Virtual Reality
DOI 10.1007/s10055-010-0158-6

In-Place Augmented Reality

Oriel Bergig · Nate Hagbi · Jihad El-Sana · Klara Kedem · Mark Billinghurst

Received: 22 July 2009 / Accepted: 8 March 2010
© Springer-Verlag London Limited 2010

Abstract In this paper, we present a vision-based approach for transmitting virtual models for Augmented Reality, which we name In-Place Augmented Reality (IPAR). A two-dimensional representation of the virtual models is embedded in a printed image. We apply computer vision techniques to interpret the printed image and extract the virtual models, which are then overlaid on the printed image. The main advantages of our approach are: (1) the image of the embedded virtual models and their behaviors are understandable to a human without using an AR system and (2) no database or network communication is required to retrieve the models. To demonstrate the technology and test its usability, we implemented several applications and performed a user evaluation. We discuss how the proposed technique can be used for the development of applications in different domains such as education, advertisement, and gaming.

O. Bergig (✉) · N. Hagbi · J. El-Sana · K. Kedem
The Visual Media Lab, Ben-Gurion University, Beer-Sheva, Israel
e-mail: bergig@cs.bgu.ac.il

N. Hagbi
e-mail: natios@cs.bgu.ac.il

J. El-Sana
e-mail: el-sana@cs.bgu.ac.il

K. Kedem
e-mail: klara@cs.bgu.ac.il

M. Billinghurst
The HIT Lab NZ, University of Canterbury, Christchurch, New Zealand
e-mail: mark.billinghurst@hitlabnz.org

Keywords Augmented Reality content · Content transmission · Model embedding · Dual perception encoding · In-Place Augmented Reality

1 Introduction

Augmented Reality (AR) enhances user perception by supplementing the real world with virtual content. Usually, the virtual content is stored in a library or model file and fetched by the AR system for rendering. In this paper, we present an alternative approach where the content is encoded in the real world itself.

Content libraries in AR systems are maintained either locally or remotely. Updating such libraries that are stored locally on AR devices is often a complicated task. On the other hand, remote databases are easy to update, but they require network communication. As more AR applications are developed, the amount of virtual content that is created rapidly increases. Since a large number of providers are involved in creating the content, managing and indexing content become a challenge. In addition, retrieving the content by the user may be complicated in some scenarios. For example, consider an AR advertising application where the user can point their mobile phone at an image in a newspaper and see a virtual scene superimposed over the printed image. In this case, to retrieve AR virtual models over the cellular network, the user could send an instant message containing the identification number of the advertisement to the appropriate provider. The AR application content is then downloaded to the phone over the cellular network. When the advertising application is running, it tracks a pattern in the newspaper advertisement and overlays the virtual models on top. In such a scenario, the user is required to take an active role in retrieving the content, which requires several manual operations before the content appears.

In many scenarios, the content must be published and updated frequently (e.g. newspapers, brochures, public displays). An Augmented Reality application for these scenarios might require a pattern to be included with the content in addition to the electronically distributed virtual content. In the previous advertising example, the advertiser publishes the advertisement and a pattern in the newspaper to make the virtual content available over the cellular network.

Designing AR systems and content delivery networks to properly scale over time is also a challenging task. Such systems must meet unpredictable demand bursts. In the advertising example, the advertisement may be used as a coupon on a limited time sale, which can cause a large number of simultaneous access requests for the same AR content and place significant load on the content servers.

A convenient way to generate virtual models without having to publish and maintain a virtual content library can lay the foundations for a new type of AR applications and broader acceptance of the technology.

This paper introduces the idea of vision-based transmission of AR models. The proposed approach combines the advantages of local and remote libraries. Vision-based transmission does not require a database or communication and still provides a flexible way for updating content in scenarios where printed content is updated frequently. The transmission is achieved by embedding the virtual content in images, which are imaged and extracted by the AR system. The extracted models are then used for virtual augmentation of the real image. The input to the system is authored in an offline stage and consists of a set of models and behaviors that are encoded into an image. We refer to geometry and texture information as the model, while behavior refers to additional information about it, such as animation, user interaction, modeling cues, and physical properties. This work describes examples that are a first step toward useful AR applications that retrieve content from captured images, rather than digitally from content libraries.

We name the approach In-Place Augmented Reality (IPAR), since the AR content is embedded in the same place (e.g. on printed paper) on which it is viewed. Figure 1 illustrates a simple example of our approach demonstrated on a physical map, where colors encode elevation. Using our vision-based approach, when the atlas book is imaged, the terrain elevation is inferred from the map and color scale. A virtual model of the terrain is then generated and overlaid back onto the book page. To further enrich user perception, the colors of the map are used as texture.

Our approach is aimed to provide *dual perception*. Namely, we encode models and behaviors in a way that reflects the content also to a person not using an AR system (see Fig. 3a). When viewed using an AR system, the appropriate model is augmented (see Fig. 3b).

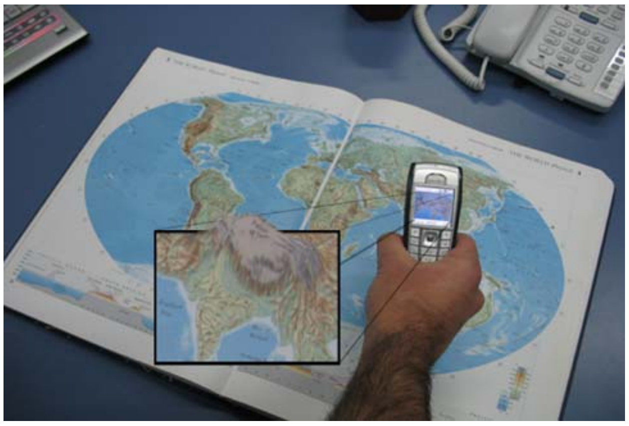

Fig. 1 In-Place Augmented Reality demonstrated using a physical map from an atlas book. Note the elevation of Tibet. The terrain model is extracted according to the *colors* of the map and augmented back onto the book with no requirement for a content library. The mobile device illustrates a possible platform of usage

One of the main advantages of our approach is that content delivery is made easier. Since the AR tracking information and the AR content are contained in a single printed image, there is no need to ask the user to send an identification code to the content provider. At the same time, content providers do not have to worry about distributing their content electronically over a communication network. Another advantage of our approach is that a content library and the infrastructure required to support it, such as a communication network, are not required. The dual perception property is another advantage. Since the printed image reflects the augmented content, interaction with the printed image becomes more natural.

In-Place Augmented Reality inherits some of the limitations common in vision-based methods. The imaging device capabilities directly affect the quality of extracted models. The dual perception property makes highly detailed models difficult to encode. It is also not clear yet how 3D models with general structure can be encoded. In addition, the basic approach is limited to applications where content lacks spatial context. Finally, only a limited amount of information can currently be encoded in a single image.

The rest of this paper is structured as follows. In the next section, we describe related work. Section 3 describes the concept and details regarding model embedding, acquisition, and extraction. Section 4 discusses layering of information and Sect. 5 provides details about our implementation. In Sect. 6, we describe several IPAR applications, and Sect. 7 concludes and describes future work.

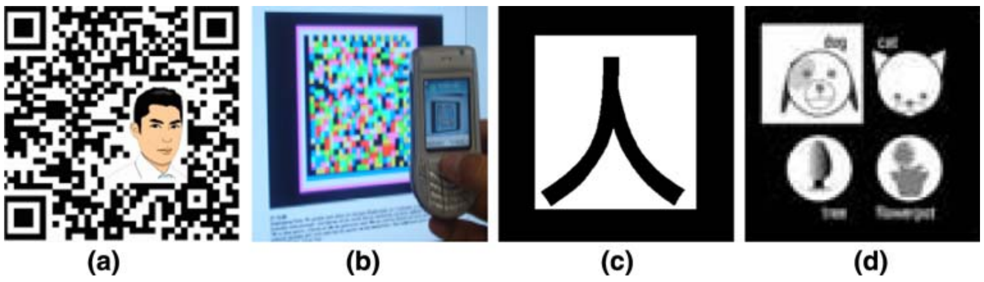

Fig. 2 Vision-based methods for content indexing: **a** Design QR-Codes (1994), **b** 4D barcodes, **c** ARToolKit and **d** AR storyboard

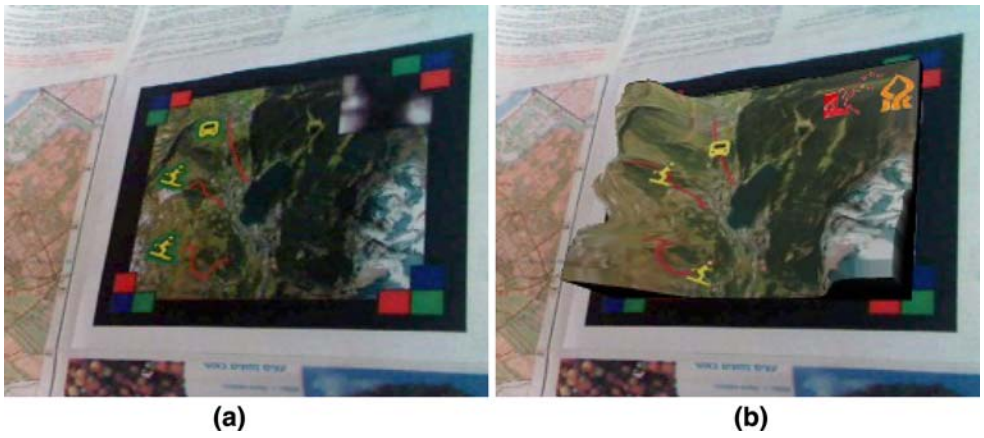

Fig. 3 Augmentation of a Saint Moritz map: **a** the embedded image and **b** the augmented image

2 Related work

This work extends our previous work on In-Place Augmented Reality (Hagbi et al. 2008). Here, we provide a user study on 12 students to test whether the suggested approach can embed new information to users that is otherwise difficult to infer from printed paper. We describe new IPAR applications and their use. We introduce the ability to overlay several layers of information and to use different page layouts for encoding content. Finally, we provide implementation details to make the approach easily reproducible.

The general concept of relying on vision for retrieving content is not new. Barcodes are widely used as a vision-based mechanism for identification (Saarelma 2005) and can contain relatively complicated information. For example, the Nintendo Gameboy e-reader (2001) uses cards with entire games encoded in the barcode pattern. When the game cards are scanned through the e-reader, the game starts. Each individual e-card can save 2,064 bytes vertically and 1,296 bytes horizontally, which is enough storage to encode a complete mini-game.

Barcodes can also be used to identify a remote source of content. In barcodes, encode webpage addresses. When scanned using a cellular phone, the webpage is accessed through the network and browsed using the phone.

Barcodes typically appear as a collection of black dots or lines. However, this does not have to be the case. Design QR-Codes (1994) incorporate images into barcodes in order to make them visually pleasant (see Fig. 2a). Barcodes can also be dynamic. Four-dimensional barcodes (Langlotz and Bimber 2007) allow transmitting of information to a handheld device as a stream of barcode images (see Fig. 2b).

In vision-based AR registration schemes, visual markers are commonly used. These typically serve two purposes: (1) identifying real objects and (2) calculating the pose of the camera relative to real objects. For example, ARToolKit (Kato and Billinghurst 1999) uses square fiducial markers, where the pattern inside the square allows identifying the marker and the square itself allows calculating the camera pose. The identity of the marker is then used to identify which model should be retrieved from the content library (see Fig. 2c), and the camera pose allows correctly aligning the retrieved virtual model with the real world. ARTag (Fiala 2004), MXRToolKit (2004), and stbTracker are other examples of vision-based libraries that use a similar two-step technique to identify markers and calculate camera pose from them.

There are also examples where marker patterns are used to convey richer content. In the AR Storyboard (Shin et al. 2005), patterns are used to compose stories, where properties are assigned to characters by combining the appropriate patterns (see Fig. 2d). In (Tsung-Yu et al. 2007), an AR system makes use of barcodes to determine the applicative context, according to which a remote database server replies with context-aware content. Other AR environments using barcodes are described in (Rekimoto and Ayatsuka 2000). In the AR PlayStation game 'The Eye of Judgment' (2007), cards are used to identify models and actions that can be performed. The cards are placed on a tracked board, and their appearance reflects the content they identify. However, in all of these cases, the amount of content encoded in the AR marker is limited, there is no visual information about the model behaviors encoded, and the marker patterns generally have a barcode-like appearance.

The main distinguishing factor of our work is that the AR content itself is retrieved from natural-looking real world images by using computer vision techniques and augmented back onto the images. In contrast, the previously mentioned methods rely on the imaging device for transmitting an identification code, which is subsequently used to retrieve the appropriate model from an external database. In contrast, in our approach, the model geometry itself is extracted from the image, and no external library has to be employed. Furthermore, unlike most barcode applications, our approach aims to provide dual perception for keeping the encoded image understandable also without the use of an AR system.

Various visual languages are understandable without using any computer, and some of these languages are very common. Pictography is the expression of words and ideas through standard languages that omit unnecessary details, e.g. icons, road signs, and Chinese characters. Visual languages are also common in cartography. Non-spatial features are commonly described by legends and spatial information, such as terrain topography and temperature, which are described using color scales. Different approaches have been described in the literature for analyzing graphical annotations. For example, mathematical and physical laws can be automatically interpreted (Davis 2007; LaViola and Zeleznik 2004; Landay and Myers 2001). Such visual languages could be visualized using the IPAR approach.

In the next section, we describe the concept of In-Place Augmented Reality in detail and provide some usage examples.

3 In-Place Augmented Reality

In augmented reality applications, virtual models are typically created by artists and stored in a content library. At runtime, these models are retrieved from the library, registered and rendered onto the scene, appearing to overlay some real world content. In our In-Place Augmented Reality approach, model details are encoded in real world images, then captured by the augmented reality system, and extracted from the scene in order to be rendered as virtual models back over the real images.

The process pipeline of an In-Place Augmented Reality system is shown in Fig. 4. The pipeline consists of three main stages: *Authoring, Content Retrieval,* and *Application Use.* In the authoring stage, representations of virtual models and their interactive behaviors are encoded in an image and printed on real paper (see Fig. 3a). This step is performed once for each printed image. Next, during runtime, content retrieval is performed, where the AR content is acquired and extracted from the printed image. This step can also be performed only once. Finally, while the application is running, for each image frame captured by the system, the extracted virtual models are registered and rendered over the real page. The final augmented result is shown in Fig. 3b. In the rest of this section, we describe each of these steps in detail.

3.1 Embedding models and behaviors

In-Place Augmented Reality deals with embedding models in images to allow their extraction when captured by the AR system. The models and their behaviors are embedded as *objects* and *annotations*, respectively. A model consists of geometry and texture, while behavior refers to the way the model acts and is displayed in real time. An object is a planar representation of a model. It consists of images representing the geometry and the texture. Annotations of objects represent model behavior. For example, an animation path for an object could be represented as a continuous line drawn on the real image. We define a rendering order for the models according to their desired appearance and arrange the corresponding objects according to this order. Objects are surrounded by a predefined color background, and annotations are marked in a visually distinct predefined color enabling them to be easily identified by our system. To associate an annotation with an object, we place the annotation next to the object.

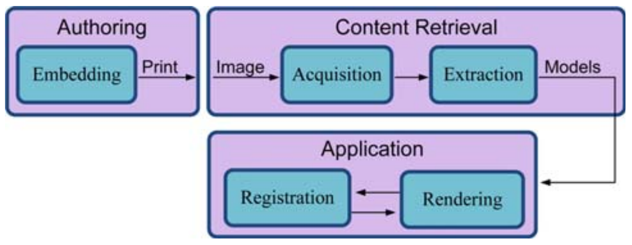

Fig. 4 The In-Place Augmented Reality pipeline

In Fig. 5, we illustrate our model representation using a map of Saint Moritz. The background defines the terrain's texture, and the geometry of the terrain is specified by an elevation map. The elevation map is scaled down and placed as a legend in the top right corner. Two skiers and a bus are embedded as icons (surrounded by a green contour), and the red paths annotate their animated routes. The result of the embedding step is depicted in Fig. 3. When captured by the AR system, the icons for the bus and skiers are animated following the red paths on the terrain model.

In addition to specifying path animations, annotations on the image can also be used to define user interaction. Figure 6 illustrates a boat object (surrounded by a green contour) annotated for user interaction (a red hand in the image). The red hand allows the user to move the boat using keyboard input. Modeling cues modifying the geometry of objects can also be assigned using annotations. For example, a property of the sea model is specified by a (red) wave modeling cue. The height and frequency of the drawn wave are used to specify the runtime properties of the virtual sea model from several predefined possibilities.

Furthermore, annotations can define physical properties for objects. For example, the skier can slide down the mountain by embedding a gravitation annotation (an arrow pointing down) next to it instead of the animation path.

In our current prototype, we support the embedding of textured 2.5D models. The types of behaviors and features that are available include model path animation, user interaction, modeling cues, and physical properties. We also support the embedding of several layers of information, as will be described in Sect. 4.

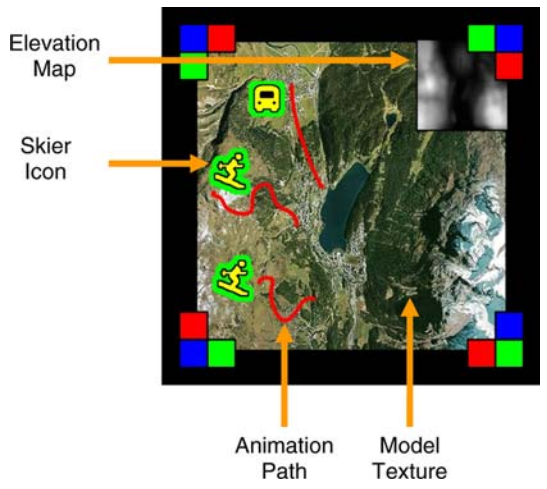

Fig. 5 The Saint Moritz map image is made of four components: the model texture, which is a satellite image of the area, the geometry of the terrain, which is represented as an elevation map, two skiers and a bus objects, which are given as icons, and the animation path, which is drawn as a *curve*

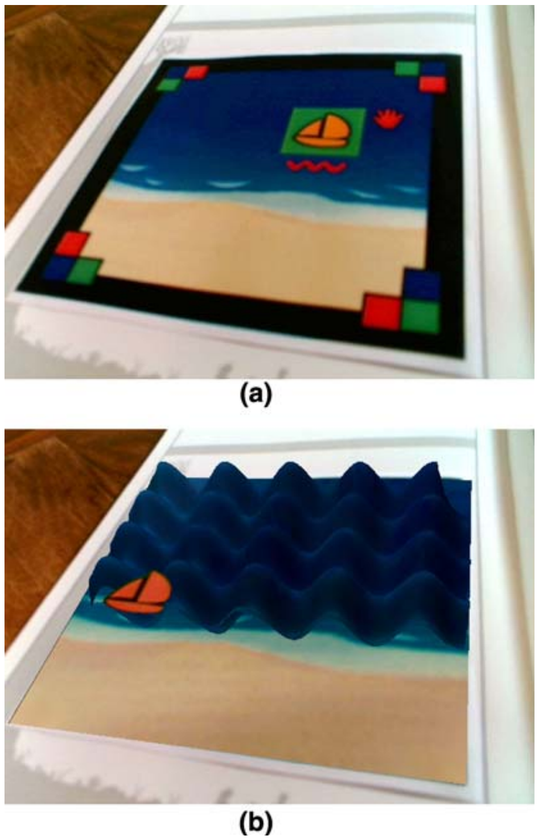

Fig. 6 Interactive ocean artwork. **a** The boat is annotated for user interaction using a *red hand symbol* and the sea is annotated by a wave modeling cue using a *red wave symbol*. **b** The augmented result. Note the wave effect and the location of the boat

3.2 Model acquisition

At runtime, the model properties and behaviors encoded in the printed image need to be extracted by the AR application. The image is captured by the camera, and if needed, we first rectify the image by applying the appropriate projective transformation. The transformation is calculated from the vertices of the surrounding black rectangular frame, using the method described in (Hartley and Zisserman 2003). Figure 7a shows the image of the real picture captured by the camera, and Fig. 7b illustrates its rectified version.

To overcome lighting differences and reduce the effect of poor camera quality, we first apply image enhancements to the captured image. Blurring caused by the low resolution of the imaging device is often present. To reduce this effect, we perform temporal averaging by keeping a short history of the rectified and registered frames. We use the RGB model to represent pixel colors. We assume the intensity of each RGB color channel varies independently and linearly with the distance to the corners of the image. To deal with color distortion, we add three reference color squares to each corner of the image.

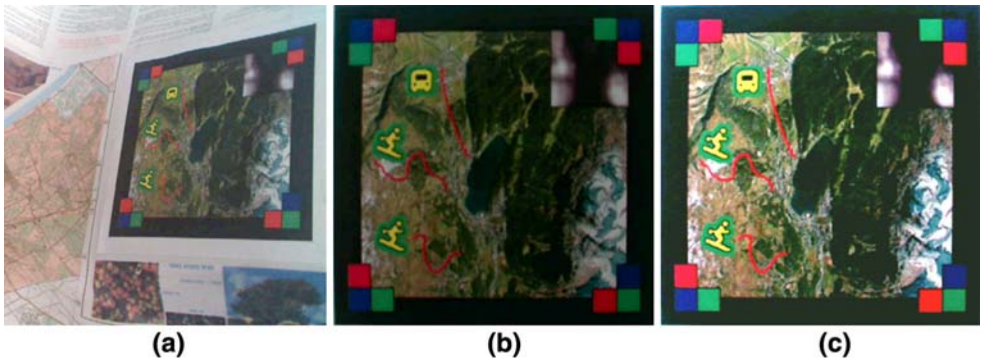

Fig. 7 Acquisition of the Saint Moritz map: **a** the embedded image captured by the camera, **b** the projectively rectified image and **c** the rectified image is averaged, and *colors* are corrected

Through the acquisition step, we interpolate each pixel's color according to these reference colors. For details on the implementation of this step, please refer to Sect. 5.2. The result of the acquisition step is illustrated in Fig. 7c. As can be seen in the final version, the image is rectified, and its colors are enhanced. The virtual model details and behaviors are then extracted from the image in a process we describe in the next section.

3.3 Extraction of models and behaviors

In the extraction step, we reverse the procedure performed in the embedding step, which is explained in Sect. 3.1. The enhanced image is analyzed, and objects with their annotations are interpreted to yield a collection of virtual models with assigned behaviors. The extraction step involves the following operations: (1) identifying objects and generating models, (2) identifying annotations that correspond to predefined behaviors, (3) assigning the behaviors to the models, and (4) generating a background model. We now describe each step, leaving implementation details to Sect. 5.

Figure 8 illustrates the first two steps. Objects are marked by a surrounding identifying contour that distinguishes them from the background (the green contour in our prototype examples). To extract the objects from the image, we perform binarization that preserves this identifying feature (see Fig. 8a) followed by a connected component analysis. Finally, we generate a mask of the object itself (see Fig. 8b). We then generate a flat model with the retrieved object image as texture.

Annotations also have an identifying feature that makes them easy to extract from the background image (red color in the examples described here). Identifying the annotations also starts with binarization that preserves this feature (see Fig. 8c, e). The annotation templates are known to the system in advance (see Fig. 8d). We perform template matching to find their location. Once annotations are

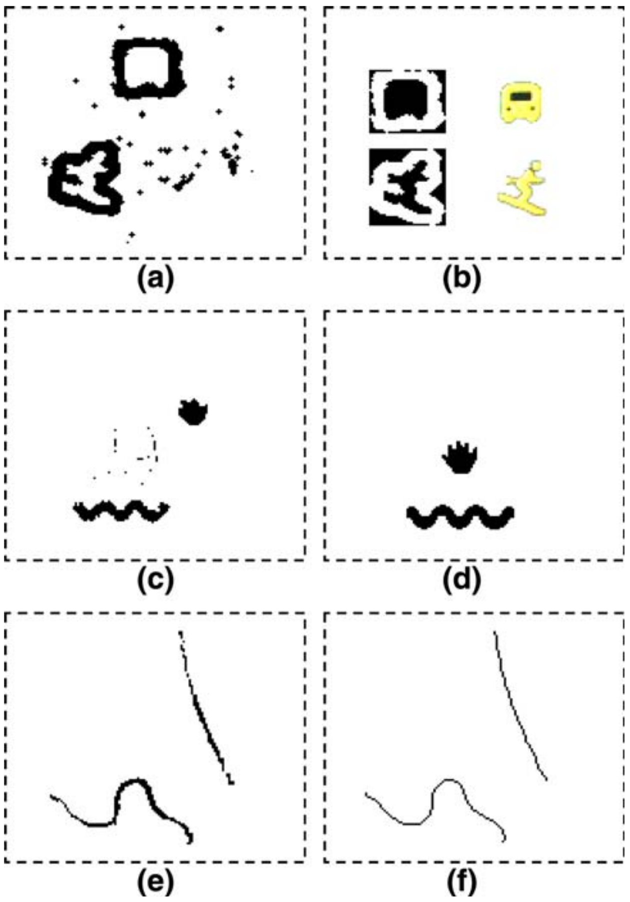

Fig. 8 Extraction of objects and their annotations for the examples of Figs. 5 and 6. **a** Extraction of object background, **b** extracted object masks and textures, **c** extracted wave motion and user control annotations, **d** wave and user control templates, **e** extracted path annotations and **f** Linked and thinned paths

extracted, the predefined behaviors they represent are assigned to their nearest models in the image. For example, when a hand annotation is identified, the system wires the keyboard arrows to the model so that it can be moved by the user. At runtime, pressing the keyboard arrows will cause the object to move.

Animation paths are marked as continuous red curves. Hence, path candidates are pixels that remain unidentified after the previous step (see Fig. 8e). To extract the paths, we perform edge linking followed by thinning, as described in (Lam et al. 1992), and scan the connected components of the resulting image. This yields thin and continuous curves even where curves are slightly broken (see the result in Fig. 8f).

Finally, we create the virtual terrain model from the elevation map and the base image. First, the elevation map is stretched to fit the size of the base image, assigning height to each pixel. The base image is then corrected and used as a texture. Parts of the image that were occluded by annotations, objects, and color squares are filled-in using the in-painting method described in (Telea 2004). We in-paint the original image, rather than the color balanced one. This gives a rough approximation of the lighting conditions in the real scene, which allows the rendered content to visually fit well into the scene. Additional texture layers can be defined as well. These additional layers are also stretched to the size of the base image and superimposed as semitransparent images. See Sect. 4 for more details about layering.

3.4 Registration

Once models have been successfully extracted, live augmentation and AR viewing begins. For each captured video frame, the models are registered onto the real image and rendered accordingly. We implemented our system using both the BazAR model–based tracking package (Lepetit and Fua 2006) and the ARToolKit marker–based registration package (Kato and Billinghurst 1999). Only one tracking solution is used at a time. ARToolKit traditionally uses a pattern surrounded by a black frame (Fig. 2c). In our implementation, however, we do not make use of ARToolKit patterns. However, we still have to consistently determine the orientation of the frame, originally inferred in ARToolKit from the orientation of the pattern. To recover the correct orientation, we use a distinct arrangement of colored squares in the corners of the black frame to distinguish each of the corners. The corner positions in the 2D image can then be passed through ARToolKit to calculate the 3D pose.

When BazAR is used, registration is performed based on the features of the embedded image, and the black frame is not required. In this case, the user must first take a frontal image of the scene in order to initialize BazAR. We use a high resolution image to perform the extraction step and pass a reduced resolution image to BazAR to increase the tracking efficiency.

3.5 Rendering

In the rendering stage, the extracted models are rendered onto the scene and modified according to the behaviors assigned to them. Modeling cues are first applied to the geometry at each frame. The effect of user actions is then applied to the appropriate models. Next, animations are calculated and updated. For example, the new position of an object with a path annotation is calculated at each frame, and the object is transformed accordingly.

4 Layering

In-Place Augmented Reality is an effective approach when there is more than one layer of information to be presented on the same model. For example, a world map may consist of different layers, providing different types of information about the world. IPAR can be used to visualize several layers by embedding them in the scene itself. Figure 9 depicts an example of embedding four different layers of the Far East area. The large image shows a view from outside the atmosphere (blue marble layer), and the three other layers are terrain elevation, population density, and average temperature layers.

The blue marble image is used as the base layer, while the others are rescaled and placed in the right side of the printed image. Although we use a black frame for tracking, as explained in Sect. 3.4, the additional layers do not have to be located inside the frame. In the acquisition step, each of the layers is first extracted. We then create a flat model with the base image as texture. We extract the other layers as well and wire keyboard keys to each of them. Pressing one of the keys toggles the visibility of the corresponding layer. When a geometry layer is toggled on, the flat model is replaced with an elevated terrain. When a texture layer is toggled on, it is added to the other visible texture layers. The result allows the user to see in runtime a transparent overlay of any of the possible combinations of the layers.

5 Implementation and performance

We implemented an In-Place Augmented Reality platform and several prototype application based upon it to demonstrate the feasibly of the concept, test performance, and perform an initial user evaluation. In this section, we first describe the system configuration and then turn to describe the implementation details of the image enhancement and extraction steps. Finally, we discuss system performance.

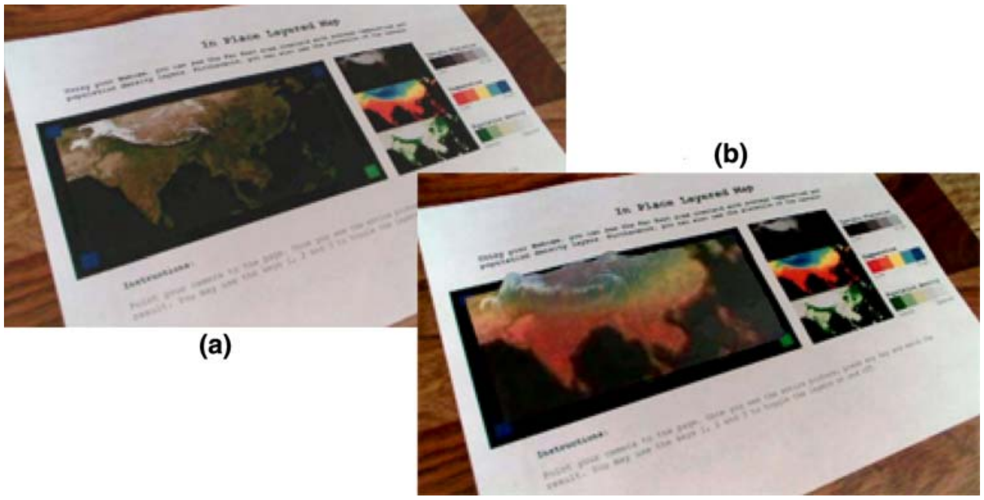

Fig. 9 Far East area **a** as seen from outside the atmosphere (blue marble view) with additional three layers: Terrain elevation where the intensity translates to height, surface temperature where *red* means hot and *blue* means cold, and population where density is marked in *green*, and **b** as seen from the AR view of the geographic information viewer application with the terrain elevation as 2.5 dimensional geometry and with the temperature overlaid semi-transparently

5.1 System configuration

We implemented the system in C++, using the Intel OpenCV (1999) image processing library, OpenGL for rendering, ODE for simulating physical properties, and the ARToolKit and BazAR libraries for registration and tracking. We tested the system using several datasets and received satisfying results. The hardware consisted of a laptop with a 1.83 GHz Intel dual core processor, 1 GB of RAM, and a 128 MB ATI graphics card running the Windows XP operating system. We tested the prototypes with two cameras: a CMOS 960 × 720 pixel, 30 fps webcam, and a CCD 640 × 480 pixel, 15 fps webcam. The embedded representation was printed on regular A4 paper, using a 600dpi laser color printer.

5.2 Image enhancement

As explained earlier, we perform color correction on the acquired image. Small colored squares used for reference are added to the four corners of the black boundary (see Fig. 7). We chose green, red, and blue colors as reference colors and used those colors in our examples to identify features. Different color models can be used as well. The RGB model allowed us to encode only three color references in a relatively small area around each corner. For efficiency reasons, we precalculate for each pixel p_i in the image its Euclidean distance from the four corners and store these distances in a table. In effect, since the table is symmetric along its two diagonals, we only have to store a fourth of the table. We measure channel intensity reduction of each color square and the intensity increase of the black border with respect to their original known colors. Equation 1 is used to calculate the corrected value of the c channel in the pixel p_i, according to the r-th corner.

$$\delta_i^{r,c} = (p_i^c - b_r^c)\frac{1}{v_r^c}, \qquad (1)$$

where p_i^c is the value of the c channel in the pixel p_i, b_r^c is the c channel value measured on the black frame in the r-th corner, and v_r^c is the value of the c channel measured in the corresponding color reference square in the r-th corner. The color value range of each channel is [0, 1].

To propagate this information to each pixel in the image, we calculate a weighted average according to the distances of a pixel to the corner color squares and correct the intensity accordingly. Equation 2 is used to update the value of each pixel.

$$P_i^c = \frac{\sum_r d_i^r \delta_i^{r,c}}{\sum_r d_i^r}, \qquad (2)$$

where the range of d_i^r is [0, 1]. It reflects the inverse ratio of the distance from p_i to the r-th corner. The nearer a pixel is to the corner, the greater its distance value.

5.3 Object extraction

Objects are extracted based on their identifying green background. Binarization is performed to keep the green background by employing three different thresholds T_r, T_g, and T_b for the red, green, and blue channels, respectively. A pixel is included in the binary image if its color values satisfy $P_r < T_r$, $P_b < T_b$, and $P_g > T_g$, for predefined thresholds P_r, P_g, and P_b. Connected component analysis is then performed by first dilating the binary image with a small round structuring element to recover lost pixels, and then flooding and labeling. We ignore components with area smaller than a predefined threshold.

5.4 Identifying annotations

To identify annotations, we use a method similar to that used for object extraction, this time searching for red pixels. As with model extraction, we first perform binarization to acquire the red pixels, which identify annotations. We then apply normalized cross-correlation template matching between the set of supported annotations and the binary image. Since the image is large, we perform the cross-correlation on the discrete Fourier transform of the image, which gives fast and robust identification. We assign annotations to their nearest objects using the Euclidean distance transform of the objects and the annotations.

5.5 Performance

In this section, we describe the performance of our system. In particular, we describe the content retrieval process, including image acquisition and model extraction. The performance of the AR application, namely registration and rendering at each frame, depends on the tracking module and is thus not included here. Using the ARToolKit to track frames of 960 × 720 pixels, we were able to achieve interactive frame rates of 15 fps.

Content retrieval is performed once and introduces a one-time delay. Image acquisition is performed first, and we were interested in measuring the delay introduced by each of its sub-steps, as shown in Table 1. Image acquisition starts with warping the 960 × 720 pixel or 640 × 480 pixel source image to a predefined 300 × 300 rectified image using the four outer corners of the black frame. We then accumulate a number of frames to perform temporal averaging and finally color interpolation.

We found that the rectified image size is the dominant factor in the performance of these steps. In our prototype system, a rectified image of 300 × 300 pixels or less was sufficient to robustly perform extraction. In Table 1, we describe results with other sizes as well. The acquisition step took only 537 ms for a 300 × 300 pixel image. Since this step is performed only once before starting to view the content, the delay is acceptable.

Next, we measured the time it takes to perform the model extraction step and its relation to the number of different annotations and objects in the image. Table 2 shows the results for three sets of experiments we performed. In Mode A, the scene consists of a single object, a single annotation, and a single geometry. In Mode B, the scene consists of 10 objects, 10 annotations, and a single geometry. In Mode C, the scene consists of 20 objects, 20 annotations, and a single geometry. We also studied the effect of the size of the rectified image on the performance, and we include results with different sizes in the table. As expected, the performance drops as the amount of content and the size of the rectified image increase. However, since 300 × 300 pixel images are sufficient for the extraction, and since this step is performed only once during the initialization of the AR application, the delay is acceptable.

Table 1 The runtime of the image acquisition sub-steps on rectified images of three sizes (in pixels)

	200 × 200	300 × 300	400 × 400
Rectification	15 ms	31 ms	39 ms
Averaging	16 ms	38 ms	78 ms
Color Interpolation	203 ms	468 ms	828 ms

Table 2 Runtime for extracting models from different images

	200 × 200	300 × 300	400 × 400
Mode A	62 ms	125 ms	234 ms
Mode B	218 ms	453 ms	922 ms
Mode C	391 ms	813 ms	1,688 ms

6 Applications and user evaluation

We have demonstrated the concept of In-Place Augmented Reality with four sample applications: (1) Saint Moritz ski site guide, (2) Geographic Information Viewer, (3) Surface Texturizer, and (4) Interactive Ocean Artwork. The applications were used to perform an initial user evaluation as described next.

The Saint Moritz ski site guide application was used to perform an objective test to answer the following question: Can augmentation of printed paper with embedded information help users assess the information correctly?

We picked 12 students at random from our university campus. The students were asked to examine the Saint Moritz elevation map and the satellite image with ski routes depicted in Fig. 10. Their tasks were to asses which skiing route is the longest, the shortest, the steepest, and the most moderate. Our hypothesis was that most students will only answer these questions correctly using the IPAR application. If so, it indicates that augmenting understandable information on the paper provides additional information to the user that is hard to asses otherwise.

The subjects were first asked to answer all the questions without using the IPAR application and then later using the application. We first explained the users what is a satellite image and that in our case it contained a skiing site. We explained the red curves represented ski routes and how to

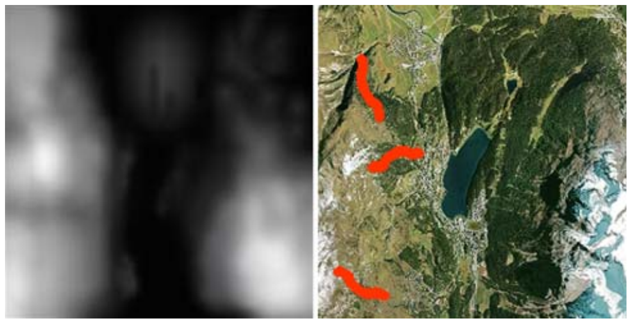

Fig. 10 Elevation and texture layers of the Saint Moritz ski site used for user evaluation. Users were asked to order the ski routes by their length and steepness using the printed paper and then using the IPAR application and the paper

read the elevation layer. We verified the subjects understood the correlation between the elevation layer and the satellite image. We let the students play with the application on another elevation layer and another texture layer (without skiing route information) before performing this task so they can experience the effect of height information on the terrain texture. Below are the questions we asked:

Q1 Do you understand how the elevation layer is correlated to the satellite image, for example that the area on the left is much higher that the area in the center?

Q2 From the ski routes in red, which one is the longest and which one is the shortest?

Q3 From the presented ski routes in red, which one is the steepest and which one is the most moderate?

Q4 Did you use the elevation layer to answer Q2 and Q3?

All the students answered "yes" to Q1. Table 3 summarizes the answers to Q2, Q3, and Q4. The answers indicate that users achieved better results in Q2 and Q3 using the IPAR application. We also checked the time it took users to complete the tasks and found no significant difference between the cases. It is interesting to note that only four out of twelve students used the elevation map information to answer Q2. The other eight students said they had not felt the need to use the elevation information to assess the route length. Six of the students understood the correlation between the terrain height and the length of the skiing routes only after using the application. Nine students used the elevation information for Q3, yet only three found the correct answer without the IPAR application.

The Geographic Information Viewer application was used for conducting a subjective user study based on the map depicted in Fig. 9. Our main interest was to compare how easy it is for users to asses a known correlation. Places of higher altitude are usually colder and their population is sparser. We assumed it would be easier for users to asses this correlation using the IPAR application rather than without.

We performed this test with the 12 students from before. All of the students were able to notice the correlation in our data with and without the IPAR application after examining the different layers for a few seconds. Most of the users had known the correct correlation before starting the experiment and only needed to have a quick look to verify their prior knowledge holds in the example. We then asked them:

Q5 Do you find it easier to verify that places of higher altitude are colder and their population is sparser with or without the application?

All 12 users found it easier to assess the correlation between the three presented layers using the Geographic Information Viewer application than without using it.

We implemented a variant of the Geographic Information Viewer application, named Surface Texturizer. Using the Surface Texturizer, users can capture textures in a freestyle manner from the real world and overlay them by stretching terrains extracted from printed elevation maps. The most appealing elevation map for users was an elevation map we created for the face of Jean-Luc Picard, a Star Trek character. Users enjoyed using different textures from the real world on his face model extracted from the elevation map, as well as mixing different textures with semi-transparency between texture layers. We implemented this application to check the fun factor of a possible mini-game built on top of our framework.

We asked all the subjects to rate the different applications they had tried from 1 to 10. All the applications received scores between 7 and 10 with neither meaningfully outstanding the others. The average fun factor was 8.3.

Table 3 Summary of the user evaluation performed with the Saint Moritz IPAR application on 12 students

	Correct ordering without application	Correct ordering with application	Used elevation information
Ordering routes by length	5	10	4
Ordering routes by steepness	3	11	9

We were also interested in checking how users unfamiliar with AR relate to the IPAR experience in comparison to traditional marker-based AR. We presented the students the ski site map depicted in Fig. 3. We placed an ARToolKit marker next to it, for which a digital copy of the same map was retrieved from a file and augmented in the exact same way. All of the users preferred the In-Place version, since they felt their interaction with it was more natural, especially because the printed map reflected what they could see in the AR view. In addition, they liked to be able to understand what the image without using any device.

The interactive ocean artwork application was used with a set of different annotations during the development process of the IPAR framework. Its main purpose was to evaluate different annotation schemes.

From the applications we implemented and the user study conducted, we conclude that: (a) It is possible to extract several layers and register them reliably for users to analyze the augmented scene. (b) The augmented combinations of layers of information make it easier for users to assess the relations between them. (c) Combination of elevation information with texture layers can give rise to conclusions about the terrain, which would be very difficult to reach otherwise. (d) It was more fun for users to assess geographic relations on printed maps using an IPAR application than without it. (e) Use cases where the camera is hand-held are complicated for users and may result in low quality of the extraction. A stationary camera, on the other hand, is easier, but users are required to rotate and tilt the printed page to experience the 3D effect of the augmentation.

In-Place Augmented Reality applications can be useful in a wide range of different domains. In education, for example, physics books can include figures of physical scenarios demonstrating the discussed material. When viewed by an IPAR application, the figures can be animated according to the physical laws demonstrated. Furthermore, the student could interact with the system, allowing him or her to modify different physical properties and immediately see the effect. Another domain that can benefit from the advantages of the proposed approach is advertising. Advertisers creating content for IPAR applications can enjoy a high level of flexibility and simplicity in distribution and maintenance of augmented advertisements. Gaming is another domain that can be influenced by the introduction of the In-Place Augmented Reality idea. In addition to all the above advantages, users should be able to author their own versions of games. For example, one can modify the board of a given game and enjoy interactivity in a new dimension.

7 Conclusion and future work

In this paper, we have introduced the concept of In-Place Augmented Reality, which is vision-based transmission of AR content independent of a content library. The idea is based on embedding models into images and extracting them in runtime, where the embedded image is intuitive and understandable to a human also without using an AR system.

In our vision-based approach, models are embedded in the scene itself, so no database is required to store the models. Thus, the amount of models that can be augmented does not have direct implications on the design and maintenance of the system. Furthermore, the number of concurrent users does not affect the AR system and is hence unlimited. The approach is also useful when no network communication can be established. In addition, since the AR virtual content is encoded in the real image that the content is supposed to augment, separate distribution of content is not required.

In this work, we focused on 2.5D models. We are investigating methods for representing 3D models in printed images that are understandable also without the use of an AR system.

A natural platform for In-Place Augmented Reality is mobile phones and mobile devices in general. These can be easily pointed to information encoded in the real world and enhance it. We intend to port the implementation to such devices and create a flexible and usable framework.

Finally, although we have conducted a user study on 12 students with our prototype applications, we would like to conduct experiments on larger groups of people. We would like to learn how easy it is for users to understand the functionality of the AR scene from the embedded objects and annotations and how natural it is for people to interact with AR content in this way.

A uniform In-Place Augmented Reality framework with a rich visual language running on different platforms can make AR available to many users and content creators. It also allows us to explore different application areas for the potential of AR. Our work is a first step in what promises to be a fresh and exciting new research direction.

Acknowledgments We would like to thank the reviewers for their constructive comments. This work was supported by the Lynn and William Frankel Center for Computer Sciences.

References

Davis R (2007) Magic paper: sketch-understanding research. Computer 40:34–41

Design QR-Code (1994). http://www.denso-wave.com/qrcode/

Fiala M (2004) ARTag, an improved marker system based on ARToolkit, NRC institute for information technology

Hagbi N, Bergig O, El-Sana J, Kedem K, Billinghurst M (2008) In-Place Augmented Reality. In: 7th IEEE and ACM international symposium on mixed and augmented reality (ISMAR'08), pp 135–138

Hartley R, Zisserman A (2003) Multiple view geometry in computer vision, 2nd edn. Cambridge University Press, Cambridge

Intel OpenCV (1999). http://opencvlibrary.sourceforge.net/

Kato H and Billinghurst M (1999) Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In: 2nd international workshop on augmented reality, IWAR99, San Francisco, USA

Lam L, Lee SW, Suen CY (1992) Thinning methodologies-A comprehensive survey. IEEE Trans Pattern Anal Mach Intell 14:879

Landay JA, Myers BA (2001) Sketching interfaces: toward more human interface design. Computer 34:56–64

Langlotz T and Bimber O (2007) Unsynchronized 4D barcodes. International Symposium on Visual Computing, pp. 363–374

LaViola J and Zeleznik R (2004) MathPad2: a system for the creation and exploration of mathematical sketches. ACM Trans Graph (Proceedings of SIGGRAPH04) 23: 432–440

Lepetit V and Fua P (2006) Keypoint recognition using randomized tree, Trans Pattern Anal Mach Intell 28, Nr. 9, pp. 1465–1479

MXRToolKit (2004). http://mxrtoolkit.sourceforge.net/

Nintendo e-Reader (2001). http://en.wikipedia.org/wiki/Nintendo_e-Reader

Open Dynamics Engine (2001). http://www.ode.org/

Rekimoto J and Ayatsuka Y (2000) Cybercode: designing augmented reality environments with visual tags. In: Proceedings of DARE 2000 on designing augmented reality environments

Saarelma H (2005) Printed codes—patent survey. Graphic Arts in Finland 34:1–11

Semacode (2004). http://semacode.com/

Semapedia (2005). http://www.semapedia.org/

Shin M, Kim BS, and Park J (2005) AR storyboard: an augmented reality based interactive storyboard authoring tool. In: Fourth IEEE and ACM international symposium on mixed and augmented reality (ISMAR'05), pp 198–199

Studierstube Tracker (2008). http://handheldar.net/stbtracker.php

Telea A (2004) An image inpainting technique based on the fast marching method. J Graph Tools 9:25–36

The Eye of Judgment (2007). http://www.eyeofjudgment.com/

Tsung-Yu L, Tan-Hsu T and Yu-Ling C (2007) 2D barcode and augmented reality supported english learning system. In: Computer and information science Melbourne, Australia, pp 5–10