Immersive Simulation for Computer Vision

Wilhelm Burger,[1] Matthew J. Barth[2]
and Wolfgang Stürzlinger[3]

[1] Johannes Kepler Univ., Dept. of Systems Science, A-4040 Linz, Austria
wilbur@cast.uni-linz.ac.at
[2] University of California, College of Engineering, Riverside, CA 92521-0425, U.S.A.
[3] Johannes Kepler Univ., Dept. f. Graphics & Parall. Process., A-4040 Linz, Austria

Abstract. Synthetic imagery has often been considered unsuitable for demonstrating the performance of vision algorithms and systems. We argue that (despite many remaining difficulties) simulation and computer graphics are at a point today that make them extremely useful for evaluation and training, even for complex outdoor applications. This is particularly valuable for autonomous and robotic applications, where the lack of suitable training data and ground truth information is a severe bottleneck. Extensive testing in a simulated environment should become an integral part of the systems development and evaluation process to reduce the possibility of failure in the real world. We describe ongoing efforts towards the development of an "Immersive Perception Simulator" and discuss some of the specific problems involved.

1 Introduction

Vision systems for autonomous and reactive robotic applications need to perform reliably under a variety of conditions. Since these systems are quite complex, formal verification methods usually do not exist and extensive testing provides the only way to ascertain the required performance. In a typical robot vision design process, testing of individual system components is limited to pre-recorded images and image sequences that are processed in a passive fashion. Closed-loop testing can only be performed *after* the complete system has been integrated and deployed on the real robot. Frequently, the task environment or even the actual robot is not accessible at all during system development, e.g., in many space applications. In the past, we also have seen several outdoor demonstrations of autonomous robots end in embarrassing or disastrous ways. Moreover, systems equipped with adaptation and learning, which are becoming increasingly important in computer vision, often require many training runs and additional feedback from the environment to converge to their optimal performance. Obviously, the lack of test data (particularly those with corresponding ground truth data) and realistic testing facilities constitutes a severe bottleneck in the vision system design process.
Thus two important needs exist:

- A large number of test cases (perhaps two orders of magnitude more than

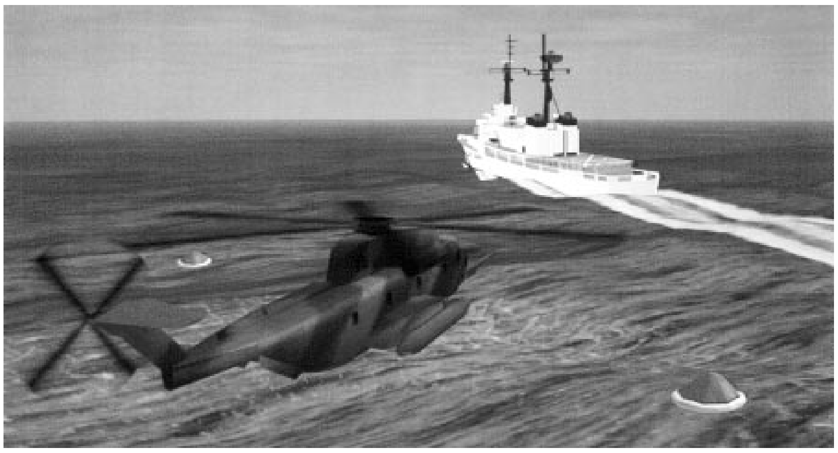

Fig. 1. Synthetic image demonstrating the degree of realism and sophistication available from current simulators. This example includes physics-based object and surface models, shading, focusing effects, motion blur, and atmospheric conditions. Although some effects are not modeled by this simulator, a perception system that performs badly on this kind of imagery can be expected to fail in a similar situation in reality (image source: Evans & Sutherland).

what exists now) with access to ground truth information is necessary for tuning and validating vision systems.

– It is necessary to have an *environment* that allows for the testing of complete vision systems *before* deployment, thus eliminating the high risk of system damage or mission failure.

The key question is, where should all these test cases come from? A promising solution to this problem is the use of advanced simulation techniques. Simulation and computer graphics are at a point today that allow the generation of a large variety of scenes realistic enough to fool the human eye. (Figure 1). Airplane pilots around the world spend thousands of hours inside flight simulators to get prepared for their sensitive tasks. Why shouldn't the same technology be used to prepare artificial vision systems? Of course, there have been pioneering efforts at various places, where different kinds of environmental simulation have been was applied for specific vision tasks, using special-purpose tools, and at different stages of the vision development process. Here we want to make a case for the *consistent* and *pervasive* use of simulation at all stages of the computer vision design process. We believe that the two main reasons for the underdeveloped role of simulation in computer vision are: (a) a wide skepticism among vision researchers against synthetic imagery in general and (b) the lack of an easy-to-use and widely available simulation environment that would avoid expensive "home-brew" solutions.

The reservations against synthetic imagery in the vision community are reasonable if one considers the state-of-the-art in image synthesis ten years ago.

Most images then simply looked "too clean" and synthetic to reasonably replicate the difficulties of image analysis on real imagery. Surfaces were usually simple (blocks world) and without texture, and the generation of sophisticated and complex outdoor scenes was beyond imagination. Today things are different and we believe the skepticism against synthetic imagery in general is mostly obsolete. It is still important, however, to make sure that the results obtained from simulation are relevant in the corresponding reality, i.e., that the simulation results best match physical reality and are not biased to simplify computer vision algorithms. There will always be details in reality that cannot be modeled in a synthetic world, because any model is only an abstraction of reality no matter how specific it is. Thus *we cannot assume that a (vision) system which performs flawlessly in a simulated world is guaranteed to succeed in reality. However (as applies to airplane pilots) if it* <u>*does*</u> *fail in a simulator test, then it will likely have trouble in the real world!*

In the following, we describe our work towards an "immersive" simulation environment for perception-based robotic applications. While the focus is on "computer vision" throughout the text, most arguments equally apply to other forms of sensory inputs, including lasers, radar, infrared, and (to some extent) sound.

2 Why Simulation?

Simulation has always played some role in image processing, computer vision, and control system engineering. Specific vision examples are the ALV testbed at the University of Maryland [2], the map-based simulation environment developed at Hughes [8], and the IUA simulator at UMASS [15]. Yet, until now, there has been *no* simulation environment available that could support the complexity of complete perception and control systems, and provide the required realism at the same time. We are therefore developing a comprehensive simulation system with capabilities that are critical for solving the complicated design and testing problems that are typical in computer vision. The foremost expectations from this approach are:

a. Simulation allows *extended testing* in arbitrary environments, under changing environmental conditions, and in extreme situations that may be rarely encountered in reality but may be crucial for system operation. The simulations can be run *unsupervised*, at any time of day, under any weather conditions, and on several computers in parallel. The increased number of situations and test cases explored in this way should improve system reliability dramatically.

b. *Ground truth information*, essential for validating many vision tasks, is available at any time. In contrast, currently used pre-recorded image data come either with very limited ground truth data, if any.

c. *Adaptation and learning* at all vision system levels can be performed efficiently and autonomously. Large sets of training examples can be processed without human intervention in both supervised and unsupervised modes.

d. Realistic testing of individual modules in a complete system environment is possible at a very early phase, even when the real robot or other parts of the system (e.g., sensors) are *unavailable.* The evaluation of system design alternatives is possible without actual deployment on the robot.

e. The *costs* for designing, building, and running the robot can be reduced. Simulation results will generally influence the robot's design, thus avoiding costly redesigns later in the process. Further, the transition between testing and deployment of the system can be streamlined. Usually, transferring a vision system from the lab onto the real robot requires a major logistic effort.

f. Processing *hardware requirements* and *real-time capability* can be estimated much earlier in the design process. Usually, hardware performance requirements are difficult to estimate before the complete system is running. Real-time behavior can be simulated even when the required high-speed hardware is still unavailable during system development.

Commercial flight simulators are among the systems that would meet many of these specifications. However, they are expensive, physically large, designed for a very specific purpose, and they have to cut corners to be fast enough (i.e., by not implementing reflections, glare, shadows of moving objects, etc.). Although flight simulators have actually been used with computer vision systems (closed-loop experiments for automatic aircraft landing [6]), their sophisticated mechanical design (i.e., hydraulics for emulating aircraft motion) is unnecessary for most perception tasks.

3 The Immersive Perception Simulator (IPS)

The IPS is a new software environment intended to support a wide range of simulation and evaluation tasks in computer vision. Its main purpose is to provide a simple, flexible, and cost-effective mechanism for testing and tuning perception-based reactive systems in a sufficiently realistic environment. The core of the IPS is based on public-domain software modules, which supports the intention to make this system freely available to the vision and robotics community. Flexibility is achieved by providing an abstract interface protocol that simplifies the communication with existing vision systems, e.g., KHOROS, KBVision, and others.

3.1 IPS Architecture

The basic architecture of the IPS consists of three main components: (a) the perception-based robot system, (b) the virtual environment, and (c) the IPS interface (Figure 2). The vision and robot control system interacts with the virtual environment simulator in a closed-loop fashion, where all communication is handled by the IPS interface. The main flow of data consists of *sensory* data generated by the simulator and *action commands* from the robot system. It should be noted that the robot system can be implemented in software, hardware, or as a combination of both. The necessary synchronization mechanisms are

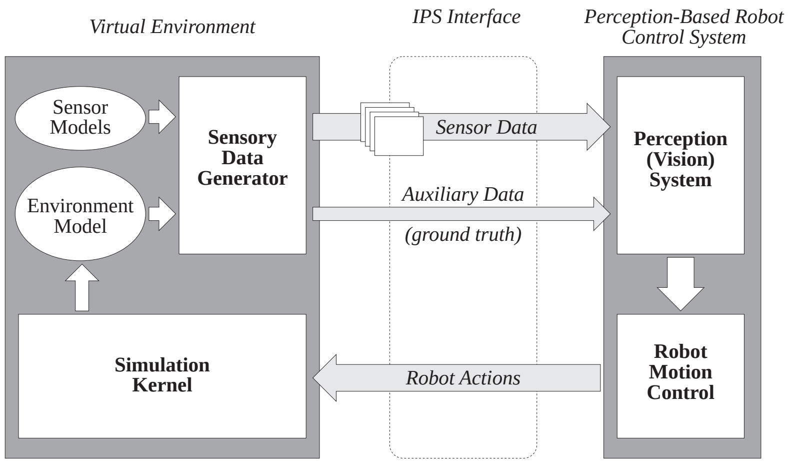

Fig. 2. Immersive Perception Simulator (IPS) simplified architecture. The perception-based robot system interacts with the virtual environment through the interface layer in a closed-loop fashion, exchanging mainly sensory data, ground-truth data, and action commands.

provided by the IPS interface, which is implemented on the UNIX process level.

3.2 The Virtual Environment

The main components of the virtual environment in Figure 2 are the *environment model*, which is a description of the environment and the state of the robot at any point in time that is maintained by what we call the *simulation kernel*. The environment model is used, in conjunction with the appropriate sensor models, to produce realistic images of the scene viewed by the robot's sensors. The structure of the actual simulator resembles that of contemporary "virtual reality" (VR) systems [9] and, in fact, we are borrowing several technological ingredients from VR, such as the modeling of object dynamics and collision detection. However, the main thrust of VR research today is on the implementation of sensors and manipulators for man/machine interaction and high simulation speed, which are not important in our case. On the other hand, the degree of physical realism that we require is not nearly available from any existing VR system. Thus the IPS could be considered an extremely realistic but "slow-motion" VR system.

3.3 Sensory Data Generation

While physical realism is the ultimate goal and requirement of the IPS, there is always a tradeoff between the amount of modeled detail and available computing resources. To make the simulator useful, rendering times must be kept within a certain range. They should probably be shortest at the beginning of the development cycle, when many alternatives still need to be evaluated, and can be longer towards the end when the perception system is converging to its optimal performance. Fortunately, the amount of detail and realism required at the early stages will also be lower than towards the end, so that in general a tradeoff should be feasible. The conclusion is that a *single* rendering approach cannot cover the whole range of applications and the different requirements encountered during the system development process. The solution in IPS is to provide several *different* rendering modules that can can be invoked selectively but use a single environment model.

Some of the available options for rendering visual scenes are listed in Table 1, ranging from simple polygon shaders on the low (but fast) end up to sophisticated (but slow) ray-tracing techniques. At the high end, i.e., for generating

Table 1. Comparison of image synthesis techniques.

Class	Method	Implem.	Features	Availability
1	polygon shading	SW/HW	flat-shaded polygons	
2	flat shading with z-buffer	SW/HW	depth values	
3	Goraud shading with z-buffer	SW/HW	smooth shading, simple fog, point light sources	SGI entry models
4	Phong shading with z-buffer	SW/HW	highlights	
5	texture mapping with z-buffer	SW/HW	surface textures, simple shadows	SGI high end, flight simulators
6	reflection mapping with z-buffer	SW/HW	reflections	SGI next generation
7	ray-tracing	SW	refraction, real camera model, area light sources with penumbra, realistic material models	common ray-tracers
8	ray-tracing + global illumination simulation	SW	indirect illumination	*Radiance*
9	ray-tracing + global illumination simulation + "participating" media	(SW)	realistic clouds, scattering	current research

the maximally realistic images, we are using an enhanced version of the *Radiance* rendering system, which is described below. The choice of the lower-end techniques depends upon the (vision) application and the available rendering hardware. On state-of-the-art graphics workstations (e.g., HP, SGI, Sun), interfaces to hardware-accelerated Goraud shading, Phong shading, and texture mapping (classes $3-5$) are going to be provided. On these platforms, limited availability of ground-truth data is provided by reading the z-buffer contents.

Also, hardware-accelerated techniques do generally not allow to implement non-standard sensor models, but this may be tolerable at the early test stages.

3.4 The *Radiance* Rendering System

Radiance[14], a physics-based rendering system for producing photo-realistic images of complex scenes, is the main rendering tool in the IPS. Although initially developed for applications in architecture and lighting design, *Radiance* is currently the most widely used non-commercial tool for general photo-realistic image synthesis. *Radiance* considers both direct and indirect (global) light sources and uses a combination of deterministic and stochastic ray-tracing techniques to balance between speed and accuracy. It supports a wide variety of object shapes, materials, and textures, and accepts many different CAD input formats for describing the scene. Parallel (distributed) processing and limited animation are also supported.

The most important *deficiencies* of *Radiance* for its application in the IPS are related to the sensor model. *Radiance* uses a simple pinhole camera and ideal shutter model and we are currently extending the package to consider the following effects:

Lens distortion: Real lens systems, wide-angle lenses in particular, are not entirely free from geometric distortions. As a consequence, straight lines in 3-D do not generally map onto straight lines in the 2-D image. However, this assumption is frequently made for vision algorithms that depend on calibrated imagery, such as binocular stereo algorithms using epipolar planes or the Hough transform for finding straight lines and parametric curves.

Depth-of-field effects: In the pinhole camera model, every object in the scene is in sharp focus. This is not the case with a real (thick) lens, where images have a finite "depth of field" that depends primarily on the focal length, the aperture setting, the focus setting, and the distance of the object. This is mainly important for close-range viewing, such as in robotic applications, but can probably be ignored in many outdoor tasks. Practical solutions for simulating the depth-of-field effect exist [11, 5].

Motion blur: Image motion (and thus motion blur) is induced by a moving camera, a moving object, or a combination of both. Due to occlusion effects, moving shadows, etc., motion blur is an extremely complex phenomenon and its simulation time-consuming [12]. Distributed ray-tracing, as proposed in [5], appears to be the only solution that allows arbitrary object shapes and motion paths.

Enhanced outdoor environments: Since *Radiance* is not primarily aimed at simulating outdoor scenes, we are enhancing the system to support the generation of fractal terrain models, plant shapes, and corresponding textures.

3.5 Ground-Truth Data

The acquisition of ground-truth data is usually expensive and tedious (e.g., through separate theodolite or radar measurements [7, 13]), therefore the possibility to access highly reliable ground-truth data is a key motive for using simulation in many vision applications. Rendering systems like *Radiance* do not have this access capability built-in, but most of the internal data structures are already available and *Radiance* is currently being extended to provide this functionality. The direct availability of such data on hardware-based renderers is of course limited. Each application requires specific ground-truth data, including local measures such as pointwise 3-D depth (z-value), local surface orientation, local surface curvature, surface color, material type, and object identity. Selective query mechanisms for these modalities are included in the IPS interface protocol.

4 Selected Vision Problems

For many of mainstream vision problems, such as 3-D scene reconstruction, shape-from-X techniques, sensor calibration, multi-sensor integration, motion analysis, obstacle avoidance, autonomous navigation, terrain interpretation, etc., the huge amount of test data provided by an immersive simulation, with simultaneous access to ground-truth data, should allow to boost the accuracy and robustness of many algorithms. While space does not permit to expand on these issues here, the areas of active vision, adaptation and learning, and performance evaluation deserve some additional thoughts:

Active Vision: The idea of active vision is to control the sensor(s) in a goal-direction fashion, thus allowing, e.g., to focus on specific parts of the scene, track moving objects, or to compensate for platform motion [1]. Since the resulting systems are inherently closed-loop, testing is usually only possible when the whole setup is complete and operational. Simulation allows to perform meaningful experiments at a much earlier phase and without finalizing the mechanical design. Typical hardware components for active vision are motorized pan/tilt heads, motorized lenses, stereo vergence mechanisms, and gyroscopes, all of which need to (and can) be properly modeled within the simulation system.

Adaptation and Learning: Although results have been disappointing in the past, there are good reasons to believe that adaptation and learning can increase the robustness and flexibility of current vision techniques. There are many forms of learning applicable for vision, including statistical parameter estimation, clustering, function approximation, structural learning, self-organization, and neural network training. Existing applications include low-level processing, feature selection and grouping, model acquisition from examples, map learning, and 3-D object recognition [4, 3]. The use of immersive simulation can provide several important solutions with respect to the learning problem in vision:

a. Large sets of realistic examples can be created and processed with reasonable effort.

b. Supervised learning is possible without human intervention, since ground-truth data and actual model information are available from the simulator.

c. Closed-loop (or "exploratory") learning, where specific, critical training data are generated in response to the learning progress, can be performed if needed.

Of course, learning in this environment is not restricted to the vision system alone but is equally important for all other aspects of agent control. As a result, future vision and robot control system may have to spend just as many (and probably more) hours "inside the simulator" as human pilots before they are ready to perform their tasks reliably.

Performance Evaluation and Prediction: While measuring the performance of a *passive* perception system is, at least, difficult, evaluating a vision-based, closed-loop control system is almost impossible before the complete system is actually deployed. Pre-recorded test image sequences are appropriate for performance comparison of passive vision systems but are insufficient for testing *reactive* systems. Instead, we can use simulation models to perform comparative testing. In such a test suite, the system would not only be required to interpret given *images*, but to solve a particular *task* within the (virtual) environment. Effects of variations in sensor performance, such noise, jitter vibration, dynamic range limitations, etc., can be obtained without additional costs in the simulation process.

The *prediction* of complete system performance under real conditions is of equal importance, particularly for mission-critical applications. For real-time system development, it is often necessary to evaluate different hardware configurations, which are either physically available or only in the form of software emulators. Specific system components may be gradually replaced by real-time hardware components as during the development process. Tools that support this kind of complex and heterogeneous engineering process are urgently needed and emerging just now [10].

5 Summary

In the past, experiments on synthetic imagery have generally not been considered conclusive for a system's performance under real operating conditions. We believe that state-of-the-art simulation and computer graphics can provide the degree of detail and realism required to make synthetic experiments just as important as experiments on real data. The underlying assumption is that, if a vision system *does* perform well on large sets of sufficiently realistic synthetic imagery, there is a good chance it will show similar performance in reality. More importantly, if the system *fails* even on synthetic data, there will be little hope for reliable operation in practice.

The need for testing perception systems without sufficient test data or in closed-loop applications has led to many *ad hoc* simulation setups that are limited to very specific tasks. A generic simulation environment, such as the IPS described here, should reduce this expensive "reinventing-the-wheel" syndrome and make comprehensive testing more practical. In addition to providing extensive training sets for vision system validation, immersive simulation technology facilitates the implementation and testing of complex reactive systems, for which sensory data cannot be obtained off-line. This is of particular importance for the design of future perception-based systems, such as planetary rovers and similar critical systems.

References

1. Y. Aloimonos. Purposive and qualitative vision. In *Proc. DARPA Image Understanding Workshop*, pages 816–825, 1990.
2. M. Asada. Map building for a mobile robot from sensory data. *IEEE Trans. Systems, Man, and Cybernetics*, 37(6):1326–1336, 1990.
3. B. Bhanu and T. Poggio. Special section on learning in computer vision. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(9):865–919, September 1994.
4. K.W. Bowyer, L.O. Hall, P. Langley, B. Bhanu, and B. Draper. Report of the AAAI fall symposium on machine learning and computer vision: What, why and how? In *Proc. DARPA Image Understanding Workshop*, pages 727–731, 1994.
5. R.L. Cook. Stochastic sampling and distributed ray tracing. In A.S. Glassner, editor, *An Introduction to Ray Tracing*, pages 161–199. Academic Press, 1989.
6. E.D. Dickmanns and F.R. Schell. Autonomous landing of airplanes by dynamic machine vision. In *Proc. IEEE Workshop on Applications of Computer Vision*, pages 172–179, Palm Springs, CA, December 1992.
7. R. Dutta, R. Manmatha, L R. Williams, and E.M. Riseman. A data set for quantitative motion analysis. In *Proc. Conf. on Computer Vision and Pattern Recognition*, pages 159–164, June 1989.
8. K.E. Olin and D.Y. Tseng. Autonomous cross-country navigation: an integrated perception and planning system. *IEEE Expert*, pages 16–30, August 1991.
9. K. Pimentel and K. Teixeira. *Virtual Reality: Through the New Looking Glass.* Windcrest Books, 1993.
10. J. Pino, S. Ha, E. Lee, and J.T. Buck. Ptolemy: A framework for simulating and prototyping heterogeneous systems. Technical report, UC Berkeley, August 1992. EECS Dept.
11. M. Potmesil and I. Chakravarty. Synthetic image generation with a lens and aperture camera model. *ACM Trans. Graphics*, 1(2):85–108, April 1982.
12. M. Potmesil and I. Chakravarty. Modeling motion blur in computer-generated images. *Computer Graphics*, 17(3):389–399, July 1983.
13. B. Sridhar, R. Suorsa, P. Smith, and B. Hussien. Vision-based obstacle detection for rotorcraft flight. *Journal of Robotic Systems*, 9(6):709–727, September 1992.
14. G.J. Ward. The RADIANCE lighting simulation and rendering system. In *Proc. SIGGRAPH Conference*, pages 459–472, Orlando, FL, July 1994. ACM.
15. C.C. Weems, C. Brown, J.A. Webb, T. Poggio, and J.R. Kender. Parallel processing in the DARPA Strategic Computing vision program. *IEEE Expert*, 6(5):23–38, October 1991.