Cyber-Shepherd: A Smartphone-based Game for Human and Autonomous Swarm Control

Hayden Wade and Hussein A. Abbass

***Abstract*— Human and Autonomous control of a swarm of robots is a non-trivial task compounded by the multi-agent nature of a swarm and the tight-coupling in swarm dynamics. Our previous work has been successful in designing controllers to shepherd a swarm of sheep by learning from human demonstrations. However, humans could get disengaged easily if the platform to collect demonstrations is uninteresting. In this paper, we propose a smartphone based Shepherding game as a platform to collect data on the behaviour of Shepherding models. The game is designed to allow both humans and artificial intelligence (AI) empowered shepherds to control the swarm of sheep. The game offers a plug-and-play ability for different human and AI controllers of the shepherd. The game logs all movements made by the human during controlling the shepherd for use by supervised learning algorithms to develop autonomous controllers. We present and evaluate different components of the game.**

I. Introduction

Shepherding is a behaviour largely observed in sheepdogs within agriculture to undergo feeding or shearing [1]. Shepherding is defined as "controlling a flock of artificial sheep with an artificial sheepdog." ([2, p. 1]) This ability to move large numbers of self-autonomous agents generated some interest from machine learning researchers [3] including the use of genetic algorithms [4] to develop accurate shepherding behaviours. Shepherding has been used in applications such as robotics for the autonomous control of multiple robots [5], agriculture by using drones as an alternative to sheep dogs, and for crowd control [6].

The 'Shepherding Problem' has been tackled in several different ways since the concept of artificial flocking was introduced by Craig Reynolds [7]. Jyh-Ming Lien's model [8] is one example of a shepherding model which would benefit from a shepherding game platform. The model utilises an interactive motion planning method, where a human can provide visual hints through laser pointers to assist the shepherding model in determining the best motion planning path. The model is able to navigate an environment with obstacles and can use a single or multiple shepherds. The human-computer interaction approach taken in their work demonstrated that the method "gains both efficiency and accuracy" in shepherding. Furthermore, the model also utilises adaptive road-map techniques [9] to enable more effective shepherding behaviours. A single shepherd was able to achieve locomotion such as side-to-side motion or road-maps around obstacles which subsequently reduced flock separation and travelling time. Despite the fact that this model can be used within a number of different environments with single and multiple shepherds, it still requires the assistance of a human to accurately shepherd. Lien states that the system itself could be improved through enabling the algorithm to learn from these user inputs.

Strömbom [10] illustrated an algorithm where each self-propelled agent in a flock was subject to local attraction-repulsion type forces from the flock and interactions from the shepherd itself. Each agent aims to stay away from the shepherd while remaining close to its n nearest neighbours. By setting a specified target location, it was found that "the shepherd tends to initially collect the agents until they are cohesive, at which point it starts to drive the herd". Strömbom emphasised how the shepherd switches between this driving and collecting behaviour until the agents are delivered to the target. Through this behaviour, a side-to-side motion emerged consequently, which mimicked sheepdog behaviour. Strömbom's algorithm can successfully herd "80 or more sheep" whilst other shepherding models required multiple shepherds to successfully herd that number.

Our previous work [11], [12] proposed the use of machine education to design syllabus for educating an autonomous neural-based shepherd. The success of a supervised learner in [11], [12] relies on the existence of a large number of demonstrations collected during humans controlling the shepherd. However, this was proved to be a tedious task. Humans could get disengaged easily after a few demonstrations if the simulation environment is uninteresting. Moreover, the simulation environment lacked the flexibility and modular design to allow for multiple control models to be tested.

In this paper, a smartphone-based shepherding game is developed to provide a means to collect significant data using a variety of models with ease. We call the game Cyber-Shepherd. It equally offers consistency when comparing the performance of different models. Gamification allows for a large quantity of user behaviour and actions to be recorded. We chose smartphones considering their wide availability and ease of use. Globally, smart-phone-based games make up 41% of the games market revenue [13]. The use of smartphones opens the possibility for data collection remotely by transmitting the data through a mobile connection to a cloud server.

II. Gamification for shepherding

Sebastian Deterding's paper [14] defines gamification as "the use of video game elements in non-gaming systems to improve user experience (UX) and user engagement". Deterding continues to illustrate that while there are numerous gamified applications in "finance, health, sustainability...and tutorials", there is a body of research looking to "games with purpose piggybacking game-play behaviour to solve human

information tasks". Whilst Deterding does not mention the application of gamification to AI behaviour, it is evident that this body of knowledge could be implemented within several other fields of work, in particular, to collect data on user input to shape AI behaviour.

Gamification has been demonstrated in several fields in recent years. A good example is discussed by Andrea Rizzoli with the development of a program called "SmartH2O" [15] which is described as "providing users information on their consumption in quasi real-time". The platform uses a gamified user interface which implements a leadership board and a physical reward system which provides an incentive for decreased water consumption. It was found that SmartH20 users consumed 20% less water than non-users.

Another example is seen through Danae Vara's work [16] which discusses a program called Meeco designed to "guide, encourage and commit people to live eco-friendlier lives in a more social and enjoyable way". The game implements a scoring system that rewards if "the user repeats an action constantly" and "keeps being eco-friendly". There are smaller goals which the user obtains rewards from, however the gamification aspect comes into play by motivating multiple users to undertake an eco-friendlier lifestyle.

This concept harps with the idea of a Shepherding model game test platform, where the user input for control of shepherds could be used as shepherding behaviour to teach an AI [17] through machine learning. Deterding himself predicts the influence of gamification by stating that "Gamified systems 'in the wild' provide new objects of inquiry in an unprecedented variety, data quality and scale".

Our proposed Shepherding game will implement aspects of gamification to motivate user involvement and as a result, would help collect data on shepherding behaviour that could be used to influence the development of more accurate shepherding models or could be used directly to teach an AI what actions are effective.

III. Game Design and Description

We define three goals for the design of the game to enable achieving the intent. The first goal is to implement a software design that enables dual use of the game to act as both an engaging experience for a user as well as a test platform for the testing of various autonomous shepherding models.

The second goal is to allow these shepherding models within the game to produce data that can be compared among many other models; thus, data standardisation is important. The game needs to be developed to allow the game-play from each level to be recorded in real-time.

The third goal is to develop a game platform that enables human input through a touch interface that can control the shepherd. The human demonstrations could be used to train an AI similar to the work in [12].

The functional and performance requirements were determined for the game and traced back to the goals as depicted in Table I.

Identifier	Requirement Outline	Traceability
FR01	The game shall implement an object-orientated design for modularity.	Goal 1
FR02	The game shall take user input through a touch screen.	Goal 3
FR03	The game shall be implemented on Android phones with various screen resolutions.	Goal 3
FR04	The game shall use Strömbom's Algorithm as the AI for shepherd.	Goal 1
FR05	The game shall use Reynold's flock behaviour for the sheep.	Goal 1
FR06	The game shall collect game-play interaction data.	Goal 2
FR07	The game shall be a 2D, isometric pixel art game.	Goal 3
FR08	The shepherd and sheep shall interact and not overlap obstacles.	Goal 3
PR01	The game shall run at a minimum of 30FPS.	Goal 3

TABLE I: Functional and Performance Requirements of Cyber-Shepherd

A number of design constraints were considered. Firstly, due to the nature of Android Studio which was used to design the game, the game was not designed for the use on Apple phones. This was the case as Apple have their own development kits. Furthermore, since the processors within phones are considerably slower than their computer counterparts, careful design and optimisation of the game needed to be considered to ensure the game could meet the requirement.

Since the average gamer spends approximately 24 minutes playing games on mobile device each day [19], it was determined a game with simple mechanics and rules would be the most suitable. The game places either the user, the Shepherding algorithm or a previous replay in control of a shepherd that must herd all the sheep clusters on the map to the target, a green square area.

The game requires the user to collect all the sheep clusters on the map and shepherd them to the green target in the shortest time possible. The score is inversely proportional to the time spent on the level and proportional to the number of sheep shepherd. In certain levels, there are obstacles on the way, which the user must herd the sheep around to successfully complete the level.

The game design involved determining the rules of the game along with the user interface structure and design. Furthermore, the gamification elements connecting the underlying data collection aims to the user engagement and enjoyment were also considered.

The game is a 2-dimensional isometric game that utilises pixel art as its design. Elements of cyberpunk are used to create a futuristic urban environment. As such, the game was named 'Cyber Shepherd' to reflect this design. The environment starts off as a country area with spread out trees and grassy areas. The user must herd sheep towards a green grid that triggers a level success as shown in the second image in Figure 1. As the levels progress, the player moves through the game world towards a city environment where obstacles such as buildings, or fence lines force the player to avoid these obstacles to achieve the best score.

Cyber Shepherd has three distinct game modes; Play, simulate and replay. The play game mode enables the user to take control of the shepherd and shepherd the sheep towards the target. The user uses the touch pad on the screen to move the shepherd and the game-play is saved in real-time to a text file.

The simulate game mode enables the user to watch the shepherding algorithm (in this case, Strömbom's Algorithm) conduct the level and see observe it performs. This mode is also recorded to a text-file.

The replay game mode can be selected on any level that has previous game-play available from the play game mode. When this game mode is selected, the last game-play of the user can be watched.

The game implements several menu screens that continue the cyberpunk theme of the game. These menus allow the user to choose which level they wish to play and change settings such as sound and music volume.

The game implements 15 distinct levels with each progressing level providing the user through increasing complexity by changing shepherding and environmental parameters. Any level can be selected by the user through the menu screens. The level is defined to be completed when the global centre of mass of the sheep is in the target area.

With each level that is played, the same data is collected for each game-play iteration. The parameters that are changed to increase complexity include the number of sheep, the number and position of sheep clusters, sheep dispersion, obstacles and initial shepherd positions. Other variables such as the noise level and relative speed/shepherd speed can be changed, however these were kept constant for the level design of the game. Figure 1 shows the parameters that affect the complexity of the level along with the data that can be collected from each level which are influenced by these complexity parameters.

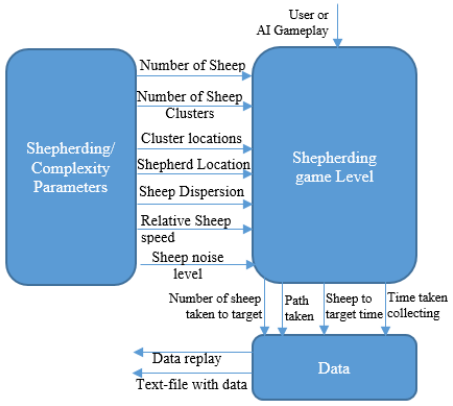

Fig. 1: Diagram of the complexity factors that influence each level along with the output parameters collected.

The first five levels are designed as simple tutorials to guide the user around the mechanics of the game. The first level has the sheep cluster nearby and the user simply has to drive the cluster along the road to the green target. This can be seen in the first image in Figure 2. The second level is similar however the player must first find the cluster and collect the sheep which are more dispersed. The next three levels familiarise the player to finding and collecting 2 clusters of sheep and avoiding obstacles on the way to the target.

The first parameter that was changed to increase complexity is the size of a single sheep cluster, ranging from 20 to 70 over five levels. The initial positions of the shepherd, sheep cluster and the target are varied through the five levels. This tests the performance of both the user and the shepherding algorithm in similar scenarios. Level 10 can be seen in Figure 2 as the middle image. In the top left hand side of the image, the target location can be seen.

The levels then progress to the last five levels which implement several environmental obstacles which the user/AI must detect and avoid to successfully shepherd the sheep to the target. In this case, levels 11 and 12 involve obstacles with a single cluster of sheep. The remaining levels involve multiple shepherd clusters and obstacles that must be avoided in order to complete the level. Level 11 can be seen in Figure 2 as the last image. The shepherd can be seen moving across a bridge which forces the shepherd to navigate through a narrow path.

Each level implements a tile map constructed in the program Tiled. This program allows each level to be constructed almost entirely visually. It enables the visual game environment to be designed easily and obstacles defined and the initial shepherd, cluster and target locations can also be chosen visually. This information can then be imported by the game from the level file.

Fig. 2: Cyber Shepherd Levels 1, 10 and 11 respectively

The shepherding model takes the parameters imported by the game architecture from the tiled map along with the hard-coded boundary size and number of sheep for that level and initializes the shepherding model. In the case the user chooses to simulate the level, the shepherding model will then calculate the new position of the sheep and shepherd each time-step to get the GCM of the sheep to the target. If the user chooses to play the game, the shepherding model outputs the position of the sheep and take the players user input for the shepherd position. This information is then used by the game architecture to render the game and to output the information to the data collection subsystem.

Data Collection is performed within the game to enable the possibility for analysis of game-play. Both user and AI game modes record the game-play. When a level is chosen, the initial parameters related to that level including obstacle, player, sheep cluster and target locations are saved to an external file. The Data collection class creates a new unique file for each iteration of game-play. Once this has been created, the game starts rendering and with each frame, the coordinates of the sheep and the shepherd are recorded to the text-file. Data collection stops if the user either exits the game or finishes the level.

Previous game-play of any level can be replayed within the game. If the user chooses the 'Replay' mode of a particular level, it will replay the last recorded game-play from that level or give a 'No replays' message if there are no replays available. When game-play is available, the Data Replay class reads through the file and saves the initial parameters to variables and saves the coordinates of the sheep and shepherd each frame to array lists. This information is then used by the Game screen to replay each frame of the game play.

IV. Algorithms and Implementation Environment

From the requirements set for this paper and using the game design structure produced within the preliminary design phase, the software design was developed. The game was implemented in Java using Android Studio. Furthermore, the gaming Framework LibGDX was used, which provided a number of libraries which aided the development of the graphical component of the game and enabled the gamification aspects of the game to be implemented.

Within the design phase of the paper, research was performed on each of these components to determine the best design structure for the game. Specifically, the use of 'LibGDX' required a number of libraries to be researched and implemented. The class structure was determined as a result of this research along with the requirements set for the game.

The class diagram in Figure 3 shows how the game operates for each of the three modes available. When the user selects to either play or simulate the game, the shepherding algorithm classes are initialised. In this case, the initial position of the target, shepherd and sheep are taken from the properties of the map through the Map Handler by the Game Screen and passed to the master call class. The master call class then initialises the positions of the sheep at the cluster location along with the initial position of the shepherd and the target. The individual positions of each sheep at that cluster position are determined by a particular seed. The master call class contains a number of methods that set these variables for the algorithm along with a number of set parameters for the shepherding and sheep behaviours. It also contains update methods used to call the Shepherd and Sheep Objects classes to update the position of the shepherd and sheep respectively based on each others locations each frame.

The Shepherd class is used to determine the next position of the shepherd based on its relative distance to the sheep

Algorithm 1 Shepherding Game Pseudo-code

```
initialise tile-map level();
initialise level(shepherd location, cluster locations, target location, obstacles);
save level parameters to text-file();

while sheep GCM != target location do
    if game state = user then
        shepherd location = user Input();
    else if game State = simulation then
        shepherd location = get Strömbom shepherd location(sheep locations);
    else if game State = replay then
        shepherd location = shepherd location from text-file(time-step);
        sheep locations = sheep locations from text file(time-step);
    if game state = user || game State = simulation then
        sheep locations = update Sheep Locations(player Location);
        save Frame to file(sheep locations, shepherd location, time elapsed);
    render to screen();
end
save score to scoreboard();
level success screen(time elapsed, score);
```

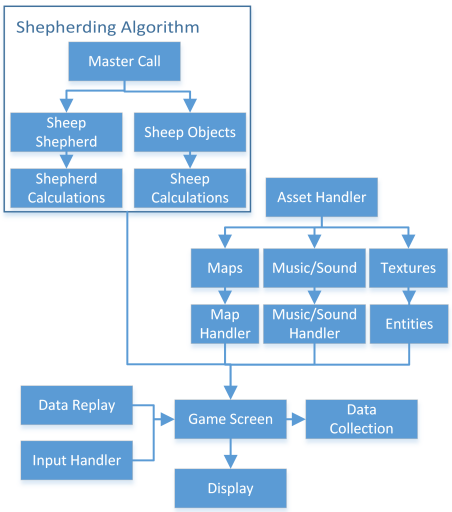

Fig. 3: Class Diagram of the Shepherding Game

and their relative distance to the target. Furthermore, there is also an obstacle detection method, which uses the coordinates from the Map Handler to check and respond if any collision occurs. The Shepherd Calculations class contains methods that determines the attraction and repulsion to other shepherds. These methods involving multiple shepherds are kept if multiple shepherds are used in future work.

The Sheep Calculations class determines the direction of repulsion from other sheep and from the shepherd. Furthermore, the direction of attraction to other sheep and the shepherd is also calculated. From this information, the Sheep Objects class determines the new direction and position of each sheep. Furthermore, the Sheep Objects class also implements obstacle detection similar to the method used for the shepherd. These methods for obstacle detection and avoidance were implemented for the use in the game. From each of these classes, the master call takes the position and direction of each sheep and the shepherd and outputs it to the Game Screen.

The remaining classes focuses on the implementation of the gamification, and as such, are classic gaming classes that we do not discuss here.

V. Game Validation

Testing was performed thoroughly on Cyber Shepherd through the use of the international standard ISO/IEC12207:2017[20] to define the activities of the testing methodology. A bottom-up methodology was implemented to ensure each element of the game functions as required. Firstly, unit testing was performed on all the methods contained within each class of the game architecture. This involved performing a black box test that used sample inputs and analysed each output. Any errors or discrepancies in any method resulted in further white-box testing and analysis.

Integration testing was also performed between all the methods and classes that have an interface within the game architecture. This was performed for every interface within the architecture and any errors resulted in further white-box testing. Subsequent system and acceptance testing was then performed through thorough game-play.

In order to test if the game was designed and developed in line with the requirements set for this paper, each functional component of the game was tested using a number of test cases. Each test case which specified that each level was tested involved the testing of each of the 15 levels 5 times. 5 repetitions was chosen due to the time it takes to collect the data and since the behaviour between repetitions did not vary significantly. The other test cases were tested 10 times to ensure that no bugs or unexpected crashes occur. Ten repetitions were undertaken due to the complexity of these specific subsystems, a negative testing approach was utilised to find any bugs.

The first test case was performed to test the overall experience of the game. Each of the menus were accessed and each button was touched to ensure that the touch screen was reacting correctly and that the button sounds and commands were working as desired.

The second test involved watching the AI perform the shepherding behaviour on every level. The path that the AI took along with its behaviour in collecting and driving the sheep was observed. It was found that the behaviour of the shepherd matched the behaviour expected from Strömbom's behaviour. An example of this test can be seen in Figure 4, which shows the behaviour of the algorithm on level 8. The level involved collecting then driving a single cluster to the target.

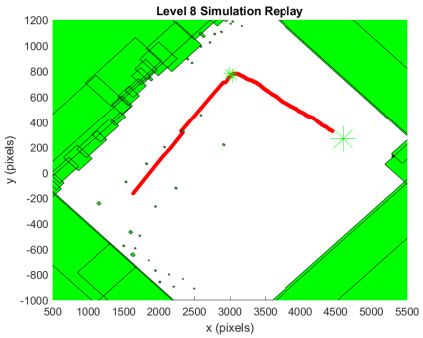

Fig. 4: Path taken by the AI to shepherd a single cluster to the target. The large green asterisks designates the target while the smaller green asterisks designates the initial cluster position.

The third test was performed similarly to the testing of the Shepherding AI, however focus was placed on observing the behaviour of the flock. It was found that the flock's behaviour performed in a way that aligned with Reynold's flock behaviour and as such, was considered to pass. This test was performed on every level to ensure there were no discrepancies in the flocks behaviour.

The fourth test involved testing the data collection and replay components of the game. Since this forms a critical part of future analysis on shepherding models from this paper, significant attention was placed on ensuring it performed without any faults. Each level was played 5 times and each replay was observed and compared to the actual game-play. Since the replay component utilised the data collection data, watching the replay of game-play on each level was sufficient to test both the collection and replay components.

The fifth test involved playing the game and moving the shepherding around all areas that can be accessed within the boundaries of each level. This was performed to ensure that the order of the map layers were rendered correctly and that the player was rendered relative to the layers correctly. Each level was played once with the shepherd moved around each map completely to complete this test case.

The sixth test case involved testing the interaction of both the shepherd and the sheep with the obstacles. This was performed by firstly moving the Shepherding into each obstacle on each level and ensuring no undesired behaviour occurred. After this, the sheep clusters were shepherded and forced to interact with each obstacle on each level as well. This was tweaked and tested multiple times to ensure all undesired behaviour was eliminated. An example of how

this behaviour was observed and analysed can be seen in Figure 5.

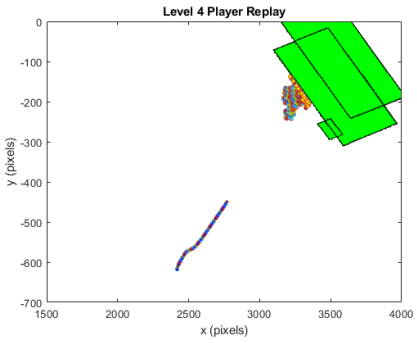

Fig. 5: Test undertaken on level 4 to test sheep and shepherd interaction with obstacles.

The last test case was performed to ensure that the game maintained a playable frame rate on each level and menu screen. The game was connected to a computer and the frame-rate observed in real time. It was found that when levels were played with numbers of sheep greater than 70, the frame rate decreased from 30 to an average of 9 FPS on a Samsung Galaxy S5. Optimisation of the obstacle detection algorithms increased this to 16FPS which was still too low for playability. As such, the number of sheep was kept below a maximum of 50 sheep. To rectify this issue in future work, a more modern Android phone will rectify this performance issue. The shepherding algorithm caused a significant load on the phone and illustrates the potential performance issue on certain devices for Strömbom's algorithm.

Requirements traceability was also conducted to ensure that the test cases were designed in a way to verify the requirements of the game. This was important to ensure that the design of the game still aligned with the goals of the research and reinforces the successful implementation of the requirements. The test case set was designed to trace back to every requirement set in the requirements analysis. Through this, it was found that all functional requirements were successfully implemented and discovered that the performance requirement needs further work.

VI. Game Results

By collecting data from both user and AI game-play, significant amount of data can be collected on the performance of a human versus a particular shepherding model. This information can provide an insight into certain decisions a human and AI might make similarly and any differences that may result. From a significant amount of data that can be collected from the game, a dataset with successful game-play attempts from both humans and AI can be constructed.

To align with this goal of the Shepherding game platform, each level was played by both a human and AI 10 times. The minimum, average and maximum time taken for both the user and the AI was collected and plotted to illustrate the comparative performance of each case. This can be seen in Figure 6. For the levels which only the human was plotted,

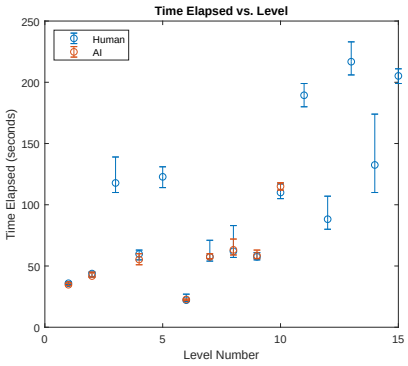

Fig. 6: Minimum, maximum and average time Elapsed for a human and AI in each level.

the AI could not complete the level. This was the case due to either the presence of obstacles or since the AI could not correctly collect and herd the sheep clusters.

It can be seen for levels between 1 and 5 that both the human and AI could complete, the time elapsed was very similar. The minimum and maximum values did not vary significantly. This was the case due to the simplicity of the shepherding tasks in each of these levels. The shepherd was simply required to find the shepherd cluster and drive them to the target in these cases. For levels 3 and 5, there were 2 clusters that first needed to be collected, with level 5 also containing obstacles. In these cases, the AI could not successfully collect the sheep and collided with the boundary for all ten repetitions performed.

For levels 6 to 10, both the human and AI could complete most of the levels excluding level 10. These levels involved finding and driving the single shepherd cluster to the target. The AI had a smaller range of values with an average value almost identical to the human. In the case of level 10, the AI could only complete the task twice out of the 10 repetitions. This resulted in an average value very similar to the human in the cases it did succeed, however it only had a 20% success rate.

Levels 11 to 15 required the shepherd to navigate obstacles such as buildings and fence lines to shepherd the sheep to the target. With these levels, it became evident one of the shortcomings of Strömbom's algorithm, which does not enables the AI to make a decision on how to navigate obstacles. As such, it was found that the AI could not complete any of the levels for the 10 repetitions performed. However, it was clear that the human could perform each level with varying success. A large margin of error for the human can be seen for level 14. This was the case due to the large spread-out area of the map and the varying paths taken to complete the task.

The path taken by both a human and AI to conduct each level successfully can be seen in Figure 7. Each level was played by both the human and AI 5 times. In the levels where only blue paths can be seen, the AI could not complete the level, correlating with the results seen in Figure 7. The large pink asterisks represents the target while the two yellow

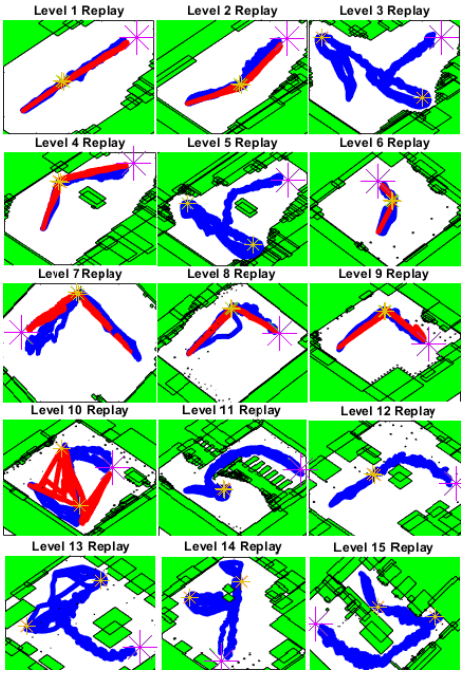

Fig. 7: Human and AI paths taken to successfully the sheep clusters to the target. Blue represents the human path while red represents the AI path.

asterisks represent the initial sheep cluster positions. This graph highlights how the behaviour of both a human and an AI could be compared. In particular, the cases where the AI could not complete a level (i.e. Levels 3, 5 and 11-15) can be analysed. It can be seen for these cases, the shepherd had to interact closely with obstacles and to navigate these to collect and drive the sheep clusters. As such, they highlight the clear shortcoming in Strömbom's algorithm which can be seen to be limited to open paddock environments.

VII. Conclusion

A Shepherding game was developed to be used as a platform to test and collect data on the behaviour of Shepherding models with both AI and human-based shepherding control. Due to a number of gaps in current shepherding models, this game was developed to provide an accurate benchmark of how each model performs in a variety of different scenarios. Furthermore, the data collected from user game play can be used within future work to accelerate the data collection process for areas of research such as reward-based learning and curriculum learning. The results from the game-play of both a human user and the AI in the levels provided within the game show clearly within only a few repetitions of play how the game can highlight any strengths or weaknesses in shepherding models.

References

[1] J. Aalders, "Dog-livestock interactions: Influences on the behavioural responses of sheep to working dogs in the herding environment," 2014.

[2] P. Cowling and C. Gmeinwieser, "Ai for herding sheep," in *Proceedings of the Sixth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*. Bradford: Association for the Advancement of Artificial Intelligence, 2010, pp. 2–7.

[3] C. K. Go, B. Lao, J. Yoshimoto, and K. Ikeda, "A reinforcement learning approach to the shepherding task using sarsa," in *2016 International Joint Conference on Neural Networks (IJCNN)*, July 2016, pp. 3833–3836.

[4] J. Brulé, K. Engel, N. Fung, and I. Julien, "Evolving shepherding behavior with genetic programming algorithms," *CoRR*, vol. abs/1603.06141, 2016.

[5] W. Lee and D. Kim, "Autonomous shepherding behaviors of multiple target steering robots," vol. 17, 11 2017.

[6] K. Fujioka and S. Hayashi, "Effective shepherding behaviours using multi-agent systems," in *2016 IEEE Region 10 Conference (TENCON)*, Nov 2016, pp. 3179–3182.

[7] C. W. Reynolds, "Flocks, herds and schools: A distributed behavioral model," in *Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '87. New York, NY, USA: ACM, 1987, pp. 25–34.

[8] J.-M. Lien and E. Pratt, "Interactive planning for shepherd motion," 2009.

[9] O. B. Bayazit, J.-M. Lien, and N. M. Amato, *Better Group Behaviors Using Rule-Based Roadmaps*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 95–111.

[10] D. Strömbom, R. P. Mann, A. M. Wilson, S. Hailes, A. J. Morton, D. J. T. Sumpter, and A. J. King, "Solving the shepherding problem: heuristics for herding autonomous, interacting agents," *Journal of The Royal Society Interface*, vol. 11, no. 100, 2014.

[11] N. R. Clayton and H. Abbass, "Machine teaching in hierarchical genetic reinforcement learning: Curriculum design of reward functions for swarm shepherding," in *IEEE Congress on Evolutionary Computation*. Wellington, New Zealand: IEEE, 2019.

[12] A. Gee and H. Abbass, "Transparent machine education of neural networks for swarm shepherding using curriculum design," in *International Joint Conference on Neural Networks*. Budapest, Hungary: IEEE, 2019.

[13] Newzoo, "Global games market report," Newzoo," Report, 2018.

[14] S. Deterding, M. Sicart, L. Nacke, K. O'Hara, and D. Dixon, "Gamification. using game-design elements in non-gaming contexts," in *CHI '11 Extended Abstracts on Human Factors in Computing Systems*, ser. CHI EA '11. New York, NY, USA: ACM, 2011, pp. 2425–2428.

[15] A. E. Rizzoli, A. Castelletti, P. Fraternali, and J. Novak, "Demo abstract: Smarth2o, demonstrating the impact of gamification technologies for saving water," *Computer Science - Research and Development*, vol. 33, no. 1, pp. 275–276, Feb 2018.

[16] D. Vara, E. Macías, S. Gracia, A. Torrents, and S. Lee, "Meeco: Gamifying ecology through a social networking platform," in *2011 IEEE International Conference on Multimedia and Expo*, July 2011, pp. 1–6.

[17] K. M. Khalil, M. Abdel-Aziz, T. T. Nazmy, and A.-B. M. Salem, "Mlimas: A framework for machine learning in interactive multi-agent systems," *Procedia Computer Science*, vol. 65, pp. 827 – 835, 2015, international Conference on Communications, management, and Information technology (ICCMIT'2015).

[18] B. Morschheuser, K. Werder, J. Hamari, and J. Abe, "How to gamify? a method for designing gamification," 01 2017.

[19] V. Analytics, "Leveling up your mobile game: Using audience measurement data to boost user acquisition and engagement," Verto Analytics," Report, 2016.

[20] "Systems and software engineering – software life cycle processes," International Organization for Standardization, Geneva, CH, Standard, 2017.