Symbolically Synthesized Motion Primitives for Autonomous Navigation

Zhixin Zhao, Antoine Girard, Sorin Olaru

▶ **To cite this version:**

Zhixin Zhao, Antoine Girard, Sorin Olaru. Symbolically Synthesized Motion Primitives for Autonomous Navigation. IEEE Conference on Decision and Control, Dec 2024, Milan, Italy. hal-04678873

HAL Id: hal-04678873
https://hal.science/hal-04678873v1

Submitted on 27 Aug 2024

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Symbolically Synthesized Motion Primitives for Autonomous Navigation

Zhixin Zhao[1], Antoine Girard[1], Sorin Olaru[1]

***Abstract*— This paper proposes a novel approach to navigationfor autonomous vehicles that leverages symbolic control methods and system translational and rotational invariance properties. By decomposing in-plane motions into translations and rotations, the approach constructs corresponding motion primitives that enable efficient offline controller design and avoid computationally expensive discretization of the whole state space. The resulting controllers achieve complex trajectories through concatenation of motion primitives. At the same time the safe corridor given by this method will provide safety guarantee for the whole mission.**

I. INTRODUCTION

In this paper, we address one of the fundamental problems in autonomous navigation: given a series of waypoint regions, how to determine as simply as possible the required control input to navigate the specified region. The problem has received the attention of different authors with a wide range of solutions from MPC [8], [12] to mixed integer optimization [1], [9] or potential fields techniques [3], [6] to mention just a few.

When discussing the simplicity, the motion primitives [2], [4] are considered effective to address this problem as they decompose the movement into atomic elements able to be performed sequentially in order to achieve a global mission. When pre-designed offline, the motion primitives offer a library of control input sequences that can be utilized by a path planner to produce feasible basic trajectories. These feasible trajectories can be combined online with small computational effort to generate more complicated trajectories. Based on this philosophy there are some applications of motion primitive as in [5], [11]. These primitives are designed to accommodate nonlinear dynamics and control input constraints while ensuring computational efficiency.

The contribution of this paper resides in the redefinition of the concept of motion primitives for planar motion by exploiting a certain invariance of the underlying dynamics and subsequent offline design of a state feedback controller using symbolic control techniques [14]. Symbolic control makes it possible to synthesize controllers for nonlinear systems with state and input constraints by solving an associated discrete synthesis problem in a symbolic domain [7], [13], [14]. Specifically, we have designed two motion primitives, particularly suitable for planar dynamics: the rotation primitive and the translation primitive. We represent them in the form of set combinations, and by applying invariant properties under specific coordinate transformations, the controllers corresponding to these two primitives can be reused. By concatenating these motion primitives, we can realize the online tracking of arbitrary waypoints in the 2D plane. At the same time, the safe corridor under the action of the controller will be given to ensure the safety of the control. Compared to an approach fully relying on symbolic control (see e.g. [13]), our approach does not require discretizing the full state space, thus greatly reducing the computational complexity, while providing the guarantees of formal correctness.

The paper is structured as follows. Section II formulates the problem, specifies the class of systems under consideration and delineates the control objective and methodology. Section III introduces the motion primitive design. Section IV outlines the methodology for online tracking using designed motion primitives. Finally, Section V provides a numerical example showing relevant results.

II. PROBLEM FORMULATION

Consider a forward complete nonlinear discrete-time system of the form

$$z(t+1) = f(z(t), u(t)),\ t \in \mathbb{N} \tag{1}$$

where $z(t) \in \mathbb{R}^n$ denotes the state of the system, and $u(t) \in \mathbb{R}^m$ represents the control input. In addition, the system is subject to constraints on state and input variables characterized by sets $\mathbb{Z} \subseteq \mathbb{R}^n$ and $\mathbb{U}$ a compact subset of $\mathbb{R}^m$, respectively.

Within this framework we are interested in systems describing the evolution of a vehicle in a planar environment. Hence, the state vector is assumed to be composed of the following components $z(t) = (x_1(t), x_2(t), \theta(t), \omega(t))$ where $x_1(t), x_2(t) \in \mathbb{R}$ represent the planar coordinates and $\theta(t) \in \mathbb{R}$ is an angle representing the heading of the vehicle. The additional vector $\omega(t) \in \mathbb{R}^l$ consists of all other possible states necessary to model the motion independent of coordinates and heading (e.g. velocity, acceleration or actuator dynamics). If $l = 0$, then $z(t) = (x_1(t), x_2(t), \theta(t))$. The simplest example of such nonlinear system is the unicycle model expressed as:

$$\begin{aligned} x_1(t+1) &= x_1(t) + u_1(t)\cos(x_3(t)) \\ x_2(t+1) &= x_2(t) + u_1(t)\sin(x_3(t)) \\ \theta(t+1) &= \theta(t) + u_2(t) \end{aligned} \tag{2}$$

We assume in the following that the state constraints for (1) is of the form $\mathbb{Z} = \mathbb{X} \times \mathbb{R} \times \Omega$ where $\mathbb{X} \subseteq \mathbb{R}^2$ and Ω is a compact subset of $\mathbb{R}^l$.

[1] Université Paris-Saclay, CNRS, CentraleSupélec, Laboratoire des signaux et systèmes, 91190, Gif-sur-Yvette, France. firstname.lastname@centralesupelec.fr

A. Structural properties of the dynamics

It is assumed that the dynamics of (1) possess invariance properties under specific coordinates transformations. In continuous time, this notion has been explored according to symmetry property [5]. In the present work, a discrete-time framework will be considered and the system dynamics will be assumed to exhibit rotational and translational invariance as formally described next.

Assumption 1: Let us denote $\delta f(z,u) = f(z,u) - z$ and

$$V(v_1, v_2, \alpha) = \begin{bmatrix} v_1 \\ v_2 \\ \alpha \\ \mathbf{0}_l \end{bmatrix}, \; M(\alpha) = \begin{bmatrix} R(\alpha) & 0 & 0 \\ * & 1 & 0 \\ * & * & I_l \end{bmatrix}$$

where $R(\alpha)$ is the rotation matrix of angle α:

$$R(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{bmatrix}.$$

Then, for all $z \in \mathbb{Z}$, $u \in \mathbb{U}$, and $v_1, v_2, \alpha \in \mathbb{R}$, it holds

$$\delta f\left(z + V(v_1, v_2, \alpha), u\right) = M(\alpha)\delta f\left(z, u\right). \tag{3}$$

Proposition 1: Under Assumption 1, consider initial states $z_0, z_0' \in \mathbb{Z}$ with $z_0' = z_0 + V(v_1, v_2, \alpha)$. Consider an input sequence $u(\cdot)$ and the associated trajectories $z(\cdot)$, $z'(\cdot)$ of (1) starting from z_0 and z_0', respectively. Then, for all $t \in \mathbb{N}$, it holds

$$z'(t) - z_0' = M(\alpha)(z(t) - z_0). \tag{4}$$

Proof: We proceed by induction. Clearly, (4) holds at $t = 0$. Let us assume that (4) holds at some $t \in \mathbb{N}$, then it follows that

$$z'(t) - z(t) = z_0' - M(\alpha)z_0 + (M(\alpha) - I_n)z(t).$$

From the structure of $M(\alpha)$ and since $z_0' = z_0 + V(v_1, v_2, \alpha)$, we can conclude that there exists $v_1(t), v_2(t) \in \mathbb{R}$ such that $z'(t) = z(t) + V(v_1(t), v_2(t), \alpha)$. Then, from (3), we get

$$z'(t+1) - z'(t) = M(\alpha)(z(t+1) - z(t)).$$

This equation, together with the induction hypothesis yields

$$\begin{aligned} z'(t+1) - z_0' &= z'(t+1) - z'(t) + z'(t) - z_0' \\ &= M(\alpha)(z(t+1) - z(t)) + M(\alpha)(z(t) - z_0) \\ &= M(\alpha)(z(t+1) - z_0). \end{aligned}$$

Hence, (4) holds for all $t \in \mathbb{N}$. ∎

B. Control objective and methodology

The objective is to design a methodology able to handle autonomous navigation missions. We assume that we are given a sequence of waypoints in $\mathbb{X}$, characterized by regions $\mathbb{X}_0, \mathbb{X}_1, \ldots, \mathbb{X}_N \subseteq \mathbb{X}$. We aim at synthesizing a feedback control strategy such that the vehicle starting from any position of $\mathbb{X}_0$ reaches the regions $\mathbb{X}_1, \ldots, \mathbb{X}_N$, in that order. This is formally stated in the following:

Problem 1: Given a system of the form (1) satisfying Assumption 1, state and input constraints $\mathbb{Z} = \mathbb{X} \times \mathbb{R} \times \Omega$ and $\mathbb{U}$, and a sequence of waypoint $\mathbb{X}_0, \mathbb{X}_1, \ldots, \mathbb{X}_N \subseteq \mathbb{X}$, synthesize a feedback control strategy such that for all initial states $z_0 = (x_{1,0}, x_{2,0}, \theta_0, \omega_0) \in \mathbb{X}_0 \times \mathbb{R} \times \Omega$, the closed-loop trajectory of (1) satisfies the following

$$\begin{aligned} &\exists 0 \leq t_1 \leq \cdots \leq t_N, \text{ such that} \\ &\qquad z(t_i) \in \mathbb{X}_i \times \mathbb{R} \times \Omega \text{ and } \forall t \leq t_N, \; z(t) \in \mathbb{Z}. \end{aligned} \tag{5}$$

In this paper, we propose a methodology for solving *Problem* 1 based on the following principles:

- The navigation problem is decomposed in a sequence of motion primitives chosen carefully so as to ensure sequencial feasibility and safety of the path;
- The motion primitives are designed to offer a rich selection for navigation but in the same time they represent simple geometric transformations to a library of elementary motion primitives (rotation or translation);
- For each elementary motion primitive, a "correct-by-design" controller is synthesized using symbolic control techniques.

III. PRIMITIVES FOR MOTION CONTROL

This section aims to introduce the concept of motion primitives considered in this paper. The elementary ones will be described first and then the associated controller synthesis problem will be addressed. Once these elements are available we will broaden the scope and discuss the design knobs.

A. Elementary motion primitive

Motion primitives are formally defined as follows:

Definition 1: A *motion primitive* $\mathcal{P}$ is given by triple of sets:

- the *starting set* $S_0 \subseteq \mathbb{R}^n$,
- the *target set* $T_f \subseteq \mathbb{R}^n$, and
- the *safety corridor* $\hat{S} \subseteq \mathbb{R}^n$, with $S_0, T_f \subseteq \hat{S}$.

and is denoted $\mathcal{P} = \langle S_0, T_f, \hat{S} \rangle$. A motion primitive is said to be *feasible* if there exists a controller $C_{\mathcal{P}} : \hat{S} \to \mathbb{U}$ and a time bound $p \in \mathbb{N}$ such that all closed-loop trajectories of (1) initialized in the starting set S_0 reach the target set T_f in at most p time steps, while staying in the safety corridor $\hat{S}$ along the way.

We denote the set mapping representing the one-step constrained predecessor of a set $T \subseteq \mathbb{R}^n$:

$$Pre_f(T, \hat{S}) = \left\{ z \in \hat{S} | \; \exists u \in \mathbb{U}, f(z,u) \in T \right\}.$$

Then, let us define iteratively the sequence of sets given by $Pre_f^0(T_f, \hat{S}) = T_f$ and for all $i \geq 1$:

$$Pre_f^i(T_f, \hat{S}) = Pre_f(Pre_f^{i-1}(T_f, \hat{S}), \hat{S}).$$

Essentially, $Pre_f^i(T_f, \hat{S})$ denotes the set of states from which trajectories of (1) can reach the target T_f in exactly i steps while staying in $\hat{S}$ along the way.

Proposition 2: A motion primitive $\mathcal{P}$ is feasible if and only if there exists $p \in \mathbb{N}$ such that

$$S_0 \subseteq \bigcup_{i=0}^{p} Pre_f^i(T_f, \hat{S}). \tag{6}$$

Proof: The proof is a direct consequence of the fact that $T_f \cup Pre_f^1(T_f, \hat{S}) \cup \cdots \cup Pre_f^i(T_f, \hat{S})$ is the set of states from which one can control trajectories of (1) to reach the target T_f in at most i steps while staying in $\hat{S}$ along the way. Then, (6) ensures feasibility of motion primitive $\mathcal{P}$. Moreover, one can choose any controller $C_{\mathcal{P}} : \hat{S} \to \mathbb{U}$ such that for all $i = 1, \ldots, p$

$$\forall z \in Pre_f^i(T_f, \hat{S}),\ f(z, C_{\mathcal{P}}(z)) \in \bigcup_{j=0}^{i-1} Pre_f^j(T_f, \hat{S}).$$

■

Remark 1: Let us denote by $p^*(\mathcal{P})$ the smallest $p \in \mathbb{N}$ such that (6) holds. $p^*(\mathcal{P})$ provides an upper-bound on the time needed to execute the motion primitive $\mathcal{P}$. We want to emphasize that for two motion primitives $\mathcal{P} = \langle S_0, T_f, \hat{S} \rangle$ and $\mathcal{P}' = \langle S_0', T_f', \hat{S}' \rangle$, with $S_0 \supseteq S_0'$, $T_f \subseteq T_f'$ and $\hat{S} \subseteq \hat{S}'$ it holds that $p^*(\mathcal{P}) \geq p^*(\mathcal{P}')$.

By selecting the sets S_0 and T_f, various movement patterns can be derived, ensuring that the trajectory remains within the pre-defined safe corridor $\hat{S}$. In particular, there are two movements that will be considered as elementary primitives in the present work:

- The translation in a pre-defined direction (without loss of generality we will consider this to be the positive direction of x_1):

$$Proj_\theta(S_0) \subseteq [-\delta_\theta, \delta_\theta],$$
$$T_f = \{z(t) + V(v_1, 0, 0) | z(t) \in S_0, v_1 \geq 0\}$$

  where $Proj_\theta$ denotes the projection over the θ variable.
- The rotation from an arbitrary initial heading towards a pre-defined direction (here, without loss of generality, the positive direction of x_1):

$$[-\pi, \pi] \subseteq Proj_\theta(S_0),$$
$$Proj_{x_2}(T_f) \subseteq [-\delta_x, \delta_x],$$
$$Proj_\theta(T_f) \subseteq [-\delta_\theta, \delta_\theta].$$

  where δ_x, δ_θ are the admissible tolerance on the respective state components.

For convenience, we use the simplest geometries, rectangles (hyperrectangles in high dimensions), to specify the motion primitives above as in Fig.1.

B. Synthesis

The objective of this section is to present the methodology employed in constructing controllers for motion primitives through the use of symbolic control techniques. Here we briefly recall the symbolic control techniques (and rely on e.g. [14], [13], [7] for further implementation details).

Symbolic control involves over-approximating the dynamics of (1) by that of a transition system with finite sets of states and inputs. The states and inputs of the transition systems correspond to symbols, each of which represents a subset of the states and inputs of (1), obtained by discretizing both the state space and the control input (see e.g. [13] for

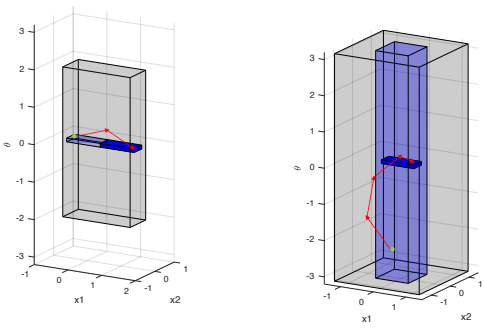


Fig. 1: Examples of motion primitives in the 3-dimensional state space, corresponding to (2): translation primitive (left); rotation primitive (right). The initial set S_0, the target set T_f and the safe corridor $\hat{S}$ are represented in light blue, dark blue, and gray, respectively.

implementation details). This approach allows us to represent a continuous model as a transition system $\Sigma = (\mathcal{Q}, \mathcal{U}, \mathcal{F})$, where $\mathcal{Q}$ denotes a set of symbolic states, $\mathcal{U}$ signifies the set of symbolic inputs, and $\mathcal{F} : \mathcal{Q} \times \mathcal{U} \to 2^{\mathcal{Q}}$ represents the set-valued transition map (symbolic counterpart of the dynamics (1)). Then, a controller for a motion primitive of (1) can be synthesized by solving the corresponding problem in the symbolic domain, where $\mathcal{Q}_0$, $\mathcal{Q}_f$ and $\hat{\mathcal{Q}}$ denote the symbolic counterparts of the starting set S_0, the target set T_f and the safe corridor $\hat{S}$.

This can be done in a similar way as described in section III-A using the symbolic one-step constrained predecessor of a set $\mathcal{T} \subseteq \hat{\mathcal{Q}}$:

$$Pre_{\mathcal{F}}(\mathcal{T}, \hat{\mathcal{Q}}) = \left\{ q \in nbs_{\mathcal{F}} \cap \hat{\mathcal{Q}} \,\middle|\, \begin{matrix} \exists u \in enab_{\mathcal{F}}(q) \\ \mathcal{F}(q, u) \subseteq \mathcal{T} \end{matrix} \right\} \quad (7)$$

where $enab_{\mathcal{F}}(q) = \{u \in \mathcal{U} | \mathcal{F}(q, u) \neq \emptyset\}$ and $nbs_{\mathcal{F}}(q, Q) = \{q \in Q | enab_{\mathcal{F}}(q) \neq \emptyset\}$. Similarly we can compute the i-step reachable sets of $\mathcal{Q}_f$ and check the inclusion of $\mathcal{Q}_0$, similar to Proposition 2.

Let us emphasize that since the transition system has finite sets of states and inputs, the implementation of the approach is straightforward. Moreover, the synthesized controller comes with formal guarantees when applied to (1) [13].

1) Specific construction methods: We begin by selecting a discretized region along with the initial set and target set within it. By progressively increasing the value of p until the condition in (6) is verified, we can ascertain the feasibility of the chosen motion primitive. Meanwhile the discretized region can be validated as a safe corridor. More specifically, we first choose a discrete region with a large area to ensure that p exists and is finite. Then we gradually reduce the size of discretized region until we reach a critical point where p no longer exists. This process enables us to refine the safe corridor and provide improved precision along the execution of the motion primitive.

It is worth mentioning that for practical reasons, we use an

interval over-approximation $\hat{\mathcal{F}}$ of $\mathcal{F}$, which guarantees that $\mathcal{F}(q,u) \subseteq \hat{\mathcal{F}}(q,u)$. Such interval over-approximations can be computed using interval reachability analysis (see [10]). The design of controllers can be done for a set of initial states due to the set-based design process. This fact, along with the use of over-approximation in the actual construction process, provides flexibility/robustness in the overall motion control process. This aspect will not be further elaborated on or quantified but is worth to be considered an additional feature of the present approach.

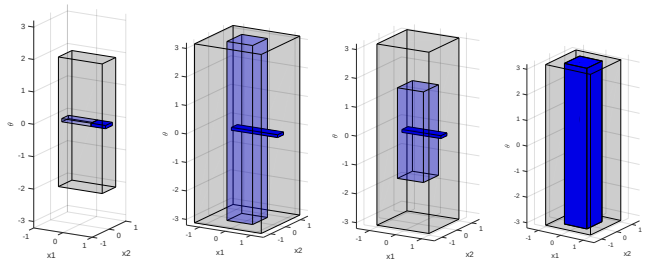
2) Parameters related to symbolic control: When employing symbolic control, different parameter configurations can significantly impact the effectiveness of the controller. Smaller grid sizes in the state space and finer control input intervals resulting in smoother trajectories. However, such improvements come at the cost of increased computational complexity. Also there exist multiple feasible control symbols underlying the definition of the sets in (7). While the reachable sets are the essential features for the definition of the primitives, a control selection criteria should be customized in order to allow control implementation and the execution of the primitive in itself. For example, when designing translation primitives, our goal may be to select a control input that maximize speed while minimizing angular change. When designing rotational primitives, we may prioritize the feasible controller that minimizes velocity.

C. Design notes

By adjusting parameters of the motion primitives, we can generate various motion patterns, forming what we refer to as a library of motion primitives. We can customize certain motion primitives as required, thereby enhancing the flexibility of motion control.

Here, we introduce some particular parametric designs to enrich the library of motion primitives.

- The *length of a translation primitive* in the translational direction is related to the speed of movement, which is usually set according to the maximum length that can be moved in one time step. In Fig.2a a deceleration primitive is depicted where the target set of the translation primitive is shortened, it mandates the controlled object to move at a slower speed. This can be applied to many situations. For example, the deceleration before rotation can result in a smaller steering radius, which means we can choose a smaller safe corridor.
- The *size of the target set* is typically associated with the smoothness of the trajectory. When we need smoother rotation paths, the rotation primitives can be designed as shown in Fig. 2b. The length of the target region extends along the desired direction.
- We can also design rotational primitives tailored to specific *ranges of rotation angles*. For example, as depicted in Fig. 2c, it rotates the states within the range of $[-\pi/2, \pi/2]$. Although this design of rotation primitive is no longer universal, a narrowed safe corridor can be found.
- We can also design a *stop primitive*, as shown in Fig. 2d, where the starting set equals the target set, implying that the trajectory is permanently confined to a certain area. In this case the set inside the safe corridor can be identified as a p-invariant set [15]



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

Fig. 2: Customized motion primitives

IV. SUPERVISED TRACKING CONTROL

This section introduces the method for online tracking of waypoints using offline designed motion primitives and corresponding controllers.

A. Families of motion primitives

We employ a family of motion primitives to represent all the transformed motion primitives obtained by applying certain operation to the elementary ones. This transformation is represented formally in terms of a bijective mapping, denoted by $p_\pi : \mathbb{R}^n \to \mathbb{R}^n$, parameterised by the vector $\pi = (v_1, v_2, \alpha) \in \mathbb{R}^3$, and defined as follows:

$$p_\pi(z) = M(\alpha)z + V(v_1, v_2, \alpha)$$

with its inverse $p_\pi^{-1}(z) = M(\alpha)^{-1}(z - V(v_1, v_2, \alpha))$. Using this transformation, for any given set $S \subseteq \mathbb{R}^n$,

$$p_\pi(S) = \{p_\pi(z) | z \in S\}.$$

Further, this can be extended to derive the family of motion primitives as follows:

$$p_\pi(\mathcal{P}) = \langle p_\pi(S_0), p_\pi(T_f), p_\pi(\hat{S}) \rangle$$

Proposition 3: If $\mathcal{P}$ is feasible, then for all $\pi = (v_1, v_2, \alpha) \in \mathbb{R}^3$, $p_\pi(\mathcal{P})$ is also feasible.

Proof: Given the feasibility of a motion primitive, there exists a controller $C_\mathcal{P}$ such that for any state $z_0 \in S_0$, $z(n) \in T_f$, and $z(k) \in \hat{S}$ for all $k \in [0, n]$. At time 0, consider $z_0' = p_\pi(z_0) \in p_\pi(S_0)$ and we apply the following controller:

$$C_{p_\pi(\mathcal{P})}(z'(k)) = C_\mathcal{P}(p_\pi^{-1}(z'(k)))$$

Let us assume that at time $k \in [1, n-1]$, we have:

$$z'(k) = p_\pi(z(k))$$

Then by applying the following control inputs we can determine $z'(k+1)$.

$$u'(k) = C_{p_\pi(\mathcal{P})}(z'(k)) = u(k)$$

From *Proposition* 1, we have:

$$z'(k+1) = M(\alpha)(z(k+1) - z_0) + p_\pi(z_0)$$
$$z'(k+1) = p_\pi(z(k+1))$$

Then we can conclude that $z'(k) = p_\pi(z(k))$ for all $k \in [0, n]$. Thus $p_\pi(\mathcal{P})$ is also feasible. ■

This basic result shows that a reduced set of elementary primitives and the transformations enabled by the rotational and translational invariance of the dynamics provide a rich family of motion primitives which can be used next for effective real-time motion control.

B. Concatenation

Each motion primitive represents a collection of trajectories, and our objective is to reach the target point by concatenating these motion primitives, which must follow certain criteria. If $\mathcal{P}^k$ and $\mathcal{P}^{k+1}$ are any two motion primitives provided in sequential order and need to be concatenated, then following rules need to be satisfied :

$$T_f^k \subseteq S_0^{k+1} \tag{8}$$

The implementation of principle (8) should be manifested in the design of motion primitives. We consider two main types of concatenation in planar movement.

- *Translation-Rotation-Translation*: A translation primitive is concatenated with a rotation primitive and then another translation primitive. The rotational primitives must be designed to accommodate the mismatch between two translation motions with different heading angles. Technically, the starting set for the rotation primitive includes the circumcircle of the rectangle from the preceding translation primitive, as shown in Fig.3.
- *Translation-Translation*: Continuously concatenate translation primitives in one direction until reaching a region adjacent to the target waypoint as illustrated in Fig.4. When the distance between the center of the target set and the waypoint is less than $length/2$ of the translation primitive, the target waypoint is reached and the primitive is considered to be executed.

Remark 2: Given the degrees of freedom in designing various motion primitives, one can adjust the size of the target set. For instance, by adding a deceleration primitive, as depicted in Fig. 2a, at the endpoint, we can obtain a comparatively smaller target set. Since this is adjustable, in this paper, we simplify the discussion by considering the target set as depicted in Fig.4 to always be selected as a subset of the given waypoint region $\mathbb{X}_i$.

C. Primitives scheduler and real time tracking

To address the problem described in Problem 1, we divide the tracking of waypoints into two parts. The real time tracking part and the primitives scheduler module. As shown in Fig.5. The real time tracking part is responsible for updating the state using (1) and monitoring whether the current state belongs to the target set of the current motion primitive denoted as $p_\pi(T_f^k)$. The *Primitives scheduler*

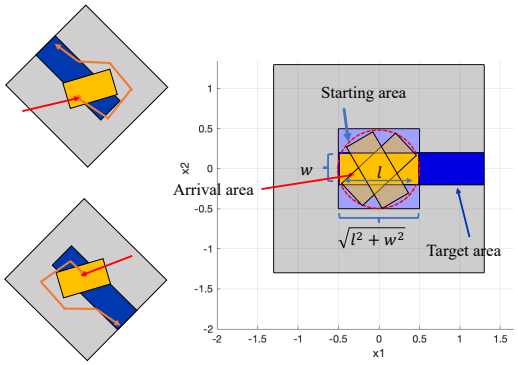


Fig. 3: Concatenation between rotation and translation primitives. Safe corridors(in gray), starting area (light blue), target region(dark blue), the potential range of target set (yellow region and red circle)

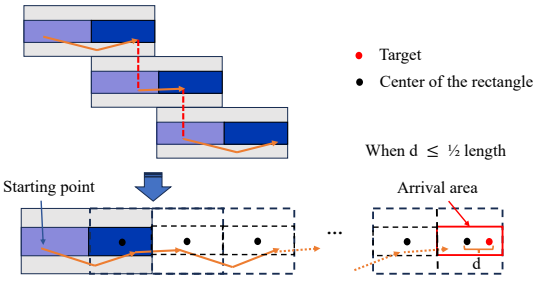


Fig. 4: Concatenation between translation primitives (in blue). Safe corridors (in gray), the starting set (light blue), the target set(dark blue), trajectory (in orange), target waypoint/region (in red)

module uses as inputs the target waypoint $\mathbb{X}_i$ and the result of the event monitoring module. This event-based module is activated if $z(t) \in p_\pi(T_f^k)$ and the delivered output is the information concerning the next motion primitive (including its transformation parameters p_π and p_π^{-1}), the associated controller $C_{\mathcal{P}^{k+1}}$, the next safe corridor $p_\pi(\hat{S}^{k+1})$ and the next target set $p_\pi(T_f^{k+1})$, with $k \in [0, n-1]$ where n represents the total number of required motion primitives to be executed up to the next waypoint.

With all these elements we can resume the proposed control method properties by the next result.

Proposition 4: The *Problem* 1 is feasible with proposed control strategy if the union of the safe corridor obtained for the whole path is a subset of $\mathbb{Z}$.

Proof: For any initial state $z_0 = (x_1(0), x_2(0), \theta(0), \omega(0)) \in \mathbb{X}_0 \times \mathbb{R} \times \Omega$, and the next target waypoint $(\bar{x}_1, \bar{x}_2)$ is characterized by region $\mathbb{X}_1$.

The desired direction is calculated as $\alpha = \arctan\left(\frac{\bar{x}_2 - x_2}{\bar{x}_1 - x_1}\right)$. If $|\theta(0) - \alpha| \leq \delta$, then we should employ a sequence of translation primitive $(\mathcal{P}_t^0, \ldots, \mathcal{P}_t^n)$. If $|\theta(0) - \alpha| > \delta$, the next primitive should be a rotation primitive then concatenate with a sequence of translation primitive : $(\mathcal{P}_r^0, \mathcal{P}_t^1, \ldots, \mathcal{P}_t^n)$ where n determined in the manner described in the previous section.

For each feasible motion primitive, states can reach the target set in at most p time steps. Therefore there exists

$$t_i \leq p_1 + p_2 + \cdots + p_n$$

such that

$$z(t_i) \in p_\pi(T_f^n) \subseteq \mathbb{X}_1 \times \mathbb{R} \times \Omega$$

and all the states in this trajectory

$$z(t) \in p_\pi(\hat{S^0}) \cup p_\pi(\hat{S^1}) \cup \cdots \cup p_\pi(\hat{S^n}).$$

It is clear that the above argument still holds when the different waypoints region $\mathbb{X}_i$ are given in order. If the obtained union of safe corridor is a subset of $\mathbb{Z}$, then problem 1 is feasible.

■

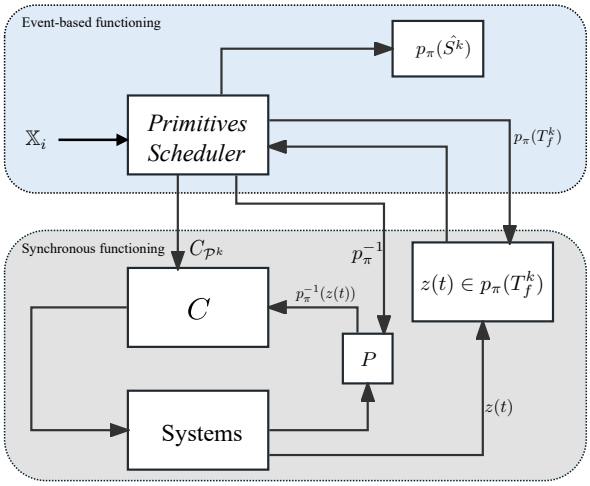


Fig. 5: Real time tracking of waypoints

V. Numerical motion control illustration

Let us revisit the dynamical system (2) within a numerical example of complex motion control subject to additional control input constraints:

$$\mathbb{U} = [0.25, 1] \times [-1, 1]. \tag{9}$$

The size of the grid is selected as $dx = [0.1, 0.1, 0.1]^T$, and we use 100 symbolic inputs. We aim for the controlled object to move at maximum speed whenever possible. Therefore, when designing the controller using symbolic control, we prioritize selecting discrete controls where $u_1 = 1$. Correspondingly, the start and target regions of the translation primitives are of length 1 in the x_1 direction.

Fig.6a and Fig.6c illustrate the trajectories corresponding to two different controllers along with their respective safety corridors. Fig.6b and Fig.6d depict the evolution of the angle, the distance to the waypoints (5 in total). Is worth to notice the piecewise monotonic behaviour of this signal, the jumps corresponding to a switch of waypoint. As show in the same figure, u_1 is selected as 1 in most cases, except when the state is approaching the frontier of the safety corridor ($t = 10s$ and $17s$ in Fig.6d). In these particular situations, the speed is constrained to a decrease to ensure safety.

For 6a we chose a relatively wide starting and target set, which also corresponds to a wider safe corridor, the resulting trajectories are smoother. We use 56700 symbolic states to construct the rotation primitive and 20160 for the translation primitive. The computation time is about 2 minutes in total with a PC 1.4 GHz Intel Core i5. For Fig. 6c, we chose a narrower range of start and target sets, resulting in a narrower safety corridor, We use 42588 symbolic states to construct the rotation primitive and 10080 for the translation primitive. The computation time is about 1 minutes.

The results show that these off-line designed controllers can cope with arbitrary motions in the two-dimensional plane, while we can flexibly adjust the parameters according to the control objectives.

VI. Conclusion

In this paper, we propose a method for synthesizing motion primitives using symbolic control techniques and for creating more intricate motions by stitching together these primitives. Our approach is applicable to a range of planar motion scenarios, including systems with translational and rotational invariance such as robots or autonomous vehicles. By employing this method, we significantly reduce the computational burden associated with symbolic control while still preserving some of its robustness and safety benefits.

In future work, we plan to extend the application of this method to more complex models and explore potential combination with other online control techniques to enhance control performance.

References

[1] Rubens JM Afonso, Marcos ROA Maximo, and Roberto KH Galvao. Task allocation and trajectory planning for multiple agents in the presence of obstacle and connectivity constraints with mixed-integer linear programming. *International Journal of Robust and Nonlinear Control*, 30(14):5464–5491, 2020.

[2] Calin Belta, Antonio Bicci, Magnus Egerstedt, Emilio Frazzoli, Eric Klavins, and George J Pappas. Symbolic control and planning of robotic motion [grand challenges of robotics]. *Departmental Papers (ESE)*, page 232, 2007.

[3] Omer Cetin, Sefer Kurnaz, Okyay Kaynak, and Hakan Temeltas. Potential field-based navigation task for autonomous flight control of unmanned aerial vehicles. *International Journal of Automation and Control*, 5(1):1–21, 2011.

[4] Emilio Frazzoli and Francesco Bullo. On quantization and optimal control of dynamical systems with symmetries. *Proceedings of the 41st IEEE Conference on Decision and Control, 2002.*, 1:817–823 vol.1, 2002.

[5] Emilio Frazzoli, Munther A Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE transactions on robotics*, 21(6):1077–1091, 2005.

[6] Shuzhi Sam Ge and Yun J Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous robots*, 13:207–222, 2002.

[7] Antoine Girard and Alina Eqtami. Least-violating symbolic controller synthesis for safety, reachability and attractivity specifications. *Automatica*, 127:109543, 2021.

[8] Thomas M Howard, Colin J Green, and Alonzo Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In *Field and Service Robotics: Results of the 7th International Conference*, pages 69–78. Springer, 2010.

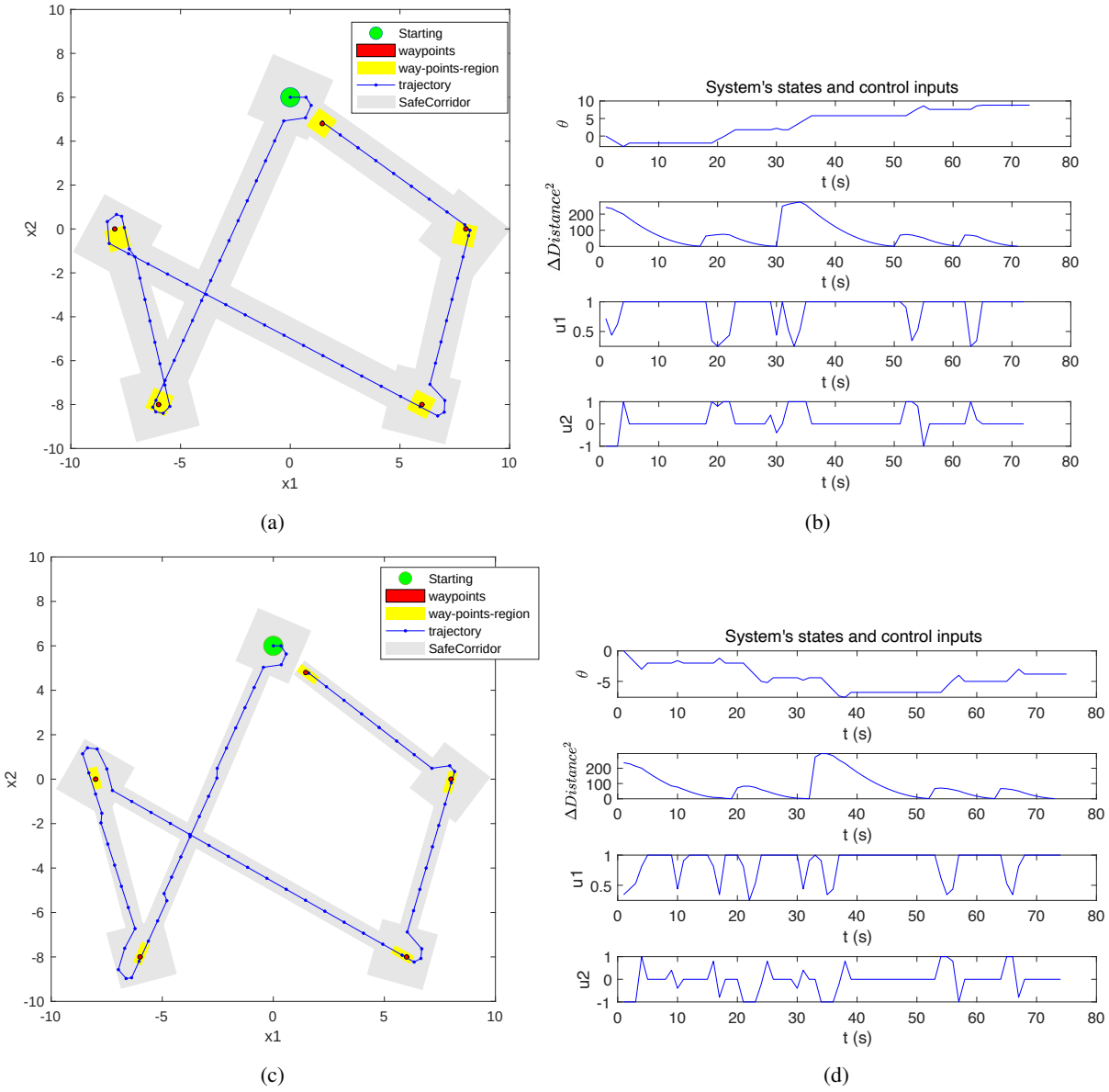


Fig. 6: The left figures illustrate the trajectory of the system alongside the corresponding safety corridor (depicted in gray), whereas the right figures display the system's state alongside the control inputs.

[9] Daniel Ioan, Ionela Prodan, Sorin Olaru, Florin Stoican, and Silviu-Iulian Niculescu. Mixed-integer programming in motion planning. *Annual Reviews in Control*, 51:65–87, 2021.

[10] Pierre-Jean Meyer, Alex Devonport, and Murat Arcak. *Interval reachability analysis: Bounding trajectories of uncertain systems with boxes for control and verification*. Springer Nature, 2021.

[11] Aditya A Paranjape, Kevin C Meier, Xichen Shi, Soon-Jo Chung, and Seth Hutchinson. Motion primitives and 3d path planning for fast flight through a forest. *The International Journal of Robotics Research*, 34(3):357–377, 2015.

[12] Ionela Prodan, Sorin Olaru, Ricardo Bencatel, Joao Borges de Sousa, Cristina Stoica, and Silviu-Iulian Niculescu. Receding horizon flight control for trajectory tracking of autonomous aerial vehicles. *Control Engineering Practice*, 21(10):1334–1349, 2013.

[13] Gunther Reissig, Alexander Weber, and Matthias Rungger. Feedback refinement relations for the synthesis of symbolic controllers. *IEEE Transactions on Automatic Control*, 62(4):1781–1796, 2016.

[14] Paulo Tabuada. *Verification and control of hybrid systems: a symbolic approach*. Springer Science & Business Media, 2009.

[15] Zhao Zhixin, Antoine Girard, and Sorin Olaru. Nonlinear model predictive control based on k-step control invariant sets. In *European control conference - Stockholm*, 2024.