Expert Systems with Applications 41 (2014) 2174–2185

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Mobile augmented reality based context-aware library management system

Adrian Shatte, Jason Holdsworth, Ickjai Lee *

School of Business (IT), James Cook University, Cairns, QLD 4870, Australia

ARTICLE INFO

Keywords:
Augmented reality
Library management
Mobile computing
Agent programming
Context-aware

ABSTRACT

Mobile augmented reality has gained popularity in recent years due to the technological advances of smartphones and other mobile devices. One particular area in which mobile augmented reality is being used is library management. However, current mobile augmented reality solutions in this domain are lacking in context-awareness. It has been suggested in the literature that agent programming may be suitable at overcoming this problem, but little research has been conducted using modern mobile augmented reality applications with agents. This paper aims to bridge this gap through the development of an agent-based, mobile augmented reality prototype, titled Libagent. Libagent was subjected to five experiments to determine its suitability, efficiency, and accuracy for library management. The results of these experiments indicate that agent-based mobile augmented reality is a promising tool for context-aware library management.

1. Introduction

With personalized smartphone applications rapidly gaining popularity in recent years, there has never been a greater need for the delivery of context-sensitive information. The portability of smartphone devices means that more often these devices are being used in the field to support a variety of occupations and scenarios. In many jobs, people are tasked with the management of sizeable inventories of items. However, due to the limited capacity of the human short term memory, we can only manipulate six or seven items concurrently (Keppel and Underwood, 1962; Miller, 1956). This limitation of our cognition can result in errors that amount to significant losses of time and money.

One particular domain in which this problem is evident is the library. In a library, contextual information is very important as books are organized geographically based on subject and catalogue number. Further, books are constantly being moved, loaned, or misplaced. Without additional support, a user may find the task of searching for books difficult, especially in larger libraries. On the other hand, librarians are tasked with the job of stack maintenance. Stack maintenance refers to the re-shelving of material that has been removed from a shelf, or shifting existing material within a stack to make room for new materials.

Current library systems rely on a searchable database, where users can enter keywords to locate catalogue numbers of desired books. Such systems require the user to first locate the correct shelf, and then visually scan through the catalogue numbers to find the appropriate book. Depending on the number of books in a particular category, the degree of similarity between each catalogue number will decrease, thus increasing the search space for the user and the difficulty of finding the book. Additional problems such as incorrect sorting of the books or books being misplaced within the library can also prove challenging. Research in the field of mobile Augmented Reality (AR) and agent programming may provide a solution for context-aware library management.

AR is a novel technology that can be used to enhance a physical environment by overlaying virtual content through a visual interface (Zhou et al., 2012). While the technology has been available since the 1990s, the rapid advances in mobile technology over the past decade have provided powerful and convenient platforms for AR applications (Van Krevelen and Poelman, 2010). The major benefit of mobile devices for AR is that the technology is ubiquitous and easily accessible to consumers. Additionally, the computing power of a mobile device can provide users with the tools to deal with large inventories of physical items, helping to reduce errors and increase performance.

An immediate benefit of AR technology for library management is that it can be used to replace the catalogue number with a simpler visual cue, for example a shape or a colour. Current catalogue numbers require users to scan a shelf and read several book spines to determine their proximity to the desired book. In contrast, an AR application could display a prominent visual cue (e.g. a yellow exclamation mark) over the spine of the desired book, thus significantly simplifying the task of searching for a book. Additionally,

* Corresponding author. Tel.: +61 740421083.
E-mail addresses: Adrian.Shatte@my.jcu.edu.au (A. Shatte), Jason.Holdsworth@jcu.edu.au (J. Holdsworth), Ickjai.Lee@jcu.edu.au (I. Lee).

0957-4174/$ - see front matter Crown Copyright © 2013 Published by Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.eswa.2013.09.016

AR could be used to aid with sorting, for example visual cues such as arrows, ticks, and crosses could indicate the steps necessary to bring a shelf back to the correct order.

Agent programming could be used alongside AR technology to provide context-awareness to a mobile library management application. By definition, agents are hardware- or software-based entities that exhibit the characteristics of autonomy, social ability, reactivity, and proactivity (Wooldridge et al., 1995). These characteristics make agents ideal for providing context-sensitive information efficiently in dynamic environments. The fact that a library shelf can change its state regularly (e.g. books being moved, added, or erroneously sorted) means that up-to-date information is necessary to reduce frustration and improve efficiency of library staff and users. Researchers have claimed that agents have the potential to improve the delivery of personalized content, but little research has been conducted using agents with modern mobile applications (Maes et al., 1994).

The aim of the research presented in this paper is to determine the benefits of a mobile library management system based on agent programming and AR technologies. It is clear that there is a need for context-awareness in mobile AR applications. While studies in the 1990s (Maes et al., 1994; Nagao, 1998) and early 2000s (Zaslavsky, 2004) suggested the use of agents to approach this challenge, there are few examples of agent-based context-awareness in modern mobile AR literature. However, based on these early studies it is evident that agent programming concepts may vastly improve the user experience for mobile AR applications, in terms of spatial localization and user-centric information. To determine these benefits, we propose a prototype, titled Libagent, and subject it to several experiments. These experiments are designed to determine the correctness, robustness, usability and ease of use in both individual and collaborative environments. The results of these experiments indicate that our system utilizes these technologies effectively to improve on current mobile AR solutions for library management.

The remainder of this paper is structured as follows. Section 2 provides the definitions and context of the major technologies and paradigms used in our research, including mobile AR, agent-based AR, and library management systems. Section 3 explains the framework of our proposed library management system (Libagent), including tools used and main algorithms for library-based tasks. Section 4 describes the research methodologies and results of our experiments, including a discussion of these results in terms of limitations and improvements. Finally, Section 5 concludes the paper with final remarks and recommendations for future research in the field of agent-based mobile AR.

2. Preliminaries

2.1. Mobile augmented reality

AR refers to a live and real-world image that has been enhanced or diminished by virtual content through a camera interface (Zhou et al., 2012). AR technology aims to simplify everyday tasks by complementing the user's perception of and interaction with the real world. There are many possible applications for AR technology. Carmigniani et al. (2011) categorise four main types of applications that have been cited in AR research: advertising and commercial, entertainment, education, and medical (Carmigniani et al., 2011).

AR sits within the broader spectrum of mixed reality proposed by Milgram and Kishino (Van Krevelen and Poelman, 2010). The Reality-Virtuality continuum defines a spectrum of mixed reality applications ranging from no augmentation (real environment) to complete virtualization (virtual environment). In virtual environments (sometimes called virtual reality) and augmented virtuality, real objects are added to virtual environments. AR differs in that it provides local virtuality and consists of augmenting virtual objects into a real world environment.

In order to provide a dynamic experience, AR needs to register the virtual content with the real world. This process is known as tracking. There have been several different methods for tracking and orientation proposed in the AR literature. In general, these approaches can be divided into three classes: visual, non-visual, and hybrid (Zhou et al., 2012). Visual methods include fiducial markers, feature detection, and edge detection. Non-visual approaches rely on additional inertial sensors (e.g. compass and accelerometer) to provide location and orientation information. Hybrid approaches utilize a combination of both methods by complementing the visual system (a camera) with sensor-based data.

While the technology for AR has been available since the 1960s (Van Krevelen and Poelman, 2010), only recently has it experienced an explosion in popularity. This is partially attributed to the rapid advances in mobile technology. Due to the recent influx of smartphones, the challenge of providing context-sensitive information is at the forefront of the field. The fact that mobile devices have become portable and ubiquitous means that location-sensitive information is more important than ever. A report published by the Pew Internet and American Life Project found that 74% of smartphone owners use their phone to get real-time, location-based information (Zickuhr, 2012).

A study conducted by Olsson et al. (2013) gained information about the expectations of mobile AR users in the context of a shopping environment. The case studies involved in this research were conducted in 2009, when mobile AR was experiencing an upheaval due to the release of powerful smartphone devices. This research found that users expect multifaceted services and information from AR applications, including proactivity, relevance, and context-sensitivity. The research concludes that mobile AR has great potential to offer rich and contextual information that is very specific to a time and a place. There are many areas of mobile AR that can benefit from accurate context-awareness. One example use case for context-aware mobile AR is wayfinding. Wayfinding is the process of orientation used by a person to navigate through a physical environment. AR navigation requires a high degree of positional accuracy to succeed (Smit and Barnett, 2010). Recent solutions have attempted sparse localization by positioning fiducial markers throughout a building and offloading most navigational tasks to the user through activity-based instructions (Mulloni et al., 2011; Mulloni et al., 2012). While such a system is highly effective in a static environment, it lacks the ability to respond to dynamic changes in the environment that might affect navigation tasks.

GAT (Rodriguez-Sanchez et al., 2013) is a system that attempts to overcome the static nature of wayfinding applications by allowing users the ability to generate their own wayfinding applications. The system provides a multiplatform app generator so that users can deploy customized wayfinding applications to tourists on all of the popular platforms (e.g. Android, iOS, and Windows phone). GAT uses sensors (e.g. BlueTooth and GPS) alongside visual recognition of QR codes to gain a user's context, and crawls the internet to determine points of interest in the surrounding area. Benefits of GAT include support for both indoor and outdoor navigation and context-specific points of interest. Similarly, the MobileAR browser developed by Engelke et al. (2012) allows developers to create and deploy rich AR applications using web technologies, including HTML5 and JavaScript.

Another use case is to provide location-based information to tourists. MobiAR is an application developed for the Android platform that provides tourists with AR-based information about accommodation, restaurants, and other points of interest (Marimon et al., 2010). MobiAR utilizes both Global Positioning System

(GPS) data obtained from the mobile device as well as recognizing real-world locations using predefined reference images. By incorporating both methods to detect location, the system can provide a more accurate location and deliver better content to the user. Other examples of tourism-based AR are prevalent in the literature (Lee et al., 2012; Kounavis et al., 2012; Furata et al., 2012).

While tourism systems such as MobiAR can be effective in outdoor locations, GPS satellites fail to provide positional information when indoors. On the other hand, sparse localization can provide users with rough context-awareness, but it may not be robust to dynamic environments. Thus, there is room for additional methods for context-awareness in mobile AR systems. It has been suggested in the literature that software agents are an ideal candidate to provide personalized content (Maes et al., 1994).

2.2. Agent augmented reality

Agent programming is a software paradigm first proposed in the 1990s that takes concepts from theories in the field of artificial intelligence and applies them to the field of distributed systems (Fabio et al., 2007). The term'agent' is ambiguous and has been used to describe software entities in different domains, for example in artificial intelligence, database environments, and computer networks (Baldauf et al., 2007). Most definitions agree that an agent is a software component that exhibits autonomy and acts on behalf of another entity in pursuit of its own agents (Fabio et al., 2007). While it is possible to have a single-agent system, it is more common for a system to consist of multiple agents (known as a multi-agent system).

Further to this definition, agents are characterized by the attributes of *autonomy*, *social ability*, *reactivity*, and *proactivity* (Wooldridge et al., 1995). The characteristic of *autonomy* means that agents can operate without the direct intervention of humans or other entities, and have control over their actions and internal state. *Social ability* refers to the capacity for agents to interact with other agents as well as with their users. The *reactivity* characteristic allows agents to perceive their own environment and respond efficiently to any changes within that environment. *Proactivity* means that in addition to responding to their environment, agents can take initiative and exhibit goal-directed behaviour. It is also implied that agents characterize *situatedness*; agents operate within an environment, sense events that occur in that environment, and act upon those events (Holz et al., 2011).

Agent-based programming is similar to object-oriented programming, and agents can be compared to"objects". That is, an agent has an interface for message sending and message retrieval that is independent of its internal data structures and algorithms. Agents communicate with other entities (including other agents) in an expressive agent communication language. This language allows agents to exchange logical information and data, including individual commands and scripts or programs. In this sense, agents can communicate goals and intentions, allowing other agents to adapt or perform tasks to assist those goals (Genesereth and Ketchpel, 1994).

According to Maes et al. (1994), agent technology has the potential to deliver personalized content to applications. One of the limitations of the current research in mobile AR is a lack of robust context-awareness for location-based applications. In the past, agents have been considered as a means for providing contextual information to AR applications. However, much of the research on agent-based AR is decades old (conducted in the 1990s), so it has limited relevance to the powerful mobile devices of today. Thus, it is important to consider the use of software agents on modern mobile AR systems to determine effectiveness and usefulness in terms of context-awareness.

2.3. AR-based library management system

Libraries are another area in which the benefits of context-aware mobile AR have been considered. In a library, contextual information is very important as books are organized geographically based on subject and catalogue number. Further, books are constantly being moved, loaned, moved, or misplaced. Without additional support, a user may find the task of searching for books difficult, especially in larger libraries. Before recommendations can be made to improve mobile AR for libraries using agents, it is important to consider the contributions and limitations of current research in the literature. It should be noted that none of the current solutions discussed in this section utilize agent programming.

One of the important considerations for developing a mobile AR library management system is the tracking technique used to recognize books. Commonly, books are organized in a vertical orientation with only the book spines visible to the library user. This provides a challenge to AR tracking, as book spines generally have a small surface area and limited distinguishing visual features. There are two main tracking techniques discussed in the literature: feature-based tracking and fiducial marker tracking. The benefits and disadvantages of each method are explored below.

The research presented by Chen et al. (2010) demonstrates an example of a hybrid system for compiling an inventory of books in a library using smartphones. The proposed system uses edge detection algorithms running on a server to separate the book spines in a captured image. Each individual spine is then compared to a database of predetermined visual features to determine a match. Additionally, the system also offloads sensor data (compass, accelerometer tracing, WiFi signatures) to the server which calculates a rough location inside the building where the image was captured. These algorithms allow a user to generate a location-based inventory of books in a building. However, the proposed system achieved only a 74.5% recognition rate. Further, the researchers fail to mention the accuracy of the location-tracking algorithm.

Another method for tracking that has been used in library management is the use of markers. A study conducted in 2003 utilizes mobile AR on a touchscreen device to determine the current position of books on a shelf and aid users in returning books to the correct position on a shelf (Reitmayr and Schmalstieg, 2003). The system (named ARLib) uses frame markers attached to the shelves for tracking purposes. Further, the application stores a model of the library's shelves, positions of individual books and additional metadata. While ARLib can provide a good overview of the location of books, the system does have limitations. First, the application itself stores the entire model of the library (shelf positions and book positions) so it may not be scalable to larger, real-world libraries. Another limitation is that the system does not know the actual location of the book - only the position where it is supposed to be located. This means that if the book or other books are removed from the shelf, the information provided by the AR display may be confusing rather than helpful.

A more recent mobile AR library management system that uses frame markers is ShelvAR, developed by the Miami University Augmented Reality Research Group (Brinkman, 2012). ShelvAR exploits the markers' unique ID by placing a unique marker on the spine of each book. When the shelf is viewed through the phone's viewfinder, simple overlays indicate whether the books are in the correct position on the shelf. While ARLib uses markers for context-awareness, ShelvAR uses the markers to identify single books.

While each library management system mentioned in this paper utilize different methods for tracking books, they all share similar weaknesses. One of the main weaknesses of current mobile AR systems in the literature is that they offer limited context-awareness. While the design presented by Chen et al. (2010) attempts to use the embedded sensors for location-tracking, it was not

proven to be a robust solution. Similarly, it is questionable whether ARLib and ShelvAR would cope with the many possible states in which books and bookshelves can exist. As bookshelves are in a constant state of flux, a successful library management system needs to be robust to all of the challenges faced in a library. We propose a new framework that combines mobile AR with agent-based programming to achieve this.

3. Framework of mobile AR agent-based library management system

3.1. Framework

Fig. 1 shows our proposed framework for a library management system based on mobile AR and agents. We call this system Libagent. Libagent processes two types of input data: Tropicat data and frame markers attached to book spines. Tropicat is a library management system that is currently being used in James Cook University.[1] It provides typical library functions such as catalog search, book/journal browsing, request placing, renewals, and item availability. Tropicat serves as a data bank that can provide Libagent with context-aware information and additional book/journal information. Frame markers on the other hand are used by our AR component to provide a visual connection to the environment, and display context-sensitive information accordingly. The way in which Libagent uses the input information to produce the visual cue outputs is explained throughout this section of our paper.

Libagent consists of three main components: mobile AR, context-awareness (agent programming) and algorithms for sorting, searching and dynamic context-aware updates. Libagent utilizes three main software packages for development: Vuforia [2] for augmented reality, Java Agent DEvelopment framework (JADE) [3] for agent programming, and Blender [4] for 2D/3D modeling. These packages were selected for their open nature, interoperability, and compatibility.

3.2. Android platform

The Android platform was selected for deployment and testing of our prototype. According to Gartner (2013), the Android operating system has a global market share of 79% as of Q2 2013 in terms of smartphone sales to end users. This is compared to iOS (14.2% market share), Windows phone (3.3%), and others (3.5%). Other advantages of using Android include ease of development and deployment to a wide range of devices (Hall and Anderson, 2009). It should be noted that an identical system to our platform could be developed on iOS. However, this is beyond the scope of our current project, which focuses on the efficiency and usability of an AR based, context-aware library management system.

3.3. Mobile AR

While there are several AR frameworks available for mobile programming, our prototype utilizes Vuforia due to its license-free nature which allows for simple development and distribution of applications. The Vuforia SDK provides support for iOS, and Android, and the Vuforia code is implemented using both the Android APIs and C++ (running through the Java Native Interface). The Vuforia SDK provides 512 unique markers (see Fig. 2) that can be used for tracking. Each marker is mapped to a corresponding identification number ranging from 0–511, and this identification number is

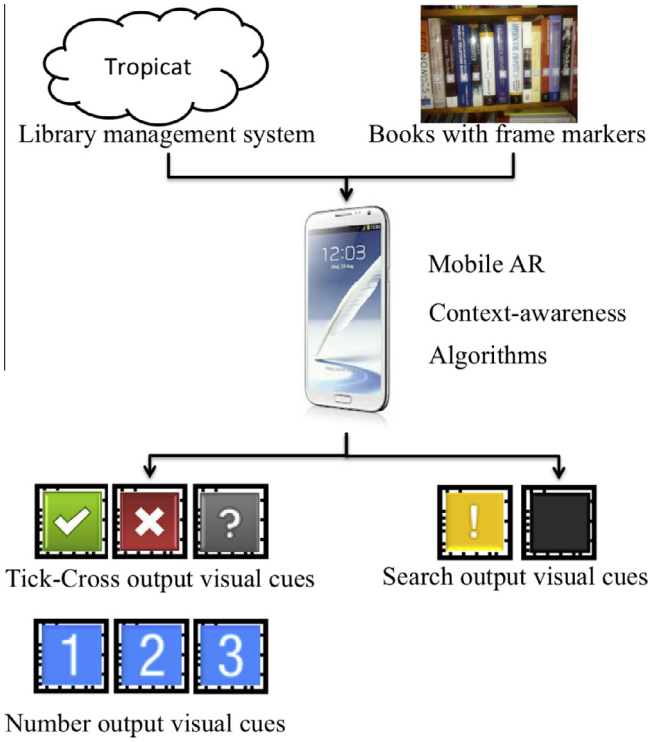

Fig. 1. Framework of mobile AR-based context-aware library management system.

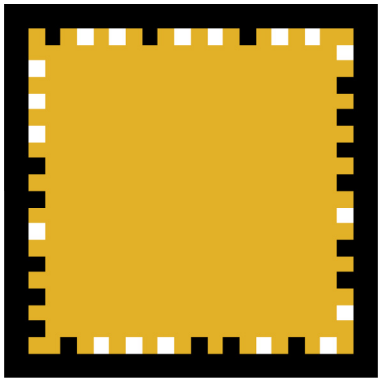

Fig. 2. An example of Vuforia marker.

used by Vuforia to display the correct 2D/3D model for each marker. When compared to image targets, it is clear that fiducial markers provide additional benefits for tracking. These benefits include flexibility (markers can be resized and repositioned with ease), uniform distinctiveness (individual markers with minimal, visual differences can be simultaneously detected and distinguished), efficiency (faster processing) and scalability (markers require less feature-based information to be sorted).

3.4. Context-awareness

For our prototype we used the JADE framework to provide agent functionality. Previous research has indicated that the JADE platform is powerful and can be used to implement thousands of agents without a loss in performance (Chmiel et al., 2005). While our system will use only simple agents (less than ten), we also selected JADE for its compatibility with Android. The software libraries provided with JADE are implemented in the Java

[1] http://www.jcu.edu.au.
[2] http://qualcomm.com/solutions/augmented-reality/.
[3] http://jade.tilab.com/.
[4] http://blender.org.

programming language, which means JADE can easily be referenced and used by an Android application which is also based on Java.

3.5. Visual cues

Due to the limited screen space available on a mobile device, the visual cues provided by the AR system need to be straightforward and simple. Thus, a 2D plane was modelled to represent visual cues due to its robustness to changing textures (i.e. a simple square), minimal use of memory, and simplicity. There are three output types used in our study: sorting visual cues, searching visual cues, and context-aware textual cues. The sorting output type has two modes (tick-cross output visual cue mode and number output visual cue mode) as shown in Fig. 1. A user can toggle between these sorting modes by tapping the screen. The tick-cross visual cues represent whether or not each book is in the correct order on a shelf. On the other hand, the number output visual cues display the correct position of the currently visible books (i.e. how they should be ordered to achieve correctness). The searching visual cue represents a selected search result whilst the context-aware textual cue presents context-sensitive information in text, such as the number of books missing and the number of books on loan from a given subsection of shelf.

It should be noted that for best performance in Vuforia, the size of the texture files are 32×32 pixels. When viewing several markers on the screen of a mobile device simultaneously, it is uncommon for the marker to surpass this size, so minimal resizing allows the application to perform more efficiently. Table 1 displays textures used in Libagent.

3.6. Dynamic texture updating

Many applications utilize Vuforia to display static models. However, our prototype requires dynamic models that adapt to the regular changes occurring on the library shelf. As the native Vuforia code loads its textures into memory upon initialization, that part of the system needs to be reloaded every time a change occurs.

To ensure that this an efficient process, the Java side of the application loads one instance of each texture at run time. Simple strings consisting of comma-separated integers (e.g. 0,1,1,0) are passed between the functions to indicate which markers need to be loaded next (i.e. 0 = cross, 1 = tick). Thus, the Java code replaces the current textures with the new set of textures before calling a native reinitialization.

3.7. Implementing JADE with Vuforia

The process of incorporating JADE into Vuforia and Android is simple and requires a reference to the JADE library from an Android project. For the current prototype, three main agents are deployed. These agents are:

- *BookDBAgent:* This agent is responsible for querying a database that contains the catalogue of books or items corresponding to the markers.
- *ShelfStateCheckerAgent:* This agent is responsible for providing information about the current state of the shelf being viewed through the camera. It works with the *BookDBAgent* to relay information about the ID numbers of any currently visible markers, in order to determine the number of books on loan, the number of books missing, and the metadata for those books.
- *RecommendationsAgent:* This agent is responsible for providing recommendations about similar books to the user. This occurs when the desired book being searched for is unavailable or checked out. This agent also works with the *BookDBAgent* to relay information about the desired book and gain information about similar titles.

These agents are initialized upon the startup of Libagent. Each device running Libagent has its own local set of these agents, and they are utilized in the following algorithms.

3.8. Algorithms

The following algorithms are utilized by Libagent to perform library-based tasks for the user.

3.9. Agent methods

Each agent in Libagent is defined as a Java class that extends the Agent class. Every agent class incudes a *setup ()* function, where any behaviours or services performed by the agent must be initialized. In our prototype, each agent was provided with a *TickerBehaviour*. A *TickerBehaviour* is similar to a standard Java timer, whereby the agent performs a specified behaviour after a defined interval of time has passed.

For example, the *BookDBAgent* has an interval of one second in its *TickerBehaviour*. During each tick, the agent checks the currently visible markers (which are stored in the tracking phase of Vuforia). If the set of markers that are currently visible are different from those recognized in the previous tick, the agent queries a remote database. The information returned by the database includes book titles, book authors, and book marker IDs. This information is utilized by the other agents in the following algorithms.

3.9.1. Sorting

The sorting feature of the application is important for both librarians and casual library users. If a single book is out of place on a shelf, the user may experience great frustration when trying to locate it. The fact that each marker has a unique ID can be exploited for sorting.

Vuforia provides a function for retrieving the current trackable's ID. Thus, it should be a simple matter of passing these IDs to another function for additional processing. However, this technique is complicated by the fact that Vuforia always detects markers by ID ascending. That is, no matter where the markers are positioned in the real-world, Vuforia will always detect them in order from smallest ID to largest ID. Therefore, the application has no information on which order the markers exist in the real world.

One technique that can be used to determine the order of the markers is to transpose the three-dimensional coordinates of the marker to the two-dimensional coordinates of the mobile interface. The horizontal coordinates of each marker can then be referenced with a marker's unique ID number and the system can provide sorting instructions. This method is illustrated in Fig. 3.

Assuming that the position of the camera always represents a fixed horizontal reference point 0, then a frame marker that is located directly in line with the camera will also have a horizontal position of 0. However, if a frame marker is positioned to the right of the camera position, it will have a positive horizontal coordinate (e.g. 10). Conversely, if a frame marker is positioned to the left of the camera, it will have a negative horizontal coordinate (e.g. −10). Fig. 4 shows screenshots of the sorting feature of Libagent. It first displays the tick-cross output visual cues mode and the number output visual cues mode when toggled.

3.9.2. Context-aware information

Utilizing the same information that the sorting algorithm obtains, the software agents can determine context-sensitive information about missing books and books on loan. The first step of

this process is to query the library catalogue to determine shelf location, based on the currently visible minimum marker and the currently visible maximum marker. The agents download this data from Tropicat and compare the results with the visible markers. Books on loan are determined easily as this metadata is already part of the library catalogue (i.e. checked in vs. checked out). On the other hand, missing books are determined by comparing the remaining book IDs to the currently visible IDs. This information is displayed to the user in as text in the top right corner of the screen. A user can tap on this section to open a list of the missing or loaned books (see Fig. 5 For this system to work, Libagent ensures that the shelf is first sorted.

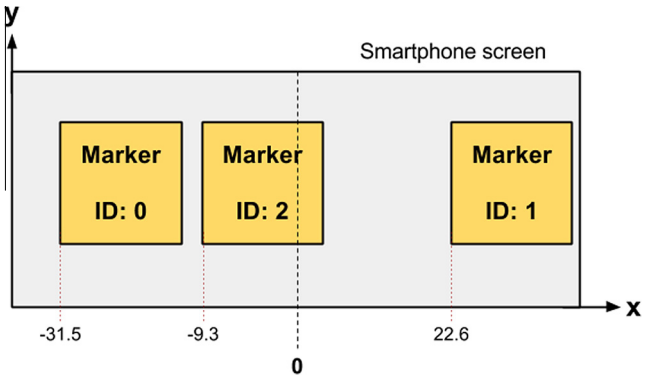

Fig. 3. Detecting frame markers based on position relative to camera.

Fig. 4. Three steps in the Libagent sorting process.

3.9.3. Searching

The searching algorithm utilizes the same data as the context-sensitive information. Based on search terms, a remote, library catalogue database is queried (using a simply MySQL-based web service). The results are displayed to the user in a list on the screen. Once the user selects the desired result, the AR component of the application resets all of its textures, setting a yellow exclamation mark for the desired book and graying out all other textures. Fig. 6 shows a screenshot of this feature in action.

4. Experimental results

In order to determine the benefits of incorporating agents into a mobile AR system, we conducted several experiments. As Libagent was developed primarily for the context of library management, these experiments involve library-based tasks, such as book sorting, book searching, and retrieving information about the shelf. This section will describe the methodologies, results, as well as provide an interpretation of these results.

4.1. Hardware

The device used for testing in the studies was a Samsung Galaxy Nexus smartphone running Android 4.1.2. At the time of testing, this device's specifications are on par with other mid to high-level

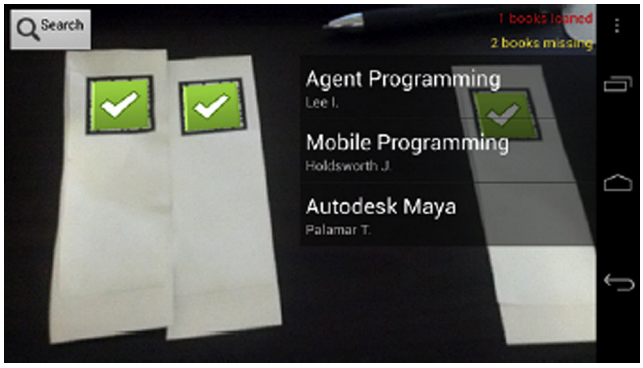

Fig. 5. Libagent displaying context-sensitive shelf information.

Fig. 6. Highlighted search result through Libagent interface.

phones on the market. This device was selected for its larger screen size and memory capacity, which are beneficial features for AR.

Note that our system was briefly tested on other Android devices, including a Samsung Galaxy Y (lower memory, camera and screen size specifications) and a Samsung Galaxy Note II (higher memory, camera and screen size specifications). While we experienced no deficiencies on these devices, the Samsung Galaxy Nexus device was used in all experimentation to reduce inconsistencies in the experimental data. Table 2 shows specifications of Samsung Galaxy Nexus.

4.2. Study 1: robustness of marker technology

As real world libraries can exist in diverse environments, there are a number of extraneous variables that may affect the performance of a mobile AR system. While marker technology is robust to certain visual noise and viewing techniques (i.e. distance, poor lighting, and so on), changes in the environment can result in false positives or negative recognitions, which is an undesirable outcome for tasks like sorting and searching. Thus, a preliminary study was conducted to determine the suitability of markers for use in library environments. Additionally, this study aimed to determine the ideal marker characteristics for successful tracking in a library situation.

This experiment utilized several frame markers taken from the set of sample markers provided by Vuforia. As this experiment is testing marker robustness, the ID of each marker is not important. Each marker was printed at a size of 150 mm^2 with a 35 mm border of white paper. For the experiment, these markers were affixed to book spines and positioned on a shelf. This setup mimics the organisation of books in a real world library. The following conditions were tested:

Distance variations

- *Close distance:* The markers were tracked at a distance of 20 cm, 30 cm, and 40 cm to determine whether all markers could be tracked simultaneously at these distances.
- *Intermediate distance:* The markers were tracked at a distance of 50 cm, 65 cm, and 80 cm to determine whether all markers could be tracked simultaneously at these distances.
- *Long distance:* The markers were tracked at a distance of 100 cm and 200 cm to determine whether all markers could be tracked simultaneously at these distances.

The maximum distance at which the markers were successfully tracked by Libagent was found to be 70 cm. The continuous autofocus functionality of the testing device aided in this, as the phone was able to refocus itself so that it could track the marker at different distances. The distance varied slightly based on the light quality of the surrounding environment. Additionally, it was found that the distance of tracking can be increased if the inside of the marker is coloured. A subsequent test found that for a marker with an orange background the distance of tracking could be increased to 100 cm.

Lighting variations

- *Direct light:* The camera flash was turned on whilst tracking, casting light directly onto the markers that were currently being viewed by the camera.
- *Indirect light:* The main fluorescent lights of the testing room were turned on, which resulted in a lesser amount of light being cast directly onto the book spines.
- *Low light:* The main fluorescent lights of the testing room were turned off, allowing only sunlight from the windows to be cast onto the book spines

It was found that the performance of Vuforia for tracking small frame markers was severely decreased when lighting was poor. In almost all instances, tracking performance was increased by turning on the camera flash to provide artificial light. Similar to the distance experiments, colouring the inside of a marker yielded significantly increased performance under poorer lighting conditions. Based on these findings, the subsequent experiments utilized coloured markers and the camera flash to ensure accurate marker tracking.

4.3. Study 2: correctness of context-aware agents

This study involved testing the correctness of the agents in the system. Specifically, this experiment focused on the correctness of the *ShelfStateCheckerAgent*, which works in collaboration with the *BookDBAgent*. Recall that these agents are used within the system to determine the contextual state of the shelf (or any given subsection of the shelf) in terms of missing books and books on loan.

In order to control for extraneous factors, a replica of a subsection of the James Cook University library was printed onto rectan-

Table 1
Textures used in Libagent.

Texture	Purpose
	These textures are for sorting visual cues in particular for the tick-cross output visual cue mode
	The green tick texture is selected to represent an item that is in the correct order
	The red cross texture is selected to represent an item that is not in the correct order
	The grey question mark texture represents an item which Libagent is unsure about. This could be caused by a failure of the application to recognize the marker ID. The colour grey is selected due to its neutral appearance. As all markers will begin with this texture (it may take a split second for recognition to occur), this marker should not be too distracting as to remove focus from more important markers
	These textures are for sorting visual cues in particular for the number output visual cue mode
  	The blue number textures are selected to represent the correct order of the currently visible books. When a marker has been identified as being in the incorrect order (represented by the red cross), the system will provide instructions to the user so that the shelf can be organized correctly. As there may be multiple markers that are incorrectly placed simultaneously, the system requires a method to show all errors in a non-confusing manner. Thus, all visible markers are given a number, indicating each book's correct position in ascending order. As the numbers are only low (1–9), it is assumed that most users will have the ability to sort them quickly. The color blue is selected for this marker so as to stand out from the other colors (green, red, grey, and yellow)
	These textures for searching mode.
	The yellow exclamation mark texture is selected to represent selected search results when the application is in search mode
	Alternatively, non-desired markers are greyed out

gular cards. These cards included a Dewey Decimal number (e.g. 134 AR) and a frame marker which had been assigned to this Dewey number in the database. Various combinations of these cards were presented to the application (starting with no missing books), and gradually books were removed from the shelf. The number of missing books that Libagent calculated was recorded and compared to the real number of missing books (known by the experimenter). Initially, the experiment started with 30 total cards, and the agents were tested for correctness over four scenarios. The difficulty of these scenarios (in terms of real world applicability) varied from simple (common situation) to extreme (rare situation).

The results of this study are presented in Table 3. In all four conditions, the application provided the correct information about missing books, which demonstrates the robustness of Libagent. No false positives or false negatives were identified with Libagent in this experiment, and 100% recall and precision were recorded. This is a significant improvement which clearly demonstrates the robustness of Libagent.

4.4. Study 3: Libagent vs. JCU Tropicat

The second study aimed to determine the effectiveness of an agent-based AR system at improving efficiency and reducing errors in library-based tasks. Libagent was compared to a current library system (installed on a computer). This system is called Tropicat and it is installed at the James Cook University (JCU) Library. Tropicat allows users to search keywords to retrieve a listing of relevant books from the catalogue. Users are then required to locate the book on the shelf using a Dewey decimal catalogue number (e.g. 139 RAN).

Twenty-one participants were sampled from JCUs student population in Townsville, North Queensland, Australia. Participants' age ranged from eighteen to sixty, with frequency of library usage ranging from rare to frequent. In order to control extraneous variables, a replica of a subsection of the JCU library catalogue was created using thirty markers and Dewey numbers printed onto rectangular cards. Additionally, a local database of books was created for Libagent to use. The information in this database includes book title, author, catalogue number, and marker ID.

Participants were given a demonstration of each system (Libagent and Tropicat) and then asked to complete tasks using each system. Efficiency of task completion was recorded as time taken to complete a task. The first task was a seek and sort task, whereby a random book was placed in the incorrect position. In the Tropicat condition, participants were required to find the erroneous book and place it back in the correct position using only the call numbers on the spine. This task required basic knowledge of ascending numbers and alphabetical order.

In the Libagent condition, participants were required to seek out the erroneous book using AR, and then sort it back to the correct position using recommendations provided by Libagent. Before sorting began, the order of the books was randomly shuffled. The user was then asked to hold the device above the markers to obtain visual cues (ticks or crosses to indicate correctness). Upon locating an erroneous book, the participant would tap on the screen to obtain sorting instructions (numbered visual cues). Using this information, the participant sorted the shelf back to the correct order and then checked their work once more using the tick-cross visual cues. Each task was randomized per participant, and each book was also randomized. Each condition was repeated twice per participant.

The second task was a search task, whereby a participant was required to search for and locate different books on the shelf. In the Tropicat condition, participants were required to find the target book by first searching the JCU Tropicat system to find the call number, and then search the shelf for that call number. For example, if a user was asked to search for a book on PASCAL programming, they could type in keywords including book title or author name to bring up a list of results, and then manually search for the desired book in that list. The list of results includes book title, book author, book call number (in Dewey Decimal format), and book status (e.g. checked in).

In the Libagent condition, participants were asked to search for a book using the application's search interface and then seek it through the viewfinder of the phone. For example, if a user was searching for a book on Java Programming, they were required to tap on the search button and enter a keyword. A list of results was presented to the user with similar information to Tropicat. Upon tapping the desired result, it was highlighted by the AR component using the appropriate texture (yellow exclamation mark). The target book was randomized per condition, and the order of conditions was also randomized per participant. Each condition was repeated twice per participant.

The data collected during this experiment was imported into a statistical analysis program. Table 4 shows the means and standard deviations for the sorting and searching tasks. Note that two outliers are removed from the sorting task and one outlier is removed from the search task to meet the normality assumption. After conducting repeated measures t-test for the sorting task, no significant difference was found between the mean time taken to complete the sorting task with Libagent and without it (t_{18} = -.629, p =.537) with 95% of confidence. Similarly, no significant difference was found between the mean time taken to complete the searching task with Libagent and without it (t_{19} = -.105, p =.917) with 95% of confidence.

Other noteworthy findings from this study were that the application provided accurate and correct results in both the sorting and searching tasks (no false positives and false negatives). While some participants made mistakes during the Tropicat sorting task (which took additional time to recover from), no mistakes were recorded

Table 2
Specifications of Samsung Galaxy Nexus.

Display	4.65" HD (1280 × 720)
Memory	1 GB RAM
OS	1.2 GHz dual core processor
Camera	5MP continuous auto focus
	LED flash
	Zero shutter lag
Sensors	Accelerometer
	Gyro
	Compass
	Proximity/Light

Table 3
Libagent's performance on missing books.

Condition	Results
Simple (one book missing)	0 loaned, 1 missing
Intermediate (five books missing)	0 loaned, 5 missing
Difficult (ten books missing)	0 loaned, 10 missing
Extreme (twenty books missing)	0 loaned, 20 missing

Table 4
Means and StDevs for the sorting/searching tasks.

Task	Mean (seconds)	StDev
Sorting (Tropicat)	34.20	9.55
Sorting (Libagent)	36.34	10.56
Searching (Tropicat)	30.84	11.31
Searching (Libagent)	31.19	10.59

during the Libagent sorting condition. This experiment demonstrates that Libagent achieves better performance in effectiveness (with no false positives and false negatives) with no significant additional time requirement.

4.5. Study 4: perceived usefulness and perceived ease of use

The fourth study involved a qualitative and quantitative survey that was filled out by participants. This survey was adapted from the Perceived Usefulness Perceive Ease of Use (PUEU) questionnaire first devised by Davie (Davis, 1989). Participants for this component were recruited from Study 3, as those participants had already been exposed to our system. Thus, there were twenty-one participants involved in this study. The participants were asked to fill out a survey based on their experiences with Libagent and provide opinions.

The data obtained from the PUEU questionnaire consists of two types. First, the twenty-one participants rated the system in terms of PUEU on a scale of 1–7 (unlikely to likely). On average, participants rated all twelve questions a score of 6 or higher as shown in Table 5. This indicates that the users found the system easy to use and did not foresee any major issues with its use for library-based tasks.

The second set of data gained from the questionnaire is opinions on positive and negative aspects of the Libagent system. The following information represents the common opinions expressed by participants in the survey:

Positive aspects

- Easy to use and requires very little training;
- Eliminates the need to learn the library catalogue system;
- Fast recognition;
- Makes a repetitive and difficult task a lot easer;
- Does not require librarian assistance;
- Would be good for users with cognitive impairments (i.e. dyslexia).

Negative aspects

- Requires adequate lighting;
- If the device is moved too quickly, the camera cannot read the symbols;
- It is difficult to hold the device steady with one hand;
- False positive and false negative recognitions can occur when moved quickly;
- The use of colours may not be suitable for colour blind users;
- The size of the screen may be an inhibitorfor some users.

Positive aspects support the robustness and usefulness of Libagent while some negative aspects could be improved with hands-free wearable systems such as Google glass or better markers and lightings. An interesting point was raised by one participant in that our system may assist users with cognitive impairments, such as Dyslexia. Dyslexia is a condition that is characterised by difficulty in fluent reading, accurate comprehension, and phonological decoding (Grigorenko, 2001). As our system replaces the necessity to rely on the Dewey decimal classification (which uses alphabetical and numeric sequences) with a symbol and colour based alternative, such users may experience fewer difficulties in library-based tasks. Additionally, this could be extended to users from another country who do not speak the language in which the Dewey numbers have been encoded.

Table 5
Means and StDevs for the PUEU (scale of 1–7).

Question	Mean	StDev
Using the system would enable me to accomplish library-based tasks more quickly	6.48	0.60
Using the system would improve my performance in the library	6.48	0.60
Using the system would increase my productivity in the library	6.57	0.60
Using the system would enhance my effectivness when using the library	6.43	0.75
Using the system would make it easier to use the library	6.76	0.44
I would find the system useful in the library	6.76	0.54
Leraning to operate the system would be easy for me	6.71	0.56
I would find it easy to get the system to do what I want it to do	6.57	0.75
My interaction with the system would be clear and understandable	6.67	0.58
I would find the system to be flexible to interat with	6.43	0.68
It would be easy for me to become skillful at using the system	6.76	0.44
I would find the system easy to use	6.81	0.40

4.6. Study 5: individual vs. collaborative sorting

The fifth study aimed to compare the performance of Libagent users individually and collaboratively in a simple sorting task. This experiment was conducted with 63 university students mostly in their last year of undergraduate studies, or first year of postgraduate studies. All participants are currently studying an IT program at James Cook University, and randomization was used to control for differences in cognitive and learning skills. Participants were divided into two groups: an individual sorting group (21 students) and a collaborative sorting group (42 students). The individual sorting group was disjoint from the collaborative sorting group. Note that both groups produced 21 sorting runs, as students in the collaborative sorting group worked in pairs.

In the individual sorting group, each student was expected to perform multitasking (holding a hand-held device whilst sorting) whilst in the collaborative sorting group each pair is assumed to do a single task (one student holding the device and another student sorting objects). In the collaborative sorting task, students were allowed to communicate to collaboratively solve the sorting task.

In both groups (individual vs. collaborative), participants were presented with 7 markers that had been incorrectly sorted. First, participants were asked to sort the markers using only the Dewey decimal catalogue numbers on the book spines. Then, participants were asked to perform the same task using the instructions provided by Libagent. The time taken to complete these sorting tasks were recorded for further analysis.

Table 6 lists the mean times in seconds for the sorting tasks for each group with and without the app. This data clearly shows the sorting time is faster for the collaborative sorting group using the app. This implies that cognitive load must reduce for each student in the collaborative sorting group, allowing them to focus on a single behaviour during the sorting task. In contrast, individual students must focus on holding the device and interacting with it, whilst simultaneously manipulating the markers in the real world.

Table 6
Means and StDevs for the sorting/searching tasks.

Group	Task	Mean (s)	StDev (s)
Individual	without app	21.62	10.84
	with app	27.18	10.87
Collaborative	without app	27.81	4.29
	with app	8.99	1.06

Table 7
Significance *t*-test (df:20, $p = 0.1{:}1.72$, $p = 0.05{:}2.09$, $p = 0.01{:}2.85$).

Task	*t*-test statistic
Individual no app vs. Individual app	1.66
Collaborative no app vs. Collaborative app	11.91
Individual app vs. Collaborative no app	2.01

Based on this data, we conducted further analysis. Table 7 presents the results of paired-sample *t*-tests for the comparisons between sorting groups and app usage. We found that the app provided a significant improvement in sorting time for individuals in the collaborative group. This implies that each member of a collaborative pair has reduced cognitive load (in terms of multitasking), and can focus on performing a single task more efficiently.

In contrast, Libagent did not significantly change the performance of an individual student. This implies that focusing on the app for guidance or just relying on common-sense makes no real difference in overall performance. This seems plausible since an individual must be continually switching between using the app and moving the markers. The positive effect of using the app is negated by the negative effect of multitasking between behaviours.

4.7. Discussions

4.7.1. Benefits of Libagent

Based on the results of our experiments, it is clear that there are benefits of using agents to support AR applications. The second study concludes that our agent-based AR system for library management is accurate at providing context-sensitive information about the state of a shelf. This information reveals very quickly (< 1 s) the number of books missing or on loan from the current shelf space, and also provides the metadata for these items. This information can be highly valuable to both library users and librarians alike.

While experienced library staff may have an increased ability to recognize a missing item, such skills require an intimate knowledge of the catalogue and the numbering system. The majority of library users will not have this knowledge, and will benefit from the speed of recognition provided by Libagent. Additionally, librarians (and all humans) are limited in terms of short-term memory capacity, which means that Libagent will be faster in scenarios that involve a large number of missing items (e.g. the difficult and extreme conditions in our experiment). For this reason, we can conclude that the agents can assist in overcoming cognitive limitations in library-based tasks.

4.7.2. Limitations of our research

Despite the results of our third study not providing evidence that our system is more efficient than traditional methods, there are some methodological concerns that may have contributed. First, the participants of the experiment may not have had enough time to become proficient with Libagent. The participants of the study consist of students from different backgrounds, including different ages (varying from 18 to 60), different levels of library experience (rare to frequent), and assumedly different experiences with technology. The fact that participants had limited time to experience Libagent may have hindered their performance. In contrast, many users will have searched for a book using traditional methods, so this task may have been faster in the given circumstances.

Another methodological concern may have been an error in sampling. While the current study recruited participants of all ages and level of library experience, it may have been more beneficial to recruit participants with less library experience. For users who are already proficient in traditional methods of sorting and searching, the new application may have appeared to make these tasks more difficult. On the other hand, users with less library experience may have performed better with Libagent, due to their limited experience in both traditional methods and the new application. A larger sample size may also be necessary to provide an accurate result.

Finally, the fact that users were expected to perform the library tasks on a small subsection of books (30) is not a true representation of the way in which a library is used. In reality, there are thousands of books present in a library, and a user must first locate the correct shelf before attempting to locate the desired book on that shelf. The search space of a real library is much larger, meaning that users are faced with hundreds of books on any given shelf, all contributing to the difficulty of finding a single book. In contrast, a small sample of 30 books may not prove to be difficult at all, resulting in faster response times from participants in the study.

4.7.3. Error reduction

While an agent-based AR system may not save time for users in sorting and searching tasks (according to our results), there are still great benefits for the system to reduce errors. As was concluded in the first study, the application is highly accurate at determining which books are missing from the shelf at any given time. This makes the task of searching for a book in the library a lot easier, as the user is efficiently provided with information about the context of the shelf. Additionally, the sorting task completed by the participants in our second study reports accurate results. Libraries employ staff to keep the shelves in order, however this task can prove tedious and error-prone. Thus, the elimination of errors could provide saving of time and money.

Once again, the error saving benefits may be caused by the applications ability to overcome the limitations of the human short-term memory. Short-term (or working) memory holds limited information for a short duration while a person completes an everyday task. The average person's short-term memory has the capacity to hold between five and seven items for around 15 to 30 s (Miller, 1956; Keppel and Underwood, 1962). Further, individuals who are skilled in a task (such as librarians) may have strategies in place to recall greater amounts of information within this range (Ericsson et al., 1980). Thus, the present design would be most useful when dealing with a range of books greater than seven. This indicates that the present study's implementation for library-based tasks may represent a realistic use of Libagent for solving a real-world problem.

4.7.4. Device shaking

A limitation of the application that was discovered during the testing phase was its weakness to device shaking (as mentioned by participants in the PUEU). While the current design only experienced minimal shaking caused by a human holding the device, other real-world applications may be prone to more intense shaking (e.g. a windy environment). Thus, such mobile AR applications would benefit from a system to reduce the effect of shaking on AR tracking. There are currently solutions for this problem cited in the literature.

One current method for reducing the effects of a shaking screen is NoShake (Rahmati et al., 2009). NoShake utilizes the accelerometer sensor that is embedded in most consumer electronic devices available on the market today. The system dynamically compensates for device shaking by shifting the screen content in the opposite direction of the shake. By incorporating a content stabilization system such as NoShake into the Libagent application, the accuracy and success-rate of marker tracking may be increased, allowing for a more efficient sorting experience.

With the current trend moving into the field of wearable computers, this limitation could be remedied within the next decade. Despite the fact that the concept of wearable computing (and prototypes of such concepts) have existed for several years, these prototypes were not portable or feasible for everyday use. The new age of wearable computing has benefited from the advances in technology, with convenient devices such as smart watches and smart glasses approaching ubiquity. One example is Google's Project Glass.[5]

Having a user wear such a device (i.e. "smart glasses") would provide the user with the freedom to manipulate the real world using both hands simultaneously, whilst keeping the interface steady. Study 5 concluded that there were multitasking effects present with Libagent, and that performance for sorting significantly increased when a user focused their attention to one task only. By removing the need to hold a device steady with their hands, a user should have perform a task more efficiently and more accurately. Migrating the application to such a device would have other benefits, as it would overcome the small screen size (mentioned as awkward by one of the participants in Study 4). This is because the user would essentially be able to view the world regularly through their own eyes (as opposed to through a screen), with the augmentations appearing more naturally.

4.7.5. Deployment in a real library

Our deployment and experimentation with Libagent was limited in scope. We used a very small subsection of a single library shelf (30 books), which in itself is not representative of a real-world library. According to the U.S Department of Education National Center for Education Statistics, the average number of books in public school libraries in the USA in 2008 was 2,015 per 100 students. This number had increased from 2004, when 1,891 books per 100 students was reported (NCES, 2012).

While our prototype is not currently programmed to deal with hundreds of thousands of books, the technologies utilized (i.e. agents and AR) are capable of scaling up to much larger sets of data. For example, marker systems (such as QR codes) can allow for hundreds of thousands of permutations, depending on the number of unique bits displayed in the marker. A potential limitation of using Vuforia for marker tracking is that the SDK provides on 512 unique markers, however there exists the possibility to use a custom marker type. For example, the developers of Shelvar created a new set of markers that can be scaled to include thousands of unique possibilities (Brinkman, 2012).

Another possibility may be to reuse the 512 markers in different sections of the shelf space, alongside another method (e.g. feature or image tracking) to gain a shelf-context prior to tracking the individual markers. The system would become aware of the shelf it is viewing and gain information about that shelf (e.g. book order and meta data) from a remote server. When using a mobile AR system it is possible for the user to view only a small amount of books at any given time, so there is no need for the system to store data about thousands of individual books simultaneously.

Other factors to consider in a real world library are the additional library-based tasks facing a user. For example, before locating a book, wayfinding is necessary to assist the user in navigating the physical shelf on which the book is located. Currently, libraries assist users in this task with visual signs to indicate which books exist on which shelf. Future researchers in this field should investigate the potential of using mobile AR wayfinding alongside AR library management tools to provide a complete solution.

[5] http://www.google.com/glass/start/.

5. Conclusion

The current study aimed to investigate the effectiveness and usefulness of agent-based programming for providing context-sensitive information to mobile AR applications. This was achieved through the design and development of an application titled Libagent. The final prototype was able to assist with difficult sorting tasks (for both groups and individuals), searching for books, as well as provide context-sensitive information about missing books and books on loan. While individual users did not appear to achieve faster results with Libagent when compared to traditional methods, the application provided correct results all of the time – a better result than humans performing the task alone. Additionally, participants of the study (both individuals and groups) rated the system highly in terms of perceived usefulness and perceived ease of use.

The main contributions of the research presented in this paper are threefold. Firstly, it is a contribution to the field of agent programming on modern devices. According to Chmiel et al. (2005), the increasing power of hardware has resulted in the perfect environment for developing and studying the nature of distributed, multi-agent systems (Chmiel et al., 2005). Secondly, the combination of both AR and agent-based technologies for context-awareness is a novel contribution, as this is an area that has experienced limited research on modern day devices. Thirdly, a robust framework of Libagent combining mobile, agents and AR for library management is a novel contribution.

Based on our results, it is clear that there are benefits for combining mobile AR with agent-based technology. With further modifications, this system could provide an efficient alternative to the repetitive library tasks that are currently performed by humans. Furthermore, the system could be expanded into other areas, such as warehouse inventory management. Our prototype, Libagent, provided only a minimal implementation of software agents to assist with basic tasks. Thus, it is recommended that future research in mobile AR consider the use of software agents to enhance the information provided by such systems, and to provide more insights into the benefits of agent technologies for mobile AR.

References

Baldauf, M., Dustdar, S., & Rosenberg, F. (2007). A survey on context-aware systems. *International Journal of Ad Hoc and Ubiquitous Computing, 2*, 263–277.

Brinkman, B. (2012). Shelvar. <http://www.shelvar.com/>.

Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E., & Ivkovic, M. (2011). Augmented reality technologies, systems and applications. *Multimedia Tools and Applications, 51*, 341–377.

Chen, D. M., Tsai, S. S., Girod, B., Hsu, C. H., Kim, K. H., & Singh, J. P. (2010). Building book inventories using smartphones. In *Proceedings of the International Conference on Multimedia* (pp. 651–654). ACM.

Chmiel, K., Gawinecki, M., Kaczmarek, P., Szymczak, M., & Paprzycki, M. (2005). Efficiency of jade agent platform. *Scientific Programming, 13*, 159–172.

Davis, F. D. (1989). Perceived usefulness, perceived ease of use, and user acceptance of information technology. *MIS Quarterly*, 319–340.

Engelke, T., Becker, M., Wuest, H., Keil, J., & Kuijper, A. (2012). Mobilear browser-a generic architecture for rapid ar-multi-level development. *Expert Systems with Applications, 40*(7), 2704–2714.

Ericsson, K. A., Chase, W. G., & Faloon, S. (1980). Acquisition of a memory skill. *Science, 208*, 1181–1182.

Fabio, B., Giovanni, C., Dominic, G., et al. (2007). *Developing Multi-Agent Systems with JADE*. Wiley.

Furata, H., Takahashi, K., Nakatsu, K., Ishibashi, K., & Aira, M. (2012). A mobile application system for sightseeing guidance using augmented reality. In *2012 Joint 6th International Conference on Soft Computing and Intelligent Systems (SCIS) and 13th International Symposium on Advanced Intelligent Systems (ISIS)* (pp. 1903–1906). IEEE.

Gartner, 2013. Gartner says smartphone sales grew 46.5 percent in second quarter of 2013 and exceeded feature phone sales for first time. <http://www.gartner.com/newsroom/id/2573415/>.

Genesereth, M. R., & Ketchpel, S. P. (1994). Software agents. *Communication of the ACM, 37*, 48–53.

Grigorenko, E. L. (2001). Developmental dyslexia: An update on genes, brains, and environments. *Journal of Child Psychology and Psychiatry, 42*, 91–125.

Hall, S. P., & Anderson, E. (2009). Operating systems for mobile computing. *Journal of Computing Sciences in Colleges, 25*, 64–71.

Holz, T., Campbell, A., O'Hare, G., Stafford, J., Martin, A., & Dragone, M. (2011). Mira – mixed reality agents. *International Journal of Human-Computer Studies, 69*, 251–268.

Keppel, G., & Underwood, B. J. (1962). Proactive inhibition in short-term retention of single items. *Journal of Verbal Learning and Verbal Behavior, 1*, 153–161.

Kounavis, C.D., Kasimati, A.E., Zamani, E.D., & Giaglis, G. 2012. Enhancing the tourism experience through mobile augmented reality: Challenges and prospects. In: the Proceedings of the 2nd Advances in Hospitality and Tourism Marketing & Management (AHTMM 2012) Conference, Corfu, Greece.

Lee, G. A., Dunser, A., Kim, S., & Billinghurst, M. (2012). Cityviewar: A mobile outdoor ar application for city visualization. In *IEEE International Symposium on Mixed and Augmented Reality (ISMAR-AMH)* (pp. 57–64). IEEE.

Maes, P. et al. (1994). Agents that reduce work and information overload. *Communications of the ACM, 37*, 30–40.

Marimon, D., Sarasua, C., Carrasco, P., Álvarez, R., Montesa, J., Adamek, T., Romero, I., Ortega, & M., Gascó, P. (2010). Mobiar: Tourist experiences through mobile augmented reality. Telefonica Research and Development Barcelona, Spain.

Miller, G. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review, 63*, 81–97.

Mulloni, A., Seichter, H., & Schmalstieg, D. (2011). Handheld augmented reality indoor navigation with activity-based instructions. In *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services* (pp. 211–220). ACM.

Mulloni, A., Seichter, H., & Schmalstieg, D. (2012). Indoor navigation with mixed reality world-in-miniature views and sparse localization on mobile devices. In *Proceedings of the International Working Conference on Advanced Visual Interfaces* (pp. 212–215). ACM.

Nagao, K. (1998). In T. Ishida (Ed.), *Agent augmented reality: Agents integrate the real world with cyberspace. Community Computing: Collaboration over Global Information Networks*. John Wiley & Sons.

NCES (2012). Chapter 7: Libraries and educational technology. <http://www.nces.ed.gov/programs/digest/d11/ch7.asp>.

Olsson, T., Lagerstam, E., Kärkkäinen, T., & Väänanen-Vainio-Mattila, K. (2013). Expected user experience of mobile augmented reality services: a user study in the context of shopping centres. *Personal and Ubiquitous Computing, 17*, 287–304.

Rahmati, A., Shepard, C., & Zhong, L. (2009). Noshake: Content stabilization for shaking screens of mobile devices. In *IEEE International Conference on, Pervasive Computing and Communications, 2009. PerCom 2009* (pp. 1–6). IEEE.

Reitmayr, G., & Schmalstieg, D. (2003). Location based applications for mobile augmented reality. *Proceedings of the Fourth Australasian User Interface Conference on User Interfaces 2003* (Vol. 18, pp. 65–73). Australian Computer Society, Inc..

Rodriguez-Sanchez, M., Martinez-Romo, J., Borromeo, S., & Hernandez-Tamames, J. (2013). Gat: Platform for automatic context-aware mobile services for m-tourism, Expert Systems with Applications.

Smit, M., & Barnett, R. J. (2010). A comparison of augmented reality indoor navigation systems with traditional techniques.

Van Krevelen, D., & Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality, 9*, 1.

Wooldridge, M., Jennings, N., et al. (1995). Intelligent agents: Theory and practice. *Knowledge Engineering Review, 10*, 115–152.

Zaslavsky, A. (2004). Mobile agents: Can they assist with context awareness? In *IEEE International Conference on, Mobile Data Management, 2004. Proceedings* (pp. 304–305). IEEE.

Zhou, F., Duh, H., & Billinghurst, M. (2012). Trends in augmented reality tracking, interaction and display: A review of ten years of ismar. In *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008. ISMAR* (pp. 193–202). IEEE.

Zickuhr, K. (2012). Three-quarters of smartphone owners use location-based services. Technical Report. Pew Research Center.