SkyNetz: A Playful Experiential Robotics Simulator

Robin Han, Dominik Auer, Sarah Edenhofer, Sebastian von Mammen
Organic Computing Group, University of Augsburg, Email: {robin.han, dominik.auer}@student.uni-augsburg.de,
{sarah.edenhofer, sebastian.von.mammen}@informatik.uni-augsburg.de

***Abstract*—We introduce SkyNetz, a playful interactive robotics simulator for computer science students. Its focus lies on the visualization of a probabilistic robot localization algorithm considering noise in the sensors and actuators of the robot on the one hand, as well as an environment filled with obstacles which damage the robot on contact on the other hand. The goal of the simulation is training students on the intricacies of the algorithm and to develop a notion on the impact of the considered factors such as the degree of sustained sensory noise. In order to facilitate learning and promote exploration, we embed a game mode that conveys the basic interactions with the simulator and the factors shaping the robot's behaviour. In the game, the player helps the simulated robot to reach its destination with as little damage as possible. This is done by setting waypoints for the robot by adjusting the parameters of the deployed localization algorithm as well as the quality of sensors and the accuracy of the robots movements. By playing with these parameters, the user playfully learns their effects, which are visualized in 3D - contrary to the hard mathematical approach presented in books. SkyNetz also has the capacity to communicate with a real robot, to show its current position and position estimates. In the long run, this will provide the foundation for novel augmented reality games. The paper includes a general introduction to the topic of interactive robot simulation, background on the specific problem of localization estimation, the presentation of our approach and the results from a small user study.**

I. Introduction

At the dawn of the robotic age, a variety of robotics software has been emerging. Robot simulators have become an essential tool in every roboticist's toolbox as building a real robot is costly and performing learning algorithms in the physical world is time consuming. A well-designed simulator makes it possible to rapidly test algorithms, design robots, and perform regression testing using realistic scenarios. Furthermore, all the parameters are displayed on screen, making it easier to keep track of them and additional real-world surveillance sensors are not needed [1]. The version of SkyNetz presented in this paper[1] was engineered for serving as an interactive, playful simulator that teaches its players how different parameters influence the probabilistic localization of a robot. The robot moves autonomously in the xz-plane of a simulated world, trying to reach a target and to avoid obstacles on its way. Any collisions damage the robot, which is used as a gamification element. As in reality, the robot's sensors and actuators need to cope with noise, which makes it impossible to directly and accurately measure its movement. Rather, it has to be estimated based on assumed degrees of noises affecting its perception and actions. The estimation, i.e. the robot's belief, is calculated by a so-called Pseudo Kalman Filter, which we derived from the Kalman Filter using some simplifications. SkyNetz always visualizes the probabilistic belief of the robot alongside its actual position. Now the user needs to make sure that the robot finds its path despite the given complex entangling network of sensors, actuators, noise, obstacles and targets. One way to help the robot get on its way is the strategic placement of waypoints. Another approach is the manual adjustment of algorithmic parameters, or tinkering with its hardware configuration.

The remainder of this paper is structured as follows. Section II presents a number of similar projects towards the simulation of robots and how they inspired the development of SkyNetz. Section III-A gives a sketchy overview on the SkyNetz. Section III-B briefly describes the inner workings of SkyNetz. Next, in Section III-C we describe how the user can interact with the system. Section IV evaluates how well SkyNetz achieved its goal of teaching probabilistic robotics. Finally, Section V summarizes the work and gives a preview towards the further development of SkyNetz.

II. Related Work

Like any other field of engineering, robotics makes heavy use of simulations. Among those, there are some focusing on robot-environment interactions, which are especially interesting in the scope of our research project. USARSim [2] is a high fidelity multi-robot simulator, that constitutes the simulation engine used to run the Virtual Robots Competition within the Robocup initiative [3]. USARSim can be a very effective tool for teaching due to its open source nature and the free availability of the underlying simulation engine for academic purposes [4]. Webots [5] is a professional development environment for building, programming and testing mobile robots widely used for educational purposes. Robots can easily be equipped with sensors and actuators from libraries and then tested in physically realistic worlds. Webots is used by over 1225 universities and research centres worldwide [5]. Sindbad [6] is a Java 3D multi-robots-simulator, developed for scientific and educational purposes in the field of Evolutionary Robotics. It is dedicated to researchers and programmers, providing a neural network library and a framework for evolutionary computing, including concrete implementations of genetic algorithms, evolution strategies and genetic programming.

Aside more serious simulators, there also exist several video games with a similar simulation character. Due to the gamification aspect, these examples were considered during research on how to keep the user engaged into continuing

[1]You can find binaries at www.vonmammen.org.

interaction with the simulation. The video games presented in the following achieve this by providing a series of challenges that the player needs to solve:

Besiege [7] is an upcoming physics based building game in which the player constructs outlandish medieval siege engines to solve challenges. Players select from a collection of mechanical parts that can be connected together to build a machine. Each level has a goal, such as "destroy the windmill" or "kill 100 soldiers". Although the goals are relatively simple, the wide variety of possible approaches allows for exploration and experimentation.

Another inspiring game is the space flight simulator Kerbal Space Program (KSP) [8] to which notable members of the space industry have already taken an interest, including NASA and Elon Musk of SpaceX. The gameplay of KSP consists of constructing spacecraft or aircraft out of a provided set of components and launching them from the in-game space center's launch pad or runway. Then they go on to complete their desired mission while averting catastrophic failure, such as running out of fuel or electricity, or the spacecraft breaking apart due to structural problems, otherwise being unable to succeed. These missions are player set or proposed in the form of "contracts" with set parameters to achieve.

III. SkyNetz: Top to Bottom

In this section, we present an overview over SkyNetz shedding light on the overall look and feel of the simulator, on its components' interplay and the user's opportunities for interaction.

A. Overview

In Fig. 1, a screenshot of the SkyNetz simulator is shown. On the left, a menu is offered to inspect and alter the configuration of the robot and its algorithmic parameters with checkboxes, sliders and text fields to specify exact input values. The 3D simulation is shown on the bottom right and can be controlled through the menu above. In order to keep the objects distinguishable, they are represented by coloured blocks. The robot is displayed in red, the waypoints are yellow and the belief of the robot about its whereabouts is projected into the scene in blue. The more confident the robot is about its position, the narrower is the Gaussian bell. As soon as the belief surface intersects with the robot's current waypoint (yellow), the waypoint is considered passed and marked accordingly (dark yellow), as seen in the waypoint in the background in Fig. 1. Afterwards, the robot proceeds with moving towards the next waypoint. Using the mouse, the user can control the camera to adjust the viewing perspective.

B. Models and Methods

Robot and obstacles are two types of units in the simulation, which are generally represented as axis-aligned boxes. The robot moves across the ground plane towards a predefined target position (waypoint). It does so in three steps: Rotating towards the target, moving towards it and rotating again. The algorithm for motion simulation implemented in SkyNetz is derived from the odometric motion sampling algorithm used for sampling the motion of particles [9]. Collision detection prevents the robot from moving through the obstacles. However, only the final position at the end of every iteration step is checked for collisions, which is sufficient for most parameter settings. But it may cause inconsistencies if the robot's movements between two simulation steps are too big. Internally, the robot maps the sensory information about its location x to a measurement z at time t. Considering noise, the robot supposes that x_t cannot be assumed to represent a deterministic pose of the robot. Rather, it assumes its location to be centred at x_t with Gaussian normal distribution. The resultant belief about its pose is denoted as $bel(x_t)$, whereas the mean of the Gaussian distribution, corresponding to the most probable location, is denoted as μ_t. The standard deviation vector σ_t represents the insecurity in every dimension of the belief. Our simulated robot periodically executes the localization algorithm to maintain $bel(x_t)$. To this end, the robot relies on the accuracy of its sensors and actuators. We implemented a Pseudo Kalman filter to simulate this functionality and predict x_t. This Pseudo Kalman filter calculates $bel(x_t)$ in a two-step process using the previous belief $bel(x_{t-1})$, the control information u_t, which denotes the most probable relative movement, and the measurement z_t. In the first step, the filter makes a prediction, in the second step it incorporates the new measurement depending on a constant called Pseudo Kalman gain to correct the prediction's error. Our simplified version is not mathematically equivalent to an actual Kalman filter but its result is a very close approximation and suffices for the purpose of the given simulator.

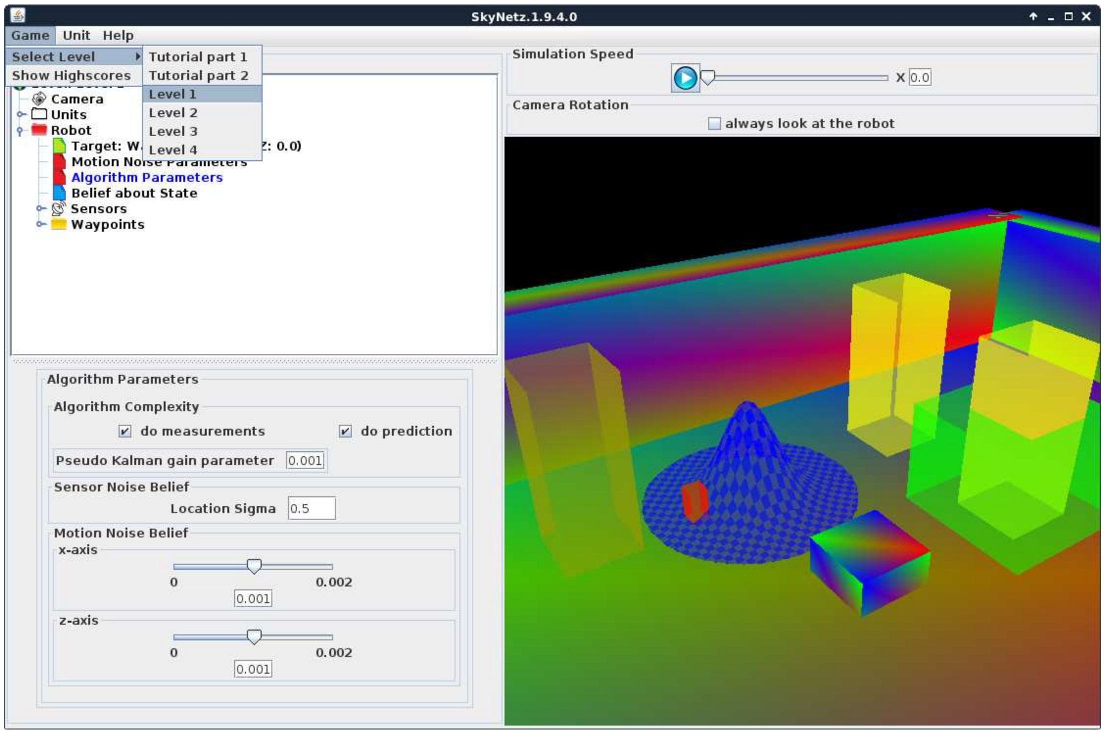
C. User Interactions and Game Mechanisms

Game mode is level-based, with increasing difficulties and different kinds of challenges faced at higher levels.

SkyNetz is split into several levels. The goal of each level is to get the robot (red cube) to its target (green cube), cf. Fig. 1. To do so, the robot will follow its waypoints (yellow cube). The robot does not exactly know where it is, but maintains a belief about where it thinks it is, which is represented by the blue bell. The broader the bell, the more insecure the robot is about its location. In later levels, the user will have to set various parameters, including waypoints.

Currently, SkyNetz consists of a two-part tutorial and four levels. In the first level, there is an obstacle between the robot and the target. To move the robot safely to the target, the user has to place waypoints in order to avoid the obstacle. In the second level, the robot has to make it along a narrow corridor while avoiding collisions. In the third level, the robots GPS-Sensor has a higher error and the player has to find a way to fix that issue. In level 4, the robot must find its way around an obstacle, but has a faulty sensor.

The benchmark for the player's performance is the amount of collisions the robot endures until the target is reached. To achieve a good result, the user is able to interact upon the simulation by adjusting sensor accuracy, motion noise and parameters for the Pseudo Kalman filter. These are the Pseudo



Simulation Speed X 0.0

Camera Rotation ☐ always look at the robot

Algorithm Parameters

Algorithm Complexity ☑ do measurements ☑ do prediction

Pseudo Kalman gain parameter 0.001

Sensor Noise Belief Location Sigma 0.5

Motion Noise Belief
x-axis 0 0.002 0.001
z-axis 0 0.002 0.001

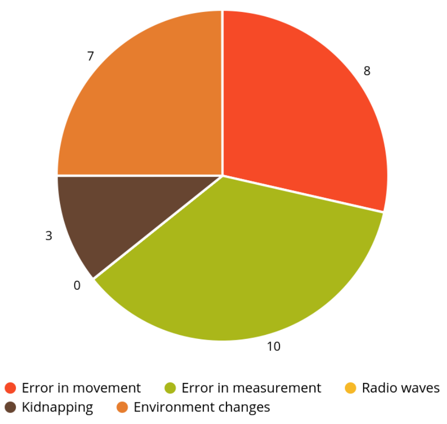
Figure 1: A screenshot of the SkyNetz simulator while in level one. The component inspection tree can bee seen on the left, while the rendered 3D-scene can be seen on the right.

Kalman gain parameter, the belief about motion noise and measurement noise, and whether movement prediction and measurement prediction should be executed. When the user increments the noisiness of the sensor, the robot's belief will be more prone to errors. However, if the user also increments the robots belief about the noisiness of the sensor, the effect is weakened as the Pseudo Kalman Filter knows about the miserable quality of the sensor and will therefore weigh the measurements less. If the user chooses to reduce the belief about the noisiness of the sensor, then Pseudo Kalman Filter will weigh the measurement more and a faulty sensor has a more fatal effect. When increasing only the error of the motion parameters without increasing the belief about the motion error, the robot does not know that his motion is faultier and the Pseudo Kalman filter will still relay on the motion prediction although the measurement has more value in this case. When the belief about the motions error is also increased, the Pseudo Kalman Filter will weigh the measurement more, resulting in a better pose estimation. When motion error is increased too much, the robot will not be able to control its movement anymore and his movements will be random. By deactivating the measurements, the Pseudo Kalman filter will only do motion prediction and its insecurity will rise over time. When deactivating motion prediction, the robot's pose will only be inferred from measurements and therefore the belief will lag behind the robot's true state. Relocating game objects and creating new ones is also allowed in some levels. For example, relocating the robot results in the belief being dislocated from the previous x_t. However, if measurements are activated, the robot can relocate its belief. How fast the robot relocates depends on further parameters like the robot's assumption about the noisiness of the sensor and the insecurity of its position. To reach the target, the robot follows waypoints which can also be modified by the user.

IV. Evaluation of Survey

A survey with twelve anonymous participants was performed in order to test the simulation. The intentions were to measure how well SkyNetz could teach understandings of probabilistic robotics and what improvements could be made. Participants of the survey were instructed to try out SkyNetz and to answer a number of questions afterwards, based on their experience with the simulation.

More than 80% of the participants rated themselves as beginners in the field of probabilistic robotics. Still, they were able to answer subject-based questions mostly correct (cf. Fig. 2a) after playing with the simulation. More than 90% thought that playing SkyNetz improved their knowledge about probabilistic robotics (Fig. 3a). More than 50% of the participants would recommend or strongly recommend SkyNetz (Fig. 4). While the colorful look was praised, 18% of the participants gave a good or very good rating for the user interface (Fig. 5), the GUI was mentioned several times when we asked, which part of SkyNetz needs the most improvement. Particularly the setting of waypoints was considered difficult. In conclusion, SkyNetz is hard to use for people that are not



(a) Answers to the question *"What are the reasons why a mobile robot does not always know where it is?"*. Most subjects were able to answer the question with "Error in measurement" correctly, with many selecting all of the four correct reasons, while none selected the incorrect answer "radio waves of other robots".

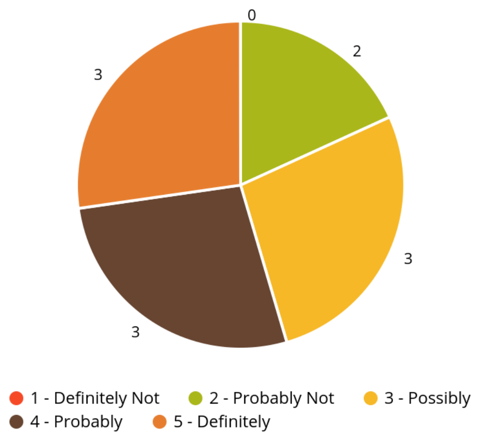
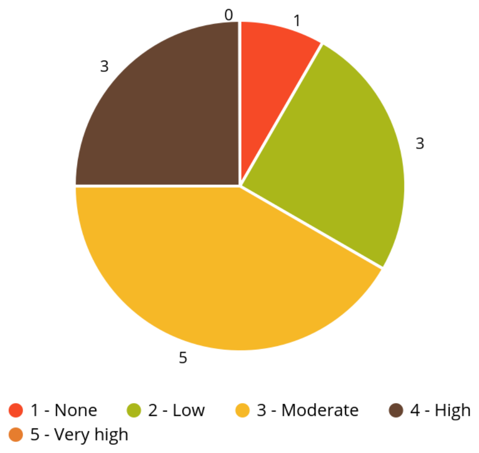


Figure 4: Answers to the question *"Would you recommend playing SkyNetz for people interested in probabilistic robotics?"*. The portion of positive recommendation was higher than the negative/neutral part.



(a) Answers to the question *"How much did playing SkyNetz improve your knowledge about probabilistic robotics?"*. Over 90% could improve with SkyNetz according to their own opinion.

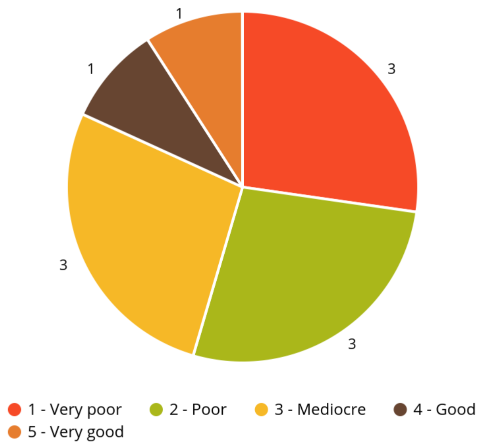


Figure 5: Answers to the question *"How would you rate the user interface?"*. Feedback on the UI showed mixed results.

acquainted with computer science, but it can help to teach students the basics of probabilistic robotics.

V. Summary & Future Work

This article introduced SkyNetz as an interactive probabilistic robotics simulator. It features a motion prediction model based on a pseudo implementation of the Kalman Filter. The robot can localize itself on the map without being damaged, but still the implementation of a correct Kalman Filter would improve the didactic value of the simulation. Users get a better understanding of probabilistic robotics by using the interactive simulation, so computer science students interested in probabilistic robotics should be encouraged to use it, while people not familiar with the topic might be overwhelmed by the complexity. Therefore, a better tutorial, which highlights GUI-elements and explains them in more detail, should be implemented. Furthermore, a mechanism to drag and drop waypoints inside the rendered 3D-scene would improve the usability.

References

[1] O. Michel, "Webots: Professional mobile robot simulation," *Journal of Advanced Robotics Systems*, vol. 1, no. 1, pp. 39–42, 2004.

[2] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: a robot simulator for research and education," in *Robotics and Automation, 2007 IEEE International Conference on*, April 2007, pp. 1400–1405.

[3] ——, *RoboCup 2006: Robot Soccer World Cup X*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, ch. Bridging the Gap Between Simulation and Reality in Urban Search and Rescue, pp. 1–12.

[4] EpicGames, "Unreal engine," http://www.unrealengine.com, accessed: 2016-03-17.

[5] Webots, "Webots," http://www.cyberbotics.com, accessed: 2016-01-14.

[6] L. Hugues and N. Bredeche, *From Animals to Animats 9: 9th International Conference on Simulation of Adaptive Behavior, SAB 2006, Rome, Italy, September 25-29, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, ch. Simbad: An Autonomous Robot Simulation Package for Education and Research, pp. 831–842.

[7] S. Games, "Besiege," http://www.besiege.spiderlinggames.co.uk/, accessed: 2016-01-19.

[8] Squad, "Kerbal space program," https://kerbalspaceprogram.com/en/, accessed: 2016-01-19.

[9] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005.