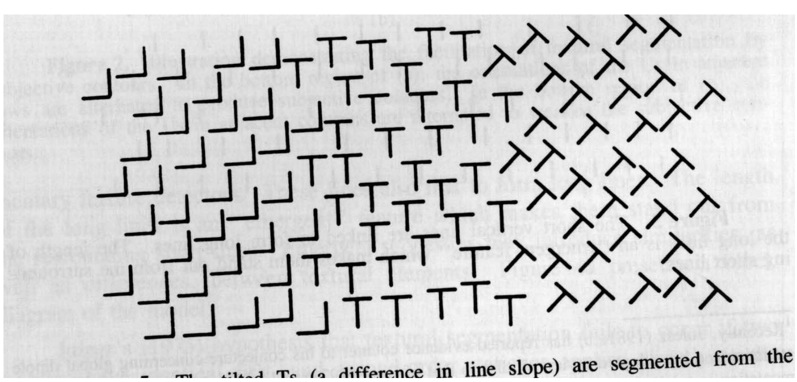
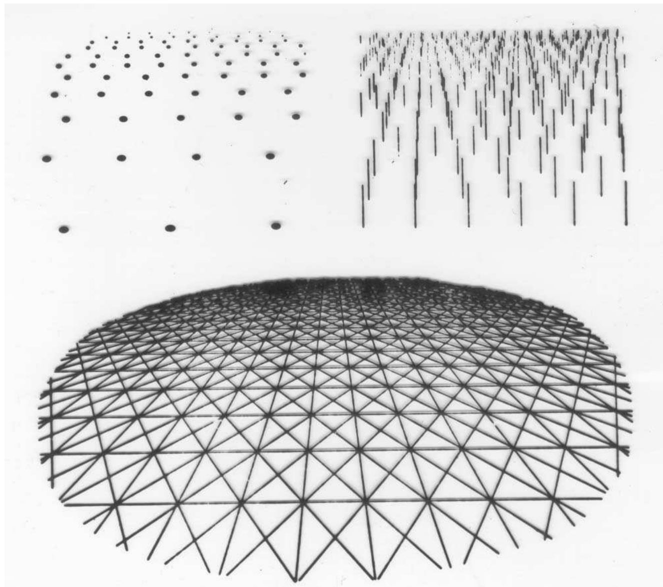
Texture

- 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

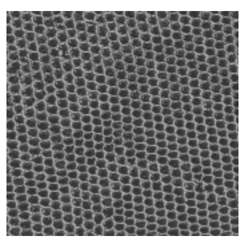
Texture Discrimination

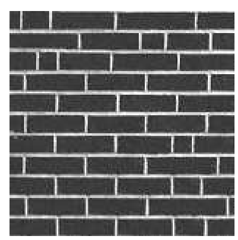


Shape from Texture



Modeling Texture





- What is texture?
  - An image obeying some statistical properties
  - Similar structures repeated over and over again
  - Often has some degree of randomness

Malik & Perona's Filters

Gabor filter kernels — product of symmetric Gaussian with oriented sinusoid

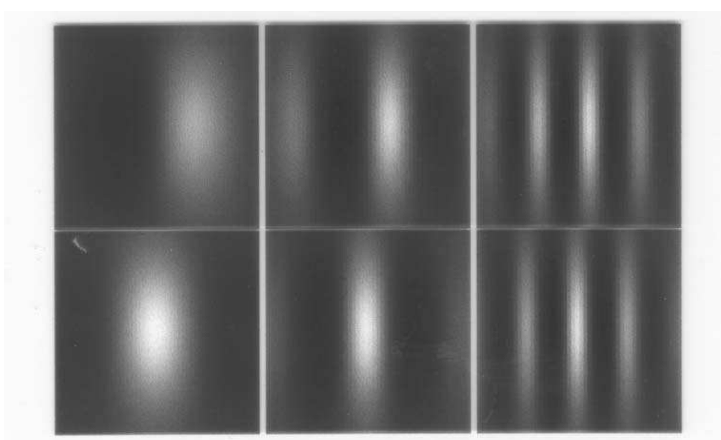
⇒ smoothed derivative kernels

DOOG filters — difference of oriented Gaussians

+ DOG filters

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

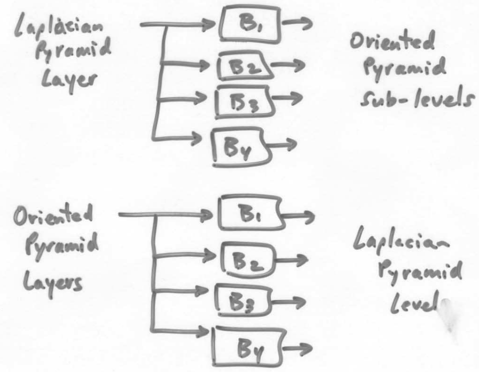
$$\begin{cases} G_{symmetric}(x,y) = \cos(k_x x + k_y y)\,\exp-\left\{\frac{x^2+y^2}{2\sigma^2}\right\} \\ G_{antisym}(x,y) = \sin(k_0 x + k_1 y)\,\exp-\left\{\frac{x^2+y^2}{2\sigma^2}\right\} \end{cases}$$



Steerable (i.e., Oriented) Pyramids

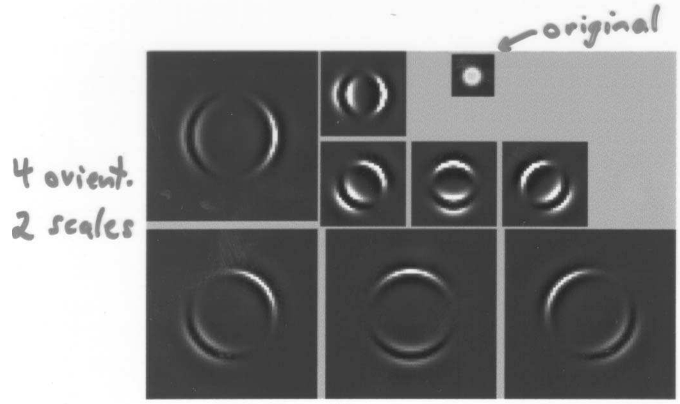
- Laplacian pyramid not appropriate for texture analysis because it does **not** encode orientation info.

⇒ ~~Oriented~~ Steerable pyramids (Simoncelli)

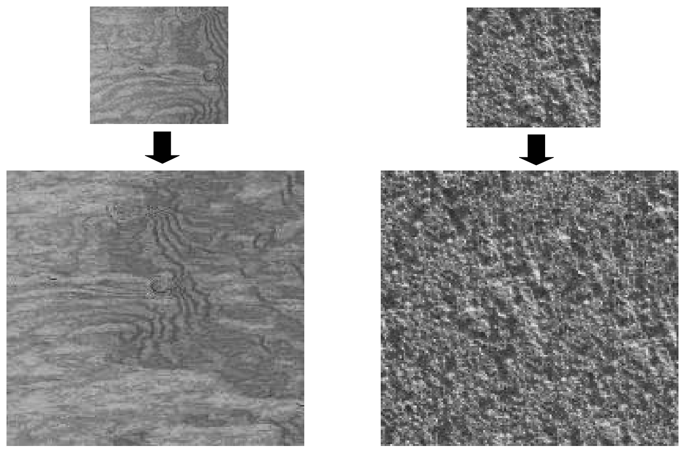


Steerable Pyramids

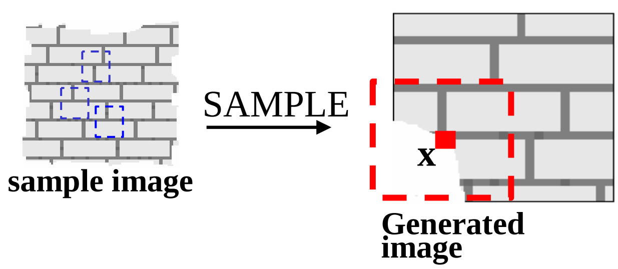
- Multiresolution, multi-orienatation image decomporition



Texture Synthesis [Efros & Leung, ICCV 99]



Synthesizing One Pixel



- What is $P(\mathbf{x}|\text{neighborhood of pixels around x})$
- 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

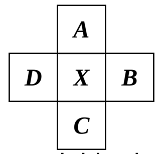
A Markov random field (MRF)

- generalization of Markov chains to two or more dimensions

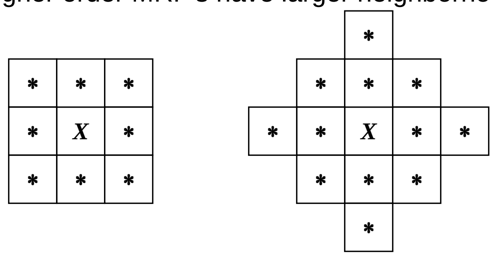
First-order MRF:

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

$$P(\mathbf{X}|\mathbf{A},\mathbf{B},\mathbf{C},\mathbf{D})$$

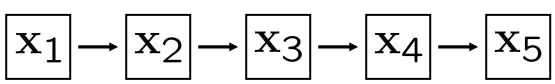


- Higher order MRF's have larger neighborhoods



Markov Chain

- Markov Chain
  - a *sequence* of random variables $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$
  - $\mathbf{x}_t$ is the **state** of the model at time t

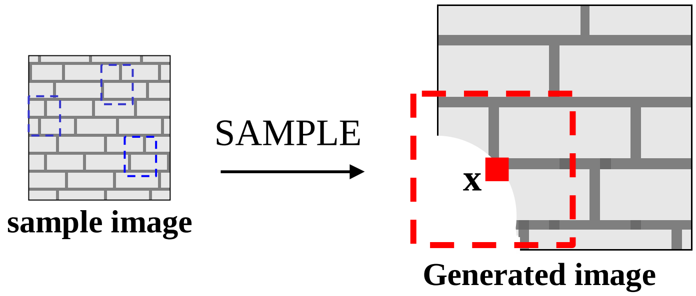


  - **Markov assumption**: each state is dependent only on the previous one
    - dependency given by a **conditional probability**:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1})$$

  - The above is actually a *first-order* Markov chain
  - An *N'th-order* Markov chain: $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \ldots, \mathbf{x}_{t-N})$

Really Synthesizing One Pixel

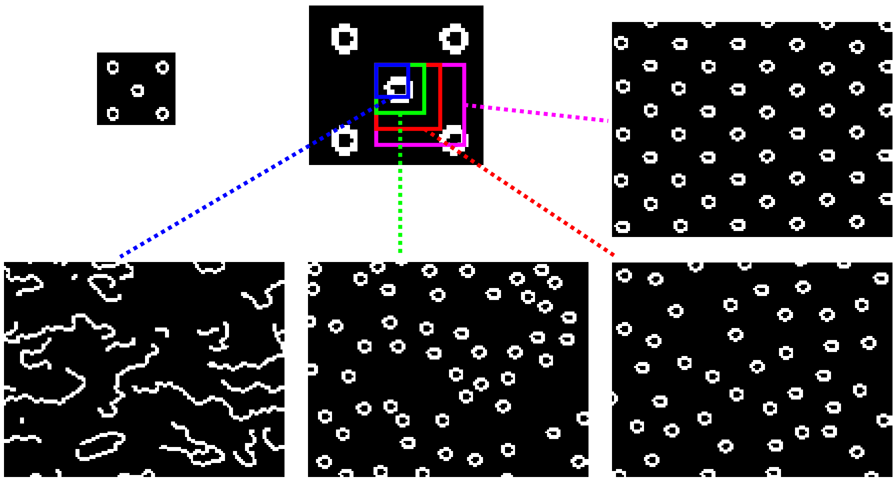


– An exact neighborhood match might not be present
– So we find the **best** matches using SSD error and randomly choose between them, preferring better matches with higher probability

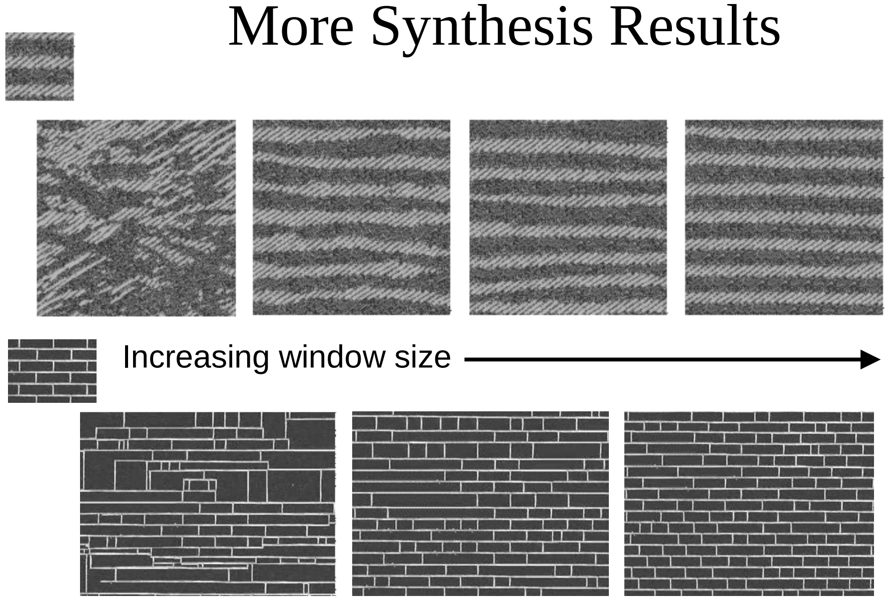
Growing Texture

– Starting from the initial image, "grow" the texture one pixel at a time

Window Size Controls Regularity

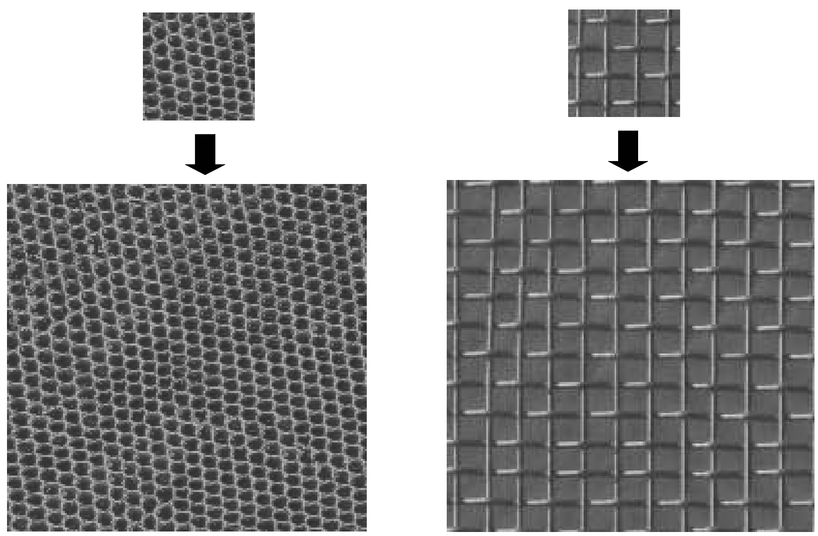


More Synthesis Results

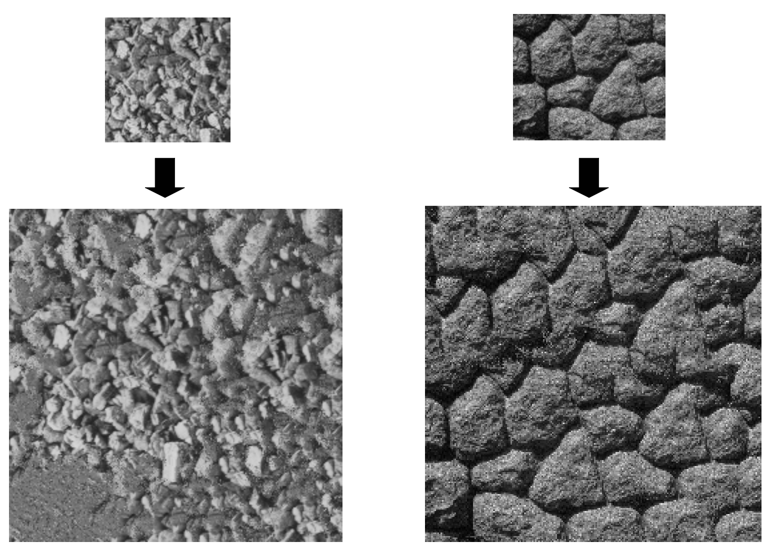


More Results

reptile skin   aluminum wire

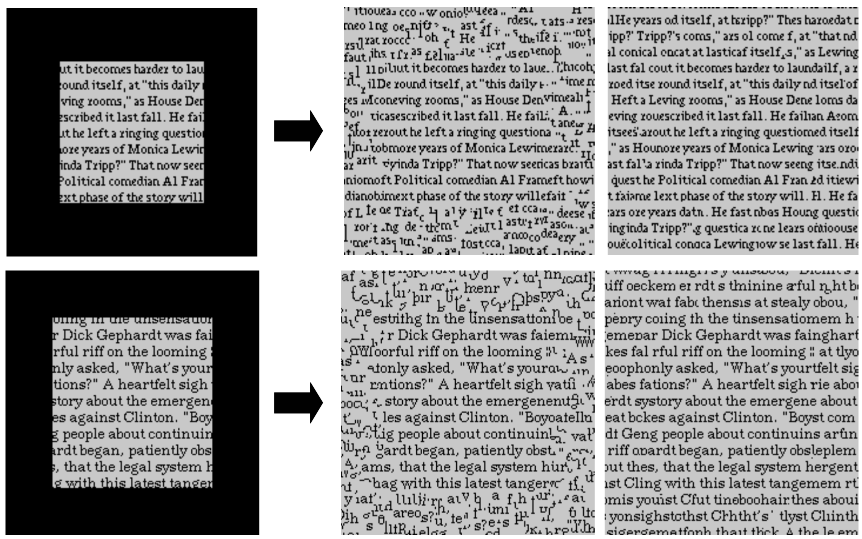


Failure Cases

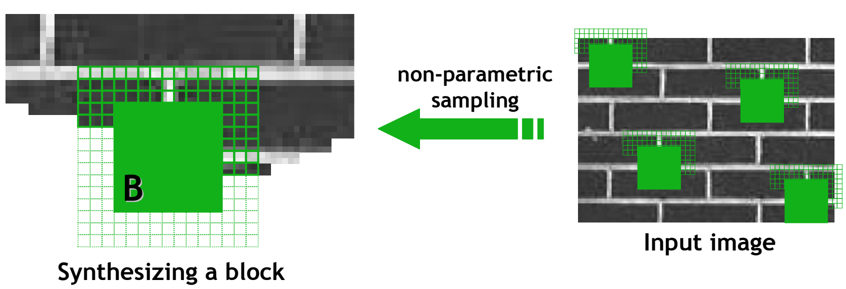


Growing garbage   **Verbatim copying**

Image-Based Text Synthesis



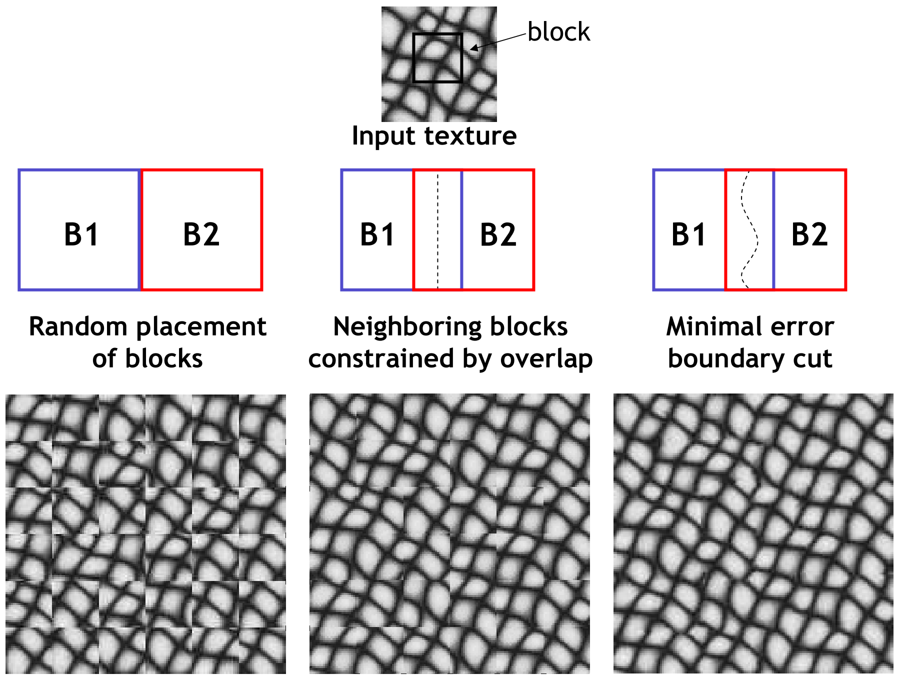
Efros & Leung ’99 Extended



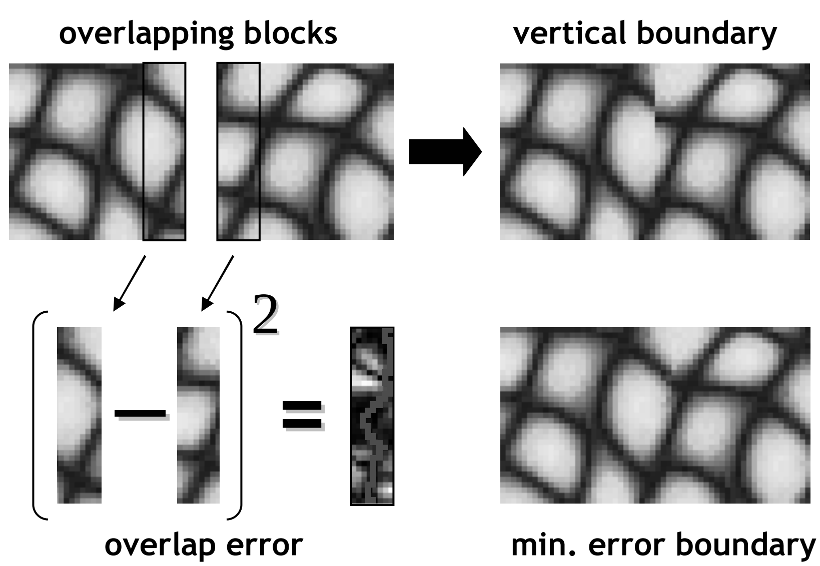
- Observation: neighbor pixels are highly correlated

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



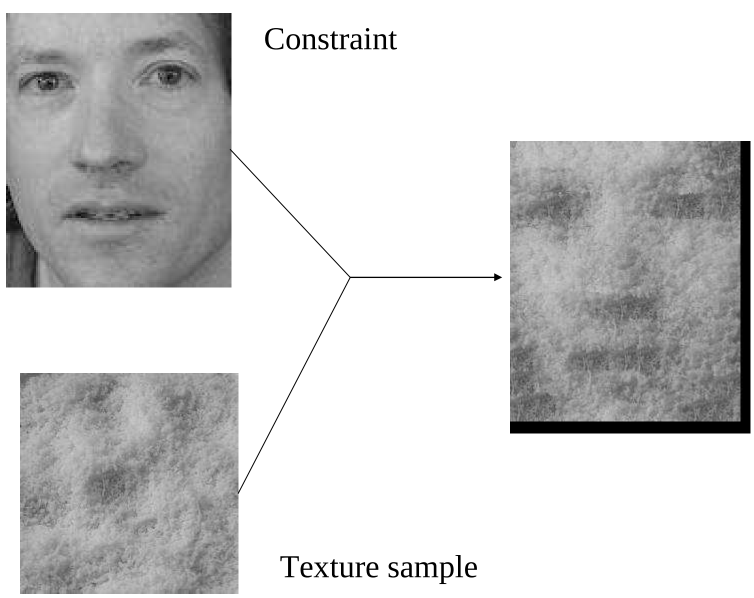
Minimal error boundary



Philosophy

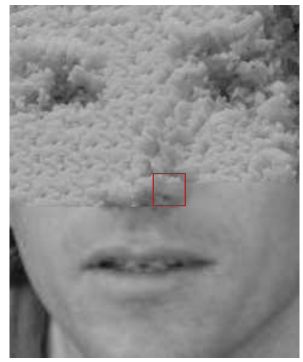
- The “Corrupt Professor’s Algorithm:”
  - Plagiarize as much of the source image as you can
  - Then try to cover up the evidence
- Rationale:
  - Texture blocks are by definition correct samples of texture, so the only problem is connecting them together

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

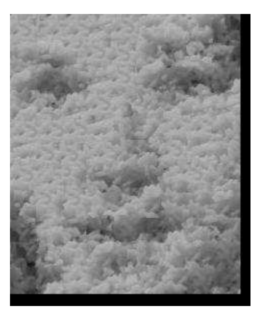


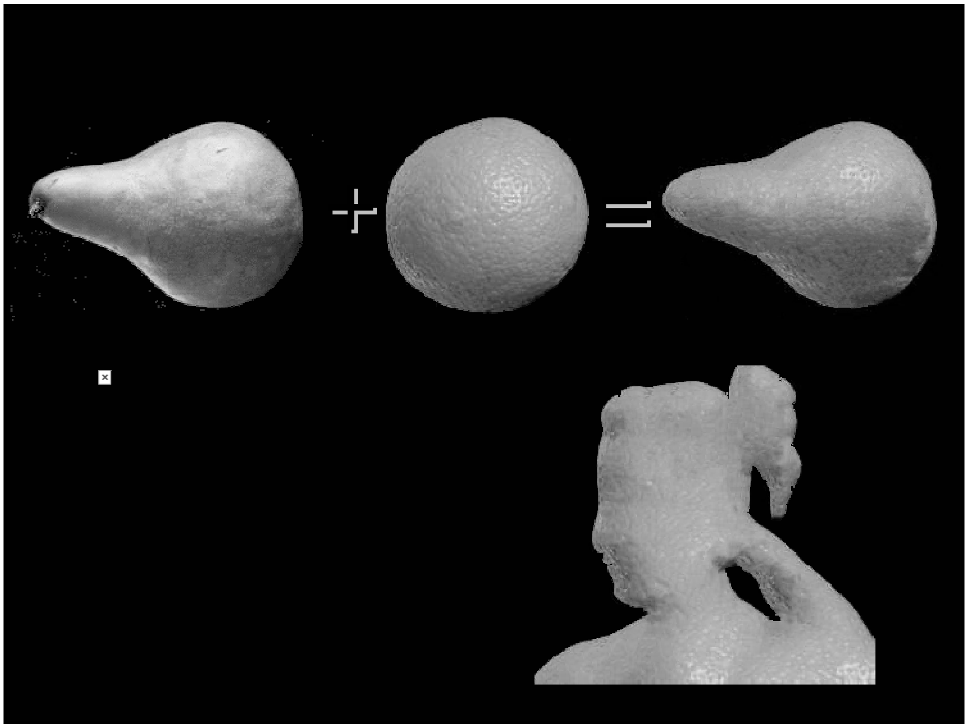
Then, just add another constraint when sampling: similarity to luminance of underlying image at that spot

parmesan

 +  = 

rice

 +  = 



Shape from Texture

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

- Witkin's Method
  - No model for texels
  - Assume natural textures do <u>not</u> mimic projection effects
  - Isotropy Assumption: All surface orientations equally likely <u>and</u> all edge orientations equally likely

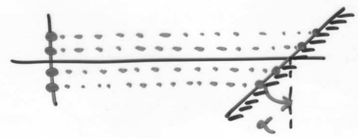
⇒ 1) Model geometric distortion on edge orientations

2) Statistically model distribution of surface orientations and surface marking orientations

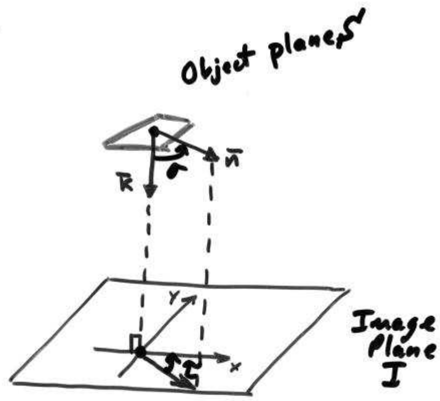
Imaging process distorts surface texture in 2 ways:

1. Distance
   Farther objects appear smaller
   Area subtended by solid angle Ω
   $= d^2\Omega$ where $d =$ distance to surface

2. Orientation



As slant α increases, foreshortening makes point appear closer together as $\frac{1}{\cos\alpha}$



Goal: Given image of a planar surface, recover (σ, τ)

$0 \le \sigma \le \frac{\pi}{2}$

$-\frac{\pi}{2} \le \tau \le \frac{\pi}{2}$

- 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

β = angle between $\beta(s)$ and x-axis is S

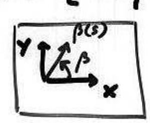
What is projection of $\beta(s)$, β, $C(s)$ into image I?

Assume orthographic projection
(⇒ 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 (σ, τ) (I's x-axis = tilt direc)
   Initially, assume $\tau = 0$
   $\Rightarrow (x, y, 0) \xrightarrow{\sigma, 0} (x\cos\sigma, y, x\sin\sigma)$

3. Orthographic Projection onto I plane
   $\Rightarrow (x, y, z) \longrightarrow (x, y)$

Combining 3 steps:

Edge orientation $[\cos\beta, \sin\beta]$ in S

projects to $[\cos\beta \cos\sigma, \sin\beta]$ in I

$$\Rightarrow \tan\beta^* = \frac{\sin\beta}{\cos\beta \cos\sigma} = \frac{\tan\beta}{\cos\sigma}$$

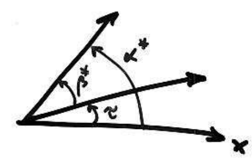
$$\Rightarrow \beta^* = \tan^{-1}\left(\frac{\tan\beta}{\cos\sigma}\right)$$

= angle b/w observed edge orient. and tilt direc (= x-axis)

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

$$\boxed{\alpha^* = \tan^{-1}\left(\frac{\tan\beta}{\cos\sigma}\right) + \tau}$$

where α^* = observed edge orientation wrt I's x-axis



- Isotropy Assumption ⇒ all surface orientations equally likely [redacted] and all edge orientations equally likely

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

likelihood of $\beta = \beta_0$ in $[0, \pi]$; likelihood of $\tau = \tau_0$ in $[0, \pi]$; likelihood of $\sigma = \sigma_0$

$$= \frac{\sin\sigma}{\pi^2}$$

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

$$= \frac{1}{\pi} \frac{\cos\sigma}{\cos^2(\alpha^* - \tau) + \sin^2(\alpha^* - \tau)\cos^2\sigma}$$

i.e. says what histogram should look like for given (σ, τ)!

Assuming edge orientations in I are **independent** $\Rightarrow$

$$\text{p.d.f.}\left(A^* = \{\alpha_1^*, \ldots, \alpha_n^*\} \mid \sigma, \tau\right) = \prod_{i=1}^{n} \text{p.d.f.}\left(\alpha_i^* \mid \sigma, \tau\right)$$

By Bayes Rule:

$$\text{p.d.f.}\left(\sigma, \tau \mid A^*\right) = \prod_{i=1}^{n} \frac{\pi^{-2} \sin\sigma \cos\sigma}{\cos^2(\alpha_i^* - \tau) + \sin^2(\alpha_i^* - \tau)\cos^2\sigma}$$

Maximum likelihood estimator for (σ, τ) $\Rightarrow$

1. Apply Edge operator
2. Compute edge orientation histogram A^*
3. For each (σ, τ) compute p.d.f. $(\sigma, \tau \mid A^*)$
4. Select (σ, τ) which is **maximum**.

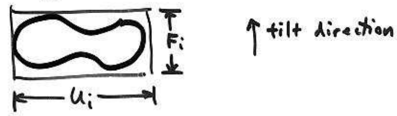
Shape from Texture Using Texels

- Projective distortion changes **size** of texels due to **distance**, and **shape** of texels due to **foreshortening**

1. Detect Texels
   - $\nabla^2 G_\sigma$ detectors find centers of "spots" of varying sizes
   - Connected components analysis used to define texels
2. Estimate single planar surface that is maximally consistent with texels "painted" on surface (no micro-relief)

Heuristic: **Texel-area gradient:**
Area of texels decreases w/ distance and **slant** angle.
Fastest in direction of **tilt**

- Approximate texel area using bounding box:



$$A_i \approx u_i F_i$$

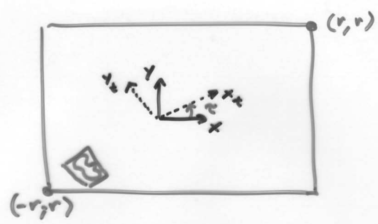
- Relate image texel area, A_i, to physical texel area, scene plane orientation, (σ, τ), and angle to texel from optical axis, θ

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

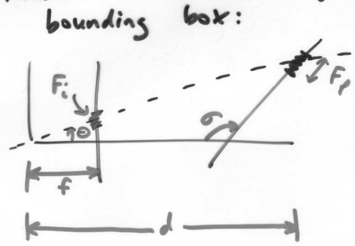
where A_c = area of texel at image center

$$\theta = \tan^{-1}\left((x \cos\tau + y \sin\tau)(r/f)\right)$$

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



- Relate to dimensions of texel's bounding box:



$$\theta = \tan^{-1}\left(x_t (r/f)\right) \qquad r = \text{image radius}$$

$$= \tan^{-1}\left((x\cos\tau + y \sin\tau)(r/f)\right)$$

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

$$= F_c \left(1 - \tan\theta \tan\sigma\right)^2$$

where F_c = foreshortened dimension of texel at <u>image center</u>

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, σ, τ) space to maximize fit with observed texel areas, A_i's.
- Discretize A_c, σ, τ values
- Use coarse-to-fine search

```
for A_c = minAc to maxAc do
  for σ = 0 to π/2 do
    for τ = 0 to π do
      fit := 0
      for t=1 to num-texels do
        Actual-area := area(t)
        expected-area := A_c (1 - tanθ tanσ)^3
        region-fit := max(expected, actual) / min(expected, actual)

      fit := fit +
          (actual-area *
              |texel-contrast| *
                  exp - { region-fit^2 / 4 }
      od
      if fit > best-fit
         then best-fit := fit
    od
  od
od
```

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

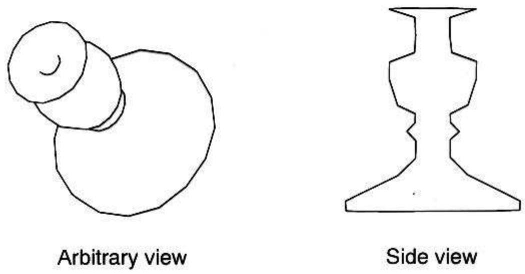
- Our approach

  Control position relative to the surface

  - Maintain fixation
  - Measure relative viewing direction changes
  - Compute occluding contour curvatures

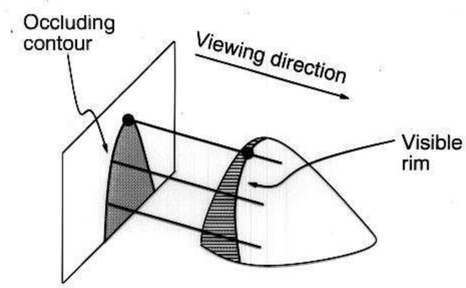
Motivation for the Approach

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



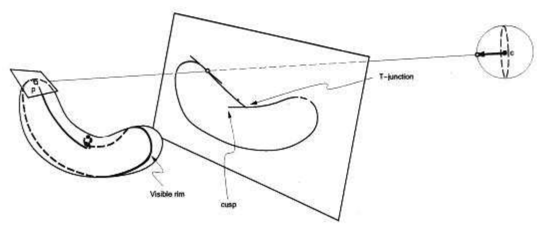
- Recover
  1. Principal directions
  2. Principal curvatures
- Assume orthographic projection

Occluding Contour

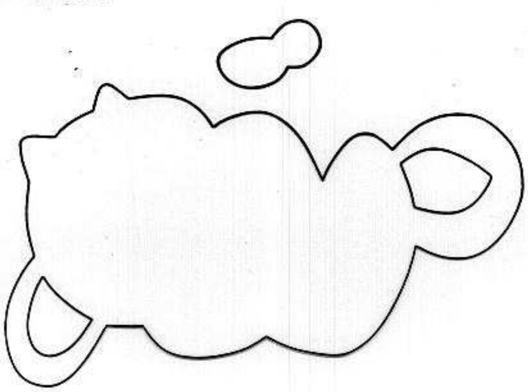
Rim: Points where surface tangent plane contains the visual ray, $\vec{v}$: $\vec{v} \cdot \vec{n} = 0$ (aka Limb, Contour Generator)

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

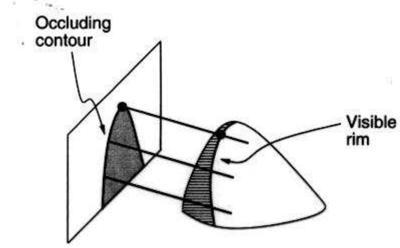


Silhouette



~~Silhouette~~ **Occluding Contour**

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

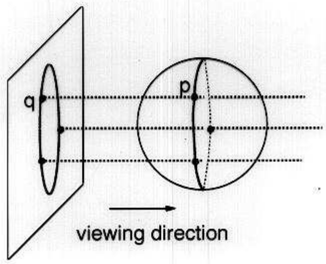
- Geometry is surface-dependent
- Projection of a limited set of surface points
- Geometry is viewpoint-dependent

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

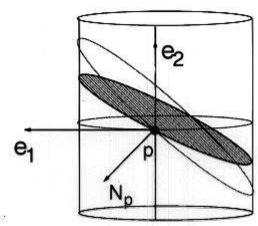
Overview

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

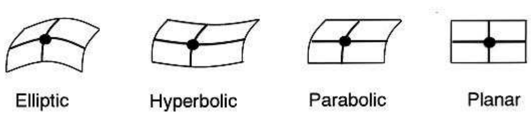


  2. Change position to recover the local surface shape at that point
  3. Select a new point for shape-recovery
- Special case: Surfaces of revolution

Normal Sections



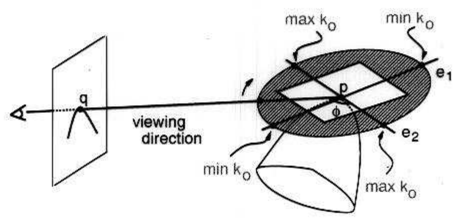
Principal curvatures = Normal curvatures along e_1, e_2



Shape from Occluding Contour

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

$$k_o^{-1} = k_1^{-1} \cos^2 \phi + k_2^{-1} \sin^2 \phi$$



Implications.

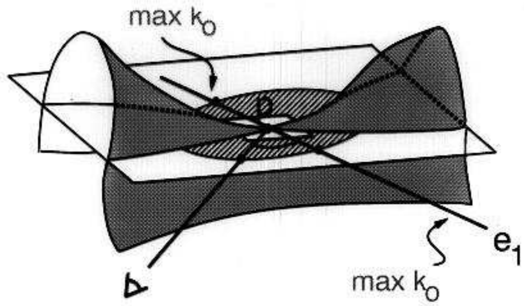
1. $k_o = k_1$ if the viewing direction is along e_2
2. If k_o, k_1, ϕ are known we can find k_2
3. $k_o(\phi)$ has only two maxima and two minima, along e_2 and e_1 respectively

The Shape-Recovery Algorithm

1. Compute k_o for the selected point at initial viewpoint
2. Compute point's tangent plane
3. Determine the direction of increasing k_o on point's tangent plane
4. Move in that direction until k_o is maximized

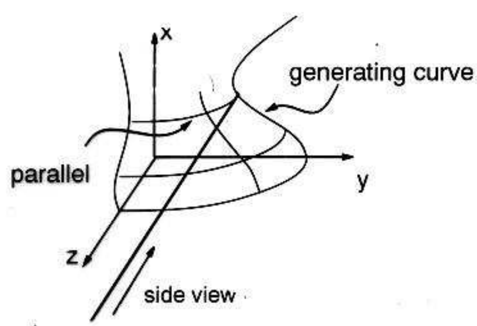
   Now, $k_o = k_1$
5. Measure the angle ϕ between the initial and current viewing direction
6. Compute k_2 from ϕ, k_1, and the initial value of k_o

Hyperbolic Points



Surfaces of Revolution

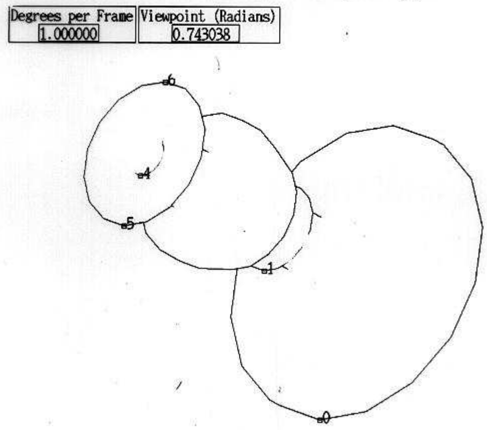
Local shape reveals global surface properties



- One of the principal directions corresponds to a side view

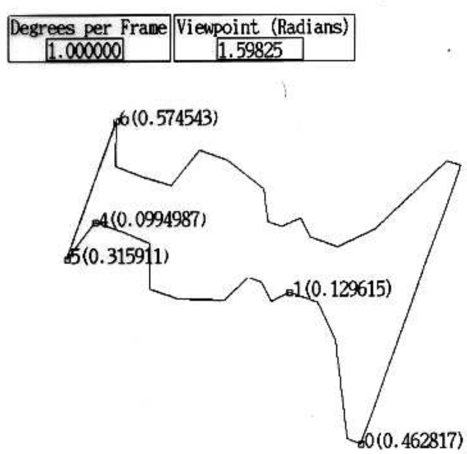
Results

Candlestick View 1 (arbitrary)



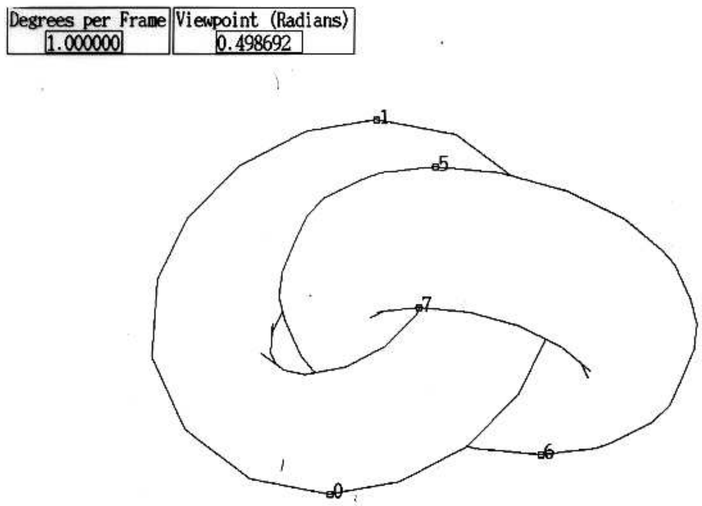
Results

Candlestick View 2 (curvature maximum)



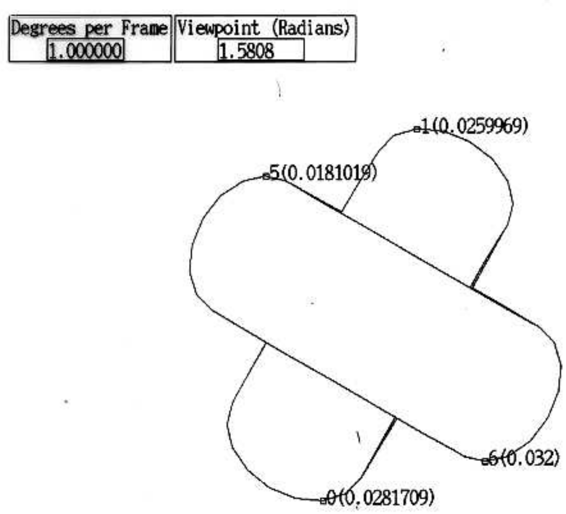
Results

Tori View 1 (arbitrary)



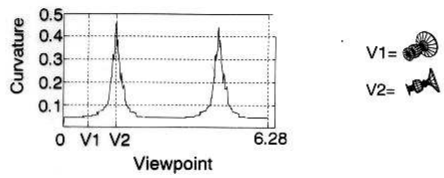
Results

Tori View 2 (curvature maximum)

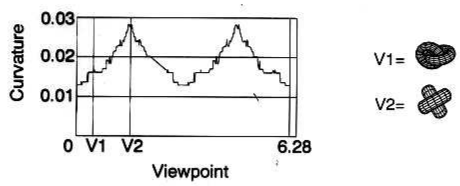


Curvature Variation with Viewpoint

- Candlestick:

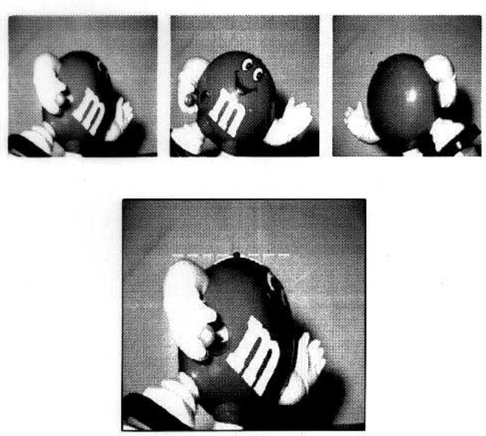


- Tori

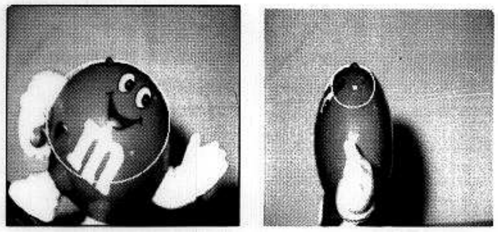


- Curvature variations decrease when $k_2 \rightarrow k_1$

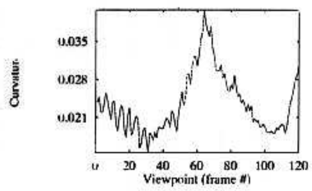
Results



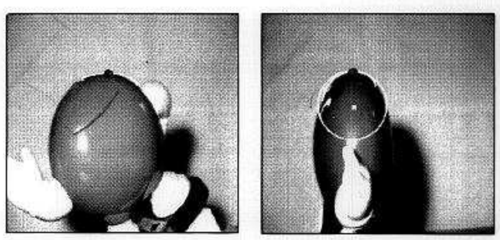
Results



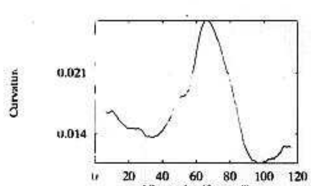
Curvature variation with viewpoint:



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)