DIRECTIONAL EXTENSIONS OF SEEK STEERING BEHAVIOR

Michal Jakob

Gerstner Laboratory, Department of Cybernetics,
FEE, Czech Technical University
Technicka 2, 166 27 Prague 6

and

Illusion Softworks,
Opatovicka 4, 110 00 Prague 1

E-mail: jakob@labe.felk.cvut.cz

KEYWORDS

steering, motion control, autonomous agents, computer games

ABSTRACT

Steering behaviors present an important component of low-level control of autonomous agents in computer games, animation and robotics. The article extents existing steering behaviors with the *directed seek* behavior, which steers the agent to arrive at a given waypoint in a given direction while considering minimum turn radius of the agent. In contrast to existing ad-hoc approaches, the presented solution is geometrically well-founded, robust, computationally efficient and compatible with other steering behaviors. Three-dimensional generalization of the steering algorithm as well as its several extensions are also presented. Directed seek behavior was implemented and has been tested as a part of the motion control system in a commercial action-strategy computer game.

1 INTRODUCTION

Practically all computer games involve moving agents, whether it be game characters, vehicles or other objects. The fundamental requirement is that the movement is realistic. Foremost, the agents must obey physical laws holding in the game world, which is the requirement secured by the game physic subsystem. Most computer games, however, involve goal-oriented intelligent agents. The minimum requirement on such agents to appear realistic is the ability to move towards locations specified by their higher-level goals while avoiding mobile and static obstacles. The game subsystem implementing such functionality is called the *steering subsystem*. Originating in the research on collision avoidance in robotics [BK89, KB], steering nowadays plays an important role in computer games, animation, virtual reality as well as in the control of real mobile robots. Its importance is reflected by the fact that it is one of the six areas of game artificial intelligence in which standardization is pursued [NPK03].

The article proceeds as follows. After a brief introduction to steering, we introduce and describe a new steering behavior called the *directed seek*. We explain the rationale behind it and present its algorithmic implementation. Next, we extended the behavior also for directed seeking in 3D space. We touch on the experience with the implementation of directed seek in a commercial computer game, and finally, we discuss its several possible extensions.

2 STEERING FOR AUTONOMOUS AGENTS

Steering is reactive, non-deliberative movement of physical agents based on the local environment surrounding every single controlled steered entity. [NPK03] In the three-level agent motion control framework introduced by Reynolds [Rey99], steering comprises the middle level of the hierarchy. The lower level called *locomotion* is responsible for actual articulation of the motion while the higher level called *action selection* deals with the deliberative part of motion control, i.e., path-planning and path-finding.

The core of the steering control system are steering behaviors. These are reactive, stateless procedures that take local information about the environment surrounding an agent as input, and produce a steering goal as output. The steering goal is often expressed as the steering force which should be applied to an agent to obtain the desired movement. An alternative possibility, followed in this article, is to output directly the desired direction towards which the agent should turn as the steering goal [AOM03].

Basic steering behaviors include, among others, *seek, arrival, pursuit, wander, path* and *wall following* [Rey99]. The basic behaviors can be combined to produce more complex behaviors such as *queuing* and *flocking*. Combined steering behaviors are also used for the control of groups of characters moving in herds and formations [VBOM00]. In practice, the most frequently used behavior is the seek behavior, which attempts to steer the agent so that it approaches a specified target location (*waypoint*). In addition to being used alone, the seek behavior is a crucial part of many other behaviors, eg. pursuit and path following.

2.1 Directed Seeking

It is often useful to specify not only the target location but also the target agent heading for the seek maneuver. Situations in which it is useful include arriving at an attack location so that a target is in front of the agent (for agents with limited angular attack range), pursuing a target and trying to match its predicted heading (eg. for aerial dogfight) or directed formation movement (when used for multiple agents). Because majority of real-world agents cannot turn in place, it is necessary to incorporate minimum turn radius of the agent in the steering procedure to obtain feasible trajectories.

The above given situations could be – at least partially – dealt with using the standard seek behavior (eg. by using several subsequent waypoints or by continually adjusting the waypoint location). Such ad-hoc solutions, however, are generally more complex, hard-to-debug and cannot usually take the minimum turn radius into account.

In contrast, the *directed seek* behavior introduced in this article is explicitly designed to account for waypoint direction and agent minimum turn radius. As such it produces smooth and simple paths, while staying computationally efficient and easy-to-combine with other steering behaviors.

3 DIRECTED SEEK BEHAVIOR

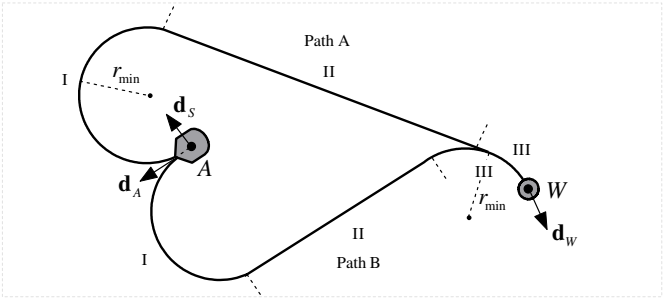

Figure 1: Two Possible 3-Segment Paths (steering vector $\mathbf{d}_s$ for path A is drawn)

3.1 Problem Definition

The input of directed seek procedure consists of (Figure 1):

1. Current agent location A and current agent direction $\mathbf{d}_A$ (by direction/heading, we understand a unit-length vector).
2. Waypoint location W and waypoint direction $\mathbf{d}_W$
3. Agent minimum turn radius r_{min}.

The output of the procedure is the direction $\mathbf{d}_s$ the agent should turn to to arrive at W in direction $\mathbf{d}_W$.

We first consider steering in 2D space, i.e., on a plane. The generalization to 3D space is given in Section 4. For the moment, we assume that the turn radius does not depend on the agent speed. Furthermore, we assume that the agent can move only forwards, i.e., it cannot perform 3-point turns. Both the extension to speed-dependent turn radius and to backward moving agents will be discussed in Section 6.

3.2 Geometric Formulation

A path towards a waypoint followed by the directed steer maneuver consists of three segments/stages (Figure 1). No steering is applied in segment II. In segments I and III, the agent makes tightest possible turn with its minimum radius r_{min}. Depending on the position and orientation of A and W, there can be up to two 3-segment paths. In majority of cases, one of them is the overall shortest path from A to W, although a 5-segment paths can be shorter if A and W are closer than $2r_{min}$. Nevertheless, we do not presently consider 5-segment paths in the directed steer algorithm.

Key to determining the steering direction is to determine the type of path (A or B), which the steering maneuver follows, and the current segment of the path. The former is discussed in Section 3.4, the latter can be decided based on the geometrical formulation of the problem (Figure 2).

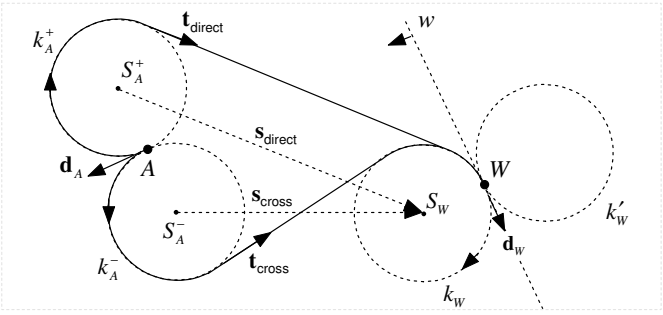

Figure 2: Three-Segment Paths with Osculating Circles

Here, k_A and k_W are circles of r_{min} radius. Directions $\mathbf{d}_A$ and $\mathbf{d}_W$ are tangent to circles k_A^+, k_A^- and k_W at points A and W, respectively. Circle k_W lies at the same side of line w as A. All circles are oriented so that their orientations agree with the directions of $\mathbf{d}_A$ and $\mathbf{d}_W$, respectively (a closed curve c is positively oriented if its interior is on the right when travelling along it). We denote orientation of circle k as $\mathtt{O}(k)$.

Since all the circles k_A^+, k_A^- and k_W are fully defined by the inputs of the directed seek procedure, the problem of determining the current path segment is transformed into determining common tangents of pair of appropriate circles.

3.3 Common Tangent Problem

Depending on their relative position, two circles can have anywhere between zero and four common tangents (except for a very special case of two identical circles, i.e., two circles with the same radius and center points; identical circles have infinite number of common tangents). These tangents can be divided into *direct* tangents, which have both circles on the same side (lines t_{direct}^{++} and t_{direct}^{--}), and *cross* tangents, which have the circles on opposite sides (lines t_{cross}^{+-} and t_{cross}^{-+}). Circles of the same radius, which we deal with in our problem, always have

two direct tangents and anywhere between zero and two cross tangents. Moreover, their direct tangents are parallel with the line connecting their centers.

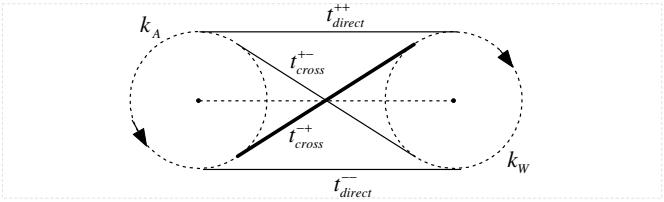

Figure 3: Common Tangents (admissible tangent is drawn in bold)

For 3-segment paths, however, we are interested only in the tangent whose direction is in accord with the orientation of both circles, and which goes from k_A to k_W. It can be easily seen that there is at most one such *admissible* tangent for two oriented non-identical circles. If both circles have the same orientation, then it is a direct one, if they have opposite orientations, it is a cross one. So while two identically oriented circles always have one common admissible tangent, two oppositely oriented circles have a common admissible tangent only if they do not overlap.

Since k_A^+ and k_A^- have opposite orientations, one of them always has the same orientation as k_W and thus has an admissible common tangent with it. Common admissible tangent then corresponds to segment II of the agent path. Therefore, there always exists at least one 3-segment path from A to W. Based on the type of tangent, we speak of a *direct path* and a *cross path* to the waypoint.

3.4 Choosing the Better Path

In case there are two 3-segment paths, we have to decide which one the agent should follow. We base this decision on the total angle an agent turns along its path to the waypoint. Path total angle is defined as

$$\Phi(\mathbf{d}_A,\mathbf{d}_W,\mathbf{d}_t) = \varphi^{\mathtt{0}(k_A)}(\mathbf{d}_A,\mathbf{d}_t) + \varphi^{\mathtt{0}(k_W)}(\mathbf{d}_t,\mathbf{d}_W) \tag{1}$$

where $\varphi^{\mathtt{0}}(\mathbf{d}_1,\mathbf{d}_2) \in [0,2\pi)$ is the *directed* angle between $\mathbf{d}_1$ and $\mathbf{d}_2$, measured from $\mathbf{d}_1$ to $\mathbf{d}_2$ in direction $\mathtt{0}$. Direction $\mathbf{d}_t$ is the direction of the straight segment of the path, i.e., either $\mathbf{t}_{\text{direct}}$ or $\mathbf{t}_{\text{cross}}$ (see Figure 2). Path total angle is computed for both possible paths, and the one with the smaller value is followed.

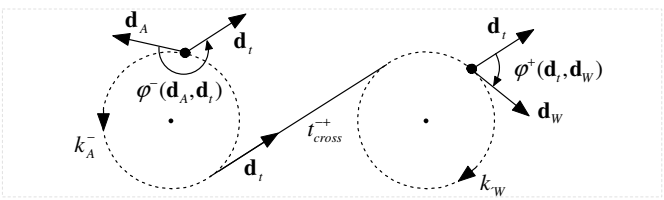

Figure 4: Path Angles

3.5 Achieving Reactivity

For the sake of simplicity and high reactivity, the steering behaviors are stateless, i.e. the steering procedures do not hold any information about the state of a steering maneuver between successive procedure calls. The directed seek procedure thus cannot record in which stage it currently is – in each call it has to again determine the current stage solely from its inputs (as described in Section 3.1). The stage can be determined using the following two vectors (see Figure 2):

1. vector $\mathbf{s}_{\text{direct}} = S_W - S_A^{\mathtt{0}(k_W)}$, i.e. the vector connecting the centers of circles pertaining to the direct path (which always exists)
2. vector $\mathbf{d}_A$

The following conditions hold for individual path stages:

Stage I

$$||\mathbf{s}_{\text{direct}}|| > 0 \wedge \mathbf{s}_{\text{direct}} \nparallel \mathbf{d}_A \tag{2}$$

Stage II

$$||\mathbf{s}_{\text{direct}}|| > 0 \wedge \mathbf{s}_{\text{direct}} \parallel \mathbf{d}_A \tag{3}$$

Stage III

$$||\mathbf{s}_{\text{direct}}|| = 0 \tag{4}$$

Conditions (2) and (4) follows directly from the definition of path segments. Condition (3) is obviously met when the direct path is followed. It is, however, met also for the cross path, because in case the cross common tangent exists, the direct common tangent also exists, and both tangents coincide.

Since the three conditions are mutually exclusive, they allow to unambiguously determine the path segment an agent currently is in. For practical use, it is useful to add some tolerance to equality and parallelism tests.

3.6 Directed Seek Algorithm

By combining the evaluation of the path total angle and the test described in the previous section, the directed seek procedure can determine all information necessary to compute the steering direction. The algorithm implementing this reasoning is depicted in Figure 5.

4 GENERALIZATION TO 3D SPACE

The essentially two-dimensional directed seek algorithm can be extended to 3D space by considering a steering plane Ω with a 2D coordinate system $\mathcal{F}$. The 3D situation is then projected to Ω, the 2D directed seek algorithm is applied, and the resulting 2D steering vector transformed back into 3D space.

The steering plane Ω is defined by agent location A, waypoint location W and its direction $\mathbf{d}_W$. Coordinate system $\mathcal{F}$ is defined by origin W and two orthonormal basis vectors

$$\mathbf{f}_x = d_W \tag{5}$$

Let S_W be the center of the waypoint tangent circle k_W so that k_W lies on the same side of w as A.
Let $\mathtt{o}_W$ and $-\mathtt{o}_W$ be the orientation of k_W and its opposite.
Let $S_A^{\mathtt{o}W}$ and $S_A^{-\mathtt{o}W}$ be centers of agent tangent circles $k_A^{\mathtt{o}W}$ and $S_A^{-\mathtt{o}W}$ (as depicted in Figure 2).
Let $\mathbf{s}_A^+$ and $\mathbf{s}_A^-$ be the right and left side vector of agent A (perpendicular to agent heading $\mathbf{d}_A$).
$\mathbf{s}_{\text{direct}} := S_W - S_A^{\mathtt{o}W}$; $\mathbf{s}_{\text{cross}} := S_W - S_A^{-\mathtt{o}W}$
if $||\mathbf{s}_{\text{direct}}|| = 0$:
    [*stage III*]
    Do the tightest $\mathtt{o}_W$-oriented turn towards $\mathbf{d}_W$: $\mathbf{d}_s = \min_{\mathbf{d}_S}(\mathbf{s}_A^{\mathtt{o}W}, \mathbf{d}_W)$
    ($\min_{\mathbf{w}}(\mathbf{u}, \mathbf{v})$ is the vector of $\mathbf{u}$ and $\mathbf{v}$ which forms a lower angle with $\mathbf{w}$; for implementation reasons, it is used to limit the steering direction to form at most the right angle with agent current heading.)

else
    if $\mathbf{s}_{\text{direct}} \parallel \mathbf{d}_A$:
        [*stage II*]
        No steering needed, proceed in the current direction
    else
        [*stage I*]
        if $||\mathbf{s}_{\text{cross}}|| \leq 2r_{min}$:
            [*only the direct path exists*]
            Do the tightest $\mathtt{o}_W$-oriented turn towards $\mathbf{t}_{\text{direct}}$: $\mathbf{d}_s = \min_{\mathbf{d}_S}(\mathbf{s}_A^{\mathtt{o}W}, \mathbf{t}_{\text{direct}})$
        else
            [*both direct and cross paths exist*]
            Determine path total angle $\Phi(\mathbf{d}_A, \mathbf{d}_W, \mathbf{d}_t)$ for both variants
            if the cross path has lower Φ
                Do the tightest $-\mathtt{o}_W$-oriented turn towards $\mathbf{t}_{\text{cross}}$: $\mathbf{d}_s = \min_{\mathbf{d}_S}(\mathbf{s}_A^{-\mathtt{o}W}, \mathbf{t}_{\text{cross}})$
            else
                Do the tightest $\mathtt{o}_W$-oriented turn towards $\mathbf{t}_{\text{direct}}$: $\mathbf{d}_s = \min_{\mathbf{d}_S}(\mathbf{s}_A^{\mathtt{o}W}, \mathbf{t}_{\text{direct}})$

Figure 5: Two-Dimensional Directed Steer Algorithm

and

$$\mathbf{f}_y = \mathbf{d}_{WA} - \langle \mathbf{d}_{WA}, \mathbf{f}_x \rangle \mathbf{f}_x \tag{6}$$

where $\langle ., . \rangle$ is the dot product and

$$\mathbf{d}_{WA} = \frac{A - W}{||A - W||} \tag{7}$$

is a unit vector directed from the waypoint to the current agent location. Vector $\mathbf{f}_y$ is thus a unit vector parallel with Ω, orthonormal to $\mathbf{f}_x$ and directed to the half space containing A. We can now express A, $\mathbf{d}_A$, W and $\mathbf{d}_W$ in in $\mathcal{F}$, apply the 2D directed seek procedure and transform the returned 2D steering direction $\mathbf{d}_S^2 = (d_{Sx}^2, d_{Sy}^2)$ back to 3D space as

$$\mathbf{d}_S = d_{Sx}^2 \mathbf{f}_x + d_{Sy}^2 \mathbf{f}_y \tag{8}$$

An important remark is in order. While A, W and $\mathbf{d}_W$ can be exactly expressed in $\mathcal{F}$, $\mathbf{d}_A$ cannot as it generally has a component $\mathbf{d}_A^\perp$ perpendicular to Ω. However, because the resulting steering direction $\mathbf{d}_S$ is by definition parallel with Ω, the perpendicular component $\mathbf{d}_A^\perp$ gradually vanishes during the maneuver. Agent heading becomes parallel with the steering plane Ω, and the agent finishes the seek maneuver along a planar path exactly as in the 2D case. A problem arises if $\mathbf{d}_A^\perp$ is too large and the waypoint too close, in which case it might happen that $\mathbf{d}_A^\perp$ does not vanishes entirely, and the agent reaches the waypoint not perfectly aligned with $\mathbf{d}_W$. Fortunately, such problem is rare in real-world scenarios and can be fully eliminated if a condition on some minimum distance between A and W is added as a requirement of the directed steer algorithm.

The generalization given above does not present a truly 3D algorithm because, in each moment, the steering takes place on a plane. This does not mean, however, that the resulting path is planar. The steering plane changes with the movement of the agent, resulting in a non-planar path and allowing to steer the agent into proper target location and direction even if $\mathbf{d}_A$ does not initially belong to the steering plane Ω (with the limitation discussed in the previous paragraph).

5 IMPLEMENTATION

Directed seek behavior was implemented and has been tested as a part of the motion control system in a commercial game. Experiments performed so far has confirmed its smooth interoperability with other steering behaviors. Moreover, the algorithm exhibited good robustness. Agents were able to reach a waypoint in the specified direction even in situations, in which the directed seek was interfered with collision avoidance maneuvers.

6 POSSIBLE MODIFICATIONS AND EXTENSIONS

Speed-dependent turn radius It is often useful to specify not only the desired heading at a waypoint but also the desired speed. Acceleration and deceleration can be easily integrated in the directed seek behavior in case the agent minimum turn radius is independent of the agent speed.

In reality, however, this is rarely the case. The current agent speed v_A and the desired waypoint speed v_W can differ, and so can differ the minimum turning radii $r_{min}(v_A)$ and $r_{min}(v_W)$.

Experiments showed that thanks to its reactivity, the direct steer algorithm works quite well even in this situation. Nevertheless, we currently test several modifications which make the original algorithm more robust in case of speed-dependent r_{min}. The first and the easiest modification always take the maximum of $r_{min}(v_A)$ and $r_{min}(v_W)$ for the computation of the steering direction. The other modification explicitly considers different radii of k_A and k_W in the directed seek algorithm.

Opposite half space waypoint tangent circles A straightforward modification is to consider both waypoint tangent circles k_W and k'_W (Figure 2) for the determination of the best path. In some situations, using the circle k'_W in the opposite half space might result in a path with lower path total angle.

Backward motion The spectrum of possible directed seek paths can be extended by allowing agents to move backwards, and thus to perform 3-point turns. Although the directed steer could be in principle extended to account for such situations, a question arises whether the reactive design would still be efficient and useful, or whether the control of such maneuvers should be rather left to the higher-level action-selection subsystem.

Truly 3D directed steer algorithm In contrast to a pseudo 3D algorithm given in Section 4, a truly 3D steering algorithm considers a full 3D situation at each moment. Three-segment path in this case is not planar and can be decomposed into two planar parts. Curved segment I then lies in the first plane, curved segment III in the other one, and straight segment II in the intersection of both planes. Since repetitive determination of the two steering planes is computationally expensive and the pseudo 3D algorithm turned out sufficient for most cases, we have not so far decided to implement the fully 3D algorithm. Nevertheless, it remains an interesting option for future research.

7 CONCLUSIONS

In this paper, we introduced the directed seek behavior, which attempts to steer an agent so that it reaches a given target location with a given heading. In contrast to existing approaches to directed seeking, the presented solution is geometrically well-founded, easy-to-combine with other steering behaviors, and takes agent minimum turn radius into account. The result is a robust and computationally efficient behavior which can be used for directed steering in both 2D and 3D space. Several extensions of the behavior and future research directions were also outlined. The directed seek behavior has been successfully implemented and is gradually extended as a part of the steering subsystem in a commercial computer game.

Acknowledgements

The research presented in this paper was supported by Czech Technical University grant No. CTU0306313.

BIOGRAPHY

MICHAL JAKOB obtained a master degree in computer science in 2001 at Czech Technical University Prague. Since then, he has been pursuing a PhD in artificial intelligence there. His main research interests include machine learning and reasoning in multi-agent systems. At the end of 2002, he joined Illusion Softworks game studio as an artificial intelligence designer/programmer. He currently works there on the development of an A-class action-strategy title.

References

[AOM03] Heni B. Amor, Oliver Obst, and Jan Murray. Fast, neat and under control: Inverse steering behaviors for physical autonomous agents. Technical Report 12–2003, Universität Koblenz-Landau, 2003.

[BK89] Johann Borenstein and Yoram Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Systems, Man, and Cybernetics*, 19(5):1179–1187, 1989.

[KB] Yoram Koren and Johann Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1398–1404.

[NPK03] Alexander Nareyek, Nick Porcino, and Mark Kolenski. AI interface standards: The road ahead. A roundtable discussion. In *Game Developers Conference*, 2003.

[Rey99] Craig Reynolds. Steering behaviors for autonomous characters. In *Game Developers Conference*, 1999.

[VBOM00] Jim Van Verth, Victor Brueggemann, Jon Owen, and Peter McMurry. Formation-based pathfinding with real-world vehicles. In *Game Developers Conference*, 2000.