Pervasive Maps: Explore and Interact with Pervasive Environments

Geert Vanderhulst Kris Luyten Karin Coninx
Hasselt University – transnationale Universiteit Limburg
Expertise Centre for Digital Media
Wetenschapspark 2, 3590 Diepenbeek, Belgium
Email: {geert.vanderhulst,kris.luyten,karin.coninx}@uhasselt.be

***Abstract*—Efficient discovery of nearby devices and services is one of the preconditions to obtain a usable pervasive environment. Typical user interfaces in these environments hide the heterogeneity of the environment for end-users which often makes it hard to perceive the provided functionality. We present Pervasive Maps, an approach and tool that allows to create an intuitive user interface for exploring and controlling the environment. Pervasive Maps offers user-oriented views on the user's environment based on pictures of this environment. We show how users can model, explore and finally interact with complex pervasive environments using Pervasive Maps.**

***Keywords*-Pervasive services; ubiquitous computing; spatial user interface; mobile interaction**

I. Introduction

In a pervasive computing environment, the available computing resources and services determine the tasks that are supported. Recognizing supported tasks is fairly difficult because computing resources are distributed and sometimes invisibly integrated in the environment's physical infrastructure. Besides, the heterogeneity of computing resources in a pervasive environment not only imposes technical challenges to make resources communicate, but also demands for intuitive interaction strategies: end-users need a means to address both invisible resources and the physical resources surrounding them. This is particularly difficult in unfamiliar environments where users first need to become aware of the embedded resources and the tasks they support. With more and more computing resources and software services being embedded in our surroundings, as envisioned by Weiser [16], the digital part of our world will become even more complex and as a result harder to perceive. In this paper we present an approach to improve the visibility of the functionality offered by a pervasive computing environment and to support physical as well as remote interactions with the environment. We use the term 'pervasive application' to refer to end-user interfaces that help to orchestrate resources and their services to accomplish tasks in a pervasive environment. We argue that in analogy with desktop systems, uniform tools (e.g. a 'start menu') are required to integrate and locate resources and applications in a pervasive computing environment. Although several frameworks provide tool-support to some extent, tools are often designed for developers and not for end-users or focus on specific application domains.

Figure 1. In Pervasive Maps, a pervasive computing environment is represented by annotated photos that reveal the environment's resources and applications that provide access to related functionalities.

We propose *Pervasive Maps* (PM) as a scalable 'desktop' solution with integrated tools for modeling and interacting with pervasive environments. Pervasive Maps tries to simplify end-user control over a pervasive environment not by abstracting it, but by adopting the most concrete representations one can find of such an environment: pictures. Studies have for instance shown that pictures assist people in finding their way in previously unexplored environments [12], [14]. End-users with no technical expertise can use the PM editor to create interactive user interfaces for navigating their environment by taking pictures of the environment and annotating them with spatial and resource-specific informations. Developers can also query the environment to discover available resources. A cornerstone of our approach is the decoupling of the complex technical side of a pervasive computing system and its end-user facade. The novel aspects are twofold. On the one hand, we include a step-by-step process, supported by a tool, that walks the *end-user* through the construction of a digital representation of the environment (section III). On the other hand, we present a Web interface for exploring a pervasive environment using variants of a classical file explorer and a 'start menu' (section IV). Furthermore, we discuss the architecture of PM (section V) and outline two case studies which illustrate how our approach can be used in practice (section VI).

978-1-4244-5328-3/09/$25.00 ©2009 IEEE

II. Background and related work

Many architectures have been presented to cope with the dynamic nature of a pervasive computing environment: new computing resources give rise to new services. This brought up the need to perceive and control the current state of the environment, for instance by visualizing hidden artefacts and providing control interfaces. We discuss existing solutions in this field and compare them with our approach.

A. Middleware infrastructures and interaction tools

Well-adopted protocols such as UPnP [1] have become a default means to make services embedded in appliances accessible by the outside world. These frameworks provide methods to discover computing devices and their services, but only return limited information about discovered entities. As a result, it remains difficult for end-users to differentiate between similar resources based on a description, especially in complex environments with many embedded computing resources. Various other middleware systems have been proposed for building context-aware pervasive services [5], [15], [9]. This landscape of diverse technologies for creating and deploying pervasive services demands for a generic solution to discover and leverage them in useful ways. Current out of the box products such as the Philips Pronto series[1] deliver an integrated user interface for multi-room control, but merely deal with IR/RF codes, rely on proprietary protocols and lack support for context-aware third-party applications. Research infrastructures that support interaction with pervasive computing environments using mobile devices include ICrafter [10] and the Personal Universal Controller (PUC) [8]. In ICrafter, services register their user interfaces with the ICrafter Interface Manager (IM). The IM is a service on its own with a user interface that migrates to the user's personal device. This interface provides an overview of the available services in the environment and a custom user interface is aggregated to control selected services. The PUC further extends this approach by delivering automatically generated user interfaces to control appliances within an environment. The main focus in these frameworks is on the generation of user interfaces to *control* appliances rather than on a user interface to *explore* an environment as in our research.

B. Visualizing a pervasive environment

Interaction with a pervasive environment based on video stills and photographs has been researched before [11], [4], [13]. The NaviCam [11] and the Digiscope [4] focus on real-time use and hence can not be used outside the target environment. For instance, the Digiscope uses a semi-transparent tablet that provides users with an augmented reality view of invisible information in the environment. In [13], a 'u-Photo' is annotated with eyemarks: physical entities that appear on a photo, representing pervasive services. By clicking on an eyemark, a user interface for interacting with the service is overlay on the photo. u-Photos are generated on the fly by taking pictures of an environment in which visual markers are embedded. Whereas u-Photo dynamically delivers annotated photos for interaction with the user's current environment, Pervasive Maps provides a full desktop interface in which photos are integrated in advance. In our solution, photos are primarily used to build up a model of the environment and in second stage as a means for discovering and interacting with remote resources from mobile devices. We consider 'services' to be part of the underlying architecture, and rather focus on 'resources' embedded in 'places' in the environment's user interface to closely match the real world situation.

The Personal Environment Controller (PECo) [6] incorporates a 3D visualization of a physical environment in conjunction with access to personal media. Devices are represented in the virtual environment as interactive 3D objects and provide access to control user interfaces. The PECo system is similar to our approach in the sense that it relies on a digital representation of the environment to interact with the real world. However, designing 3D models is a complex task which is unlikely to be performed by end-users. In Pervasive Maps end-users design their own virtual environment by taking pictures and annotating them. Moreover, our system is different from PECo in the way resources and applications are treated: a resource is 'opened with' a suitable application and is not limited to a UPnP device with an embedded user interface. Instead, we allow passive objects with no computing power to become interactive parts of the virtual environment as well.

III. Modeling the environment

We use a dynamic environment model as internal data structure to represent the environment context. This model is built up from *place*, *sight*, *resource* and *application* concepts and relationships that apply between these concepts. Resources are organized into places that make up the environment and sights provide views on the various places and their embedded resources. Since resources are diverse in terms of embedded technology, visibility and mobility, we subdivide them in four categories:

- A *resources* are (mobile) interaction devices with an embedded computing platform such as mobile phones and UMPCs. This class of resources is typically used to interact with the environment, but can also host and share services that give rise to new pervasive applications.
- B *resources* are everyday, often stationary computer-augmented resources such as a smart fridge or a networked light. These resources are part of the physical environment, but also have a digital dimension that can be accessed through A resources.
- C *resources* are invisibly embedded in the environment, but can be discovered by applications. An example is a

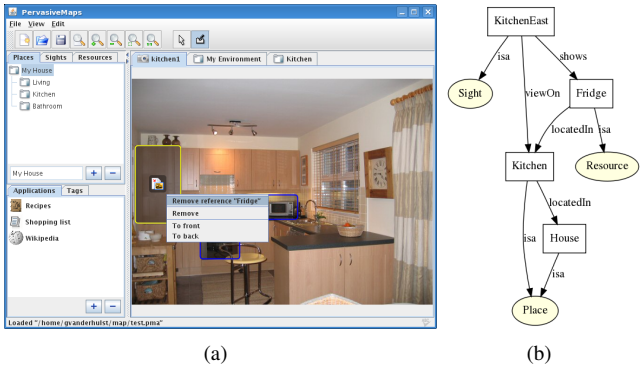
[1] http://www.pronto.philips.com/

light sensor whose output is processed by an application.
- D *resources* correspond to physical objects without computing platform that refer to other resources or applications. For example, a pile of DVDs could point to a media player, whilst a television panel can act as an abstraction for the set-top box it is attached to.

Most B, C and D resources are known in advance and have a fixed location in the physical environment, while A resources are often mobile and not known in advance. Hence we can distinguish between resources that can be integrated in advance in an environment model and resources that announce their presence at runtime to seamlessly integrate with the pervasive computing environment. We present a tool that enables end-users to create a personalized model of their environment. This model is created beforehand, but continuously adapts to changes that occur when resources enter or leave the environment on the fly.

A. Integrate places, sights and resources

Pervasive Maps uses a graph structure that correlates resources and their associated places as a model for a pervasive environment. An environment is subdivided in different locations (called 'places') and sights on a location reveal the resources contained in a particular place. End-users can create digital representations of the environment with the PM editor, as shown in figure 2. The modeling process consists of four steps, discussed below.



(a) (b)

Figure 2. End-users can create a model of their environment using the PM editor (a). The tool produces a graph of the environment relating places, sights, resources and applications (b).

Define places. First, we build up a spatial model of the environment. The end-user identifies different places of interest in the environment which are hierarchically organized based on their relative locations. For example, a place 'house' contains a place 'first floor' with places 'kitchen', 'living room', etc. By subdividing an environment in different places, we can better deal with the environment's complexity and navigate to resources based on their relative location, i.e. the place they are in. For each place in the environment, an optional floorplan diagram is provided which is created with tools such as Google SketchUp. Floorplan sketches are then imported in the PM editor and annotated with spatial information such as the dimensions of a place and its orientation (e.g. north-east).

Define sights. The second step consists of picturing the different places in one's environment using a digital camera. In particular, images that clearly depict B or D resources embedded in a place are of interest here. Currently, we assume camera shots are taken frontally and wider images are preferred over close ups as they generally contain more reference points that help to map the contents of a digital image onto the real world and vice-versa.

Define resources. Next, the third step involves integrating available resources into the digital environment. Hereby the end-user is assisted by the computing system which automatically proposes to integrate B and C resources it discovers on the network. D resources, i.e. resources that are not networked, should be defined manually with a name and icon. Additional attributes describe a resource's capabilities and technical bindings.

Connect places, sights and resources. Finally, places, sights and resources need to be connected. Places and resources can be marked on imported images. This relates a selected area on the image with a particular place or resource, similar to tagging people on photos that appear on social network sites such as Facebook[2]. Since photos provide a 2D view on a 3D environment, objects might overlap, e.g. when resources occlude each-other. By means of a context menu with 'bring to front' and 'bring to back' actions, z-orderings are added to tagged objects. Hence we partially reconstruct relevant 3D information that has been lost in the photo. Furthermore, sights are marked as a spot on a place's floorplan which corresponds to the location where the sight image was taken.

Note that we can derive extra information from the environment model that was not explicitly provided by end-users. For example, since a sight provides a view on a place, all resources that were marked on the sight are also located in the place. We can retreive the relative distances for the resources that are marked on a place's floorplan. When the user's position is known, the relative distance between the user and the resources can also be calculated. It is also possible to derive facts from sights such as 'the microwave is situated left from the fridge'. Moreover, when sights and places are further annotated with spatial information (e.g. compass headings, see section IV-C) we can acquire information about the direction in which a resource is located, e.g. in the north-east of the room.

B. Integrate applications

We define a pervasive application as a front-end for one or more distributed services. In this context, a service is a functional component published on the network, while an

[2]http://www.facebook.com/

application corresponds to an end-user interface that interacts with services in its back-end. The ensemble of a service and its embedded user interface such as a UPnP service with a HTML presentation page can be considered as an application as well. We focus on applications because users are already accustomed to add, remove and use applications on desktop computers. An application in Pervasive Maps is identified by a URL from where its presentation can be downloaded and has extra properties that specify the type of user interface (e.g. HTML or VoiceXML), version information, etc. The PM editor can discover applications on the network (i.e. service and user interface ensembles) and includes an installer for third-party applications. We use AJAX-based applications since they run on almost every device with a Web browser. A distinction is made between private and public applications.

A *private* application runs in the resource's firmware and is exclusively used to control the resource it resides on. For example, an application embedded in a kitchen appliance is private, as it can only be used to steer that appliance and e.g. not a similar one which will run its own copy of the application. A *shared* application runs on any computing device from where it can operate other resources, similar to a word processor that can open and save documents on a network. An example is a light application that takes as input a light resource whose state it can observe and change. Publicly available (Web) applications such as a weather forecast service or an online recipe book can be considered shared applications as well. Private applications are associated with a single resource, while shared applications can operate a range of similar resources or are resource-independent.

We relate resources with applications through meta-data: tags attached to resources and applications are matched at runtime. A positive match indicates that an application can receive a resource as input parameter (see section III-C).

C. Tag and search

The environment model is further enriched with semantics describing the integrated resources and applications. End-users can attach *isa* tags to resources that describe the type of the resource, e.g. a fridge, a movie collection, etc. Likewise, applications can be annotated with *domain* and *resource* tags. Domain tags describe the domain of an application (e.g. 'cooking', lights', etc) and are used to categorize applications. Resource tags, on the other hand, help to link resources and applications on the fly. For example, an application with resource tag 'kitchen appliance' will be associated with all resources tagged as 'kitchen appliciance'.

We use the WordNet lexicon[3] to disambiguate between the different meanings a word might have. When tagging a resource or application, the user enters a keyword which is looked up in the WordNet lexicon. The user then selects the intended meaning of the word from a list of word senses which is attached as a tag to the resource. The linguistic relations that apply between words can then be exploited to search for resources and applications. For example, an application tagged with the keyword 'light' semantically matches a search term 'lamp', provided that both keywords are used in the same sense (i.e. a source of illumination). Similar, a keyword 'piano' will match a tag 'musical instrument' as the former is a hypernym of the latter.

[3]http://wordnet.princeton.edu/

IV. Exploring the environment

An environment model created using the PM editor is rendered as a Web interface optimized for mobile devices. This interface provides access to the available applications, either by selecting them from a menu or by browsing through the graph of places, sights and resources.

A. Application-based navigation

Pervasive applications can be accessed in a traditional way from a list of application names and icons. However, since applications dynamically become (un)available, the list of applications can grow long, making navigation difficult. To overcome this, we render a tag cloud from the different domain tags an application has assigned, as depicted in figure 3. Domain tags automatically categorize applications and allow end-users to locate applications even without knowing their names. This is particularly useful to find applications related to a certain domain such as cooking or to find an application suitable for a certain task such as adjusting the temperature.

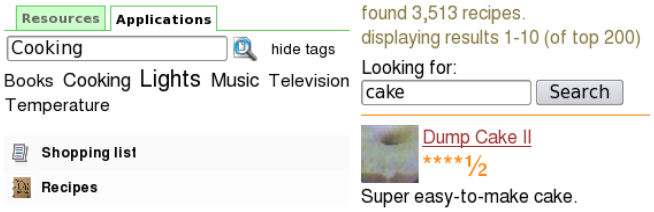


Figure 3. Applications are organized by their tags. An online recipe book that was integrated in the environment as a shared application is found using a 'cooking' domain tag and can be linked with e.g. an oven via a resource tag.

B. Resource-based navigation

Resources are either listed or shown on top of a map. In the list view, places act as folders that hold resources; navigation is menu-based. Opposed to this, the map view is photo-based: users navigate places and sights by selecting interactive parts on floorplan and sight images. For example, a door marked on a photo showing part of the kitchen might point to the living place. Images reveal available, possibly invisibly embedded resources in the environment that can be accessed from anywhere. To highlight resources, we render photos semi-transparent and overlay them with solid

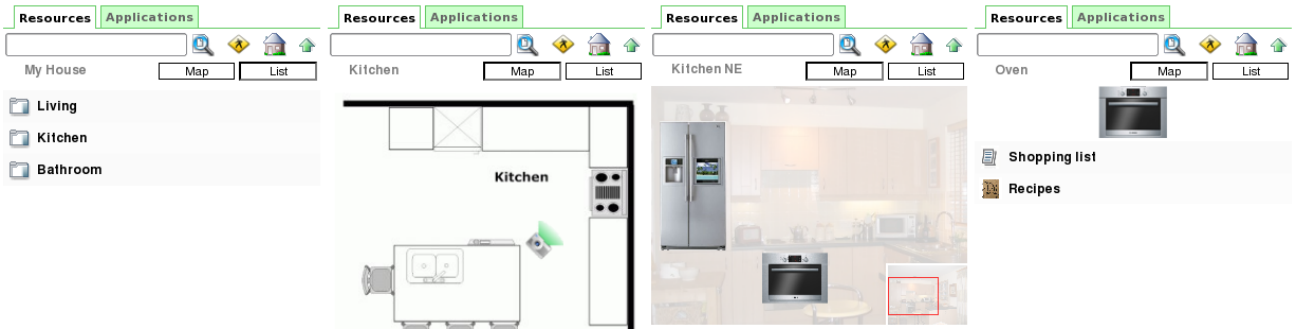


Figure 4. A combination of list- and photo-based views provide access to the resources embedded in the environment. From left to right, these views show a list of places and resources; an interactive floorplan of a place with a sight reference on it; an interactive sight on which resources are marked; a list of applications associated with a selected resource.

pictures of resources. In dense places with many resources, z-orderings help to differentiate between nearby resources and resources further away. However, we acknowledge that we currently do not take into account resources which are completely occluded by other resources; this situation should be avoided when taking a picture, if possible.

Figure 4 illustrates the concept of resource-based navigation: different views of the environment help to locate resources and find relevant applications. Note that users can switch between list and map views on the fly; the navigation context is preserved. One can zoom and pan floorplan and sight images, making them usable on devices with a limited screen size as well. In the environment depicted in the figure, we attached a resource tag labeled 'kitchen appliance' to applications related to the domain of cooking. These applications thus become associated with real-world kitchen appliances such as an oven – tagged as 'oven', a hyponym of 'kitchen appliance'. When navigating to a resource and selecting one of its related applications, the application is loaded with the resource as input so that the application can adapt to the context of use. For example, when opened with an oven resource, the recipes application shown in figure 3 could only suggest oven dishes.

C. Location and orientation tracking

Context-awareness and in particular location-awareness is considered key for interaction with pervasive environments. Indoor tracking systems are still being investigated extensively in order to improve their accuracy and to reduce costs and setup time. Due to the diversity of real-time positioning techniques (e.g. RFID-based such as LANDMARC [7] or Wifi-based as discussed in [2]) we have not considered a default solution, but rather outline how different solutions could be used in conjunction with resource-based navigation in Pervasive Maps. The places and sights that are part of an environment include spatial markers as a special type of meta-information. These markers can be assigned arbitrary data and are added to a place's floorplan using the PM editor. For example, when used in combination with an RFID-based tracking system, the RFID-tag in the real-world would be matched with a spatial marker in the digital world having the same identifier. Location data is then obtained from the digital marker such as the place it is part of and its relative position w.r.t. this place. Tracking solutions that rely on signal strength (e.g. wifi-based solutions) can use spatial markers as reference points to store sample data. In this case, a place is overlay with (a grid of) location markers, depending on the tracking system at hand. Algorithms translating real-time positioning data into user-centric location information run as services on a user's handheld device. By walking through the real-world environment with a spatial tag that can be tracked by a positioning system and marking the current location on a floorplan using a mobile computing device, end-users can learn the pervasive computing system to become location-aware.

Apart from location, the direction someone is pointed to gives insight into the resource(s) one is facing. This information can be exploited to list just those resources one is looking at, or to present a sight matching the user's view in the real-world that reveals the available resources on a digital photo.

D. Augmented reality

An environment modeled using PM can be considered a virtual environment in which resources, sights and places refer to the real-world. By embedding tags in the physical environment that refer to the virtual environment, digital and real worlds get intertwined. These tags can then be used for several purposes:

- Locate oneself in an unfamiliar environment, similar to 'I am here' information panels. Scanning a tag shows information about the current place the user is in.
- Access the digital dimension of a resource. Scanning the tag pops up a list of applications related to the resource.
- Create a dedicated 'start menu' in the real world composed of tags. Scanning a tag loads an application.

Augmenting one's surroundings with digital references contributes to an enhanced interaction experience [11], [4]. Al-

though PM does not require an environment to be augmented with digital artifacts, users can benefit from augmented places, e.g. to avoid navigation in the virtual world.

V. Pervasive Maps architecture

Many pervasive software architectures are service-oriented and not application-oriented, meaning they offer programmatic access to distributed functionalities, but lack a proper integration of user interfaces that make features available to end-users. Although service frameworks such and UPnP offer support for attaching UIs to services (e.g. through a presentation URL), their main focus is on discovery and control interfaces. In Pervasive Maps, however, the main focus is on end-user applications (i.e. user interfaces) which rely on one or more services in their back-end. The architecture of PM consists of three major parts:

- *PM host:* The PM host platform is installed on a computer connected to the environment's internal network, denoted as the PM gateway. It serves a Web interface that provides access to the environment's resources and applications.
- *PM client:* A PM client platform is optionally installed on personal devices. It assists end-users in seamlessly accessing a PM environment by taking away the need to start a Web browser and manually browse to a PM host.
- *PM editor:* The PM editor is a design and configuration tool for the PM host (see section III). It generates an environment model from user input, includes an application installer and supports basic user administration.

A. Pervasive Maps host

The PM host is a modular platform written in Java with a built-in Web server. The core task of the PM host is to serve a dynamic Web interface for interacting with the environment. This interface is rendered from an environment model provided by the PM editor. It communicates through Javascript and JSON with its Java-based back-end implementation, just like any other deployed Web application. When a resource is opened with an application, the resource object is serialized into a compact JSON variant (known as 'RISON') and attached as a parameter to the application's URL so that an application can quickly validate its input and execute. An authentication component is used to identify the users connected to an environment. Apart from security reasons, the notion of users and the devices they are using is useful information for context-aware applications [3].

B. Pervasive Maps clients

Personal devices, denoted as A resources according to the classification in section III-A, fulfill a double role in the environment: they primarily act as a means for interacting with the environment but also offer a computing platform on which services can be installed. To access the PM Web interface from a device, no other software but the device's native Web browser is required. However, since the IP address of the PM host gateway might be unknown, we install a minimal PM client application on personal computing devices that discovers a PM host and forwards its presentation URL to a Web browser. In addition, PM clients can be extended with plugins to access the device's hardware/sensors. For example, we extended an iPhone PM client with a plugin to support scanning QR tags with the phone's built-in camera. Other plugins can be developed to further exploit the interaction capabilities of a device.

VI. Case studies

As a first evaluation of our system we have implemented two case studies, depicted in figure 5. The first case study addresses the scenario of a user visiting an interactive museum. In this scenario, the user explores and interacts with artifacts in the museum, both from within the physical museum as from a remote location. The second case study discusses a pervasive application to enhance the experience of playing a musical instrument with projected music scores.

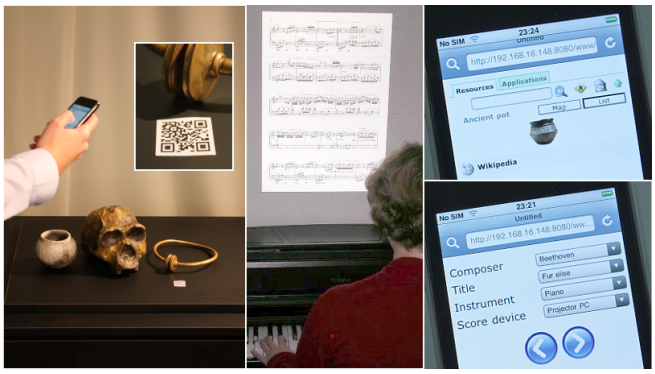


Figure 5. An iPhone running the PM Web interface is used to interact with a museum environment and to select and navigate digital scores.

A. Museum environment

We augmented a lab environment with ancient historical objects such as old vases to simulate a single room museum. The Pervasive Maps framework was deployed on a computer in our museum with two applications installed: one application provides a user interface to control the background music playing in the museum and the other one uses Wikipedia to provide background information about the museum's artifacts. Using the PM editor, we designed a place 'museum' and different resources tagged as 'artifact' that correspond to the objects in the museum. As a first experiment within this case study, we asked 8 participants with different backgrounds to take pictures of the museum using a digital camera and to integrate and annotate these as sights in the virtual environment. This included marking the various artifacts on the image and linking them with resources we previously integrated. Next, we asked them to remotely visit the museum over the internet, using their

own laptop and preferred Web browser. To guide the visit, we posed a few questions about the artifacts in the museum. The answer to these questions could be found by opening an artifact with the Wikipedia application. Furthermore we asked test users about the background music that was playing in the museum, which they could find out by navigating to an application tagged 'music'.

In a second experiment, we attached printed QR-codes to the artifacts in the museum to identify them in the digital world. These codes were generated from the artefact's URIs using the BeeTagg online code generator[4] which automatically optimizes the size of a URI to fit on a QR-code. We also added QR-tags on the four walls of the museum room which correspond to spatial markers. Using the PM editor, we integrated spatial markers on the room's floorplan and added a compass heading to each of them. In 'map mode', the spatial information obtained by scanning a spatial marker is used to pan and rotate the place's floorplan. In 'sightseeing mode', the spatial context of the user is used to select the nearest sights the user's device is pointed to. We asked the same participants of the previous experiment to physically visit the museum using an iPhone with the BeeTagg reader application installed. When a QR-code is scanned, the BeegTagg reader translates the code into a URI. This URI is then passed as a parameter to the PM Web interface which is loaded in a browser.

After these experiments we presented the participants with a survey in which they had to indicate their degree of agreement with a number of statements on a five-point Likert scale. The statements and results of the survey are presented in table I. We only included average results since the distributions of responses were all consistent with a standard deviation of less than one. Results acknowledge that the PM editor is still a prototype application and needs usability upgrades. Nevertheless, users were able to create a digital museum environment and interact with it through a Web browser. They particularly stressed their interest in a PM user interface for their own places.

	Statement	Result
S1	It was easy to learn how to use the PM editor.	3.0
S2	Taking pictures of the museum and annotating them in the PM editor was a positive experience.	3.2
S3	Interacting with the museum from a Web browser felt intuitive.	3.8
S4	I had no difficulties in finding more information about artifacts in the museum.	4.0
S5	I found it easy to find out what kind of music was playing in the museum.	4.1
S6	A PM interface to observe and control resources in my own house or apartment would be useful.	4.5
S7	I believe it is simple to map photos on the real world and locate resources that can be interacted with.	3.0
S8	A combination of digital photos and physically tagged resources improves interaction with the environment.	4.0

Table I
SURVEY STATEMENTS AND AVERAGE RESULTS.

B. Digital scores

In this case study we focused on the integration of a dedicated prototype application in Pervasive Maps. We designed a scalable pervasive application for musicians that displays digital scores in the physical environment which assist them with playing their instruments. The setup of the environment is as follows. The PM software is installed on a gateway computer and its Web interface is accessed through an iPhone. A small PC running a linux-based OS is connected to a projector aimed at the wall above a piano. The computer runs a UPnP service implemented in less than 50 lines of Perl code that exports a control interface for navigating PDF documents. The service leverages XPDF as document renderer and is published on the network using a Perl UPnP DeviceManager implementation[5]. Incoming control requests such as 'SetDocument' or 'NextPage' are dispatched to Perl functions that pass data to an XPDF process. Furhermore, we created 'DigiScores', an AJAX-based Web application that provides access to an archive of score documents indexed on title and composer. DigiScores expects as input a PDF service for rendering scores and a musical instrument to determine the type of scores needed, as piano scores differ from guitar scores for instance. Service and instrument type can be selected from the application's user interface along with preferred scores as shown in figure 5. The set of available PDF services is discovered using an integrated UPnP proxy which is also used to invoke their functions. Listing 1 shows part of the JavaScript code that initiates the DigiScores application.

Next, we modeled a PM environment that consists of a single place, a living room, with a single sight showing a piano which we marked and annotated with a WordNet tag. The tag stipulates the marked area on the sight corresponds to a piano in the sense of keyboard instrument, and not to low loudness which is an alternative meaning of the noun 'piano'. Next, we installed the DigiScores application via the PM editor and attached it a WordNet tag labeled 'musical instrument'. Using the iPhone, we can now directly access the DigiScores application via the PM application menu. Alternatively, one can navigate to the piano resource and 'open it' with the DigiScores applications whose tag semantically corresponds with the one from the piano. While the DigiScore application does not know about the piano resource, the resource is linked with the current application state, i.e. the selected PDF service, instrument type, last scores, etc, which is automatically cached. By bookmarking the piano as a QR tag and attaching it to the piano in the real world, the DigiScores application becomes directly

[4]http://www.beetagg.com/

[5]http://perlupnp.sourceforge.net/

accessible on the iPhone and will remember the correct settings from a previous session.

```
var r = rison.decode_object(params['r']); // input resource
for (var i=0; i<r.tags.length; ++i) // set instrument type
        if (isSupportedInstrument(r.tags[i].lemma)
                { setInstrument(r.tags[i].lemma); break; }
var e = pm.getEnvironment(); // reference to environment model
var xpdfs = pm.findServices('upnp:xpdf'); // find services
for (var i=0; i<xpdfs.length; ++i) {
        var d = e.getResourceById(xpdfs[i].runsOn);
        var s = new XPDFService(xpdfs[i].URL);
        addScoreDevice(d.name,s);
}
```

Listing 1. The DigiScores application initiates by analyzing its input and discovering suitable services.

VII. Conclusions

We have presented Pervasive Maps as an approach to explore and interact with a pervasive environment using mobile devices. Our approach is inspired by desktop operating systems in which a file explorer and a start menu are indispensable tools. We introduce similar tools for interaction with a pervasive computing environment based on a classification of the different types of resources that inhabit a typical environment and the observation that pervasive services are often invisible background processes instead of applications with an end-user interface. We developed a tool that enables end-users to model their own environment by taking pictures and annotating these with extra information. The places and resources identified in the modeling phase can then be explored and interacted with through a Web interface, also from remote locations. Furthermore, we can improve co-located interaction by embedding tags such as QR codes in the physical environment which refer to resources or applications in the digital environment.

A first user study pointed out that the appreciation factor of Pervasive Maps is high, although the PM editor still lacks a satisfactory stage of usabilty. To improve the modeling tool, users suggested to inlude a step-by-step wizard for creating a digital version of their environment.

References

[1] *Universal Plug and Play (UPnP).* http://www.upnp.org/.

[2] Yiqiang Chen, Zhuo Sun, Juan Qi, Derek Hao Hu, and Qiang Yang. LoSeCo: Location-based Search Computing for Pervasive Device Augmentation. In *Proc. of the 7th IEEE Int. Conf. on Pervasive Computing and Communications (PERCOM'09)*, pages 1–6. IEEE Computer Society, 2009.

[3] Anind K. Dey. Understanding and Using Context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.

[4] Alois Ferscha and Markus Keller. Real Time Inspection of Hidden Worlds. In *Proc. of the 7th IEEE Int. Symposium on Distributed Simulation and Real-Time Applications (DS-RT'03)*, pages 51–59. IEEE Computer Society, 2003.

[5] Robert Grimm. One.world: Experiences with a Pervasive Computing Architecture. *IEEE Pervasive Computing*, 03(3):22–30, 2004.

[6] Ali A. Nazari Shirehjini. A novel interaction metaphor for personal environment control: direct manipulation of physical environment based on 3D visualization. *Computers & Graphics*, 28(5):667–675, 2004.

[7] Lionel M. Ni, Yunhao Liu, Yiu Cho Lau, and Abhishek P. Patil. LANDMARC: Indoor Location Sensing Using Active RFID. In *Proc. of the 1st IEEE Int. Conf. on Pervasive Computing and Communications (PERCOM'03)*, pages 407–415. IEEE Computer Society, 2003.

[8] Jeffrey Nichols and Brad A. Myers. Controlling Home and Office Appliances with Smart Phones. *IEEE Pervasive Computing*, 5(3):60–67, 2006.

[9] Hubert Pham, Justin Mazzola Paluska, Umar Saif, Chris Stawarz, Chris Terman, and Steve Ward. A Dynamic Platform for Runtime Adaptation. In *Proc. of the 7th IEEE Int. Conf. on Pervasive Computing and Communications (PERCOM'09)*, pages 1–10. IEEE Computer Society, 2009.

[10] Shankar Ponnekanti, Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd. ICrafter: A Service Framework for Ubiquitous Computing Environments. In *Proc. of the 3rd Int. Conf. on Ubiquitous Computing (UbiComp'01)*, pages 56–75. Springer-Verlag, 2001.

[11] Jun Rekimoto and Katashi Nagao. The World through the Computer: Computer Augmented Interaction with Real World Environments. In *"Proc. of the 8th ACM Symposium on User Interface Software and Technology (UIST'95)"*, pages 29–36, 1995.

[12] Tracy Ross, Andrew J. May, and Simon Thompson. The Use of Landmarks in Pedestrian Navigation Instructions and the Effects of Context. In *Proc. of the 6th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'04)*, pages 300–304, 2004.

[13] Genta Suzuki, Shun Aoki, Takeshi Iwamoto, Daisuke Maruyama, Takuya Koda, Naohiko Kohtake, Kazunori Takashio, and Hideyuki Tokuda. u-Photo: Interacting with Pervasive Services Using Digital Still Images. In *Proc. of the 3rd Int. Conf. on Pervasive Computing (PERVASIVE'05)*, pages 190–207, 2005.

[14] Johannes Schning; Keith Cheverst; Markus Lchtefeld; Antonio Krger; Michael Rohs; Faisal Taher. Photomap: Using Spontaneously taken Images of Public Maps for Pedestrian Navigation Tasks on Mobile Devices. In *Proc. of the 11th Int. Conf. on Human-Computer Interaction with Mobile Devices and Services (MobileHCI'09)*. ACM, 2009.

[15] Geert Vanderhulst, Kris Luyten, and Karin Coninx. ReWiRe: Creating Interactive Pervasive Systems that cope with Changing Environments by Rewiring. In *Proc. of the 4th IET Int. Conf. on Intelligent Environments (IE'08)*, pages 1–8, 2008.

[16] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, 1991.