - Shape
  - Motion
  - Dynamics

Perception for planning

- Initial state (bottom up)
  - Segment & name objects
  - Associate goal with objects
- Extra object information (top down)
  - Shape recognition
- Check feasibility of actions
  - Taking an object
  - Moving an object
  - Placing an object

Incorporating perception

- Recognition/shape information:
  - Reachability
  - Stackability
- Approaches
  - External predicates
    - Precomputed (PRE)
    - On-the-fly (INT)***
  - Plan checking
    - Filtering (FILT)
    - Replanning (REPL)

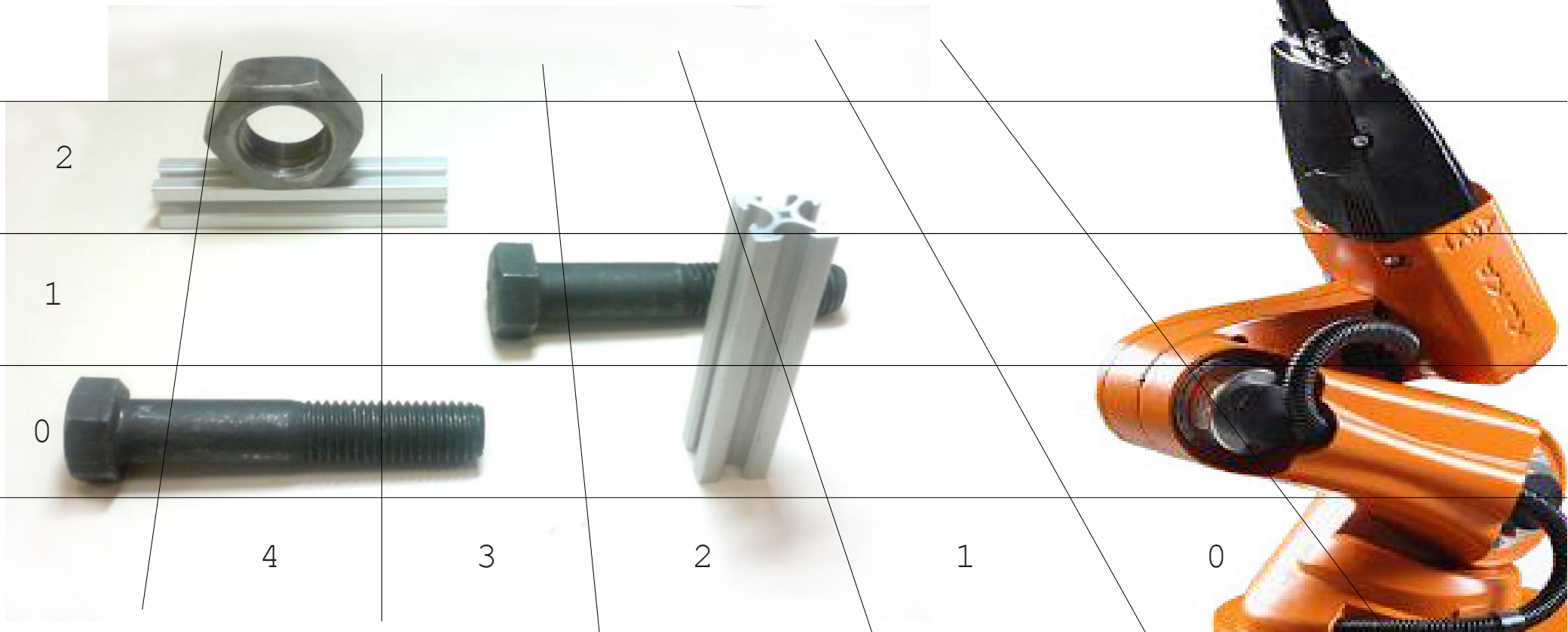
*** INT not examined in this work

Pick and place domain

C+ Fluents

is_at (scobj3) = scobj1
is_at (scobj1) = loc_4x2
is_at (bolt1) = loc_4x0
is_at (scobj4) = loc_2x1
is_at (scobj2) = loc_2x0

ori_is (scobj3) = horiz_y
ori_is (scobj1) = horiz_x
ori_is (bolt1) = horiz_x
ori_is (scobj4) = horix_x
ori_is (scobj2) = vert

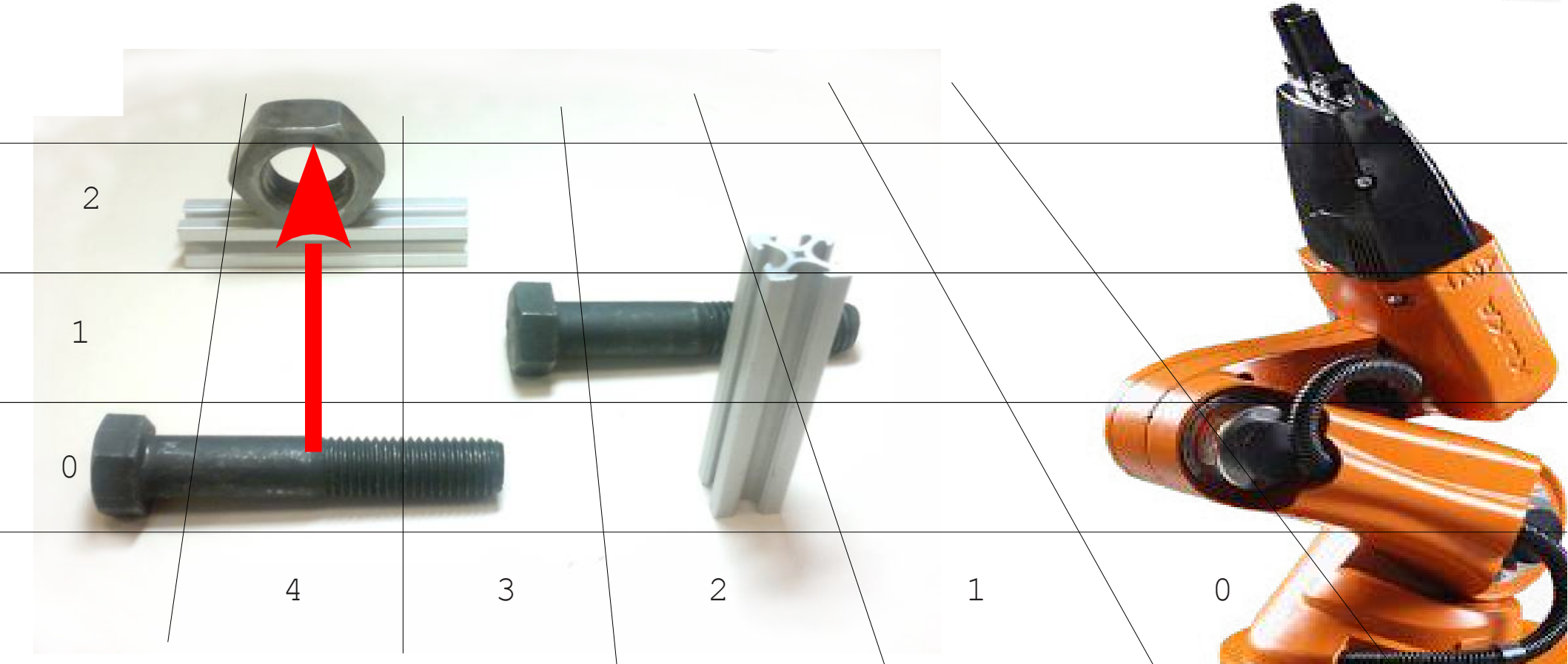


C+ Actions

move(obj, loc, orient) **causes** *is_at(obj) = loc*

move(obj, loc, orient) **causes** *ori_is(obj) = orient*

caused *false if is_at(obj) = loc ∧ is_at(obj') = loc*



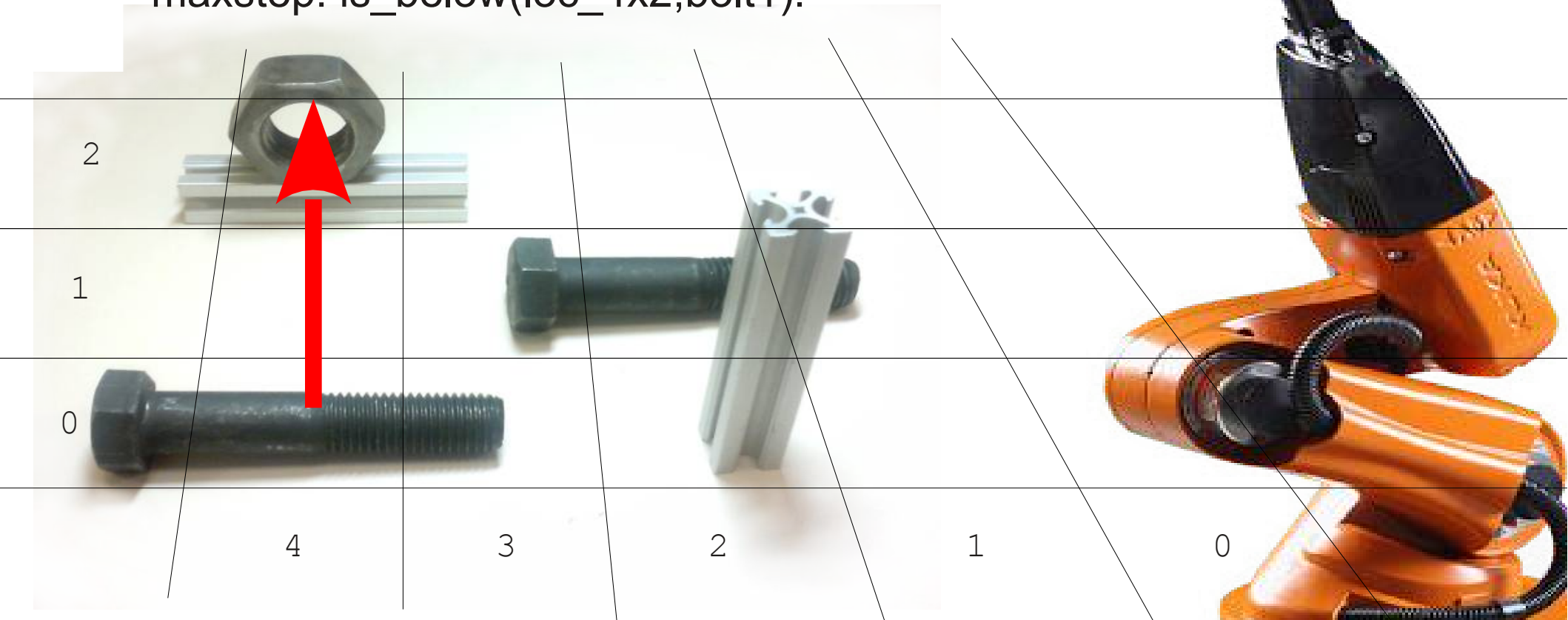
C+ Query

:- query

maxstep :: 0..3;

0: is_at(scobj1)=loc_2x0, ori_is(scobj1)=vert,
is_at(bolt1)=loc_4x0, ori_is(bolt1)=horiz_x,
…;

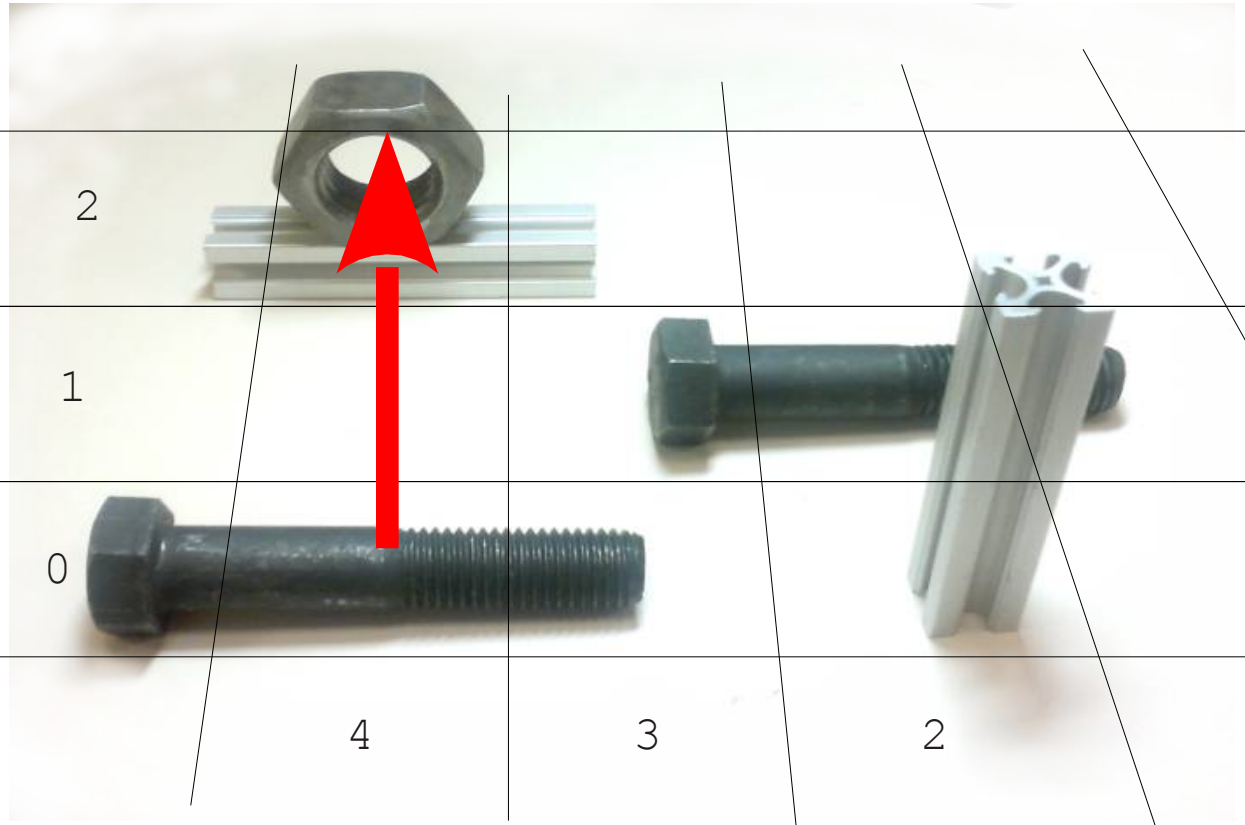
maxstep: is_below(loc_4x2,bolt1).



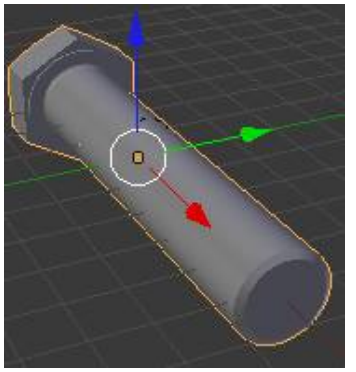
Perception: input

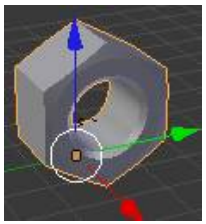
- Input:
  - Kinect XYZRGB point cloud
  - Query:

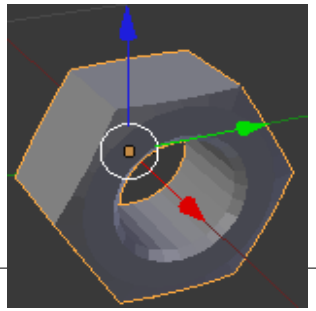
    `move bolt1 from loc_4x0 to loc_4x2`
  - Object database

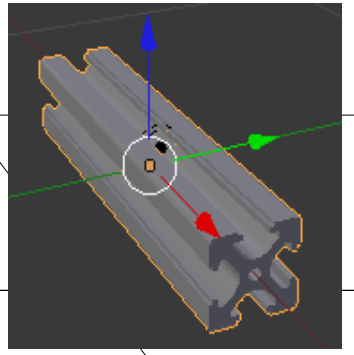


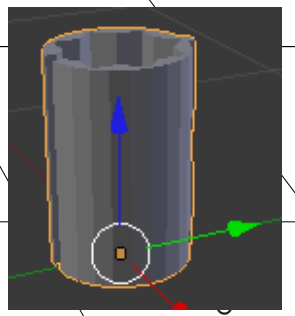
Object database

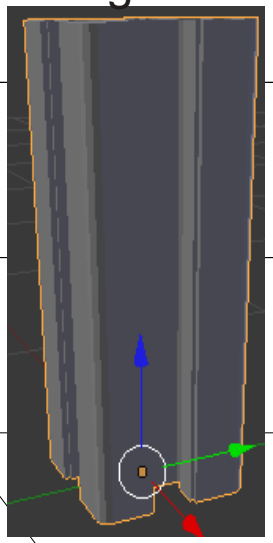

bolt


nut


bignut


aluprof


adapter

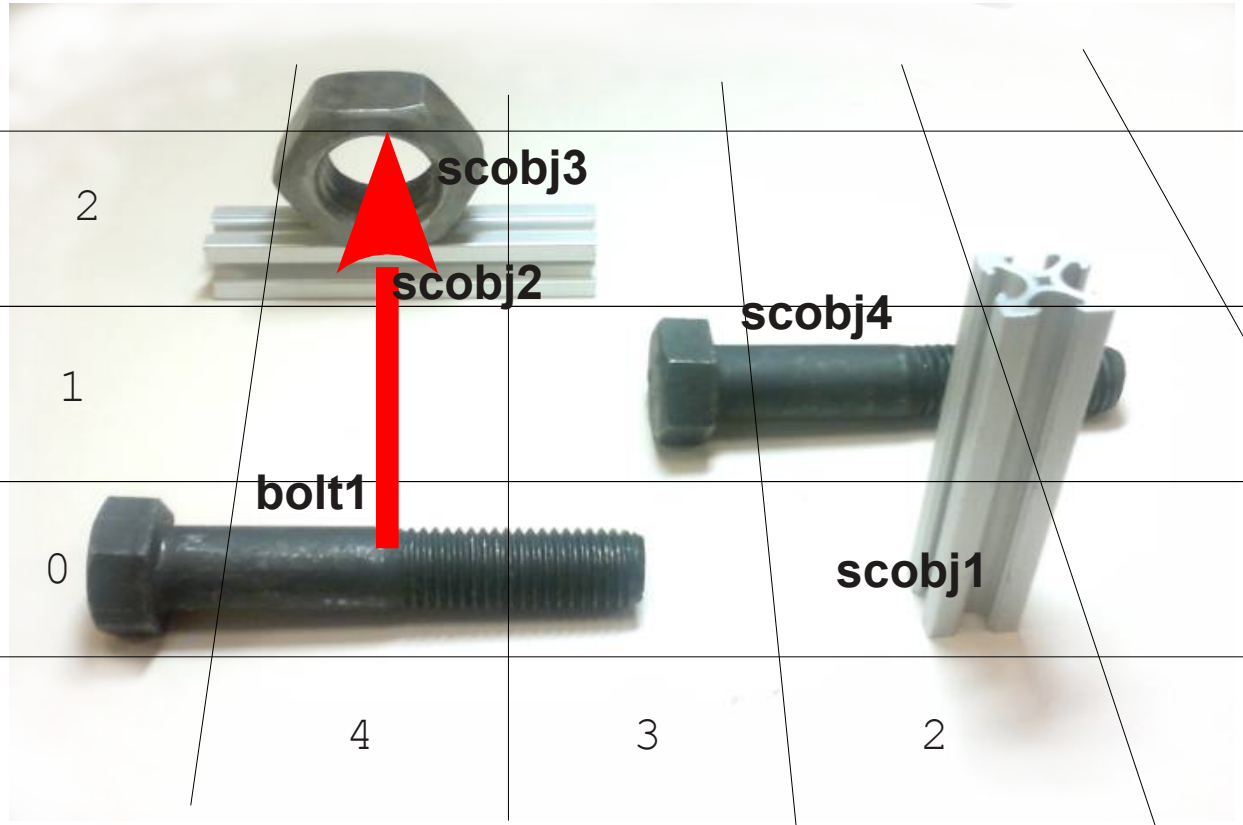

bigaluprof

Perception: segmentation, association

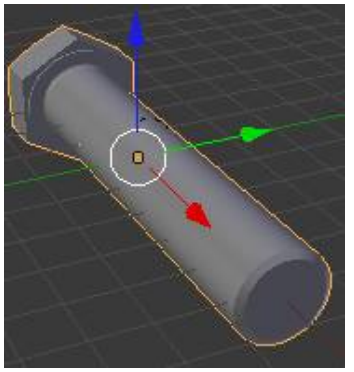
- Input:
  - Kinect XYZRGB point cloud
  - Query:

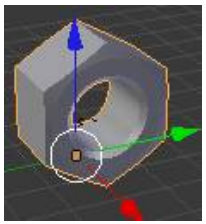
    ```
    move bolt1 from loc_4x0 to loc_4x2
    ```

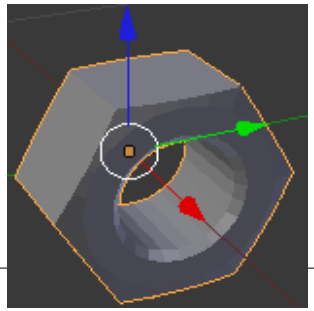
  - Object database
- Bottom-up phase (segmentation, association)

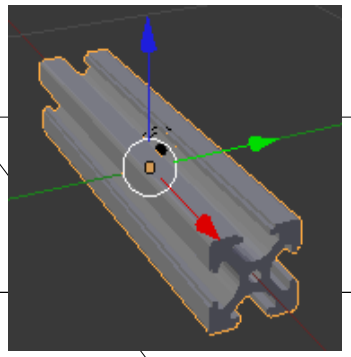


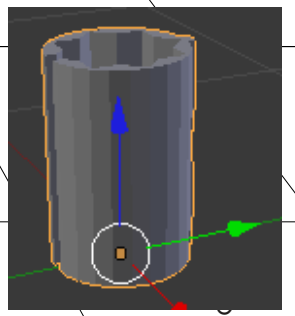
Object database

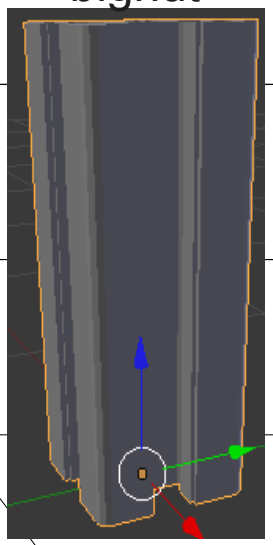

bolt


nut


bignut


aluprof


adapter

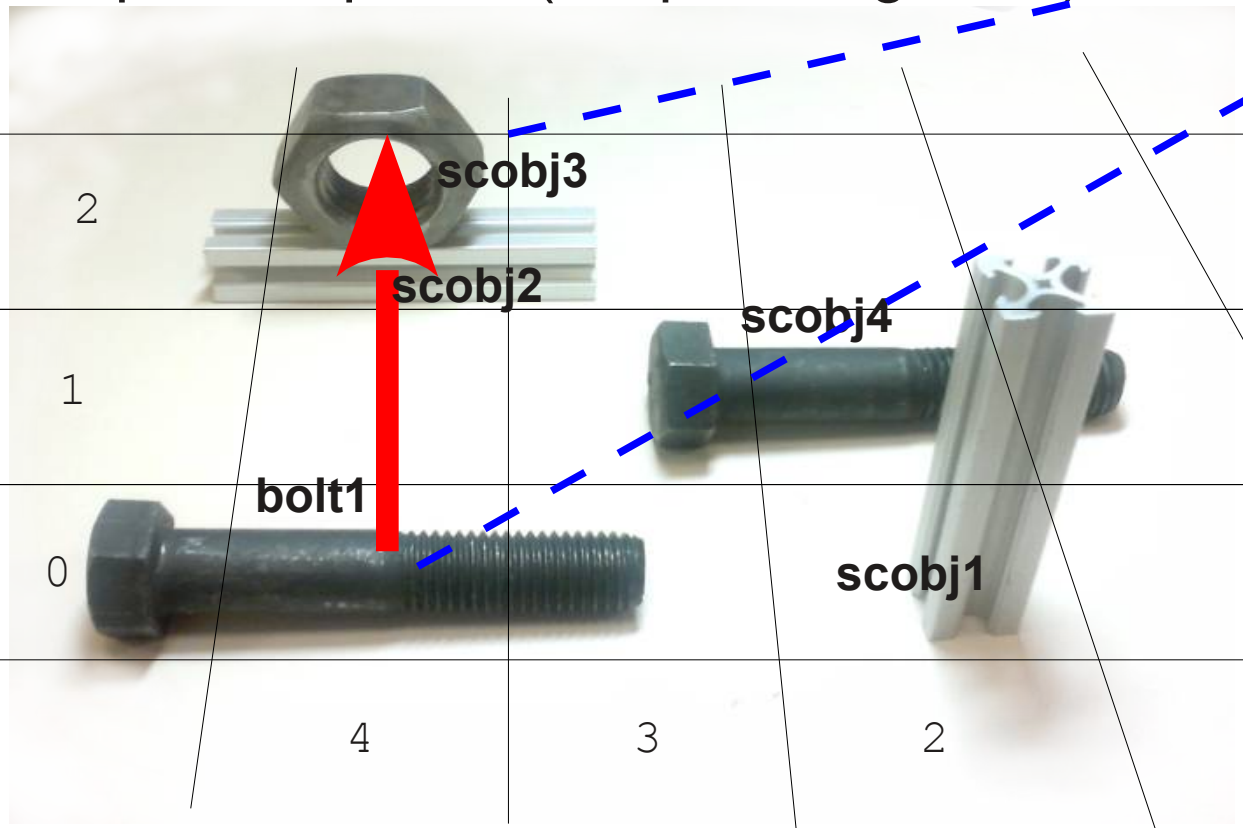

bigaluprof

Perception: shape recognition

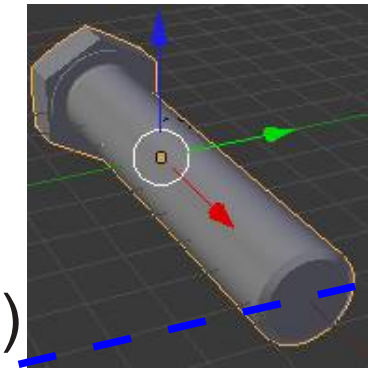
- Input:
  - Kinect XYZRGB point cloud
  - Query:

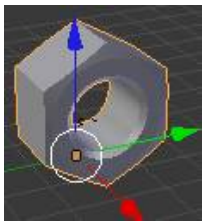
    ```
    move bolt1 from loc_4x0 to loc_4x2
    ```

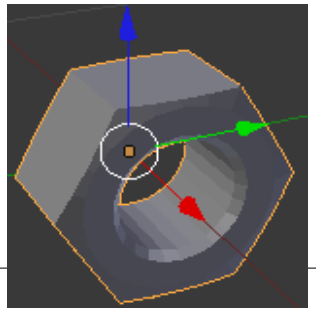
  - Object database
- Bottom-up phase (segmentation, association)
- Top-down phase (shape recognition)

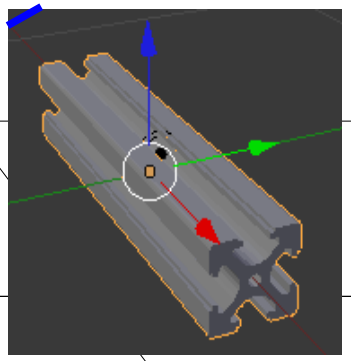


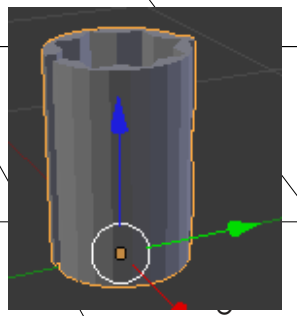
Object database

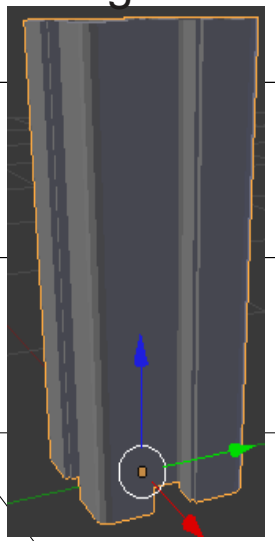

bolt


nut


bignut


aluprof


adapter


bigaluprof

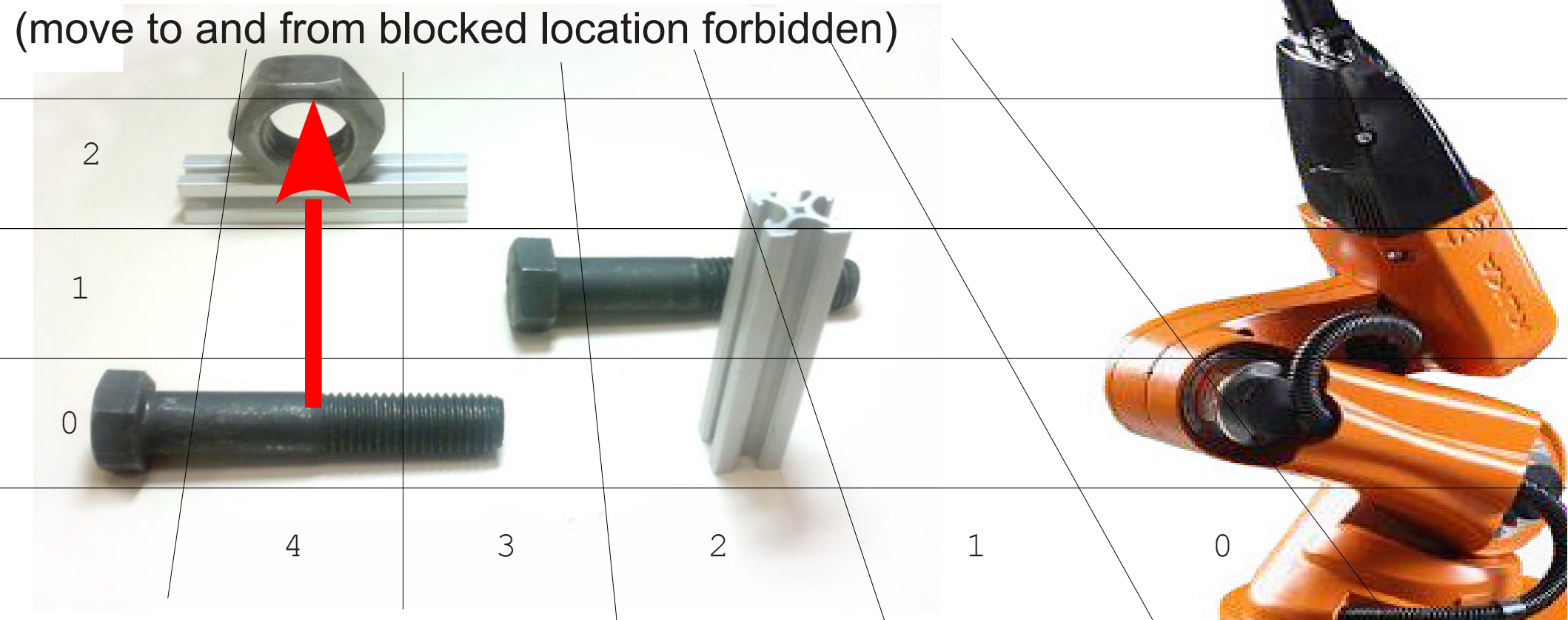
Perception via external predicates (PRE)

Stackability:

nonexecutable *move(obj,obj',orient)* **if** *ori_is(obj')= orient'*
(where `unstackable_ext`*(obj,orient,obj',orient') holds)*

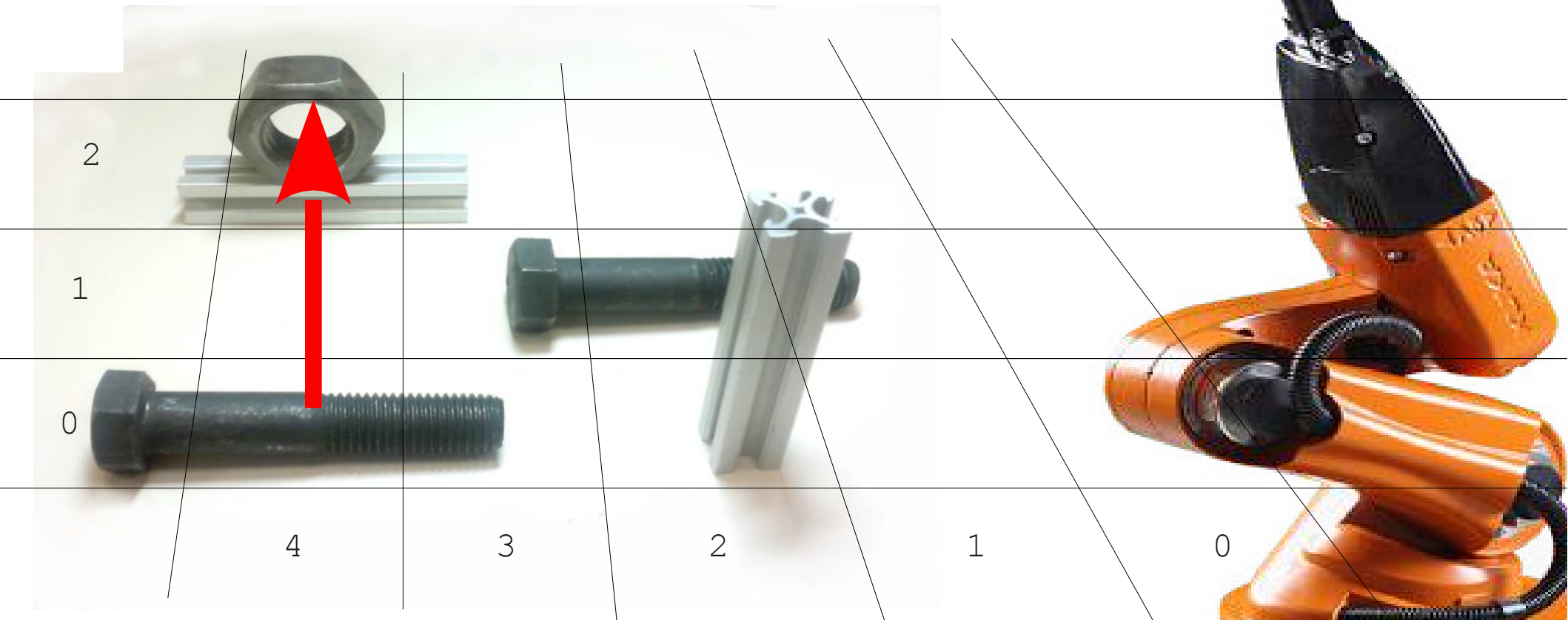
Reachability:

… `reach_blocked_ext`*(obj,loc,orient,obj',loc',orient') …*
(move to and from blocked location forbidden)



Perception via filtering (FILT)

- For each plan:
  - *What stacks are attempted?*
  - *What reaches may be blocked?*
- Obtain relevant object shapes
- Check feasibility of stacks and reaches

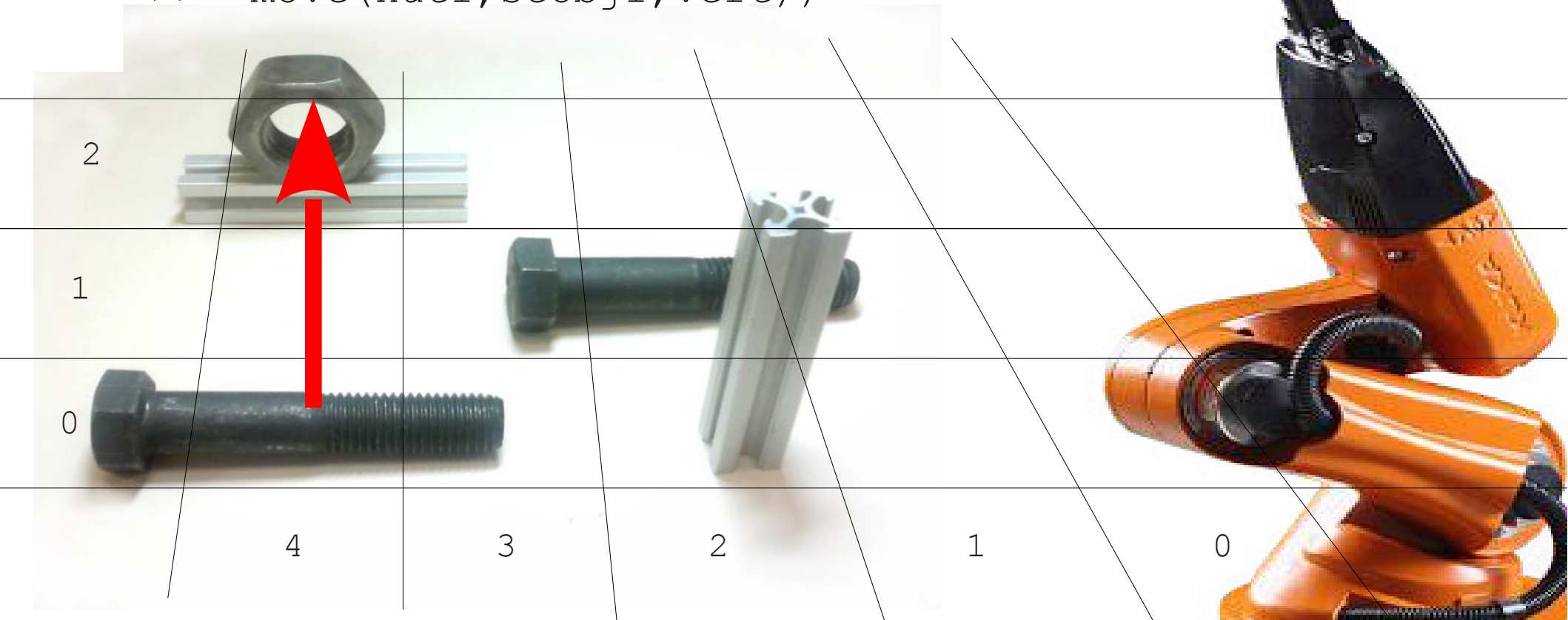


Perception via replanning (REPL)

- When infeasible plan encountered generate new constraints and rerun query:

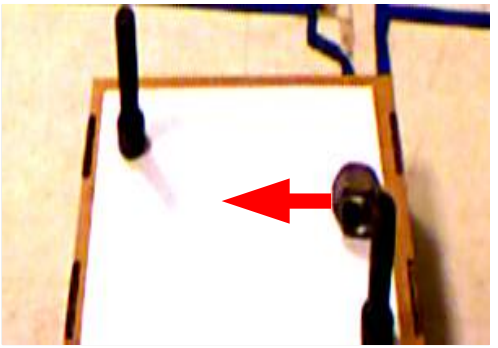
  e.g.

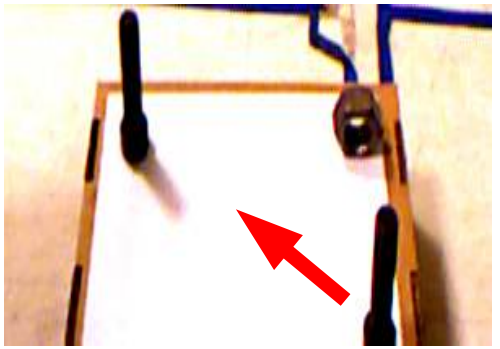
```
T=<maxstep-1 ->> (T: ori_is(scobj1)=vert
->> -move(nut1,scobj1,vert))
```

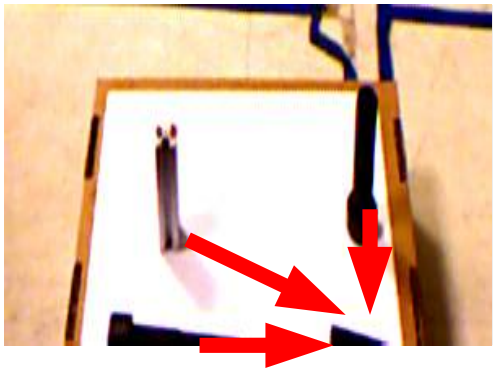


Experiments

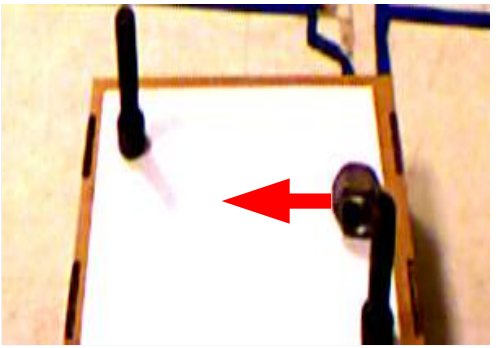
Experiments

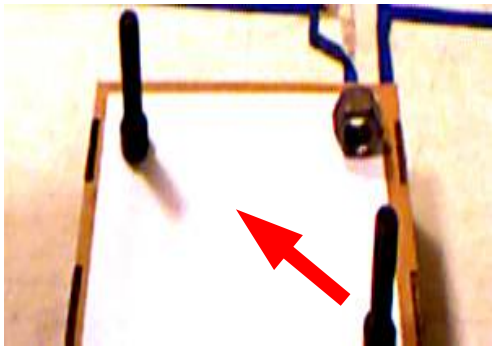


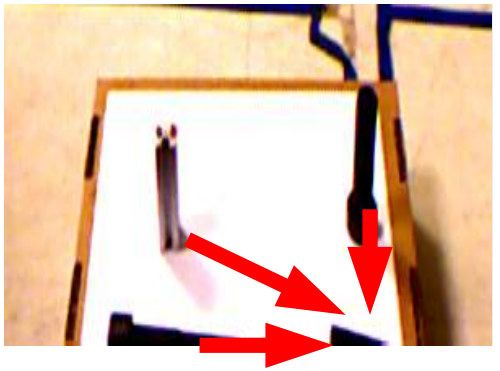




Number of infeasible plans checked (to calculate one plan)



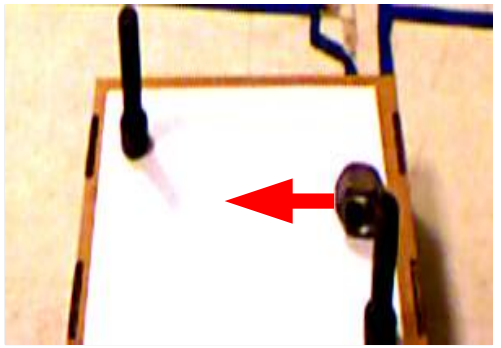
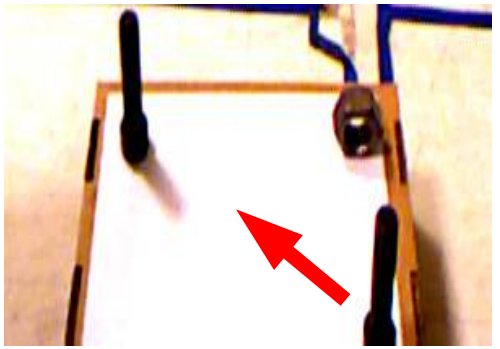
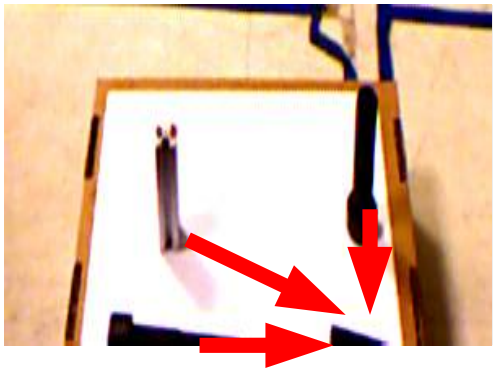




PRE	0	0	0
FILT	61	0	1760
REPL	1	0	1

n=5

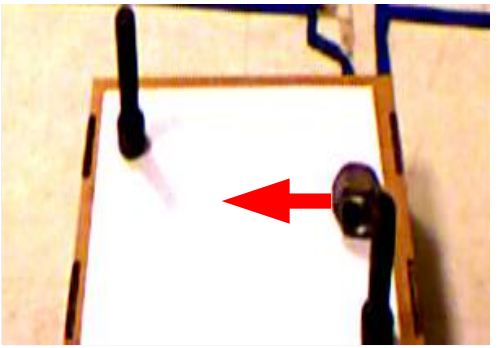
Number of perceptual queries (to calculate one plan)

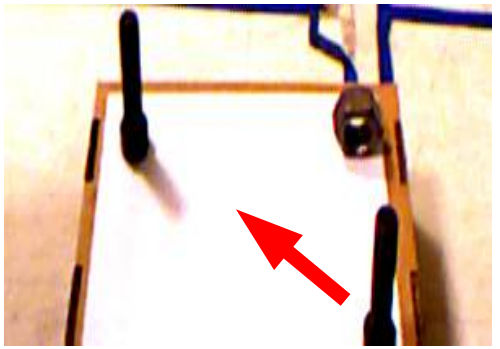




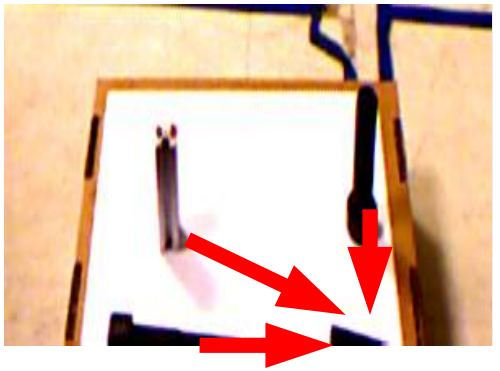
PRE	3	3	4
FILT	2	0	4
REPL	1.4	0	4

n=5

Number of infeasible plans checked (to calculate 100 plans)



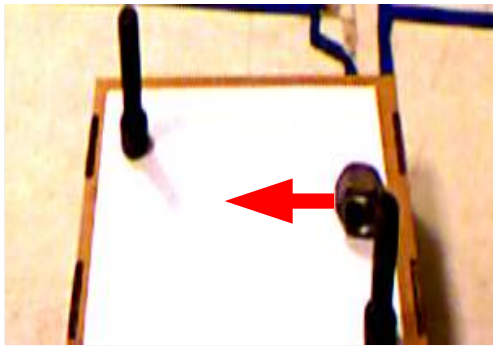
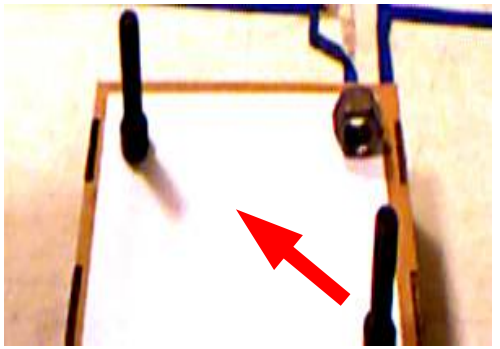
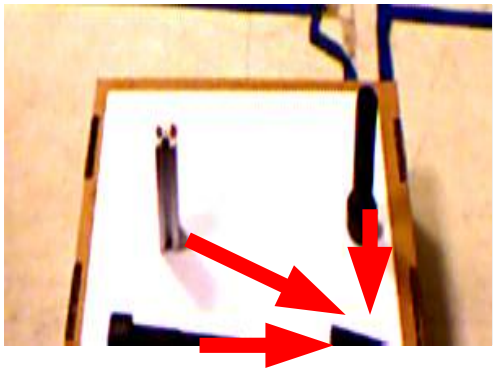




PRE	0	0	0
FILT	505.3	6.0	1760
REPL	1.5	0	1.0

n=5

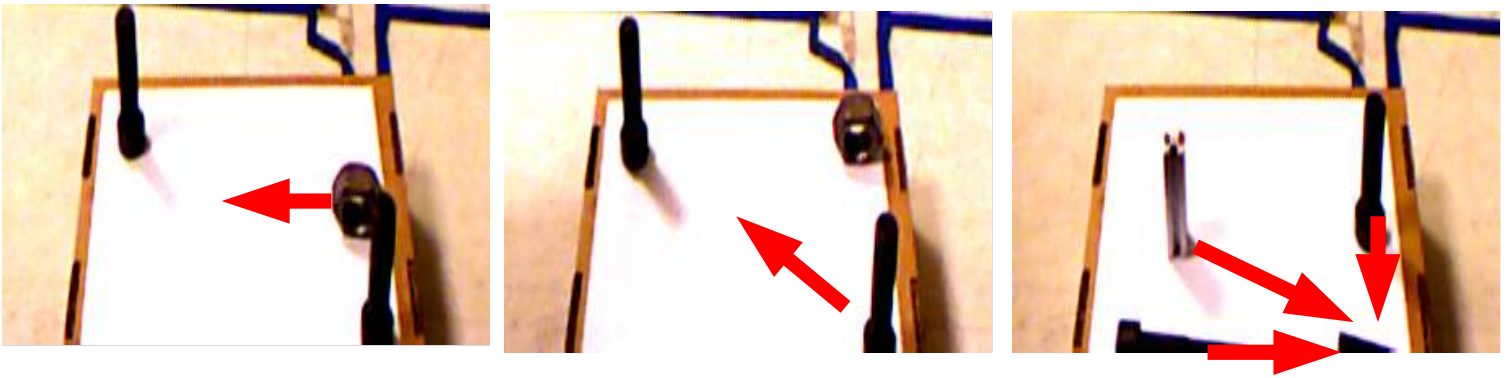
Number of perceptual queries (to calculate 100 plans)

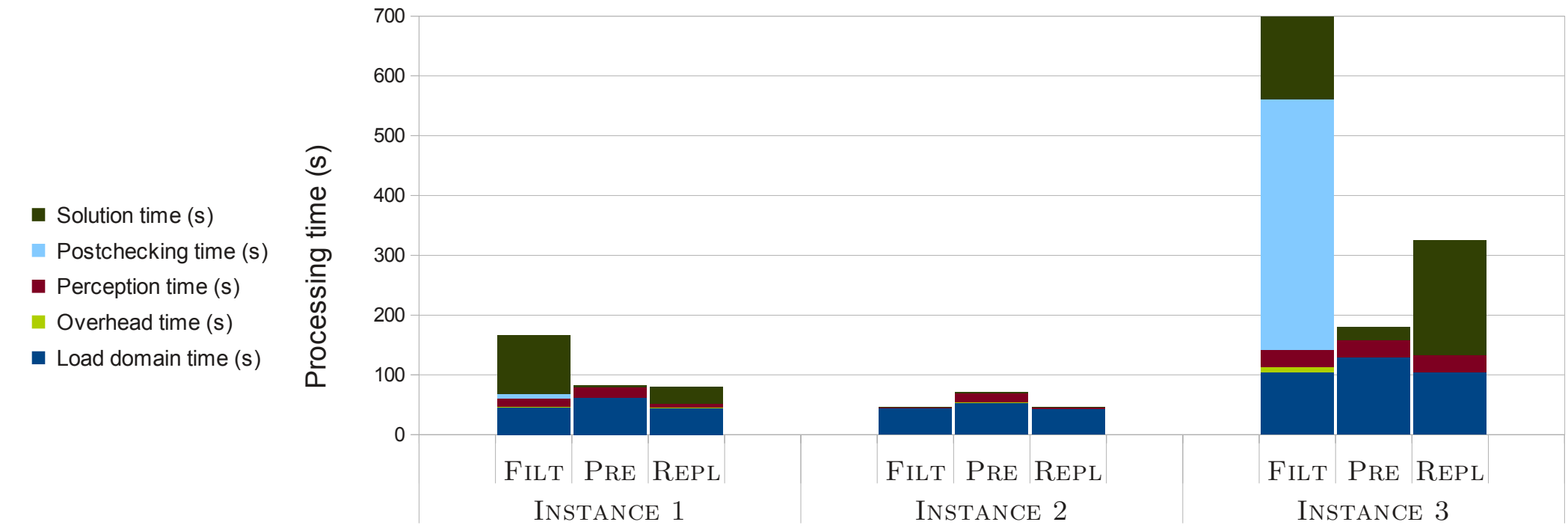




PRE	3	3	4
FILT	3	1.2	4
REPL	1.6	0.8	4

n=5

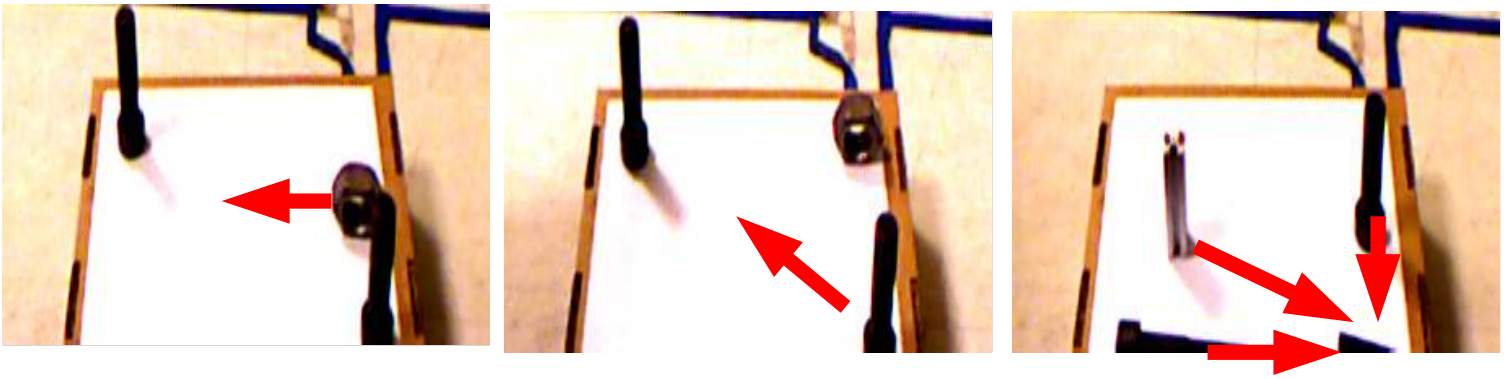
Computation time (for 1 plan)

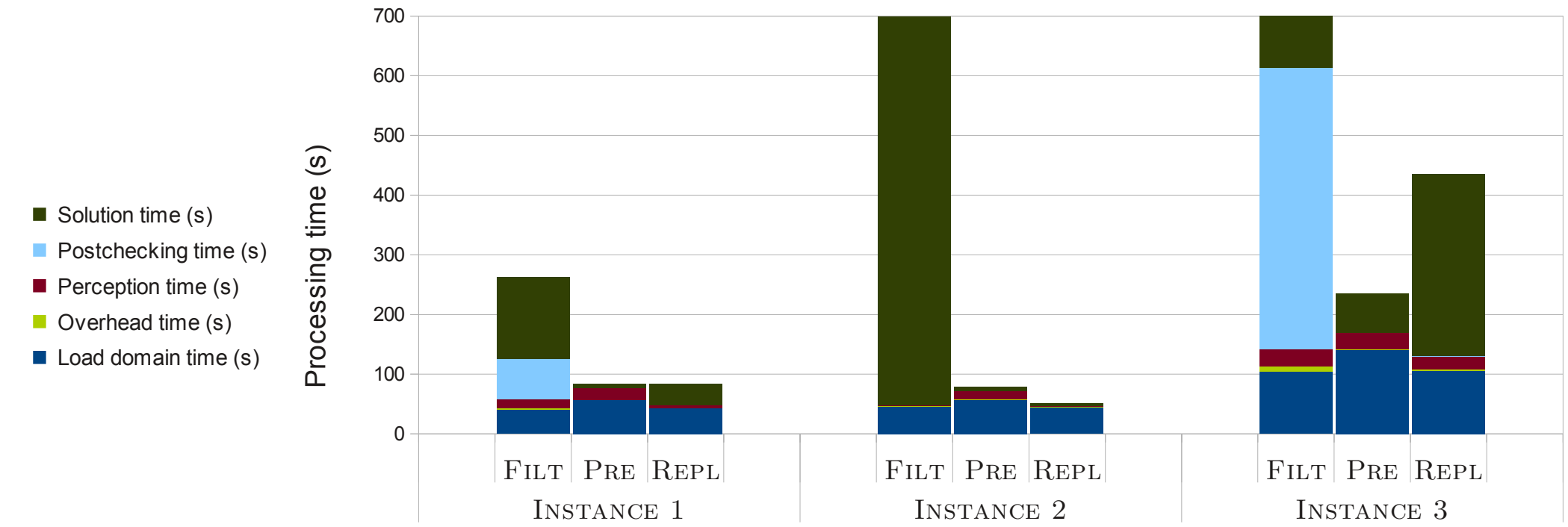




n=5

Computation time (for 100 plans)





n=5

Conclusions

- PRE:
  - No need to check infeasible plans
  - External computation referred to in domain description
  - Unnecessary perceptual computation
- FILT:
  - Many infeasible plans
  - Unnecessary perceptual computation
- REPL:
  - Very small number of infeasible plans
  - Minimizes perceptual computation
  - Reduces initial domain load