Multi-Objective Optimization and Decision-Making in Context Steering

Alexander Dockhorn, Sanaz Mostaghim
Otto von Guericke University
Magdeburg, Germany
alexander.dockhorn@ovgu.de, sanaz.mostaghim@ovgu.de

Martin Kirst, Martin Zettwitz
Polarith GmbH
Magdeburg, Germany
martin.kirst@polarith.com, martin.zettwitz@polarith.com

***Abstract*—This work concentrates on decision-making for autonomous movement of agents to simultaneously optimize several objectives which occur in their local environment. Such behavior can be achieved with steering algorithms, which have originally been designed for moving numerous agents simultaneously where occasional uncertainties are not noticeable by players. Nevertheless, concentrating on single individuals can reveal major flaws in their movement patterns such as oscillatory movement. For avoiding such problems, game makers are forced to develop higher-level abstractions for handling game-relevant special cases. Thus, eliminating the initial benefit of steering behaviors to be highly modular, lightweight, and controllable. This work enhances the context steering approach by Fray, which introduced discretized contextual information in the aggregation of a steering behavior's components. We combine this method with multi-criteria decision-making for controlling the agent's velocity direction and magnitude. The resulting approach is tested based on selected scenarios which show that the resulting approach is well suited to improve the agent's smooth and natural movement. Based on our observations we propose suitable parameterizations of the designed method and discuss advantages and disadvantages of made enhancements.**

***Index Terms*—Context Steering, Autonomous Movement, Multi-Criteria Optimization, NPC, AI**

I. Introduction

The development of games often requires the design of believable agents. In recent years, game worlds have become more complex and dynamic, which can make the development of suitable movement algorithms a daunting task. As a result, moving entities can get stuck or show other forms of unrealistic movement. This can break the players' immersion and reduce their enjoyment of the game world. Hence, flexible and robust methods for movement control are required.

The two most common approaches for AI movement are represented by path-planning and steering algorithms. While path-planning methods, such as A* [1], [2] or navigation meshes [3], [4] make use of global planning to find an optimal path, their applicability is considerably constrained in complex dynamic scenarios. In addition, the situation becomes even more challenging if the game strongly relies on physics for controlling characters, so the AI also has to consider forces and their resulting impact on the movement. The dynamics of a world and complex movement constraints can render path-finding algorithms inapplicable, due to the size of the resulting search space and the frequent re-planning required to accommodate for the environment's dynamics.

978-1-6654-3886-5/21/$31.00 ©2021 IEEE

An alternative approach is the use of steering algorithms which limit the agent's search to its local environment. Instead of making a single global decision, the agent moves by making a sequence of short-term decisions which should bring the agent closer to its goal. The original steering approach [5] uses a collection of behaviors which each return a preferred movement direction. To determine the agent's movement direction, the directions returned by each behavior are aggregated. Steering has proven to be a valuable tool for game designers, due to the simplicity of its underlying behaviors and the potential complexity that can emerge from their combination.

This flexibility comes with the drawback of having to fine-tune the agent's movement by painstakingly optimizing a large number of interdependent parameters. With an increasing number of simple behaviors and parameters, this process can become quite complex. In particular, a problem arises in the dependency of many variables, which is caused by the aggregation of the individual behavior decisions. Typical problems include deadlocks or oscillatory movement [6], in which the decisions of underlying behaviors contradict each other.

The steering approach proposed by Fray [7] is tackling these problems by integrating the context of each behavior's decision into account during the aggregation process. This allows for more fine-grained control of the aggregation process and has shown able to eliminate some sources of unrealistic movement but cannot avoid deadlocks entirely. In this work, we propose to extend Fray's approach by a multi-objective decision-making process for stable agent movement.

The main contributions of this work are:

- **Multi-Criteria Decision-Making for Context Steering:** Given our formalization of context steering, we propose a multi-objective view of the steering problem and methods to solve it.
- **Methods for Smoothing the Steering Behavior:** History Blending has been proposed by Fray [8] to reduce the oscillating behavior of context steering agents. Since the approach has never been clearly defined, we propose methods for smoothing the steering values in a single time step and over consecutive time steps. Furthermore, we formalize the interpolation of neighboring context values, another principle that has been proposed by Fray [8], to

approximate the optimal steering direction based on the receptors' perceptions.

In the following section, we will provide a brief overview of the principles of steering, context steering, and a review of related work on the optimization of steering behaviors. In Section III we present our multi-objective context steering algorithm and formalize methods for smoothing the agent's behavior. In Section IV we will introduce several test scenarios and discuss the observed performance of the proposed approach. We conclude this work in Section V by summarizing our results and discussing possible applications and ideas for possible enhancements to be analyzed in future work.

II. Background and Related Work

In the following, we briefly summarize the original steering approach proposed by Reynolds (Section II-A). Furthermore, we review the context steering approach by Fray, which represents a well-known extension on which this work will be based on (Section II-B). In Section II-C, we describe how multiple behaviors are combined in context steering. Thereafter, we review the fundamentals of multi-criteria decision-making (Section II-D), which will be used in Section III. This section ends with a brief overview of related work on optimizing agent movement and multi-objective optimization for agent control (Section II-E) to place this work into the context of others.

A. Steering

Steering algorithms represent a class of movement algorithms that reduce the path-finding problem to a series of short-term decisions, whereas each decision takes the agent's local environment into account. In this work, we will consider the agent's environment to be a 2-dimensional, dynamically changing world. This environment can consist of multiple objects which can draw the agent's interest or represent objects to be avoided.

Reynolds [5], [9] proposed the use of multiple simple behaviors to define the building blocks of a more complex steering agent. Each of these steering behaviors observes the agent's local environment and returns velocities, consisting of a direction vector and its magnitude, to react to the current game situation. Most common examples are the behaviors *seek* and *flee*. The former returns velocities pointing towards objects the AI agent is interested in, whereas the *flee* behavior returns velocities pointing away from objects an agent wants to avoid. Moreover, the way these vectors are computed can be altered by the use of additional steering forces. For example, such forces might be used such that the resulting motion path always describes an arc instead of a direct line for simulating a physics-oriented movement even if no physics engine is involved. After their acquisition, the velocity vectors returned by each behavior are aggregated and result in a single velocity used for the agent's movement. This final vector can be altered with weight-based and/or priority-based methods depending on the specific game and situation.

Due to the simple principles of underlying movement behaviors, steering has proven to be an accessible tool for AI designers in the games industry. Although steering behaviors are designed to be lightweight and simple to combine, in practice, they come with major problems which especially stand out when looking at single computer-controlled individual instead of flocks and herds. Because of its aggregation, steering can result in a movement that appears unreasonable to the player in specific situations, such as the example shown in Figure 1. The weighted aggregation of a seek behavior pulling towards the green diamond and pushing away from red circles, may result in a deadlock in which the agent stands still because the aggregation of each behavior's movement vector yields a null vector. Similar problems occur in dynamic environments, in which agents can show oscillating movement behavior.

B. Context Steering

As we will see in the following section, changing the aggregation process has the potential to improve on the observed problems. Fray's proposal of context steering [8] origins from the observation of discussed deadlock situations. He argued that such problems can be avoided by adding more information to the aggregation process.

Instead of just aggregating the proposed movement direction of each behavior, those will return a context map instead. The context map consists of a behavior's rating of each movement direction. To generate such a context map, the agent will have a set of receptors $r \in R$ to perceive the agent's surroundings in their associated direction v_r. Based on this observation each behavior generates a rating of each direction, e.g. interest or danger, and writes the result in a scalar array hence worth called context map.

Figure 2a shows the generation of a seek behavior's context map. The value of each cell is dependent on the type of object the behavior is sensitive to, the distance of the agent pos_a and the object pos_o, and the angle ω between the steering direction v_s and the direction the receptor is pointing to v_r.

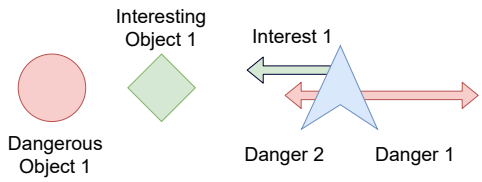
The angle can be determined through:

$$\omega = cos^{-1} \frac{\langle \vec{v_r}, \vec{v_s} \rangle}{\|\vec{v_r}\| \|\vec{v_s}\|} \in [0, \pi] \tag{1}$$

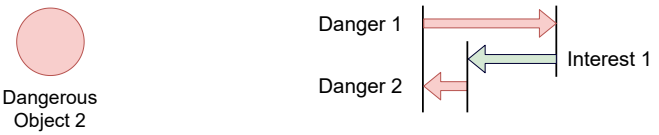
If the angle is smaller than a given threshold $\theta \in [0, \pi]$, the receptor is affected by the object, and a context value z_r greater than 0 is calculated.

$$z_r = f(\omega) g(\|\vec{v_s}\|) \in [0, 1] \tag{2}$$

The context value z_r depends on the exact location of the detected object o in relation to the agent a and its related receptor r. The function f maps the angle ω from the domain $A \in [0, \theta]$ to a value in the range of $U \in [0, 1]$. Function $g(\|\vec{v_s}\|)$ maps the distance of the detected object $\|\vec{v_s}\|$ which is in the domain $[\xi_{min}, \xi_{max}]$ to a value in the range of $[0, 1]$. Given the thresholds ξ_{min} and ξ_{max} we can denote the agent's detection range, which effectively results in the agent ignoring all objects outside of this range. In case multiple objects are in range, only the highest context value will be written to the context map.

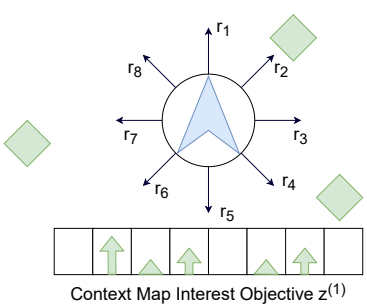


(a) Exemplary steering vectors of a seek and flee behavior.

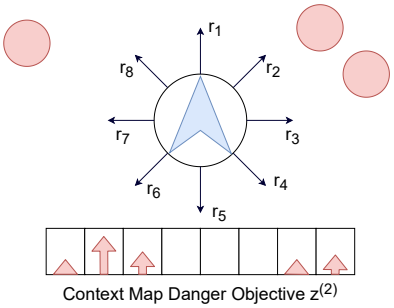


(b) Resulting deadlock in which movement vectors contradict each other

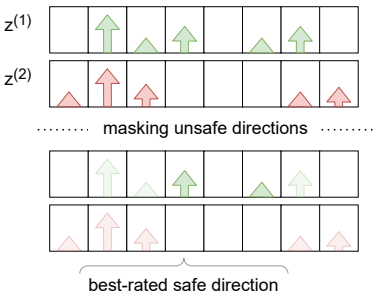
Fig. 1: An example where a steering scenario (a) can lead to to a deadlock (b). Green diamonds represent objects that the blue agent is interested in. The red circles on the left and right represent dangerous objects.



(a) Context map of a *seek* behavior.



(b) Context map of a *flee* behavior.



(c) Combined context map.

Fig. 2: Example of the context steering aggregation process.

The mapping type used for $f(\omega)$ and $g(\|\vec{v_s}\|)$ depends on the desired agent behavior. For a seek behavior that shall find close objects, an inverse linear mapping should be used for $f(\omega)$ and $g(\|\vec{v_s}\|)$. Other interesting mappings may be squared mapping, square-root mapping, or their inverse versions.

C. Combining Behaviors and Context Maps

For the aggregation of multiple behaviors, we first define how two behaviors with the same objective can be aggregated in a single context map, after which we describe Fray's approach for combining context maps of different objectives.

In context steering, multiple behaviors can be defined with the same objective in mind, e.g. minimizing the distance to objects of interest. Those context maps can easily be aggregated to $z^{(agg)}$ by using the maximum value for the related entries of the two context maps $z^{(1)}$ and $z^{(2)}$.

$$z_i^{(agg)} = \max\{z_i^{(1)}, z_i^{(2)}\}, \qquad \forall i : 1, \ \ldots, |R| \tag{3}$$

Besides combining multiple behaviors with the same objective into one context map, the context steering algorithm proposed by Fray [8] is also able to handle multiple objective functions. He specifically focused on the objectives of minimizing the distance to interesting objects while avoiding dangerous objects.

Figure 2c illustrates an example for combining context maps of different objective functions. To combine these into a single steering direction, first, the lowest value in the danger map will be determined (i). All directions with a higher danger value will be eliminated as they are considered to be too dangerous (ii). For the remaining directions, we choose the direction with the highest interest value as the agent's steering direction.

While this process has shown to yield a more stable and robust steering behavior, the aggregation process is still ignoring much of the available information. In this work, we will propose the use of multi-criteria optimization methods for the aggregation of context maps with different objective functions.

D. Multi-criteria Optimization (MCO)

MCO methods can be applied whenever there are two or more conflicting objective functions to be optimized at the same time. Within this work, we will use the following notation:

$$\begin{aligned} \text{minimize} \quad & \{f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_k(\mathbf{x})\} \\ \text{subject to} \quad & \mathbf{x} \in S \end{aligned} \tag{4}$$

whereas f_i represents an objective function used to rate an agent's movement direction and a total of $k \geq 2$ objective functions will be optimized at the same time.

Each *objective functions* $f_i : \mathbb{R}^n \to \mathbb{R}$ which span the *objective space* $\mathbb{R}^k$. The *decision (variable) vectors* $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ belong to the (nonempty) *feasible region* (set) S which is a subset of the *decision variable space* $\mathbb{R}^n$. The image of the feasible region which is a subset of the objective space is denoted by Z and called *feasible objective region*, and its elements are called *objective (function) vectors* or *criterion vectors* and denoted by $\mathbf{f}(\mathbf{x})$ or $\mathbf{z} = (z_1, z_2, \ldots, z_k)^T$, where $\forall i = 1, \ldots, k : z_i = f_i(\mathbf{x})$ are *objective (function) values* or *criterion values* [10, p. 5].

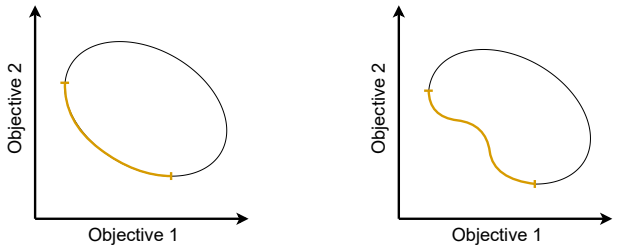


Fig. 3: Example MCO problems; left: convex, right: concave. The Pareto optimal sets for minimizing the two objectives are highlighted in orange.

It is important to note that there is no total order for solutions because of the conflicting objectives. Thus, the notion of *Pareto dominance* is applied to determine the optima of our MCO problem. A decision vector $\mathbf{x}^* \in S$ is *Pareto optimal* if there is no other decision vector $\mathbf{x} \in S$ such that $\forall i = 1, \ldots, k : f_i(\mathbf{x}) \leq f_i(\mathbf{x}^*)$. In addition, an objective vector $\mathbf{z}^* \in Z$ is Pareto optimal if there is no other objective vector $\mathbf{z} \in Z$ such that $\forall i = 1, \ldots, k : z_i \leq z_i^*$ and $\exists j : z_j < z_j^*$, or equivalently, $\mathbf{z}^*$ is Pareto optimal if the decision vector corresponding to it is Pareto optimal [10]. The set of all Pareto optimal solutions is called *Pareto optimal set* [10] or *Pareto front*, and all solutions resulting by the subsequently introduced methods in Section III-B are elements of this set. Figure 3 visualizes the Pareto Fronts of two exemplary MCO problems.

An MCO problem is shaped either convex or concave as also illustrated in Figure 3. The shape determines which methods for obtaining Pareto optimal solutions are convenient to use because the ability to find specific solutions differ from method to method. In Section III-B, we discuss the problem shapes that can be expected in the context of multi-objective context steering and how this reflects on our selection of a suitable MCO method.

E. Learning and Optimization of Steering Behavior

Since we are going to propose a new method for the aggregation of context steering and therefore the optimization of their movement behavior, we are briefly reviewing alternative methods in this section. Despite the common use of (context) steering for designing an agent's movement behavior, there is not much literature on extensions of the baseline approaches. Most notably, Fray has reviewed the context steering approach in his recent work [11], in which he discussed context steering for racing agents. In this, he also suggested the use of history blending, which can help in reducing observed oscillating behavior. For this purpose, the agent blends between context maps of consecutive time-steps for a smoother transition.

Another line of work concentrates on the optimization of a steering agent's behavior. The development of steering agents involves setting up a set of rules and behaviors that will ultimately be combined to result in an agent's steering behavior. This results in a large number of parameters to be tuned, which is especially complex due to these parameters being dependent on each other. To address the high dimensionality of the optimization problem, Gerdelan and O'Sullivan [12] proposed the use of an evolutionary algorithm to tune the rules of a fuzzy controller for steering. Their evaluations show that this system is able to produce a better controller configuration than the hand-tuned reference set. However, due to the nature of the aggregation, this system is still prone to deadlocks.

An alternative process has been explored by Croitoru [13] who tried to learn steering behaviors to approximate given trajectories. For this purpose, a particle swarm optimization-based method has been proposed to derive agent steering behaviors based on Reynolds' boids model [9].

None of these steering methods specifically address the multi-objective nature of the steering problem. In the following, we are going to propose a multi-objective aggregation process that should help in reducing the number of deadlocks and overall increase the believability of the agent's movement.

Algorithm 1 SAMPLEOBJECTIVES
in: $(O, B, V, \theta, \xi_{min}, \xi_{max})$ | **in+out:** (D, I)

set all objective values of D and I to 0
for all objects $o \in O$ **do**
  for all behaviors $b \in B$ **do**
    $\vec{v_s} \leftarrow$ CLASSICSTEERINGBEHAVIOR(b, o)
    if $\|\vec{v_s}\| < \xi_{min}$ **or** $\|\vec{v_s}\| > \xi_{max}$ **then**
      continue (with next element, skip current)
    $s \leftarrow 1 - (\|\vec{v_s}\| - \xi_{min}) \div (\xi_{max} - \xi_{min}) \in [0, 1]$
    $\vec{v_s} \leftarrow \vec{v_s} \div \|\vec{v_s}\| \cdot s$
    for $k \leftarrow 1, \ldots, |V|$ **do**
      $\vec{v_r} \leftarrow V[k]$
      $\omega \leftarrow cos^{-1}(\langle \vec{v_r}, \vec{v_s} \rangle \div \|\vec{v_r}\| \|\vec{v_s}\|) \in [0, \pi]$
      if $\omega \leq \theta$ **then**
        $z_r \leftarrow f(\omega) g(\|\vec{v_s}\|) \in [0, 1]$
        if o is danger **then**
          $D[k] \leftarrow$ MAX$(D[k], z_r)$
        else
          $I[k] \leftarrow$ MAX$(I[k], z_r)$

III. METHODOLOGY

This work enhances the context steering approach by interpreting sampled geometric data as two conflicting objective functions $f_d(x) = z_d \in [0, 1]$ (danger) and $f_i(x) = z_i \in [0, 1]$ (interest) with $x = 1, \ldots, n$ sample points. The notation $f(x)$ states that $f_d(x)$ and $f_i(x)$ can be applied respectively. From an agent's point of view, dangerous objects can be in the same direction as interesting objects, which result in an MCO problem to be solved.

A. Sampling Objective Values

We combine a set of steering behaviors B with context maps as described in Section II-C, whereby iterated objects O are either interest or danger. So they can be associated with their objective array $D = f_d(x)$ or $I = f_i(x)$. The set of context map receptors R determines the resolution of our discrete MCO problem with $n = |V| = |D| = |I|$. The

SAMPLEOBJECTIVES procedure (see Algorithm 1) aggregates discrete functions and additionally gets the inputs θ_ω as angle threshold and $[\xi_{min}, \xi_{max}]$ as perception range.

For simplification in this work, we only use SEEK. In practice, e.g., an agent's finite-state machine (FSM) controls active steering behaviors based on the current situation. For an improved run-time, iterated objects should be limited to objects relevant and visible for an agent, depending on the current scenario. Therefore, data structures that enhance spatial queries might be applied.

B. MCO Decision-Making

The two sampled discrete functions $f_d(x)$ and $f_i(x)$ form our 2D MCO problem and the 2D objective space, respectively. Since we formulate a minimization problem, we need to negate $f_i(x)$ to find solutions having minimum danger and maximum interest.

$$\begin{aligned} &\text{minimize} \quad \{f_d(x), -f_i(x)\} \\ &\text{subject to} \quad x \in \{1, 2, \ldots, n\} \end{aligned} \tag{5}$$

An agent sampling one interest and one dangerous object results in the typical ellipsoid MCO problem shape, as in Figure 3. The greater the number of sampled objects, the more concave problem shapes arise. For finding a Pareto-optimal solution that corresponds to a suitable movement vector (receptor) for each frame, we investigate three methods.

The first evaluated MCO solver is the *weighting method* [14], [15], although we expect that this approach might lead to sub-optimal results since it misses solutions in concave problem shapes. The configuration of the weighting coefficients $w_d, w_i \in [0, 1]$ with $w_d + w_i = 1$ selects specific solutions out of the Pareto-optimal set by collapsing the 2D problem to a conflict-free 1D function. Hence, a solution can easily be obtained by simply selecting the minimum value.

$$\begin{aligned} &\text{minimize} \quad w_d f_d(x) - w_i f_i(x) \\ &\text{subject to} \quad x \in \{1, 2, \ldots, n\} \end{aligned} \tag{6}$$

A more promising technique is the *ε-constraint method* [16], since it also considers concave solutions. The negated interest function $-f_i(x) \in [-1, 0]$ is defined to be minimized, whereby the danger function $f_d(x) \in [0, 1]$ is limited to a configurable upper bound $\varepsilon_d \in [0, 1]$ and applied as a constraint. Thus, an agent considers corresponding movement directions only with a configured limited danger.

$$\begin{aligned} &\text{minimize} \quad -f_i(x) \\ &\text{subject to} \quad f_d(x) \leq \varepsilon_d, \quad x \in \{1, 2, \ldots, n\} \end{aligned} \tag{7}$$

As a third approach, we test a *hybrid method* [17], [18], which represents a combination of the two preceding methods. Similar to the *ε-constraint method*, it is also capable of finding solutions in concave areas of MCO problem shapes.

$$\begin{aligned} &\text{minimize} \quad w_d f_d(x) - w_i f_i(x) \\ &\text{subject to} \quad f_d(x) \leq \varepsilon_d, \qquad x \in \{1, 2, \ldots, n\} \end{aligned} \tag{8}$$

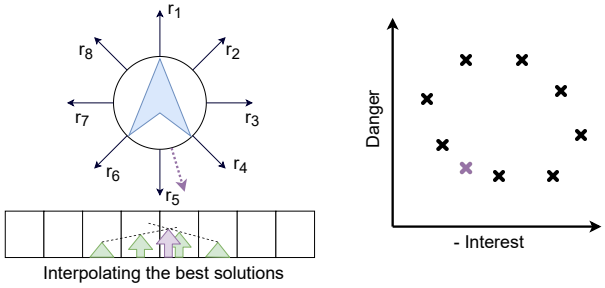


Fig. 4: Context interpolation of the Pareto-optimal solution (r_5); (left) determining the steering direction (adapted from Fray [8]); (right) resulting solution in objective space.

C. Gaussian Blurring and History Blending

To further improve results, we optionally apply two methods proposed by Fray for stabilizing decided directions before the MCO because more inert decisions might appear subjectively more natural [8].

Objective functions $f_d(x)$ and $f_i(x)$ can be smoothed. As implemented in this work, we separately apply an approximated discrete *Gaussian filter* $h_\sigma(x)$ with $\sigma = 0.4$. Since adjacent values decrease in variation, smoothing can reduce sudden changes between multiple MCO solutions from frame to frame.

$$h_\sigma(x) = \big[0.10558, 0.78884, 0.10558\big]^T \tag{9}$$

$$(f * h_\sigma)(x) = \sum_{k=-\infty}^{\infty} f(k) h_\sigma(x - k) \tag{10}$$

As a second method, we memorize discrete objective functions $f^{t-1}(x)$ of past frames and combine them with current values $f^t(x)$. So during MCO solving, agents favor solutions already made before. This method is our variant of a *history blending* approach, whereby the parameter $\alpha \in [0, 1]$ determines the proportion applied for blending past and present values.

$$(f^{t-1} + f^t)_\alpha(x) = \alpha f^{t-1}(x) + (1 - \alpha) f^t(x) \tag{11}$$

D. Context Interpolation

After MCO, we refine the Pareto-optimal solution by linear interpolation of directly neighboring values. This makes it possible to compensate for discretization disadvantages and prevents sudden unnatural changes in direction from frame to frame. Hereto, we use approximated gradients analogously for both objective functions. Supplemental to Fray's approach [8], we formalized the method for both adjacency directions:

$$\nabla f(x - 1) = f(x - 1) - f(x - 2) \tag{12}$$

$$\nabla f(x + 1) = f(x + 1) - f(x) \tag{13}$$

Next, let $z_r = m \cdot p_r + c$ be a linear equation (with slope m and intercept c) for interpolation point p_r to check

whether z_r has better interest without violating a possibly given danger constraint. This method is applied analogously for both existing adjacency directions. Moreover, we can interpolate the associated receptor w.l.o.g. $\vec{v_r} = p_r V[x-1] + (1-p_r)V[x]$ between the original receptor and its respective adjacent, as shown in Figure 4.

$$z_r = \nabla f(x-1) \cdot p_r + f(x-1) \tag{14}$$

$$z_r = \nabla f(x+1) \cdot p_r + f(x) - \nabla f(x+1) \tag{15}$$

$$p_r = \frac{f(x) - \nabla f(x+1) - f(x-1)}{\nabla f(x-1) - \nabla f(x+1)} \tag{16}$$

IV. Evaluation and Results

In this section, we describe the scenarios and metrics to test our approach. We evaluate and discuss different methods and parameter setups that we described in the previous section.

A. Test Scenarios

In our scenarios, we use standard use-cases in game-AI that consist of the following components:

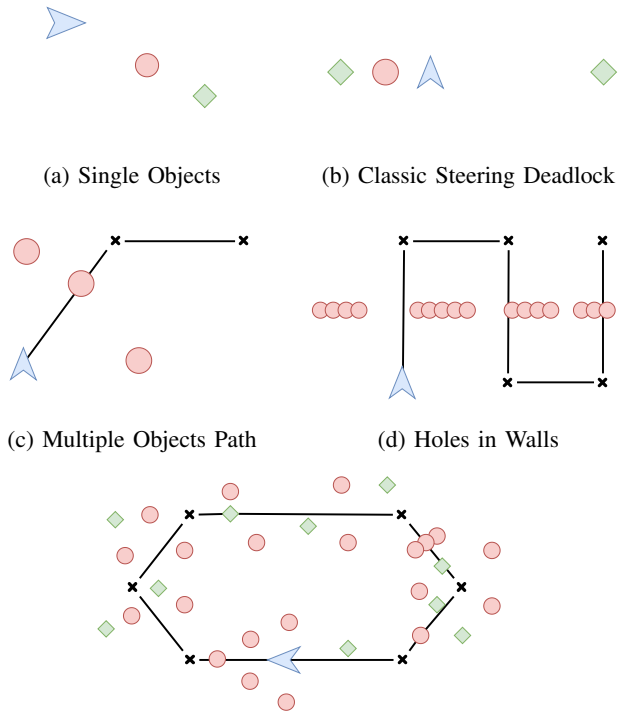
- **Agents**: triangular-shaped character that is only able to rotate and move forwards
- **Collectible**: objects of interest that can be picked up by the agent and result in objective *interest*
- **Obstacles**: objects that should be avoided and result in objective *danger*
- **Paths**: linear routes of ordered way-points that the agent should follow iteratively; The currently active way-point is perceived as objective *interest*

To measure the performance in each test case, we use the following metrics:

- **Pick-Ups:** number of collectibles the agent reached, which should be maximized
- **Collisions:** number of collisions between the agent and the obstacles, which should be minimized
- **Distance Traveled:** the accumulated euclidean distance between each update step of the agent, which is to be minimized as well

The scenarios are designed to highlight special features or flaws of the methods introduced in this work. They represent standard use cases as well as more complex situations that come close to real-world applications. To guarantee equal conditions in all tests, we define that each scene only consists of static instead of dynamically moving objects. Note, this is no limitation to our methods since the perception of the environment is frame-based within its fixed update frequency so that objects are perceived as static nevertheless. Additionally, it is easier to consistently design challenging benchmark situations. The following scenarios (cf. Figure 5) always contain a single agent and are described as follows:

a) Single Objects: The scene consists only of a single *interest* and a single *danger* object. From the start on, the agent cannot travel on a direct path to the collectible, since the dangerous object will block the way. As an optimal result, the agent reaches the collectible without colliding with the obstacle.



(a) Single Objects (b) Classic Steering Deadlock

(c) Multiple Objects Path (d) Holes in Walls

(e) Complex Object Arrangement

Fig. 5: Test scenarios: Agent = blue polygon, Collectible (*Interest*) = green diamond, Obstacle (*Danger*) = red circle, Path way point = cross, Path = black line

b) Classic Steering Deadlock: This scene describes conflicting solutions that need high-level logic. The agent is placed in the center. To the left, there is a collectible covered by an obstacle. To the right, there is a collectible that is slightly further away than the left one. In this situation, classic steering approaches [5], without additional case handling, fall into deadlocks since steering algorithms usually consider the left collectible more interesting. Hence, it completely ignores the additional circuitous route to overcome the obstacle.

c) Multiple Objects Path: In this scenario, the agent should move back and forth along a path covered with three obstacles. Optimal results exhibit no collisions, and a minimum traveled distance while reaching all way-points.

d) Holes in Walls: The scenario consists of multiple closely arranged obstacles that form game-typical structures like walls with passable holes. Therefore, the agent should follow a path and pass these holes whereby the world features holes of varying size. Especially the smaller ones require careful navigation to avoid collisions with any of the obstacles.

e) Complex Object Arrangement: The last scenario consists of a problem space that is typical for racing or space games. The agent should follow a closed path (circuit), whereby it can pick up collectibles. In an optimal result, the agent manages to reach all collectibles without colliding with any obstacles while minimizing the distance traveled.

TABLE I: Variable parameters whose modification is evaluated.

	Variable parameter	Default value
Context steering	Context map resolution r	32
	Angle threshold θ	89°
	History blending, Gaussian blurring	Disabled
MCO	Danger weight w_d	0.49
	Interest weight w_i	0.51
	Danger constraint ε_d	0.56
	Context interpolation	Enabled

TABLE II: Test-results based on the default parameterization.

	MCO	Pick-Ups	Collisions	Distance
a) Single objects	Weighting	$1/1$	0	8.98
	Constraint	$1/1$	0	6.94
	Hybrid	$1/1$	0	8.72
b) Classic steering deadlock	Weighting	$2/2$	0	18.38
	Constraint	$2/2$	0	15.33
	Hybrid	$2/2$	0	17.38
c) Multiple objects	Weighting	$0/0$	1	44.11
	Constraint	$0/0$	0	48.74
	Hybrid	$0/0$	0	∞
d) Holes in walls	Weighting	$0/0$	4	77.05
	Constraint	$0/0$	0	82.34
	Hybrid	$0/0$	0	∞
e) Numerous objects	Weighting	$8/10$	2	129.55
	Constraint	$10/10$	0	179.61
	Hybrid	$9/10$	0	210.86

B. Parameters

This work introduces a variety of parameters that can be categorized either as *variable* or *fixed*. Fixed parameters remain the same for each scenario and test run so that they depict fundamental conditions for all examinations. These parameters are sub-grouped to *Context steering* (perception range, objective mapping weights, and active steering behavior), *AI mechanics* (velocity magnitude, AI update frequency, and agent rotation speed), and the *Scenario setup* (radius of collision boundary). Most remarkable in this setup are the objective mapping weights such that the agent slightly favors collectibles over waypoints since they are mapped to the same objective *interest*. In contrast, the variable parameters directly influence the behavior and decision of the agent, respectively. Table I contains all variable parameters and their default values whose modifications and resulting influences are evaluated. As before, parameters are classified corresponding to their thematic affiliation. The post-processing methods such as history blending and Gaussian blurring are deactivated by default to obtain initial unbiased results. The default writing condition angle ensures that selected context map vectors must result in a positive dot product when compared with steering vectors. Interpolation is used by default to support the fluent decision-making process.

C. Results and Observations

As shown in Table II, the three methods perform differently well. Most notably, the constraint method has shown superior performance in all scenarios since it outperforms the two other methods in all metrics most of the time. The constraint method is the only one that reaches all collectibles, never hits an obstacle, and always finishes the test scenarios. In scenario *c* and *d*, the weighting method performs better in terms of the traveled distance but collides with obstacles. In contrast, the hybrid method gets stuck in deadlocks in these two scenarios.

We observed that the agents move smoothly for the scenarios with a single obstacle but sometimes tend to jitter if there are multiple obstacles due to slight rotational updates between the time steps. In this case, the agent cannot decide to move either left or right since both are valid and of equal quality. In the following, we show the influence of the variable parameters.

1) Context Map Resolution: Lowering the resolution to a minimum of $r = 8$ has no negative impact in the first three scenarios but leads to more unstable movement so that agents fluctuate between decided directions. In scenario *d* and *e*, the traveled distances become significantly larger. A greater resolution leads to more stable results and better overall performance in terms of our metrics but comes at a higher computational cost because of the linear run-time complexity. Hence, high resolutions are not needed since our methods significantly benefit from the context interpolation in Section III-D. To obtain similar, jittery-free results without interpolation, the resolution has to be greater by a factor of two for the first three scenarios and by a factor of three for the last two scenarios. Hence the resolution correlates with the complexity of the scene.

2) Writing Condition Angle: Lesser values reduce agents' awareness and lead to a more strict movement behavior so that agents are less sensitive to obstacles. This slightly reduces the distances traveled. For angles $\theta < 50°$, the number of collisions increases significantly for all MCO methods. Increased angles destabilize the movement of agents and increase jittery movement. Angles with $\theta > 120°$ have shown to result in deadlocks for all MCO methods.

3) History blending and Gaussian blurring: If enabled, the context steering significantly stabilizes movement behavior and reduces jittery without altering numeric results in Table II. Only traveled distances are slightly affected, but their proportions remain the same. The constraint method benefits the most.

4) Danger/Interest Weight: Lesser *danger* weights w_d result in greater *interest* weights w_i and vice versa due to the partition of unity $w_d + w_i = 1$. So, w.l.o.g. only the modification of the danger weight is examined. The effects are similar for each scenario. Danger weights $w_d < 0.5$ result in careless agents that move closer to obstacles and increase the number of collisions. In contrast, danger weights $w_d > 0.5$ increase traveled distances and disable agents to pick up most of the collectibles in the last scenario. The latter has shown to also increase the number of deadlocks.

5) Danger Constraint: As for weights, the effects of altering constraints are similar for all scenarios. For constraints $\varepsilon_d < 0.5$, the number of collisions increases significantly, whereby values $\varepsilon_d > 0.6$ disable agents to pick up most of the collectibles in the last scenario and provoke deadlocks.

6) Context Interpolation: Disabling context interpolation does not alter numeric results except for increasing traveled distances, but it dramatically destabilizes the overall movement behavior.

7) General Observations: The complexity of the problem space correlates with the number of objects in the scene. Therefore, the constraint method shows the best overall performance. Problems arise if the collectibles are nearly co-linear since the problem space degenerates to a line, causing the agent to jitter, especially using the constraint method. The post-processing methods yield a relief since history blending supports the agent to stay constant. A practical advantage of the weighted and the constraint method is the more straightforward setup since there is only a single control parameter each, w or ε, to adjust the overall behavior even though multiple behaviors will increase the complexity.

V. Conclusion and Future Work

In this work, we presented a fast, optimal, and natural behaving extension to the problem of multi-objectives that arise from the basics of context steering [8] and overcomes the problems of classic steering [5]. Therefore, we applied and compared three different multi-criteria optimization algorithms whereby the constraint method performs best and offers an easily understandable control parameter ε. This affects how brave an agent acts, i.e. how close it moves to obstacles. Due to the power of combining different behaviors, game designers are able to build natural and robust movement controllers that are based on local decision-making. With a few parameter changes, different AI personalities can emerge from the designers choices. Further, widely used path-finding methods can easily be integrated using waypoints as collectibles.

The evaluation shows that agents move autonomously and naturally most of the time, but in some circumstances, they tend to jitter, e.g. if there is a large number of objects in the scene such that the context map becomes unstructured. We recommend limiting the perception range as well as the use of post-processing methods such as history blending and context interpolation. Based on our observations, the latter enables the possibility to reduce the resolution of the sensor to improve the computation time and enable features like AI level-of-detail with respect to the sensor resolution, as proposed by Fray [8].

Even though we showed that our proposed method yields reasonable and stable decisions that overcome the flaws of classical steering approaches [5] and provide rich extensions to the problem of multi-objectives that arise from the basics of context steering [8], other problems are yet to be solved. More advanced MCO algorithms might be more suitable, such as the elastic constraint method or Benson's method [19]. Since context steering is an extension to the basic steering algorithms, it contains the same amount of parameters. Additionally, a new parameter for the MCO-solver has been introduced. Hence, combinations of behaviors result in a large set of parameters to be optimized so there is a need for automated parameter tuning. Evolutionary algorithms or neural networks may prove suitable in tuning such a large amount of parameters. Additionally, some parameters might be more sensitive to specific situations. Such correlations need to be examined to allow for effective online tuning of the parameters. In this work, we focused on 2D planar-shaped sensors, where context interpolation is an effective way to smooth the movement of an agent and effectively reduce the number of required receptors on the sensor, and hence the computational effort. Since a 3D spherical sensor comes with a naturally high number of receptors, there is a need to keep their count as small as possible. Unfortunately, this is non-trivial for three-dimensional problem spaces since the intersecting lines of the gradient descent of the neighboring receptors in 2D become intersecting planes resulting in an over-determined linear system in 3D. Therefore, other context interpolation methods would be required to effectively reduce the number of required sensors in 3D space.

References

[1] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[2] ——, "Correction to a formal basis for the heuristic determination of minimum cost paths," *SIGART Bull.*, no. 37, pp. 28–29, 1972.

[3] X. Cui and H. Shi, "An overview of pathfinding in navigation mesh," *International Journal of Computer Science and Network Security*, vol. 12, no. 12, pp. 48–51, 2012.

[4] W. G. van Toll, A. F. Cook, and R. Geraerts, "A navigation mesh for dynamic environments," *Comput. Animat. Virtual Worlds*, vol. 23, no. 6, pp. 535–546, 2012.

[5] C. W. Reynolds, "Steering behaviors for autonomous characters," in *Game developers conference*, vol. 1999, 1999, pp. 763–782.

[6] M. Kirst, "Multicriteria-optimized context steering for autonomous movement in games," *Master's thesis, Otto-von-Guericke University Magdeburg*, 2015.

[7] A. Fray, "Steering Behaviours Are Doing It Wrong," https://andrewfray.wordpress.com/2013/02/20/steering-behaviours-are-doing-it-wrong/, Feb. 2013, accessed: 2021-04-03.

[8] ——, "Context steering: behavior driven steering at the macro scale," in *Game AI Pro 2: Collected Wisdom of Game AI Professionals.* CRC Press, 2015, pp. 183–193.

[9] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th annual conference on Computer graphics and interactive techniques - SIGGRAPH '87.* ACM Press, 1987.

[10] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research & Management Science. Springer, 1999.

[11] A. Fray, "Context steering," *Game AI Pro 360: Guide to Movement and Pathfinding*, 2019.

[12] A. Gerdelan and C. O'Sullivan, "A genetic-fuzzy system for optimising agent steering," *Computer Animation and Virtual Worlds*, vol. 21, no. 3-4, pp. 453–461, 2010.

[13] A. Croitoru, "Deriving low-level steering behaviors from trajectory data," in *2009 IEEE International Conference on Data Mining Workshops*. IEEE, 2009, pp. 583–590.

[14] S. Gass and T. Saaty, "The computational algorithm for the parametric objective function," *Naval Research Logistics Quarterly*, vol. 2, pp. 39–45, 1955.

[15] L. Zadeh, "Optimality and non-scalar-valued performance criteria," *IEEE Transactions on Automatic Control*, vol. 8, no. 1, pp. 59–60, 1963.

[16] Y. Y. Haimes, L. S. Lasdon, and D. A. Wismer, "On a bicriterion formulation of the problems of integrated system identification and system optimization," *IEEE Transactions on Systems Man and Cybernetics*, vol. 1, no. 3, pp. 296–297, 1971.

[17] R. E. Wendell and D. N. Lee, "Efficiency in multiple objective optimization problems," *Mathematical Programming*, vol. 12, no. 1, pp. 406–414, 1977.

[18] H. Corley, "A new scalar equivalence for pareto optimization," *IEEE Transactions on Automatic Control*, vol. 25, no. 4, pp. 829–830, 1980.

[19] M. Ehrgott, *Multicriteria Optimization*, 2nd ed. Springer, 2005.