Interfacing Matlab/Simulink with V-REP for an Easy Development of Sensor-Based Control Algorithms for Robotic Platforms

Riccardo Spica, Giovanni Claudio,
Fabien Spindler and Paolo Robuffo Giordano

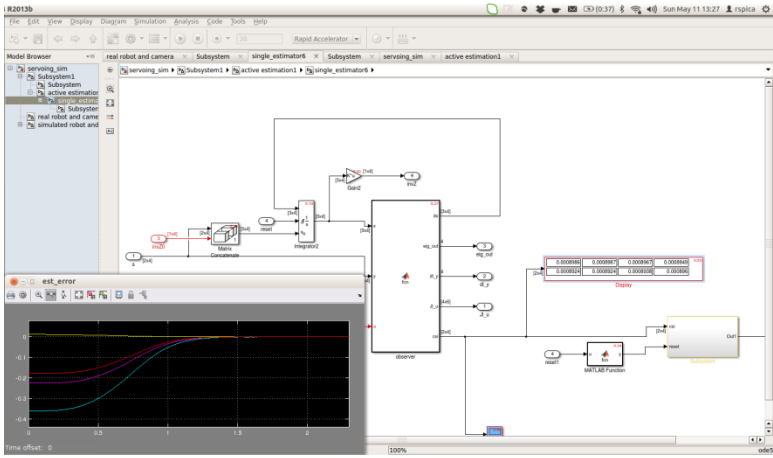
Lagadic group

Inria Rennes Bretagne Atlantique & IRISA

http://www.irisa.fr/lagadic

Why using Matlab/Simulink

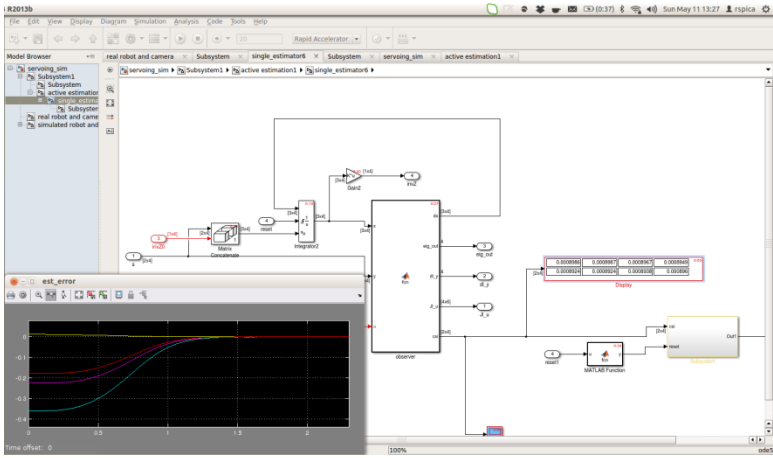
- fast prototyping (debugging+testing) complex control algorithms
  - debugging, scopes, displays
  - post-processing, plotting, logging
- powerful scripting language
- huge library + File Exchange



- integrate external C/C++ code (MEX files and s-function)
- automatic code generation
  - speed up execution time
  - deployment to other platforms (e.g., the robot itself)

Why using Matlab/Simulink

- fast prototyping (debugging+testing) complex control algorithms
  - debugging, scopes, displays
  - post-processing, plotting, logging
- powerful scripting language
- huge library + File Exchange



- integrate external C/C++ code (MEX files and s-function)
- automatic code generation
  - speed up execution time
  - deployment to other platforms (e.g., the robot itself)
- missing a rigid body dynamics simulator

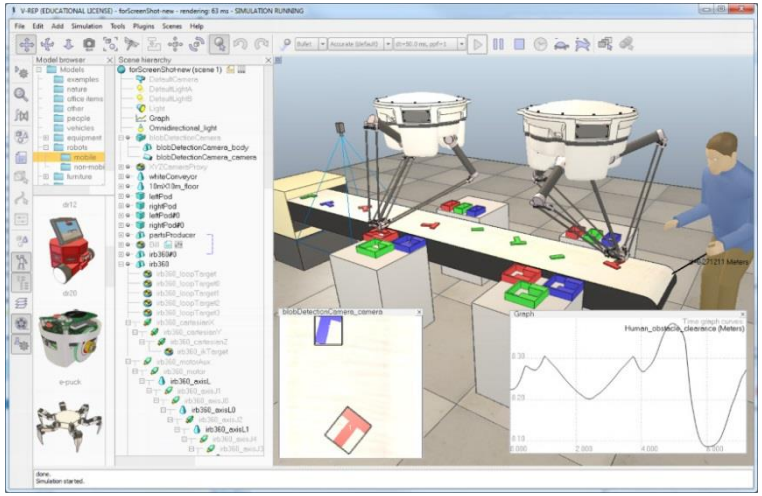
Why using a simulator

- easy testing and debugging algorithms
- no risk to damage real robots
- no need to have a real robot (useful for teaching)
- create different virtual environments or particular testing conditions
- testing faster than real time

Why using a simulator

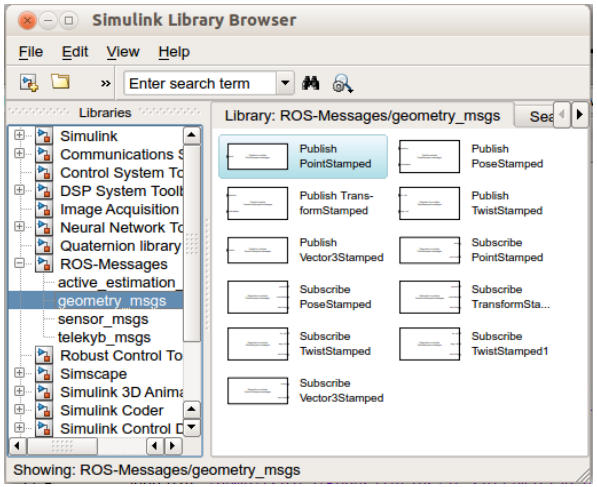
- easy testing and debugging algorithms
- no risk to damage real robots
- no need to have a real robot (useful for teaching)
- create different virtual environments or particular testing conditions
- testing faster than real time

- V-Rep
  - open source and free for academics
  - provides sensors, mechanisms, robots
  - algorithms for collision detection, planning, inverse kinematics,
  - support different simulation modes
  - highly customizable (e.g. C++ plugins)
  - support for ROS communication framework



Interfacing Matlab and V-Rep with ROS

- open source
- widely spread benefits support from a big developer community
- provides many standard drivers/algorithms
- provides additional tools for logging/plotting/debugging
- same code with the simulated and real robots (just change some nodes and topic name)
- native support by V-Rep + custom plugin vrep_ros_bridge
- official support for Matlab on the way but many unofficial solutions available (e.g. integrate ROS in C++ s-function)



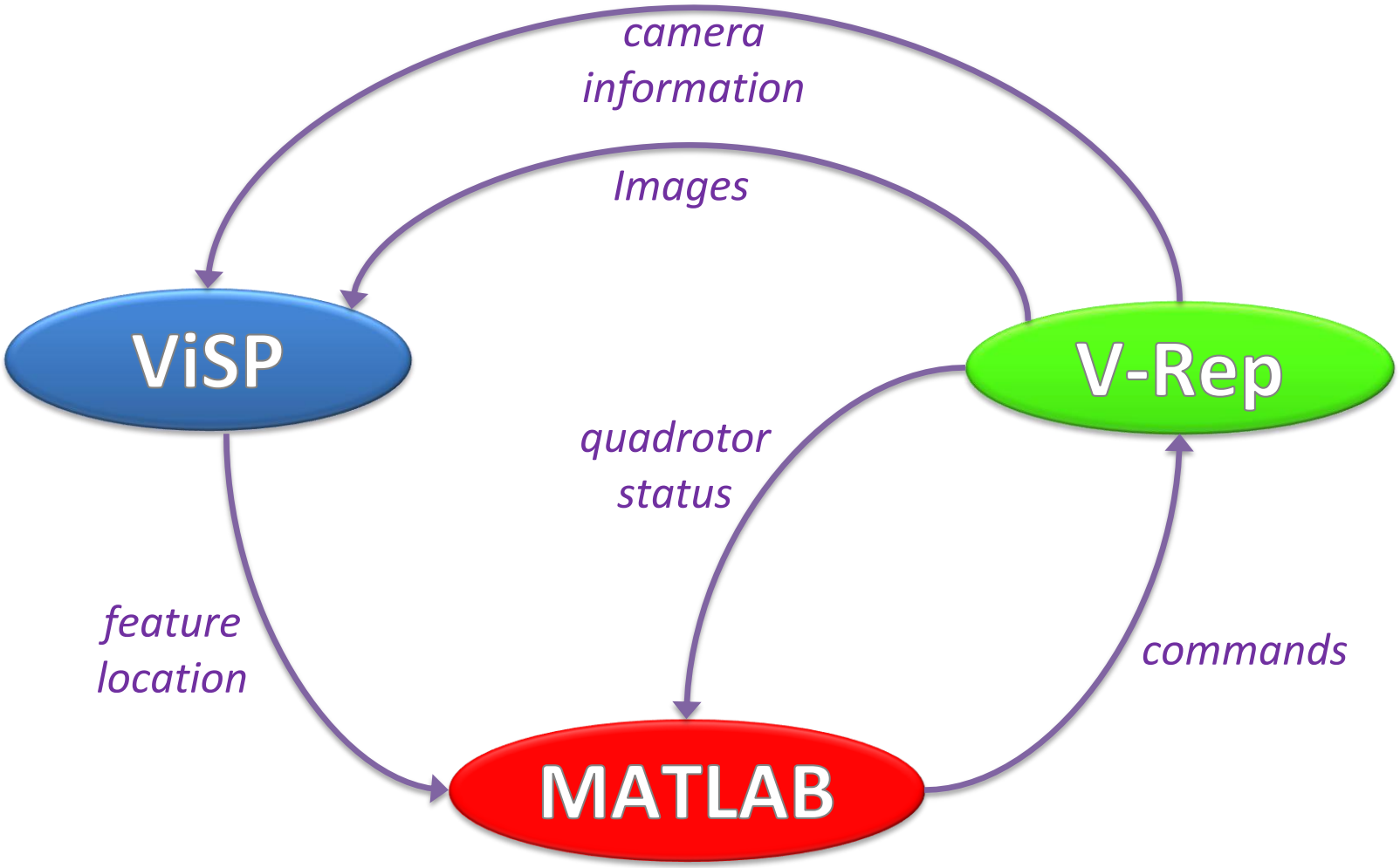
Synchronizing Matlab and V-Rep (?)

- synchronize both softwares with the system time (real time execution)

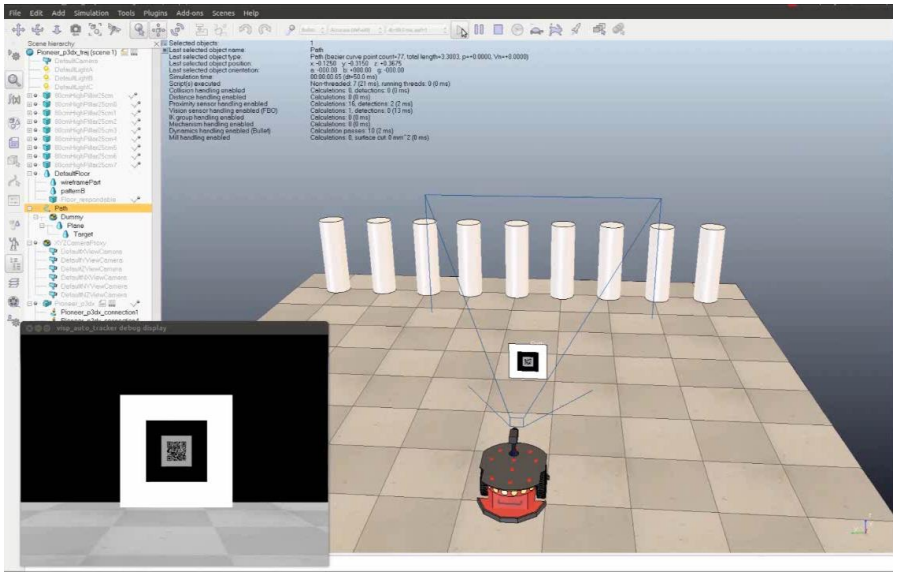
- V-Rep supports real-time and real-time-factor running

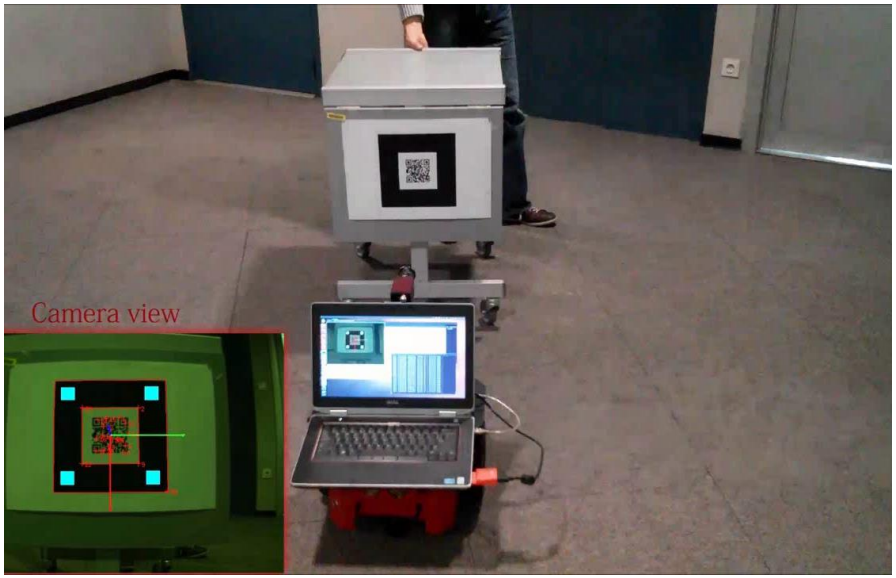
- in Matlab (my understanding):
  - official solution: use a real time target in code generation
  - our “quick & dirty” solution: use a custom s-function block that pauses the simulation to keep simulation time and system time synchronized

Demo: visual servoing for a quadrotor and a manipulator



Demo: visual servoing on simulated/real Pioneer





Links and Contacts

- Lagadic: http://www.irisa.fr/lagadic/
- ViSP: http://www.irisa.fr/lagadic/visp/
- vrep_ros_bridge:
  - ROS wiki: http://wiki.ros.org/vrep_ros_bridge
  - GIT Repository: https://github.com/lagadic/vrep_ros_bridge
  - Demo: http://www.irisa.fr/lagadic/demo/demo-vrep/

- Contacts:
  - Giovanni Claudio: giovanni.claudio@inria.fr
  - Riccardo Spica: riccardo.spica@inria.fr
  - Fabien Spindler: fabien.spindler@inria.fr