

## Project Outline

Define metrics to compare the coverage, quality and efficiency of steering algorithms

Perform initial experiments on algorithm parameters independently to study parameter effects on metrics

Create a framework that can automatically perform a multivariate parameter dependent analysis to search for optimal steering algorithm parameter settings

## Metrics

### Coverage:

The success  $succ(A)$  of a steering algorithm  $A$  is computed as the ratio between the number of scenarios in a scenario set that a steering algorithm can successfully solve (reference agent reaches its goal within a time limit without any collisions).

### Quality:

The quality with respect to distance  $qual(A)$  of a steering algorithm  $A$  is calculated with respect to the reference agent. The value is normalized based on the static optimal path for the reference agent to reach its target in the given scenario.

### Efficiency:

A metric that quantifies the computational efficiency  $eff(A)$  of an algorithm  $A$ . Unlike the previous metrics, it is not straightforward to provide an ideal upper bound for  $eff(A)$ .

To provide a basis for normalization, we assume that 10% of all computational resources are allocated to the steering algorithm.

For every scenario, the maximum time  $t_{max}$  allocated to every steering agent per frame is  $\frac{0.1}{desiredFPS \cdot N}$  seconds, where  $N$  is the number of agents in the scenario.

$t_{avg}$  is the average time spent computing a steering decision per frame for all agents.

$$eff_s(A) = \begin{cases} 1 & \text{if } t_{avg} < t_{max} \\ \frac{t_{max}}{t_{avg}} & \text{otherwise} \end{cases} \quad eff(A) = \frac{\sum_{s \in S} eff_s(A)}{|S|}$$

$eff(A)$  is the aggregate value of  $eff_s(A)$  over a set of scenarios  $S$ .

## Optimization Formulation

Define a weighted objective function:

$$f(A, w_1, w_2, w_3) = w_1 \cdot succ(A) + w_2 \cdot qual(A) + w_3 \cdot eff(A)$$

where  $w_1, w_2, w_3$  are normalized weights which determine the influence of coverage, quality, and performance on the objective respectively.

Finding the optimal steering algorithm parameters that maximize the objective is formulated as the following optimization problem:

$$v_{opt}^{(w_1, w_2, w_3)} = \max_{v \in V} f(A, w_1, w_2, w_3)$$

where  $V$  is the vector of parameters for algorithm  $A$ .

## Uni-Variate Analysis

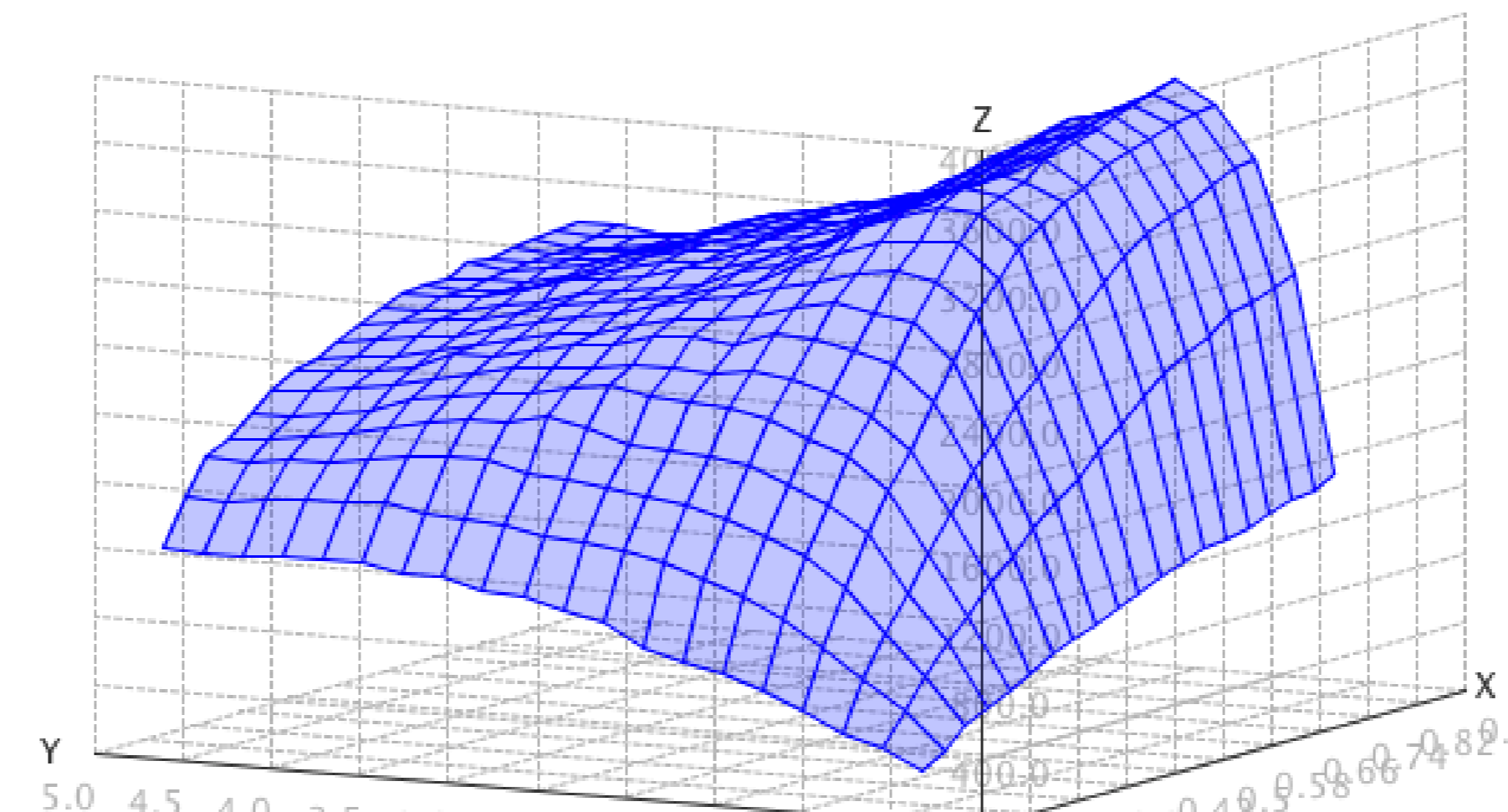
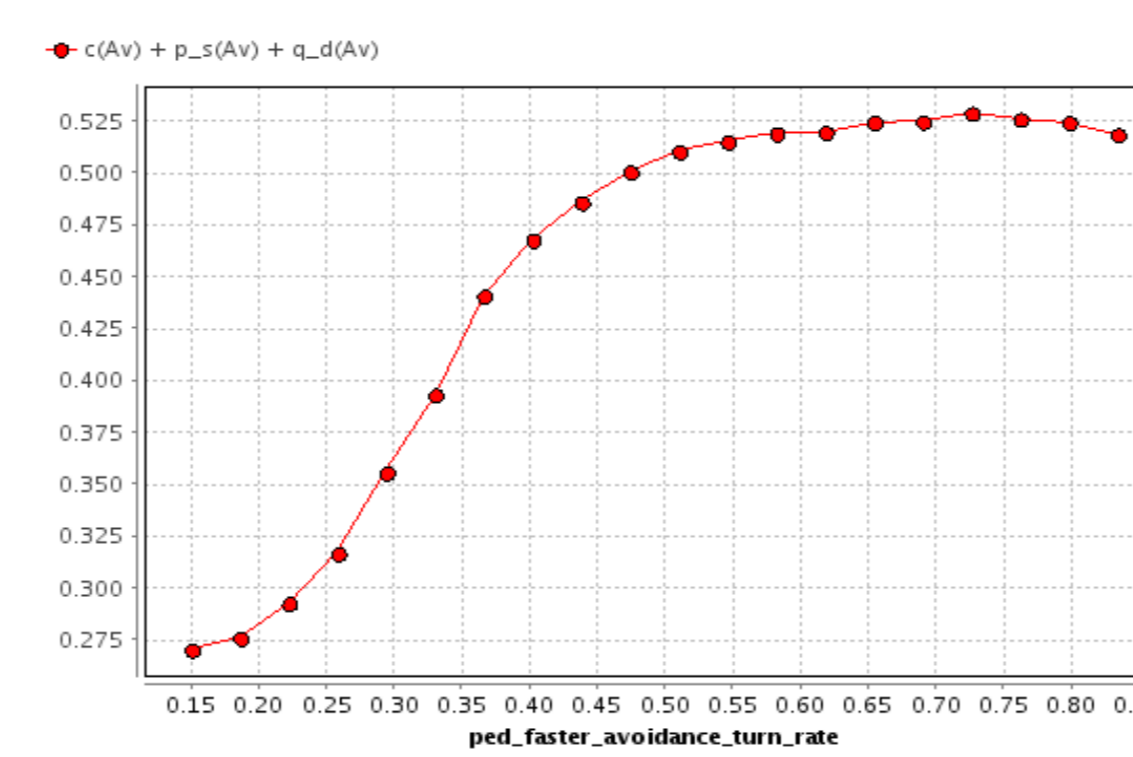
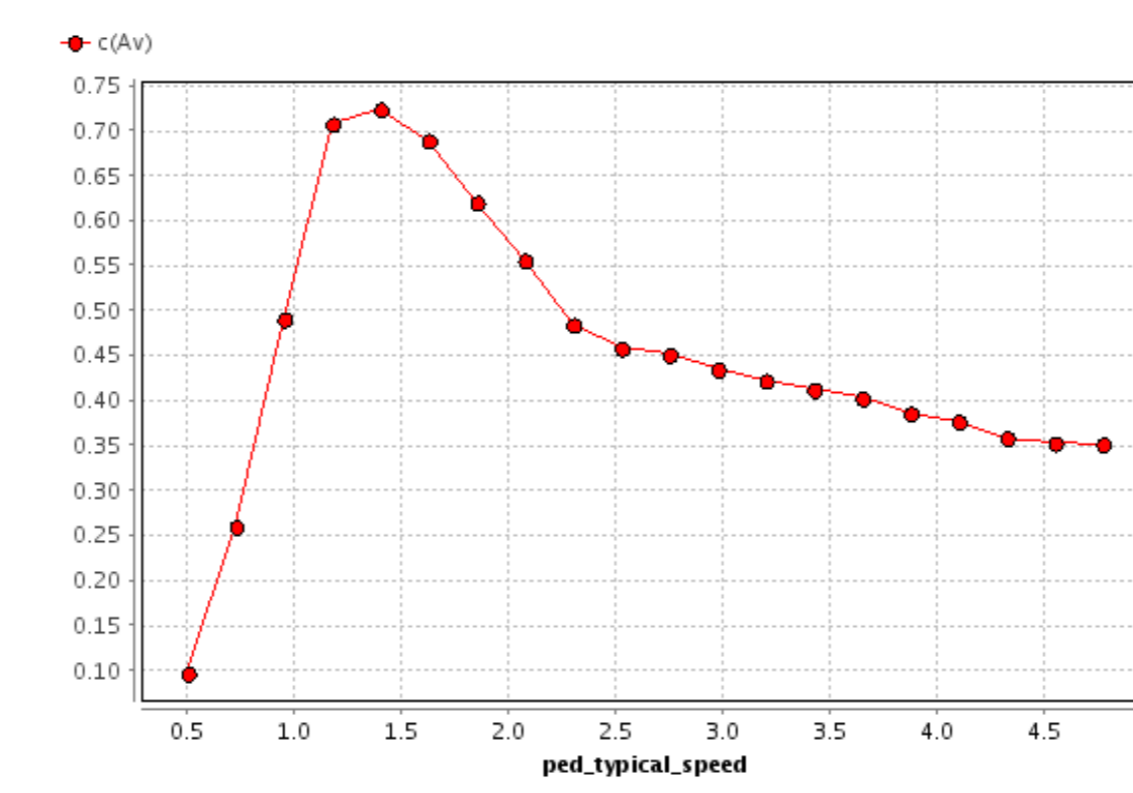
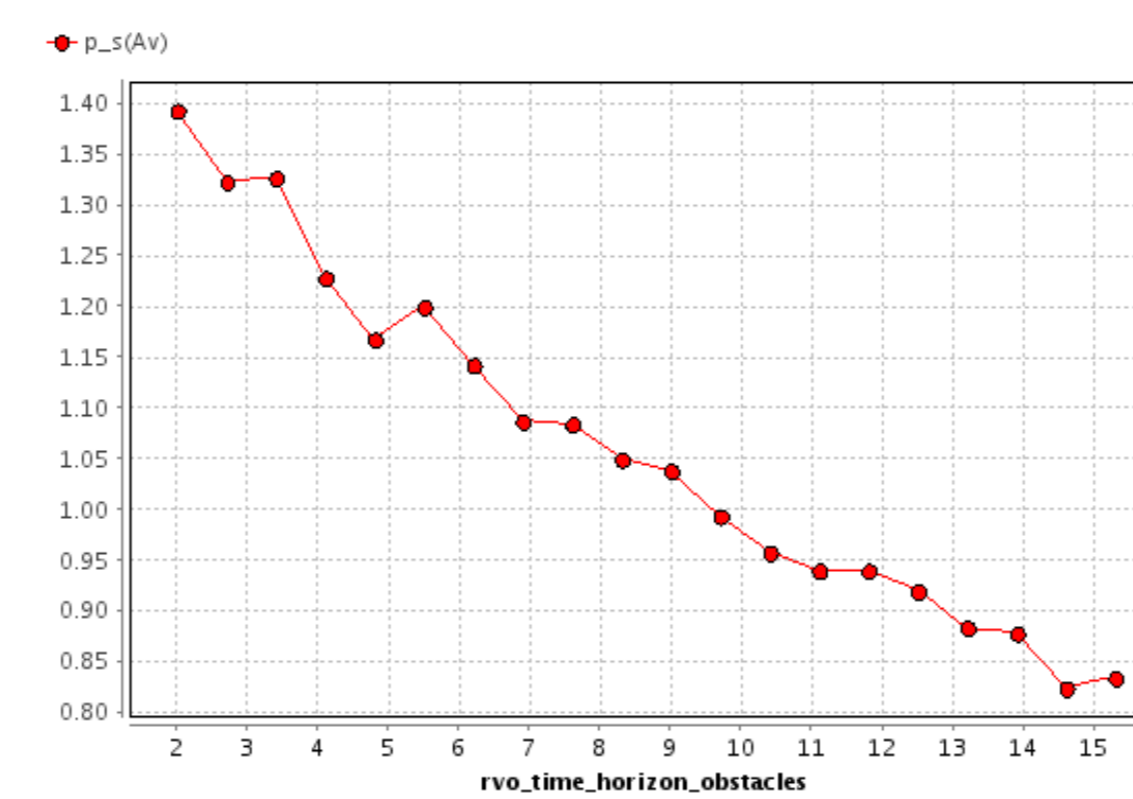
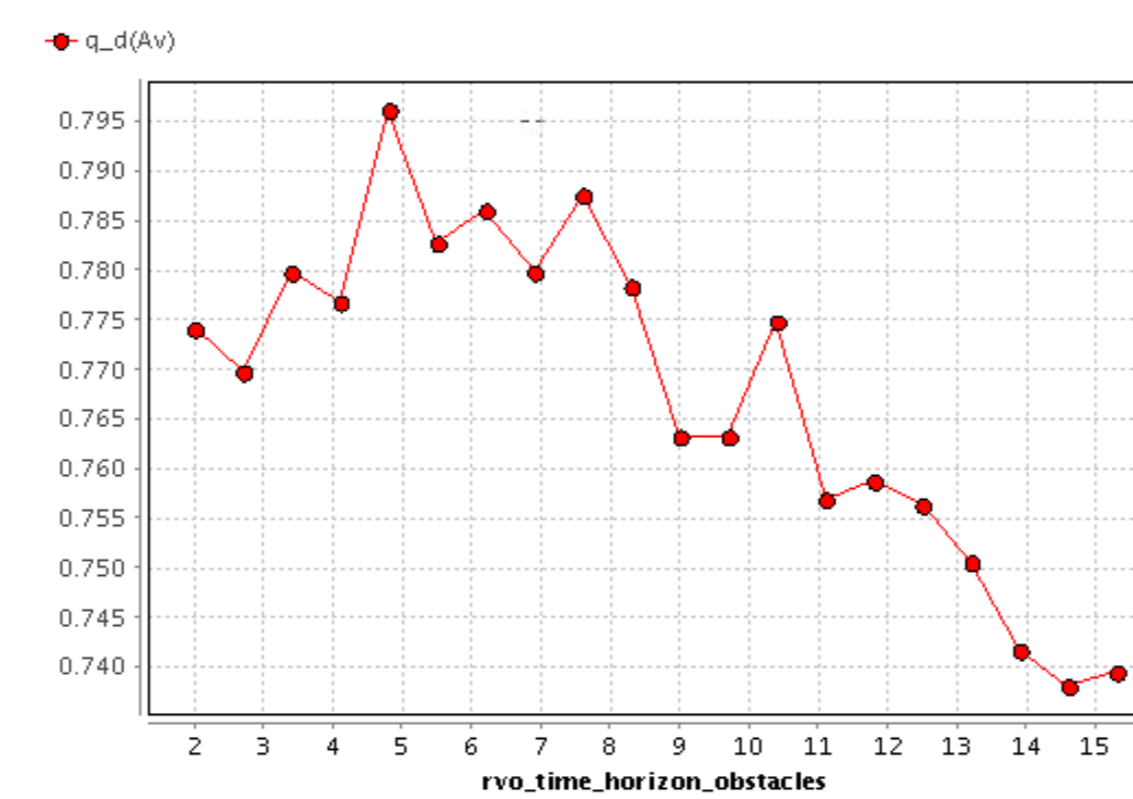


Figure 1: The left 4 graphs illustrate the behaviours of selected parameters. The above graph shows the results of evaluating the cross product of two parameters.

Our experiments show that parameter settings can have a significant effect on the performance of a steering algorithm. The experiments also reveal that specific parameters have a direct effect on specific metrics.

In some cases, adjusting thresholds allows algorithms to handle more difficult scenarios. In other cases, allocating more resources for the algorithms produces higher quality results at the expense of efficiency.

## Multi-Variate Analysis

To maximize the performance of a steering algorithm in the space of its parameters, we need to solve a multi-variate optimization problem.

We chose to use the Covariance Matrix Adaptation Evolution Strategy technique.

As expected, the multi-variate approach produces better results than the uni-variate for both algorithms over every metric.

## Steering Algorithms

### PPR:

A hybrid framework that combines reaction, prediction and planning. It is an example of a rule based system for agent based steering and has 39 independent parameters.

### Example Parameter:

*avoidance-turn-rate* defines the turning rate adjustment speed in proportion to speed. *query-radius* controls the radial distance around an agent used to predict collisions with other objects and agents.

### RVO2:

Uses optimal reciprocal collision avoidance to efficiently steer agents in large-scale crowds.

### Example Parameter:

*max-neighbors* defines the maximum number of nearby agents taken into consideration when making steering choices. *time-horizon* refers to the time for which an agent's computed velocity is safe with respect to other agents.

## Findings and Analysis

The framework is most effective on the **PPR algorithm**, increasing *Coverage* by 28% and *Quality* by 21%, and *Efficiency* is almost tripled.

The **RVO2 algorithm** shows small changes in *Coverage* and *Quality*. The most significant change is in *Efficiency*, which is doubled while only sacrificing 1% of RVO2's coverage metric.

Algorithm	v	succ	qual	eff	comb
PPR	DEF	0.620	0.826	0.022	0.511
	UNI	0.748	0.880	0.046	0.603
	MUL	0.899	0.907	0.061	0.605
RVO2	DEF	0.500	0.919	0.324	0.650
	UNI	0.530	0.925	0.621	0.726
	MUL	0.537	0.929	0.637	0.737

The table above compares the default (DEF), uni-variate (UNI) and multi-variate (MUL) values for optimizing the PPR and RVO2 algorithms.

## Results

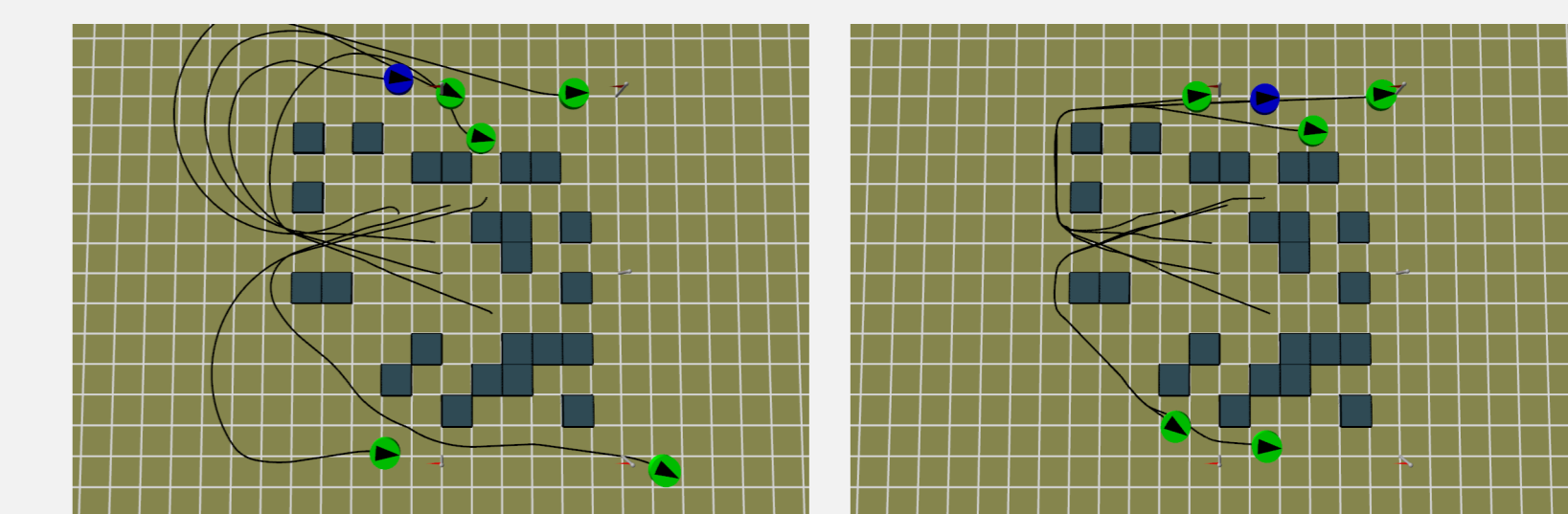
### PPR Parameter Settings

Parameter Name	$v_{min}$	$v_{max}$	$v_{def}$	$v_{opt}^{(1,0,0)}$	$v_{opt}^{(0,1,0)}$	$v_{opt}^{(0,0,1)}$	$v_{opt}^{(3,3,3)}$
ped_max_speed	1	5	2.6	2.418	1.303	3.518	1
ped_typical_speed	0.5	5	1.3	1.1916	1.331	0.634	0.73
ped_max_force	8	22	14	13.910	13.722	12.481	14.75
ped_max_speed_factor	0.6	4.7	1.7	1.22	1.723	2.188	1.434
ped_faster_speed_factor	0.55	4.2	1.31	1.520	0.982	1.595	3.11
ped_slightly_faster_speed_factor	0.4	3.4	1.15	0.978	1.044	2.065	0.641
ped_slightly_slower_speed_factor	0.15	1.2	0.77	0.951	0.446	0.801	0.434
ped_typical_speed_factor	0.5	1.5	1	0.578	0.571	1.200	0.5
ped_slower_speed_factor	0.1	1	0.5	0.109	0.1	0.538	0.1
ped_cornering_turn_rate	0.83	3.76	1.9	1.957	1.399	0.830	3.466
ped_adjustment_turn_rate	0.03	1.54	0.16	0.409	0.403	0.578	1.236
ped_faster_avoidance_turn_rate	0.15	1.87	0.55	1.347	0.414	1.471	1.524
ped_typical_avoidance_turn_rate	0.08	0.75	0.26	0.470	0.204	0.696	0.38
ped_braking_rate	0.5	1.5	0.95	0.540	0.554	0.590	1.326
ped_comfort_zone	0.7	2.8	1.5	1.794	1.302	1.731	2.526
ped_query_radius	5.0	21.0	10.0	10.325	9.175	5.000	10.49
ped_similar_direction_dot_product_threshold	0.78	0.999	0.94	0.83	0.952	0.780	0.815
ped_same_direction_dot_product_threshold	0.89	0.999	0.99	0.929	0.997	0.999	0.999
ped_oncoming_prediction_threshold	-0.99	-0.78	-0.95	-0.948	-0.893	-0.78	-0.78
ped_oncoming_reaction_threshold	-0.99	-0.78	-0.95	-0.940	-0.852	-0.78	-0.891
ped_wrong_direction_dot_product_threshold	0.23	0.78	0.55	0.440	0.434	0.628	0.23
ped_threat_distance_threshold	3	16.8	8	8.09	8.447	7.264	7.698
ped_threat_min_time_threshold	0.37	1.45	0.8	0.735	0.392	0.769	0.829
ped_threat_max_time_threshold	1.22	8.77	4	3.758	3.913	3.515	1.56
ped_predictive_anticipation_factor	2.33	8.39	5	5.08	4.921	6.816	2.433
ped_reactive_anticipation_factor	0.33	2.31	1.1	1.103	1.121	0.448	1.939
ped_crowd_influence_factor	0.11	0.61	0.3	0.474	0.378	0.116	0.28
ped_facing_static_object_threshold	0.08	0.61	0.3	0.285	0.272	0.502	0.08
ped_ordinary_steering_strength	0.001	0.2	0.05	0.115	0.197	0.027	0.2
ped_oncoming_threat_avoidance_strength	0.05	0.4	0.15	0.266	0.117	0.239	0.333
ped_cross_threat_avoidance_strength	0.73	0.999	0.9	0.820	0.73	0.955	0.999
ped_max_turning_rate	0.02	0.23	0.1	0.226	0.23	0.230	0.17
ped_feeling_crowded_threshold	1	8	3	3.189	3.406	3.439	5
ped_scoot_rate	0.17	0.78	0.4	0.707	0.274	0.170	0.78
ped_reached_target_distance_threshold	0.1	0.9	0.5	0.812	0.495	0.100	0.9
ped_dynamic_collision_padding	0.02	0.43	0.2	0.404	0.177	0.422	0.43
ped_furthest_local_target_distance	10	50	20	20	20	16.895	20
ped_next_waypoint_distance	255	260	256	256	256	256.561	256
ped_max_num_waypoints	10	50	20	20	20	20	21

### RVO2 Parameter Settings

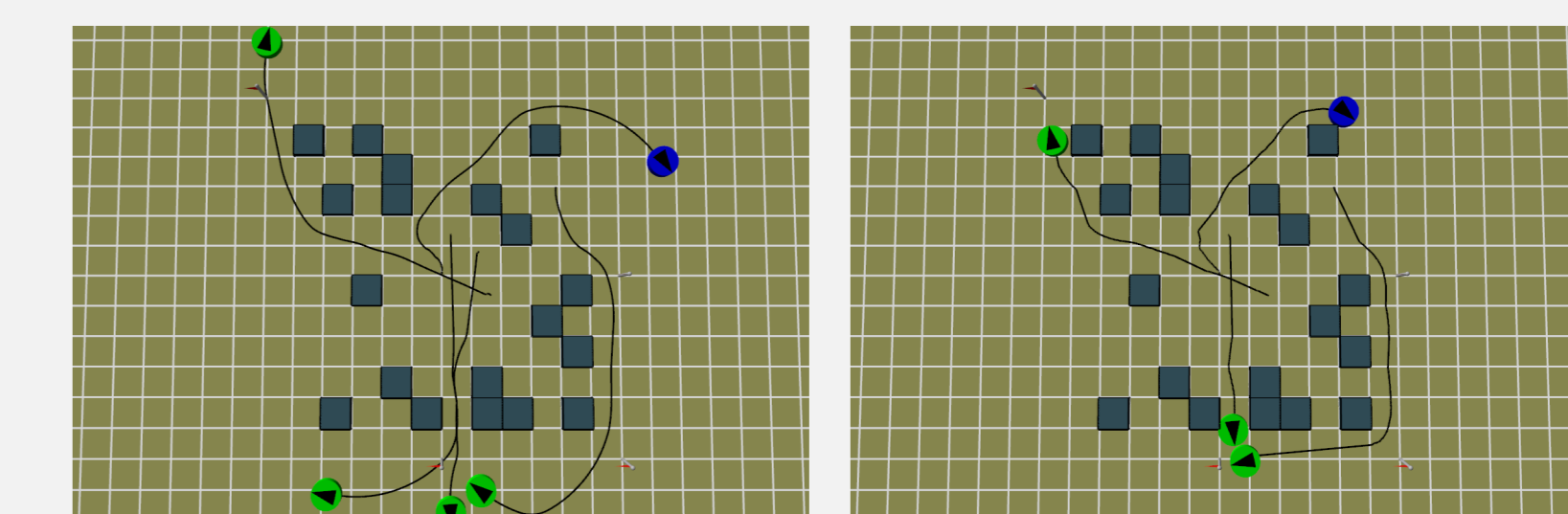
Parameter Name	$v_{min}$	$v_{max}$	$v_{def}$	$v_{opt}^{(1,0,0)}$	$v_{opt}^{(0,1,0)}$	$v_{opt}^{(0,0,1)}$	$v_{opt}^{(3,3,3)}$
rvo_neighbor_distance	2	22	15	14.774	14.366	16.068	14.976
rvo_time_horizon	2	16	10	5.941	13.958	15.705	10.044
rvo_max_speed	1	3.2	2	2.68	2.394	1.037	1.947
rvo_time_horizon_obstacles	2	16	7	5.66	3.88	2.041	7.031
rvo_max_neighbors	2	22	10	11.1855	10.382	2.077	10.061

The above tables indicate the parameter (v) settings for both algorithms for differently weighted optimizations.



Left: Default settings

Right: Settings optimized for  $qual(A)$



Left: Default settings

Right: Settings optimized for  $succ(A)$

## Conclusion

Thus far this framework is effective at evaluating an algorithm's sensitivity to parameter alterations.

Using this framework we can identify poorly chosen values for parameters and optimize parameter settings according to combinations of metrics.

For most applications, the framework can be adapted to evaluate and optimize parameter settings for different metrics, test sets and objectives.