To appear, 2011 International Conference on Cyber-Physical Systems (ICCPS)
http://www.cds.caltech.edu/~murray/papers/otwm11-iccps.html

Distributed Synthesis of Control Protocols for Smart Camera Networks

Necmiye Ozay, Ufuk Topcu, Tichakorn Wongpiromsarn and Richard M. Murray

***Abstract*— We consider the problem of synthesizing control protocols for smart camera networks where the goal is to guarantee that certain linear temporal logic (LTL) specifications related to a given surveillance task are met. We first present a centralized control architecture for assigning pan-tilt-zoom (PTZ) cameras to targets so that the specification is met for any admissible behavior of the targets. Then, in order to alleviate the computational complexity associated with LTL synthesis and to enable implementation of local control protocols on individual PTZ cameras, we propose a distributed synthesis methodology. The main idea is to decompose the global specification into local specifications for each PTZ camera. A thorough design example is presented to illustrate the steps of the proposed procedure.**

I. INTRODUCTION

Video surveillance has become a standard for indoor/outdoor security applications and traffic monitoring [1]. A substantial research effort has been devoted to develop reliable computer vision algorithms for background subtraction, object detection/recognition/identification and tracking [2]. A higher level goal is to build systems that can reason about the sensed and processed data in order to perform complex tasks or make high level decisions about the dynamic environment monitored. Distributed smart cameras are real-time distributed embedded systems performing computer vision tasks such as tracking and recognition [3]. Smart camera systems can sense and process data; and can report "interesting" events via a communication unit. Complementing such systems with mobile cameras, such as pan-tilt-zoom (PTZ) cameras can provide increased autonomy [3]. Moreover, as argued by Soatto [4], active cameras have an information theoretic advantage over the stationary ones.

In this paper, we consider a hierarchical visual surveillance system similar to that in [5]. The idea is to supplement a stationary camera network used for tracking with additional PTZ cameras to guarantee certain logic specifications. We assume that the tracking subsystem tracks (almost in real-time) the targets in the area of interest using a multi-camera multi-target tracking algorithm and reports the target positions. Active recognition subsystem, consisting of PTZs, is responsible for zooming into each target to capture high-resolution images that can be used for recognition. Our goal is to automatically synthesize a control protocol that assigns PTZ cameras to targets to ensure that each target has a close-up taken before leaving the area. The challenge in this problem is two-fold: (i) the target behaviors are not known a priori, (ii) the number of the targets in the area at a given time could be greater than the number of PTZ cameras available.

The networked visual surveillance system we consider is a complex system combining cyber aspects like communication between the cameras, computation both for video processing and decision-making, with physical aspects like dynamics of the PTZ cameras. Design and verification of distributed sense and control systems that are interacting with a potentially dynamic environment is a challenging task. In a networked system, distributing the decision-making to individual agents has several advantages. First, since the information and computation load decreases, such distributed protocols can be implemented on simple devices, on each PTZ in our case. Second, a distributed protocol is more robust to failures in that even though one of the controllers fails, the other parts of the system will continue to function. However, these advantages come along with certain design challenges. In particular, the information flows and the cooperative behavior between distributed agents should be taken into account in the design.

We consider this synthesis problem in a discrete setup where we use a discrete abstraction of the admissible behavior of the targets and PTZ camera dynamics. In general, one can start with a continuos or hybrid model and reduce it to a discrete one [6], [7], [8]. Our goal is to synthesize a supervisory control protocol that determines at each time to where the PTZ cameras should zoom in so that certain security related specification is satisfied. We use linear temporal logic (LTL) as a specification language [9], [10] and exploit the tools for controller synthesis for LTL specifications [11], [12]. Then, we consider decomposing the global specification into local ones in order to enable distributed synthesis and implementation of local control protocols on each PTZ. Synthesizing distributed implementations from global specifications is generally hard [13], [14], [15]. However for certain architectures, it is possible to synthesize distributed controllers for local specifications [16]. We show that if the PTZ cameras in the network are weakly coupled (e.g. if their areas of coverage do not totally overlap), it is possible to derive local specifications. We further show that when these local specifications are too conservative, it is possible to refine them by adding new specifications through the existing interfaces along which PTZ cameras can collaborate. These ideas are illustrated with an example for which the design steps are explained in detail.

The rest of the paper is organized as follows. In section II, we summarize linear temporal logic and the digital design synthesis problem. We recast the problem of control

This work is supported in part by MuSyC.

N. Ozay, U. Topcu and R.M. Murray are with Control and Dynamical Systems, California Institute of Technology, Pasadena, CA 91125, USA. T. Wongpiromsarn is with Singapore-MIT Alliance for Research & Technology, Singapore. necmiye, utopcu, murray@cds.caltech.edu; nok@smart.mit.edu

protocol synthesis for PTZ cameras into a digital design synthesis problem in section III. In section IV, we propose a centralized and a distributed design procedure to solve this problem. We work through the details of the design procedure on an example in section V. Finally, section VI concludes the paper with some remarks and directions for future research.

II. BACKGROUND

Formal methods are mathematical techniques for rigorously analyzing a design to ensure system correctness. These approaches rely on constructing a mathematical representation of a system (i.e., a model) and its specification (i.e., desired properties). Examples of such mathematical objects typically used in modeling systems include finite state machines, differential equations, timed automata and hybrid automata. ω-regular languages and temporal logics are widely used to describe system specifications [17]. Thanks to their expressive power, a wide class of properties including deadlocks, livelocks, correctness of system invariants, safety, stability and non-progress execution cycles can be precisely specified.

In this section, we first describe linear temporal logic, which is used throughout the paper as a specification language. Then, we provide a brief summary of automatic synthesis of digital designs that satisfy a large class of properties expressed in linear temporal logic even in the presence of an adversarial environment [12].

A. Linear temporal logic

Temporal logic is a branch of logic that implicitly incorporates temporal aspects and can be used to reason about infinite sequences [17], [10], [18]. Its use as a specification language was introduced by Pnueli [9]. Since then, temporal logic has been demonstrated to be an appropriate specification formalism for reasoning about various kinds of systems, and has been utilized to specify and verify behavioral properties in various applications [19], [20], [21], [22]. The version of temporal logic we employ in this paper is LTL. Before we formally describe LTL, we define some of the relevant concepts.

Definition 1: A system consists of a set V of variables. The *domain* of V, denoted by $dom(V)$, is the set of valuations of V. A *state* of the system is an element $v \in dom(V)$.

Definition 2: An *atomic proposition* is a statement on system variables v that has a unique truth value ($True$ or $False$) for a given value of v. Let $v \in dom(V)$ be a state of the system and p be an atomic proposition. We write $v \Vdash p$ if p is $True$ at the state v. Otherwise, we write $v \nVdash p$.

Definition 3: A *finite transition system* is a tuple $\mathbb{T} = (\mathcal{V}, \mathcal{V}_0, \mathcal{R})$ where $\mathcal{V}$ is a finite set of states, $\mathcal{V}_0 \subseteq \mathcal{V}$ is a set of initial states, and $\mathcal{R} \subseteq \mathcal{V} \times \mathcal{V}$ is a transition relation.

An *execution* of a finite transition system is an infinite sequence of its states $\sigma = v_0 v_1 v_2 \dots$ where $v_0 \in \mathcal{V}_0$, for each $i > 0$, $v_i \in \mathcal{V}$ and $(v_{i-1}, v_i) \in \mathcal{R}$.

LTL has two kinds of operators: logical connectives and temporal modal operators. The logic connectives are those used in propositional logic: *negation* ($\neg$), *disjunction* ($\vee$), *conjunction* ($\wedge$) and *material implication* ($\rightarrow$). The temporal modal operators include *next* ($\bigcirc$), *always* ($\Box$), *eventually* ($\Diamond$) and *until* ($\mathcal{U}$). An LTL formula is defined inductively as follows:

1) any atomic proposition p is an LTL formula; and
2) given LTL formulas φ and ψ, $\neg\varphi$, $\varphi \vee \psi$, $\bigcirc\varphi$ and $\varphi \,\mathcal{U}\, \psi$ are also LTL formulas.

Other operators can be defined as follows: (a) $\varphi \wedge \psi \triangleq \neg(\neg\varphi \vee \neg\psi)$, (b) $\varphi \rightarrow \psi \triangleq \neg\varphi \vee \psi$, (c) $\Diamond\varphi \triangleq True\, \mathcal{U}\, \varphi$, and (d) $\Box\varphi \triangleq \neg\Diamond\neg\varphi$.

A *propositional formula* is one that does not include temporal operators. Given a set of LTL formulas $\varphi_1, \dots, \varphi_n$, their *Boolean combination* is an LTL formula formed by joining $\varphi_1, \dots, \varphi_n$ with logical connectives.

Semantics of LTL: An LTL formula is interpreted over an infinite sequence of states. Given an execution $\sigma = v_0 v_1 v_2 \dots$ and an LTL formula φ, we say that φ *holds at position* $i \geq 0$ of σ, written $v_i \models \varphi$, if and only if (iff) φ holds for the remainder of the execution σ starting at position i. The semantics of LTL is defined inductively as follows:

1) For an atomic proposition p, $v_i \models p$ iff $v_i \Vdash p$;
2) $v_i \models \neg\varphi$ iff $v_i \not\models \varphi$;
3) $v_i \models \varphi \vee \psi$ iff $v_i \models \varphi$ or $v_i \models \psi$;
4) $v_i \models \bigcirc\varphi$ iff $v_{i+1} \models \varphi$; and
5) $v_i \models \varphi \,\mathcal{U}\, \psi$ iff there exists $j \geq i$ such that $v_j \models \psi$ and $\forall k \in [i, j), v_k \models \varphi$.

Based on this definition, $\bigcirc\varphi$ holds at position i of σ iff φ holds at the next state v_{i+1}, $\Box\varphi$ holds at position i iff φ holds at every position in σ starting at position i, and $\Diamond\varphi$ holds at position i iff φ holds at some position $j \geq i$ in σ.

Definition 4: An execution $\sigma = v_0 v_1 v_2 \dots$ *satisfies* φ, denoted by $\sigma \models \varphi$, if $v_0 \models \varphi$.

Definition 5: Let Σ be the set of all executions of a system. The system is said to be *correct* with respect to its specification φ, written $\Sigma \models \varphi$, if all its executions satisfy φ.

B. Synthesis of a digital design: a two-player game approach

In many applications, systems need to interact with their environments and whether they satisfy the desired properties depends on the behavior of the environments. For example, the feasibility of a surveillance task depends on how the targets move and one should aim at designing a camera network that could achieve a given task for a wide class of target motions. In this section, we briefly describe the work of Piterman, et al. [12]. We refer the reader to [12] and references therein for detailed discussion of automatic synthesis of a finite state automaton from its specification.

We refer the controllable part of the system (i.e., PTZ cameras) as *plant*. When we say *system*, we refer the combined behavior of the environment and the controlled plant. From Definition 5, for a system to be correct, its specification φ must be satisfied by all of its executions regardless of the behavior of the environment in which it operates. Thus, the environment can be treated as an adversary and the synthesis

problem can be viewed as a two-player game between the plant and the environment: the environment attempts to falsify φ while the plant attempts to satisfy φ. Let $\mathcal{E}$ and $\mathcal{P}$ be the variables of the environment and the plant respectively. A state $s = (e, p)$ of the game is in $dom(\mathcal{E}) \times dom(\mathcal{P})$. A transition of the game is a move of the environment $\mathcal{R}_e$ followed by a move of the plant $\mathcal{R}_p$. A *strategy* for the plant is a partial function $f : (s_0 s_1 \dots s_t, p_t) \mapsto p_t$ which chooses a move of the plant among its allowable moves based on the state sequence so far and the behavior of the environment. In this sense a *control protocol* is a winning strategy for the plant such that for all behaviors of the environment the specification is met. We say that φ is *realizable* if such a control protocol exists, that is the system can satisfy φ no matter what the environment does.

For specifications in the form of the so called *Generalized Reactivity(1)* formulas, Piterman, et al. show that checking its realizability and synthesizing the corresponding automaton can be performed in polynomial time in the number of states of the game automaton. In particular, we are interested in a specification of the form

$$\varphi \doteq (\varphi_e \rightarrow \varphi_s) \tag{1}$$

where roughly speaking, φ_e characterizes the assumptions on the environment and φ_s describes the correct behavior of the system, including the valid transitions the plant can make. We refer the reader to [12] for precise definitions of φ_e and φ_s. Note that since Eq. (1) is satisfied whenever φ_e is *False*, i.e., whenever the assumptions on the environment φ_e are violated, then the correct behavior φ_s of the system is not ensured, even though the specification φ is satisfied.

If the specification is realizable, the digital design synthesis tool such as JTLV [12] generates a finite state automaton that represents a set of transitions the plant should follow in order to satisfy φ. Assuming that the environment satisfies φ_e, then at any instance of time, there exists a node in the automaton that represents the current state of the system and the system can follow the transition from this node to the next based on the current knowledge about the environment. However, if φ_e is violated, the automaton is no longer valid, meaning that there may not exist a node in the automaton that represents the current state of the system, or even though such a node exists and the system follows the transitions in the automaton, the correct behavior φ_s is not guaranteed.

If the specification is not realizable, the synthesis tool provides an initial state of the system starting from which there exists a set of moves of the environment such that the system cannot satisfy φ. The knowledge of the nonrealizability of the specification is useful since it provides information about the conditions under which the system will fail to satisfy its desired properties.

The main limitation of the synthesis of finite state automata is the state explosion problem. In the worst case, the resulting automaton may contain all the possible states of the system. For example, if the system has n variables, each can take any value in $\{1, \dots, M\}$, then there may be as many as M^n nodes in the automaton. On the other hand, if the system can be decomposed into N subsystems each having around n/N variables and the specification can be divided into N pieces each depending only on the corresponding variables, then there would be N automata with size in the order of $\sqrt[N]{M^n}$. We exploit this observation in order to overcome the state explosion problem. In particular, we propose a distributed synthesis scheme that starts with decomposing the specification so that it is possible to solve smaller synthesis problems for each PTZ camera separately.

III. PROBLEM FORMULATION

We would like to recast the problem of assigning PTZ cameras to targets into a game as in Section II-B. To this effect, in what follows we summarize the system model and the system specification (which consists of the desired behavior of the system and the assumptions on the environment). For simplicity, we assume that the area monitored is divided into cells as in Fig. 1. We also assume that all the cells are within the area of coverage of at least one of the PTZ cameras.

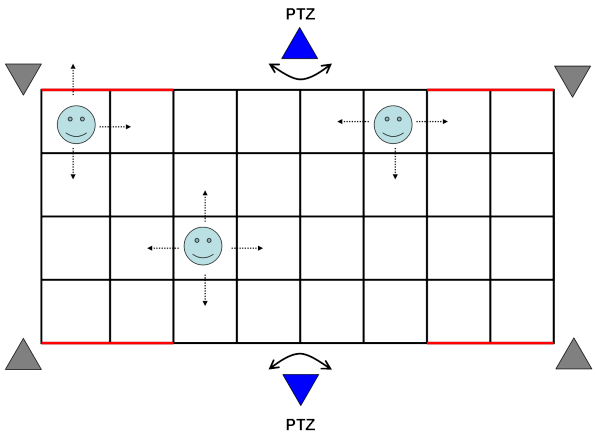

Fig. 1. PTZ Camera Network Setup. Gray triangles are stationary cameras. Blue triangles are PTZ cameras. Red lines denote enterances/exits of the area.

We first define the system model which is an abstraction that captures the PTZ camera properties such as field of view and dynamics. By field of view, we mean the region a PTZ camera sees when it zooms in to get a high resolution image. All possible regions from which a PTZ camera can take high resolution images is referred to as the area of coverage of the camera. The active recognition subsystem to be controlled consists of N_{PTZ} PTZ cameras. When zoomed in, the field of view of each camera i is limited. This limitation together with the camera dynamics can be specified either as, a PTZ camera can capture high resolution images from C_1 neighboring cells in one time step; or by taking the delays into account as, a PTZ can traverse at most C_2 neighboring cells and can capture at most C_1 images from the C_2 cells it scanned in a time step. Using this information, it is possible to build a finite transition system that models how the controllable variables of the game (i.e., PTZ cameras) can evolve.

Next, we characterize all possible behaviors of the environment against which correct behavior of the system is

expected. If there is no information with regard to the motion capabilities of the targets (i.e., there is no restriction on the environment behavior), the controller needs to deal with a large amount of uncertainty in which case many interesting tasks would be unachievable. Assumptions restricting the behavior of the environment based on the knowledge about the target motion should be incorporated into the model in order to enrich the set of achievable properties. Such an environment model can be learnt, for instance, by examining the statistics of target tracks within the area of interest over a period of time. Some sample assumptions modeling the environment can be as follows:

- There can be at most N_p people in the area at the same time.
- There can be at most one person in a cell at a given time.
- Every person always eventually exits the area.
- Everyone remains at least T time steps in the area.
- People can only enter and exit through designated enter/exit spots (e.g. red lines in Fig. 1).
- A person can move to one of the neighboring cells or stay at the same cell in one time step.

Finally, we state the system requirement that needs to be guaranteed by the system as long as the assumptions on the environment hold true. In particular, we want each person to be zoomed-in (i.e., have a high resolution picture taken) at least once when they are in the area. It is also possible to include progress requirements such as a certain cell or all cells should be scanned infinitely often, for instance, for checking left/unoccupied items.

The control protocol decides which camera should zoom into which cells taking into account the target locations, current configurations of the cameras, dynamics of the cameras and dynamics of the targets. If a camera zooms into a cell occupied by a target, it takes a high resolution image of that target. We do not keep target identity after a target exits the area[1]. We next define the variables and their domains that are used in formal problem statement.

Environment Variables: There are N_p variables $\mathbf{e}^{(i)}$, $i = \{1, \ldots, N_p\}$ for the environment. Each $\mathbf{e}^{(i)}$ has three fields, $(x^{(i)}, n^{(i)}, isZoomed^{(i)})$. $x^{(i)}$ denotes the location of the target corresponding to the variable i and takes values from a set $\mathcal{L}$ which contains the possible target[2] locations including an element c_0 representing the out of the area. $n^{(i)}$ is a counter that takes values in $\{0, 1, \ldots, T\}$ that shows for how long target i stayed in the area. $isZoomed^{(i)}$ is a boolean flag that takes the value $True$ after the target has been zoomed-in and $False$ otherwise.

[1]If someone exits the scene and enters back in, we treat it as a new person and want to take a high resolution image of him/her. Note that without zooming in, it is hard to tell if he/she is a person that has already been seen earlier since identification/face recognition engines work more reliably with high resolutions images. Therefore, this assumption is indeed desired in a surveillance system.

[2]With slight abuse of terminology, we say "target" or "target i" to refer to the target associated with the i^{th} environmental variable rather than a specific target identity.

Controllable (Plant) Variables: There are N_{PTZ} states $\mathbf{p}^{(j)}$, one for each PTZ camera. Each $\mathbf{p}^{(j)}$ has C_1 fields, $(z^{(j),1}, \ldots, z^{(j),C_1})$, that correspond to the locations of the last C_1 cells scanned by camera j with $z^{(j),k} \in \mathcal{L} \backslash c_0$ for all j, k. We drop the index k when the cameras can take a high quality image from a single location (i.e., cell) at a time step.

We call $x^{(i)}$'s independent environment variables since their evolution is solely determined by the target motions. On the other hand, we call $n^{(i)}$ and $isZoomed^{(i)}$ dependent since the former is a function of target motion and the latter is an output of the interaction between the targets and the PTZ cameras.

The system properties, environment assumptions and requirements listed above can be encoded using LTL formulas in the environment and controllable variables. For instance, the assumption that every person always eventually exits the area can be written as: $\square\lozenge(x^{(i)} = c_0)$ for all $i \in \{1, \ldots N_p\}$; or the assumption that everyone remains at least T time steps in the area can be written as: $\square\left((x^{(i)} \neq c_0 \wedge n^{(i)} < T) \rightarrow \bigcirc(x^{(i)} \neq c_0)\right)$ for all i. Using LTL, we can express the system model, the assumptions φ_e on the environment variables and the desired behavior φ_s of the system with a single formula of the form (1), i.e., whenever the environment variables satisfy their assumptions, then the system meets its requirements. The problem we would like to solve can be formally stated as follows:

Problem 1: Synthesize a control protocol such that (1) holds.

This problem can be solved using the technique described in Section II-B. If the specification is realizable, the procedure in [12] gives us a control protocol. If it is unrealizable, the procedure provides a counterexample for which there is no control strategy to prevent a target leave the scene without having a high resolution image taken. For the latter case, one can modify the design of the camera network (e.g. increase the number of PTZs) and check the realizability again.

IV. SYNTHESIS OF CONTROL PROTOCOLS

A. Centralized Control Protocol Synthesis

In this section, we consider a centralized control architecture as shown in Fig. 2. In this set-up, a control protocol that solves Problem 1 is implemented on the discrete planner. This system is centralized in the sense that a central discrete planner needs to (i) collect all the tracking data from static cameras and current PTZ positions, (ii) decide on the strategy based on the current system state, and (iii) send the appropriate target assignments, the PTZs should follow, back to the continuous controllers on PTZs. All the communication and control scheme is governed by these rules and has the discrete planner at its center.

In this scheme, the discrete planner uses data from each target and each camera to make a decision. Although the centralized design is less conservative in terms of realizability, due to state explosion problem, it is inefficient in design phase. Since each PTZ camera has an embedded controller, it is be possible to design a decentralized control

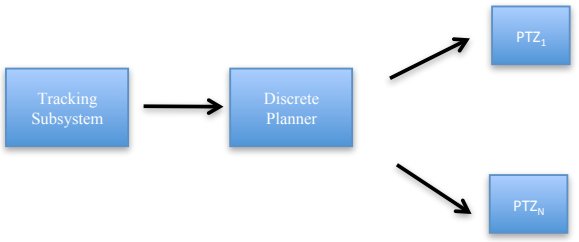

Fig. 2. Centralized control architecture for camera network. Arrows denote the communication.

protocol. This can be done, for instance, by incorporating some communication constraints into the centralized design as in [23] so that the resulting controllers can be implemented on individual PTZs. However, a more efficient approach would be not only to decentralize the implementation of the control protocol but also to distribute the computations in the synthesis which we discuss next.

B. Distributed Control Protocol Synthesis

Modern PTZ cameras have onboard computational capabilities. Instead of having a central discrete planner as in Fig. 2, it is possible to implement local planners on each PTZ. This requires synthesizing a distributed control protocol. Although it is not always possible to achieve the performance of a centralized design; when the system is composed of weakly coupled subsystems, it is possible to distribute the design without too much conservatism. We now exploit this idea to synthesize distributed control protocols that can be implemented on individual PTZs. This approach not only leads to a compositional system but also substantially reduces the computational cost of the synthesis procedure. The corresponding architecture is shown in Fig. 3. In this set-up each PTZ has its own local planner. The tracking subsystem sends only the tracks related to a PTZ camera's mission to that PTZ (e.g. tracks of targets within that camera's area of coverage). PTZ cameras can also communicate with each other and cooperate if necessary.

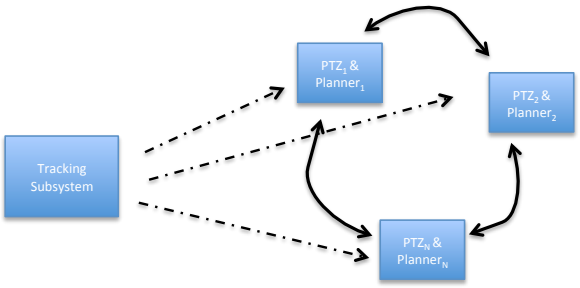

Fig. 3. Distributed control architecture for camera network. Arrows denote the communication. Dashed line from tracking subsystem represents the fact that only partial track information is communicated to the PTZ cameras.

Next, we discuss how the global specification can be decomposed into local ones so that the control protocols that are implemented on each local planner can be synthesized separately[3]. The following proposition is the basis of our approach.

Proposition 1: Let φ_e, φ_{e_1}, φ_{e_2}, φ_s, φ_{s_1} and φ_{s_2} be LTL formulas that contain variables only from the respective sets of environment variables $\mathcal{E}$, $\mathcal{E}_1$, $\mathcal{E}_2$ and system variables $\mathcal{S}$, $\mathcal{S}_1$, $\mathcal{S}_2$. Let $\mathcal{P}$, $\mathcal{P}_1$, $\mathcal{P}_2$ be the sets of all controllable variables in $\mathcal{S}$, $\mathcal{S}_1$, $\mathcal{S}_1$ that satisfy $\mathcal{P}_1 \cup \mathcal{P}_2 = \mathcal{P}$ and $\mathcal{P}_1 \cap \mathcal{P}_2 = \emptyset$. If the conditions:

1) any execution of the environment that satisfies φ_e; also satisfies $(\varphi_{e_1} \wedge \varphi_{e_2})$,
2) any execution of the system that satisfies $(\varphi_{s_1} \wedge \varphi_{s_2})$; also satisfies φ_s,
3) and, there exist two control protocols that make the local specifications $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ true,

hold, then implementing these two control protocols together would lead to a system where the global specification

$$\varphi \doteq (\varphi_e \rightarrow \varphi_s)$$

is met.

Proof: The conditions on $\mathcal{P}$, $\mathcal{P}_1$ and $\mathcal{P}_2$ ensure that the synthesized local control protocols do not conflict and can be implemented separately at the same time[4]. Let $\nu_1 \doteq ((\varphi_{e_1} \rightarrow \varphi_{s_1}) \wedge (\varphi_{e_2} \rightarrow \varphi_{s_2}))$ and $\nu_2 \doteq ((\varphi_{e_1} \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_1} \wedge \varphi_{s_2}))$. It can be shown that any execution of the system that satisfies ν_1, also satisfies ν_2. That is, if there exist control protocols as in condition 3 of the proposition, the system meets the specification ν_2 when these control protocols are implemented simultaneously. Conditions 1 and 2 respectively mean $\varphi_e \rightarrow (\varphi_{e_1} \wedge \varphi_{e_2})$ and $(\varphi_{s_1} \wedge \varphi_{s_2}) \rightarrow \varphi_s$ are tautologies (i.e., they evaluate to $True$ for any execution). Hence, it follows that for all executions that satisfy ν_2, $(\varphi_e \rightarrow \varphi_s)$ is satisfied. Therefore, for all executions that satisfy ν_1, specification φ is met. ■

There are two factors that should be taken into account while choosing e_j, s_j and φ_{e_j}, φ_{s_j}, $j \in \{1, 2\}$. The first is the number of variables involved in the local synthesis problems. If the possible valuations of the variables involved in local specifications $\varphi_{e_j} \rightarrow \varphi_{s_j}$ are substantially less than the possible valuations of the variables in the global specification, then distributed synthesis would be computationally more efficient than the centralized one (assuming the lengths of LTL formulas for the global and the local specifications are of the same order). The second is the conservatism of the distributed synthesis. Since Proposition 1 provides only sufficient conditions, it is possible that even if the centralized problem is realizable, the local specifications in decentralized synthesis may be unrealizable. Indeed, let the sets of executions be defined as:

$$\Sigma_e = \{\sigma | \sigma \models \varphi_e\}; \quad \Sigma_{e'} = \{\sigma | \sigma \models (\varphi_{e_1} \wedge \varphi_{e_2})\};$$
$$\Sigma_s = \{\sigma | \sigma \models \varphi_s\}; \quad \Sigma_{s'} = \{\sigma | \sigma \models (\varphi_{s_1} \wedge \varphi_{s_2})\}.$$

[3]For notational simplicity, we discuss decomposing a system into two subsystems. The generalization to N subsystems follows similar lines.

[4]If the same controllable variable is included in two different local specifications, the corresponding local control protocols might assign different moves to this variable. Hence, these protocols can not be implemented simultaneously.

Condition 1 in Proposition 1 implies that $\Sigma_{e'} \supseteq \Sigma_e$, whereas condition 2 implies that $\Sigma_{s'} \subseteq \Sigma_s$. Local variables and specifications should be chosen so that conditions 1 and 2 are satisfied. Moreover, the conservatism can be reduced by choosing φ_{e_j} and φ_{s_j} such that $\Sigma_{e'}$ is as small as possible, and the set $\Sigma_{s'}$ is as large as possible in the sense of set inclusion order.

In a PTZ camera network the choice of local specifications and variables involved in them is guided by physical constraints such as network topology, areas of coverage of cameras; and domain knowledge such as knowledge of motion patterns of the targets. When the system consists of weakly coupled subsystems (e.g. not totally overlapping areas of coverage), it may be possible to find a tradeoff between conservatism and computational complexity. To start with, we fix the communication rules between cameras. We assume that all PTZ cameras receive the locations (i.e., tracks) of targets within their areas of coverage from the tracking subsystem. By assumption, the $isZoomed$ and n values of a target that gets into the area of coverage of a PTZ from outside of the area monitored, is $False$ and 1, respectively. Targets that have been already in the area but just enter to the area of coverage of k^{th} PTZ, should have been in the area of coverage of l^{th} PTZ, for some $k \neq l$. Then the $isZoomed$ and n values of such targets are sent from the k^{th} PTZ to the l^{th} PTZ.

For simplicity, let us assume that there are two PTZ cameras, denoted by PTZ_1 and PTZ_2. We next give brief guidelines on how to decompose the global specification into local ones for each PTZ camera. A local specification for a PTZ camera only involves the dynamics of that camera and targets within that camera's area of coverage. For instance, the requirement that no one exits the area before having a high resolution image taken can be divided into local requirements such that no target within the area of coverage of PTZ_i ($i \in \{1,2\}$) exits the area before being zoomed in. This requirement together with the camera dynamics constitute φ_{s_i}. Note that with such a local requirement, PTZ_i does not aim to capture images of targets that leave its area of coverage through PTZ_j's area of coverage. Targets entering the area of coverage of PTZ_i through the area of coverage of PTZ_j, $j \neq i$, and targets' allowable motion within PTZ_i's area of coverage are included in the local environment assumption φ_{e_i} for $i,j \in \{1,2\}$. Following these simple guidelines, it is possible to derive local specifications of the form $(\varphi_{e_i} \rightarrow \varphi_{s_i})$ that satisfies the assumptions and first two conditions in Proposition 1. Therefore, if Condition 3 in Proposition 1 holds, it is possible to synthesize distributed control protocols which guarantee that the global specification is met.

On the other hand, if Condition 3 in Proposition 1 does not hold, we attempt to refine the local specifications to make them realizable. While deriving the local specifications, we initially assume no collaboration between cameras. However, the boundary (or intersection) of areas of coverage of different PTZ cameras forms a natural interface through which cameras can collaborate. The requirement that PTZ_1 zooms into some of the targets that leave its area of coverage through the area of coverage of PTZ_2, restricts the possible environment behavior that PTZ_2 needs to handle. Let ϕ_1 be such a requirement for PTZ_1 and ϕ_1' be the corresponding assumption on the environment variables of PTZ_2. Assume the local specification $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ for PTZ_1 is realizable and $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ is unrealizable. Now, if the refined local specifications

$$\varphi_{e_1} \rightarrow (\varphi_{s_1} \wedge \phi_1) \tag{2}$$

and

$$(\phi_1' \wedge \varphi_{e_2}) \rightarrow \varphi_{s_2} \tag{3}$$

are realizable, implementing control protocols corresponding to the Eqs. (2)-(3) together would lead to a system where the global specification is met. When $(\varphi_{e_2} \rightarrow \varphi_{s_2})$ is unrealizable, specification (3) could be realizable since the set of executions of the environment for the latter is a subset of that of former. However, if $(\varphi_{e_1} \rightarrow \varphi_{s_1})$ is unrealizable, specification (2) would be unrealizable as well. Hence, when both local specifications are unrealizable initially, one should impose constraints on both sides of the equations simultaneously in the refinement step. That is, one should add formulas ϕ_2, ϕ_2' to tighten the requirement of PTZ_2 and the assumption on the environment of PTZ_1 which gives local specifications of the form

$$\begin{array}{c}(\phi_2' \wedge \varphi_{e_1}) \rightarrow (\varphi_{s_1} \wedge \phi_1) \\ (\phi_1' \wedge \varphi_{e_2}) \rightarrow (\varphi_{s_2} \wedge \phi_2).\end{array} \tag{4}$$

To avoid circular reasoning [18] in (4), we restrict ϕ_i, ϕ_i' to be a safety (i.e., invariance) formulas that use only the temporal operators $\bigcirc$ and $\Box$. Also, the current values of the dependent environment variables which depend on the moves of the controllable variables (e.g. $isZoomed$ in this application) involved in ϕ_i should not depend on the current move of the controllable variables. This second restriction is to ensure the well-posedness of the two-way interaction between the PTZ cameras and related to Moore interfaces [24]. If ϕ_i, ϕ_i' are chosen according to these restrictions and if there exist control protocols that satisfy the refined local specifications in Eqs. (4), then implementing these two control protocols together would result in a system that meets its global specification.

Remark 1: In practice, formulas ϕ_i can be chosen using engineering intuition as demonstrated with an example in Section V. However, it is desirable to be able to systematically choose them. We are currently working on developing a general procedure for this purpose.

V. EXAMPLE

In this section, we illustrate the proposed control protocol synthesis methods on a simple example both in a centralized setup and a distributed setup. First, we state the domains of the variables introduced in Section III for this example; and express the specification for the centralized design using LTL. Then, we derive local specifications and show how they can be refined to reduce the conservatism in distributed synthesis. For synthesizing controllers we used TuLiP [25],

a software package for automatic synthesis of embedded control software, which provides a user-friendly interface to JTLV [12].

Figure 4 shows the layout of the area monitored. The area is partitioned into 12 cells with labels from the set $C = \{c_1, c_2, \ldots, c_{12}\}$. There are two PTZ cameras; one on the left and one on the right side of the area. The area of coverage of the left PTZ is limited to the cells $C_l = \{c_1, \ldots, c_6\}$; and that of the right PTZ is limited to the cells $C_r = \{c_7 \ldots, c_{12}\}$. The field of view of each PTZ covers single cell. We assume that there could be at most 3 targets on each side of the area (a total of $N_p = 6$ targets) at a given time. $x_t^{(i)} \in C \cup \{c_0\}$ is the position of the i^{th} target at time t, where c_0 denotes the outside of the area. Hence the number of targets in the area at time t is the cardinality of the set $\{x_t^{(i)} | x_t^{(i)} \neq c_0, i \in \{1, \ldots, 6\}\}$. We model the permissible target motion with simple rules. At each time step a target either remains in the cell it was in the previous step, or moves to a neighboring cell (transition to diagonal neighbors is not allowed). Also, the targets can exit through designated doors; that is, $\forall i\ x_{t+1}^{(i)} = c_0$ only if $x_t^{(i)} \in \{c_0, c_1, c_3, c_{10}, c_{12}\}$. Similarly, for entering the area the following rule holds: $\forall i, t$ if $x_t^{(i)} = c_0$ then $x_{t+1}^{(i)} \in \{c_0, c_1, c_3, c_{10}, c_{12}\}$. We further assume that people are "uniformly" distributed within the area or not packed in a particular region. More precisely, there can be at most 3 people in C_r and at most 3 people in C_l at a given time. All the rules regarding a target's motion can be collected into an LTL formula of the form

$$\psi_{x,i} \doteq \bigwedge_{k \in \{0,\ldots,12\}} \Box \left((x^{(i)} = c_k) \rightarrow \bigcirc \left(\bigvee_{l \in L_k} (x^{(i)} = c_l) \right) \right) \tag{5}$$

for appropriate choices of k and L_k. In (5) L_k is the set of the cells target is allowed to move from the current cell c_k in one time step; and it is a function of all x's for the targets on the boundary of C_r and C_l to preserve uniformity constraint. In addition, everyone always eventually leaves the area which can be expressed as

$$\psi_{f,i} \doteq \Box\Diamond(x^{(i)} = c_0). \tag{6}$$

We assume that a target remains in the area for at least $T = 3$ time steps

$$\psi_{d,i} \doteq \Box \Big((x^{(i)} \neq c_0 \wedge n^{(i)} < T) \rightarrow \bigcirc(x^{(i)} \neq c_0) \Big) \tag{7}$$

where the variable $n^{(i)}$, counting the number of time steps target i remains inside, satisfies

$$\begin{aligned} \psi_{n,i} \quad \doteq \Box \big[& \big((x^{(i)} = c_0) \rightarrow (n^{(i)} = 0) \big) \wedge \\ & \big(\big(\bigcirc(x^{(i)} \neq c_0)\big) \rightarrow \big(\bigcirc(n^{(i)}) = \min(n^{(i)} + 1, 3)\big) \big) \big] . \end{aligned} \tag{8}$$

In order to take a high resolution image of a target at c_i, the camera should zoom into that single cell c_i. We assume that the camera dynamics are significantly faster than the target dynamics. In particular, for the controllable variables $z_t^{(j)}$ with $j = l$ (or, $j = r$) that denote the cell the left (right) camera points to, we assume $\forall t\ z_t^{(j)} \in C_j$ without any further restriction. The LTL formula for this expression is

$$\psi_{c,j} \doteq \Box \bigvee_{l \in C_j} (z^{(j)} = c_l) \ \text{ for } \ j \in \{l, r\}. \tag{9}$$

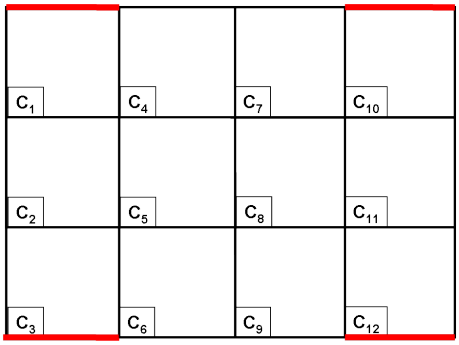

Fig. 4. Layout used for the illustrative example.

The goal is to synthesize a control protocol for the cameras that determines $z_t^{(j)}$ given $z_{0:t-1}^{(j)}$ and $x_{0:t}^{(i)}$ so that it is guaranteed that no target leaves the area before having a high resolution image taken. $isZoomed^{(i)}$ is the boolean variable that indicates whether a high resolution image of the i^{th} target has been taken yet. Let,

$$\begin{aligned} o_{1,i} \doteq \ & \big((x^{(i)} = z^{(l)}) \vee (x^{(i)} = z^{(r)}) \big) \rightarrow \\ & \big(\bigcirc(isZoomed^{(i)}) \big) \\ o_{2,i} \doteq \ & \big(isZoomed^{(i)} \wedge (x^{(i)} \neq c_0) \wedge \bigcirc(x^{(i)} \neq c_0) \big) \rightarrow \\ & \big(\bigcirc(isZoomed^{(i)}) \big) \\ o_{3,i} \doteq \ & \big((x^{(i)} \neq z^{(l)}) \wedge (x^{(i)} \neq z^{(r)}) \wedge \neg isZoomed^{(i)} \big) \rightarrow \\ & \big(\bigcirc(\neg isZoomed^{(i)}) \big) \\ o_{4,i} \doteq \ & \big((x^{(i)} = c_0) \big) \rightarrow \big(\bigcirc(\neg isZoomed^{(i)}) \big) . \end{aligned}$$

The LTL formulas $isZoomed^{(i)}$s follow are:

$$\psi_{o,i} \doteq \bigwedge_{k \in \{1,\ldots,4\}} \Box o_{k,i}. \tag{10}$$

The camera network system is initialized when the area is empty which identifies the initial conditions as

$$\psi_{init,i} \doteq ((x^{(i)} = c_0) \wedge (n^{(i)} = 0) \wedge \neg isZoomed^{(i)}). \tag{11}$$

Let $escape_i$ be a proposition, which indicates that target i leaves the area without having a high resolution image taken, defined as follows:

$$\begin{aligned} escape_i \doteq \ & \big((x^{(i)} \neq c_0) \wedge \neg isZoomed^{(i)} \wedge \bigcirc(x^{(i)} = c_0) \wedge \\ & \bigcirc(\neg isZoomed^{(i)}) \big) . \end{aligned}$$

Then, the goal of the discrete controller can be written as:

$$\psi_g \doteq \Box \left(\bigwedge_{i \in \{1,\ldots,6\}} \neg escape_i \right) . \tag{12}$$

Collecting all environmental assumptions into a single formula, we obtain:

$$\varphi_e \doteq \bigwedge_{i \in \{1,\ldots,6\}} (\psi_{x,i} \wedge \psi_{f,i} \wedge \psi_{d,i} \wedge \psi_{n,i} \wedge \psi_{o,i} \wedge \psi_{init,i}). \tag{13}$$

Similarly, the requirement is:

$$\varphi_s \doteq \psi_g \wedge \psi_{c,l} \wedge \psi_{c,r}. \tag{14}$$

By using TuLiP [25], it is possible to find a centralized control protocol that satisfies the specification

$$\varphi \doteq \varphi_e \rightarrow \varphi_s. \tag{15}$$

Such a control protocol can be implemented using a centralized architecture as in Fig. 2.

A. Distributed Synthesis

We present a decomposition of the specification for distributed synthesis for the example above. The constraints on the areas of coverage of the cameras and the assumption that people are not dense either on the left or right side of the area enables defining local specifications of the form $\varphi_e^j \rightarrow \varphi_s^j$, $j \in \{l, r\}$. Each PTZ camera has partial track information that includes only the locations of the targets within their own area of coverage. Additionally, the PTZs have some limited communication capability through which they can exchange n and $isZoomed$ values of the targets that cross from one side of the area to the other. We derive the local specification that is used for synthesizing the control protocol for the left PZT camera (i.e., $j = l$); the control protocol for the right camera is the same due to symmetry.

Since there could be at most 3 people in cells C_l, we consider $x^{(i)} \in C_l \cup \{c_0\}$ for $i \in \{1, 2, 3\}$. Unlike the centralized design, the right side of the area is also represented by c_0. The rules for the target motion is the same when $x_t^i = c_k$ for $k \in \{1, 2, 3\}$. Since the left PTZ camera does not have the information about the targets on the right, we need to modify the assumptions on the target motion to account for people passing from one side to the other. We assume that a target in c_4, c_5 or c_6 can freely pass to the right side. That is, $L_4 = \{0, 1, 4, 5\}$, $L_5 = \{0, 2, 4, 5, 6\}$ and $L_6 = \{0, 3, 5, 6\}$ where L_k is defined similarly as in Eq. (5). Also, whenever $x^{(i)} = c_0$ for some i, someone can enter from the right side of the area as well as the entrances; that is $L_0 = \{0, 1, 3, 4, 5, 6\}$. Finally we obtain the following target motion model:

$$\psi_{x,i}^l \doteq \bigwedge_{k \in \{0,\ldots,6\}} \square \left((x^{(i)} = c_k) \rightarrow \bigcirc \left(\bigvee_{l \in L_k} (x^{(i)} = c_l) \right) \right). \tag{16}$$

We replace the condition that every target always eventually leaves the area with

$$\psi_{f,i}^l \doteq \square\lozenge(x^{(i)} = c_0), \tag{17}$$

which means every target always eventually leaves the left side of the area. When combined with the symmetric condition $\psi_{f,i}^r$ for the right side of the area, a target who keeps passing from the left to the right and vice versa infinitely often will satisfy the environment assumption although such a behavior is not allowed by the assumption (6). However since any behavior compatible with (6) is also compatible with $\psi_{f,i}^l \wedge \psi_{f,i}^r$, the condition 1 in Proposition 1 holds.

We assume a target should remain in the area for at least $T = 3$ time steps. However, while defining the rules related to $n^{(i)}$, we need to make a distinction between outside of the area and the right side of the area. Taking into account the facts that a target on the left side of the area can only get out of the area from c_1 or c_3 and someone crossing from right to left might have already stayed in the area for a while, we have:

$$\begin{aligned} \psi_{d,i}^l \doteq \; & \square \big(((x^{(i)} = c_1 \vee x^{(i)} = c_3) \wedge (n^{(i)} < T)) \rightarrow \\ & \bigcirc (x^{(i)} \neq c_0) \big) \end{aligned} \tag{18}$$

and

$$\begin{aligned} \psi_{n,i}^l \doteq \; & \square \big[\big((x^{(i)} = c_0) \rightarrow (n^{(i)} = 0) \big) \wedge \\ & \big(((x^{(i)} \neq c_0) \wedge \bigcirc (x^{(i)} \neq c_0)) \rightarrow \\ & \quad \big(\bigcirc (n^{(i)}) = \min(n^{(i)} + 1, 3) \big) \big) \wedge \\ & \big(\big((x^{(i)} = c_0) \wedge \bigcirc (x^{(i)} = c_1 \vee x^{(i)} = c_3) \big) \rightarrow \\ & \quad \big(\bigcirc (n^{(i)}) = 1 \big) \big) \big]. \end{aligned} \tag{19}$$

Camera dynamics remain the same; that is $\psi_c^l \doteq \psi_{c,l}$. As for $isZoomed$, we have

$$\begin{aligned} o_{1,i}^l \doteq \; & (x^{(i)} = z^{(l)}) \rightarrow \big(\bigcirc (isZoomed^{(i)}) \big) \\ o_{2,i}^l \doteq \; & \big(isZoomed^{(i)} \wedge (x^{(i)} \neq c_0) \wedge \bigcirc (x^{(i)} \neq c_0) \big) \rightarrow \\ & \big(\bigcirc (isZoomed^{(i)}) \big) \\ o_{3,i}^l \doteq \; & \big((x^{(i)} \neq z^{(l)}) \wedge \neg isZoomed^{(i)} \big) \rightarrow \\ & \big(\bigcirc (\neg isZoomed^{(i)}) \big) \\ o_{4,i}^l \doteq \; & \big((x^{(i)} = c_0) \wedge (\bigcirc (x^{(i)} \neq c_4 \wedge x^{(i)} \neq c_5 \wedge \\ & x^{(i)} \neq c_6)) \big) \rightarrow \big(\bigcirc (\neg isZoomed^{(i)}) \big) \end{aligned}$$

which lead to the formula

$$\psi_{o,i}^l \doteq \bigwedge_{k \in \{1,\ldots,4\}} \square o_{k,i}^l. \tag{20}$$

Initial conditions are the same as before (i.e., $\psi_{init,i}^l \doteq \psi_{init,i}$). We need to modify the proposition $escape_i$ as follows:

$$\begin{aligned} escape_i^l \doteq \; & \big(((x^{(i)} = c_1) \vee (x^{(i)} = c_3)) \wedge \neg isZoomed^{(i)} \wedge \\ & \bigcirc (x^{(i)} = c_0) \wedge \bigcirc (\neg isZoomed^{(i)}) \big). \end{aligned}$$

The goal of the planner on the left PTZ is

$$\psi_g^l \doteq \square \left(\bigwedge_{i \in \{1,2,3\}} \neg escape_i^l \right). \tag{21}$$

Collecting all environmental assumptions into a single formula, we obtain:

$$\varphi_e^l \doteq \bigwedge_{i \in \{1,2,3\}} (\psi_{x,i}^l \wedge \psi_{f,i}^l \wedge \psi_{d,i}^l \wedge \psi_{n,i}^l \wedge \psi_{o,i}^l \wedge \psi_{init,i}^l). \tag{22}$$

Similarly, the requirement is:

$$\varphi_s^l \doteq \psi_g^l \wedge \psi_c^l. \tag{23}$$

From Proposition 1, it follows that if there exist two local control protocols satisfying $(\varphi_e^l \rightarrow \varphi_s^l)$ and $(\varphi_e^r \rightarrow \varphi_s^r)$; then implementing these control protocols on local planners will guarantee that the global specification in (15) is met. However, using TuLiP we determine that (23) is unrealizable which is a certificate that there exists no control protocol that guarantees (23) (nor the symmetric counterpart for the right PTZ). The specification (23) is not realizable mainly because there is no collaboration on the boundary of the two regions.

Each camera tries to ensure none of the targets gets out of the area without being zoomed. Yet, they do not guarantee any exposure of the targets leaving their area of coverage to cross to the other camera's area of coverage. Hence, when multiple un-zoomed targets cross to the left of the area from the right, the left PTZ can not guarantee that they will all be zoomed in before they leave the area.

If the cameras cooperate by restricting the number of un-zoomed targets that cross from one side to the other, it might be possible to achieve realizability. Strengthening the specification for the right camera by including additional requirements, that ensure some of the targets passing from right to left to be zoomed-in, restricts the allowable behavior of the environment for the left camera. Therefore, both sides of (23) should be refined simultaneously due to the symmetry of the system considered in this example. We refine φ_e^l by assuming that two un-zoomed targets do not cross from right side of the area to the left at the same time. Let in be a proposition defined as

$$\begin{aligned} in_{i,j}^l \doteq \quad & (x^{(i)} = c_0) \wedge \Big(\bigvee_{k\in\{4,5,6\}} \bigcirc(x^{(i)} = c_k)\Big) \wedge \\ & \bigcirc(\neg isZoomed^{(i)}) \wedge (x^{(j)} = c_0) \wedge \\ & \Big(\bigvee_{k\in\{4,5,6\}} \bigcirc(x^{(j)} = c_k)\Big) \wedge \bigcirc(\neg isZoomed^{(j)}), \end{aligned}$$

then, the additional assumption on the environment is

$$\varphi_{e,refine}^l \doteq \Box \left(\bigwedge_{i,j\in\{1,2,3\}, i\neq j} \neg in_{i,j}^l \right). \tag{24}$$

Similarly, let out be defined as

$$\begin{aligned} out_{i,j}^l \doteq \quad & \Big(\bigvee_{k\in\{4,5,6\}} (x^{(i)} = c_k)\Big) \wedge \bigcirc(x^{(i)} = c_0) \wedge \\ & \bigcirc(\neg isZoomed^{(i)}) \wedge \Big(\bigvee_{k\in\{4,5,6\}} (x^{(j)} = c_k)\Big) \wedge \\ & \bigcirc(x^{(j)} = c_0) \wedge \bigcirc(\neg isZoomed^{(j)}). \end{aligned}$$

The additional requirement that represents the cooperative effort of left PZT at the boundary is

$$\varphi_{s,refine}^l \doteq \Box \left(\bigwedge_{i,j\in\{1,2,3\}, i\neq j} \neg out_{i,j}^l \right). \tag{25}$$

Finally we obtain the refined specification:

$$\varphi_{refined}^l \doteq \big((\varphi_{e,refine}^l \wedge \varphi_e^l) \rightarrow (\varphi_s^l \wedge \varphi_{s,refine}^l)\big). \tag{26}$$

We used TuLiP [25] to verify the realizability of this specification and to synthesize local control protocols. Implementing these control protocols on local planners of PTZs guarantees that the global specification φ in (15) is met. A simulation for this example is shown in Figure 5 where each PTZ camera moves according to its local control protocol. As seen in the figure, no one leaves the area before being zoomed in.

VI. CONCLUSIONS AND FUTURE WORK

A. Conclusions

In this paper, we considered the problem of designing control protocols for PTZ cameras within a smart camera network where the goal is to guarantee certain temporal logic specifications related to a given surveillance task. We recast this problem into a two-player game between the targets and the PTZ cameras. We employed the digital design synthesis method of Piterman et al. [12] to synthesize control protocols. However, this method does not scale well with increasing number of variables due to the state explosion problem. To partially alleviate this problem, we proposed a distributed synthesis procedure which is based on decomposing the global specification into local ones so that it is possible to implement local controllers on each PTZ. We also presented some preliminary ideas as to how the local specifications can be refined in order to reduce conservatism by imposing cooperation between the PTZ cameras.

B. Discussions and Future Directions

The proposed distributed control protocol synthesis methodology is restrictive and should be considered as an initial step. We now discuss these specific restrictions and potential extensions. If the local specifications are unrealizable it could be either because of the conservatism of Proposition 1 or because the global specification is unrealizable. Obviously, refining the local specifications would be pointless if the global specification is unrealizable. One of the drawbacks of our approach is its lack of providing any insight about this. An interesting research direction is to use the counterexamples for local specifications to search for unrealizability certificates for the global specification. Also, a better characterization of the LTL formulas that can be decomposed and LTL formulas that can be used in the refinement procedure is necessary. Another direction for current research is to automate the refinement procedure, for instance using counterexample guided approaches (e.g., [26]) or random formula generation.

For the PTZ cameras, we considered a cyclic topology, but it is possible to consider different network topologies as well. For instance in a traffic monitoring application a serial topology along the traffic flow would be more appropriate. It is worth studying how different interconnections/interactions of the subsystems affect the design. In this paper, we fixed the communication rules between the cameras a priori. Designing communication rules subject to some communication constraints within the synthesis procedure is a worthwhile endeavor. Finally, we are exploring applications of our distributed synthesis methodology in different control protocol design problems, such as those arising in vehicle management systems or autonomous robotic teams.

REFERENCES

[1] P. Remagnino, C. S. Regazzoni, G. A. Jones, and N. Paragios, eds., *Video-Based Surveillance Systems: Computer Vision and Distributed Processing*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[2] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2002.

[3] B. Rinner and W. Wolf, "An introduction to distributed smart cameras," *Proceedings of the IEEE*, vol. 96, pp. 1565–1575, October 2008.

[4] S. Soatto, "Actionable information in vision," in *Proceedings of the International Conference on Computer Vision*, October 2009.

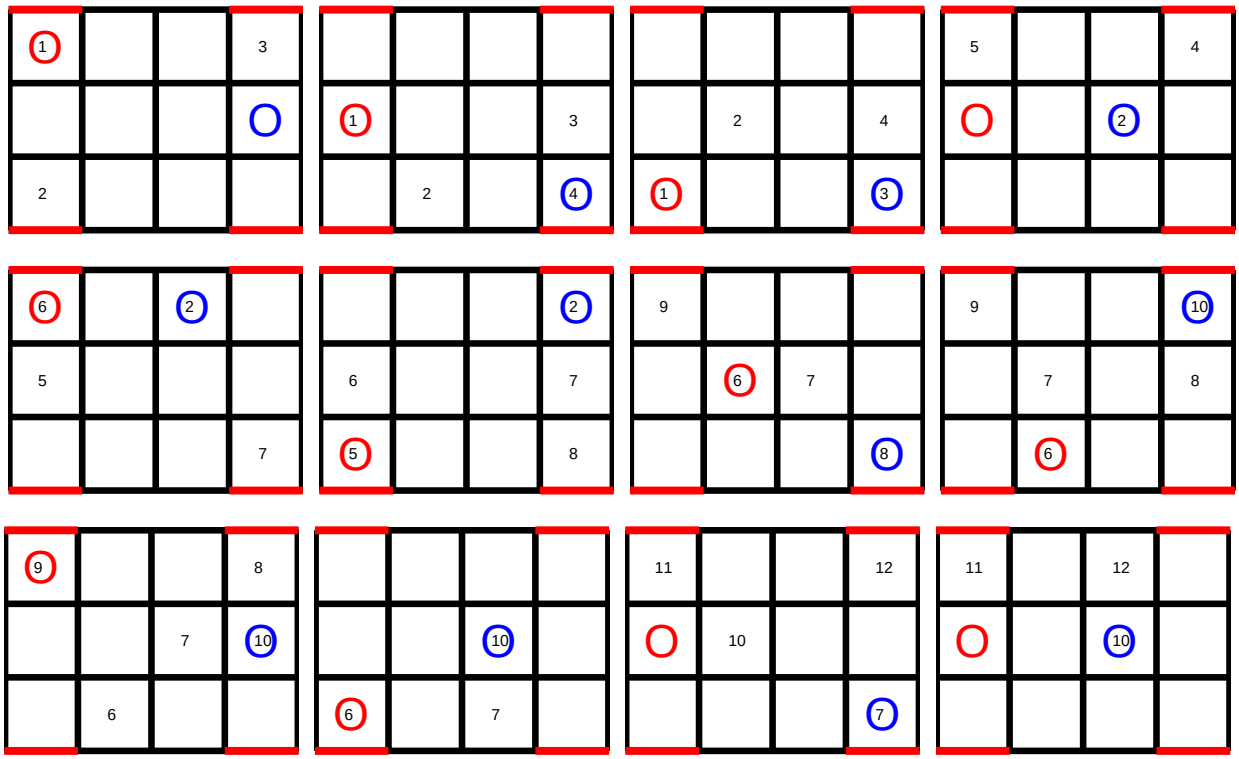

Fig. 5. Results for a sample run of the system. Numbers denote the target identities. Red (blue) circles indicate the cells left (right) PTZ camera zoom in according to the distributed control protocol. Snapshots from first twelve time steps are shown ordered from left to right starting on the first row.

[5] C. Micheloni, G. Foresti, and L. Snidaro, "A network of co-operative cameras for visual surveillance," *Vision, Image and Signal Processing, IEE Proceedings* -, vol. 152, pp. 205 – 212, apr. 2005.

[6] R. Alur, T. A. Henzinger, G. Lafferriere, and G. J. Pappas, "Discrete abstractions of hybrid systems," in *Proceedings of the IEEE*, pp. 971–984, 2000.

[7] P. Tabuada and G. J. Pappas, "Linear time logic control of linear systems," *IEEE Transactions on Automatic Control*, vol. 51, no. 12, pp. 1862–1877, 2006.

[8] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning," *IEEE Transactions on Automatic Control*, 2010. submitted.

[9] A. Pnueli, "The temporal logic of programs," in *Proc. of the 18th Annual Symposium on the Foundations of Computer Science*, pp. 46–57, IEEE, 1977.

[10] Z. Manna and A. Pnueli, *The temporal logic of reactive and concurrent systems*. Springer-Verlag, 1992.

[11] E. Asarin, O. Maler, A. Pnueli, and J. Sifakis, "Controller synthesis for timed automata," in *IFAC Symposium on System Structure and Control*, pp. 469–474, 1998.

[12] N. Piterman, A. Pnueli, and Y. Sa'ar, "Synthesis of reactive(1) designs," in *Verification, Model Checking and Abstract Interpretation*, vol. 3855 of *Lecture Notes in Computer Science*, pp. 364 – 380, Springer-Verlag, 2006. Software available at http://jtlv.sourceforge.net/.

[13] A. Pneuli and R. Rosner, "Distributed reactive systems are hard to synthesize," in *SFCS '90: Proceedings of the 31st Annual Symposium on Foundations of Computer Science*, (Washington, DC, USA), pp. 746–757 vol.2, IEEE Computer Society, 1990.

[14] M. Mukund, "From global specifications to distributed implementations," in *Synthesis and control of discrete event systems*, pp. 19–34, Kluwer, 2002.

[15] E. Filiot, N. Jin, and J.-F. Raskin, "Compositional algorithms for ltl synthesis," in *Automated Technology for Verification and Analysis*, pp. 112–127, 2010.

[16] P. Madhusudan and P. Thiagarajan, "Distributed controller synthesis for local specifications," in *Automata, Languages and Programming* (F. Orejas, P. Spirakis, and J. van Leeuwen, eds.), vol. 2076 of *Lecture Notes in Computer Science*, pp. 396–407, Springer Berlin, 2001.

[17] C. Baier and J.-P. Katoen, *Principles of Model Checking*. MIT Press, 2008.

[18] E. M. Clarke, O. Grumberg, and D. A. Peled, *Model Checking*. MIT Press, 1999.

[19] A. Pnueli, "Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends," *Current Trends in Concurrency. Overviews and Tutorials*, pp. 510–584, 1986.

[20] A. Galton, ed., *Temporal Logics and Their Applications*. San Diego, CA: Academic Press Professional, Inc., 1987.

[21] G. Holzmann, "The theory and practice of a formal method: NewCoRe," in *Proc. of the IFIP World Computer Congress*, vol. 1, pp. 35–44, North-Holland Publ., 1994.

[22] S. Cerrito and M. C. Mayer, "Using linear temporal logic to model and solve planning problems," in *AIMSA*, pp. 141–152, 1998.

[23] M. Kloetzer and C. Belta, "Automatic deployment of distributed teams of robots from temporal logic motion specifications," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 48–61, 2010.

[24] S. Tripakis, B. Lickly, T. A. Henzinger, and E. A. Lee, "On relational interfaces," in *EMSOFT*, pp. 67–76, 2009.

[25] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "TuLiP: a software toolbox for receding horizon temporal logic planning," in *HSCC*, 2011. submitted. Software available at http://www.cds.caltech.edu/tulip.

[26] E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Counterexample-guided abstraction refinement," in *Computer Aided Verification* (E. Emerson and A. Sistla, eds.), vol. 1855 of *Lecture Notes in Computer Science*, pp. 154–169, Springer Berlin, 2000.