Autonomous and Mobile Robotics
Prof. Giuseppe Oriolo

An Introduction to V-REP with an Application to Motion Planning

Paolo Ferrari
outline

• introduction to V-REP
  • basic elements
  • dynamic modeling
  • C++ plugins
  • Matlab/Simulink interface
• application to motion planning
  • task-constrained motion planning with moving obstacles
  • problem formulation
  • approach
  • V-REP simulations
the V-REP simulator

• V-REP = Virtual Robot Experimentation Platform

• a robotic simulator: a software environment aimed at generic robotic applications (not only motion planning)

• relatively new (2014), produced by Coppelia Robotics

• free and open source

• available on Windows, Linux and Mac

• example of applications
  • fast prototyping and verification
  • fast algorithm development
  • hardware control
  • etc

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
the V-REP simulator

• provides physical engines for **dynamic** simulations

• allows the simulation of **sensors**

• its functionalities can be easily extended using **many programming languages** (C/C++, Python, Java, Lua, MATLAB, Octave, Urbi) and **programming approaches** (remote clients, plugins, ROS nodes, …)

• provides a large and continuously growing **library of robot models**

Oriolo: AMR - *An Introduction to V-REP with an Application to Motion Planning*
three central elements

- **scene objects** (how to build a robot)
  - basic building blocks
  - 12 different types
  - can be combined each other

- **calculation modules** (how to simulate a robot)
  - 5 basic modules
  - can be combined each other

- **control mechanisms** (how to control a robot)
  - 6 methods or interfaces
  - more than 7 programming languages
  - 6 methods can be used at the same time
scene objects

• how to **build** a robot
scene objects: basic components

• shapes
  • rigid mesh objects that are composed of triangular faces
  • can be grouped/ungrouped
  • different types (random, convex, pure shapes)

• joints
  • revolute: rotational movement
  • prismatic: translational movement
  • screw: translational while rotational movement
  • spherical: three rotational movements

• a robot model can be created through a hierarchical structure including (at least) shapes and joints
scene objects: sensors

- **proximity sensors**
  - more than simple ray-type detection
  - configurable detection volume
  - fast minimum distance calculation within volume

- **vision sensors**
  - render the objects that are in their field of view
  - embedded image processing
  - two different types: orthographic projection-type (e.g., close-range infrared or laser range finders) and perspective projection-type (e.g., camera-like sensors)

- **torque/force sensors**
  - measure applied force/torque (on 3 principal axes)
scene objects: other components

- paths/trajectories
  - allow 6D definitions
  - can be easily created
    (by importing a file or defining control points)
- cameras
  - perspective/orthographic projection
  - can track an object while moving
- lights: omnidirectional, spotlight, directional
- mirrors: reflect images/light, auxiliary clipping frame
- graphs: draw 3D curves, easily exportable
- mills: cutting operations
- dummies: auxiliary reference frame
calculation modules

- how to simulate a robot

Collision detection

Minimum distance calculation

Forward / Inverse kinematics

Physics / Dynamics

Path / motion planning

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
calculation modules

- forward/inverse kinematics
  - can be used for any kinematic chain (closed, redundant, …)
  - build inverse kinematic (IK) group
  - different techniques for inverse kinematics (pseudoinverse, damped least square)
  - accounts for joint limits and obstacle avoidance
- minimum distance computation
  - can be used between any pair of meshes
  - very fast and optimized
- collision detection
  - can be used between any pair of meshes
  - scene objects can be defined as collidable or not
calculation modules

• path/motion planning
  • can be performed for any kinematic chain
  • holonomic/non holonomic path planning
  • requires the specification of: start/goal configuration, obstacles, robot model
  • uses the OMPL library

• physics/dynamics
  • enable dynamic simulations (gravity, friction, …)
  • four different physical engines: Bullet, ODE, Vortex, Newton (ordered by computational demand)
  • dynamic particles to simulate air or water jets
control mechanisms

• how to control a robot
control mechanisms

- **local** approach: control entity is internal
  - embedded script: associated to a single robot
  - add-on: can only execute minimalistic code
  - **plugin**: most general tool, fast computation, written in C++

- **remote** approach: control entity is external
  - ROS node: bridge between V-REP and ROS
  - custom solution: client/server paradigm using the BlueZero framework
  - **remote API client**: communication between VREP and an external application (e.g., Matlab/Simulink)
dynamic modeling of a robot

• building a dynamic model in V-REP is very easy: no equations are needed

• it requires a few simple steps:
  • 1) import a CAD model of the robot
  • 2) associate to each body of the robot its dynamic parameters: mass, center of mass, inertia matrix

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
dynamic modeling of a robot

• 3) build a **model tree**: a tree that represents all hierarchical information of the kinematic chains (links and joints)
**C++ plugins**

- uses the V-REP regular APIs (more than 450 functions available)
- produces a shared library (e.g., .so for Linux and .dll for Windows)
- automatically loaded by V-REP at program start-up
- can be integrated with other C++ libraries (e.g., Eigen, Octomap, etc)
- two main applications
  - extend V-REP's functionality through user-written functions (e.g., motion planning algorithms, controllers, …)
  - used as a wrapper for running code written in other languages
- a single plugin can manage more than one robot
- fast execution (particularly suited for motion planning)

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
C++ plugins

• at each event in the V-REP interface, a corresponding message is sent to the plugin

• each message triggers the execution of a particular portion of the code in the plugin

before starting the simulation (“offline” functions)

during the simulation (“online” functions)
Matlab/Simulink interface

- uses the **V-REP remote APIs** (more than 100 functions available)
- an interface for sending/receiving commands to/from V-REP
- two main blocksets: **V-REP sink** and **V-REP source** respectively sends/reads values from V-REP joints

- **V-REP** and Matlab/Simulink times automatically synchronized
- Matlab/Simulink commands start/stop V-REP simulations

*Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning*
Matlab/Simulink interface

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
**summary**

- **V-REP main advantages**
  - very good online documentation
  - four different physics engines
  - most complete open source software for dynamic simulations
  - very good set of APIs and control mechanisms
  - very fast software development when one gets the V-REP structure

- **V-REP main drawbacks**
  - vision sensors have high computational payload
  - since it is a huge software, it is not so friendly at the beginning
  - multi-robot simulations have high computational payload
  - collision checking library is slow
task-constrained motion planning with moving obstacles

• consider
  • a robot whose configuration $q$ takes values in an $n$-dimensional configuration space
  • an assigned task path $y_d(s), s \in [s_{ini}, s_{fin}]$, that takes values in an $m$-dimensional task space
  • an environment populated by fixed and moving obstacles
• assume
  • the robot is redundant wrt the task ($n > m$)
  • obstacle trajectories are known
• the TCMP-MO problem consists in finding a feasible, collision-free configuration space trajectory that allows the robot to exactly execute the assigned task
problem formulation

• kinematic case
  • robot described by the kinematic-level model $\dot{q} = v$
  • generalized velocities can be expressed as $\dot{q} = \dot{s}q'$

• dynamic case
  • robot described in the Euler-Lagrange form $B(q)\ddot{q} + n(q, \dot{q}) = \tau$
  • generalized accelerations can be expressed as $\ddot{q} = \ddot{s}q'' + \dot{s}q'$

• a solution to the TCMP-MO problem consists of a configuration-space trajectory $q(t)$, composed by a path $q(s)$ and a time history $s(t)$, such that:
  • joint velocity limits (kinematic case) or joint velocity/torque limits (dynamic case) are respected
  • the assigned task path is continuously satisfied
  • collisions are always avoided

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
approach

• an offline randomized algorithm

• builds a tree, similarly to the RRT, to search for a solution in the planning space

  • a vertex contains a state of the robot and an associated time instant

  • an edge between two adjacent vertexes represents a feasible collision-free subtrajectory in the configuration space

• makes use of $N$ samples of the assigned path $\{y_1, \ldots, y_k, \ldots y_N\}$

• each sample is located in correspondence of a value of the parameter $s$ in the predefined sequence $\{s_1 = s_{ini}, \ldots, s_k, \ldots s_N = s_{fin}\}$

• the algorithm has been implemented as a C++ plugin for V-REP

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
planning for the kinematic case

- root the tree at \((q_{ini}, 0)\)
- iteratively
  - randomly select a sample \(y_{rand}\)
  - compute an IK solution \(q_{rand} = f^{-1}(y_{rand})\)
  - randomly assign to \(q_{rand}\) a time instant \(t_{rand}\)
  - search the closest vertex \((q_{near}, t_{near})\) to \((q_{rand}, t_{rand})\), and extract \(s_k\) associated to \(q_{near}\)
  - randomly choose two values of \(\dot{s}\) (forward/backward motions)
  - randomly choose \(\tilde{w}\)
  - numerically integrate (forward/backward motions)
    \[ q' = J^#(\pm y'_d + k_p e_y) + (I - J^#J)\tilde{w} \]
  - if no violation occurs, add new vertices and edges to the tree
planning for the dynamic case

- root the tree at \((q_{ini}, \dot{q}_{ini}, 0)\)
- iteratively
  - randomly select a sample \(y_{rand}\)
  - compute an IK solution \(q_{rand} = f^{-1}(y_{rand})\)
  - randomly assign to \(q_{rand}\) a time instant \(t_{rand}\) and a generalized velocity \(\dot{q}_{rand}\)
  - search the closest vertex \((q_{near}, \dot{q}_{near}, t_{near})\) to \((q_{rand}, \dot{q}_{rand}, t_{rand})\), and extract \(s_k\) associated to \(q_{near}\)
  - randomly choose two values of \(\ddot{s}\) (accelerating/decelerating motions)
  - randomly choose \(\ddot{z}\)
  - numerically integrate (accelerating/decelerating motions)
    \[
    q'' = J^\#(y_d'' - J'q' + k_p e_y + k_d e_y') + (I - J^#J)\ddot{z}
    \]
  - if no violation occurs, add new vertices and edges to the tree

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
V-REP simulations

A General Framework for Task-Constrained Motion Planning with Moving Obstacles

Massimo Cefalo, Giuseppe Oriolo

Robotics Lab, DIAG
Sapienza Università di Roma

February 2018

Oriolo: AMR - An Introduction to V-REP with an Application to Motion Planning
references


• V-REP Regular APIs, link: http://www.coppeliarobotics.com/helpFiles/en/apiFunctionListAlphabetical.htm

• M. Cefalo, G. Oriolo, “A general framework for task-constrained motion planning with moving obstacles”– Robotica, vol. 37, pp. 575-598, 2019