Uncertainty Boundaries for Complex Objects in Augmented Reality

Jiajian Chen, Blair MacIntyre

School of Interactive Computing

Georgia Institute of Technology

ABSTRACT

Registration errors between the physical world and computer-generated objects are a central problem in Augmented Reality (AR) systems. Some existing AR systems have demonstrated how to dynamically estimate registration errors based on estimates of spatial errors in the system. Using these error estimates, these systems also demonstrated a number of ways of ameliorating the effects of registration error. One central part of this previous work was the creation and use of error regions around objects; unfortunately, the analytic methods used only created accurate regions for simple convex objects. In this paper, we present a simple and stable algorithm for generating the uncertainty regions for complex objects, including non-convex objects and objects with interior holes. We demonstrate how our approach can be used to create a set of more accurate error-based highlights in the presence of registration error, and also be used as a general highlighting mechanism.

KEYWORDS: AR, registration error, non-convex objects, spatial uncertainty.

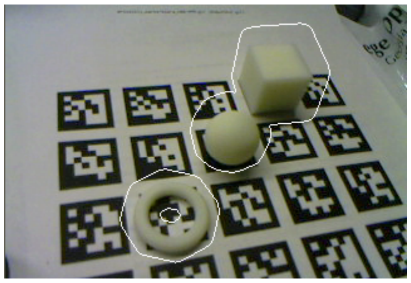
INDEX TERMS: I.3.3 [Computer Graphics]: Picture/Image Generation - Viewing Algorithms; I.3.6 [Computer Graphics]: Methodology and Techniques - Interaction Techniques.

1 INTRODUCTION

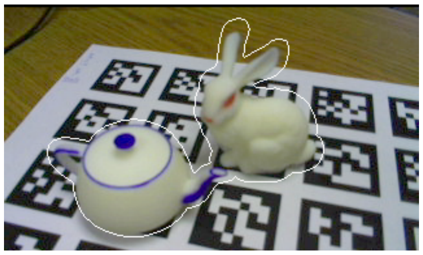
All Augmented Reality (AR) applications rely on spatial information about the physical world. Due to the spatial uncertainty of sensors and the latency between sensing and rendering, registration errors are inevitable in most practical AR systems. In some existing systems like OSGAR [3] and AIBAS [5], while registration errors can cause problems for users, both with respect to understanding the intent of augmentations and with respect to user interaction [2], if the system has a model of the sensor errors it is possible to estimate and thus ameliorate the impact of the inevitable registration error. A lot of work has been done to deal with registration errors, rather than just eliminate them; others have worked on mitigating the impact of registration error without explicitly computed error estimates. For example, [4] uses multi-modal interfaces to help users locate objects in AR, but it heavily relies on finding the selection gesture accurately. [6] uses a statistical approach based on a worst case estimate of registration error.

Given an estimate of registration error, a variety of approaches could be taken to help users understand the impact of the error. Perhaps the most obvious approach is to modify the visual presentation to directly convey the spatial uncertainty. For example, an appealing method to highlight the location of an object would be to draw an expanded version of the boundary of the object such that the object will fall within the region, given the registration error estimate, as shown in Figure 1. This is one approach that works in OSGAR [3]. OSGAR works correctly on simple convex objects, such as a cube or a cylinder, but yields an unsatisfying approximation on non-convex objects.

{johnchen, blair}@cc.gatech.edu



(a) Simple and complex shapes: a cube, a ball and a torus.



(b) Correct boundaries for complex objects: a teapot and a bunny.

Figure 1. The boundary algorithm for non-convex objects.

In this paper, we present an alternative image-based method to establish a more accurate boundary for non-convex objects and show how these more accurate boundaries can be used to create a collection of interesting visual effects for highlighting objects in AR. While the algorithm is simple, it allows us to demonstrate a collection of highlight techniques (shown in Figure 1 and 3), and would be a useful addition to an AR visualization toolbox even without considering the impact of registration error.

2 AN ACCURATE BOUNDARY FINDING METHOD

The goal of this algorithm is to establish a more precise boundary for a non-convex object in AR. As with most AR applications, we assume that we have a sufficiently accurate model of the objects to be augmented. For the example images in the paper, we ensured an accurate correspondence by reversing the typical process: we started with a set of object models, and printed physical versions on a 3D printer. These include simple objects (cube, sphere), a simple non-convex object (torus) and two complex objects (the Stanford Bunny and the teapot).

We track the objects by placing them at known locations on a sheet of fiducial markers with ARToolkitPlus. However, any technique for tracking the camera would also work. The algorithm is designed to integrate with a system such as OSGAR, and assumes the existence of a 2D registration error estimate represented as an ellipse in screen space. In these examples, we use a fixed sized error ellipse for simplicity.

2.1 Boundary Finding Algorithm

We use the following methods to find the exact boundary of this real object from a video frame:

(0) Obtain the pose of the physical object to be augmented.
(1) Set up a render target. Clear the background to black and drawing color to white;
(2) Disable lighting and render the virtual objects to the frame buffer as solid white silhouettes;
(3) Read the render target back into memory;
(4) Dilate the boundary; image dilation with a structure element (e.g. a disk) is a standard operation in image morphology.
(5) Find the boundary edges and produce a "mask image"
(6) Modify the current frame with the dilated image and produce various visual effects.

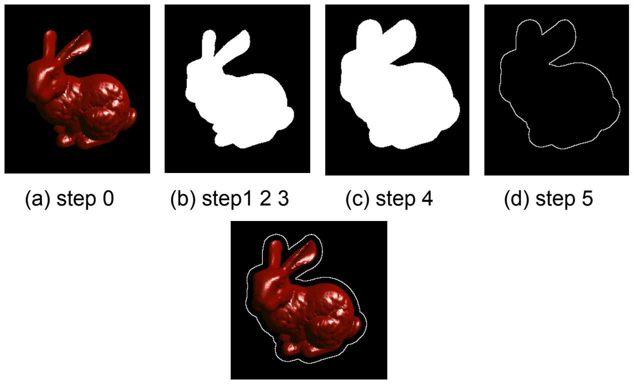


Figure 2. Find the boundary for complex objects

Figure 2 illustrates this procedure to find the dilated boundary of a bunny and combine them together. This method is simple and stable and it can deal with multiple complex objects in the AR scene. Multiple objects can be grouped or treated individually. Most importantly, the radius of the disk can be chosen to present the uncertainty of each real object (the bigger the radius is, the bigger the registration error estimate is).

We implemented a GPU version and a non-GPU version for this algorithm. The GPU version renders objects to texture and skips step 3 and combines step 4 and 5 as a single pass in the fragment shader. In the non-GPU version, we use the frame buffer to render the intermediate objects, reading the images back using *glReadPixels*. We will discuss their performance below.

2.2 Displaying Uncertainty with Different Effects

Given the tighter fitting silhouettes and outlines generated by this algorithm, it is straightforward to produce a variety of interesting visual highlights around the real objects in an AR system. The edge boundary itself, shown in Figure 1, is one such highlight.

Figure 3 demonstrates two other highlighting techniques. In the left two pictures the physical objects have been highlighted with different colors. The right one shows a dimming effect achieved by dimming everything outside the region, intended to simulate a spotlight falling on the objects. Notice the inner circle of the torus in the right one is correctly dilated.

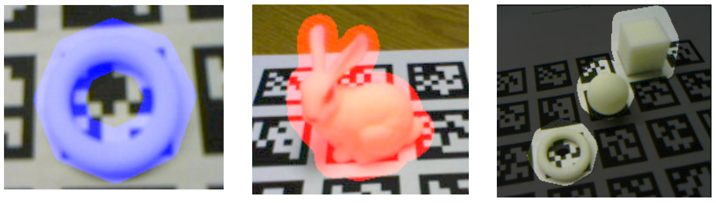


Figure 3. Examples of two highlighting techniques

3 Architecture for Real Time AR

The first issue is the speed. The GPU version draws the silhouette in two passes: render the objects to texture and map this texture on the screen quad. A fragment shader is used when we do texture mapping in the second pass. This shader dilates and draws the object silhouette in the GPU directly so it eliminates expensive data copy between the frame buffer and memory. It can easily run at 60+fps on a laptop with an ATI X1300 card. However, if we want to apply this technique in handheld devices which do not support shaders, we need the non-GPU version. This version achieved 20~30 fps on the same laptop. The second issue is how to properly integrate it into an AR system. The GPU version is suitable to integrate and will not introduce any delay. In the non-GPU version *glReadPixels* becomes a bottleneck for real time performance. In this case, an alternative approach would be to use smaller render targets for each object, drawing the objects at the correct orientation and scale such that they are centered in and fill their viewport. Then, the highlight can be rendered using texture mapping until the pose of the object changes enough to require re-rendering, and the texture hardware can take care of scaling the highlight to the appropriate size.

4 Conclusion

In this paper, we have leveraged a simple approach to finding a dilated 2D screen-based silhouette to demonstrate a variety of highlighting techniques that would be useful in the presence of registration error. The algorithm works with complex, non-convex objects, and only assumes an estimate of registration error is available in the form of a 2D ellipse in screen-space. Regardless of the specific algorithm, these highlight techniques would be a useful addition to an AR visualization toolbox. The visual aesthetic of having the highlight graphics loosely enclose the object without touching it is appealing, possibly lessening the potential for the graphics to negatively impact the performance of a worker using AR.

References

[1] B. Bell, S. Feiner, and T. Hollerer. View management for virtual and augmented reality. In *Proc of 14th UIST*, pages 101-110, 2001.
[2] E. M. Coelho, B. MacIntyre, S. Julier. Supporting interaction in augmented reality in the presence of uncertain spatial knowledge. In *Proc of the 18th UIST*, pages 111-114, 2005.
[3] E. M. Coelho, B. MacIntyre, and S. Julier. osgAR: A scenegraph with uncertain transformations. In *ISMAR 2004*, pages 6–15, 2004.
[4] E. Kaiser, A. Olwal, D. McGee, et al. Mutual dissambiguation of 3d multimodal interaction in augmented and virtual reality. *In ICMI 2003*, pp 12–19, 2003.
[5] C. Robertson, B. MacIntyre. Adapting to Registration Error in an Intent-Based Augmentation System. In *Virtual and Augmented Reality Applications in Manufacturing*, pages 143-163, 2003.
[6] A. Olwal, H. Benko, and S. Feiner. Senseshapes. Using statistical geometry for object selection in a multimodal augmented reality system. In *ISMAR 2003*, pages 300–301, 2003.