SnakeSIM: a *ROS*-based Control and Simulation Framework for Perception-Driven Obstacle-Aided Locomotion of Snake Robots*

Filippo Sanfilippo[1,2], Øyvind Stavdahl[1] and Pål Liljebäck[1]

***Abstract*— Biological snakes are capable of exploiting roughness in the terrain for locomotion. This feature allows them to adapt to different types of environments. Snake robots that can mimic this behaviour could be fitted with sensors and used for transporting tools to hazardous or confined areas that other robots and humans are unable to access. Snake robot locomotion in a cluttered environment where the snake robot utilises a sensory-perceptual system to perceive the surrounding operational environment for means of propulsion can be defined as *perception-driven obstacle-aided locomotion* (POAL). The initial testing of new control methods for POAL in a physical environment using a real snake robot imposes challenging requirements on both the robot and the test environment in terms of robustness and predictability. This paper introduces *SnakeSIM*, a virtual rapid-prototyping framework that allows researchers for the design and simulation of POAL more safely, rapidly and efficiently. *SnakeSIM* is based on the Robot Operating System (ROS) and it allows for simulating the snake robot model in a virtual environment cluttered with obstacles. The simulated robot can be equipped with different sensors. Tactile perception can be achieved by using contact sensors to retrieve forces, torques, contact positions and contact normals. A depth camera can be attached to the snake robot head for visual perception purposes. Furthermore, *SnakeSIM* allows for exploiting the large variety of robotics sensors that are supported by ROS. The framework can be transparently integrated with a real robot. To demonstrate the potential of *SnakeSIM*, a possible control approach for POAL is considered as a case study.**

***Index Terms*— perception-driven obstacle-aided locomotion, snake robots, rapid-prototyping, ROS.**

I. Introduction

Biological snakes may push against rocks, stones, branches, obstacles, or other irregularities in the terrain for locomotion, which allows them to be remarkably adaptable to different types of environments. Snake robots that can mimic this variety of behaviour could open up a variety of possible applications for use in challenging real-life operations, such as explorations of earthquake-hit areas, pipe inspections for the oil and gas industry, fire-fighting operations, and search-and-rescue activities. Snake robot locomotion in a cluttered environment where the snake robot utilises a sensory-perceptual system to exploit the surrounding operational space and identifies walls, obstacles, or other external objects for means of propulsion can be defined as *perception-driven obstacle-aided locomotion* (POAL) [1], [2]. The underlying idea is shown in Fig. 1. The snake robot exploits the environment for locomotion by using augmented information: potential push-points are shown as cylinders, while achievable contact reaction forces are illustrated by arrows.

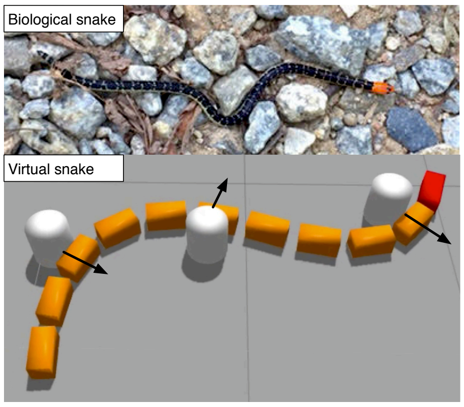


Fig. 1: Biological snakes push against salient features in the environment (top). Illustration of the same principle applied to a virtual snake robot (bottom).

The development of POAL is known to be challenging because of the complex interaction between the snake robot and the immediate environment. Furthermore, testing new control methods for POAL in a real setup environment is very difficult because potential collisions may damage both the robot and the environment. This process may also be time consuming. In contrast, a realistic simulator framework may enable researchers to develop control algorithms for POAL in a practical, efficient and safe simulation setup. Robotic simulators are commonly used in the design and testing of control algorithms. Related to this, the Robot Operating System (ROS) [3] has emerged as a de facto standard for robot software architecture in the research community in recent years. The primary goal of ROS is to provide a common platform to make the design of capable robotic applications quicker and easier. Some of the features it provides include hardware abstraction, device drivers, message-passing and package management. In conjunction with ROS, Gazebo 3D simulator [4] can be adopted to accurately and efficiently simulate robots in complex indoor and outdoor environments. Gazebo also provides a robust physics engine, high-quality graphics, and convenient programmatic and graphical inter-

*This work is supported by the Research Council of Norway through the *Young research talents* funding scheme, project title "SNAKE - Control Strategies for Snake Robot Locomotion in Challenging Outdoor Environments", project number 240072.

[1]F. Sanfilippo, Ø. Stavdahl and P. Liljebäck are with the Dept. of Engineering Cybernetics, Norwegian University of Science and Technology (NTNU), 7491 Trondheim, Norway. filippo.sanfilippo@ntnu.no

[2]F. Sanfilippo is aslo with the Dept. of Science and Industry Systems, Faculty of Technology, Natural Sciences and Maritime Sciences, University of Southeast Norway (USN), Post box 235, 3603 Kongsberg, Norway.

faces. In this perspective, ROS serves as the interface for the robot model, while Gazebo is used to simulate both the robot and its operational environment.

Even though ROS and Gazebo provide advanced features for general robotic applications, a comprehensive collection of tools, libraries, and conventions specifically designed for rapid-prototyping [5] POAL is still missing. The main contribution of this work is the development of *SnakeSIM*, a rapid-prototyping framework for POAL, and the integration of this missing technology with ROS. *SnakeSIM* makes it possible to simulate the snake robot model in a virtual environment cluttered with obstacles. Different sensors can be added to the robot. Contact sensors can be adopted to retrieve forces, torques, contact positions and contact normals so that tactile perception can be achieved. A depth camera can be attached to the snake robot head for visual perception purposes. Moreover, *SnakeSIM* allows for exploiting the large variety of robotics sensors that are supported by ROS. The framework can be transparently integrated with a real robot. This integrated framework will enable researchers to develop control algorithms for POAL more safely, rapidly and efficiently. To demonstrate the potential of *SnakeSIM*, a possible control approach for POAL is considered as a case study in this paper.

The paper is organised as follows. A review of the related research work is described in Section II. *SnakeSIM*, the proposed virtual rapid-prototyping framework for POAL of snake robots is presented in Section III. As a case study, a novel control algorithm for POAL is described in Section IV. In Section V, related simulation are carried out to validate the proposed framework. Finally, conclusions and future work are outlined in Section VI.

II. Related research works

The application of software rapid-prototyping methods for POAL has been quite limited in existing literature. One example of such work is the general motion planning framework for body shape control of snake robots developed by our research group [6]. The applicability of the framework was demonstrated for two-dimensional straight-line path following control, and for implementing body shape compliance in environments with obstacles. In [7], the same framework was extended to motion in three-dimensional space. However, the proposed framework still adopts a quite simplified representation of the environment and does not provide specifically designed tools for rapid-prototyping.

In [8], [9], a simulation tool was proposed, namely the *Modular Snake Robot Simulator*. This tool is a rigid-body-dynamics based physics simulator, which features several customisations such as body dimensions, actuator control and response functions, and environment setup, among others. Similarly, a physics-based simulation system for development and optimisation of snake robot locomotion patterns was proposed in [10]. The system provides a graphical user interface (GUI), which allows for interaction with the simulation at runtime and for supervising it. Long-term optimisations can be performed as background processes without GUI. However, most of these previous works mainly provide general tools for snake robot locomotion, while an exhaustive selection of tools, libraries, and conventions specifically designed for rapid-prototyping POAL is still lacking.

To the best of our knowledge, an integrated rapid-prototyping framework for designing and testing effective control algorithms for POAL is still missing.

III. Framework architecture

A. Design guidelines

When considering the design guidelines to develop the proposed rapid-prototyping framework, the following requirements were taken into account:

- flexibility: the framework must offer the possibility of collecting different sensor information from the simulated scene;
- reliability: as a research tool, the framework must be easy to maintain, modify and expand by adding new components and features;
- integrability: the framework must allow for transparent integration with real robots in the future.

For these reasons, ROS [3] is adopted as a common platform for implementing the proposed rapid-prototyping framework and as the interface for the snake robot model. Together with ROS, the Gazebo 3D simulator [4] is adopted to facilitate seamless simulations. Gazebo is one of the most-known simulators among researchers for robotic applications. Gazebo is designed to accurately reproduce the dynamic environments a robot may encounter. All simulated objects have mass, velocity, friction, and numerous other attributes that allow them to behave realistically when pushed, pulled, knocked over, or carried. These features can be used as integral parts of the proposed framework. In addition to ROS and Gazebo, the RViz (ROS visualisation) [11] visualisation tool is adopted to visualise and monitor sensor information retrieved in real-time from the simulated scenario. In particular, the normal and tangent vectors to the snake robot at the contact points are visualised and continuously updated according to the simulated scenario. The integration between ROS, Gazebo and RViz is shown in Fig. 2.

B. Hierarchical organisation

The proposed control framework is hierarchically organised, as shown in Fig. 3. Consideing the standard functions and capabilities of guidance, navigation, and control (GNC) [12], the following abstraction levels are defined:

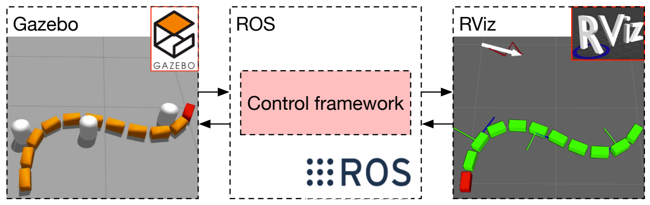


Fig. 2: The integration between ROS, Gazebo and RViz.

- Guidance: this level is responsible for achieving the functions of sensing, mapping and localisation. The snake robot's sensory-perceptual data are used to produce a representation of the surrounding environment. This level performs parsing or segmenting of low-level sensor data (e.g. point-clouds) into higher-level and more manageable information, including positions of the obstacles, their geometries and other relevant attributes, as well as the positions, directions and magnitudes of any relevant contact forces between the robot and the environment;
- Navigation: this level is responsible for decision making in terms of where, when and how the robot should ideally move. External system commands (e.g. preset mission objectives, a joystick operated by a human operator or an external system) and the snake robot's perception data provide the input to this level. The expected output from this level is the robot's desired trajectory (e.g. path and velocity information);
- Control: this level is the core of the proposed control framework. It enables researchers to develop their own alternative control method for POAL. The level does not impose any limitation regarding its internal implementation. However, each possible control method must be compliant with the provided interfaces of the framework. The inputs to this level are the desired trajectory, as well as any relevant information from the above guidance level (sensor data). The objective of the control level is to derive the required setpoints for the individual snake robot actuators in order to follow the desired trajectory. This control action will preferably be based on the high-level informtion from the guidance level, but lower-level information like the actual position and magnitude of contact forces might be necessary depending on the actual algorithm employed in the control level.

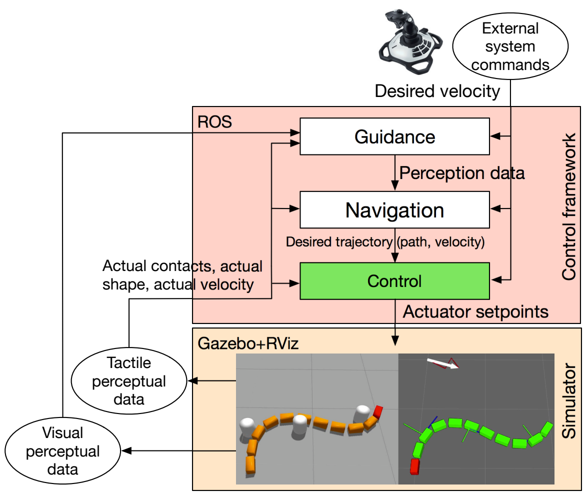


Fig. 3: The proposed framework architecture for *SnakeSIM*.

C. UML diagrams

On the base of the selected design guidelines and by considering the system requirements, the Unified Modeling Language (UML) [13] is adopted to better highlight the main features of the snake robot that is intended to be modelled withing *SnakeSIM*. Note that the UML is used hereafter to present blueprints of the snake robot (the real implementation of the system may differ).

A UML use case diagram is shown in Fig. 4 to describe the set of processes (use cases) that an external system or a human operator (subject) should or could perform in collaboration with the proposed framework. In particular, an "External system/human operator" operates the snake robot, which performs POAL. To achieve POAL, the action of guidance (i.e. sensing, mapping, localising), navigation (i.e. path planning, trajectory planning) and control (i.e. calculate actuator setpoints) are required.

A UML calss diagaram is shown in Fig. 5 to describe the structure of the proposed framework by showing the considered classes, their attributes, operations (or methods), and the relationships among objects. The "External system/human opeator interface" makes it possible to set the desired velocity of the snake robot through the "Guidance" component of the "Robot interface", which also include the "Navigation" and "Control" classes. The "Sensor interface" is responsable for sensing from both visual and tactile sensors, while, the "Actuator interface" is responsable for actuaing the joint actuators.

A UML state machine diagram is shown in Fig. 6 to describe the discrete behavior of the designed framework through finite state transitions. The snake robot can be into four main states: "PowerOff", "Idle", "Not locomoting", and "Locomoting" (locomotion can be achieved only when there are at least three simulataneos push-points, see Section IV). The most interesting state is the "Locomoting" state, which show the processes of guidance, navigation and control.

D. ROS implementation

As highlighted in Fig. 3, the proposed control framework is implemented in ROS, while Gazebo is adopted to provide seamless simulations, and RViz is used to visualise and monitor sensor information retrieved in real-time from the simulated scenario.

1) Simulated scenario: To carry out our experiment, a simulation scenario is built in Gazebo reproducing a cluttered environment. In particular, cylindrical objects or other shapes are placed in the scene and used as obstacles.

2) Snake robot model: To take full advantage of ROS, the snake robot model is implemented according to the Universal Robotic Description Format (URDF) [14]. The URDF is a specific file format used in ROS to describe all elements of a robot, such as links, joints, actuators and sensors. Simulated controllers are then adopted to actuate the joints of the snake robot. Simulated contact sensors are used to retrieve collisions with obstacles.

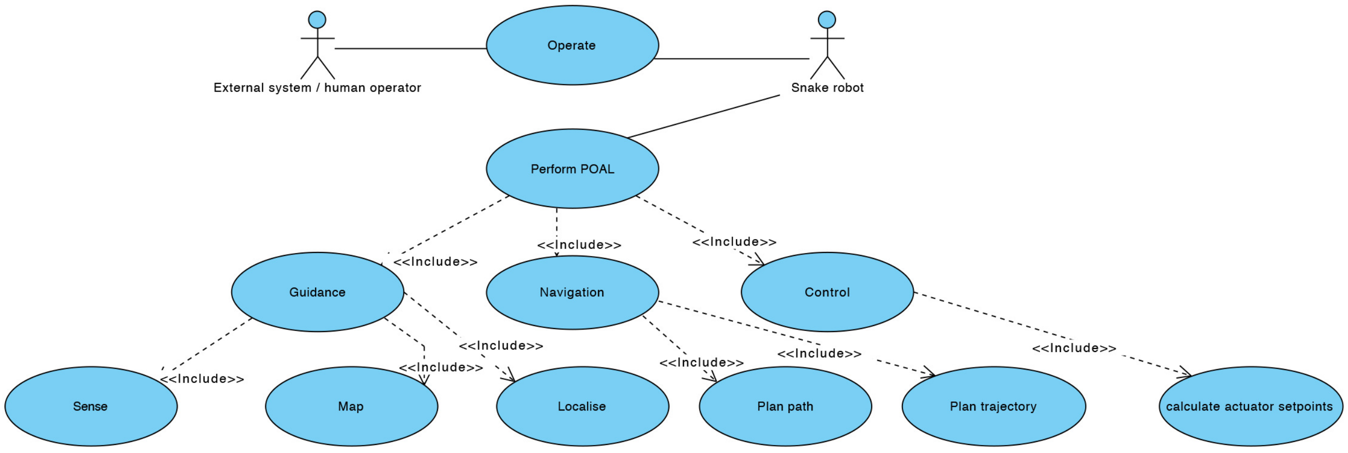


Fig. 4: The considered use case diagram.

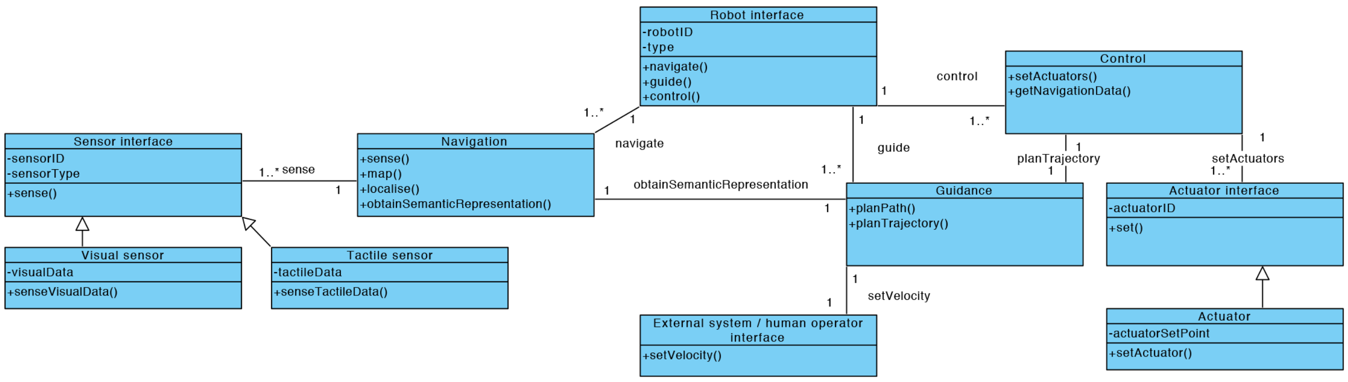


Fig. 5: The considered class diagram.

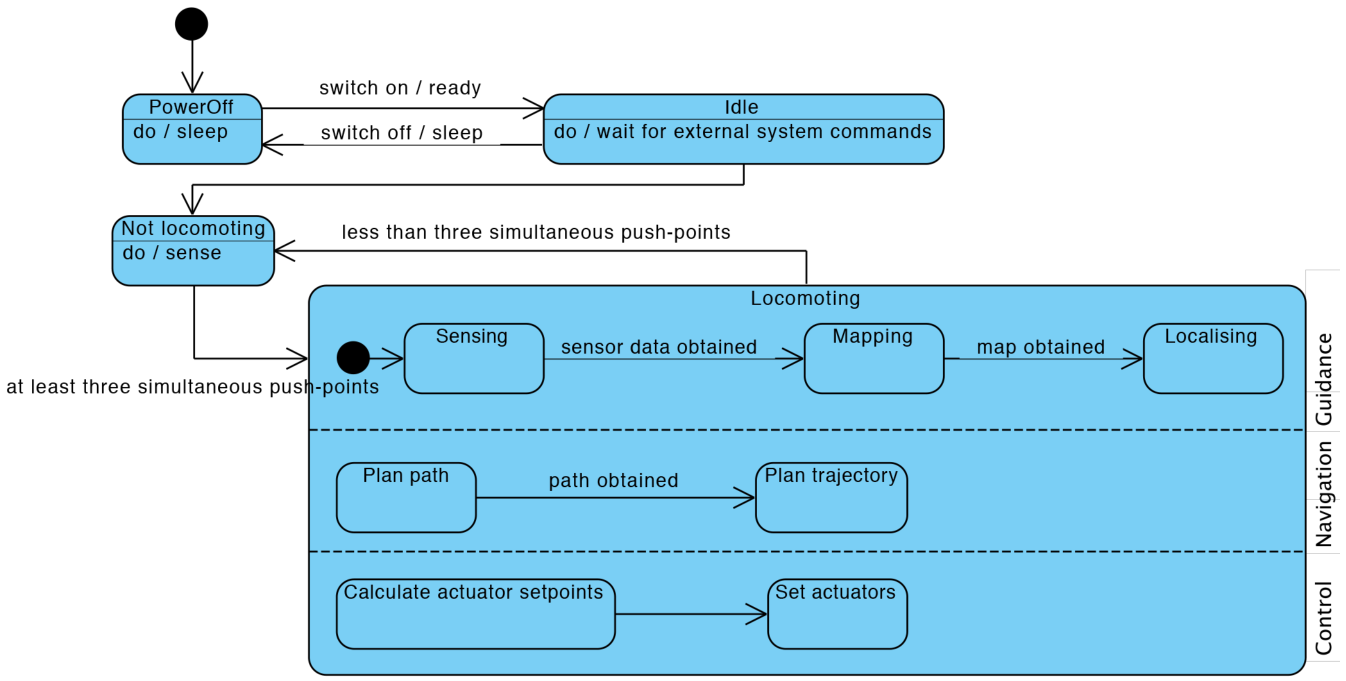
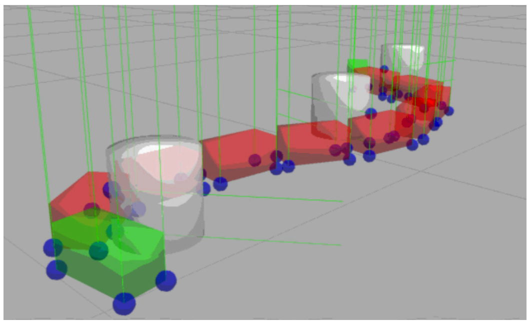


Fig. 6: The considered state machine diagram.

3) Snake robot sensors: Simulated contact sensors are adopted to retrieve bump contacts. In particular, the gazebo_ros_bumper_sensor is adopted [15]. Forces, torques, contact positions and contact normals can be retrieved. The retrieved bump contacts are shown in Fig. 7-a, while the retrieved coordinate axes are shown in Fig. 7-b. This


(a)

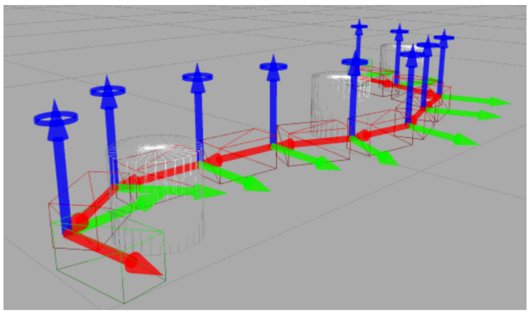

(b)

Fig. 7: (a) the retrieved bump contacts. (b) the retrieved coordinate axes.

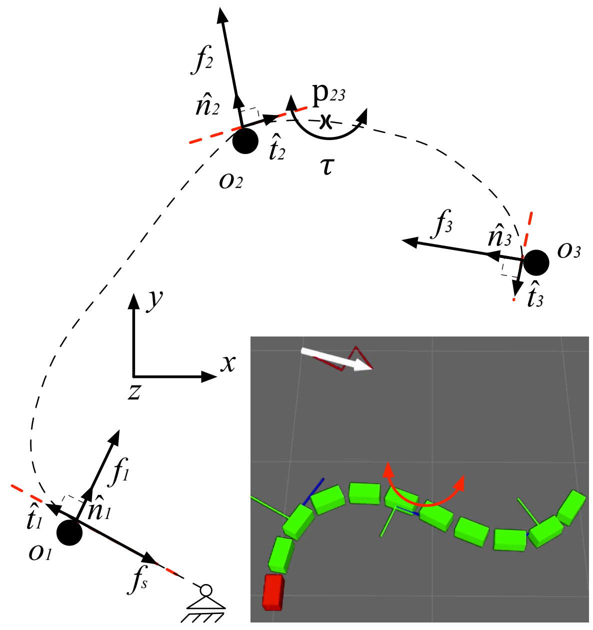


Fig. 8: The *obstacle triplet model*. Each obstacle is represented by a round dot, while a possible path is suggested by the dashed line. At the bottom right corner, the corresponding information is visualised in RViz.

information can be used to achieve tactile perception. A depth camera can be attached to the snake robot head for visual perception purposes. In addition, *SnakeSIM* allows researchers for exploiting the large variety of other robotics sensors that are supported by ROS.

IV. Case study

SnakeSIM allows researchers to design, develop and test alternative control algorithms for POAL. The level where researchers can implement their own controllers is the control layer of the proposed hierarchical architecture. Each possible control method must be compliant with the provided interfaces of the framework.

As a case study, a novel control algorithm for POAL, which was previously designed by our research group based on the foundations proposed in [16], is developed and tested using *SnakeSIM*. This control algorithm is briefly summarised in this section. For further details, the reader is referred to [17]. The aim of the selected control algorithm is to seek for a pragmatic approach to POAL by reducing the problem from a multi-dimensional formulation to only a two dimensional instance with one direction along the path and the other direction across the path.

A. The obstacle triplet model

According to a review of lateral undulation as it occurs in nature [18], at least three simultaneous push-points are necessary for this type of motion to take place. Based on this evidence, the control model for the case where the snake robot is in contact with three simultaneous push-points is considered, as shown in Fig. 8. The three simultaneous push-points are selected at alternating sides of the path and indicated as o_1, o_2, o_3. The term *obstacle triplet model* is adopted to describe this particular scenario.

The following assumptions are considered:

1) a path, $S(s)$, with s being the path length parameter, is known. The obstacle locations, o_1, o_2, o_3, are also known;
2) the snake is always on the path $S(s)$;
3) the snake is planar and discrete (joints and links are numbered from left to right);
4) there is no ground or obstacle friction;
5) the snake is at rest;
6) the snake tail link is tethered to the ground, as shown in Fig. 8. The tether is unactuated. No tangential movements are allowed. The tail is not restricted in any other way. A tensile force, f_s, acts along the tangent at o_1. Note that, the only reason why the tether is introduced is to justify a static analysis – which is of course an approximation of the real case;
7) the snake is perfectly rigid except at the point where an internal torque can be applied. The obstacles are perfectly rigid and fixed to the ground surface;
8) we choose to apply an internal torque, τ, at a known point, p_{23}, on the path between o_2 and o_3, as shown in Fig. 8.

As shown in Fig. 8, the triplet of contacts generate the forces f_1, f_2, f_3, which are normal to the snake body at each contact point. The normal and tangent unit vectors are indicated as $\hat{n}_i$ and $\hat{t}_i$ respectively, where $i = 1,2,3$.

According to the assumption 8, an internal torque, τ, is

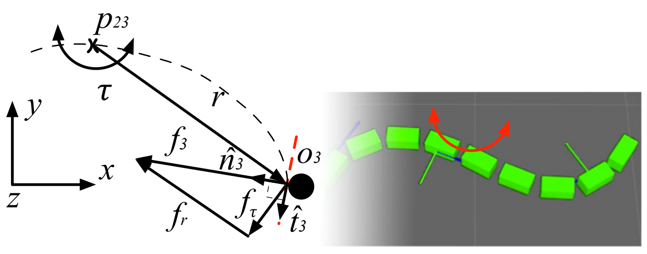


Fig. 9: A detail showing the internal torque, τ, applied at a known point, p_{23}, on the path (e.g. snake) between o_2 and o_3. At the bottom right corner, the corresponding information is visualised in RViz.

applied at a known point, p_{23}, on the path (e.g. snake) between o_2 and o_3. This produces a counter force, f_τ, acting at the obstacle o_3, as shown in detail in Fig. 9. The torque radius is denoted as r and it is known because of assumption 1, 2 and 8. Since there is no friction by assumption 4, the total contact force from o_3 on the snake must be perpendicular to the tangent at o_3. Since the contact force, f_3, is normal to the snake body at the point o_3, the following relation is valid:

$$f_3 \cdot \hat{t}_3 = 0. \tag{1}$$

Referring to Fig. 9, the contact force, f_3, can be obtained as:

$$f_3 = f_\tau + f_r, \tag{2}$$

where, by using the definition of torque, the counter force, f_τ, can be expressed as:

$$f_\tau = r \times \tau, \tag{3}$$

while f_r is the force component parallel to the torque radius, r, and by definition can be expressed as:

$$f_r \triangleq |f_r| \frac{r}{|r|}. \tag{4}$$

By combining (2), (3) and (4), the contact force, f_3 can be rewritten as:

$$f_3 = r \times \tau + |f_r| \frac{r}{|r|}, \tag{5}$$

which, because of (1), can be rewritten as:

$$(r \times \tau + |f_r| \frac{r}{|r|}) \cdot \hat{t}_3 = 0. \tag{6}$$

This in turn can be rewritten as follows by using the distributive property of the dot product and the anti-commutative property of the cross product:

$$|f_r| \frac{r}{|r|} \cdot \hat{t}_3 = (\tau \times r) \cdot \hat{t}_3. \tag{7}$$

It follows that:

$$|f_r| = \frac{(\tau \times r) \cdot \hat{t}_3}{\frac{r}{|r|} \cdot \hat{t}_3}. \tag{8}$$

Consequently, because of (5) and (8), f_3 can be rewritten as:

$$f_3 = r \times \tau + \left[\frac{(\tau \times r) \cdot \hat{t}_3}{\frac{r}{|r|} \cdot \hat{t}_3} \right] \frac{r}{|r|}. \tag{9}$$

By considering the assumption 5 of static conditions, the following force balance equation can be obtained:

$$f_s + f_1 + f_2 + f_3 = 0, \tag{10}$$

where, f_s is the tensile force that need to be counterbalanced, f_3 is given by (9), while f_1, f_2 are unknown variables. The directions of the contact forces, f_1, f_2, are known as they are given by the normals at each contact point. Since there are 2 unknown variables, one more equation is required to completely determine the considered system. The torques exerted on the robot about the global origin by the external forces can be considered as follows:

$$o_1 \times (f_s + f_1) + o_2 \times f_2 + o_3 \times f_3 = 0. \tag{11}$$

By combining (9), (10) and (11), the considered system is now completely determined. The bending torque τ can be uniquely computed at any point. In other words, given any point, s, on the path, it is possible to uniquely express the bending torque as a function of f_s, f_1, f_2, f_3:

$$\tau(s) = f(f_s, f_1, f_2, f_3). \tag{12}$$

Equivalently, this means that the tensile force, f_s, can be obtained as a function of $\tau(s), f_1, f_2, f_3$:

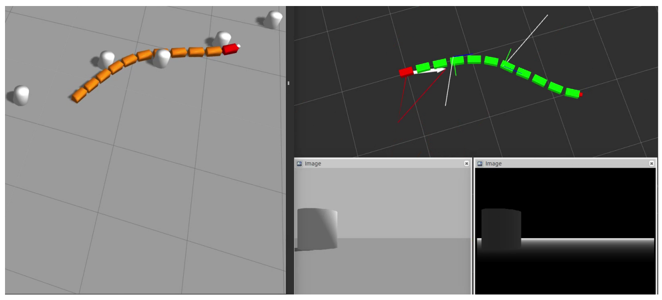
$$f_s = g(\tau(s), f_1, f_2, f_3). \tag{13}$$

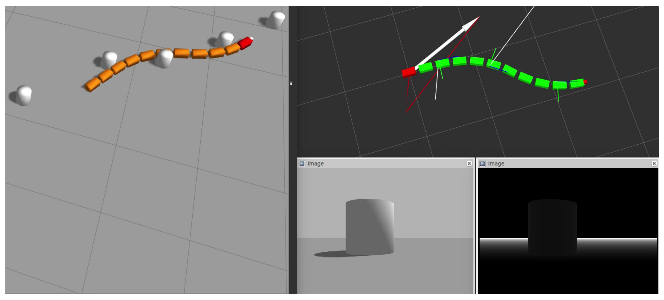
Therefore, the only necessary control variable is $\tau(s)$.

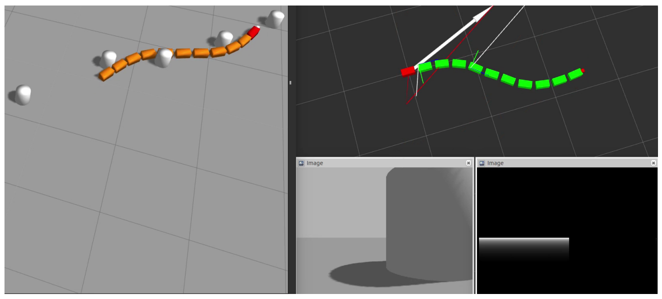
V. Simulation results

The control approach presented in Section IV as a case study was implemented by using *SnakeSIM*. A related simulation was carried out over a time period of 10 s in order to illustrate the use of the framework. For the simulation a desired propulsion acceleration, was indirectly set by imposing $f_s = 35N$. It should be noted that the joint 6 is controlled by the propulsion controller, while all the others joints are controlled in position mode. As shown in Fig. 10, a sequence of screenshots is taken from *SnakeSIM* (for each screenshot, the left snapshot is from Gazebo and the right snapshot is from RViz) demonstrating the effectiveness of the considered control algorithm for POAL. Note that this sequence shows the action forces retrieved by the tactile sensors and the images captured by the depth camera mounted on the snake robot head. During this sequence, the link 6 is rotated upwards, pushing on the obstacle to its left. The whole snake is then pushed slightly forward. It should be noted that the force polygon is also shown (red polygon) for each screenshot highlighting the resultant propulsion force (white arrow). During the same simulation, the torques applied on all joints are monitored during the propulsion phase, as shown in Fig. 11.

In Fig. 12 and Fig. 13, the joint angles and the applied joint torques are shown respectively for the most relevant joints during the propulsion phase of the considered simulation. In this case, the first joint to be controlled in propulsion mode was joint 7, since link 7 was initially in contact with the central obstacle. As soon as there is an applied torque value, the joint angles suddenly deviates far from their


(a)


(b)


(c)

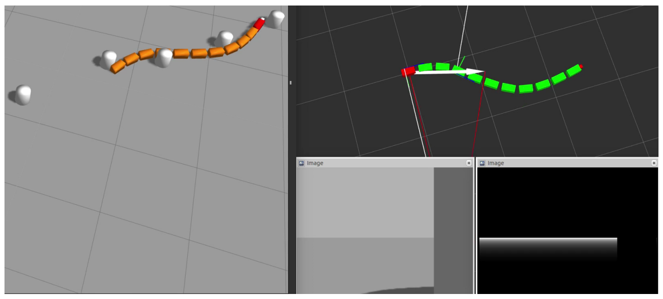

(d)

Fig. 10: (a), (b), (c), (d) A sequence of screenshots taken from *SnakeSIM* (for each screenshot, the left snapshot is from Gazebo and the right snapshot is from RViz). The action forces and the images captured by the depth camera are shown.

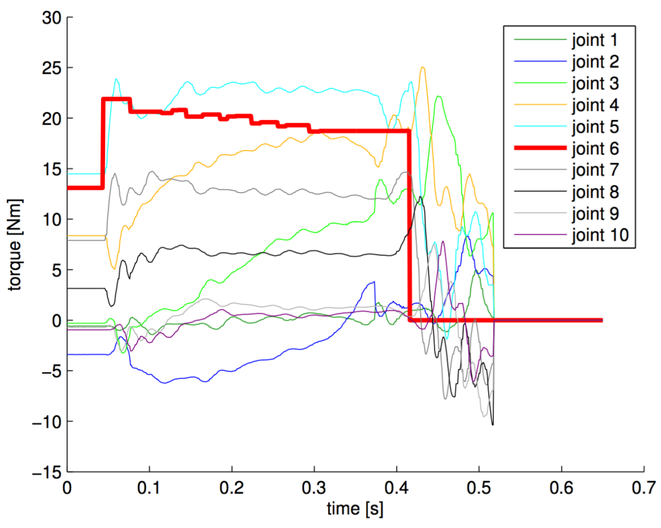


Fig. 11: Joint torques applied during the propulsion phase. The joint 6 is controlled by the propulsion controller, while all the others joints are controlled in position.

corresponding reference values. As the snake robot continues to propel forward, the propulsion torque is transfered from joint 7 to the next joints 6, 5 and 4, propagating backward until the snake robot loses contact with the obstacle at its tail.

VI. Conclusions and Future Work

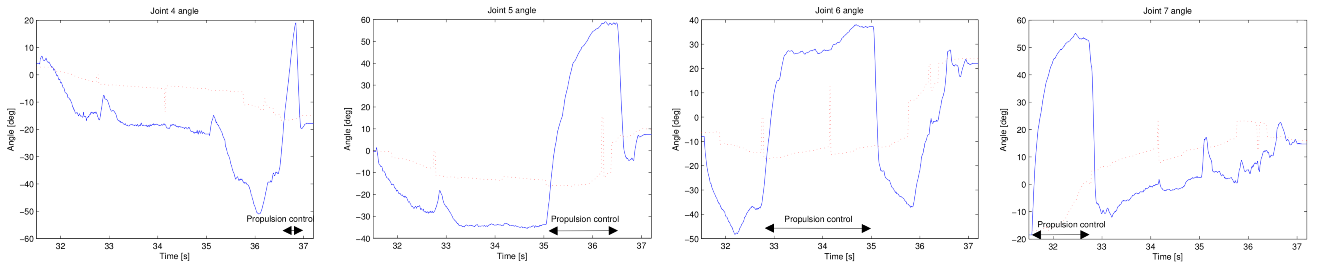
SnakeSIM, a virtual rapid-prototyping framework that allows for the design and simulation of control algorithms for perception-driven obstacle-aided locomotion (POAL) was presented in this paper. The framework proposed is integrated with ROS and enables researchers to develop control algorithms for POAL in a simulated environment with Gazebo. This integration makes the development of POAL algorithms more safe, rapid and efficient. Different sensors can be simulated both for tactile as well as visual perception purposes. Once the development phase is terminated, the designed control algorithms can be tested on a real prototype and continuously tuned with real sensor data. The integration with a real snake robot prototype, the Mamba snake robot [19], is currently ongoing.

The framework is built on open-source software. It is the opinion of the authors that the key to maximising the long-term, macroeconomic benefits for the robotics industry and for academic robotics research relies on the closely integrated development of open content, open standards, and open source. *SnakeSIM* contributes towards this same strategy. In the future, different control algorithms for POAL may be designed and tested. The framework may also be adopted as an educational tool.

As future work, the proposed approach needs to be validated with physical experiments. The ultimate goal of the *SnakeSIM* framework is to facilitate effective experimentation with different control algorithms at the level where the obstacle triplet model is currently plugged in.

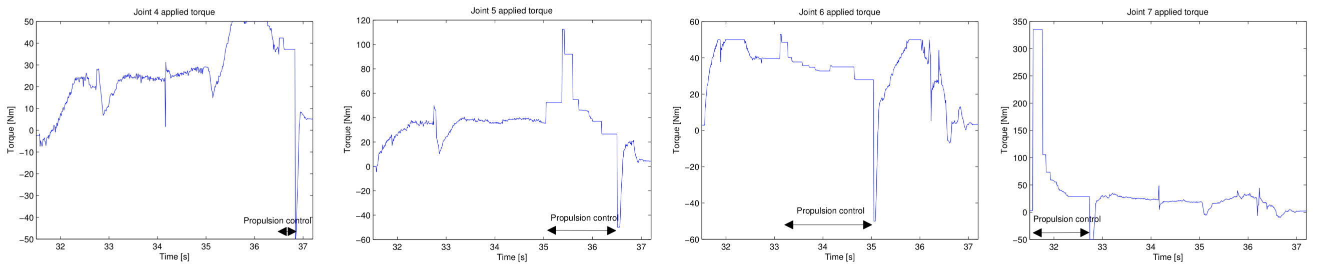
Acknowledgement

The authors gratefully acknowledge the contribution of the students Stian Grøttum Danielsen and Guillaume Adam in the implementation of this work.



(a) (b) (c) (d)

Fig. 12: (a), (b), (c), (d) Relevant joint angles during the propulsion phase. The dashed red lines are the reference angles, while the blue lines are the measured angles. The two-sided arrows in the plots show the approximate time that specific joint is controlled for propulsion generation.



(a) (b) (c) (d)

Fig. 13: (a), (b), (c), (d) Relevant joint torques during the propulsion phase. The two-sided arrows in the plots show the approximate time that specific joint is controlled for propulsion generation.

REFERENCES

[1] F. Sanfilippo, J. Azpiazu, G. Marafioti, A. A. Transeth, Ø. Stavdahl, and P. Liljebäck, "A review on perception-driven obstacle-aided locomotion for snake robots," in *Proc. of the 14th International Conference on Control, Automation, Robotics and Vision (ICARCV), Phuket, Thailand*, 2016, pp. 1–7.

[2] F. Sanfilippo, J. Azpiazu, G. Marafioti, A. A. Transeth, Ø. Stavdahl, and P. Liljebäck, "Perception-driven obstacle-aided locomotion for snake robots: the state of the art, challenges and possibilities," *Applied Sciences*, vol. 7, no. 4, p. 336, 2017.

[3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA), workshop on open source software*, vol. 3, no. 3.2, 2009, p. 5.

[4] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 3, 2004, pp. 2149–2154.

[5] J. Won, K. DeLaurentis, and C. Mavroidis, "Rapid prototyping of robotic systems," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 4, 2000, pp. 3077–3082.

[6] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "Compliant control of the body shape of snake robots," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 4548–4555.

[7] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "A 3d motion planning framework for snake robots," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014, pp. 1100–1107.

[8] J. Leon, L. Paez, and K. Melo, "Modular snake ros," in *Proc. of the ROS Developer Conference (ROSCon2013)*, 2013.

[9] K. Melo, J. Leon, A. di Zeo, V. Rueda, D. Roa, M. Parraga, D. Gonzalez, and L. Paez, "The modular snake robot open project: Turning animal functions into engineering tools," in *Proc. of the IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1–6.

[10] D. Krupke, N. Hendrich, J. Zhang, and H. Zhang, "Gait optimization based on physics simulation of 3d robot models with a modular robotic simulation system," in *Proc. of the 18th International Conference on CLAWAR 2015*, 2016, pp. 612–619.

[11] H. R. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, no. 2, pp. 337–345, 2015.

[12] F. Kendoul, "Towards a Unified Framework for UAS Autonomy and Technology Readiness Assessment (ATRA)," in *Autonomous Control Systems and Vehicles*, ser. Intelligent Systems, Control and Automation: Science and Engineering, K. Nonami, M. Kartidjo, K.-J. Yoon, and A. Budiyono, Eds. Springer Japan, 2013, no. 65, pp. 55–71, dOI: 10.1007/978-4-431-54276-6_4. [Online]. Available: http://link.springer.com/chapter/10.1007/978-4-431-54276-6_4

[13] J. Rumbaugh, I. Jacobson, and G. Booch, *Unified modeling language reference manual, the*. Pearson Higher Education, 2004.

[14] Open Source Robotics Foundation. (2016, May) Tutorial: Using a URDF in Gazebo. [Online]. Available: http://gazebosim.org/tutorials/?tut=ros_urdf

[15] Open Source Robotics Foundation. (2016, May) Tutorial: Using Gazebo plugins with ROS. [Online]. Available: http://gazebosim.org/tutorials?tut=ros_gzplugins

[16] C. Holden and Ø. Stavdahl, "Optimal static propulsive force for obstacle-aided locomotion in snake robots," in *Proc. of the IEEE International Conference on Robotics and Biomimetics (ROBIO), Shenzhen, China*, 2013, pp. 1125–1130.

[17] F. Sanfilippo, Ø. Stavdahl, G. Marafioti, A. A. Transeth, and P. Liljebäck, "Virtual functional segmentation of snake robots for perception-driven obstacle-aided locomotion," in *Proc. of the IEEE Conference on Robotics and Biomimetics (ROBIO), Qingdao, China*, 2016, pp. 1845–1851.

[18] Z. Y. Bayraktaroglu and P. Blazevic, "Understanding snakelike locomotion through a novel push-point approach," *Journal of dynamic systems, measurement, and control*, vol. 127, no. 1, pp. 146–152, 2005. [Online]. Available: http://cat.inist.fr/?aModele=afficheN&cpsidt=16829403

[19] P. Liljebäck, Ø. Stavdahl, K. Pettersen, and J. Gravdahl, "Mamba - A waterproof snake robot with tactile sensing," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2014, pp. 294–301.