Electronic Letters on Computer Vision and Image Analysis 12(2):40-53, 2013

Generation and Rendering of Interactive Ground Vegetation for Real-Time Testing and Validation of Computer Vision Algorithms

Nico Hempe* and Jürgen Rossmann* and Björn Sondermann*

* *Institute for Man-Machine Interaction, RWTH Aachen University, Aachen, Germany*

Received 22th Oct 2012; accepted 4th Mar 2013

Abstract

During the development process of new algorithms for computer vision applications, testing and evaluation in real outdoor environments is time-consuming and often difficult to realize. Thus, the use of artificial testing environments is a flexible and cost-efficient alternative. As a result, the development of new techniques for simulating natural, dynamic environments is essential for real-time virtual reality applications, which are commonly known as Virtual Testbeds. Since the first basic usage of Virtual Testbeds several years ago, the image quality of virtual environments has almost reached a level close to photorealism even in real-time due to new rendering approaches and increasing processing power of current graphics hardware. Because of that, Virtual Testbeds can recently be applied in application areas like computer vision, that strongly rely on realistic scene representations. The realistic rendering of natural outdoor scenes has become increasingly important in many application areas, but computer simulated scenes often differ considerably from real-world environments, especially regarding interactive ground vegetation.

In this article, we introduce a novel ground vegetation rendering approach, that is capable of generating large scenes with realistic appearance and excellent performance. Our approach features wind animation, as well as object-to-grass interaction and delivers realistically appearing grass and shrubs at all distances and from all viewing angles. This greatly improves immersion, as well as acceptance, especially in virtual training applications. Nevertheless, the rendered results also fulfill important requirements for the computer vision aspect, like plausible geometry representation of the vegetation, as well as its consistence during the entire simulation. Feature detection and matching algorithms are applied to our approach in localization scenarios of mobile robots in natural outdoor environments. We will show how the quality of computer vision algorithms is influenced by highly detailed, dynamic environments, like observed in unstructured, real-world outdoor scenes with wind and object-to-vegetation interaction.

Key Words: Computer Graphics, Modelling of Natural Phenomena, Stereo Vision, Development in-the-loop.

1 Introduction

Ground vegetation like grass and shrubs is an essential element of natural outdoor scenes, due to the fact, that it covers huge landscape areas. When thinking of virtual outdoor environments, realistically looking and

Correspondence to: <hempe@mmi.rwth-aachen.de>

Recommended for acceptance by <Angel Sappa>
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autonoma de Barcelona, Barcelona, Spain

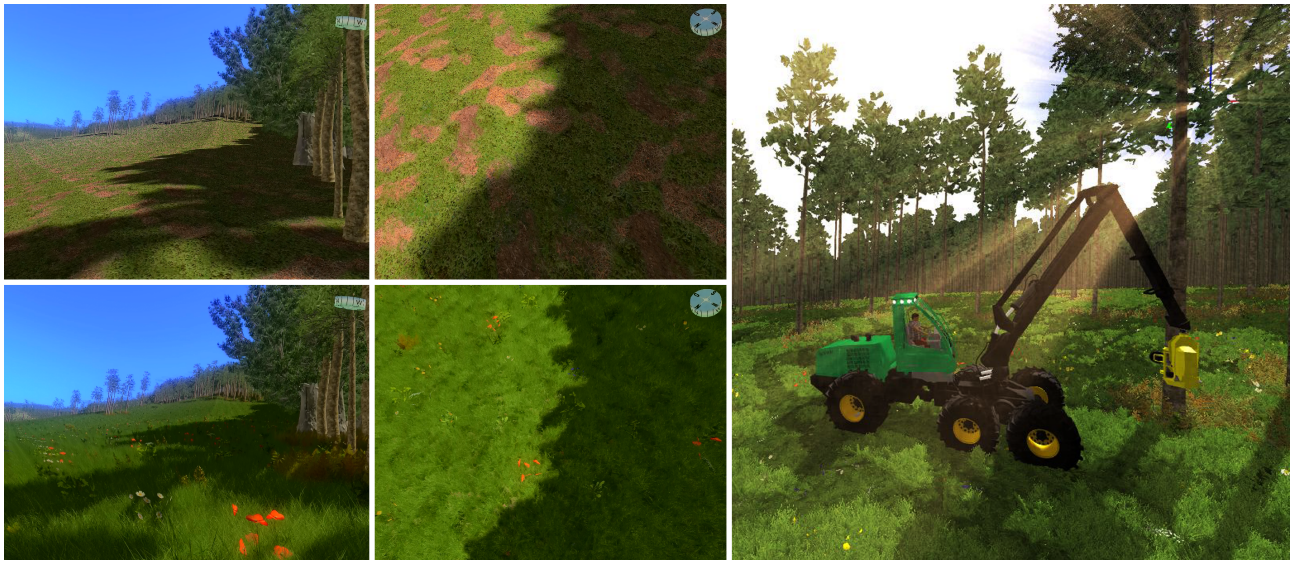

Figure 1: Left: Screenshots of different views of a large outdoor scene taken with and without our ground vegetation rendering approach. Right: Screenshot taken from our current wood harvester virtual training simulation.

interactive gound vegetation greatly improves immersion. Especially in virtual training simulations, a high level of immersion can drastically improve acceptance of the trainees. However, the rendering and animation of realistically looking, interactive ground vegetation is still a challenging task in current real-time simulation systems as it consists of innumerable amounts of single structurfilies. Modelling each individual blade requires a huge modelling effort and geometry, making it impossible to apply it for real-time ground vegetation rendering of arbitrary sceneries. Up to now, various grass simulations have been developed that deliver realistic results including wind and interaction at acceptable frame rates. They mostly suffer from high modeling effort, difficult parameterization or limited kinds of vegetation, like grass only. Moreover, they are hardly applicable for large scenes due to performance reasons. Thus, ground vegetation is usually ignored and represented by simple ground textures. However, ground vegetation is a highly dynamic element, thus, a representation by a static ground texture is unsatisfactory in many application areas.

In a recent paper, we introduced a novel ground vegetation rendering approach that is able to render and generate interactive ground vegetation at large scenes with excellent performance [1] and low modelling effort. Furthermore, our approach also features object-grass interaction with a hugh amount of objects for the whole scene. We focused on the integration of our approach into an existing, complex real-time VR simulation system to improve the visual appearance in virtual training simulations. The left part of figure 1 shows screenshots of a large outdoor scene. The top row was rendered with flat ground textures, the bottom row uses our ground vegetation rendering approach. The right part shows a screenshot of our current virtual forestry work machine simulation. It becomes clear, that realistic rendering of ground vegetation greatly improves immersion, as well as acceptance of virtual environments. Nevertheless, the realistic rendering of natural outdoor scenes can also be beneficial in other application areas not directly related to real-time rendering, which are tackled in this contribution.

Regarding computer vision, real-world testing and evaluation of newly developed algorithms is hardly applicable, expensive or time consuming in many cases. Thus, these algorithms are mostly tested under laboratory conditions or with pre-recorded video clips [18]. Interactive ground vegetation is mostly ignored under laboratory conditions leading to synthetic results while the usage of pre-recorded video clips can lead to unpredicted results when applied to real scenes due to changing conditions. Therefore, a dynamic approach provided by Virtual Testbeds is preferable. In addition to the simulation of landscapes and virtual worlds the current challenges are the precise simulation of various technical components like sensors and their integration as well as

the validation of the developed components [20]. Seen from the computer vision aspect, the realistic simulation of digital cameras and their specific properties like ramdom noise or geometrical distortion is also part of our Virtual Testbed [22]. Such a camera simulation builds the basis for the usage of virtual scenes for computer vision purposes.

The generated results of our ground vegetation rendering approach also fulfill important requirements like plausible geometry representation of the vegetation as well as its consistence over the entire simulation. Thus, it can also be applied for improving and real-time testing of computer vision algorithms of mobile robots in natural outdoor environments. Time-consuming real-world testing can be reduced and the flexibility of virtual scenes allows for testing in various different scenarios right from the beginning of the development process.

In this contribution, we apply our approach on computer vision algorithms in Virtual Testbeds and show the importance of realistic ground vegetation simulation in the field of mobile outdoor robotics. In section 2, related work regarding ground vegetation rendering and virtual testbeds for computer vision is presented. In section 3, our ground vegetation rendering approach is introduced. Section 4 focuses on the application of computer vision algorithms and shows the importance of realistic ground vegetation. Different testing scenarios, real and virtual ones, using the SeekurJr. mobile robot are shown and the results are compared. Finally, section 5 concludes the presented work.

2 Related Work

In recent years, many different techniques were developed that focus on the rendering and animation of a huge number of plants required for grass and shrubs rendering in large natural sceneries. These techniques can roughly be categorized into two basic approaches; volumetric representations and billboard-based techniques. Shell texture techniques [6] are one example of volumetric approaches, which create the illusion of volumetric grass fields. Multiple layers of transparent shells that represent the body of the grass blade are stacked and translated along the surface normal. By including wind vectors to the translation, wind animations can be simulated. However, collision handling is difficult to apply and the shell structure is clearly noticeable when viewed up close. Due to the specific look of the results, this technique is mainly used for fur and hair rendering [7]. Real volumetric textures are used by Decaudin [13], who assembled texture slices to form a volumetric dataset of a small section of forest at different locations and composites them together using traditional volume rendering methods. To apply realistic lighting to these techniques, ray tracing based algorithms are required, which are computationally expensive.

Thus, the most common way to currently represent grass and shrubs with a good balance between realism and performance is achieved by billboard-based approaches using textures of several grass blades. The transparency in the texture enables the representation of objects with complex shapes such as plants and shrubs. In contrast to volumetric techniques, textured quad approaches can efficiently be animated and rendered. Pelzer [12] used two perpendicular polygons containing a section of grass to render large fields of grass with simple wind animation, achieving excellent performance and realistic appearance. This approach was extended to whole landscape rendering by Behrendth et al. [11], who used billboard clouds to represent complex objects.

To achieve a natural appearance of the grass fields, various animation strategies have been developed. While the approach presented by Pelzer [12] uses procedural primitive animation, Ramraj [8] introduced a simple wave effect to simulate wind. Wang et al. [9] integrated explicit grass-grass collision detection to allow for interactive wind animations. The grass animation was further improved by Banisch and Wuthric [10], who used a spring-mass model to simulate realistic grass movement behaviour. A grass simulation approach capable of handling object-grass interaction and collision was introduced by Orthmann et al. [4]. Their approach features complex collision handling with the exact collider geometry, thereby resulting in a very realistic collisions reaction of the grass objects. Furthermore, they subdivided the grass objects into several regions, that are animated using a spring mass model. The collision animation and geometry updating is performed on the GPU resulting a a good overall performance for small and medium sized scenes.

Virtual Reality has been used for virtual prototyping for a long time. Only recently, modern graphics hardware and new rendering approaches provide the possibility to generate highly detailed and interactive outdoor scenes, which are perfectly suitable for testing computer vision algorithms. However, just a few low detailed, rendered image sequences are currently available to the public for computer vision benchmarking. Hence, these sequences are of limited suitability for effective testing of high level computer vision applications such as driver assistance or robot navigation applications [18]. Nevertheless, image data from synthetic environments is extremely important for the evaluation and the comparison of computer vision algorithms, such as stereo matching and optical flow, as a simulation system can also provide the corresponding ground truth [18]. Ground truth and the ability to run a large number of test cases is necessary for tuning and validating vision systems even before deployment [17]. Furthermore, the test cases have to be repeatable in every simulation step, so that the results of two different test cases can be compared afterwards [20].

3 Ground Vegetation Simulation

Different approaches for grass simulation and rendering have been developed so far providing individual strength and weaknesses. Our ground vegetation simulation focuses on the practical and flexible application in large sceneries. The application areas include virtual training simulation as well as computer vision aspects. Thus, the following development goals need to be addressed:

- **Easy integration:** The integration of ground vegetation into existing or new sceneries must be practicable with little modelling effort. This is an important requirement for Virtual Testbeds, as they are not limited to a specific application area.
- **Low render cycle times:** Computation and rendering times must be held low to keep enough time for other rendering modules as well as computer vision testing algorithms, which can also be computationally expensive. This also includes GPU calculation times if implemented using GPGPU.
- **Close-to-Reality visual representation:** Ground vegetation needs a realistic appearance from every viewing angle and distance.
- **Reproducibility and consistence:** The whole simulation must be repeatable in exactly the same way, which is another important feature of Virtual Testbeds. Thereby, occuring errors in the testing algorithms can easily be analyzed and fixed.
- **Sufficiently accurate object-grass interaction:** Objects within the scene should interact with the grass in a physically plausible way, even for complex models. Moreover, permanent changes like tire tracks and squeezed grass need to be simulated as they influence feature detection, for example.

Due to the fact, that the observer will hardly get so close to the ground that visual differences between single grass blade rendering and textured quads of grass clumps become noticeable, we forgo costly single grass blade rendering. This assumption is also true regarding computer vision, as most applications are based on mainstream digital cameras with a relatively low image resolution. Due to these facts, billboard-based approaches like named in section 2 are the best choice here, which use textures of whole clumps of grass, plants or flowers. Figure 2 illustrates the rendering process. Quads are distributed over the scene showing a specific plant contained in a vegetation texture atlas shown on the left. This texture atlas contains all plants that can occur in the scene and is classified by vegetation categories like grass, foilage plants, flowers or forest ground vegetation like fern.

Our approach distributes the ground vegetation of the whole scene directly at load time using a fully reproducible approach to allow for reproducibility and consistence of the test scene. However, this procedure demands for a low memory footprint of the required data regarding large scenes. Thus, we do not rely on

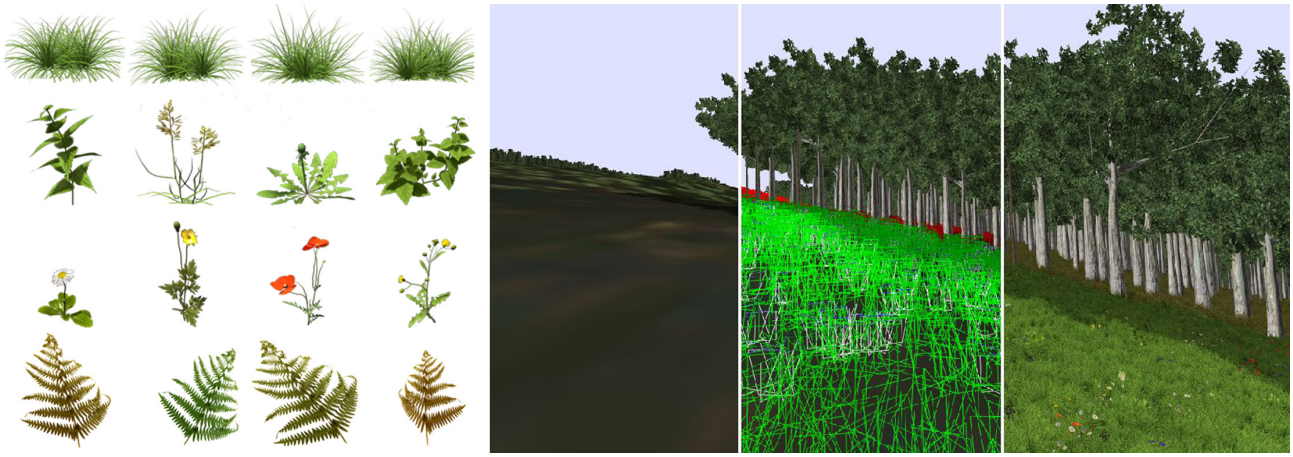

Figure 2: Left: Example of a texture atlas for vegetation rendering. Right: Illustration of the ground vegetation rendering process using textured quads.

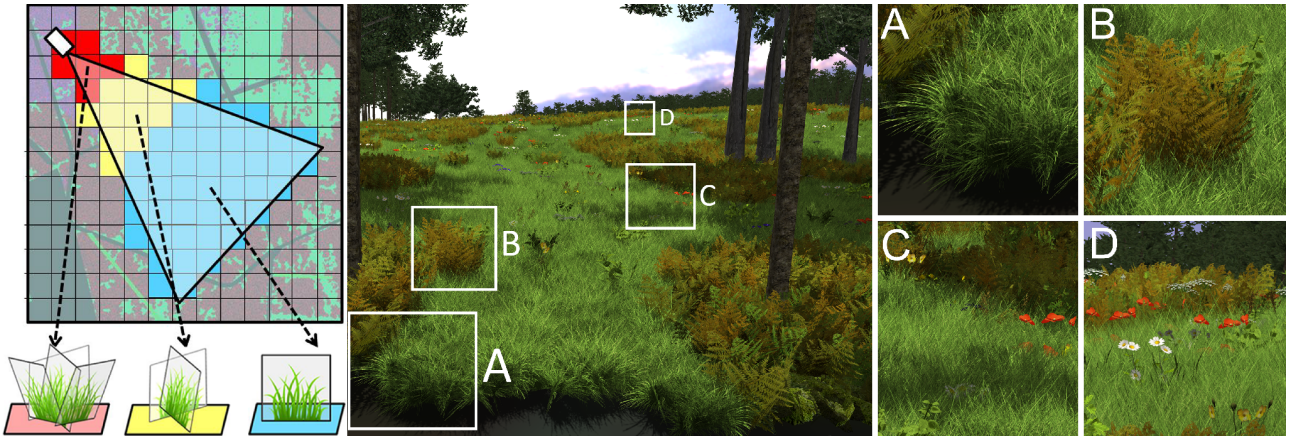

Figure 3: Left: Illustration of different level-of-detail representations. Right: Amplification of different sections of a rendered scene.

complex plant object representations like used by Orthmann et al. [4]. Instead, we developed a novel approach where each plant object is represented as a single point only, indicating it's position on the ground. During the rendering process, the plant objects geometry is automatically generated by a geometry shader directly. This approach only requires very little data in order to reconstuct renderable plants. Our plant objects only consists of the position, the up direction, as well as a parameter vector containing the vegetation type, an orientation angle and a random value used to prevent uniform appearance of the rendered plants. When adding interaction, only a vector for the collision forces and bending direction needs to be added. Moreover, the automatic geometry generation allows for fully dynamic level-of-detail (LOD) switching. No explicit modification of the plant objects is required at run-time and no multiple copies of differently complex grass object representations need to be held in memory. The left side of Figure 3 illustrates the LOD determination and the resulting grass object complexity. We differentiate between three different complexity levels. plant objects close to the camera are expanded using four textured quads with fixed orientations and different horizontal and vertical angles. Thus, the illusion of dense vegetation is achieved even if looking from above (see middle part of figure 1). The second level of detail only consists of two textured quads. This is sufficient to generate a dense vegetation at mid-range. plant objects far away are drawn as single screen-facing billboards, as they are not noticeable at high distances. Furthermore, they cover the ground quite well, even if the camera is viewing from different angles.

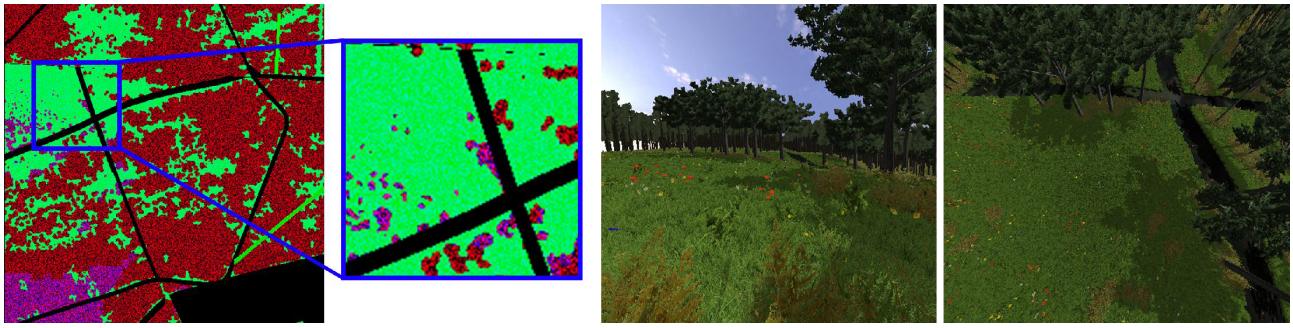

Figure 4: Left: Example of a grass map. Different vegetation types are encoded into the grass map's color channels. Right: Screenshots of the amplified area with corresponding ground vegetation.

3.1 Distributing Ground Vegetation

Due to the fact, that modeling each single position of the grass layer or modifying existing ones requires a huge modeling effort, we rely on a grass map for easy and efficient vegetation distribution. Grass maps are used by many vegetation rendering approaches, which are suitable for large sceneries. Our approach encodes the type and density of the vegetation into the color channels. Each channel represents a row in the vegetation texture atlas while the channel's intensity defines the vegetation density. Figure 4 shows an example of a grass map and the resulting rendered ground vegetation. Green marks grassland, red marks forested areas and blue marks shrubs and wild flowers. Other kinds of grass map implementations and interpretations presented in recent approaches can easily be used instead. Moreover, the specific plants represented by the texture atlas can easily be exchanged to match other vegetation zones with minimal effort.

The usage of a grass map allows for easy modeling of the ground vegetation layer with a common painting program. However, the grass map can also be generated using more advanced techniques. When thinking of geodata servers (GIS), available data does not only consist of geometry but also semantic data, describing the environment in more detail. Our simulation system is capable of interpreting data received from GIS servers in various ways to enhance the simulation [2, 3]. Typically, information like land coverage, infrastructure or even vegetation types can be used to generate a grass map representing a realistic mapping of the real-world vegetation.

3.2 Organizing Vegetation Data

In order to optimize vegetation data regarding render performance, data size and flexibility, a novel data structure was developed for our approach. The ground geometry is tiled into a regular grid of 500x500 meters representing a scene tile. The scene tile stores the grass map, that is mapped to the geometry. The scenery tile is further separated into smaller grass fields of 25x25 meters size, as illustrated in figure 5.

Each grass fields contains the offset to the corresponding sub-sample of the scene tile's grass map and is further used for culling as well as level of detail determination. The grass field represents the basic unit for the grass renderer and each grass field is handled separately. In contrast to approaches using a huge single vertex buffer for all plant objects [4, 5], we use separate vertex buffers for each grass field. This simplifies memory and interaction handling while still achieving high performance due to smaller memory blocks that need to be updated. Moreover, techniques like vertex array objects allow for rapid buffer switching. Finally, each generated plant object is arranged in plant clusters of 5x5 meters size. Depending on the grass map's value of a specific location, a corresponding count of plant object positions is generated and adjusted to the surface geometry. All plant objects are distributed using a predefined random seed number. Thus, the exact distribution of all single plants can be repeated in a later simulation pass. Furthermore, all animation like wind and interaction use the simulation system's clocked time steps. Therefore, tested computer vision algorithm

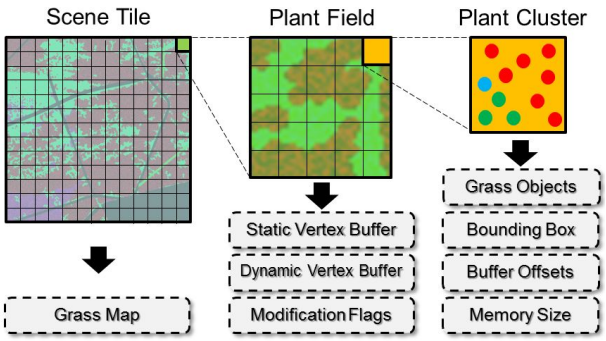

Figure 5: Illustration of the vegetation data structure.

can easily be debugged and tested. Plant clusters as well as static and dynamic buffers of the grass fields are important for efficient collision handling and are further explained in the following section.

3.3 Wind Animation and Interaction

The vegetation animation is categorized into two groups: wind animation and object-grass interaction. To minimize updating effort, the wind animation is added to the plant objects at run-time directly in the geometry shader. Thus, no buffer updates are necessary for grass fields that do not contain interaction. Due to performance reasons, the wind animation is based on procedural vertex animations as suggested by Pelzer [12], which is sufficiently realistic for wind animation. To prevent a uniform look, random values stored in each plant object are used as animation and speed offset. All data necessary for rendering and wind animation is stored in the grass field's static vertex buffer. After initialization, this buffer never needs to be updated. Only data necessary for interaction handling like the collision forces or modification variables are stored in a separate, relatively small dynamic vertex buffer, leading to a great overall performance. Due to the fact, that only relatively small areas of the whole grass field interact with objects, the grass field is further subdivided into plant clusters as mentioned before. Each plant cluster contains the bounding box of the contained plant objects as well as offsets and sizes referring to the plant field's vertex buffers. Thus, only small ranges of the grass field's vertex buffer need to be updated, which drastically reduces the computing costs and transfer times for collision handling and movement updates.

Large scenes can consist of several thousands of objects, where most of them do never interact with the grass. Thus, we use specific collider objects like suggested by [14]. Collider objects can be attached to the scene's objects. Interacting objects can be very complex making exact collision handling costly even when performed on the GPU [4]. Regarding the performance demand of this approach, we decided to implement a simplified collision model. Thus, our collider objects contain a simplified hull encasing the underlaying complex geometry. Considering a mobile robot driving through a grass field, collider extensions are attached to the wheels, thus, only the wheels and its collision hulls need to be taken into account when updating the grass fields. Moreover, this approach allows for the simulation of local wind breezes. Spherical collider extension can be moved through the scene using random radii and forces to generate the effect of local turbulences. Downwinds of airborn vehicles like quadcopters can also be simulated, that whirl up ground vegetation.

3.4 Vegetation Rendering Process

The simulation process is illustrated in figure 6. At simulation start, the static and dynamic vertex buffers of the grass tiles are initialized and filled, regarding the values read from the grass map. The plant objects are arranged into the corresponding plant cluster and it's bounding box is generated. Finally, the buffer offset and size of the plant cluster are stored to be able to modify it for collision handling. The initialization step is fully

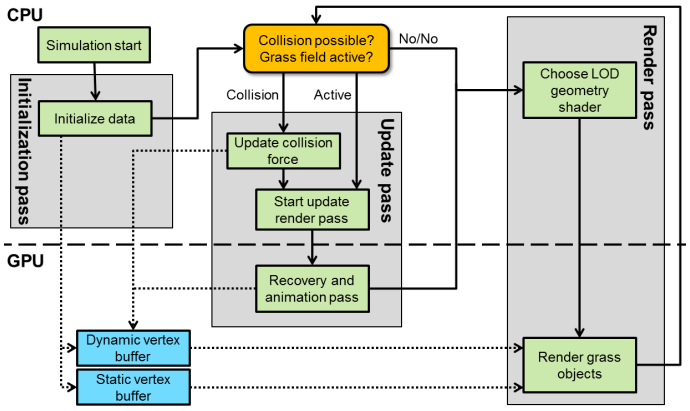

Figure 6: Illustration of the ground vegetation simulation and rendeing process.

configurable with parameters for vegetation density, height and visual variances, to adjust our approach to the desired application area.

The collision handling is separated into two steps: collision handling and recovering. In our approach, collision handling is performed on the CPU. All objects containing a collider extension, which are arranged in an octree structure for fast collision testing. The tree is traversed and checked for plant field collision. If a collision has been detected, the grass field is marked as active and the current time stamp is stored. Subsequently, intersecting plant clusters are determined, the corresponding vertex buffer range of the dynamic buffer is mapped and the collision forces are updated. The status value of the plant object is used to mark it for specific update handling required by the subsequent GPU pass. Currently, we differentiate between four possible states: inactive, recovering, felled and cut. Per default, all states are set to inactive. If a collision was handled in the CPU pass, the state value is set to recovering. If the collision forces reach a defined maximum or the plant objects exceeds a maximum bend, the state is set to felled, marking the plant object as a felled object. The cut state is handled similar, marking vegetation that was mowed.

After the CPU collision processing, the plant objects need to be animated corresponding to the collision result. This processing step takes place on the GPU. If the grass field is marked as active, it is rendered using an animation vertex shader that updates its movement and streams the results back to the dynamic vertex buffer (using stream out). To minimize the amount of the required draw calls, the GPU update pass is carried out for a whole grass field and not per plant cluster, like in the CPU pass. In our implementation the movement is updated with a simple spring-mass equation [10]. Active grass fields are also handled for subsequent updates in following frames, even if no collision is detected any more. This allows for recovering of the plant objects from the position that resulted from the collision back to it's resting position. After a defined amount of time, the grass field is assumed to be completely recovered and is marked as inactive to exclude it from further update-related processing. Figure 7 illustrates the collision handling pass. Active fields are marked in red while currently colliding and recovering plant objects are marked in blue. After a defined amount of time, the plant fields are assumed to be fully recovered and set to inactive again. Thus, we can grant optimal performance as only grass fields are updated that actually need collision or recovery handling.

After the update pass, the geometry is ready for rendering. View-frustum culling is applied to determine the visible grass fields. Depending on the current distance to the camera, a matching geometry shader is chosen to generate adequate plant objects. Due to the novel point-based grass blade representation, the LOD-switching can directly be performed at run-time without the need for memory updates or the storage of multiple geometric grass blade representations per object. The three shader sets (one for each level of detail) are activated and all corresponding grass fields are rendered. The LOD geometry shader also calculates the current wind animation and adds the result to the generated vertices, which already contain the object interaction movement from the update pass. The fragment shader applies lighting calculations depending on the current up direction of the

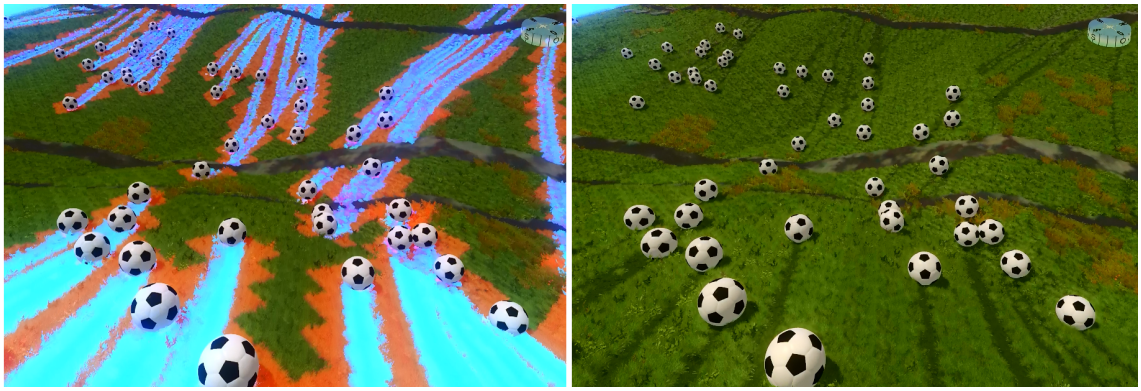

Figure 7: Left: Active grass tiles are marked in orange, actually updated plant objects are marked in blue. Right: Footballs leave tracks in the vegetation layer. If collision forces and bending angles exceed a defined limit, the state value of the grass object is set to felled.

plant objects. Moreover, the plant objects' random values are used to alter the lighting intensity and plant object sizes to generate a natural, non-uniform appearance with some variation. For shadowing purposes, common shadow mapping techniques are applied.

To prevent the need of computational costly depth sorting when using semi-transparent vegetation textures instead of binary ones, the alpha-to-coverage technique can be used, which is supported by modern graphics cards [15]. The number of subpixels that will be filled with the current pixel color are determined by the current alpha value. Subsequently, the subpixels are blended while resolving the multiple resolution to the final image resolution. This approach generates realistic grass fields with good performance. The whole grass renderer only consists of a single fragment shader for lighting and shading, thus it can easily be replaced by a more complex one to achieve more accurate results. Due to the fact, that only collider objects that actually collide with the grass influence the performance, huge amounts of collider objects can be realized with good performance. The performance mostly depends on the amount of active grass fields that need to be updated.

4 Testing Computer Vision Algorithms

The simulation of highly detailed outdoor scenes is very important when synthetic environments are used to handle some of the engineering and testing problems for computer vision [17]. Performance evaluation is an important subject in computer vision [19] and as ground truth is easily obtained using simulation systems, evaluation of stereo and optical flow algorithms are usually performed on computer rendered images.

For benchmarking reasons, we attached a collider object actually colliding with the grass to every single plant cluster of a grass field (25 collider objects per grass field), which represents a worst case scenario. On our test system (Core i7 4x2.93GHz, NVidia GeForce GTX580, 1024x768, 4xFSAA) the updating time for 10 grass fields (250 colliders) take less than 1 ms, 100 grass fields (2500 collider) are updated in 10 ms and 400 grass fields (whole 500x500 meters grass tile, 10000 colliders) are updated in 40 ms. However, this worst case scenario will hardly ever occur in real applications, hence, realistic updating times will be significantly lower making this approach practical for large amounts of collider objects even in real-time critical applications.

The possibility to generate highly detailed and dynamic scenes, which are repeatable in every simulation step, is a further development from virtual simulators to virtual testbeds [20]. Combined with advanced sensor simulation techniques [21] it is possible to generate image data which can be used for testing and evaluating basic image processing algorithms from stereo depth estimation or optical flow up to complex vision applications such as driver assistance systems or mobile robot navigation.

The techniques of highly detailed environment simulation are currently combined with the full dynamic simulation of forestry working machines and mobile robots for the development and testing of self-localization

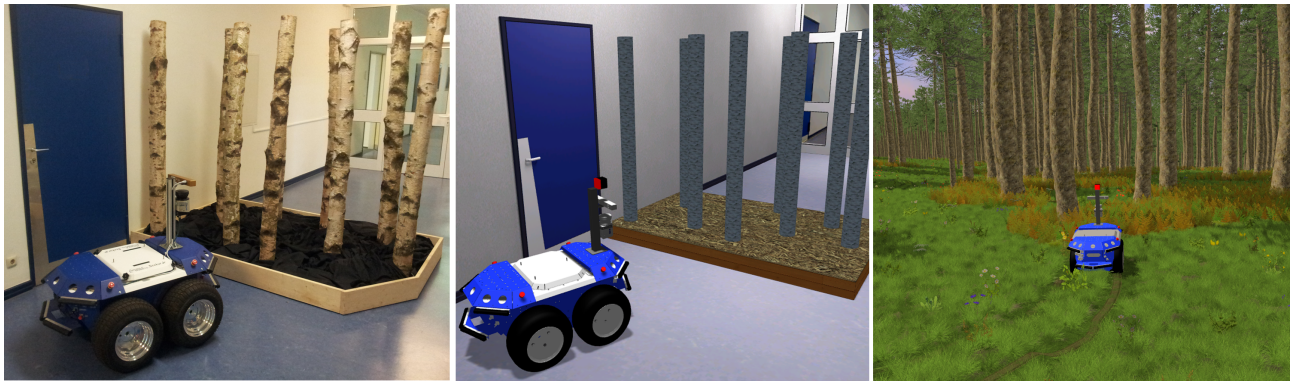

Figure 8: Left: Real test environment under laboratory conditions. Middle: Common virtual testing environment for testing of computer vision algorithms. Right: Close-to-reality virtual test environment using dynamic ground vegetation for more accurate testing results.

applications in forest environments. We implemented a landmark-based localization framework, which is able to estimate the current position of mobile systems on the basis of landmarks observed by optical sensors (currently from 2d laser scanners, stereo cameras and time-of-flight cameras) [23] and a navigation map generated from remote sensing data containing landmarks of the test area [24]. The navigation map can be updated directly by the robot system with newly observed landmarks. In forest environments trees are frequently occurring landmarks.

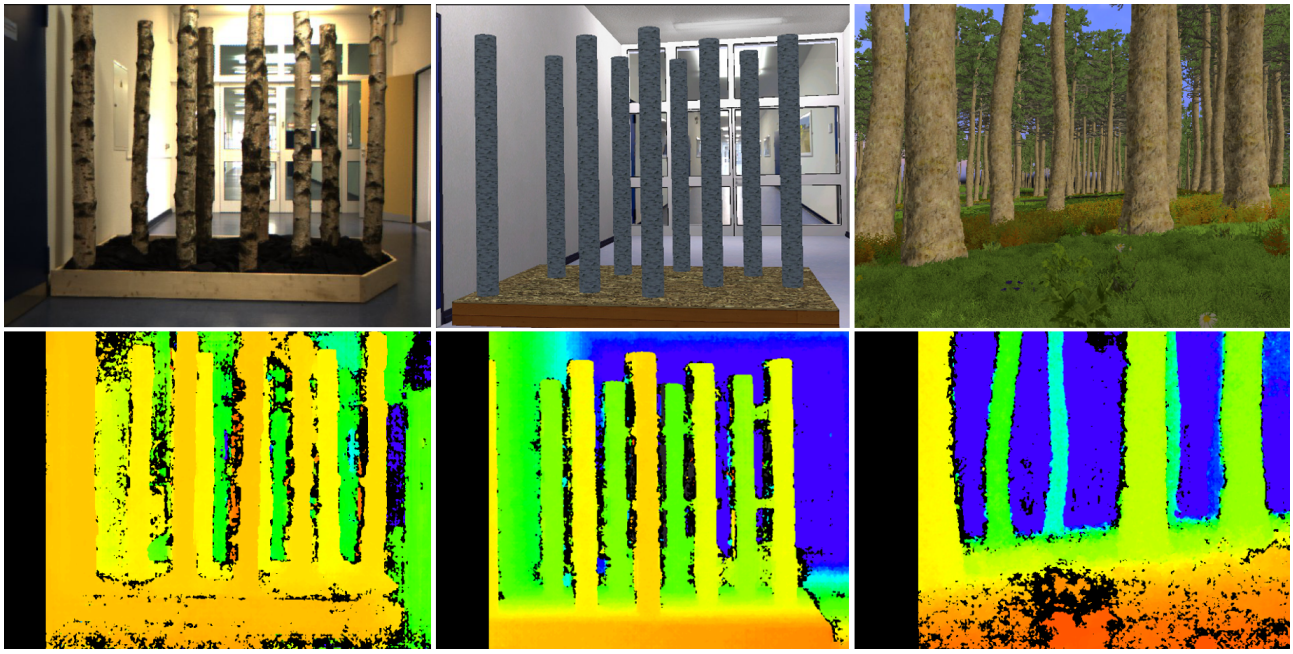

Figure 9: Colored visualizations of the depth and stereo matching results. Left: Real-world test environment. Middle: Common virtual test environment. Right: Close-to-Reality Virtual Testbed.

Figure 8 shows different environments for testing and evaluation of computer vision algorithms. A common real-world testing environment under laboratory conditions is shown on the left. If a real-world testing environment is not available, VR simulations including stereo camera simulations as described in [22] can be used instead, also delivering results comparable to real ones. The image on the right shows a complex, close-to-reality testing environment including our ground vegetation simulation. Results achieved from this Virtual

Testbed are much closer to reality than any other approach so far. By simulating fine ground vegetation structures in combination with wind and object interaction, it is possible to synthetically generate complex non-static scenes to test computer vision algorithms in many different levels of detail.

Figure 9 shows a simulated scene with highly detailed ground vegetation rendering, the corresponding ground truth and the result of a semi global matching (SGM) stereo disparity estimation. The ground truth of the very same scene without any ground vegetation is also displayed showing the difference between static scene sequences as provided by [18] and dynamic virtual testbeds. Latter are using up-to-date rendering techniques as high dynamic range lighting and dynamic scene geometry. As the geometry of dynamic scenes changes in the course of simulation time due to wind simulation and object interaction, the scene is sufficiently complex to initially test and parameterize high-level vision applications already in early stages of the design process, leading to much earlier hardware requirements processing and estimation of real-time capability [17].

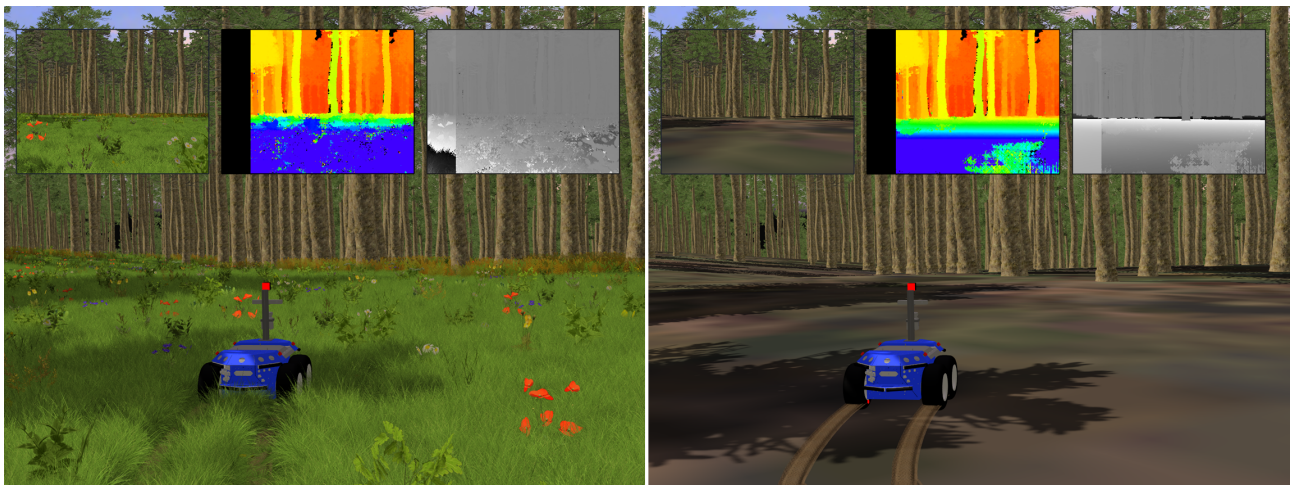

Figure 10: Left: Test scene with high detailed ground vegetation. Right: Same scene without ground vegetation. The displayed overlays show one of the stereo input images, the SGM disparity map (color coded from cold to warm), the difference image between the ground truth disparity data and the SGM disparity map.

For the evaluation of stereo disparity measures, we use the ground truth data generated by the simulation system and calculate a disparity map corresponding to the baseline and intrinsic parameters of the stereo camera system. Thus, it is possible to easily assess the stereo matching algorithm directly in units of disparity pixels. Figure 10 shows two simulated scenes with and without ground vegetation. An evaluation of the disparity map generated using SGM stereo disparity estimation is given in figure 11. While the matching algorithm has major problems finding corresponding pixels on plain-colored surfaces, the matching results on highly detailed ground vegetation is much more reliable, although the very fine ground vegetation cannot be reconstructed in detail. In our test scene, we achieved mean errors to the amount of five pixels difference in disparity with ground vegetation and errors over fifteen pixels difference with low detailed surfaces. The difference images in figure 11 show the signed pixel difference between the ground truth disparity and the stereo estimated disparity shifted towards the middle of the 8bit gray scale image. Thus the histograms have their peak around the value of 127 with standard deviations of up to 28 using no ground vegetation and values up to 18 with ground vegetation turned on.

The interaction between robots and their environment is another important issue of computer vision methods for robot navigation. Tracks left by the robot can change the appearance of the environment in a way, so that feature detection algorithms will have problems to recognize certain features afterwards. On the other side, the interaction of robots with their environment results in new features, which can be used for navigation or localization purposes. For example, the mars exploration rovers (MER) used visual odometry algorithms for movement verification. The feature detector was optimized for feature-laden natural terrain, and often failed

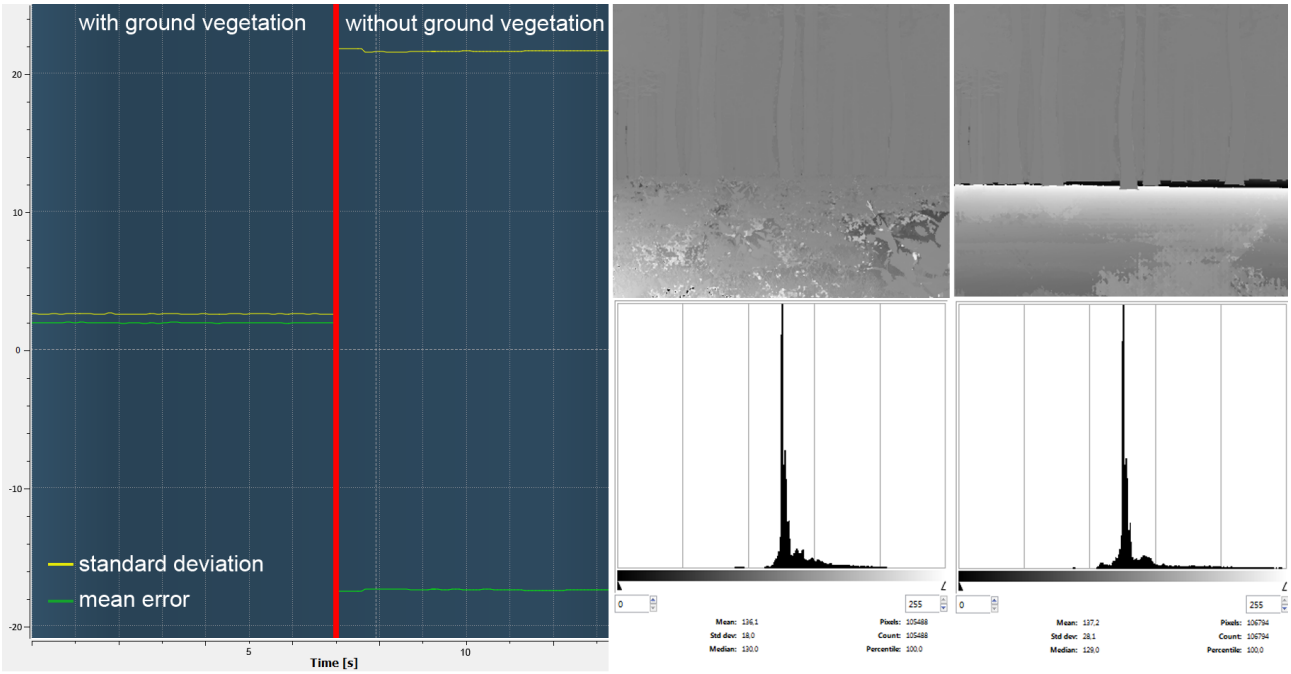

Figure 11: Left: mean error and standard deviation between SGM and ground truth disparity data with and without ground vegetation. Middle: Difference image and its histogram of the scene including ground vegetation. Right: Same information for scene without ground vegetation.

to find sufficient unique features in the piles of sand. Finally they used the tracks left by the vehicle as they provided enough features to enable convergence. This is the reason, why the MSL rover curiosity currently exploring the Gale Crater on mars has a very prominent tire tread pattern producing tracks on the surface which can easily be tracked for visual odometry pruposes. By simulating tire tracks on both vegetated and non-vegetated surfaces the benefit of tracks left by the robots tires can be examined directly in the virtual environment.

Especially in the development of new robot vision approaches, testing in virtual environments is very comfortable as the robot hardware does not need to be available during the entire development process. Furthermore, the simulation results can be used to influence the requirements of hardware design, avoiding costly redesigns later in the process. By gradually increasing the details of the simulated environment according to the development status of the vision algorithms it is possible to test basic functionalities as well as entire robot navigation applications with ground truth in one virtual testbed provided by our simulation system. Figure 10 shows the difference between a static test scene and a highly detailed scene with dynamic ground vegetation and tire tracks. The quality of the stereo matching results differ conspicuously. The depth estimation of the flat ground results in substantial error due to missing features, known from stereo matching of indoor environments with plain-colored walls and floors. In contrast, the highly detailed scene provides enough features for satisfying stereo matching and depth estimation, where even the geometry of particular plants can be recognized.

5 Conclusion and Outlook

In this paper, we presented a practical ground vegetation simulation, capable of generating and rendering huge vegetation fields on arbitrary geometry with excellent performance. We have successfully realized a grass simulation that meets the most important requirements of complex simulation systems: The generation and rendering of naturally looking grass from all lines of sight and wind as well as object-grass interaction to grant an immersive, interactive virtual environment. Due to the fact, that interaction and collision handling

is only calculated when actually needed, it also performs with minimal workload overhead achieving great performance, even in large sceneries.

Moreover, the rendered results have an appearance so close to reality, that they can also be applied to other application areas that rely on realistic imagery like in the area of computer vision. Our approach also fulfills important requirements like plausible geometry representation of the vegetation as well as its consistence over the entire simulation. Results of stereo matching in commonly used real-world testing environments as well as the results achieved with imagery generated by our Virtual Testbed including realistic ground vegetation are compared. We have shown that Virtual Testbeds are capable of supporting the development of novel computer vision algorithms as they deliver close to reality imagery. Regarding natural outdoor scenes, modern rendering techniques like the presented ground vegetation approach further improve Virtual Testbeds for testing and development of computer vision applications. By comparing the results of data acquired with and without the shown ground vegetation approach, we noticed a significant improvement on stereo matching results using high dynamic scene rendering as observable in real-world outdoor environments. Moreover, they are easily configurable and available at lower costs, as they run in real-time even on standard consumer PCs.

We are currently investigating more application areas of Virtual Testbeds in the area of computer vision. Validating the simulating components of the Virtual Testbeds by means of real hardware and environments allows for interpolation to new domains. Transferring the results from self-localization of forest machinery to the localization and mapping of robots on extraterrestrial planetary surfaces, we use rocks and craters as landmarks. As the required rock density cannot be warranted in the whole mission area, we implemented the possibility to fall back on tracking algorithms extending the absolute localization approach by a relative localization component. As the MER and MSL missions on mars, we use a visual odometry approach to fill the gap between terrain with enough landmarks separated by open ground with few or no landmarks. As features can change or even be destroyed over time in outdoor terrain, it is of great importance to be possible to simulate these scenarios in a Virtual Testbed to keep the number of real outdoor missions low, especially in the early development process, as the costly real world tests should be carried out in more advanced stages of development confirming the results of the virtual tests. As simulated test scenarios are cheap, easily to reproduce and able to be run unsupervised, intensive testing in Virtual Testbeds is an important and recurring step during the entire development.

References

[1] N. Hempe, J. Rossmann, ”Efficient Real-Time Generation and Rendering of Interactive Grass and Shrubs for Large Sceneries”, *Proceedings of the 13th IASTED International Conference on Computer Graphics and Imaging (CGIM 2012)*, 1:240-247, 2012.

[2] N. Hempe, J. Rossmann, R. Waspe, ”Geometric Interpretation and Optimization of Large Semantic Data Sets in Real-Time VR Applications”, *Proceedings of the ASME 2012 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference (IDETC/CIE 2012)*, 1:1-10, 2012.

[3] J. Rossmann, M. Schluse, M. Hoppen, R. Waspe, ”Gis-based virtual testbeds and their application to forestry and city simulation”, *In Proceedings of the ASME 2010 World Conference on Innovative Virtual Reality (WINVR 2010)*, 2010.

[4] J. Orthmann, C. Rezk-Salama, A. Kolb, ”GPU-based Responsive Grass”, *Journal of WSCG (17)*, 1:65-72, 2009.

[5] D. Whatley, ”Toward Photorealism in Virtual Botany”, *GPU Gems 2*, 1:7-25, 2005.

[6] B. Bakay, W. Heidrich, ”Real-Time Animated Grass”, *Proceedings of Eurographics (short paper)*, 2002.

[7] J.E. Lengyel, E. Praun, A. Finkelstein, H. Hoppe, "Real-Time Fur over Arbitrary Surfaces", *ACM Symposium on Interactive 3D Graphics*, 1:227-232, 2001.

[8] R. Ramraj, "Dynamic Grass Simulation and Other Natural Effects", *Game Programming Gems 5*, 1:411-419, 2005.

[9] C. Wang et al., "Dynamic modeling and rendering of grass wagging in wind", *Computer Animation and Virtual Worlds*, 1:377-389, 2005.

[10] S. Banisch, C.A. Wthrich, "Making Grass and Fur Move", *Journal paper of the 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2006.

[11] S. Behrendt, C. Colditz, O. Franzke, J. Kopf, O. Deussen, "Realistic realtime rendering of landscapes using billboard clouds", *Eurographics Computer Graphics Forum 24(3)*, 1:507-516, 2005.

[12] K. Pelzer, "Rendering countless blades of waving grass", *GPU Gems*, 1:107-121, 2004.

[13] P. Decaudin, F. Neyret, "Rendering forest scenes in real-time", *Proceedings of Eurographics Symposium on Rendering, Rendering Texhniques '04*, 1:93-102, 2004.

[14] S. Guerraz, F. Perbet, D. Raulo, F. Faure, M.-P. Cani, "A Procedural Approach to Animate Interactive Natural Sceneries", *IEEE Computer Society*, 2003.

[15] K. Myers, "Alpha-to-Coverage in Depth", *Shader X5*, 1:69-74, 2006.

[16] K. Boulanger, S.N. Pattanaik, K. Bouatouch, "Rendering grass in real time with dynamic lighting", *IEEE Journal of Computer Graphics Applications (29)*, 1:32-41, 2009.

[17] W. Burger, M.J. Barth, "Virtual Reality for Enhanced Computer Vision", *Proceedings of the IFIP 5.10 Workshop on Virtual Environments*, 1994.

[18] T. Vaudrey, C. Rabe, R. Klette, J. Milburn, "Differences between stereo and motion behaviour on synthetic and real-world stereo sequences", *Image and Vision Computing New Zealand, 23rd International Conference (IVCNZ 2008)* , 1-6, 2008.

[19] R. Klette, S. Stiehl, M. Viergever, V. Vincken, "Performance Evaluation of Computer Vision Algorithms", Kluver, Amsterdam, 2000.

[20] J. Romann, M. Schluse, B. Sondermann, M. Emde, M. Rast, "Advanced Mobile Robot Engineering with Virtual Testbeds", *Proceedings for the 7th German Conference on Robotics*, 331-336, 2012.

[21] M. Emde, J. Romann, B. Sondermann, N. Hempe, "Advanced Sensor Simulation in Virtual Testbeds: A Cost-Efficient Way to Develop and Verify Space Applications", *Proceedings of AIAA Space 2011 American Institute of Aeronautics and Astronautics (AIAA) Conference and Exposition*, 2011.

[22] J. Rossmann, T. Steil, M. Springer, "Validating the Camera and Light Simulation of a Virtual Space Robotics Testbed by Means of Physical Mockup Data", *Proceedings of 11th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2012.

[23] J. Rossmann, B. Sondermann, M. Emde, "Virtual Testbeds for Planetary Exploration: The Self-Localization Aspect", *Proceedings of 11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2011.

[24] J. Rossmann, N. Wantia, M. Springer, O. Stern, H. Mller, M. Ellsiepen, "Rapid Generation of 3D Navigation Maps for Extraterrestrial Landing and Exploration Missions: The Virtual Testbed Approach", *Proceedings of 11th Symposium on Advanced Space Technologies in Robotics and Automation (ASTRA)*, 2011.