Planning in Dynamic Environments Through Temporal Logic Monitoring

Bardh Hoxha and **Georgios Fainekos**
Arizona State University
660 W 6th St, STE 203
Tempe, AZ 85281
{bhoxha, fainekos}@asu.edu

Abstract

We present a framework that enables online planning for robotic systems in dynamic environments. The PLANrm framework presented in this work utilizes the theory of robustness and monitoring of Metric Temporal Logic (MTL) specifications to inspect and modify available plans to both avoid obstacles and satisfy specifications in a dynamic environment. The use of MTL allows the practitioner to set complex event and timing based specifications that need to be satisfied in the execution of the plan. The monitoring algorithm inspects the possible paths in a bounded window and selects and adjusts a path to satisfy the specifications. In this paper, we present initial results on the framework and an extended summary of the algorithmic results. The approach is illustrated using a running example of a car-like model with a number of MTL specifications.

Introduction

The problem of motion planning has been researched extensively in the past few decades. As a result, robotic systems have been successfully deployed in a variety of fields, from disaster response and recovery to autonomous warehouse systems, autonomous driving and industrial manufacturing. In general, the operating environment is dynamic, where new obstacles may appear, or the mission may change. To support planning for such environments, we propose the PLANrm framework that incorporates a Metric Temporal Logic (MTL) (Koymans 1990) monitoring algorithm (Dokhanchi, Hoxha, and Fainekos 2014) that inspects possible plans in a bounded time window and selects a path that satisfies changing MTL specifications.

MTL is a formal language which enables practitioners to express complex specifications that take into account both the occurrence and timing of events through time. It enables the definition of specifications such as "*if you pass through region A within a five second period, then visit region B while avoiding region C*" or "*always visit the refueling region R before visiting region B*".

Temporal Logic has been previously utilized to synthesize controllers for motion-planning of dynamical systems. In (Fainekos, Kress-Gazit, and Pappas 2005; Fainekos et al. 2009; Kloetzer and Belta 2007), the authors convert Linear Temporal Logic (LTL) specifications to Büchi automata and create abstraction models for the dynamical systems. Then they utilize automata-based methods to synthesize controllers. One of the main challenges with this approach are the increasing computational costs due to both the length of the specification and the level of abstraction of the dynamical system. Other varieties of Temporal Logic such as MTL have been also utilized in planning. In (Kabanza 1995), the author uses MTL to create synchronized plans for teams of agents. In (Karaman and Frazzoli 2008b), the authors consider the vehicle routing problem with MTL specifications. In (Ding, Lazar, and Belta 2014), for online planning, the authors present an approach for receding horizon control for finite deterministic systems subject to LTL formulas. In (Raman et al. 2014), the authors present a method for developing model predictive controllers for systems that can be represented as a mixed integer-linear program. The controller is developed subject to Signal Temporal Logic (Maler and Nickovic 2004) specifications.

In this work, we consider the online planning problem for dynamic environments where both the environment and mission may change. Furthermore, MTL enables the specification of time bounds for temporal operators and thus the definition of more elaborate specifications. Our planning framework utilizes the notion of robustness of MTL specifications (Fainekos and Pappas 2006; 2009), with a bounded-time monitoring algorithm (Dokhanchi, Hoxha, and Fainekos 2014), to enable online planning for dynamic environments. Finally, we illustrate our framework with Matlab Simulink using a car-like model from the Robotics toolbox (Corke 2011) with various MTL specifications.

Summary of Contributions: We present an online planning problem for robotic systems where the environment is dynamically changing and mission compliance specifications may be updated. To the authors knowledge, this is the first attempt to include specification updates in an online planning problem for dynamically changing environments. In the following, we provide an overview of our solution and the planning framework. We illustrate the approach using an example of a car-like model with a number of MTL specifications.

Acknowledgments: This work has been partially supported by award NSF CNS 1446730.

Preliminaries

Metric Temporal Logic

Metric Temporal Logic (Koymans 1990) enables reasoning over quantitative timing properties of boolean signals. In this paper, we use the standard fragment of MTL with bounded future, but also we allow the use of past time operators. In the following, we assume we a constant sampling rate for the robot and thus any specification on time reduces to a specification on sampling points.

Definition 1 ($\text{MTL}^{<+\infty}_{+pt}$ Syntax) *Let AP be the set of atomic propositions and I be any non-empty interval of $\mathbb{N}$, and $\overline{I}$ be any non-empty interval of $\mathbb{N} \cup \{+\infty\}$. The set $MTL^{<+\infty}_{+pt}$ formulas is inductively defined as*

$$\varphi \ ::= \ \top \mid p \mid \neg\varphi \mid \psi \vee \varphi \mid \psi \mathcal{U}_I \varphi \mid \psi \mathcal{S}_{\overline{I}} \varphi$$

where $p \in AP$ and $\top$ stands for true.

The propositional operators conjunction ($\wedge$) and implication ($\rightarrow$) are defined the usual way. Other temporal operators are defined as follows.

Future operators:

$\bigcirc$ (Next) as $\bigcirc\varphi \equiv \top \mathcal{U}_{[1,1]} \varphi$

$\Diamond$ (Eventually) as $\Diamond_I \varphi \equiv \top \mathcal{U}_I \varphi$

$\Box$ (Always) as $\Box_I \varphi \equiv \neg\Diamond_I \neg\varphi$

The intuitive meaning of the $\psi \mathcal{U}_{[a,b]} \varphi$ operator at sampling time i is a follows: ψ has to hold at least until φ becomes true within the time interval of $[i + a, i + b]$ in the future.

Past operators:

$\odot$ (Previous) as $\odot\varphi \equiv \top \mathcal{S}_{[1,1]} \varphi$

$\Diamond$ (Eventually in the past) as $\Diamond_{\overline{I}} \varphi \equiv \top \mathcal{S}_{\overline{I}} \varphi$

$\boxdot$ (Always in the past) as $\boxdot_{\overline{I}} \varphi \equiv \neg\Diamond_{\overline{I}} \neg\varphi$

The intuitive meaning of the $\psi \mathcal{S}_{[a,b]} \varphi$ operator at sampling time i is as follows: since φ became true in the past interval $[i - b, i - a]$, ψ must hold till now (current time i).

The atomic propositions in our case label subsets of the output space $\mathcal{Y}$. Each atomic proposition is a shorthand for an arithmetic expression of the form $p \equiv g(y) \leq c$, where $g : \mathcal{Y} \rightarrow \mathbb{R}$ and $c \in \mathbb{R}$. We define an observation map $O : AP \rightarrow \mathcal{P}(\mathcal{Y})$ such that for each $p \in AP$ the corresponding set is $O(p) = \{y \mid g(y) \leq c\} \subseteq \mathcal{Y}$.

Robustness of $\text{MTL}^{<+\infty}_{+pt}$ Formulas

The robustness estimate, formally presented in (Fainekos and Pappas 2009; Abbas et al. 2013), is used to indicate how well a signal satisfies (or falsifies) MTL specifications. In the rest of the paper, we will use the terms *time state sequence* and *(execution or simulation) trace* interchangeably. We define a timed state sequence as $\mu = \mu_0 \mu_1 \mu_2 \ldots \mu_m$ where $\mu_i = (\tau_i, s_i)$ represents a tuple of the time stamp and the vector containing the values of the state variables at sampling instance i. Given a timed state sequence μ of a system output trajectory and a MTL specification φ, the robustness estimate returns a value $\rho \in \mathbb{R} \cup \{-\infty, \infty\}$. A positive (resp. negative) robustness means that the system is satisfied (resp. falsified) for a particular system output trajectory. In the following, we use the notation $[\![\varphi]\!]$ to denote the robustness estimate with which μ satisfies the specification φ. In addition to the boolean semantics, the robustness measure ρ defines how robustly μ satisfies or falsifies φ.

Formally, we define the robust semantics of $\text{MTL}^{<+\infty}_{+pt}$ as follows. Using a metric d (Seda and Hitzler 2008), we can define a distance function that captures how far away a point $x \in X$ is from a set $S \subseteq X$.

Definition 2 (Signed Distance) *Let $x \in X$ be a point, $S \subseteq X$ be a set and d be a metric. Then, we define the Signed Distance from x to S to be*

$$\textbf{Dist}_d(x, S) := \begin{cases} -\inf\{d(x, y) \mid y \in S\} & \text{if } x \notin S \\ \inf\{d(x, y) \mid y \in X \backslash S\} & \text{if } x \in S \end{cases}$$

where inf is the infimum.

Using Definition 2, we provide the formal definition for the robustness semantics.

Definition 3 ($\text{MTL}^{<+\infty}_{+pt}$ Robustness Semantics) *Let s be a trace $s : \mathbb{N} \rightarrow X$, and O be an observation map $O : AP \rightarrow P(X)$, then the robust semantics of any formula $\varphi \in MTL^{<+\infty}_{+pt}$ with respect to s is recursively defined as:*

$$\begin{aligned}
[\![\top]\!](s, i) &:= +\infty \\
[\![p]\!](s, i) &:= \textbf{Dist}_d(s(i), O(p)) \\
[\![\neg\varphi]\!](s, i) &:= -[\![\varphi]\!](s, i) \\
[\![\psi \vee \varphi]\!](s, i) &:= [\![\psi]\!](s, i) \sqcup [\![\varphi]\!](s, i) \\
[\![\psi \mathcal{U}_{[l,u]} \varphi]\!](s, i) &:= \bigsqcup_{j=i+l}^{i+u} \left([\![\varphi]\!](s, j) \sqcap \bigsqcap_{k=i}^{j-1} [\![\psi]\!](s, k) \right) \\
[\![\psi \mathcal{S}_{[l',u'\rangle} \varphi]\!](s, i) &:= \\
&\bigsqcup_{j=max\{0,i-u'\}}^{i-l'} \left([\![\varphi]\!](s, j) \sqcap \bigsqcap_{k=j+1}^{i} [\![\psi]\!](s, k) \right)
\end{aligned}$$

where $\sqcup$ stands for max, $\sqcap$ stands for min, $p \in AP$, $l, u, l' \in \mathbb{N}$ and $u' \in \mathbb{N} \cup \{\infty\}$. Furthermore, the symbol $\rangle$ in $\mathcal{S}_{[l',u'\rangle}$ will be) when $u' = +\infty$ and] when $u' \neq +\infty$.

Monitoring of $\text{MTL}^{<+\infty}_{+pt}$ Formulas

The monitoring algorithm, formally presented in (Dokhanchi, Hoxha, and Fainekos 2014), enables the online robustness estimation of $\text{MTL}^{<+\infty}_{+pt}$ formulas during system execution or simulation. Given a $\text{MTL}^{<+\infty}_{+pt}$ formula φ and a current time state sequence step i, the algorithm determines the number of time steps needed in the future (resp. past) needed to compute the robustness of φ. We refer to the number of time steps as the window size. Next, a dynamic algorithm is utilized to calculate the robustness estimate. The future (resp. past) window size for specification φ is called the horizon (resp. history) and denoted by $hrz(\varphi)$ (resp. $hst(p)$). The finite horizon and the history are defined recursively as follows:

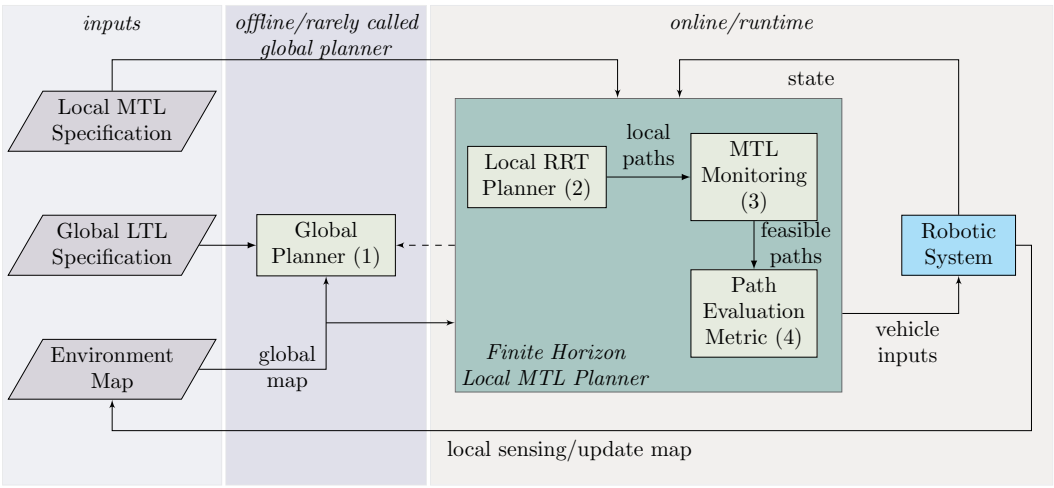

Figure 1: The planning process using the PLANrm framework.

Horizon:
$hrz(p) = 0$
$hrz(\neg\psi) = hrz(\psi)$
$hrz(\psi \ \mathbf{OP} \ \varphi) = max\{hrz(\psi), hrz(\varphi)\}$
$hrz(\psi \mathcal{U}_{[l,u]} \varphi) = max\{hrz(\psi) + u - 1, hrz(\varphi) + u\}$
$hrz(\psi \mathcal{S}_{[l',u'\rangle} \varphi) = max\{hrz(\psi), hrz(\varphi)\}$

History:
$hst(\neg\psi) = hst(\psi)$
$hst(\psi \mathcal{U}_{[l,u]} \varphi) = max\{hst(\psi), hst(\varphi)\}$
$hst(\psi \ \mathbf{OP} \ \varphi) = max\{hst(\psi), hst(\varphi)\}$
$hst(\psi \mathcal{S}_{[l',u'\rangle} \varphi) =$
$\begin{cases} max\{hst(\psi) + u' - 1, hst(\varphi) + u'\} & \text{if } u' \neq +\infty \\ max\{hst(\psi) + l' - 1, hst(\varphi) + l'\} & \text{if } u' = +\infty \end{cases}$

where $p \in AP$. Here, **OP** is any binary operator in propositional logic, and ψ, φ are $\text{MTL}^{<+\infty}_{+pt}$ formulas.

Problem Formulation

Our high level goal is to provide an online planning framework for dynamic environments where part of the mission specifications may be updated. We aim to do so by utilizing the notion of robustness and monitoring of MTL specifications. Formally, we aim to solve the following problem:

Problem 1 *Given a model of a robotic system Σ and a path p that satisfies an LTL formula φ, design an online controller which computes local paths which still satisfy φ and, moreover, they satisfy a set $\{\psi_i\}$ of potentially dynamically changing local MTL mission requirements in a dynamically changing environment.*

The planning framework is presented in Fig. 1. Given a global LTL specification φ, a global planner is utilized to generate a path from the initial position to the goal position that satisfies φ. Then, the online, finite horizon local MTL planner generates plans that take into account potentially changing MTL specifications until the destination is reached. In certain conditions, the local planner may not find a valid local path that satisfies the specification. In those cases, an updated plan from the global planner is requested. However, at this point, we must make sure that the future obligations from the local MTL requirements are preserved.

In the following, we review motion planning over LTL specifications and then we provide an overview of our online planning framework for solving Problem 1 through an example.

Review of LTL Planning

Motion planning over LTL specifications has a long history (Loizou and Kyriakopoulos 2004; Fainekos, Kress-Gazit, and Pappas 2005; Kloetzer and Belta 2007; Karaman and Frazzoli 2008a; Bhatia, Kavraki, and Vardi 2010; Plaku and McMahon 2013). One of the earliest works in LTL planning in robotics appears in (Lamine and Kabanza 2002). There, the authors use LTL runtime monitoring to check for violations of LTL specifications while the robot executes its plan. Also, the framework includes a simulation based planner with LTL goals in order to choose a behavior that would best satisfy the LTL goals.

A general approach to the high-level planning problem with LTL is as follows. First, a discretized abstraction M of the workspace using decomposition techniques is generated (Fainekos, Kress-Gazit, and Pappas 2005). Then, the LTL specification is converted into a Büchi automaton B. Next, a search algorithm over the product automaton $M \times B$ is utilized to find a feasible plan. In the LTL planning framework, partially known environments can be handled as well. For example, see the recent work in (Guo, Johansson, and Dimarogonas 2013), where the authors present an approach to locally update the product automaton when the environment changes. Closer in spirit to our work are the works by (Plaku, Kavraki, and Vardi 2009; Bhatia, Kavraki, and Vardi 2010), where the authors present an approach that combines high-level planners with sampling-based low-level planners to explore for feasible paths that satisfy LTL specifications.

In general, finding a tractable method for motion planning over MTL requirements is still an open problem. The approach utilized for LTL formulas is not feasible for MTL. MTL formulas can be represented by timed automata (Alur

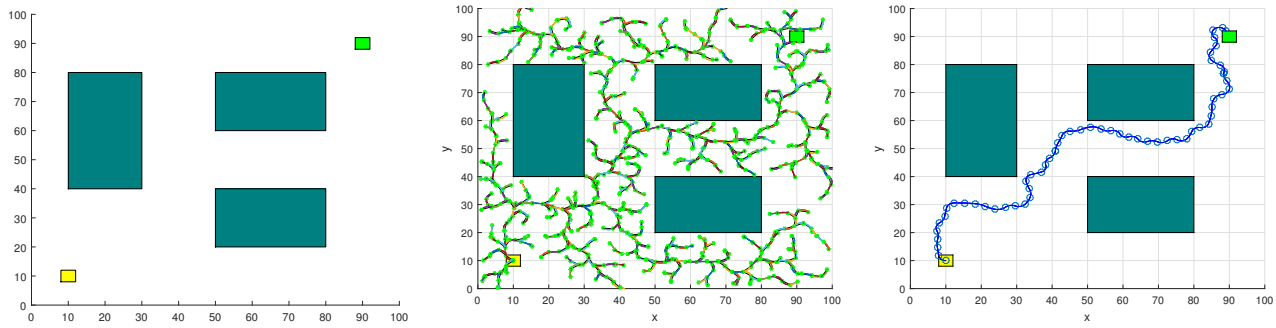

Figure 2: Left: Environment map with the initial position for the robot (yellow) at $x_{init} = (10, 10, \pi)$ and the goal position for the robot (green) at $x_{goal} = (90, 90, 0)$. Middle: RRT generated on the configuration space considering the vehicle dynamics. Right: Best path based on the RRT from x_{init} to x_{goal}.

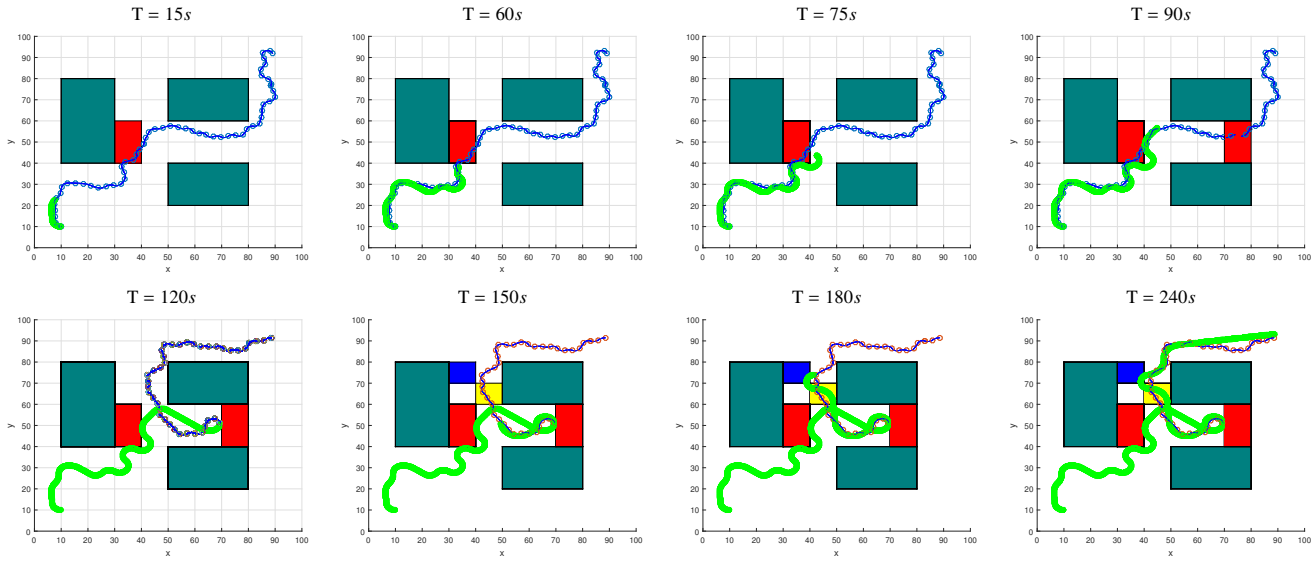

Figure 3: Vehicle progression from $x_{init} = (10, 10, \pi)$ to $x_{goal} = (90, 90, 0)$ at various times in the path traversal. The blue path represents the shortest path obtained by the global planner. The green path represents the path traversed by the vehicle. The red regions represent areas where the vehicle should not go through. Also, every time the vehicle is in the yellow set then within the next $20s$ the vehicle should enter set blue set.

and Dill 1994) and currently algorithms exist only for fragments of MTL (Maler, Nickovic, and Pnueli 2006). Therefore, in this work, we utilize LTL for the global specification.

Planning in Dynamic Environments and Specifications

In real world applications of robotic systems, both the environment and the mission can change dynamically as the system is running. In the latter case, we assume that such changes are defined as MTL specifications. We aim to utilize history and horizon window sizes for MTL specifications to determine the timespan for the local plan generator in order to generate specification compliant plans. For the local planner, we utilize MTL since it can capture specifications that include timing intervals for temporal operators. In many applications it is often necessary to capture timing properties.

We present an overview of the planing framework in Fig. 1. First, a global planner (1) is utilized to generate paths considering the environment and system dynamics. For example, using the framework presented in (Bhatia, Kavraki, and Vardi 2010). Then, the global plan is executed. If there are any local MTL formulas that must be satisfied, then a local RRT planning framework is executed. The Local Planner (2) generates paths from the current system location, while considering a dynamically changing environment and specification. The set of paths generated are then classified as feasible and infeasible through MTL Monitoring (3). Out of the feasible local plans, in the Path Evaluation Metric (4) stage, the framework choses the one that most closely corresponds to the global established path. Here, we can utilize a weighted average of the robustness metric, in conjunction with a similarity measure, such as the euclidean distance, to chose one path from the set of feasible paths. It is important to note that the local plan needs to stay close to the global plan, either in terms of path distance, regions or

some other high-level abstraction. In case no feasible paths are found, the Local Planer (2) requests an update from the global planner (1) and the process repeats until the destination is reached.

To illustrate the approach we utilize the car-like model from the Robotics Toolbox (Corke 2011). A rear wheel is fixed to the body, and a front wheel which rotates to steer the vehicle. We assume that the vehicle only moves forward. The configuration of the vehicle is represented by generalized coordinates $q = (x, y, \theta) \in C \subset SE(2)$. The kinematic model of the vehicle is presented with the following equations of motion:

$$\dot{x} = v\cos\theta$$
$$\dot{y} = v\sin\theta$$
$$\dot{\theta} = \frac{v}{L}\tan\gamma$$

where (x, y, θ) represent the (x, y) position and θ orientation of the robot, v represents the forward speed, L represents the length of the vehicle and γ is the steering angle. In our study, the only input to the system is the steering angle with a constant velocity. For more details on the model see (Corke 2011). The Matlab Simulink model is also provided in the toolbox. In Fig. 2 we present the environment along with the initial and goal positions. We also present the global plan generated using an RRT algorithm.

In Figure 3, we illustrate vehicle progression in a dynamically changing environment and mission specification. The mission goal is to go from initial position $x_{init} = (10, 10, \pi)$ to $x_{goal} = (90, 90, 0)$. At T = $15s$, a specification $\phi = \Box\neg a$ is introduced where a is the set $[30, 40] \times [40, 60]$. The specification states that set a should never be visited. At T = $60s$ and T = $75s$, the local planner generates a path that avoids set a. At T = $90s$, the specification is updated to $\phi = \Box(\neg a \wedge \neg b)$, where b is the set $[70, 80] \times [40, 60]$. At T = $120s$, the local planner fails to find a path towards x_{goal} and therefore requests an updated global plan. The global planner then presents a new route to x_{goal}. At T = $150s$, the specification is updated to $\phi = \Box(\neg a \wedge \neg b) \wedge \Box(c \rightarrow \Diamond_{[0,20]} d)$, where c is the set $[40, 50] \times [60, 70]$ and d is the set $[30, 40] \times [70, 80]$. In addition to the previous specification, the updated specification states that every time the vehicle is in set c then within the next $20s$ the vehicle should enter set d. At T = $120s$, the local planner provides a plan that satisfies the specification. It should be noted that it strays away from the plan provided by the global planner in order to satisfy the updated specification. At T = $240s$, the vehicle reaches the destination.

In practice, when an updated local MTL specification is received, we should verify that it does not contradict the global LTL specification. In certain scenarios, we can utilize the work presented in (Fainekos 2011; Dokhanchi, Hoxha, and Fainekos 2015) to detect and debug contradictory or inconsistent specifications.

The Local Planner (2) used in this framework generates a set of trajectories by slightly modifying the steering angle at a number of control points within the specification horizon/history window. Namely, given a horizon/history window size H, the steering angle is modified at t_i equidistant control points such that $H = \sum_{i=1}^{n} t_i$. This method is aimed to improve the smoothness of the trajectories generated as opposed to general RRT methods. This local planning method is illustrated in Figure 4. To generate the RRT, we utilize a slightly modified version of the algorithm provided in the Robotics Toolbox (Corke 2011). The spread-based method utilized is inspired by the works presented in (Von Hundelshausen et al. 2008; Yu et al. 2012) which may be used to generate smooth vehicle motion plans.

The set of trajectories generated from the local planner are then tested over the specifications using the MTL Monitoring (3) algorithm. We use S-TaLiRo (Annapureddy et al. 2011; Hoxha et al. 2014) to conduct the robustness computation over MTL formulas. In Figure 4, we illustrate this process with the specification $\varphi = \Box(a \rightarrow \Diamond_{[0,2]} b)$ where a is the set $[10, 12] \times [9, 11]$ and b is the set $[13, 15] \times [12, 14]$.

Once the set of feasible trajectories is provided, we utilize the Path Evaluation Metic (4) to choose the best path. Here, we can utilize the robustness metric, in conjunction with a trajectory similarity measure, to choose a path that corresponds closely to the global path provided by the global planner. We refer to this as the *selection criterion*. In our example, we utilize a weighted average of the robustness metric and the similarity measure. For a similarity measure, we utilize a Euclidean distance measure, i.e. for two trajectories T_1 and T_2, $E_d(T_1, T_2) = \sum_{i=1}^{n} \|(T_1(i), T_2(i))\| / n$, where $T(i)$ represents the vehicle position at time i.

Determining the best *selection criterion* is a challenging problem, since non-optimal choices may lead to livelock situations where the vehicle moves but does not make overall progress towards the goal. If a livelock does happen, it is important to detect it and mitigate the issue by possibly requesting a new plan from the global planner. This topic will be covered in future work.

Conclusion and Future Work

In this work, we presented an overview of the framework that enables online planning for robotic systems in dynamic environments where the mission specification may change during system operation. We illustrated the framework on a car-like model from the Robotics Toolbox (Corke 2011).

As future work, we will consider the challenges presented in the previous section. In certain situations, when no feasible paths are presented by the local planner, an updated global path is requested by the global planner. However, in those cases, the future obligations from the local MTL specification should be maintained. Also, we will investigate candidates for the *selection criterion* and conduct experimental results to evaluate them. We will also investigate methods for determining livestock conditions and how to resolve them. Finally, we plan to utilize the PLANrm framework on a more complex hybrid system and evaluate the performance and computational overhead during this process.

References

Abbas, H.; Fainekos, G. E.; Sankaranarayanan, S.; Ivancic, F.; and Gupta, A. 2013. Probabilistic temporal logic falsi-

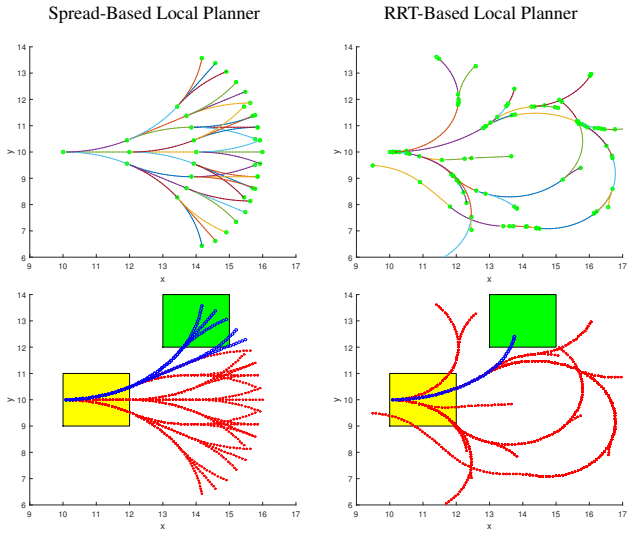

Figure 4: Top: Left: Local planner used in the bicycle model example for generating smooth and evenly distributed trajectories. Right: An additional method for generating local plans through RRTs. Bottom: Trajectories that satisfy (resp. falsify) specification $\varphi = \Box(a \rightarrow \Diamond_{[0,2]} b)$ represented in blue (resp. red) where a is the set $[10, 12] \times [9, 11]$ and b is the set $[13, 15] \times [12, 14]$.

fication of cyber-physical systems. *ACM Trans. Embedded Comput. Syst.* 12(2s):95.

Alur, R., and Dill, D. L. 1994. Theory of timed automata. *Theoretical Computer Science* 126(2):183–235.

Annapureddy, Y. S. R.; Liu, C.; Fainekos, G. E.; and Sankaranarayanan, S. 2011. S-taliro: A tool for temporal logic falsification for hybrid systems. In *Tools and algorithms for the construction and analysis of systems*, volume 6605 of *LNCS*, 254–257. Springer.

Bhatia, A.; Kavraki, L. E.; and Vardi, M. Y. 2010. Sampling-based motion planning with temporal goals. In *International Conference on Robotics and Automation*, 2689–2696. IEEE.

Corke, P. I. 2011. *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer.

Ding, X.; Lazar, M.; and Belta, C. 2014. Ltl receding horizon control for finite deterministic systems. *Automatica* 50(2):399–408.

Dokhanchi, A.; Hoxha, B.; and Fainekos, G. 2014. On-line monitoring for temporal logic robustness. In *Runtime Verification*, 231–246. Springer.

Dokhanchi, A.; Hoxha, B.; and Fainekos, G. 2015. Metric interval temporal logic specification elicitation and debugging. MEMOCODE.

Fainekos, G., and Pappas, G. J. 2006. Robustness of temporal logic specifications. In *Formal Approaches to Testing and Runtime Verification*, volume 4262 of *LNCS*, 178–192. Springer.

Fainekos, G., and Pappas, G. J. 2009. Robustness of temporal logic specifications for continuous-time signals. *Theor. Comput. Sci.* 410(42):4262–4291.

Fainekos, G.; Girard, A.; Kress-Gazit, H.; and Pappas, G. J. 2009. Temporal logic motion planning for dynamic robots. *Automatica* 45(2):343–352.

Fainekos, G. E.; Kress-Gazit, H.; and Pappas, G. J. 2005. Temporal logic motion planning for mobile robots. In *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*, 2020–2025. IEEE.

Fainekos, G. E. 2011. Revising temporal logic specifications for motion planning. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 40–45. IEEE.

Guo, M.; Johansson, K. H.; and Dimarogonas, D. V. 2013. Revising motion planning under linear temporal logic specifications in partially known workspaces. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 5025–5032. IEEE.

Hoxha, B.; Bach, H.; Abbas, H.; Dokhanchi, A.; Kobayashi, Y.; and Fainekos, G. 2014. Towards formal specification visualization for testing and monitoring of cyber-physical systems. In *Int. Workshop on Design and Implementation of Formal Tools and Systems*.

Kabanza, F. 1995. Synchronizing multiagent plans using temporal logic specifications. In Lesser, V., ed., *Proceedings of the First International Conference on Multi–Agent Systems*, 217–224. San Francisco, CA: MIT Press.

Karaman, S., and Frazzoli, E. 2008a. Complex mission optimization for multiple-uavs using linear temporal logic. In *American Control Conference, 2008*, 2003–2009. IEEE.

Karaman, S., and Frazzoli, E. 2008b. Vehicle routing problem with metric temporal logic specifications. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, 3953–3958. IEEE.

Kloetzer, M., and Belta, C. 2007. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *Robotics, IEEE Transactions on* 23(2):320–330.

Koymans, R. 1990. Specifying real-time properties with metric temporal logic. *Real-Time Systems* 2(4):255–299.

Lamine, K. B., and Kabanza, F. 2002. Reasoning about robot actions: A model checking approach. In *Advances in Plan-Based Control of Robotic Agents*, volume 2466 of *LNCS*, 123–139. Springer.

Loizou, S. G., and Kyriakopoulos, K. J. 2004. Automatic synthesis of multi-agent motion tasks based on LTL specifications. In *Proceedings of the 43rd IEEE Conference on Decision and Control*.

Maler, O., and Nickovic, D. 2004. Monitoring temporal properties of continuous signals. In *Proceedings of FORMATS-FTRTFT*, volume 3253 of *LNCS*, 152–166.

Maler, O.; Nickovic, D.; and Pnueli, A. 2006. From MITL to Timed Automata. In *Proceedings of FORMATS*, volume 4202 of *LNCS*, 274–289. Springer.

Plaku, E., and McMahon, J. 2013. Combined mission and motion planning to enhance autonomy of underwater vehi-

cles operating in the littoral zone. In *Workshop on Combining Task and Motion Planning at IEEE International Conference on Robotics and Automation (ICRA13)*.

Plaku, E.; Kavraki, L. E.; and Vardi, M. Y. 2009. Falsification of ltl safety properties in hybrid systems. In *Proc. of the Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, volume 5505 of *LNCS*, 368 – 382.

Raman, V.; Donzé, A.; Maasoumy, M.; Murray, R. M.; Sangiovanni-Vincentelli, A.; Seshia, S.; et al. 2014. Model predictive control with signal temporal logic specifications. In *Decision and Control (CDC), 2014 IEEE 53rd Annual Conference on*, 81–87. IEEE.

Seda, A. K., and Hitzler, P. 2008. Generalized distance functions in the theory of computation. *The Computer Journal* 53(4):bxm108443–464.

Von Hundelshausen, F.; Himmelsbach, M.; Hecker, F.; Mueller, A.; and Wuensche, H.-J. 2008. Driving with tentacles: Integral structures for sensing and motion. *Journal of Field Robotics* 25(9):640–673.

Yu, H.; Gong, J.; Iagnemma, K.; Jiang, Y.; and Duan, J. 2012. Robotic wheeled vehicle ripple tentacles motion planning method. In *Intelligent Vehicles Symposium (IV), 2012 IEEE*, 1156–1161. IEEE.