Efficient Image-Based Projective Mapping using the Master Texture Space Encoding

David Guinnip, David Rice, Christopher Jaynes
Dept. of Computer Science
University of Kentucky
Lexington, KY 40506
jaynes@netlab.uky.edu

Randall Stevens
ArchVision, Inc.
110 South Upper St.
Lexington, KY 40507
rstevens@archvision.com

ABSTRACT

We introduce an encoding technique that supports efficient view-dependent image-based rendering applications that combine photorealistic images with an underlying surface mesh. The representation increases rendering efficiency, reduces the space required to store large numbers of object views, and supports direct image-based editing for realistic object manipulation. The *Master Texture Space* encoding transforms an original set of exemplar images into a set of Master Textures that share a globally consistent set of texture coordinates based on underlying object geometry and independent of camera positions used to create the exemplar views.

An important property of the master texture space is that an arbitrary but fixed pixel position in all the Master Textures correspond to the same point on the object surface. This property increases rendering efficiency for real-time dynamic image-based rendering applications because new texture coordinates do not have to be loaded as a function of viewpoint. In addition, changes in a Master Texture image can be rapidly propagated to all views to add/remove features, generate new viewpoints, and remove artifacts to a pre-existing image-based scene. Results presented here demonstrate that the technique reduces real-time rendering rates by 1.6 milliseconds per frame for reasonably complex models on a commodity graphics card. Two example scenarios demonstrate how the Master Texture encoding supports efficient update of an image-based model.

Keywords

Image-based rendering, texture mapping, compression, image processing.

1. INTRODUCTION

Image-based rendering has emerged in recent years as an approach to rendering three-dimensional scenes [Lip80] without the representation of an overly complex geometric model. As opposed to pure geometric approaches, the image-based approach uses real-world images of an object or scene as the basis for rendered viewpoints to achieve photorealism.

The image-based approach maximizes realism (by using real-world imagery) while minimizing rendering time. Image-based rendering is particularly successful for complex objects that are cumbersome to represent in a purely model-based domain (e.g. Cars, Trees, and People). Today, image-based content is used in both still renderings, off line animations, and in real-time interactive applications [Arc02].

Journal of WSCG, Vol.11, No.1., ISSN 1213-6972
WSCG'2003, February 3-7, 2003, Plzen, Czech Republic.

There have been many techniques introduced for representing image-based content [Che93, Deb98, Gor96, Lav94, Lev96]. All of these techniques can be described as view dependent, in that images captured from known positions in the world are dynamically selected (or interpolated) based on the current rendering view. Each technique utilizes some geometric representation of the object to be rendered in order to incorporate the image-based model into a scene. Model complexity varies from a single plane to complex surface meshes, onto which images are statically or projectively mapped.

This continuum of full geometry versus complete image-based rendering offers users the ability to represent objects based upon application needs. Points in this continuum can yield different levels of detail depending upon the application requirements. A single polygon and 200 views may be a sufficient for a model of a person that is to be inserted into an office scene and rendered from a distance. On the other hand, a vehicle, rendered from nearby may exhibit significant perspective distortion that is not modeled in the available viewpoints. In this case, a simple object mesh can be combined with the available images to improve the rendered result.

This paper will focus on view-dependent projectively texture mapped (VDTM) objects with an underlying mesh of moderate complexity. We introduce an

encoding technique designed to support photorealistic image-based rendering, and view-dependent projective texturing of objects. The encoding process transforms an original set of object views into *Master Texture Space*, so that the new images share a globally consistent, single set of texture coordinates regardless of the camera positions used to create the initial images.

This global set of texture coordinates only needs to be stored once and is invariant with respect to the number of images used for the image-based model. For standard conditions the Master Texture encoding can achieve reasonable compression rates while increasing rendering efficiency.

The encoding process guarantees that Master Texture images are pixel-wise aligned to one another. This alignment property supports efficient image-based editing, straightforward view interpolation, and efficient updating of previously captured image-based scenes.

2. IMAGE-BASED APPROACHES AND RELATED WORK

At the heart of image-based rendering techniques is direct processing of image data to derive new views of an object or scene. Typically, a set of exemplar images is captured of a scene of object from many viewpoints. These images are, combined with known camera positions, are the basis for rendering new views either by directly placing the captured image data into the scene or first by interpolating between known views. Surface constraints, (e.g. that the model is piecewise planar), can be used to constrain the interpolation process. As opposed to a purley image-based approach, explicit use of a low-resolution model has been shown to improve rendering quality and extend the range of viewpoints available for novel views [Nis99, Deb96].

The Master Texture encoding is applicable to image-based rendering techniques that combine a geometric model with view dependent texture mapping (VDTM). Techniques described in [Deb96] employ VDTM to achieve photorealistic novel views from a sparse set of images by projecting interpolated pixel data directly onto low-resolution models. A distinct advantage of this approach is the ability to realistically integrate image-based models into computer-generated scenes. Even low-fidelity geometric representation supports model interaction within its environment (e.g. illumination, shadow casting). Combination of image-based views of an object with an underlying surface representation can be efficiently implemented as a projective texture map [Hec91], and is widely supported in rendering engines, real-time displays, and graphics hardware.

Image-based modeling and rendering techniques that do not exploit a geometric representation [Mat02, Che93, Lav94] use morphing maps of adjacent images to render a new images from viewpoints not contained in the original set of basis views. Interpolation requires accurate camera calibration to the object coordinate system, or known relative calibration between the exemplar views for constrained epipolar interpolation [Car01]. Although approaches that use partially reconstructed surfaces or range data to constrain the interpolation process [Deb98, Mcm97] have been introduced, interpolative methods still rely heavily on accurate point correspondences between images and are typically only robust given a dense sampling of base images.

Explicit estimation of the Plenoptic function has advantages similar to the view dependent texture mapping in that novel viewpoints can be reconstructed from a set of basis images without modeling the optic flow of pixels as they move from one camera to another [Mcm95]. Light-Field rendering [Lev96] and the Lumigraph [Gor96, Bue01] reduce the dimensionality of the plenoptic function while utilizing a sparse mesh representation. A drawback these techniques share is the dependence on a large number of images for accurate estimation. Furthermore, these approaches are not widely supported in hardware and do not support real-time image-based rendering.

Regardless of the particular tradeoff between exemplar views and geometry that each of these approaches make, rendering quality and resolution is limited by the number of basis images that can be stored and efficiently rendered. The large number of exemplar images, calibration information, and object mesh required for VDTM pose a significant challenge to efficient storage and rendering of photorealistic models. As a result, researchers have been addressing representation, manipulation, and efficient rendering of image-based data [Che02, Deb98].

In work perhaps most similar to our own, [Nis01] describes an encoding scheme that stores all the views of a particular model face in a single image texture. This representation has the advantage of being amenable to off-line compression methods and results show that the technique is able to achieve between 5:1 and 15:1 compression rates with little or no loss in image fidelity. However, manipulation of the representation is cumbersome and, more importantly, rendering of an object encoded using the Eigentexture [Nis01] do not support real-time applications because texture coordinates are computed at rendering time based on viewpoint.

Our approach is motivated by the observation that a single set of texture coordinates for any number of basis views is desirable for render-time compression and efficiency. The Master Texture Space preserves image fidelity contained in the exemplar views, achieves reasonable compression, and facilitates efficient rendering and manipulation of the encoded images.

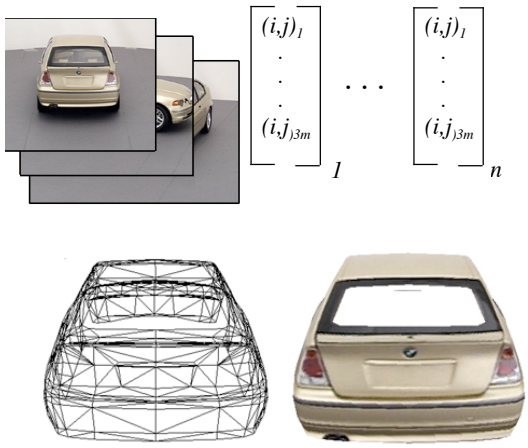

Figure 1: Traditional view-dependent projective texture mapping approach. Exemplar images of a real-world object from *n* viewpoints (top left) are dynamically combined with a mesh using a set of texture coordinates based on the current view.

3. MASTER TEXTURE GENERATION

Current approaches to projective texture mapping first select or create an appropriate image of the scene using the exemplar images based on the view of the object to be rendered. This image is then projectively mapped to the object mesh to support lighting effects, and induce depth effects (such as foreshortening) not present in the mesh alone [Deb96]. Because it is infeasible to compute texture coordinates in real-time using the available projection matrices, the mapping between object vertices and image coordinates must be precomputed and stored as a set of texture coordinates with each image. Given a mesh containing *m* triangular faces, and *n* exemplar images, between *3m*n* and *m*n* texture coordinates must be stored with the model. In total, then, the images, model, and texture coordinates are used at render time to facilitate a dynamically texture mapped, photorealistic model. Figure 1 depicts this general approach.

Although VDTM successfully combines photorealistic images with simple geometric models, the amount of data required to achieve a high fidelity, accurate representation of the object is significant.

The general approach to computing the Master Texture space encoding is to precompute the relationship between all exemplar images and the object mesh. Mesh elements are projected into all exemplar images to produce a corresponding set of image regions (facets) that are extracted, warped and stored in the Master Texture space.

When facets are moved from the original image space to the Master Texture space, a new set of texture coordinates are created under the constraint that all Master Textures share a single set of coordinates. Given a mesh containing *m* faces associated with *n* exemplar images, only *3m* coordinates are needed to completely represent the relationship between mesh faces and any number of example images.

The encoding process requires a significant amount of processing. It is important to note, however, that this is a one time encoding cost that is not recomputed at rendering time. As will be shown in Section 4, advantages of the Master Texture representation outweigh this one-time cost.

The Master Texture encoding guarantees that a pixel, $(i,j)_k$, in Master Texture image *k* corresponds to the same point on the surface as pixel *(i,j)* in all other Master Textures. This *pixelwise correspondence* is an important advantage of the Master Texture encoding both because it represents a significant compression over storing all texture assignments explicitly, and image-based operations such as interpolation can occur on the now pixel-aligned data directly in Master Texture space.

The Master Texture Encoding Algorithm

The first step in creating a set Master Texture images from a set of *n* different exemplar images is to project each of the *m* mesh triangles into each view to produce *n*m* image facets. For a single mesh triangle, *n* facets, comprised of 3 image coordinates, $(u,v)_{1,2,3}$, are generated for each of the *n* views using the known projection matrix corresponding to each view:

$$\sum_{n=1}^{n}\sum_{m=1}^{m}\sum_{c=1}^{3}\begin{bmatrix} u_{nmc}/s_{nmc} \\ v_{nmc}/s_{nmc} \\ s_{nmc} \end{bmatrix} = P_n \begin{bmatrix} x_{mc} \\ y_{mc} \\ z_{mc} \\ 1.0 \end{bmatrix}$$

(Equation 1)

For non-degenerate cases, a mesh triangle corresponds to a triangular region in each image. Pixel data within each triangular region is rasterized to produce a facet containing image information Next, the *n* different facets corresponding to a single mesh face *m*, are transformed to be of uniform size and shape.

All facets are warped to a right triangle of width *w* and height *h*, where *w* and *h* are the maximum independent width and height of any the *n* facets corresponding to a single mesh face. For a given *w* and *h*, an affine warp, given by the matrix ***C*** of Equation 2, is applied to each of the *n* facets:

$$A = \begin{bmatrix} x_0 & y_0 & z_0 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_0 & y_0 & z_0 \\ x_1 & y_1 & z_1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_2 & y_2 & z_2 \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ 0 \\ w \\ 0 \\ 0 \\ h \end{bmatrix}$$

$$A^{-1}B = C \qquad \textbf{(Equation 2)}$$

Where ***A*** is the set of initial image coordinates of the mesh triangle, ***B*** is a vector containing the target coordinates of the warped facet. Note that ***B*** is constructed so that the resulting facets are all aligned with the image axes. Figure 2 shows three facets from a single face *m*, as seen from different views of an object before warping (inset top row) and after they have been warped to Master Texture space.

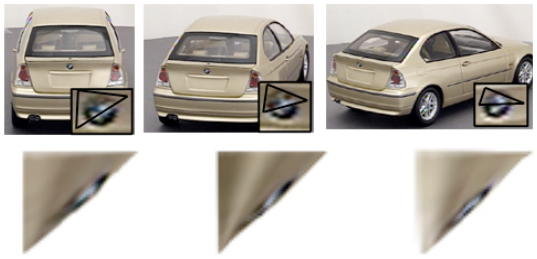

Figure 2: Example of facet extraction and warping. (top row) Three images of a real-world car object. A single mesh face corresponding (insert on each view, top row) as seen in each image. (bottom row) Facets are extracted and warped to uniform shape and size.

Once all facets corresponding to a particular mesh face have been warped, they are of uniform size and shape regardless of their initial projectively warped shape. This step ensures that corresponding facets in different Master Textures can be pixelwise aligned and that the facets are of maximum fidelity.

Ultimately, each exemplar view will be discarded and replaced by a corresponding Master Texture. Each Master Texture image is composed of *m* different facets corresponding to the *m* different mesh faces as seen from that view. The *m* different facets are placed into the Master Texture according to a packing algorithm that minimizes unused pixels in the Master Texture image.

An exemplar view is selected and the *m* facets, produced for that view are packed into a new image. The algorithm proceeds from the largest to the smallest facet, placing each into the Master Texture as efficiently as possible. Prior to packing, each facet is paired with a facet of similar area and reflected across the *x* and *y*-axes. A bounding box is fit to the result. Figure 3 shows two facets before and after the facets been oriented and fit to a bounding box. The resulting bounding region is packed into the new Master Texture image.

Figure 3: Example of facet combination and packing. (top row) Facets belonging to the same view are paired according to size. Facet at top right is rotated and combined with next largest available to fit both into a bounding box (bottom left). Result is inserted into a Master Texture image using the packing algorithm.

Bounding boxes are packed into the image at the leftmost and topmost position that the bounding box will fit. This process is repeated until all *m* facets corresponding to a single image have been placed into the Master Texture for that view. Figure 4 shows a Master Texture derived from a single view of the car example.

When the process terminates, a *packing map* that determines how all facets are placed into the Master Texture is stored. Because each facet in a Master

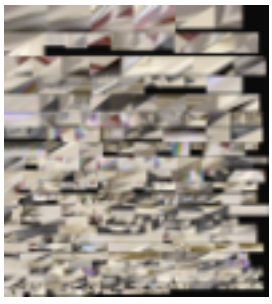

Figure 4: A master texture image computed for view 17 of the BMW model shown in Figures 1 and 2. Resulting image resolution is 438x686.

Texture corresponds to *n-1* other facets that are exactly the same size and shape, this same packing map can be used to efficiently generate the remaining *n-1* Master textures. These new images, the mesh, and a single set of texture coordinates are all that is

(a) (b) (c)

Figure 5: Results of the Master Texture encoding applied to the three different test objects. Each example shows an exemplar image (top left), object mesh (bottom left), example Master Texture , and the rendered view (at right). (a) Lamp example, n views, m polygons. (b) Helicopter example, 36 views and 436 polygons. (c) Vehicle example, 28 views and 820 polygons. For related videos the reader is encouraged to visit www.metaverselab.org/research/digital-objects/master/index.html.

needed for efficient, view-dependent rendering of the object using the available images. Because only redundant information was destroyed in the encoding process, renderings that use the Master Texture encoding have the same fidelity as those that make explicit use of the original image data.

4. RESULTS AND EXAMPLE APPLICATIONS

The master texture space-encoding algorithm was tested on several datasets to demonstrate compression rates and efficient use of the Master Texture space. We demonstrate the encoding on datasets of varying mesh complexity and varying numbers of exemplar images. The criteria for evaluating the approach are 1) efficiency of rendering a Master Textured object from changing viewpoints, 2) compression rate of the Master Texture encoding as compared to traditional texture mapping 3) efficiency in editing the image-based data while it is stored in the Master Texture representation.

Three datasets were used to understand the behavior of the Master Texture encoding (shown in Figure 5). Exemplar images for the Lamp and Helicopter objects were rendered using a very-high resolution object mesh with surface markings. Exemplar images of the third object, a BMW 325ti, were captured from the real word. These images were calibrated from a target placed in view of the exemplar images.

Encoding times are dependent on mesh resolution and number of exemplar images. Typically encoding times range from 8 minutes, for 436 polygons to 26 minutes for 8020 polygons on a Pentium IV 1.7Ghz machine.

Once encoded into Master Texture space, rendering efficiency is improved by removing the need for multiple sets of texture coordinates. The transfer time required to move new texture coordinates from system memory into the graphics pipeline must be taken into account when rendering complex image-based data that cannot all be stored on a graphics card. For a polygon mesh of 8020 polygons we measured this transfer time to be 1.6 milliseconds for each view on a Pentium IV 1.7Ghz machine with a 4x AGP bus and a GForce III graphics card. Using the Master Texture encoding, this fixed, per-view, overhead is eliminated.

Mesh Size	Views	Image Size	Master Texture Size	Compression Ratio
Lamp				
1842	90	640x480	466x932	1 : 2.2
Heli				
436	360	720x576	316x540	1 : 2.4
6,604	72	720x576	620x1214	1 : 1.76
7,259	36	720x576	820x1650	3 : 1
Car				
651	28	640x480	438x686	1 : 1.1

Table 1: Compression ratios for three different datasets of varying mesh size, exemplar image size, and number of views.

Compression rates each of the three different objects were measured and, in the case of the "Heli" dataset, mesh complexity was varied. Compression rates are measured as the ratio of total bytes required for traditional VDTM versus VTDM using the Master Texture encoding. Table 1 summarizes the results of the experiment.

Compression rates, particularly for complex meshes and many exemplar views, are not dramatic. This trend is demonstrated by the "Heli" model in Table 1. As mesh complexity and the number of views increases, mesh faces are likely to be seen as small (foreshortened) triangles in some views. These are resized to the maximum image triangle as seen in any view during the encoding process. This increases the resulting Master Texture size and decreases compression rates. This is unsurprising, in that the master texture encoding was developed to support image-based rendering scenarios. As the mesh size increases, the geometry begins to dominate the image data and the situation begins to approximate traditional geometric rendering. In cases were the mesh size is reasonable, the Master Texture encoding significantly decreases the amount of storage required for VDTM while increasing rendering efficiency.

Image-Based Editing

Once a set of images are captured under controlled conditions or rendered for an image-based rendering dataset, post-editing of the image data is a cumbersome process [Sei98]. For example, using a calibrated camera to capture multiple views of an indoor scene may be used to render photorealistic architectural walkthroughs. If the scene is to be modified (i.e. a billboard advertisement may be inserted), all images must be modified by back projecting the geometric position of the new object into all views using the camera calibration information. This requires $16*n*v$ multiplies for an object of v vertices, seen in n different exemplar views.

Alternatively, an image-based approach may be taken by painting the new billboard into each view appropriately. Editing of an exemplar view, however, requires that pixel correspondences are known for all exemplar views containing the new object. Discovering correspondences can be costly and, particularly in the case of real-world imagery, may be prone to error.

Because Master Texture images are pixel-wise aligned across all views regardless of the relative viewing geometry between the model and each view, changes to any one of the master textures can be easily propagated to all views. This is can be performed efficiently without the need to rediscover pixel correspondences.

We demonstrate this technique using two examples. In one example, changes made directly to an exemplar image are propagated to all views to support photorealistic rendering of the object containing the new data. In a second, more complex example, changes on the underlying geometry are first rendered to create a new image that is encoded using a known packing map to produce a new Master Texture image that can then be incorporated into the master texture space.

4.1.1 Direct Image-Based Editing of an existing Master Texture Encoded Space

A dataset was first created by capturing 28 controlled views of a car rotating in front of a calibrated digital camera. Exemplar views were captured at a resolution of 640x480 pixels. A corresponding geometric model containing 651 polygons was created by hand digitizing points on the object. The object mesh and an exemplar view are shown in Figure 5c).

Using the mesh, images, and corresponding camera calibration information, each exemplar view was

Figure 6: An image-based editing example. Original image and corresponding Master Texture shown at top left. Edited image and new Master texture are shown top column (a license plate has been added in this case). Pixelwise subtraction of the new Master Texture from the old reveals pixels that have changed (top right). These pixels are written directly into all Master Texture views to update the complete model that can then be viewed from new positions (bottom row).

encoded into a Master Texture space. This results in 28 Master Texture images. An example Master Texture image created from one exemplar is shown in Figure 6 (top left).

Exemplar view, *I*, directly behind the car was editied by inserting a licence plate. The resulting image, *I'* was encoded to produce a new Master Texture image, *M'*. Figure 6 (top row) shows the initial exemplar image *I*, its Master Texture encoding *M*, the edited exemplar *I',* and the new Master Texture, *M'*.

Since *I* and *I'* were both encoded using the same procedure, *M* and *M'* will also be identical except for the edited regions. Therefore subtraction of *M* and *M'* yields a set of pixels, Δ, in the Master Texture space that correspond to edited regions for all views. Figure 6 (top right) shows a difference image containing only pixels from the set Δ.

All views in the Master Texture space are updated through a pixel-wise addition of Δ. This operation requires no processing to determine (or search) for pixel correspondences and does not make explicit use of the camera calibration information (that has already been implicitly used in the creation of the Master Texture Space). Furthermore, pixelwise addition of image arrays can be easy implemented in hardware. The bottom row of Figure 6 shows the car rendered from three views using the modified Master Texture images.

4.1.2 Updating Master Texture from Changes in an Arbitrary View

A second example demonstrates how the Master Texture space supports efficient updating of images when a new view is introduced that does not correspond to an existing exemplar image (as in the previous example). Using techniques similar to the one shown here, relighting of the object mesh, changes to material properties on the surface, shadowing effects, and other object space updates, can be rapidly propagated into an existing Master Texture space.

Starting with the original dataset described in section *4.1.1*, a tree model and light source are placed into the virtual scene containing the car mesh. A shadow is then cast on matte shaded mesh surface. In a traditional image-based rendering scenario, update of all existing exemplar views with the new lighting effect requires significant computation.

In order to propagate the new effect into Master Texture space, a viewpoint that shows the desired effect (in this case the shadow) is selected. From that view, the shaded mesh is rendered both with and without the desired effect enabled, resulting in two new images *I* and *I'*, respectively. An image of the object mesh with the shadow effect enabled is shown in Figure 7 (top left).

Both images are then encoded into the master texture space using the same packing map P of the original

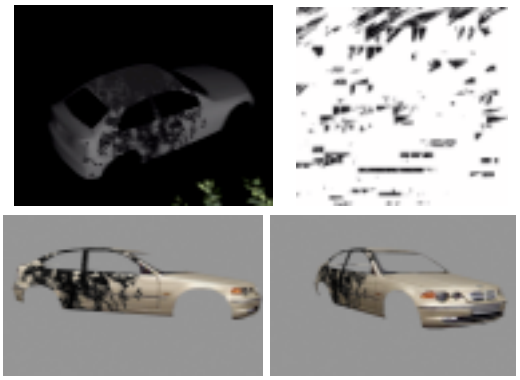

Figure 7: An image-based editing example. This example adds a shadow to the sequence of master images by projecting a computer-generated shadow onto the geometric model of the car.

dataset. As before, a set of difference pixels are derived by subtracting the two new Master Texture images *M* and *M'*. The difference image is shown Figure 5 (top right). These differences are propagated into all views through pixel wise addition across the *n* Master Texture views.

Once a new set of Master Textures has been computed, the new image-based model can be dynamically observed without having to recompute the effects of the shadow at render time. Figure 7 (bottom row) shows two views of the new model texture mapped with the modified Master Textures.

In this example, the shadow will appear consistent in scenes where the car and shadow are stationary. Figure 8 depicts the new image-based model placed into a scene containing the shadow-casting tree. Using image-based techniques for efficiency, static shadows for all objects in the scene are precomputed and texture mapped onto a planar surface (Figure 8a). Placement of the old image-based model into the scene results in car without shadow and a perceptually inconsistent scene (Figure 8b). The new image-based scene appears consistent without the need for expensive shadow casting operations to be computed at render time, and can now be viewed from any view in real-time (Figure 8c) (see www.metaverselab.org/research/mastertexture for related videos).

CONCLUSION

We have introduced the master texture encoding and have demonstrated how it can be used to improve compression and efficiency for view-dependent texture mapping applications. Examples, presented here, show that for sparse meshes and large numbers of images, the master texture encoding achieves over 50% compression for common image-based, view-dependent texture mapping scenarios.

(a) (b) (c)

Figure 8: Rapid relighting of a Master Texture dataset. (a) Scene containing precomputed shadows. (b) Car object added before relighting effect. Inconsistent shadow cues destroy realism. (c) Object added to scene after the master textures have been modified.

The pixel-wise alignment of the master texture images supports rapid modification of the master texture space that can be directly propagated into all views without explicit use of image calibration information. We are currently exploring how the master texture encoding can support entire scenes rather than a single object. We expect that a single set of globally consistent texture coordinates can be applied to site walkthroughs, for example

5. REFERENCES

[Arc02] ArchVision RPC technology. Real trees, real people. www.rpcnet.com

[Bue01] Buehler, C., Bosse, M., McMillan, L., Gortler, S., Cohen, M. Unstructured Lumigraph Rendering. In *Proceedings ACM SIGGRAPH 2001*, pp. 425-432 August 2001.

[Car01] Carpenter, C.S., Seales, B., Jaynes, C., and Stevens, R. Automated basis-view and match-point selection for the ArchVision RPC image-based model. In *Proc. of the Inter. Conf. on Multimedia*, September 2001.

[Che93] Chen, S. E. Williams, L. and View interpolation for image synthesis. In *Proceedings, ACM SIGGRAPH 1993*, pp. 279-288 July 1993.

[Che02] Chen, W-C., Bouguet, J-Y., Chu M., Grzeszczuk, R. Light Field Mapping: Efficient Representation and Hardware Rendereing of Surface Light Fields. In *Proceedings, ACM SIGGRAPH 2002*, pp. 447-456 2002.

[Deb96] Debevic, P., Taylor,C., and Malik, J. Modeling and rendering architecture from photographs: A hybrid-geometry and images-based approach. In *Proceedings, ACM SIGGRAPH 1996*, pp. 11-20, August 1996.

[Deb98] Debevic, P., Yizhou, Y.,and Borshukov, G. Efficient View-Dependent Image-Based Rendering with Projective Texture-Mapping. *Eurographics Rendering Workshop 1998*, pp. 105-116, June 1998.

[Gor96] Gortler, S., Greszczuk, R., Szeliski, R., and Cohen., M. The lumigraph. In *Proceedings*, *ACM SIGGRAPH 1996,* pp. 43-54, August 1996.

[Hec91] Heckbert, P. and Moreton, H. Interpolation for polygon texture mapping and shading. State of the art in Computer graphics: Visualization and Modeling, 1991.

[Lav94] Laveau, S. and Faugeras, O. 3-D scene representation as a collection of images. *In Proceedings of 12th Inter. Conf. on Pattern Recognition*, Volume 1, pp. 689-691, 1994.

[Lev96] Levoy, M. and Hanrahan, P. Light field rendering. In *Proceedings*, *ACM SIGGRAPH 1996*, pp. 31- 42, 1996.

[Lip80] Lippman, A., Movie-Maps: An Application of the Optical Videodisc to Computer Graphics. In *Proceedings*, *ACM SIGGRAPH 1980*.

[Mat02] Matusik, W., Pfister, H., Ngan, A., Beardsley, P., Ziegler, R., and McMillan, L. Image-Based 3D Photography using Opacity Hulls. In *Proceedings*, *ACM SIGGRAPH 2002*, pp.427-437, 2002.

[Mcm97] McMillan, L. An Image-based Approach to Three-Dinensional Computer Graphics. PhD thesis, U. of North Carolina, Chapel Hill, 1997.

[Mcm95] McMillan, L., and Bishop, G. Plenoptic Modeling: An image-based redering system. In *Proceedings*, *ACM SIGGRAPH* 1995.

[Nis99] Nishino, K., Sato, Y,, Ikeuchi, K. Appearance compression and synthesis based on 3d model for mixed reality. *ICCV '99*, pp. 38-45, 1999.

[Nis01] Nishino K., Sato Y. and Ikeuchi K., Eigen-Texture Method: Appearance Compression and Synthesis based on a 3D Model, in *IEEE PAMI*, 23:11, pp.1257-1265, Nov., 2001.

[Pul97] Pulli, K., Cohen, M., Duchamp T., Hoppe, H., Shapiro, L., and Stuetzle, W. View-based rendering: Visualizing real objects from scanned range and color data. In *Proc. of 8th Eurographics Workshop on Rendering*, St Etienne, France, pp. 22-34, June 1997.

[Sei98] Seitz, S. and Kutulakos, K. Plenoptic Image Editing. In *Proceedings of 5th ICCV,* pp. 17-24, 1998.