DOI: 10.1111/cgf.14631
ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2022
D. L. Michels and S. Pirk
(Guest Editors)
Volume 41 (2022), Number 8

Cognitive Model of Agent Exploration with Vision and Signage Understanding

Colin Johnson[1] and Brandon Haworth[2]

University of Victoria, Canada
[1]colin@colinjohnson.net
[2]bhaworth@uvic.ca

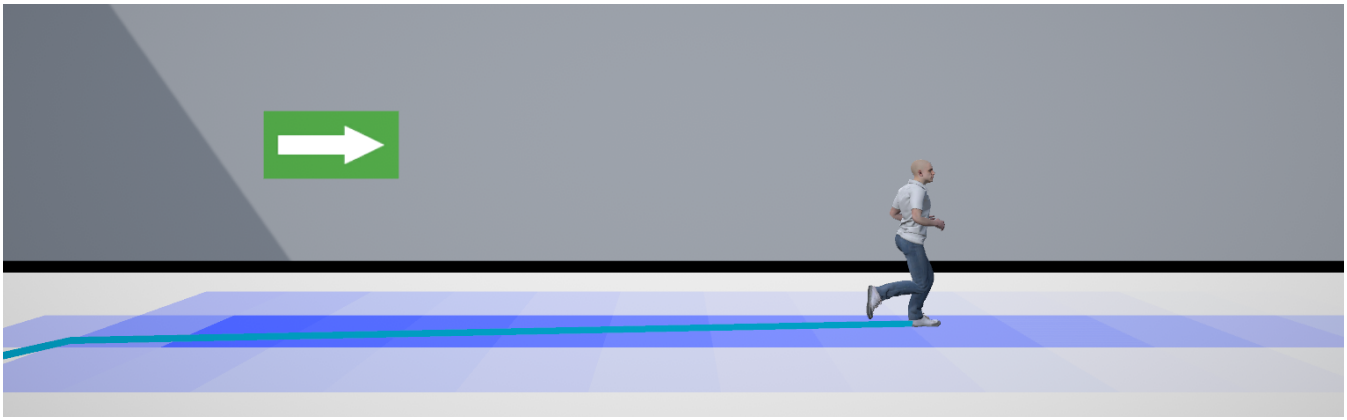

Figure 1: *An agent navigates based on information communicated by a directional arrow sign. The information is encoded and visualized on a spatial grid which planning and steering interact with.*

Abstract

Signage systems play an essential role in ensuring safe, stress-free, and efficient navigation for the occupants of indoor spaces. Crowd simulations with sufficiently realistic virtual humans provide a convenient and cost-effective approach to evaluating and optimizing signage systems. In this work, we develop an agent model which makes use of image processing on parametric saliency maps to visually identify signage and distractions in the agent's field of view. Information from identified signs is incorporated into a grid-based representation of wayfinding familiarity, which is used to guide informed exploration of the agent's environment using a modified A algorithm. In areas with low wayfinding familiarity, the agent follows a random exploration behaviour based on sampling a grid of previously observed locations for heuristic values based on space syntax isovist measures. The resulting agent design is evaluated in a variety of test environments and found to be able to reliably navigate towards a goal location using a combination of signage and random exploration.*

CCS Concepts

• ***Computing methodologies*** → *Multi-agent systems; Image manipulation;* • ***Applied computing*** → *Computer-aided design;*

1. Introduction

A signage system is an information system which makes use of visual cues to assist in navigation through a built environment. Signage systems provide invaluable through navigational aids such as "you are here" maps, directional arrows, and exit signs, which guide occupants along optimal paths in unfamiliar environments. While it is generally preferable to optimize the actual layout or floor plan of the environment, the application of signage is an effective post-hoc solution to ease or eliminate the confusion commonly experienced during wayfinding tasks. In addition to reducing the stress of navigation, effectively designed signage systems can optimize the flow

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

of pedestrians, and greatly improve safety during emergency egress scenarios. Similarly, visually striking or landmark features such are art installations, statues, lighting etc, impact the navigation of humans in complex spaces.

Due to this importance in the functioning of built, or real environments, there would be an expectation that virtual human modelling would have rich models for information gathering and exploration via signage and visual information sources. However, there is little work in this area at the intersection of visible environment information, agent modelling, and and information driven agent behaviour. The majority of the literature in crowd simulation focuses on global goal information assumptions and long term path planning. Global goal information assumptions that allow for long-term path planning may take into account area costs, route complexity, smart route choices, or even occupation density. However, there are few if any methods that view the world from the agent's perspective and allow the modelling of complex interactions between signage information, the distribution of that information in the environment, and exploratory navigation driven by that information.

Here, crowd simulation provides a compelling solution. Crowd simulation, agent-based crowd simulation, is a valuable tool in numerous industries and research areas that allows a practitioner or researcher to model large groups of autonomous entities interacting in a shared environment. From an animation perspective having smart agents improves the ability of animators to generate high quality scenes of interacting characters. This reduces costs and potential dangers inherent in the filming of large-scale live action scenes and reduces overhead and costs in the animation of large-scale animated scenes.

In this paper, we propose a novel agent-based, vision-based, crowds method for smart agents that interact not only with each other but more intelligently with their environment. We take a vision based approach, allowing each agent to actually see the environment and base their gaze movement on the value, or saliency, of objects in the visual field including signage and distractions. This visual field information allows our proposed agents to find and seek visual information for navigation and directly interact with the visual catchment areas of signs, art, distractions, etc. We propose an information map, inspired by recent advancements in human-like navigation, that encodes the distribution of information gained from several classes of signage. Finally, we propose a two stage navigation algorithm: a modified A* algorithm that accounts for the information value distributed in the environment and an exploration behaviour that accounts for spatial features by encoding their value in the form of SpaceSyntax measure on a pre-computed visibility graph. We show our method is capable of producing compelling behaviours automatically and that the parametric model affords deep authoring capabilities.

2. Related Work

2.1. Salient Objects

A first step to building a vision-based agent is deciding not just what in the scene is visible, but also what is interesting enough to be noticed by this agent. The inherent visual noticeability of an object is referred to as its saliency. One option for identifying salient areas and objects in an image is to use machine-learning-based methods like SALICON [JHDZ15], which have been trained on real-world image data. When such approaches are too slow for real-time multi-agent simulations, or fail to generalize well to simulated environments, an alternative is to use *parametric saliency maps* [KCH*21]. These are a recently proposed method of Modelling visual attention for autonomous agents. A parametric saliency map (PSM) is a 2D grayscale texture generated by applying a custom shader to the agent's camera. The fragment shader outputs lighter pixels in areas corresponding to more salient areas of the scene. This approach is fast and can produce similar results to those obtained from machine learning based methods, when sufficient metadata is available.

$$S' = W(S_d w_d + S_F w_F + S_v w_v + S_R w_R + S_I w_I)(S_M w_M)(S_A w_A) \quad (1)$$

The calculation, shown in equation 1, uses a weighted sum of various factors including speed v, angular speed R, interestingness I, depth d orientation F semantic masking M and visual attention A. The resulting value S' is the lightness of the resulting pixel, representing saliency.

2.1.1. Legibility of Signage

The fact that an agent can see a sign does not necessarily mean they can read it. Both user studies and information-theory-based approaches [DKT*17] have found that the area in which a sign is legible forms an approximate circle tangent to the surface of the sign and called the visual catchment area (VCA).

One approach to considering the legibility of signage during navigation is to take an information-theoretic approach [DTK*19]. Dubey et al. use information theory to quantify wayfinding information by considering the entropy associated with a sign. In their model, agents proceed towards a sign until this entropy is reduced past a set threshold, after which the agent is likely enough to have gained information, and receives a directional vector indicating to where it should proceed.

An alternative method is to use a binary check for signage visibility i.e. the sign is either visible, or it is not. To determine whether an agent is within the VCA of a sign, [XFG*07] considers the distance as well as the angular separation between the agent and the sign; the result is a circle given by equation 2. If an agent is within this circle and directly facing the sign, then the sign should be legible. In this equation, b is half of the size of the minimum recognizable element, and θ is the angular separation between the agent and the sign. The definition of the minimum recognizable element depends on the type of signage being modelled, but it is generally decided based on the size of the smallest text or pictogram on the sign as [XFG*07]:

$$\left(\frac{b}{sin(\theta)}\right)^2 = x^2 + \left(y - \frac{b}{tan(\theta)}\right)^2 \quad (2)$$

2.2. Space Syntax

When no other information is available, human wayfinding is based, in part, on the spatial configuration of an environment as

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

visible from the agent's isovist. An agent's isovist is the volume of space visible from their current position. Space syntax refers to a set of theories deriving metrics based on the intervisibility of locations in an environment; these include neighborhood size, which measures the number of vertices visible from a given location, and clustering coefficient, which measures the proportion of vertices which are connected versus the number that could possibly be connected [Tur01]. Past research has shown that uninformed exploration can be modelled surprisingly well using these metrics, even when compared to other more complex decision making strategies [PT02]. Furthermore, isovist measures of visual connectivity have been shown to be statistically significant in predicting route choice probabilities for users in a virtual reality experiment, as well as being effective in optimizing the configuration of environments to guide occupant route choices [DMS*22]. By making decisions using geometrical measures which could be reasonably estimated by the agent using information available in current or past isovists, we can avoid basing exploration on unrealistic global information. Additionally, our method proposes more than one information layer and integrates more than one visibility layer (visual perception with a max distance, and visual catchment areas) as well to avoid limitations of visibility graphs and isovists [SVF21].

2.3. Wayfinding Familiarity

Exploration is not always based solely on the configuration of the environment, as measured by space syntax. In many cases there is additional information available either through navigational aids like signage, or in a cognitive map developed through past experience navigating the environment. This additional information can be thought of as wayfinding familiarity, which varies throughout an environment. Higher familiarity results in an increased ability to make optimal decisions to reach the goal.

One approach to modelling this familiarity involves subdividing the environment into a hexagonal grid and assigning integer counts $C_{x,z}$ to each cell, such that larger values indicate greater wayfinding familiarity [RP22]. The hexagonal grid provides equal distance measurements for long-term planning, even in the diagonal. The agent is then guided by the path produced by an A* search with a heuristic function $h(\mathbf{p}_{x,z})$, given in equation 3, that assigns higher priority to cells with values closer to the maximum count C_{max}. When the agent has no familiarity with the current location, i.e. it is standing in a hex with a counter value of zero, it proceeds by randomly exploring the environment.

$$h(\mathbf{p}_{x,z}) = |\mathbf{p}_{x,z} - \mathbf{p}_{goal}| + \lambda_{x,z} \tag{3}$$

$$\lambda_{x,z} = 2(C_{max}), \quad when\ c_{x,z} = 0 \tag{4}$$

$$\lambda_{x,z} = \frac{C_{max}}{C_{x,z}}, \quad when\ c_{x,z} > 0 \tag{5}$$

2.4. Existing Crowds Solutions

A variety of end-to-end solutions for the generation, optimization, validation, and simulation of building layouts, crowds, and signage systems already exist. *AUTOSIGN* is a tool for signage system optimization, which includes agent-based validation of signage layouts [DKM*20]. Spatial Human Accessibility graph for Planning and Environment Analysis (*SHAPE*) is a path-based simulation framework that takes a physiological approach to evaluating wayfinding in 3D multilevel environments [Sch21]. *BuildingEXODUS* is software which focuses on the simulation of large-scale emergency egress situations; it includes signage as one of an extensive set of parameters [RNG12]. Existing solutions for the application of crowd simulation to studying wayfinding using both signage and uninformed exploration frequently include several limitations. Some models overly simplify the representation of the agent or its cognitive model of the environment. Others consider vision in only a basic sense, such as by basing visibility checks on simple ray casts [DTK*19]. Approaches using only ray casts fail to model the noticeability and legibility of signage. The use of global shortest-path routing algorithms like A* is another common issue [Sch21]. Shortest path routing fails to model the varied sub-optimal paths taken as a result of realistic wayfinding in unfamiliar environments. Work in this area is mainly focused on density adaptive planning, where agents intelligently avoid overly crowded areas [VTCIG12]. Furthermore, few current systems take into account the influence of distractions in the environment which draw attention away from the wayfinding task and can significantly alter the flow of crowds [KHKF21, KHKF20]. However, gaze modelling and interactions are a rich area in the literature. For example, there are methods to simulate gaze between real and virtual humans [NBR*16]. Most closely related to this work are those gaze methods which incorporate relative scoring mechanisms [GT09]. Recent works focused on the interaction between density and gaze behaviour reveal the effect local surroundings can have on gaze and movement behaviour and point toward incorporating vision based gaze models in crowd simulation [BHO*20]. There are a few vision-based crowds methods that have emerged [OPOD10] which includes density-adaptive synthetic-vision [HOD15]. All of these methods point toward a desire in the crowd animation field to have intelligent agents which automatically react to visual stimuli and also have intelligent behaviours. Our paper proposes the first fully parametric method to gain information from visible signage directly through agent based vision and explore accordingly to the gained information in an autonomous and intelligent manner, including when there is little or no information or the agent is new to the environment.

3. Overview

The aim of this work is to develop a performant multi-agent simulation system in which agents visually identify and observe signage using a more realistic method than simple ray cast visibility checks. This information should then be used to construct a cognitive map to assist in wayfinding to a goal location. To achieve this agent behaviour while improving upon certain aspects of existing models, there are several key challenges to overcome. Firstly, a representation of vision needs to be chosen, as well as a method of identifying objects of interest within that representation. Then, the information conveyed by the objects of interest which are selected needs to be encoded and stored. Finally, the stored navigational information must be used to guide future wayfinding.

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

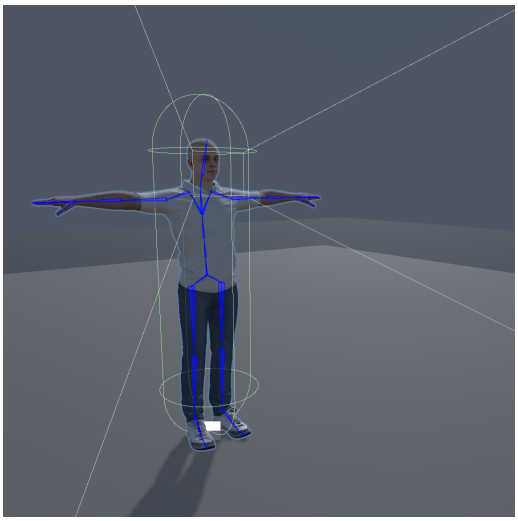

Figure 2: *Structure of the agent. Animation rigging is shown in blue, physics rigid body in green, and camera frustum in gray.*

3.1. Input Data

Before the beginning of the simulation, input geometry representing the simulation environment must be labelled and pre-processed. The user of the software must specify the locations and sizes of all the signs that should be available to the agents. Several parameters must be set for each sign: path length, sign type, indicated direction, strength of information, and goal location. The setting of these can be automated for different classes of signs and signs with explicit world locations. For each agent in the scene, a start position and a global goal position must be specified. For the geometry preprocessing step, we calculate walkability and store the information in a shared *obstacle grid*, this is one of several grid layers which are available to the agent during the simulation. The resolution of the obstacle grid is set by the user as a parameter of the model. In the tests conducted, a square grid representation with a resolution of $1m^2$ was used. A grid cell is considered walkable if a rectangular prism with a square base matching the size of the grid cell and a height equal to the maximum height of the agent does not intersect any obstacle geometry in the scene.

3.2. Agent Avatar

The chosen representation of the agent is relatively straightforward. For physics, the agent is represented as a capsule-shaped rigid body with a width of $0.3m$. For the purposes of animating the agent, any standard humanoid rigging can be used with a 3D model as well as idle, walking, and running animations. This character model is placed inside the capsule geometry to simplify collision checking. A perspective camera is attached relative to the humanoid rigging's head bone, such that the view from the camera represents the area of the scene visible to the agent. This structure is shown in Figure 2 with these key components highlighted.

3.2.1. Animation

The agent's animation is especially important because the positioning of the agent's rigging has a direct effect on the movement of its head bone, which, in turn, moves the agent's camera and impacts the information available at any given time. Changes to the agent's animation result in changes to the outcome of the simulation. We animate the agent's avatar by combining idle, walk, and run animations with a blend tree, then dynamically modify these combined animations by enforcing inverse kinematics (IK) constraints on the head and chest bones to set the agent's gaze direction. When the agent is not fixated on an object in the scene, the weights of these IK constraints are set to zero, and the gaze direction is decided based on the most recent movement direction. When an object is fixated on, the weights of the IK constraints are ramped up, and the position of the agent's look-at target point is linearly interpolated to the world position of the target object. The speed of IK constraint ramp-up and movement of the look-at point are configurable as parameters of the model, which control maximum the speed at which the agent's gaze changes direction. The location of the look-at point is set by behaviour tree nodes which process information collected by the agent camera, subsequent image processing and feature extraction.

3.2.2. Local Steering

The agent's high-level wayfinding behaviours decompose the process of navigating to a global goal into sequences of local goals. Certain agent behaviour nodes push locations in the environment to a local waypoint queue. The agent's local steering system continually consumes and navigates towards successive waypoints from this queue. When the agent reaches a waypoint, that waypoint is removed from the queue. When one of the agent's behaviors is interrupted by a higher priority behavior with an alternative navigation goal, the local waypoint queue is cleared. For example, if the agent is exploring, then notices a sign to approach, the waypoints associated with the random exploration behavior are discarded and replaced with the sequence of waypoints required to navigate towards the sign.

The simulations described in this paper used a simple model in which the agent's movement is driven solely by force applied to its rigid body in the direction of the next goal point in the queue. The calculation of this force is shown in equation 6. The agent's maximum walking speed is controlled by the k_{walk} parameter, and the speed at which the agent stops in controllable by the k_{drag}. This local steering model was chosen for simplicity while investigating other agent behaviours; a more powerful predictive collision avoidance system would be a drop in replacement by substituting the appropriate behaviour tree node.

$$\mathbf{F} = k_{drag}\mathbf{v} + k_{walk}\frac{\mathbf{p}_{waypoint} - \mathbf{p}}{|\mathbf{p}_{waypoint} - \mathbf{p}|} \tag{6}$$

3.3. Identification of Salient Objects

Salient objects are items in the environment which are visually interesting enough that the agent should slow down or stop, adjusting its gaze towards the object in question while it gains information

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

and investigates. We refer to this behaviour as the agent becoming fixated on the object. We consider two categories of salient objects. Firstly, there are signs, which when fixated upon provide useful wayfinding information, then we have all other distractions, which offer no relevant wayfinding information and simply slow the agent down as it proceeds to its goal. When a sign is identified, the agent will approach it until it is close enough to be readable. For all other distraction types, the agent simply stops and observes the distraction until it is no longer interesting, this simplification was chosen to avoid handling the complexity of modelling the decision making process of whether to stop, slow, or approach a distraction, while still having distractions impact the agent's time to reach its goal.

The process of identifying salient objects begins with the parametric saliency map shader [KCH*21], as described in section 2.1, being applied to the agent's head camera. To the standard PSM calculation given in equation 1, we add an additional "information decay" multiplier. This multiplier, $D_i(t)$, is used to reduce the saliency of objects that the agent has already observed. It is calculated at time t for salient object i using equation 7. The rate of information decay, $\dot{D}$, for each observing agent, and the initial amount of information, $D_i(0)$, for each salient object are controllable as a parameters for each agent and salient object in the simulation environment. The calculation also takes into account the angular separation between the current look-at point and the target object relative to the agent, such that when the agent is looking at an object head-on, the saliency decay is faster. The saliency decay multiplier is applied to the standard PSM fragment shader calculation according to equation 8.

$$D_i(t+\Delta t) = D_i(t) + \Delta t(\dot{D})(\hat{\mathbf{d}}_{current} \cdot \hat{\mathbf{d}}_{target}) \tag{7}$$

$$S_i(t) = S_i'(1 - D_i(t)) \tag{8}$$

After the modified parametric saliency map is rendered to a texture, key points are identified on that texture using a feature extraction pipeline. The pipeline applies thresholding, erosion and dilation image processing operations to reduce noise and create a binary image, and finally performs simple blob detection to identify key points that the agent might want to focus on. The limits of the thresholding and the kernel size for opening are controllable as parameters which effect the minimum saliency of objects noticeable by the agent. An example of the parametric saliency map and corresponding key points is shown in figure 3.

Having identified key points corresponding to areas in the agent's field of view which are candidates to fixate on, we need to choose between them. The choice is done randomly, with the selection probabilities of each point being weighted by the size of the corresponding blob. The selection probability of the ith salient object with total saliency S_i for an agent who is currently seeing n candidate objects is given by equation 9. By using this method of weighted random selection, objects tend to be fixated on in an approximate order of descending saliency.

$$P(obj_i) = \frac{S_i}{\sum_j^n S_j} \tag{9}$$

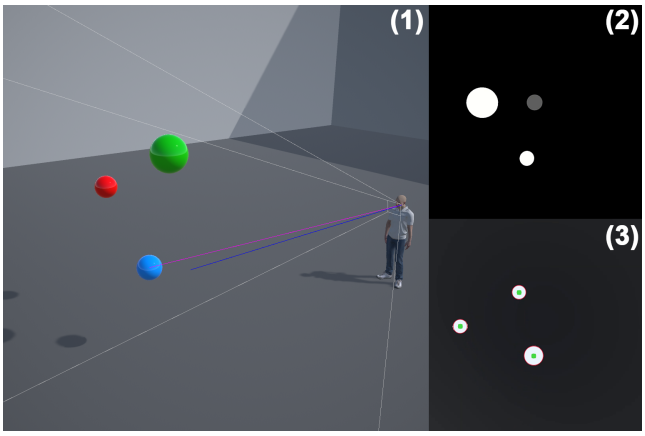

Figure 3: *(1) View of the scene with agent's gaze being interpolated towards one of three salient objects. The current gaze direction is shown in blue, the target direction is shown in pink, and the camera frustum is shown in gray. (2) the parametric saliency map representation of the agent's viewpoint. One of the circles, corresponding to the blue ball is not as a bright as the others due to saliency decay. (3) The result of the image processing pipeline on the saliency map from just before the agent fixated on the blue orb. The blue orb was selected randomly from the three green key points.*

Once a key point has been stochastically selected, the location on the vision texture in pixel coordinates must be mapped to the corresponding object's metadata, so that the agent can integrate the wayfinding information associated with that object into its decision making process. A simple approach would be using a ray cast through the agent's head camera's near plane at the key point position. Unfortunately, this approach is unreliable since blobs can be irregularly shaped and composed of multiple salient objects, resulting in the ray to failing to intersect with the objects of interest, or the wrong objects being selected.

A more reliable and faster method is to use a second shader which encodes unique salient object IDs in a 32 bit scalar *object ID texture*. We can mask the object ID texture based on the selected feature in the parametric saliency map. Each pixel in the masked area corresponds to an object ID. Since a salient blob might be made up of multiple salient objects, we make a second random selection between the objects that the blob is composed of, weighted by the frequency of the IDs associated with the corresponding pixels. This randomly selected object is the one that the agent fixates on. The entire process of selecting an object within the agent's partial isovist and obtaining it's information is summarized in algorithm 1.

3.4. Navigational Information

3.4.1. Obtaining Navigational information

If a salient object is fixated upon, and that object is a sign, then the agent needs to gain information as a result of having observed it. The agent immediately adjusts its gaze to face towards the sign to

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

Algorithm 1 Saliency Map Feature Extraction Pipeline

1. Generate parametric saliency map
2. Convert to binary image (threshold)
3. Erode and Dilate (opening)
4. Extract connected components (blob detection)
5. Select a key point, weighted by saliency
7. Extract corresponding salient object IDs from id texture
8. Select a salient object, weighted by frequency
9. Map pixel coordinates to salient object metadata

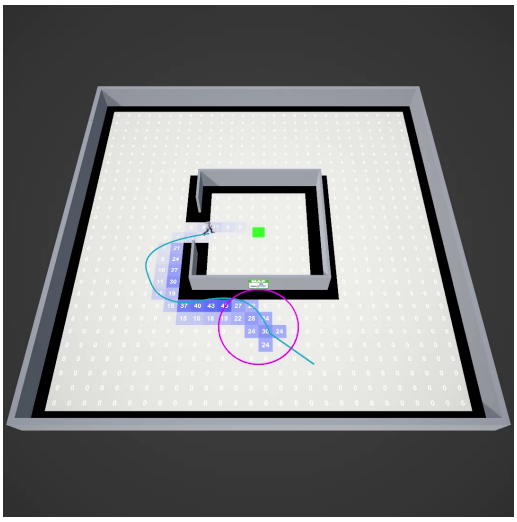

Figure 4: *The path taken by the agent is indicated by the blue trail. We see that the agent starts in a position with the map visible, but outside the VCA. The path taken goes first into the VCA, represented as a pink circle, to receive signage information. Then, the agent proceeds to the goal by following signage information in the navigation grid. The agent's navigation grid values are shown by the shaded grid on the floor of the scene.*

maximize the information gain rate, but does not initially gain any information until it is within the sign's visual catchment area, as defined by equation 2. In that equation, our model uses a constant minimum angular separation of 0.29, consistent with values in the NFPA Life Safety Code Handbook [XFG*07] and leaves the size of the minimum recognizable element to be set as a parameter for each sign in the scene by the user of the system. The centre of the sign's VCA is added to the local waypoint queue such that the agent moves close enough that the sign is legible, this behavior can be observed in figure 4. The information gain rate of of the agent, $\dot{D}$ in equation 7, drops off towards the edge of the VCA, so that the saliency does not decrease until the sign can be read and understood. When the saliency of the corresponding salient blob drops to zero, the sign is considered to have been read by the agent. After this point, the sign is no longer salient and the information associated with it is integrated into the agent's navigation grid.

3.4.2. Encoding Navigational information

Once the agent has fixated on and finished observing a sign, the information associated with that sign must be stored so that it can be used during the wayfinding process. To assist in representing navigational information, we introduce another grid layer over the environment for each agent, the *navigation grid*. The resolution of this grid is configurable as a parameter of the model, and was set to $1m^2$ per cell for most testing. All cells marked as walkable on the obstacle grid are associated with integer counters in the agent's navigation grid. Familiarity for any grid cell is capped at a configurable value. All counters are initially zero, and represent the agent's wayfinding familiarity with the corresponding area of the environment. Larger values indicate greater familiarity.

When the agent observes a sign, the sign increases these counters along a path of grid cells; this path is determined by the type of sign observed. There are two types of signs supported by the system: global signs, which represent maps, and directional signs, which represent arrows. Global signs increase counter values along the grid cells corresponding to the optimal path from the sign to the goal. Directional signs increase counter values along grid cells intersecting a line starting at the sign and going in the direction indicated by a user-specified vector controlling where the sign tells the agent to go. Both kinds of signs have several common parameters: strength, distance, and falloff. The amount that the grid cell counters increase, as specified by the strength parameter, linearly decreases to zero along the number of cells on the path specified by the distance parameter. The spread of information to cells adjacent to the path is controlled by the information falloff parameter. Parameters controlling the integration of signage information into the agent's navigation grid are summarized in table1.

In figures 4, 5, 9, 10, and 11 global signs are represented as "you are here" maps, while directional signs are represented as white arrows. This symbology was selected because it portrays the kind of information each kind of sign is intended to model; however, the agent does not employ image processing to "read" the signs, rather, once the agent has noticed a sign and entered its VCA, it directly accesses the sign's metadata and updates its navigation grid.

3.4.3. Acting on Navigational Information

The approach used to pick paths based on information in the navigation grid is similar to the human-like path planning approach, except that square grid cells are used instead of hexagonal tiles [RP22]. When an agent finds itself in an area of the environment with nonzero navigation familiarity, wayfinding is guided by an A* algorithm with a modified heuristic similar to 3. The only difference is that instead of a static multiplier of 2 being used in the calculation of the $\lambda_{x,z}$ term in the heuristic, the strength of the agent's preference for nonzero grid cells is a user-configurable value. The modified heuristic is designed such that the agent prefers paths through cells with higher familiarity, even if those cells do not represent the shortest path. The agent strongly prefers not to move through cells with zero familiarity. This approach works because the heuristic is not admissible; the agent overestimates the cost of moving into an unfamiliar cell. A* is not guaranteed to return a shortest path when using a heuristic that is not admissible, and will instead prioritize paths which pass through more familiar areas of

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

Parameter	Description
Sign Type	Categorical. Either "directional" or "global". Directional signs increase navigation information along a straight line. Global signs increase information along the shortest path to the goal location.
Distance	Integer. Maximum length of the trail of information left by this sign. A larger value causes the sign to guide the agent further towards the goal.
Strength	Integer. Initial magnitude of navigation familiarity increase provided by this sign. A larger value causes the path communicated to be more strongly preferred during wayfinding.
Falloff	Integer. Reduction in strength when recursively propagating the familiarity increase to adjacent cells.
Direction	3D Vector. Applies only to directional signs and indicates the direction the sign is suggesting the agent proceed in.
Goal Location	3D Point. Applies only to global signs and indicates the location of the goal the sign is leading the agent to. x

Table 1: *Signage metadata controlling navigation grid updates.*

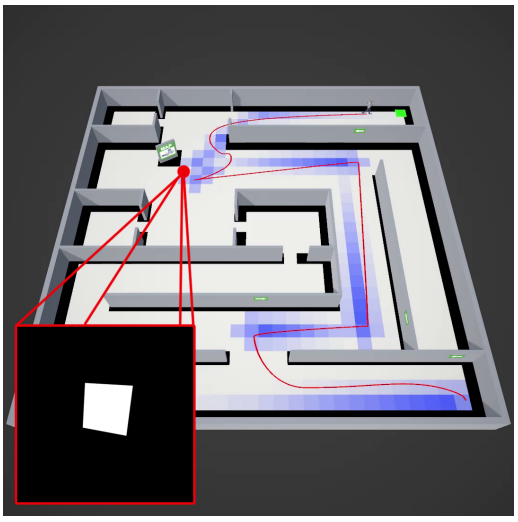

Figure 5: *The red trail shows the path of agent navigating towards a goal using both global (map) and directional (arrow) signage. The square in the bottom left shows the agent's head camera parametric saliency map as it approached the map. Darker blue grid cells correspond to areas of the environment where the agent has higher wayfinding familiarity as indicated by the integer counters in the navigation grid. Wayfinding using the navigation grid is interrupted when the agent notices a new sign to read.*

the environment. When the agent enters an area of the environment with which it has nonzero familiarity, as indicated by its navigation grid, it calculates a complete path to the goal. The grid cells along this path are pushed to the local steering waypoint queue and followed until either the agent reaches its goal, steps into a grid cell with zero familiarity, or becomes distracted. The result of using this path planning method is shown in figure 5, wherein an agent navigates through an environment with multiple paths to a goal using both global and directional signage.

3.5. Simulating Random Exploration

While the agent is in areas of the scene with nonzero wayfinding familiarity from signage, it can navigate using the modified A* approach described in the previous section. However, when there is no such information available on the navigation grid, the agent must randomly explore the environment in search of either the goal or more signage to follow. The simulation of random exploration is based primarily on visibility graph metrics. At the start of each simulation, a dense undirected graph is generated wherein each grid cell is a node, and two nodes are connected if they are visible from each other. Visibility between grid cells is determined by a ray cast between the locations where the agent's head camera would be if they were standing in the middle of each of two grid cells. From this graph we easily can calculate a variety of visibility graph measures representing the environment's configuration. These visibility graph measures are stored in a grid which is shared between all agents to avoid re-computation. To model random exploration, each agent maintains an *exploration grid* layer over the environment. As the agent navigates the environment, grid cells are marked as "seen" if they are currently, or have been previously, visible. Only these cells are considered as targets for exploration. Each exploration grid cell contains an *exploration heuristic* measuring the desirability of navigating to that location with the intention of gaining additional information about the environment.

3.5.1. Calculating the Exploration Heuristic

At each exploration grid cell, an exploration heuristic is calculated using equation 11. Each pre-calculated visibility graph measure h_i from the corresponding static visibility grid cell is multiplied by its weight w_i; in the current version of the agent model, there are $n = 2$ metrics considered: neighbourhood size and clustering coefficient. The chosen metrics and their corresponding weights are set by the user to enable control over the agent's preferences during random exploration. Multiplied to this sum of visibility graph heuristics are two additional factors used to avoid revisiting grid cells or selecting a cell which is too close to a previously selected cell.

$$w_{distance} = max(0, min(\frac{|\mathbf{p}_{node} - \mathbf{p}_{agent}|}{d_{ideal}}, 1)) \tag{10}$$

$$H_{exp} = (\sum_{i=1}^{n} h_i w_i)(w_{distance})(w_{familiarity})^c \tag{11}$$

Each time a cell on the exploration grid is visited during random exploration, a familiarity counter c is increased for that cell

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

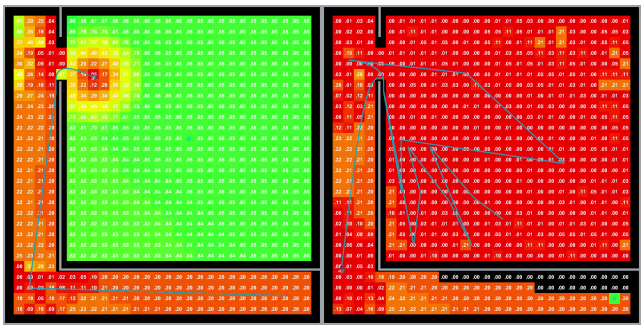

Figure 6: *The agent's path is shown in blue. Grid cell coloring indicates exploration heuristic value with red cell indicating lower exploration heuristic values and green more desirable higher heuristic values. The random exploration behavior works well to guide agents out of enclosed spaces, but is slow to navigate out of highly connected regions.*

by a user-specified constant $\Delta c \in \mathbb{N}$. The counters for adjacent cells are recursively increased by $c - 1$ while $c > 0$ and the adjacent cell is marked at "seen". A familiarity penalty in the range [0, 1], $w_{familiarity}$, is also set by the user. This penalty is raised to the power of the node's familiarity counter, such that the heuristic value of a cell decreases the more times it has been visited. This penalty is needed to avoid having the the agent get stuck in areas with initially high exploration heuristic.

The second multiplier involves $w_{distance}$, which is calculated as the straight line distance between the agent's current position and the candidate exploration grid cell. This distance can be thought of as the magnitude of the homing vector to a previously observed location; the ability of animals, including humans, to calculate the magnitude of this homing vector through the process of path integration is a well established navigation mechanism which may form the basis for cognitive maps of environments [Wan16]. The addition of this factor further discourages the agent from staying in the same location while exploring.

3.5.2. Using the Exploration Grid

The exploration grid cells with the highest heuristic values are considered for selection as the next location to explore to. A target location is selected randomly from the candidate locations. The agent considers all "seen" exploration grid cells as candidate points for random exploration, including those which are not in its current isovist. This is important because it means that the agent can remember previously seen locations that look tempting to explore and go back to them in the future. However, since the local steering model only works for straight line paths, this means that the selected random exploration destination cannot be naively added to the local waypoint queue. Instead, a shortest path is computed within the grid cells which have been marked as "seen"; since the path can only use areas that the agent has seen, it is not necessarily a global shortest path. Examples of this exploration behavior are seen in figure 6, which shows two opposite situations: exploring from a confined area to a highly connected one, and vice versa.

3.6. Coordinating Agent Behaviors

Agent behaviors are coordinated by an event-driven behavior tree. A simplified version of this behavior tree is shown in figure 7, wherein the five primary agent behaviors are labelled (1) through (5). These behaviors are ranked by increasing importance such that any lower-priority (higher number) behavior can be interrupted by a higher priority (lower number) behavior. The activation of each sequence is subject to a conditional node which functions as a predicate controlling the activation of the corresponding behavior nodes. The placement of the agent's behavior tree in the context of the other elements of the simulation system is shown in Figure 8. The functionality of these behavior groups, in order of descending priority, are as follows:

1. The highest priority behavior is to stop when the goal is reached. Regardless of whatever else is going on in the scene, the agent will always stop and do nothing once it reaches its goal, interrupting whatever is was doing before reaching it. The conditional for this behavior group is a check to see if the agent is sufficiently close to the goal location, as configured by the *goal threshold* parameter.
2. The second behavior group's conditional node checks for the detection of a salient object by the agent's PSM feature extraction pipeline from section 3.3. This behavior is only allowed to activate if the agent doesn't already have a completely informed path to the goal location in its navigation grid; this prevents the agent from looking at more signs when it already knows where to go. When triggered, this behavior guides the agent into the VCA if a sign was detected, observes the object until it is no longer salient, then updates the navigation grid appropriately as in section 3.4.
3. The goal seeking group's conditional checks if the goal location is visible within the agent's isovist at any time. If it is, then the agent will immediately clear its local waypoint queue and navigate straight to the goal location.
4. The final conditional behavior group is triggered by the agent residing in a navigation grid cell with a nonzero counter value. This behavior has the agent follow the navigation grid as in section 3.4.3 until it is interrupted by a higher priority behavior, or strays into a region with zero wayfinding familiarity.
5. The fifth behavior group has no conditional; it is the fallback behavior which is used when there is nothing else for the agent to do. The behavior tree nodes in this group control the heuristic-based random exploration behavior described in section 3.5.

Figure 9 shows the results of the simulation in an environment with multiple agents demonstrating coordination of various behaviors in the agent behavior tree. We see the red trail agent initially exploring out of a snaking hallway using the heuristic-based random exploration, which guides it to the more highly connected central room. The red agent then sees map in that room, approaches its VCA, observes it, and adds a trail of information to its navigation grid along the optimal path to the goal, which it then follows. The blue trail agent has a similar experience to the red agent, except that it navigates out from its initial location using two directional arrow signs. Once it exits its starting hallway, its navigation grid following is interrupted by seeing the map. The green agent starts in an open room, where it is initially distracted by a red sphere.

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

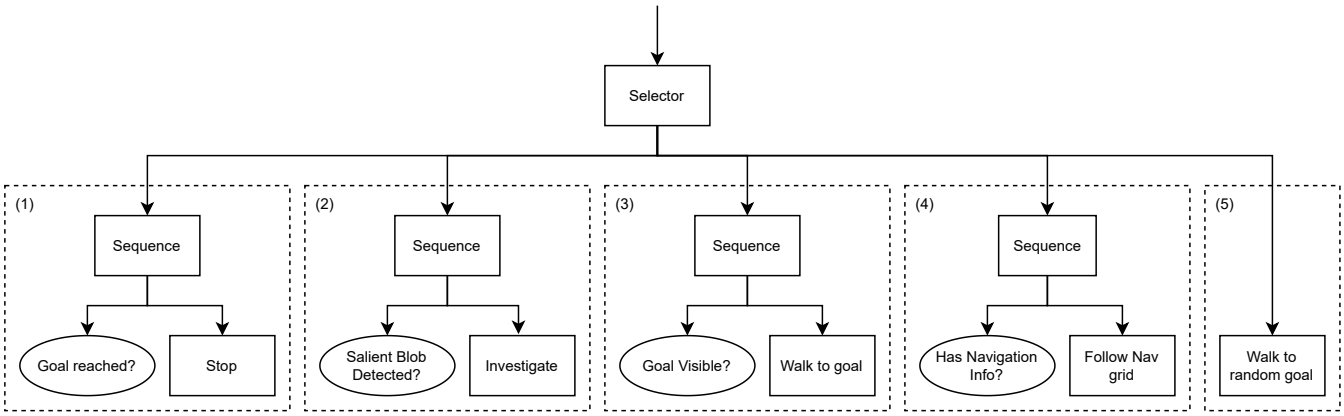

Figure 7: *A simplified representation of the agent model's event-driven behavior tree. Conditional nodes are displayed as circles. There are five main subtrees containing groups of behaviors. The behavior groups are ordered from left to right by priority; any behavior group to the left of another group can interrupt that group if its conditional node is evaluated as true.*

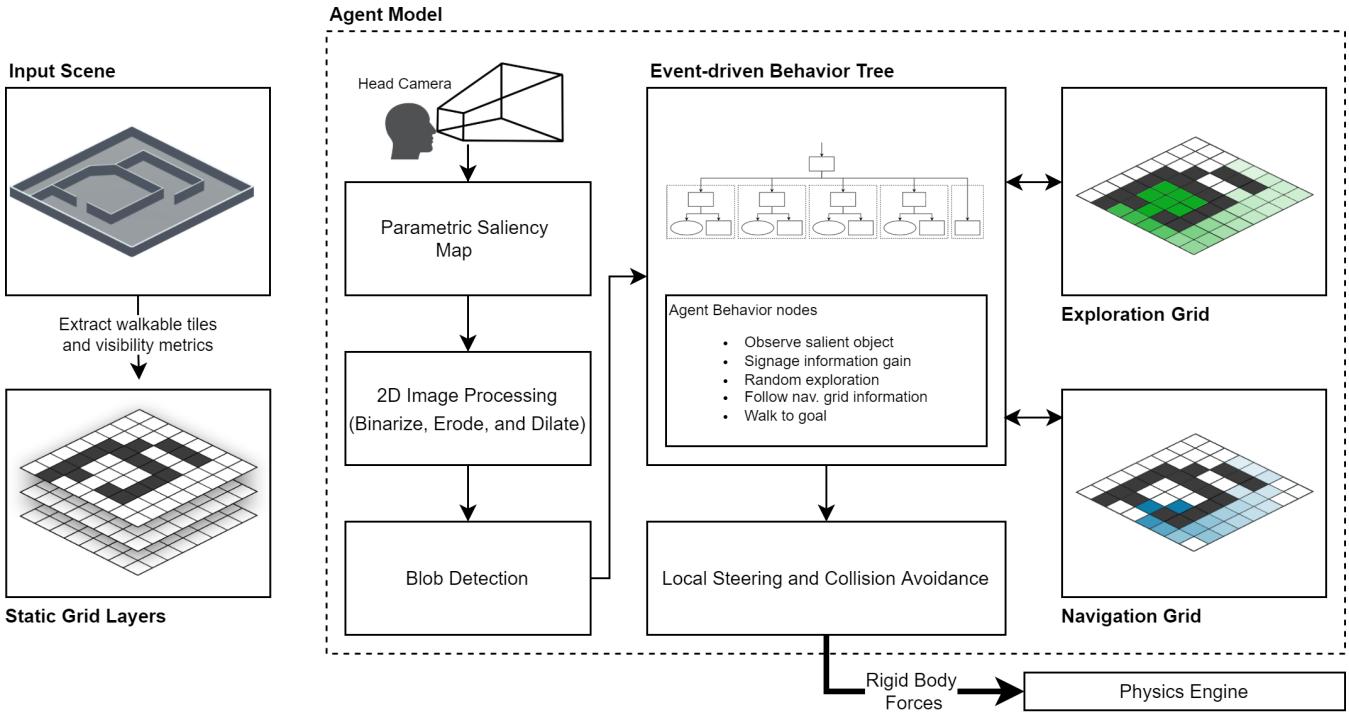

Figure 8: *Overall agent design. The simulation environment is preprocessed to extract static grid layers including walkability and visibility metrics. Each agent has a camera whose output is processed by the PSM shader and image processing pipeline. This data is integrated into the navigation grid, which is used in conjunction with the Agent's grid of exploration heuristics to guide wayfinding decisions coordinated by the event-driven behavior tree. The model's outputs are rigid body forces which move the agent through the scene.*

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

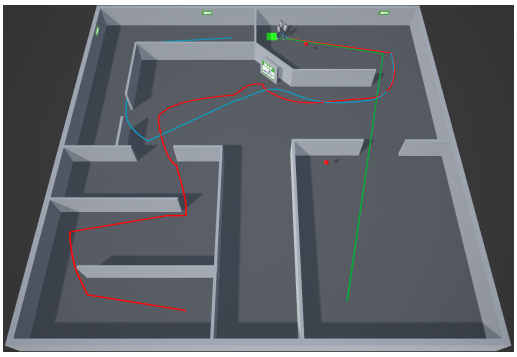

Figure 9: *A complex scene containing multiple agents whose paths are represented by red, green, and blue trails. This environments requires agents to make use of multiple behavior tree groups to reach the goal.*

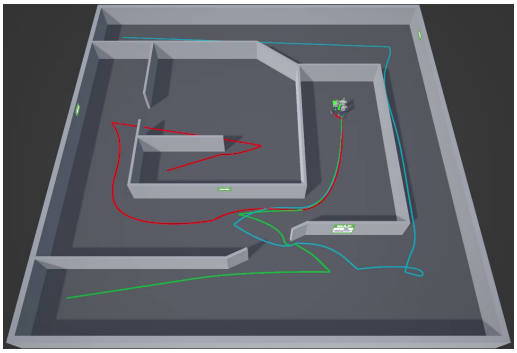

Figure 10: *A complex scene showing the paths of the agents in red, green, and blue. All the agents use a combination of random exploration and various types of signage in the scene to find the goal more quickly than with completely random movement, and more slowly than an unrealistic shortest path approach.*

The distraction provides no navigational information, so the agent runs across the central room to an arrow sign near the goal. Before reaching the goal tile, once they enter the top right room, all three agents have their "walk to goal when visible" behavior interrupted to observe a non-signage distraction displayed as a red sphere.

Figure 10 and Figure 11 shows the results of the simulation in an environment with multiple agents in increasingly more difficult scenarios. Figure 10 demonstrates a simulation where agents obtain enough information to initially proceed the correct direction but require exploration to find the next signage. In Figure 11, we significantly increase the difficulty of the environment by adding more signage, conflicting signage for different goals, and two groups with conflicting paths to their goals that they must find in the environment using signs and exploration. This leads to a classic crossing groups scenario where emergent lane forming from the steering control helps resolve the scenario.

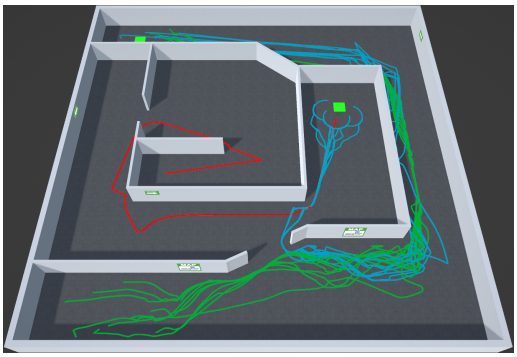

Figure 11: *A complex scene showing the paths of the agents in red, green, and blue. Here, there are blue and green groups which interact and steer among each other. The scenario geometry is the same as Figure 10 but more signage and conflicting signage for directions have been added. The green and blue groups also have conflicting paths to their goals which leads to a crossing groups scenario.*

4. Evaluation

In this section we evaluate the proposed model from a few different perspectives. First we look at the cost of random exploration since this is the baseline, no information case. Then we evaluate the computational performance.

4.0.1. Random Exploration

The chosen method of exploration tends to encourage agents to seek wide-open areas where they can see more of the environment, this can be seen in left side of Figure 6, wherein the agent picks exploration grid cells with high neighborhood size as the next destination; it goes straight for the entrance to the large room. This is an effective strategy for many environments because in a larger, more connected area, the agent is more likely to notice some signage which can be used to find the goal.

The worst performance of the exploration heuristic is observed when the agent starts in a highly connected region and needs to go into a small passage to find the goal. In this case, the agent must fully explore the open area such that the cells in the room are visited enough times that familiarity multiplier causes the lower neighborhood size nodes to be preferred. This worst-case situation is shown in the right side of Figure 6, and could be likely be improved by adjusting the exploration heuristic's weighting of visibility metrics and the size and spread of the familiarity penalty.

4.1. Computational Performance

Depending on the complexity of the scene, the system can accommodate around a dozen agents while maintaining real time performance at around 60 frames per second on a AMD Ryzen 3 3100 Processor, GeForce GTX 1070 GPU machine with 32 GB of memory. If real time performance is not desired, an arbitrarily high number of agents can be accomodated, limited only by the system memory. Profiling reveals that the primary bottleneck is the image processing pipeline. Generation of the parametric saliency maps is fast,

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

but the feature extraction pipeline is slow, accounting for over 50 percent of the total CPU time of the simulation. This bottleneck occurs because the implementation of the PSM processing pipeline is executed on the CPU, while the PSM textures themselves are rendered on the GPU; PSM textures must be transferred from the GPU to the CPU at every update step. This is a slow blocking operation. For these tests, visual information was rendered to 64x64 resolution textures. Increasing this resolution results in a more severe performance bottleneck. The current implementation of neighborhood size is $O(n^2)$ and clustering coefficient is $O(n^3)$ on the number of walkable cells, where n is the number of walkable cells. Tests were conducted using $900m^2$ grids with a resolution of $1m^2$, at this resolution, startup times ranged from 10 to 60 seconds.

5. Conclusion

The proposed approach encodes the complexity of modelling human wayfinding behaviour in a configurable method that affords authoring scenes with intelligently exploring agents. The key components of this agent model are vision-based identification of signage, a per-agent navigation grid for informed wayfinding, static grid layers to encode walkability and visibility metrics, and a precomputed spatial metrics based exploration behaviour. The agents avoid following the simple shortest-path routes to their goals that would be typical of models including unrealistic global information, while also moving through the scenes faster than they would without additional information from signage.

5.1. Limitations and Future Work

We show that our proposed model is robust and produces interesting non-trivial paths in complex situations, however, there are certainly some limitations to the current approach. Firstly, there is no mechanism for multi-objective signage systems, as would be seen in buildings with multiple possible destinations. To study signage layouts of this type, separate signage layouts and simulations must be conducted for each objective. Also, signs are "all or nothing" in the sense that an agent can't receive partial information from a partially visible sign. Secondly, the agent always tries to take a straight line path to a sign's visual catchment area, and to a goal if it sees one. This causes issues if a sign is visible but not accessible across something like an atrium, in this case the agent ends up walking into a wall forever while it tries to reach the sign's VCA; this issue can be mediated by adding a timeout to group (2) in the behaviour tree, though this does not address the underlying issue. While the current method of following the navigation grid works well for optimal and slightly sub-optimal paths, there is currently no way to have a sign which is so misleading that it guides the agent to a dead end; this an unavoidable side-effect of the modified A* approach.

The current performance of the model could be improved substantially by developing a GPU-based implementation of the saliency processing and object tracking pipeline. Currently blob tracking and image processing are a computational bottleneck.

There are several social effects which could be added to the simulation to increase realism. If an agent sees another agent looking in a direction, they should also want to look in that direction; this is a known phenomenon in real human behaviour [GHS*12]. The local steering model should be swapped out for something that includes social forces [HM95] and short-term prediction [KHvBO09] to reduce collisions with nearly agents and obstacles. Additional realism could also be added if, when uninformed, agents had a tendency to follow other agents that might have might have more information.

Currently, the approach to gaze control is overly simplified when the agent doesn't have a specific object to focus on. A possible future improvements would be to consider gaze direction as part of the configuration space sampled by the agent during it's random exploration behaviour. This would mean exploration goals would include going to a specific location, as well as looking in a specific direction.

6. Acknowledgements

The research was supported in part by NSERC Discovery, Canada [funding reference number RGPIN-2021-03541].

References

[BHO*20] BERTON F., HOYET L., OLIVIER A.-H., BRUNEAU J., LE MEUR O., PETTRÉ J.: Eye-gaze activity in crowds: impact of virtual reality and density. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)* (2020), IEEE, pp. 322–331.

[DKM*20] DUBEY R. K., KHOO W. P., MORAD M. G., HÖLSCHER C., KAPADIA M.: Autosign: A multi-criteria optimization approach to computer aided design of signage layouts in complex buildings. *Computers & Graphics 88* (2020), 13–23.

[DKT*17] DUBEY R. K., KAPADIA M., THRASH T., SCHINAZI V. R., HOELSCHER C.: Towards an information-theoretic framework for quantifying wayfinding information in virtual environments. In *CAID@ IJCAI* (2017).

[DMS*22] DUBEY R. K., MORAD M. G., SOHN S. S., XUE D., THRASH T., HÖLSCHER C., KAPADIA M.: Cognitively grounded floorplan optimization to nudge occupant route choices. *Available at SSRN 4003119* (2022).

[DTK*19] DUBEY R. K., THRASH T., KAPADIA M., HOELSCHER C., SCHINAZI V. R.: Information theoretic model to simulate agent-signage interaction for wayfinding. *Cognitive computation 13*, 1 (2019), 189–206.

[GHS*12] GALLUP A. C., HALE J. J., SUMPTER D. J. T., GARNIER S., KACELNIK A., KREBS J. R., COUZIN I. D.: Visual attention and the acquisition of information in human crowds. *Proceedings of the National Academy of Sciences - PNAS 109*, 19 (2012), 7245–7250.

[GT09] GRILLON H., THALMANN D.: Simulating gaze attention behaviors for crowds. *Computer Animation and Virtual Worlds 20*, 2-3 (2009), 111–119.

[HM95] HELBING D., MOLNÁR P.: Social force model for pedestrian dynamics. *Physical review. E, Statistical physics, plasmas, fluids, and related interdisciplinary topics 51*, 5 (1995), 4282–4286.

[HOD15] HUGHES R., ONDŘEJ J., DINGLIANA J.: Davis: density-adaptive synthetic-vision based steering for virtual crowds. In *Proceedings of the 8th ACM SIGGRAPH Conference on Motion in Games* (2015), pp. 79–84.

[JHDZ15] JIANG M., HUANG S., DUAN J., ZHAO Q.: Salicon: Saliency in context. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1072–1080.

[KCH*21] KREMER M., CARUANA P., HAWORTH B., KAPADIA M., FALOUTSOS P.: Psm: Parametric saliency maps for autonomous pedestrians. In *Motion, Interaction and Games* (New York, NY, USA, 2021), MIG '21, Association for Computing Machinery.

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.

[KHKF20] KREMER M., HAWORTH B., KAPADIA M., FALOUTSOS P.: Watch out! modelling pedestrians with egocentric distractions. In *Motion, Interaction and Games*. 2020, pp. 1–10.

[KHKF21] KREMER M., HAWORTH B., KAPADIA M., FALOUTSOS P.: Modelling distracted agents in crowd simulations. *The Visual Computer 37*, 1 (2021), 107–118.

[KHvBO09] KARAMOUZAS I., HEIL P., VAN BEEK P., OVERMARS M. H.: *A Predictive Collision Avoidance Model for Pedestrian Simulation*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 41–52.

[NBR*16] NARANG S., BEST A., RANDHAVANE T., SHAPIRO A., MANOCHA D.: Pedvr: Simulating gaze-based interactions between a real user and virtual crowds. In *Proceedings of the 22nd ACM conference on virtual reality software and technology* (2016), pp. 91–100.

[OPOD10] ONDŘEJ J., PETTRÉ J., OLIVIER A.-H., DONIKIAN S.: A synthetic-vision based steering approach for crowd simulation. *ACM Transactions on Graphics (TOG) 29*, 4 (2010), 1–9.

[PT02] PENN A., TURNER A.: Space syntax based agent simulation. Springer-Verlag, 2002.

[RNG12] RONCHI E., NILSSON D., GWYNNE S.: Modelling the impact of emergency exit signs in tunnels. *Fire Technology 48*, 4 (2012), 961–988.

[RP22] RAHMANI V., PELECHANO N.: Towards a human-like approach to path finding. *Computers & Graphics 102* (2022), 164–174.

[Sch21] SCHWARTZ M.: Human centric accessibility graph for environment analysis. *Automation in Construction 127* (2021), 103557.

[SVF21] SCHWARTZ M., VINNIKOV M., FEDERICI J.: Adding visibility to visibility graphs: Weighting visibility analysis with attenuation coefficients. In *Proceedings of the 12th Annual Symposium on Simulation for Architecture and Urban Design* (San Diego, CA, USA, 2021), SimAUD '21, Society for Computer Simulation International.

[Tur01] TURNER A.: Depthmap: a program to perform visibility graph analysis. In *Proceedings of the 3rd International Symposium on Space Syntax* (2001), vol. 31, Citeseer, pp. 31–12.

[VTCIG12] VAN TOLL W. G., COOK IV A. F., GERAERTS R.: Real-time density-based crowd simulation. *Computer Animation and Virtual Worlds 23*, 1 (2012), 59–69.

[Wan16] WANG R. F.: Building a cognitive map by assembling multiple path integration systems. *Psychonomic Bulletin & Review 23*, 3 (2016), 692–702.

[XFG*07] XIE H., FILIPPIDIS L., GWYNNE S., GALEA E. R., BLACKSHIELDS D., LAWRENCE P. J.: Signage legibility distances as a function of observation angle. *Journal of fire protection engineering 17*, 1 (2007), 41–64.

© 2022 The Author(s)
Computer Graphics Forum © 2022 The Eurographics Association and John Wiley & Sons Ltd.