"Modeling Urban Environments using Texture and Displacement Maps"

Miguel Chover†, Roberto Vivó‡, Ricardo Quirós†, Xavier LLuch‡

† Departamento de Informática. Universidad Jaume I.
Campus de Penyeta Roja. 12071 Castellón. España
Tel: ++ 34 64 345771 Fax: ++ 34 64 344858
e-mail: chover@inf.uji.es

‡ Departamento de Sistemas Informáticos y Computación.
Universidad Politécnica.Camino de Vera s/n.
46071 Valencia. España

Abstract

Conventional modeling techniques usually require large sets of primitives to represent real-world surface features in detail. Therefore, we are interested in obtaining methods to simplify the design and representation of complex surfaces. In this paper we describe a representation model for urban environments that combines both texture mapping and displacement maps. Our main contributions are the introduction of the concept of urban environment, and the application of two-dimensional color, displacement, and material maps to this new concept. We also propose rendering techniques that adjust the level of detail of the scene according to distance and provide smooth transitions between bump rendering algorithms.

1. Introduction

The work we discuss in this paper specifies a model that simplifies the design and specification of features related to urban environments, like facades, sidewalks, etc. This model provides a fine level of detail representation for surfaces commonly encountered in such environments.

Designing a street using some CAD application typically requires modeling each of its buildings individually. Visual appearance can be improved by using texture maps to represent different material properties [Nuzz80]. The model can be made as complex as necessary in order to represent all the elements of a facade: balconies, windows, doors, ledges, etc. It is sufficient to define new surfaces using well-known modeling techniques to improve the level of detail of each facade. The model we propose, however, only requires surfaces to represent the facades itself, but not its surface detail. Instead we use texture and displacement maps to represent the detail that classically has be represented with additional surfaces. This way we can represent a street by three single surfaces: one for the ground, and two for the facades on each side. We thus avoid the problem of manipulating and rendering large numbers of surfaces just to represent a single facade or street. There are similar models for representing natural scenes, like the model by Gardner [Gard84].

We start the construction of a urban scene by placing a basic set of flat and cylindric surfaces in the model. We then apply color, displacement and material mappings to those basic surfaces. Basic surfaces are defined using parametric functions. This simplifies the process of texture mapping.

One of the advantages of our model is that it allows the specification of a geometric perturbation together with the definition of the parametric function of the surface. That way we avoid the construction of large complex polygon meshes, like those required by earlier displacement methods [Cook84] [Upst90]. This approach also eliminates the storage requirements associated to polygon approximations.

We render the images using displacement functions that adjust the level of detail of the scene according to distance and provide smooth transitions between different bump mapping algorithms. Our goal with this approach is to generate just the necessary detail, thus avoiding the complexity of displaying surfaces that are two far from the viewer.

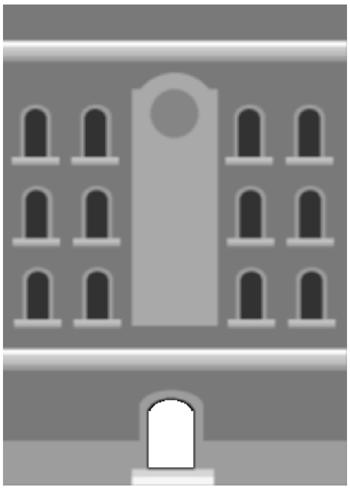
2.- Designing the Maps

After placing the set of basic surfaces into the scene, we design and choose different texture and displacement maps for each surface. We design the maps using basic two-dimensional primitives like circles, squares, ellipses, and more general curves. Then we apply different fill patterns to the drawings obtained using two-dimensional primitives. Fill patterns can be colored, diffused, hatched, and can be easily generated using a sequential process in which each pattern is used as a framework for the design of the next one.

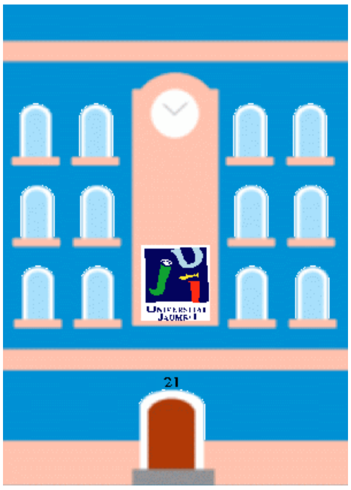
First, we design a displacement pattern that specifies the initial geometry of the perturbed surface. To do so, We use a color palete, where each entry is associated with a different depth level. For example, if we want to design a building, then we will assign to a window a color representing a depth in the displacement pattern larger than that of a ledge or a balcony. Note that here we are limited by the number of color indices in our palete, currently 256. The resulting image of the displacement map is the one we use as a basis to generate the color patterns (color plate 1). That image is stored in a two-dimensional array.

Our next step is the definition of a color pattern. We use the displacement map as a reference for the color pattern, so that we can use it to appropriately choose fill colors according to the geometric characteristics of the displaced surface. The color pattern is not limited by the number of colors of a particular palete, but it can rather be chosen from a wider selection of possibilities. For example, we offer all the possibilities of any standard 2D design application, like text, imported images, image composition, etc. The result is a color pattern that may even have blank areas to be filled later using a specific material's map. That color pattern is stored as an array of RGB values, as depicted in color plate 2.

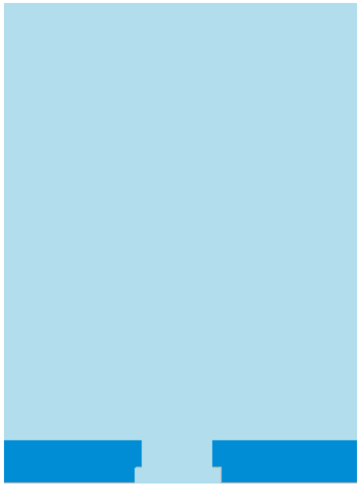
Finally, we use the material map to specify the materials to be assigned to the different parts of our original surface. We specify the material map using the color map as a reference. In this case we use a color palete whose entries are associated to different material types. To avoid distortion problems due to texturing over displaced surfaces, we define most of the materials using three-dimensional texture functions (color plate 3).



Color plate 1
Displacement map.



Color plate 2
Color map.



Color plate 3
Material map.

3. Computing the Displacements

Parametric surface modeling using depth maps is a technique that was developed in 1993 by Chover et al. [Chov93]. This technique belongs to a broader class of methods to displace and deform parametric surfaces [Chov96]. A displaced surface is defined using three functions of two parameters u and v, that take values in the domain [0,1]. To obtain all points on the surface it suffices to traverse the parametric space. We use the following equation to obtain the new position vectors of the displaced surface:

$$\mathbf{S}'(u,v) = \mathbf{S}(u,v) + F(u,v) \cdot \mathbf{N}(u,v) \qquad u,v \in [0..1]$$

where S(u,v) represents the original non-perturbed surface, F(u,v) represents the displacement map, and N(u,v) represents the direction of the normal vector to the original surface. This equation can be rewritten as:

$$x'(u,v) = x(u,v) + F(u,v) \cdot N_x(u,v)$$
$$y'(u,v) = y(u,v) + F(u,v) \cdot N_y(u,v)$$
$$z'(u,v) = z(u,v) + F(u,v) \cdot N_z(u,v)$$

Given this definition, we can obtain the tangent vectors to the surface in each parametric direction by taking derivatives with respect to both variables u and v. This approach, however, can be very complicated, since the displacement functions are likely to have discontinuities in their first derivatives. To avoid this problem we obtain the tangent vectors to the displaced surface by sampling the displacement function in a narrow interval around the point of interest. Again, sampling is done along both parametric directions u and v. We can thus express the tangent vectors to the surface using the following equations:

$$x'_u = \frac{(x'(u+\varepsilon,v) - x'(u-\varepsilon,v))}{2 \cdot \varepsilon} \qquad x'_v = \frac{(x'(u,v+\varepsilon) - x'(u,v-\varepsilon))}{2 \cdot \varepsilon}$$
$$y'_u = \frac{(y'(u+\varepsilon,v) - y'(u-\varepsilon,v))}{2 \cdot \varepsilon} \qquad y'_v = \frac{(y'(u,v+\varepsilon) - y'(u,v-\varepsilon))}{2 \cdot \varepsilon}$$
$$z'_u = \frac{(z'(u+\varepsilon,v) - z'(u-\varepsilon,v))}{2 \cdot \varepsilon} \qquad z'_v = \frac{(z'(u,v+\varepsilon) - z'(u,v-\varepsilon))}{2 \cdot \varepsilon}$$

Then we normalize the resulting vectors and determine the normal at any point on the surface by computing the cross product of the two vectors.

$$\mathbf{N}'(u,v) = \mathbf{S}'_u \times \mathbf{S}'_v \qquad u,v \in [0..1]$$

The advantage of this technique is that it allows us to redefine the geometry of a surface without recomputing its polygon representation. Furthermore, it also simplifies texture mapping, since you no longer need to adapt your texture maps to the new polygon representation.

4. Texture.

Given the newly defined geometry of a displace surface we apply the textures by using both the color and material maps selected by the designer for each surface. We apply color textures by superimposing the surface with the appropriate color pattern. Empty regions or regions with special features are left for the application of material patterns. These patterns consist of different three-dimensional texture functions that allow the representation of a broad collection of materials like marble, bricks, etc. and surface finishes like glass, stone, etc. To apply such texture functions we use the following algorithm:

```
Function Urban Texture
Input u,v,x,y,z,nx,ny,nz,
      color_pattern,material_pattern

     col=color_pattern[u,v]
     mat=material_pattern[u,v]
     If NOT empty(col)
          If NOT empty(mat)
               Get(mat,material_features)
          Else
               Get(DEFAULT_MATERIAL,material_features)
          Endif
```

```
        Else
             If NOT empty(mat)
                  Material(mat,x,y,z,col,nx,ny,nz)
                  Get(mat,material_features)
             Else
                  Material(DEFAULT_MATERIAL,x,y,z,col,nx,ny,nz)
                  Get(DEFAULT_MATERIAL,material_features)
             Endif
        Endif

   Output col,nx,ny,nz,material_features
   End Urban Texture
```

Here we use a different approach for each of the texture functions.

Texture Functions. We use two types of textures techniques: texture mapping, and three-dimensional solid textures. We apply texture mapping by superimposing a color pattern on the parametric surface, and solid textures by evaluating a three-dimensional function that depends on the contents of the entries to the material map.

Texture mapping. Given a color pattern we apply the associated texture by coloring the point of the surface according to its entry in the array representing the pattern. A problem associated to this technique is the distortion that results from applying a flat color pattern to a curved surface [Ma 88]. This problem is more evident on displaced surfaces, because the texture gets stretched and compressed when mapping directly from texture space onto the surface's parametric space. A possible solution to this problem is to use tree-dimensional projected textures [Peac85]. The idea is to project a flat texture onto three-dimensional space, so that the color of a point is obtained by evaluating a three-dimensional space function.

In our system, however, distortion is not a problem, since we define a color pattern in terms of the surface's displacement map. That way it suffices to specify the right area on the displacement map for the color pattern to be applied to right area of the displaced surface. For example, if we color a ledge differently, then the geometry of that portion of the surface corresponding to the ledge will be colored adequately as shown in figure 1. There may occur some distortion, but we do not observe it, if we assign the same color to the entire displaced portion of the base surface.

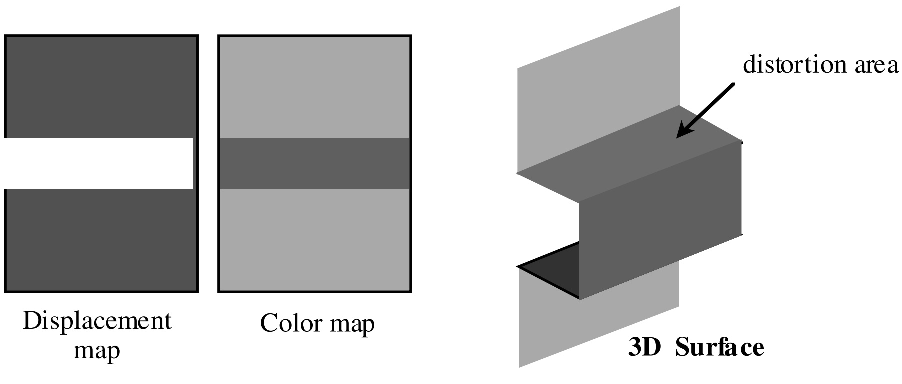


Figure 1 *Distortion of a color texture.*

Solid Textures. We apply solid textures by selecting appropriate three-dimensional functions according to the entries of the material maps. These functions are relevant, because they do not pose the distortion problems that superimposed patterns do.

Our system includes functions due to Perlin [Perl85] and Peachey [Peac85], plus our own set of material functions. The former ones allow us to represent wood, marble, etc. The latter ones allow us to represent materials like brick, stone, granite, etc., that are more common to urban environments. Brick textures, for instance, are obtained by dividing the space into blocks organized in a specific way (figure 2). Each block is the divided into an inner portion made of brick and an outer portion made of concrete.

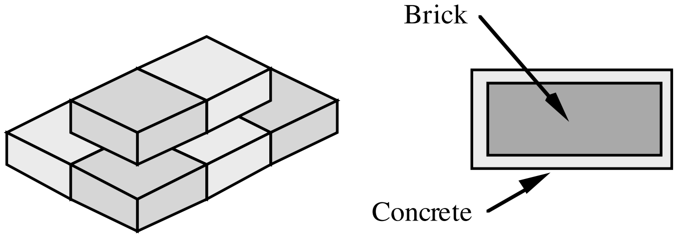


Figure 2. *A solid texture to represent bricks.*

To represent stone we use Perlin's noise function [Perl85], which models irregularities by perturbing the normal to the surface. Similarly, to represent granite we use the same function to model the color changes of the material.

5 Implementation

We described the basic characteristics of our system. In this section we explain how we use it to accurately render realistic urban environments.

We star with a description file of the basic surfaces of the scene. Given that description we construct for each surface a displacement, a color and a material map in that order. The sequential ordering of those three steps is critical to avoid distortion problems in the final rendering phase. Once the scene has been completely described, rendering is performed by applying the displacement maps to the parametric surfaces and feeding the color and material maps into the texturing algorithm of our system (figure 3).

Rendering. Rendering can be done in either of two ways: (i) using a specific algorithm for rendering parametric surfaces, or (ii) converting the surface representation to a polygon mesh and rendering it using specialized hardware. In either case we can use displacement functions where the level of detail of the perturbations varies according to the distance of the surface to the viewer. Such functions are stored in a pyramid structure similar to those utilized for mip-maps [Will83].

The rendering phase of our system includes the following algorithms:

- *Visibility*. We use a modified version of Catmull's algorithm [Catm74].
- *Shading*. We use Phong shading [Bui-75] with a Lambertian diffuse reflection function and multiple light sources.
- *Rendering*. We use Williams' Z-buffer algorithm [Will78].

To allow smooth transitions between bump mapping algorithms we also use another algorithm due to Chover [Chov96]. This algorithm uses displacement maps to render surfaces closer to the viewer and normal perturbations to render surfaces further from the viewer. For intermediate surfaces smooth transitions are achieved by varying the displacement linearly with the distance and interpolating the normals at the same time.

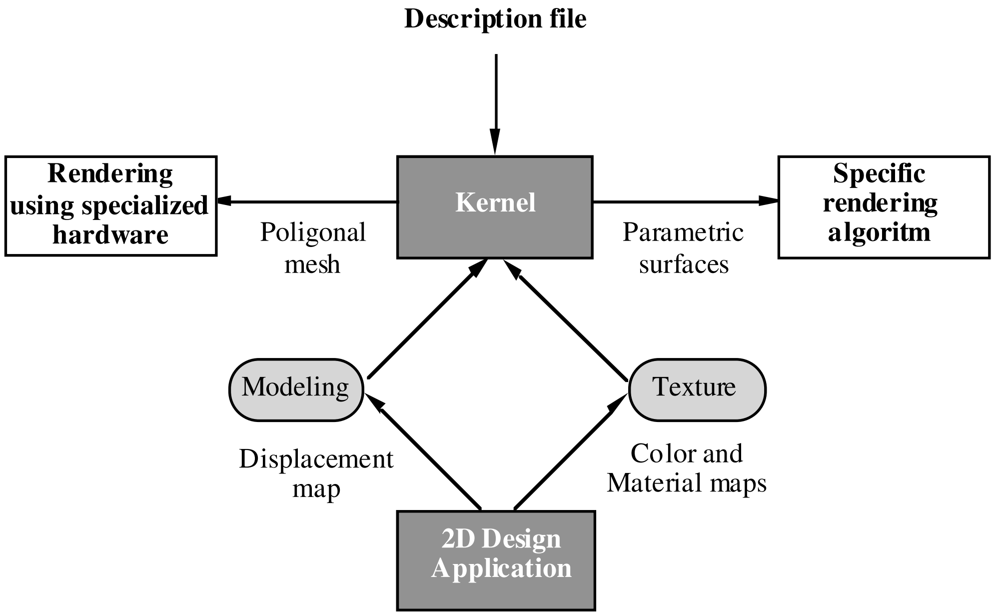


Figure 3. *Module decomposition of the urban modeling application.*

Results. Color plate 4 shows an example image generated by our system using the maps depicted in color plates 1, 2, and 3.

6 Conclusions

We have developed a system that allows realistic rendering of urban environments by using an extension of the displacement and texturing methods introduced by Chover, Vivó, and Quirós [Chov93]. The main advantages of our system are the application of the concept of urban environment and the use of a minimal number of surfaces to represent the objects of the environment. We add detail to those surfaces by using both displacement and texturing techniques. We thus avoid the need of large polygon representations, which require each its own texture map and material description. Implementing urban environments with our approach provides an easy-to-use and compact representation for scenes that classically required a lot of effort to create and manipulate.

Other features of our system are a representation with multiple levels of detail to render surface detail according to the distance to the viewer, and an algorithm that produces smooth transitions between different level-of-detail representations [Chov96]. Finally, an important contribution of our work is the definition of three-dimensional texture functions that avoid distortion problems when using two-dimensional textures together with displacement maps.

At the application level, we believe that our system can be very useful to model interesting urban effects. For example, we can easily cover the buildings of a city with snow by displacing a surface upwards along the direction of its normal. This technique was used before by Chover to simulate snow accumulated on tree branches [Chov95].

The images presented in this paper show that our system allows simple and compact models of urban scenes with good visual results. However, there are some slight problems that still need to be addressed. Designing a balcony, for instance, can not be entirely done by displacing a flat surface. Similarly, there are other details in facades that require some improvements in our system. We are currently in the process of finding solutions to those problems.

Color Plate 4.
Urban Environment

7. Bibliography

[Catm74] E. Catmull, "Computer display of curved surfaces", Proc. of the conf. on Comp. Graphics: Pattern Recognition and Data Structures, 1974, pag. 4--17

[Cook84] Robert L. Cook, "Shade trees", Computer Graphics (SIGGRAPH '84 Proceedings), 1984, Julio, vol. 18, pag. 223--231

[Chov93] Miguel Chover, Roberto Vivó y Ricardo Quiros, "Síntesis de Imagen con Hipertexturas", CEIG´93, pag. 276--287

[Chov95] M. Chover et al., "Texture, Displacement and Immersion: A Model for Tree Rendering", WSCG'95, Feb., pag. 69--78

[Chov96] M. Chover, "Un modelo de desplazamiento, deformación y textura, para la representación de características macrogeométricas sobre superficies paramétricas", Tesis. DSIC., Univ. Politécnica de Valencia, 1996, Feb., pag. 173

[Gard84] Geoffrey Y. Gardner, "Simulation of Natural Scenes Using Textured Quadric Surfaces", Computer Graphics, 1984, Julio, vol. 18, pag. 11--20

[Ma 88] Song D. Ma and Hong Lin, "Optimal texture mapping", Eurographics '88, 1988, Septiembre, pag. 421--428

[Nuzz80] V. Nuzzolese, "Obtaining Buildings Perspective Views Complete of Materials Textures", CAD 80, 1980, vol. 4, pag. 109--117

[Peac85] Darwyn R. Peachey, "Solid Texturing of Complex Surfaces", Computer Graphics (SIGGRAPH '85 Proceedings), 1985, Julio, vol. 19, pag. 279--286

[Perl85] Ken Perlin , "An Image Synthesizer", Computer Graphics (SIGGRAPH '85 Proceedings), 1985, Julio, vol. 19, no. 3, pag. 287--296

[Upst90] Steve Upstill, "The Renderman Companion", Ed. Addison-Wesley, 1990, pag. 455

[Will78] L. Williams, "Casting cuved shadows on curved surfaces", Computer Graphics, 1978, vol. 12, no. 3, pag. 270--274

[Will83] Lance Williams, "Pyramidal Parametrics", Computer Graphics (SIGGRAPH '83 Proceedings), 1983, Julio, vol. 17, no. 3, pag. 1--11