

Texture Mapping And Texture Synthesis

Spring CSE 291

Cindy Wang

02/20/03

UCSD

1

Organization

- Part 1-- Fundamentals of Texture Mapping
 - Mapping
 - Scanning algorithm
 - Anti-aliasing
- Part 2--Non – Parametric Sampling Texture Synthesis
 - Algorithm overview
 - Details
 - Limitation

UCSD

2

Reference

- Paul S. Heckbert. *Survey of Texture Mapping*, IEEE Computer Graphics and Applications, Nov. 1986.
<http://www-2.cs.cmu.edu/~ph/textsurv.pdf>
- Paul S. Heckbert. *Fundamentals of Texture Mapping and Image Warping*, Master's Thesis, University of California, Berkeley, 1989.
<http://www2.cs.cmu.edu/~ph/textfund/textfund.pdf>

UCSD

3

Introduction

- Texture Mapping
 - texture mapping is a successful technique to create high-quality image synthesis without the tedium of modeling and rendering every details of a surface
- Results depend on some key elements
 - Mapping
 - Filtering
 - Sampling

UCSD

4

Coordinate Systems

- ❑ Texture Space
 - ✓ 2-D space of surface textures
- ❑ Object Space
 - ✓ 3-D coordinate system in which geometries are defined
- ❑ World Space
 - ✓ A global coordinate system that is related to each object's local object space using 3d modeling transformations
- ❑ Screen Space
 - ✓ The coordinate system of display

UCSD

5

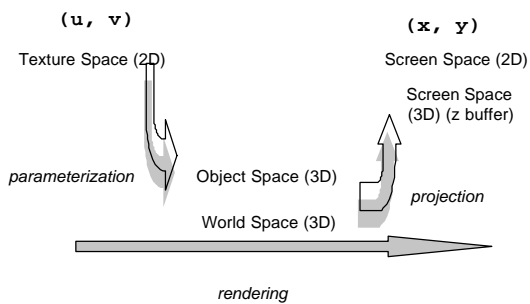
Goal of Texture Mapping

- ❑ Definition
 - ✓ the mapping of a function onto a surface in 3-D
- ❑ 2-step mapping process
 - ✓ The source image(texture) is mapped onto a surface in 3-D object space
 - ✓ The surface is then mapped to the destination image by the viewing projection

UCSD

6

Spaces mapping



UCSD

7


Parameterization

- ☞ Mapping a 2-D texture onto a 3-D surface requires surface parameterization
- ☞ Affine mapping
- ☞ Bilinear mapping
- ☞ Perspective mapping

UCSD

8

Affine mapping



- ❑ scales, rotations, translations, shears
- ❑ Preserve parallel lines and equispaced points along lines
- ❑ Triangle keeps shape after mapping
 - o vertices have $Au+Bv+C$ form

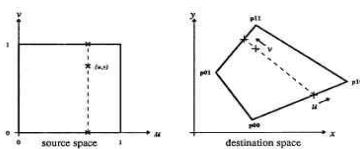
UCSD
9

Bilinear mapping

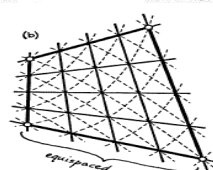
- ❑ Mapping a square into quadrilateral
- ❑ Preserve horizontal / vertical lines
- ❑ Diagonal lines are distorted

UCSD
10

Example of Bilinear mapping



(b)

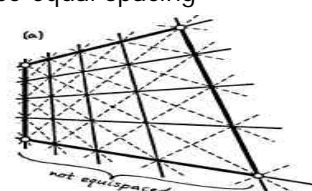


UCSD
11

Perspective Mapping

- ❑ A better parameterization choice for planar quadrilaterals
- ❑ Preserve lines at all orientations
- ❑ Sacrifice equal spacing

(c)



UCSD
12

Comparison of mappings

	Preserve lines	Preserve parallel	Preserve equi-distance	Invertible matrix	Degree of freedom
Affine	Yes	Yes	Yes	Yes	6
Bilinear	Not diagonals	No	Not diagonals	No	8
Perspective	Yes	No	No	Yes	8

UCSD

13

Screen scanning

```

For y (screen row)
  For x (screen column)
    Compute u(x,y) and v(x,y)
    copy SCR[x,y] = TEX[u,v]
    
```

- ☞ most common
- ☞ mapping invertible
- ☞ random access texture

UCSD

14

Texture Scanning

```

For v (texture row)
  For u (texture column)
    Compute x(u,v) and y(u,v)
    SCR[x,y] = TEX[u,v]
    
```

- ☞ simple
- ☞ invertible
- ☞ can result in holes and overlaps

UCSD

15

Two pass

```

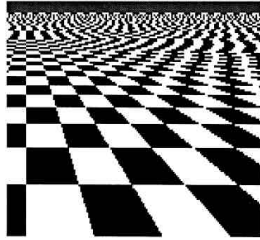
For v
  for u
    compute x(u,v)
    copy TEX[u,v] to Temp[x,v]
  For x
    for v
      compute y(x,v)
      copy Temp[x,v] to SCR[x,y]
  ☞ Work well for affine and perspective mapping
    
```

UCSD

16

Aliasing

- Aliasing is a result of high frequency signals
- Moiré pattern near horizon
- High frequency: black/white alternate frequently



UCSD

17

Anti-Aliasing

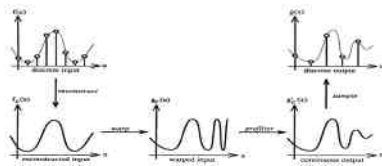
- Point sample at high frequency
 - Associate sample rate with local intensity variance.
- Low pass filtering before sampling
 - Preferable
 - Input must be band-limited (can be solved by signal processing knowledge)

UCSD

18

Four steps of Anti-Aliasing

- Reconstruct continuous signal
- Warp the signal
- Low pass filter the signal
- Resample the signal



UCSD

19

Space Variant Filtering

- Shape varies with space location
- Pre-image of pixels approximated as quadrilaterals or ellipses
- Cross sectional shape:
 - Ideal Filter: $\text{sinc}(x)$
 - Practical Filter: box, triangular, cubic B-spline, Gaussian



a



b



c

UCSD

20

Direct Convolution

- ❏ Computes weighted average of texture maps
- ❏ Previous works
- ❏ EWA (Elliptical Weighted Average)
- ❏ Cost per screen pixel: proportional to the number of texture pixels

UCSD

21

EWA

- ❏ EWA (Elliptical Weighted Average) filter
- ❏ Space-variant filter
- ❏ Pixels circular and overlapping
- ❏ Weighted average from concentric ellipses
- ❏ Computes average in texture space

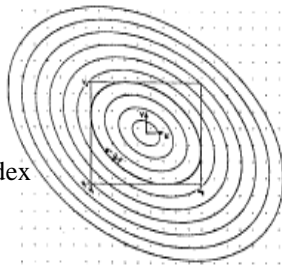
UCSD

22

EWA sample region

- ❏ Arbitrarily oriented ellipse defined by two vectors
- ❏ Using a biquadratic function as its radial index

$$Q(u, v) = Au^2 + Buv + Cv^2$$

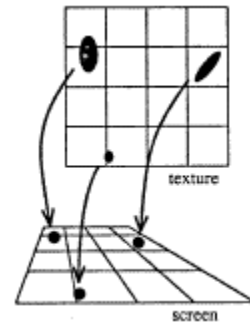


UCSD

23

EWA: Space-variant

- ❏ Space-variant sample areas
- ❏ Circle in screen space maps to ellipse in texture space
- ❏ Greater sample accuracy



UCSD

24

Prefiltering

- ☞ Purpose
 - Speed up filtering process
- ☞ Two data structures
 - Pyramid—most commonly used
 - Integrated array

UCSD

25

Pyramid

- ☞ A hierarchical data structure
- ☞ Building pyramid
 - Divide texture as $n \times n$ texture areas
 - Build pyramid iteratively
- ☞ Space complexity
 - $4/3$ of original data

UCSD

26

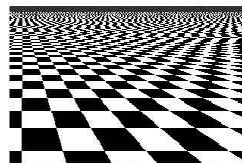
Pyramid

- ☞ Assume pre-image texture area is squares
- ☞ Compute texture value by using trilinear interpolation method
- ☞ Each screen pixel requires at most 8 texture data

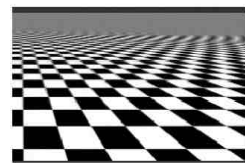
UCSD

27

Result of anti-aliasing



(a) Point sampling.



(b) Trilinear interpolation on a pyramid.

UCSD

28

Part 2: Texture Synthesis by Non-parametric Sampling

UCSD

29

Reference

- Alexei A. and Thomas K Leung
Texture Synthesis by non-parametric sampling
IEEE International Conference on Computer Vision, Corfu, Greece, September 1999

UCSD

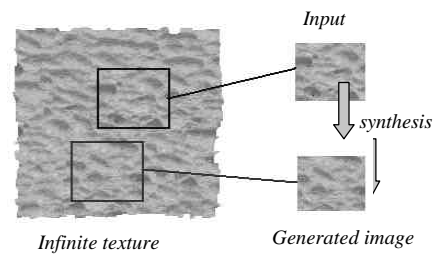
30

Problem of texture synthesis

- ✓ Define texture as some visual pattern on an *infinite* 2D plane, which, at some scale, has a *stationary* distribution
- ✓ Given a finite sample from some texture (an image).
- ✓ Assume: the sample is large enough
- ✓ Goal: to synthesize other samples from the same texture

UCSD

31

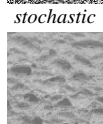
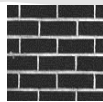


UCSD

32

The challenge

- ☞ Traditionally, textures can be classified as
 - Regular
 - Stochastic
- ☞ In real life, textures lie between these two extremes and need a single model
- ☞ So, how to analyze and model textures?



Both??

UCSD

33

Problem of producing language

- ☞ Shannon proposed a way of generating English-looking text using N-grams
- ☞ Basic idea:
 - Assume a generalized Markov chain
 - Using a large sample language, compute probability distributions of each letter given N-1 previous letter
 - Repeatedly sample the Markov chain to produce new letters

UCSD

34

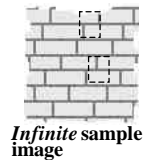
Non parametric sampling Algorithm

- ☞ Motivated by the way of modeling language using Markov chain
- ☞ Texture is “grown” pixel by pixel, outwards from an initial seed.
- ☞ Model texture as a MRF
- ☞ conditional pdf of pixel given its neighbors synthesized thus far is computed directly from the sample image

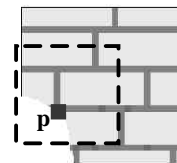
UCSD

35

Synthesizing one pixel



SAMPLE



Generated image

- Assuming Markov property,
- Compute conditional probability distribution of p , given the neighborhood window
- Instead of constructing a model, directly search the input image for all such neighborhoods to produce a histogram for p
- To synthesize p , just randomly pick one

UCSD

36

Compute probability distribution

- Based on MRF, assume p is independent of $I \setminus w(p)$.
- Define a set

$$\Omega(p) = \{w' \subset I_{real} : d_{perc}(w', w(p)) = 0\}$$

- Estimate conditional pdf of p with a histogram of all center pixel values in $\Omega(p)$

UCSD

37

Practical synthesizing a pixel

- In practice, find $\Omega'(p) \approx \Omega(p)$

$$\Omega'(p) = \{w'_{best} \subset I_{real} : d_{perc}(w'_{best}, w(p)) < \epsilon\}$$
- So we find the **best** match using normalized sum of squared distances metric – SSD error.
 - ✓ weighted by a Gaussian to emphasize local structure, and take all samples within some distance from that match

UCSD

38

Synthesizing a texture



- Take 3*3 random patch from sample as a seed
- Only match on the known values in the window $w(p)$
- Still use Gaussian-weighted SSD to compute pdf of p

UCSD

39

Some details

- This algorithm doesn't guarantee the pdf of p is valid
- So, we should note:
 - ✓ Synthesize those pixels which have most known neighborhood pixels first
 - ✓ Using Gaussian – weighted SSD is important
 - ✓ To make sure the new pixel agrees with its closest neighborhood pixels
 - ✓ Approximates reduction to a smaller neighborhood window if data is too sparse

UCSD

40

Randomness parameter

- ☞ The width w of the window size is the only parameter to be set by users.
- ☞ The randomness of texture changes with window size parameter
- ☞ Window size should be large enough to capture the stationary of sample texture

UCSD 41

An example

The diagram illustrates how a single window size can be used to generate different textures. A central square window contains a few white dots. Lines radiate from this window to five other squares, each showing a different texture: a sparse pattern of dots, a pattern of dots with some larger circles, a pattern of dots with some larger circles and some larger squares, a pattern of dots with some larger circles and some larger squares and some larger rectangles, and a pattern of dots with some larger circles and some larger squares and some larger rectangles and some larger irregular shapes.

UCSD 42

More Results...

sample

5 11 15 23

UCSD 43

More Results...

5 15 23

UCSD 44

