HAIL: Modular Agent-Based Pedestrian Imitation Learning

André Antakli, Igor Vozniak, Nils Lipp, Matthias Klusch and Christian Müller

German Research Center for Artificial Intelligence (DFKI)
Stuhlsatzenhausweg 3, 66123 Saarbruecken, Germany
firstname.lastname@dfki.de

Abstract. In the area of autonomous driving there is a need to flexibly configure and simulate more complex individual pedestrian behavior in critical traffic scenes which goes beyond predefined behavior simulation. This paper presents a novel human-oriented, agent-based pedestrian simulation framework, named HAIL, that addresses this challenge. HAIL allows to simulate human pedestrian behavior through means of imitation learning by virtual agents. For this purpose, HAIL combines the 3D traffic simulation environment OpenDS with an integrated imitation learning environment and hybrid agents with AJAN. For predictive behavior planning on the tactical and strategical level, AJAN is extended with Answer Set Programming. For pedestrian behavior imitation learning on the operational level, HAIL utilizes the module InfoSalGAIL for generation of pedestrian paths learned from demonstration by its human counterpart as expert. Among others, an application example has been demonstrated that HAIL can be applied to solve a common challenge in the Neural Network domain, namely the out-of-distribution (OOD), e.g. never shown scenarios would raise an uncertainty prediction level, by unison work of the two different behavior generation frameworks.

Keywords: Pedestrian Simulation Framework · Multi-Agent System · Imitation Learning

1 Introduction

Pedestrian simulations are mostly considered in crowd scenarios. In such simulations the individual pedestrians are mathematical functions called particles, that can implement only a limited variety of emerging behavior. In state-of-the-art traffic simulation frameworks like Carla[1] or LGSVL[2] pedestrians follow only predefined trajectories. However, if higher-order behavior is to be simulated because the focus is set on the individual pedestrian, these models are no longer suitable, since various aspects of a pedestrian, such as activity planning through to actual movement, are no longer covered. In general (cf. [10]), a distinction is made between three layers of pedestrian behavior: the strategic level includes

[1] Carla: https://carla.org/
[2] LGSVL: https://www.lgsvlsimulator.com/

high-level decision-making, e.g. activity or trip planning based on interests; the tactical level, which divides the high-level activity plan into intermediate targets and tries to achieve them using atomic actions; and the operational level, which implements the actual action like a walking task and adjusts pedestrian speed, gait, and alignment. The agent paradigm is suitable for encapsulating these layers into a single autonomous entity. Especially the fields of video games or robotics have successfully used this model. In these areas, the simulation environment is represented abstractly in the Strategic and Tactical layer of the agent and is often processed via Behavior Trees. Instead, the Operational layer works directly with environmental properties, like its geometry or physics. Nevertheless, all simulation systems either use pre-modelled approximations of real human behavior [12] ,[18], or only certain aspects of the behavior model are considered but hardly transferable to other simulation scenarios due to their lack of modularity [11], [19].

To this end, we developed a novel approach called HAIL (human-oriented agent-based imitation learning) for the simulation of pedestrians in virtual traffic scenes. The resulting framework follows the above mentioned pedestrian behavior model, and combines modular predictive agents with imitated real pedestrian behavior in order to simulate more realistic traffic situations. With HAIL, it is possible to imitate demonstrated expert behavior of on-street walking on the operational level. For this purpose, HAIL leverages the imitation learning approach InfoSalGAIL [20] and the 3D driving simulation software OpenDS[3] (version 6.0) to set up traffic environment and visualize imitated pedestrian behavior in it. Finally, the agent system AJAN [2] is used in HAIL to represent more complex pedestrian behaviors on the strategic and tactical level based on both intrinsic and extrinsic pedestrian needs. The behavior model in AJAN relies on so called SPARQL-BTs, which was extended by means of Answer Set Programming (ASP) to realize utility-based foresighted activity planning.

The paper is structured as follows. Section 2 gives a brief introduction into the background required and discusses relevant state of the art on pedestrian agent engineering and pedestrian imitation learning. In Section 3, we present our contribution HAIL, the interplay of its components and describe how ASP is integrated for foresighted action planning. In Section 4, an application example in the context of simulated pedestrians is presented. Finally, we conclude the paper in Section 5.

2 Related Work

2.1 Pedestrian Agent Engineering

In the field of pedestrian simulation we mainly find solutions to simulate crowds, see PTV Viswalk[4], VADERE [5] or PEDSIM[6]. According to [18], these solutions

[3] OpenDS - open source driving simulation: https://opends.dfki.de/
[4] PTV Viswalk: https://www.ptvgroup.com/de/loesungen/produkte/ptv-viswalk/
[5] VADERE Crowd simulation: http://www.vadere.org/
[6] PDESIM - pedestrian crowd simulation: http://pedsim.silmaril.org/

are often based on social force, cellular or magnetic force models and are inspired as well by the work of [16]. The individuals of a crowd are purely reactive agents who do not pursue their own goals. The direct interaction between individuals is usually not considered in these approaches and the individual modeling of a autonomous higher order behavior is not given. However, if, for example, critical situations in road traffic are to be simulated in which the vehicle gains a "close" view of individual pedestrian behavior, these solutions are no longer useful. Due to the immense efforts that have recently been made in the field of autonomous driving, the need for individual pedestrian behavior simulation has increased. Prominent traffic simulation environments available in this context are Carla and LGSVL. These solutions only have primitive pedestrian models that usually follow a predefined path. Since Carla is based on the Unreal Engine it is possible to model the individual pedestrian behavior with Behavior Trees[7] (BT). BTs are widely used in the gaming industry and in robotics (see [15]) to realize deliberative agents. However, a predictive utility-based behavior covering intrinsic and extrinsic needs is hard to be implemented with BTs.

Modular agent engineering with AJAN. AJAN (Accessible Java Agent Nucleus) is an agent engineering framework, in which SPARQL-enhanced BTs, so called SPARQL-BTs (SBT) are used as an agent behavior model and agent models are defined in RDF (Resource Description Framework). Beside of dictionaries with key value pairs, which are often used in other BT solutions, the RDF data model is domain-independent and thus a more flexible and in combination with SPARQL a powerful alternative [5]. For an intuitive modeling of AJAN agents, a web editor is provided. AJAN has already been used to control virtual humans, see [2]. AJAN is available as open-source software[8] and is used in our approach to control single virtual pedestrians. An AJAN agent has one or more behaviors, each consisting of a SBT and a corresponding RDF-based execution knowledge (EKB), which stores internal behavior knowledge (e.g. procedural variables); one agent specific RDF-based knowledge base (AKB), storing internal agent knowledge, which can be accessed by all agent behaviors[9]; one or more events and goals, each holding RDF data; and one or more agent endpoints, which are the agent's interfaces to its domain and forward incoming RDF messages as events. Behaviors are linked to such events or goals but can also create these. If an event occurs, the behaviors linked to it are executed. SBTs are used to perform contextual SPARQL queries for state checking (e.g. realized with a SPARQL-ASK query), updating, constructing RDF data used for action executions, or to control the internal execution of an AJAN agent behavior. Furthermore, SBTs are defined in RDF, whereby a semantic description of the behaviors they implement is available. SBTs use standard BT primitives (see [7]) and are processed like typical BTs[10].

[7] Unreal - BTs: https://docs.unrealengine.com/InteractiveExperiences/BehaviorTrees
[8] AJAN-service: https://github.com/aantakli/AJAN-service
AJAN-editor: https://github.com/aantakli/AJAN-editor
[9] Not like EKs, where only the corresponding agent behavior has access to.
[10] AJAN uses LibGDX-BTs: https://github.com/libgdx/gdx-ai/wiki/Behavior-Trees

2.2 Imitation of Pedestrians in Simulated Environment

Behavior Cloning (BC) and Apprenticeship Learning (AL) are common approaches to address imitation learning challenges. Considering BC it suffers from moderate generalization due to compounding errors and covariant shift [17]. In contrary, AL tends to reconstruct the reward function [1] at high computational costs because of solving a reinforcement learning problem in the training loop. Generative Adversarial Imitation Learning (GAIL) [8], is a prominent approach in solving AL problems. The objective of which is to learn the optimal strategy for a given task without estimating an explicit reward function. An extension of GAIL was introduced in InfoGAIL [13], where the policy of a simulated car agent is estimated based on the mixture of expert trajectories, adding a direct relationship to the latent variables as in [6].

Pedestrian imitation learning with InfoSalGAIL. For the imitation learning module of HAIL, we selected the InfoSalGAIL [20] system. In particular, this system uses saliency maps of experts (recorded during a comprehensive study) for a more human-like imitation of virtual pedestrian walking behavior. It was shown that visual attention, represented in the form of saliency maps, indeed plays an important role in trajectory generation. However, the service-oriented architecture of HAIL also allows to integrate other imitation learning modules than InfoSalGAIL.

In InfoSalGAIL, the imitated behavior of a simulated pedestrian is considered safe or risky. This classification is based on the learned expert trajectories and the traffic areas entered with them, such as streets (risky) or crosswalks (safe) (cf. Fig. 3). In the context of a visual attention model, the task is to identify the most probable area of interest at any given point in time, which can be seen as a set of highlighted pixels as shown in Fig. 1. The training objective or loss function (cf. Equation 1) is defined as follows and has been experimentally shown to be efficiently working for the generation of different types of pedestrian walking behaviors:

$$\min_{\theta,\psi,\Delta} \max_{\omega} \mathbb{E}_{\pi_\theta}[D_\omega(s_{vis,sal}, a)] \; + \mathbb{E}_{\pi_E}[D_\omega(s_{vis,sal}, a)] - \lambda_0 \eta(\pi_\theta) - \lambda_1 L_1(\pi, Q) - \Delta_E(s) - \lambda H(\pi) \quad (1)$$

where π stands for the agent's policy, π_E the policy of the subject, D is the discriminative classifier with the overall goal to distinguish state-action pairs (synthetic vs. real). $H(\pi) \triangleq \mathbb{E}\left[-\log \pi(a|s)\right]$ denotes the γ - discount casual entropy of the policy π_θ as defined by [4]. λ_1 is the hyper-parameter for the information maximization regularization term L_1 as in [13]. The term $\eta(\pi_\theta) = \mathbb{E}_{s\sim\pi_\theta}[s_r]$ reflects the tendency towards learning of the desired behavior, thus, stands for the main reinforcement learning component which is obtained directly from the simulator. Δ_E is the pixel-wise loss based on the binary cross-entropy function, which helps to optimize the saliency map generation objective.

3 HAIL Framework

3.1 Overview

The HAIL framework is designed to imitate fore-sighted human-like behavior of pedestrians in previously unseen virtual traffic scenarios based on prior knowledge of prerecorded expert demonstrations, e.g. walking in virtual environment. HAIL can also be used to simulate pedestrians in traffic scenarios for which no prior training data exist. Fig. 1 shows the high-level architecture of HAIL, which consists of three main components, namely OpenDS as the simulation software, AJAN for controlling pedestrian agents, and InfoSalGAIL as Imitation Learning framework to account for the human-like trajectories learned during the training given the ground truth data of recorded subjects.

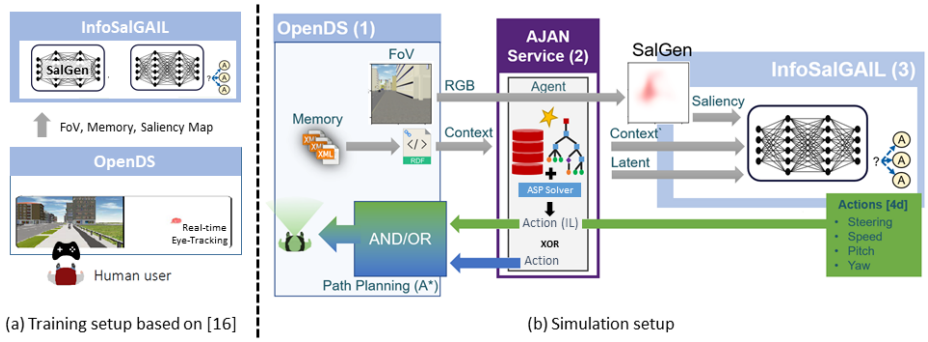


Fig. 1: HAIL framework overview. (a) Setup (cf. [20]) to learn from expert demonstrations. (b): Field of View (FoV) is the input image from OpenDS, where the memory contains current and previous simulation information. Path planning (A*) is supported within OpenDS to deliver a path if no imitation model in a OOD situation. SalGen generates saliency maps for the FoV images, where the context and latent variable are used as input information for the policy generator. The $4d$ action vector comes from InfoSalGAIL to control the pedestrian.

The starting component of HAIL is the open-source driving simulation software OpenDS (cf. Fig. 1 b(1)), which is used to simulate and visualize virtual traffic scenarios. OpenDS manages a scenario with a three-dimensional scene with semantic information about objects like traffic lights but also points-of-interest (POI), simulated vehicles, integrated path planning (A*) and atomic actions on the operational level. Such actions include performing animations like waving or operating a traffic light, navigating a pedestrian agent to a given destination via path planning and directly setting the steering vector of a pedestrian agent. In addition, it provides a training environment (cf. Fig. 1 (a)) for the imitation learning used in HAIL [20]. Moreover, it manages AJAN agents controlling simulated pedestrians and provides them with dynamic scenario information.
AJAN (cf. Section 2.1), the "man-in-the-middle" component of HAIL (cf. Fig. 1 b(2)), is used for decision-making of the pedestrian agent, to execute OpenDS

atomic actions, and to obtain new avatar transformations for it via the imitation learning service InfoSalGAIL. In this context, two strategies can be applied: Either (i) AJAN controls the virtual pedestrian using input from InfoSalGAIL, if an imitation model is available for a given agent state; or (ii) AJAN controls the pedestrian directly via OpenDS actions, if no such model exists. Which strategy is chosen actually depends on the SBTs and goals of the particular agent and its beliefs. When the beliefs are updated by OpenDS, the decision-making with SBTs is triggered. This involves evaluating which capabilities are available in the current state to achieve an existing goal.
InfoSalGAIL (cf. Section 2.2), the imitation learning component of HAIL (cf. Fig. 1 b(3)), allows to bring human-like behavior into the simulation through means of learning an optimal navigation policy by means of expert demonstrations. Unlike [20], where the latent code was manually set throughout the simulation, AJAN is responsible for dynamically defining the latent code based on its initial knowledge and knowledge gained throughout the run, therefore makes the overall simulation system more flexible.

3.2 Integration

For the integration of the HAIL components or services, RDF-based information is exchanged over HTTP between these; the interaction between components is summarized in the following.
OpenDS to AJAN: OpenDS initializes an AJAN-controlled pedestrian agent and defines its initial beliefs and goals. This includes the pedestrian agent position and information about the given traffic scene at $time = 0$. After initialization, updates on scene changes are sent to the agent, after which OpenDS listens for calls to perform atomic actions. An update includes "seen" POIs, positions of dynamic objects such as simulated vehicles and other pedestrians, but also states of virtual objects such as traffic lights. In order to simulate a human-like perception module, OpenDS has been extended with an additional visual module to cover the POIs only and the distances to the same, if they fall within the field of view (FoV) of the pedestrian agent. For the use of InfoSalGAIL, additional information about the current body orientation of the pedestrian agent, its speed and "view" (RGB image in combination with yaw and pitch angles for the head) as well as historical information about previously executed actions is also broadcast. The agent receives these updates and sends them to an SBT which updates the agent knowledge base and decides which navigation strategy to use. When it is recognized that an InfoSalGAIL model exists[11] in the current situation to perform a navigation task, the input data required by InfoSalGAIL is forwarded by the agent, otherwise OpenDS-based navigation is used.
AJAN to InfoSalGAIL: An important aspect of AJAN, besides to which destination to navigate for a given goal, is to determine the latent variable used in InfoSalGAIL. The latent variable specifies how risky a pedestrian should navigate. For example (cf. Fig. 3, Scenario A), a simulated pedestrian may initially

[11] Upon initialization, an AJAN agent receives RDF descriptions of available InfoSalGAIL models defining trained street configurations.

try to cross the street because of an approaching car. However, if the pedestrian agent realizes that the potential danger has passed, it decides to riskily cross the street. To execute InfoSalGAIL, the data received from OpenDS is merged with the adapted latent variable and forwarded by AJAN. After receiving the information, InfoSalGAIL responds immediately with a new action input for the pedestrian agent, which is forwarded by AJAN to OpenDS. The $4d$-input vector consists of *turning angle*, *speed*, and head orientation (*pitch* and *yaw* angles).
Execution in OpenDS: OpenDS provides multiple HTTP/RDF endpoints to perform pedestrian agent actions, such as *"set transformation"* in case we use InfoSalGAIL, or *"walk to target"* in case we use the path planner. During simulation, these endpoints listen for incoming action commands from AJAN. For example, if a target (a 3D-vector) is received to which the pedestrian agent should walk to, a path to the target is generated using A* and applied to the avatar. However, if the avatar transformation based on the output of InfoSalGAIL is to be adjusted directly, this action input is applied to the avatar and its history and the RGB image of the FoV are updated accordingly.

3.3 AJAN Extensions

AJAN is used in HAIL to realize the strategic and tactical layers of a simulated pedestrian and therefore to control its navigation. For this purpose, a destination is selected and, depending on the strategy, performed directly via OpenDS navigation or with InfoSalGAIL. A set of destinations to be reached sequentially can be manually defined using SBTs. In order to implement more complex scenarios in which the agent dynamically creates navigation sequences based on intrinsic or extrinsic needs, we implemented an ASP-SBT node[12] for reasoning, problem solving or to plan intention sequences. For this purpose, we adapted the RDF-to-ASP translation approach in [9] to translate RDF-based AJAN agent beliefs into ASP rules. Table 1 shows the five most important transformation rules with an example in Fig. 2 left.

Table 1: Basic RDF-to-ASP transformation rules.

	RDF Version:	⇔	**ASP Version:**
Triple:	<S> <P> <O>	⇔	_t("S","P","O")
Graph:	<IRI> { <S> <P> <O> }	⇔	_g(_t("S","P","O"), "IRI")
IRI:	IRI	⇔	"IRI"
Literal:	"I"^^<XSD>	⇔	_l("I", "XSD")
Blank:	_:blank123	⇔	_b("blank123")
Prefix:	@prefix react: <IRI>	⇔	_p("react","IRI")

If beliefs are available for solving navigation problems, then additional ASP rules (e.g., planning rules, see Fig. 4 and [3] section 2) need to be added. Pedestrian behavior could also integrate the social context of the environment (e.g., traffic norms modeled ins ASP [14]) such that the trajectories that are generated by the ASP planner can be evaluated from a normative point of view. The specified problem can then be solved using an ASP solver[13]. If no stable model is found

[12] ASP-SBT-node: https://github.com/aantakli/AJAN-service/tree/master/pluginsystem/plugins/ASPPlugin
[13] In AJAN we use the Potassco clingo solver: https://potassco.org/clingo/

then the ASP-SBT node returns FAILED as status otherwise SUCCEEDED; each model is stored by the agent as a RDF named graph (cf. Fig. 2 right).

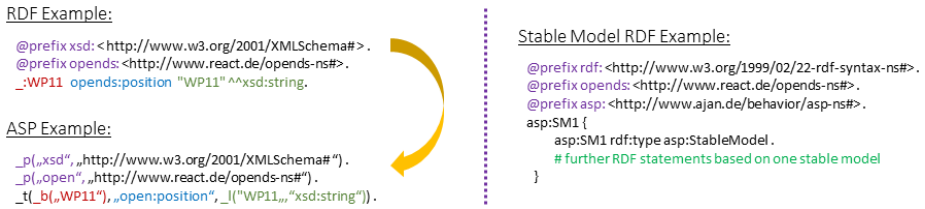


Fig. 2: Left: RDF-to-ASP example. Right: RDF representation of a stable model.

4 Application Example

This section outlines two applications of HAIL for using learned pedestrian street crossing behavior with a foresighted agent. AJAN is used to detect these situations during simulation and to send appropriate requests based on the configured agent model to InfoSalGAIL. The situations that the imitation learning model cannot mimic due to an unevenly balanced training dataset, e.g., navigating in an "unseen" street configurations, are handled by built-in actions in AJAN and OpenDS and meant to solve the OOD challenge of imitation learning.

4.1 Scenarios Description

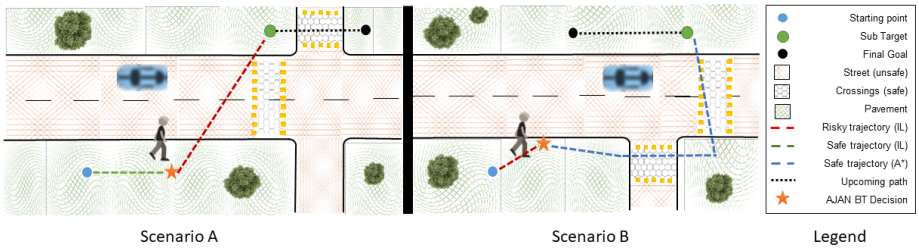


Fig. 3: Chosen use-case scenarios. Left: Scenario A - stands for the use-case shown to IL framework in the form of experts demonstrations. Right: Scenario B - out-of-distribution (OOD) scenario that only partially is known ("seen") to the IL framework. In scenario B, only part of the path (red trajectory) can be guided by the trained model, with the remaining path (blue trajectory) being realized by A* path planner.

Fig. 3 shows the traffic scenarios in which a pedestrian agent shall navigate from a given starting point to a destination like its human counterpart. In the considered scenarios A and B multiple strategies are possible, e.g., risky crossing of the road directly to the destination. Which strategy is finally chosen depends on different parameters like configured maximum time to reach the target, degree of risk aversion and interests. Strategies can also be dynamically switched in response to events, such as an approaching vehicle. In scenario A, e.g., the pedestrian agent initially walks safely because it has seen an approaching vehicle; if there is no longer any potential danger, the agent changes its strategy and crosses the

road directly and riskily to the intermediate destination and then walks on to its final location. In scenario B, however, the pedestrian first behaves riskily and then also safely. During the simulation, the pedestrian agent must dynamically decide whether to reach the respective destinations with InfoSalGAIL or A*.

4.2 Agent Model

HAIL enables to configure different pedestrian agents by setting their behavioral parameters like the available definitions of InfoSalGAIL imitation models, the degree of risk taking, and individual interests. These parameters are taken into account during execution of the pedestrian agent SBTs, hence affect its behavior in the scene and are set while agent initialization and updated in runtime.
The application SBT defines the processing of strategy or rout choices depending on the given pedestrian agent state. In each simulation step the agent receives input from OpenDS with which this SBT is executed. While in one sub-tree of the SBT incoming information is saved and the will to take risks is adapted, the simulated behavior is implemented in parallel in another sub-tree as follows. If a critical event occurs, then the SBT is aborted, otherwise, the first step of the iteration is to use an ASP SBT-node to create multiple weighted routes to the agent goal. Then, a strategy is selected based on the agent risk-taking and the rout costs. Further, it is checked whether an InfoSalGAIL model exists for the selected strategy, and if positive, this model is used next; otherwise, the path planning in OpenDS is used.
For planning weighted routes the agent parameters, the ASP SBT-node (see Sec. 3.3), the scene configuration graph, which is stored in the agent knowledge and transformed into ASP, and the rules shown in Fig. 4 are used. These rules are using the RDF/ASP scene configuration to plan possible strategies. The result after solving the ASP problem are 0 to n routes with their costs, which are available to the SBT as named graphs.

```
1   cost(0,0).
2   time(1..steps).
3   edge(I,O,C) :-  _t(E,"rdf:type","opends:Edge"), _t(E,"opends:out",O), _t(E,"cost",C), _t(I,"opends:risky",O), not risky.
4   number(C) :- C = #count{N : node(N)}.
5   1 { walktTo(G,T) : node(G) } 1 :- time(T).
6   at(G,T) :- walktTo(G,T), at(A,T-1), edge(A,G,_).
7   cost(T,C1+C2) :- walktTo(X,T), at(Y,T-1), edge(Y,X,C1), cost(T-1,C2), Y!=X.
8   cost(Max) :- Max = #max{C : cost(T,C)}.
9   :- walktTo(G,T), not edge(A,G,_), at(A,T-1).
10  :- not at(X,T), T = steps, goal(X).
```

Fig. 4: ASP navigation planning rules, generating 0 to n stable models each containing one weighted *walkTo*-action sequence. In line 3 RDF based edges are defined and possible edges are removed, if risk-taking is not considered. In 5 a search space is built and filtered by constraints in lines 9 and 10. In lines 7 and 8 costs are calculated.

Fig. 5 depicts several views of scenario A: (A) shows the scene configuration as a graph available to the agent in RDF. Red edges are marked as risky and green edges as safe. Based on this graph, a route is calculated via ASP in which interests (green node) are taken into account by decreasing the route costs. (B) is a partially risky route to an intermediate destination at time 0 of scenario A. Instead, (C) is a safe route of the same scenario and time. (D) displays the graphical representation of the semantic description of an available InfoSalGAIL

model. Both routs or strategies (B) and (C) are matching this model. InfoSalGAIL is used then, by the presented SBT for navigation. The same model can be used in scenario B, if the pedestrian wants to riskily reach the intermediate destination. If it is to be reached safely, the navigation planning in OpenDS must be used instead, if no matching imitation model is available. The choice of a strategy depends mainly on the agent's willingness to take risks. This depends on the current agent state and is based on its initial configuration and incoming events. If, as in scenarios A and B, such an event is received, the originally safely (scenario A) or risky (scenario B) strategy is discarded (star in Fig. 3) by AJAN and a new route is computed and executed by HAIL or A*.

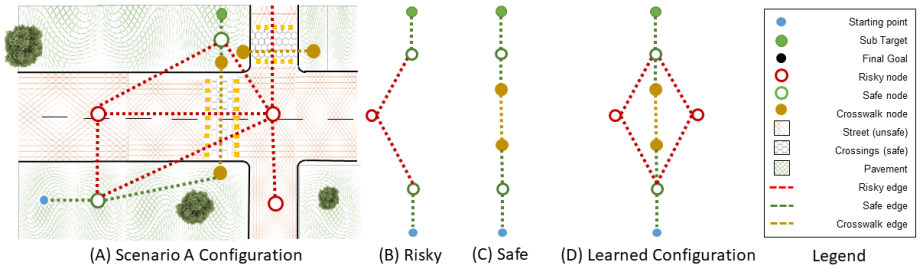


Fig. 5: (A) street configuration in OpenDS. (B) and (C) generated ASP plans for risky and safe behavior. (D) street definition for available imitation models.

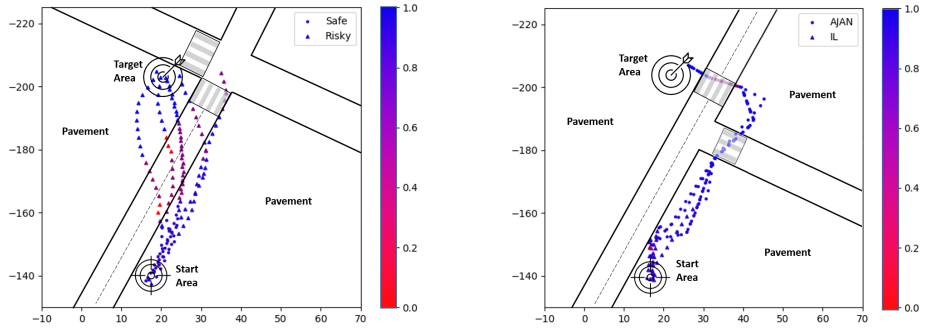
4.3 Imitation Model

For the application scenarios, we train imitation models given the training data (see [20]), e.g., pairs consisting of the FoV images and saliency maps, including memory information of six human subjects on nine different scenarios (∼140K pairs) in accordance to German-in-Depth-Accident-Study (GIDAS), excluding the safe part of the use-case scenario B (special OOD scenario). Each model represents a single street configuration (for which there is an RDF representation). Here, the risk-free and high-risk navigation was considered in a scene with a zebra crossing. As input signals, InfoSalGAIL accepts an RGB image of size *224x224x3* and prior information from frames $t-1$ and $t-2$, respectively. This information comes from OpenDS and is passed to InfoSalGAIL via AJAN with a scenario-dependent latent code (safe or risky) for a given trained model. The resulting *4d* action (turn, speed, yaw, pitch) is then passed back to OpenDS.

4.4 Experimental Evaluation

For training, validation and testing purposes with OpenDS and InfoSalGAIL, we utilized a Tesla V100 (32GB vRAM) GPU under ∼14GB of vRAM due to usage of a pre-trained model for the saliency generator provided in [20], and a 2018 MacBook with Windows 10 to control the pedestrian agent with AJAN. For the processing of incoming data from OpenDS, the forwarding of information required by InfoSalGAIL, and the passing of resulting 4d-actions, AJAN needs no more than ∼17ms. If a navigation plan needs to be created with the ASP-SBT node, no more than ∼94ms are additionally required for the application

scenarios. A detailed evaluation of ASP based action planning is presented in [3]. The generated pedestrian agent trajectories for the application scenarios A[14] and B are shown in Fig. 6.



(a) Safe to Risky behaviour switch (b) IL to AJAN (A*) switch

Fig. 6: (a): Plots of generated pedestrian agent trajectories of Scenario A, where the change from safe to risky behavior is triggered by AJAN depending on the road situation, e.g. potential dangerous collisions by an arriving car. (b): Plots of the trajectories of scenario B, where the agent is faced with the OOD problem (new street layout). Accordingly, navigation path is taken from A*-planning, where less variation is seen.

5 Conclusion

We presented a novel approach, named HAIL, for modular agent-based pedestrian imitation learning in traffic scenarios to generate human-like trajectories under the constraints of previously unseen traffic scenarios. HAIL combines the AJAN, OpenDS and InfoSalGAIL subsystems to realize the tactical and strategic as well as the operational level of pedestrian behavior, making HAIL suitable for the generation of critical traffic scenarios and tests. We presented an application example to show how HAIL virtually imitates real pedestrian trajectories on the one hand and how OOD scenarios are solved on the other hand. For this, ASP is used to decide whether an imitation model is available for partial execution of navigation plans and to dynamically create these plans.

Acknowledgements The work described in this paper has been funded by the German Federal Ministry of Education and Research (BMBF) through the project REACT (grant no. 01/W17003) and in part by Huawei Munich Research Center.

References

1. Abbeel, P., Ng, A.Y.: Apprenticeship learning via inverse reinforcement learning. In: Proc. 21st Int. Conf. Machine Learning (ICML) (2004)

[14] Videos of the FoV of the pedestrian agent in scenario A can be found at: https://cloud.dfki.de/owncloud/index.php/s/HAf5wQtMAx3F9K5

2. Antakli, A., Spieldenner, T., Rubinstein, D., Spieldenner, D., Herrmann, E., Sprenger, J., Zinnikus, I.: Agent-based web supported simulation of human-robot collaboration. In: Proc. Int. Conf. Web Information Systems and Technologies (WebIST) (2019)
3. Antakli, A., Zinnikus, I., Klusch, M.: Asp-driven bdi-planning agents in virtual 3d environments. In: Proc. 14th German Conference Multiagent System Technologies (MATES). Springer (2016)
4. Bloem, M., Bambos, N.: Infinite time horizon maximum causal entropy inverse reinforcement learning. In: Int. Conf. Decision and Control. IEEE (2014)
5. Brambilla, M., Facca, F.M.: Building semantic web portals with a model-driven design approach. In: Web Technologies: Concepts, Methodologies, Tools, and Applications, pp. 541–570. IGI Global (2010)
6. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: Proc. 29th Int. Conf. Neural Information Processing Systems (NeurIPS). pp. 2172–2180 (2016)
7. Colledanchise, M., Ögren, P.: Behavior trees in robotics and AI: An introduction. CRC Press (2018)
8. Ho, J., Ermon, S.: Generative adversarial imitation learning. In: Proc. 29th Int. Conf. Neural Information Processing Systems (NeurIPS) (2016)
9. Ianni, G., Martello, A., Panetta, C., Terracina, G.: Efficiently querying rdf (s) ontologies with answer set programming. Logic and Computation **19**(4) (2009)
10. Ishaque, M.M., Noland, R.B.: Behavioural issues in pedestrian speed choice and street crossing behaviour: a review. Transport Reviews **28**(1), 61–85 (2008)
11. Karamouzas, I., Heil, P., Van Beek, P., Overmars, M.H.: A predictive collision avoidance model for pedestrian simulation. In: International workshop on motion in games. pp. 41–52. Springer (2009)
12. Lee, J., Li, T., De Vos, M., Padget, J.: Using social institutions to guide virtual agent behaviour. In: The AAMAS Workshop on Cognitive Agents for Virtual Environments (CAVE-2013). Citeseer (2013)
13. Li, Y., Song, J., Ermon, S.: Infogail: Interpretable imitation learning from visual demonstrations. In: Proc. 30th Int. Conf. Neural Information Processing Systems (NeurIPS) (2017)
14. Panagiotidi, S., Nieves, J.C., Vazquez-Salceda, J.: A framework to model norm dynamics in answer set programming. In: Proc. Int. Workshop Multi–Agent Logics, Languages, and Organisations (MALLOW). vol. CEUR 494 (2009)
15. Paxton, C., Hundt, A., Jonathan, F., Guerin, K., Hager, G.D.: CoSTAR: Instructing collaborative robots with behavior trees and vision. In: Proc. IEEE Int. Conf. Robotics and Automation (ICRA). IEEE (2017)
16. Reynolds, C.W.: Steering behaviors for autonomous characters. In: Game developers conference. vol. 1999, pp. 763–782. Citeseer (1999)
17. Ross, S., Bagnell, D.: Efficient reductions for imitation learning. In: Proc. 13th Int. Conf. Artificial Intelligence and Statistics (2010)
18. Teknomo, K., Takeyama, Y., Inamura, H.: Review on microscopic pedestrian simulation model. arXiv preprint arXiv:1609.01808 (2016)
19. Vizzari, G., Crociani, L., Bandini, S.: An agent-based model for plausible wayfinding in pedestrian simulation. Engineering Applications of Artificial Intelligence **87**, 103241 (2020)
20. Vozniak, I., Klusch, M., Antakli, A., Müller, C.: Infosalgail: Visual attention-empowered imitation learning of pedestrians in critical traffic scenarios. In: Int. Conf. Neural Computation Theory and Application (NCTA). IEEE (2020)