CourseWare Wiki

/ b181 / courses / b4m36uir / labs / lab01

NAVIGATION

- B4M36UIR
- Classification
- Tasks
- Laboratory Exercises
  - Lab01 - Introduction to V-REP and Open-Loop Robot Locomotion Control
  - Lab02 - Exteroceptive sensing and Reactive-based Obstacle Avoidance
  - Lab03 - Grid and Graph-based Path Planning Methods
  - Lab04 - Incremental Path Planning
  - Lab05 - Randomized Sampling-based Algorithms
  - Lab06 - Curvature-constrained local planning in RRT
  - Lab07 - Asymptotically Optimal Randomized Sampling-based Path Planning
  - Lab08 - Multi-goal path planning and data collection path planning - TSP-like formulations
  - Lab09 - Data Collection Path Planning with Remote Sensing (TSPN)
  - Lab10 - Curvature-constrained Data Collection Path Planning ((D)TSPN)
  - Lab11 - Game Theory in Robotics
  - Lab12 - Game Theory in Robotics contd.
  - Lab13 - Game Theory in Robotics contd.
- Lectures
- For students
- Tutorials

ALL COURSES

- Winter 2018 / 2019
- Summer 2017 / 2018
- Older

B4M36UIR & BE4M36UIR

−Table of Contents

Lab01 - Introduction to V-REP and Open-Loop Robot Locomotion Control

Motivations and Goals
Become familiar with the V-REP robotic simulator
Be able to control hexapod walking robot
Tasks (teacher)
Familiarize with a simple "nature inspired" robot locomotion to control a hexapod walking robot
Task01(**3 Points**) Create a "motion primitive" function to abstract the robot motion control
Lab resources
Task01 resource package

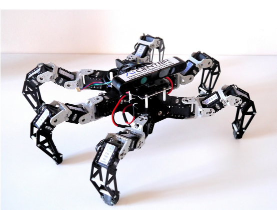
Robotic Simulator V-REP

V-REP is a powerful cross-platform 3D simulator based on a distributed control architecture: control programs (or scripts) can be directly attached to scene objects and run simultaneously in a threaded or non-threaded fashion. It features advanced physics engines which allows to simulate real-world physics and object interactions (collisions, object dynamics, etc.).

V-REP control methods
V-REP Python remote API documentation
V-REP C++ remote API documentation

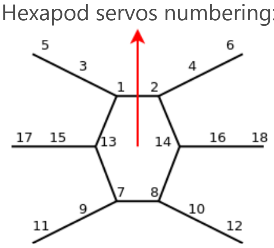
V-REP Python remote API tutorial

Hexapod model for V-REP



Hexapod model

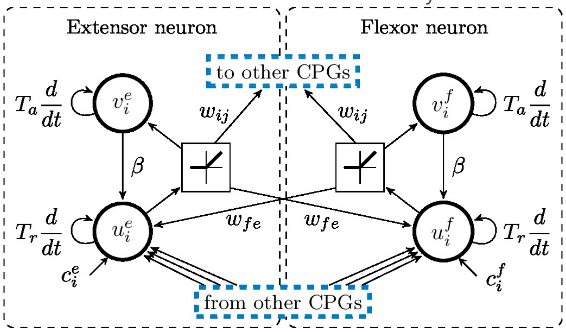
Hexapod servos numbering:



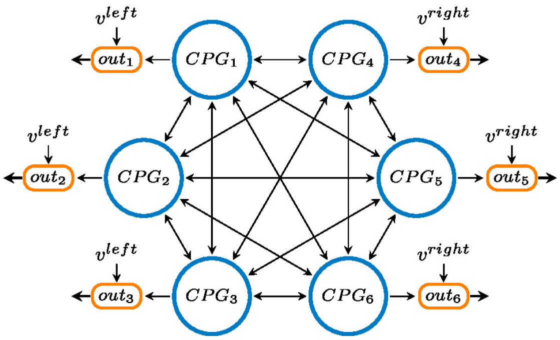
Multi-legged Robot Locomotion Using Central Pattern Generator

Central Pattern Generator (CPG) is a biologically inspired neural network that produce rhythmic patterned outputs [1]. CPGs are composed from individual neurons connected by mutual inhibition. The most used model and structure of the CPG is a Matsuoka oscillator [2]. The main problem is parameter tweaking of individual connections to achieve limit cycles in the neural network [3]

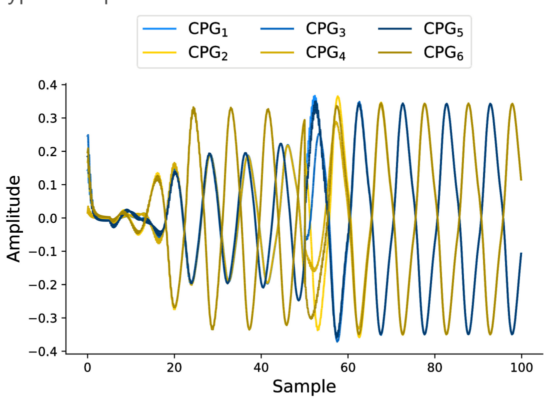
We will use a Matsuoka oscillator formed by four neurons in mutual inhibition.



The CPGs are connected in a network where each leg is driven by one CPG [4]. The purpose of the CPG network is to synchronize the oscillations of the connected CPGs to produce a motion gait. Output of each CPG is shaped to produce signals that drive individual joint actuators.



Typical output of the CPG network and the transition between different gaits.



Translation of the CPG output on the actuators can be done directly [1], or using the post-processing of the signal and inverse kinematics [4] [5] to calculate the foot-tip trajectories. In our work we are using the direct approach to map the CPG output to the joint angles. The output shaping blocks are fed with the control signal, which determines the speed of the robot legs on right and left sides respectively, which enables the robot with the differential steering.

Open-loop Control of the Robot in V-REP (3 Points)

In intelligent robotics the vital task for the robot is the navigation. Hence, the robot has to be aware of its position with respect to the goal and then find a suitable way to achieve it. In this course we are interested mostly in the artificial intelligence and planning; hence, the localization is provided in 6 Degrees Of Freedom (DOF) in global coordinates externally.

The open-loop control considers only the current position (x, y) and orientation ϕ of the robot which means that the current state of the robot is described as triplet (x, y, ϕ). The goal position is defined by its position (x_{goal}, y_{goal}). Besides, it may be beneficial to adjust the robot heading when it arrives at the goal position to, e.g., perform an inspection task. Therefore, the goal location is described by a triplet $(x_{goal}, y_{goal}, \phi_{goal})$.

Note that the real robot can never reach the precise goal position; hence, a small neighborhood of the goal position with a diameter of usually a half size of the robot is considered as suitable for navigation.

1)

A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: A review", In Neural Networks, Volume 21, Issue 4, 2008, Pages 642-653

2)

K. Matsuoka, "Sustained oscillations generated by mutually inhibiting neurons with adaptation." Biological cybernetics, Volume 52, Issue 6, 1985, Pages 367-376.

3)

L. Righetti, A. J. Ijspeert, "Design methodologies for central pattern generators: an application to crawling humanoids." Proceedings of robotics: Science and systems, 2006, Pages 191-198

4)

G. Zhong, L. Chen, Z. Jiao, J. Li and H. Deng, "Locomotion Control and Gait Planning of a Novel Hexapod Robot Using Biomimetic Neurons," in IEEE Transactions on Control Systems Technology, Volume PP, Number 99, Pages 1-13

5)

P. Milička, P. Čížek, and J. Faigl, “On chaotic oscillator-based central pattern generator for motion control of hexapod walking robot,” in ITAT, CEUR Workshop Proceedings, Volume 1649, 2016, Pages 131–137.

courses/b4m36uir/labs/lab01.txt · Last modified: 2018/10/01 01:31 by cizekpe6