Collision Avoidance between Avatars of Real and Virtual Individuals

René van den Berg[1], Juan Manuel Rejen[2], and Rafael Bidarra[1]

[1] Delft University of Technology
rene3.berg@planet.nl, r.bidarra@tudelft.nl
[2] iOpener Media GmbH
manuel@iopenermedia.com

Abstract. One of the most difficult challenges of associating virtual environments and real-world events deals with integrating real-world, tracked individuals into a virtual environment: while virtual individuals may interact with the avatars of real-world individuals, the latter obviously do not respond to the behaviour of the virtual individuals. To prevent collisions or unrealistic behaviour of either, one needs to 'artificially' modify the trajectory of the tracked individual. Moreover, after such modifications, one should end up returning that avatar back to the accurate representation of the real world.

In this paper we propose a strategy for solving these problems based on generic control functions, which are flexible enough to be fine-tuned on a domain-dependent basis. Control functions determine an offset from the tracked individual's actual trajectory and, when carefully parameterized, guarantee a smooth and automatic recovery after each trajectory modification. We discuss this approach, illustrate it with a variety of Gaussian control functions, analyze the believability of the resulting trajectory modifications, and conclude that control functions provide a sound and powerful basis for improving strategies of collision-avoiding trajectory modification and recovery. Most examples of domain-specific strategies discussed here are taken from the upcoming application area of real-time racing games, which provides both easily recognizable and very attractive situations, and has been the original motivation for this research.

Keywords: Trajectory modification, collision evasion, real-world integration.

1 Introduction

According to [4], a virtual environment (VE) is "a multi-sensory real-time simulation that immerses the participant in a multi-dimensional (usually 3D) graphical space, allows freedom of movement within the space, and supports interactions including the modification of most features of the space itself." For the purposes of this paper, the most important part of this definition is the fact that a VE supports interactions: the user can project himself into an environment and interact with it as if it were real.

Taking this concept one step further, we might imagine a VE that *is* real: a virtual copy of an actual environment. If we can copy each relevant static feature of a real-world environment, and also track its relevant dynamic features, the VE may present the user with a virtual clone of an actual environment.

A. Egges, R. Geraerts, and M. Overmars (Eds.): MIG 2009, LNCS 5884, pp. 1–12, 2009.

In such an environment, players may then interact with other individuals through avatars. For example, suppose we create a racing game in which the user is allowed to test his skills against the professional drivers, who are racing at the track at that very same time. The actions of each opponent are tracked and transmitted over the Internet to players at home, so that they may not only see, but even participate in the race [5].

Alas, such players can only participate in any real-world environment using *asymmetric interactivity*. Asymmetric interactivity means that although the player and the VE are influenced by the state of the real world, the real-world environment is not altered by any virtual interaction, nor does any real-world individual directly act upon any purely virtual object. Thus, the interaction must be simulated locally. For example, a real-world vehicle's trajectory may pass through a virtual object. To avoid collisions or even overlap in such cases, the trajectory of either the real-world or the virtual individual (the player's avatar) needs to be modified — but the player will not tolerate being moved around by external forces. Also, it is up to the active party to prevent collisions — if any individual is performing an overtaking maneuver, the responsibility of evading the other individual lies with him. Therefore, the system must intelligently modify the trajectory of the virtual individual.

Any modifications to a real-world individual's trajectory must be performed in a way that is guaranteed to recover, stable, and realistic or, at least, believable. A modification that is guaranteed to recover means that the influence of another individual on the behavior of the virtual individual decreases as the distance between them increases, until it is 0 at a certain point. Stability means that care should be taken to decrease the probability that the modification of one individual's trajectory brings it so close to another that the *other* individual will need to start maneuvering. Realism, or the less stringent requirement of believability, dictates that an observer or the user is not able to distinguish whether the trajectory perceived was modified or not, or at least the alterations are not disturbing in any way.

Our main contribution is the notion of a *control function*, a generic strategy to perform trajectory modifications on the avatars of real-world individuals in a way that is guaranteed to recover with time, and allows for the customization of the specific trajectory modification strategy based on current circumstances and the type of environment.

The rest of this paper is structured as follows: in the next section, we will present some previous research that is related to ours. Section 3 will highlight the context in which our approach can be used, followed by an explanation of the basic approach in Section 4. Section 5 provides the theory for possible extensions to the control function approach, and Section 6 will present some results, followed by conclusions and future work in Section 7.

2 Related Work

The specific domain of interaction between dynamic real-world individuals and virtual individuals is still very new. However, there are some areas with strong similarities to our current research area.

The field of robotic navigation is one example. Some interesting research has been performed in the area of motion planning in the presence of moving obstacles, proving

that the problem of finding an optimal solution to the problem is intractable [9]: the complexity of the problem prevents a solution from being found fast enough to be used in all but the most trivial cases. However, nearly optimal planning and collision avoidance are still thoroughly being researched. Recent results from this research include the idea of deforming precomputed roadmaps discussed in [3]. This approach features roadmaps made up of nodes, chained together by elastic links, which deform in the presence of obstacles.

Another field with strong relations to the current research is the simulation of human crowds or other groups. This field makes strong use of short-term predictions, collision-avoiding trajectories and goalseeking. A lot of emergent behavior can be seen in these simulations, provided they are adequately programmed: lane formations, for example, naturally occur in some synthesized environments, as well as in the real world.

Given that a model exists that can adequately simulate this behavior, it can also be used for predicting it. Discomfort fields [10], the use of local potential fields and the separation between high-level static path planning and local collision avoidance [11] are some of the important ideas that can be applied to the research at hand. Also, future work may incorporate some simple psychology for path prediction or modification, as proposed in, for example, [7].

Finally, an interesting angle on path prediction and recovery from incorrect movements is presented in [1].

3 Domain and Problem Characteristics

In this section, some important aspects of trajectory modification strategies are presented. This knowledge is required to understand the problems our approach deals with, and the assumptions made in developing our solution. A more in-depth treatment of related issues can be found at [12].

First, the real-world individuals are tracked within the real-world environment. This data is then transmitted using any suitable means to the local VE. It is in this local VE that we apply trajectory modification strategies.

To determine whether the trajectory of a real-world individual can be represented as-is or requires modification, it is necessary to obtain accurate trajectory predictions for both the real-world and the virtual individuals. The real-world individual's trajectory is often easier to predict, given accurate input variables. A simple physical extrapolation will suffice for most scenarios, since most "real" individuals are not inclined to sudden behavior alterations. Examples of such extrapolations are the various Dead Reckoning models, which are commonly used in such diverse applications as computer games and navigation systems to reduce network load [4], [1]. However, different environments require different Dead Reckoning models, as has been extensively researched in [8].

The behavior of the virtual individual is more difficult to predict: the player is more likely to "misbehave", or do something unexpected. In a car race, for example, it is reasonable to assume that no real-world individual will drive against traffic on purpose. A player is a lot more likely to misbehave in such circumstances, for a myriad of reasons, among which are the lack of social pressure, the willingness to test limits, and the fact that there is, after all, no actual danger involved. But predicting the virtual individual's trajectory *is* the most important: if no virtual individuals are near any real-world

individuals, no modification to the real-world individuals' trajectory is ever needed. If the user is not purposely misbehaving, a mix between a short-term physical model and a longer-term strategic model is ideal for prediction. Examples of this approach are the Hybrid Model [2], which alternates between a Dead Reckoning prediction and a strategical prediction, and the Hybrid Method [6], which enhances Dead Reckoning prediction with longer-term Interest Schemes.

Assuming that some adequate prediction scheme can be found, the predictions for both the virtual individual and each real-world individual are calculated and compared. If their predicted positions are, at some future point, too close together, overlap or collisions may occur. To prevent this, an evasive maneuver performed by the real-world individual is simulated — but only if that individual is the "active" party in the impending collision. In a car race, for example, the frontmost individual would not make space for the overtaking individual. Which individual, if any, performs the evasive maneuver should be determined by the system based on the current circumstances.

Another characteristic is that the prediction algorithm used for the real-world individuals depends on the nature of the interaction. If the tracking data of the real-world individuals is prerecorded, or sufficiently delayed in their presentation, the near future of their trajectory is already known. This future may be used as an "extremely accurate prediction". However, this necessitates an estimation of the predictability of that future; if it includes "surprising" elements, it should be rejected as a prediction and replaced by some other, more "naive" prediction scheme.

4 Basic Approach

We propose to control the evasive maneuver using a *control function*, which describes the modifications to the trajectory of the real-world individual. Trajectory modifications are only ever applied to real-world individuals, since they have no full knowledge of their surroundings in the virtual environment, as opposed to the virtual individuals. Furthermore, we focus here on theory aspects that are valid for all applications, leaving application-specific analyses, such as tuning physical constraints, for future work.

4.1 Basics of Control Functions

An n-dimensional control function is a combination of one or more mathematical functions $f(d) : \mathbb{R}^m \rightarrow \mathbb{R}^n$, with $m < n$, which determines the offset from the actual trajectory for an evading individual. The input variable d is the distance between the centers of the two individuals in their *unmodified* positions, and the output variables are offsets in certain directions. This notion of "certain directions" is kept deliberately vague, since control functions can be applied to a variety of coordinate systems, such as Cartesian, polar or cylindrical coordinates. In this paper, we will assume that the primary axis is always perpendicular to the tangent of the trajectory.

Control functions might not reflect physical restrictions, so it is up to the person using them to make sure that the chosen function is suitable for the application at hand in terms of maneuverability, velocity adjustments, et cetera.

Furthermore, instead of the normal distance between the individuals, we use the signed distance to distinguish between being ahead or behind the other individual. A

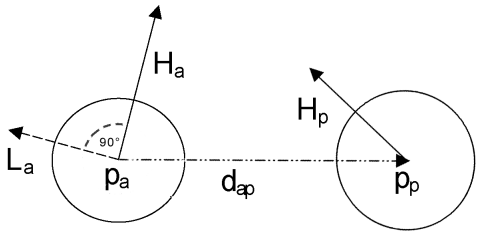


Fig. 1. The important variables of the trajectory modification. $\mathbf{p_a}$ and $\mathbf{p_p}$ are the positions of the active and passive individuals, respectively. $\mathbf{H_a}$ and $\mathbf{H_p}$ are their respective heading vectors, or the tangent vectors of their trajectories. $\mathbf{d_{ap}}$ is the vector from the active to the passive individual and $\mathbf{L_a}$ is the active individual's normalized *left-hand vector*, the primary axis for modification.

positive distance then indicates that the active individual is ahead of the passive one, that is, $\mathbf{d_{ap}} \cdot \mathbf{H_a} > 0$, and vice versa (see Fig. 1).

However, this signed distance might never be 0. This causes the output variable to display discontinuous behavior if the control function is not symmetric around $d = 0$, since in that case the distance, at some point, suddenly switches signs without passing through 0. To prevent this, the input variable used is the signed magnitude of the difference vector $\mathbf{d_{ap}}$ between the individuals projected onto the heading vector H_a, which is calculated by $\mathbf{d_{ap}} \cdot \mathbf{H_a}/|\mathbf{H_a}|$. This projected signed distance is then guaranteed to be continuous. The heading vector should not exhibit sudden changes, because otherwise the projected signed distance will behave erratically. For most environments, this is a reasonable requirement. If it is not, the passive individual could be projected onto the active individual's trajectory instead.

A very welcome consequence of using the distance as the input variable, is that the graph of the control function horizontally scales with velocity difference: no matter what the velocity difference between the individuals is, the resulting trajectory modification will always look smooth. This is illustrated in Fig. 2.

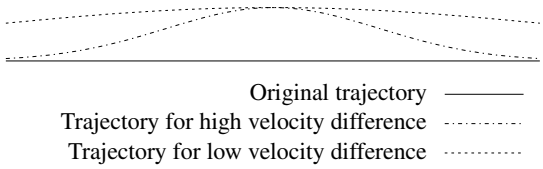


Fig. 2. The automatic "stretching" of the deformation function due to low velocity difference

4.2 Requirements and General Observations

To be suitable as a control function, a mathematical function $f(x) : \mathbb{R} \rightarrow \mathbb{R}^n$ should fulfill some requirements.

First of all, it should accurately reflect the capabilities of the individual for whom the modification strategy is meant. This includes, amongst others, the requirement of

continuity, so as not to yield any sudden changes in position or direction of movement.

Second, it should have *finite support*, which means that $f(x) > 0$ for only a finite set of input values. This region of support should contain $x = 0$ and be nearly symmetric around the y axis. This allows for a smooth transition between the original and the modified trajectory at the start and end of the maneuver.

Another general rule is that most functions should often be centered around the y axis. Also, the function should generally increase over exactly one interval and decrease over exactly one other interval: "swaggering" mostly decreases the believability of the resulting trajectory.

4.3 1-Dimensional Control Functions

A 1-dimensional (1-D) control function has an output value which is used as an offset in a direction perpendicular to the $\mathbf{H_a}$ vector. A default direction must be determined beforehand, so that the sign of the output becomes meaningful; in this paper, we assume that the default direction is L_a (see Fig. 1), 90 degrees clockwise from H_a as viewed from above, or "left". The trajectory will mostly be modified to the same side the active individual is on with respect to the passive individual: to the left if it is on the left, and vice versa.

To determine whether to modify the trajectory to the left or the right side, the following formula, which makes use of the *signum* function, is used:

$$\operatorname{sgn}(\mathbf{L_a} \cdot \mathbf{d_{ap}}) * \operatorname{sgn}(\mathbf{H_a} \cdot \mathbf{H_p}). \tag{1}$$

The result of this equation is multiplied by the outcome of the control function to yield a signed offset magnitude, which should be multiplied by $\mathbf{L_a}$ and added to p_a to obtain the active individual's new position. Special care should be taken in case Equation 1 yields 0. This may occur in two cases: if the heading vectors are perpendicular to each other, the evasion should occur as determined by first part of the equation, but if p_p lies on the line determined by H_a, the sign should be calculated as $-\operatorname{sgn}(\mathbf{L_a} \cdot \mathbf{H_a})$.

An excellent example of a 1-D control function is the Gaussian function. While it does not strictly comply with the "finite support"-rule, some of its other properties make it useful as a control function in every other respect — most importantly its nicely bell-shaped graph. The generic formula for a Gaussian function g is:

$$g(x) = ae^{-\frac{(x-b)^2}{2c^2}} \tag{2}$$

for any set of real values for $a \neq 0$, b and $c \neq 0$. These parameters are, interestingly enough, nicely reflected in the shape of the graph: while a is reflected in the amplitude of the Gaussian, and thus the magnitude of the trajectory modification, b relates to its location along the x-axis and c is a measure for the width of the bell-shaped curve. The influence of the parameters is visualized in Fig. 3.

A slight detractor for the Gaussian function is that it can never satisfy $f(x) = 0$. This means that if the Gaussian function is used as a control function, the trajectory modification will never stop being active. However, the Gaussian function falls off quickly

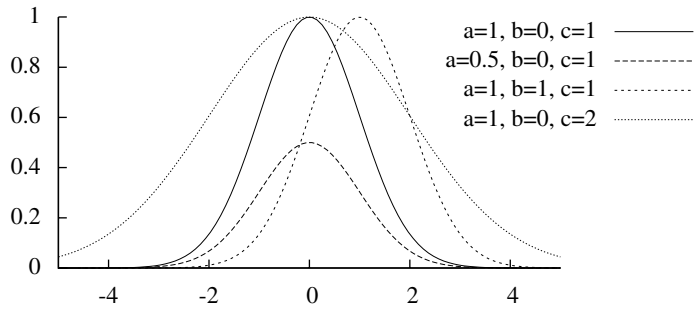


Fig. 3. The influence of the Gaussian parameters

enough as its input value increases or decreases to use a simple cutoff, fulfilling the finite support-requirement as well.

5 Advanced Control Function Extensions

In this section, we discuss two extensions to control functions: a 2-dimensional control function approach, and automatic configuration of control function variables.

5.1 2-Dimensional Control Functions

A 2-dimensional control function adds a longitudinal offset to the perpendicular offset. This can be used to fine-tune the trajectory generated by a 1-D control function. The net effect of an extra modification dimension is an apparent speed modification. Extra care should be taken to prevent unrealistic or impossible behavior, since the direction of movement is modified; for example, to seemingly slow down a bit, the individual moves less in the forward direction, while the movement in the perpendicular direction stays the same. The result of this modification may become unrealistic if the perpendicular movement overshadows the longitudinal movement.

The effect can be visualised as taking the graph of the primary, 1-D control function, and displacing each point on it in the longitudinal direction, where the displacement is controlled by a second 1-D control function. In the context of 2-D control functions, this new, 1-D part of the function is referred to as the *longitudinal control function*, while the previously discussed primary control function is referred to as the *perpendicular control function*.

In general, there are some simple rules to keep in mind when selecting a mathematical function to be used as the longitudinal control function:

- If the input and the output value have the same signs, the distance between the individuals is increased by the control function.
- A negative output value puts the individual "behind" its actual location. Conversely, a positive value means the individual is ahead of its true location.
- Anywhere the longitudinal deformation function has a positive derivative, the individual accelerates. By the same token, a negative derivative slows it down.

– Both the function and its first and second derivative should never have extremely high or low values, because this may have an adverse effect on the plausibility of the resulting movement. If the individual is projected too far ahead, for example, this may result in a velocity that is higher than the physical maximum of that individual.

Any function which follows these rules is fit to be used as a longitudinal deformation function. Keep in mind that a 2-D control function may be either a combination of multiple 1-D control functions, or a mathematical function $f(d) : \mathbb{R} \rightarrow \mathbb{R}^2$.

Our prime example of combining two 1-D control functions to yield a 2-D control function is using the Gaussian function as the perpendicular control function and its derivative as the longitudinal control function. This approach is especially useful since both functions can be manipulated by the same parameters b and c, as discussed in the previous section. Also, the Gaussian derivative has one distinct region where the result is positive and one region where the result is negative, and these regions are rotationally symmetric around the point $(b, 0)$. All this means that it can be used to accelerate first and then decelerate when the other individual is passed, or the other way round.

Moreover, if the same c value is used for the longitudinal and the perpendicular control function, the *inflection point* of the perpendicular modification coincides with the minimum or maximum longitudinal offset. This inflection point is where the perpendicular acceleration switches signs, which means that the perpendicular velocity starts decreasing where it was previously increasing, or vice versa.

5.2 Automatic Variable Configuration

Most control functions will have some parameters, allowing the system to control their shape, like the a, b and c parameters for the Gaussian function. However, any set of precomputed values will fall short in some situations. Thus, it is better to calculate appropriate values at runtime. Automatic parameter optimization is an important area of AI research, which has yielded various well-known approaches. These can also be applied to the problem at hand. However, some knowledge of how the parameters affect the maneuver may be useful.

The value of a should be calculated as a function of, amongst others, the minimum evasion distance necessary and the velocity difference between the active individual and the passive individual. The value of b should depend mostly on strategic considerations — for example, in a racing environment, a negative b would be used to represent a blocking maneuver after overtaking: the active individual "swoops in" sharply and cuts off the passive individual. However, a positive b would represent "sticking" to the actual course a little longer to be able to use the slipstream generated by the passive individual.

The value for c is the most difficult one: if this parameter is set too low, activating the algorithm may result in an unrealistic trajectory with high perpendicular velocities. However, setting it too high will result in unnecessary maneuvering and increasing recovery time. Also, if the value of c is not chosen carefully, this may result in "snapping" behavior: the displacement calculated as the trajectory modification behavior is activated is not as close to 0 as it should be. This is because the c dictates that an overtaking maneuver is started as soon as the individuals are m meters apart, while the impending collision is only detected when they are n meters apart. Now if $n < m$, the

offset $f(d)$ at the start of the maneuver is not 0, thus causing an extremely unrealistic instantaneous perpendicular movement. The value of c, then, is mostly a function of the difference in velocities: if this difference is higher, the c value should also be higher to allow for a longer "preparation distance" and eliminate sharp turns and high perpendicular velocities. However, if the velocity difference is lower, the c should be kept low, since the maneuver will otherwise "wake up" too late and snap into activity. A possible function for calculating c would be $c = k * \Delta v$, where Δv is the difference in velocity.

6 Results and Evaluation

We tested our theories by using the aforementioned control functions in both ideal and normal conditions. Two sets of tests were performed. One set deals with a synthetic environment with only one straight road, so that perpendicular displacement is only caused by the control function. In the other set of tests, data from an individual tracked at the Zandvoort circuit was used to test the interaction between real and virtual individuals in a realistic environment.

Table 1. Algorithm settings for testing environments

Test	Test1	Test2	Test3
a_1	5.0	5.0	5.0
a_2	-5.0	10.0	-25.0
b	0.0	0.0	0.0
c	0.5	2.5	1.0
v_a	10.0	50.0	70.0
v_p	5.0	25.0	60.0
v_{diff}	5.0	25.0	10.0

Table 2. Velocity changes resulting from the algorithms

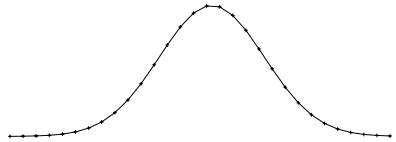
Parameter		Test1	Test2	Test3
	a_1	5.0	5.0	5.0
	a_2	-5.0	10.0	-25.0
	b	0.0	0.0	0.0
	c	0.5	2.5	1.0
	v_a	10.0	50.0	70.0
	v_p	5.0	25.0	60.0
	v_{diff}	5.0	25.0	10.0
1D Contr. Func.	$max(v_{err})$	0.45	0.18	0.066
2D Contr. Func.	$max(v_{err})$	9.99	-3.99	24.97
2D Contr. Func.	$avg(v_{err})$	3.19	1.46	8.70

6.1 Specially Prepared Environments

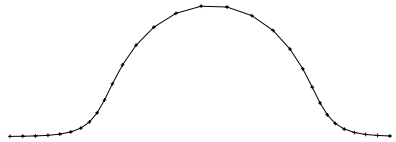
Two vehicles were created, one behind the other. They were both instructed to follow the road's middle line. Both vehicles were given a constant velocity, and that of the rearmost was the highest, necessitating overtaking maneuvers. In Table 1, the configuration variables for three different tests are shown. The a_1 and a_2 mean that these values are set for the perpendicular and longitudinal direction, respectively, while v_a and v_p refer to the velocities of the active and passive individuals. In each test, the b and c values were kept the same for the longitudinal and perpendicular control functions.

In Test 1, a small velocity difference is tested in a low velocity environment. Test 2 represents a large velocity difference in a medium-velocity environment, while Test 3 shows a small velocity difference in a high-velocity environment.

Obviously, it is hard to render the resulting 2-D time varying dataset on paper. Even though it is easy to visualize the trajectory itself, it is harder to visualize the velocities



(a) Test 1 using 1-D Gaussian control function



(b) Test 1 using 2-D Gaussian control function

Fig. 4. Subsampled trajectories for Test 1, synthetic environment. Vertically upscaled 5 times for visual inspection.

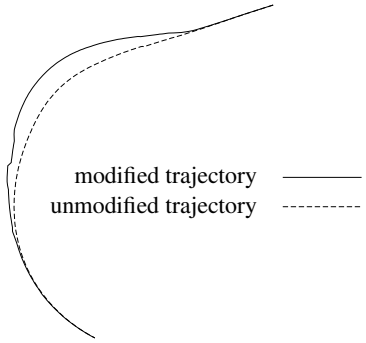


Fig. 5. Resulting Trajectories for one test run (1-D control function, realistic environment)

that go with the datapoints. Therefore, graphs were created using regular sampling from the output data: by rendering only one out of every n datapoints in the graph, velocities can be inferred from the distances between consecutive points. To allow for easier visual inspection, the perpendicular (vertical) direction is upscaled 5 times in the graphs. The shape difference between the 1-D and 2-D control function graphs in Fig. 4 is purely due to the addition of a longitudinal control function. Obviously, the maneuver resulting from the 2-D control function has more chance of properly avoiding a collision, but this comes at a price: its longitudinal velocity will be somewhat higher at some points, thus increasing the risk of violating physical restrictions, as seen in Table 2.

6.2 Realistic Environments

The performance of the algorithms was also evaluated using data tracked at the Zandvoort circuit. The dataset contains some noisy measurements, which shows in the output. By using subsampling and interpolation techniques, new datasets were created from the only available dataset for individuals with the same trajectory, but a different velocity. This was done by multiplying the basic velocity by different factors. Also, a number of datapoints was trimmed from the start of the dataset, so that each vehicle would have a different starting point. These types of datasets can be combined in a sheer endless number of ways, so only some representative results are presented.

When plotting the results, it turns out they are somewhat more erratic, largely a result of the noisy nature of the dataset. For example, in Fig. 5, the modified trajectory is not completely smooth. This is caused by incorrect measurements of location of the passive individual. This brings up an interesting and important point: control functions are highly sensitive to noise in the measured data. If control functions are to be successfully applied, prefiltering of location and heading data is absolutely required.

The fact that most of the errors in the resulting trajectory are a result of the noise instead of the algorithm can be proven by the “back-calculation” of the Gaussian. In

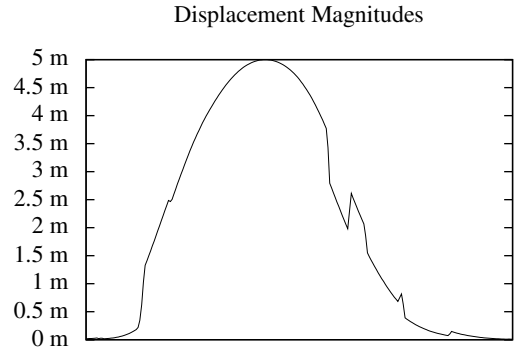


Fig. 6. The magnitudes of the actual offsets for one test

the areas where the input parameter to the Gaussian function increases or decreases monotonously, the shape of the graph should be exactly that of a (stretched) Gaussian. Indeed, it turns out that calculating $m(x_{err}, y_{err})$ for each resulting displacement, where $m(x_{err}, y_{err}) = \sqrt{x_{err}^2 + y_{err}^2}$ and x_{err} and y_{err} are the displacements in x and y direction, respectively, yields a graph that is globally shaped like a Gaussian, but contains some spikes, as shown in Fig. 6. The smallest of these spikes can be explained by the fact that the input parameter is not completely monotonous, but the larger spikes result from noisy data. Non-monotonicity of the input parameters is not problematic, since the Gaussian approach was designed to deal with it, but non-continuity *is* a serious problem.

7 Conclusions and Future Work

Some conclusions can be drawn from our research:

- The 1-D Control Function provides a useful and adequate modification strategy for some environments, *if* its parameters are correctly configured. In the Gaussian control function, especially c, which determines the length of the curve, should be carefully adjusted, preferably at real-time, based on the current circumstances.
- The 2-D control function algorithm is somewhat more dangerous and may increase or decrease velocities by large amounts if the parameters are not very carefully configured, as seen in Table 2. The results may be nice, since the shape of the perpendicular control function can be modified to better suit the current circumstances, but the configuration is somewhat harder, and failure will often be dramatic.

The most important opportunities for future work is the research of different control functions. The Gaussian function is an adequate control function for some environments, but there is no doubt that there are functions better suited to any number of different circumstances. Some of these functions may be created by first determining the desired shape and then performing Fourier analysis on the resulting graph. While this would yield a function that violates the finite-support requirement, it is trivial to reduce the function to only one period of the signal. It is important to realize that control

functions are more useful if their shape can be configured using parameters determined at real-time to adapt to the current circumstances. The automatic configuration of these variables is another interesting area for future work.

References

1. DeCarpentier, G.J.P., Bidarra, R.: Behavioral assumption-based prediction for high-latency hiding in mobile games. In: Proceedings CGAMES 2007 (2005)
2. Delaney, D., Ward, T., Mcloone, S.: On reducing entity state update packets in distributed interactive simulations using a hybrid model. In: Proceeding of the 21st IASTED International Multi-conference on Applied Informatics, pp. 10–13 (February 2003)
3. Gayle, R., Sud, A., Lin, M., Manocha, D.: Reactive deformation roadmaps: motion planning of multiple robots in dynamic environments. In: IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2007, pp. 3777–3783 (2007)
4. Gossweiler, R., Laferriere, R., Keller, M., Pausch, R.: An introductory tutorial for developing multiuser virtual environments. Presence: Teleoperators and Virtual Environments 3(4), 255–264 (1994)
5. System for Simulating Events in a Real time Environment, U.S. Patent Application No. 12/106,263; for more information contact iOpener Media, `http://www.iopenermedia.com`
6. Li, S., Chen, C., Li, L.: A new method for path prediction in network games. Comput. Entertain. 5(4), 1–12 (2007)
7. Musse, S., Thalmann, D.: A hierarchical model for real time simulation of virtual human crowds. IEEE Transactions on Visualization and Computer Graphics 7(2), 152–164 (2001)
8. Pantel, L., Wolf, L.: On the suitability of dead reckoning schemes for games. In: NetGames 2002: Proceedings of the 1st workshop on Network and system support for games, pp. 79–84. ACM, New York (2002)
9. Reif, J., Sharir, M.: Motion planning in the presence of moving obstacles. J. ACM 41(4), 764–790 (1994)
10. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: SIGGRAPH 2006: ACM SIGGRAPH 2006 Papers, pp. 1160–1168. ACM, New York (2006)
11. Van Den Berg, J., Patil, S., Sewall, J., Manocha, D., Lin, M.: Interactive navigation of multiple agents in crowded environments. In: SI3D 2008: Proceedings of the 2008 symposium on Interactive 3D graphics and games, pp. 139–147. ACM, New York (2008)
12. Project website, `http://graphics.tudelft.nl/Game_Technology/vandenBerg`