Generalizations of Steering - A Modular Design

Lars Wagner, Christopher Olson
Faculty of Computer Science
Otto von Guericke University
Magdeburg, Germany
{lars.wagner; christopher.olson}@ovgu.de

Alexander Dockhorn
Faculty of Electrical Engineering and Computer Science
Leibniz University
Hannover, Germany
dockhorn@tnt.uni-hannover.de

***Abstract*—Steering and its many variants have been used to model multi-agent behavior in simulated worlds. Each variant of the steering algorithm introduces its own concepts and components. This work aims to generalize those in a modular building block design which allows to recreate existing steering algorithms and draft new versions. The concept is demonstrated in small-scale study on aggregation functions which shows the impact of a single building block on the agents' behavior.**

***Index Terms*—Steering, Context Steering, Aggregation Functions, Dempster-Shafer Theory**

I. Introduction

Path-finding for large swarms of agents is a complex problem in which even modern solutions can struggle to find a solutions due to the large search space [1]. Steering [2] has shown to be an effective alternative to path-finding for controlling many-agents. In classic steering, each agent consists of multiple simple behaviors, which each propose a target movement direction based on the local neighborhood of an agent. The final movement direction is simply determined by aggregating the decisions of each behavior.

Since classic steering methods do not use any information besides the current neighborhood of an agent, deadlocks can frequently occur. While those deadlocks might not be apparent in the observation of the general behavior of a swarm, they may break the immersion of a player that directly perceives it. Therefore, many methods have made extensions of the classic approach, to avoid frequent deadlock situations.

Arguably, the most prominent among them is the context steering approach by Fray [3]. By not just aggregating the target directions of an agent's behaviors but actually comparing their preferences for each possible direction, more information is available during the aggregation process. The resulting steering algorithm adds more parameters to the overall steering process, but tuned agents have shown to result in more reliable behavior.

One of the big changes in context steering was the introduction of context maps. Those represent different objectives that the agent can follow or avoid, such as danger and interest. While every singular behavior will modify the values of these context maps, filled maps need to be aggregated to make a selection of the agent's final movement direction. The aggregation approach by Fray [3] simply filters the directions with a high danger value and selects among the remaining directions with the highest interest value. The approach has been generalized to a multi-objective context steering proposed by Kirst [4], [5] in which multi-objective decision-making functions are used to select the most promising direction.

The works by Fray and Kirst have each observed drawbacks of specific algorithmic components in the steering process. To overcome them, they have replaced components with the result of a more general steering method. First interested in overcoming the stiffness of the aggregation process by Fray [3], we have been working on its generalization to arbitrary functions and studied their impact on the agent's behavior. Intrigued by this approach, we continued by breaking up the steering algorithm into mostly independent building blocks.

In the following section, we will be presenting our modular steering approach, which is based on components that are frequently found in the existing works on steering. Naturally, previous algorithms can be seen as instances of the general steering concept, while the latter will help in the design of new alternatives. As one example, we discuss possible implementations of the aggregation mechanism in Section III. We conclude the discussion of our steering model in Section IV and mention perspectives for future work.

II. Generalizations of Context Steering

We believe that splitting up the steering algorithm into configurable components will make it easier to identify alternative implementations and design custom agents for specific environments. Given an analysis of previous work, we boiled down the concept of steering to *perceiving* (and *memorizing*) the environment to *make decisions by aggregating* the preferences of *multiple behaviors*. Furthermore, we emphasize the importance of *execution and control* in complex physical simulation and real-world environments. Highlighted words mark the building blocks of our generalized steering concept which is also depicted in Figure 1. The following subsections will discuss the components, their addressed problems, and some of their implementations in more detail.

A. Perception

The perception component ensures the observation of the agent's current surroundings, but may also include global information to better guide the decision-maker. In previous

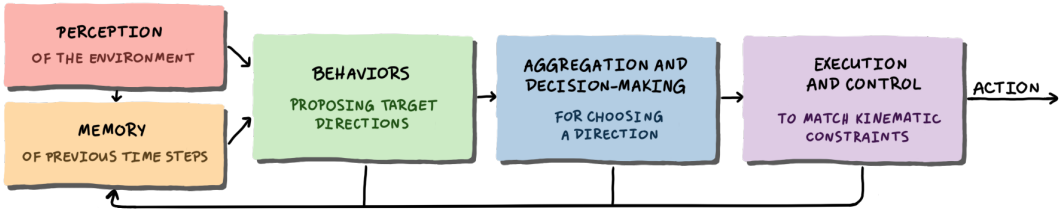

Fig. 1: Building blocks and reference implementations of generalized steering.

work, the perception of steering agents has usually been ensured by directly accessing an object's properties or using ray-casts/laser sensors to measure the angle of and distance to the agent [2]. Other works have added a globally planned path to a distant object, which the agent should follow [5] to help navigating large environments. Other sensors include but are not limited to radar/lidar and 2D/3D cameras. The quantity and quality of these sensors impact the amount of information that the agent needs to process. For instance, increasing the number of ray casts, will increase the granularity and can result in smoother movement. Additionally, the fusion of information from multiple modalities is a challenging task but may provide the agent with an even better picture of its environment [6].

Steering in simulated environments (e.g., games) often provides the agent with information on the type of a perceived object since this information is readily available in the object's data. The same may be hard to ensure in real-world environments, in which the classification of an object may depend on multiple observable or even hidden properties. While simple agents might just care for an object's color, other implementations are based on markers or sophisticated image classification [7].

Next to a single agent that perceives its environment, a multi-agent swarm can be considered. Here, agents can be enabled to share information via communication. This may be used to localize themselves in the swarm or to be aggregated for even more information on the environment.

B. Memory

While previously described steering approaches aim to be memory-less, this necessarily results in an agent that is prone to deadlocks. Similar to local path-finding [8], storing information on the environment may help the decision-making in later time steps. A simple approach has been proposed in terms of history blending [5], in which the previous time step ($f(x_{t-1})$) or its context values are remembered and combined in a weighted average (α) with values from the current time step ($f(x_t)$).

$$h(x_t, x_{t-1}, \alpha) = \alpha f(x_t) + (1 - \alpha) f(x_{t-1}) \qquad (1)$$

This approach can easily be extended to store data from multiple previous time steps. However, this linearly increases the amount of data storage required given the number of time steps to be stored. As an alternative, we can make use of a latent vector representation as it is used in recurrent neural networks. Training such a network, e.g. an LSTM, has shown to effectively store information of previous time-steps [9].

Next to storing information on the local neighborhood of the agent, we can use previous observations to build a map of static obstacles. Simultaneous Localization and Mapping (SLAM) [10] tracks the agent's position and merges the agent's observations to build a consistent map of its environment. Both can be used to define heuristics to effectively avoid deadlocks by punishing areas that have already been visited or setting exploration goals for yet unvisited areas. Similar approaches are used in heuristic search [11] to make locally optimal decisions and guarantee finding a solution. Extending SLAM to multi-agent scenarios is still an active research topic [12] with potential applications in multi-agent steering.

C. Behaviors and Movement Preferences

Behaviors are one of the key ideas of the steering concept. Instead of defining a single complex behavior, we split it into multiple simple behaviors. Those can be much easier to design and implement, and their combination has the potential to result in much more complex behavior. Each behavior should process information made available by the agent's perception and memory components. Thereby, they can either propose a single target vector, output a preference over possible movement directions, or return a probability distribution over all directions.

Typical behaviors include a wide range of strategies for moving toward points of interest or avoiding dangerous areas [13] or circling around a target. More sophisticated behaviors use the information of the perception and memory components to predict the movement of dynamic objects and either cut their way or flee from them more effectively [5]. Furthermore, an agent's behavior may take its role or position in a swarm into account and respond accordingly [14].

D. Aggregation and Decision-Making

Aggregation and decision-making are so closely coupled, that we cannot break them apart into two separate building blocks. It is often up to the decision-making algorithm how the information is to be processed to make a decision.

While classic steering simply adds up all the target vectors returned by the behaviors, context steering first aggregates them into context maps (e.g., interest or danger) and then selects a target direction. The standard aggregation method is storing the maximal preference per behavior in the respective context map. It boils down the available information to a single value while information on the distribution is lost.

We propose to generalize this procedure by defining a function that aggregates the target directions or preferences

of all behaviors. Thereby, the aggregation function becomes an interesting end-point for (deep) learning-based approaches. Apart from that, other descriptive measures can be used, e.g. mean or median preference for a robust aggregation, or min-preference for designing a risk-aware agent. Similarly, probabilistic, information-theoretic methods and fuzzy approaches [15], [16] can be implemented to aggregate the preferences of all behaviors.

Traditionally, the two objectives of interest and danger have been considered in steering. While danger values have been used to filter the set of possible directions, the use of multi-objective decision-making algorithms has been proposed by Dockhorn et al. [5] to allow for a more granular decision-making process. This opens algorithms to handle more than two objectives. Additionally, we can use Pareto-dominance to filter movement directions and weighting-based approaches to change the priority of our objectives.

E. Execution and Control

Moving from simulations to real-world environments results in uncertainties in the actuators and sensors [17]. To compensate those, we introduce an execution and control component which acts as a mediator between the agent and its environment. This component can be used to adapt to the agent's kinematic constraints such as maximum speed/acceleration and compensate for the agent's size for avoiding collisions.

III. Comparison of Aggregation Methods

One key item of the proposed building blocks is the aggregation function which has previously not received much attention. We demonstrate the impact of the aggregation function in two experiments, i.e. an escape and a cooperative collection task. Figure 2 shows the two exemplary scenarios, which use a similar room layout. In the escape scenario, all agents need to reach one of the two escape points to be safe. During the cooperative collection task, we randomly place items in the environment, which need to be collected by the agents. A total of 200 items will have to be collected, while a maximum of 10 will be active at the same time.

In our test, we will be using an agent with the following setup: **Perception:** A radial sensor with a radius that covers about one-tenth of the room will be used. Collectible items, walls, and other agents will be recognized while being in this radius. The positions of the escape points will always be known to the agents. A total of 16 steering directions will be evaluated. **Memory:** The perception of the previous time-step will be stored and used via history-blending (cf. Equation (1)). **Behaviors:** Two simple behaviors will be used. The seek behavior will guide the agents closer to the target while returning a maximal value in case a steering direction directly points towards the target. A flee behavior will be used to steer away from close dangers. **Aggregations and Decision-Making:** The agent's movement direction will be determined using context steering. Interest and danger will be processed separately. We will compare the effects of different aggregation functions to combine multiple context values. Used aggregation

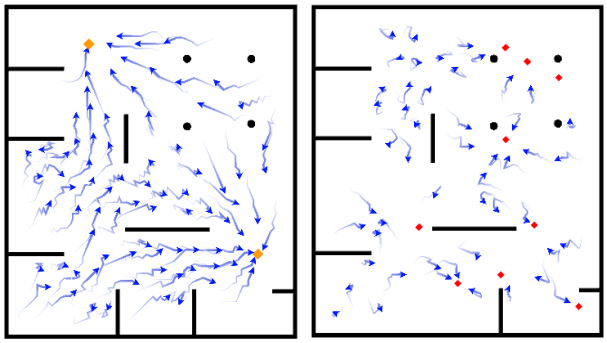

Fig. 2: Escape (left) and collection (right) evaluation settings.

functions will be described below. **Execution and Control:** In our simulated environment, agents will always walk directly in the direction of the chosen movement direction with constant speed. Drag will be ignored.

Context steering as proposed by Fray [3] has been using the maximum operator as an aggregation function. We exchange this function with the median and sum of context values overall perceived objects. Additionally, we propose Dempster-Shafer aggregation as an information-theoretic approach to the aggregation of context values. Here, each object $o \in O$ in the agent's vicinity is defining a mass distribution m_o over all sensors representing the preference for the corresponding movement direction and the certainty (c_o) as the a reciprocal function of its distance to the agent. Masses are multiplied by the object's confidence and the uncertainty is used in remaining calculations ($u_o = 1 - c_o$). Masses are combined by a variation of Dempster's rule of combination which takes the uncertainty of each mass assignment into account and determines the joint mass ($m_{1,i}$) of the first i objects by:

$$\text{for } i \text{ in } \{1, \ \ldots, \ |O|\}:$$
$$m_{1,i} = u_i m_{1,i-1} + m_{1,i-1} m_i + u_{1,i-1} m_i$$
$$u_{1,i} = u_{1,i-1} u_i$$

In the escape experiment, we have tracked the time it takes for a number of agents to reach an exit point. We repeated the experiment for 25 trials while randomizing the initial agent positions. Sometimes agents get trapped in a deadlock situation. While this rarely happened, we decided to stop each simulation after one minute, declaring all agent's left as stuck. With an average escape time of about 10 seconds, this has shown to be more than enough time for agents to leave. Figure 3 shows the evacuation times of the first 90 out of 100 agents and the distribution of agents getting stuck. During the collection task, we tracked the time it takes to collect a number of items and the relative time in between two collection events. Figure 4 shows the respective results.

Both experiments show the large impact of the aggregation function. In both experiments, the Dempster-Shafer aggregation rule performed best, closely followed by Sum and Max. Median has been the worst option during the collection task, while the Sum operator seems to be most prone to deadlocks.

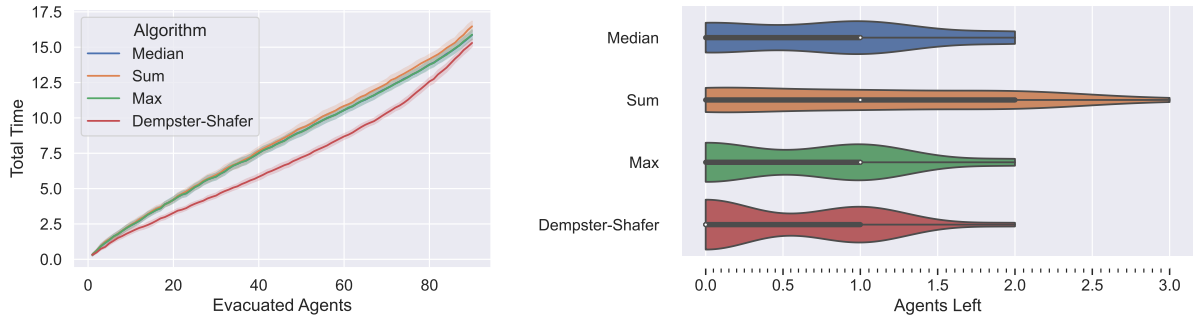

Fig. 3: Comparison of aggregation functions for completing the escape scenario tracking the total time it takes agents to evacuate and the distribution of agents getting stuck.

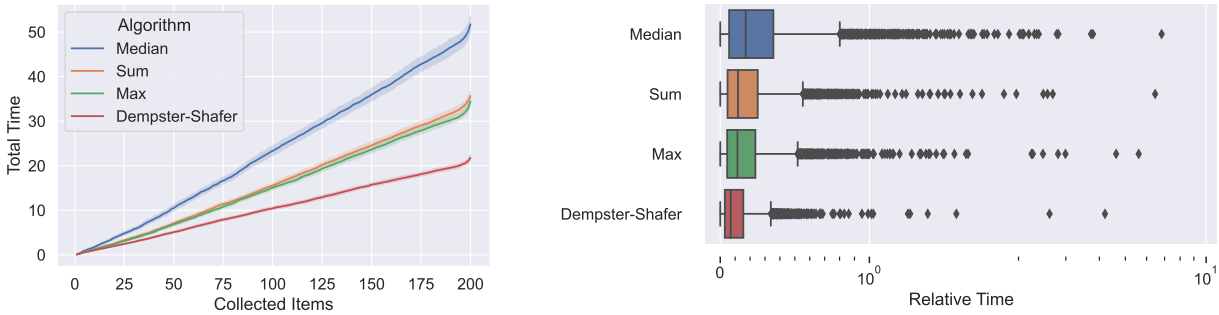

Fig. 4: Comparison of aggregation functions for completing the collection scenario tracking the total time it takes agents to collect a number of items and the distribution of the time in between tracking two targets.

IV. Conclusion

Motivated by the observations we made while reading previous works on steering, we proposed a general building block model for steering algorithms. We believe that this structure can serve as a guideline to identify and design new steering algorithms in the future. We used the design to create four context steering agents with differing aggregation functions and compared their performance in two tasks. This is far from a comprehensive comparison of aggregation functions but shows the great impact such a small parameter can have. In upcoming experiments, we want to further compare different implementations of each building block and their impact on the behavior of a single agent and a swarm.

References

[1] S. Mai and S. Mostaghim, "Modeling pathfinding for swarm robotics," in *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 190–202.

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

[3] A. Fray, "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.

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

[5] A. Dockhorn, S. Mostaghim, M. Kirst, and M. Zettwitz, "Multi-objective optimization and decision-making in context steering," in *IEEE Conference on Games, COG*. IEEE, 2021. [Online]. Available: https://doi.org/10.1109/CoG52621.2021.9619155

[6] H. Wu, M. Siegel, R. Stiefelhagen, and J. Yang, "Sensor fusion using dempster-shafer theory [for context-aware hci]," in *IMTC/2002. Proceedings of the 19th IEEE Instrumentation and Measurement Technology Conference (IEEE Cat. No.00CH37276)*, vol. 1, 2002, pp. 7–12 vol.1.

[7] D. Kragic and M. Vincze, "Vision for robotics," *Found. Trends Robot*, vol. 1, no. 1, p. 1–78, jan 2009.

[8] F. Peralta, M. Arzamendia, D. Gregor, D. G. Reina, and S. Toral, "A comparison of local path planning techniques of autonomous surface vehicles for monitoring applications: The ypacarai lake case-study," *Sensors*, vol. 20, no. 5, p. 1488, Mar. 2020. [Online]. Available: https://doi.org/10.3390/s20051488

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997.

[10] H. Temeltas and D. Kayak, "Slam for robot navigation," *IEEE Aerospace and Electronic Systems Magazine*, vol. 23, no. 12, pp. 16–19, 2008.

[11] R. E. Korf, "Real-time heuristic search," *Artificial Intelligence*, vol. 42, no. 2, pp. 189–211, 1990.

[12] M. Kegeleirs, G. Grisetti, and M. Birattari, "Swarm slam: Challenges and perspectives," *Frontiers in Robotics and AI*, vol. 8, 2021.

[13] 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*. ACM Press, 1987.

[14] P. UG, "Polarith ai documentation," http://docs.polarith.com/ai/, 2020, accessed: 2020-02-28. [Online]. Available: http://docs.polarith.com/ai/

[15] I. L. Bajec, M. Mraz, and N. Zimic, "Boids with a fuzzy way of thinking," in *Proceedings of ASC 2003*, 2003, pp. 58–62.

[16] M. Mraz, N. Zimic, and I. L. Bajec, "Fuzzy model of bird flock foraging behavior," in *Proc. EUROSIM 2007*, 2007.

[17] S. Mai, N. Traichel, and S. Mostaghim, "Driving swarm: A swarm robotics framework for intelligent navigation in a self-organized world," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022.