Texture

• What is texture?
  – Easy to recognize, hard to define
  – Deterministic textures (“thing-like”)
  – Stochastic textures (“stuff-like”)

• Tasks
  – Discrimination / Segmentation
  – Classification
  – Texture synthesis
  – Shape from texture
  – Texture transfer
  – Video textures

Texture Discrimination
What is texture?
- An image obeying some statistical properties
- Similar structures repeated over and over again
- Often has some degree of randomness
Melnik & Perona's Filters

Gabor filter kernels - product of symmetric Gaussian with oriented sinusoid
⇒ smoothed derivative kernels

DOOG filters - difference of Gaussian filters

⇒ spot and bar filters are many scales, orientations, and phases

\[
\begin{align*}
G_{\text{symm}}(x,y) &= \cos(k_x x + k_y y) \exp\left[-\frac{x^2 + y^2}{2 \sigma^2}\right] \\
G_{\text{unsym}}(x,y) &= \sin(k_x x + k_y y) \exp\left[-\frac{x^2 + y^2}{2 \sigma^2}\right]
\end{align*}
\]
Steerable (i.e., Oriented) Pyramids

- Laplacian pyramid not appropriate for texture analysis because it does not encode orientation info.

- Oriented pyramids (Simoncelli)

Steerable Pyramids
- Multiresolution, multi-orientation image decomposition

4 orient.
2 scales
Synthesizing One Pixel

- What is $P(x|\text{neighborhood of pixels around } x)$
- Find all the windows in the image that match the neighborhood
  - consider only pixels in the neighborhood that are already filled in
- To synthesize $x$
  - pick one matching window at random
  - assign $x$ to be the center pixel of that window
Markov Random Field

A Markov random field (MRF)

• generalization of Markov chains to two or more dimensions

First-order MRF:

• probability that pixel $X$ takes a certain value given the values of neighbors $A$, $B$, $C$, and $D$:

$$P(X|A, B, C, D)$$

• Higher order MRF’s have larger neighborhoods

Markov Chain

• Markov Chain
  – a sequence of random variables $X_1, X_2, \ldots, X_n$
  – $X_t$ is the state of the model at time $t$

$$X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow X_5$$

• Markov assumption: each state is dependent only on the previous one
  • dependency given by a conditional probability:

$$p(X_t|X_{t-1})$$

• The above is actually a first-order Markov chain
• An $N$th-order Markov chain:

$$p(X_t|X_{t-1}, \ldots, X_{t-N})$$
Really Synthesizing One Pixel

- An exact neighborhood match might not be present
- So we find the best matches using SSD error and randomly choose between them, preferring better matches with higher probability

Growing Texture

- Starting from the initial image, “grow” the texture one pixel at a time
Window Size Controls Regularity

More Synthesis Results

Increasing window size
More Results

reptile skin

aluminum wire

Failure Cases

Growing garbage

Verbatim copying
Image-Based Text Synthesis

**Idea:**

- Observation: neighbor pixels are highly correlated

**Effros & Leung ’99 Extended**

- **Observation:** neighbor pixels are highly correlated

**Idea:** unit of synthesis = block

- Exactly the same but now we want \( P(B|N(B)) \)

- Much faster: synthesize all pixels in a block at once
**Minimal error boundary**

- Overlapping blocks
- Vertical boundary

Overlap error

\[
\begin{pmatrix}
\begin{array}{c}
\text{overlap error}
\end{array}
\end{pmatrix}^2 = \begin{pmatrix}
\end{pmatrix}
\]

Min. error boundary
Philosophy

• The “Corrupt Professor’s Algorithm:”
  – Plagiarize as much of the source image as you can
  – Then try to cover up the evidence
• Rationale:
  – Texture blocks are by definition correct samples of texture, so the only problem is connecting them together

Texture Transfer
Texture Transfer

• Take the texture from one object and “paint” it onto another object
  – This requires separating texture and shape
  – That’s HARD, but we can cheat
  – Assume we can capture shape by boundary and rough shading

Then, just add another constraint when sampling: similarity to luminance of underlying image at that spot

parmesan

+ =

rice

+ =
Shape from Texture

- Main Idea: Projection distorts texture geometry that depends on surface shape and geometry

- Witkin’s Method
  - No model for texture
  - Assume natural textures do not mimic projection effects
  - Isotropy Assumption:
    All surface orientations equally likely and all edge orientations equally likely

- Model geometric distortion on edge orientations

- Statistically model distribution of surface orientations and surface marking orientations
Imaging process distorts surface texture in 2 ways:

1. Distance
   Further objects appear smaller
   Area subtended by solid angle \( \Omega \) is
   \[ d^2 \Omega \] where \( d \) = distance to surface

2. Orientation

As slant \( \alpha \) increases, foreshortening makes point appear closer together as

\[ \frac{1}{\cos \alpha} \]

Goal: Given image of a planar surface, recover \((\xi, \eta)\)

\[ 0 \leq \xi \leq \frac{\pi}{2} \]
\[ -\frac{\pi}{2} \leq \eta \leq \frac{\pi}{2} \]
Features: Edge orientations of surface markings

Geometric Model
Given curve \( C(s) \) on surface \( S \)
Let \( \beta(s) = \) direction of \( C(s) \)'s tangent at \( s \)

\[ \beta = \text{angle between } \beta(s) \text{ and } x-axis \text{ is } S \]

What is projection of \( \beta(s), \beta, C(s) \) into image \( I \)?

Assume orthographic projection
(\( \Rightarrow \) foreshortening effects only)

Let \( C^*(s) = \) orthographic projection of \( C(s) \) in \( I \)

To compute \( C^*(s) \):
1. Put \( C(s) \) in \( I \) using \( I \)'s coords
   \[ \beta(s) = [\cos \beta, \sin \beta] \]

2. Rotate \( I \) by \((\theta, \tau)\) \( \Rightarrow \) \( x \)-axis
   Initially, assume \( \tau = 0 \) (x-axis along
   \[ (x, y, 0) \rightarrow (x \cos \theta, y, x \sin \theta) \]

3. Orthographic projection onto I plane
   \[ (x, y, z) \rightarrow (x, y) \]
Combining 3 steps:
Edge orientation $[\cos \beta, \sin \beta]$ in $\Sigma$
projects to $[\cos \beta \cos \theta, \sin \beta]$ in $I$.
Rightarrow $\tan \beta^* = \frac{\sin \beta}{\cos \beta \cos \theta} \Rightarrow \tan \theta^* = \frac{\sin \beta}{\cos \theta}$

$\Rightarrow \theta^* = \tan^{-1} \left( \frac{\tan \beta^*}{\cos \theta} \right)$

= angle by observed edge orientation and tilt direct (x-axis)

More generally, if $\theta \neq 0$ then

$\theta^* = \tan^{-1} \left( \frac{\tan \beta^*}{\cos \theta} \right) + \theta$

where $\theta^*$ = observed edge orientation wrt $\Sigma$'s x-axis

---

- Isotropy assumption $\Rightarrow$ all surface orientations equally likely
  and all edge orientations equally likely.

$\Rightarrow p.d.f. (\beta, \sigma, \tau) = \frac{1}{\pi} \cdot \frac{1}{\pi} \cdot \sin \sigma$

$\Rightarrow \quad \text{likelihood of } \beta = \beta_0 \text{ in } [0, \pi]$.
$\Rightarrow \quad \text{likelihood of } \tau = \tau_0 \text{ in } [0, \pi]$.
$\Rightarrow \quad \text{likelihood of } \sigma = \frac{\sin \sigma}{\pi^2}$

Find pdf ($\theta^*(\beta) \mid \sigma, \tau$)

$\quad = \frac{1}{\pi} \cdot \frac{\cos \sigma}{\cos \theta^*(\beta) + \sin^2(\theta^*(\beta)) \cos \sigma}$

i.e. says what histogram should look like for given $(\theta, \tau)$.
Assuming edge orientations in \( I \) are independent, \( \Rightarrow \)
\[
p.d.f. (A^* = \{ \alpha_i^*, \ldots, \alpha_n^* \} \mid \sigma, \tau) \]
\[
= \prod_{i=1}^{n} p.d.f. (\alpha_i^* \mid \sigma, \tau)
\]
By Bayes Rule:
\[
p.d.f. (\sigma, \tau \mid A^*) = \frac{\prod_{i=1}^{n} \pi^2 \sin \sigma \cos \sigma}{\cos^2(\alpha_i^* - \tau) + \sin^2(\alpha_i^* - \tau) \cos^2 \sigma}
\]
Maximum likelihood estimator for \((\sigma, \tau) \Rightarrow \)
1. Apply Edge operator
2. Compute edge orientation histogram \( A^* \)
3. For each \((\sigma, \tau)\) compute p.d.f. \( p.d.f. (\sigma, \tau \mid A^*) \)
4. Select \((\sigma, \tau)\) which is maximum.

---

Shape from Textures Using Textures

- Projective distortion changes size
  of textures due to distance and
  shape of textures due to
  foreshortening

1. Detect Textures
   - \( \Omega^2 \) detectors find centers
     of "spots" of varying sizes
   - Connected components analysis
     used to define textures

2. Estimate single planar surface that
   is maximally consistent with textures
   "painted" on surface (no micro-variation)

   Heuristic: Textural-area gradient:
   Area of textures decreases
   with distance and slant angle.
   Fastest in direction of tilt
• Approximate texel area using bounding box:

A_i \approx W_i F_i

• Relate image texel area, A_i, to physical texel area, scene plane orientation, (\theta, \tau), and angle to texel from optical axis, \Theta:

A_i = A_c \left(1 - \tan \theta \tan \Theta \right)^3

where A_c = area of texel at image center

\Theta = \tan^{-1} \left(\frac{\sqrt{(x \cos \tau + y \sin \tau)^2 + r^2}}{f}\right)

where (x, y) = texel center coords
f = image width
r = image radius

• Relate to dimensions of texel's bounding box:

\Theta = \tan^{-1} \left(\frac{k_x}{k_y} \frac{r}{d}\right)

where k_x, k_y = image radius

\Rightarrow F_i = F_p \frac{r}{d} \cos \theta \left(1 - \tan \theta \tan \Theta \right)^2

= F_c \left(1 - \tan \theta \tan \Theta \right)^2
where \( F_c \) = foreshortened dimension of texel at image center

Similarly,

\[
U_i = U_c (1 - \tan \theta \tan \sigma)
\]

\[
A_i = F_i U_i = A_c (1 - \tan \theta \tan \sigma)^3
\]

- Search \((A_c, \sigma, \tau)\) space to maximize fit with observed texel areas, \(A_i\)'s.
- Discretize \(A_c, \sigma, \tau\) values
- Use coarse-to-fine search
Recovering Shape By Purposive Viewpoint Adjustment

Kiriakos N. Kutulakos Charles R. Dyer

Computer Sciences Department
University of Wisconsin
Madison, Wisconsin USA
**Approaches to Recovering Shape**

- **Range sensors**
  Accuracy, distance and resolution limited

- **Stereo**
  Surface texture required

- **Shape from Shading**
  Surface reflectance characteristics required

- **Shape from (Static) Contour**
  Ambiguous: many-to-1 mapping from shape to contour

---

**Active Shape-Recovery**

How can we recover surface shape using an observer able to move?

- **Current approaches.**
  Use a shape-from-motion module
  (e.g., [Cipolla & Blake; ICCV90])
  - Known viewer velocities and accelerations
  - Compute velocities and accelerations of image points

- **Our approach**
  Control position relative to the surface
  - Maintain fixation
  - Measure relative viewing direction changes
  - Compute occluding contour curvatures
Motivation for the Approach

Some views provide more information than others about the shape of a surface:

- An active observer can use these views to recover shape information

Goal: Recover shape for points on the visible rim

- Recover
  1. Principal directions
  2. Principal curvatures
- Assume orthographic projection
Occluding Contour

Rim: Points where surface tangent plane contains the visual ray, $\vec{v}$:
\[ \vec{v} \cdot \vec{n} = 0 \]
(also: Limb, Contour Generatrix)

Visible Rim: Rim points and in view (i.e., not occluded)

Occluding Contour: Projection of the visible rim on image. Collection of open and closed smooth curves; endpoints are Cusps or T-junctions

Silhouette
Using the Occluding Contour

Occluding contour properties

- Dependence on viewpoint & shape well-understood
- Provides shape in absence of markings & surface reflectance information
- Can be efficiently tracked (Blake et al., 1993)
- Recoverable from
  - stereo (Vaillant & Faugeras, 1992)
  - viewpoint control (Kutulakos & Dyer, 1994)
Properties of Occluding Contour

- Geometry is **surface-dependent**
- Projection of a limited set of surface points
- Geometry is **viewpoint-dependent**

Assumptions

- Smooth, opaque, stationary object (can be non-convex)
- Parallel projection
- Image features used: occluding contour only
- Observer moves on a sphere around object
- Angular changes in viewpoint known
Overview

- Basic steps.
  1. Select a point on the visible rim
     (elliptic or hyperbolic)

2. Change position to recover the local surface
   shape at that point
3. Select a new point for shape-recovery

- Special case: Surfaces of revolution

Normal Sections

Principal curvatures = Normal curvatures along $\epsilon_1, \epsilon_2$

Elliptic  Hyperbolic  Parabolic  Planar
Shape from Occluding Contour

Relation between the occluding contour curvature and local surface shape [Blaschke]:

\[ k_n^{-1} = k_1^{-1} \cos^2 \phi + k_2^{-1} \sin^2 \phi \]

Implications:
1. \( k_n = k_1 \) if the viewing direction is along \( e_2 \)
2. If \( k_n, k_1, \phi \) are known we can find \( k_2 \)
3. \( k_n(\phi) \) has only two maxima and two minima, along \( e_2 \) and \( e_1 \) respectively

The Shape-Recovery Algorithm

1. Compute \( k_2 \) for the selected point at initial viewpoint

2. Compute point's tangent plane

3. Determine the direction of increasing \( k_2 \) on point's tangent plane

4. Move in that direction until \( k_2 \) is maximized
   Now, \( k_2 = k_1 \)

5. Measure the angle \( \phi \) between the initial and current viewing direction

6. Compute \( k_2 \) from \( \phi, k_1, \) and the initial value of \( k_2 \)
Hyperbolic Points

Surfaces of Revolution

Local shape reveals global surface properties

- One of the principal directions corresponds to a side view
Results

Candlestick View 1 (arbitrary)

<table>
<thead>
<tr>
<th>Degrees per Frame</th>
<th>Viewpoint (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>0.785398</td>
</tr>
</tbody>
</table>

Results

Candlestick View 2 (curvature maximum)

<table>
<thead>
<tr>
<th>Degrees per Frame</th>
<th>Viewpoint (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>1.570796</td>
</tr>
</tbody>
</table>

Points:
- 6(0.574543)
- 4(0.094467)
- 5(0.315911)
- 1(0.129635)
- 2(0.462817)
Results

Tori View 1 (arbitrary)

<table>
<thead>
<tr>
<th>Degrees per Frame</th>
<th>Viewpoint (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>0.498622</td>
</tr>
</tbody>
</table>

Results

Tori View 2 (curvature maximum)

<table>
<thead>
<tr>
<th>Degrees per Frame</th>
<th>Viewpoint (Radians)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00000</td>
<td>1.5938</td>
</tr>
</tbody>
</table>
Curvature Variation with Viewpoint

- Candlestick:

![Graph: Curvature Variation with Viewpoint]

- Tori:

![Graph: Curvature Variation with Viewpoint]

- Curvature variations decrease when \( k_2 \rightarrow k_1 \)

Results

![Images: Results]
Results

Curvature variation with viewpoint:

Results

Curvature variation with viewpoint:
Successes and Contributions

Our active approach has a number of features:

- Recovers principal curvatures and principal directions
- Qualitative motion control
- Visual processing consists of curvature measurements on the occluding contour
- Recovers correct axis and generating curve of surfaces of revolution

What is the role of special views in an active context?

Long version of this paper available via ftp at ftp.cs.wisc.edu (Technical Report #1035)