Dynamic Shader Lamps : Painting on Movable Objects

Deepak Bandyopadhyay [1]   Ramesh Raskar [2]   Henry Fuchs [1]

[1] University of North Carolina at Chapel Hill   [2] Mitsubishi Electric Research Lab

Abstract

We present a Dynamic Spatially Augmented Reality *system for augmenting movable 3D objects in an indoor environment using multiple projectors. We describe a real-time system for applying virtual paint and textures to real objects simply by direct physical manipulation of the object and a "paint brush" stylus. We track the objects and the "paintbrush", and illuminate the objects with images that remain registered as they move, to create the illusion of material properties. The system is simple to use and we hope it may herald new applications in diverse fields such as visualization, tele-immersion, art and architecture. The system currently works with tracked objects whose geometry was pre-acquired and models created manually, but it is possible to extend it, by adding cameras to the environment, to acquire object geometry automatically and use vision-based tracking for the object and paintbrush.*

Keywords: Applications, 3D Painting, Spatially Augmented Reality, Projector, Tracking, Human-Computer Interface

Additional Keywords: Augmented Reality, Dynamic, Virtual Reality

1   Introduction

Augmented Reality techniques aim to supplement the user's view of the real world with virtual objects. Spatially augmented reality [27] is a paradigm in which these virtual objects are rendered directly in the user's physical space, on real-world objects, using projectors or flat-panel displays. This allows multiple independent viewers to see an augmented version of their surroundings in stereo without the need for head tracking or head-mounted displays.

1.1   Shader Lamps

In previous work by some of the authors[28], we introduced **Shader Lamps**, a special case of Spatially Augmented Reality where multiple projectors are used to render a virtual object that has the same shape as the physical object used as a display surface. The approach is to "lift" the lighting and material properties of the real object into the model being projected, so that we can replace a physical object illuminated by white light with a neutrally colored object with projected illumination. As long as the final illumination that reaches the eye is the same, this rearrangement of terms in the optical path works and the augmented object looks quite realistic.

We described techniques to solve the problems we faced, including intensity correction for surfaces that are oblique to the projector, a feathering technique to blend the overlapping images of projectors on a static object, and easy projector calibration. This technique can be used to graphically animate the object, illuminate it with virtual lights, or change its material properties so it appears to be made of some other material.

1.2   Painting on Movable Objects

We describe in this paper an extension of the idea of Shader Lamps to movable objects. These objects are allowed to move in arbitrary

Figure 1: An enthusiastic young user (Miriam Mintzer Fuchs, age 9 years) demonstrates our easy to use system for 3D painting on movable physical objects. Note that the color palette on the table and the color on the spherical tip of the "paint brush" are projected.

ways, retaining their current state of shading as they do so. A 3D painting interface allows interactive shading of the objects. The user of this system can draw something on the real object or apply a texture to it, while moving it around and rotating it freely.

Technologically, our system comprises a hand-held tracker used as a paintbrush providing user input, and two projectors as display devices for projecting color patterns onto the faces of the object. Furthermore, we let the object move, and the projected color patterns stay registered on its faces and move with it.

1.3   Applications

Our painting interface for real objects could be used to modify other properties such as the object's illumination from virtual lights, or to interactively animate it. We are hopeful that this capability will be useful for a variety of applications. For example, in wide-area immersive videoconferencing[26], the paint system could be used to modify real objects and replicate the modification on the remote site as a means of communication. There may be applications in cosmetics and fashion where different colors may be projected on skin or clothes while the subject moves around, and modified interactively. In the creation of architectural models, our techniques would allow interactive annotation, rearrangement and visualization of building structures with different materials. Finally, we hope that it introduces a new artistic mode of virtual painting and augmentation of real objects, with applications in showroom and exhibition floors, art galleries, museums and maybe in future, in the home. For further details, see Section 7.

The rest of this document is organized as follows. Section 2 details the previous work and our contributions in the areas of 3D painting and AR visualization. Sections 3 and 4 deal with our initial proof-of-concept implementation, the issues that arose and the

decisions made during its development. Section 5 describes the 3D painting subsystem and the user interface. Section 6 summarizes the initial reactions of users of the paint system. Section 7 discusses possible applications of our system in diverse fields such as visualization, art, architecture, medicine and cosmetics. Section 8 enumerates some of the issues that remain to be addressed and how we are planning to deal with them in the short to medium term, and section 9 concludes with a summary and an indication of the challenges and future possibilities for research in this area.

2   Related Work

3D paint programs allow a user to modify (paint) directly onto the model rather than in texture space. Hanrahan and Haeberli[11] painted onto parameterized 3D meshes using a mouse interface, using an item buffer to pick the polygon and point of modification. This approach was extended to 3D input devices by Agrawala et al[1], who built a system that allowed painting on scanned models of real objects, moving a tracked brush around the real object to control the painting and provide haptic feedback.

The important differences between [1] and our work are that in our system the object is dynamic and we paint directly onto the real object at the same time as the model, using projectors to display directly on the movable physical object. This results in an easier to use interface where the user's attention is focused on the object rather than divided between the real object and the model.

Among rendering systems for realizing spatially augmented display on real objects, our system is based on Shader Lamps [28], already described in Section 1.1. Our system demonstrates one of the simplest applications of this technique. There is some other recent work in the area of using real objects in the environment as a luminous-tangible display and user interface[13, 33, 32].

Our work also draws heavily from the 3D paint engine of the inTouch haptic painting system [9]; the method used here is a scan conversion of each triangle to be painted, in texture space, so that any arbitrary brush function (see Section 5.2) can be simulated.

Recently, there has been work on CavePainting [14], where the emphasis is on using natural interactions with an immersive display surface (a CAVE^{TM}[5]) to create three-dimensional paintings. Our system, like theirs, uses a projected palette interface for painting (a similar interface first appeared in [6]). However, we paint on real objects, as opposed to a virtual environment with no object physically there. Our system also yields a better sense of immersion than the CAVE, with a smaller setup cost[28]. However, we are limited by a cumbersome acquisition process any time the object to be painted is to be changed, occlusion, user-induced shadows and an inability to change scale. Refer to the limitations in Section 8.

2.1   Contributions

Our main contribution is a set of techniques for interactively changing the *appearance properties* of a neutrally colored physical object, which may be static or movable. We discuss our implementation of a prototype system with two projectors (display devices) and a tracked object and paintbrush (3D input devices), and discuss how it would scale for more display devices, more objects or vision-based tracking.

3   Geometric Configuration

3.1   Choice of Objects

While we would like to be able to project on any arbitrary object, and indeed theoretically we can, the results would not always be visually compelling. The ideal object must be diffuse or matte (not
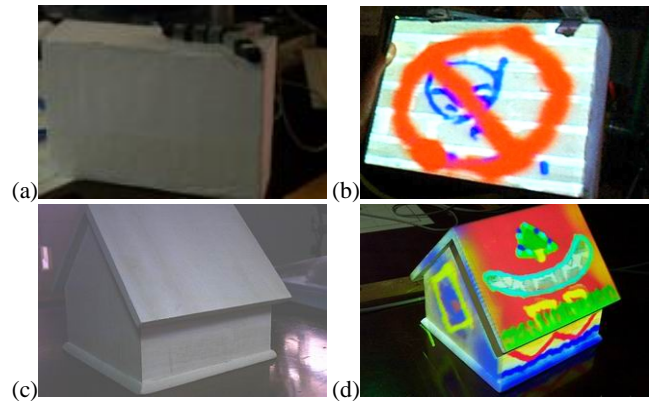

Figure 2: Augmented movable objects. a) White cuboid with optical tracker attached b) illuminated with interactively painted images c) a birdhouse with magnetic tracker mounted d) illuminated with projected imagery painted by the user.

specular) so that we may project any arbitrary illumination onto it. It must be *neutral* (not dark or colored) so that we can get a wider range of colors by fully exploiting the limited dynamic range of the projector. It should be lightweight (for easy manipulation) and readily available. The proof of concept can be shown even with an object of small polygon count, though the system will work with medium-sized polygon count objects as well.

We chose a cuboid (made from a white cardboard box) as it satisfies all the above properties and is sufficiently simple to model as a first object without sacrificing any of the power of the underlying techniques. So for example we can move it around arbitrarily and it looks unique at each orientation; and we can paint on its faces, or around the corners and edges so that the colors are shared between faces. The next object that we chose is a wooden bird house. Both objects are shown in white light as well as projected on with interactively painted images in Figure 2.

3.2   Tracking and Coordinates

We use an optical tracker (the Flashpoint^{TM} 5000 from Image Guided Technologies Inc., Denver, Colorado) as well as a Fastrak magnetic tracker from Polhemus Inc. In each case, we attach a sensor onto the object being tracked, and use an optical or magnetic stylus as the paintbrush.

A simplification we make is that the world coordinate frame is the same as the tracker coordinate frame, whereas the object coordinates are specified in the coordinate frame of the sensor that is rigidly attached to the object. This allows us to read the matrix that gives the sensor position and orientation in tracker coordinates, multiply it with each point on the object (a constant in object coordinates), and directly get the world coordinates at which the point is to be rendered.

3.3   Projector Setup and Calibration

Two projectors are mounted in the ceiling facing each other, as shown schematically in Figure 3. The projectors tilt downwards and their field of view and zoom are set to cover the tracking volume. They have a large overlap within the tracking volume to ensure that all faces of the object are illuminated by at least one projector while it is being tracked.

Each projector needs to be calibrated so that it can draw correctly in world coordinates. For a static world, finding the internal
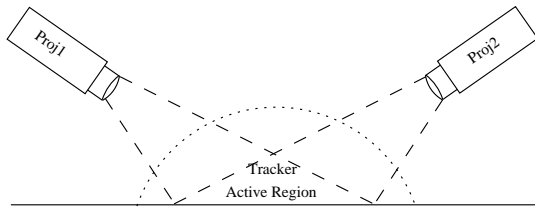
Figure 3: Schematic diagram of projector setup. Two projectors face each other and projecting downwards, cover the volume where tracking is active (shown as a hemisphere).

parameters of the projector and the rigid transformation between the coordinate system of the world and the projector is enough. Raskar et al in [28] solve this problem. They move a projected cross-hair in each projector's image space, to coincide with known fiducials in the world or on the object, and note down the 2D position corresponding to each fiducial. Then the 2D points along with the 3D coordinates of the fiducials are input to a Matlab program that solves for the 4x4 perspective projection transformation (treating the projector as dual of a camera as in [24]) and decomposes it into intrinsic and extrinsic parameter matrices. The rendering program can then use these matrices to render the model so that it is registered on the physical model.

In our calibration process, instead of using fiducials we use the tracker stylus to measure 3D positions in the world corresponding to projected 2D points. We add a stipulation that the calibration points span the overlap of the projection and tracking volumes, and be roughly uniformly distributed across this intersection volume. This ensures that the calibration is good no matter where and in which orientation we place the object.

Figure 4: Projector calibration. The two sets of cross hairs give 2D projector coordinates. The corresponding 3D point is measured with a stylus (tracker).

3.4 Registration

A projector has good registration when we can use it to draw accurately in world coordinates and on the tracked object. Static registration is improved by improving the calibration, modeling the object more accurately and more accurate tracking. For static or slow moving objects, we observe good registration in regions where the calibration sample points were taken.

The dynamic registration problem in our system is for graphics to real world registration when the object is moving, and it arises due to latency in the system. Dynamic registration error causes the static or moving user to perceive shearing in the rendered object, and this hampers their sense of presence. We do not face the viewpoint-to-graphics registration problem as currently we do not render specularities and other view-dependent effects.

4 Modeling and Rendering

4.1 Modeling

The paint system assumes that the model of the real object to project on has been pre-acquired, by scanning or point sampling, surface reconstruction and texture coordinate assignment (parameterization). The model representation used is essentially the same as in Shader Lamps [28]. To acquire the vertices of this model, we use a tracked stylus to measure points on the object, and transform them to object coordinate space. For creating the edge list, we use a surface reconstructor[22] to create the mesh connectivity.

The texture coordinates are stored per vertex, along with the model. Right now texture coordinates are assigned manually, but for any very complex example they will have to be automatically generated. One way to do this is to use a cylinder or sphere mapping and blow out all triangles from a central axis or point to the surface of a cylinder or sphere respectively, and then map coordinates on the cylinder or sphere onto a rectangular map. Special care has to be taken that no triangles share the same region of a texture map, or else painting in one triangle will cause inadvertent and absurd changes in the texture of another. This constraint can be incorporated into a texture coordinate optimization stage [29, 19] that follows the generation stage.

4.2 Lighting and Material Properties

Intensity Correction: As detailed in [28], the shaded model has to be processed per-pixel to increase the intensity to compensate for the fact the it is rendered on an oblique surface and the visible light intensity is not the same as the rendered intensity, but is less by a factor of the cosine of the angle the surface normal makes with the projector's z axis. Since normals are defined per-face (we calculate them at the beginning), and the scene changes dynamically, it is not feasible to use the method specified in [28], rendering the object as white and diffuse and reading back the intensity values. So instead we do the following - transform each normal into homogeneous coordinates, and then divide the intensity at each vertex of the triangle by the transformed z coordinate (cosine of the angle of falloff).

Material Properties: We have chosen textures for our objects from the world of everyday construction materials (brick, various marbles, wood, stone). These together with other material properties (diffuse and specular colors and reflectance coefficients, specular exponent) are used to initially render the object.

4.3 Rendering

The shaded model is currently rendered on two Sony projectors driven by two channels of one graphics pipe of an SGI InfinityReality2 engine[23]. The code is designed to be fully portable to a PC with a multiple monitor card, or individual PCs communicating over a network. The projectors are set up as shown in Figure 3, to best cover the tracking volume and also minimize the overlap surface area, so that wherever the object is moved within the region, in most orientations it gets full coverage on most of its surfaces, with double intensity (overlap) artifacts on as few surfaces as possible.

On one graphics pipe we are able to achieve nearly 60 fps of rendering speed, with one or more tracker updates being read per frame time. The latency of 4-5 frames (optical tracker) and about 2 frames (magnetic tracker) between the tracker and the projector

display is a significant factor leading to break in presence when the object or the brush is moved at moderate to high speeds.

Since the user is not head-tracked in our prototype system, there is no view-dependent (specular) lighting. Our diffuse shading looks correct from all user viewpoints and allows multiple simultaneous users, following the paradigm of spatially augmented reality[27, 25]. View dependent effects should be easy to add for a single tracked user, as described in previous work[26].

5 Interactive 3D Painting

Once the objects have been illuminated with shader lamps, the next step is to provide an interface that lets the user modify the attributes of the shader lamps (and hence of the object) in real time. This is an interaction process similar to that of painting the model surface, so we implemented it as a 3D painting application for spatially augmented reality. Our system can paint color and texture directly onto the object.

We use a painting scheme based on the inTouch system[9] We maintain one or more texture maps for the model, with texture coordinates assigned to all the model vertices. We now describe the steps in the painting process.

5.1 Contact Determination by Proximity Detection

We compute an axis-aligned bounding box for the model and then check the position of the brush head, transformed into model coordinates, against this bounding box. If it is outside, we stop further testing; this gives us an early rejection test when looking for triangles to paint. When the check succeeds, we find the perpendicular distance between the brush head center and the plane of every triangle within the bounding box. For all triangles within an *admittance threshold radius* (a brush parameter), we make sure the perpendicular dropped from the brush center falls within the triangle or upto an *outside triangle margin* (normally a constant times the brush radius) outside its edges. This helps to make sure that the blob of paint is not clipped to the triangle within which it lies but stretches across to neighboring triangles. We collect a list of triangles that have met both criteria for sending to the next stage.

Note that this process could be improved vastly with an oriented bounding box for the model, or some other form of space subdivision, most effectively hierarchical subdivision as in an AABB or OBB-tree or an octree[8, 7, 21, 10]. When the brush is spherical, the proximity test (between a point and a polygonal surface) is best handled as a sequential test within a region culled using bounding boxes. However for a more realistic brush geometry a proximity package such as PQP [16] may be used.

5.2 Brush Function

The brush function (B), used in blending in the brush color with the color of a point, provides the factor α to be used for the blend as a function of 3D point position, given the position of the brush. We assume a spherical geometry for the brush head and a corresponding spherical distribution of the function around the brush center c, with the simple formula for brush function at point X:

$$B(X) = 1 - (r/R)^2 \qquad (1)$$

where r is the distance from c to X, and R is the constant "brush radius".

B is always between 0 and 1 for $r < R$. Thus for this brush function to work properly, the brush admittance threshold must be less than or equal to the brush radius. When it is nearly equal, a very fine film of paint is deposited on triangles at the fringe of the

threshold. This when coupled with a larger than normal brush radius leads to what we call the *airbrush* or *spray can* mode. In *contact painting* mode, we use a smaller brush radius, with a threshold set to the radius of a physical sphere attached to the brush head, and a spherical brush function centered at the contact point rather than the center point of the sphere. *Texture painting* is similar to contact painting, but it's brush function is thresholded. If r/R is less than the threshold, B is 1, else it is 0. Thus instead of blending we get superimposition.

5.3 Painting as Triangle Rasterization

One by one, the triangles chosen to paint are scan converted or rasterized using a special routine taken from [9]. The routine steps through 2D points on the triangle's texture map and corresponding 3D points interpolated from the 3D positions of its vertices. We have to be specially careful that no texel on the border of two or more triangles is rasterized twice, or it will receive double the regular dose of color for any brush position, leading to discontinuities in paint strokes. To fix it, we set a 'painted bit' when these texels are first modified and then ignore any write to a painted texel in the same rasterization pass. An improved rasterizer would avoid this problem altogether.

5.4 Texture Map Modification

The brush function evaluated at the 3D point (X) is used to calculate the new value in the texture map at its corresponding 2D position, (x). The formula used is a simple alpha blending:

$$Color(x) = Color(x)*(1-B(X))+BrushColor*B(X) \quad (2)$$

Here B is the brush function and BrushColor is the current color of the brush.

5.5 User Interface

We demonstrate in our accompanying video a simple physical user interface where the object to be painted is hand-held and movable so that the user can expose its various surfaces and paint on them. The paint-brush stylus is similarly hand-held, and we attach a sphere to its tip so that it appears lit up in the current brush color, and the notion of brush function becomes intuitive. We choose the brush radius to be the actual sphere radius, plus a tolerance of 2-6 mm to compensate for inexact alignment of the stylus tip with the sphere center. Thus the contact paint works just that way - on contact, the paint threshold is breached and painting occurs. Airbrush or spray painting also works the same way but from a distance. When the stylus has a button on it (eg. the magnetic stylus), we use the button press as a trigger for spray painting; this is again similar to spray painting in the real world.

Additionally, a portion of the table is converted into another physical UI object, the *palette*. Our simple palette is a projected rectangle with a texture map that shows the available colors as well as painting options (analogous to the tool tray used by Fitzmaurice in his GraspDraw application, [6]). We treat the palette just like a normal object, but a collision of the brush on specific parts of the palette (corresponding to what is drawn there) produces different outputs, which can be mapped to actions such as color changes, paint mode changes and other such events. Our user interface remains the same, regardless of the tracking technology in use and the object being painted.

6 Initial User Reactions

Since the preliminary proof of concept implementation of our system has just been completed, systematic user data is not yet avail-
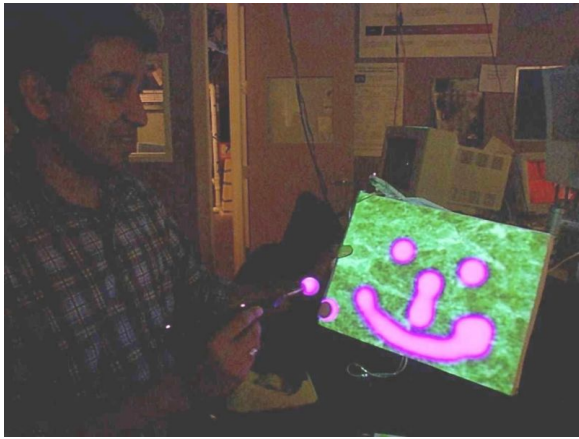
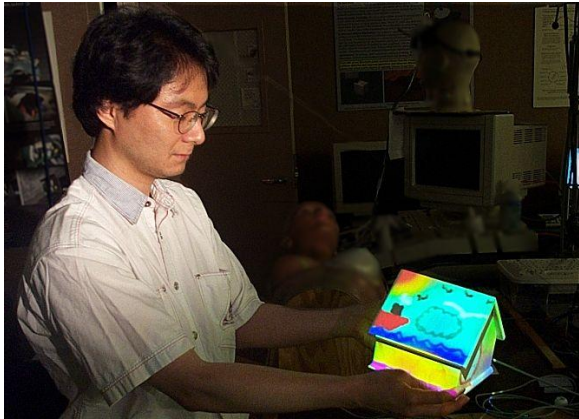Figure 5: User demonstrating a painted box, tracked optically.

Figure 6: User holding a painted bird house, tracked magnetically.

able. The working system was demonstrated to visiting graphics researchers and students, a film crew, and the 9-year old daughter of one of the authors. Figures 1, 5 and 6 show users holding objects and painting on them.

The system seems to appeal to users because of its premise of the creation of art on real objects in an augmented setting. Furthermore, the degree of immersion is quite high for the 9-year old; she paints and casually rotates the object while painting, seemingly oblivious that the illumination is projected, and without noticing any loss in registration while doing so.

To the question of how this compared with other painting programs - 2D image based, 3D model based or fully immersive (virtual), the response was that this is certainly different from those applications and has a whole new range of applications encompassing those visualized for shader lamps. One user reinforced our feeling that the feedback of real paint-based and digital artists would be valuable. Advanced technical users loved the artistic capabilities, but also pointed out that limitations of the tracking technology and its latency bounds were the significant performance bottleneck as well as the break in presence for the system. In this regard, the magnetically-tracked system was better, since within its (smaller) working volume it worked in all orientations, unlike the optical tracker, and had appreciably lower latency.

7 Applications

We present here some of our thoughts on a few applications where it would make a difference.

Tele-immersion. Imagine a wide-area networked videoconferencing setup such as the Office of the Future [26], with multiple projectors and cameras at each end doing scene acquisition and rendering in real time. Painting on objects in the workspace and passing them around provides a tangible (physical, real-world) means of interaction among the local users, that can be replicated at the remote site using our techniques to project on an identical object moved independently, or else using another stereo display technology. This could be a significant addition to the interaction techniques currently available in such systems.

Tangible User Interface Objects. A simple variation on the painting interface can make the object to be painted behave like a palette, which can be carried around and used to select a function to perform. Our techniques can transform any ordinary object into a physical user interface object [13, 33, 32] for interacting with Augmented Reality.

Architecture. We can use our system to visualize how a structure will look with different materials and lighting conditions, with the added benefit that the structures could be moved around (if one were planning a city). The painting interface could be used for placement of materials, or for architectural *annotation*, i.e. drawing markings which stay in place as the structure is moved around, and provide some useful information. Since all these changes are virtual, reversible and undoable, our system scores over traditional annotation methods. However, not being able to scale a physical model is a limitation for this application.

Art. Our 3D painting system, along with a new interaction style, also induces a new "artistic style" for Augmented Reality the way [14] does for immersive VR - to beautify the surfaces of everyday objects or make them appear to be made of some other material. Though our implementation does not yet offer the full suite of tools an artist would need or try to simulate physically-based paint like [4] does, our experience is that people love to tinker with the tools available and they can create aesthetically pleasing designs. Our framework is open and extensible to incorporate advanced painting and paint simulation, as well as combination with sophisticated vision-based tracking methods [15, 18] and a modeling package for deformable objects to produce, for instance, a spatially augmented simulation of sculpting with clay[20].

Medical and Cosmetic. Introducing Spatially Augmented Reality Painting instead of the HMD based systems currently used in medical AR research [3, 30] might be a radical idea right now, but if we can get tracking of a deformable object (the human body) to work accurately, we can at least use it for annotation. In cosmetics, fashion and allied industries, it can be useful to visualize how a variety of products (such as make-up) would look on the body, or how different colors and patterns would look on clothing, without actually trying them on. The objects to be painted on tend to behave rigidly with respect to the body frame, or they can be tracked using motion capture technology.

Theatre. A straightforward extension of the previous application is to have "intelligent spotlights" on performing artists, that would light up in color a part of an artist's body throughout the performance. The "painting" interface in this application could be modified so that the director could aim a laser pointer to indicate the next target to light up.

8 Limitations and Future Work

There are a number of not well-solved problems with the current system and the whole paradigm of painting on movable objects. We list the significant ones here:

- **Tracker Latency:** The *latency* problem will not go away soon but can only be minimized. The display and the projector each add one frame of latency. The optical tracker gets bottlenecked in the serial port driver and has a large latency of 5-6 frames, where one frame is $1/60^{th}$ second. The magnetic tracker's own latency is supposed to be well under a frame[12], but we observe that the network (between the tracker client and server) and buffering of the readings contribute around 2 frames.

  Rather than go back to the case of not moving the object (static shader lamps[28]) or moving it in a slow, restricted manner (as in bolting it to a turntable or a mechanical arm) to reduce latency, the preferred solution is to fine-tune the tracking code and add prediction [2] to it to offset most of the effects of latency.

- **Tracker Range:** The Flashpoint optical tracker's sensors can track only in a restricted set of orientations, when all two or three LEDs on the sensor are visible to the infra-red detector. This limitation can be addressed by building a custom sensor so that always at least three LEDs are visible; three visible LEDs can be used to track the object with six degrees of freedom. Also, there has to be a line of sight (LOS) from LEDs to detector, which is lost if the user accidentally occludes an LED, resulting in loss of tracking. Of course, magnetic tracking avoids such problems altogether, though its range is limited to 76cm[12]. When the magnetic receiver is within range of the transmitter, signal to noise ratio (SNR) of the tracker reading is above a threshold; this guarantees the specified positional accuracy of 0.08cm.

- **Geometric Issues:** The cumbersome process (manual or semi-automatic) of acquiring a model for an object before using it with our system can be avoided by using cameras and structured light to acquire scene geometry in real time as described in [26]. Cameras can also be used for vision based tracking of the object, especially since its geometry is known[15, 18].

  Since by definition our system requires the presence of a real physical object, it does not support **scaling** or multi-resolution painting, a limitation when compared to other painting systems for virtual reality.

- **Projection:** Dynamic blending in real-time in regions where multiple projectors overlap on the object is a problem not yet solved. The static case was solved in [28] by intensity feathering. However, in our system with the surfaces moving around it is hard to solve for these weights in real time unless one can drastically minimize the area of overlap by partitioning the set of surfaces between the projectors in some way. Thus in our prototype implementation we ignore overlap, and it shows up as double intensity in some regions. Double intensity coupled with non-ideal static registration contributes to imperfect overlap between the projected images, when both projectors are contributing to a surface at a large oblique angle.

  Another related issue is of **limited dynamic range** of the projectors. The augmented version of the object looks best when there is very little ambient light; ambient white light tends to wash out the colors and materials due to superposition and interference. Due to the technology of current projector devices there is very little that can be done to resolve this in the immediate future. It turns out that in a lot of our application domains, controlled lighting is a given. Also, in an environment where only the projectors provide lighting, this problem does not occur if we are able to solve the dynamic blending problem.

Finally, there are traditional projector limitations such as limited depth of field, differences in color levels[17], non-uniform intensity within the projected region.

- **Scalability:** In order to extend the *range* of operation of the system to cover the large room-sized environments we visualize it will be used in, we need more projectors and higher tracking range. For more projectors, the current solution scales readily (note though that the blending problem that we haven't solved is made more complex by the need to scale to n projectors). Projector placement to cover the space efficiently has been investigated [31]. Extending the tracking range is trickier, as is setting up the projectors and tracker to get the highest possible volume of intersection between the projection volume and the tracking volume.

- **Occlusion and Shadows** The complete illumination of non-trivial physical objects using projectors is difficult for some objects which are self-shadowing. For locally concave objects, our techniques will still work if we add more projectors to fill out the occluded region. Shadows induced by the user on the object by occluding the projected light are more difficult to compensate for. One possibility with multiple projectors is to track the user's approximate position and then turn off some areas of projected illumination that would cast a shadow, switching to alternate projectors to display the same illumination. This is not a major problem in the current system as humans are conditioned to working without occluding the sources of light in their environment, and easily find a position from where they can paint without significant occlusion.

- **Extensions:** The term 'dynamic' as used in our implementation refers to "movable rigid" objects, whereas in its true sense it could refer to deformable objects or objects that were not earlier modeled but are introduced into the environment. Each of these can be supported by extracting the object geometry in real time.

9 Conclusions

We have presented a new system for 3D spatially-augmented painting and projection on real objects. It requires that there be two projectors facing each other, a tracker attached to the object (and in the paint brush), the projectors both be calibrated to the tracker's frame of reference, the object be reasonably dull-colored and diffuse, and that the object's geometry and texture coordinates be pre-acquired.

Some of the major issues that we dealt with in the design of our system are listed below:

- **Suitability** of any real object for being projected and painted on.
- **Tracking and Updates** of the real object's position and orientation as the user moves it around.
- **Setup and Calibration** of the projectors to display in world coordinates.
- **Modeling** of the real object into 3D geometry and texture coordinates.
- **Display and Registration** of the projections onto the object.
- **Lighting and Material** properties of the object, and how to modify them with image-based illumination
- **Rendering**, the actual projection step from two projectors, updated simultaneously for all-around viewing of the object.
- **3D Painting** with brush functions (see section 5.2) and mapping from 3D points to texture coordinates.

- **User Interface** design to keep the interaction with the system simple, so that anyone can use it without a learning period.

The system was tested with a cuboidal object, which is good enough to test the registration of vertices and edges in the real object and its projected version, as well as a moderately complex polygonal object. The same framework will work with an arbitrary number of overlapping projectors and an arbitrarily complex polygonal object. The performance for high polygon count models can be accelerated using hierarchical collision detection techniques to pick the surfaces to paint. Some of the issues that remain to be solved are dynamic blending for projector overlap, increasing tracker range, improving the dynamic range of the display in bright light conditions, reducing or compensating for latency further to increase the sense of presence, and developing compelling applications. The system represents a technological milestone in the creation of an AR painting space, and commercial applications are needed to drive its further development.

Further details and high-resolution color images are available at the URL: http://www.cs.unc.edu/~debug/papers/DSLpaint

References

[1] Maneesh Agrawala, Andrew C. Beers, and Marc Levoy. 3d painting on scanned surfaces. In *1995 Symposium on Interactive 3D Graphics*, pages 145–150, April 1995.

[2] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-Through HMD. In *SIGGRAPH 94*, pages 197–204, July 1994.

[3] Michael Bajura, Ryutarou Ohbuchi, and Henry Fuchs. Merging virtual objects with the real world : Seeing ultrasound imagery within the patient. In *SIGGRAPH '92 Conference Proceedings*, volume 26, pages 203–210, July 1992.

[4] Bill Baxter, Vincent Scheib, Ming Lin, and Dinesh Manocha. Dab : Interactive haptic painting with 3D virtual brushes. In *SIGGRAPH 2001 Conference Proceedings*, August 2001.

[5] Carolina Cruz-Neira, Daniel J. Sandin, and Thomas A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 135–142, August 1993.

[6] George W. Fitzmaurice. *Graspable User Interfaces*. PhD thesis, University of Toronto, 1996.

[7] S. Gottschalk. *Collision Queries Using Oriented Bounding Boxes*. PhD thesis, Department of Computer Science, University of N. Carolina, Chapel Hill, 2000.

[8] S. Gottschalk, M. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. In *SIGGRAPH '96 Conference Proceedings*, pages 171–180, 1996.

[9] Arthur Gregory, Stephen Ehmann, and Ming C. Lin. intouch : Interactive multiresolution modeling and 3d painting with a haptic interface. In *IEEE Virtual Reality 2000 Conference Proceedings*, March 2000.

[10] Arthur Gregory, Ming C. Lin, Stefan Gottschalk, and Russell Taylor. A framework for fast and accurate collision detection for haptic interaction. In *IEEE Virtual Reality 1998 Conference Proceedings*, 1998.

[11] Pat Hanrahan and Paul Haeberli. Direct wysiwyg painting and texturing on 3D shapes. In *Computer Graphics, Proceedings of SIGGRAPH 90*, volume 24, pages 303–310, August 1990.

[12] Polhemus Inc. 3space fastrak user's manual.

[13] Hiroshi Ishii and Brygg Ullmer. Tangible bits: Towards seamless interfaces between people, bits and atoms. In *CHI*, pages 234–241, 1997.

[14] Daniel F. Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H. Laidlaw, and Jr. Joseph J. LaViola. Cavepainting : A fully immersive 3d artistic medium and interactive experience. In *2001 Symposium on Interactive 3D Graphics*, pages 85–93, March 2001.

[15] D. Koller, G. Klinker, E. Rose, D. Breen, R. Whitaker, and M. Tuceryan. Real-time Vision-Based camera tracking for augmented reality applications. In Daniel Thalmann, editor, *ACM Symposium on Virtual Reality Software and Technology*, New York, NY, 1997. ACM Press.

[16] Eric Larsen, Stefan Gottschalk, Ming C. Lin, and Dinesh Manocha. Fast Proximity Queries with Swept Sphere Volumes. Technical Report TR-99-018, Department of Computer Science, University of N. Carolina, Chapel Hill, 1998.

[17] Aditi Majumder, Zhu He, Herman Towles, and Greg Welch. Color Calibration of Projectors for Large Tiled Displays. In *IEEE Visualization*, 2000.

[18] Richard Marks. Natural interfaces via real-time video. In *SIGGRAPH 2000 Conference Abstracts and Applications*, page 237, 2000.

[19] S. R. Marschner. *Inverse Rendering for Computer Graphics*. PhD thesis, Department of Computer Science, Cornell University, 1998.

[20] Kevin T. McDonnell, Hong Qin, and Robert A. Wlodarczyk. Virtual clay : A real-time sculpting system with haptic toolkits. In *2001 Symposium on Interactive 3D Graphics*, pages 179–190, March 2001.

[21] D. Meagher. Geometric modeling using octree encoding. In *Computer Graphics and Image Processing*, volume 19, pages 129–147, June 1982.

[22] Gopi Meenakshisundaram and Shankar Krishnan. Surface reconstruction using lower-dimensional delaunay triangulation. In *Eurographics 2000 Conference Proceedings*, June 2000.

[23] J. S. Montrym, D. R. Baum, D. L. Dignam, and C. J. Migdal. Infinitereality: A real-time graphics system. In *SIGGRAPH 97*, pages 293–302, August 1997.

[24] R. Raskar. *Projector Based 3D Graphics*. PhD thesis, Department of Computer Science, University of N. Carolina, Chapel Hill, 2000.

[25] R. Raskar, G. Welch, and W.-C. Chen. Table-top spatially augmented reality : Bringing physical models to life with projected imagery. In *Second International Workshop on Augmented Reality (IWAR'99)*, pages 11–20, November 1999.

[26] R. Raskar, G. Welch, M. Cutts, A. Lake, L. Stesin, and H. Fuchs. The office of the future: A unified approach to image-based modeling and spatially immersive displays. In *SIGGRAPH '98 Conference Proceedings*, pages 179–188, 1998.

[27] R. Raskar, G. Welch, and H. Fuchs. Spatially augmented reality. In *First IEEE Workshop on Augmented Reality (IWAR'98)*, pages 11–20, November 1998.

[28] Ramesh Raskar, Greg Welch, Kok-Lim Low, and Deepak Bandyopadhyay. Shader lamps : Animating real objects with image-based illumination. In *Eurographics Rendering Workshop 2001*, August 2001.

[29] P.-P. J. Sloan, Weinstein D. M., and J. D. Brederson. Importance driven texture coordinate optimization. In *Eurographics 1998 Conference Proceedings*, volume 17, June 1998.

[30] Andrei State, Mark A. Livingston, Gentaro Hirota, William F. Garrett, Mary C. Whitton, Henry Fuchs, and Etta D. Pisano (MD). Technologies for augmented reality : Realizing ultrasound needle-guided biopsies. In *SIGGRAPH '96 Conference Proceedings*, pages 439–446, August 1996.

[31] Wolfgang Stürzlinger. Imaging all visible surfaces. In *Graphics Interface'99 Conference Proceedings*, pages 115–122, June 1999.

[32] John Underkoffler and Hiroshi Ishii. Urp: A luminous-tangible workbench for urban planning and design. In *CHI*, pages 386–393, 1999.

[33] John Underkoffler, Brygg Ullmer, and Hiroshi Ishii. Emancipated pixels: Real-world graphics in the luminous room. In Alyn Rockwood, editor, *SIGGRAPH 1999, Computer Graphics Proceedings*, pages 385–392, Los Angeles, 1999. Addison Wesley Longman.
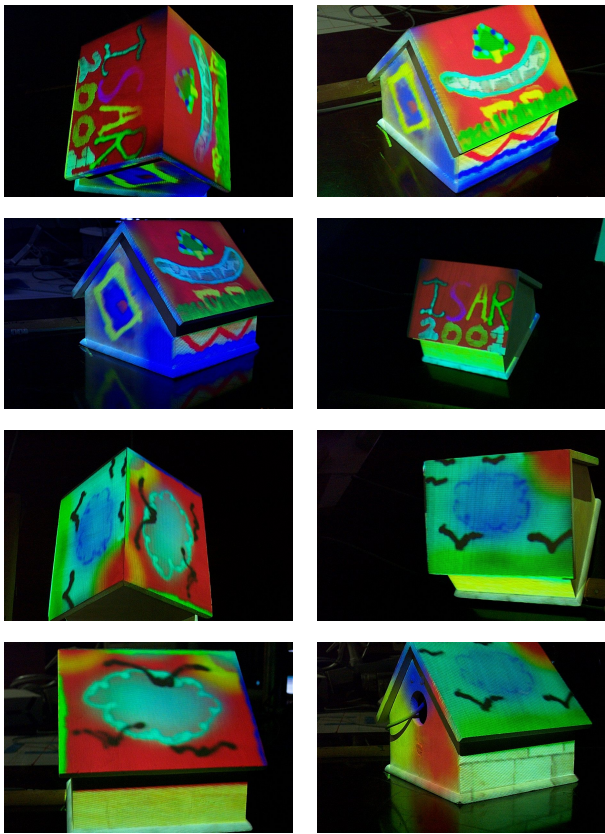
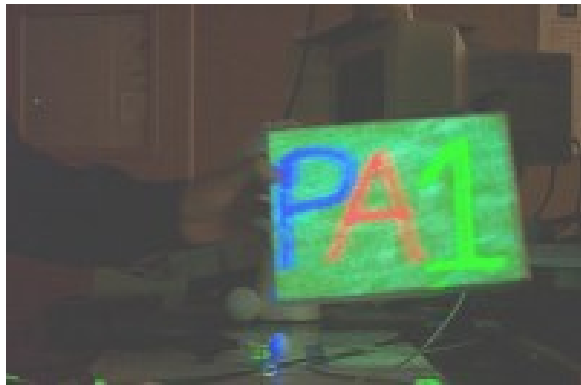Figure 7: Paintings created on the birdhouse using paint, air and texture brushes, seen from different orientations.

Figure 8: Demonstration of the latency in the tracking pipeline. Notice the blurring and trails artifacts.

Figure 9: User trial with Miriam Mintzer Fuchs, 9 years old, painting on a movable object.

Figure 10: User trial of Dynamic Shader Lamps with other users holding painted objects.