Improved Obstacle Relevancy, Distance, and Angle for Crowds Constrained to Arbitrary Manifolds in 3D Space

Brian C. Ricks and Parris K. Egbert

Department of Computer Science
Brigham Young University
Provo, Utah, USA

Abstract

Recent work has proposed crowd simulation algorithms on arbitrary manifolds in 3D space. These algorithms simulate crowds on far more realistic surfaces than previously possible, including multi-story structures, science fiction scenarios, and habitats for insects and other animals that can walk on walls. However, current implementations can have distinct artifacts, including collision false positives and false negatives. Also, current implementations fail to account for the cylindrical shape of the characters being simulated. The resulting crowds move unnaturally and have obvious collisions. After identifying the cause of these artifacts, we propose an algorithm that does not struggle from these false positives or false negatives and correctly accounts for the non-spherical shape of agents. The resulting crowds move on large surfaces (over 100k triangles) running with a thousand agents in real-time.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation; Simulation and modeling [I.6.8]: Types of Simulation—Animation

1. Introduction

Crowd simulation plays a crucial part in populating virtual worlds, whether for special effects in film, interactive settings in games, or planning with architectural tools. However, most crowd simulation algorithms focus on 2D planes instead of arbitrary surfaces. Some algorithms try to simulate crowds on arbitrary surfaces, like Torchelsen et al. [TSO*10], but these algorithms struggle with either false negatives or false positives and fail to handle the non-spherical shape of agents. We propose an improved relevancy algorithm for crowd simulation on non-planar surfaces that resolves these issues.

At the heart of crowd simulation algorithms is a function that takes each agent and finds the best heading and velocity for collision-free movement. If we call this function *ObstacleAvoidance*, then this function can be written as:

$$ObstacleAvoidance : R \in (\theta \times d), a \rightarrow \theta', v'$$

Where R is a set of tuples in $(\theta \times d)$ (where θ is the angle to a nearby obstacle and d is the distance to that obstacle), and a is the location and orientation of an agent. *ObstacleAvoidance* uses the heading and distance to obstacles to choose a change in motion for collision free movement. This change in motion is represented by the outputs of *ObstacleAvoidance*, with θ' and v' representing the agent's change in angle and velocity respectively.

The set of algorithms that implement the function *ObstacleAvoidance* (or a very similar function) includes social forces [HM95], RVO [VdBLM08], HiDAC [PAB07], and anticipation [OPOD10]. Finding the correct angle and distance values for the set R for crowds on a 2D plane is straightforward, but for crowds on arbitrary manifolds in 3D space, finding R is quite difficult.

Current techniques that work on arbitrary surfaces suffer with clear artifacting that limits their applicability. Additionally, these algorithms fail to account for the non-spherical shape of characters, leading to obvious agent collisions. To facilitate crowd simulation on arbitrary manifolds in 3D space for film, games, and architecture, we propose an algorithm for finding R in real-time that leads to natural, collision-free movement and that lacks the artifacts of previous work.

DOI: 10.2312/conf/EG2012/short/073-076

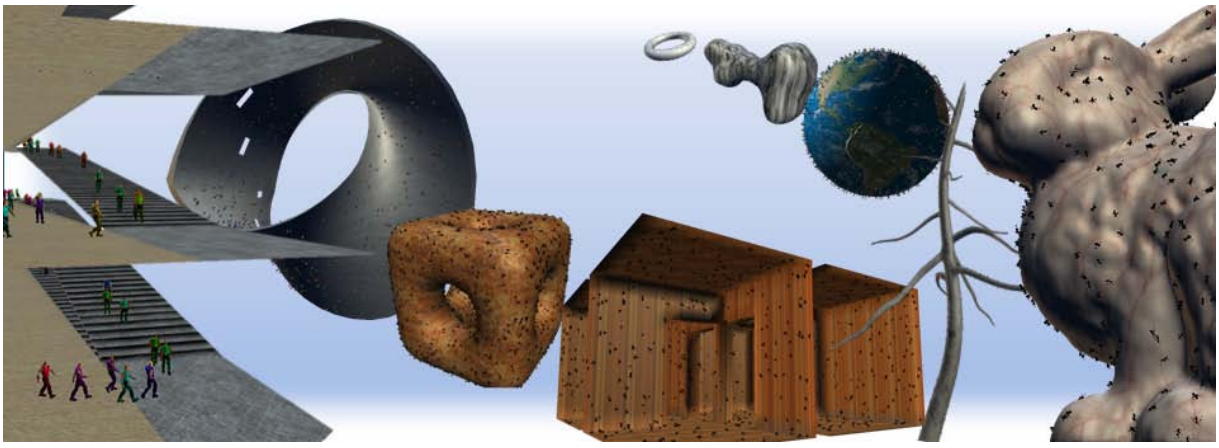

Figure 1: *3D Surfaces where our algorithm produces natural crowds, eliminating the artifacts of previous work. Stanford bunny data courtesy the Stanford 3D scanning repository.*

2. Artifacts in Current Approaches

In order to use an implementation of *ObstacleAvoidance* on arbitrary surfaces in 3D space, a crowd simulation algorithm needs two functions, *Relevant* and $Offset$, to choose obstacles that are relevant and to determine the angle and distance to those relevant obstacles.

The *Relevant* function quickly finds a subset of all the obstacles in a scene that are relevant to the current agent. Formally, *Relevant* can be defined as:

$$Relevant : a, O \rightarrow O_{Relevant} \in O \tag{1}$$

where a is the agent in question, O is the set of all obstacles, and the result $O_{Relevant} \in O$ gives us obstacles that are relevant to *ObstacleAvoidance*. Without a *Relevant* function, crowd simulation algorithms can be $O(n^2)$ since the angle and distance between each pair of agents must be calculated. A constant-time *Relevant* function that returns a maximum number of obstacles reduces this to $O(n)$ since each agent only uses the angle and distance to a fixed number of agents in its *ObstacleAvoidance* function. A classic example of a *Relevant* function is presented by Reynolds [Rey06] who proposed a grid structure for quickly finding obstacles around a given agent. (Note that although Reynolds' work has a *Relevant* function, the problem of flocking in 3D space is very different than our problem in the domain of crowds constrained to arbitrary manifolds.)

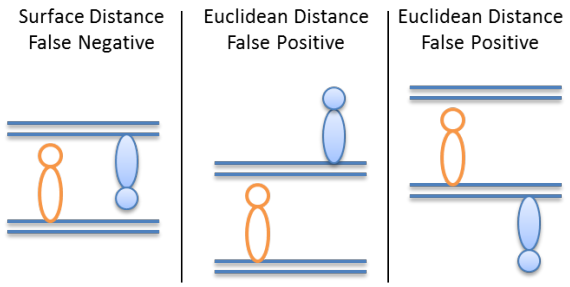

Figure 2: *Examples of scenarios where simply using surface distance or Euclidean distance produces collision false positives or false negatives.*

Once the *Relevant* function has found a small subset of obstacles, the $Offset$ function loops over each obstacle in $O_{Relevant}$ and finds the angle and distance to the obstacle where Off is a set containing a tuple in the power set of angles and distances.

$$Offset : a, o \in O_{Relevant} \rightarrow Off \in (\theta \times d) \tag{2}$$

For this work we consider the output of $Offset$ to be a set of tuples even though most implementations of $Offest$ return a single tuple. Allowing $Offset$ to return a set of tuples will prove to be a key innovation in allowing our crowd simulation algorithm to remove the artifacts of previous work. Implementing $Offset$ on a 2D surface is simple to understand, but on arbitrary surfaces in 3D space it is far more complicated. There are two main approaches: surface offset and Euclidean offset, both of which struggle with clear artifacts.

One can implement $Offset$ using a surface path algorithm (A*, fast marching methods [Set99], etc.) and calculate the distance to an obstacle as the sum of each segment in the path and the angle as the angle of the first segment of the path. Unfortunately, this fails to produce artifact-free crowd simulation since agents can collide even when their surface distance is extremely high. Figure 2 depicts an agent in question (the orange agent on the floor) and the obstacle agent (the blue agent on the ceiling). The orange agent is at most a few meters from the blue agent, but the surface distance is the distance from the orange agent to the nearest wall, up

the wall, and back across the ceiling. Even with an accurate surface distance algorithm, we would incorrectly think that the orange and blue agents are not about to collide even though they really are. The angle results of a surface path implementation of $Offset$ will produce similar artifacts.

The other method for determining angles and distances is to use a Euclidean measure (similar to Torchelsen et al. [TSO*10]). Given a vector $v_{obstacle}$ that is the vector between the agent in question a and $obstacle$, Euclidean distance is the length of $v_{obstacle}$. Euclidean angle is found by projecting $v_{obstacle}$ onto the plane tangent to a. Torchelsen et al. accurately point out that Euclidean offset does not struggle with the false negative issues of a surface distance algorithm. However, instead of suffering from false negatives like the surface implementations, a Euclidean implementation of $Offset$ suffers from the opposite problem: collision false positives. Consider the center and right examples in Figure 2. In both cases a Euclidean distance approach will inaccurately report that the orange agent is about to collide with the blue agent, and the agents will evade each other even though there is no imminent collision. This unnatural effect is very obvious in the most common crowd simulation environment on arbitrary surfaces: multi-story buildings.

Neither the surface path nor the Euclidean implementation of $Offset$ address the last issue with crowds constrained to arbitrary surfaces in 3D space: the non-spherical shape of virtual characters. On arbitrary surfaces in 3D space, two agents can be on surfaces that are perpendicular to each other (see the right-most image in Figure 4). If the crowd simulation algorithm only represents characters with a small sphere near each agent's feet, the heads of the two agents may be on a collision course even if the spheres are not. Enlarging the spheres removes this problem but makes each person have an enormous personal space bubble, causing jams even when agents could easily pass each other.

3. Removing Artifacts

We propose key improvements to the *Relevant* and $Offset$ functions that significantly enhance the believability of crowd simulation of surfaces in 3D space. Instead of combining both a surface path and Euclidean implementation into $Offset$, we have found that a far easier solution lies in an improved *Relevant* function. Notice that in all false positive cases (see Figure 2), the obstacle agent that the relevant agent incorrectly avoids is not visible to the relevant agent. Thus, by improving *Relevant* to discard agents that are not visible to the agent in question, a Euclidean implementation of $Offset$ would not have false positives and the resulting unnatural motion.

A full synthetic vision algorithm could be used to determine which agents are mutually visible, but we have been unable to implement such an algorithm that runs at real-time speeds. Instead, we propose leveraging what we know about

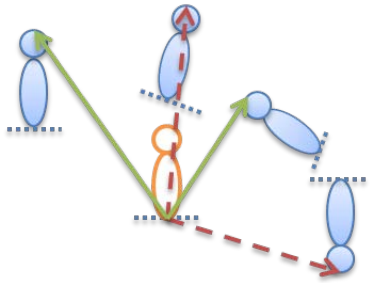

Figure 3: *Our improvement to the Relevant function. The relevant agent (orange) checks if the vector to possible obstacle agents penetrates an approximate surface at each agent's feet (dotted blue line). If the vector penetrates these surfaces (dashed red arrow) the obstacle agent is not considered relevant. If the vector does not penetrate (lighter green arrow) the agent is considered relevant.*

the position of agents to make a close approximation to their visibility (see Figure 3). The key fact is that since each agent is constrained to a manifold, we know there is a surface at the feet of each agent. Using this fact, we add two additional checks to our *Relevant* function for each obstacle agent. We find the vector from the agent in question to the head of the agent which is proposed as an obstacle. We call this vector v_{head}. We then assume that the surface the obstacle agent is standing on can be locally approximated by a one meter radius circle at its feet and that the normal of this circle is the same as the obstacle agent's normal. If v_{head} penetrates this circle, then we remove this agent from the set $O_{Relevant}$. Likewise, we assume the agent in question is standing on a locally flat surface and check to see if v_{head} penetrates this agent's surface on the way to the obstacle agent. This approximation is not perfect, but in practice there are no false positives or false negatives in the crowd movement. As we discuss in our results, our crowds are significantly more realistic than those produced by previous work.

This improvement to *Relevant* does not handle the non-spherical nature of agents. To handle this additional complication and produce very realistic crowd motion, we further improve the definition of $Offset$ to account for the non-spherical nature of agents. Instead of one small sphere or one giant sphere, we define each character with three spheres stacked in a way that approximates a cylinder. For the center of each of these spheres, we use a traditional implementation of $Offset$ and return the union of tuples. Formally, if o_{up} is the normal of the obstacle agent for which we want a more realistic representation, then our implementation of $Offset$ is as follows:

$$Offset_{Improved} = \bigcup_{i=0}^{2} Offset(a, o + i \cdot o_{up})$$

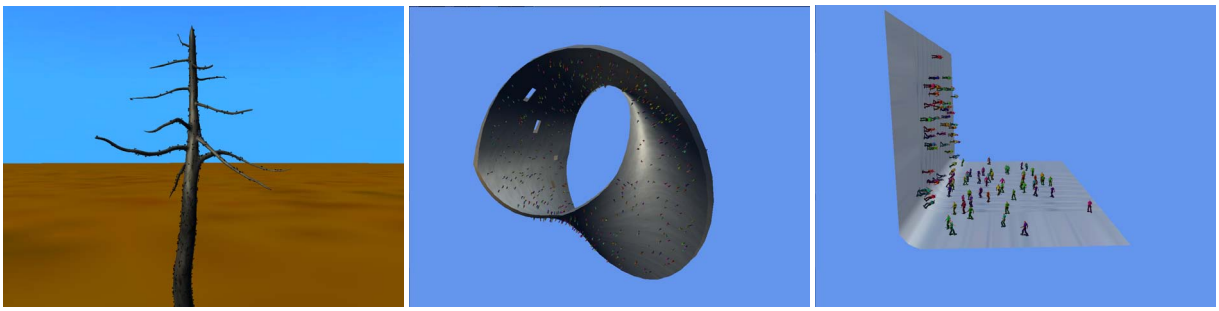

Figure 4: *Examples of where improved distance and angle measures are essential for accurate movement, including ants on thin tree branches, humans on a Mobius strip, and science fiction characters walking on walls.*

Figure 5: *An example surface that brings agents close together on the ceiling and floor. Without an accurate distance and angle function, agents will run into each other.*

These improvements have a minuscule computational footprint, have a clear impact on the motion of the agents in a crowd, and can handle intentionally complex scenes like that in Figure 5.

4. Results

For our tests we generated crowds of 1,000 agents across an array of surfaces inspired by architecture, nature, outer space, insect habitats, and topologically unique surfaces (as seen in Figure 1). These models range from a dozen triangles to over a hundred thousand triangles, which surpasses the highest triangle count model used in Torchelsen et al. [TSO*10] and which we believe is the highest triangle count model reported in the crowd simulation literature.

We compared our crowds against a surface path-based algorithm and a Euclidean algorithm similar to Torchelsen et al. [TSO*10] and found our crowds moved noticeably smoother than previous work. For our quantitative tests, we calculated collision percentages using social forces [HM95] and reciprocal velocity obstacles [VdBLM08]. The median collision rates were very low: .05% and .009% respectively. We also calculated the speed of our algorithm. In all our meshes, including the Stanford Bunny and an abstract art piece with over 100k traingles, our algorithm ran at over 33 frames-per-second.

References

[HM95] HELBING D., MOLNAR P.: Social force model for pedestrian dynamics. *Physical review 51*, 5 (1995). 2, 5

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG) 29*, 4 (2010). 2

[PAB07] PELECHANO N., ALLBECK J., BADLER N.: Controlling individual agents in high-density crowd simulation. *ACM SIGGRAPH/Eurographics symposium on Computer animation* (2007), 99–108. 2

[Rey06] REYNOLDS C.: Big fast crowds on ps3. *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames* (2006), 113–121. 3

[Set99] SETHIAN J.: Fast marching methods. *SIAM review* (1999), 199–235. 3

[TSO*10] TORCHELSEN R., SCHEIDEGGER L., OLIVEIRA G., BASTOS R., COMBA J.: Real-time multi-agent path planning on arbitrary surfaces. *ACM SIGGRAPH symposium on Interactive 3D Graphics and Games* (2010), 47–54. 2, 4, 5

[VdBLM08] VAN DEN BERG J., LIN M., MANOCHA D.: Reciprocal velocity obstacles for real-time multi-agent navigation. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on* (2008), 1928–1935. 2, 5