Texture Mapping

Michael Kazhdan

(600.457)

HB Ch. 14.8,14.9
FvDFH Ch. 16.3, 16.4.5, 16.6
Textures

We know how to go from this… to this

J. Birn
Textures

But what about this... to this?

J. Birn
Textures

• How can we go about drawing surfaces with complex detail?

Target Model
Textures

- How can we go about drawing surfaces with complex detail?

Direct:
- Tessellate in a complex manner and then associate the appropriate material properties to each vertex.
Textures

- How can we go about drawing surfaces with complex detail?

**Indirect:**
- Use a simple tessellation and use the location of surface points to look up the appropriate color values
Textures

• Advantages:
  ◦ The 3D model remains simple
  ◦ It is easier to design/modify a texture image than it is to design/modify a surface in 3D.
Textures

Implementation:

• Associate a *texture coordinate* to each vertex \( v \): 
  \[ s^v = \{ s^v_1, \ldots, s^v_n \} \quad (0 \leq s^v_i \leq 1, \ n \in \{1,2,3\}) \]

• When rasterizing, *interpolate* to get the texture coordinate to at a pixel: 
  \[ s^p = \{ s^p_1, \ldots, s^p_n \} \]

• *Sample* the texture at \( s^p \) to get the color at \( p \).
Example: Brick Wall
Example: Brick Wall

\[ s^v = (0,1) \quad s^v = (1,1) \]

\[ s^v = (0,0) \quad s^v = (1,0) \]

+
2D Texture

- Coordinates described by variables $s$ and $t$ and range over interval $(0,1)$
- Texture elements are called texels
- Often 4 bytes (rgba) per texel
Texture Mapping

- Scan conversion:
  - Interpolate texture coordinates down/ across scan lines

\[ (s, t) = \alpha (s_1, t_1) + \beta (s_2, t_2) + \gamma (s_3, t_3) \]
3D Rendering Pipeline (for direct illumination)

3D Primitives

- 3D Modeling Coordinates

 Modeling Transformation

- 3D World Coordinates

 Camera Transformation

- 3D World Coordinates

 Lighting

- 3D Camera Coordinates

 Projection Transformation

- 2D Screen Coordinates

 Clipping

- 2D Screen Coordinates

 Viewport Transformation

- 2D Image Coordinates

 Scan Conversion

- 2D Image Coordinates

 Image

Texture mapping
Interpolation

Given color/normal/depth/texture values at the vertices of a triangle, we get the value at a pixel by:

- Interpolating along the edges of the triangle as we traverse the scan lines
- Interpolating along the scan line as we traverse the pixels

\[ A = (1 - \alpha) \cdot d_1 + \alpha \cdot d \]
\[ B = (1 - \beta) \cdot d_2 + \beta \cdot d_3 \]
\[ d = (1 - \gamma) \cdot A + \gamma \cdot B \]
Scan Conversion

How do we average information from the three vertices of a triangle?

- Interpolate using weights determined by the 2D screen space projection.
- Interpolate using weights determined by the 3D locations.

It’s easier to do the interpolation in 2D because we are in 2D when rasterizing.

Is there a difference?
Scan Conversion Example

A line segment in 2D projected onto a 1D screen.

How should we interpolate the information from vertices $p_1$ and $p_2$ at the pixel corresponding to ray $R$?

$$z = 0 \quad z = 1$$
Scan Conversion Example

A line segment in 2D projected onto a 1D screen.

1. \( R \) intersects the projected line segment in the middle:
   - We should use equal contributions from \( p_1 \) and \( p_2 \).
Scan Conversion Example

A line segment in 2D projected onto a 1D screen.

1. \( R \) intersects the projected line segment in the middle:
   - We should use equal contributions from \( p_1 \) and \( p_2 \).

2. \( R \) intersects the 2D line segment closer to \( p_1 \):
   - We should use more information from \( p_1 \) than from \( p_2 \).
Scan Conversion Example

A line segment in 2D projected onto a 1D screen.

Recall: The 2D point \((x, z)\) maps to the point \((x/z)\) in 1D.

If \(p_1 = (x_1, z_1)\) and \(p_2 = (x_2, z_2)\), to find the blending value \(\alpha\) for a pixel falling at position \(x\) in the screen we need to solve:

\[
(1 - \alpha)(x_1, z_1) + \alpha(x_2, z_2) \rightarrow (x, 1)
\]

\[
((1 - \alpha)x_1 + \alpha x_2, (1 - \alpha)z_1 + \alpha z_2) \rightarrow (x, 1)
\]

\[
\frac{(1 - \alpha)x_1 + \alpha x_2}{(1 - \alpha)z_1 + \alpha z_2} = \frac{x}{1}
\]
Scan Conversion Example

A line segment in 2D projected onto a 1D screen.

• How do we interpolate correctly?

Recall: The 2D point \((x, z)\) maps to the point \((x/z)\) in 1D.

If \(p_1 = (x_1, z_1)\) and \(p_2 = (x_2, z_2)\), to find the blending value \(\alpha\) for a pixel falling at position \(x\) in the screen we need to solve:

\[
\frac{(1 - \alpha)x_1 + \alpha x_2}{(1 - \alpha)z_1 + \alpha z_2} = \frac{x}{1}
\]

This is not the same as solving for the blending value in the image plane:

\[
(1 - \alpha) \frac{x_1}{z_1} + \alpha \frac{x_2}{z_2} = \frac{x}{1}
\]

To compute the interpolation weights correctly, we need to perform a perspective divide:
Texture Mapping

Linear interpolation of texture coordinates in screen space

Correct interpolation with perspective divide

Hill Figure 8.42
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Filtering

• Texture mapping applications
  ◦ Modulation textures
  ◦ Illumination mapping
  ◦ Bump mapping
  ◦ Environment mapping
  ◦ Shadow maps
Option: Unfold/Map Entire Surface

[Piponi2000]
Option: Unfold/Map Entire Surface

• Tricky, because mapped surface may have severe distortions

• However, because texture is continuous, may be easier to think about

Gu et al. 2003
Option: Unfold/Map Entire Surface

• Tricky, because mapped surface may have severe distortions

• However, because texture is continuous, may be easier to think about

In general, it is impossible to parameterize a complex shape to a simple base domain so that both angles and areas are preserved
Option: Make an Atlas

charts  atlas  surface

[Sander2001]
Option: Make an Atlas

• Less distortion on each little piece of atlas
• Need to pack to patches to reduce wasted space in texture image
• Need to enforce continuity across seam patch boundaries
• May be more difficult to think about the relationships between the different pieces
Overview

• Texture mapping methods
  ◦ Parameterization
  ◦ Filtering

• Texture mapping applications
  ◦ Modulation textures
  ◦ Illumination mapping
  ◦ Bump mapping
  ◦ Environment mapping
  ◦ Shadow maps
Texture Filtering

Must sample texture to determine color at each pixel in image

Angel Figure 9.4
Texture Filtering

Sample texture to determine color at each pixel.

- If the transformation from screen space to texture space does not preserve area, we need to compute the average of the pixels in texture space.

Average over many pixels
Texture Filtering

Size of filter depends on the projective deformation

- Can prefilter images for better performance
  - Mip maps
  - Summed area tables

Average over many pixels
Mip Maps

- Keep textures prefILTERed at multiple resolutions
  - For each pixel, use the mip-map closest level(s)
  - Fast, easy for hardware

Texture hierarchy

Average over many pixels

Screen
Mip Maps

• Keep textures prefiltered at multiple resolutions
  ◦ For each pixel, use the mip-map closest level(s)
  ◦ Fast, easy for hardware

Texture hierarchy

Average over a few pixels

Screen
Mip Maps

• Keep textures prefiltered at multiple resolutions
  ◦ For each pixel, use the mip-map closest level(s)
  ◦ Fast, easy for hardware

Again: we’re trading aliasing for blurring!
Mip Maps

• Keep textures prefILTERED at multiple resolutions
  ◦ For each pixel, use the closest mip-map level(s)
  ◦ Fast, easy for hardware

• This type of filtering is isotropic:
  ◦ It doesn’t take into account that there is more compression in the vertical direction than in the horizontal one

Again: we’re trading aliasing for blurring!
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle:

\[
\text{Sum}([a, b] \times [c, d]) = \int_{a}^{b} \int_{c}^{d} f(x, y) \, dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula:

\[
\int_a^b \int_c^d f(x, y) dy \, dx = \int_0^b \int_0^d f(x, y) dy \, dx - \int_0^b \int_0^c f(x, y) dy \, dx - \int_0^a \int_0^d f(x, y) dy \, dx + \int_0^a \int_0^c f(x, y) dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_a^b \int_c^d f(x, y)dy \, dx = \int_0^b \int_0^d f(x, y)dy \, dx - \int_0^b \int_0^c f(x, y)dy \, dx - \int_0^a \int_0^d f(x, y)dy \, dx + \int_0^a \int_0^c f(x, y)dy \, dx
\]
Summed-Area Tables

Key Idea:

- Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

\[
\int_a^b \int_c^d f(x, y) dy \, dx = \int_0^b \int_0^d f(x, y) dy \, dx - \int_0^b \int_0^c f(x, y) dy \, dx - \int_0^a \int_0^d f(x, y) dy \, dx + \int_0^a \int_0^c f(x, y) dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula:

\[
\int_a^b \int_c^d f(x,y) \, dy \, dx = \int_0^b \int_0^d f(x,y) \, dy \, dx - \int_0^b \int_0^c f(x,y) \, dy \, dx - \int_0^a \int_0^d f(x,y) \, dy \, dx + \int_0^a \int_0^c f(x,y) \, dy \, dx
\]
### Summed-Area Tables

**Key Idea:**

- Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

- Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_a^b \int_c^d f(x, y) dy \, dx = \int_0^b \int_0^d f(x, y) dy \, dx - \int_0^b \int_0^c f(x, y) dy \, dx - \int_0^a \int_0^d f(x, y) dy \, dx + \int_0^a \int_0^c f(x, y) dy \, dx
\]
Summed-Area Tables

Key Idea:

• Approximate the summation/integration over an arbitrary region by a summation/integration over an axis-aligned rectangle.

• Perform the integration quickly by pre-computing integrals and leveraging the formula

\[
\int_a^b \int_c^d f(x,y) dy \, dx = \int_0^b \int_0^d f(x,y) dy \, dx - \int_0^b \int_0^c f(x,y) dy \, dx - \int_0^a \int_0^d f(x,y) dy \, dx + \int_0^a \int_0^c f(x,y) dy \, dx
\]
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:

\[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 0</td>
<td></td>
</tr>
<tr>
<td>0 3 1 1</td>
<td></td>
</tr>
<tr>
<td>4 2 0 1</td>
<td></td>
</tr>
<tr>
<td>1 2 1 3</td>
<td></td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

• Precompute the values of the integral:

\[
S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx
\]

• Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  4  0</td>
<td></td>
</tr>
<tr>
<td>0  3  1  1</td>
<td></td>
</tr>
<tr>
<td>4  2  0  1</td>
<td></td>
</tr>
<tr>
<td>1  2  1  3</td>
<td></td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

• Precompute the values of the integral:

\[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

• Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  4  0</td>
<td></td>
</tr>
<tr>
<td>0  3  1  1</td>
<td></td>
</tr>
<tr>
<td>4  2  0  1</td>
<td></td>
</tr>
<tr>
<td>1  2 1 3</td>
<td>1  3</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_{0}^{a} \int_{0}^{b} f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

Original image

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Summed area table

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
</table>
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

**Original image**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>4</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Summed area table**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
<td>7</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

• Precompute the values of the integral:

\[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

• Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 0</td>
<td></td>
</tr>
<tr>
<td>0 3 1 1</td>
<td></td>
</tr>
<tr>
<td>4 2 0 1</td>
<td>5</td>
</tr>
<tr>
<td>1 2 1 3</td>
<td>1 3 4 7</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

- Precompute the values of the integral:
  \[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

- Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1  2  4  0</td>
<td></td>
</tr>
<tr>
<td>0  3  1  1</td>
<td></td>
</tr>
<tr>
<td>4  2  0  1</td>
<td></td>
</tr>
<tr>
<td>1  2  1  3</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5  9</td>
</tr>
<tr>
<td></td>
<td>1  3  4  7</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Pre-Process)

• Precompute the values of the integral:

\[ S(a, b) = \int_0^a \int_0^b f(x, y) \, dy \, dx \]

• Each texel is the sum of all texels below and to the left of it

<table>
<thead>
<tr>
<th>Original image</th>
<th>Summed area table</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 2 4 0</td>
<td>6 15 21 26</td>
</tr>
<tr>
<td>0 3 1 1</td>
<td>5 12 14 19</td>
</tr>
<tr>
<td>4 2 0 1</td>
<td>5 9 10 14</td>
</tr>
<tr>
<td>1 2 1 3</td>
<td>1 3 4 7</td>
</tr>
</tbody>
</table>
Summed-Area Tables (Run-Time)

• Given a pixel on the screen that maps to a rectangle in texture space use the summed area table to compute the average:

\[
\text{Sum}([1,3] \times [2,3]) = S(3,3) - S(0,3) - S(3,1) + S(0,1) = 26 - 6 - 14 + 5 = 11
\]

\[
\text{Average}([1,3] \times [2,3]) = \frac{\text{Sum}([1,3] \times [2,3])}{\text{Area}([1,3] \times [2,3])} = \frac{11}{6}
\]
Overview

• Texture mapping methods
  ○ Parameterization
  ○ Filtering

• Texture mapping applications
  ○ Modulation textures
  ○ Illumination mapping
  ○ Bump mapping
  ○ Environment mapping
Modulation textures

Map texture values to scale factor

Modulation

\[ I = \overline{T(s,t)} \left( I_E + K_A I_{AL} + \sum_i \left( K_D \langle \vec{N}, \vec{L}_i \rangle I_i + K_S \langle \vec{V}, \vec{R}_i \rangle^n I_i \right) \right) \]
Illumination Mapping

Map texture values to any material parameter

Modulation

Diffuse

\[ I = I_E + K_A I_{AL} + \sum_i \left( T(s,t) (\vec{N}, \vec{L}_i) I_i + K_S (\vec{V}, \vec{R}_i)^n I_i \right) \]
Illumination Mapping

Map texture values to any material parameter

Modulation  Diffuse

Note that we need to evaluate the texture at each pixel but can still use the interpolated lighting values \( \langle \vec{N}, \vec{L}_i \rangle \)

\[
I = I_E + K_A I_{AL} + \sum_i \left( T(s, t) \langle \vec{N}, \vec{L}_i \rangle I_i + K_S \langle \vec{V}, \vec{R}_i \rangle^n I_i \right)
\]
Illumination Mapping

Map texture values to any material parameter

- Modulation
- Diffuse
- Specular

\[ I = I_E + K_A I_{AL} + \sum_i \left( K_D \langle \vec{N}, \vec{L}_i \rangle I_i + T(s, t) \langle \vec{V}, \vec{R}_i \rangle^n I_i \right) \]
Illumination Mapping

Map texture values to any material parameter

Modulation  Diffuse  Specular

Again, we don’t need to re-compute most of the lighting calculation $\langle \vec{V}, \vec{R}_i \rangle^n$

\[
I = I_E + K_AI_{AL} + \sum_i \left( K_D \langle \vec{N}, \vec{L}_i \rangle I_i + T(s,t) \langle \vec{V}, \vec{R}_i \rangle^n I_i \right)
\]
Bump Mapping

• Recall that many parts of our lighting calculation depend on surface normals

\[ I = I_E + K_AI_{AL} + \sum_i \left( K_D \langle \vec{N}, \vec{L}_i \rangle I_i + K_S \langle \vec{V}, \vec{R}_i \rangle^n I_i \right) \]
Bump Mapping

\[ n_0 \quad n_1 \]

P. Rheingans
Bump Mapping

Phong shading performs per-pixel lighting calculations with the interpolated normals, approximating a smoothly curved surface.
Bump Mapping

Phong shading performs per-pixel lighting calculations with the interpolated normals, approximating a smoothly curved surface.

With bump maps, we encode the normals in the texture.

P. Rheingans
Bump Mapping

Phong shading performs per-pixel lighting calculations with the interpolated normals, approximating a smoothly curved surface.

With bump maps, we encode the normals in the texture. This allows Phong shading to give the appearance of a bumpy surface.

P. Rheingans
Bump Mapping
Bump Mapping

Note that bump mapping does not change object silhouette
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture values are dynamically reflected off surface patch.

Set the texture coordinates based on the direction of the reflected view direction.
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture values are dynamically reflected off surface patch.

Set the texture coordinates based on the direction of the reflected view direction.

Changing the position of the camera changes the texture coordinates.
Environment Mapping

- Generate a spherical/cubic map of the environment around the model.
- Texture values are dynamically reflected off surface patch.

Set the texture coordinates based on the direction of the reflected view direction.

Changing the position of the camera changes the texture coordinates.
Environment Mapping

• Generate a spherical/cubic map of the environment around the model.

• Texture values are dynamically reflected off surface patch

Set the texture coordinates based on the direction of the reflected view direction

Changing the position of the camera changes the texture coordinates.
Environment Mapping

Texture values are dynamically reflected off surface patch.