COSC426 Research project: ARDog for Interactive Advertising and Entertainment

Kris Nicholson, Johannes Pagwiwoko, Brenda Zheng
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch, New Zealand
{kvn12, jaa62, czh23} @ student.canterbury.ac.nz

October 20, 2008

ABSTRACT

Animated augmented reality agents provide a compelling experience for users. They are grounded in the real world, so they can interact with physical objects, unlike traditional virtual agents. In this paper we describe ARDog, our prototype augmented reality virtual agent designed to be used as an interesting advertising medium. It uses computer vision techniques to track physical objects such as a ball, allowing users to interact with it in a tangible way. Users can feed the dog to curb hunger and play with the ball to keep it entertained. After developing a demonstration, we are satisfied with the results and how users feel about playing with the dog. However more work is required to make the dog seem more lifelike.

1 Introduction

Augmented Reality (AR) is the idea of augmenting the real world with virtual information. Real objects are tracked in 3D so that virtual objects can be associated with them. It is an important area in human computer interaction because it allows humans to interact with computers in a natural way.

One area of research in the field of AR is virtual agents. An AR virtual agent is a virtual character which is anchored in the real world. In this way they can help a user with a task more effectively than a virtual agent without the use of AR, since they are a part of the task space and are able to interact with the physical aspect of the task at hand.

Virtual agents are also interesting because of their ability to engage and entertain the user. An AR virtual agent has the advantage that it can interact with the user in a more tangible and natural way so that they do not feel like they are interacting with a computer. Using this engagement, people could be drawn to a particular cause like a form of advertisement.

In our work, we describe our system, ARDog, which provides an interactive AR virtual agent which looks and behaves like a pet dog. Users are able to interact with the dog by rolling a ball for it to chase and by giving it a bone as food when it is hungry.

We propose that this system could be used, not just as entertainment, but for advertising as well. An organisation such as the RSPCA could use the virtual ARDog to raise the issues regarding abandoned dogs. By using a relevant virtual agent, it should provide a link from their current experience to the real dogs that need help.

In this paper, we investigate previous work with animated virtual agents and the use of vision in AR. We then describe the design of our prototype system and some preliminary results.

2 Related Work

There are two areas of related work that we investigated: autonomous virtual agents for augmented reality, and computer vision applications in augmented reality.

2.1 The Use of Autonomous Agents in Mixed Reality Applications

Wagner, et al. [1] present a handheld AR educational application in which a virtual character can teach users about art history.

Wagner, et al. investigate how AR technology can be used to bring virtual characters into the real world and compare AR characters to other types of virtual characters. They also have gone further, unlike previous works with AR characters, where they present results from a conducted user study. Their main focus is particularly to explore the effect of different virtual character representations on user engagement, enjoyment and educational outcomes in learning a particular set of tasks.

For the study, they develop a handheld AR educational game in which a virtual character teaches users about art history. As users try to complete their objective, they can summon the virtual character to provide hints and or directly help them in their tasks. The users interact with five different representations of the virtual character: Text only, Text & Audio, 2D Image, 3D virtual character, and an AR character.

They find that for a virtual character to be engaging, an audio element has to be present. Audio improves a virtual character's perceived friendliness and the realism of the character. They also found that an animated 3D virtual character gives a significant boost in how users perceive the realism of virtual characters compared to the 2D, Text & Audio or Text Only counterparts.

The most interesting discovery made in this paper is that 3D characters and AR characters are equally part of the real world unless the AR character interacts with objects in the real world. Otherwise, there is little merit in having an AR virtual character. In addition, although the AR character exists in users' physical space, there is no actual physical interaction between the AR character and the users.

These two weaknesses can be greatly improved by having an AR character that can interact with their environment (other AR or virtual objects) and also using Computer Vision techniques or Tangible Interface to allow interaction between users and the AR characters.

Maes [2] talks about the potentials, challenges and applications of life-like autonomous agents in the field of entertainment, but especially in Virtual Reality (VR).

Maes covers the topic of artificial life in the field of entertainment. The author briefly describes the idea of artificial life and autonomous agents as a gentle introduction. They in particular suggest why the field of entertainment is one of the more promising applications for life-like autonomous agents.

However, the author states that there are still challenges, especially in creating entertaining agents. One of the most obvious challenges is that most of these characters are extremely simple-minded; they demonstrate a very predictable behaviour and do not seem to be convincing. Interaction with autonomous agents in real time turns out to be a major challenge as well. For autonomous agents to have life-like behaviours, they have to be non-mechanistic, non-predictable and spontaneous.

The author then proposes some possible approaches to overcome these challenges: That the agents should be modelled as distributed, decentralized systems consisting of small competence or "expert" modules that are able to communicate with each other. This allows them to be more robust, reactive and adaptive to external events. They also must be able to

react and interact within their environment in which they live in for them to be convincing.

The author also encourages designers of autonomous agents to think more about the user and to study how users perceive the virtual characters. The author mentions ALIVE [3] as an example (case study) of an application that has successfully implemented autonomous agents, having adaptive life-like behaviours, and conclude that it requires an interdisciplinary approach that combines human sciences with artificial intelligence to build a successful life-like autonomous agent.

Maes suggests the idea of having an interdisciplinary approach to build a successful autonomous agent and also encourage designers to think more about the users. We think Wagner, et al. achieve this as part of their user study.

Both papers by Wagner, et al. and Maes have given us useful insights and information regarding the potentials, challenges and the proposed solutions or approaches to build a successful, immersive AR interactive advertisement. We learn from Wagner, et al. that it requires user-agent interactions, agent-environment interactions and audio element to have a more convincing AR characters. We also learn from Maes that there are some required AI techniques that needs to be implemented to have a non-predictable, adaptable, dynamic and spontaneous autonomous agents.

Barakonyi, et al. [4] describe an animation framework for augmented reality. In this framework they use a puppeteering metaphor to provide a storytelling architecture.

A puppet is at the lowest level and represents an AR agent which can be a virtual 3D or physical agent (a robot). A puppeteer knows how to get an agent to perform a goal by performing small tasks. A choreographer informs the puppeteer which goal it should be trying to achieve currently, such as moving to an object and picking it up. Finally, a director is at the highest layer and represents application logic.

As part of their report, they describe an example pilot application they created using their framework, AR Lego. In this application, two agents are used to help the user perform machine maintenance. One is a virtual repairman which is a virtual animated agent. The other is a LEGO Mindstorms robot, which is the machine to be maintained. The virtual repairman indicates to the user how to perform the next step by doing the action virtually himself. The physical robot is tracked, so the virtual agent is able to find the location of the robot as well as the location that the next piece needs to go. As soon as a step such as mounting an engine is completed, the robot can attempt to turn it on. If unsuccessful, the previous step will need to be repeated or corrected at the instruction of the virtual agent.

The paper describes ways for a virtual agent to interact with physical objects. In particular, they will watch and react to changes in locations of physical objects. We would like our ARDog to be able to interact with physical objects such as a ball also, however, we will not be using physical agents. It also describes an interesting hierarchical design for the autonomy of agents. This may influence how we implement our dog's actions. However, we do not plan on the dog having particularly complex AI.

The idea of physical and virtual agents working together is intriguing, but an example with a larger emphasis on what the physical agent does in this respect would have been useful.

In a following paper from Barakonyi, et al. the work of their earlier paper is extended, but this time the focus is on the mobile nature of augmented reality agents.

Here the authors describe a system where virtual agents can be transferred between

different interfaces automatically based on the suitability of that device to the task at hand. A central database is used to store information about agents and the places that they can reside.

It also stores persistent information about the agent, such as user preferences. An agent brain (or server) knows about the current state of the agent and controls migration between different habitats depending on the task at hand. For example, the lego agent could move to a larger screen if the current construction step is too detailed to fit on a currently small screen.

An example application is described, where the agent could actually be a character that is to be modelled and animated at a character animation studio. The modeller would create the initial model on his PC. This could be transferred to a PDA and taken to the animator's workstation. A producer could then take the character to a meeting and move it to a projector for a presentation.

The paper describes a very interesting model for networking devices with a central brain to control the agent. This will be relevant if we decide to have the ability to have multiple viewers interacting with the dog collaboratively as the actions that the dog takes and its current state will need to be controlled by a central server.

For the second paper, although the example provides a good explanation of how the system works, it does not seem to be particularly valuable. In particular the PDA seemed to be an unnecessary step in the process for what they described. If a single user was using the system, then the PDA would conceptually allow the user to know that they have the agent with them. But to transfer between machines, why not just do it directly through the network.

2.2 Vision-based recognition in AR applications

2.2.1 Gesture Recognition

Moritz and his colleagues [6] introduce a computer vision-based gesture recognition AR interface. In order to recognize gestures, the first step is to capture motion and configuration of objects; the second is to classify the captured information.

The authors claim that gesture recognition systems can be classified into model-based and appearance-based systems. Model-based systems require creation of a geometric model of the hand. Appearance based systems concentrated on each pixel of the captured image.

The authors apply low-level segmentation. In order to overcome the invariance of intensities problem, RGB colours are transformed to a colour space regardless of intensity. Pointing, clicking and another five gestures are recognized in this system.

The authors built a computational simple gesture recognition system in an AR interface. The low-level segmentation is applied for the detection and recognition. Chromaticity is involved to achieve invariance to the intensity.

The colour of objects is used as a feature for gesture recognition. Although the illumination intensity could be normalized, the illumination colour is difficult to be transformed. In addition, the author reveals that cameras have a limited dynamic intensity range. These areas can be improved in further work.

ARDog uses vision-based tracking to monitor real objects such as a ball, while gesture recognition system is computer vision-based for the AR interface. Some knowledge addressed by the authors of gesture recognition can be applied with ARDog project, such as RGB colour transformation in the segmentation process. ARDog requires the

similar pre-segmentation as the gesture recognition in AR interface.

2.2.2 ARKB: 3D vision-based augmented reality keyboard

Lee & Woo [7] introduce a wearable 3D Augmented Reality Keyboard that allows a user to input text without using a traditional keyboard or mouse. ARKB capture markers using a camera on a see through head mounted display and uses the ARToolkit tracking engine to align 3D virtual objects.

ARKB has a natural interface and offers users a more realistic AR experience. The author claims that ARKB doesn't require any physical sensors to observe the collision between fingers and the virtual keyboard. Only 3D position information is used to monitor such collisions.

The ARKB recognizes a user's hand according to the colour of skin. It enables interaction between a virtual object and the user. The author states that the obtained 3D position information provides the AR keyboard on a marker.

ARKB also provides audiovisual feedback when there is collision between a key and a finger to offer a more ideal typing feeling.

The authors show that ARKB uses stereo camera on a see through HMD. This can help to determine the depth of 3D image. Therefore, the collision between fingers and AR keyboard can be detected.

The ARKB offers a natural interface rather than a traditional interface. However, there is an image delay which influences the accuracy. The accuracy of ARKB could be improved by reducing this delay.

ARDog uses vision-based tracking to monitor real objects such as a ball. This is quite similar to the ARKB. Such knowledge addressed in the article describing ARKB can be adopted to improve ARDog such as using stereo camera, which could be applied to monitor the collision between a user's hand and ARDog. However, a stereo camera has not been used in this project and we have not implemented hand tracking.

3 Design and Implementation of ARDog

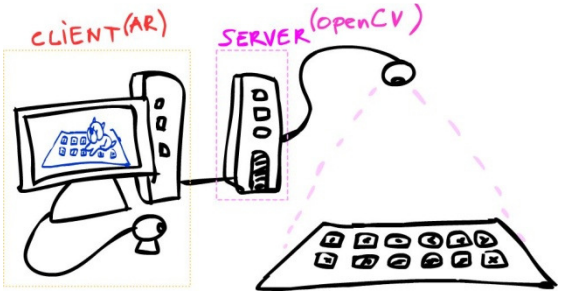


Figure 1: Concept for ARDog

3.1 ARDemo Setup

To demonstrate our idea, we created a prototype demo, named ARDog. The physical demo has three parts; a wooden enclosure with an ARToolkit multimarker attached, a server with a camera connected suspended above the enclosure looking directly down, and a client computer.

The server and enclosure together form the demo installation. Ideally, this would be a persistent setup, perhaps installed as part of a booth for the SPCA or other organization. The server runs an application built with the OpenCV library to optically track physical objects for users to interact with. It also waits for a client connection. Currently it can only serve one client.

To set up the server initially, a hot key should be pressed to record the background frame. Using the mouse, the colours to track should then be selected from the window. The first colour selected represents the colour of the ball, and the second colour selected represents the colour of the bone.

The client is currently a standard laptop computer, but could potentially be a mobile device. It runs an application built with the

OSGART library which combines the optical Augmented Reality tracking of ARToolkit with the powerful OpenSceneGraph graphics library. The client connects to the server and receives the 2D position of physical objects in the enclosure. Using this information it displays the ARDog to the user and shows its behaviour and responses to users' interaction.

3.2 The Dog & content pipeline

As we have discussed in the previous section, Wagner, Billinghurst and Schmalstieg had came to a conclusion on how to produce convincing virtual autonomous agents. And thus, we aim to have our virtual dog (named Shiba, of the Japanese Shiba Inu breed) to possess the following properties:

- Shiba should exhibit dog characteristics through its actions and appearance
- Shiba should be able to interact with the users, and also its environments
- Shiba should be able to make sounds so that it appears alive

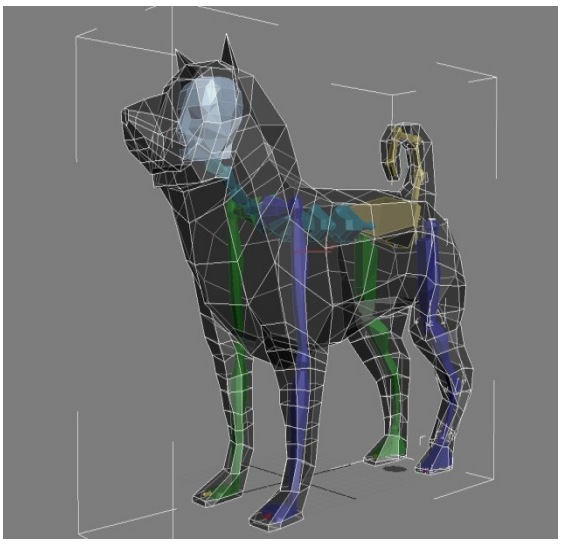
All the modelling, rigging, basic texturing and animation work were done in Autodesk 3D Studio Max [8]. We decided to use the trial version of Autodesk 3D Studio Max 9 which is licensed for non-commercial purposes. We chose 3D Studio Max because it is commonly used by the mainstream gaming and 3D application developer community, and therefore there would be much support from the online community. This would greatly aid us to learn quickly how the art pipeline works considering the short time span of this project.

We found two pipelines in which the produced 3D Studio Max assets can be exported into OpenSceneGraph:

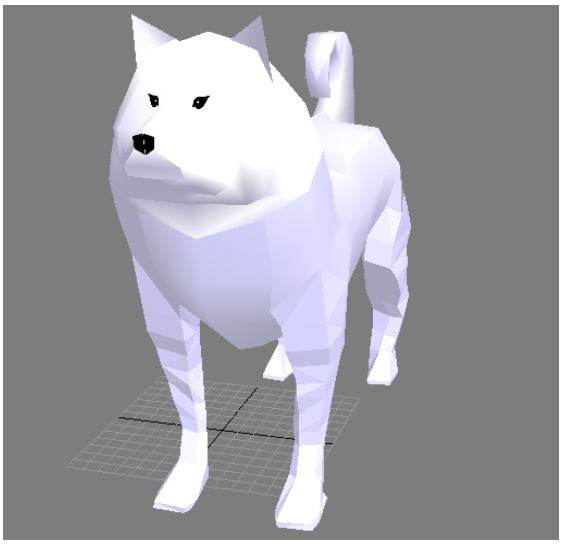
1. Exporting 3D Studio Max's skeleton, mesh, texture and animation assets into Cal3D [9] assets using Cal3Dexporter, and then, using the osgCal2 [10] adapter plugin (or possibly the ReplicantBody[12] animation library) to provide a layer of transparency between OpenSceneGraph and Cal3D assets.
2. Exporting the max files directly into OpenSceneGraph by using osgExp[12] (also known as osgMaxExp) exporter plugin.

At first, we planned to use the first pipeline because it would provide us with greater flexibility in creating the animations. Also, there would be a possibility to control the dog (such as controlling Shiba to look at the direction of the camera).

We managed to convert our assets into Cal3D assets but not the texture assets. We were also faced with many problems in getting osgCal2



(b) A modified human biped for the dog



(a) Shiba, the virtual dog

Figure 2: 3D Models for the virtual agent, Shiba

to interface with OpenSceneGraph. And furthermore, the amount of documentation for both Cal3D and osgCal2 are very scarce. Thus, considering the time that we had on this project, we opted for the second method, that is, exporting the 3D Studio Max assets directly into OpenSceneGraph's native binary and ASCII formats (IVE and OSG respectively).

The second method, however, has some drawbacks. The more complex the assets, the bigger the IVE and OSG files size will be, and the file size could grow to an unmanageable size. Therefore, we decided to use a low polygon count, simple textures and simple looped animations so as to minimize the size as much as possible. The osgExp exporter also provides 3D Studio Max with OpenSceneGraph's VisibilityGroup Helper to hide the skeletal model from being visible and Sequence Helper to export the animations into OpenSceneGraph.

3.3 Vision-based Tracking

ARDog uses vision-based tracking to monitor real objects such as ping pong ball and bone made of plastic. This is done through two computer vision techniques: background subtraction and colour detection.

3.3.1 Background Subtraction

As the camera on the server side is always fixed, we use background subtraction to identify moving objects. After we obtain a static background (Figure 3a), moving objects can be detected according to the difference between current video frame (V_c) and the background frame (V_b), where:

$$|V_c - V_b| > Threshold$$

Then, we generate a new image that ignores all pixels from background frame, and only shows the moving objects. (Figure 3b)

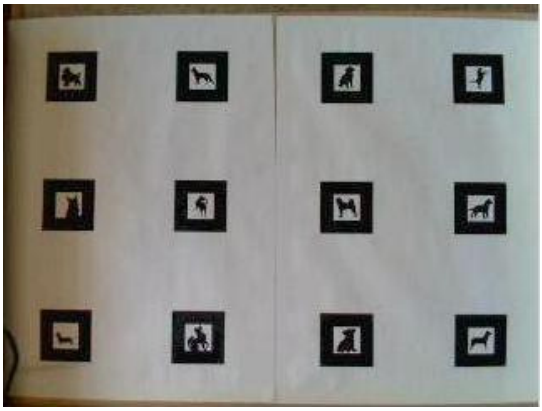
The advantage of using background subtraction in our project is to increase the accuracy of objects recognition, as the range of objects has been reduced after background subtraction.

3.3.2 Colour detection

We use colour detection to identify objects. In order to find a green ball in the image, we first take an image that contains only the ball (Figure 4b) and calculate its 2D hue-saturation histogram. The histogram could have a strong maximum for the green color (Figure 4c). Then, back projection is calculated according to the histogram to get the image, where bright pixels correspond to green colors (Figure 4d).

The back projection image is then smoothed and thresholded in order to construct the binary image. Smoothing and thresholding can eliminate some noise to a certain extent. Edge detection is then used to find the contours of the binary image. A contour is a set of pixels that surround and enclose an area.

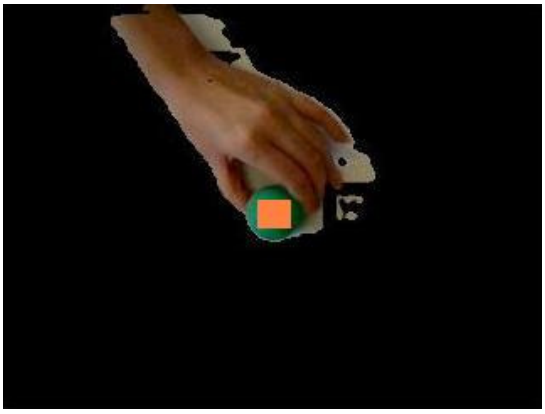
The smallest upright bounding rectangle of each pixel set is then calculated. The centre coordinate of the upright bounding rectangle that represents the position of the object is sent to the server.



(a) Background Frame

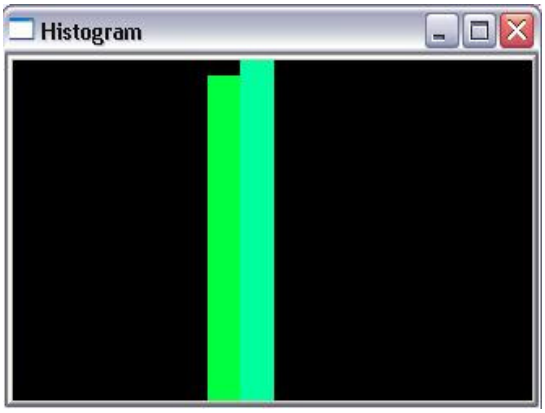
(b) Image after background subtraction shows only moving objects

Figure 3: Background subtraction

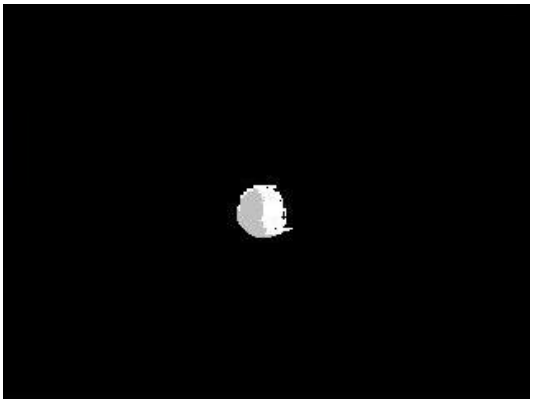


(a) Selecting the colour to be tracked

(b) Image containing only the object



(c) Hue histogram for the object



(d) Back projection image

Figure 4: Colour detection

3.4 Dog Behaviour and interaction

For a user to be engaged and enjoy playing with the ARDog, it will need to behave and react in a way that is comparable to a real dog.

Interaction with the dog using real, physical objects should provoke interesting responses. There should be a small sense of unpredictability so that the user can experiment with various interactions and see how it affects the dog.

To achieve this, we developed a simple state based system, where the dog can be in various states based on its current action. It also has several variables covering the dog's wants or needs so that it can make a decision when given multiple choices.

Currently the dog's hunger and boredom are modelled. Over time the dog will become hungrier and more bored. We developed a small number of behaviours for the dog to act out when certain criteria are met.

If not presented with any object, the dog will be idle. While idle, the dog will stay in one point, wagging its tail.

When presented with the ball, the dog will chase the ball as shown in Figure 5 and become idle once it reaches it. In a typical situation however the user will be rolling the ball, catching it and then rolling it again. If the dog is at maximum hunger, it will no longer chase the ball as it has a lack of energy. While chasing the ball, the dog's boredom will decrease quickly.

Figure 5: Shiba chasing the ball

If presented with food (represented by a bone in our demo), the dog will move to the food and begin eating it as long as it is hungry enough (currently 20% hunger in our demo) as shown in Figure 6. Food takes the highest priority for the dog, so it will eat the food even if it is very bored. While eating, the dog's hunger will decrease quickly.

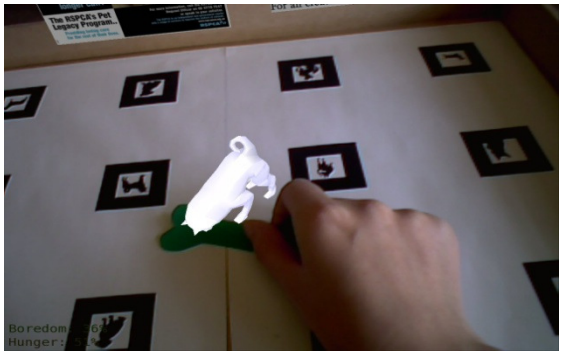


Figure 6: Shiba eating the bone

4 Results

We conducted an informal evaluation of our ARDog by inviting some postgraduate and undergraduate students to play with it and asked for their feedback. We received very promising feedback. Many described ARDog as "cool". We received very good reviews and responses from them.

It was interesting to see how natural it was for a small crowd of people to gather without much effort on our part.

This shows that ARDog is compelling enough to attract people's attention, and our virtual dog successfully exhibits convincing dog characteristics. Some had commented that the dog could be furthered improved, especially with more texture to add realism and to be distinct and easily recognisable.

5 Future Work

In spite of the good feedbacks that we received from fellow postgraduate and undergraduate students, we think that there are still many improvements that can be done to our ARDog.

5.1 Formal evaluation

The first thing that can be done is that ARDog needs some formal and proper evaluations. There are two key questions that need to be answered:

- *On interaction techniques*: How much realism and how usable are the computer vision based interactions compared to the tangible user interface based or other kind of interaction?

- *On interactive advertisement*: How effective is ARDog as an interactive advertisement? Is it more appealing or more frustrating than the 'normal' advertisement technologies and techniques that bombard us today?

5.2 The dog

Although we are generally satisfied with how Shiba turned out, there are still improvements that can be made. From the feedbacks that we received from students, we think that Shiba's apperance could be furthered improved by having its body textured. At the moment, the only part of his body that is textured is his face. Shiba could also have more actions to perform with associated animations. More varied sound effects could also help.

If we had more time, we would also revisit osgCal2 to reduce the problem of having potentially high IVE file size for our 3D assets, to blend animations and to control the skeleton in real time. This would allow us to have smarter behaviours like having Shiba's head to look into the direction of the camera.

5.3 Occlusion

In our prototype demo we used a simple wooden enclosure to encapsulate the interaction space. When viewing the area from certain angles, the dog and advertisements would be drawn over these boundaries. Because we know the dimensions of the structure, it should be fairly simple to construct an equivalent 3D model to use for occlusion. This would increase the level of realism and immersion for users.

There are other occlusion problems, such as drawing over the physical ball and users' hands. However these are more difficult problems to solve. Although we track the position of the ball, network lag would hinder attempts to model occlusion for it. It may still be worth doing when the ball is stationary. Occlusion for user's hands is even more difficult. In any of these cases, modern AR technology such as cameras that find depth information could be used also.

5.4 Shadow

To further unite the ARDog with the real world, all virtual objects contained within ARDog application should cast a shadow onto the ground surface containing the markers. To do this, a ground plane would need to be drawn over the real ground surface. This would again bring up the problem of occlusion since it would cover the ball and user's hands. These problems would need to be solved for this to be effective.

The shadow can either be implemented as part of the model in 3D Studio Max or generated in real time directly in OpenSceneGraph using osgShadow.

5.5 Multiple Clients

One of our initial reasons for the client/server architecture for the ARDog was that multiple users could interact and play with the ARDog collaboratively. We decided to keep things simple and just handle one client connection to the server. For multiple users a unified view of the dog's current state, position and action would be needed. Currently all of this information is handled by the client, meaning that if multiple clients were to connect, they would see different dogs and would not be collaborating. This would require extra information, such as the dog position and state, to be sent across the network; however object positions would no longer be sent so network traffic would not increase considerably.

5.6 More Behaviour and Interactions

Currently, the ARDog only has several interactions as well as several behaviours. Because of this, the level of engagement with a user is limited and they will become bored with it fairly quickly. Some other properties the dog could have, such as tiredness or bladder, could be associated with sleeping or going to the toilet.

One interaction we were planning on implementing at first was to allow a user to actually "pet" the dog with their hand or finger. This is perhaps possible using our current vision techniques, but some modifications would be necessary to make it robust. The major problem is that every person's skin colour is different and the system would need to be initialised each time.

6 Conclusion

Although the concept of virtual agents has been somewhat thoroughly researched, their effectiveness has not been thoroughly explored. We present ARDog, an Augmented Reality application which could be use for advertising.

We believe that we have successfully created an interactive advertisement application with a compelling animated virtual avatar, which manages to engage audiences. This is done particularly by the use of computer vision interaction techniques which greatly increases the sense of immersion and engagement with users.

Finally, we have also discussed ways in which ARDog can be further improved. One of the most important tasks left to do is to conduct a formal evaluation on the effectiveness of the computer vision as a means to interact with AR objects or AR virtual agents, and also to evaluate the effectiveness of AR in interactive advertising. This would give us a more solid understanding on how to effectively interact with AR virtual characters, how we perceive AR virtual characters, and how AR virtual characters can be used to augment the effectiveness of our daily tasks.

We conclude that ARDog fulfils our minimum aims: that it has to be compelling and the message that ARDog is a kind of advertisement seems to come across as users know the context they are in while interacting with the virtual dog.

References

[1] Wagner, D., Billinghurst, M., and Schmalstieg, D. 2006. How real should virtual characters be?. In Proceedings of the 2006 ACM SIGCHI international Conference on Advances in Computer Entertainment Technology (Hollywood, California, June 14 - 16, 2006). ACE '06, vol. 266. ACM, New York, NY, 57.

[2] Maes, P. 1995. Artificial life meets entertainment: lifelike autonomous agents. Commun. ACM 38, 11 (Nov. 1995), 108-114.

[3] P. Maes, T. Darrell, B. Blumberg, A. Pentland, "The ALIVE system: full-body interaction with autonomous agents," ca, p. 11, Computer Animation 1995.

[4] Barakonyi, T. Psik, and D. Schmalstieg. 2004. “Agents That Talk And Hit Back: Animated Agents in Augmented Reality,” Proceedings of the 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality, IEEE Computer Society.

[5] I. Barakonyi, and D. Schmalstieg. 2006. “Ubiquitous animated agents for augmented reality,” Proceedings of the 5th IEEE/ACM International Symposium on Mixed and Augmented Reality, pp. 145-154.

[6] Storring, M., Moeslund, T. B., Liu, Y., & Granum, E. (2004) “Computer vision-based gesture recognition for an augmented reality interface.” IASTED International Conference on Visualization, Image, and Image Processing, pp. 766-771.

[7] Lee, M., & Woo, W. (2003) “ARKB: 3D vision-based Augmented Reality Keyboard.” International Conference on Artificial Reality and Telexisitence (ICAT03), pp. 54-57.

[8] “Autodesk - 2D and 3D Design Software for the Architecture, Engineering, Construction, Manufacturing, and Entertainment Industries,” October, 2008; http://usa.autodesk.com/.

[9] “Cal3D - 3d character animation library,” October, 2008; https://gna.org/projects/cal3d/.

[10] “osgCal2 - Adapting cal3d to OpenSceneGraph,” October, 2008; http://osgcal.sourceforge.net/.

[11] “ReplicantBody - a character animation toolkit,” October, 2008; http://www.vrlab.umu.se/research/replicantbody/.

[12] “OSGExp - An Open Source Exporter from 3ds max to OpenSceneGraph,” October, 2008; http://osgmaxexp.wiki.sourceforge.net/.

Figure 7: Shiba, the Augmented Reality Dog