Dynamic Obstacle Representations for Robot and Virtual Agent Navigation

Eric Aaron and Juan Pablo Mendoza

Department of Mathematics and Computer Science
Wesleyan University
Middletown, CT 06459

Abstract. This paper describes a reactive navigation method for autonomous agents such as robots or actors in virtual worlds, based on novel *dynamic tangent* obstacle representations, resulting in exceptionally successful, geometrically sensitive navigation. The method employs three levels of abstraction, treating each obstacle entity as an *obstacle-valued function*; this treatment enables extraordinary flexibility without pre-computation or deliberation, applying to all obstacles regardless of shape, including non-convex, polygonal, or arc-shaped obstacles in dynamic environments. The unconventional levels of abstraction and the geometric details of dynamic tangent representations are the primary contributions of this work, supporting smooth navigation even in scenarios with curved shapes, such as circular and figure-eight shaped tracks, or in environments requiring complex, winding paths.

1 Introduction

For autonomous agents such as robots or actors in virtual worlds, navigation based on potential fields or other reactive methods (e.g., [3,4,6,9,10]) can be conceptually elegant, robust, and adaptive in dynamic or incompletely known environments. In some methods, however, straightforward geometric representations can result in ineffective obstacle avoidance or other navigation difficulties. In this paper, we introduce reactive navigation intelligence based on *dynamic tangent* obstacle representations and repellers, which are locally sensitive to relevant obstacle geometry, enabling effective navigation in a wide range of environments.

In general, reactive navigation is fast and responsive in dynamic environments, but it can be undesirably insensitive to some geometric information in complicated navigation spaces. In some potential-based or force-based approaches, for instance, a circular obstacle would be straightforwardly treated as exerting a repulsive force on agents around it, deterring collisions; as an example, Figure 1 illustrates an angular repeller form employed in [5,7,8], in which a circle-shaped obstacle obs_i repels circle-shaped agent A by steering A's heading angle away from all colliding paths. (See Section 2 for additional information on this kind of angular repeller.) Straightforwardly, the repeller representation of obstacle obs_i is based on the entire shape of obs_i. Such a straightforward connection between the entire shape of an *obstacle entity* and the repeller representation of that entity, however, is not always so successful. Some common obstacle entities, for example, have shapes inconsistent with otherwise-effective navigation methods; for

C. Butz and P. Lingras (Eds.): Canadian AI 2011, LNAI 6657, pp. 1–12, 2011.

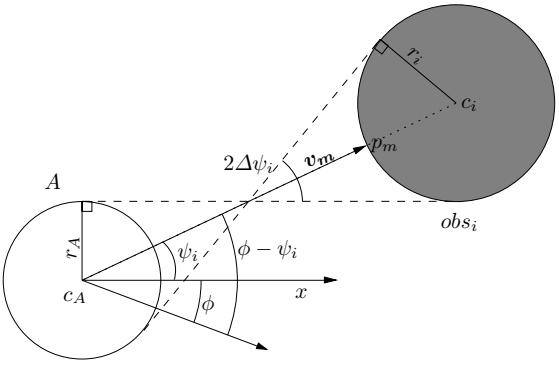

Fig. 1. Obstacle avoidance, with agent A, obstacle obs_i, and other elements as labeled. When heading angle ϕ is inside the angular range delimited by the dotted lines—i.e., when some point of A is not headed outside of obs_i—A is steered outside of that angular range, avoiding collision.

example, navigation methods requiring circle-based obstacle representations [2,3,5,7] can be ineffective with obstacles that have long, thin shapes, such as walls.

Indeed, our work is motivated by difficulties in applications requiring navigation near or along walls in dynamic environments, such as boundary inspection [1] or navigation in hallways. For this paper, we distinguish between *boundary-proximate* and *boundary-distant* navigation: Boundary-proximate behaviors require navigation along obstacle boundaries, whereas boundary-distant behaviors require only collision avoidance, which tends to deter proximity to obstacle boundaries. Boundary-distant reactive behaviors can often be straightforwardly achieved by, e.g., potential-based navigation that employs forceful repellers and ignores details such as concavities in obstacle shapes. Boundary-proximate reactive behaviors, however, are more challenging. This paper is thus focused on boundary-proximate behaviors (although as noted in Section 5, our method supports both kinds of behavior), presenting efficient, geometrically sensitive, dynamic obstacle representations that support boundary-proximate navigation.

In particular, this paper introduces dynamic tangent (*DT*, for short) obstacle representations as intermediaries between obstacle entities and repeller representations. Dynamic tangent-based *DT navigation* treats each obstacle entity as an *obstacle-valued function*, which returns an obstacle representation: For each agent A, at each timestep in computing navigation, each perceived obstacle entity (e.g., a wall, a polygon, another navigating agent) is represented by a dynamic tangent obstacle representation; each obstacle representation is mathematically modeled as an *angular range of repulsion*—or, alternatively, as the part of the obstacle entity within that angular range from which A is repelled. Hence, unlike other approaches in which only two levels of information are reflected in obstacle modeling, DT navigation employs a three-tiered structure:

1. the *obstacle entity*—the full geometric shape of the entity in the environment;
2. the *obstacle representation*—the DT representation abstracted from the obstacle entity, i.e., the locally usable geometry upon which a repeller form is based;
3. and the *repeller representation*—the mathematical function encoding the angular repulsion ascribed to the obstacle entity.

The additional level of abstraction in this three-level structure and the geometric details of our DT representations are the primary contributions of this paper. The mathematical functions for the repeller representations in our DT navigation are similar to those of a standard mathematical form described in [2,5,8], and based on only three arguments: the minimum distance from agent A to a nearest point p_m on an obstacle entity; the difference $\phi - \psi$ between heading angle ϕ of A and angle ψ from A to p_m; and an angular range of repulsion $\Delta\psi$, which controls the spectrum of heading angles from which the obstacle will repel the agent. The resulting DT representations are effective and efficient, and they satisfy desirable properties of obstacle representations—such as *proximity*, *symmetry*, and *necessary and sufficient repulsion*—that underlie successful navigation in other methods (see Section 4).

Indeed, as detailed in Section 3, DT representations enable exceptionally effective, geometrically sensitive navigation without pre-computation or deliberation. Even in simulations of non-holonomic agents (e.g., robots) moving at constant speed, DT representations result in smooth, successful navigation in scenarios with complicated paths, moving walls, or curved shapes such as circular or figure-eight shaped tracks.

2 Repellers

The repellers underlying our DT representations are based on the same mathematics as in [5,7], although our repellers are capable of expressing a wider range of configurations. In this section, we describe these repellers, summarizing previous presentations of the underlying mathematics and emphasizing particulars of our design.

In broad terms, DT navigation is an application of dynamical systems-based behavior modeling [8]. For this paper, and consistent with related papers [5,7], agents are circular and agent velocity is constant throughout navigation—obstacle avoidance and target seeking arise only from the dynamical systems governing heading angle ϕ. As noted in [2] and elsewhere, velocity could be autonomously controlled, but our present method shows the effectiveness of DT representations even without velocity control.

The repellers themselves in DT navigation are angular repeller functions, dynamically associated with obstacles based on local perception, without pre-computation. Most of the mathematical ideas underlying these functions remain unchanged from [5], thus retaining strengths such as *competitive behavioral dynamics* and some resistance to problems such as local minima. (See [5,7,8] for details about strengths of this particular repeller design.) Conventionally, with these repellers, every obstacle entity is represented using only circles, and circles' radii determine the associated repeller representations; in the underlying mathematics [5], however, repellers do not actually depend on circles' radii, but only on angular ranges of repulsion. Previous presentations do not emphasize that these angular ranges need not be derived from radii, nor that requiring a circle (thus a radius $r \geq 0$) for the repellers can restrict the expressiveness of obstacle representations: A point obstacle at the intersection of the dotted lines in Figure 1, for instance, would have the same angular range of repulsion as obstacle obs_i; because no entity could be smaller than a point, no smaller range could occur from an obstacle

at that location. Smaller ranges, however, could be productively employed by variants of these previous techniques, for more flexible and sensitive navigation dynamics. In this section, we describe our repellers, which are based on angular ranges, and in Section 3, we describe the particular angular ranges calculated for DT navigation.

To briefly summarize work from [7] and other papers, the evolution of agent heading angle ϕ during navigation is determined by angular repellers and angular attractors in a dynamical system of the form

$$\dot{\phi} = |w_{tar}|f_{tar} + |w_{obs}|f_{obs} + \textit{noise}, \tag{1}$$

where $\dot{\phi}$ is the time derivative of ϕ (by convention, dotted variables are time derivatives), f_{tar} and f_{obs} are functions representing targets and obstacles, respectively—the contributions of attractors and repellers to agent steering—and w_{tar} and w_{obs} are weight functions for each term. (The *noise* term prevents undesired fixed points in the dynamics.) Thus, at each timestep, $\dot{\phi}$ is calculated for steering at that moment, naturally and reactively accommodating new percepts or changes in the environment. A target is represented by a simple sine function:

$$f_{tar} = -a\sin(\phi - \psi_{tar}). \tag{2}$$

This function induces a clockwise change in heading direction when ϕ is counter-clockwise with respect to the target (and symmetrically, counter-clockwise attraction when ϕ is clockwise), thus attracting an agent.

Obstacle functions are more complicated, encoding *windowed repulsion* scaled by distance, so that more distant repellers have weaker effects than closer ones, and repellers do not affect agents already on collision-free paths. (Full details are in [7], which we only concisely summarize here.) For an obstacle obs_i, the angular repeller in DT navigation is the product of three functions:

$$R_i = \frac{\phi - \psi_i}{\Delta\psi_i}\, e^{1-|\frac{\phi-\psi_i}{\Delta\psi_i}|} \tag{3}$$

$$W_i = \frac{\tanh(h_1(\cos(\phi-\psi_i)-\cos(\Delta\psi_i+\sigma)))+1}{2} \tag{4}$$

$$D_i = e^{-\frac{d_m}{d_0}}. \tag{5}$$

Function R_i is an angular repeller with angular width $\Delta\psi_i$, centered around heading-angle value ψ_i (see Figure 1); windowing function W_i limits repulsion to have significant effects only within $\Delta\psi_i$ (plus a safety margin σ) from ψ_i; and scaling function D_i limits the strength of the overall repulsion based on d_m, the minimum distance between the agent and the obstacle. (Designer-chosen constant d_0 is a scaling parameter for D_i.) Each repeller, then, is a product $f_{obs_i} = R_i \cdot W_i \cdot D_i$, and for navigation, contributions of individual repellers are summed to $f_{obs} = \sum_i f_{obs_i}$ and then combined with f_{tar} in the weighted sum of Equation 1 to control steering. The weights themselves are determined by a system of competitive dynamics for reactive behavior selection; full details are available in [5].

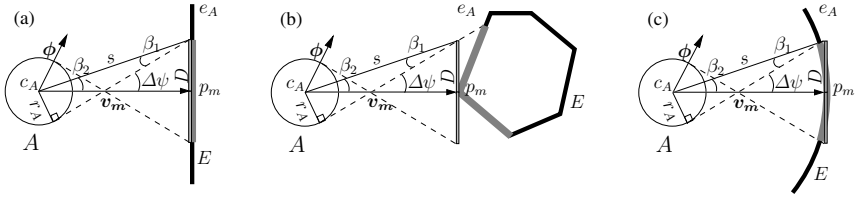

Fig. 2. Default, non-boundary case DT representations for various obstacle entity shapes. In all cases, $2\Delta\psi$ is chosen to be the angular range subtended by a segment of length $2D$ constructed around point p_m, perpendicular to vector $\boldsymbol{v_m}$.

3 Dynamic Tangent Obstacle Representations

Because of the additional level of abstraction for DT representations, the dynamic sensitivity of the repellers in Section 2 is enhanced by increased flexibility and local sensitivity to geometry, without deliberation or pre-computation. DT representations are constructed from locally relevant portions of obstacle entities' shapes, which for this paper are presumed to be always either straight lines or arcs of circles. Based on this, we consider three possible cases for any relevant component shape of an obstacle: *straight line*, *convex* non-line, or *concave* (i.e., a boundary of a linear, convex, or concave portion of the obstacle). Processes for DT construction are similar in each case, so we here present details of geometry applicable to all cases, with case-specific details in the following subsections.

In any of the cases, given agent A and obstacle entity E, the DT representation of E can be seen as the portion of E within a reactively calculated angular range of repulsion for E; in Figures (e.g., Figure 2), we conventionally indicate such a portion by thicker, lighter colored boundary lines on E. For this paper, the angular range (from A) defining that portion of E is the subtended range of a line segment, as shown in Figure 2; in general, as Figure 2 also suggests, this *DT segment* is locally tangent to E at a *projection* point p_m of minimal distance between A and E. More specifically, the segment is oriented perpendicular to the vector $\boldsymbol{v_m}$ that joins the center of A to p_m, and it is centered at p_m, extending a distance D in each direction, where D is an agent- or application-specific parameter. Parameter D thus determines angular range $\Delta\psi$—the DT segment represents an angular repeller of range $2\Delta\psi$, with range $\Delta\psi$ in each direction around $\boldsymbol{v_m}$ (see Figure 2). In examples in this paper, D is constant over a navigation (rather than, e.g., $\Delta\psi$ being constant over a navigation), so $\Delta\psi$ relates to $|\boldsymbol{v_m}|$ in desirable ways: For example, as A gets closer to E, the subtended range widens, resulting in greater repulsion.

In the default (i.e., non-boundary) conditions in each of the three cases, the angular range $2\Delta\psi$ subtended by the segment of length $2D$ around p_m can be found from elements labeled in Figure 2:

$$\Delta\psi = \beta_1 + \beta_2 = \sin^{-1}\left(\frac{r_A}{s}\right) + \sin^{-1}\left(\frac{D}{s}\right) \tag{6}$$

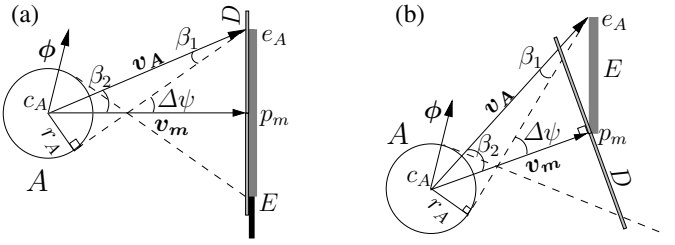

Fig. 3. Boundary case DT construction for a straight line obstacle entity. In the direction agent A is heading with respect to vector $\boldsymbol{v_m}$, the entity's subtended angular range is smaller than a standard DT representation's subtended angular range, so the range of the DT representation is modified to prevent repulsion from non-colliding paths.

As non-default, boundary cases, we consider instances where the angular range of the resulting repeller is wider than the angular range subtended by the original obstacle entity E, as shown in Figures 3–5. In these cases, to prevent needless repulsion from collision-free headings, DT representations have angular range exactly equal to that of E in *the direction the agent is headed* with respect to $\boldsymbol{v_m}$; the computations underlying these representations depend on the shape of E, as described below.

3.1 Straight Lines

When the obstacle entity E is a wall or some other straight line shape, finding p_m for a DT representation is straightforward: If there is a perpendicular from the center of agent A to E, then p_m is the intersection of E and the perpendicular; otherwise, p_m is the closest endpoint of E to A. In default cases, construction of the DT segment—a portion of E—and the resulting repeller is also straightforward.

In boundary cases, it is necessary to find the subtended range of E with respect to A in the direction that A is heading around E. For this, the process is the same whether $\boldsymbol{v_m} \perp E$ (Figure 3a) or not (Figure 3b). First, find endpoint e_A of E in the direction that A is headed. The appropriate angular range for repulsion is then given by:

$$\Delta\psi = \sin^{-1}\left(\frac{r_A}{|\boldsymbol{v_A}|}\right) + \cos^{-1}\left(\frac{|\boldsymbol{v_m}|^2 + |\boldsymbol{v_A}|^2 - |e_A - p_m|^2}{2|\boldsymbol{v_m}||\boldsymbol{v_A}|}\right) \tag{7}$$

3.2 Convex Shapes

For a convex chain of straight lines, finding p_m is again straightforward for agent A, and desired range $\Delta\psi$ can be computed similarly to the process in Subsection 3.1, using vertices for boundary cases (see Figure 4a). For a convex circular arc (Figure 4b), however, the subtended angular range is not necessarily defined by its endpoints. Consider such an arc to be defined by the radius r_o and center-point c_o of its defining circle C_o, as perceived by A, and by the visible angular range of C_o included in the arc, defined by angles θ_i and θ_f with respect to c_o and the positive x axis. Then, it is again straightforward to find closest point p_m to A, and in default cases, the DT segment and associated angular range follow immediately.

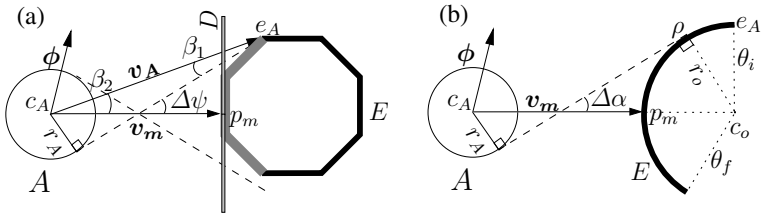

Fig. 4. Boundary case DT construction for convex shapes. For convex circular arcs (b), it must be determined if the subtended angle $\Delta\alpha$ between agent A and the entire circle C_o (of which the arc is a portion) is the angle $\Delta\psi$ from which the DT representation is derived.

In boundary cases, it remains to find the angle subtended by the arc, to compare to the angle subtended by the DT segment. To do this, we find endpoint e_A similarly to the straight line case, and we observe that the desired subtended angular range is limited by either e_A or by the point called ρ in Figure 4b, which defines (one side of) the subtended angular range $2\Delta\alpha$ between A and the entire circle C_o. The remainder of the DT construction then follows as before, with angular range of repulsion determined by either parameter D or the appropriate boundary condition described here.

3.3 Concave Shapes

Unlike the convex case, concave shapes bring safety concerns: Given non-holonomic agents with constant forward velocity, some environments cannot be navigated safely, such as a corner or box requiring sharper turns than motion constraints allow agents to make. In DT navigation, we prevent such difficulties by automatically approximating each concave corner by an arc with a radius large enough to be safely navigable: first, using properties of agent velocity and geometry, we calculate the agent's minimum radius for safe turns, r_{min}; then, when computing navigation, an unsafe corner is effectively modeled by a navigable arc, as in Figure 5a. (We also presume that all arcs in the environment have radius at least r_{min}, although tighter arcs could similarly be modeled by larger, navigable ones.) To approximate only the minimum amount necessary, DT representations treat the corner as if an arc of radius r_{min} were placed tangent to the lines forming the corner, as in Figure 5a; after finding half of the angle formed by the corner, θ_c, the distance d_c from the corner at which the arc should begin is $d_c = \frac{r_{min}}{\tan(\theta_c)}$. It is straightforward to find point p_m and manage boundary cases with endpoints of the chain, thus completing the definition of the representation.

For concave arcs, procedures are similar but complementary to those for convex arcs. If A is not located between c_o (the center of the circle from which the arc is derived) and the arc itself, closest point p_m to agent A is an endpoint of the arc; otherwise, p_m is the point on the arc in the same direction from c_o as c_A. Endpoint e_A is found by following the arc clockwise from p_m if agent heading direction is clockwise from p_m, and counter-clockwise otherwise (see Figure 5b). For boundary cases, the subtended angular range of a concave arc is always determined by endpoint e_A, and can thus be found similarly to previous cases.

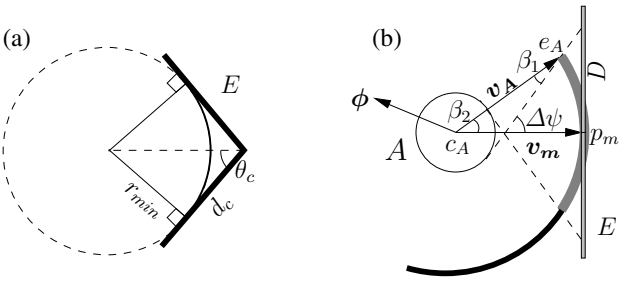

Fig. 5. Concave corner and arc shapes, with labeled features pertinent to DT computation: (a) approximating a corner by an arc; (b) finding a subtended angle

4 Properties of Obstacle Representations

As part of designing DT representations, we identified several desirable properties that reactive obstacle representations could possess, properties that clearly motivated previous work such as [5,7]. As part of evaluating DT representations, we present our list of these properties and very briefly describe how DT representations satisfy them.

Proximity and Symmetry. The focal point for computing repulsion is a point p_m on obstacle entity E of minimal distance from agent A. Thus, the nearest point on E to A—with which A might in principle collide soonest—is also the nearest point of the obstacle representation of E to A, enabling appropriate distance-based effects of repulsion. Furthermore, repulsion is centered around p_m and associated vector $\boldsymbol{v_m}$ (see Figures 1 and 2), so the reactive, local obstacle representation aptly does not determine in which general direction A heads around E—the representation steers A around E in the direction A was already heading, symmetrically around $\boldsymbol{v_m}$, regardless of the heading of A.

Necessary and sufficient repulsion. The repulsive range of the obstacle representation corresponds to exactly the heading angles along which the agent would collide with the obstacle entity.

Reactivity. Obstacle avoidance dynamically applies to both stationary and moving obstacles, without pre-computation or non-local knowledge.

Mathematical parsimony. Each obstacle entity is represented by a single repeller, neither overloading agent computations nor requiring needless mathematical machinations. This enables straightforward utility in a range of scenarios.

Previous obstacle representations that satisfy these properties were effective only in substantially restricted environments, i.e., consisting of only circular obstacles (see, e.g., Figure 1). Our DT representations are far more flexible, also applying to non-circular obstacles, and they directly satisfy properties of reactivity, mathematical parsimony, proximity, and symmetry, as well as a relaxed sense of necessary and sufficient repulsion: Because DT-based repellers are bounded by the maximum angular range of the obstacle entity remaining in the direction A is heading (toward endpoint e_A), no collision-free paths in that direction are repelled. Furthermore, due to proximity and

symmetry properties, each time repulsion is calculated, any possible collision point would be at or very near p_m. Thus, for large enough D with respect to agent size and navigation calculations, any colliding path would be within the range of repulsion of a DT-based repeller, and no collision-free path would ever be in that range.

5 Demonstrations

To test DT navigation, we created a simple, OpenGL-based simulator and simulated navigation in several scenarios. For each agent A, obstacle boundaries were locally perceived, and perception was straightforwardly implemented so that portions of obstacles occluded by other obstacles were not perceived, but other entities were perceived with unlimited range in all directions. Default parameter values for the repellers of Section 2 and [5,7] were $d_0 = 2.0$, $\sigma = 0.4$, and $D = 4r_A$ unless otherwise noted, and navigation was in a 12 by 12 unit world, with velocity held constant at 0.3, isolating heading angle ϕ as the only behavioral variable governing navigation, as in [5,7]. Tests were of two general kinds: basic testing to calibrate the value of D and establish general DT effectiveness in common scenarios; and testing in complex environments.

5.1 Basic Testing and Calibration

The default value $D = 4r_A$ in our demonstrations was chosen after experimentally determining the effect of D on navigation. In general, greater values of D lead to repellers with greater angular ranges of repulsion, but for an obstacle entity that subtends a large angular range, it is not always desirable for a repeller to subtend that entire range. For example, such large repellers can preclude the *boundary-proximate* navigation (Figure 6) on which this paper is focused, as discussed in Section 1. *Boundary-distant* navigation, in contrast, can be supported by such large repellers, but boundary-distant navigation can also be readily supported by appropriate DT representations (Figure 7). The local sensitivity enabled by DT representations, however, is not fully exploited in boundary-distant applications.

For finer-tuned, boundary-proximate navigation, we first calibrated D for appropriate sensitivity in DT representations. To do this, we ran experiments with a single agent A navigating along a wall, which indicated that distance d_m of agent A from the wall systematically varied with the value of D. We also considered a thought experiment—i.e., among many differences between an elephant and an ant, they maintain different safety margins when walking along a wall—and thus selected an agent size-dependent value of $D = 4r_A$, where r_A is the radius of agent A; this results in a d_m of between one and two radii for agents, which seems safe but not excessive. We then tested DT navigation in the basic scenarios shown in Figure 6, each with a convex or concave obstacle. In each scenario, agents started at 100 randomly selected positions spanning the left sides of their worlds, and DT navigation achieved perfect performance: Every agent reached its target without collision.

5.2 Complicated Environments

We also tested agents in more complicated environments, as shown in Figure 9. In the Hallways scenario, approximating an indoor layout with 3×2-sized office-obstacles

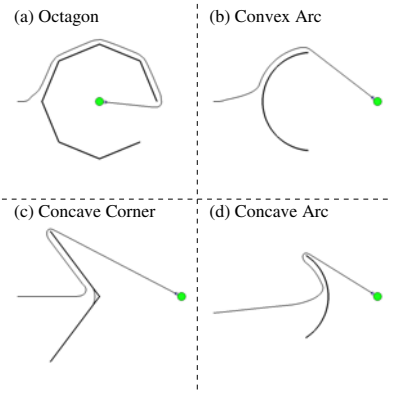

Fig. 6. Basic scenarios for demonstrations of purely reactive boundary-proximate navigation. Each image contains an obstacle entity, a target (green circle), and a sample trajectory.

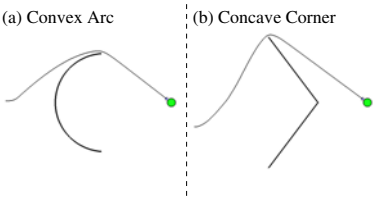

Fig. 7. Demonstrations of DT-based boundary-distant navigation. Agents started out facing the target but turned quickly, taking a smooth, efficient path to the target.

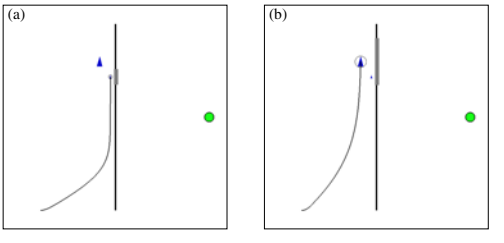

Fig. 8. Two different-sized agents, sizes $r_A = 0.1$ and 0.3, reaching parallel paths along a wall, each from a setting of $D = 4r_A$. Figures show the target locations (green circles), trajectories, and DT representations of the wall for each.

(the inner rectangles) and hallway width roughly 10-to-20 times r_A, agents navigated from 100 starting positions in the left of their world to a sequence of five target locations (Figure 9a), requiring extensive navigation and turning. Purely reactive DT navigation performance was perfect in all tested variants of this hallway scenario, including versions with additional circular obstacles, stationary or moving, in hallways.

The Polygons scenario (Figure 9b) incorporates navigation around a moving wall, which rotates in the center of the space, and a variety of convex polygons. In these experiments, agents navigated to five target locations (similar to those in the Hallways scenario), requiring a full traversal of the horizontal space; because of the additional difficulty posed by this scenario, the values of d_0 and σ were raised to 2.25 and 0.6, for repulsion at greater distances. Tests of DT navigation showed very good performance: Of 100 agents tested, starting from positions spanning the left side and top of this environment, 99 reached all targets without colliding. (Avoiding the moving wall proved difficult, perhaps due to the restriction to constant velocity.)

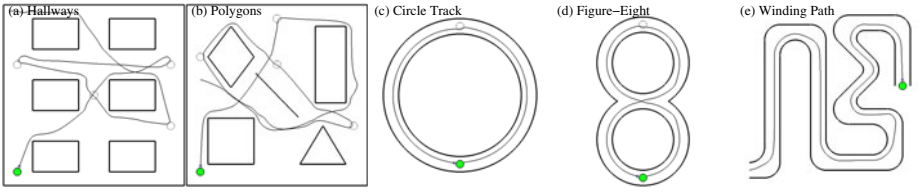

Fig. 9. Different scenarios in which DT navigation was tested, showing target locations and an example agent trajectory in each: (a) Hallways; (b) Polygons; (c) Circle Track; (d) Figure-Eight; (e) Winding Path

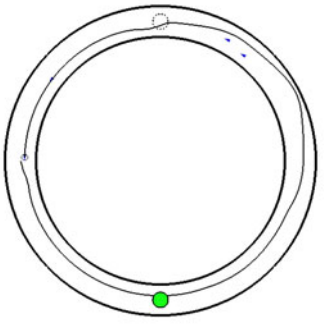

Fig. 10. A race-like run in the Circle Track scenario, including target locations and a trajectory of a fast agent that steered around slower agents

The remaining tests in complex environments were performed in scenarios with curved shapes (Figure 9c–e): a Circle Track; a Figure-Eight; and a Winding Path. The Winding Path scenario illustrates how even purely reactive DT navigation can succeed along a very complicated path, and the Figure-Eight scenario shows successful navigation in a closely bounded, curved environment. In the Figure-Eight and Circle Track scenarios, targets were alternatingly placed on the top and bottom of the tracks (indicated in Figure 9), to keep agents looping around the tracks. Race-like demonstrations were also run on the Circle Track (Figure 10), with up to four agents at different speeds, all successfully avoiding each other and the boundaries of the track while running.

6 Conclusion

This paper presents a new, *dynamic tangent*-based navigation method, which treats obstacle entities as *obstacle-valued functions*: Each agent represents each obstacle as an angular repeller, dynamically adjusted during navigation to support successful performance. The *obstacle representation* level of abstraction enables enhanced geometric sensitivity while retaining desired properties of obstacle representations. Simulations demonstrate that DT navigation is successful even in applications where agents must navigate closely around obstacle shapes and scenarios with a moving wall or complicated environments requiring circular or winding paths. DT representations might also

be effective in a wider range of environments if based on context-dependent variations in the value of D or with learning-based adaptations; the fact that DT representations require so few parameters may facilitate developmental or learning-based approaches.

Acknowledgments. The authors thank Clare Bates Congdon and anonymous referees for comments on previous versions of this paper.

References

1. Easton, K., Burdick, J.: A coverage algorithm for multi-robot boundary inspection. In: Int. Conf. Robotics and Automation, pp. 727–734 (2005)
2. Goldenstein, S., Karavelas, M., Metaxas, D., Guibas, L., Aaron, E., Goswami, A.: Scalable nonlinear dynamical systems for agent steering and crowd simulation. Computers and Graphics 25(6), 983–998 (2001)
3. Huang, W., Fajen, B., Fink, J., Warren, W.: Visual navigation and obstacle avoidance using a steering potential function. Robotics and Autonomous Systems 54(4), 288–299 (2006)
4. Khatib, O.: Real-time obstacle avoidance for manipulators and mobile robots. Int. Journal of Robotics Research 5(1), 90–98 (1986)
5. Large, E., Christensen, H., Bajcsy, R.: Scaling the dynamic approach to path planning and control: Competition among behavioral constraints. International Journal of Robotics Research 18(1), 37–58 (1999)
6. Paris, S., Pettré, J., Donikian, S.: Pedestrian reactive navigation for crowd simulation: A predictive approach. Computer Graphics Forum 26(3) (2007)
7. Schöner, G., Dose, M.: A dynamical systems approach to task-level system integration used to plan and control autonomous vehicle motion. Robotics and Autonomous Systems 10(4), 253–267 (1992)
8. Schöner, G., Dose, M., Engels, C.: Dynamics of behavior: Theory and applications for autonomous robot architectures. Robotics and Autonomous Systems 16(2-4), 213–245 (1995)
9. Shao, W., Terzopoulos, D.: Autonomous pedestrians. Graphical Models 69(5-6), 246–274 (2007)
10. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. ACM Trans. on Graphics 25(3), 1160–1168 (2006)