REWARD (MIS)DESIGN FOR AUTONOMOUS DRIVING

W. Bradley Knox[*,1,2], **Alessandro Allievi**[1,2], **Holger Banzhaf**[3], **Felix Schmitt**[4], **Peter Stone**[2]

[1]*Robert Bosch LLC*
[2]*The University of Texas at Austin*
[3]*Robert Bosch GmbH*
[3]*Bosch Center for Artificial Intelligence*

ABSTRACT

This paper considers the problem of reward design for autonomous driving (AD), with insights that are also applicable to the design of cost functions and performance metrics more generally. Herein we develop 8 simple sanity checks for identifying flaws in reward functions. The sanity checks are applied to reward functions from past work on reinforcement learning (RL) for autonomous driving, revealing near-universal flaws in reward design for AD that might also exist pervasively across reward design for other tasks. Lastly, we explore promising directions that may help future researchers design reward functions for AD.

1 Introduction

Treatments of reinforcement learning often assume the reward function is given and fixed. However, in practice, the correct reward function for a sequential decision-making problem is rarely clear. Unfortunately, the process for designing a reward function (i.e., *reward design*)—despite its criticality in specifying the problem to be solved—is given scant attention in introductory texts.[2] For example, Sutton and Barto's standard text on reinforcement learning [2018, p. 53–54, 469] devotes 0.3% of its pages to reward design in the absence of a known performance metric. Anecdotally, reward design is widely acknowledged as a difficult task, especially for people without considerable experience doing so. Additionally, the problem of reward design is highly related to the more general problem of designing performance metrics for optimization—whether manual or automated optimization—and is equivalent to designing cost functions for planning and control (Section 2), making a discussion of reward design relevant beyond reinforcement learning.

As we will argue in Section 3, reward design for autonomous driving is a particularly difficult instance of reward design. Given the high envisioned impact of autonomous driving, the RL for AD community needs to consider reward design carefully if reinforcement learning will have a significant role in the development of AD. To that end, in Section 4 we develop 8 sanity checks for identifying flaws in reward functions and apply them to published reward functions used for autonomous driving. The majority of these reward functions—often all of them—fail the tests that are part of the first 3 sanity checks (Sections 4.1–4.3). In Section 5, we further explore obstacles to reward design for AD through initial attempts to design three attributes of an AD reward function, uncovering obstacles to doing so that a fellow designer should consider. Section 5 also includes a review of some government-mandated performance metrics, a discussion of reward *learning* for AD, and a proposal for designing reward for AD with a financial currency as its unit.

We do not seek to condemn past efforts—which are excellent in many regards—but rather to provide understanding of common issues in reward design for AD and some guidance on identifying them. The contributions of this paper include a deep discussion of what reward design for AD should entail, the development of 8 simple sanity checks for reward or cost functions, application of these sanity checks to identify prevalent flaws in published reward functions for AD (flaws that anecdotally appear common throughout RL, beyond AD), identifying pervasive *trial-and-error reward design* (see Section 4.5), and revealing obstacles that arise in our initial attempt to design reward for AD. In

[*]Corresponding author: `brad.knox@us.bosch.com`

[2]Unless otherwise noted, any discussion herein of reward design focuses on the specification of the environmental reward, *before any shaping rewards are added.* We also largely focus on manual reward specification, which differs from inverese reinforcement learning and other methods for learning reward functions (discussed in Section 5.4).

particular, we do not claim to solve the problem. Instead, we provide guidance for future efforts to design reward and other performance metrics for autonomous driving.

2 Background: objectives, utility functions, and reward functions

To support our discussion of reward design for AD, we first review what a reward function is. When attempting to decide between alternative decision-making or control algorithms—or, equivalently in this discussion, *policies*, which map each state to a probability distribution over actions—a common and intuitive approach is to define a *performance metric* J that scores each policy π according to $J : \pi \rightarrow \mathbb{R}$. If $J(\pi_A) > J(\pi_B)$, then π_A is better according to J.[3]

Definition of the performance metric J creates a ranking over policies and identifies the set of optimal policies, both of which are helpful for optimizing behavior in sequential decision-making tasks. However, different definitions of the performance metric typically create different rankings and sets of optimal policies, which generally change the result of a learning algorithm optimizing according to some J. *Put simply, bad design of a performance metric function creates misalignment between what the designer—or other stakeholders—considers to be good behavior and what the learning algorithm does. A perfect optimization algorithm*[4] *is only as good as the performance metric it is optimizing to.*

$J(\pi)$ is typically defined as the expectation over outcomes created by π, where these outcomes can be directly observed but $J(\pi)$ cannot. More specifically, in episodic framings of tasks, $J(\pi)$ is defined as an expectation of $G(\tau)$, which is the performance over each trajectory τ created by the policy. I.e., $J(\pi) = E_{\pi,D}[G(\tau)]$, where τ is a trajectory generated by following π from a starting state drawn according to some initial state distribution D. The mean of $G(\tau)$ from trajectories generated therefore can serve as a practical, unbiased estimator of $J(\pi)$.[5]

The policy-performance metric J and the *trajectory-performance metric* G go by many names in the various fields that address the sequential decision-making problem. Figure 1 shows some of these various terms. In many subfields, J is called an *objective* if its definition explicitly includes the goal of maximizing it (or of minimizing it, in some contexts). In utility theory, G is a *utility function* and $J(\pi)$ is the *expected utility*. In planning and control theory, both $J(\pi)$ and $G(\tau)$ can be referred to as *cost* (and optimization seeks to minimize cost rather than maximize it, as we otherwise assume here). In evolutionary algorithms, $J(\pi)$ is referred to as the *fitness* of a policy. Fitness may simply be the mean of $G(\tau)$ over n samples of τ, not the expectation $G(\tau)$. In *reinforcement learning (RL)*, each transition within the trajectory elicits a *reward* signal according to a *reward function* $R : S \times A \times S \rightarrow \mathbb{R}$. The input for R is the state, action, and next state in the transition. In the undiscounted setting, the sum of a trajectory τ's rewards is its *return*, which is reinforcement learning's word for $G(\tau)$. Therefore, $G(\tau) = \sum_{t=1}^{(T-1)} R(s_t, a_t, s_{t+1})$, where trajectory $\tau = (s_1, a_1, s_2, a_2, ..., s_{T-1}, a_{T-1}, s_T)$. Policy-improvement algorithms in RL seek to maximize the expected return, $J(\pi)$.

$$\underbrace{G(\tau)} = \sum_{t=1}^{(T-1)} \underbrace{R(s_t, a_t, s_{t+1})}$$

Field		
reinforcement learning	return	reward
motion planning	-1 × cost	-1 × cost
control theory	-1 × cost	-1 × cost
evolutionary algorithms	fitness	-
utility theory	utility	-
optimization	objective (max or min)	-
-	performance metric	-
-	score	-

Figure 1: Relationships of terminology among related fields. Note that *fitness* and *objective* are similar in meaning to *return* but not identical, as we describe in Section 2.

We will largely use the terminology of RL, but we will refer to trajectory-level performance metrics as utility functions, since "return function" is not a common concept. Also, where feasible, this paper's discussions focus on trajectory-level utility functions, Gs, rather than reward functions. We do so for clarity, having judged subjectively that the consequences of a utility function are more analytically accessible than those of a reward function.

[3]Roughly speaking, many learning algorithms for sequential-decision making tasks can be thought of as looping over two steps. (1) Policy evaluation: approximately evaluate $J(\pi)$ for one or more policies via gathered experience. (2) Policy improvement: use the evaluation of step 1 to choose a new policy π or several new policies to evaluate.

[4]We use "learning" and "optimization" interchangeably in this article.

[5]Episodic task framings are the norm in reinforcement learning for AD and are assumed throughout this paper unless otherwise noted.

3 The challenge of reward design for autonomous driving

We now consider four aspects of reward design for AD specifically that appear to make it more difficult than typical reward design.

Utility function depends on numerous attributes First, driving is a *multiattribute problem*, meaning that it encompasses numerous attributes that each contribute to the utility of driving. These attributes may include measurements of progress to the destination, time spent driving, collisions, obeying the law, fuel consumption, vehicle wear, passenger experience, and various impacts on the world outside the vehicle. These external impacts include those on people in other cars, pedestrians, bicyclists, and the government entity that builds and maintains driving infrastructure, as well as pollution and the climate more broadly. Defining a trajectory-performance metric G for AD requires (1) identifying all such attributes and specifying them quantitatively and (2) combining them into a utility function that outputs a single real-valued number.[6]

Utility function depends on a large and context-dependent set of stakeholders Second, the utility function G should conform to stakeholders' interests. In AD, these stakeholders might include users such as passengers; end consumers; partnering businesses like taxi and ride-sharing companies; automotive manufacturers; governmental regulators; providers of research funding; nearby pedestrians, passengers, bicyclists, and residents; and broader society. These stakeholders and the weight given to their interests will differ among vehicles (e.g. with different manufacturers) and will even differ for the same car in different contexts. One such context that could affect G is the driving region, with which values, preferences, and driving culture may differ substantially. Therefore *ideal* reward design for AD might require designing numerous reward functions (or Gs, more generally). Alternatively, the aforementioned stakeholder-based context could be part of the observation signal, permitting a single monolithic reward function. Such a reward function would allow a policy to be learned across numerous stakeholder-based contexts and generalize to new such contexts.

Lack of rigorous methods for evaluating a utility function Third, when choosing a utility function for algorithmic optimization in some context(s), a critical question arises: given a set of stakeholders, how can one utility function be deemed better or worse than another, and by how much? We have not yet found research on how to measure the degree of a utility function's conformity to stakeholders' interests. We also have not found formal documentation of current common practice for evaluating a utility function. Anecdotally, such evaluation often involves both subjectively judging policies that were trained from candidate utility functions and reflecting upon how the utility function might be contributing to undesirable behavior observed in these trained policies. Setting aside the dangers of allowing specific learning algorithms to inform the design of the utility function (see the discussion of trial-and-error reward design in Section 4.5), this common approach has not been distilled into a specific process, has not been carefully examined, and in practice varies substantially among different designers of reward functions or other utility functions. However, our sanity checks in Section 4 represent useful steps in this direction.

Elicits naïve reward shaping A fourth difficulty may be specific to designing a per-step feedback function like a reward function or a cost function. Driving is a task domain with *delayed* feedback, in that much of the utility of a drive is contained at its end (e.g., based on whether the goal was reached). For drives of minutes or hours, such delayed information about performance renders credit assignment to behavior within a trajectory difficult. In part because of this difficulty, reward shaping appears quite tempting and its naïve application is extremely common in research we reviewed (see Section 4.1).

4 Sanity checks for reward functions

In this section we develop a set of 8 conceptualy simple sanity checks for critiquing and improving reward functions (or cost functions, equivalently). Table 1 summarizes these sanity checks. Many of the tests apply more broadly to any (trajectory-level) utility function. We demonstrate their usage through critically reviewing reward functions used in RL for AD research.

The 19 papers we review include every paper on RL for autonomous driving we found that had been published by the beginning of this survey process at a top-tier conference or journal focusing on robotics or machine learning [10, 18, 23, 19, 6, 45, 17, 40, 31, 22, 24, 20, 16], as well as some papers published at respected venues focused more on autonomous driving [26, 7, 47, 27, 38, 3]. Of the 19 papers, we arbitrarily designated 10 as "focus papers", for which we strove to exhaustively characterize the reward function and related aspects of the task description, typically through detailed correspondence with the authors (see Section 4.5). These 10 focus papers are detailed in Appendices A and C.

We present 8 sanity checks below, 4 each in their own detailed subsections and 4 in the final subsection, Section 4.5. These tests have overlap regarding what problems they can uncover, but each sanity check entails a distinct inquiry. Their application to these 19 papers reveals multiple prevalent patterns of problematic reward design.

[6]Multiattribute utility functions still have a single performance metric for optimization, unlike multi-objective utility functions, an alternative beyond the scope of this paper.

	Sanity check failures	**Brief explanation**	**Potential intervention**
1	Unsafe reward shaping	If reward includes guidance on behavior, shaping exists and safety needs to be addressed.	Remove reward shaping or change to be safe. Be sure to separately define true reward function and shaping reward.
2	Human-reward mismatch in preference orderings	If there is human consensus that one trajectory is better than another, reward function should agree.	Change reward function to align its preferences with human consensus.
3	Undesired risk tolerance via indifference points	Assess a reward function's risk tolerance via indifference points and compare to a human-derived acceptable risk tolerance.	Change reward function to align its risk tolerance with human-derived level.
4	Learnable loophole(s)	If learned policies show a pattern of undesirable behavior, consider whether it is explicitly encouraged by reward.	Remove encouragement of the loophole(s) from the reward function.
5	Missing attribute(s)	If desired outcomes are not part of reward function, it is indifferent to them.	Add missing attribute(s).
6	Redundant attribute(s)	Two or more reward function attributes include measurements of the same outcome.	Eliminate redundancy.
7	Trial-and-error reward design	Tuning the reward function based on RL agents' performances is ill-advised.	Design your true reward function in isolation from your solution(s). Separately define any shaping reward, which can be tuned.
8	Incomplete description of problem specification	Missing descriptions of reward function, termination conditions, discount factor, or time step duration may indicate insufficient consideration of the problem specification.	In research papers, write the full problem specification and why it was chosen. The process might reveal issues.

Table 1: Sanity checks one can perform to ensure a reward function does not suffer from certain common problems. Each sanity check is described by what problematic characteristic to look for. Failure of any of the first 4 sanity checks identifies problems with the reward function; failure of the last 4 checks should be considered a warning.

4.1 Identifying unsafe reward shaping

In the standard text on artificial intelligence, Russell and Norvig assert, "As a general rule, it is better to design performance metrics according to what one actually wants to be achieved in the environment, rather than according to how one thinks the agent should behave" [35, p. 39]. In the standard text on RL, Sutton and Barto [37, p. 54] agree in almost the same phrasing, adding that imparting knowledge about effective behavior is better done via the initial policy or initial value function. More succinctly, *specify how to measure outcomes, not how to achieve them*. Exceptions to this rule should be thoughtfully justified.

Yet using rewards to encourage and hint at generally desirable behavior—often with the intention of making learning more efficient and tractable when informative reward is infrequent or inaccessible by most policies—is intuitively appealing. This practice has been formalized as reward shaping, where the learning agent receives the sum of true reward and shaping reward. The boundary between these two types of rewards is not always clear in AD, where the "true" objective is not given; nonetheless, some rewards more clearly are one of the two types.

The dangers of reward shaping are well documented [34, 30, 21]. These dangers include creating "optimal" policies that perform catastrophically. Perhaps worse, reward shaping can appear to help by increasing learning speed without the reward designer realizing that they have, roughly speaking, decreased the upper bound on performance by changing the reward function's preference ordering over policies.

There is a small canon of reward-shaping research that focuses on how to perform reward shaping with certain *safety* guarantees [30, 48, 4, 9, 14]. *Safety* here means that the reward shaping has some guarantee that it will not harm learning, which differs from its colloquial definition of avoiding harm to people or property. A common safety guarantee is *policy invariance*, which is having the same set of optimal policies with or without the shaping rewards. Any attempts to shape rewards should be informed by this literature on safe reward shaping. For all techniques with such guarantees, shaping rewards are designed *separately* from the true reward function. Also, if possible, the utility function G that arises from the true reward should be equivalent to the main performance metric used for evaluating learned policies.

We now formulate the above exposition as a sanity check. Unsafe reward shaping can be identified by first identifying reward shaping—without regard to safety—and then determining whether the designers of the reward function are either following a known safe reward shaping method or have a persuasive argument that their shaped rewards are safe.

Application to AD Acknowledging the subjectivity of classifying whether reward is shaped when shaping is not explicitly discussed, we confidently judge that, of the 19 papers we surveyed, 13 included reward shaping via one or more attributes of their reward functions [10, 23, 19, 26, 45, 7, 17, 40, 31, 22, 24, 38, 16]. Another 2 included reward attributes which could arguably be considered reward shaping [46, 3]. Examples of behavior encouraged by reward shaping in these 13 papers are staying close to the center of the lane [19], passing other vehicles [26], not changing lanes [17], increasing distances from other vehicles [45], avoiding overlap with the opposite-direction lane [10, 23], and steering straight at all times [7]. Other examples can be found in Appendix B. All of these encouraged behaviors are heuristics for *how to achieve good driving*—violating the aforementioned advice of Russell, Norvig, Sutton, and Barto—and it is often easy to construct scenarios in which they *discourage* good driving. For example, the reward shaping attribute that penalizes changing lanes [17] would discourage moving to lanes farther from other vehicles or pedestrians, including those acting unpredictably. Many of the examples above of behaviors encouraged by shaping rewards might be viewed as metrics that are highly and positively *correlated* with performance. As Amodei et al. [2] discussed, rewarding behavior that is correlated with performance can backfire, since strongly optimizing such reward can result in policies that trade increased accumulation of the shaping rewards for large reductions in other performance-related outcomes, driving down the overall performance. This concept has been memorably aphorized as Goodhart's law: "When a [proxy] measure becomes a target, it ceases to be a good measure." [36, 13].

Of the 13 papers which use reward functions we are confident are shaped, 8 were in the set of 10 focus papers [10, 23, 19, 26, 45, 7, 17, 40]. None of the 8 papers explicitly described the separation between their shaping rewards from their true rewards, and none discussed policy invariance or other guarantees regarding the safety of their reward shaping. Of these 8 papers, only Jaritz et al. and Toromanoff et al. acknowledged their usage of reward shaping, and only the former discussed its undesirable consequences. Jaritz et al. write "the bots do not achieve optimal trajectories ... [in part because] the car will always try to remain in the track center", which their reward function explicitly incentivizes. Further, in most of these 8 papers with reward shaping, the performance of learned policies was not compared in terms of their return but rather according to one or more other performance metrics, obscuring how much the undesirable behavior (e.g., frequent collisions) was a result of the RL algorithm or of the reward function it was optimizing against. 3 of these 8 papers did report return [45, 17, 7].

4.2 Comparing preference orderings

Although it is difficult for humans to score trajectories or policies in ways that are consistent with utility theory, simply judging one trajectory as better than another is sometimes easy. Accordingly, one method for critiquing a utility function is to compare the utility function's trajectory preferences to some ground-truth preferences, whether expressed by a single human or a decision system among multiple stakeholders, such the issuance of regulations. This preference comparison test of a utility function G is $\tau_A \prec \tau_B \iff G(\tau_A) < G(\tau_B)$, where $\prec$ means "is less preferred than". Finding a τ_A and τ_B for which this statement is false indicates a flaw in the utility function but does not by itself evaluate the severity of the flaw. However, severity is implied when one trajectory is strongly preferred. Note that we focus here on evaluating a utility function, which differs from learning a utility function from preferences over trajectories or subtrajectories (see Section 5.4), after which this sanity check can be applied.

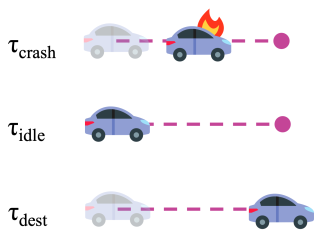

Figure 2: Illustrations of the abstract trajectories used in Sections 4.2 and 4.3.

Application to AD We apply this comparison by choosing two trajectories such that, under non-exceptional circumstances, one trajectory is strongly preferred. We specifically let τ_{crash} be a drive that is successful until crashing halfway to its destination and let τ_{idle} be the safe trajectory of a vehicle choosing to stay motionless where it was last parked. Figure 2 illustrates τ_{crash} and τ_{idle}. Of the 10 focus papers, 9 permit estimating $G(\tau_{crash})$ and $G(\tau_{idle})$. Huegle et al. [17] does not (see Appendix C). For the calculation of these utilities here and later in this paper, we assume reward is not temporally discounted in the *problem specification*—which is generally considered correct for episodic tasks [37, p. 68] like these—despite nearly all papers' adherence to the current best practice of discounting future reward to aid deep reinforcement learning *solutions* (as discussed by Pohlen et al. [33]).

We presume that any appropriate set of stakeholders would prefer a vehicle to be left idle rather than to proceed to a certain collision: $\tau_{crash} \prec \tau_{idle}$. Yet of these 9 evaluated reward functions, only 2 have reward functions with the correct preference and 7 have reward functions that would prefer τ_{crash} and its collision. These 7 papers are identified on the left side of Figure 3, under $\tau_{idle} \prec \tau_{crash}$. We do not calculate utilities for the reward functions of the 9 papers that were *not* in the set of focus papers, but our examination of these other reward functions suggests a similar proportion of them would likewise have an incorrect ordering.

Calculation of returns for these trajectories allows a much-needed sanity check for researchers conducting RL-for-AD projects, avoiding reward functions that are egregiously dangerous in this particular manner.

4.3 Comparing indifference points

A more complex form of preference comparison reveals problems in the 2 papers that passed the test in Section 4.2. For three trajectories $\tau_A \prec \tau_B \prec \tau_C$, the continuity axiom of utility theory states that there is some probability p such that a rational agent is indifferent between (1) τ_B and (2) sampling from a Bernoulli distribution over τ_A and τ_C, where τ_C occurs with probability p [44]:

$$G(\tau_B) = pG(\tau_C) + (1-p)G(\tau_A). \tag{1}$$

This indifference point p can often be compared to a ground-truth indifference point derived from human stakeholders that reveals their risk tolerance.

Application to AD To apply this test of preferences over probabilistic outcomes to AD, we add τ_{succ} to τ_{crash} and τ_{idle} from Section 4.2, where τ_{succ} is a trajectory that successfully reaches the destination. $\tau_{crash} \prec \tau_{idle} \prec \tau_{succ}$. Therefore, choosing p amounts to setting permissible risk of crashing amongst otherwise successful trips. In other words, a policy that has higher risk than this threshold p is less preferred than one that refuses to drive at all—as a human driver might wisely do when faced with a significant probability of collision, such as during severe weather. Figure 3 displays the calculated p converted to a more interpretable metric: km per collision[7] at which driving is equally preferable to not deploying a vehicle. For comparison, we also plot estimates of collisions per km for various categories of humans. As the figure shows, of those 9 focus papers that permit this form of analysis, 0 require driving more safely than a legally drunk US 16–17 year old teenager. The most risk-averse reward function by this metric [6] would approve driving by a policy that crashes 4666 times as often as our estimate of drunk 16–17 year old US drivers.

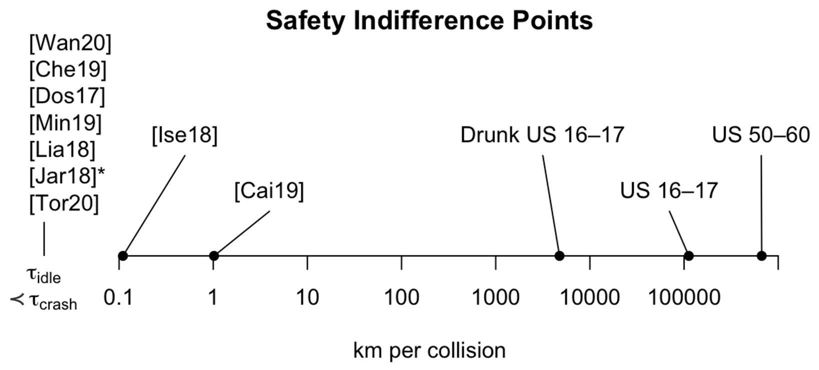

Figure 3: Estimates of kilometers per collision at which various published reward functions are indifferent regarding whether they prefer safely declining to move or driving with a certain km per collision rate. Higher is better. Papers are referenced by the first 3 letters of their first author's name and the last two digits of their publication year. *Che19* refers to the paper by Jianyu Chen. The 3 points on the right designate estimates of actual km per collision for the age group among US drivers with the most km per collisions (50–60 year olds) and the least (16–17 year olds) [39], as well as a rough estimate of km per collision for a drunk 16–17 year old (from applying a 37x risk for blood alcohol concentration ≥ 0.08, as estimated by Peck et al. [32]). *The task domain of Jaritz et al. [19] was presented as a racing video game and therefore should not be judged by real-world safety standards.

An argument against this test—and more broadly against requiring the utility function to enforce driving at or above human-level safety—is that penalizing collisions too much could cause an RL algorithm to get stuck in a conservative local optimum of not moving, having assessed that its current policy is not safe enough for driving. This issue however can potentially be overcome by creating sufficiently good starting policies or by performing reward shaping explicitly and rigorously. Further, there is a significant issue with the argument above, the argument that the reward function should encourage the RL algorithm to gather driving experience by being extremely lenient with collisions. Whether a

[7]Kilometers per crash at indifference point $= ([p/(1-p)] + 0.5) \times$ distance of a successful path, where $p/(1-p)$ is the amount of τ_{succ} trajectories per half-length τ_{crash} trajectory at the indifference point.

specific weighting of a collision penalty will effectively discourage collisions without making the vehicle avoid driving at all is dependent on the performance of the current policy. When the policy is being improved (our primary use case for an AD reward function), the value of effective weightings would increase as the RL algorithm improves its driving. The true reward function is part of a task specification and should not change as a function of the policy. However, such dynamic weighting could be achieved by defining the true reward function to have a risk tolerance that is desirable in real-world situations then adjusting the weight given to collisions via a form of dynamic reward shaping: the collisions weight would start small and be gradually increased to scaffold learning while the policy improves, eventually reaching its true weight value. In this strategy, reward shaping is temporary and therefore policy invariance is achieved once shaping ends.

4.4 Identifying learnable loopholes

Once a designed reward function is used for learning a policy, observable patterns of undesirable behavior might emerge. When such behavior increases utility, it is often referred to as *reward hacking* or *specification gaming* (terms that implicitly and unfairly blame the agent for correctly optimizing a flawed utility function). Colloquially, such technically legal violations of the spirit of the law are called *loopholes*. Existence of a loophole implies reward shaping, since a loophole reveals a mismatch between the behavior that achieves the highest utility and the desired behavior. The RL literature provides numerous catastrophic examples of such loopholes [34, 1, 30], which often involve learned trajectories that actually loop in physical space to repeatedly accrue some reward but prevent long-term progress (see Figure 4).[8] However, these loopholes can also be subtle, not dominating performance but nonetheless limiting it, and in such cases are more difficult to find through observations of learned behavior. In general, both subtle and blatant loopholes might be found by observing undesirable behavior and reflecting that the reward function encourages that behavior. A more rigorous improvement on such reflection is to estimate the utility (or return, equivalently) for an observed trajectory that contains the undesirable behavior and also estimate the utility for that trajectory modified minimally to not contain the undesirable behavior; if the less desirable trajectory receives more utility, then a loophole likely exists.

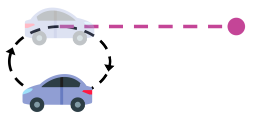
Figure 4: Illustration of a common learnable loophole, in which the agent moves in circles to repeatedly collect reward for progress, never reaching the goal.

Application to AD We do not observe blatantly catastrophic loopholes in the RL-for-AD literature, perhaps because most reward functions for AD thus far appear to have been designed via trial and error (see Section 4.5); in such trial-and-error design, any such catastrophic loopholes would likely get caught, and the reward function would then be tuned to avoid them. However, the learned-policy limitation that Jaritz et al. [19] discussed (see Section 4.1) is an example of a learned loophole in RL for AD.

4.5 Sanity checks for which failure is a warning

In addition to the sanity checks described above, other methods can be used to raise red flags, indicating potential issues in the reward function that warrant further investigation.

Missing attributes Utility functions sometimes lack attributes needed to holistically judge the performance of a trajectory. Specifically, as discussed in Amodei et al. [2], omission of an attribute "implicitly expresses indifference" regarding that attribute's measurements of outcomes. A simple example in autonomous driving is to ignore passenger experience while including an attribute that increases utility for making progress towards a destination; the learned behavior may exhibit high acceleration and jerk, which tend to harm passenger experience but do not directly affect the utility function. Section 3 contains a list of potential attributes for autonomous driving that may help in this evaluation. Identifying missing attributes and adding them is one solution. When inclusion of all missing attributes is intractable, Amodei et al. propose penalizing a policy's impact on the environment. However, this approach, called *impact regularization*, shares potential issues with reward shaping.

Redundant attributes Similarly, a utility function may have redundant attributes that encourage or discourage the same outcomes. Such overlap can overly encourage only part of what constitutes desirable outcomes. The overlap can also complicate a reward designer's understanding of the attributes' combined impact on the utility function. For instance, an autonomous driving utility function might include two attributes, one that penalizes collisions and one that penalizes repair costs to the ego vehicle. Both attributes penalize damage to the ego vehicle from a collision. Yet each attribute also includes a measurement of outcomes that the other does not: a collision penalty can discourage harm to people and external objects, and a repair-costs penalty discourages driving styles that increase the rate of wear. When redundant attributes exist in the utility function, solutions include separating redundant aspects of multiple attributes into a new attribute or removing redundant aspects from all but one attribute. Executing these solutions is not always straightforward, however. In our example above, perhaps the *collision* penalty could be separated into components

[8] More examples can be found at this webpage: "Specification gaming examples in AI"

that measure harm to humans and animals, harm to external objects, and increased repair costs for the ego vehicle's maintainer. Then the repair-costs component of the collision penalty could be removed, since it is already contained within the penalty for overall repair costs.

Trial-and-error reward design Alternatively, if a paper presents its AD reward function without justifying its design, then the design may have occurred through a process of trial and error. This *trial-and-error reward design* process involves designing a reward function, testing an RL agent with it, *using observations of the agent's learning to tune the reward function*, and then repeating this testing and tuning process until satisfied. This process is also described by Sutton and Barto [37, p. 469]. Since the reward function itself is being revised, typically one or more other performance metrics are employed to evaluate learned policies. Those performance metrics could be based on subjective judgment or be explicitly defined.

One issue with this trial-and-error reward design is that the specification of the reinforcement learning problem should not in principle be affected by the preferred solution to the problem, unless perhaps the reward designer's intention is to do reward shaping. More practically, another issue is that this manual reward-optimization process might overfit. In other words, trial-and-error reward design might improve the reward function's efficacy—whether measured by the reward designer's subjective evaluation, another performance metric, or otherwise—in the specific context in which it is being tested, but then the resultant reward function is used in other untested contexts, where its effects are unknown. Factors affecting trial-and-error reward design include both the RL algorithm and the duration of training before assessing the learned policy. In particular, after the designer chooses a final reward function, we suspect they will typically allow the agent to train for a longer duration than it did with the multiple reward functions evaluated during trial-and-error design. Further, any comparison of multiple RL algorithms will tend to unfairly favor the algorithm used during trial-and-error design, since the reward function was specifically tuned to improve the performance of that RL algorithm.

Trial-and-error reward design for AD is widespread: of the 8 papers whose authors shared their reward design process with us in correspondence, all 8 reported following some version of defining a linear reward function and then manually tuning the weights or revising the attributes via trial and error until the RL algorithm learns a satisfying policy [10, 18, 6, 17, 26, 7, 45, 40].

Incomplete problem specification in research presentations If a paper on RL for AD does not exactly specify the reward function, discount factor, termination conditions, and time step duration used, then this omission is also a red flag that may indicate to the reader that the reward function may be considered merely an implementation detail. Only one of the 10 focus papers thoroughly described the above portions of their problem specification Liang et al. [23]. We learned the other 9 papers' specifications through correspondence with their authors. To be clear, such omission of the details of one's problem specification does not *necessarily* indicate problems in reward design but rather is, we conjecture, positively correlated with reward design issues.

5 Exploring the design of an effective reward function

In contrast to the previous section—which focuses on how *not* to design a reward function—this section instead considers *how* to design one. This section contains exploration that we intend to be preliminary to a full recommendation of a specific reward function for AD or of a process for creating one.

5.1 AD performance metrics beyond RL

One potential source of reward-design inspiration is the performance metrics that have been created by communities beyond RL researchers.

Prominent among these are the metrics used by regulatory agencies and companies developing AD technology. Many such metrics express the *distance per undesirable event*, which incorporates two critical utility-function attributes: making progress and avoiding failure states like collisions. Specifically, the California Department of Motor Vehicles (DMV) requires reporting of *miles per disengagement*, where a disengagement is defined as the deactivation of the vehicle's autonomous mode and/or a safety driver taking control from the autonomous system. Criticisms of this metric include that it ignores important context, such as the complexity of driving scenarios, the software release(s) being tested, and the severity of the outcomes averted by the safety drivers' interventions. The California DMV also requires a report to be filed for each collision, through which a *miles per collision* measure is sometimes calculated [11]. This metric is vulnerable to some of the same criticisms. Also, note that disengagements often prevent collisions, so safety drivers' disengagement preferences can decrease miles per disengagement while increasing miles per collision, or vice versa; therefore, the two complementary metrics can be combined for a somewhat clearer understanding of safety. Another metric is miles per fatality, which addresses the ambiguity of a collision's severity. In contrast to the

8 attributes we listed in Section 3 as important for an AD utility function, each of the above metrics only covers 2 attributes—distance traveled and a count of an undesirable event.

Performance metrics for AD have also been designed by other robotics and artificial intelligence communities. In particular, per-time-step cost functions developed by the planning and control communities could be converted to reward functions by multiplying their outputs by -1. Additionally, insight might be gained by examining the reward functions learned by techniques reviewed in Section 5.4. However, a review of such cost functions and learned reward functions is beyond the scope of this paper.

5.2 An exercise in designing utility function attributes

To get a sense of the challenges involved in expressing these attributes, let us consider in detail three attributes that we listed previously in Section 3: progress to the destination, obeying the law, and passenger experience. We assume that each attribute is a component in a linear reward function, following the common practice. For simplicity, we focus on the scenario of driving one or more passengers, without consideration of other cargo.

We see one obvious candidate for **progress to the destination**.[9] This candidate attribute is the proportion of progress towards the destination, expressed as a value ≤ 1 and calculated at time step t as

$$\frac{(\textit{route length from position at time } t) - (\textit{route length from position at time } t-1)}{\textit{route length from start}},$$

which adds to 1 over an entire successful trajectory. (Route length can be calculated in various ways, such as the distance of the shortest legal path that can be planned.)

This approach initially appears reasonable but nonetheless has at least one significant issue: each equal-distance increment of progress has the same impact on the attribute. To illustrate, consider two policies. One policy always stops exactly halfway along the route. The other policy reaches the destination on half of its trips and does not start the trip otherwise. With respect to the progress-to-the-destination attribute proposed above, both polices have the same expected performance. Yet the performance of these two policies do not seem equivalent. For our own commutes, we authors certainly would prefer a ride-sharing service that cancels half the time than a service that always drops us off halfway to our destination.

This dilemma leaves open the question of how to calculate progress to the destination as an attribute. Perhaps it should be based upon the utility derived by the passenger(s) for being transported to some location. We suspect this utility would *generally* be lowest when the drop-off point is far both from the pickup location and from the destination. A highly accurate assessment of this utility would seemingly require information about what the passenger(s) would do at any drop-off location, including their destination. For instance, if a drug store is the destination, a passenger's utility at various drop-off locations will differ greatly based on whether they plan to purchase urgently needed medication or to buy a snack. Such information is unlikely to be available to autonomous vehicles, but nonetheless a coarse estimate of a passenger's utility for a specific drop-off location could be made with whatever information that is available. Also, perhaps passengers could opt in to share that reaching their destination is highly consequential for them, allowing the driving policy to reflect this important information.

Obeying the law is perhaps even trickier to define precisely as an attribute. This attribute could be expressed as the penalties incurred for breaking laws. Unfortunately, such penalties come in different units, with no obvious conversion to the other units—such as fines, time lost interacting with law enforcement and court systems, and maybe even time spent incarcerated—making difficult the combination of these penalties into a single numeric attribute. An additional issue is that modeling the penalties ignores the cost to others, including to law enforcement systems. If such *external costs* are included, the attribute designer might also want to be careful that some of those external costs are not redundantly expressed in another attribute, such as a collisions attribute. A further challenge is obtaining a region-by-region encoding of driving laws and their penalties.

Passengers are critical stakeholders in a driving trajectory, and **passenger experience** appears important because their experiences are not always captured by other metrics like collisions. For example, many people have experienced fear as passengers when their driver brakes later than they prefer, creating a pre-braking moment of uncertainty regarding whether the driver is aware of the need to slow the vehicle, which they are. Though the situation was actually safe in hindsight, the late application of brakes created an unpleasant experience for the passenger. To define passenger experience as an attribute, one candidate solution is to calculate a passenger-experience score through surveys of passenger satisfaction, perhaps by averaging all passengers' numeric ratings at the end of the ride. However, surveys

[9]For reasons hinted at in this subsection, we do not focus on the simpler binary attribute of reaching the destination, which ignores that drop-off points other than the destination can have many different effects on passengers' utilities.

rely on biased self report and might be too disruptive to deploy for every trajectory. In experimental design, when surveys are asking for respondents' predictions of their own behavior, it is advisable to consider whether instead the experiment could rely on observations of behavior; this guideline prompts the question of whether future passenger behavior—such as choosing to use the same AD service again—might be more useful as part of the attribute than direct surveys. Lastly, passenger experience is an underdefined concept, despite our confidence that it is important to include in the overall AD utility function, leaving open the question of what exactly to measure.

Combining these attributes is also difficult, since the units of each attribute might not be straightforwardly convertible to those of other attributes. The reward function is commonly a linear combination of attributes; however, this linearity assumption could be incorrect, for example if the utility function needs to be the result of a conjunction over attributes, such as a binary utility function for which success is defined as reaching the destination without collision. Also, weight assignment for such linearly expressed reward functions is often done by trial and error (see Section 4.5), possibly in part because the researchers lack a principled way to weigh attributes with different units.

5.3 A financial utility function for AD

One potential solution to the challenge of how to combine attributes is to express all attributes in the same unit, so they can be added without weights. Specifically, a financial utility function might output the change in the expectation of net profit across all stakeholders caused by τ. Utilities expressed in currency units are common in RL when profit or cost reduction are the explicit goals of the task, such as stock trading [28] and tax collection [25], but we are unaware of its usage as an optimization objective for AD.

To create such a financial utility function for AD, non-financial outcomes would need to be mapped to financial values, perhaps via an assessment of people's willingness to pay for those outcomes. We have been surprised to find that some non-financial outcomes of driving have a more straightforward financial expression than we initially expected, providing optimism for this strategy of reward design. For example, much effort has gone towards establishing a *value of statistical life*, which allows calculation of a monetary value for a reduction in a small risk of fatalities. The value of statistical life is used by numerous governmental agencies to make decisions that involve both financial costs and risks of fatality. The US Department of Transportation's value of statistical life was \$11.6 million US Dollars in 2020 [42, 43].

5.4 Methods for learning a reward function

Instead of manually designing a reward function, one can instead learn one from various types of data. Methods of learning reward functions include inverse reinforcement learning from demonstrations [29, 51], learning reward functions from pairwise preferences over trajectories [49, 8, 5], and inverse reward design from trial-and-error reward design in multiple instances of a task domain [15]. Traditionally, these approaches assume that reward is a linear function over pre-specified attributes and that only the weights are being learned, so the challenges of choosing attributes remain. Approaches that instead model reward via expressive representations like deep neural networks [50, 12] could avoid this challenge. Another issue is that there is no apparent way to evaluate the result of reward learning without already knowing the utility function G, which is not known for many tasks like autonomous driving; blindly trusting the results of learning is particularly unacceptable in safety-critical applications like AD. For these methods, the sanity checks we present in Section 4 could provide partial evaluation of such learned reward functions.

6 Conclusion

In the US alone, 1.9 trillion miles were driven in 2019 [41]. Once autonomous vehicles are prevalent, they will generate massive amounts of experiential data. Techniques like reinforcement learning can leverage this data to further optimize autonomous driving, whereas many competing methods cannot do so (e.g., behavioral cloning) or are labor-intensive (e.g., manual design of decision-making). Despite the suitability of RL to AD, it might not play a large role in its development without well-designed objectives to optimize towards. This paper sheds light on the problematic state of reward design for AD, and it provides arguments and a set of sanity checks that could jump start improvements in reward design for AD. We hope this paper provokes conversation about reward specification—for autonomous driving and more broadly—and adds momentum towards a much-needed sustained investigation of the topic.

From this paper, we see at least four impactful directions for further work. The first supports the practicality of learning from a utility function that passes our first three sanity checks. Specifically, in the work we reviewed, the challenges of exploration and reward sparsity were partially addressed by reward shaping and low penalties for collisions. One could empirically demonstrate that, while learning from a reward function that passes the corresponding sanity checks, these challenges can instead be addressed by other methods. Second, one could craft a reward function or other utility function for AD that passes our sanity checks, includes attributes that incorporate all relevant outcomes, addresses the issues discussed in Section 5.2, and passes any other tests deemed critical for an AD utility function. Third, one could develop more comprehensive methods for evaluating utility functions with respect to human stakeholders' interests. Finally, the community would benefit from research that constructs best practices for the manual design of reward functions for arbitrary tasks.

Acknowledgements

We thank Ishan Durugkar and Garrett Warnell for their valuable feedback. We also thank the authors of the focus papers who generously answered our questions about their papers' details. This work is a collaboration between Bosch and the the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (CPS-1739964, IIS-1724157, NRI-1925082), ONR (N00014-18-2243), FLI (RFP2-000), ARO (W911NF-19-2-0333), DARPA, Lockheed Martin, GM, and Bosch. Peter Stone serves as the Executive Director of Sony AI America and receives financial compensation for this work. The terms of this arrangement have been reviewed and approved by the University of Texas at Austin in accordance with its policy on objectivity in research.

References

[1] Dario Amodei and Jack Clark. Faulty reward functions in the wild. `https://blog.openai.com/faulty-reward-functions`, 2016. Accessed: 2020-04-16.

[2] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety. *arXiv preprint arXiv:1606.06565*, 2016.

[3] Szilard Aradi, Tamas Becsi, and Peter Gaspar. Policy gradient based reinforcement learning approach for autonomous highway driving. In *IEEE Conference on Control Technology and Applications (CCTA)*, pages 670–675. IEEE, 2018.

[4] John Asmuth, Michael L Littman, and Robert Zinkov. Potential-based shaping in model-based reinforcement learning. In *Twenty-third AAAI Conference on Artificial Intelligence*, pages 604–609, 2008.

[5] Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International Conference on Machine Learning (ICML)*, pages 783–792. PMLR, 2019.

[6] Panpan Cai, Yuanfu Luo, Aseem Saxena, David Hsu, and Wee Sun Lee. Lets-drive: Driving in a crowd by learning from tree search. In *Robotics: Science & Systems (RSS)*, 2019.

[7] Jianyu Chen, Bodi Yuan, and Masayoshi Tomizuka. Model-free deep reinforcement learning for urban autonomous driving. In *IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2765–2771. IEEE, 2019.

[8] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4299–4307, 2017.

[9] Sam Michael Devlin and Daniel Kudenko. Dynamic potential-based reward shaping. In *11th International Conference on Autonomous Agents and Multiagent Systems*, pages 433–440. IFAAMAS, 2012.

[10] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.

[11] Francesca M Favarò, Nazanin Nader, Sky O Eurich, Michelle Tripp, and Naresh Varadaraju. Examining accident reports involving autonomous vehicles in california. *PLoS one*, 12(9):e0184952, 2017.

[12] Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2018.

[13] Charles AE Goodhart. Problems of monetary management: the UK experience. In *Monetary theory and practice*, pages 91–121. Springer, 1984.

[14] Marek Grzes. Reward shaping in episodic reinforcement learning. In *AAAI Conference on Artificial Intelligence*. ACM, 2017.

[15] Dylan Hadfield-Menell, Smitha Milli, Pieter Abbeel, Stuart J Russell, and Anca Dragan. Inverse reward design. In *Advances in Neural Information Processing Systems (NIPS)*, pages 6765–6774, 2017.

[16] Mikael Henaff, Yann LeCun, and Alfredo Canziani. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. In *7th International Conference on Learning Representation (ICLR)*, 2019.

[17] Maria Huegle, Gabriel Kalweit, Branka Mirchevska, Moritz Werling, and Joschka Boedecker. Dynamic input for deep reinforcement learning in autonomous driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[18] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

[19] Maximilian Jaritz, Raoul De Charette, Marin Toromanoff, Etienne Perot, and Fawzi Nashashibi. End-to-end race driving with deep reinforcement learning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2070–2075. IEEE, 2018.

[20] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[21] W. Bradley Knox and Peter Stone. Reinforcement learning with human and MDP reward. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. IFAAMAS, June 2012.

[22] Changjian Li and Krzysztof Czarnecki. Urban driving with multi-objective deep reinforcement learning. In *18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 359–367. IFAAMAS, 2019.

[23] Xiaodan Liang, Tairui Wang, Luona Yang, and Eric Xing. CIRL: Controllable imitative reinforcement learning for vision-based self-driving. In *European Conference on Computer Vision (ECCV)*, pages 584–599, 2018.

[24] Guan-Horng Liu, Avinash Siravuru, Sai Prabhakar, Manuela Veloso, and George Kantor. Learning end-to-end multimodal sensor policies for autonomous navigation. In *Conference on Robot Learning (CoRL)*, pages 249–261, 2017.

[25] Gerard Miller, Melissa Weatherwax, Timothy Gardinier, Naoki Abe, Prem Melville, Cezar Pendus, David Jensen, Chandan K Reddy, Vince Thomas, James Bennett, et al. Tax collections optimization for new york state. *Interfaces*, 42(1):74–84, 2012.

[26] Kyushik Min, Hayoung Kim, and Kunsoo Huh. Deep distributional reinforcement learning based high-level driving policy determination. *IEEE Transactions on Intelligent Vehicles*, 4(3):416–424, 2019.

[27] Branka Mirchevska, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker. High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning. In *21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2156–2162. IEEE, 2018.

[28] Yuriy Nevmyvaka, Yi Feng, and Michael Kearns. Reinforcement learning for optimized trade execution. In *23rd International Conference on Machine Learning (ICML)*, pages 673–680, 2006.

[29] A.Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *Seventeenth International Conference on Machine Learning (ICML)*, 2000.

[30] A.Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. *Sixteenth International Conference on Machine Learning (ICML)*, 1999.

[31] Chris Paxton, Vasumathi Raman, Gregory D Hager, and Marin Kobilarov. Combining neural networks and tree search for task and motion planning in challenging environments. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6059–6066. IEEE, 2017.

[32] RC Peck, MA Gebers, RB Voas, and E Romano. Improved methods for estimating relative crash risk in a case—control study of blood alcohol levels. In *Joint Meeting of The International Council on Alcohol, Drugs & Traffic Safety (ICADTS) and The International Association of Forensic Toxicologists (TIAFT)*, 2007.

[33] Tobias Pohlen, Bilal Piot, Todd Hester, Mohammad Gheshlaghi Azar, Dan Horgan, David Budden, Gabriel Barth-Maron, Hado Van Hasselt, John Quan, Mel Večerík, et al. Observe and look further: Achieving consistent performance on atari. *arXiv:1805.11593*, 2018.

[34] J. Randløv and P. Alstrøm. Learning to drive a bicycle using reinforcement learning and shaping. In *Fifteenth International Conference on Machine Learning (ICML)*, pages 463–471. Citeseer, 1998.

[35] Stuart Russell and Peter Norvig. *Artificial intelligence: a modern approach*. 2020.

[36] Marilyn Strathern. 'Improving ratings': audit in the British university system. *European review*, 5(3):305–321, 1997.

[37] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[38] Yichuan Tang. Towards learning multi-agent negotiations via self-play. In *IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, 2019.

[39] B.C. Tefft. Rates of motor vehicle crashes, injuries and deaths in relation to driver age, United States, 2014-2015. AAA Foundation for Traffic Safety `https://aaafoundation.org/rates-motor-vehicle-crashes-injuries-deaths-relation-driver-age-united-states-2014-2015/`, 2017. Accessed: 2020-10-07.

[40] Marin Toromanoff, Emilie Wirbel, and Fabien Moutarde. End-to-end model-free reinforcement learning for urban driving using implicit affordances. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7153–7162, 2020.

[41] U.S. Dept. of Transportation. Traffic volume trends. `https://www.fhwa.dot.gov/policyinformation/travel_monitoring/20jultvt/20jultvt.pdf`, July 2020.

[42] U.S. Dept. of Transportation. Departmental guidance on valuation of a statistical life in economic analysis. `https://www.transportation.gov/office-policy/transportation-policy/revised-departmental-guidance-on-valuation-of-a-statistical-life-in-economic-analysis`, April 2021.

[43] U.S. Dept. of Transportation. Departmental guidance: Treatment of the value of preventing fatalities and injuries in preparing economic analyses. `https://www.transportation.gov/sites/dot.gov/files/2021-03/DOTVSLGuidance-2021Update.pdf`, March 2021.

[44] John Von Neumann and Oskar Morgenstern. *Theory of games and economic behavior*. Princeton university press, 1944.

[45] Jingke Wang, Yue Wang, Dongkun Zhang, Yezhou Yang, and Rong Xiong. Learning hierarchical behavior and motion planning for autonomous driving. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[46] Pin Wang, Hanhan Li, and Ching-Yao Chan. Continuous control for automated lane change behavior based on deep deterministic policy gradient algorithm. In *IEEE Intelligent Vehicles Symposium (IV)*, pages 1454–1460. IEEE, 2019.

[47] Pin Wang, Hanhan Li, and Ching-Yao Chan. Quadratic q-network for learning continuous control for autonomous vehicles. In *Machine Learning for Autonomous Driving Workshop at the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[48] Eric Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19:205–208, 2003.

[49] Christian Wirth, Riad Akrour, Gerhard Neumann, Johannes Fürnkranz, et al. A survey of preference-based reinforcement learning methods. *Journal of Machine Learning Research*, 18(136):1–46, 2017.

[50] Markus Wulfmeier, Peter Ondruska, and Ingmar Posner. Maximum entropy deep inverse reinforcement learning. *arXiv preprint arXiv:1507.04888*, 2015.

[51] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Twenty-third AAAI Conference on Artificial Intelligence*, volume 8, pages 1433–1438, 2008.

A Reward functions in the 10 focus papers

In this Appendix section, we describe reward functions and other problem specification details from the 10 papers that we evaluate in Section 4. We also closely examined 9 other papers, but because the patterns in the 10 focus papers below were so consistent, we did not pursue clarifying these 9 additional papers' problem specifications sufficiently for us to be able to characterize them with confidence at the same level of detail as the focus papers described below.

In this section, papers are listed alphabetically by first author's last name. A † marks information obtained in part or fully through correspondence with the paper's author. Additionally, we include time limit information but do not typically report whether the RL agent is updated with a terminal transition when time expires and an episode is stopped, since we rarely have such information. However, we suspect that most of these time limits are meant to make training feasible and are not actually part of the problem specification, and we further suspect that agents are correctly not updated with a terminal transition upon time exhaustion.

A.1 LeTS-Drive: Driving in a Crowd by Learning from Tree Search [6]

Reward function The reward function is the unweighted sum of 3 attributes, with the authors' stated purpose of each in brackets:

- [efficiency] -0.1 at each non-terminal time step (and 0 for a terminal transition);
- [smoothness] -0.1 if the action includes non-zero acceleration (0 otherwise); and
- [safety] $-1000 \times (v^2 + 0.5)$ upon collision with a pedestrian or a static obstacle, where v is the driving speed in m/s† (0 otherwise).

Time step duration Time steps are 100 ms.

Discount factor The discount factor $\gamma = 1$.†

Episodic/continuing, time limit, and termination criteria The task is episodic. Regarding the time limit, episodes are computationally stopped after 120 seconds (in simulation time) / 1200 time steps, which is used for calculating the success rate. However, if a trajectory is stopped at 120 seconds, none of the trajectory is used to update the value function, making it somewhat optimistic.† Termination criteria are collisions and the agent reaching the goal.†

A.2 Model-free Deep Reinforcement Learning for Urban Autonomous Driving [7]

Reward function The reward function is the unweighted sum of 5 attributes:

- $min(\text{speed}, 10 - \text{speed})$, the ego vehicle's speed in m/s, with a penalty for going too fast;
- $-0.5 * (\text{steering angle})^2$, a penalty for the magnitude of the steering angle in radians;
- -10 upon a collision (0 otherwise);
- -1 upon leaving the lane, incurred if the distance between the ego vehicle's center and the closest point on the provided route's polyline is greater than 2m (0 otherwise);†and
- -0.1 every step to encourage quickly reaching the goal.

Reward is calculated for every 100 ms CARLA time step, but it is received by the agent only at its decision points, every 400 ms. That received reward is the sum of the four 100 ms rewards.

Time step duration Time steps are effectively 400 ms because "frame skip" is used to keep the chosen action unchanged for 4 frames, each of which is 100 ms (as in other research within the CARLA simulator).

Discount factor The discount factor $\gamma = 0.99$ and was applied at every decision point (i.e., every 400 ms).†

Episodic/continuing, time limit, and termination criteria The task is episodic. The time limit is 500 time steps (i.e., 50 s).† Termination criteria are getting to the goal state, collisions, leaving the lane, and running out of time.†

A.3 CARLA: An open urban driving simulator [10]

Reward function The reward function is the weighted sum of 5 attributes:

$r = (1)\Delta d + (0.05)\Delta v + (-0.00002)\Delta c + (-2)\Delta s + (-2)\Delta o.$

These attributes are

- Δd, the change in distance traveled in meters along the shortest path from start to goal, regularly calculated using the ego's current position;
- Δv, the change in speed in km/h;
- Δc, the change in collision damage (expressed in range $[0, 1]$);
- Δs, the change in the proportion of the ego vehicle that currently overlaps with the sidewalk; and
- Δo, the change in the proportion of the ego vehicle that currently overlaps with the other lane.

Time step duration Time step information is not described in the paper, but 0.1 s is the time step duration reported by all other papers that use the CARLA simulator and report their time step duration.

Discount factor The first author suspects $\gamma = 0.99$ (which is typical for A3C).†

Episodic/continuing, time limit, and termination criteria The task is episodic. The time limit for an episode is the amount of time it would take a car to follow the shortest path from the start state to a goal state, driving at 10 km/h. Termination occurs upon collision or reaching the goal.

A.4 Dynamic Input for Deep Reinforcement Learning in Autonomous Driving [17]

Reward function The reward function is the unweighted sum of 3 attributes:

- 1 given every step;
- $-\frac{|v - v_{desired}|}{v_{desired}}$; and
- -0.01 if the current action is a lane change (0 otherwise).

Time step duration Time steps for the agent are 2 seconds.

Discount factor The discount factor $\gamma = 0.99$.†

Episodic/continuing, time limit, and termination criteria The task is continuing.† There is no time limit nor termination criterion.†

A.5 Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning [18]

Reward function The reward function is the unweighted sum of 3 attributes:

- -0.01 given every step;
- -10 if a collision occurred (and 0 otherwise); and
- $+1$ when the agent successfully reaches the destination beyond the intersection (and 0 otherwise).

Time step duration Time steps are 200 ms, though some actions last 2, 4, or 8 time steps (i.e., the agent may frame skip).

Discount factor The discount factor $\gamma = 0.99$.†

Episodic/continuing, time limit, and termination criteria The task is episodic. In unoccluded scenarios, episodes are limited to 20 s. In occluded scenarios, episodes are limited to 60 s. Termination occurs upon success, running out of time, or collision.

A.6 End-to-End Race Driving with Deep Reinforcement Learning [19]

Reward function Four different reward functions are evaluated, three of which are new and intentionally add reward shaping:

- $r = v$
- $r = v(\cos\alpha - d)$ ("Ours")
- $r = v(\cos\alpha - \max(d - 0.5w, 0))$ ("Ours - w/ margin")
- $r = v(\cos\alpha - \frac{1}{1+\exp -4(|d|-0.5w})$ ("Ours - sigmoid")

Above, v is the vehicle velocity component in the direction of the lane's center line, d is the distance from the middle of the road, α is the difference between the vehicle's heading and the lane's heading, and w is the road width.

Time step duration Time step duration is 33ms, with 1 step per 30 FPS frame.† The game will pause to await the RL agent's action.

Discount factor The discount factor $\gamma = 0.99$.†

Episodic/continuing, time limit, and termination criteria The task is episodic. No time limit is enforced. The termination criteria are the vehicle stops progressing or the vehicle goes off-road or the wrong direction.

A.7 CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving [23]

Reward function The reward function is the unweighted sum of 5 attributes:

- penalty for steering angles in ranges assumed incorrect for current command (e.g., going left during a turn-right command);
- speed (km/h), with penalties for going too fast on a turn and limits to speed-based reward when not turning (to keep under a speed limit);
- -100 upon collision with vehicles or pedestrians and -50 upon collision with anything else (e.g., trees and poles) (and 0 otherwise);
- -100 for overlapping with the sidewalk (and 0 otherwise); and
- -100 for overlapping with the opposite-direction lane (and 0 otherwise).

Time step duration Time steps last 100 ms, the same as has been reported in other research conducted within the CARLA simulator.

Discount factor The discount factor $\gamma = 0.9$.

Episodic/continuing, time limit, and termination criteria The task is episodic. The time allotted for an episode is the amount of time to follow the "optimal path" to the goal at 10 km/h. Termination occurs upon successfully reaching the destination, having a collision, or exhausting the allotted time.

A.8 Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination [26]

Reward function The reward function is the unweighted sum of 4 attributes:

- $\frac{v-40}{40}$, where v is speed in km per hour within the allowed range of $[40, 80]$ km/h;
- 0.5 if the ego vehicle overtakes another vehicle (0 otherwise);
- -0.25 if the ego vehicle changes lane (0 otherwise); and
- -10 if the ego vehicle collides (0 otherwise).

Time step duration The time step duration is the time between frames in the Unity-based simulator. The correspondence of such a frame to seconds in the simulated world was unknown to the first author.†

Discount factor The discount factor $\gamma = 0.99$.

Episodic/continuing, time limit, and termination criteria The task is episodic. There is no time limit.† Termination occurs upon collision with another vehicle or when the ego vehicle travels the full track length (2500 Unity spatial units†), effectively reaching a goal.

A.9 Learning hierarchical behavior and motion planning for autonomous driving [45]

Reward function The reward function is defined separately for transitions to terminal and non-terminal states. For transitions to terminal states, reward is one of the following:

- 100 if the goal was reached;
- -50 upon a collision or running out of time;
- -10 for a red light violation; or
- -1 if the ego vehicle is in the wrong lane.

The reward for a single *non-terminal* high-level behavioral step is the negative sum of the costs of the shorter steps within the best trajectory found by the motion planner, which executes the high-level action. Expressed as the additive inverse of cost, this reward for a high-level behavioral step is the unweighted sum of these 3 attributes:

- $\frac{-\sum_t t^2 |v_{ref} - v(t)|}{\sum_t t^2}$, which rewards speeds close to the desired speed, v_{ref};
- $\frac{-1}{1+\sum_t |v(t)|}$, which rewards based on distance traveled; and
- $\sum_t [0.02 \times d_{olon}(t) + 0.1 \times d_{olat}(t)]$, which rewards keeping larger distances from obstacles.

Distances are in meters and speed in m/s.† Desired speed v_{ref} is determined by the RL agent's high-level actions *speed_up* and *speed_down*, which each change v_{ref} by a fixed percentage. v_t is the current speed. Time t is a count from 1 over the time steps of the planned trajectory.† d_{olon} and d_{olat} are respectively the longitudinal and lateral distance from the closest vehicle within 20 meters, measured between the centers of the cars. If no vehicle is present within that range, $d_{olon} = 20$ and $d_{olat} = 3.5$.†

Time step duration A new high-level action is only chosen by the RL agent once the planned trajectory from the previous high-level action is completed. In the simulator, this results in an *average* time step of 1s for the RL-based behavior policy. The the lower-level motion planner averages 100ms per planning iteration.

Discount factor Discount factor $\gamma = 0.99$.

Episodic/continuing, time limit, and termination criteria The task is episodic. The time limit is the time required to travel the length of a randomly generated route in a CARLA town at 10 km/h. Episodes are terminated when the goal is reached or any of the following occur: a collision, driving out of the lane, or a red light violation.

A.10 End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances [40]

Reward function The reward function outputs $r = r_{speed} + 0.5 \times r_{dist} + 0.5 \times r_{heading}$ and is in $[-1, 1]$.

- r_{speed}: $1 - |s_{desired} - s_{ego}|/40$,† an attribute in $[0, 1]$ (40 km/h is the maximum $s_{desired}$†) that is inversely proportional to the absolute difference in the actual speed and the desired speed;
- r_{dist}: $-d_{path}/d_{max}$, an attribute in $[-1, 0]$, where $d_{max} = 2.0$ and d_{path} is the distance in meters from the closest point on the optimal path's spline;
- $r_{heading}$: $clip(-1, 0, b \times |\theta_{ego} - \theta_{path}|)$†, an attribute in $[-1, 0]$, where θ_{ego} is the ego vehicle's heading and θ_{path} is the heading of the optimal path spline at its closest point; and
- -1 upon termination (0 otherwise).

To determine reward, an *optimal path* is created via the waypoint API. This path is optimal with simplifying assumptions. In training, this waypoint-generated optimal path is generated by randomly choosing turn directions at intersections.

The unit for speeds is km/h, and the unit for angles is degrees. The desired speed, $s_{desired}$, is hard coded based on the presence of traffic lights and obstacles, using "priveleged" information available only during training. d_{max} is half the width of a lane (which apparently is always 2 m in CARLA). In the formula for $r_{heading}$, $b = (-1/10)$ when the optimal path's accompanying high-level command is to follow the lane or go straight through an intersection, and $b = (-1/25)$ when the command is to turn left or right through an intersection.†

We note that this reward function appears somewhat supervisory, teaching the target speed in various contexts and to stay close to the precomputed path. Further, the context required to calculate reward is not available to the agent during testing.

Time step duration Time steps last 100 ms, the same as has been reported in other research conducted within the CARLA simulator.

Discount factor Discount factor $\gamma = 0.99$.†

Episodic/continuing, time limit, and termination criteria The task is episodic. There is no time limit during training. During training, there were no successful terminations (i.e., upon reaching a destination); instead, driving continued along a procedurally generated route until an undesirable termination condition was met.† Termination conditions are when the agent is further from the optimal path than d_{max}, collisions, running a red light, and having 0 speed when neither behind an obstacle nor waiting at a red traffic light.

B Reward shaping examples per paper

Of the 19 papers we reviewed, we are highly confident that 13 include reward shaping. Below is at least one example per paper of behavior discouraged or encouraged by reward shaping. The following examples are discouraged behavior, penalized via negative reward:

- deviating from the center of the lane [19, 40, 31, 38],
- changing lanes [17],
- overlapping with the opposite-direction lane [10, 23],
- overlapping a lane boundary [16],
- delaying entering the intersection when it is the ego vehicle's right of way at a stop sign [22],
- deviating from steering straight [7],
- side-ways drifting (on a race track) [24],
- getting close to other vehicles [16], and
- having the turn signal on [38].

These following two examples are behaviors that are encouraged by positive reward:

- passing other vehicles [26] and
- increasing distances from other vehicles [45].

Additionally, 2 papers had reward attributes that were somewhat defensible as not constituting reward shaping. The reward function in Wang et al. [47] also includes a penalty correlated with the lateral distance from center of the lane, but their reinforcement learning algorithm is *explicitly* a lower-level module that receives a command for which lane to be in, and one could argue the subtask of this module is to stay inside that lane. However, being perfectly centered in the lane is not part of that high-level command, so we think the argument for considering it to be reward shaping is stronger than the argument for the alternative. A reward attribute in Aradi et al. [3] encourages being in the rightward lane if no car is in the way, which fits laws in *some* US states that require drivers to keep right unless passing. Therefore, whether their reward function includes reward shaping depends on the laws for the location of the ego vehicle.

Lastly, we are confident that 4 papers do not include reward shaping [18, 6, 20, 27].

C Calculation of trajectory returns

This appendix section describes how we estimate the return for various trajectories under each reward function. These calculations are used for two related sanity checks for reward functions: comparing preference orderings (Section 4.2) and comparing indifference points (Section 4.3).

Recall that we estimate returns for 3 different types of trajectories:

- τ_{crash}, a drive that is successful until crashing halfway to its destination;
- τ_{idle}, the safe trajectory of a vehicle choosing to stay motionless where it was last parked; and
- τ_{succ}, a trajectory that successfully reaches the destination.

We additionally remind readers that an indifference point is calculated by solving the following equation for p when $\tau_A \prec \tau_B \prec \tau_C$:

$G(\tau_B) = pG(\tau_C) + (1-p)G(\tau_A)$.

In this paper, we calculate it specifically with $G(\tau_{idle}) = pG(\tau_{succ}) + (1-p)G(\tau_{crash})$. And from p, the *safety indifference point*, expressed in km per crash, can be calculated as $((p/(1-p)) + 0.5) \times$ path length, where *path length* is the length of a successful trajectory. Note that $p/(1-p)$ expresses the amount of τ_{succ} trajectories per half-length τ_{crash} trajectory at the indifference point, making $(p/(1-p)) + 0.5$ the path lengths driven per collision.

To illustrate, assume that for successful-until-collision τ_{crash}, $G(\tau_{crash}) = -10$; for motionless τ_{idle}, $G(\tau_{idle}) = -5$; and for successful τ_{succ}, $G(\tau_{succ}) = 10$. Recall that the indifference point p is where the utility function has no preference over τ_{idle} and a lottery between τ_{crash} and τ_{succ} according to $pG(\tau_{succ}) + (1-p)G(\tau_{crash})$. For this example, $-5 = (p \times 10) + ((1-p) \times -10)$, and solving the equation results in $p = 0.25$. Therefore, the utility function would prefer driving (more than not driving) with any ratio higher than 1 success to 3 crashes. Let us also assume that a path length is 1 km. Therefore, at the indifference point, for each collision the car would drive 0.33 km on successful drives and *half* of 1 km on a drive with a collision, which is

$(p/(1-p)) + 0.5$ path lengths per collision $\times\ 1$ km per path length $= ([0.25/(0.75)] + 0.5) = 0.83$ km per collision.

Calculations of safety indifference points are given below for the two papers for which $\tau_{crash} \prec \tau_{idle} \prec \tau_{succ}$ [6, 18].

In our descriptions below, we try to name every significant assumption we make. Readers might find it useful to choose different assumptions than ours to test how sensitive our analysis is to these assumptions; we expect the reader will find no changes in the qualitative results that arise from this quantitative analysis.

C.1 General methodology and assumptions

To estimate return for some trajectory τ_i, we do not fully ground it as a sequence of state-action pairs. As we show in the remainder of this appendix section, the exact state-action sequence is not needed for estimating certain trajectories' returns under these reward functions.

For calculating reward per time step, we adhere to the following methodology.

- To determine the return/utility for a successful portion of a trajectory, we assume no unnecessary penalties are incurred (e.g., driving on the sidewalk).
- For positive attributes of reward, we choose the value that gives the maximum outcome. If the maximum is unclear, we choose outcomes for attributes that are as good or better than their best-reported experimental results. Lastly, if experimental results do not include a measure of the reward attribute, we attempt to calculate a value that is better than what we expect a typical human driver to do.
- Path lengths are the given length of the paper's driving tasks or our estimation of it. If the given information is insufficient to estimate the path length, we assume it to be 1 km.

For the papers paper below, we write out the units for each term in the equations for the return of a trajectory. We encourage the reader to refer to the paper's corresponding subsection in Appendix A to understand our calculations of return. Additionally, to aid the reader, we use specific colors for terms expressing the time limit, path length, time step duration, and speed. For reward functions that are sums of attributes, in our return calculations we maintain the order of the attributes as they were described in Appendix A, and we include 0 terms for attributes that do not affect return for the corresponding trajectory.

C.2 Assumptions and calculations for each paper

LeTS-Drive: Driving in a Crowd by Learning from Tree Search [6] Driving maps are 40 m $\times$ 40 m and are each a single intersection or curve, and from this map size we assume the path length is 40 m. This assumption appears reasonable because the car is spawned in a random location.

For the successful path τ_{succ} and the successful portion of path τ_{crash}, we use mean task time and the number of deceleration events reported for their best algorithm. The motionless τ_{idle} involves time running out, which is not actually a termination event in their method but is treated so here. The driving speed when collision occurs is assumed to be the mean speed for LeTS-Drive calculated from their Table 1's Time-to-goal and the path length.

Path length: 0.04 km.

A trajectory that is successful until collision:

$$\begin{aligned}G(\tau_{crash}) &= -515.71\\ &= (-0.1 \text{ reward / time step} \times 10 \text{ time step / s} \times 29.6 \text{ s / successful trajectory} \times 0.5 \text{ of full path length traveled})\\ &\quad + (-0.1 \text{ reward / acceleration event} \times 18.2 \text{ mean acceleration events / successful episode} \times 0.5 \text{ of full path length traveled})\\ &\quad + (-1000 \times (40 \text{ m path length}/29.6 \text{ s / successful trajectory})^2 + 0.5)\end{aligned}$$

A motionless trajectory:

$$\begin{aligned}G(\tau_{idle}) &= -120\\ &= (-0.1 \text{ efficiency reward / time step} \times 10 \text{ time step / s} \times 120 \text{ s before time limit is reached})\\ &\quad + (-0.1 \text{ smoothness reward / acceleration event} \times 0 \text{ acceleration events / episode})\\ &\quad + 0 \text{ safety reward}\end{aligned}$$

A successful trajectory:

$$\begin{aligned}G(\tau_{succ}) &= -31.42\\ &= (-0.1 \text{ efficiency reward / time step} \times 10 \text{ time step / s} \times 29.6 \text{ s / successful trajectory})\\ &\quad + (-0.1 \text{ smoothness reward / acceleration event} \times 18.2 \text{ mean acceleration events / successful episode})\\ &\quad + (0 \text{ for no collision})\end{aligned}$$

Calculation of the indifference point:

$$\begin{aligned}G(\tau_{idle}) &= pG(\tau_{succ}) + (1-p)G(\tau_{crash})\\ p &= \frac{G(\tau_{idle}) - G(\tau_{crash})}{G(\tau_{succ}) - G(\tau_{crash})}\\ p &= \frac{-120 - (-515.71)}{-31.42 - (-515.71)}\\ p &= 0.9617\end{aligned}$$

Calculation of km per collision at the indifference point:

$$\begin{aligned}[(p/(1-p)) + 0.5 \text{ path lengths per collision}] \times 0.02 \text{ km per path length} &= [(0.9617/(1-0.9617)) + 0.5] \times 0.04\\ &= 1.02 \text{ km per collision}\end{aligned}$$

Model-free Deep Reinforcement Learning for Urban Autonomous Driving [7] We assume a constant speed of 5 m/s (18 km/h), which is the most reward-giving speed for the speed-based reward attribute. This paper focuses on a specific roundabout navigation task, which presumably would be a shorter route than those used in the more common CARLA benchmarks first established by Dosovitskiy et al. [10]. Accordingly, differing from our assumptions for most other CARLA evaluations, we assume a 0.125 km path length, the distance which can be achieved at the above speed in exactly half of the permitted 50 s time limit. We further assume that the steering angle is always 0 (avoiding a penalty) and that the agent never leaves its lane except upon a collision. Also, recall that reward is accrued at 100 ms time steps (whereas discounting is applied at 400 ms time steps).

Path length: 0.125 km.

A trajectory that is successful until collision:

$$\begin{aligned}G(\tau_{crash}) &= 601.5\\ &= (5 \text{ m / s} \times (((125 \text{ m path length} \times 0.5 \text{ of full path length traveled})/5 \text{ m / s}) \times 10 \text{ time step / s}))\\ &\quad + (0 \text{ for no deviations from 0 steering angle}) + (-10 \text{ for the collision}) + (-1 \text{ for leaving the lane before the collision})\\ &\quad + (-0.1 \text{ constant reward / time step} \times (((125 \text{ m path length} \times 0.5 \text{ of full path length traveled})/5 \text{ m / s}) \times 10 \text{ time step / s}))\end{aligned}$$

A motionless trajectory:

$$\begin{aligned}G(\tau_{idle}) &= -50.0\\ &= (0 \text{ for 0 speed throughout})\\ &\quad + (0 \text{ for no deviations from 0 steering angle}) + (0 \text{ for no collision}) + (0 \text{ for never leaving the lane})\\ &\quad + (-0.1 \text{ constant reward / time step} \times (50 \text{ s time limit} \times 10 \text{ time step / s})\end{aligned}$$

A successful trajectory:

$$\begin{aligned}G(\tau_{succ}) &= 1225.0\\ &= (5 \text{ m / s} \times ((125 \text{ m path length}/5 \text{ m / s}) \times 10 \text{ time step / s}))\\ &\quad + (0 \text{ for no deviations from 0 steering angle}) + (0 \text{ for no collision}) + (0 \text{ for never leaving the lane})\\ &\quad + (-0.1 \text{ constant reward / time step} \times ((125 \text{ m path length}/5 \text{ m / s}) \times 10 \text{ time step / s}))\end{aligned}$$

CARLA: An open urban driving simulator [10] For a successful drive, we assume a 1 km path, a change in speed from start to finish is 60 km/h, and that no overlap occurs with sidewalk or other lane. For a trajectory with a collision, we assume the collision damage is total (i.e., 1) and the ego vehicle completely overlaps with sidewalk or other lane at 60 km/h.

Path length: 1 km.

A trajectory that is successful until collision:

$$\begin{aligned}G(\tau_{crash}) &= 501.00\\ &= (500 \text{ m path length} \times 1 \text{ reward / m traveled}) + (0.05 \times 60 \text{ km / h increased over trajectory})\\ &\quad + (-0.00002 \times 1 \text{ for full collision damage}) + (0 \text{ for never overlapping with the sidewalk})\\ &\quad + (-2 \times 1 \text{ for fully leaving the lane before collision})\end{aligned}$$

A motionless trajectory:

$$\begin{aligned}G(\tau_{idle}) &= 0\\ &= (0 \text{ for no distance traveled}) + 0 \text{ km / h increased over trajectory}\\ &\quad + (0 \text{ for no collision damage}) + (0 \text{ for never overlapping with the sidewalk}) + (0 \text{ for never leaving the lane})\end{aligned}$$

A successful trajectory:

$$\begin{aligned}G(\tau_{succ}) &= 1003\\ &= (1000 \text{ m path length} \times 1 \text{ reward / m traveled}) + (0.05 \times 60 \text{ km / h increased over trajectory})\\ &\quad + (0 \text{ for no collision damage}) + (0 \text{ for never overlapping with the sidewalk}) + (0 \text{ for never leaving the lane})\end{aligned}$$

Dynamic Input for Deep Reinforcement Learning in Autonomous Driving [17] Because a collision appears impossible in this task, this reward function was not involved in the analysis of preference orderings and indifference points.

Navigating Occluded Intersections with Autonomous Vehicles using Deep Reinforcement Learning [18] We assume that the path is 6 lane widths, which is roughly what the "Left2" turn requires and that each lane is 3.35 m wide (based on 11 feet appearing common enough, e.g., in `https://mutcd.fhwa.dot.gov/rpt/tcstoll/chapter443.htm`). For the successful drive, we assume 4 s was required. We focus on the unoccluded scenario with a 20 s time limit.

Path length: $(3.35 \text{ m / lane width} \times 6 \text{ lane widths / path length})/1000 \text{ m / km} \approx 0.02$ km.

A trajectory that is successful until collision:

$$\begin{aligned} G(\tau_{crash}) &= -10.1 \\ &= (0.5 \text{ of full path length traveled} \times (4 \text{ s / successful trajectory}/0.2 \text{ s / time step}) \times -0.01 \text{ reward / time step}) \\ &\quad + (-10 \text{ for collision}) \\ &\quad + (0 \text{ for not reaching the goal}) \end{aligned}$$

A motionless trajectory:

$$\begin{aligned} G(\tau_{idle}) &= -1 \\ &= ((20 \text{ s time limit}/0.2 \text{ s / time step}) \times -0.01 \text{ reward / time step}) \\ &\quad + (0 \text{ for no collision}) \\ &\quad + (0 \text{ for not reaching the goal}) \end{aligned}$$

A successful trajectory:

$$\begin{aligned} G(\tau_{succ}) &= 0.8 \\ &= ((4 \text{ s / successful trajectory}/0.2 \text{ s / time step}) \times -0.01 \text{ reward / time step}) \\ &\quad + (0 \text{ for no collision}) \\ &\quad + (1 \text{ for reaching the goal}) \end{aligned}$$

Calculation of the indifference point:

$$\begin{aligned} G(\tau_{idle}) &= pG(\tau_{succ}) + (1-p)G(\tau_{crash}) \\ p &= \frac{G(\tau_{idle}) - G(\tau_{crash})}{G(\tau_{succ}) - G(\tau_{crash})} \\ p &= \frac{-1 - (-10.1)}{0.8 - (-10.1)} \\ p &= 0.8349 \end{aligned}$$

Calculation of km per collision at the indifference point:

$$\begin{aligned} [(p/(1-p)) + 0.5 \text{ path lengths per collision}] \times 0.02 \text{ km per path length} &= [(0.8349/(1-0.8349)) + 0.5] \times 0.02 \\ &= 0.11 \text{ km per collision} \end{aligned}$$

End-to-End Race Driving with Deep Reinforcement Learning [19] Note that the domain is a car-racing video game, so safety constraints differ from autonomous driving driving. For successful driving, we assume 72.88 km/h, which is the average speed reported, and a 9.87 km track. We assume that the ego vehicle's heading is always aligned with the lane and the car is always in the center of the lane.

Path length: 9.87 km.

A trajectory that is successful until collision:

$$\begin{aligned} G(\tau_{crash}) &= 532980 \\ &= ((9.87 \text{ km path length} \times 0.5 \text{ of full path length traveled} \times (1/72.88 \text{ km / h}) \times 3600 \text{ s / h} \times 30 \text{ time steps / s}) \\ &\quad \times 72.88 \text{ km / h} \times 1 \text{ reward per km / h}) \\ &\quad + (0 \text{ for heading always aligned with the lane}) + (0 \text{ for always at the lane center}) \end{aligned}$$

A motionless trajectory:

$$\begin{aligned} G(\tau_{idle}) &= 0 \\ &= (0 \text{ for 0 km / h always}) \\ &\quad + (0 \text{ for heading always aligned with the lane}) + (0 \text{ for always at the lane center}) \end{aligned}$$

A successful trajectory:

$$\begin{aligned}G(\tau_{succ}) &= 1065960\\ &= ((9.87 \text{ km path length} \times (1/72.88 \text{ km / h}) \times 3600 \text{ s / h} \times 30 \text{ time steps / s})\\ &\quad \times 72.88 \text{ km / h} \times 1 \text{ reward per km / h})\\ &\quad + (0 \text{ for heading always aligned with the lane}) + (0 \text{ for always at the lane center})\end{aligned}$$

CIRL: Controllable Imitative Reinforcement Learning for Vision-based Self-driving [23] For successful drives, we assume a 60 km / h speed that is within speed limit. We assume no penalties are incurred. When a collision occurs, we assume overlap the opposite-direction lane for 1 s (which has an equivalent impact as overlap with the sidewalk for 1 s), a 60 km / h speed that is within the speed limit, no steering-angle penalty, and collision with a vehicle specifically. As for other CARLA-based research, we assume a 1 km successful trajectory, which creates a 6 minute time limit.

Path length: 1 km.

A trajectory that is successful until collision:

$$\begin{aligned}G(\tau_{crash}) &= 16900\\ &= (0 \text{ for always-0 steering angle})\\ &\quad + (60 \text{ km / h}\\ &\quad\quad \times ((1 \text{ km path length} \times 0.5 \text{ of full path length traveled})/60 \text{ km / h}) \times 3600 \text{ s / h} \times 10 \text{ times steps / s})\\ &\quad + (-100 \text{ for collision with a vehicle}) + (0 \text{ for never overlapping with the sidewalk})\\ &\quad + (-100 \text{ for overlapping with the opposite-direction lane} \times 1 \text{ s of overlap} \times 10 \text{ times steps / s})\end{aligned}$$

A motionless trajectory:

$$\begin{aligned}G(\tau_{idle}) &= 0\\ &= (0 \text{ for always-0 steering angle}) + (0 \text{ for always } 0 \text{ km / h})\\ &\quad + (0 \text{ for no collision}) + (0 \text{ for never overlapping with the sidewalk})\\ &\quad + (0 \text{ for never overlapping with the opposite-direction lane})\end{aligned}$$

A successful trajectory:

$$\begin{aligned}G(\tau_{succ}) &= 36000\\ &= (0 \text{ for always-0 steering angle})\\ &\quad + (60 \text{ km / h}\\ &\quad\quad \times (1 \text{ km path length}/60 \text{ km / h}) \times 3600 \text{ s / h} \times 10 \text{ times steps / s})\\ &\quad + (0 \text{ for no collision}) + (0 \text{ for never overlapping with the sidewalk})\\ &\quad + (0 \text{ for never overlapping with the opposite-direction lane})\end{aligned}$$

Deep Distributional Reinforcement Learning Based High-Level Driving Policy Determination [26] For a successful trajectory and the successful portion of the trajectory with a collision, we assume 17 overtakes per km (based on "Distance" in meters and "Num overtake" statistics shown in Fig. 7 in their paper and assuming those statistics are taken from a good trajectory) and an average of 1 lane change per overtake. We also assume that the car is always driving at 80 km / h, the speed that accrues the most reward. Since the minimum speed is 40 km / h and stopping in this paper's task—highway driving—would be unsafe, we instead assume the vehicle can decline to be deployed for 0 return. Since the first author did not know the duration of a time step in simulator time, we assume a common Unity default of 30 frames per second (and therefore 30 time steps per second) and that Unity processing time equals simulator time; consequently, 0.033 s time steps are assumed. We also assume a 1 km path, since the first author also did not have access to the path length for the task.

Path length: 1 km.

A trajectory that is successful until collision:

$$\begin{aligned}G(\tau_{crash}) &= 673.9\\ &= (((80 \text{ km / h} - 40)/40)\\ &\quad\quad \times (0.5 \text{ of full path length traveled} \times (1 \text{ km path length}/80 \text{ km / h}) \times 3600 \text{ s / h} \times 30 \text{ time steps / s}))\\ &\quad + (0.5 \text{ reward / overtake} \times (17 \text{ overtakes / km} \times 1 \text{ km path length} \times 0.5 \text{ of full path length traveled}))\\ &\quad + (-0.25 \text{ reward / lane change}\\ &\quad\quad \times (17 \text{ overtakes / km} \times 1 \text{ lane change / overtake} \times 1 \text{ km path length} \times 0.5 \text{ of full path length traveled}))\\ &\quad + (-10 \text{ for collision})\end{aligned}$$

A motionless trajectory:

$G(\tau_{idle}) = 0$

A successful trajectory:

$$\begin{aligned}G(\tau_{succc}) &= 1357.9\\
&= (((80 \text{ km / h} - 40)/40) \times ((1 \text{ km path length}/80 \text{ km / h}) \times 3600 \text{ s / h} \times 30 \text{ time steps / s}))\\
&\quad + (0.5 \text{ reward / overtake} \times (17 \text{ overtakes / km} \times 1 \text{ km path length}))\\
&\quad + (-0.25 \text{ reward / lane change} \times (17 \text{ overtakes / km} \times 1 \text{ lane change / overtake} \times 1 \text{ km path length} \times))\\
&\quad + (0 \text{ for no collision})\end{aligned}$$

Learning hierarchical behavior and motion planning for autonomous driving [45] We assume a 1 km path length and a speed of 60 km / h (or 16.67 m / s), as we do for most other CARLA evaluations. We also assume that the desired speed v_{ref} is always 60 km / h, making the first per-time-step component result in 0 reward each step, and that high-level RL time steps last exactly 1 s, which is their reported mean duration. Lastly, we assume that no other vehicles ever are closer than 20 m from the ego vehicle. Because the path length is assumed to be 1 km, the time limit is 360 s (based on the time limit information in Appendix A).

Path length: 1 km.

A trajectory that is successful until collision:

$$\begin{aligned}G(\tau_{crash}) &= 174.8\\
&= (0 \text{ for always matching } v_{ref})\\
&\quad + ((((1 \text{ km path length} \times 0.5 \text{ of full path length traveled})/60 \text{ km / h}) \times 3600 \text{ s / h} \times 1 \text{ RL time steps / s})\\
&\qquad \times (-1/(1 + (10 \text{ planning time steps / RL time step} \times 16.67 \text{ m / s}))))\\
&\quad + ((((1 \text{ km path length} \times 0.5 \text{ of full path length traveled})/60 \text{ km / h}) \times 3600 \text{ s / h} \times 1 \text{ RL time steps / s})\\
&\qquad \times 10 \text{ planning time steps / RL time step} \times ((0.02 \times 20 \text{ m}) + (0.1 \times 3.5 \text{ m})))\\
&\quad + (-50 \text{ for a collision})\end{aligned}$$

A motionless trajectory:

$$\begin{aligned}G(\tau_{idle}) &= -3711.2\\
&= ((360 \text{ s time limit} \times 1 \text{ RL time steps / s})\\
&\qquad \times (-|16.67 \text{ m / s} - 0 \text{ m / s}| \times 1 \text{ remaining terms after constant speed differential is moved outside summation})\\
&\quad + ((360 \text{ s time limit} \times 1 \text{ RL time steps / s})\\
&\qquad \times (-1/(1 + (10 \text{ planning time steps / RL time step} \times 0 \text{ km / h}))))\\
&\quad + ((360 \text{ s time limit} \times 1 \text{ RL time steps / s})\\
&\qquad \times 10 \text{ planning time steps / RL time step} \times ((0.02 \times 20 \text{ m}) + (0.1 \times 3.5 \text{ m})))\\
&\quad + (-50 \text{ for running out of time})\end{aligned}$$

A successful trajectory:

$$\begin{aligned}G(\tau_{succ}) &= 549.6\\
&= (0 \text{ for always matching } v_{ref})\\
&\quad + ((((1 \text{ km path length})/60 \text{ km / h}) \times 3600 \text{ s / h} \times 1 \text{ RL time steps / s})\\
&\qquad \times (-1/(1 + (10 \text{ planning time steps / RL time step} \times 16.67 \text{ m / s}))))\\
&\quad + ((((1 \text{ km path length})/60 \text{ km / h}) \times 3600 \text{ s / h} \times 1 \text{ RL time steps / s})\\
&\qquad \times 10 \text{ planning time steps / RL time step} \times ((0.02 \times 20 \text{ m}) + (0.1 \times 3.5 \text{ m})))\\
&\quad + (100 \text{ for reaching the goal})\end{aligned}$$

End-to-End Model-Free Reinforcement Learning for Urban Driving using Implicit Affordances [40] As we do for other CARLA evaluations, we assume a 1 km path length. We assume a speed of 30 km / h, based on the first author's report that 40 km/h was their maximum speed. We also assume termination occurs upon reaching some destination, which is only true during testing, since training involves driving until termination by failure. No additional reward is given at such successful termination. For τ_{crash} and τ_{succ}, we assume that the ego vehicle is always moving at the speed, location, and heading. We also assume that the 0-speed termination condition is not applied immediately but rather at 10 s and later; this assumption is made to avoid terminating at the starting time step, when the vehicle is spawned with a speed near 0 km / h.

Path length: 1 km.

A trajectory that is successful until collision:

$$\begin{aligned} G(\tau_{crash}) &= 599 \\ &= (1 \text{ reward for } s_{ego} \text{ always matching } s_{desired} \\ &\quad \times (1 \text{ km path length} \times 0.5 \text{ of full path length traveled}) \times (1/30 \text{ km / h}) \times 3600 \text{ s / h} \times 10 \text{ time steps / s}) \\ &\quad + (0 \text{ for always } d_{path} = 0) \\ &\quad + (0 \text{ for } \theta_{ego} \text{ always matching } \theta_{path}) \\ &\quad + (-1 \text{ for collision}) \end{aligned}$$

A motionless trajectory:

$$\begin{aligned} G(\tau_{idle}) &= 25 \\ &= (1 - (|0 \text{ km / h} - 30 \text{ km / h } s_{desired}|/40)) \times (10 \text{ s before 0-speed termination condition is applied} \times 10 \text{ time steps / s}) \\ &\quad + (0 \text{ for always } d_{path} = 0) \\ &\quad + (0 \text{ for } \theta_{ego} \text{ always matching } \theta_{path}) \end{aligned}$$

A successful trajectory:

$$\begin{aligned} G(\tau_{succ}) &= 1200 \\ &= (1 \text{ reward for } s_{ego} \text{ always matching } s_{desired} \times (1 \text{ km path length}/30 \text{ km / h}) \times 3600 \text{ s / h} \times 10 \text{ time steps / s}) \\ &\quad + (0 \text{ for always } d_{path} = 0) \\ &\quad + (0 \text{ for } \theta_{ego} \text{ always matching } \theta_{path}) \end{aligned}$$