Controlling Wild Bodies Using Discrete Transition Systems

Leonardo Bobadilla Oscar Sanchez Justin Czarnowski
bobadil1@uiuc.edu sanche14@uiuc.edu jczarno2@uiuc.edu
Katrina Gossman Steven M. LaValle
kgossma2@uiuc.edu lavalle@uiuc.edu
Department of Computer Science
University of Illinois
Urbana, IL 61801 USA

***Abstract*—This paper proposes methods for achieving basic tasks such as navigation, patrolling, herding, and coverage by exploiting the wild motions of very simple bodies in the environment. Bodies move within regions that are connected by gates that enforce specific rules of passage. This leads to a hybrid systems approach in which the behaviors define a discrete transition system. Tasks can even be specified using a Linear Temporal Logic (LTL) formula and are converted into a multibody implementation that satisfies the formula. Common issues such as dynamical system modeling, precise state estimation, and state feedback are avoided. The method is demonstrated in a series of experiments that manipulate the flow of weasel balls (without the weasels) and Hexbug Nano vibrating bugs.**

I. Introduction

In everyday life we see many examples of independently moving bodies that are gracefully corralled into behaving in a prescribed way. For example, when the free breakfast area closes in a hotel, the manager usually locks the door from the outside so that no one else can enter, but people eating are able to finish their meals and leave. This has the effect of clearing everyone from the room without people feeling that they have been tightly controlled or coerced. People install a "doggie door" on their house door to enable pets to move in either one direction or both. In a subway system, turnstiles cause people to flow in prescribed directions to ensure that proper fares are paid.

These scenarios, and many others, involve numerous bodies moving together in one environment with two important principles:

1) Each body moves independently, possibly with a "mind of its own", in a way that seems to exhaustively explore its environment.
2) The bodies are effectively controlled without precisely measuring their state and without forcefully actuating them.

On the other hand, robots are typically controlled in the opposite way. There is a large modeling burden, which includes system identification (learning the equations of motion) and constructing a map of the environment. Powerful sensors are used for mapping and localization of the robots. Filters are developed that provide state estimates at all times so that plans or policies can make actions depend on state feedback. In the case of multi-robot coordination, further complications arise. Difficult coordination strategies may be required. Furthermore, careful communication between robots and possibly a central controller is often necessary. For some tasks, we wonder whether all of these issues can be avoided altogether.

(a) (b)

Fig. 1. a) One vehicle of study is a $4 weasel ball; b) it consists entirely of a battery and slowly oscillating motor mounted to a plastic shell.

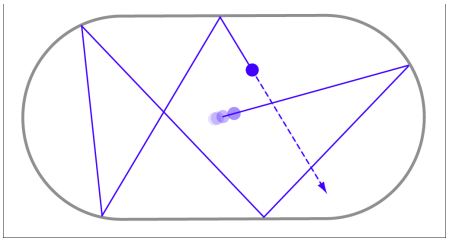


Fig. 2. The Bunimovich stadium is a well-known example of an ergodic system [7]. The "hockey puck" will strike every open set along the boundary as it travels forever. (Figure courtesy of Wikipedia.)

By taking inspiration from the everyday life principles above, we propose an unusual paradigm to control multiple robots. In contrast to most approaches, we start with a "wildly behaving" body for which its precise equations of motion are unknown; it is far from stable, and has little or no sensing capabilities. Our main "vehicle" of study is a $4 weasel ball (see Figure 1), which has no sensors, no computation, and one

motor, which oscillates constantly at about 2Hz.

In our experiments, the particular choice of body is not critical (Section V-A will show other systems). We instead care only about its high-level motion properties. We informally consider a body to be *wild* if when placed into a bounded region $r \subset \mathbb{R}^2$, it moves along a trajectory that strikes every open interval along the boundary of r infinitely often. This concept is closely related to the notion of *topological transitivity* in the branch of dynamical systems known as *dynamical billiards* [44]. An example is shown in Figure 2, which can be imagined as a billiard ball that bounces off of the table sides forever. A strong system property that arises in that work and achieves our required wild behavior is *ergodicity*.[1] It is even known that in the space of all simple (including nonconvex) polygons and all initial configurations for the body, the resulting trajectory is ergodic except for cases that lie in a set of measure zero [24], [25], [28]. The idea of exploiting wild motions in robotics is reminiscent of the randomization work by Erdmann [14] and designing robot systems with ergodic dynamics by Shell et al. [42].

How do we control such systems? We are first inspired by the power of abstraction used in hybrid systems [6], [12], [19], [21], [26], [46]. As is common in many approaches, we partition the state space into a finite set of regions over which a discrete transition system is defined. Whereas it is common in hybrid system approaches to derive state-feedback control laws while vehicles are within continuous regions [12], [20], [30], [36], [43], we simply let our "vehicle" behave wildly.

One unusual aspect of our approach is that we embed simple mechanisms in the environment that force the bodies to achieve goals while remaining wild. To control each body, we design *gates* that appear only along region boundaries and connect to other regions. When a body strikes a gate, the gate will induce our planned behavior, which might be to remain in the region or transition to another region. In this sense, the gate "gently guides" the body. The gates themselves have configurations that determine what type of passage is allowed between adjacent regions. The gates can be fixed in advance (*static gates*), can have actuators that change configurations (*controllable gates*), or can have their configurations changed mechanically by absorbing energy from the bodies (*pliant gates*). This way of controlling bodies leads to many interesting open questions regarding the space of tasks that can be solved and the overall system complexity required to solve them.

Our approach draws inspiration from several areas, including *nonprehensile manipulation* [15], [23], [40], vibrating plates [5], [41], [47], and billiard models of quantum computing [22]. Even more closely related are designing *virtual fences* to control herds of cows [8] and designing fire evacuation strategies to safely "herd" humans out of a burning building [9]. We are also inspired by the family of work that converts high-level specifications into low-level control laws for the hybrid system [29], [39], [45]. In particular, much of our work uses the Linear Temporal Logic (LTL) framework that has been developed in several recent works [1], [16], [17], [20], [30], [31], [32], [33], [34], [35], [36], [43], [48]. The idea is to express a complicated task using a logical formula and then converting the specification into a control law that satisfies the formula, thereby accomplishing the task.

The paper is organized as follows. Section II presents some preliminary concepts, including the interaction between the wild body, the gates, and the regions. Sections III and IV present our approach for the cases of a single and multiple bodies, respectively. Section V presents experiments and Section VI concludes the paper with some promising directions for future research. This paper is built upon two earlier conference publications [2], [4].

[1]In this context, ergodicity does not necessarily have anything to do with probabilities, as in the more commonly seen case of ergodicity in Markov chains.

II. The Overall Design

A. Connectivity between regions and gates

Consider a planar workspace $E \subseteq \mathbb{R}^2$ that is partitioned into an obstacle set O and a finite set of bounded cells with connected open interior, each of which is either a *region* or a *gate*; Figure 3 shows a simple example. The following conditions are imposed: 1) No region shares a boundary with any other region; 2) no gate shares a boundary with any other gate; 3) every region shares a boundary with at least one gate; 4) if a gate and a region share a boundary, then the boundary is a connected interval (rather than being a point or being disconnected). Let R denote the set of all regions and G denote the set of all gates. The union of all $r \in R$, all $g \in G$, and O yields E.

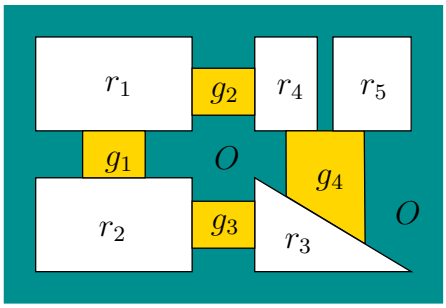


Fig. 3. An example arrangement of five regions and four gates.

B. Wild bodies

We now place a *body* b into the workspace. The body is assumed to be "small" with respect to the sizes of regions, gates, and their shared boundaries. It is therefore modeled geometrically as a point even though it may have complicated shape, kinematics, and dynamics. We assume that the body moves in a wild, uncontrollable way, but the trajectory satisfies the following high-level property:

> *For any region $r \in R$, it is assumed that b moves on a trajectory that causes it to strike every open*

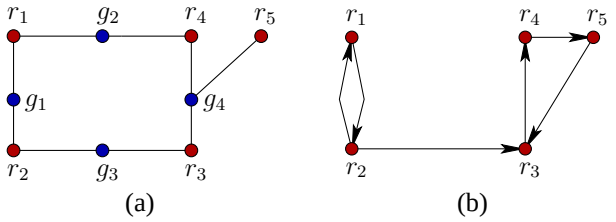


Fig. 4. a) A bipartite graph representation of the arrangement of regions and gates from Figure 3. b) A flow graph that corresponds to one particular composite mode. Each gate configuration allows alternative possible flow directions between every pair of regions that are adjacent to the gate: g_1 allows bidirectional flow; g_2 allows no flow; g_3 allows flow from left to right; g_4 allows clockwise flow among r_3, r_4, and r_5.

interval in ∂r (the boundary of r) infinitely often, with non-zero, non-tangential velocities.

A body that satisfies this property is called *wild*. We can now imagine that a wild body travels on a path through the bipartite graph shown in Figure 4(a), with transitions occurring only if specific gates allow it, which is the next topic.

C. Gate configurations and flow graphs

Every gate $g \in G$ has an associated finite set $C(g)$ of *local configurations* that determine the flow of bodies between adjacent regions. Let C denote the *global configuration space*, which is defined as the k-fold Cartesian product of $C(g)$ over every $g \in G$ and $k = |G|$.

Let $R(g)$ denote the regions adjacent to a gate $g \in G$. For the example in Figure 3, $R(g_4) = \{r_3, r_4, r_5\}$. For each region pair $r, r' \in R(g)$, the local configuration $c \in C(g)$ could allow one of four body flows:

1) Allow bidirectional passage between r' and r.
2) Allow passage only from r to r'.
3) Allow passage only from r' to r.
4) Block all passage between r and r'.

For each global configuration $(c_1, \ldots, c_k) \in C$, a *flow graph* $F(c)$ is a directed graph that is defined as follows. The set of vertices is R. A directed edge in $F(c)$ exists from r to r' if and only if there exists some gate $g_i \in G$ with $r, r' \in R(g)$ and g allows passage from r to r' while in global configuration $(c_1, \ldots, c_k)$. Note that a fixed global configuration $(c_1, \ldots, c_k)$ fixes a particular local configuration in $c_i \in C(g)$. Figure 4(b) shows an example of a possible flow graph for the regions and gates of Figure 3.

D. Static and dynamic gates

The "control" in our system occurs by designing the behavior of gates. The simplest case is a *static gate*, which means that $|C(g)| = 1$, thereby fixing the flow between adjacent regions. For example, if $R(g) = \{r, r'\}$, then the gate g might permanently allow bodies to flow from r to r', but not from r' to r. Otherwise, we obtain a *dynamic gate*, for which $|C(g)| > 1$. See Section V-B for examples of implemented gates.

Once dynamic gates are present, an interesting question arises: What information is available to use a feedback in prescribing the configuration? This is a standard control issue. For our purposes, the complete information case will correspond to each gate having at its disposal the local configuration of all gates and the region that contains the body (or the regions corresponding to all bodies, if there are multiple bodies). In general, however, our systems will operate with less than complete information by using whatever sensor observations and information states are available. An interesting problem is determine the minimal amount of sensing and filtering needed to solve a specified task in our framework.

III. Controlling One Wild Body

This section presents a method for designing gates that effectively control a wild body that must visit regions in a prescribed way.

A. Specifying tasks in LTL

We want to specify tasks in some high-level way, possibly starting from structured English or some simple logic. We chose Linear Temporal Logic (LTL) due to its increasing popularity and available toolkits; see [13]. The syntax includes a set Π of propositions, propositional logic symbols, and some temporal operators. Formulas ϕ are constructed from atoms $\pi \in \Pi$ using the grammar [11]:

$$\phi ::= \pi \mid \neg\phi \mid (\phi \vee \phi') \mid \bigcirc \phi \mid \phi \mathcal{U} \phi',$$

in which $\mathcal{U}$ and $\bigcirc$ are temporal operators meaning *until* and *next*, respectively. A formula is considered *true* or is said to *hold* based on the truth values of the propositions at each state and time and the semantic rules of the various logic symbols and operators. The full specification of LTL semantics is not included here. For $\phi \mathcal{U} \phi'$ to hold, it means that there is a state at which ϕ' holds and ϕ holds at every state before it. For $\bigcirc\phi$ to hold, it means that ϕ holds at the next state. Other operators and logic symbols can be derived from the grammar: *conjunction* ($\wedge$), *implication* ($\Rightarrow$), *equivalence* ($\Leftrightarrow$), *eventually* ($\Diamond$), and *always* ($\Box$).

We want to express tasks in terms of the regions that are visited by the body. Therefore, let $\Pi = \{\pi_1, \pi_2, ..., \pi_n\}$ be a set of Boolean propositions for which π_i is true if and only if the body is in $r_i \in R$. Examples of task specifications are [34]:

- Navigation: $\Diamond \pi_1$
- Sequencing: $\Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond(\pi_3 \wedge \cdots \Diamond \pi_k) \cdots)$
- Coverage: $\Diamond \pi_1 \wedge \Diamond \pi_2 \wedge \cdots \Diamond \pi_k$
- Avoiding regions: $\neg(\pi_1 \vee \pi_2 \cdots \vee \pi_k) \mathcal{U} \pi_{final}$
- Patrolling: $\Box(\Diamond \pi_1 \wedge \Diamond \pi_2 \wedge \ldots \Diamond \pi_k)$.

B. Discrete abstraction of motion

We now define a discrete transition system

$$S_1 = (R, r_0, \rightarrow_1), \tag{1}$$

in which r_0 is the initial region and R is the set of system states. A discrete transition system can be thought of as a

directed graph in which paths or walks are possible system trajectories. The set of vertices is R and the an edge from r to r' exists if the transition relation $r \rightarrow_1 r'$ is true. There is an edge from r to r' if and only if there exists a gate some gate $g_i \in G$ with $r, r' \in R(g)$. You can think of S_1 as representing a maximal flow graph in the sense that it includes as many directed edges as possible, based only on regions that are adjacent through some gate. The approach, which is discussed next, is to determine which particular transitions are needed to yield a region sequence that satisfies a desired LTL formula ϕ.

C. The method

Suppose that an environment contains a set R of regions and that an LTL formula ϕ that expresses the task in terms of the region-based propositions in II. The approach is summarized as follows:

1) Design a body that is wild with respect to every region in R.
2) Start the system with the body in r_0.
3) Apply a standard model-checking algorithm to ϕ to determine a (possibly infinite) region sequence $\tilde{r} = (r_0, r_1, \ldots)$ that satisfies ϕ.
4) Ensure that each transition from r_i to r_{i+1} occurs by setting the global gate configuration appropriately. The resulting execution satisfies ϕ.

In the third step, widely available model checking software, such as NuSMV [10] or SPIN [27], can be used to produce $\tilde{r}$.

What gate designs are needed to ensure that $\tilde{r}$ is executed? Suppose that a transition from r_i to r_{i+1} must occur and g is the gate through which they are adjacent. If it is known that the body has arrived in r_i, then the local gate configuration $c \in C(g)$ should be set so that the flow graph contains an edge from r_i to r_{i+1} and there is no edge from r_i to any other region.

Depending on $\tilde{r}$, much less information may be needed during execution to determine the global gate configuration. Suppose that in $\tilde{r}$ there does not exist any $i < j$ for which $r_i = r_j$ and $r_{i+1} \neq r_{j+1}$. In other words, if the body is in $r_i = r_j$, then the next required region is unique and does not depend on the particular position in $\tilde{r}$. Call this the *uniqueness condition*. In this case, static gates are sufficient for forcing the body follow the region sequence $\tilde{r}$ and satisfy ϕ.

D. A simple example

This environment is used for some experiments in Section V. In Figure 5, there are five regions $R = \{r_0, r_1, r_2, r_3, r_4\}$ and five gates $G = \{a, b, c, d, e, f\}$, shown in blue. Suppose that the LTL formula is

$$\phi = \Diamond(\pi_2 \wedge \Diamond(\pi_1 \wedge \Diamond(\pi_0 \wedge \Diamond\pi_4))). \quad (2)$$

After running NuSMV on the system S_1 and ϕ, the sequence $\tilde{r} = (r_2, r_1, r_0, r_4)$ is returned. Since this satisfies the uniqueness condition, an implementation with static gates is sufficient:

1) Set gate c to allow passage from r_2 to r_1.
2) Set gate b to allow passage from r_1 to r_0.
3) Set gate a to allow passage from r_0 to r_4.

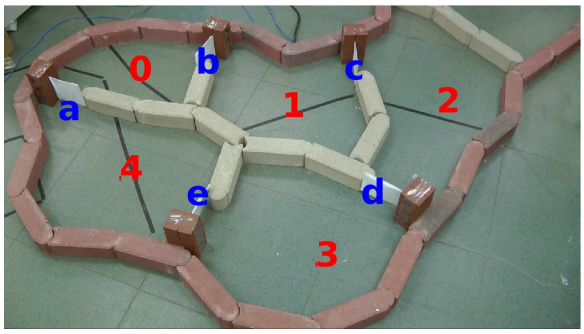


Fig. 5. An example of an arrangement of five regions and five gates.

More complicated examples, which require sensing and the greater expressive power of LTL are given in Section V.

IV. Controlling Multiple Wild Bodies

Now consider extending the ideas of Section III to control multiple bodies. The bodies are not able to communicate or coordinate with each other. However, they are allowed to collide with each other. Each body is assumed to be wild in each region, in spite of these collisions. We have observed in experiments that interference with other bodies does not prevent them from contacting the boundary and becoming wild; however, in theory this depends on the particular mechanics of the body.

A. Discrete abstraction for multiple bodies

One of the main issues with multiple bodies is distinguishability. To model various cases, a collection $B = \{b_1, \ldots, b_k\}$ of k bodies can be assigned *labels* by a function $\lambda : B \rightarrow L$. For example, L could be a set of colors, $L = \{blue, red, yellow, green\}$.

At one extreme, the bodies could be completely distinguishable. In this case, λ is one-to-one, yielding a unique label for every body. Each body can then be controlled independently by using the concepts from Section III. An LTL formula ϕ_i is defined for each $b_i \in B$. This yields a region sequence $\tilde{r}_i$ that satisfies ϕ_i. Imagine the execution. At any time, there are k flow graphs, one for each b_i. It is possible to design discriminating gates which allow only b_i to pass. If each body has its own associated gates and will be blocked by all others, then every b_i is handled independently by its own gates. If the gates are shared between bodies, however, then conflict occurs in cases in which bodies b and b' are in some region r: b must transition to r', and b' must transition to another region. The gate between r and r' must allow b to pass, but block b'. The gate's local configuration space could be designed to generate this behavior; however, the implementation may be difficult.

At the other extreme, the bodies could be completely indistinguishable. In this case, λ assigns the same label to all bodies and there is no reason for the gates to distinguish between them. Tasks are then described in terms of the *number*

of bodies in each region. Let a *distribution* d of bodies be a k-dimensional vector $d = (c_1, \ldots, c_n)$, in which each component c_i is a nonnegative integer representing the count (number of bodies) for each region. For k bodies and n regions, note that $c_1 + \cdots + c_n = k$. Let D_k be the set of all possible distributions for a given k (n is assumed to be fixed well in advance). The size of D_k is $\binom{n+k-1}{k}$, which from combinatorics is the number of ways to place k balls (bodies) into n boxes or urns (regions).

Once the graph of regions and gates is defined, a discrete transition system of the form

$$S_k = (D_k, d_0, \rightarrow_k). \tag{3}$$

naturally captures the possible transitions between distributions of k bodies.

The relation $d \rightarrow_k d'$ is true if and only if d' can be obtained from d by the passage of a single body through a gate. This uses region adjacency constraints as in the case of S_1 from Section III-B. For example, suppose there are $n = 4$ regions and $k = 12$ bodies. Suppose $d = (2, 3, 5, 2)$ and $d' = (2, 4, 5, 1)$. For $d \rightarrow_k d'$ to be true, there must be a gate between the second and fourth regions, which allows a body to transition from the fourth region to the second. For each transition, exactly two components of the distribution are allowed to change: One is incremented and the other is decremented.

It is possible to extend the transition systems to the case of *partially distinguishable* bodies, which could correspond to assigning them nonunique colors. In the limiting case, each body receives a unique color, making them fully distinguishable. To make a transition system for k bodies, the state space would be R^k, which is the k-fold Cartesian product of R. The transitions can be assigned in a standard way if the gates are independent; however, it becomes more complicated if gates are shared between bodies, as mentioned above. If there are i bodies of the same color, then the corresponding i components in R^k are replaced by D_i to express the distribution of i bodies, due to their mutual indistinguishability. These extensions lead to interesting open questions, however, we restrict the remainder of the paper to the completely indistinguishable case.

B. The methods

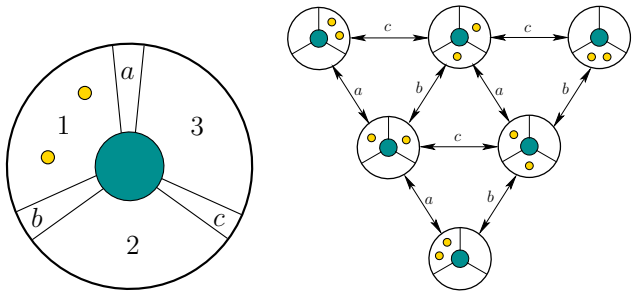
Assume that the initial distribution is given. To express tasks in LTL, let Π be the set of all propositions of the form π_d for every $d \in D_k$. An LTL formula ϕ can then be defined to express any task that involves distributions of bodies across the regions. The method follows in the same way as in Section III-C, which yields a distribution sequence $\tilde{d} = (d_0, d_1, \ldots)$ that satisfies ϕ. To ensure that the execution follows $\tilde{d}$, each gate must be aware of the current distribution to allow the transition, if appropriate. Each time a transition occurs, the count for the adjacent regions is incrementally modified.

An alternative method will now be given. The solutions so far are deterministic in the sense that the sequence $\tilde{r}$ or $\tilde{d}$ must be predictably executed. However, it is usually that case that many alternative sequences would also satisfy the formula. For the case of a single body, it might not be necessary to force it along the precise sequence $\tilde{r}$ of regions. For the case of multiple bodies, the situation is even worse because the bodies are prescribed to follow a precise sequence $\tilde{d}$ of distributions. For example, to arrive from a distribution of $(4, 0, 0)$ to $(0, 0, 4)$, the first transition should be $(4, 0, 0) \rightarrow (3, 1, 0)$. In the next transition, however, both $(3, 1, 0) \rightarrow (3, 0, 1)$ and $(3, 1, 0) \rightarrow (2, 2, 0)$ make progress toward the goal.

For the simple task of navigating one body to region r_g, a nondeterministic flow graph that solves the problem can be constructed as follows. Define a distance function $\rho : R \rightarrow \mathbb{N}$, in which $\rho(r)$ is the number of regions encountered on the shortest path in the bipartite graph (recall Figure 4) from r to r_i. Note that $\rho(r_g) = 1$. The values of ρ can be calculated by simple breadth-first search over the graph. Each static gate g can be configured as follows: If $\rho(r) < \rho(r')$ for a pair $r, r' \in N(g)$, then $m \in M(g)$ is set to allow flow from r' to r. In the flow graph, this construction generally allows multiple out edges from a single region. It does not matter which gate the body crosses in this case; any gate transition causes progress to be made.

C. A simple example

For the case of completely indistinguishable bodies, Figure 6(a) shows an example that has three regions $R = \{r_1, r_2, r_3\}$ and three gates $G = \{a, b, c\}$. Suppose that each gate allows the bodies to transition in either direction, depending on its configuration. The discrete transition system S_2 is given by (3) for $k = 2$. A distribution sequence $\tilde{d}$ for S_2 corresponds to a walk through the graph shown in Figure 6(b). Each edge is labeled with the gate that is crossed by the body that caused the transition.



(a) (b)

Fig. 6. a) An example with three regions, three gates, and two bodies; b) a graph that for which the vertices are D_2, the set of possible distributions, and the edges correspond to possible transitions.

Consider the following task. Suppose that both bodies are initially in r_1, as shown in Figure 6(a). The task is to bring them to r_3, then r_2, and then return to r_1. A corresponding LTL formula is

$$\Diamond(\pi_{(2,0,0)} \wedge \Diamond(\pi_{(0,0,2)} \wedge \Diamond(\pi_{(0,2,0)} \wedge \Diamond\pi_{(2,0,0)}))). \tag{4}$$

A possible solution trajectory for S_2 is depicted in Figure 7 as a sequence of body distributions for which transitions are caused by setting gate directions.

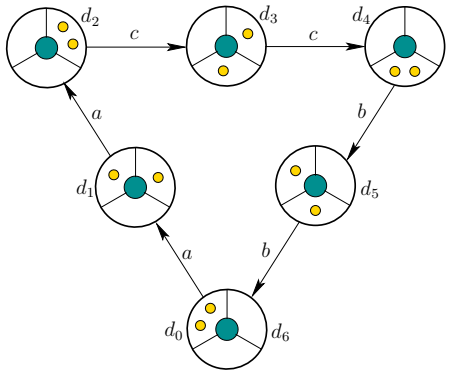
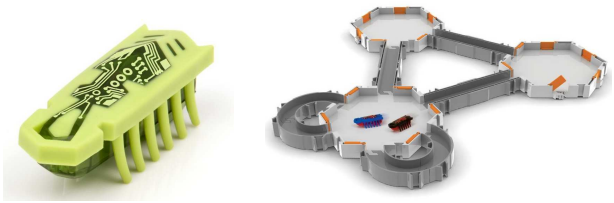


Fig. 7. An example sequence $\tilde{d} = (d_0, \ldots, d_6)$ of distributions that satisfies the LTL formula given in (4).



(a) (b)

Fig. 8. a) The vibrating Hexbug Nano toy also exhibits the wild property and is used in experiments; b) it in fact comes equipped with a "habitat" that it nicely explores.

V. Experiments

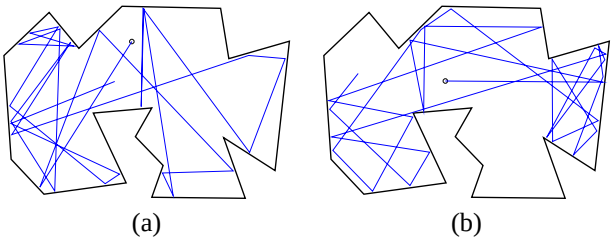
We designed and developed low-cost hardware to illustrate the methodology. For the executions, the printed frames in this section do not do justice to the execution of the system. Full videos appear at

`http://msl.cs.uiuc.edu/dts/`

A. Wild body implementations

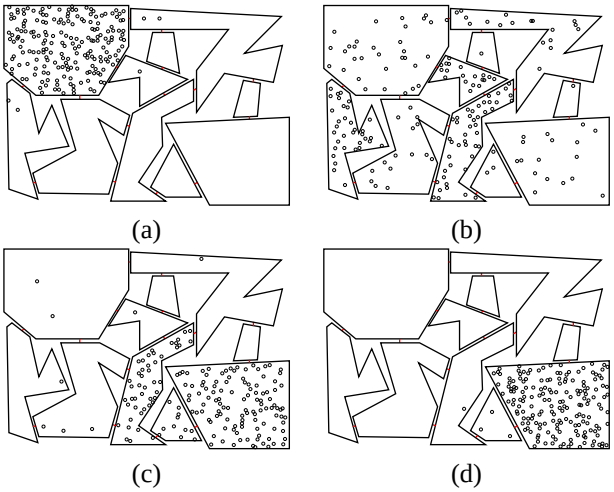
The first task is to find bodies that appear to fulfill the wildness condition of Section II-B. The main body used in our work is the weasel ball, which was shown in Figure 1. It costs around $4 US and consists of a plastic ball of radius 8.5cm that has only a single offset motor inside which oscillates at about 2Hz. We performed hundreds of experiments that consisted of placing one or more balls into regions and observing their motions. Without fail, they easily strike any reachable place along the boundary of a region, therefore becoming a suitable candidate for a wild body. We acknowledge, however, that no model of its mechanics is provided here and it is not formally proved to be wild. It is only verified experimentally.

An alternative to the weasel ball is the *Hexbug Nano* (Figure 8), which is a cheap (around $10 US) vibrating toy that looks like the end of a toothbrush with rubberized bristles and a vibrating motor mounted on top. This highly popular toy has been demonstrated to explore complex habitats with regions and gates, which can be purchased. In this case, the gates are opened (allowing bidirectional flow) or closed (blocking all flow) manually by the user. Recent kits even provide static gates that permit flow in one direction only.



(a) (b)

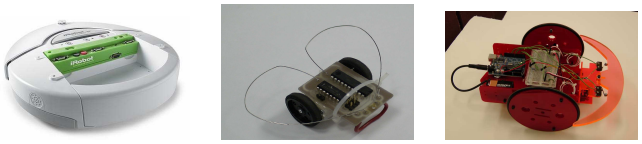
Fig. 9. a) A sample trajectory from a body that moves straight and then bounces at a random angle; b) a sample trajectory for the case of bouncing at the angle of incidence (as in an ideal billiard ball).



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

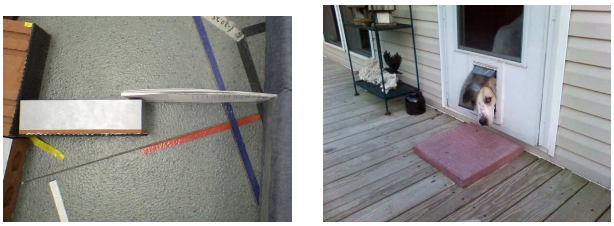
Fig. 10. A simulation of 200 robots in a complex environment with 10 regions. Each robot moves straight until it hits a boundary and then reflects at a random angle. The robots are guided from the upper left region to the lower right region using static gates.

Numerous other possibilities exist for wild bodies. Simple robot platforms can be used, which incorporate sensor feedback to determine that boundary contact has occurred. For example, consider a differential drive robot with a contact sensor. The robot rolls straight until it hits a wall. It then rotates a random amount and continues moving straight. Simulation trajectories are shown in Figure 9(a) for this motion model. Another reasonable motion model is to deterministically bounce, like in the Bunimovich stadium (recall Figure



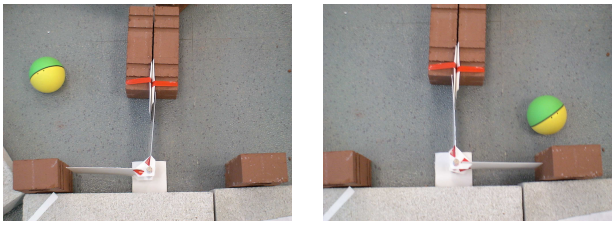
(a) (b) (c)

Fig. 11. Some simple robots that we have used to implement the straight-line motion with random bouncing: a) A Roomba iCreate; b) a simple home-brew robot made from a pager motor (total cost less than $30 US); c) a SERB open-source robot constructed from acrylic, cheap motors, and an Arduino microcontroller (total cost less then $120 US).



(a) (b)

Fig. 12. a) A static, directional gate can be implemented making a flexible "door" from a stack of paper; in this case, the body can transition only from the bottom region to the top; b) this works much like a "doggie door".



(a) (b)

Fig. 13. A pliant gate with two modes: a) a ball can pass from left to right, but its blocked the other way; b) a ball can only pass from right to left.

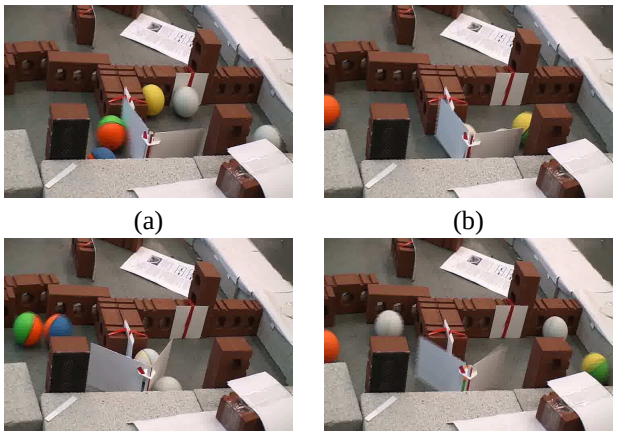
2); see Figure 9(b). Figure 10 shows 200 simulated robots accomplishing a simple task under these bouncing models. We have implemented the random bounce angle model in several robots, which are shown in Figure 11. Experiments with simple, wild mobile robots are covered in [3]; however, this paper is restricted to simpler bodies.

B. Gate designs

Recall from Section II-D that gates may be either static or dynamic. A simple way to engineer a successful static gate is illustrated in Figure 12. A body moving from the bottom region to the top region can pass through the right side by bending the paper; a body moving in the other direction is blocked. This simple setup was reliable in experiments. In other experiments, we have used ramps and ledges as static gates that allow flow in one direction only.

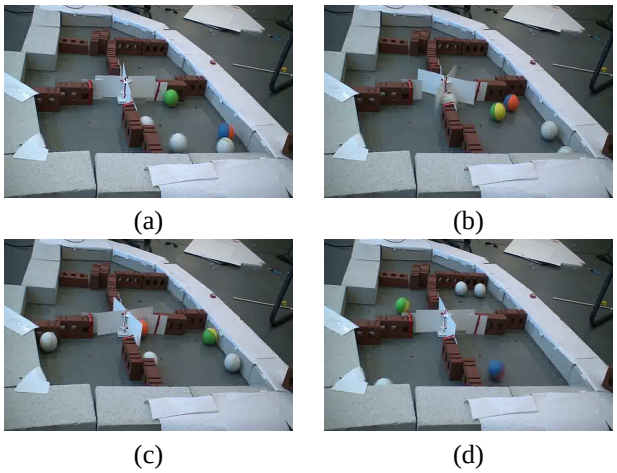
Now consider dynamic gates. There are two categories of dynamic gates, depending on whether external energy is used to actuate the gates. If the gate configuration changes only as a result of forces applied by the passing body, then the gate is called *pliant*. Otherwise, it is called a *controllable* gate.

First consider the case of a pliant gates. Figure 13 shows a simple example in which an "L"-shaped door is placed on a swivel base. In this case, the door is made of cardboard panels attached to a straw. The door pivots when the body passes through. Figure 14 shows an experiment that illustrates this for five weasel balls. On its own, this gate enforces the constraint that the number of bodies per room must remain roughly constant, even though any ball is allowed to pass in either direction. This behavior can be thought of as an alternating prisoner exchange.



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

Fig. 14. a) Initially, two bodies are in the right region and three are in the left; the gate configuration allows a right to left transition; b) after 18 seconds a body crosses right to left, changing the gate mode; c) after 40 seconds a body moves left to right, changing the gate configuration again; d) number of bodies in each region alternates between two and three for the rest of the experiment.

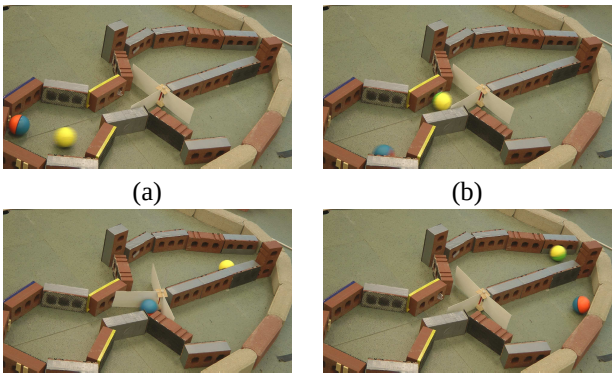


(a) (b) (c) (d)

Fig. 15. a) Initially, the five bodies are together in one region and only clockwise transfers are allowed; b) after 20 seconds a body changes regions and counterclockwise transfers are allowed; c) 5 seconds later a body changes regions; d) after 92 seconds, the bodies occupy all four regions.

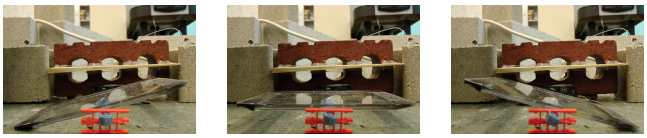
We also designed and implemented a four-way revolving door, which is a pliant gate that has four adjacent regions. It is only allowed to rotate up to 90 degrees and alternates between two modes: 1) Allowing a clockwise transfer and 2) allowing a counterclockwise transfer. An experiment with five weasel balls is illustrated in Figure 15. Our final pliant gate design is shown in Figure 16. In this case, the gate configuration determines which region will receive the body, which alternates after each transition.

Many interesting open questions remain with regard to pliant gates. What is the family of tasks that can be solved entirely using pliant gates? What types of mechanisms for pliant gates can be designed? This depends on the body design. Is it possible to allow capacitance by storing and then releasing



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

Fig. 16. This pliant gate alternates between the destination regions by using a rotating "T" shape. a) Initially, two bodies are in one region; b) the gate transfers the first body to the upper region; c) the gate then transfers the second body to the lower region; d) both bodies become trapped in separate regions. If there are $2n$ bodies in the initial region, then this gate would place n bodies in each destination region.



(a) (b) (c)

Fig. 17. The three gate configurations: a) the gate allows a body to cross in the left to right direction, b) the gate prevents bodies from crossing in either direction, and c) the gate allows an body to cross in the right to left direction.

bodies? Would this allow additional tasks to be accomplished?

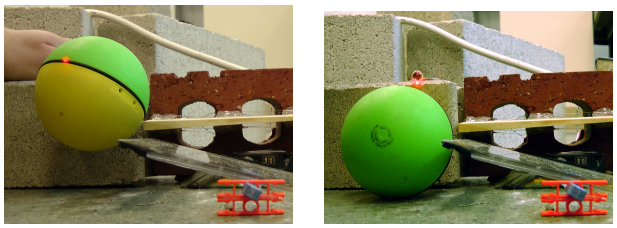
Now consider controllable gates. In this case, what information is available to determine when to change the configuration? This requires the incorporation of sensors that provide necessary information during execution.

Our controllable gate is made from a piece of acrylic in the form of a *ramp*. By tilting the ramp, the direction of the gate is altered, and we can obtain three gate configurations to execute the gate actions, as seen in Figure 17.

The acrylic ramp element is attached to Futaba S3003 servo motors using standard servo horns. Servo motors were chosen for this application because they are inexpensive (around $8 US each) and allow precise control of output angle by the use of negative feedback. Additionally, the only control input required is a Pulse-Width Modulation (PWM) signal, which is easily generated by most microcontrollers.

Simple sensor feedback is provided to the gate. A body crossing is detected through the use of optical emitter-detector pairs. Laser pointers were chosen because they are inexpensive (about $3 US each) and easily aimed. The laser pointers were modified to run on external battery packs and held in place by simple armature mounts (about $3 US each). Simple photodiodes (about $2 US each) were mounted on the opposite side to detect the laser beams.

A change in voltage is observed when a body crosses the beam, thereby blocking the laser beam from reaching the photodetector. As can be seen in Figure 18, the laser beam/photodetector pairs are placed so that only an body which has just crossed a gate causes a beam crossing.



(a) (b)

Fig. 18. a) A ball that has just crossed the gate interrupts the laser beam, while b) a body simply moving within a region does not interrupt the laser beam, which is visible just above the ball in the picture.

As previously mentioned, the ramp-type gates are implemented using servo motors. The angular position of these servo motors is determined by the duty cycle of the PWM signal they receive. For this purpose, we used an Arduino Mega microcontroller board based on the Atmel ATmega1280 microcontroller. This platform was chosen because it is easy to configure and inexpensive (about $35 US). Additionally, Arduino documentation and code examples are plentiful.

C. Single-body experiments

Several experiments are presented using the method presented in Section III-C. We chose typical tasks specified using LTL, similar to those in [34]. If the solution region sequence $\tilde{r}$ enabled it due to the uniqueness condition of Section III-C, we solve the problem using static gates. Otherwise, the system of controllable gates, as described in Section V-B were sufficient for enforcing any solution $\tilde{r}$ to be achieved during execution.

Several experiments were conduced with a weasel ball in an environment of approximately 2 by 3 meters and five gates; see Figure 19. For the region and gate names, recall Figure 5. The specification of the task that we would like to achieve is: "Starting in r_2, go to r_4". An LTL formula that captures this specification is $\phi = \Diamond\pi_4$. We applied NuSMV on the corresponding discrete transition system S_1 and the formula ϕ. The output region sequence $\tilde{r} = (r_2, r_3, r_4)$ implies that gates d and e should allow transitions from r_2 to r_3 and from r_3 to r_4. The execution is shown in Figure 19.

We also demonstrated patrolling by introducing the LTL formula

$$\phi = \Box(\Diamond\pi_0 \wedge \Diamond\pi_1 \wedge \Diamond\pi_3), \tag{5}$$

which means that r_0, r_1, and r_3 must be visited infinitely often. An infinite sequence $\tilde{r}$ that cycles through all regions was returned by NuSMV. A gate configuration that implements the sequence is shown in Figure 20 along with part of the actual execution. The ball visits attempts to visit the required regions infinitely often (in reality, its battery dies).

Figure 21 shows an example in which regions must be visited in a particular sequence. Suppose that we want a subsequence of $\tilde{r}$ to visit regions in the following order: r_0

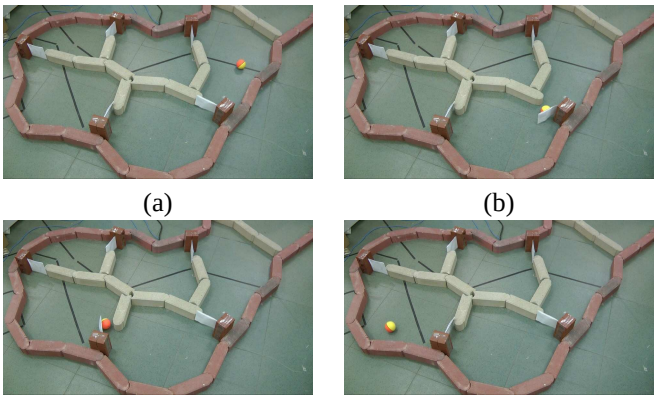


Fig. 19. "Starting in r_2, go to r_4": a) The weasel ball is placed initially in r_4 (leftmost); b) after 30 seconds strikes gate d and enters r_3; c) after 105 seconds it strikes gate e and d) moves into r_4, which completes the task.

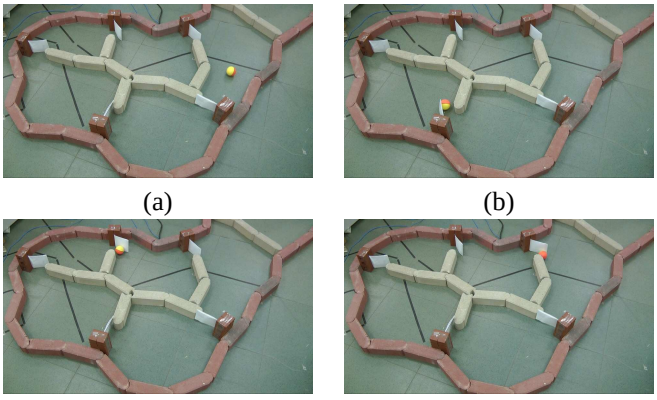


Fig. 20. "Patrol regions r_0, r_3 and r_1": a) The ball starts its route; b) after 107 seconds it has entered two new regions; c) after 212 seconds it has visited most regions; d) after 225 seconds, it completes a tour of all regions, and continues.

(upper right), r_1 (upper left), r_2 (lower left), r_3 (lower right), r_2, r_1, r_0. An LTL formula that achieves this is

$$\phi = \Diamond(\pi_0 \wedge \Diamond(\pi_1 \wedge \Diamond(\pi_2 \wedge \Diamond(\pi_3 \wedge \Diamond(\pi_2 \wedge \Diamond(\pi_1 \wedge \Diamond\pi_0)))))). \tag{6}$$

The experiment for this example appears in Figure 21.

Finally, Figure 22 shows a more complicated example. The LTL formula that describes the task is:

$$\begin{aligned}\phi =& \pi_1 \wedge \bigcirc\Box(\neg\pi_1 \wedge \Diamond\pi_2 \wedge \Diamond\pi_3 \wedge (\pi_5 \rightarrow \bigcirc\pi_4) \wedge \\ & \forall_{j\neq 4}(((\pi_4 \wedge \bigcirc\pi_j) \rightarrow \neg\Diamond(\pi_j \wedge \bigcirc\pi_4)) \\ & \wedge ((\pi_j \wedge \bigcirc\pi_4) \rightarrow \neg\Diamond(\pi_4 \wedge \bigcirc\pi_j))))\end{aligned} \tag{7}$$

Starting in r_1, we want the robot to patrol r_2 and r_3, requiring that the body moves to r_5 after being in r_4 (and not reversed), and all flows incident to r_4 are constrained to move in one direction for all time ; moreover, r_1 must be avoided once it is visited. In 0.031 seconds, the NuSMV package (running under Ubuntu 10.04 on a PC with Intel Core 2 Quad 2.4GHz

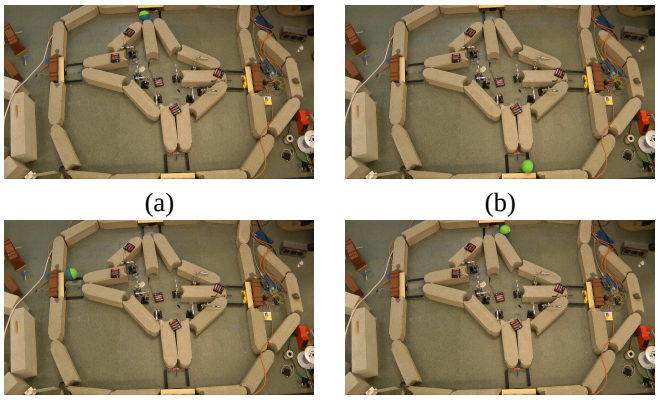


Fig. 21. A coverage task: a) The body crosses into the upper-left region; b) after 15 seconds, the body crosses into the lower-right region, completing the coverage; c) after 50 seconds, the body crosses into the upper-left region on the return trip; d) after 240 seconds, the body returns to the upper-right region.

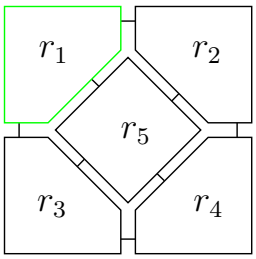


Fig. 22. An example that involves 5 regions and 8 gates.

processor and 4GB of memory) returned the infinite sequence

$$\tilde{r} = r_1(r_5 r_4 r_2 r_5 r_4 r_3)^+ \tag{8}$$

in which $^+$ denotes that the subsequence repeats forever. This was by far the longest running time of any applications of NuSMV to our LTL formulas for a single body. The policy was successfully implemented in simulation. Note that it cannot be implemented with static gates because $\tilde{r}$ does not satisfy the uniqueness condition from Section III-C.

D. Multiple-body experiments

The first two experiments in this section apply the first method of Section IV-B. The controllable gate setup shown in Figures 17 and 18 is sufficient to implement any sequence of body distributions produced by a model checker. Using the controllable gates, we implemented several tasks, such as: "Starting with all four bodies in r_0 (upper-right), cover all four regions simultaneously and then meet again in r_3 (lower-right)". One way to achieve this is to define

$$\phi = \Diamond(\pi_{(1,1,1,1)} \wedge \Diamond\pi_{(0,0,0,4)}). \tag{9}$$

See Figure 23 for the implementation.

The second experiment involves two bodies in the environment shown in Figure 22. For this task we want the bodies to meet in all of the outer regions (r_1, r_2, r_3, r_4) for dual

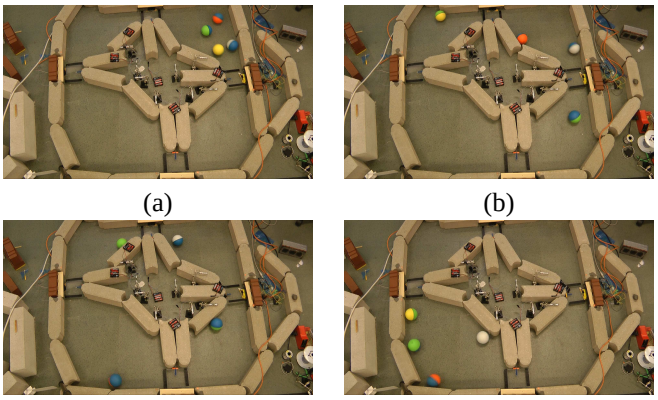


Fig. 23. A group splitting and coverage example: a) The 4 bodies begin together in the upper-right region; b) after 37 seconds the bodies begin to split; c) after 45 seconds bodies have split completely into independent regions; d) after 240 seconds bodies reconvene in the lower-left region.

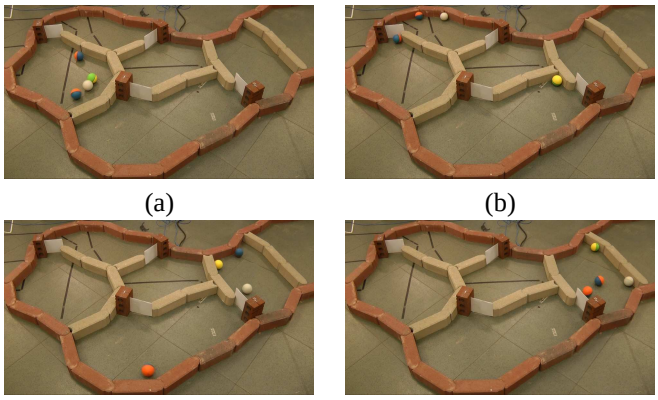


Fig. 24. Navigation with multiple balls: a) Four weasel balls are started in the left-most region; b) after 48 seconds some progress is made; c) after 262 seconds, all but one ball have arrived at the destination; after 270 seconds, all four balls have arrived.

patrolling (two at a time in a region) of each region; r_5 is constrained to have space for only one body at a time; moreover, after r_1 is visited, at least one body has to be there until r_4 is visited; the same restriction holds for regions r_3 and r_2, respectively. Also, after any of the bodies visits r_5, it must move to r_4. The corresponding LTL formula is

$$\begin{aligned}\phi =&\Box(\neg\pi_{(0,0,0,0,2)} \wedge \Diamond\pi_{(2,0,0,0,0)} \wedge \Diamond\pi_{(0,2,0,0,0)}\wedge\\ &\Diamond\pi_{(0,0,2,0,0)} \wedge \Diamond\pi_{(0,0,0,2,0)}\wedge\\ &((\pi_{(1,0,0,0,1)} \vee \pi_{(1,0,0,1,0)} \vee \pi_{(1,0,1,0,0)} \vee \pi_{(1,1,0,0,0)}) \rightarrow\\ &((\pi_{(1,0,0,0,1)} \vee \pi_{(1,0,0,1,0)} \vee \pi_{(1,0,1,0,0)} \vee \pi_{(1,1,0,0,0)}\vee\\ &\pi_{(2,0,0,0,0)})U\pi_{(1,0,0,1,0)}))\wedge\\ &((\pi_{(0,0,1,0,1)} \vee \pi_{(0,0,1,1,0)} \vee \pi_{(0,1,1,0,0)} \vee \pi_{(1,0,1,0,0)}) \rightarrow\\ &((\pi_{(0,0,1,0,1)} \vee \pi_{(0,0,1,1,0)} \vee \pi_{(0,1,1,0,0)} \vee \pi_{(1,0,1,0,0)}\vee\\ &\pi_{(0,0,2,0,0)})U\pi_{(0,1,1,0,0)}))\wedge\\ &((\pi_{(1,0,0,0,1)} \vee \pi_{(0,1,0,0,1)} \vee \pi_{(0,0,1,0,1)}) \rightarrow\\ &\bigcirc(\pi_{(1,0,0,1,0)} \vee \pi_{(0,1,0,1,0)} \vee \pi_{(0,0,1,1,0)}))\wedge\\ &(\pi_{(0,0,0,1,1)} \rightarrow \bigcirc\pi_{(0,0,0,2,0)}))\end{aligned} \tag{10}$$

The NuSMV package found the following solution in 0.304 seconds:

$$\begin{aligned}\tilde{d} = &((2,0,0,0,0),(1,0,0,0,1),(1,0,0,1,0),(0,0,0,1,1),\\ &(0,0,0,2,0),(0,0,1,1,0),(0,0,2,0,0),(0,0,1,1,0),\\ &(0,1,1,0,0),(0,1,0,1,0),(0,2,0,0,0),(1,1,0,0,0),\\ &(2,0,0,0,0)),\end{aligned} \tag{11}$$

which is a distribution sequence that satisfies ϕ. This solution was implemented in a simulation of the bodies and gates.

Now recall the alternative method from Section IV-B, which allows nondeterminism to move multiple bodies to a goal region r_g. Figure 24 shows a navigation task which modes 4 bodies from r_4 to r_2. In another experiment, shown in Figure 25, 50 weasel balls are guided from a starting to a goal region,

Fig. 25. In this experiment, 50 weasel balls were successfully manipulated from a source region into a destination region.

in an environment with 6 regions and 6 gates. The regions are complicated shapes, some with interior obstacles, and the gates are narrow. It took around 40 minutes for all 50 balls to arrive in the goal due complicated regions, small gates, and a long tail distribution on arrival times. A nondeterministic version of patrolling with multiple bodies is shown in Figure 26.

Some experiments are also shown for the case of Hexbug Nanos. By placing a small piece of paper in the doorway between two regions, we have implemented a simple way to enforce one-way flow. The paper is allowed to bend in one direction, but is blocked in the other. Figure 27 shows an experiment in which 4 Nanos were induced to flow from the leftmost region to the rightmost region by designing two one-way gates out of strips of paper. Figure 28 shows another experiment. In this case, a small environment was constructed from inexpensive Magna-Tiles and 10 Nanos are controlled to flow from an initial region to a goal region. In this case, each gate was implemented by stacking tiles on the ground. Each tile is approximately 5mm wide. To induce a flow from region r to region r', we stack n tiles to make the floor of r, and $(n-1)$ tiles for the floor of r'. Each Nano then experiences a

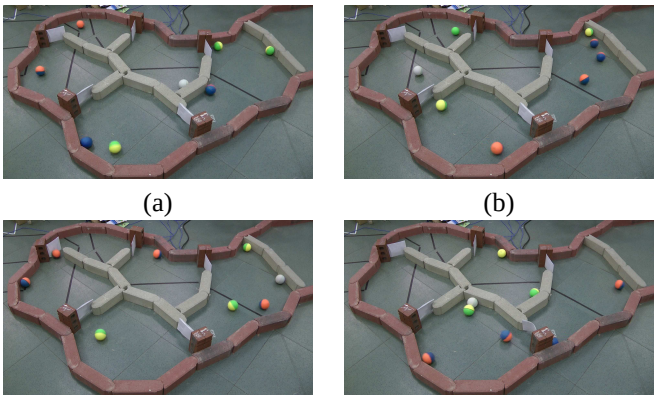


Fig. 26. "Patrol all regions indefinitely": a) 8 bodies are placed in arbitrary regions and static gates are configured in a cycle; b) 3 minutes later, most bodies have changed places; (c) after 6 minutes the patrolling continues; d) configuration 12 minutes after the initial setup

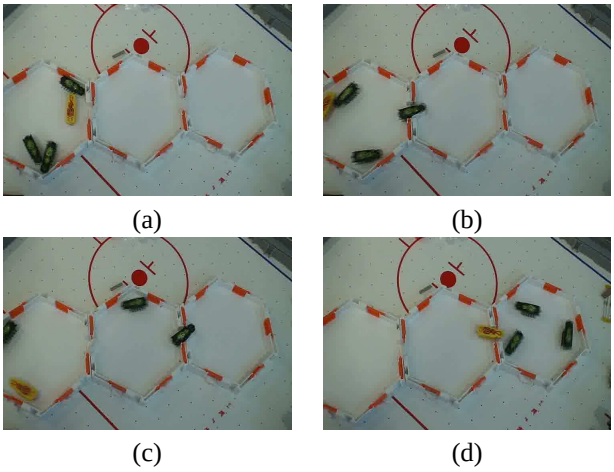


Fig. 27. Navigation of Hexbug Nanos: a) Initially, all the four Nanos are together in the left region; b) after 10 seconds one Nano changes regions; c) after 17 seconds one Nano crosses from the second region to the third; d) after 70 seconds all Nanos are in the third region.

small dropoff on the boundary between r and r' and is unable to return to r from r'. This induces the directional flow.

It is clear from these examples that allowing nondeterministic region transitions leads to dramatic complexity reduction and performance improvements. We would ideally like to take complicated LTL formulas and synthesize an automaton that expresses all region sequences that satisfy the formula while simultaneously determining what sensors and gate configuration combinations are sufficient for achieving the task. This remains for future work.

VI. Conclusions

We developed an approach to solve a variety of common robotic tasks by placing wildly moving bodies into a complicated environment and then gently guiding them through gates that can be reconfigured. This avoids many common difficulties such as system identification, heavy sensing, filtering and estimation, map building, localization, coordination, and communication. Tasks can be specified using a high-level logic, such as LTL, and then gate configurations are set to satisfy the formula and achieve the desired task. Several experiments were shown for weasel balls and Hexbug Nanos performing tasks such as navigation, patrolling, and coverage. Various types of gates were designed, including static, pliant, and controllable with sensor feedback. With the successes so far, it seems we have barely scratched the surface on the set of possible systems that can be developed in this way to solve interesting tasks.

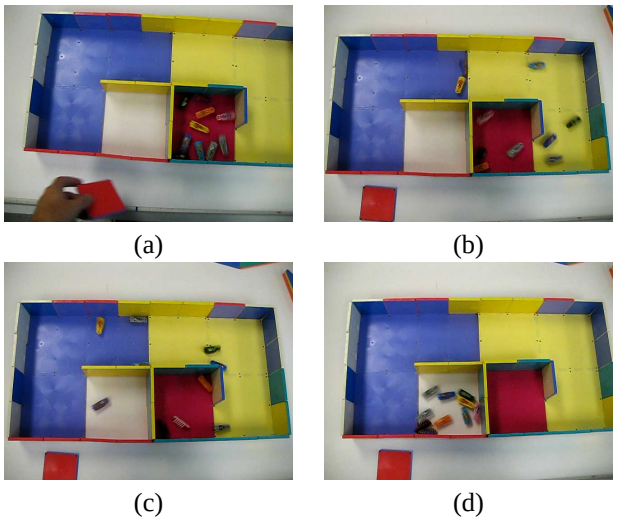


Fig. 28. A Nano experiment involving 4 regions (red, yellow, blue, and white). The task is to move 10 Nanos from the red region to the white region: a) Initially, all 10 Nanos are placed on the highest platform (the red square at the bottom); b) after 5 seconds, significant progress has been made; c) after 8 seconds the first Nano arrives in the white goal region; d) after 44 seconds, all 10 Nanos arrive in the goal region.

Designing information-feedback plans: A remaining challenge is to formally characterize the minimal amount of sensing information that is needed to switch gate configurations and accomplish desired tasks. For each task the requirement may be different. A *plan* or *control law* can generally be expressed as an information-feedback mapping $\pi : \mathcal{I} \rightarrow C$, in which C is the global configuration space for gates and $\mathcal{I}$ is an *information space* that takes into account actuation histories and sensor observation histories (see Chapter 11 of [37]). Recall that for each global configuration, there is a corresponding flow graph. We can therefore imagine π as specifying a dynamic flow graph, which changes its flow as new information becomes available.

There are many possible choices for $\mathcal{I}$, depending on the kind of sensors and filters that are developed. By taking a minimalist approach, we use the weakest sensors and filters that can nevertheless accomplish the task. Therefore, we avoid the case in which each information state in $\mathcal{I}$ specifies the precise configuration and velocities of every body and the configurations of all of the gates. We could consider *time feedback*, for which $\mathcal{I} = T = [0, t]$, an interval of time.

We will also use simple sensors that detect whether a body has passed in or out of a gate. If Y represents the set of all sensor outputs, then we obtain *sensor feedback* plans of the form $\pi : Y \rightarrow C$. For the experiments of Section V sensor feedback was sufficient to switch the gates. More generally, the sensor readings could be aggregated into a filter that updates an information state after each new reading. The filter information state is then used as feedback for the plan. See [38] for related details.

Analyzing execution times: Another important direction is to analyze the time it takes to enter the gate for various motion models, region shapes, and gate widths. Can objective criteria be formulated for the motion and then optimized through a simple motion strategy for the body? Furthermore, statistical analysis might enable us to predict the expected time to completion for a task, which is currently a weakness of our approach.

Toward useful applications: To achieve more useful tasks, we envision enhancing the bodies with limited amounts of sensing, controllable actuation, and computation. This substantially changes the power of the bodies. For example, a body can use its sensors and decide on the gate configuration for itself. This results in a "virtual" gate, much in the same way that artificial walls can be set up when using the popular Roomba vacuum cleaner. For example, suppose that region boundaries are simply marked on the floor by colored tape. We have performed some early experiments in which an iRobot Create equipped with a cheap color sensor can move over colored tape on the floor, deciding whether to "bounce" from the tape or pass through it, depending on the mode. The tape and color sensor simulate the gate. This extension has worked well in recent experiments and appears in [3].

We hope to develop wild bodies that solve more useful tasks. As a step in this direction, we have equipped one weasel ball with a small Wi-Fi module and microcontroller, allowing it to use Wi-Fi connections while wilding moving around. This enables more interesting tasks to be performed, such as Wi-Fi-based SLAM [18]. Better performance could be obtained from a specialized radio signal source. We imagine that a collection of wild bodies would be useful for exploration and mapping if equipped with appropriate sensors for this purpose. As another task, we could equip each body with an Annoy-a-tron circuit board, which costs around $13 US and emits a loud, piercing sound at irregular intervals, without warning. We could program the bodies to diffuse in a hostile indoor environment and then switch into an "annoy" mode during which the building inhabitants are constantly distracted by tiny devices stationed in unknown locations. To accomplish more tasks, the basic control and coordination is provided by allowing wild motions and traveling through the discrete transition systems, and we are free to enhance the robots however we like.

Acknowledgments: This work was supported in part by NSF grant 0904501 (IIS Robotics), NSF grant 1035345 (CNS Cyberphysical Systems), DARPA SToMP grant HR0011-05-1-0008, and MURI/ONR grant N00014-09-1-1052. The authors are grateful for the helpful suggestions of Hadas Kress-Gazit and the anonymous reviewers.

REFERENCES

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

[2] L. Bobadilla, K. Gossman, and S. M. LaValle. Manipulating ergodic bodies through gentle guidance. In *Proceedings IEEE Conference on Robot Motion and Control*, 2011.

[3] L. Bobadilla, F. Martinez, E. Gobst, K. Gossman, and S. M. LaValle. Controlling wild mobile robots using virtual gates and discrete transitions. In *American Control Conference*, 2012.

[4] L. Bobadilla, O. Sanchez, J. Czarnowski, K. Gossman, and S. M. LaValle. Controlling wild bodies using linear temporal logic. In *Proceedings Robotics: Science and Systems*, 2011.

[5] K.-F. Böhringer, V. Bhatt, B. R. Donald, and K. Goldberg. Algorithms for sensorless manipulation using a vibrating surface. *Algorithmica*, 26:389–429, 2000.

[6] M. S. Branicky, V. S. Borkar, and S. K. Mitter. A unified framework for hybrid control: Model and optimal control theory. *IEEE Transactions on Automatic Control*, 43(1):31–45, 1998.

[7] L. A. Bunimovich. On the ergodic properties of nowhere dispersing billiards. *Communcations in Mathematical Physics*, 65:295–312, 1979.

[8] Z. Butler, P. Corke, R. Peterson, and D. Rus. Virtual fences for controlling cows. In *IEEE International Conference on Robotics and Automation*, pages 4429–4436, 2004.

[9] L. G. Chalmet, R. L. Francis, and P. B. Saunders. Network models for building evacuation. *Fire Technology*, 18(1):90–113, 1982.

[10] A. Cimatti, E. Clarke, E. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An open source tool for symbolic model checking. In *Computer Aided Verification*, pages 241–268, 2002.

[11] E. M. Clarke, O. Grumberg, and D. A. Peled. *Verification of Reactive Systems: Formal Methods and Algorithms*. MIT Press, Cambridge, MA, 1999.

[12] M. Egerstedt and X. Hu. A hybrid control approach to action coordination for mobile robots. *Automatica*, 38(1):125–130, January 2001.

[13] E. A. Emerson. Temporal and modal logic. *Handbook of theoretical computer science*, 8:995–1072, 1990.

[14] M. A. Erdmann. Randomization in robot tasks. *International Journal of Robotics Research*, 11(5):399–436, October 1992.

[15] M. A. Erdmann and M. T. Mason. An exploration of sensorless manipulation. *IEEE Transactions on Robotics & Automation*, 4(4):369–379, August 1988.

[16] G. E. Fainekos. Revising temporal logic specifications for motion planning. In *Proceedings IEEE International Conference on Robotics & Automation*, 2011.

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

[18] B. Ferris, D. Haehnel, and D. Fox. WiFi-SLAM using Gaussian process latent variable models. In *International Joint Conference on Artificial Intelligence*, 2007.

[19] R. Fierro, A. Das, V. Kumar, and J. P. Ostrowski. Hybrid control of formations of robots. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 157–162, 2001.

[20] C. Finucane, G. Jing, and H. Kress-Gazit. LTLMoP: Experimenting with language, temporal logic and robot control. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1988–1993, 2010.

[21] E. Frazzoli, M. A. Dahleh, and E. Feron. Robust hybrid control for autonomous vehicles motion planning. Technical Report LIDS-P-2468, Laboratory for Information and Decision Systems, Massachusetts Institute of Technology, 1999.

[22] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3-4):219–253, 1982.

[23] K. Y. Goldberg. Orienting polygonal parts without sensors. *Algorithmica*, 10:201–225, 1993.

[24] E. Gutkin. Billiards in polygons. *Physica D*, 19:311–333, 1986.

[25] E. Gutkin. Billiards in polygons: A survey of recent results. *Journal of Statistical Physics*, 83(1/2):675–737, 1996.

[26] E. Haghverdi, P. Tabuada, and G. J. Pappas. Bisimulation relations for dynamical, control, and hybrid systems. *Theoretical Computer Science*, 342(2-3):229–261, September 2005.
[27] G. J. Holzmann. *The SPIN model checker: Primer and reference manual*. Addison Wesley Publishing Company, 2004.
[28] S. Kerckhoff, H. Masur, and J. Smillie. Ergodicity of billiard flows and quadratic differentials. *Annals of Mathematics*, 124(2):293–311, 1986.
[29] M. Kloetzer and C. Belta. Temporal logic planning and control of robotic swarms by hierarchical abstractions. *Robotics, IEEE Transactions on*, 23(2):320–330, 2007.
[30] M. Kloetzer and C. Belta. Automatic deployment of distributed teams of robots from temporal logic motion specifications. *IEEE Transactions on Robotics and Automation*, 26(1):48–61, 2010.
[31] H. Kress-Gazit. *Transforming high level tasks to low level controllers*. PhD thesis, University of Pennsylvania, 2008.
[32] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.
[33] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Temporal-logic-based reactive mission and motion planning. *IEEE Transactions on Robotics and Automation*, 25(6):1370–1381, December 2009.
[34] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Temporal logic motion planning for mobile robots. In *Proceedings IEEE International Conference on Robotics and Automation*, 2005.
[35] H. Kress-Gazit, G.E. Fainekos, and G.J. Pappas. Where's Waldo? sensor-based temporal logic motion planning. In *Proceedings IEEE International Conference on Robotics and Automation*, 2007.
[36] M. Lahijanian, J. Wasniewski, S. B. Andersson, and C. Belta. Motion planning and control from temporal logic specifications with probabilistic satisfaction guarantees. In *Proceedings IEEE International Conference on Robotics & Automation*, pages 3227–3232, 2010.
[37] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Also available at http://planning.cs.uiuc.edu/.
[38] S. M. LaValle. Sensing and filtering: A tutorial based on preimages and information spaces. *Foundations and Trends in Robotics*, 2012. To appear.
[39] S. G. Loizou and K. J. Kyriakopoulos. Automatic synthesis of multi-agent motion tasks based on LTL specifications. In *Proceedings IEEE Conference Decision and Control*, pages 153–158, 2005.
[40] M. T. Mason. *Mechanics of Robotic Manipulation*. MIT Press, Cambridge, MA, 2001.
[41] D. Reznik, E. Moshkoich, and J. Canny. Building a universal planar manipulator. In K. F. Bohringer and H. Choset, editors, *Distributed Manipulation*, pages 147–171. Kluwer, Norwell, MA, 2000.
[42] D. A. Shell, C. V. Jones, and M. J. Mataric. Ergodic dynamics by design: A route to predictable multirobot systems. In *Multi-Robot Systems: From Swarms to Intelligent Automata*, pages 291–297, 2005.
[43] S. L. Smith, J. Tumova, C. Belta, and D. Rus. Optimal path planning under temporal logic constraints. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3288–3293, 2010.
[44] S. Tabachnikov. *Geometry and Billiards*. American Mathematical Society, Providence, Rhode Island, 2005.
[45] P. Tabuada and G. J. Pappas. Linear time logic control of discrete-time linear systems. *Automatic Control, IEEE Transactions on*, 51(12):1862–1877, 2006.
[46] C. Tomlin, G. J. Pappas, and S. Sastry. Conflict resolution for air traffic management: A study in multiagent hybrid systems. *IEEE Transactions on Automatic Control*, 43(4):508–521, April 1998.
[47] T. Vose, P. Umbanhowar, and K. M. Lynch. Sliding manipulation of rigid bodies on a controlled 6-DoF plate. In *Proceedings of Robotics: Science and Systems*, Los Angeles, CA, USA, June 2011.
[48] M. Wu, G. Yan, Z. Lin, and Y. Lan. Synthesis of output feedback control for motion planning based on LTL specifications. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5071–5075, 2009.