[COR: Number]

Creating Virtual Reality Scenarios for Pedestrian Experiments Focusing on Social Interactions

Daniela Briola[1,*], Francesca Tinti[1] and Giuseppe Vizzari[1]

[1]*University of Milano Bicocca, Italy*

Abstract

Designing and running real world pedestrian experiments can be complex, costly, and it can even have ethical implications. Virtual Reality can represent an alternative enabling the execution of experiments in a virtual environments, with synthetic humans (i.e. agents) interacting with human subjects. To achieve a high degree of realism, such virtual humans should behave realistically from many points of view: in this paper, we focus on how they move inside the environment. We propose the design, and first prototype, of a new tool based on Unity, simplifying the setup of realistic scenarios for experiments in VR with humans. In particular, this tool lets the modeler integrate external pathfinding models so as to achieve realistic and believable scenarios for experiments with human subjects.

Keywords

Virtual Reality, Intelligent Agents, Pathfinding, Pedestrian Simulation, Experiments with human subjects

1. Introduction

The continuous evolution of technology has opened up new frontiers in the scientific research, enabling the exploration of complex phenomena through innovative approaches. Among these emerging technologies, one of the most fascinating ones is the Virtual Reality (VR). VR has garnered significant interest due to its ability to provide substantial benefits across a wide array of applications, including gaming, training, architectural design, social skills training, surgical procedures, and more. This technology offers the unique opportunity to replicate experiences that would otherwise be inaccessible in the real world, offering an unparalleled level of realism.

For example, one particularly promising application area for VR is the study of pedestrian dynamics in building [1, 2]. Interest in the study of human movement within buildings is motivated by the need of understanding how people interact with architectural spaces, as this has a significant impact on crucial aspects such as the design of environments, safety and efficiency of people flows. This field of research faces substantial challenges when conducted in real-life environments: the intricate nature of most pedestrian infrastructures, coupled with the inherent variability of human behavior in such settings, makes it difficult to control experimental scenarios and external factors. Additionally, conducting controlled field experiments to investigate pedestrian behavior in hazardous situations is often impeded by ethical considerations related to

WOA 2024: 25th Workshop "From Objects to Agents", July 8-10, 2024, Forte di Bard (AO), Italy

*Corresponding author.

daniela.briola@unimib.it (D. Briola); f.tinti@campus.unimib.it (F. Tinti); giuseppe.vizzari@unimib.it (G. Vizzari)

the mental and physical well-being of participants. Moreover, these field experiments typically require significant investments of labor and finances.

Virtual reality emerges as a promising tool in this context, making it possible to realistically simulate the architectural environment and accurately monitor participants' behavior during experiments, reducing costs and problems. In comparison to field experiments, VR provides the capability to realistically simulate architectural environments and meticulously monitor participants' behavior during experiments. This includes having comprehensive control over the experimental setup and collecting precise behavioral data concerning pedestrian movement and decision-making processes. Furthermore, VR allows participants to immerse themselves virtually in hazardous environments without facing actual physical risks.

However, to achieve an experiment in VR offering comparable experience with respect to a field one, many aspects need to be addressed in order to get high presence and immersion in the VR experience, as discussed in [3, 4, 5]: one crucial aspect of particular significance we are going to focus on is related to virtual humans in the VR environment. For VR to be effective in pedestrian studies, the virtual humans must move in a realistic manner. If they do not, participants may not perceive them as real humans, potentially invalidating the entire experiment. Achieving a proper level of realism cannot be obtained, for instance, by manually specifying the path that the virtual human has to follow without adapting it to the presence of the real human performing the experiment, or by exploiting standard algorithms such as A*, which finds the shortest path but does not accurately reflect the paths a real human would choose. Virtual humans need to be adaptive to the behaviors of each other and to the VR experiment subject, and they need to have a plausible behavior: this could be achieved by integrating a pedestrian simulator to guide the virtual humans [6].

The aim of this work is to support modelers and researchers who need to exploit VR to setup and run experiments with humans participants, focusing on studying the impact of social influence on wayfinding within a complex building or an open (but restricted) area, both in normal and emergency situations. More broadly, it aims to facilitate the study of human behavior in social scenarios. The desired system should support the setting up of the overall experiment, from the configuration of the environment to the programming of virtual humans. These virtual humans, which can be seen as partially autonomous agents, are needed to populate the environment and simulate pedestrians moving within it. It should also offer a way to import and exploit pedestrian (or more generally behavioural) models obtained with different techniques such as ML, to ensure that virtual humans move and act in a realistic way. This includes following trajectories, but also avoiding other agents while navigating the environment, and performing actions in a lifelike manner.

We are designing a project, based on Unity (a state of the art Game Engine largely adopted both in industry and in academy), which will include pre-configured virtual humans (avatars) with possible animations, pathfinding and social interaction behaviours, along with a support to distribute and orchestrate them within the environment. Additionally, the tool will automatically collect a wide set of data during the running of the experiment, facilitating the analysis of participant behavior, movement and other relevant metrics.

We present in this paper an high level description of this tool and discuss how (and why) to import already exiting pathfinding models can be the starting point to create realistic virtual humans in VR: to make a concrete example, we present a pedestrian model based on a curriculum

learning approach, already developed by our research group, and we briefly discuss how it could be integrated into the proposed tool.

The paper is organized as follow: Section 2 clarifies why VR can be beneficial for performing experiments with human subjects and which are the requirements for achieving a good experience, offering an overview of already existing pedestrian models that could be integrated. Section 3 describes the overall requirements we have for the tool we are envisioning and developing in Unity, while Section 4 describes what we have already achieved and Section 5 presents the pedestrian model we are integrating in the tool. Section 6 describes the open points and future parts we would like to integrate in our tool, and the conclusions.

2. VR for Experiments with Human subjects

As anticipated in the introduction, running an experiment with human subjects in a Virtual Reality scenario may represent a valid alternative to perform such experiment in reality, especially if many people are needed to set up scenarios, like for example in experiments with crowds or pedestrian studies. In such cases, apart from the involved subjects that will take part in the study and will be monitored and interviewed, many other humans are needed as actors in order to build a realistic environment (the scenario) for the experiment. Second, with respect to what the topic of the experiment is, having both the actors and the subject moving around the environment may require specific limitations to grant security, making the concrete running of the experiment often impossible in reality. An example could be studying how pedestrians move when immersed within a crowd in public places, such as a crosswalks, or during an evacuation in a metro station. Situations like these occur in real life but they are too challenging to study through field observations due to their complexity and potential danger. At the same time, they are also difficult to replicate accurately as it is challenging to elicit genuine and unfiltered reactions from participants. In such cases, VR can be exploited both to replicate the physical environment and to replace human actors with virtual humans (often referred to as avatars or virtual agents) [7, 8]. However, there are several requirements that must be met for the VR simulation to be effectively adopted.

First of all, an high degree of presence and immersion are needed: obtaining them is still an open issue in the VR research area, as it involves addressing numerous aspects, ranging from software to hardware challenges. To cite some of those more important in the over mentioned experiments for crowd and pedestrian studies, high fidelity models of environments are needed, with sounds and lights managed in a realistic way, as well as hardware that supports high level of immersion, such Head Mounted Displays (which isolate participants from the real world). Additionally, realistic avatars, both in appearance and movement, are crucial. This aspect is the focus of the proposed tool, as studying how a human subject navigates an environment populated with other humans requires realistic avatars, but also a "simple enough" way to orchestrate them during the simulation (they should move in the overall as a realistic crowd). Failing to achieve this realism in avatar behavior compromises the comparability of the resulting VR experience with a real-world one and the reliability of collected data.

For example, in recent collaborations we are studying how humans decide the path to follow in a multilevel building, and if their choices depend, and how, on the presence of other people

in the building. To set up such experiments, a realistic model of a multilevel building is needed, featuring alternative paths connecting levels and rooms to provide participants with varied options for navigation, and to study which ones they opt for. Additionally, the setup requires numerous avatars realistic in their aspects, capable of following different paths (controlled by the modeler in terms of points to be reached) and performing different animations. These animations may include moving, stopping to engage in conversation, and changing direction at predefined timestamps. Organizing and coordinating such avatars is complex, and the modeler cannot specify, for each avatar, every single step of its path. Instead, the modeler usually specifies only the "waypoints" (the necessary points that must be reached during the movement), and then the Game Engine is in charge of concretely computing the path to reach them [9]. However, if the computed path is not realistic (in the sense that it does not resemble a path a human would naturally follow), the resulting avatar's movement can be perceived as fake by the human subject, thus resulting in a loss of sense of presence in the VR simulation. To address this, we would like to use more realistic models of avatar movement, exploiting results from AI where models have been studied from years in order to achieve for example agents able to avoid each others, perceiving points of interest in the environments and so on.

2.1. Virtual Humans: Unity NavMesh versus Realistic pathfinding

Many VR experiments are conducted using Unity[1], a game engine widely used for creating interactive 3D games and experiences. Its intuitive interface and comprehensive documentation make it accessible to users with diverse technical backgrounds, which is one of the reasons why we chose it for our project. Furthermore, Unity's Asset Store[2] provides pre-made assets and tools, speeding up the development and enabling the creation of detailed, realistic settings with minimal effort. Among its many features, Unity provides a comprehensive system for developing characters capable of intelligently navigating the game world using navigation meshes. This system consists of a NavMesh and a NavMesh Agent. The NavMesh is a data structure that defines the walkable areas in the game world, facilitating pathfinding between different walkable locations. The walkable areas are automatically created from the scene's geometry by identifying points where the agent can stand and connecting these points to form a surface overlaying the scene's geometry. This surface, called the navigation mesh (NavMesh), is represented as a collection of convex polygons (an example can be seen in Figure 1b). The NavMesh Agent component is essential for developing characters that can navigate towards their objectives while avoiding collisions with other agents. These agents use the NavMesh to understand the game world and to maneuver around both static and dynamic obstacles. The NavMesh Agent component manages both the pathfinding and movement control of a character, ensuring smooth and intelligent navigation. To determine a path between two locations in Unity, the start and destination points are first mapped to their nearest polygons. Unity employs the A* algorithm to search through these polygons, generating a sequence known as a corridor. The agent navigates this corridor by steering towards the next visible corner. For individual agents, this method is straightforward, but for multiple agents, the need to avoid collisions can

[1] https://unity.com/
[2] https://assetstore.unity.com/

complicate navigation. Instead of following a static path, agents dynamically adjust their routes by continuously recalculating their corridor and steering towards the next corner.

While Unity's Navigation system is useful and already integrated, it has some limitations. When it comes to local avoidance (the ability of characters to navigate around each other without collisions), Unity's built-in system can struggle, particularly in crowded scenes. The method it uses (RVO), is not always up to the task of managing dense crowds smoothly. This can result in characters bumping into each other or moving in erratic, unrealistic ways, which can break the immersion. Additionally, the quality of the paths generated by Unity is not always perfect: paths might be jagged or inefficient, requiring developers to implement additional path-smoothing techniques. Without these adjustments, character movement can appear unnatural, as if they're taking a series of sharp turns rather than following a smooth, logical route.

Unity's system limitations generally originate from the use of the A* algorithm. While A* remains a popular algorithm due to its simplicity and effectiveness, it struggles in complex environments, prompting ongoing advancements in the field. Over the years, numerous variations and extensions of A* have been developed to enhance its efficiency [10], but multi-agent navigation presents new challenges. In these scenarios, agents must avoid not only static obstacles but also other moving agents. Traditional numerical tools for finding precise solutions are often expensive and time-consuming. This is where Reinforcement Learning (RL) has shown significant promise [11]. RL allows agents to make decisions based on a parameterized policy optimized through training data. RL algorithms can also be combined with traditional pathfinding methods to improve performance [12]. More recently, a study [13] explored an RL-based approach, defining a perception model that provides agents with information about nearby agents, obstacles, and the final goal. The action model regulates the agent's velocity vector, considering angle variations and acceleration/deceleration. Our model, briefly described later on, builds on this approach but employs a single training process based on a curriculum method.

3. Goals and design of the proposed tool

The tool we are designing and prototyping in Unity should support these functionalities:

1. Providing at least two ready-to-use realistic models, including one for multilevel environments.
2. Supporting the customization of available avatars (for example in the color of their clothing or skin), and offering a support in the definition of their path in the environment (as a sequence of points to be reached). This includes specifying animations at each point along the path.
3. Supporting a way to associate to each avatar a different model of movement (the one offered by Unity, the one presented in Section 5, other already existing models coherent with the Unity Avatar management).
4. Automatically tracking and storing subjects' movements data (in terms of position, rotation, timestamp), its head rotation (and eye tracking if the HMD supports it), and similar data.

5. Supporting the modeler in enabling avatars to interact with the environment and the objects inside it, allowing avatar actions to dynamically change the state of the environment.

4. Design, and partial implementation, of the Unity tool

Unity Setup for User Locomotion

In order to create realistic simulations and to enable users to navigate the environment, we implemented a combination of open-world navigation and a modified version of the usual steering locomotion [14]. The open-world approach allows participants to freely explore the environment, while steering locomotion lets users initiate continuous simulated self-motion toward their desired destination using a joystick. This system facilitates effective exploration and interaction within the virtual environment, enhancing the sense of presence and realism. However, continuous steering locomotion is known to induce cybersickness [15, 14]. To address this issue, we modified the continuous steering locomotion to a forward "snap" movement, which mimics the step-by-step motion of walking. This technique has been shown to minimize motion sickness while maximizing the sense of presence [16]. So, participants control their forward movement using the hand controller, while the direction of their movement is guided by their head rotations. The XR Rig element (the GameObject representing the user in the VR environment) has been identified as an obstacle, so that autonumous avatars will avoid it while moving, enhancing the presence in the simulation.

Requirement 1

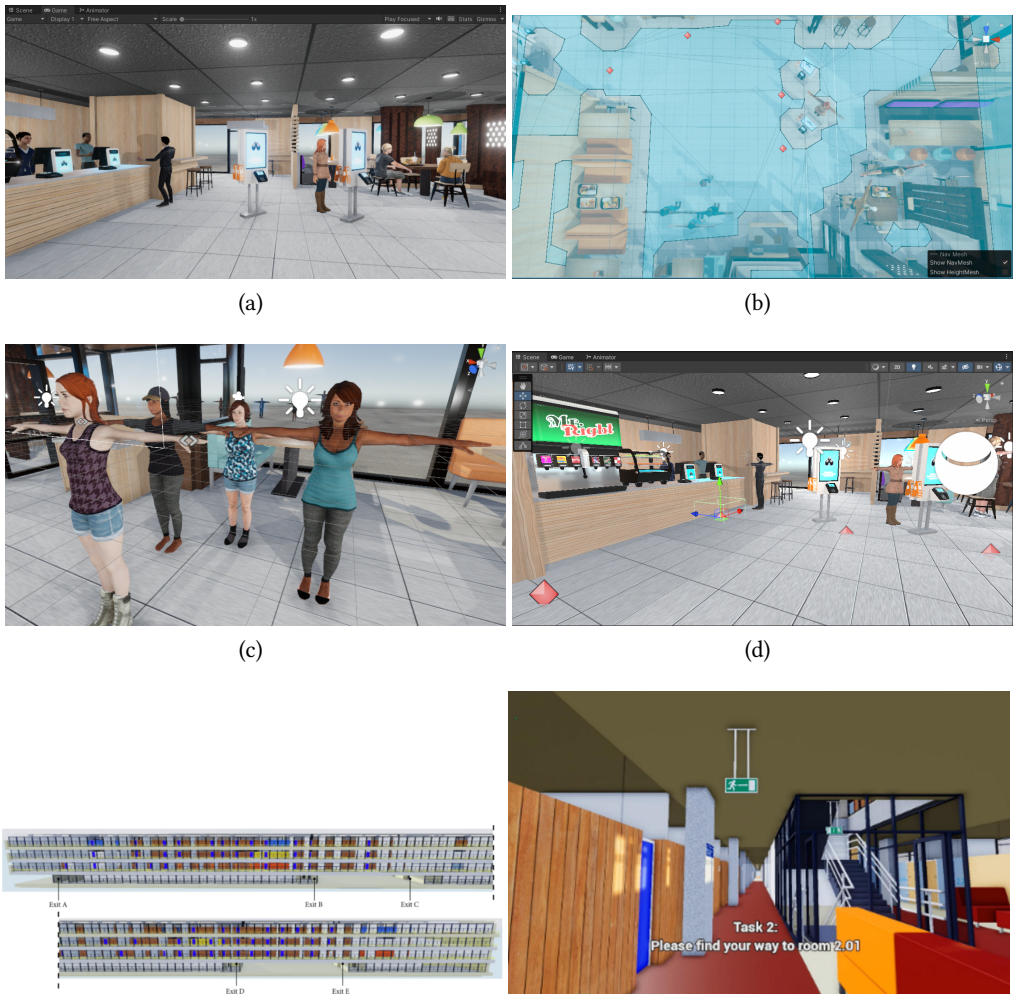
With respect to the requirements list reported in Section 3, our work focused on exploring the Asset Store in order to find a prefabricated environment offering realism and customization options. The “Fast Food Restaurant Kit”[3] caught our eye as it replicates a contemporary fast food (on a single floor): given the widespread familiarity with such environments, we believed it is the ideal setting, as most people understand the typical dynamics of a fast-food restaurant, such as ordering and waiting for food. Additionally, this environment allows us to create scenarios that can be reproduced in real life, which is crucial for setting up experiments with human subjects to study how they move in crowded environments. The package includes a sample scene that we modified to add more exits, enhancing navigation for participants and avatars (see Figure 1a). The scene is fully customizable and already features realistic textures and lighting. Additionally, the kit contains numerous prefabs, allowing for further adjustments and improvements. For what concerns the model of a multilevel building, we are still searching for a free model similar to the one proposed in [17, 18] (Figure 1e, 1f), that we already used in a tool similar to the one we are developing [19], but based on UnrealEngine[4].

Requirement 2

We provide two types of avatars in the tool, which can be easily added by the modeler to the scenario: background avatars to create a lifelike atmosphere, and navigating avatars programmed to follow specific paths (Figure 1c). To investigate social interactions between pedestrians, we added a NavMesh surface in our environment (Figure 1b), specifying the surfaces

[3] https://assetstore.unity.com/packages/3d/environments/fast-food-restaurant-kit-239419

[4] https://www.unrealengine.com/en-US/unreal-engine-5



(a) (b) (c) (d) (e) (f)

Figure 1: (a) A screenshot of the Fast Food model, in the background it is possible to see the added exit and (b) a (partial) view of the NavMesh surface used by the NavMesh agents, (c) avatar ready to be assigned a path, (d) waypoints (red diamonds) to be reached by an avatar. From [17, 18], (e) model of a multilevel building seen from the side (the model is truncated due to its elongated nature), and (f) view of the staircases of the multilevel building.

where agents can walk on, and added autonomous avatars in it. The avatars serve a dual purpose: investigating pedestrian dynamics and enhancing the environment's realism. We exploited two different free avatars packages: background avatars were sourced from Mixamo[5], which offers a wide variety of free models and animations, although they lack extensive customization options, while navigating avatars were obtained from the Unity Learn website's crowd simulation

[5]https://www.mixamo.com/

project[6], which provides customizable avatars along with walk, idle, and run animations, as well as a pre-made Animator Controller (see Figure 2b). The Unity Animator Controller manages a set of animation clips and their transitions, so in this case it's quite useful as it ensures that the transitions between animations are preconfigured and ready-to-use. This is especially convenient since integrating navigation with character animations to achieve smooth, natural movement can be complex. In fact it's not always easy to achieve a seamless blend between animated movements and pathfinding. The navigating avatars are ready-to-use and equipped with Unity's NavMesh Agent component that enables them to move around the environment using Unity's Navigation System. However, as previously mentioned, this system has its limitations, which is the reason we want to change it.

To enhance control over the avatars' paths and customize their actions, we developed a script that allows us to set waypoints directly from the Editor (Figure 2a). The waypoints are simple GameObjects, marked with a red icon for easy visualization in the Editor but invisible during runtime. These GameObjects can be placed anywhere within the walkable areas. In Unity, GameObjects are fundamental entities representing every object in the game, such as characters, objects, and scenery. While they don't perform any actions on their own, they serve as containers for Components that provide actual functionality. Our waypoints are basic empty GameObject with the sole purpose to designate target locations for the avatars (Figure 1d), and the script is designed to read the coordinates of these waypoints. This setup allows us to specify an array of locations from the Editor that the avatars need to reach in sequence (each avatar has its own waypoints list). This script also enables us to categorize avatars as either store customers or passersby. Store customers will stop at various locations within the store for a period of time customizable from the Editor, for example to look at kiosks with the menu or wait near the drinking machines, while passersby will simply walk within the environment from one waypoint to another, adding dynamic movement to the scene. This approach allows us to precisely manage the movement of all avatars, ensuring a realistic and easily controllable simulation environment. Please note that by now avatars are only able to run animations, but not to concretely interact with the environment (for example to order something from the kiosks: this is the next aspect we will face in future research (Requirement 5)).

As shown in Figure 2a, locations can be added or removed from the list of waypoints (called "nodes") in the Editor. To simplify the process, we created a Prefab[7] of the nodes: a Prefab allows to create, configure, and store a GameObject, complete with all its components, property values, and child GameObjects, as a reusable asset. Any edit made to a Prefab asset is automatically applied to all instances of that Prefab, enabling broad changes across the entire project without needing to edit each instance individually. The "Delay" value property and the "is Customer" boolean property are interconnected: if the boolean is set to True, the avatar is considered a customer, as previously mentioned, and will pause at each waypoint for the duration specified by the "Delay" value. If the boolean is set to False, the avatar will simulate a passerby and will navigate between waypoints without stopping. A subtype of the waypoint Prefab is the one used to destroy an avatar, which can be used if we want to make an avatar "disappear" from the scene (possibly to be positioned not in the sight of view of the human subject).

[6] https://learn.unity.com/project/crowd-simulation
[7] https://docs.unity3d.com/Manual/Prefabs.html

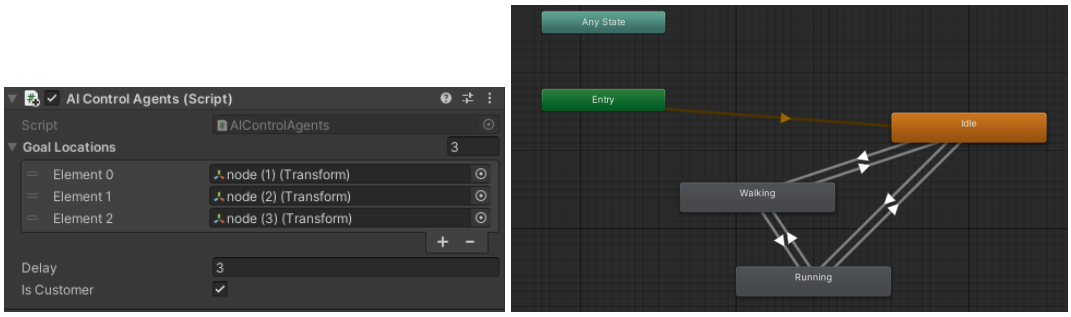
Requirement 3 We are still working on offering a simple way to import, and then use, models for pedestrian movement. In the next section we will briefly introduce an already existing pedestrian model developed by our research group, and we will report on the design for its integration into our tool, so that to offer an alternative algorithm to move the autonomous avatars.

Requirement 4 and 5 Requirement 4 is under development, so that to track and store a large set of data, and to directly produce output files to be analyzed with PedPy [20]. Requirement 5 instead requires a careful and deep research study, since we are studying how to integrate BDI Agent model [21] and A&A abstraction [22] into Unity, in order to get the autonomous avatar to really interact with the environment and be more "intelligent" and able to plan their overall behaviour during the simulation.

5. The integrated pathfinding algorithm

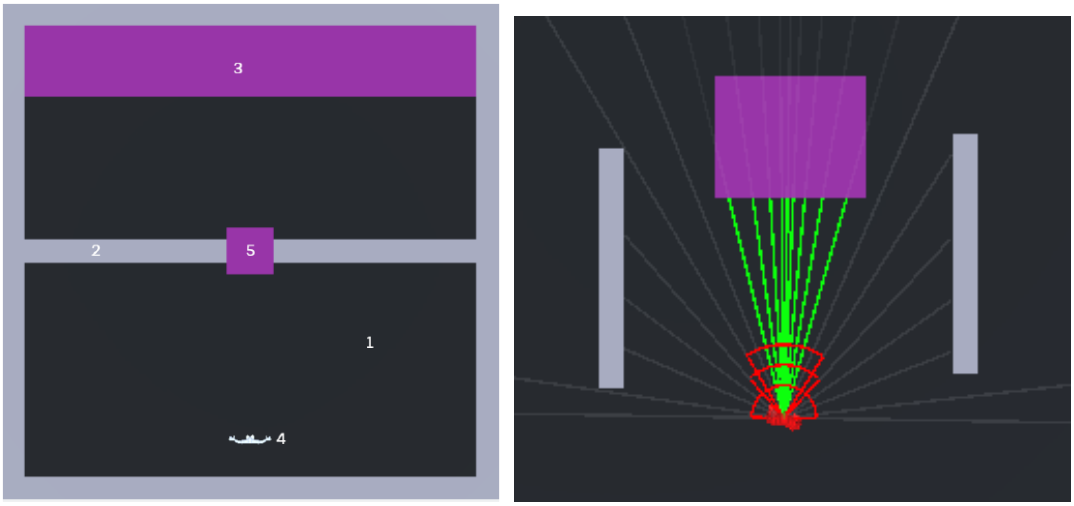
In line with recent research results aiming at exploiting Deep Reinforcement Learning (DRL) techniques [23] for achieving a sufficiently *general* behavioural model for a pedestrian agent positioned and moving in an environment, our research group recently proposed a contribution employing a curriculum [24] based approach that, together with a carefully designed reward function, allowed us to exploit expertise on the simulated pedestrian phenomenon and to achieve a behavioural model for a pedestrian agent showing promising results. First results [25] and a more recent tune (to appear) of this approach showed the practical possibility to achieve plausible results in terms of ability of avoiding other agents and in finding a suitable path from a starting point to a destination one, crossing intermediate points too.

The system used to generate this model is based on Unity: the scenarios, agents and their perceptive capabilities, as well as the monitoring components for acquiring data and to model all what is needed to train and then test the model, are implemented as Unity Prefabs and C# scripts. Unity does not directly include components supporting DRL techniques, but the



(a) (b)

Figure 2: (a)The script component used to add waypoints to the path of an agent, specifying if the agent is a costumer or not and the time he has to wait before moving to the next waypoint and (b) the animator controller with the walk, idle and run animations



(a) Environment elements (b) Rays of perception

Figure 3: (a) Environment elements: walkable area (1), walls (2), final target (3), agent (4), intermediate target (5); (b) rays of agent perception.

ML-Agents toolkit[8] provides both an extension of the Unity environment as well as a set of Python components enabling training and using DRL based agents.

So, we decided to use this pedestrian model as a first attempt to import and use it in our new Unity based tool. In this section, we report some general information regarding the model and some results to let the user understand the differences between its usage and those achieved with a standard NavMesh Unity agent: later on we will discuss how we are proceeding regarding its integration into our tool.

5.1. Reinforcement Learning Pedestrian Model

To generate the model we adopted environments of 20 × 20 metres surrounded by walls, with different internal structures. Walls and obstacles are represented in gray, while violet rectangles are intermediate and final targets not preventing the possibility of moving through them, but they are perceivable by agents such as gateways, mid-sections of bends, exits, and they support agent's navigation of the environment. The modeler must therefore perform an annotation of the environment before using it, as showed for example in Figure 3(a).

Agents perceive the environments by means of a set of projectors generating *rays* extending up to 14 m (in the current setup) that provide indications on what is hit and its distance from the agent (Figure 3(b)). The agent is also provided with cones in which it can detect the presence of walls and other agents, for supporting close range avoidance behaviours. The regulation of the velocity vector related to agent's movement (magnitude and direction of walking speed) is the only action managed by the action model. In line with the literature [26], agents take

[8] https://github.com/Unity-Technologies/ml-agents

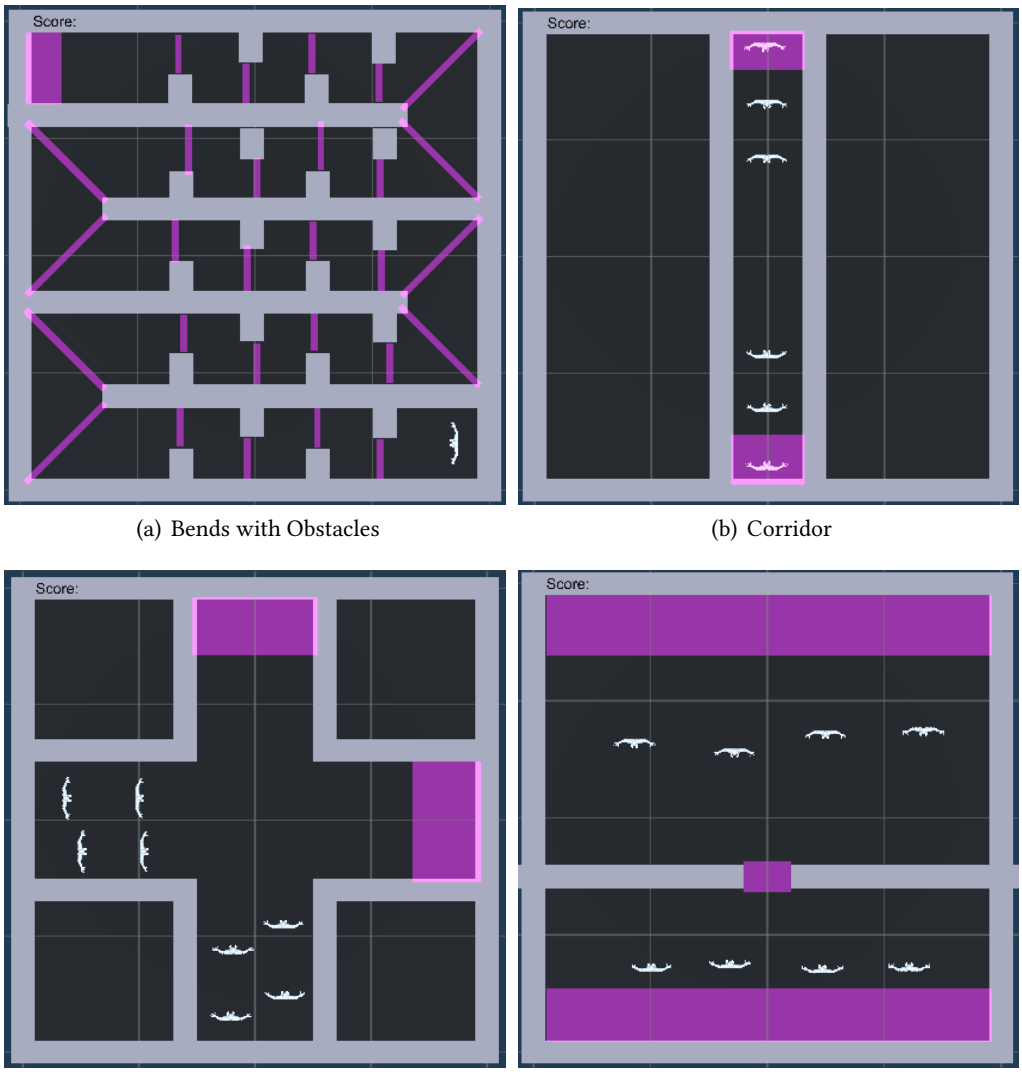
three decisions per second. Each agent is provided with an individual desired velocity sp_{des} that is drawn from a normal distribution with average of 1.5 m/s and a standard deviation of 0.2 m/s. Agent's action space has been therefore defined as the choice of two continuous values in the [-1,1] interval that are used to determine a change in velocity vector, respectively for magnitude and direction. The first element causes a change in the walking speed, while the second element of the decision determines a change in the agent's direction.

The cumulative reward we handcrafted increases only upon reaching the final target or a valid intermediate target (one that leads towards the final target, but reached only once). Other actions (associated to an implausible choice or simply to the fact that another decision turn has passed without reaching the final target) will instead yield a negative reward. Reaching the end of a training phase without reaching the final and intermediate targets will lead to a substantial negative reward.

We exploited the Proximal Policy Optimization (PPO) [27], a state–of–the–art RL policy–based algorithm, as implemented by ML-Agents.

Finally, we adopted a Curriculum Learning approach to generate our model, which is a particular form of supervised ML where examples increasing in difficulty are presented during the training, illustrating gradually more complicated situations to the learning algorithm. This approach is foreseen to speed up the learning phase and to get a model usually more prone to generalize to unknown environments. The used curriculum starts with very simple environments where the agent learns how to steer to look for the final target and walks towards it with just perimetral walls, then it has to face situations in which the environment is narrow (a basic corridor) and in which bends are present. Then social interaction is introduced, first of all with agents with compatible directions, then with conflicting ones, in geometries presenting bends and even bottlenecks. A selection of training environments is shown in Figure 4.

To let the reader understand the difference in the achieved pedestrian behaviours obtained with our model with respect to the standard algorithm offered by Unity with its NavMesh agents, we show in Figure 5 the resulting paths and velocity of an agent with NavMesh (Figure 5(a) and 5(c)) and one equipped with our model (Figure 5(b) and 5(d), in two different environments, one for each row): the agent exploiting A* clearly follows the shortest path, resulting in going really near to corners, turning as narrow as possible and walking straight forward between the intermediate targets, while the other agent follows the same overall paths but in a more "human" way, that is, adjusting sometimes its direction and not going so tangent to walls and corners. Also, please note that NavMesh agents have the complete view and knowledge of the environment (including the intermediate targets), so their path is computed with this information, and changes are made only if dynamic obstacles (for example other walking agents) appear at runtime: instead, agents exploiting ML-Agent library and our model are limited in their sight (as said before, they see only up to 14 meters and in a limited field of view), so they recalculate their path with respect to what they can see at runtime. This is evident for example comparing Figures 5(a) and 5(b), where the NavMesh agent already knows that the intermediate target is around the corner and directly orientates in that direction, while the other agent, as a normal human, will start exploring the environment (going forward) until it can see the intermediate target: only at that point it will turn towards it. This behaviour is more realistic, and is more similar to something we would like to exploit in a VR simulation, where the user should "confuse" an autonomous avatar with a human person.



(a) Bends with Obstacles (b) Corridor

(c) Intersection (d) Bidirectional Door

Figure 4: A selection of training environments.

5.2. Integration

The integration of the pedestrian model is still under work: the ML-Agent library saves the achieved models in a format that can be directly used within Unity without the need to have the ML-Agent trainer running (or even installed locally in the machine running the specific Unity instance). Nevertheless, when running, avatars exploiting these policies need to perceive the environment as those used for the training, so that to provide input for the model in the correct way. So, we are studying if it is simpler to import ML-Agents (in the sense of their body structure, rays, components and so on) in our tool, and modifying them in order to add

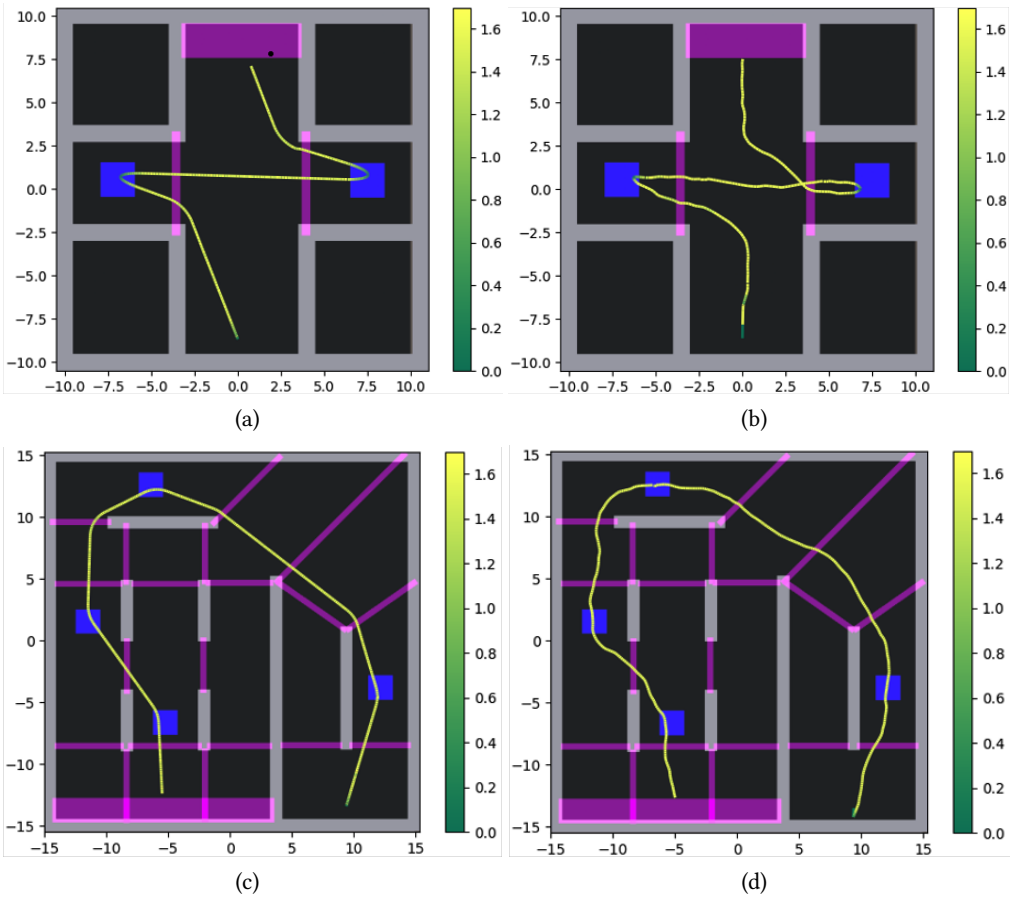


Figure 5: Comparison between the paths of a Navmesh agents (a) and (c) and those followed by agents moved with the model described in Section 5 (b) and (d)

the animator controller and the scripts to manage the list of waypoints to be reached, or vice versa, that is, enhancing the already exploited avatars to make them able to interface with the pedestrian model.

Furthermore, we will surely have to modify the elements in the environment to tag them as final or intermediate targets, or as walls and obstacles (as described in Section 5.1 and depicted in Figure 3(a)): this can be long and tedious, since we are now in a realistic and complex environment, with a lot of entities all along the scenario (tables, chairs, kiosks and so on), so we will help the modeler with an automatic script to tag all these elements as "obstacles", leaving to him only the burden to manually tag the intermediate and final targets.

6. Conclusion

We presented an high level description of the requirements for a new VR tool we are implementing, based on Unity, foreseen to support modelers in the creation of scenarios with

virtual humans moving in a realistic way, discussing how (and why it would be of paramount importance) to import already exiting pathfinding models.

Currently we are working on the integration of avatars exploiting the model described in Section 5, which should be concluded in the next month, as on the implementation of functions to collect real time data (subject movements, head rotation, avatars trajectories and so on).

A more complex study is going on regarding how to make the autonomous avatars more intelligent, in the sense of making them guided by internal goals (in a BDI fashion) and not simply by a list of waypoints to be followed, and more interactive with the environment, so that when the modeler is creating a new autonomous avatar specifying its goals, these can be then translated into concrete actions over the environment: we are studying how to exploit something related to the Agent and Artifact paradigm, since the Unity environment and interacting objects should be made available, and known, to the avatars to make them able to act over them. The general and long term idea is to create a VR tool able to support the management of the avatars (in their movements, but not necessarily only in this aspect) thanks to the integration of both Machine Learning approaches and more symbolic and logic ones, to let the modeler exploit approaches, techniques and tools coming from both the research areas, which can provide different, but we think complementary, benefits.

Second, since we are also researching on more psychological aspects of human interaction, in particular how proxemics impacts the way humans move in an environment and approach other humans (that can be autonomous avatars too) [28, 29, 30], we would like to offer in our tool another type of autonomous avatar (like we are doing for avatars guided by the pedestrian model discussed in this paper) that should in this case be guided by proxemics factors: this would open to further studies with humans, maybe focusing more on social interactions rather than on pathfinding and pedestrian movements, but still under the same overall goal of studying how humans interact and move in an environment populated with other people, exploiting VR.

References

[1] M. Fu, R. Liu, Y. Zhang, Do people follow neighbors? An immersive virtual reality experimental study of social influence on individual risky decisions during evacuations, Automation in Construction 126 (2021) 103644. URL: https://www.sciencedirect.com/science/article/pii/S0926580521000959. doi:10.1016/j.autcon.2021.103644.

[2] J. Lin, L. Cao, N. Li, Assessing the influence of repeated exposures and mental stress on human wayfinding performance in indoor environments using virtual reality technology, Advanced Engineering Informatics, Volume 39 (2019). URL: https://www.sciencedirect.com/science/article/pii/S1474034618304014.

[3] M. I. Berkman, E. Akan, Presence and immersion in virtual reality, Lee, N. (eds) Encyclopedia of Computer Graphics and Games. Springer, Cham (2024). URL: https://link.springer.com/referenceworkentry/10.1007/978-3-031-23161-2_162.

[4] J. Mütterlein, The three pillars of virtual reality? investigating the roles of immersion, presence, and interactivity (2018).

[5] J. Jerald, The VR book: Human-centered design for virtual reality, Morgan & Claypool, 2015.

[6] S. Deb, D. W. Carruth, R. Sween, L. Strawderman, T. M. Garrison, Efficacy of virtual reality in pedestrian safety research, Applied ergonomics 65 (2017) 449–460.

[7] M. Nelson, A. Koilias, S. Gubbi, C. Mousas, Within a virtual crowd: Exploring human movement behavior during immersive virtual crowd interaction (2019). URL: https://doi.org/10.1145/3359997.3365709. doi:10.1145/3359997.3365709.

[8] J. Lin, R. Zhu, N. Li, B. Becerik-Gerber, Do people follow the crowd in building emergency evacuation? a cross-cultural immersive virtual reality-based study, Advanced Engineering Informatics, Volume 43 (2020). URL: https://www.sciencedirect.com/science/article/pii/S1474034620300094.

[9] J. Lin, L. Cao, N. Li, How the completeness of spatial knowledge influences the evacuation behavior of passengers in metro stations: A vr-based experimental study, Automation in Construction, Volume 113 (2020). URL: https://www.sciencedirect.com/science/article/pii/S0926580519312452.

[10] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, E. Gunawan, A systematic literature review of a* pathfinding, Procedia Computer Science, Volume 179 (2021). URL: https://www.sciencedirect.com/science/article/pii/S1877050921000399.

[11] R. Junges, F. Klügl, Programming agent behavior by learning in simulation models, Applied Artificial Intelligence, 26(4), 349–375 (2012). URL: https://www.tandfonline.com/doi/full/10.1080/08839514.2012.652906.

[12] H. Qiu, Multi-agent navigation based on deep reinforcement learning and traditional pathfinding algorithm (2020). arXiv:2012.09134.

[13] F. Martinez-Gil, M. Lozano, F. Fernández, Emergent behaviors and scalability for multi-agent reinforcement learning-based pedestrian models, Simulation Modelling Practice and Theory, Volume 74 (2017). URL: https://www.sciencedirect.com/science/article/pii/S1569190X17300503.

[14] J. Clifton, S. Palmisano, Effects of steering locomotion and teleporting on cybersickness and presence in hmd-based virtual reality, Virtual Reality 24, 453–468 (2020). URL: https://link.springer.com/article/10.1007/s10055-019-00407-8.

[15] C. G. Christou, P. Aristidou, Steering versus teleport locomotion for head mounted displays, Augmented Reality, Virtual Reality, and Computer Graphics (2017). URL: https://link.springer.com/chapter/10.1007/978-3-319-60928-7_37.

[16] M. P. Jacob Habgood, D. Moore, D. Wilson, S. Alapont, Rapid, continuous movement between nodes as an accessible virtual reality locomotion technique (2018) 371–378. URL: https://ieeexplore.ieee.org/abstract/document/8446130. doi:10.1109/VR.2018.8446130.

[17] Y. Feng, D. C. Duives, S. P. Hoogendoorn, Using virtual reality to study pedestrian exit choice behaviour during evacuations, Safety Science 137 (2021) 105158. URL: https://www.sciencedirect.com/science/article/pii/S0925753521000011. doi:https://doi.org/10.1016/j.ssci.2021.105158.

[18] Y. Feng, D. C. Duives, S. P. Hoogendoorn, Development and evaluation of a vr research tool to study wayfinding behaviour in a multi-story building, Safety Science 147 (2022) 105573. URL: https://www.sciencedirect.com/science/article/pii/S092575352100415X. doi:https://doi.org/10.1016/j.ssci.2021.105573.

[19] Y. Feng, D. C. Duives, S. P. Hoogendoorn, Development and evaluation of a vr research

tool to study wayfinding behaviour in a multi-story building, Safety science 147 (2022) 105573.

[20] T. Schrödter, T. P. D. Team, Pedpy - pedestrian trajectory analyzer, 2024. URL: https://doi.org/10.5281/zenodo.10814490. doi:10.5281/zenodo.10814490.

[21] A. S. Rao, M. P. George, Bdi agents: from theory to practice, 1995. URL: https://api.semanticscholar.org/CorpusID:269838374.

[22] A. Omicini, A. Ricci, M. Viroli, Artifacts in the a&a meta-model for multi-agent systems, Auton. Agents Multi Agent Syst. 17 (2008) 432–456. URL: https://doi.org/10.1007/s10458-008-9053-x. doi:10.1007/S10458-008-9053-X.

[23] R. S. Sutton, A. G. Barto, Reinforcement Learning, an Introduction (Second Edition), MIT Press, 2018. ISSN: 01406736.

[24] Y. Bengio, J. Louradour, R. Collobert, J. Weston, Curriculum learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, ICML '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 41–48. URL: https://doi.org/10.1145/1553374.1553380. doi:10.1145/1553374.1553380.

[25] G. Vizzari, T. Cecconello, Pedestrian simulation with reinforcement learning: A curriculum-based approach, Future Internet 15 (2023). URL: https://www.mdpi.com/1999-5903/15/1/12. doi:10.3390/fi15010012.

[26] S. Paris, S. Donikian, Activity-Driven Populace: A Cognitive Approach to Crowd Simulation, IEEE Computer Graphics and Applications 29 (2009) 34–43.

[27] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms, CoRR abs/1707.06347 (2017). URL: http://arxiv.org/abs/1707.06347. arXiv:1707.06347.

[28] S. Bandini, D. Briola, A. Dennunzio, F. Gasparini, M. Giltri, G. Vizzari, Distance-based affective states in cellular automata pedestrian simulation, Nat. Comput. 23 (2024) 71–83. URL: https://doi.org/10.1007/s11047-023-09957-y. doi:10.1007/S11047-023-09957-Y.

[29] S. Bandini, D. Briola, A. Dennunzio, F. Gasparini, M. Giltri, G. Vizzari, Integrating the implications of distance-based affective states in cellular automata pedestrian simulation, in: B. Chopard, S. Bandini, A. Dennunzio, M. A. Haddad (Eds.), Cellular Automata - 15th International Conference on Cellular Automata for Research and Industry, ACRI 2022, Geneva, Switzerland, September 12-15, 2022, Proceedings, volume 13402 of *Lecture Notes in Computer Science*, Springer, 2022, pp. 259–270. URL: https://doi.org/10.1007/978-3-031-14926-9_23. doi:10.1007/978-3-031-14926-9_23.

[30] M. Giltri, S. Bandini, F. Gasparini, D. Briola, Furthering an agent-based modeling approach introducing affective states based on real data, in: A. L. C. Bazzan, I. Dusparic, M. Lujak, G. Vizzari (Eds.), Twelfth International Workshop on Agents in Traffic and Transportation co-located with the the 31st International Joint Conference on Artificial Intelligence and the 25th European Conference on Artificial Intelligence (IJCAI-ECAI 2022), Vienna, Austria, July 25, 2022, volume 3173 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2022, pp. 124–136. URL: https://ceur-ws.org/Vol-3173/9.pdf.