Small Radius Spheres in Output Space of Nonholonomic Systems

Arkadiusz Mielczarek and Ignacy Duleba
Dept. of Cybernetics and Robotics, Wroclaw University of Science and Technology, Janiszewski St. 11/17, Wroclaw, Poland

Keywords: Nonholonomic System, Configuration Space, Output Space, Sphere, Motion Planning.

Abstract: In this paper small radius spheres of driftless nonholonomic systems in an output space are analyzed. The nonholonomic systems appear frequently in mobile robotics. An algorithm is provided to compute the spheres extensively using a directional optimization and spherical coordinates. Illustrating examples are provided for two two-input nonholonomic systems. Results presented in this paper are important in motion planning of nonholonomic systems with outputs as a ready-to-use receipt is given how to shift a point in an output space in a desired direction. In practice, an effective short-distance motion planner is required while planning a motion in a space polluted with obstacles.

1 INTRODUCTION

Recently, nonholonomic systems are frequently encountered in robotics. Wheel mobile robots (Altafini, 2001) (with trailers (Duleba, 2018)), nonholonomic manipulators (Nakamura et al., 2001), nonholonomic mobile manipulators (Bayle et al., 2003) free-floating robots (Vafa and Dubowsky, 1990) belong to this class. After modeling kinematics and dynamics of the systems, the next task is to plan their motion, i.e. design an algorithm to steer the systems from one point of a configuration space to another (Duleba, 1998; LaValle, 2006). Even more demanding task is to plan the motion optimally with respect to a distance to the goal or an energy expense quality functions. Usually, this task is difficult due to non-linearity of a system and a small number of controls to steer in a high dimensional configuration space. In order to simplify a little bit this task, in a robotic literature, a small radius spheres are considered when a short distance motion is planned. In this case the non-linearity is encapsulated in a local, around a given point in a configuration space, description of vector fields defining a system and their descendants. Such spheres for nonholonomic systems in a configuration space were analyzed in the scope of sub-Riemannian geometry (Jean, 2014). It is much simpler to predict how a small radius nonholonomic sphere in a configuration space looks like rather than to calculate its real shape. A more practical task arises when a nonholonomic system is considered together with an output function relating configurations with coordinates of an output space. Usually, a dimension of the output space is smaller than dimension of a configuration space as some coordinates of the latter space are not important (for example in a collision avoidance task). In this paper a task of computing an exact shape of small radius spheres in an output space of driftless nonholonomic systems is addressed. The shape is important in motion planning as it shows energy cheap and expensive directions of motion. A small range motion is also useful while planning a path in an obstacle cluttered environment. In a proposed method to get a small radius spheres in an output space a generalized Campbell-Baker-Hausdorff-Dynkin (gCBHD) formula will be exploited (Strichartz, 1987). Locally, around a given configuration, the formula allows to predict a motion in a configuration space corresponding to an assumed set of controls and vector fields generated from the system equations and evaluated at the configuration. When the motion is transferred into the output space via Jacobian of the output function and optimized within a space of controls a desired sphere is constructed.

The paper is organized as follows. In Section 2 a terminology is recalled and some required tools are presented. In Section 3 the task of construction of small radius spheres in an output space of nonholonomic systems is formulated and an algorithm to solve the task is presented. In Section 4 simulation results are provided. On two nonholonomic systems with different output functions sections of spheres are visualized. Section 5 concludes the paper.

Mielczarek, A. and Duleba, I.
Small Radius Spheres in Output Space of Nonholonomic Systems.
DOI: 10.5220/0007840003160322
In *Proceedings of the 16th International Conference on Informatics in Control, Automation and Robotics (ICINCO 2019)*, pages 316-322
ISBN: 978-989-758-380-3

2 MATHEMATICAL PRELIMINARIES

The gCBHD formula describes a trajectory of a differential non-autonomous system (Strichartz, 1987)

$$\dot{\boldsymbol{q}} = \boldsymbol{A}(t)(\boldsymbol{q}(t)) \tag{1}$$

initialized at a configuration $\boldsymbol{q}_0$ and given by

$$\boldsymbol{q}(t) = \exp \boldsymbol{z}(t)(\boldsymbol{q}_0)) \simeq \boldsymbol{z}(t)(\boldsymbol{q}_0) + \boldsymbol{q}_0, \tag{2}$$

where $\exp \boldsymbol{z}(t)(\boldsymbol{q}_0)$ is a solution of the following equation

$$\frac{d}{ds}\nu(s,t) = \boldsymbol{z}(t)\nu(s), \quad \nu(0,0) = \boldsymbol{q}_0.$$

Thus,

$$\nu(s,t) = exp(sz(t))(q_0) \Rightarrow \boldsymbol{q}(t) = \nu(1,t) = exp\boldsymbol{z}(t)(\boldsymbol{q}_0).$$

Later on, a special sub-class of general systems (1) will be considered, namely two-input driftless nonholonomic systems

$$\boldsymbol{A}(t)(\boldsymbol{q}(t)) = \sum_{i=1}^{m} \boldsymbol{X}_i(\boldsymbol{q})u_i, \text{ and } \quad m = 2. \tag{3}$$

In Eq. (3) vector fields $\boldsymbol{X}_i$ are called generators. Locally, when $t \to 0$, $\boldsymbol{z}(t)(\boldsymbol{q}_0)$ in Eq. (2) takes the form of a series (Strichartz, 1987)

$$\boldsymbol{z}(t)(\boldsymbol{q}_0) \simeq \sum_{r=1}^{\infty} \int_{T_r(t)} \left(\sum_{\boldsymbol{\sigma} \in P_r} c(\boldsymbol{\sigma}) \boldsymbol{E}_{\boldsymbol{\sigma}} \right) d\boldsymbol{s}^r, \tag{4}$$

where $\int_{T_r(t)} = \int_{s_r=0}^{t} \int_{s_{r-1}=0}^{s_r} \cdots \int_{s_1=0}^{s_2}$ is an r-dimensional simplex, $d\boldsymbol{s}^r = ds_1 \ldots ds_r$; P_r is a set of all permutations derived from the set $\{1,\ldots,r\}$;

$$\boldsymbol{E}_{\boldsymbol{\sigma}} = [[\ldots[\boldsymbol{A}(s_{\boldsymbol{\sigma}(1)}), \boldsymbol{A}(s_{\boldsymbol{\sigma}(2)})], \ldots], \boldsymbol{A}(s_{\boldsymbol{\sigma}(r)})] \tag{5}$$

and

$$c(\boldsymbol{\sigma}) = (-1)^{e(\boldsymbol{\sigma})} / \{ r^2 \binom{r-1}{e(\boldsymbol{\sigma})} \}, \tag{6}$$

where $e(\boldsymbol{\sigma})$ counts the number of errors in consecutive pairs of integers in a permutation $\boldsymbol{\sigma} = \{\sigma(1), \sigma(2), \ldots, \sigma(r)\}$, for example $e((1,2,3)) = 0$, $e((2,1,3)) = 1$, $e((3,2,1)) = 2$; $[\boldsymbol{V}, \boldsymbol{Z}]$ denotes a Lie bracket of vector fields $\boldsymbol{V}, \boldsymbol{Z}$ (Spivak, 1999).

In (4) many Lie brackets are to be calculated. In order to reduce the computational complexity of (4) two properties of vector fields, valid for any $\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}$, are applied

P1 the anti-symmetry: $[\boldsymbol{A}, \boldsymbol{B}] = -[\boldsymbol{B}, \boldsymbol{A}]$,

P2 the Jacobi identity:

$$[\boldsymbol{A}, [\boldsymbol{B}, \boldsymbol{C}]] + [\boldsymbol{C}, [\boldsymbol{A}, \boldsymbol{B}]] + [\boldsymbol{B}, [\boldsymbol{C}, \boldsymbol{A}]] = 0.$$

In general, vector fields are exemplifications of Lie monomials (other examples of Lie monomials are square matrices with the Lie bracket defined as

$$[\boldsymbol{X}, \boldsymbol{Y}] = \boldsymbol{X}\boldsymbol{Y} - \boldsymbol{Y}\boldsymbol{X}$$

and vectors in R^3 with the Lie bracket defined by a vector product $[\boldsymbol{X}, \boldsymbol{Y}] = \boldsymbol{X} \times \boldsymbol{Y}$). A degree of a Lie monomial (vector field) counts the number of generators appearing in the monomial. For exemplary generators $\boldsymbol{X}_1 = \boldsymbol{X}, \boldsymbol{X}_2 = \boldsymbol{Y}$ we have

$$\deg(\boldsymbol{X}) = 1, \deg([\boldsymbol{X}, \boldsymbol{Y}]) = 2, \deg([\boldsymbol{X}, [\boldsymbol{X}, \boldsymbol{Y}]]) = 3.$$

A layer contains all Lie monomials sharing the same degree. A minimal set of Lie monomials spanned by some generators is called a basis. Although all further considerations are quite general, two input systems will be used to illustrate concepts being discussed. Two input systems are the most demanding ones as the difference between the number of controls $m = 2$ and the dimension of a space where actions of controls are planned is the biggest one and more sophisticated control strategies are required to plan a motion towards a desired point. The most popular basis is the Ph. Hall one (abbreviated later on as PHB and effectively computed with an algorithm proposed in (Duleba and Khefifi, 2006)) and the first three layers of PHB spanned by $\boldsymbol{X}, \boldsymbol{Y}$ are the following

$$\begin{aligned} &(\overbrace{\boldsymbol{H}_1^1, \boldsymbol{H}_2^1}^{\boldsymbol{H}^1}, \overbrace{\boldsymbol{H}_1^2}^{\boldsymbol{H}^2}, \overbrace{\boldsymbol{H}_1^3, \boldsymbol{H}_2^3, \ldots}^{\boldsymbol{H}^3}) = \\ &= (\boldsymbol{X}, \boldsymbol{Y}, [\boldsymbol{X}, \boldsymbol{Y}], [\boldsymbol{X}, [\boldsymbol{X}, \boldsymbol{Y}]], [\boldsymbol{Y}, [\boldsymbol{X}, \boldsymbol{Y}]], \ldots) \end{aligned} \tag{7}$$

where $\boldsymbol{H}_d^r$ denotes the d-th element within the r-th layer of the PHB. For the two-input system (1), the operator (4) shifts a current state $\boldsymbol{q}_0$ to $\boldsymbol{q}_0 + \boldsymbol{z}(t)(\boldsymbol{q}_0)$ and it is expressed as a series of control-depended coefficients $\boldsymbol{\alpha}$ pre-multiplying vector fields (PHB elements) evaluated at a current configuration $\boldsymbol{q}_0$

$$\begin{aligned} \boldsymbol{z}(t) = &\alpha_1^1(t)\boldsymbol{X} + \alpha_2^1(t)\boldsymbol{Y} + \alpha_1^2(t)[\boldsymbol{X}, \boldsymbol{Y}] + \\ &+ \alpha_1^3(t)[\boldsymbol{X}, [\boldsymbol{X}, \boldsymbol{Y}]] + \alpha_2^3(t)[\boldsymbol{Y}, [\boldsymbol{X}, \boldsymbol{Y}]] + \ldots \end{aligned} \tag{8}$$

where the coefficients are equal to (Duleba and Khefifi, 2006)

$$\begin{aligned} &\alpha_1^1(t) = \int_0^t u_1(s_1)ds_1, \quad \alpha_2^1(t) = \int_0^t u_2(s_1)ds_1, \\ &\alpha_1^2(t) = \tfrac{1}{2} \int_0^t \int_0^{s_2} (u_1(s_1)u_2(s_2) - u_2(s_1)u_1(s_2))ds_1 ds_2, \\ &\ldots \end{aligned} \tag{9}$$

and depend on controls $\boldsymbol{u}$. It can be easily deduced that the number of items to calculate $\boldsymbol{\alpha}$ corresponding to high layers of vector fields will grow rapidly and

the integration of each item in Eq. (4) becomes more and more complex as product of controls integrated over a high dimensional simplex transfers variables from one integration to another making an expression to integrate more and more complex. For the very first few layers the calculations can be performed by hand, but as the number of layers increases, the calculation should be automized with the use of symbolic computation packages (Mielczarek and Duleba, 2018). The formula (8) is local (around at current configuration $\boldsymbol{q}_0$) as only in this case the tail of the series is negligible (note that all vector fields are evaluated at $\boldsymbol{q}_0$).

The system (1), (4) is (small time locally) controllable when its Lie algebra spanned by generators $\boldsymbol{g}_i(\boldsymbol{q})$ is of the full rank (Chow, 1939) or, equivalently, the Ph. Hall basis evaluated at any configuration is of the full rank

$$\forall \boldsymbol{q} \in \mathbb{Q} \quad \text{rankPHB}(\boldsymbol{q}) = n. \tag{10}$$

Condition (10) means that the system can evolve at any configuration in any direction. From a practical point of view not all coordinates of the configuration vector $\boldsymbol{q}$ are equally important. For example, for mobile robots the rotation angle of wheels is not important for checking collision of the robot. Therefore, an output function should be introduced to describe interactions of a system with its environment

$$\boldsymbol{x} = \boldsymbol{k}(\boldsymbol{q}), \quad \dim(\mathbb{X} \ni \boldsymbol{x}) = r. \tag{11}$$

Controllability of system (1), (4) with the output function (11) was discussed in (Duleba and Mielczarek, 2018). It appears that the system with output is controllable if

$$\forall \boldsymbol{x} \in \mathbb{X} \quad \text{rank}\, \boldsymbol{J}(\boldsymbol{k}^{-1}(\boldsymbol{x}))\boldsymbol{M}_{\text{PHB}}(\boldsymbol{k}^{-1}(\boldsymbol{x})) = r, \tag{12}$$

where $\boldsymbol{J}(\boldsymbol{q}) = \partial \boldsymbol{k}(\boldsymbol{q})/\partial \boldsymbol{q}$ is the Jacobian of the output function (11) and $\boldsymbol{M}_{\text{PHB}}$ is a matrix composed of elements of the Ph. Hall basis arranged in columns of the matrix.

3 TASK AND ALGORITHM

Having introduced indispensable notations and tools, we are in a position to define a task to be solved:
Task: given the system (1), (4), (11) and a configuration $\boldsymbol{q}_0$ find a small radius sphere in output space $\mathbb{X} \subset \mathbb{R}^r$ around point $\boldsymbol{k}(\boldsymbol{q}_0)$.

A sphere should have a small radius because the gCBHD formula is local, i.e. it reasonably well approximates a trajectory only if the trajectory does not leave a close neighborhood of $\boldsymbol{q}_0$ (in this case the very few items of the series (8) well approximate the infinite series). In order to satisfy this requirement, the time of applying controls is assumed to be fixed $t = T$ and the energy of controls $E(\boldsymbol{u})$ is constant and small.

It it is a common practice to express controls as linear combinations of time-dependent functions (Duleba, 1998; Jakubiak et al., 2010)

$$\boldsymbol{u}_i(t) = \sum_{j=0}^{N_i-1} \phi_j(t) p_i^j, \quad i = 1, \ldots, m, \tag{13}$$

where $\phi_j(t)$ are elements of any functional basis (polynomials, harmonic functions) and N_i is the number of variables required to describe the i-th control. Thus a vector $\boldsymbol{p}$ collects all variables p_i^j and uniquely described controls $\boldsymbol{u}$ and the energy of controls depends on $\boldsymbol{p}$.

In order to solve the task, a directional optimization technique will be applied. At first, a direction in the output space $\mathbb{R}^r$ is selected and the furthest point along this direction is searched for with an energy of motion fixed. Then, a collection of points for all admissible directions forms a sphere in an output space. In order to describe any direction in spherical coordinates $(\alpha_1, \ldots, \alpha_{r-1}, R)^T$ within $\mathbb{R}^r$ are particularly well suited. The angle coordinates $(\alpha_1, \ldots, \alpha_{r-1})^T$ describe a desired direction of motion while R its range

$$\begin{cases} w_1 = R\, c_{\alpha_1} \\ w_i = R \prod_{j=1}^{i-1} s_{\alpha_j} c_{\alpha_i} \quad i = 2, \ldots, r-1, \\ w_r = R \prod_{j=1}^{r-1} s_{\alpha_j} \end{cases} \tag{14}$$

where $c_{\alpha_j} = \cos(\alpha_j)$ and $s_{\alpha_j} = \sin(\alpha_j)$.

Detailed steps of solving the task are collected in the algorithm:

Step 1: Read-in a configuration $\boldsymbol{q}_0$. Compute necessary vector fields and evaluate them at $\boldsymbol{q}_0$ to form a matrix $\boldsymbol{M}_{\text{PHB}}(\boldsymbol{q}_0)$. For the output function (11) compute Jacobian $\boldsymbol{J}(\boldsymbol{q}_0)$

Step 2: Select a functional basis (13) and appropriate representation of controls (setting N_j in (13)) to form a vector of variables $\boldsymbol{p}$. Compute energy of controls and set its fixed (small) value E:

$$E(\boldsymbol{p}) = \sum_{i=1}^{m} \int_{t=0}^{T} \left(\sum_{j=0}^{N_i-1} \phi_j(t) p_i^j \right)^2 dt = E = const., \tag{15}$$

Step 3: Using Eq. (9), compute control dependent coefficients $\boldsymbol{\alpha}$ as a function of variables $\boldsymbol{p}$.

Step 4: Generate a mesh in $\mathbb{R}^{r-1} \ni (\alpha_1, \ldots, \alpha_{r-1})^T$ dimensional space of angle-vectors.

Step 5: For each (fixed at this stage) angle-vector $(\alpha_1, \ldots, \alpha_{r-1})^T$ repeat Steps 6-8.

Step 6: Using (14) form a set of constraints

$$\boldsymbol{w}(R) = \boldsymbol{J}(\boldsymbol{q}_0)\boldsymbol{M}_{\mathrm{PHB}}(\boldsymbol{q}_0), \quad \boldsymbol{w} = (w_1, \ldots, w_r)^T. \tag{16}$$

Step 7: Maximize the function

$$f(\boldsymbol{p}) = R, \tag{17}$$

with constraints (15), (16).

Step 8: Add the point $(\alpha_1, \ldots, \alpha_{r-1}, R)^T$ computed from Eq. (14) to the sphere in $\mathbb{R}^r$.

Step 9: Visualize the sphere (or its sections).

Some remarks concerning the algorithm:

- In Step 1 the number of vector fields has to be selected to satisfy controllability condition (12). However, when any vector field, say $\boldsymbol{Z}$ with degree $z = deg(\boldsymbol{Z})$, is used to check the condition also all other vector fields with the degree equal to z should be included into the matrix $\boldsymbol{M}_{\mathrm{PHB}}(\boldsymbol{q}_0)$ as the vector fields have the same impact on resulting series (8) as the vector field $\boldsymbol{Z}$, (Duleba, 1998).
- In Step 2, a reasonable compromise between accuracy and computational complexity should be preserved. Obviously, $\dim \boldsymbol{p} \geq r$. Increasing $\dim \boldsymbol{p}$ also accuracy of shaping the sphere increases but also (rapidly) increases a computational complexity of the optimization task solved in Step 7. Therefore, only a small redundancy in selecting $\dim \boldsymbol{p}$ is advised.
- In Step 7 a classical optimization task with constraints is to be solved. A Lagrange multiplier technique is appropriate for this task (Bertsekas, 1996).

4 SIMULATIONS

Two systems (3) with two-inputs are considered differing in dimensions of the configuration space. The first model is the unicycle, with a configuration $\boldsymbol{q} = (x, y, \theta)^T$, where the θ angle orients the robot with respect to the x-axis, and (x, y) are positional coordinates of its axle. The unicycle is described by the following equation

$$\dot{\boldsymbol{q}} = (c_\theta, s_\theta, 0)^T u + (0, 0, 1)^T v = \boldsymbol{X}(\boldsymbol{q})u + \boldsymbol{Y}(\boldsymbol{q})v, \tag{18}$$

where $c_\theta = \cos(\theta)$ and $s_\theta = \sin(\theta)$. The set of vector fields $\boldsymbol{X}, \boldsymbol{Y}$ and $[\boldsymbol{X}, \boldsymbol{Y}] = (s_\theta, -c_\theta, 0)^T$ is independent for any $\boldsymbol{q}$, thus the system is nonholonomic and a small time controllable (Chow, 1939).

The second model is a kinematic car (with the configuration $\boldsymbol{q} = (x, y, \theta, \psi)^T$ where the additional coordinate ψ denotes an angle of rotation of its wheel) described by the equation

$$\begin{aligned} \dot{\boldsymbol{q}} &= (c_\theta c_\psi, s_\theta c_\psi, s_\psi, 0)^T u + (0, 0, 0, 1)^T v \\ &= \boldsymbol{X}(\boldsymbol{q})u + \boldsymbol{Y}(\boldsymbol{q})v. \end{aligned} \tag{19}$$

After calculating

$$[\boldsymbol{X}, \boldsymbol{Y}] = (c_\theta s_\psi, s_\theta s_\psi, -s_\psi, 0)^T,$$
$$[\boldsymbol{X}, [\boldsymbol{X}, \boldsymbol{Y}]] = (-s_\theta, c_\theta, 0, 0)^T, [\boldsymbol{Y}, [\boldsymbol{X}, \boldsymbol{Y}]] = \boldsymbol{X}$$

it can discovered that the system (19) is also nonholonomic and a small time controllable as the vector fields $\boldsymbol{X}, \boldsymbol{Y}, [\boldsymbol{X}, \boldsymbol{Y}], [\boldsymbol{X}, [\boldsymbol{X}, \boldsymbol{Y}]]$ are independent everywhere.

Controls are represented in a harmonic basis, thus Eq. (13) takes the form

$$\boldsymbol{u}_i(t) = p_i^1 + \sum_{j=1}^{N_i} (p_i^{2j} \sin(j\omega t) + p_i^{2j+1} \cos(j\omega t)) \tag{20}$$

where $\omega = 2\pi/T$ and $i = 1, 2$.

In all simulations the controls act on the interval $[0, T]$ with the time horizon T fixed. The vector collecting all variables $\boldsymbol{p}$ contains only components with $p_i^j \neq 0$. Output functions used in simulations are constructed by selecting some components from the configuration. For the unicycle and the kinematic car this selection of output functions has its physical meaning (e.g. for the kinematic car, the function $\boldsymbol{k}(\boldsymbol{q}) = (x, y, \theta)^T$ can be used in planning a motion in a parallel parking task).

In all simulations a zero vector, of an appropriate size, is selected for the initial configuration $\boldsymbol{q}_0$ (Eqns. (18), (19) do not depend on x, y and the initial θ can be set to zero by appropriate rotation of (x, y) plane), while the energy $E(\boldsymbol{p})$ and the time of motion T are fixed and equal to 1.

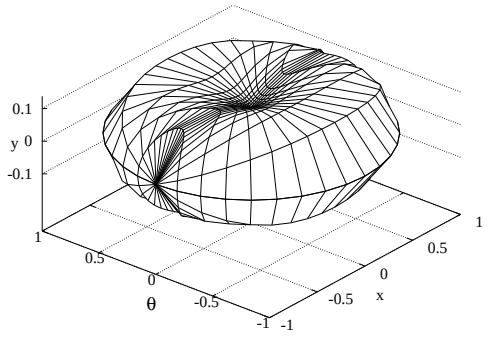

Figure 1: A nonholonomic sphere of the unicycle.

In Fig. 1 a nonholonomic sphere of the unicycle is illustrated with the identity output function $\boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{q}$ and controls with full harmonics up to degree one in

both controls

$$\begin{cases} u_1(t) = p_1^1 + p_1^2 \sin(\omega t) + p_1^3 \cos(\omega t), \\ u_2(t) = p_2^1 + p_2^2 \sin(\omega t) + p_2^3 \cos(\omega t). \end{cases} \quad (21)$$

A regular mesh used to describe possible directions of motions is given as follows

$$(\alpha_1, \alpha_2) = (\frac{\pi i}{18}, \frac{\pi j}{18}),\ i = 0, 1, \ldots, 35;\ j = 0, 1, \ldots, 18.$$

The sphere is radially symmetric with respect to the y axis and the pure motion along the y-axis is least energy-efficient.

For a given selection of $\boldsymbol{\alpha} = (\alpha_1, \alpha_2)^T$ an elementary optimization task has been solved to get the vector $\boldsymbol{p}$ realizing the maximum value of the distance R. Next, the value of $\boldsymbol{p}$ was substituted into Eq. (21) to determine controls and a numerical integration of the model (18) initialized at $\boldsymbol{q}_0$ with the controls $\boldsymbol{u}$ was invoked. The final point of the trajectory generated contributed into a numerically obtained nonholonomic sphere visualized in Fig. 2. This sphere is a close approximation of the real nonholonomic sphere for the unicycle. Both spheres are interlaced in Fig. 3.

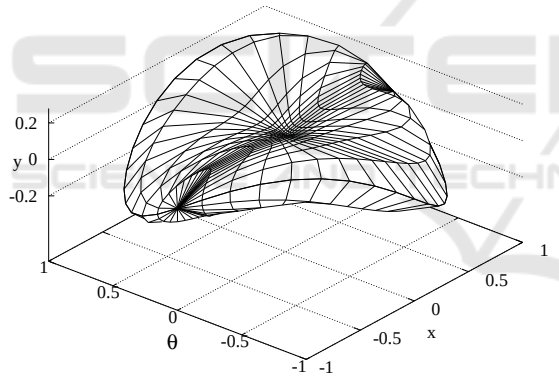

Figure 2: A nonholonomic sphere of the unicycle obtained with a numerical integration.

It can be observed that the spheres differ slightly as the sphere obtained with the algorithm and based on the gCBHD formula neglects the higher degree vector fields while the numerical procedure takes them into account implicitly. Nevertheless, the difference is small enough to plan a motion with the algorithm in a desired direction.

In Fig. 4 a section of spheres from Fig. 3 for $\theta = 0$ (the $x-y$ plain) is presented. The solid line corresponds to the sphere generated by the algorithm while the dashed line – to a numerical integration of the model. At this sections the differences are really small. From the figure one can learn also that the motion along the second degree vector field $[\boldsymbol{X}, \boldsymbol{Y}]$ acting

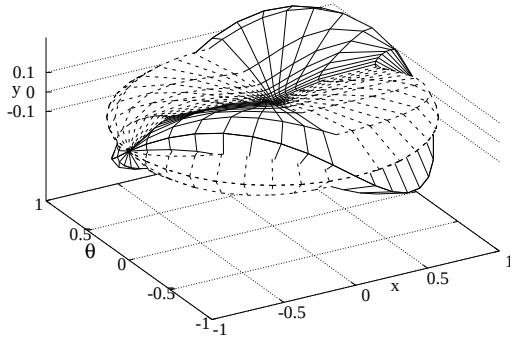

Figure 3: Nonholonomic spheres of the unicycle obtained with the algorithm presented (dashed) and the numerical integration.

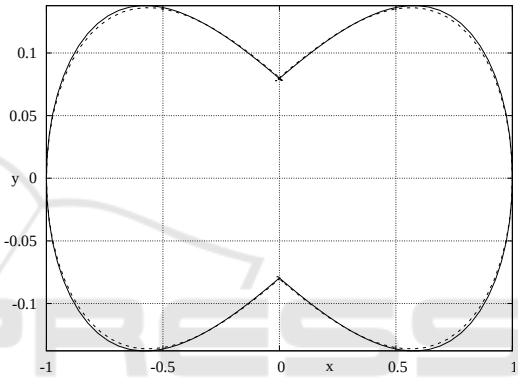

Figure 4: The section of nonholonomic spheres of the unicycle for $\theta = 0$.

at $\boldsymbol{q}_0$ along y-axis is significantly less effective than a motion along x-axis corresponding to the first degree vector field $\boldsymbol{X}$.

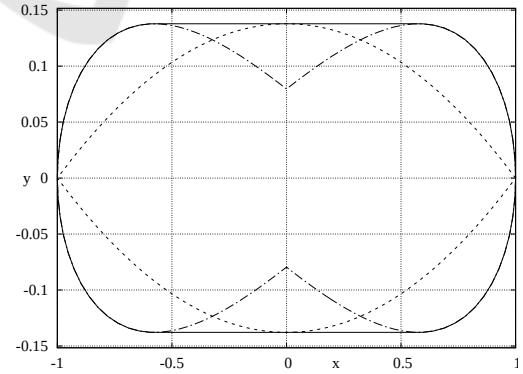

Figure 5: The unicycle with a projection onto $x-y$ plane output function. The section for $\theta = 0$ and three selections of controls.

Now an impact of representation of controls and output functions on sections of nonholonomic spheres for the unicycle robot are investigated. In Fig. 5 and

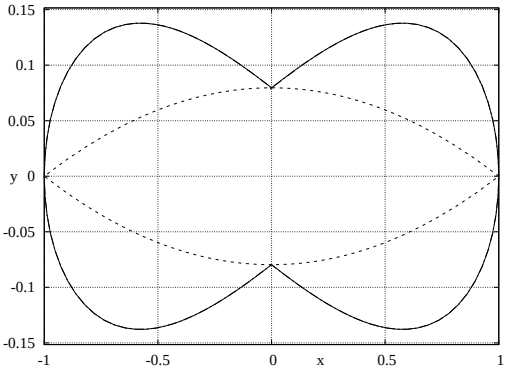

Figure 6: The unicycle with the identity output function. The section for $\theta = 0$ and three selections of controls (the solid line coincides with the dot-dashed one).

Fig. 6 three sections are presented for the output function $k(\boldsymbol{q}) = (x,y)^T$ and $k(\boldsymbol{q}) = (x,y,\theta)^T$, respectively: solid line corresponds to controls given in Eq. (21), the dashed line was obtained within the family of controls

$$\begin{cases} u_1(t) = p_1^1 + p_1^2 \sin(\omega t), \\ u_2(t) = p_2^1 + p_2^3 \cos(\omega t). \end{cases} \tag{22}$$

while the dot-dashed line corresponds to the controls

$$\begin{cases} u_1(t) = p_1^1 + p_1^3 \cos(\omega t), \\ u_2(t) = p_2^1 + p_2^2 \sin(\omega t). \end{cases} \tag{23}$$

In all cases the algorithm presented in this paper has generated the required data. From Figs. 5. 6, it can be deduced that the representation of controls is very important as spheres (and their sections) generated can differ in shapes significantly. Even when the representation is of the same length (as in Eq. (22) and (23) still the differences are visible). In practical situations one should solve the following problem: how to get reliable results keeping a representation of controls as small as possible (as it significantly decreases computational costs).

For a kinematic car the output function is selected as the identity $\boldsymbol{k}(\boldsymbol{q}) = \boldsymbol{q}$ and controls are full harmonics up to the order of two

$$\begin{cases} u_1(t) = p_1^1 + \sum_{j=1}^{2} (p_1^{2j} \sin(j\omega t) + p_1^{2j+1} \cos(j\omega t)), \\ u_2(t) = p_2^1 + \sum_{j=1}^{2} (p_2^{2j} \sin(j\omega t) + p_2^{2j+1} \cos(j\omega t)). \end{cases} \tag{24}$$

An irregular mesh of 47 points discretizing the α_1 angle and 43 points for the α_2 angle are selected to discretize the space of $\boldsymbol{\alpha}$ angles. The purpose to use the irregular mesh instead of regular one is to keep the computational complexity as low as possible without deteriorating the quality of retrieving a shape of spheres. While using a regular mesh, the vast majority of points are located in the area close to planes

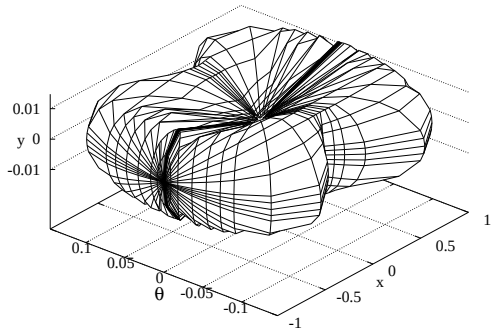

Figure 7: A section of the nonholonomic sphere of the kinematic car obtained with the algorithm for $\psi = 0$.

$x = 0$ or $\theta = 0$. In Fig. 7 a section of the nonholonomic sphere for the kinematic car calculated with the algorithm is presented when $\psi = 0$. In Fig. 8 sections of the sphere of the kinematic car are presented for $(\theta = \psi = 0)$ (on the $x-y$ plane). As before, the solid curve is generated with the algorithm while dashed one with a numerical integration of the model (19). The section shapes presented in Fig. 8 are similar, but with different values near the y axis. It is noticeable a large difference between the maximal values of x and y coordinates of points along the curves as the vector field acting along x and y axis are of different degrees.

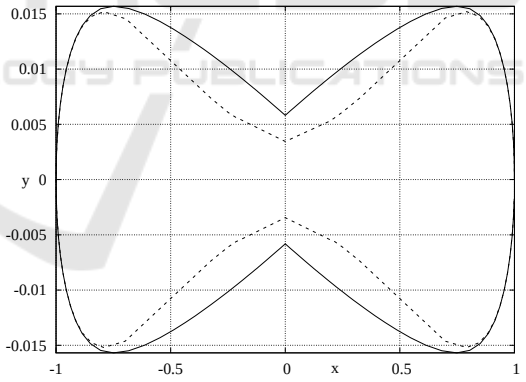

Figure 8: A section of the nonholonomic sphere of the kinematic car obtained with the algorithm for $\theta = \psi = 0$.

Programs used in simulations were written in Wolfram Mathematica, version 11.3, (Wolfram Research, 2018) and run on a computer with Intel® Core™ i5-8400 CPU and 8 GiB RAM memory. The computational time to construct a sphere for the unicycle was $210[s]$ and 36×19 elementary tasks (a directional optimization in a particular direction determined by fixing angles (α_1, α_2)) were solved which gives $0.3[s]/\text{task}$. The total computational time to construct a kinematic car sphere's section ($\psi = 0$) was $390[min]$ and 47×43 elementary tasks were solved

which gives $11.5[s]/\text{task}$.

It should be pointed out that the `NMinimize` function was used extensively to solve elementary tasks. The function tries to find the global optimum and it is time consuming. In real applications some dedicated procedures should be implemented to reduce computational costs, form example by taking into account results of previous optimizations and incorporate a local optimization (e.g. some variants of the Newton algorithm, (Nakamura, 1991)) instead of the global one.

5 CONCLUSIONS

In this paper small radius spheres were examined for nonholonomic systems considered with accompanying output functions. The presented algorithm to derive the spheres is based on the generalized Campbell-Baker Hausdorff-Dynkin formula and locally, around a given point in the configurations space, shrinks the series generated with this formula to leave only the small number of items required to preserve a small time local controllability of the system. To derive a reliable shape of the sphere a large number of optimization tasks should be solved. It was shown how to decrease the dimension of the tasks being solved by one. The simulation results shown that the selection of a representation of controls is crucial in deriving reliable shapes of the spheres. The representation should be rich enough to get constructed spheres reliable. Unfortunately, too long representations dramatically increase computational costs as many tasks with a nonlinear quality function and nonlinear constraints should be solved. Results of the paper can be used directly in motion planning algorithms of nonholonomic systems with an output function. In the algorithms, at a current configuration, only one optimization task is to be solved, thus computational costs are reasonably low.

ACKNOWLEDGEMENT

The first author was supported by the Young Researchers' Program under Grant No. 0402/0158/18 while the second author was supported by the WUST statutory grant No. 0401/0022/18.

REFERENCES

Altafini, C. (2001). Some properties of the general n-trailer. *Int. Journal of Control*, 74(4):409–424.

Bayle, B., Renaud, M., and Fourquet, J. (2003). Nonholonomic mobile manipulators: Kinematics, velocities and redundancies. *J. Intell. Robot. Syst.*, 36(1):45–63.

Bertsekas, D. (1996). *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, Belmont, Massachusetts.

Chow, W. (1939). Über Systeme von linearen partiellen Differentialgleichungen erster Ordnung. *Mathematische Annalen*, 117(1):98–105.

Duleba, I. (1998). *Algorithms of Motion Planning for Nonholonomic Robots*. WUST Publ. House, Wroclaw.

Duleba, I. (2018). Kinematic models of doubly generalized n-trailer systems. *Journal of Intelligent and Robotic Systems*, pages 1–8. on-line.

Duleba, I. and Khefifi, W. (2006). Pre-control form of the gcbhd formula for affine nonholonomic systems. *Systems and Control Letters*, 55(2):146–157.

Duleba, I. and Mielczarek, A. (2018). Controllability of driftless nonholonomic systems in a task-space. In Tchon, K. and Zielinski, C., editors, *Advances in Robotics*, pages 311–318, Publ. House of the Warsaw Univ. of Technology. in Polish.

Jakubiak, J., Tchon, K., and Magiera, W. J. (2010). Motion planning in velocity affine mechanical systems. *International Journal of Control*, 83(9):1965–1974.

Jean, F. (2014). *Control of Nonholonomic Systems: from Sub-Riemannian Geometry to Motion Planning*. SpringerBriefs in Mathematics. Springer.

LaValle, S. (2006). *Planning Algorithms*. Cambridge University Press.

Mielczarek, A. and Duleba, I. (2018). Theoretical and algorithmic aspects of generating pre-control form of the gcbhd formula. In *23nd Int. Conf. on Methods and Models in Automation and Robotics*, pages 905–909, Miedzyzdroje, Poland.

Nakamura, Y. (1991). *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley.

Nakamura, Y., Chung, W., and Sordalen, O. (2001). Design and control of the nonholonomic manipulator. *IEEE Trans. Robot. Autom.*, 17(1):48–59.

Spivak, M. (1999). *A comprehensive introduction to differential geometry*. Publ or Perish Inc., Houstron, Tx., 3rd edition.

Strichartz, R. (1987). The Campbell-Baker-Hausdorff-Dynkin formula and solutions of differential equations. *Journ. of Functional Analysis*, 72:320–345.

Vafa, Z. and Dubowsky, S. (1990). The kinematics and dynamics of space manipulators: the virtual manipulator approach. *Int. J. Robot. Research*, 9(4):3–21.

Wolfram Research, I. (2018). Mathematica, Version 11.3. Champaign, IL, 2018.