2012 IEEE International Conference on Robotics and Automation
RiverCentre, Saint Paul, Minnesota, USA
May 14-18, 2012

Sequential Composition of Robust Controller Specifications

Jerome Le Ny and George J. Pappas

Abstract— **We present a general notion of robust controller specification and a mechanism for sequentially composing them. These specifications form tubular abstractions of the trajectories of a system in different control modes, and are motivated by the techniques available for certifying the performance of low-level controllers. The notion of controller specification provides a rigorous *interface* for connecting a planner and lower-level controllers that are designed independently. With this approach, the planning layer does not integrate the closed-loop system dynamics and does not require the knowledge of how the controllers operate, but relies only on the specifications of the output tracking performance achieved by these controllers. The control layer aims at satisfying specifications that account quantitatively for robustness to unmodeled dynamics and various sources of disturbance and sensor noise, so that this robustness does not need to be revalidated at the planning level. As an illustrative example, we describe a randomized planner that composes different controller specifications from a given database to guarantee that any corresponding sequence of control modes steers a robot to a given region while avoiding obstacles.**

I. Introduction

Formalisms available for the analysis of cyber-physical systems such as robotic systems, based for example on hybrid automata mixing discrete events with differential equations, are arguably unwieldy to use for system design purposes. Descriptions based on such general formalisms lead to ubiquitous state-space explosion problems and often accidentally include undesired behaviors complicating the analysis, such as Zeno phenomena. To address these complexity issues, increasing emphasis is being placed on compositional design frameworks that allow one to build such systems from components and derive system properties from the separate analysis of the individual components [1].

In robot motion planning, several compositional frameworks have been proposed, including motion description languages [2], [3] and the maneuver automaton [4], as well as the sequential composition of funnels [5]–[7] based on preimage backchaining [8]. In these examples, a set of controllers is available to execute specific atomic behaviors, and one wishes to build more complex behaviors from these atoms. A drawback of most of these frameworks however is that they do not establish a clear or rigorous separation between the planning module and lower-level controllers. Hence they either produce computationally complex planners, which attempt at verifying low-level properties directly at the planning level without relying on dedicated control-theoretic tools such as Lyapunov analysis, or they fail to provide rigorous guarantees for the overall system, e.g., by assuming that the tracking errors of the controllers can be neglected by the planner. Moreover, the planner must generally directly integrate the complete system dynamics, in typically high-dimensional state spaces, which complicates its task significantly.

The approach of Burridge et al. [5] to sequential composition of behaviors was an important step in bridging planning and Lyapunov analysis while enforcing a certain separation between the design of the different layers. They limited their study to the class of regulating controllers however, and did not address quantitatively certain issues, such as robustness to noise, disturbance, and unmodeled dynamics. Note that composing only regulators can be difficult because the Lyapunov analysis needs to find large basins of attraction for the successive target points in the state space, see e.g. [7] for recent work in that direction.

This paper proposes a rigorous *interface* for connecting and designing relatively independently the planning and control layers. We do not advocate here the use of particular planning algorithms or controller design techniques, but instead we are interested in compact summaries of the closed loop dynamics that can be used by a planner in place of detailed differential equations. Section II introduces such an abstract notion of *controller specification*, together with a mechanism for sequentially composing them. Essentially, a specification defines a tube around a reference trajectory describing how accurately a control mode tracks this reference. Disturbances prevent perfect tracking, but the role of a controller is to quickly bring the tracking errors sufficiently close to zero, and numerous design techniques have been devised to achieve this objective with theoretical guarantees [9]. The funnels of Burridge et al. [5] correspond to a situation with no external disturbance

The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104, USA jeromel, pappasg@seas.upenn.edu.

978-1-4673-1405-3/12/$31.00 ©2012 IEEE

and all reference trajectories consisting of fixed points in the state space. We also specialize our definitions to the important case of input to output stability specifications, for which the system in each control mode is required to be input to output stable [10] with respect to the disturbance inputs and the tracking error outputs. This notion provides a suitable concrete way of parametrizing the tubes, using standard Lyapunov analysis techniques.

With a notion of control mode specification and of sequential composition of such specifications, we can then consider the general problem of searching for a sequence of specifications achieving a desired behavior. To illustrate these ideas, Section III focuses on steering a robot from one region to another while avoiding obstacles, and describes how to combine the specifications with a Rapidly-exploring Random Tree (RRT) planner [11]. The approach is then applied in Section IV to the motion planning problem for a curvature constrained vehicle equipped with a family a simple line tracking controllers satisfying formal performance specifications.

II. Robust Controller Specifications

A. Notation

For $s \in \mathbb{R}$ and a map $t \in \mathbb{R} \mapsto F(t)$, we define the time shifted map F^s by $F^s(t) := F(t+s)$. $\mathcal{P}(X)$ denotes the set of subsets of a set X. We denote the Euclidean norm of a vector x by $|x|$. $\mathcal{B}_\rho(x_0)$ denotes the Euclidean ball with center x_0 and radius $\rho \geq 0$. Finally, we denote the sup norm of a signal $t \to w(t)$ over the interval $[0,t)$ by $\|w\|_t := \sup_{s\in[0,t)} |w(t)|$, including $\|w\|_\infty$ when $t = \infty$ in the definition.

B. Controller Specifications

Consider a system with open-loop dynamics described in local coordinates by

$$\dot{x} = f(x,u,w), \tag{1}$$

$$y = Hx, \tag{2}$$

where $x(t) \in \mathsf{X} \cong \mathbb{R}^n$ is the state, u a control input, and w a disturbance input that can also account for modeling errors. We wish to steer the output $y \in \mathsf{Y} \cong \mathbb{R}^p$ (a linear transformation of the state) in the obstacle free subset $\mathsf{Y}_{\text{free}} \subset \mathsf{Y}$. For example, y can be a subset of the coordinates of x that are subject to collision avoidance constraints. We assume that H is full row rank.

We use a set of predesigned tracking controllers to steer the robot. A *control mode* m consists of a reference trajectory $t \mapsto r_m(t) \in \mathsf{X}$, together with a controller K_m specifying the input $u(\cdot)$ as long as the mode is engaged. The controller is designed so that the state x tracks r_m and hence the output y tracks the signal Hr_m. In some cases, only an output reference trajectory $y_{ref,m}$ could be specified and then we can typically define $r_m = H^\dagger y_{ref,m}$, where $H^\dagger$ is the Moore-Penrose pseudo-inverse of H, see [12, chap. 4]. In addition, a particular control setup can introduce additional disturbances denoted ν_m, e.g., measurement noise. Hence a control mode is also associated to a particular disturbance signal $w_m = [w^T, \nu_m^T]$, possibly of larger dimension than the open-loop disturbance w.

For planning purposes, we only rely on certain performance specifications for a controller and not on the knowledge of the actual controller design or implementation. More precisely, we need the following data for a given control mode.

Definition 1: A *controller specification* for a control mode m engaged on the interval $[t_m, t'_m)$ is a tuple $\sigma_m = (r_m, \mathsf{C}_m, \mathcal{E}_m, \mathcal{Z}_m, \mathsf{D}_m)$ where

1) $r_m : \mathbb{R}_+ \to \mathsf{X}$ is called the *reference trajectory*. We then define the *tracking error* $e_m : \mathbb{R}_+ \to \mathsf{X}$ by $e_m(t) = x^{t_m}(t) - r_m(t)$, $0 \leq t \leq t'_m - t_m$.
2) $\mathsf{C}_m \subset \mathsf{X}$ is called the *enabling condition* and one must have $e_m(0) \in \mathsf{C}_m$.
3) $\mathcal{E}_m : \mathbb{R}_+ \to \mathcal{P}(\mathsf{X})$ is a set-valued function of time, satisfying $\mathcal{E}_m(0) \subset \mathsf{C}_m$ and $e_m(t) \in \mathcal{E}_m(t)$ for all $0 \leq t \leq t'_m - t_m$.
4) $\mathcal{Z}_m : \mathbb{R}_+ \to \mathcal{P}(\mathsf{Y})$ is a set-valued function of time, satisfying $He_m(t) \in \mathcal{Z}_m(t)$ for all $0 \leq t \leq t'_m - t_m$.
5) D_m is a set of of admissible disturbance signals for the mode.

A controller specification represents the tracking performance of a controller for a specific reference trajectory. The enabling condition must be met by the system state in order to engage the controller. The sets $\mathcal{E}_m(t)$ and $\mathcal{Z}_m(t)$ represent the tracking errors in the state and output space respectively. Note that we can always choose $\mathcal{Z}_m(t) = H\mathcal{E}_m(t)$ in Definition 1-4). The reason for a separate definition is that one can sometime get a better estimate for the output tracking error than for the state tracking error. The next subsection provides more concrete representations of these tracking error sets. Next, we introduce a mechanism for sequentially composing motion specifications and recording the propagation of the tracking error during switching.

Definition 2: Two controller specifications $\sigma_{m_1}, \sigma_{m_2}$ for two modes m_1 and m_2 can be *sequentially composed* after a time d_1 in mode m_1, denoted $\sigma_{m_1} \rhd_{d_1} \sigma_{m_2}$, if they satisfy $r_{m_1}(d_1) + \mathcal{E}_{m_1}(d_1) \subset r_{m_2}(0) + \mathsf{C}_{m_2}$. In this case, we define the *tracking error transition map* $\tau^{d_1}_{\sigma_{m_1},\sigma_{m_2}} : \mathcal{P}(\mathsf{X}) \times \mathcal{P}(\mathsf{Y}) \to \mathcal{P}(\mathsf{X}) \times \mathcal{P}(\mathsf{Y})$ by

$$\tau^{d_1}_{\sigma_{m_1},\sigma_{m_2}}(\mathcal{E}_{m_1}(d_1), \mathcal{Z}_{m_1}(d_1)) = (\mathcal{E}_{m_2}(0), \mathcal{Z}_{m_2}(0)).$$

Hence two controller specifications can be sequentially composed after duration d_1 in mode m_1 if we know with certainty that mode m_1 steered the state close enough to $r_2(0)$ in order to engage mode m_2. In this case, the transition map transforms the set of possible tracking errors for mode m_1 at the time of switching to an initial set of possible tracking errors for mode m_2. Finally, the planning problem that we address more specifically in this paper is the following.

Problem 1: Given a set $S \subset \mathsf{X}$ of possible initial states $x(0)$ for the system, and a set $G \subset \mathsf{Y}_{\text{free}}$ of desired outputs to reach, given a family $\mathcal{F}$ of controller specifications, does there exist $K \in \mathbb{N}$ and sequences $\sigma_{m_0}, \sigma_{m_1}, \dots \sigma_{m_K} \in \mathcal{F}$ and $d_0, \dots, d_K \geq 0$ such that

1) $S \subset \mathsf{C}_{m_0}$,
2) $\sigma_{m_i} \rhd_{d_i} \sigma_{m_{i+1}}$, $0 \leq i \leq K-1$,
3) $Hr_{m_i}(t) + \mathcal{Z}_{m_i}(t) \in \mathsf{Y}_{\text{free}}$, for all $0 \leq i \leq K, 0 \leq t \leq d_i$ (i.e., collisions are avoided),
4) $Hr_{m_K}(d_K) + \mathcal{Z}_{m_K}(d_K) \subset G$.

That is, we are looking at the planning level for a sequence of specifications that is guaranteed to lead from a starting region to a goal region while keeping the output in the obstacle free space Y_{free}.

C. Input to Output Stability Specifications

The previous subsection defines specifications in abstract set-theoretic terms. In practice, we need to instantiate Definition 1 to versions that are computationally more convenient and motivated by the tools available for the analysis and design of controllers.

If some control mode m with a controller K_m tracking a known continuously differentiable trajectory $t \to r_m(t) \in \mathsf{X}$ is engaged from time t_m onward, the closed-loop system follows the dynamics, for $t \geq 0$,

$$\dot{\xi}_m = f_m(t, \xi_m, w_m), \;\; e_m(0) \in \mathsf{C}_m, \tag{3}$$
$$z_m := y - Hr_m = He_m, \;\; w_m \in \mathsf{D}_m \tag{4}$$

where e_m denotes the tracking error as in Definition 1 and the state of the closed-loop system $\xi_m = [e_m^T, \zeta_m^T]^T$ includes the state ζ_m of the controller K_m. The disturbance signal is $w_m = [w^T, \nu_m^T]$ as explained previously.

Example 2.1: Consider a dynamic controller with error feedback,

$$\dot{\zeta}_m = \eta(\zeta_m, h(e_m) + \nu_m)$$
$$u = \theta(\zeta_m),$$

where ν_m is a noise contaminating a tracking error measurement $h(e_m)$. Then we have for the closed-loop system

$$\dot{e}_m = f(e_m + r_m, \theta(\zeta_m), w) - \dot{r}_m$$
$$\dot{\zeta}_m = \eta(\zeta_m, h(e_m) + \nu_m)$$
$$z_m := He_m,$$

which is of the form (3), (4), since since $r_m, \dot{r}_m$ are known functions of time.

A convenient requirement for the control modes is that the closed-loop dynamics (3) in each mode be input to state stable (ISS) or input to output stable (IOS) [10], [13]. We use the following terminology. A function $\gamma : [0, \infty) \to [0, \infty)$ is of class $\mathcal{G}$ if it is continuous, non-decreasing and satisfies $\gamma(0) = 0$. It is of class $\mathcal{K}$ if it is of class $\mathcal{G}$ and strictly increasing. $\mathcal{GL}$ (resp. $\mathcal{KL}$) is the class of functions $[0, \infty)^2 \to [0, \infty)$ that are of class $\mathcal{G}$ (resp. $\mathcal{K}$) on their first argument and decrease to zero on their second argument.

For a given mode m with dynamics (3), let us assume available a set of J_m inequalities of the form

$$|g^i(e_m(t))| \leq \max\{\beta_m^i(|\xi_m(0)|, t), \gamma_m^i(\|w_m\|_t)\}, \tag{5}$$

for all $t \geq 0$, $e_m(0) \in \mathsf{C}_m$, $w_m \in \mathsf{D}_m^\alpha = \{w | \|w\|_\infty \leq \alpha\}$, and $i = 1, \dots, J_m$, where β_m^i are $\mathcal{GL}$ functions and γ_m^i are of class $\mathcal{G}$. In other words, the closed-loop system in mode m is locally input to output stable [10] with respect to the input disturbance w_m and the outputs $g^i(e_m)$, $i = 1, \dots, J_m$, with the minor variation that we use functions of class $\mathcal{G}$ and $\mathcal{GL}$ instead of $\mathcal{K}$ and $\mathcal{KL}$ in the definition. These inequalities can be used to give a representation of the set-valued maps $\mathcal{E}_m$, $\mathcal{Z}_m$ of Definition 1, as explained in the rest of this section.

The ISS and IOS notions [10], [13] are typically used to study asymptotic stability properties of nonlinear systems. The function β_m characterizes the transient regime of the mode, and the quantity $\gamma_m(\|w_m\|_\infty)$ the steady-state tracking error. Here however, we use the functions β_m, γ_m to abstract the dynamics of the robot over finite time intervals. Of particular interest for computations are functions β_m of the exponentially decreasing form

$$\beta_m(\xi, t) = k_m(\xi) e^{-\lambda_m t},$$

where k_m is a function of class $\mathcal{G}$, for example $k_m(\xi) = k_{0,m} |\xi|$. Assuming that the functions k_m and γ_m admit finite dimensional parametrizations, they can be stored in memory together with the decay rate λ_m. This provides a finite-dimensional abstraction of the closed-loop dynamics of mode m, including the effect of perturbations. The main advantage of using characterizations of the form (5) for computations is that Lyapunov analysis techniques are available to derive such bounds [9].

Example 2.2: Suppose that we only have the following two such inequalities for each motion specification. The first bounds the state tracking error

$$|e_m(t)| \leq \max\{\beta_m^1(|\xi_m(0)|, t), \gamma_m^1(\|w_m\|_t)\}. \quad (6)$$

The second bounds the output tracking error

$$|z_m(t)| \leq \max\{\beta_m^2(|\xi_m(0)|, t), \gamma_m^2(\|w_m\|_t)\}, \quad (7)$$

assuming it is less conservative than using $|He_m| \leq \|H\||e_m|$ in (6). In this case, $r_m(t) + \mathcal{E}_m(t)$ and $Hr_m(t) + \mathcal{Z}_m(t)$ are Euclidean balls around the references $r_m(t)$ and $Hr_m(t)$ respectively.

Consider now the transition between two motion specifications σ_{m_1} and σ_{m_2}, where σ_{m_1} is followed for a duration d_1. From the first inequality (6), we get a bound of the form $|x(t_1 + d_1) - r_{m_1}(d_1)| \leq \rho_1$, where t_1 is the time at which mode m_1 is engaged. If $\mathcal{B}_{\rho_1}(r_{m_1}(d_1)) \subset r_{m_2}(0) + \mathsf{C}_{m_2}$, we can sequentially compose the modes. Let $t_2 = t_1 + d_1$. Since $t \mapsto x(t)$ is continuous, we have

$$\begin{aligned} |e_{m_2}(0)| &= |x(t_1 + d_1) - r_{m_2}(0)| \\ &\leq \rho_1 + |r_{m_1}(0) - r_{m_2}(0)| := \rho_2. \end{aligned}$$

Hence we can define $\mathcal{E}_2(0)$ as $\mathcal{B}_{\rho_2}(r_{m_2}(0))$. Moreover, we have $|\xi_{m_2}(0)| \leq \sqrt{\rho_2 + |\zeta_{m_2}(0)|^2}$, where $\zeta_{m_2}(0)$ depends on the initialization of the controller for mode m_2. This last bound is used to replace the unknown quantity $|\xi_{m_2}(0)|$ in the inequalities (6), (7) for σ_{m_2}, and to obtain representations of the error tracking maps $\mathcal{E}_{m_2}(t), \mathcal{Z}_{m_2}(t)$.

III. Searching for a Sequential Composition

In this section, we describe a randomized algorithm to solve Problem 1, based on the RRT planner of [11], and exploiting the notion of sequential composition of specifications presented in Section II-B. The pseudocode for the algorithm is given as Algorithms 1-3. The RRT algorithm builds a graph that is eventually used to steer the system to the goal region by sequentially composing modes. A node n in the graph records

1) the specification σ_{m_n} of the mode m_n used to reach the node.
2) the duration d_n since the mode m_n was last engaged.
3) The index $\texttt{pred}(n)$ of the predecessor node in the tree.

Intuitively, a node n is associated to the point $r_{m_n}(d_n) \in \mathsf{X}$. We initialize the tree with a node with index 0, recording $\hat{x}_0$, an estimate of the initial state, and $\hat{\mathcal{E}}_0$ so that $S \subset \hat{x}_0 + \hat{\mathcal{E}}_0$, with S as in Problem 1.

Given a partially constructed tree, we create a new node by first generating a new sample point s in the free output space $\mathsf{Y}_{\texttt{free}}$. Then, we find a node in the RRT, say node n, close to this sample point according to some heuristic notion of distance $\mu : \mathcal{N} \times \mathsf{Y} \to \mathbb{R}_+$, where $\mathcal{N}$ is the set of nodes in the RRT. Generally μ involves the distance from $Hr_{m_n}(d_n)$ to s and a measure of the size of the output or state tracking error at the node. For example, we could take $\mu(n, s) = |Hr_{m_n}(d_n) - s| + \alpha \operatorname{diam}(\mathcal{Z}_{m_n}(d_n))$, for some constant $\alpha \geq 0$.

Once the sample s is produced and a node n in the tree is selected "close" to s, we create one or more new nodes from node n. The function generating new nodes is described in Algorithm 3. First, we consider a number of possible specifications that can be sequentially composed with σ_{m_n} after d_n, with reference trajectories starting from or close to $r_{m_n}(d_n)$, and preferably steering the output toward s. We also select deterministically or at random a duration T for which we consider following each of these modes.

Among the set of specifications considered to extend the tree, we include the possibility of continuing to follow mode m_n for an additional time T, and also select some other compatible specifications to potentially switch to. For example, we could switch to tracking a straight line towards s, if such a mode exists. Collision avoidance is checked at this stage, to verify that the tube around a reference trajectory does not intersect any obstacle. In case of mode switching, we use the transition map τ to evaluate the change in the tracking error sets. We select among the specifications the one that steers the vehicle closest to s, based on the "distance" function μ, and we add a corresponding node to the tree with predecessor n. It is also possible to add several nodes at this step if several motions give satisfactory performance, with the usual tradeoff due to longer nearest-neighbor search as the tree gets bigger.

Allowing to continue a given specification is important in practice to obtain tighter bounds, because tracking error sets can grow as we switch from one specification to another. For example a robot performing a maneuver can see its localization performance decrease for typical sensor packages.

IV. Example

A. Problem Formulation

The purpose of this section is to illustrate the concepts outlined above for a specific example, involving robust motion planning for a Dubins vehicle. The dynamics of the vehicle with configuration $(x, y, \theta) \in \mathbb{R}^2 \times \mathcal{S}^1$, fixed constant velocity v and minimum turning radius $1/2$ are

Algorithm 1 Robust Forward RRT Planner. $\hat{x}_0$ is the initial state estimate, G is the desired goal set.

Require: $\hat{x}_0, \hat{\mathcal{E}}_0, \hat{\mathcal{Z}}_0; G$
  $\sigma \leftarrow (r \leftarrow \hat{x}_0, \mathsf{C} \leftarrow \perp, \mathcal{E} \leftarrow \hat{\mathcal{E}}_0, \mathcal{Z} \leftarrow \hat{\mathcal{Z}}_0, \mathsf{D} \leftarrow \perp)$
  $\mathcal{T}$.addNode$(\sigma, d \leftarrow 0, \mathtt{pred} \leftarrow \perp)$
  while true **do**
    $s \leftarrow$ randomOutput$(\mathsf{Y}_{\mathtt{free}})$
    $n \leftarrow \mathcal{T}$.extend(s)
    if $r_{m_n}(d_n) + \mathcal{E}_{m_n}(d_n) \subset G$ **then**
      return $\mathcal{T}, n$
    end if
  end while

Algorithm 2 $\mathcal{T}$.extend(s). $\mathcal{T}$.nearestNeighbor(y) returns the node in $\mathcal{T}$ closest to y according to μ.

  $n_{near} \leftarrow \mathcal{T}$.nearestNeighbor$(s, \mu)$
  $n_{new} \leftarrow$ generateChildNode(n_{near}, s)
  $\mathcal{T}$.addNode(n_{new})
  return n_{new}

Algorithm 3 generateChildNode(n, s). Creates a new node with predecessor n and specification steering the system from n towards s. m_n is the mode used to reach n. closestNode returns a node closest to s from a list of candidate nodes, according to μ.

  modeList $\leftarrow$ generateCandidateMotions(n, s)
  $T =$ randomTime()
  for $\mathtt{m}$ in modeList **do**
    if $\mathtt{m} = m_n$ **then**
      {- we continue the same mode -}
      $d \leftarrow d_{\mathtt{n}} + T$
      $\sigma \leftarrow \sigma_{m_n}$
      Check for collision of the tube $Hr_{m_n}(t) + \mathcal{Z}_{m_n}(t), t \in [d_n, d_n + T]$, with obstacles
    else
      {- we switch mode -}
      Check $r_{m_n}(d_n) + \mathcal{E}_{m_n}(d_n) \subset r_{\mathtt{m}}(0) + \mathsf{C}_{\mathtt{m}}$
      $\mathcal{E}_0, \mathcal{Z}_0 \leftarrow \tau^{d_{\mathtt{n}}}_{m_{\mathtt{n}}, \mathtt{m}}$
      $\sigma \leftarrow \sigma_{\mathtt{m}}$ s.t. $\mathcal{E}_{\mathtt{m}}(0) \supset \mathcal{E}_0$, $\mathcal{Z}_{\mathtt{m}}(0) \supset \mathcal{Z}_0$
      $d \leftarrow T$
      Check for collision of the tube $r_{\mathtt{m}}(t) + \mathcal{Z}_{\mathtt{m}}(t), t \in [0, T]$, with obstacles
    end if
    if all checks passed **then**
      add to candidateNodeList the node $(\sigma, d, \mathtt{pred} \leftarrow \mathtt{n})$
    end if
  end for
  return closestNode(candidateNodeList$, s, \mu$)

described by the equations

$$\dot{x} = v\cos\theta + w_x, \quad \dot{y} = v\sin\theta + w_y,$$
$$\dot{\theta} = 2\,v\,\mathrm{sat}\left(\frac{1}{2}(u + w_\theta)\right),$$

where $\mathrm{sat}(x) = \max\{-1, \min\{1, x\}\}$. The signal $w(t) = [w_x(t), w_y(t), w_\theta(t)]^T$ represents a bounded perturbation, and we assume say $|w_x|, |w_y| \leq 0.02$, $|w_\phi| \leq 0.05$. The only available input u controls the angular rate $\dot{\theta}$. In particular, the vehicle can only move forward at fixed velocity v, and we can set $v = 1$ in the following without loss of generality. We consider a family of controllers tracking motions along half-lines with arbitrary starting point and direction.

B. Line-tracking Controller Specifications

Consider a directed half-line with orientation θ_l starting from $[r_x(0), r_y(0)]^T$. By a change to a new set of coordinates denoted (χ, δ, ϕ), the half-line becomes the χ-axis, oriented toward increasing χ-coordinates, δ is the vehicle distance to the line, and ϕ its orientation with respect to the line. The parametrization of the error tracking tubes $\mathcal{E}(t)$ and $\mathcal{Z}(t)$ as in Definition 1 for a controller tracking this half-line can be done in this coordinate system. The output coordinates are χ, δ. Assuming for simplicity that the coordinate χ can be measured exactly, the reference trajectory tracked is $r(0) = 0,\ r(t) = [\chi(t), 0]^T,\ t > 0$. That is, we do not give an open-loop specification of the χ coordinate, instead the vehicle tracks its own projection on the half-line. This simplification is possible here since the speed at which the vehicle progresses along the line is irrelevant for our basic motion planning problem.

To regulate δ and ϕ to zero, consider the simple proportional controller

$$u = -(1.3\ \delta + 0.9\ \sin\phi), \tag{8}$$

with the following assumption on the initial state

$$\mathsf{C} = \left\{\ [\chi, \delta, \phi]^T\ \middle|\ |\delta| \leq 0.9, |\phi| \leq \pi/3\ \right\}.$$

In particular, the vehicle is initially oriented in the desired direction. Assume an initial tracking error

$$\mathcal{E}(0) = \left\{[\chi, \delta, \phi]^T \middle| |\chi| \leq \epsilon_\chi, |\delta| \leq \epsilon_\delta, |\phi| \leq \epsilon_\phi\right\},$$

with $\epsilon_\delta \leq 0.9, \epsilon_\phi \leq \pi/3$. Then for $t > 0$, a Lyapunov analysis detailed in [14] shows that we can take

$$\mathcal{E}(t) = \Big\{[\chi, \delta, \phi]^T \Big| \chi = 0 \quad |\delta| \leq \max\{k_\delta(\epsilon_\delta, \epsilon_\phi)e^{-\lambda t}, \delta_\infty\}, \quad |\phi| \leq \max\{k_\phi(\epsilon_\delta, \epsilon_\phi)e^{-\lambda t}, \phi_\infty\}\Big\},$$

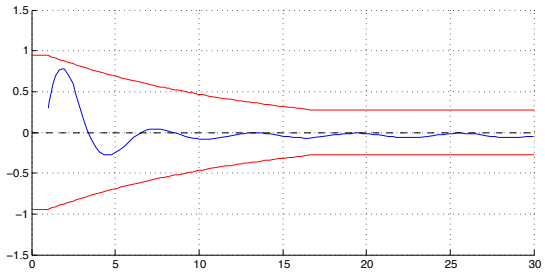

Fig. 1. Illustration of the tubular approximation for a straight line motion tracked by the vehicle, with $\phi(0) = \pi/3, \delta(0) = 0.3, \chi(0) = 1$ and perturbation $w_\chi = -0.02, w_\delta = 0.02\cos(t), w_\phi = -0.05$. The line tracked is the χ-axis, dashed.

for some functions k_δ, k_ϕ, and $\lambda = 0.0775, \delta_\infty = 0.27, \phi_\infty = 0.3469$. Fig. 1 illustrates the tube $r(t) + \mathcal{E}(t), t \geq 0$.

Finally, we consider the composition of two successive specifications. After following mode m for the duration d, the robot state is $[\chi(d), \delta(d), \psi(d)]^T \in [\chi(d), 0, 0]^T + \mathcal{E}(d)$ in the local coordinate system. We consider switching to a mode tracking a line with orientation ϕ_l with respect to the current direction, and origin $[\chi(d), 0]$ in the coordinate system of the specification σ_m. Let $\hat{\sigma}$ be the specification of this second mode. Then one can see that the composition is valid if $|\delta(d)| \leq 0.9$ and $|\phi(d)| + |\phi_l| \leq \pi/3$. Moreover, we can set

$$\hat{\mathcal{E}}(0) = \Big\{ [\hat{\chi}, \hat{\delta}, \hat{\phi}] \Big| |\hat{\chi}(0)| \leq |\delta(d)|, |\hat{\delta}(0)| \leq |\delta(d)|, \\ |\hat{\phi}(0)| \leq |\phi(d)| + |\phi_l| \Big\}$$

and use these values as $\epsilon_{\hat{\chi}}, \epsilon_{\hat{\delta}}, \epsilon_{\hat{\phi}}$. Fig. IV-B shows an example of composition of specifications reaching a desired goal set, obtained using the RRT algorithm of Section III and the specifications of this section.

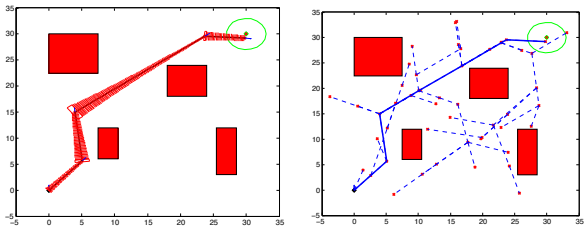

Fig. 2. Example of robust path with error tracking tubes. The vehicle starts from the lower left corner and must reach the encircled green zone with certainty. We also show the RRT tree built to obtain the solution.

V. Conclusion

This paper introduces a notion of robust controller specification, and details how to sequentially compose such specifications to establish the feasibility of a higher-level planning task, taking into account various sources of disturbance. In contrast to previously proposed motion description languages, the atoms do not specify how the controllers operate, but focus instead on their trajectory tracking performance, which can be certified by using techniques from control theory. In particular, this approach allows to refine the controllers and the planner independently. In future work it will be used to design systems that satisfy plans expressed in temporal logics, and we will study what notions of completeness can be maintained with such a decomposition.

References

[1] L. de Alfaro and T. Henzinger, "Interface theories for component-based design," in *EMSOFT 2001*, ser. LNCS 2211, 2001, pp. 148–165.

[2] R. W. Brockett, "Formal languages for motion description and map making," in *Robotics*, R. W. Brockett, Ed., vol. 41. Providence, RI: American Mathematical Society, 1990, pp. 181–193.

[3] V. Manikonda, P. S. Krishnaprasad, and J. Hendler, "Languages, behaviors, hybrid architectures and motion control," in *Mathematical Control Theory*, J. Bailleul and J. C. Willems, Eds. New-York: Springer, 1998, pp. 199–226.

[4] E. Frazzoli, M. A. Dahleh, and E. Feron, "Maneuver-based motion planning for nonlinear systems with symmetries," *IEEE Transaction on Robotics*, vol. 21, no. 6, pp. 1077–1091, December 2005.

[5] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 534–555, 1999.

[6] L. Yang and S. M. LaValle, "The sampling-based neighborhood graph: A framework for planning and executing feedback motion strategies," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 419–432, June 2004.

[7] R. Tedrake, "LQR-trees: Feedback motion planning on sparse randomized trees," *Proceedings of Robotics: Science and Systems (RSS)*, 2009.

[8] T. Lozano-Pérez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.

[9] H. Khalil, *Nonlinear Systems*, 3rd ed. Prentice Hall, 2002.

[10] E. D. Sontag and Y. Wang, "Notions of input to output stability," *Systems and Control Letters*, vol. 38, pp. 235–248, 1999.

[11] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[12] B. D. O. Anderson and J. Moore, *Optimal Control: Linear Quadratic Methods*. Dover, 2007, republication of the 1990 edition.

[13] E. D. Sontag, "Smooth stabilization implies coprime factorization," *IEEE Transactions on Automatic Control*, vol. 34, no. 4, pp. 435–443, April 1989.

[14] J. Le Ny and G. J. Pappas, "Sequential composition of robust controller specifications," University of Pennsylvania, Tech. Rep., 2012.