

Optimal Acceleration Thresholds for Nonholonomic Agents

Brian C. Ricks · Parris K. Egbert

Abstract Finding optimal trajectories for non-accelerating, nonholonomic agents is a well understood problem. However, in video games, robotics, and crowd simulations nonholonomic agents start and stop frequently. With the vision of improving crowd simulation, we find optimal paths for virtual agents accelerating from a standstill. These paths are designed for the “ideal”, initial stage of planning when obstacles are ignored. We analytically derive paths and arrival times using arbitrary acceleration angle thresholds. We use these paths and arrival times to find an agent’s optimal ideal path. We then numerically calculate the decision surface that can be used by an application at run-time to quickly choose the optimal path. Finally we use quantitative error analysis to validate the accuracy of our approach.

Keywords Nonholonomic agents · Optimization · Crowd simulation

1 Introduction

The fields of computer games, robotics, and simulation often use point-based, virtual agents to represent real entities. Frequently, these agents are nonholonomic, meaning they accelerate in the direction they are facing (see [2]). For nonholonomic agents that cannot accelerate, it has been shown that optimal trajectories are composed of lines and circular arcs [5].

However, it is non-trivial to find optimal paths for accelerating nonholonomic agents. Consider the super-

imposed paths in Fig. 1. All agents started at the origin looking right. The red agent turns in place until it sees its destination before accelerating. The green agent turns until there are $\pi/2$ radians between its heading and the destination before accelerating. The blue agent accelerates without turning in place. For this particular destination, $(-8, 0)$, the green agent arrives first, the blue agent arrives second, and the red agent arrives last. However, there exist destinations where the red agent would be the fastest and others where the blue agent would be the fastest. As we demonstrate later, there is no trivial method for choosing the shortest path.

In order to improve nonholonomic agent movement, we:

1. We derive equations for arrival time and paths for agents with arbitrary destinations. We do this for nonholonomic agents starting from a standstill.
2. We present an off-line method for numerically calculating a decision surface that finds the optimal path to any destination.
3. We find a close approximation to this surface that can be used at run time to find the optimal path for nonholonomic agents.

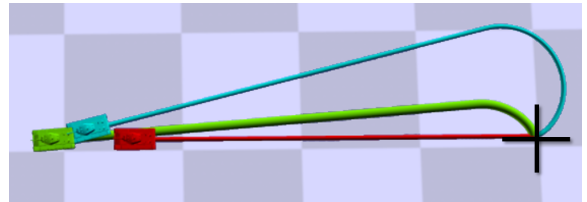


Fig. 1 Superimposed paths of agents using different acceleration angle thresholds. Agents started at the origin (right) looking right. A threshold of $\pi/2$ (green) reaches the destination first, followed by π (blue), followed by 0 (red, who reaches the destination 1.33 seconds after the first agent). An agent with a threshold of ~ 2 radians is optimal here for this destination.

B. Ricks
CCICADA Center, Rutgers University
E-mail: brian.ricks@rutgers.edu

P. Egbert
Brigham Young University
E-mail: egbert@cs.byu.edu

We provide rigorous validation for these contributions. Combined, we believe this will lead to meaningful reductions in agent travel time and computation time for games, robotics, and crowd simulations.

2 Previous Work

The most relevant previous work in terms of our results can be divided into three main parts: modeling human motion, optimal trajectories, and collision avoidance.

2.1 Reproducing Human Trajectories

One branch of virtual agent research finds patterns in human movement and reproduces them in virtual agents. The hope is that believable agent motion can be created by reproducing these patterns. This research includes studies on how we move our eyes [7], hands [10, 7], and arms [15]. Often it is assumed that humans optimize their movement over some function [1], including minimizing variance [7] or maximize smoothness [15]. Researchers do the same analysis on human trajectories. Arechavaleta et al. [2] asserted that the movement of humans can be modeled nonholonomically while minimizing the L_2 norm. Hicheur et al. [9] argued that locomotion planning is done at a higher level than footstep planning. Thus they assert that people plan trajectories first and footfalls second.

This research has an appropriate place in virtual agent movement, but should not be the sole foundation for choosing paths for agents. Unlike the human subjects in these studies, virtual agents often start and stop frequently. Also, nonholonomic agents are also used to model non-human entities. In such cases, a more generic model is required.

2.2 Optimal Trajectories

Recognizing the importance of improved trajectories for nonholonomic agents, researchers have studied path planning from a mathematical perspective. One of the key themes in this area is optimizing with respect to an agent's arrival time. Researchers have looked at fastest paths for robots moving at constant speeds. These are called Dubins paths after the work of Dubin [5]. Cockayne and Hall found the set of all reachable points using Dubins paths in a fixed amount of time [4]. Soueres and Laumond have studied how to optimally link any two configurations [13]. However, similar to previous work, all agents travel at constant speeds.

Our work deviates from Dubins paths we do not assume that agents have a constant velocity. On the contrary, we are interested in the more complex scenario where agents can accelerate from a standstill. Thus we seek a more expressive set of proofs and algorithms for choosing paths for nonholonomic agents.

2.3 Collision Avoidance

A common use of nonholonomic agents is in crowd simulation and it is related to our current and future work.

Collision avoidance has roots in Reynolds' [12] instantaneous forces collision avoidance methods. Helbing and Molnár [8] proposed a 2D method called social forces. Fiorini and Shiller [6] introduced velocity obstacles and created very believable crowds with less stalling than when using social forces. This was further extended by van den Berg et al. [3]. Moving from velocity-based collision avoidance to accelerations-based collision avoidance, Ondrej et al. [11] proposed a synthetic-vision crowd simulation algorithm. They also assert agents movement can be judged on their arrival time to destinations. For a survey of this research area see [14].

Most crowd simulation algorithms avoid collisions by finding a collision-free velocity for each agent. When there are multiple options, algorithms choose a collision-free velocity that is closest to the agent's ideal path to its destination. This ideal path is usually the straightest line from the agent's current location to the destination without regard to obstacles. Crowd simulation is highly relevant to this paper since we show that a straight line from the agent to the destination is not always the optimal path for agents. Also, since our work has direct application to crowd simulation, we do calculations in a similar ideal, obstacle-free environment.

3 Problem Definition

A nonholonomic virtual agent's motion can be defined using a differential equation: $\dot{x} = v \cos \theta$, $\dot{y} = v \sin \theta$, $\dot{v} = a$. Without loss of generality, we initially place the origin about our agent, i.e., $x_0 = y_0 = 0$ and $\theta_0 = 0$. We also assume that $v_0 = 0$.

Using these values, the initial configuration of any agent can be uniquely defined by a 5-tuple: $\langle D_x, D_y, r_m, r'_m, \theta'_m \rangle$ where D_x and D_y are values of the destination relative to the agent. (Equally, we use polar coordinates D_r and D_θ to denote the destination.) Further, r_m is the agent's maximum velocity, r'_m is the agent's maximum acceleration, and θ'_m is the agent's rate of rotation. Without loss of generality, we assume

that D_y is positive, resulting in a counter-clockwise turn by the agent to reach its destination.

When an agent plans its path, it often starts by guessing the shortest path without respect to dynamic obstacles (consider velocity obstacles or social forces). Our work focuses on this first part of agent path planning in an ideal, obstacle-free environment. Specifically, we want to take out the guessing and put in provably best choice.

The path of an agent to its destination can be thought of as having three distinct steps. In the first step the agent turns in place at rate θ'_m without any linear velocity or acceleration. We call this the *turn in place* step. Next the agent accelerates linearly while turning. If the agent reaches its maximum linear velocity r_m during this phase, it continues to turn while moving forward at this velocity. We call this the *turning with velocity* step. Lastly, when the agent is looking directly at its destination, it stops turning and continues to accelerate (or if it has already reached r_m , it continues at that speed), until it reaches its destination. We call this the *straight motion* step.

It is simple for an algorithm to transition an agent from the turning with velocity step to the straight motion step: the agent need only stop turning when it faces the destination.

The moment at which an agent transitions between the turning in place and turning with velocity steps is chosen arbitrarily by the underlying motion algorithm. We call the angle between the agent's current heading and the agent's destination when it transitions between the first two steps the *acceleration angle threshold*.

As an example, consider an agent whose destination is at $(D_x, D_y) = (-8, 0)$ (see the green/thicker path in Fig. 1). This agent's acceleration angle threshold is $\pi/2$. Thus, the agent will turn in place without any linear velocity or acceleration until the difference between its heading and the destination is $\pi/2$. This happens when it is looking down the positive y-axis. As the agent accelerates and turns, it will move in an arc toward the destination. When the agent is looking directly at $(-8, 0)$, the agent will stop turning. It will then proceed with the straight motion step until it reaches its destination.

The goal of this work is to answer the following question: Given a nonholonomic agent, how far should the agent turn in place to reach its destination in the least amount of time? To help answer this question, we introduce the function $ArrivalT$, that gives the arrival time of an agent based on its choice of acceleration angle threshold. Formally $ArrivalT : at \in [0, \pi], D_x, D_y, r_m, r'_m, \theta'_m \rightarrow t$ where at is the chosen acceleration angle threshold and t is the amount of time

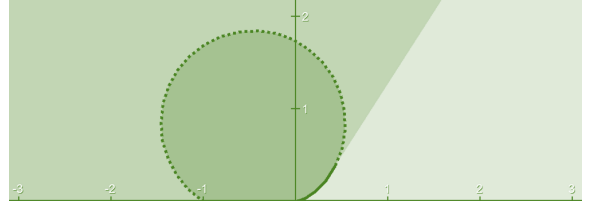


Fig. 2 Three surface subdivisions for $at = \pi$ (shown with $\theta'_m = 1, r_m = 1, r'_m = 1$). The light green and medium green areas show pre- and post-full velocity destinations respectively. The dark green area in the middle is unreachable by an infinitely small agent. The solid dark green line shows the trajectory of the agent up to time $t = \frac{r_m}{r'_m}$.

it takes an agent to reach its destination using that threshold. As we discuss later, not all angle thresholds will reach all destinations, in which case $ArrivalT$ is $t = +\infty$. We derive $ArrivalT$ in Sections 4 and 5.

We use $ArrivalT$ to find the at that results in the lowest arrival time (and hence the optimal path). To do this we use function $BestAT$. Formally, $BestAT : D_x, D_y, r_m, r'_m, \theta'_m \rightarrow at$ s.t. $\forall q \in [0, \pi], ArrivalT(at, D_x, D_y, r_m, r'_m, \theta'_m) \leq ArrivalT(q, D_x, D_y, r_m, r'_m, \theta'_m)$. We derive a very close approximation to $BestAT$ in Section 6.

4 Acceleration Angle Threshold = π

In order to derive the equation for $ArrivalT$ for an arbitrary at , we begin by solving $ArrivalT$ when $at = \pi$. We show in the next section that we can derive $ArrivalT$ for any at with only slight modification to the following equations.

Setting the acceleration angle threshold to π means an agent will never turn in place when heading toward a destination. Instead, since $D_\theta \leq \pi$, the agent will skip the turn in place step and immediately proceed to the turning with velocity step. The blue agent in Fig. 1 gives an example of paths generated by $at = \pi$.

4.1 Three Areas of the Surface

Deriving the equation for $ArrivalT$ for $at = \pi$ depends on whether or not an agent reaches full velocity before transitioning from the turning with velocity step to the straight motion step. We call destinations that an agent will look at directly before reaching full velocity *pre-full velocity destinations* and destinations that the agent will not see until after reaching full velocity *post-full velocity destinations*. We begin by discussing how to determine whether a point is a pre- or post-full velocity destination, then how to find the arrival time to each of these types of destinations in turn (see Fig. 2).

Until an agent reaches its maximum velocity, its acceleration is constant at r'_m and it will turn at a constant rate of θ'_m . Thus, the agent's velocity at time t is $r'_m \cdot t$ and its heading is $\theta'_m \cdot t$. Converting to x and y velocity:

$$x'(t) = \cos(\theta'_m \cdot t) \cdot r'_m \cdot t, \text{ and } y'(t) = \sin(\theta'_m \cdot t) \cdot r'_m \cdot t \quad (1)$$

Taking the definite integral between 0 and t :

$$\begin{aligned} x(t) &= \frac{\cos(\theta'_m \cdot t)}{\theta'^2_m} + \frac{t \cdot \sin(\theta'_m \cdot t)}{\theta'_m} - \frac{1}{\theta'^2_m} \\ y(t) &= \frac{\sin(\theta'_m \cdot t)}{\theta'^2_m} - \frac{t \cdot \cos(\theta'_m \cdot t)}{\theta'_m} \end{aligned} \quad (2)$$

We are interested in the position of the agent when it reaches its maximum velocity. This happens at time $t = \frac{r_m}{r'_m}$. We can find the location of the agent when it reaches its full velocity by replacing t with $\frac{r_m}{r'_m}$ in Eq. 2. We call this point the *full velocity point*, the x and y coordinates of which are:

$$\begin{aligned} FVX &= r'_m \cdot \left(\frac{\cos(\frac{r_m}{r'_m} \cdot \theta'_m)}{(\theta'^2_m)} + \frac{\frac{r_m}{r'_m} \cdot \sin(\frac{r_m}{r'_m} \cdot \theta'_m)}{\theta'_m} \right) - 1 \\ FVY &= r'_m \cdot \left(\frac{\sin(\frac{r_m}{r'_m} \cdot \theta'_m)}{(\theta'^2_m)} - \frac{\frac{r_m}{r'_m} \cdot \cos(\frac{r_m}{r'_m} \cdot \theta'_m)}{\theta'_m} \right) \end{aligned} \quad (3)$$

In order to determine which destinations are pre- and post-full velocity destinations, we can think of an agent walking from the origin to the full velocity point. Every point the agent looks at directly is a pre-full velocity destination. If we draw a line through the agent at time $\frac{\theta'_m \cdot r_m}{r'_m}$ parallel to the agent's heading, it is easy to prove that all points to the right of that line (or returning a negative distance) will be pre-velocity destinations. This is true since all these points are visible to the agent as it moves from the origin to the full velocity point. We thus use this line to segregate pre-full velocity destinations and post-full velocity destinations.

In order to find this segregating line we need a point and a tangent. We know the full velocity point is on this line and we derived it in Eq. 3. We also know the vector of the tangent of this segregating line, since it is the direction the agent is facing at time $\frac{r_m}{r'_m}$. The non-unit vector version of this tangent can be found by taking the velocity of the agent at time $\frac{r_m}{r'_m}$ using Eq. 1. If we drop the scaling factor in Eq. 1 we have the unit length tangent: $tangent = (\cos(\frac{r_m}{r'_m} \cdot \theta'_m), \sin(\frac{r_m}{r'_m} \cdot \theta'_m))$. An orthogonal vector to this is $(OrthX = -\sin(\frac{r_m}{r'_m} \cdot \theta'_m), OrthY = \cos(\frac{r_m}{r'_m} \cdot \theta'_m))$. Thus, the equation of the segregating line running through the agent at time $t = \frac{r_m}{r'_m}$ is:

$$-\sin\left(\frac{\theta'_m \cdot r_m}{r'_m}\right)x + \cos\left(\frac{\theta'_m \cdot r_m}{r'_m}\right)y + \frac{r_m \theta'_m - r'_m \sin\left(\frac{\theta'_m \cdot r_m}{r'_m}\right)}{r'_m \theta'^2_m} = 0 \quad (4)$$

To determine if a point is a pre-velocity destination, we can plug D_x and D_y into Eq. 4. If the result is less than 0, we know that the point is a pre-velocity destination. Interestingly, this line does not guarantee that

points on the other side are post-velocity destinations (as we discuss shortly). Additionally, the trajectory of the agent forms a curve below this dividing line that contains pre-velocity destinations. Using this segregating line, these points are incorrectly identified as post-full velocity points. However, as all these points have low D_θ and D_r , the arrival time to these points is low and this mislabeling is inconsequential (see Section 4.4).

ArrivalT can be found by first finding whether the destination is a pre- or a post-full velocity destination. Once we know which type of destination we have, we can use the appropriate equations to find arrival times. We first describe how to find *ArrivalT* for post-full velocity destinations and then for pre-full velocity destinations.

4.2 Post-Full Velocity Destinations

If we know a destination is a post-full velocity destination, we know the agent will reach the full velocity point before it transitions to the straight motion step (see Sec. 3). Since it takes $\frac{r_m}{r'_m}$ time for the agent to reach this point, we already know one part of the arrival time.

Once an agent reaches full velocity it stops accelerating and continues at the same speed while turning. Thus, the agent follows a perfect circle from the time it reaches full velocity to the time it is looking directly at its destination (see the dotted circle in Fig. 2). Finding the point where the agent is looking directly at the destination requires finding the equation for this circle and then finding the point on this circle where the agent changes from the turning with velocity step to the straight motion step. Since this is the point where the agent leaves the perfect circle trajectory, we call this the *departure point*.

To find the departure point, we first have to find the circle the agent follows after it reaches full velocity. Since the agent reaches its full velocity at the full velocity point, we know this point is on the circle. We can calculate the radius of this circle since it takes $\frac{2\pi}{\theta'_m}$ seconds for an agent to turn 2π radians, in which time the agent will go around this circle exactly once. Since the time to go around the circle once is $t = \frac{2\pi}{\theta'_m}$ and the agent's velocity is $v = r_m$, the agent will travel $\frac{2\pi}{\theta'_m} \cdot r_m = \frac{2\pi \cdot r_m}{\theta'_m}$ meters if it followed this circle all the way around. Thus, the circumference of the circle is $\frac{2\pi \cdot r_m}{\theta'_m}$. Using this, we can compute the radius of the circle: $\frac{2\pi \cdot r_m}{\theta'_m} = 2 \cdot \pi \text{Radius}$. It follows that $R = \frac{r_m}{\theta'_m}$.

We can find the center of this circle by moving in the opposite direction of $(OrthX, OrthY)$ a distance of R starting from the full velocity point. Thus, the center of the circle (CCX, CCY) in terms of equation Eq. 3

is:

$$CCX = FVX + OrthX \cdot R \text{ and } CCY = FVY + OrthY \cdot R \quad (5)$$

We can now find the exact departure point since the destination, circle center, and departure point form a right triangle. The hypotenuse of the triangle runs from the destination to the circle center. Using this information we can find the angle between the vectors running from the destination to the circle center and departure point $AngleDepDestCC$. The angle from the destination the center of the circle, $AngleDestCC$ is trivial to find. Using these we can find the angle of the vector running from the circle center to the departure point, $AngleDep = ASin(Radius/DistDestCC) + AngleDepDestCC$.

Finding the departure point is a matter of finding the distance from the destination to the departure point $DistDestDep$ and using that to find the exact location of the departure point (DPX, DPY). The points are $DPX = \cos(AngleDep) \cdot DistDestDep + DX$ and $DPY = \sin(AngleDep) \cdot DistDestDep + DY$.

Now that we have the location of the departure point we can find how long the agent is on the circle. We do this by taking the angles from the circle center to the full velocity point and then from the circle center to the departure point. The angle difference between these over 2π will give us the percentage of the circumference that the agent follows between the full velocity point and the departure point. We call this time $TCircle$. To find the time the agent spends on the straight motion step we take the distance from the departure point to the destination we found earlier ($DistDestDep$) and divide by r_m : $TDest = DistDestDep/r_m$. Combining $\frac{r_m}{r'_m}$, $TCircle$ and $TDepDest$ gives us the total time to reach the post-full velocity destination. More Formally:

$$AT_{Post}(\pi, D_x, D_y, r_m, r'_m, \theta'_m) = \frac{r_m}{r'_m} + TCircle + TDepDest \quad (6)$$

Notice that the trajectory from the full velocity point to the destination is composed of straight lines and circles. This Dubins path (see Sec. 2) is expected since the agent's velocity does not change between these points.

4.3 Pre-Full Velocity Destinations

Thus far we have talked about how to find AT_{Post} , or the time required for an agent to reach its post-full velocity destination where $at = \pi$. Now we address the issue of reaching pre-full velocity destinations.

By definition, pre-full velocity paths are composed of two parts. In the first part the agent turns while accelerating. In the second part the agent heads straight while accelerating with the possibility of continuing straight at maximum speed. The key to deriving these paths and their lengths lies in finding the transition

point between the turning portion and the straight portion. In order to find this transition point, we need to find the time when the angle of the vector running from the agent to the destination has the same heading as the agent. The heading of the agent at time t is simply $t \cdot \theta'_m$. The angle of the vector running from the agent at time t to the destination is: $Tan(\frac{D_y - y(t)}{D_x - x(t)})$ where $x(t)$ and $y(t)$ are defined in Eq. 2. Thus, we can find the transition point if we can solve the following for t

$$\theta(t) = Tan\left(\frac{D_y - \left(\frac{\sin(\theta'_m \cdot t)}{\theta'^2_m} - \frac{t \cdot \cos(\theta'_m \cdot t)}{\theta'_m}\right)}{D_x - \left(\frac{\cos(\theta'_m \cdot t)}{\theta'^2_m} + \frac{t \cdot \sin(\theta'_m \cdot t)}{\theta'_m} - \frac{1}{\theta'^2_m}\right)}\right) \quad (7)$$

There is no analytical solution to this equation and the Taylor Series approximation has too much error to be useful. However, we can reframe the problem to significantly reduce the error. Instead of thinking of an agent moving towards its destination over time, we think of the destination moving around and toward the agent over time. This frame of reference is best defined using polar coordinates where the polar coordinates give the distance and angle offset to the destination. In this frame of reference, the agent will look at its destination when the y coordinate of the destination is 0. We can define the evolution of the destination's position using differential equations as follows, with Δ being the timestep:

$$\begin{aligned} D_r(0) &= D_r \text{ and } D_\theta(0) = D_\theta \\ D_r(t) &= D_r(t - \Delta) - r'_m \cdot \Delta \\ D_\theta(t) &= D_\theta(t - \Delta) - \theta'_m \cdot \Delta \end{aligned} \quad (8)$$

Converting Eq. 8 to Euclidean values, the y value of the rotating destination with respect to time is:

$$r'_m \cdot t \cdot \theta'_m + D_y \cdot \theta'^2_m \cos(t \cdot \theta'_m) - (r'_m + D_x \cdot \theta'^2_m) \sin(t \cdot \theta'_m) \quad (9)$$

There is no analytical way of finding t when Eq. 9 equals 0. However, if we replace the Sine and Cosine functions in Eq. 9 with their second-degree Taylor Series approximations, Eq. 9 becomes tractable (Section 4.4 shows that the error of this and our other equations is minute). The new function with the Taylor Series approximations is:

$$r'_m \cdot t \cdot \theta'_m - t \cdot \theta'_m (r'_m + D_x \theta'^2_m) + D_y \theta'^2_m (1 - \frac{t^2 \cdot \theta'^2_m}{2}) \quad (10)$$

Solving for $t = 0$, we know a pre-full velocity agent will look at its destination when $t = T_{Straight}$:

$$T_{Straight} = \frac{-D_x \cdot \theta'_m + \sqrt{D_x^2 \cdot \theta'^2_m + 2D_y \theta'^2_m}}{D_y \theta'^2_m} \quad (11)$$

Starting from here, we can find the total time for the agent to reach the destination. Eq. 11 gives us the time for the agent to look at its destination. It is a simple matter of integrating to find the agent's location at the point, and hence the remaining distance to be traveled. The total time therefore becomes the time from Eq. 11 plus the remaining time the agent travels straight toward its destination (which we call RT). Combined, we can find the arrival time for an agent with a pre-full velocity destination:

$$AT_{Pre}(\pi, D_x, D_y, r_m, r'_m, \theta'_m) = T_{Straight} + RT \quad (12)$$

We now have the equations for deriving *ArrivalT* where $at = \pi$ for any destination. If a destination is a post-full velocity destination, the equations outlined in Sec. 4.2 will give the arrival time of an agent. If a destination is a pre-full velocity destination, the equations outlined in Sec. 4.3 will give the arrival time of the agent. If the destination is unreachable, *ArrivalT* returns $+\infty$.

We can combine Eq. 6 and Eq. 12 to get the full definition of *ArrivalT* where $at = \pi$. For this equation we introduce a new function $Pre : x, y \rightarrow true, false$ which returns *true* if the destination is to the right of the segregating line defined in Eq. 4 and *false* otherwise. We also define a new function $Post : x, y \rightarrow true, false$ that returns *true* if a destination is to the left of the segregating line defined in Eq. 4 and not in the unreachable area defined in Eq. 5.

$$ArrivalT(\pi, D_x, D_y, r_m, r'_m, \theta'_m) = \begin{cases} AT_{Pre}(\pi, D_x, D_y, r_m, r'_m, \theta'_m), & \text{if } Pre(D_x, D_y) \\ AT_{Post}(\pi, D_x, D_y, r_m, r'_m, \theta'_m), & \text{if } Post(D_x, D_y) \\ +\infty, & \text{otherwise} \end{cases} \quad (13)$$

4.4 Validation and Error Analysis

In order to define *ArrivalT* with $at = \pi$, we broke down destinations into two sets, pre-full velocity destinations and post-full velocity destinations. For the post-full velocity destinations we were able to derive arrival time analytically. For pre-full velocity destinations, we derived the formula for when the agent sees its destination using a Taylor Series approximation. To verify our equations and claims of low error, we compared our equations for agent arrival time *ArrivalT* to actual simulations of nonholonomic agent movement. In doing so we show that there is almost no error and that the simulation converges to our equations as the simulation's timestep converges to 0.

Fig. 3 shows the surface *ArrivalT* with $at = \pi$ across a large set of destinations. Notice that this surface has several key features we would expect. First, arrival times grow as destinations move away from the origin. We would expect this since destinations further from the origin will take longer to reach. Second, the surface is almost radially symmetric, but not quite. You can tell by counting contour lines that destinations with higher D_θ values take longer to reach. We expect this since these destinations require the agent to turn in order to reach them, thus increasing the arrival time. Third, there are no seams or jumps as our equations change from using the post-full velocity equations and pre-full velocity equations. This is because the equations are accurate enough that slight changes in the destination around the segregating line in Eq. 4 do not cre-

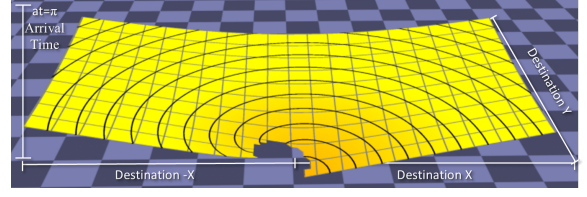


Fig. 3 Plot of how long it will take an agent ($r_m = r'_m = \theta'_m = 1$) to arrive at different destinations when $at = \pi$. The x and y axes correspond to the x and y values of destinations. The height of the curve shows time required to reach destinations. Contour lines are placed every 1s on the surface to help discern the surface's height.

ate any perceptible change the calculated arrival time. Note also that there is a gap in the surface to the left of the origin. This corresponds to the unreachable circle shown in Fig. 2. Since agents cannot reach these points when $at = \pi$, they have an infinite arrival time and are culled from the surface.

For a quantitative validation of *ArrivalT* with $at = \pi$, we generated a similar arrival time surface using the agent simulator from our crowd simulation algorithm. This simulator took time steps, moving the agent forward based on its current linear velocity and heading, and then updating the heading to turn toward the destination. The simulator continued to take steps until the agent came within a small threshold of the destination, at which time it reported the accumulated time. The closer the results of these simulations were similar to those predicted by *ArrivalT*, the more accurate we knew our equations were.

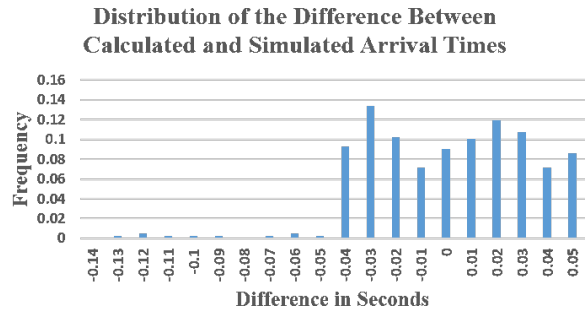


Fig. 4 Difference between arrival times calculated using *ArrivalT* with $at = \pi$ and simulation using 3,300 samples. We used a timestep of .1 in this test. The average difference was a minimal -.0015 seconds.

Fig. 4 shows the error difference between our calculated surface using *ArrivalT* and our simulated agent with a timestep of .1s. The average difference between calculated and simulated arrival times was -.0015s. while the average arrival time for the samples on our surface was 8.93s. Thus, the average difference was only 1/5953 the size of our average arrival time.

Note also that when we run a discrete simulation, it is almost impossible for the agent to arrive exactly at the destination. In the simulation we treat the destination as a circle with a radius of the timestep times r_m . Thus, in this error analysis, we would expect the simulated arrival times of an agent to a destination to be within ± 1 seconds of the true arrival time. Note that in over 99% of the comparisons in Fig. 4 the difference between our calculated arrival time and the simulated arrival time was within this $\pm 1s$ threshold. Also, almost 98% of the comparisons are within half of that threshold. This result gives further evidence that our derivation of *ArrivalT* is highly accurate.

We further looked at the error as the timestep decreases. Since many of our equations find the exact arrival time of an agent, we would expect the difference between our equations and simulated arrival times to decrease as the simulation time step approaches one over infinity. Fig. 5 shows this difference as the simulation timestep approaches 0. Notice that the error decreases exponentially as expected, or approximately linearly on a log scale.

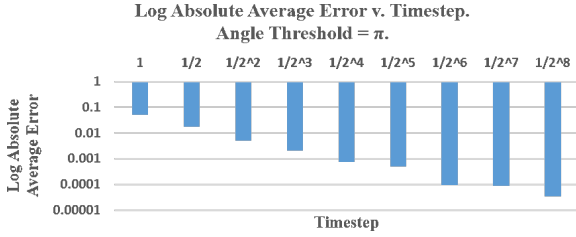


Fig. 5 Log of absolute average difference between *ArrivalT* and simulated arrival times of agents as the timestep of the simulated agents decreases exponentially. Over 3,300 samples were used in each comparison. We would expect this error to drop to zero as the timestep approaches zero, which it appears to do.

5 Arbitrary Angle Thresholds

We have derived *ArrivalT* for an agent where $at = \pi$. We can now use this to derive the solution for *ArrivalT* for arbitrary values of at .

Recall from our section on definitions (Sec. 3) that the movement of agents can be broken down into three steps: the turn in place, the turn with velocity, and the straight motion steps. When $at = \pi$, agents skip the turn in place step since all destinations have a $D_\theta \leq at = \pi$. For most destinations, however, the optimal acceleration angle threshold is not $at = \pi$. Thus we need to expand our solution to *ArrivalT* to include all values of at before we can solve *BestAT*.

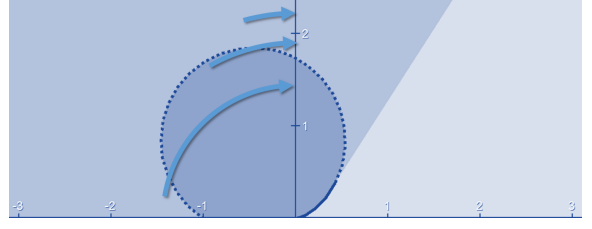


Fig. 6 How destinations with $D_\theta > \pi/2$ rotate during the turn in place step when $at = \pi/2$. Notice that this changes which destinations are reachable when compared to $at = \pi$.

Consider an agent whose acceleration angle threshold is $\pi/2$. For all destinations where $D_\theta < \pi/2$, the path of the agent to the destination would be the same as an agent with an acceleration angle threshold of π since in both cases D_θ falls underneath these thresholds. The only difference is how the agent reaches destinations with $D_\theta > \pi/2$. In these cases the $at = \pi/2$ agent would turn in place until the difference between its heading and the destination is $\pi/2$. It would then proceed to the turn with velocity step.

When $at = \pi/2$, an agent would turn until destinations with $D_\theta > \pi/2$ are $\pi/2$ radians away. For simplicity, we can think about the destination rotating about the agent instead of the agent rotating in place. Fig. 6 shows how these destinations would rotate. Notice that some destinations that were unreachable to an agent with $at = \pi$ are reachable to an agent with $at = \pi/2$ agent and vice versa. Once the destination has been rotated, the agent proceeds as if $at = \pi$.

After rotating destinations, we can use *ArrivalT* with $at = \pi$ as the key component of our solution to *ArrivalT* for any at . First, if $D_\theta > at$, we rotate destinations about the origin until they have a polar angle threshold were π , using the newly rotated destination. Formally:

$$\begin{aligned}
 & \text{ArrivalT} : at, D_x, D_y, r_m, r'_m, \theta'_m \rightarrow \text{Time} \\
 & \text{DestATDiff} = D_\theta - at \\
 & \text{TurnTime} = \begin{cases} (D_\theta - at)/\theta'_m, & \text{if } \text{DestATDiff} > 0 \\ 0, & \text{otherwise} \end{cases} \\
 & D'_x = \begin{cases} \cos(at)D_r, & \text{if } \text{DestATDiff} > 0 \\ D_x, & \text{otherwise} \end{cases} \\
 & D'_y = \begin{cases} \sin(at)D_r, & \text{if } \text{DestATDiff} > 0 \\ D_y, & \text{otherwise} \end{cases} \\
 & \text{RemainingTime} = \text{ArrivalT}(\pi, D'_x, D'_y, r_m, r'_m, \theta'_m) \\
 & \text{Time} = \text{TurnTime} + \text{RemainingTime}
 \end{aligned} \tag{14}$$

Eq. 14 satisfies our first main contribution (see Sec. 1): we have derived the arrival time for a nonholonomic agent to any destination with arbitrary movement constraints. This gives us the mathematical background to find *BestAT*, or the best acceleration angle threshold for a given destination.

Additionally, this equation gives us the exact path of a nonholonomic agent without any simulation. By using these integrals and equations, we can derive the exact location of an agent at any time. This is an important contribution since it allows applications of nonholonomic agents to analytically check for collisions without doing any relatively expensive discrete simulation. We discuss this further in our future work section (Sec. 7).

6 Optimal Angle Thresholds

The primary goal of this work is to find optimal paths for nonholonomic agents without doing simulation. We do this by deriving a function *BestAT* that finds the optimal acceleration angle threshold for an agent in terms of its arrival time. In the previous section we arrived at a definition of *ArrivalT*, the function that gives the arrival time for any acceleration angle threshold. In this section we use *ArrivalT* to find a low error, functional approximation to *BestAT*. First, note that given a destination with a polar angle of D_θ , all angle thresholds whose values are greater than D_θ will produce the same path since the agents will all immediately start accelerating while turning. Thus, if the best angle threshold for a destination is D_θ , *BestAT* could return any value in the range $[D_\theta, \pi]$. For clarity and to make *BestAT* a clearly defined function, we say that *BestAT* never returns an angle higher than D_θ .

Since we have a definition of *ArrivalT*, we can do offline computation to find the numeric approximation of the decision surface for finding the best acceleration angle threshold for a set of destinations. To do this, we iterate across a dense set of destinations in the area $x \in [-10, 10], y \in [0, 10]$. For each destination we then iterate across acceleration angle thresholds, starting with 0 and ending at D_θ . This process returns the best acceleration angle threshold for each sample in our area, which we show in Fig. 7. Similar surfaces can be shown for other values of r_m, r'_m , and θ'_m . Areas of this surface where *BestAT* = D_θ are highlighted in green. Notice that this decision surface is not perfectly flat, i.e., there is no one, hard-coded acceleration angle threshold that will result in the best arrival time for all destinations. This again validates our assertion that finding optimal paths for nonholonomic agents is non-trivial.

Unfortunately, this numerical approach to finding the optimal acceleration angle threshold for a given destination is computationally expensive and will not work in real-time as a way of choosing paths. Instead, we found a functional approximation of *BestAT* that will produce this same decision surface.

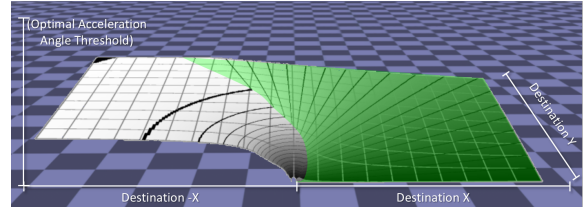


Fig. 7 *BestAT*, or the optimal acceleration angle decision surface (r_m, r'_m , and $\theta'_m=1$). This surface has two distinct regions: one where *BestAT* equals D_θ (the plateau region, highlighted green) and the sub-plateau region, where the contour lines form circles.

One approach to finding *BestAT* would be to take Eq. 14 and find its derivative with respect to at . The zero-crossings of the derivatives that correspond to local minima would determine which angle thresholds produce the shortest arrival times. Eq. 14 is composed of a series of closed form equations, so we were able to find the derivatives for each branch case using a computational mathematics package. Unfortunately, the resulting derivatives are so long that we estimate that transcribing them into this paper would take approximately 10 pages. In addition, these derivatives do not have closed-form solutions for their zero-crossings. We therefore have derived a functional approximation to *BestAT* based on empirical observation.

6.1 Plateau Region

Based on our observations, we have found several key features of this surface that hold regardless of the underlying values of r_m, r'_m , and θ'_m . We leave it to future work to show mathematically that these features are true for all values of r_m, r'_m , and θ'_m .

The primary features of this set of curves are best described using the polar definition of destinations (D_r, D_θ) . Using this polar form of the *BestAT* function, we have observed that $\forall r_g > r, \text{BestAT}(r_g, D_\theta, r_m, r'_m, \theta'_m) \geq \text{BestAT}(r, D_\theta, r_m, r'_m, \theta'_m)$. In other words, for a fixed destination angle, the optimal acceleration angle threshold monotonically increases as the polar radius of the destination increases. At some radius, the optimal acceleration angle threshold will reach D_θ , which is the maximum acceleration angle threshold that an agent can choose, and all values of *BestAT* for radii after that point will all have the same value (D_θ). Thus, the optimal acceleration angle threshold always reaches a plateau. We call the minimum polar radius for which destinations of a given polar angle have an optimal acceleration angle threshold of D_θ the *plateau point*, or simply *plateau*. Fig. 7 shows *BestAT* with the plateau region highlighted in green.

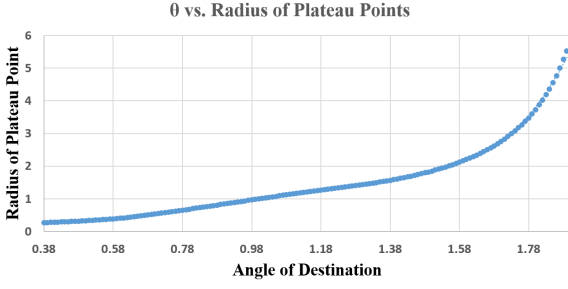


Fig. 8 Plot of D_θ vs. D_r of the plateau points the define the plateau region in Fig. 7. These points are key to our functional approximation to $BestAT$.

Our second observation is that the polar radius grows monotonically as the polar angle increases. The highlighted region in Fig. 7 gives evidence of this observation.

The location of these plateau points is key to approximating $BestAT$. Once we have a function for finding the radius of a plateau point given a radius, we can define half of the optimal acceleration angle threshold decision surface precisely. This is true since we know that any destination point whose D_r is greater than that of the plateau point will always have an optimal acceleration angle threshold of D_θ . As noted earlier, the derivative of Eq. 14 is too unwieldy to provide guidance as to where these points are. Instead, we have numerically found plateau point pairs (r, θ) for set values of r_m, r'_m , and θ'_m . We then did a curve fit across these points to find an approximation to where these plateau points are located.

For example, consider the case where r_m, r'_m , and θ'_m all equal 1, which have been the movement constraints we have used in all our figures. Fitting a fifth-degree polynomial to the plateau points in Fig. 8, we get:

$$-4.68 + 30.72D_\theta - 73.08D_\theta^2 + 84D_\theta^3 - 45.23D_\theta^4 + 9.26D_\theta^5 \quad (15)$$

To find the optimal acceleration angle threshold for a destination (D_r, D_θ) , we plug D_θ into Eq. 15 to get the plateau point for this angle. If D_r is greater than or equal to the value returned by Eq. 15, then we know the optimal acceleration angle threshold is D_θ .

6.2 Sub-Plateau Region

We call the region of surface where the polar radius is less than the plateau point the *sub-plateau region*. Again, we return to our observations of surfaces with different r_m, r'_m , and θ'_m values with an example shown in Fig. 7. Notice that in the sub-plateau region, the contour lines on the surface form perfect circles. In other words, if we find the optimal acceleration angle threshold for a point not on the plateau region, all destinations

with the same D_r and higher D_θ s will have the same optimal acceleration angle threshold. This optimal acceleration threshold value is the value of the plateau point whose radius is D_r . Another way to think about this is that each plateau point “gives” its value to all destinations whose D_r is the same and whose D_θ is greater. This results in the perfectly circular contour lines on the optimal acceleration angle threshold surfaces. More formally if $BestAT(r, D_\theta, r_m, r'_m, \theta'_m) < D_\theta$ (i.e. the pair (D_r, D_θ) is not in the plateau region), then $\forall \theta > D_\theta, BestAT(r, \theta, r_m, r'_m, \theta'_m) = BestAT(r, D_\theta, r_m, r'_m, \theta'_m)$.

Thus, in order to find the optimal acceleration angle threshold for a destination in the sub-plateau region, we can find its value by rotating around the origin until we find the plateau point whose radius matches the current D_θ . We could do this by incrementally decreasing D_θ and checking for plateau points, but it is easier to simply invert the plateau point function (e.g. Eq. 15). Based on our second observation above (that the radius of plateau points increase monotonically with their angles), we know that we can invert these functions. Returning to the case where r_m, r'_m, θ'_m all equal 1, the fifth-degree polynomial fit to the points is:

$$0.12 + 1.18D_r - 0.31D_r^2 + 0.041D_r^3 - 0.002D_r^4 + 0.00006D_r^5 \quad (16)$$

Thus, given a destination (D_r, D_θ) , our functional approximation to $BestAT$ is:

$$\begin{aligned} P &= PlateauPoint(D_r, D_\theta) \\ SP &= SubPlateauPoint(D_r, D_\theta) \\ BestAT(D_r, D_\theta) &= \begin{cases} D_\theta, & \text{if } D_r \geq P \\ SP, & \text{otherwise} \end{cases} \end{aligned} \quad (17)$$

This approximate functional solution to $BestAT$ has very low error. The drawback is that it requires fitting a fifth-degree polynomial to the plateau points for different values of r_m, r'_m , and θ'_m . Fortunately, in most virtual agent applications there is a limited set of possibilities for r_m, r'_m , and θ'_m . Thus, if these fits are done offline, a simulation algorithm can quickly approximate $BestAT$ using this baked information.

6.3 Error

We quantitatively validate our functional approximation to $BestAT$ in Eq. 17 by comparing our calculated decision surface to the one we generated numerically (Fig. 7). To generate the numerical surface we iterated over thousands of possible at values across 3,300 possible destinations. These destinations were in the polar region $D_r \in [0.38, 12], D_\theta \in [0, \pi]$ to avoid numerical issues close to the origin.

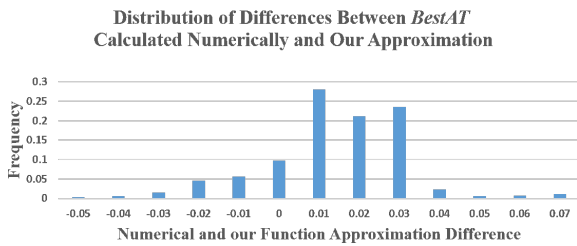


Fig. 9 Distribution of error (in radians) between the numerical solution to *BestAT* and our approximation from Eq. 17.

The average error was .009 radians with a standard deviation of .017 (see Fig. 9). Note that .009 radians is less than half a percent of $\pi/2$. This low error gives us high confidence in our approach.

7 Conclusion and Future Work

In this paper we derived the arrival time of nonholonomic agents to arbitrary destinations with arbitrary movement constraints. We also showed how to find the location of an agent at any time without having to do any discrete simulation. We also derived a numerical, offline method for finding the best arrival time surface. We analyzed this surface and showed how to define this surface with two curves for use in realtime applications. This means that any application that uses nonholonomic virtual agents can quickly find optimal paths for agents that start with no velocity.

Our future work is focused in several areas. First, we want to solve these equations for arbitrary starting velocities. Second, we want to see how this affects crowd simulation since we can find precise ideal paths to destinations. This could lead to reduced arrival times for crowd simulation agents.

References

1. Arechavaleta, G., Laumond, J.P., Hicheur, H., Berthoz, A.: Optimizing principles underlying the shape of trajectories in goal oriented locomotion for humans. *Humanoid Robots, 2006 6th IEEE-RAS International Conference on* pp. 131–136 (2006)
2. Arechavaleta, G., Laumond, J.P., Hicheur, H., Berthoz, A.: An optimality principle governing human walking. *Robotics, IEEE Transactions on* **24**(1), 5–14 (2008)
3. Van den Berg, J., Lin, M., Manocha, D.: Reciprocal velocity obstacles for real-time multi-agent navigation. *Proceedings of Robotics and Automation* pp. 1928–1935 (2008)
4. Cockayne, E., Hall, G.: Plane motion of a particle subject to curvature constraints. *SIAM Journal on Control* **13**(1), 197–220 (1975)
5. Dubins, L.E.: On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics* **79**(3), 497–516 (1957)
6. Fiorini, P., Shiller, Z.: Motion planning in dynamic environments using velocity obstacles. *The International Journal of Robotics Research* **17**(7), 760 (1998)
7. Harris, C.M., Wolpert, D.M.: Signal-dependent noise determines motor planning. *Nature* **394**(6695), 780–784 (1998)
8. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. *Physical Review* **51**(5), 4282–4286 (1995)
9. Hicheur, H., Pham, Q.C., Arechavaleta, G., Laumond, J.P., Berthoz, A.: The formation of trajectories during goal-oriented locomotion in humans. i. a stereotyped behaviour. *European Journal of Neuroscience* **26**(8), 2376–2390 (2007)
10. Lacquaniti, F., Terzuolo, C., Viviani, P.: The law relating the kinematic and figural aspects of drawing movements. *Acta psychologica* **54**(1), 115–130 (1983)
11. Ondřej, J., Pettré, J., Olivier, A., Donikian, S.: A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG)* **29**(4), 123:1–123:9 (2010)
12. Reynolds, C.: Flocks, herds and schools: A distributed behavioral model. *Proceedings of ACM SIGGRAPH Computer Graphics* **21**(4), 25–34 (1987)
13. Soueres, P., Laumond, J.P.: Shortest paths synthesis for a car-like robot. *Automatic Control, IEEE Transactions on* **41**(5), 672–688 (1996)
14. Thalmann, D., Musse, S.R.: *Crowd simulation*. Wiley Online Library (2007)
15. Todorov, E., Jordan, M.I.: Smoothness maximization along a predefined path accurately predicts the speed profiles of complex arm movements. *Journal of Neurophysiology* **80**(2), 696–714 (1998)