Path Planning for Perception-Driven Obstacle-Aided Snake Robot Locomotion

Kristian G. Hanssen
Dept. Mathematics and Cybernetics
Sintef Digital
Trondheim, Norway
kristian.gaustad.hanssen@sintef.no

Aksel A. Transeth
Dept. Mathematics and Cybernetics
Sintef Digital
Trondheim, Norway
aksel.a.transeth@sintef.no

Filippo Sanfilippo
Dept. of Engineering Sciences
University of Agder (UiA)
Grimstad, Norway
filippo.sanfilippo@uia.no

Pål Liljebäck
Eelume AS
Trondheim, Norway
pal.liljeback@eelume.com

Øyvind Stavdahl
Dept. of Engineering Cybernetics
Norwegian University of Science and Technology (NTNU)
Trondheim, Norway
oyvind.stavdahl@ntnu.no

***Abstract*—Development of snake robots have been motivated by the ability of snakes to move efficiently in unstructured and cluttered environments. A snake robot has the potential to utilise obstacles for generating locomotion, in contrast to wheeled robots which are unable to move efficiently in rough terrain. In this paper, we propose a local path planning algorithm for snake robots based on obstacle-aided locomotion (OAL). An essential feature in OAL is to determine suitable push-points in the environment that the snake robot can use for locomotion. The proposed method is based on a set of criteria for evaluating a path, and is a novel contribution of this paper. We focus on local path planning and formulate the problem as finding the best next push point and the trajectory towards it. The path is parameterised as a quadratic Bézier curve. The algorithm is implemented and tested with a simulator, employing decentralised joint controllers with references generated by a constant translation speed of the snake along the path.**

Careful design of the criteria allows us to use simple position and velocity controllers for the joints, circumventing the need for force control. However, the set of feasible paths will be restricted by this approach.

The proposed criteria can also be used in a global path planning algorithm; the local focus is due to one of the key use cases of snake robots: operating in unstructured and unknown environments.

***Index Terms*—Obstacle-aided locomotion, motion generation, gait pattern, snake robot locomotion, path planning**

I. Introduction

Path planning is necessary for navigation and for decision making in terms of where, when and how the robot should ideally move. This is especially relevant for snake robot locomotion in an environment cluttered with obstacles [1].

A traditional approach when dealing with obstacles in robotics is to try to avoid them. Collisions may make the robot unable to progress and cause mechanical stress or damage to equipment. Therefore, many studies on robot navigation have focused on obstacle avoidance. For instance, potential-field based methods for path planning have traditionally been adopted [2] to effectively model artificial force fields around objects that are either repulsive or attractive on the robot. Potential-field based methods for path planning have the advantage of being fast, but they are also prone to getting stuck in local minima when used with complex systems such as redundant snake robots. To avoid the danger of local minima, an alternative method of setting up a potential-field system that replaces rigid links by stiff (virtual) springs was presented in [3]. However, path planners for obstacle avoidance are not practically applicable when the robot must traverse terrains highly cluttered with obstacles.

A more relaxed approach to obstacle avoidance can be considered by using sensory feedback actively and efficiently. Rather than always avoiding collisions, a snake robot may be allowed to collide with obstacles, but collisions must be controlled so that no damage to the robot occurs. This methodology has a substantial advantage over traditional obstacle-avoidance methods in environments in which totally avoiding contact is impractical, such as many natural environments. This new robotic paradigm was first introduced in [4]. In [5], a general formulation of motion constraints due to contact with obstacles was presented. Based on this formulation, a new inverse kinematics model was developed that provides joint motion for snake robots under contact constraints. Based on this model, a motion planning algorithm for snake robot motion in a cluttered environment was also proposed. However, this approach is mainly based on local perception. An alternative solution to obtain better predictability and more prompt reaction considers both global and local perceptions for path generation and adaptation. Based on this assumption, a distributed path planning algorithm for snake robot navigation in an intelligent environment with distributed wireless visual sensors was proposed in [6]. Via communication links between sensors, segments of a path, as an elastic band from start position to goal position, interact with each other to react to

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.

repulsive forces from obstacles whilst maintain compliance. The compliance has to be subject to the robot kinematic constraints and the elastic band may change to rigid or even to a broken state. However, this approach requires sensors to be placed in the environment, and is not realistic for practical applications. Moreover, accommodating obstacles does not allow fully exploiting the environment for locomotion. In [7], an interesting approach for linking low-level controllers to high-level planners was proposed, and applied to a snake-like robot in an environment with obstacles.

To fully exploit roughness in the terrain for locomotion, the goal should not be to avoid or accommodate obstacles, but rather to move relative to these irregularities in a way that enables the snake robot to utilise push points for more efficient propulsion. We use the term perception-driven obstacle-aided locomotion (POAL) as locomotion where the snake robot utilises a sensory perceptual system to perceive the surrounding operational environment, for means of propulsion with obstacle-aided locomotion [8]. As an attempt to achieve this, a framework based on non-smooth dynamics for the development, analysis, and testing of forthcoming motion planning and control approaches to obstacle-aided locomotion were presented in [9]. A control strategy employing measured contact forces to maintain propulsion while simultaneously preventing the snake robot from being jammed between obstacles in its path was presented in [10]. Snake robots have a high number of degrees of freedom and this adds complexity to their control systems. To address this challenge, the possibility of simplifying the snake robot model to deal with a lower-dimensional system was considered in [11]. However, the development of motion patterns based on the mathematical model and observations of real snakes that enable the snake robot to detect and exploit obstacles by itself for efficient locomotion is still missing. Moreover, joint torque sensing and control can be a valuable tool for snake robot locomotion, but it has some challenges regarding, e.g., a need for torque sensing, accurate sensor calibration, and possible sensor failure. To the best of our knowledge, an effective path planning framework for POAL is still missing.

In this paper we present a new method for local path planning for snake robot obstacle-aided locomotion (OAL) which alleviates the need for force control of a snake robot's joints. We also propose a set of criteria for evaluating a path which will ensure forward locomotion and avoid that a snake robot gets stuck. We pose the local path planning problem as finding the best next push point which a snake robot can use for locomotion and the path the snake robot must follow in order to reach the push point. The path is parameterised as a quadratic Bézier curve, extending the framework proposed in [12]. We evaluate and verify our presented algorithm in a 2D snake robot simulator based on the framework of convex analysis and non-smooth dynamics. The simulation results show that suitable push points in front of the robot is determined, and that it is able to reach these push points successfully.

II. Method

This work is based on the framework proposed in [12], using quadratic Bézier curves to describe the shape and trajectory of the snake robot. For consistency, we follow the same notation as in the referenced paper whenever possible. Some adaptation of the framework was needed, and the relevant changes are described in subsequent paragraphs.

The path is laid out sequentially, as it is intended for a snake robot navigating in an unknown environment, discovering obstacles to be used as push points as it moves forward. We assume the entire snake will follow the path laid out, and no adaptation or updating of the path is conducted once it is constructed. However, since the snake robot consists of a discrete number of links with a non-negligible length, the assumption that the entire snake follows the entire path is clearly a simplification. To reduce this effect, we enforce the path to be straight for a fixed length around the contact point of the obstacle. As the head of the snake approaches the end of the current path, we extend the path by finding the best next obstacle to use as a push point. This means finding a feasible path to the obstacle, and the approach angle with respect to this obstacle.

A. Bézier Curves

Bézier curves are widely used in computer graphics to model smooth curves. Bézier curves are defined in terms of a set of control points $\boldsymbol{b}_0, \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n$, where $\boldsymbol{b}_0$ and $\boldsymbol{b}_n$ are the first and last control points, respectively. The curve will always pass through the end points, but in general not through the intermediate control points. A first order Bézier curve, also denoted linear Bézier curve, is simply a straight line between the control points $\boldsymbol{b}_0$ and $\boldsymbol{b}_1$. A quadratic Bézier curve is defined by the control points $\boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2$, and is a linear interpolation between the straight lines from $\boldsymbol{b}_0$ to $\boldsymbol{b}_1$ and $\boldsymbol{b}_1$ to $\boldsymbol{b}_2$. This can be expressed as

$$\boldsymbol{B}(s) = (1-s)^2\boldsymbol{b}_0 + 2(1-s)s\boldsymbol{b}_1 + s^2\boldsymbol{b}_2 \tag{1}$$

where $s \in [0, 1]$ is the curve parameter. Similar to the generalisation from linear to quadratic Bézier curves, the generalisation can be done to higher orders. In this work, we will only use quadratic Bézier curves as in [12]. The notation $\boldsymbol{B}(s; \boldsymbol{b}_0, \boldsymbol{b}_1, \boldsymbol{b}_2)$ is used when explicitly specifying the control points of a quadratic Bézier curve.

In order to enforce particular properties of the curve, we use the concept of curvature. The curvature at a point along a curve is the reciprocal of the radius of the osculating circle at that point. The curvature of a parametric curve in Cartesian coordinates $\boldsymbol{B}(s) = [b_x(s), b_y(s)]$ can be expressed as

$$\kappa = \frac{|b'_x b''_y - b''_x b'_y|}{(b'^2_x + b'^2_y)^{\frac{3}{2}}} \tag{2}$$

where curve parameter s is left out for readability, and the primes denotes derivatives with respect to s.

B. Control Framework

The control framework proposed in [12] is based on specifying the desired shape of the snake as a continuous curve, denoted the shape curve, by a set of shape control points (SCP). In addition, each SCP is assigned a tangent angle. These SCPs are then connected by quadratic Bézier curves, which define the shape curve. SCP i is denoted $\mathbf{P}_i$. In order to find this quadratic Bézier curve, the intermediate point $\mathbf{P}_{i,i+1}$ between SCP $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$ must be found, which is the crossing of the tangents from $\mathbf{P}_i$ and $\mathbf{P}_{i+1}$. However, if the tangents are parallel, such a point only exists if both SCPs are on this line. Furthermore, and not noted in [12], this crossing of the tangents might be in a direction causing a 180 degree abrupt turn at the SCP. This is clearly not the desired behavior and can be overcome by using higher order Bézier curves. We choose to continue using quadratic Bézier curves, adding additional SCPs when needed to avoid these 180 degree change of direction. In general, quadratic Bézier Curves behave more predictable and are easier to work with. We will also consider the intermediate points when searching for a feasible path, as this simplifies the implementation.

Note the slight change in notation compared to the section on Bézier curves, where also the intermediate point of a quadratic Bézier curve was denoted as a control point. For the control framework, the end points of the Bézier curves are the SCPs.

We employ the framework to construct a path in an environment with a set of obstacles. As a simplification, ground friction is neglected, although friction is included in the subsequent case study. The only forces acting on the snake will be from the obstacles, and perpendicular to the tangent of the snake at the obstacle. We denote these points as *push points*. Neglecting friction simplifies the analysis considerably. The path is constructed sequentially, so that when the head of the snake approaches the end of the current shape curve, a new push point along with a path to this push point is found, and added to the shape curve.

The snake is assumed to be perfectly aligned with the shape curve and in contact with a set of active push points $\mathbb{A}$ at location $\mathbf{L}_a$ for push point $a \in \mathbb{A}$. The unit tangent of the snake at push point a is denoted $\mathbf{T}_a$, and the unit normal vector in the direction of the resulting reaction force exerted by the obstacle on the snake, if the snake is pushing against the obstacle, is denoted $\mathbf{N}_a$. In Fig. 1 an illustration of the snake robot close to an obstacle is shown. The dashed blue line is the path to be followed, the yellow dot is $\mathbf{L}_a$ for the particular obstacle a, and the black line represent $\mathbf{N}_a$. In the following, we define a set of criteria which must be met in order for the snake to move forward.

The push points are chosen such that a rigid snake is not able to rotate about any of the push points. This means that for each push point, there must be at least two other push points, one capable of producing a torque in a clockwise direction, and one in a counter clockwise direction. We define the vector

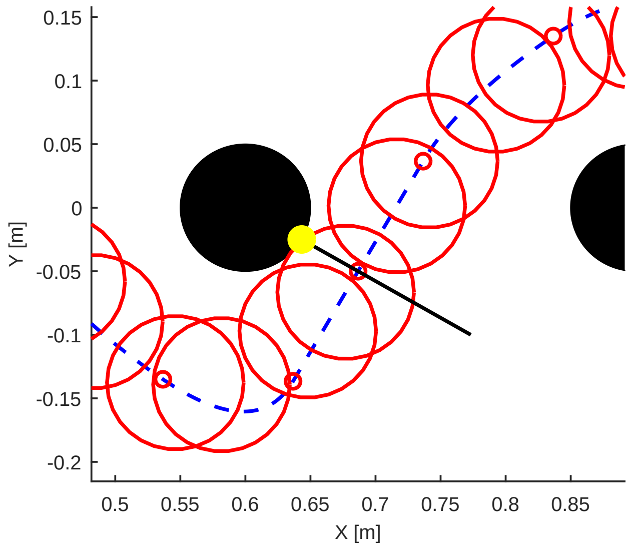

Fig. 1. Illustration of snake robot on path close to obstacle

from the location of push point $\mathbf{L}_{\hat{a}}$ to $\mathbf{L}_a$ as

$$\mathbf{D}_{\hat{a}a} = \mathbf{L}_a - \mathbf{L}_{\hat{a}} \tag{3}$$

So the torque about push point a due to a unit force vector at push point $\hat{a}$ can be expressed as

$$\tau_{\hat{a}a} = \mathbf{D}_{\hat{a}a} \times \mathbf{N}_{\hat{a}} \tag{4}$$

Note that we are only considering 2D planar motion in this paper, and there is a slight abuse of notation in eq. (4). Strictly speaking the cross product is not defined in 2D, while in eq. (4) we implicitly assume the vectors are augmented by a zero in the third dimension. Furthermore, the result of the cross product is a 3D vector itself, but only the third component will be nonzero and of interest. It is only the third component we are considering, and therefor treating it as a scalar.

In order to oppose a rotation about push point a, we require

$$\sum_{\hat{a}\in\mathbb{A}\backslash a} \max(\tau_{\hat{a}a}, 0) \geq \underline{\tau} \tag{5a}$$

$$\sum_{\hat{a}\in\mathbb{A}\backslash a} -\min(\tau_{\hat{a}a}, 0) \geq \underline{\tau} \tag{5b}$$

where $\underline{\tau}$ is a strictly positive threshold parameter. Note that the summation could also be conducted over the entire set $\mathbb{A}$ as τ_{aa} will always be zero.

It is not enough to merely ensure that the snake can not rotate about any of the push points, it must also be able to produce a propulsive force in the direction of motion. The direction of motion is a vector pointing from the tail towards the head, as an infinitesimal small gliding motion along the path will change the center of gravity in this direction. Under the assumption of no friction, the resultant forces needed to move the snake forward will be in the same direction. We represent the direction of motion by a unit vector $\mathbf{R}$, and the propulsive component of all push points are projected onto this vector, denoted f_a^r, expressed by

$$f_a^r = \mathbf{N}_a \cdot \mathbf{R} \tag{6}$$

while the total resulting force from unit vectors at each push point is then expressed as

$$F^r = \sum_{a \in \mathbb{A}} f_a^r \tag{7}$$

To summarise, for a set of active push points, we require the push points to be located and oriented in such a way that eq. (5) is satisfied for all active push points, and $F^r \geq \underline{f}$

It could be argued that only components in the desired direction should be included, not also negative terms. We choose deliberately to penalise negative terms, as they seem to greatly increase the possibility of jamming.

1) Finding a path to an obstacle: In the following, we present an approach to parameterise a path to an obstacle, respecting curvature requirements and avoiding collisions with other obstacles.

We assume the head of the snake is at the end of the current shape curve, and we would like to select an obstacle from a set of potential obstacles $\mathbb{B}$. For a particular obstacle $b \in \mathbb{B}$ and an approach angle β, the location of the resulting contact point between the snake and the obstacle is denoted $\mathbf{L}_{b,\beta}^c$, and the unit normal force vector acting on the snake at $\mathbf{L}_{b,\beta}^c$ is denoted $\mathbf{N}_{b,\beta}$. Note that strictly speaking $\mathbf{N}_{b,\beta}$ is only dependent on β, and is uniquely defined by β. $\mathbf{T}_{b,\beta}$ is perpendicular to to $\mathbf{N}_{b,\beta}$, but not uniquely defined, as a rotation of 180 degrees will also meet this requirement. This corresponds to approaching an obstacle on the left or right side. In this work, both sides are considered.

For each potential obstacle and approach angle pair (b, β), a feasible path from the current head position, assumed located at the last SCP $\mathbf{P}_k$ with orientation ψ_k, to a point $\mathbf{L}_{b,\beta}$ such that the snake will contact the obstacle at $\mathbf{L}_{b,\beta}^c$ is attempted constructed. For a snake width no width $\mathbf{L}_{b,\beta}^c$ and $\mathbf{L}_{b,\beta}$ will coincide, otherwise $\mathbf{L}_{b,\beta}$ is adjusted to $\mathbf{L}_{b,\beta}^c$ to account for the width of the snake. In this work, we simply search for a path which will avoid collision with any obstacle and satisfy a curvature requirement. As noted, we impose the requirement that the shape curve should be straight in the vicinity of push points, ensuring the particular approach angle is maintained for a snake with a finite number of links. This is done by selecting the first Bézier segment to simply be a projection of $\mathbf{P}_k$ along ψ_k for a distance of $\frac{l_c}{2}$ to $\hat{\mathbf{P}}_{k+1}$. l_c is the parameter defining the straight line segment length at each obstacle. This can be expressed as

$$\hat{\mathbf{P}}_{k+1} = \mathbf{P}_k + \frac{l_c}{2} \begin{bmatrix} \cos\psi_k & \sin\psi_k \end{bmatrix} \tag{8}$$

The hat is used in order to denote a possible path, $\hat{\mathbf{P}}$ a potential SCP. The intermediate point for the straight line segment can then be chosen freely on this line, we use the midpoint

$$\hat{\mathbf{P}}_{k,k+1} = \frac{\mathbf{P}_k + \hat{\mathbf{P}}_{k+1}}{2} \tag{9}$$

The first straight line Bézier segment is then given as

$$\boldsymbol{B}(s; \mathbf{P}_1, \hat{\mathbf{P}}_{k,k+1}, \hat{\mathbf{P}}_{k+1}) \tag{10}$$

Likewise, a backward projection along β is conducted from $\mathbf{L}_{b,\beta}$. However, the path can approach $\mathbf{L}_{b,\beta}$ from both sides. We introduce the variable $\chi \in \{-1, 1\}$, and the backward projection can then be expressed as

$$\hat{\mathbf{P}}_{k+3}^{b,\beta} = \mathbf{L}_{b,\beta} + \chi\frac{l_c}{2} \begin{bmatrix} \cos\beta & \sin\beta \end{bmatrix} \tag{11}$$

As indicated by the $k+3$ subscript, we use two quadratic Bézier segments to construct the curve between these two projected potential SCPs. The intermediate point $\hat{\mathbf{P}}_{k+3,k+4}^{b,\beta}$ between $\hat{\mathbf{P}}_{k+3}^{b,\beta}$ and $\mathbf{L}_{b,\beta}$ is again chosen as the midpoint, and the last potential SCP is

$$\hat{\mathbf{P}}_{k+4}^{b,\beta} = \mathbf{L}_{b,\beta} \tag{12}$$

In the following we operate directly on the intermediate and control points of the Bézier curves, not the interface provided by the original control framework specifying control points and tangents. The first potential intermediate control point is then given by

$$\hat{\mathbf{P}}_{k+1,k+2}^{b,\beta} = \hat{\mathbf{P}}_{k+1} + c_1 l_k^{b,\beta} \begin{bmatrix} \cos\psi_k & \sin\psi_k \end{bmatrix} \tag{13}$$

where c_1 is a positive variable to be decided and $l_k^{b,\beta}$ is the euclidean distance between $\hat{\mathbf{P}}_{k+1}$ and $\hat{\mathbf{P}}_{k+3}^{b,\beta}$, that is

$$l_k^{b,\beta} = \left\| \hat{\mathbf{P}}_{k+3}^{b,\beta} - \hat{\mathbf{P}}_{k+1} \right\| \tag{14}$$

The second potential intermediate control point $\hat{\mathbf{P}}_{k+2,k+3}^{b,\beta}$ is likewise constructed by

$$\hat{\mathbf{P}}_{k+2,k+3}^{b,\beta} = \hat{\mathbf{P}}_{k+3}^{b,\beta} + c_3 l_k^{b,\beta} \chi \begin{bmatrix} \cos\beta & \sin\beta \end{bmatrix} \tag{15}$$

where c_3 is a positive variable. The last potential SCP is constructed as a convex combination of the former potential intermediate control points as

$$\hat{\mathbf{P}}_{k+2}^{b,\beta} = (1 - c_2)\hat{\mathbf{P}}_{k+1,k+2}^{b,\beta} + c_2 \hat{\mathbf{P}}_{k+2,k+3}^{b,\beta} \tag{16}$$

where $c_2 \in [0, 1]$ is a variable to be decided.

For a particular set (b, β) there is some degree of freedom in finding the path to $\mathbf{L}_{b,\beta}$, depending on the variables $\mathbf{c} = (c_1, c_2, c_3, \chi)$. This allows searching for a path meeting certain requirements. In Fig. 2 we see an example continuation of a path for a set of $\mathbf{c}$, with SCPs marked with circles and intermediate control points with crosses. Blue color is used for the fixed control points, and red color for control points which are determined by $\mathbf{c}$. In Fig. 3 three different paths for various $\mathbf{c}$ are shown, for the same obstacle with identical approach angle.

The path from the current head to a particular obstacle (b, β) is denoted $\hat{\mathbf{S}}(\hat{s})$, and can be written

$$\begin{aligned} \hat{\mathbf{S}}(\hat{s}; b, \beta, \mathbf{c}) = \boldsymbol{B}(\hat{s} - \lfloor \hat{s} \rfloor; \hat{\mathbf{P}}_{k+\lfloor \hat{s} \rfloor}^{b,\beta}, \hat{\mathbf{P}}_{k+\lfloor \hat{s} \rfloor, k+1+\lfloor \hat{s} \rfloor}^{b,\beta}, \\ \hat{\mathbf{P}}_{k+1+\lfloor \hat{s} \rfloor}^{b,\beta}), \hat{s} \in [0, 4) \end{aligned} \tag{17}$$

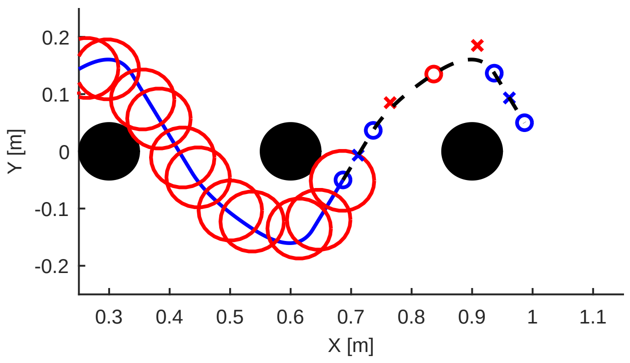

Fig. 2. Illustration of path parameterization

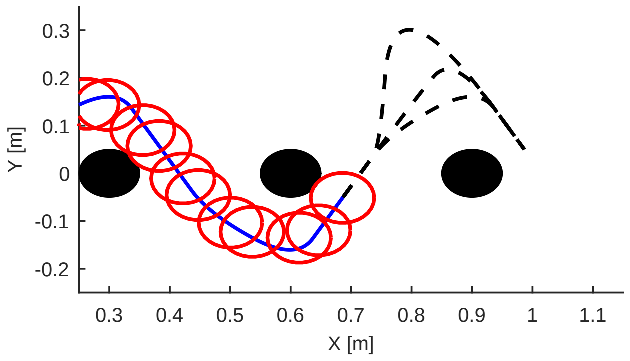

Fig. 3. Illustration of 3 various paths to an obstacle

where the dependence of the potential control points on $\mathbf{c}$ is left out for readability.

Searching for a feasible path is done by sampling the path at $\hat{s}_i,\ i \in \mathcal{I}$, and constructing a feasibility measure, one for curvature and one for obstacle obstructions. For curvature, let $\kappa\left(\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c})\right)$ denote the curvature for $\hat{s}_i$, and $\kappa^{\max}$ the maximum allowed curvature. The curvature violation $V^{\text{curv}}_{b,\beta,\mathbf{c}}$ is then formulated as

$$V^{\text{curv}}_{b,\beta,\mathbf{c}} = \sum_{i\in\mathcal{I}} \max\left(\kappa^{\max} - \left|\kappa\left(\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c})\right)\right|, 0\right) \tag{18}$$

The curvature of a Bézier curve can easily be calculated analytically by using eq. (2).

Similarly, for avoiding obstacle collisions, a function measuring the obstacle violation at a point along the path denoted $\Omega\left(\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c})\right)$, where $\Omega = 0$ when there is no chance of collision at a location. The obstacle collision measure for a path can be expressed as

$$V^{\text{coll}}_{b,\beta,\mathbf{c}} = \sum_{i\in\mathcal{I}} \Omega\left(\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c})\right) \tag{19}$$

In this work, we only deal with circular links and obstacles, hence the collision measure is simply penalizing sample points closer to an obstacle than the radii of the obstacle and the circular links combined. This can be expressed mathematically as

$$\Omega\left(\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c})\right) = \sum_{o\in\mathcal{O}} \max\left(r_l + r_o - \left\|\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c}) - \mathbf{L}_o\right\|, 0\right) \tag{20}$$

where $\mathcal{O}$ is the set of obstacles, $\mathbf{L}_o$ and r_o the center location and radius of obstacle circle o, respectively. r_l is the radius of the links.

For finding a feasible path, we formulate an unconstrained minimization problem including the curvature violation and obstacle collision in the objective by

$$J = -\mathcal{L} + \lambda_{\text{curv}} V^{\text{curv}} + \lambda_{\text{coll}} V^{\text{coll}} \tag{21}$$

Where $\mathcal{L}$ is the length of path, and λ are positive weighting parameters. For efficiently solving the optimization problems, $\Omega\left(\hat{\mathbf{S}}(\hat{s}_i; b, \beta, \mathbf{c})\right)$ was precomputed into a 2D map, using linear interpolation in this map in the solving. Furthermore, the max term in V^{curv} was replaced with a smoothed version

$$\max(x, y) \approx \max_a \frac{1}{2}\left(x + y + \sqrt{(x-y)^2 + a}\right) \tag{22}$$

which significantly improves the convergence in solving the problem. Note that no smoothing was conducted of obstacle violation term, as this seemed unnecessary in the numerical simulations conducted. However, this could be further studied.

Once a feasible path for all (b, β) have been sought, with corresponding path length $\mathcal{L}^{b,\beta}$ from the current head to the obstacle, we evaluate which obstacle and approach angle is the best.

2) Evaluating and selecting best next obstacle: In the previous sections, we have formulated two criteria to ensure that the snake is able to propel forward without rotating about any of the push points. Furthermore, we have proposed a method for finding a path respecting curvature conditions and avoiding collisions with obstacles. In the following, these criteria and paths are used in the evaluation of various obstacles, to select which obstacle to use as push point, how the snake should approach the obstacle and which contact angle to undertake.

For choosing the best next obstacle, we emphasise two criteria. It should bring us in the desired direction of motion, and it should be possible to continue motion forward when the new obstacle is reached. For the first criteria, we simply project the vector from the current head position to $\mathbf{L}_{b,\beta}$ onto the unit reference vector $\mathbf{R}$ representing the desired direction. This can be expressed as

$$J^{\text{dir}}_{b,\beta} = (\mathbf{L}_{b,\beta} - \mathbf{L}_h) \cdot \mathbf{R} \tag{23}$$

and we denote $J^{\text{dir}}_{b,\beta}$ as the direction score. The direction score will be between -1 and 1, 1 corresponding to a straight line in the same direction as R, while -1 in the opposite direction.

For evaluating the push points in terms of continued forward motion, we introduce the concept of *remaining tail*, denoted $J^{tail}_{b,\beta}$. We use remaining tail to describe how much further the snake can move forward after reaching the push points, without reaching any new obstacles. The idea is that the more remaining tail you have, the more freedom and possibilities you have in finding another new push points once you reached the relevant obstacle.

The overall objective function for selecting the best next obstacle is then formulated as

$$J(b, \beta) = J^{\text{dir}}_{b,\beta} + \mu J^{tail}_{b,\beta} \tag{24}$$

where μ is a non-negative parameter defining the relative importance of the two terms. The best next obstacle is then found as

$$(b^*, \beta^*) = \arg \min_{b \in \mathbb{B}, \beta} J(b, \beta) \tag{25}$$

The formulation presented could be posed as a Mixed Integer Non-linear Programming (MINLP) problem, and there are available solvers. However, the formulation will be non-convex even for fixed integer variables, and global properties of the solution is difficult to ensure. The solution approach here can be considered more as a brute force strategy, but it is the author's belief that more general optimization solvers can be used in the future, and the formulation is done with this in mind. In this work, we first find what we consider the best path for each obstacle and orientation, subsequently this path is used for choosing between the different obstacles and orientations. Combining this into one optimization problem would be natural.

III. Case study

1) Snake robot simulator: In this section, we provide a short introduction to a mathematical modeling approach used as a basis for a snake robot simulator. The simulator is employed for evaluating the planning approach presented in this paper.

We have in previous works [9], [13] developed a snake robot simulator based on the framework of convex analysis and non-smooth dynamics. The simulator has previously been successfully verified with live experiments on snake robot obstacle-aided locomotion [9]. With this simulation tool, we simulate a 11 link planar snake robot with isotropic friction between each snake robot link and a horizontal ground plane. Moreover, the simulator includes friction-less contact dynamics between the snake robot and circular obstacles. The model also allows for adding Coulomb friction between the snake robot and the obstacles, but this has not been a focus for this paper and is a topic for future work. Each pair of adjacent links (i and $i-1$) are connected by a 1 degree of free rotational joint i. We employ a proportional-derivative controller in order to control each joint to a specific angle of rotation and rotational velocity:

$$\tau_i = -K_P(\phi_{i,d} - \phi_i) - K_D(\dot{\phi}_{i,d} - \omega_i) \tag{26}$$

where τ_i is the torque applied to joint i, K_P, K_D are the proportional and derivative gains, $\phi_{i,d}, \dot{\phi}_{i,d}$ are the reference joint angle and joint velocity, and ϕ_i, ω_i are the actual joint angle and joint velocity for joint i.

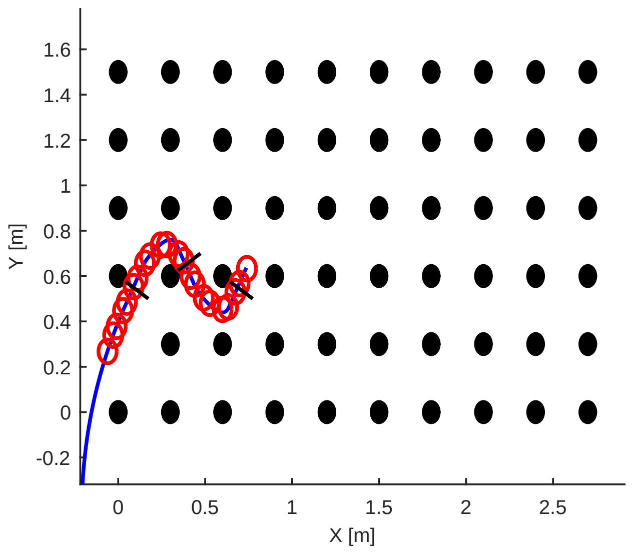

Fig. 4. Obstacles and initial configuration of the snake robot for case study

There are advantages to using non-smooth dynamics for snake robot obstacle aided locomotion. The framework allows for *instantaneous* changes in velocities. Such changes are usually associated with collisions. To this end, we can formulate in a uniform manner both the smooth and non-smooth phases of motion. This is important as obstacle aided locomotion has numerous and rapidly-changing robot-obstacle contact points.

The snake robot system parameters are the same as in [9], except the friction coefficient between the snake robot and the ground floor $\mu_T = 0.3$, numerical treatment parameters $r_H = 0.01$ and $r_T = 1.0$, proportional $K_P = 150$ Nm and derivative $K_D = 2$ Nm gains in the low-level controller for each joint.

The minimization problem in (21) was solved using Ipopt [14], with the maximum number of iterations set modestly to 20. Tuning of the parameters λ were only to provide adequate convergence to a path respecting the constraints. Optimality in terms of shortest length was not of importance in this work, the focus was on generating a feasible path. The result from the optimization was also tested with respect to feasibility, and in case of violation the result was disregarded, setting $\mathcal{L}^{b,\beta}$ to $\mathtt{inf}$. If a feasible path was found approaching from both left and right, the path with the shortest length was chosen of the two.

2) Case description: We generate an environment with obstacles regularly spaced on a Cartesian grid as depicted in Fig. 4. The obstacles are black circles with a radius of 0.05 m.

The initial configuration of the snake is manually constructed by a shape curve alternating sides of the obstacles, and an approach angle of 30 degrees. This is a configuration known to be advantageous for obstacle aided locomotion [9]. The initial configuration is also shown in Fig. 4, with the blue line representing the initial shape curve, red circles the snake robot and black lines the direction of the reaction force $\mathbf{N}_a$

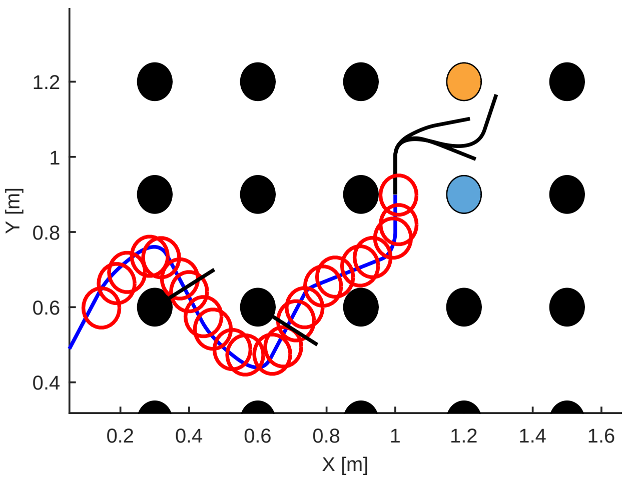

Fig. 5. Various paths considered at particular configuration

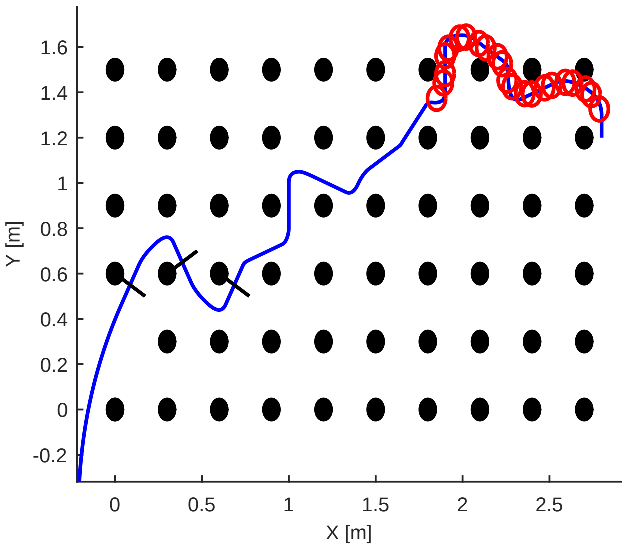

Fig. 6. Final path and configuration of snake robot

for the obstacle currently in contact with the snake robot.

When the snake robot reaches the end of the shape curve, the shape curve is extended by evaluating the different obstacles according to the previously defined objective function. The weighting term μ is set to 2. Each obstacle is evaluated for approach angles $\beta = [0\ 10 \ldots 350]$. The desired direction is set to be to the right, $\mathbf{R} = [1\ 0]^T$. In Fig. 5 three possible continuations of the path is shown.

The algorithm is able to find a path which allows the snake robot to traverse the obstacles in the desired direction. The constructed shape curve and configuration of the snake robot is shown in In Fig. 6. The motion towards the end of the curve was somewhat jerky and the entire snake robot did not follow the precise shape curve, however, the overall motion did follow the shape curve, and the snake was able to traverse the environment set up.

IV. Concluding Remarks and Further work

We have proposed a novel local path planning formulation for snake robots employing obstacle-aided locomotion. Simulations have shown the feasibility of the approach, but there is also room for substantial improvements. A stringent constraint imposed is that once the path is constructed, the entire snake robot should follow the path and no adjustment of the shape curve and contact angles are allowed. This assumption greatly simplifies the planning problem and solution space, and is a natural limitation to address in further work.

Another natural extension of the approach is to extend the horizon. It could be extended to a full global path planner, but for perception-driven planning it is more natural to extend it to a distance similar to the horizon of the snake robot's situational awareness.

References

[1] F. Sanfilippo, Ø. Stavdahl, and P. Liljebäck, "Snakesim: a ros-based control and simulation framework for perception-driven obstacle-aided locomotion of snake robots," *Artificial Life and Robotics*, pp. 1–10, 2018.

[2] M. C. Lee and M. G. Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," in *Proc. of the IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, vol. 2. IEEE, 2003, pp. 735–740.

[3] A. McLean and S. Cameron, "The virtual springs method: Path planning and collision avoidance for redundant manipulators," *The International journal of robotics research*, vol. 15, no. 4, pp. 300–319, 1996.

[4] Y. Shan, "Robot obstacle accommodation: Mechanics, control, and applications." Ph.D. dissertation, 1993.

[5] Y. Shan and Y. Koren, "Obstacle accommodation motion planning," *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 36–49, 1995.

[6] Y. Cheng, P. Jiang, and Y. F. Hu, "A distributed snake algorithm for mobile robots path planning with curvature constraints," in *Proc. of the IEEE International Conference on Systems, Man and Cybernetics*, 2008, pp. 2056–2062.

[7] M. J. Travers, J. Whitman, P. E. Schiebel, D. I. Goldman, and H. Choset, "Shape-based compliance in locomotion." in *Robotics: Science and Systems*, 2016.

[8] F. Sanfilippo, J. Azpiazu, G. Marafioti, 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.

[9] A. A. Transeth, R. I. Leine, C. Glocker, K. Y. Pettersen, and P. Liljebäck, "Snake robot obstacle-aided locomotion: Modeling, simulations, and experiments," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 88–104, 2008.

[10] P. Liljeback, K. Y. Pettersen, and O. Stavdahl, "Modelling and control of obstacle-aided snake robot locomotion based on jam resolution," in *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on.* IEEE, 2009, pp. 3807–3814.

[11] 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 International Conference on Robotics and Biomimetics (ROBIO).* IEEE, 2016, pp. 1845–1851.

[12] P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl, and J. T. Gravdahl, "A control framework for snake robot locomotion based on shape control points interconnected by bézier curves," in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on.* IEEE, 2012, pp. 3111–3118.

[13] A. A. Transeth, S. A. Fjerdingen, and P. Liljebäck, "Snake robot obstacle-aided locomotion on inclined and vertical planes: Modeling, control strategies and simulation," in *Proc. IEEE Int. Conf. Mechatronics (ICM 2013)*, 2013.

[14] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical programming*, vol. 106, no. 1, pp. 25–57, 2006.