GroundSim: Animating Human Agents for Validated Workspace Monitoring

Kim Wölfel, Tobias Werner, and Dominik Henrich

Chair for Robotics and Embedded Systems,
Universität Bayreuth, D-95440 Bayreuth, Germany,
kim.woelfel@uni-bayreuth.de,
http://robotics.uni-bayreuth.de/

Abstract. In the promising field of human-robot cooperation, robot manipulators must account for humans in the shared workspace. To this end, current prototypes integrate various algorithms (e.g. path planning or computer vision) into complex solutions for workspace monitoring. The step from research to industrial use for these solutions demands rigid validation of the underlying software with real-world and synthetic data. Related fields (e.g. human factors and ergonomics) implement toolsets to create synthetic data of human-machine interactions. However, existing toolsets employ hand-crafted motion paths or motion segments for their human agents. This limits the variety of resulting motions and implies laborious composition of animation sequences. In contrast to this, we contribute a novel approach to human animation for synthetic validation: We animate our human agents through a realistic physics simulation and we expose motion paths in a flexible and intuitive high-level editing interface. We also generate photo-realistic images of resulting animations through state-of-the-art rendering techniques. Finally, we employ these synthetic images and their ground-truth backing to validate a prototype for a workspace monitoring system and a subsequent online path planner.

Keywords: workspace monitoring, virtual environments, ground-truth testing, synthetic data, physically-based animation of humans, computer vision, robotics

1 Introduction

Traditional industrial manipulators execute automated tasks in isolated robot cells. However, the recently emerging field of robot-human cooperation envisions a shared workspace for robots and humans. Robots, particularly, must perceive human agents and other a-priori unknown objects in the robot cell in real-time to determine appropriate on-line reactions. Prototypical monitoring solutions for the shared human-robot workspace (e.g. [23]) suggest algorithms of computer vision to derive workspace occupation through varied sensor data. Subsequent stages for collision checking and path planning can incorporate the reconstructed workspace occupation to find a suitable reaction for the robot manipulator. Example reactions include speed control or adjusting the path of the robot.

Both monitoring systems and robot reactions must be validated thoroughly for the step from research to real-world applications. Full validation obviously requires real-world data. However, real-world data is not without its drawbacks. Acquiring extensive real-world data is time-consuming and error-prone, the robot cell must be available and remains occupied during tests, dummy workers and example obstacles are required, and finally no ground-truth data exists for test automation (e.g. for unit and integration tests).

Synthetic testing data promises to solve the above problems of real-world data at the cost of fidelity. The significance of validation increases with the fidelity of synthetic testing data. To test system limits, fidelity is particularly relevant for traditional failing points of computer vision: On the one hand, texturing and lighting of the rendered scenes must be photo-realistic for meaningful validation of background subtraction and segmentation. On the other hand, movement of human agents must be lifelike for meaningful validation of time-coherent obstacle detection (e.g. for approaches with tracking or online learning).

Traditional solutions use off-the-shelf rendering and animation techniques such as basic Phong lighting and animations stitched together from motion-captured segments. Opposed to this, our contribution excels in two points: We combine state-of-art real-time rendering (e.g. shadow mapping, normal mapping) with a physics simulation to drive realistic human movement. Additionally, our toolset offers a high-level and intuitive way to specify flexible motions for human workers without relying on rigid motion captures or laborous motion stitching.

2 Related Work

Over recent years, robotics simulators have become widespread. Such simulators enable the user to design and validate robot applications in virtual environments, including simulated sensors such as depth cameras or laser scanners. Some of the simulators (e.g. RoboDK [18]) focus on the simulation of robotic manipulators, partially even without a graphical user interface (GUI) (e.g. [11]). Other simulators try to cover a wide variety of robot types (e.g. Actin [1], Gazebo [10], UARSim [3], OpenRAVE [5], V-Rep [7], Webots [16]).

Closest to our contribution are simulators that offer a specific set of features: simulated physics with collision detection, photo-realistic rendering, animated human agents, a graphical user interface, and virtual sensors. Consequently, we chose Gazebo, V-Rep and UARSim for an in-depth review.

The Gazebo framework is a comprehensive simulator which is well established in the scientific community. Gazebo supports multiple high-performance physics engines (e.g. ODE [14] and Bullet [2]), it utilizes the Open source 3D GRaphics Engine (OGRE [15]) for realistic rendering and it includes virtual sensor models with or without simulated noise. Finally, Gazebo animates human agents based on predefined joint trajectories. Our contribution, in contrast, generates human movements at runtime based on a physics simulation.

The V-Rep framework is another toolset for validating robotics applications. V-Rep offers fast collision detection and a choice of four physics engines (Bullet,

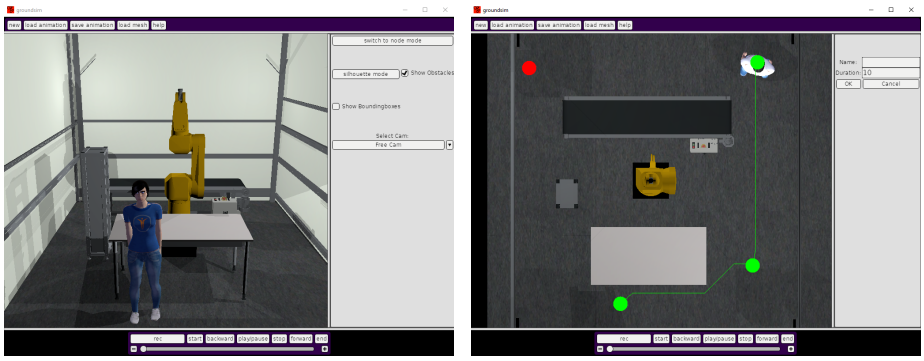

Fig. 1: GroundSim toolset: Sensor View (left) and Edit View (right)

ODE, Newton [13] and Vortex Dynamics [20]) to simulate real world rigid-body physics. The integrated rendering engine performs photo-realistic rendering with soft shadows. Virtual sensor models and a graphical user interface round out the features of V-Rep. Opposed to our contribution, V-Rep animates human agents by combining predefined motions.

The UARSim framework is a lightweight robotics simulator for research and education. UARSim builds upon the Unreal Engine, it offers virtual sensors, and it comes with a graphical user interface. Finally, UARSim can simulate human agents, but does so by means of predefined motion trajectories.

3 System Overview

This section describes the main aspects of our toolset. We then continue to highlight features of the photo-realistic rendering component, we discuss advantages of working with synthetic data, and we explain our approach to the physics-based animation of human agents.

3.1 Overview and Main Features

The GroundSim toolset serves as a **Sim**ulator for generating **Ground** Truth Data. It enables rapid validation of robotics applications. We are particularly concerned with applications that perform online monitoring of shared workspaces for adaptive path planning. To this end, GroundSim allows users to intuitively and effortlessly generate animation sequences of virtual human workers inside a robot cell. Photo-realistic rendering of these animation sequences subsequently produces virtual sensor input (e.g. as a simulated multi-camera system). Finally, combining virtual sensor input with available ground-truth backing data supports the thorough validation of software components, including online monitoring and path planning.

Our approach differs from other toolsets in how users choose the motions of human workers in the robot cell. We do not use predefined motion trajectories, but calculate motions on the fly with a physics simulation. Our graphical user

interface (see Fig. 1 (left)) enables users to specify motion sequences for workers in a robot cell through worker positions, through per-position behaviours, and through behavioural segments connecting the positions.

Our toolset operates in two different modes: *Sensor Mode* and *Editing Mode*. When operating in sensor mode (see Fig. 2 (left)), the center viewport shows a realistic rendering of the current workspace, including human agents and the robot manipulator. While arbitrary camera movement is possible, we also have fixed camera presets that mimic a real-world multi-camera setup for intuitive preview of later output. Additional features include changing object visibility, switching between photo-realistic and silhouette rendering modes (see Fig. 2), and toggling the display of obstacle bounding boxes.

In editing mode, the virtual camera of the center viewport is fixed to a bird's-eye view (see Fig. 1 (right)). Users are then able to define an animation path for the human worker by using two kind of animation primitives, *animation nodes* and *animation segments*. In particular, users place animation nodes both at positions where the human worker should interact with the environment and at goal positions for walking animations. Our toolset automatically links all consecutive animation nodes with animation segments. The planning algorithm for animation segments divides the floor into small, square tiles, thereafter adds the scene objects as obstacles, and finally executes the A* algorithm [9] to find a collision-free path for the human worker.

The resulting path p consists of a set N of animation nodes n_i, connected by a set S of animation segments s_j (see Fig. 1 (right)), where the count of animation segments c_s depends on the count of animation nodes c_n, i.e. $c_n = c_s + 1$. Users can assign a set of behaviours B_{node} or B_{segment} to each primitive of p, including behaviours such as walking, waving a hand, picking up an object, or shoving an object with the foot.

For animation nodes, we offer reasonable behaviours,

$$B_{\text{node}} \subset \{b_{\text{stand}}\} \times \{\emptyset, b_{\text{wave\_hand}}, b_{\text{kick}}, b_{\text{pick\_up}}, b_{\text{drop}}, ...\}, \tag{1}$$

while for animation segments, we expose different behaviours,

$$B_{\text{segment}} \subset \{b_{\text{walk}}, b_{\text{run}}, ...\} \times \{\emptyset, b_{\text{wave\_hand}}, b_{\text{drop}}, ...\}. \tag{2}$$

Individual behaviours are physically grounded due to the use of a physics-based animation kernel (see [4]). When generating the animation, we merge individual behaviours of each animation primitive to an overall action. This may, for instance, result in an animation sequence that combines walking with waving the hands.

3.2 Rendering Engine

For the photo-realistic visualization of animation sequences, we apply a custom rendering engine. Our engine [21] follows a command-driven architecture, where C++11 shared pointers manage opaque IDs over data of naive rendering APIs. Notably, client applications such as our toolset can issue rendering commands

Fig. 2: Photo-realistic (left) and silhouette (right) rendering modes.

(e.g. to update graphics resources or to compose virtual scenes) to data IDs from arbitrary client threads. This enables convenient integration of additional software components such as monitoring systems and path planners for synthetic, but still online validation. Supported back-ends include a photo-realistic OpenGL 3.2 path, a fallback OpenGL 1.1 path, and an NVIDIA Optix GPU ray-tracing path. The OpenGL 3.2 path features state-of-art rendering techniques such as deferred lighting (see [8]), shader-based normal mapping (see [12]), and smooth shadows through shadow mapping (see [6]). Opposed to game-centric engines (e.g. OGRE), our rendering kernel efficiently can exchange data with other software components. For example, rendering to CUDA-capable offscreen back-buffers enables overhead-less coupling with fast GPU implementations of computer vision algorithms. Finally, we use a custom GUI module to visualize arbitrary 3D viewports, for example to realize a preview of all virtual sensors.

3.3 Advantages

The advantages of our contribution for the validation of robotics applications are twofold: We supply realistic, but synthetic data instead of real-world data and we simulate characters based on a physics simulation. With respect to application validation, synthetic data surpasses its real-world equivalent in notable cases. We discuss select cases in the following, alongside respective advantages of our toolset. See Table 1 for a summary of our discussion.

We first consider validating the *workspace occupation* as an output of the monitoring system. Depending on the later application, monitoring systems build subsymbolic (e.g. voxels) or symbolic (e.g. transforms, poses) representations of workspace occupation. In real-world validation ground-truth data is not available implicitly for either kind of representation. Existing approaches thus measure or derive approximate ground-truth by limited, laborous, invasive, or expensive procedures, such as laser scanning or high-fidelity motion capturing. Opposed to this, synthetic data as generated by our toolset comes with an implicit, effortless, inexpensive and exact ground-truth: Positions, orientations and geometric meshes of obstacles and humans are readily available. Finally, we can opt to

Table 1: Comparison of real-world and synthetic data

Trait	Real-World Data	Synthetic Data
Workspace Occupation	not exact	ground-truth
Segmentation	not exact	ground-truth
Sensor Calibration	time consuming	fast and optional
Workcell Access	necessary	not necessary
Time	fixed	speed-up possible
Coverage	manual	permutations and fuzzying
Repeatability	hardly possible	possible
Modification	hardly possible	possible
Hazardous Scenarios	no	possible

perform *camera calibration*, or we can use the implicitly known, ground-truth camera parameters to correlate workspace occupation with later path planning.

Foreground-background *segmentation* [17] constitutes a rather sensitive part of monitoring systems. Therefore, the individual validation of respective software components is worthwhile. Since there is no exact ground-truth for real-world workspace occupation, the standalone validation of segmentation proves difficult. Options include hand-labeling of chosen images with an optional pass through a slow, but high-precision classifier (e.g. AdaBoost on select image features) for the remaining images. Our contribution, however, can directly generate exact ground-truth segmentations from ground-truth workspace occupation.

Apart from ground-truth concerns, synthetic data also carries advantages in usability: For instance, our toolset allows to validate robotics applications without *access* to the respective workcell. This includes off-site validation or the a-priori validation of applications at the planning stage of a robotics facility. Neither of these scenarios is possible with real-world data.

Its algorithmic nature gives our toolset further advantages. Most notably, the *time* needed to perform validation or to generate a validation sequence scales with available hardware. This is not possible with real-world validation data, which demands fixed time to record a single movement sequence. To verify our claim, we measured an example speed-up using our toolset on a fixed hardware configuration, an Intel Core i7 with 16 GB RAM and an NVIDIA GeForce GTX 1060 graphics board. Given a set of objects, workers and behaviours, we are able to estimate how many synthetic data sets our framework can generate in the same time a single real-world run would need. Table 2 sums our results. Notably, at a rate of 10 Hz, we can generate six synthetic multi-camera sequences in the same time a single real-world sequence takes to record. Finally, please note that our experiment does not consider the (negligibly small) time required to start the toolset and to place a few animation nodes for the human worker.

Closely related to time benefits are *coverage* benefits: Due to the synthetic and physically-based nature of our animations, we can easily generate additional validation sequences. Examples include permutation testing of behaviours over a fixed path or fuzzy testing with randomized animation nodes, segments, and

Table 2: Time benefits of synthetic validation data

frequency [Hz]	5	10	30	60
speed-up to real-time	13.2	6.6	2.2	1.1

behaviours. With real-world data, each of these automated runs would require a separate recording pass in the workcell.

Repeatability and *Modification* are two closely related points of concern: In the real world, exactly repeating a recorded motion sequence is mostly impossible and it is likewise difficult to partially adapt a real-world motion sequence after recording. However, both of these features are useful for validation, e.g. for testing occlusions or different camera positions. Our toolset, in contrast, can readily repeat motion sequences and partial motion segments can intuitively be adjusted. Finally, validation on synthetic data avoids certain risks of real-world validation. *Hazardous* cases, in particular, must be validated rigorously (e.g. to avoid human-robot collisions), which is not an option for real-world data.

Animating virtual characters with a physics simulation has several benefits when compared to an animation based on motion captures. One major drawback of motion captured animation is that motion retargeting is difficult even for slight adjustments to the original recording. Consequently, pre-defined motion trajectories either have to be fine-tuned by hand in a time-consuming process, or motion trajectories must be re-recorded for the desired motion. Algorithms for physics-based human animation instead are able to handle walking on uneven terrain [24] and can adjust their motions to grasped objects [4]. In the end, we decided to utilize the approach of [4] because of possible adaption to changes in the topology of the animated character, in obstacles and in grasped objects.

4 Experimental Evaluation

In this section, we validate our toolset for validation: We generate an animation sequence of a human worker in a workcell with our toolset and pass this sequence into a prototype monitoring system for shared human-robot workspaces.

4.1 Monitoring and Path Planning

The choice of monitoring and path planning systems distinctly enforces the form of validation data. Thus, we provide a short overview over both systems we wish to validate with our toolset. Our monitoring system (see [22]) accepts incoming images from an intrinsically and extrinsically calibrated multi-camera network. From these images, we derive a foreground-background segmentation (optionally, as non-binary confidences). We then merge resulting silhouettes over all cameras through an efficient and precise variant of the visual hull to derive a conservative approximation of workspace occupation. The subsequent path planning system (see [23]) incorporates multiple collaborating blackboard agents. These agents evaluate the approximated workspace occupation to suggest either collison-free or risk-minimized paths for the robot manipulator.

4.2 Validation Data

Our example animation consists of a human worker who starts at the front-right corner of the robot cell and walks to the back-right corner while waving her hand. We effortlessly designed the animation with our toolset and rendered the animation both as photo-realistic and silhouette images through a virtual multi-camera system of eight ceiling-mounted Full HD cameras. We then fed resulting images into the monitoring system and our path planning system. Finally, we compared monitoring results with exact ground-truth to evaluate monitoring.

4.3 Results

Over the course of the validation sequence, the monitoring system builds an approximation of the workspace occupation. We start by investigating effects of occlusions and conservative reconstruction compared to implicit ground-truth for human workers and other obstacles. To this end, we pass synthetic ground-truth silhouettes into the monitoring system. This procedure causes the approximated occupation to exceed the ground-truth occupation by an average of about 32% in volume. Depending on exact positions of human and robot, this occasionally makes the path planner discard occluded but otherwise viable path suggestions. Average movement times of the robot increase by an estimated 13% because occlusions occur mainly outside the usual robot paths. As a consecutive step, we feed photo-realistic virtual sensor images into the monitoring system. The monitoring system then performs foreground-background segmentation on the incoming images. Respective errors cause false-negative and false-positive classification of workspace volumes. While workspace volumes grow by only 8%, robot paths slow down considerably with a 19% decrease in robot speed. This relates to the fact that noise artifacts, unlike occlusions, appear at dislocated positions in the robot workspace and have a more prominent impact on path planning. One can thus conclude that suppressing noise (e.g. by knowledge refinement) is more important than reducing occlusions (e.g. through camera optimization). Finally, note that both evaluations and the subsequent conclusion have only been possible due to availability of implicit ground-truth for synthetic test data.

5 Conclusion

In the preceding, we have presented our contribution towards validating robotics applications. Our toolset facilitates generating human animations: Users define animations intuitively by placing animation nodes in a workcell. Our toolset then automatically finds connecting animation segments and allows users to specify per-node or per-segment behaviour for human agents. Together with implicit ground-truth, the resulting, photo-realistically rendered and physically-based animations act as input to virtual validation of monitoring systems and path planners. We have discussed the advantages of our process, including a speedup over real-world validation, implicit and exact ground-truth, and incremental adjustments to existing motion sequences.

In future work, we intend to increase the number of characters available per simulation to model collaboration between multiple humans and robots. Other goals include support for a wider variety of simulated sensors (e.g. for scenarios as in [19]), virtual online validation, and more rigid automated validation.

Acknowledgements
This work has partly been supported by the Deutsche Forschungsgemeinschaft (DFG) under grant agreement He2696/11 SIMERO.

References

1. Actin Software. www.energid.com/software/actin-simulation
2. Bullet Physics Library. www.bulletphysics.org
3. S. Carpin. USARSim: a robot simulator for research and education. ICRA, 2007.
4. S. Coros, P. Beaudoin, M. Van de Panne. Generalized biped walking control. ACM Transactions on Graphics 29.4, 2010.
5. R. Diankov, J. Kuffner. Openrave: A planning architecture for autonomous robotics. Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-08-34 79, 2008.
6. W. Donnelly, A. Lauritzen. Variance shadow maps. Interactive 3D graphics and games. ACM, 2006.
7. M. Freese et al. Virtual robot experimentation platform v-rep: A versatile 3d robot simulator. Simulation, modeling, and programming for autonomous robots, 2010.
8. S. Hargreaves, M. Harris. Deferred shading. Game Developers. Vol. 2. 2004.
9. P.E. Hart, N.J. Nilsson, B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths. Systems Science and Cybernetics SSC4, pp. 100-107, 1968.
10. N. Koenig, A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. IROS, 2004.
11. S. Lemaignan et al. Human-robot interaction in the MORSE simulator. HRI, 2012.
12. T. Möller, E. Haines, N. Hoffman. Real-Time Rendering, Third Edition. ISBN 9781568814247, AK Peters/CRC Press, 2008.
13. Newton Dynamics. www.newtondynamics.com
14. Open Dynamics Engine (ODE). www.ode.org
15. Open Source 3D Graphics Engine (OGRE). www.ogre3d.org
16. M. Olivier. Cyberbotics Ltd. Webots: professional mobile robot simulation. Advanced Robotic Systems 1.1, 2004.
17. M. Piccardi. Background subtraction techniques: a review. Systems, Man and Cybernetics, Vol. 4, 2004.
18. RoboDK. Sim. for industrial robots and offline programming. www.robodk.com
19. J. Thieling, J. Rossmann. Virtual Testbeds for the Development of Sensor-Enabled Applications. Montage Handhabung Industrieroboter. Springer Vieweg, Berlin, Heidelberg, pp. 23-32, 2017.
20. Vortex Studio. www.cm-labs.com/vortex-studio
21. T. Werner. Integration of an Interactive Raytracing-Based Visualization Component into an Existing Simulation Kernel. Master Thesis. University Bayreuth. 2011.
22. T. Werner, D. Henrich. Efficient and precise multi-camera reconstruction. Distributed Smart Cameras. ACM, 2014.
23. T. Werner, D. Henrich, D. Riedelbauch. Design and Evaluation of a Multi-Agent Software Architecture for Risk-Minimized Path Planning in Human-Robot Workcells. Montage Handhabung Industrieroboter. Best Paper Award.
24. J. Wu, Z. Popovic. Terrain-adaptive bipedal locomotion control. ACM Transactions on Graphics (TOG) 29.4, 2010.