Towards a Holistic Approach for Mobile Application Development in Intelligent Environments

Stefan Diewald, Luis Roalter, Andreas Möller and Matthias Kranz
Technische Universität München
Lehrstuhl für Medientechnik
Arcisstrasse 21
80333 Munich
stefan.diewald@tum.de, roalter@tum.de, andreas.moeller@tum.de and matthias.kranz@tum.de

ABSTRACT

"There's An App For That" – but how do we actually develop them? While smartphones and tablet PCs are getting more and more popular and their application scenarios are growing, we still develop them using only a standard integrated development environment. Although context-based services and apps do, next to network connectivity, require lots of sensor data, the tools for providing realistic sensor data during development are still immature.

Developing, testing, debugging and evaluating those next-generation context-based apps require sensor data from the mobile device – acceleration, motion, light, sound, camera and many more sensors are available. Though, the existing development tools do seriously limit application developers by not providing the data at all or only on a very limited scale. Especially for indoor environment applications such as indoor navigation, seamless interaction between public and private displays and activity recognition and monitoring, realistic sensor data are needed and simulation support during the development phase is essential.

In this paper, we present our work towards a holistic approach for mobile application development in intelligent environments. At the example of the Android mobile device platform, we show how our approach can facilitate more effective and realistic means for mobile application development.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

Keywords

Mobile devices, smart phones, Android, middleware, ROS, simulation, virtual environment, intelligent environment

MUM'11 , Dec 7-9, 2011, Beijing, China.

1. INTRODUCTION

"There's An App For That", according to Apple, is a pretty good description for the unprecedented growth of the mobile application market in the last few years. Mobile application developers have created "apps" for almost every purpose one can think of. Those applications can be easily obtained from online market places for every popular mobile device operating system: Android Market (Google), App Store (Apple), Blackberry App World (RIM), Ovi Store (Nokia) and Windows Marketplace (Microsoft). On Apple's App Store, there are currently more than 445,000 iOS apps [2]. The Android Market offers over 275,000 mobile applications and is currently growing by about 25,000 apps every month [3]. A worldwide mobile application store revenue of $15.1 billion is expected by Gartner research in 2011 [6].

In January 2011, Appcelerator and IDC surveyed 2,235 developers on perceptions surrounding mobile operating system priorities and mobile development plans in 2011 [1]. The major general trends for mobile computing can be summarized by the following three points:

- "Always connected, personal and contextual". This is the main slogan for the future. Besides cloud services, the integration of social and contextual services will define the majority of mobile experiences.
- Rapid Development: on average, each developer will work on 6.5 different mobile applications in 2011 (2010: 2.3). The main goal is to minimize time-to-market and update cycle time for not falling behind.
- Rapid Innovation: apps are becoming more and more complex and dozens of features are added.

Although more and more complex and feature-enhanced mobile applications are developed, the time spent for development, evaluation and testing should at the same time be less than 50% of the time spent in the years before, since the developers should on average work on double the number of mobile applications [1]. Integrated development environments (IDEs) and device emulators, which are available for all major platforms, support the development process. But debugging and testing of mobile applications are still largely open issues. Especially context-aware applications, which are dependent on the various sensors of mobile devices and live data, cannot be reasonably tested currently as it would be desirable with the available tools: there is no point in "sense making" without the ability to sense! Due to

the insufficient sensor simulation possibilities of the device emulators, the testing has often to be carried out manually on real hardware and outside of the actual development systems. This consumes a lot of time and personnel expenditure. At the same time, the results from testing in artificial lab environments can only cover a very limited set of scenarios.

In this paper, we describe the idea and the implementation of a toolchain that facilitates and accelerates the complete mobile application creation process. By shifting the majority of the testing and debugging process into a virtual environment with a powerful simulation engine and background middleware, the overall development time can be minimized. Automated testing with varying sceneries and scenarios can be easily performed with it. The coupling of a virtual mobile device in the simulation with a device emulator further allows testing of mobile applications that can run on real devices without changes. For our research, we have chosen the open source mobile operating system and software-platform Android [17], since changes to the device simulator or system itself can be done very easily with the source code available. The principles are though applicable for all mobile operating systems as the development follows similar approaches currently.

2. CURRENT SITUATION AND RELATED WORK

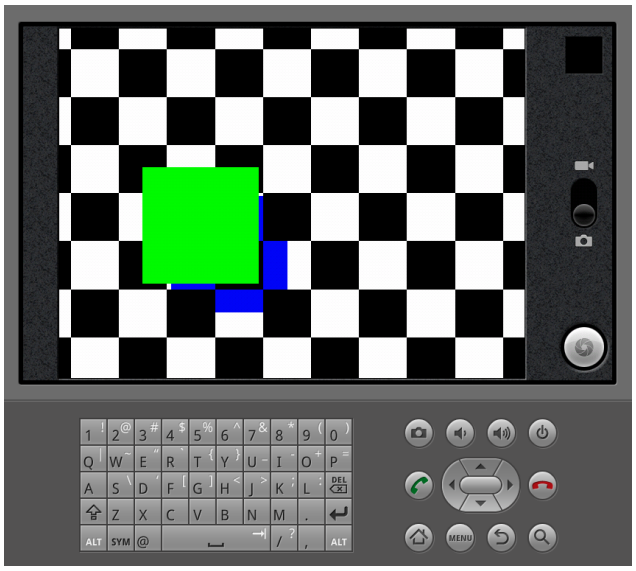
The current situation for mobile application development is described at the example of the Android platform. For getting started, a software development kit (SDK) is provided[1]. It consists of a cross-compilation toolchain and the Android device emulator which can load prepared system images for the different Android versions and hardware platforms (phone or tablet). A plugin for the integrated development environment Eclipse simplifies the source code creation and code debugging on the devices. It can further be used to control the Android emulator. The emulator offers settings for changing hardware values. For example, the screen resolution and density or the SD card size can be configured. During runtime, only a few sensors can be used and controlled:

- GPS or, more general, the position information from a GNSS (Global Navigation Satellite System), is supported when a NMEA 0183 compliant GPS device is connected to the host.
- The network connectivity and the battery status can be changed during runtime.
- The camera is only displaying a test pattern with a floating square on a checkerboard (Figure 1(a)). It cannot be fed by any external source.

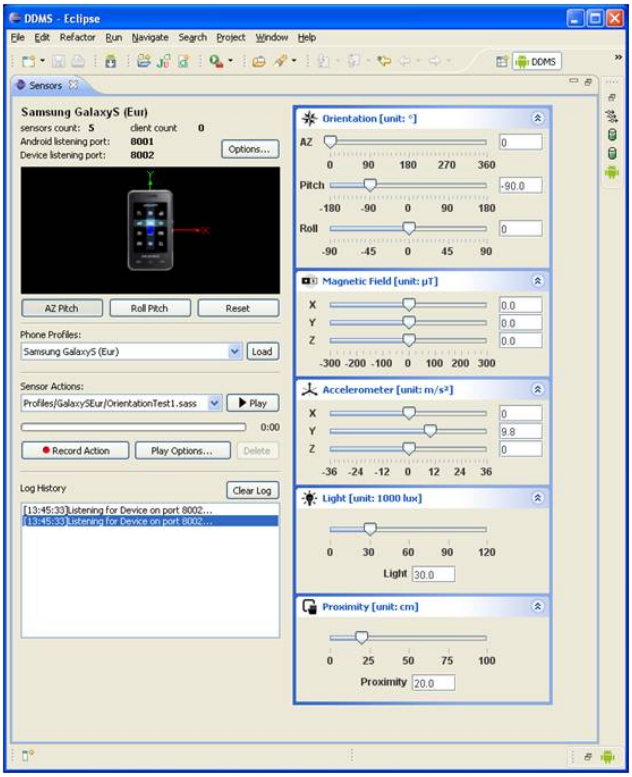
A main issue is the lack of intuitive sensor simulation support in all development environments. There is no built-in possibility to set the values for the different sensors that are commonly integrated in mobile devices, such as accelerometer, gyroscope, proximity or light sensor in a realistic way.

Since the emulator does not offer any support for external sensor data, except a real GPS device which would only emit

[1]Android SDK: `http://developer.android.com/sdk/index.html`



(a) The default, fixed camera preview of the Android emulator.



(b) The *Samsung Sensor Simulator* GUI. It enables to set several sensors' data for use with the Android emulator. Source: [22]

Figure 1. The camera preview of the Android emulator (Fig. 1(a), top) cannot be changed without adding third-party code to the application that may influence the run-time behavior. The Samsung Sensor Simulator (Fig. 1(b), bottom) can be used to adjust sensor values for the Android emulator.

a static position, there are several third-party approaches for adding those capabilities. The OpenIntents SensorSimulator [18] is a client-server application that allows setting accelerometer, compass, orientation and thermometer data. A background service has to run on the Android simulator and wrapper classes have to be used in the application instead of the standard Android *SensorManager*. The server application (on the host PC) comes with a graphical user interface (GUI) that allows to change the different sensor values via sliders and to set the device's orientation via a simple 3D representation that can be manipulated by the mouse. Gibara offers a similar client-server based solution for the camera [7]. By using a wrapper class in the application, one can feed recorded videos or live camera image from a webcam to the app. A big disadvantage of both solutions is that one has to add additional code to their application in order to use the simulated data. But adding code for performing a simulation can heavily influence the applications runtime-behaviors. In the worst case it can even happen that two different versions of an application have to be maintained: one for the emulator with the simulated sensors and one for the real device. The Samsung Sensor Simulator [22] is similar to the OpenIntents SensorSimulator, except for the fact that Samsung delivers a modified system image for the Android simulator. This allows the developers to use the standard Android API functions. Its GUI is depicted in Figure 1(b). In addition to the live manipulation of the values, the Eclipse-based tool allows to record and playback sensor data scripts. With *robotium* [9] there is a user scenario testing tool available for Android. It allows black-box testing of Activities, Dialogs, Toasts, Menus and Context Menus.

The situation is similar for other mobile operating systems and platforms. Apple's iOS Simulator does not support any hardware sensor simulation. The location framework returns a fixed unchangeable location on the simulator. Microsoft's Windows Phone Emulator, which comes with the Windows Phone SDK, has a better sensor simulation support. The accelerometer sensor simulator offers a 3D view for changing the phone's orientation. It can further replay recorded movements such as a shake input. The included location sensor simulator allows sending *position-changed* events by choosing a position on a map view. The BlackBerry simulator supports changing the tilt and orientation of the simulated smartphone in an OpenGL 3D view. The GPS position can be set via input fields. It further enables to simulate a movement from one GPS position to another by specifying a speed value. The Symbianˆ3 emulator offers a position and route simulation tool. Samsung's smartphone platform bada comes with the so called *Event Injector*. This application allows to change settings of the emulated bada device and to inject events such as incoming calls or plugging in of ear phones. Besides setting the location via a map view, the application allows simulating values of accelerometer, magnetometer, tilt sensor and proximity sensor. The emulator's camera can be fed through a connected webcam. The support of NFC simulation is a unique feature of the bada SDK. But since all these platforms are closed source, it was decided to use the open source platform Android. Basically all of the mentioned emulators can be coupled to our solution. One needs only to adapt the system to the interfaces of the respective emulators.

The idea of shifting the development and evaluation of new hardware and software into the virtual space is not completely new. Virtual product development is today used in most branches for designing, creating and evaluating early prototypes of an idea. Dongsik has shown a method for design evaluation of mobile devices using virtual reality based prototypes [10]. An approach from Lister et al. allows creating a 3D mock-up of an electronic device by simulating the complete hardware on system level [16].

3. ENHANCED MOBILE APPLICATION DEVELOPMENT

With the currently available development tools, it is not possible to create comprehensive realistic test-scenarios which can be performed automatically for sensor dependent applications. For this reason, we have created an extended and improved toolchain for the development, allowing testing more aspects of the new app software on mobile devices such as smartphones and tablet PCs. Instead of simply extending and combining the available sensor simulators, we decided to create a holistic solution that allows on the one hand automated and repeatable testing, and on the other hand a realistic and intuitive view and interaction method with the mobile device. To satisfy the requirements of the future mobile application development, our toolchain is designed to support:

- Simulation of complex interaction scenarios by taking the physicality of objects and humans into account. An example could be touching a physical object with the mobile device, e.g. for reading a NFC tag.
- Generation of pro-active, predicative and random data. An example could be the context- and location-based provision of multimedia data, e.g. in a confined room inside a building.
- Simulation of a complete mobile system (not only the software part) in order to create an overall "better" system. An example could be novel interactions, e.g. as indicated by the MagicPhone [26] scenario.
- Rapid development for interactive multimodal (mobile) applications. An example could be the rapid prototyping of an interactive public-private display interaction.

In order to reach these goals, we decided to use the concept of virtual prototyping [23]. The basis of the toolchain is a virtual environment, in which a 3D model of a mobile device can be operated. This is far more intuitive than shifting 2-dimensional sliders in external GUIs for changing 3-dimensional sensor values. The device is made functional by a software system that is interconnected with the virtual environment system and at the same time with the mobile device emulator.

3.1 ROS as middleware for distributed sensor-actuator systems

Based on our former research on the simulation of intelligent environments [21], we have chosen the Robot Operating System (ROS) as middleware. ROS is one of the major middleware systems in the domain of robotics, running for example on Willow Garage's PR2 [5] or the Fraunhofer Care-o-Bot III [8]. It has also been used for immobile robots

(ImmoBots) such as *intelligent environments* or ambient assisted living (AAL) environments [15]. The *Cognitive Office* [20] is, for example, a fully featured intelligent office environment that uses the ROS as background system for the real implementation and for a complete 3D simulation. The main advantage of this middleware is that a huge set of drivers and applications are already available. It provides device drivers, hardware and software abstraction, message passing, visualization and simulation tools (2D and 3D). In contrast to our former work, we do focus in this work on the mobile devices in intelligent environments and not the environments themselves.

The messages passing system allows fast interconnection of different so called nodes. There are several standard message types for exchanging basic types like string, integers or floats, but also more complex message types for transporting objects' poses, image data or point clouds. For exchanging special data, one can add new message types. The message system together with the hardware and software abstraction allows rapid development of additional instances for controlling or influencing the simulation flow. For example, if one wants to add noise to a simulated sensor value, a node for performing the biasing can be added between the sensor value publisher and the node that reads the value.

3.2 3D simulator Gazebo

In order to create a simulation that is something like a synthetic mirror of real world, 3-dimensional simulation is performed. The simulation is based on the robotic 3D simulator Gazebo [13]. It uses OGRE [25] for 3D rendering and the *Open Dynamics Engine (ODE)* rigid body dynamics library [24] for simulating a realistic physical behavior of virtual objects. The objects for the virtual environment can be created with all common 3D modeling tools such as Blender, Cinema4D or 3ds Max. The meshes can be connected via different types of joints that confine the degrees-of-freedom in order to enable the desired movements. By defining a mass, inertia and friction values for the virtual objects, the physics engine ODE can simulate their dynamic behavior in a realistic way.

The simulation tool Gazebo uses a robot description markup language which can be utilized for describing different kinds of intelligent robots or objects [11]. Each object consists of graphical primitives and corresponding collisions matrices. These primitive links can be connected via different joints. The joints and links can react on physical collisions in a realistic way. The collisions can at the same time also be used for launching events in the middleware. Such an event could be touching a sensor (e.g. a light switch) in the simulation, triggering the home automation system. The communication is bidirectional so that the middleware is able to control or modify the simulation as well as responding to simulation events. Since the ROS middleware can monitor and control virtual as well as real objects, it is also possible to link the real world with the virtual simulation.

An advantage of the Gazebo simulator, compared to other robotic sensor simulators, is the virtual sensor and actuator system. Sensor and actuator plugins can be assigned to any object in the simulation. Examples of available sensors are cameras, laser scanners, contact switches, force sensors or inertial measurement units (IMUs). Those sensors can be used as templates for creating other modules. For example, the laser scanner sensor could be used to model a proximity sensor [14]. It is also possible to simulate a simple battery unit that can be loaded and drained [4]. This is especially of interest for simulating mobile devices that should bear a realistic behavior. In order to model a mobile device, it is also important to have virtual displays that can be attached to any surface. For that reason, we have developed such a plugin for Gazebo which is available from our public ROS repository[2].

Gazebo is fully controllable through the ROS middleware. The physical dynamic values of every object in the simulation can be requested through a predefined background service. For controlling objects in the 3D simulation, one can apply force and wrench to the individual elements. Available route planning nodes can be used for automatic path generation through any scenery, what can simplify the creation of automated test-cases. For manual evaluation of a system, Gazebo offers an intuitive GUI for interacting with the elements in the simulation. Users can exert rotational and translational force to the physical objects. The advancing simulation time allows the evaluation of temporal behavior. Depending on the complexity of the scenery the simulation is conducted in, a faster-than-real-time simulation can be performed, if desired.

3.3 Android emulator

For simulating mobile applications that can also run on real devices without using any wrapper classes in the app itself, the Android emulator had to be modified. The principle of the system shall be described by the example of the added camera support. In order to feed the camera with any video stream from a ROS image topic that is published by the camera of the virtual mobile device in the simulation, a ROS node had to be added to emulator's system image. For that reason, we have developed a minimal ROS client in C++ that can replace the current fake hardware driver of the emulator which is responsible for drawing the test pattern. Every time an application is using the Android camera API, the ROS node starts, connects to a defined ROS master and subscribes to the image topic on which the virtual device's camera publishes its images. Whenever new image data arrive, they are copied to Android's camera buffer that can be accessed via the API. The same modification has been done for the accelerometer which can be accessed via the *SensorManager* API. The values for the accelerometer are calculated from the current physical dynamic parameters of the virtual mobile device in the simulation. Similar modifications have to be done in order to enable the other supported sensors that are described in the next subsection.

For streaming the content of the Android emulator's display back into the virtual environment, a VNC server was installed on the emulator. This allows pushing the current state of the user interface back in the simulation environment. A user "sees" what he would see on the device in the real environment. This allows realistic user interface evaluations. A VNC-to-ROS image topic converter[3] connects to the VNC server and our created virtual display maps the image onto the virtual device's display. The VNC connection can also be used for sending inputs from the virtual environment to the Android emulator. This allows simulating complete interaction scenarios. For enabling the various

[2] `https://vmi.lmt.ei.tum.de/projects/ros`

[3] VNC to ROS image topic converter: `http://www.ros.org/wiki/vnc_image_client`

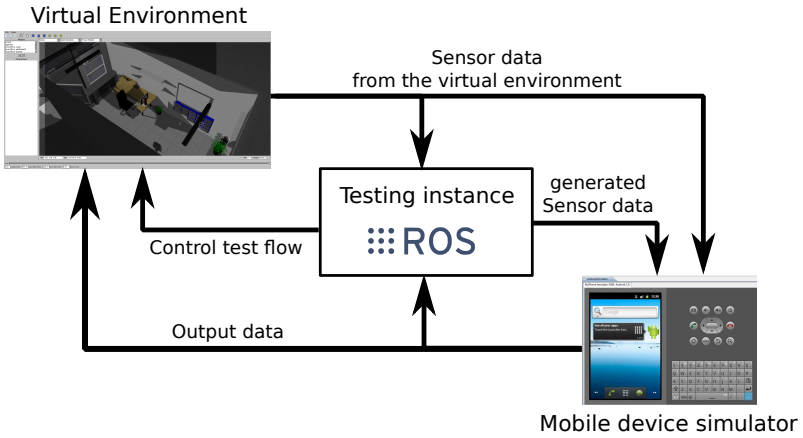


Figure 2. The structure of the proposed and integrated toolchain. A central testing instance coordinates the test flow and generates sensor data that cannot be derived from the virtual environment. It is also responsible for the evaluation of the device emulator's output. It allows recording and playback of experiments and thus simplifies the development process.

network connections from and to the Android system running on the emulator, virtual private network (VPN) support has been added by reconfiguring and recompiling the Android kernel. *OpenVPN* [19] is used for creating the VPN connection. The connection is automatically established at start-up.

3.4 The overall system

For the coordination and evaluation of automatic test scenarios, a central testing instance has to be added. The structure of the overall toolchain is depicted in Figure 2. The testing instance is responsible for the evaluation of the emulator's output. It also coordinates and controls the flow of the test process by sending commands to actuators of the virtual environment or by applying force to any involved physical object. It is further used to generate some sensors' data that cannot be simulated (yet) in the virtual environment, such as temperature or air pressure.

The virtual environment is simulated by Gazebo. The sensor data and camera images that are derived directly from the simulation are directly passed to the mobile device simulator. Currently, only the device emulator's display output is sent back to the virtual environment.

At the moment, we can simulate with our implemented toolchain the following sensors:

- **Accelerometer** and **gyroscope** by reading out the current position and orientation of the virtual model and its current linear and angular velocities.
- **Camera** via a built-in view frustum based virtual camera model.
- **Light sensor** by using a camera and calculating the image intensity.
- **Proximity sensor** by using a modified version of the included laser scanner sensor.
- **GPS** by calculating the latitude/longitude from the x,y,z coordinates.
- **NFC reader** that is triggered through a collision of NFC enabled objects.
- **Battery status** through the available battery model.

We have further created models for supporting the simulation of network connectivity, magnetic sensor, NFC writer and microphone that are not yet implemented. The following output devices are currently supported in the toolchain:

- **Light emitting diodes (LEDs)** and **displays** through our implemented dynamic texture.
- **Projectors** by using a modified version of the ROS/ Gazebo integrated texture projector.
- **Speakers** via the Gazebo integrated *Open Audio Library (OpenAL)*[4].

We intend, as part of our future work, to support the usage of haptic feedback in the simulation by adding virtual vibration motors.

4. SAMPLE DEVELOPMENT SCENARIOS

For the evaluation of the created toolchain and to highlight its potentials, we carried out several experiments using especially the camera system of the virtual device and the NFC reader. A comparison of the testing with our introduced toolchain and of manual testing on a real phone is used to show the advantages of the virtual prototyping approach.

4.1 Indoor localization using NFC tags and QR codes

In the first experiment we compared the testing process of an indoor localization method using near field communication (NFC) tags and QR codes. The NFC tags and QR codes are affixed to door plates in one of our institute's corridors. The code and the tag store the position information of the respective room. The virtual representation of the corridor is depicted in Figure 3. It was modeled using an extended interior design application. The scenery is fully textured and true to scale. The floor and wall textures correspond to the real world.

[4]Open Audio Library: `http://connect.creativelabs.com/openal/default.aspx`

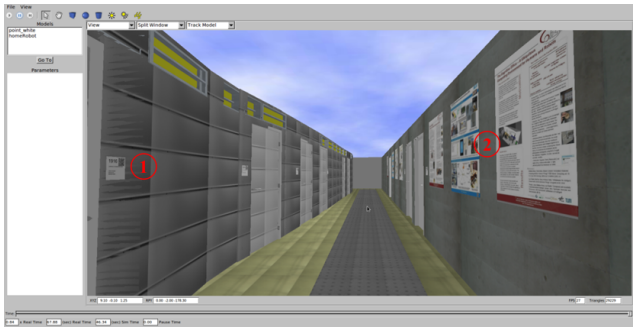


Figure 3. The Gazebo 3D simulator is showing the virtual representation of the floor. (1) The door plates next to the doors contain a QR code and a virtual NFC tag. (2) The wall-mounted posters show the same content and are at the same position as in the real corridor. The fully textured and true to scale model allows performing a rich simulation.

In the scenario a mobile device user walks along a corridor and scans several QR codes or NFC tags, in order to get their current position. This information could, for example, be used for indoor navigation solutions. For testing the scenario with a real device, the tester has to walk every time along the corridor or to use fake tags at their desk. Testing and debugging a navigation solution in this way is very hard and requires a good power of imagination. However in the virtual environment, the scenario can be played completely without leaving the office. Using one of the available route planers for ROS allows an effortless conduction of automated test scenarios. By formulating different (randomized) route constraints for every run, it is possible to create dozens of tests that vary only slightly, as it would be the case in the real world. The record and replay ability of ROS can used to save special test cases. This is important for debugging unforeseen problems.

Figure 4 shows a comparison of the real and the simulated environment. The QR code that is depicted on the door plate is detected in the real as well as in the virtual scenario. The position can be derived by evaluating the encoded data. For reading out the NFC tag, the real phone has to be put directly above the white circle in the lower right of the plate. The triggering of the read out on the simulated device is working similar. As soon as a collision between the NFC enabled virtual door plate and the virtual mobile device is detected, the data stored on the NFC tag is sent to the Android emulator. In a future version, the NFC readout can be triggered when the virtual device approaches a NFC tag and the distance falls below a definable threshold.

In Figure 5 the lighting capabilities of Gazebo are shown. The depicted example of a QR code under difficult lighting conditions can, for instance, be used to evaluate the performance of QR code readers. Reflective objects can be realized by adding appropriate shader and material scripts. This can be done without changing any source code of the simulation tools.

4.2 Indoor localization using visual information

In another indoor localization scenario, the toolchain was used without the Android emulator binding. The virtual

(a) Photography of the real world test scenario. The QR code holds an URL that contains the organizational room number. The white circle on the lower right of the door plate is a passive NFC tag.

(b) Screenshot of the virtual test scenario in Gazebo. The doorplate contains the same information as the real one. The NFC functionality is available on the whole door plate.

Figure 4. The real and the virtual Android powered smartphones in front of a door plate with QR code and NFC tag.

device is completely driven by the ROS node that made use of the camera images. In this scenario, the Parallel Tracking and Mapping (PTAM) [12] algorithm is used for estimating the camera pose in an unknown scene. Since the algorithm is not yet running on a mobile device, the testing in the real world requires to record a video and to recode the video to a format that is compatible with the application that performs the algorithm. For every desired test-case, a new video has to be captured. This consumes a lot of time and it is likely that one will be satisfied with only a small number of different recorded scenarios, which would lead to a strong limitation of test cases.

In order to allow a more comfortable evaluation of the implemented algorithm, ROS support was added to the PTAM application. This enabled to couple the visual localization algorithm to any ROS node that publishes image messages. In our case, it was connected to the camera data of the virtual mobile device in the simulation. Initial results are displayed in Figure 6: The real environment is captured and analyzed (Figure 6(a)). In the simulation environment, the

Figure 5. High fidelity shadows can be simulated in Gazebo. The example shows a simulated QR code under difficult lighting conditions.

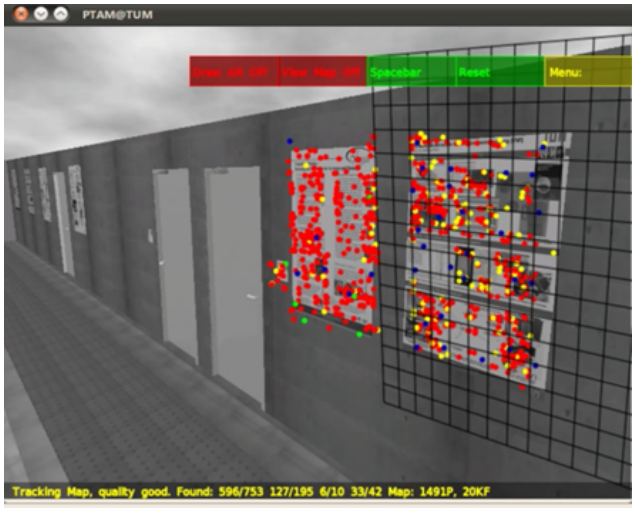
same poster is shown (Figure 6(b), the right poster). The areas and the density of the identified feature sets (colored dots in the image) are comparable. This indicates that the simulation approach does provide sufficient realism for the development.

4.3 Further possible application areas

Further application areas are home automation, and interaction with public and private displays, 2D touch interfaces and multimedia systems or novel interaction paradigms such as MagicPhone [26].

They all do require similarly rich sensor data from the mobile device which we believe our enhanced development toolchain can provide in a realistic way. The ongoing effort on improving Gazebo's material and lighting support[5] will further enhance its rendering fidelity and thus allow a more realistic simulation of vision-based projects. The incorporation of human models in the simulation can be used for performing realistic simulations of human machine interfaces. The simulation environment offers further the possibility of modeling and evaluating new input and output devices before one has to create any real hardware prototype.

(a) PTAM performed on real video input. The colored dots indicate feature sets that are used by the algorithm.



(b) PTAM in the virtual corridor. The grid indicates that the algorithm has found a reference plane and is trying to keep track of it.

Figure 6. The PTAM algorithm highlights the detected features (colored dots in the image) and draws a plane into the image. The comparison of the posters to the right shows that the algorithm performs almost identical in the real and the virtual scene.

5. CONCLUSIONS & FUTURE WORK

We have identified current shortcomings in the development process and development tools of current mobile devices at the example of the Android platform. We extended the existing toolchain by simulation capabilities, allowing the developer to access rich and realistic sensor data already during the development stage. This is expected to speed up the development of novel applications itself and to reduce the time needed for manual testing. We have implemented several examples of common interaction with mobile devices in intelligent environments to verify our approach. We outlined further application areas and thereby the potential and future impact of our implementation.

Future work will include a more formal verification by comparing directly the development of identical applications with and without the presented approach to obtain qualitative and quantitative data from professional Android app developers. The toolchain will also be used within the scope of our research oriented teaching. Students will work on various indoor location systems and the results can be compared to project outcomes from former years in which the presented toolchain had not been used.

In order to evaluate the realism of the simulated sensors in comparison to real sensors, appropriate metrics need to be defined. Future work will include finding those metrics that allow identifying weaknesses and strengths of the introduced solution.

[5]Gazebo Development Roadmap: `http://www.ros.org/wiki/simulator_gazebo/Roadmap`

6. ACKNOWLEDGMENTS

This work has been funded in parts from the German DFG funded Cluster of Excellence 'CoTeSys - Cognition for Technical Systems'.

7. REFERENCES

[1] Appcelerator Inc. and IDC International Data Group, Inc. Appcelerator IDC Mobile Developer Report, January 2011. http://www.appcelerator.com/company/survey-results/mobile-developer-report-january-2011/, Jan. 2011.

[2] AppShopper.com, LLC. App Shopper – Apps for iOS. http://appshopper.com/, Sept. 2011.

[3] AppTornado GmbH. AppBrain – Android Statistics. http://www.appbrain.com/stats/number-of-android-apps, Sept. 2011.

[4] K. Conley and J. Hsu. pr2_gazebo_plugins – ROS Wiki. http://www.ros.org/wiki/pr2_gazebo_plugins, Jan. 2010.

[5] S. Cousins. ROS on the PR2 [ROS Topics]. *Robotics & Automation Magazine, IEEE*, 17(3):23–25, 2010.

[6] Gartner, Inc. Gartner Says Worldwide Mobile Application Store Revenue Forecast to Surpass $15 Billion in 2011. http://www.gartner.com/it/page.jsp?id=1529214, Jan. 2011.

[7] T. Gibara. Obtaining a Live Camera Preview in Android. http://www.tomgibara.com/android/camera-source, May 2010.

[8] B. Graf, U. Reiser, M. Hagele, K. Mauz, and P. Klein. Robotic home assistant Care-O-bot 3 - product vision and innovation platform. In *Advanced Robotics and its Social Impacts (ARSO), 2009 IEEE Workshop on*, pages 139–144, nov. 2009.

[9] Jayway. robotium – User scenario testing for Android. http://code.google.com/p/robotium/, Aug. 2011.

[10] D. Jo, U. Yang, and W. Son. Design evaluation using virtual reality based prototypes: towards realistic visualization and operations. In *Proceedings of the 9th international conference on Human computer interaction with mobile devices and services*, MobileHCI '07, pages 246–258, New York, NY, USA, 2007. ACM.

[11] Ken Conley. XML Robot Description Format (URDF). http://www.ros.org/wiki/urdf/XML, July 2011.

[12] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

[13] N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2149–2154. IEEE, 2004.

[14] M. Kranz, P. Holleis, and A. Schmidt. DistScroll - a new one-handed interaction device. In *Proc. 25th IEEE International Conference on Distributed Computing Systems Workshops*, pages 499–505, 2005.

[15] M. Kranz, T. Linner, B. Ellmann, A. Bittner, and L. Roalter. Robotic Service Cores for Ambient Assisted Living. In *4th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth2010)*, pages 1–8, 22–25 March 2010.

[16] P. Lister, P. Watten, M. Lewis, P. Newbury, M. White, M. Bassett, B. Jackson, and V. Trignano. Electronic simulation for virtual reality: virtual prototyping. In *Theory and Practice of Computer Graphics, 2004. Proceedings*, pages 71–76, june 2004.

[17] Open Handset Alliance. Android Open Source Project. http://source.android.com/, Aug. 2011.

[18] OpenIntents. SensorSimulator. http://code.google.com/p/openintents/wiki/SensorSimulator, Aug. 2011.

[19] OpenVPN Technologies, Inc. OpenVPN – Open Source VPN. http://openvpn.net/, July 2011.

[20] L. Roalter, M. Kranz, and A. Möller. A Middleware for Intelligent Environments and the Internet of Things. In *Ubiquitous Intelligence and Computing*, volume 6406 of *Lecture Notes in Computer Science*, pages 267–281. Springer Berlin / Heidelberg, 2010. 10.1007/978-3-642-16355-5_23.

[21] L. Roalter, A. Möller, S. Diewald, and M. Kranz. Developing Intelligent Environments: A Development Tool Chain for Creation, Testing and Simulation of Smart and Intelligent Environments. In *Proc. of 7th Intl. Conference on Intelligent Environments 2011*, pages 214–221, July 2011.

[22] Samsung Electronics Co. Ltd. Samsung Sensor Simulator. http://innovator.samsungmobile.com/down/cnts/toolSDK.detail.view.do?platformId=1&cntsId=9460, May 2011.

[23] J. J. C. Schaaf and F. L. Thompson. System concept development with virtual prototyping. In *Proceedings of the 29th conference on Winter simulation*, WSC '97, pages 941–947, Washington, DC, USA, 1997. IEEE Computer Society.

[24] R. Smith. Open Dynamics Engine – User Guide. http://me.seu.edu.cn/RCBaoMing/2007/download/ode-latest-userguide.pdf, Feb. 2006.

[25] Torus Knot Software Ltd. OGRE – Open Source 3D Graphics Engine. http://www.ogre3d.org/, May 2011.

[26] J. Wu, G. Pan, D. Zhang, S. Li, and Z. Wu. MagicPhone: pointing & interacting. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, Ubicomp '10, pages 451–452, New York, NY, USA, 2010. ACM.