

# TRS: An Open-source Recipe for Teaching/Learning Robotics with a Simulator.

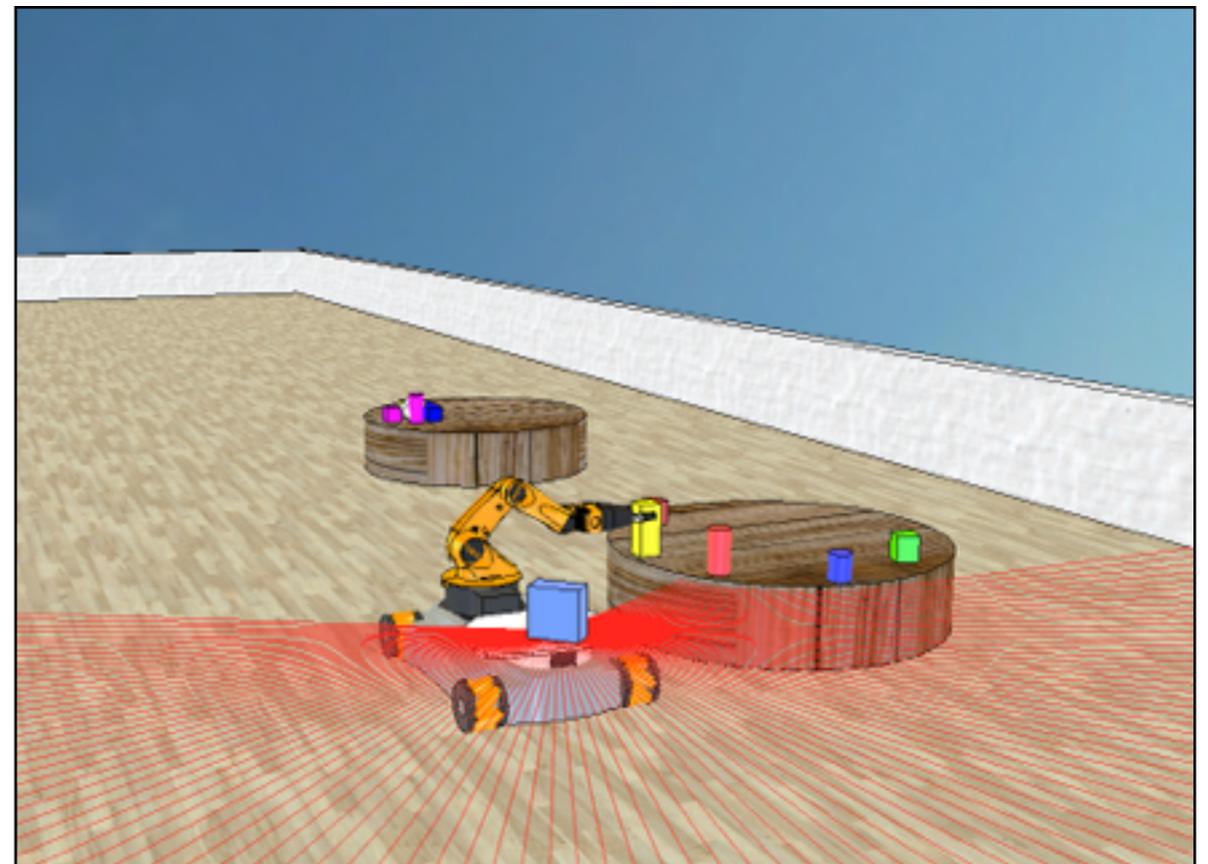
Learning Robotics with a Simulator: Setup Your Laptop in 5 min,  
Code a Control, Navigation, Vision or Manipulation Project in 100  
Lines.

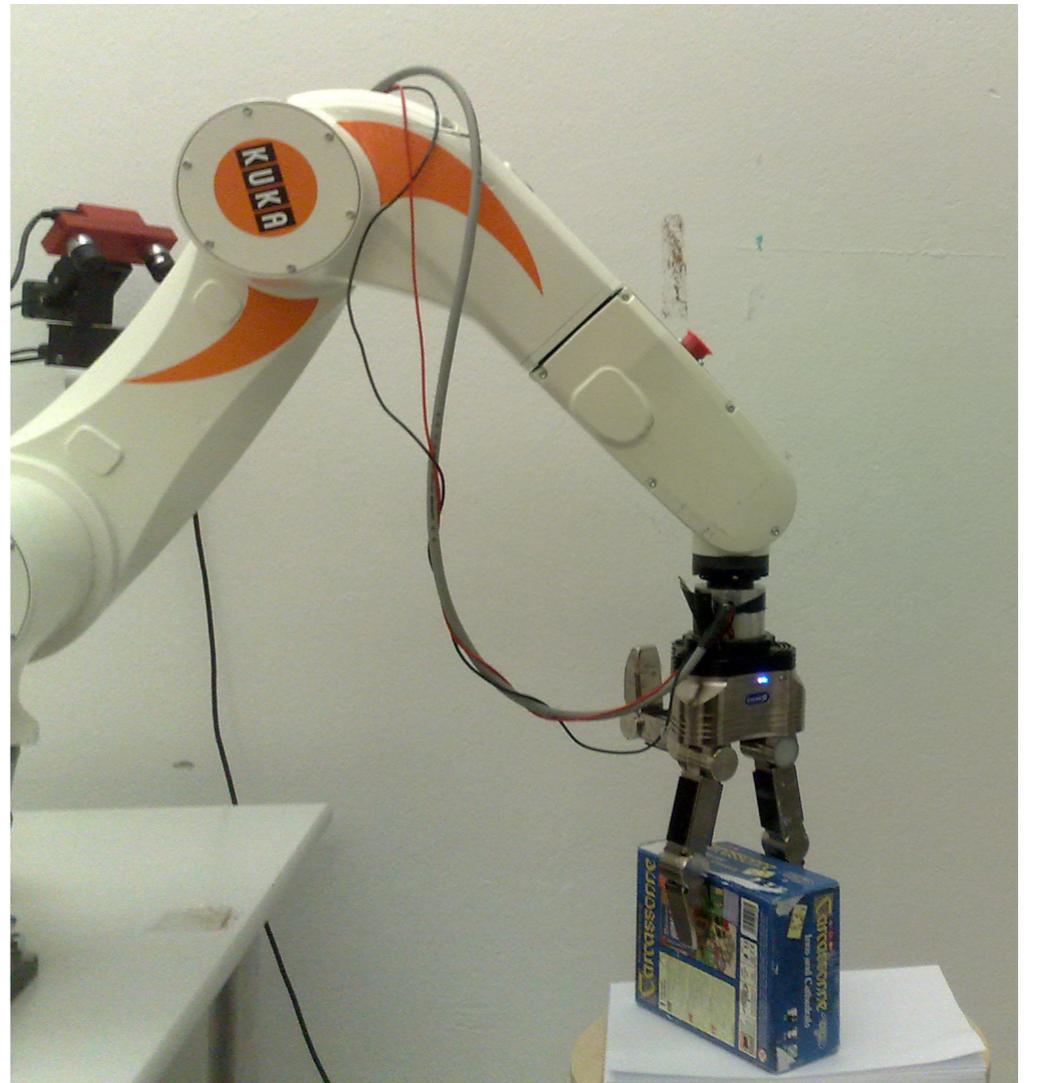
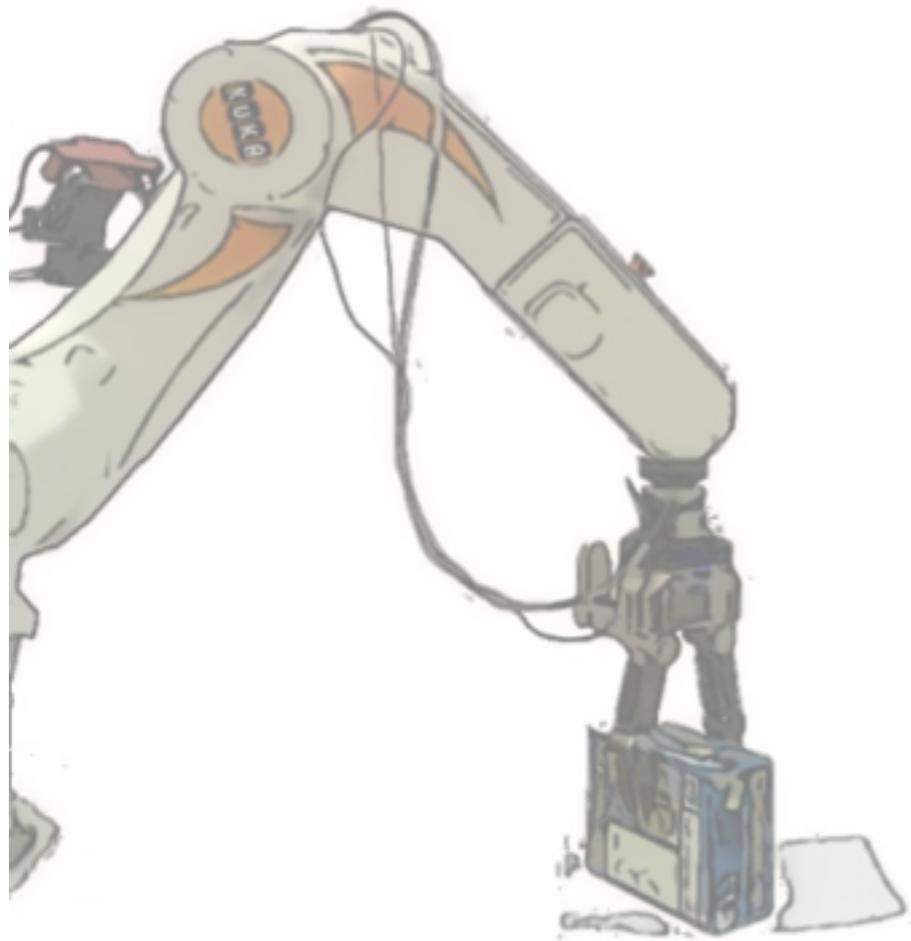
**Renaud Detry**

University of Liège, Belgium

**Peter Corke**

**Marc Freese**





**TRS: an environment that**

- **can be setup in a few minutes,**
- **allows students to code vision, navigation, manipulation in few lines of code**

# V-REP + Peter Corke's Matlab Robot Toolbox

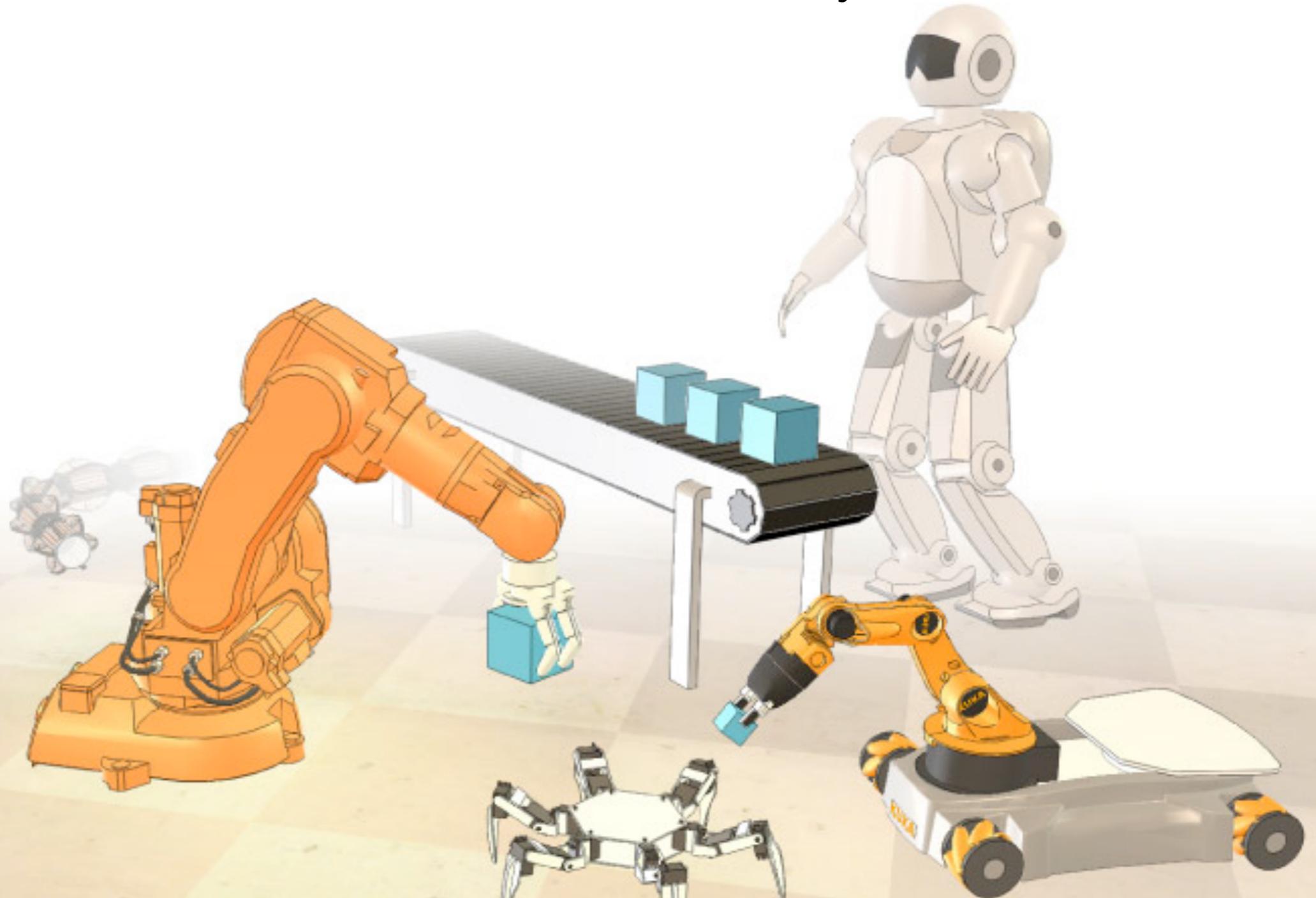
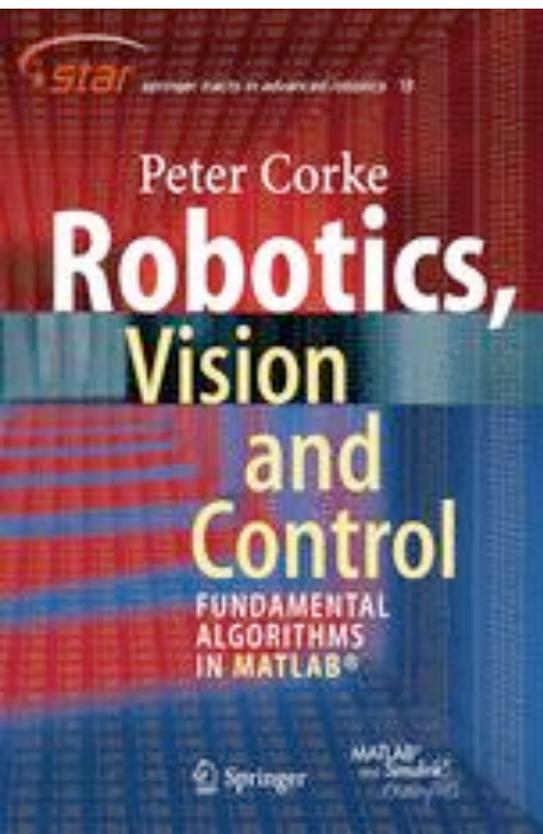
## Toolbox: Code

- Control
- Vision
- Navigation

in 100 lines!

## VREP:

- Trivial installation — Linux, Mac, Win
- Remote API for C++, Python, Matlab, ROS





GAZEBO



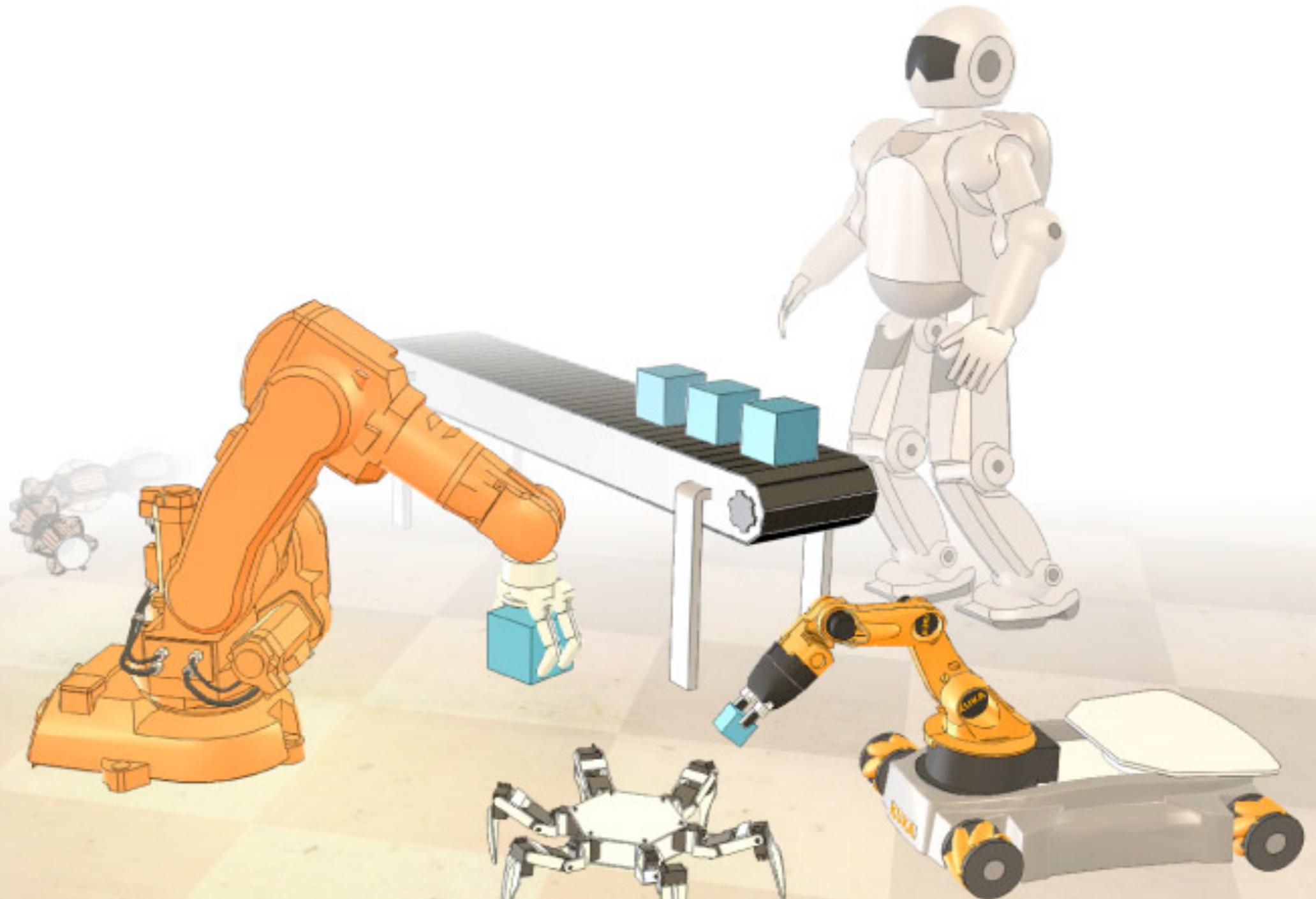
OpenRAVE

**ROBWORK**

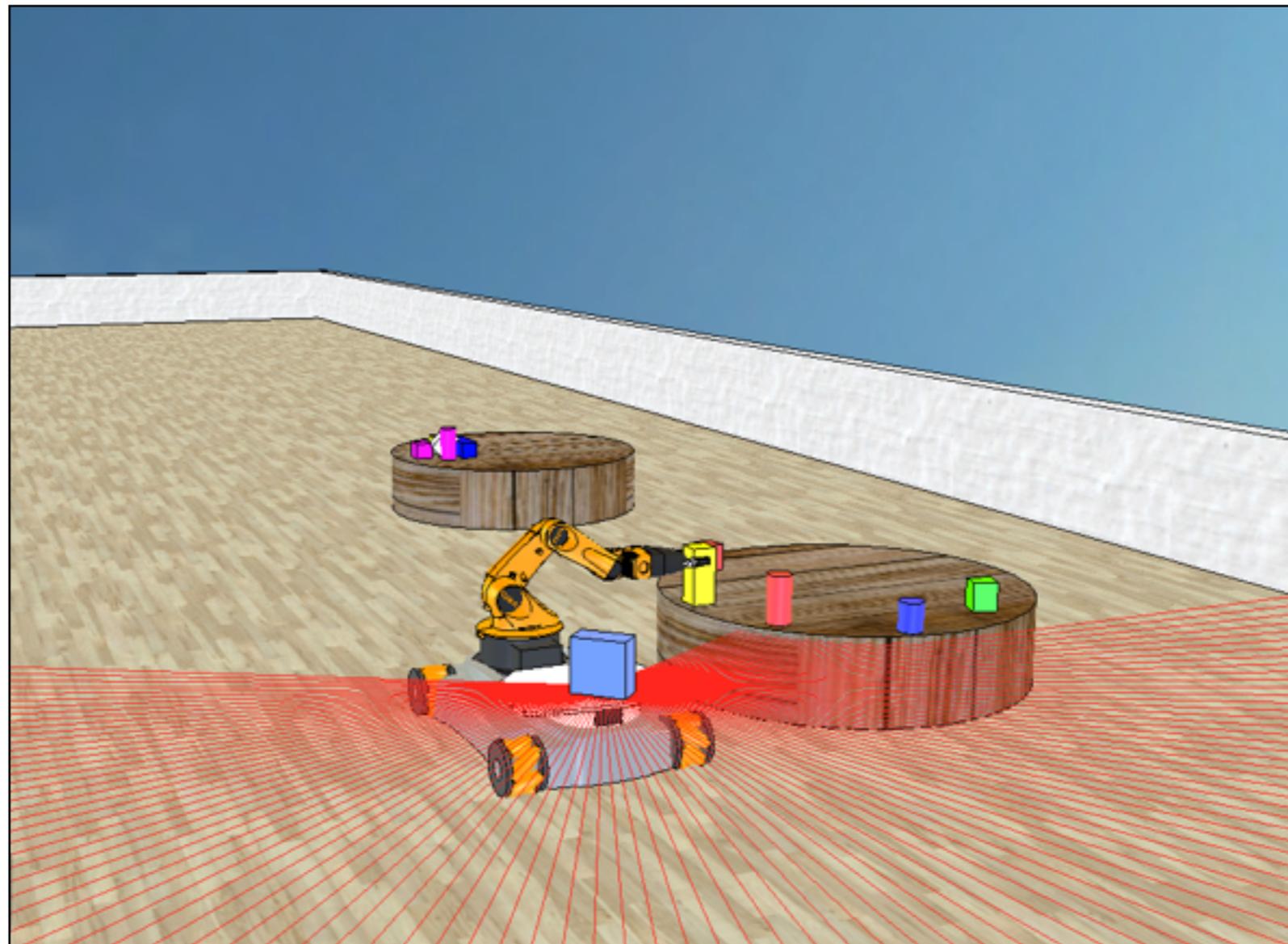
RobWork | Yet Another Robotics Library

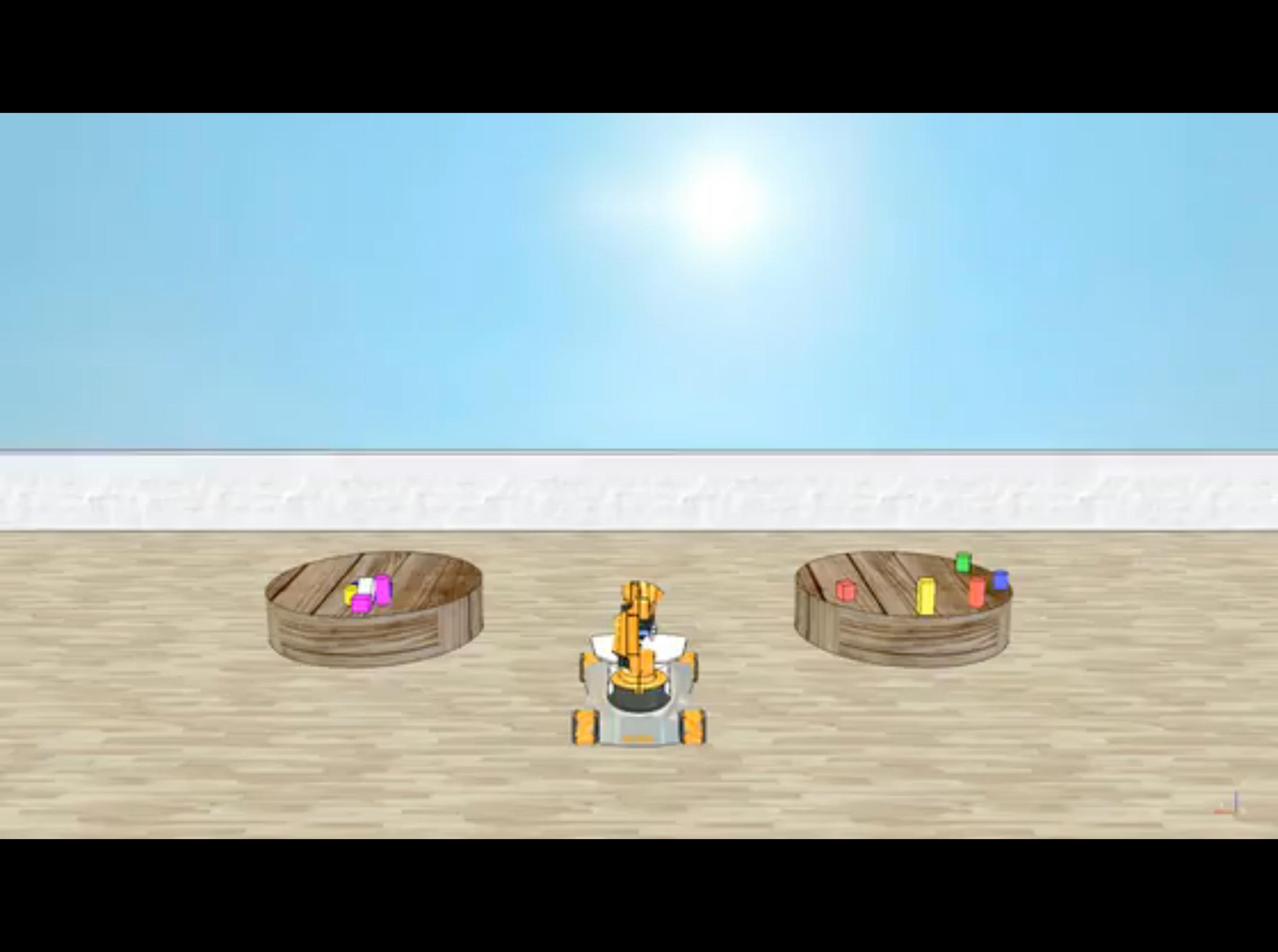
# V-REP: An Easy Robot Simulator

- Trivial installation — Linux, Mac, Win
- Remote API for C++, Python, Matlab, ROS
- Stable, fast.



- Involves: control, navigation, mapping, vision and manipulation.
- Goal: pickup groceries from a table and store them in baskets.



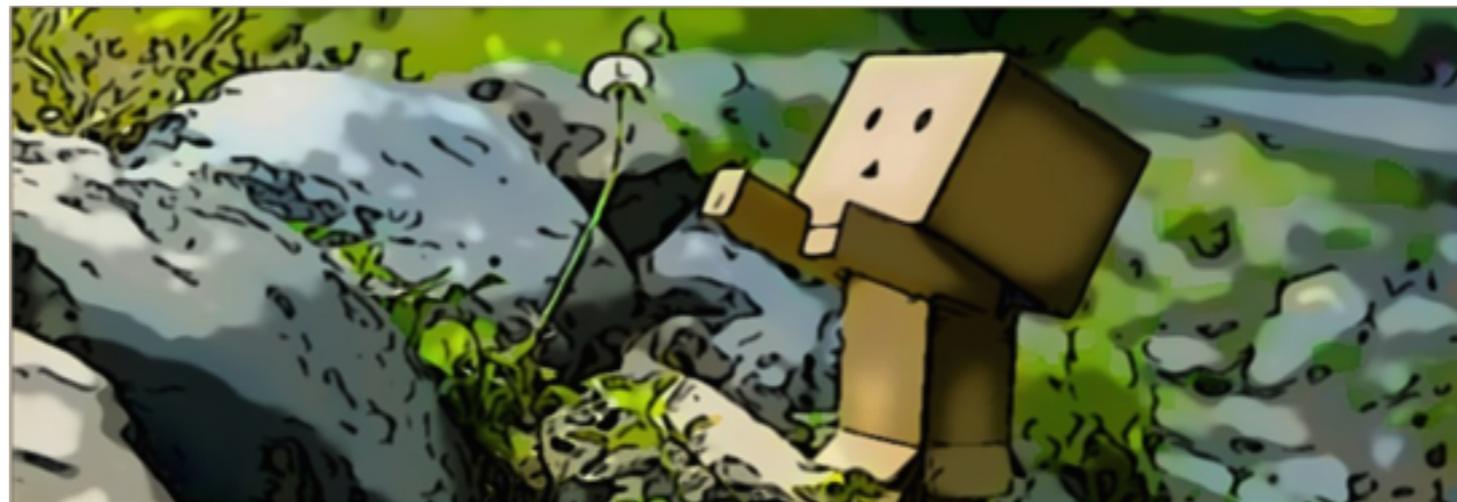


# What You Get

We provide a complete recipe for organizing the project:

- a V-REP model of a house floor and a youBot,
- a Matlab script that illustrates access the robot's sensors and actuators,
- a page that explains how to setup a laptop to work on the project,
- a page that presents the project definition: objectives, milestones, a description of the robot and the documentation of the Matlab functions that access the robot's sensors and actuators.

<http://teaching-robotics.org/trs>



**TRS:** An Open-source Recipe for Teaching/Learning Robotics with a Simulator.

:: setup a laptop in 5 minutes, write a control, navigation, vision or manipulation program in 100 lines of code

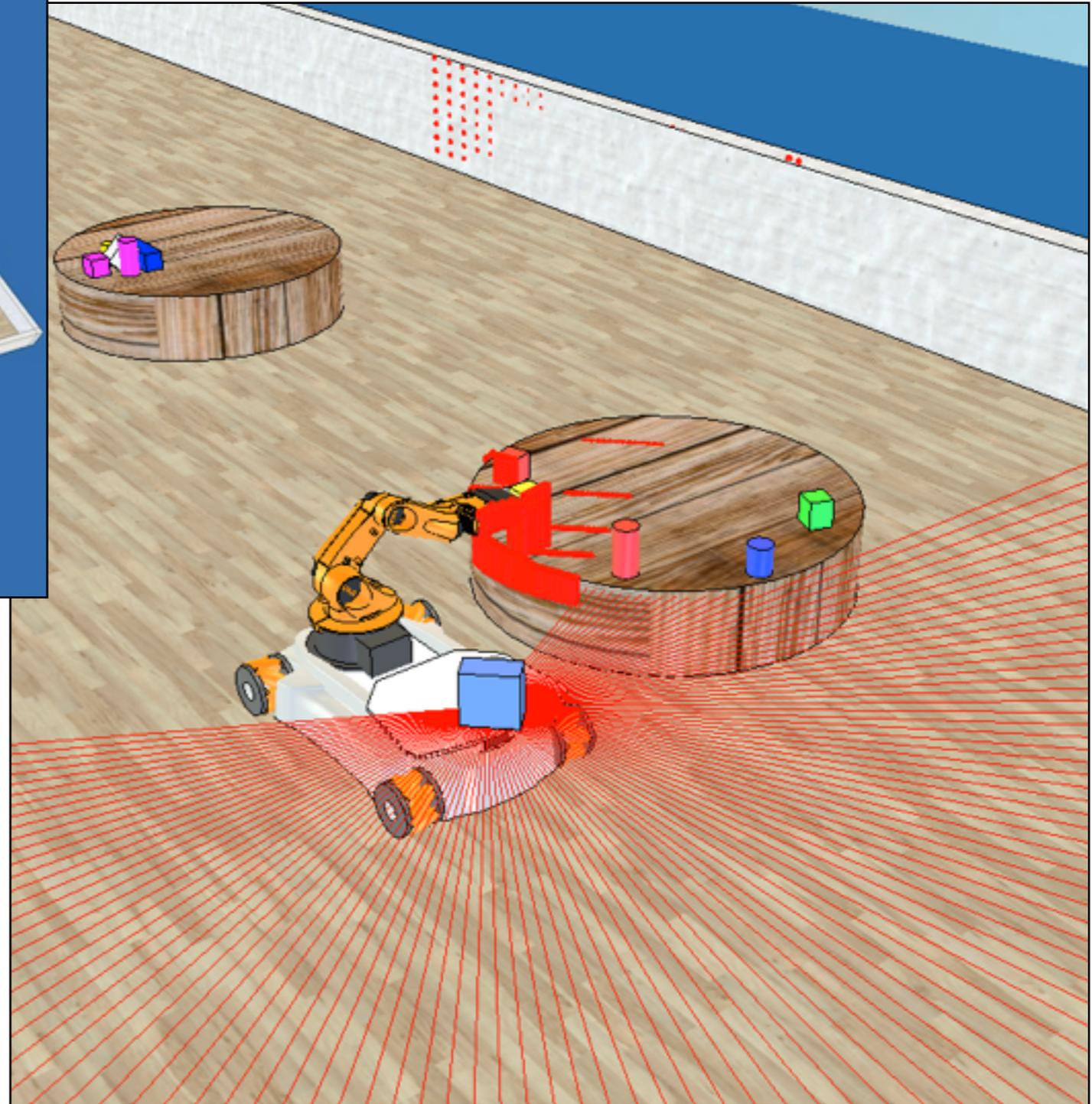
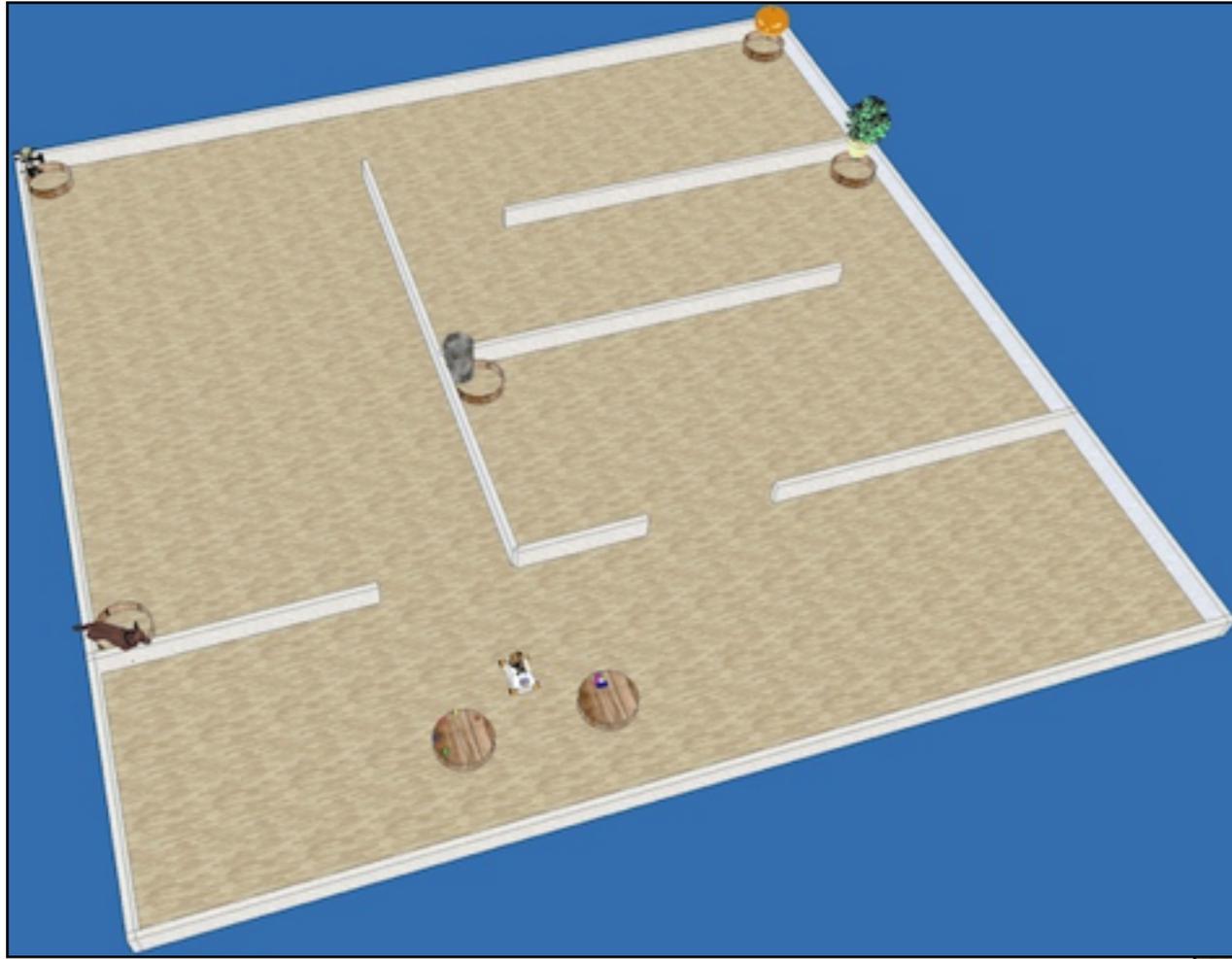
TRS

Motivation

**Motivation**

TRS is a cross-platform robot development and simulation environment that can

# 1) V-REP model of a youBot and a house



# 2) Matlab script showing how to access the youBot

```
function youbot()
% youbot illustrates the V-REP Matlab bindings.

% (C) Copyright Renaud Detry 2013.
% Distributed under the GNU General Public License.
% (See http://www.gnu.org/copyleft/gpl.html)

disp('Program started');
%Use the following line if you had to recompile remoteApi
vrep = remApi('remoteApi', 'extApi.h');
vrep=remApi('remoteApi');
vrep.simxFinish(-1);
id = vrep.simxStart('127.0.0.1', 19997, true, true, 2000, 5);

if id < 0,
disp('Failed connecting to remote API server. Exiting.');
```

```
vrep.delete();
return;
end
fprintf('Connection %d to remote API server open.\n', id);

% Make sure we close the connexion whenever the script is interrupted.
cleanupObj = onCleanup(@( ) cleanup_vrep(vrep, id));

% This will only work in "continuous remote API server service"
% See http://www.v-rep.eu/helpFiles/en/remoteApiServerSide.htm
res = vrep.simxStartSimulation(id, vrep.simx_opmode_oneshot_wait);
% We're not checking the error code - if vrep is not run in continuous remote
% mode, simxStartSimulation could return an error.
vrchk(vrep, res);

% Retrieve all handles, and stream arm and wheel joints, the robot's pose,
% the Hokuyo, and the arm tip pose.
h = youbot_init(vrep, id);
h = youbot_hokuyo_init(vrep, h);

% Let a few cycles pass to make sure there's a value waiting for us next time
% we try to get a joint angle or the robot pose with the simx_opmode_buffer
% option.
pause(.2);

% Constants:
timestep = .05;
wheelradius = 0.0937/2; % This value may be inaccurate. Check before using.

% Min max angles for all joints:
armJointRanges = [-2.9496064186096,2.9496064186096;
-1.5707963705063,1.308996796608;
-2.2863812446594,2.2863812446594;
-1.7802357673645,1.7802357673645;
-1.5707963705063,1.5707963705063 ];

startingJoints = [0,30.91*pi/180,52.42*pi/180,72.68*pi/180,0];

% In this demo, we move the arm to a preset pose:
pickupJoints = [90*pi/180, 19.6*pi/180, 113*pi/180, -41*pi/180, 0*pi/180];

% Tilt of the Rectangle22 box
r22tilt = -44.56/180*pi;

% Parameters for controlling the youBot's wheels:
forwBackVel = 0;
leftRightVel = 0;
rotVel = 0;

disp('Starting robot');

% Set the arm to its starting configuration:
res = vrep.simxPauseCommunication(id, true); vrchk(vrep, res);
for i = 1:5,
res = vrep.simxSetJointTargetPosition(id, h.armJoints(i),...
startingJoints(i),...
vrep.simx_opmode_oneshot);

vrchk(vrep, res, true);
end
res = vrep.simxPauseCommunication(id, false); vrchk(vrep, res);

plotData = true;
if plotData,
subplot(211)
drawnow;
[X,Y] = meshgrid(-5:.25:5,-5.5:.25:2.5);
X = reshape(X, 1, []);
Y = reshape(Y, 1, []);
end

% Make sure everything is settled before we start
pause(2);

[res homeGripperPosition] = ...
vrep.simxGetObjectPosition(id, h.ptip,...
h.armRef,...
vrep.simx_opmode_buffer);

vrchk(vrep, res, true);
fsm = 'rotate';

while true,
tic
if vrep.simxGetConnectionId(id) == -1,
error('Lost connection to remote API.');
```

```
end

[res youbotPos] = vrep.simxGetObjectPosition(id, h.ref, -1,...
vrep.simx_opmode_buffer);
```

```
vrchk(vrep, res, true);
[res youbotEuler] = vrep.simxGetObjectOrientation(id, h.ref, -1,...
vrep.simx_opmode_buffer);

vrchk(vrep, res, true);

if plotData,
% Read data from the Hokuyo sensor:
[pts contacts] = youbot_hokuyo(vrep, h, vrep.simx_opmode_buffer);

in = inpolygon(X, Y, [h.hokuyo1Pos(1) pts(1,:) h.hokuyo2Pos(1)],...
[h.hokuyo1Pos(2) pts(2,:) h.hokuyo2Pos(2)]);

subplot(211)
plot(X(in), Y(in), '-g', pts(1,contacts), pts(2,contacts), '*r',...
[h.hokuyo1Pos(1) pts(1,:) h.hokuyo2Pos(1)],...
[h.hokuyo1Pos(2) pts(2,:) h.hokuyo2Pos(2)], 'r',...
0, 0, 'ob',...
h.hokuyo1Pos(1), h.hokuyo1Pos(2), 'or',...
h.hokuyo2Pos(1), h.hokuyo2Pos(2), 'or');
axis([-5.5 5.5 -5.5 2.5]);
axis equal;
drawnow;

end
angl = -pi/2;

if strcmp(fsm, 'rotate'),
rotVel = 10*angdiff(angl, youbotEuler(3));
if abs(angdiff(angl, youbotEuler(3))) < 1/180*pi,
rotVel = 0;
fsm = 'drive';
end
elseif strcmp(fsm, 'drive'),
forwBackVel = -20*(youbotPos(1)+3.167);

if youbotPos(1)+3.167 < .001,
forwBackVel = 0;
vrep.simxSetObjectOrientation(id, h.rgbdCasing, h.ref,...
[0 0 pi/4], vrep.simx_opmode_oneshot);

for i = 1:5,
res = vrep.simxSetJointTargetPosition(id, h.armJoints(i), pickupJoints(i),...
vrep.simx_opmode_oneshot);

vrchk(vrep, res, true);
end
if plotData,
fsm = 'snapshot';
else,
fsm = 'extend';
end
end
elseif strcmp(fsm, 'snapshot'),
% Read data from the range camera

% Reading a 3D image costs a lot to VREP (vrep has to simulate the image). It
% also requires a lot of bandwidth, and processing a 3D point cloud (for
% instance, to find one of the boxes or cylinders that the robot has to
% grasp) will take a long time in Matlab. In general, you will only want to
% capture a 3D image at specific times, for instance when you believe you're
% facing one of the tables.

% Reduce the view angle to better see the objects
res = vrep.simxSetFloatSignal(id, 'rgbd_sensor_scan_angle', pi/8,...
vrep.simx_opmode_oneshot_wait);

vrchk(vrep, res);
% Ask the sensor to turn itself on, take A SINGLE 3D IMAGE,
% and turn itself off again
res = vrep.simxSetIntegerSignal(id, 'handle_xyz_sensor', 1,...
vrep.simx_opmode_oneshot_wait);

vrchk(vrep, res);

fprintf('Capturing point cloud...\n');
pts = youbot_xyz_sensor(vrep, h, vrep.simx_opmode_oneshot_wait);
% Each column of pts has [x;y;z;distancetosensor]
% Here, we only keep points within 1 meter, to focus on the table
pts = pts(1:3,pts(4,:)<1);
subplot(223)
plot3(pts(1,:), pts(2,:), pts(3,:), '*');
axis equal;
view([-169 -46]);

% Save the pointcloud to pc.xyz.
% (pc.xyz can be displayed with meshlab.sf.net).
fileID = fopen('pc.xyz','w');
fprintf(fileID,'%f %f %f\n',pts);
fclose(fileID);
fprintf('Read %i 3D points, saved to pc.xyz.\n', max(size(pts)));

% Read data from the RGB camera

% This is very similar to reading from the 3D camera. The comments in the 3D
% camera section directly apply to this section.

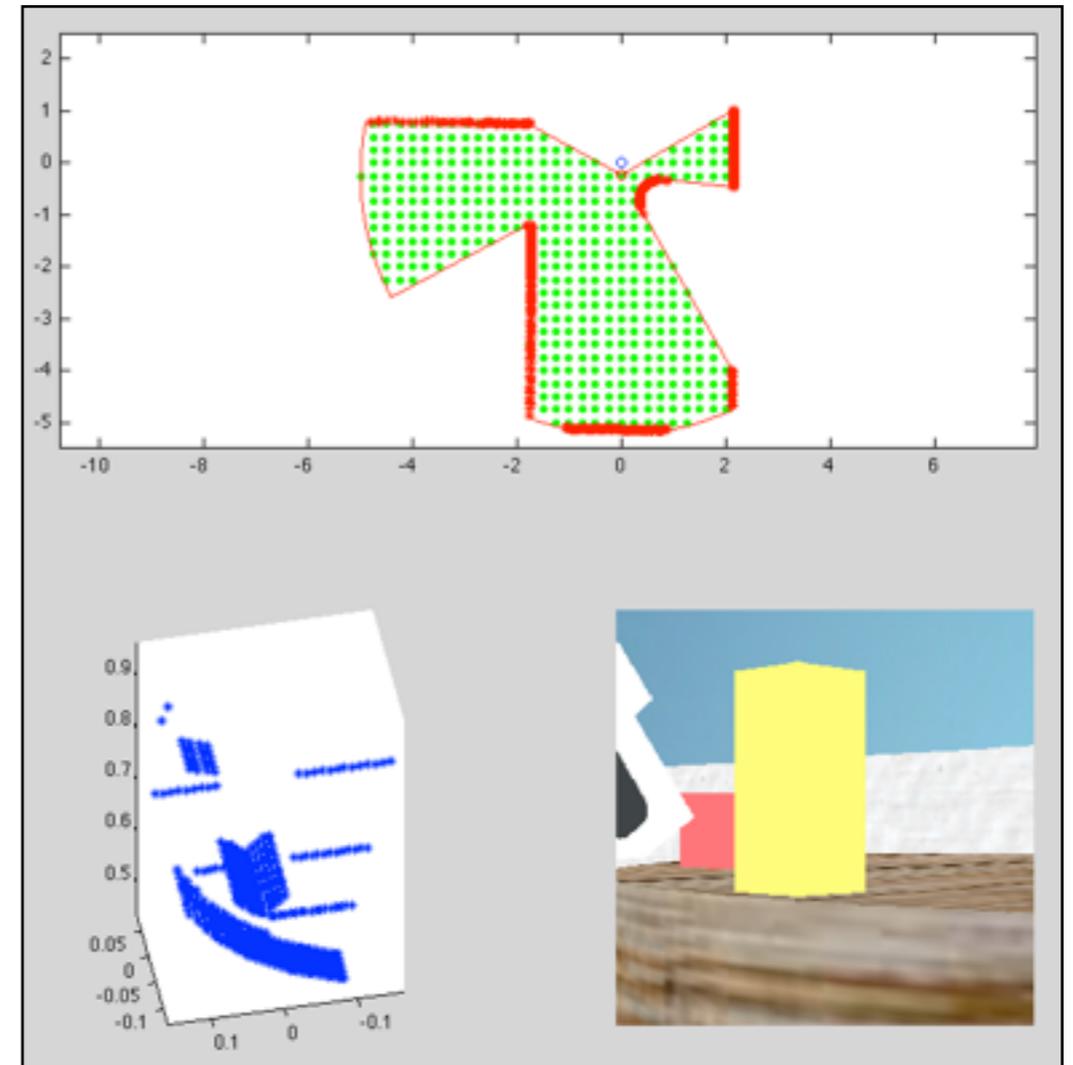
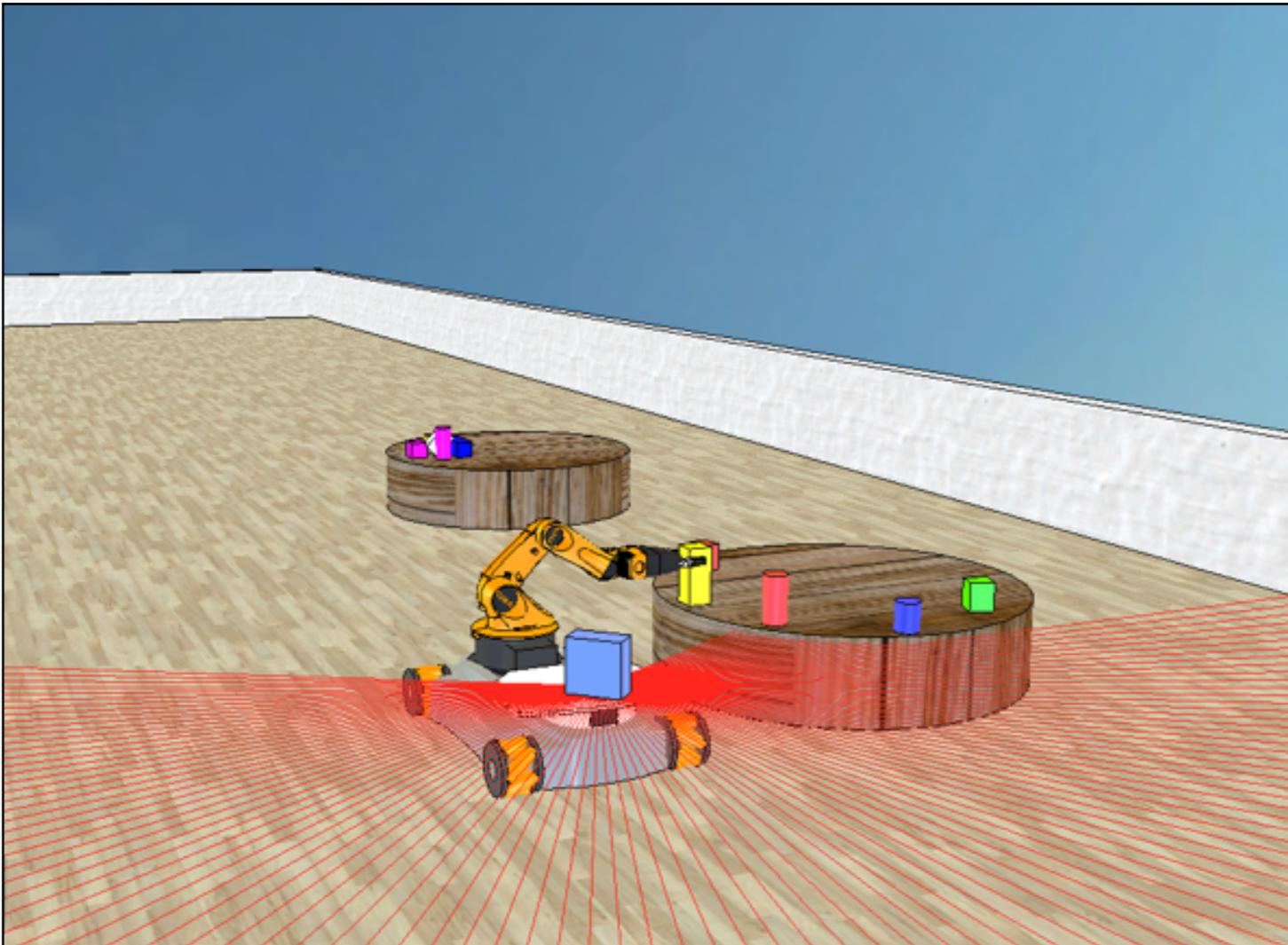
res = vrep.simxSetIntegerSignal(id, 'handle_rgb_sensor', 1,...
vrep.simx_opmode_oneshot_wait);

vrchk(vrep, res);
fprintf('Capturing image...\n');
[res resolution image] = ...
vrep.simxGetVisionSensorImage2(id, h.rgbSensor, 0,...
vrep.simx_opmode_oneshot_wait);

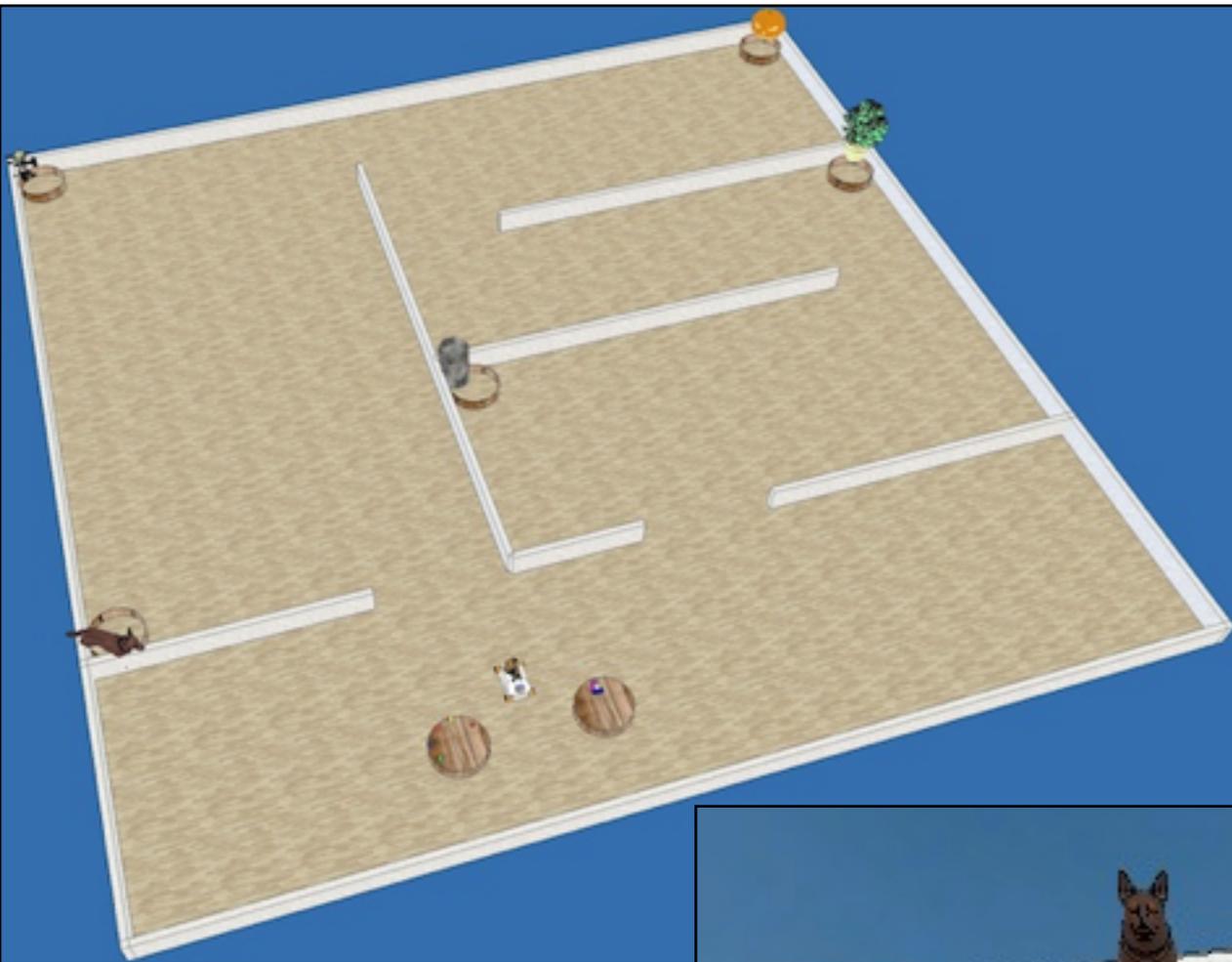
vrchk(vrep, res);
fprintf('Captured %i pixels.\n', resolution(1)*resolution(2));
subplot(224)
imshow(image);
drawnow;
fsm = 'extend';
elseif strcmp(fsm, 'extend'),
```

# Practice Navigation, Grasping, Vision in 5 Minutes

```
vrep = remApi('remoteApi', 'extApi.h');  
id = vrep.simxStart('127.0.0.1', 57689, true, true, 2000, 5);  
  
[res image] = vrep.simxGetVisionSensorImage(id, cam);  
  
vrep.simxSetJointTargetVelocity(id, wheel1, 10);
```



# Project Definition

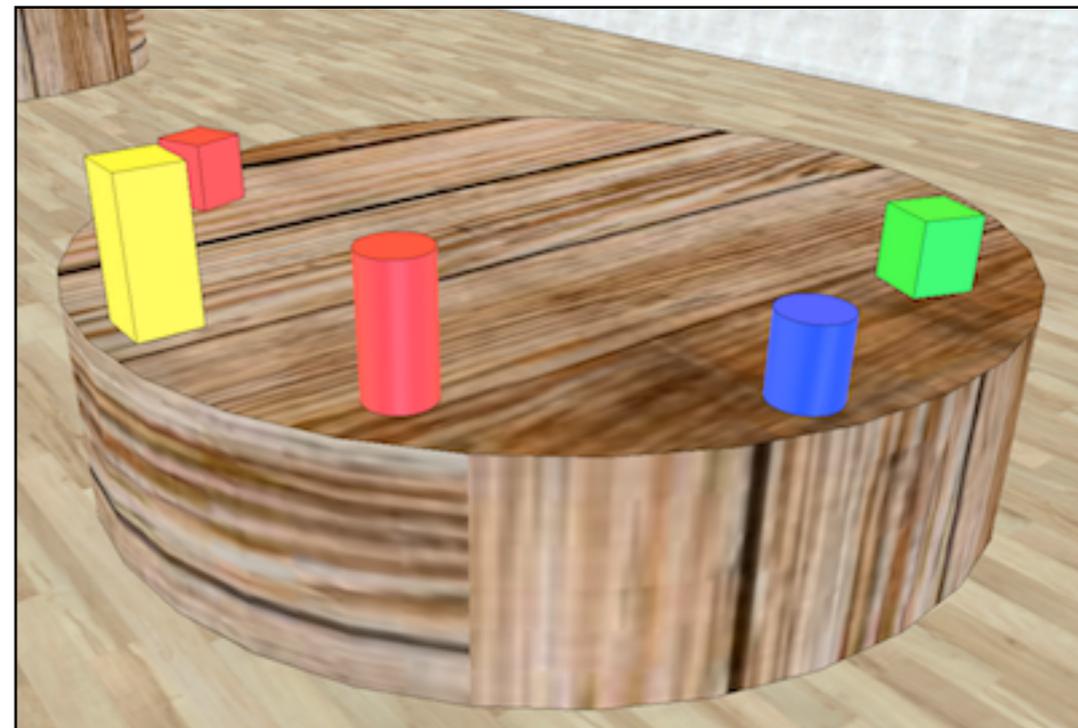
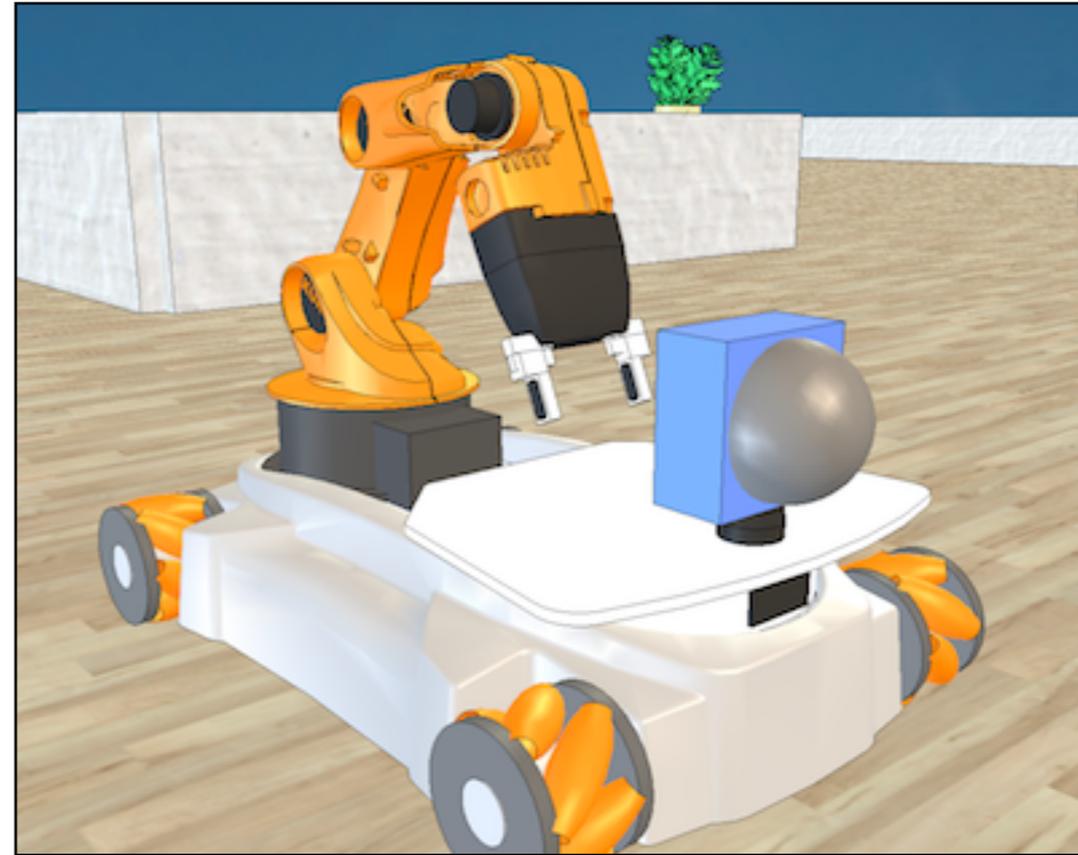


Aim of the project: put the objects five baskets distributed across the house



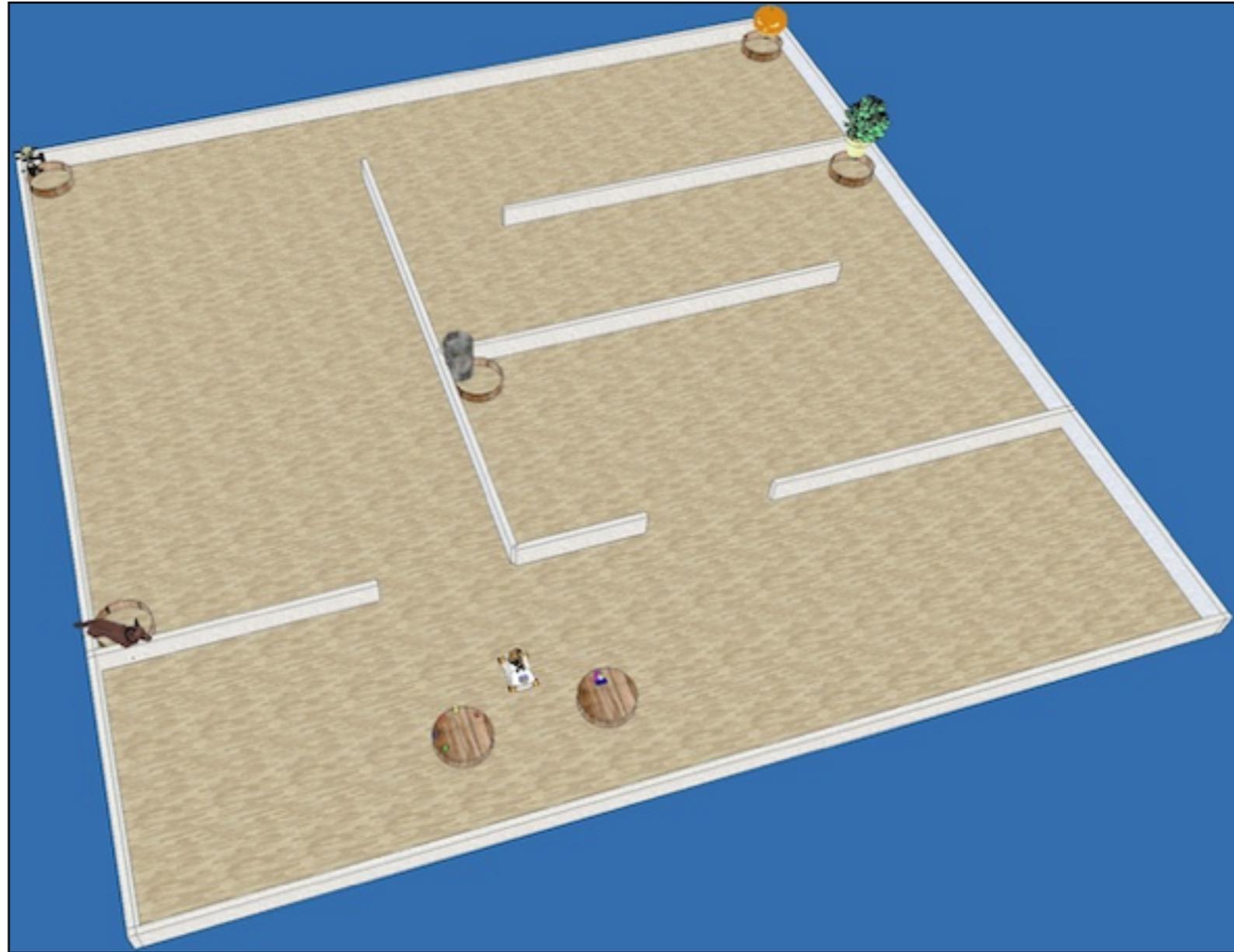
# Project Definition

- Object are either cylindrical or box-shaped.
- The bases of boxes and cylinders are fixed
- Objects are initially placed on two tables facing the youBot's starting position.
- One on one table, objects are placed upright. On the other table, stacked



# Project Definition

- Baskets are distributed around the house.
- There's a landmark object next to each basket.
- The landmark allow the robot to identify the room to which each basket belongs



# Project Definition

The robot has access to a list of instructions that tell into which basket each object must go:

- `inst(1).shape`: shape of the first object (either `box` or `cylinder`).
- `inst(1).color`: color of the first object (R, G, B values).
- `inst(1).picture`: path to an image of the landmark next to which is located the basket in which object 1 must go.
- `inst(2).shape`: shape of the second object (either `box` or `cylinder`).
- `inst(2).color`: color of the second object (R, G, B values).
- `inst(2).picture`: path to an image of the landmark next to which is located the basket in which object 2 must go.
- ...

# Milestone A: Navigation

Building a map of the house (a map of the walls and other obstacles).

1. Using accurate localization (via `simxGetObjectPosition` ON `youBot_ref` OR `youBot_center`)
2. Without using `simxGetObjectPosition` ON `youBot_ref` OR `youBot_center`.

This milestone covers the following topics:

- Navigation and mapping: with the help of RTB, students learn how to manage a map, how to handle exploration, and how to plan trajectories that avoid obstacles
- Control: TRS provides raw access to the youBot's wheels. A students learn to implement a controller that configures the robot's position and orientation in order to follow a trajectory.
- Poses and reference frames: students learn to move points and velocity vectors from the frame of the robot to the world frame and vice versa.

# Milestone B: Manipulation

Picking up objects and moving them to any room of the house (except the room where the objects start in) (not necessarily in a basket).

1. Moving only the objects from the first table (where objects stand upright), using V-REP IK. Objects can fall on the floor.
2. Moving all the objects (both tables), using V-REP IK. Objects can fall on the floor.
3. Moving all the objects (both tables), using V-REP IK. Objects *cannot* fall on the floor.
4. Moving all the objects (both tables), *without* using V-REP IK. Objects *cannot* fall on the floor.

This milestone covers the following topics:

- Articulated arms: students learn about forward and inverse kinematics.
- Vision/Fitting: locating objects, deciding where to place the gripper.

# Milestone C: Vision

Finding and identifying the baskets.

1. Finding the baskets and the tables.

2. Recognizing the landmark objects accompanying the baskets, based on the data from `instructions.mat`.

This milestone covers the following topics:

- Fitting: RANSAC or Hough, or other, to find the cylindrical baskets with the Hokuyo sensor
- Object recognition

# Milestone D: Manipulation+Vision

## Manipulation+Vision

- 1.Placing the objects into arbitrary baskets (as long as there is the same number of objects in each basket). (Requires at least B.1 and C.1.)
- 2.Placing the objects into the appropriate basket, as indicated by `instructions.mat`. (Requires at least B.1 and C.2.)

This milestone covers the following topics:

- Planning order and shortest paths for bringing the objects to the baskets.

# Milestone E: Calibration

Computing the transformation between the frame of the vision sensor, and the frame of the robot. (Without `simxGetObjectOrientation` on `rgbSensor`.)

# The Robot: Configuration Signals

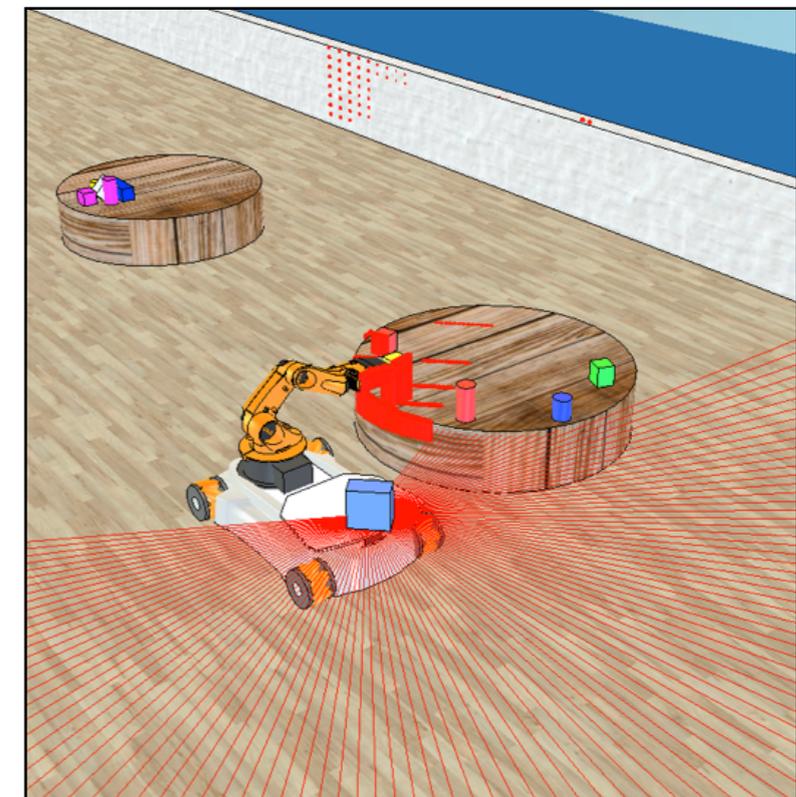
Turn vision sensors on/off:

```
handle_xyz_sensor, handle_xy_sensor, handle_rgb_sensor  
vrep.simxSetIntegerSignal(id, 'handle_rgb_sensor', 0)
```

The view angle of the depth camera and the RGB camera can be controlled via a signal named `rgbd_sensor_scan_angle`.

The `gripper_open` signal controls the gripper.

The `km_mode` signal turns the robot's inverse kinematics mode on or off.



# V-REP API: Authorized Calls

- [simxGetLastCmdTime](#)
  - [simxGetLastErrors](#)
  - [simxGetModelProperty](#)
  - [simxGetObjectChild](#)
  - [simxGetObjectFloatParameter](#)
  - [simxGetObjectGroupData](#)
  - [simxGetObjectHandle](#)
- Only for objects that are part of the robot.
- [simxGetObjectIntParameter](#)
  - [simxGetObjectOrientation](#)

Only with the following arguments:

<b>objectHandle</b>	<b>relativeToObjectHandle</b>	<b>Milestone</b>
rgbSensor	youBot_ref, youBot_center	all except E
xyzSensor	rgbSensor	(any)
rgbSensor	rgbSensor	(any)
fastHokuyo_sensor1	youBot_ref, youBot_center	(any)
fastHokuyo_sensor1	youBot_ref, youBot_center	(any)
youBot_ref, youBot_center	-1	all except A.2
youBot_gripperPositionTip	youBot_ref	all except B.4
youBot_gripperPositionTarget	youBot_ref	all except B.4
youBot_gripperOrientationTip	Rectangle22	all except B.4
youBot_gripperOrientationTarget	Rectangle22	all except B.4

- [simxGetObjectParent](#)
- [simxGetObjectPosition](#)  
See [simxGetObjectOrientation](#).
- [simxGetObjects](#)
- [simxGetObjectSelection](#)

# What You Get

- Recipe for organizing a Master-level robotics project
- The project: pickup groceries from a table and store them in baskets.
- Involves: control, navigation, mapping, vision and manipulation.

<http://teaching-robotics.org/trs>



**TRS:** An Open-source Recipe for Teaching/Learning Robotics with a Simulator.

:: setup a laptop in 5 minutes, write a control, navigation, vision or manipulation program in 100 lines of code

TRS

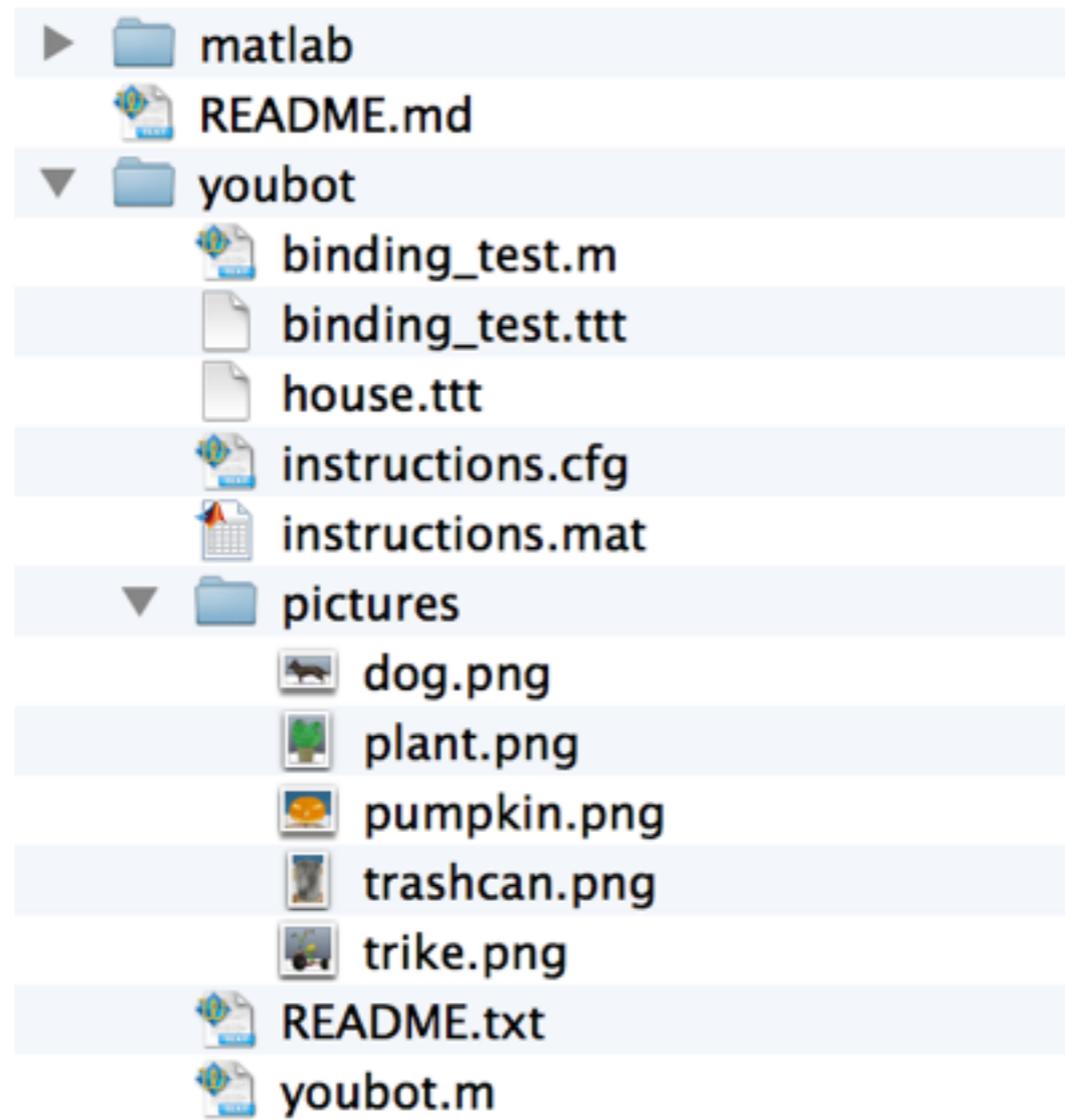
Motivation

**Motivation**

TRS is a cross-platform robot development and simulation environment that can

# How To Use

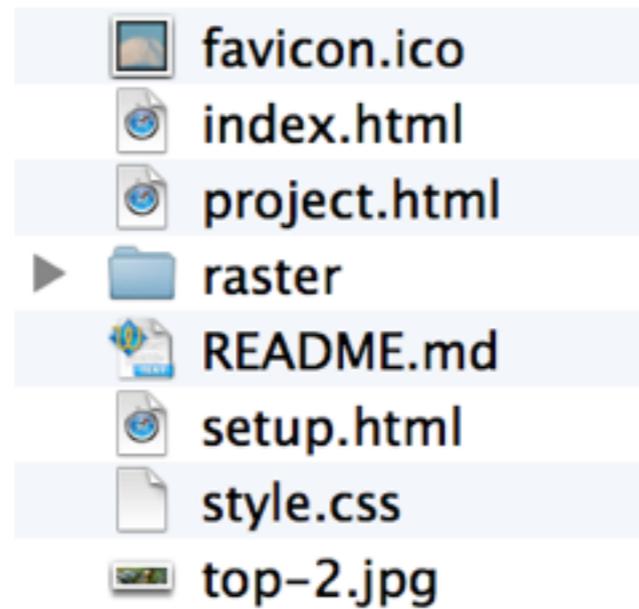
- Freely distributed via GitHub  
<https://github.com/ULgRobotics/trs.git>
- master branch: code & V-REP models (GPL)



- gh-pages branch: doc & project details (CC)

# How To Use

- Freely distributed via GitHub  
<https://github.com/ULgRobotics/trs.git>
- master branch: code & V-REP models (GPL)
- gh-pages branch: doc & project details (CC)

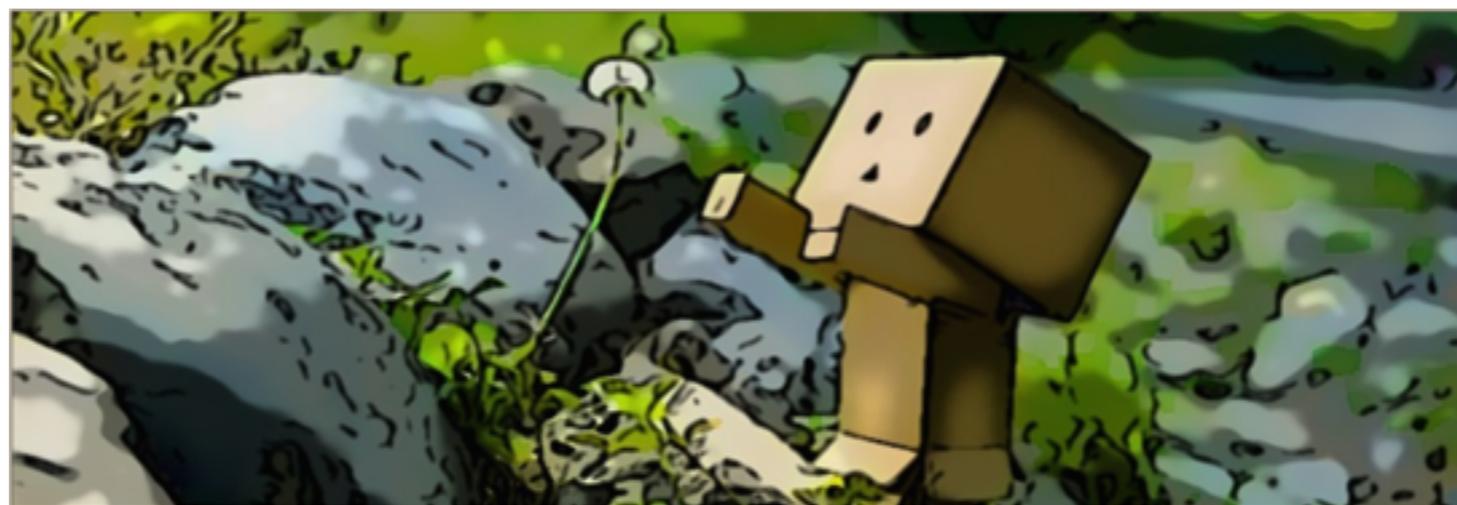


Github serves the gh-pages branch over http via [github.io](http://github.io)

# How To Use

- Freely distributed via GitHub  
<https://github.com/ULgRobotics/trs.git>
- master branch: code & V-REP models (GPL)
- gh-pages branch: doc & project details (CC)

<http://teaching-robotics.org/trs>



**TRS:** An Open-source Recipe for Teaching/Learning Robotics with a Simulator.

:: setup a laptop in 5 minutes, write a control, navigation, vision or manipulation program in 100 lines of code

TRS

Motivation

**Motivation**

TRS is a cross-platform robot development and simulation environment that can

<http://teaching-robotics.org/>

<http://roboticscourseware.org/>

**Teaching-Robotics.org:** a community-driven website for sharing resources related to teaching robotics to Master and PhD students.

Motivation

Ressources

Events

Contact

## Motivation

This website aims to offer community-driven resources related to teaching robotics.

## Ressources

TRS: An Open-source Recipe for Teaching/Learning Robotics with a Simulator.

## Events

Hong Kong, June 5 2014, at ICRA: [Workshop on using MATLAB/Simulink for Robotics Education and Research.](#)

Chicago, September 14 2014, at IROS: [Tutorial on teaching robotics with a simulator.](#)

## Contact

[Renaud Detry, University of Liege, Belgium](#)

# Upcoming Tutorials

Copenhagen, Denmark, July 9, at **ICVS**

Kobe, Japan, August 31 2015, at **Ro-Man**

Hamburg, Germany, September 2015, at **IROS**

# Learning Robotics with a Simulator: Setup Your Laptop in 5 min, Code a Control, Navigation, Vision or Manipulation

<http://teaching-robotics.org/trs>

