اصول پردازش تصویر

Principles of Image Processing

مصطفی کمالی تبریزی

۲۲ آذر ۱۳۹۹

جلسه بیست چهارم

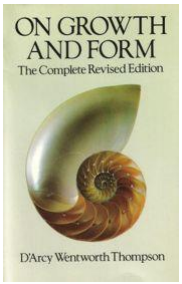
Image Morphing

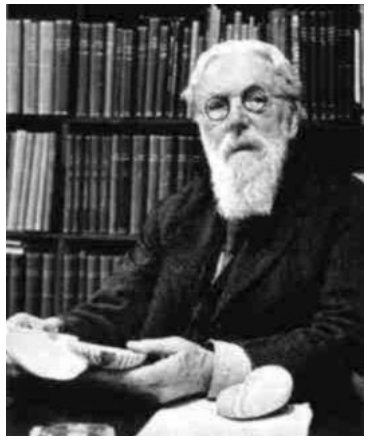
Many slides adopted from Derek Hoiem

نتیجه تمرین آقای احمد رحیمی در سال ۱۳۹۸

Image Warping in Biology

- D'Arcy Thompson
- http://www-groups.dcs.st-and.ac.uk/~history/Miscellaneous/darcy.html
- http://en.wikipedia.org/wiki/D'Arcy_Thompson
- Importance of shape and structure in evolution





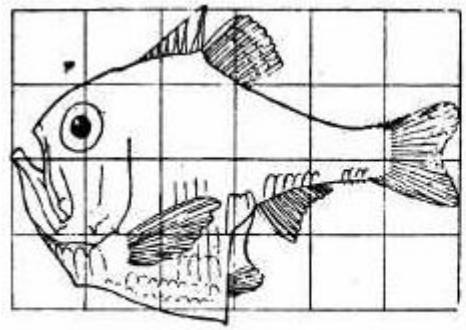


Fig. 517. *Argyropelecus Olfersi.*

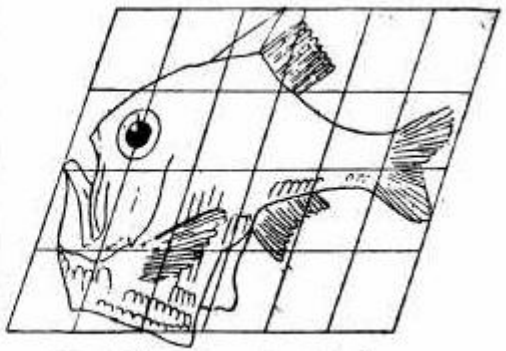
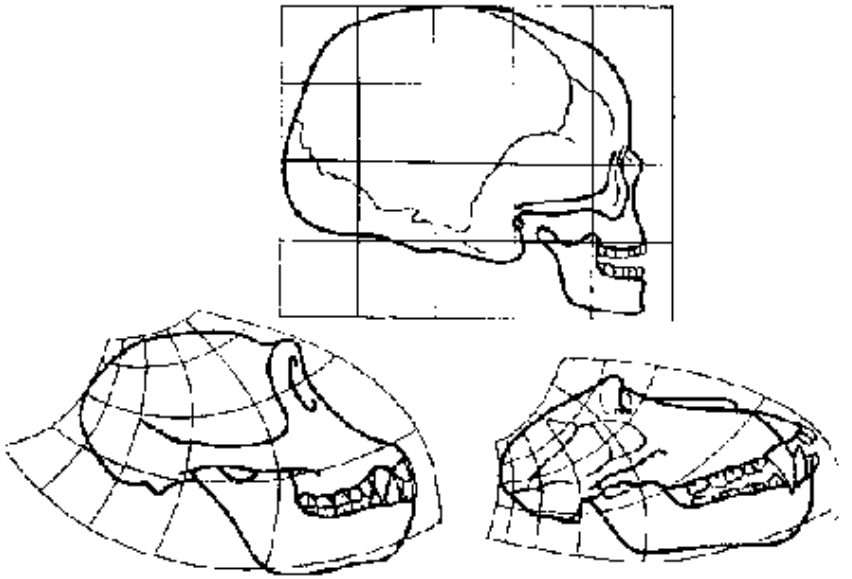


Fig. 518. *Sternoptyx diaphana.*



Skulls of a human, a chimpanzee and a baboon and transformations between them

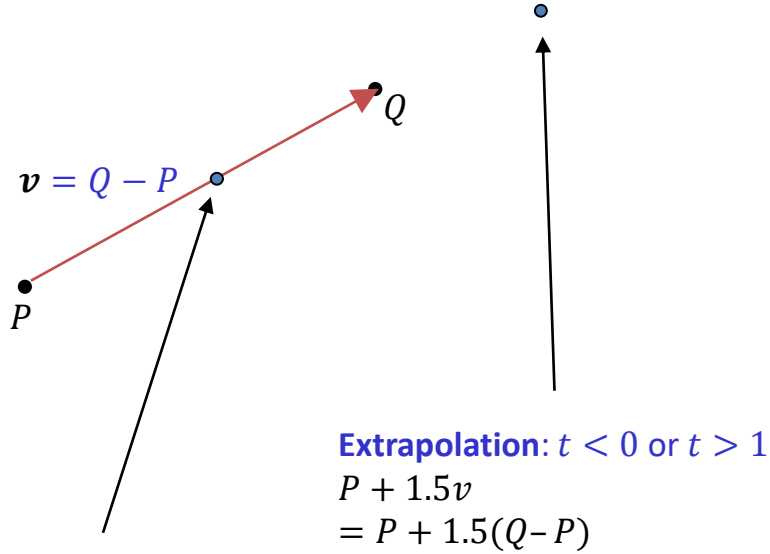
Slide by Durand and Freeman

Morphing = Object Averaging

- The aim is to find “an average” between two objects
  - Not an average of two images of objects …
  - … but an image of the average object!
  - How can we make a smooth transition in time?
    - Do a “weighted average” over time t

Averaging Points

What's the average of P and Q?



Extrapolation: $t < 0$ or $t > 1$

$P + 1.5v$
$= P + 1.5(Q - P)$
$= -0.5P + 1.5Q$ $(t = 1.5)$

Linear Interpolation

New point: $P + t * (Q - P)$

Or equivalently: $(1 - t)P + tQ$ $(0 < t < 1)$

$P + 0.5v$
$= P + 0.5(Q - P)$
$= 0.5P + 0.5Q$

- P and Q can be anything:
  - points on a plane $(2D)$ or in space $(3D)$
  - Colors in RGB $(3D)$
  - Whole images (m-by-n D)… etc.

Idea #1: Cross-Dissolve

- Interpolate whole images:
- $I_{halfway} = (1-t)\, I_1 + t\, I_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #1: Cross−Dissolve

- Interpolate whole images:
- $I_{halfway} = (1-t)\, I_1 + t\, I_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #1: Cross−Dissolve

- Interpolate whole images:
- $I_{halfway} = (1 - t)\, I_1 + t\, I_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #1: Cross−Dissolve

- Interpolate whole images:
- $I_{halfway} = (1 - t)\, I_1 + t\, I_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #1: Cross−Dissolve

- Interpolate whole images:
- $I_{halfway} = (1 - t)\, I_1 + t\, I_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #1: Cross−Dissolve

- Interpolate whole images:
- $I_{halfway} = (1-t)\, I_1 + t\, I_2$
- This is called **cross-dissolve** in film industry

- But what if the images are not aligned?

Idea #2: Align, then cross-disolve

- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

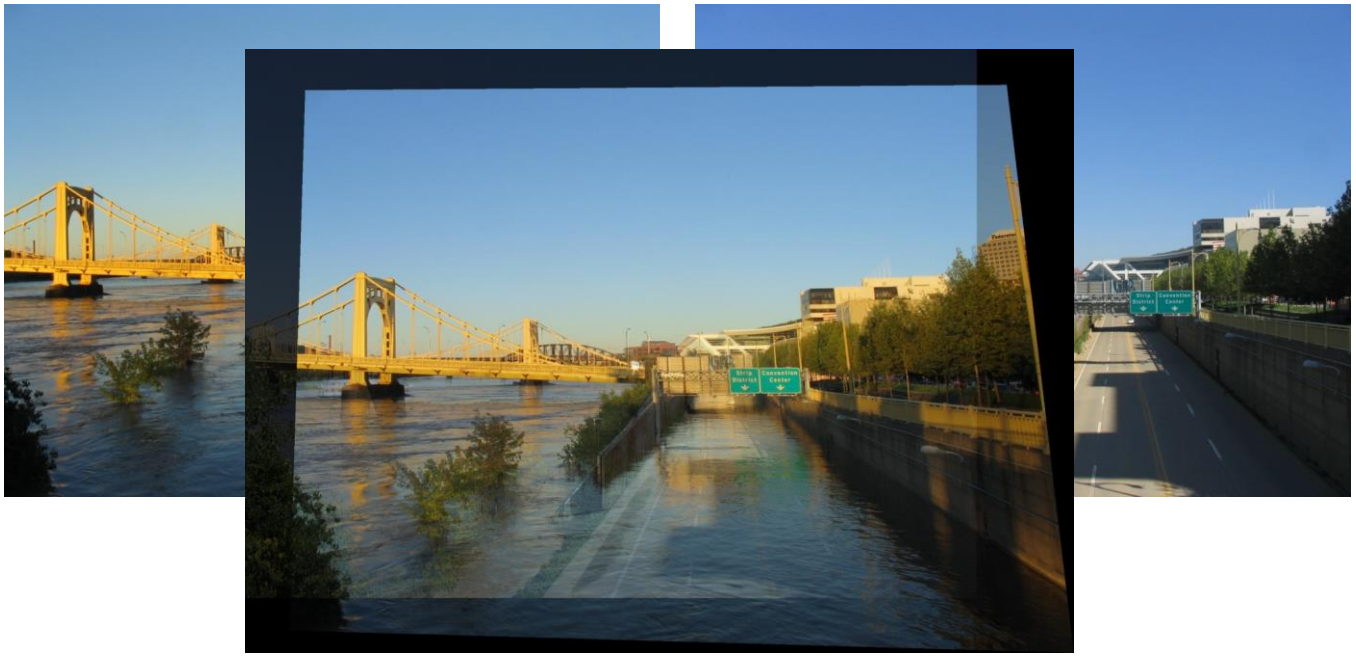
Image credit: Alyosha Efros

Idea #2: Align, then cross-disolve

- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Image credit: Alyosha Efros

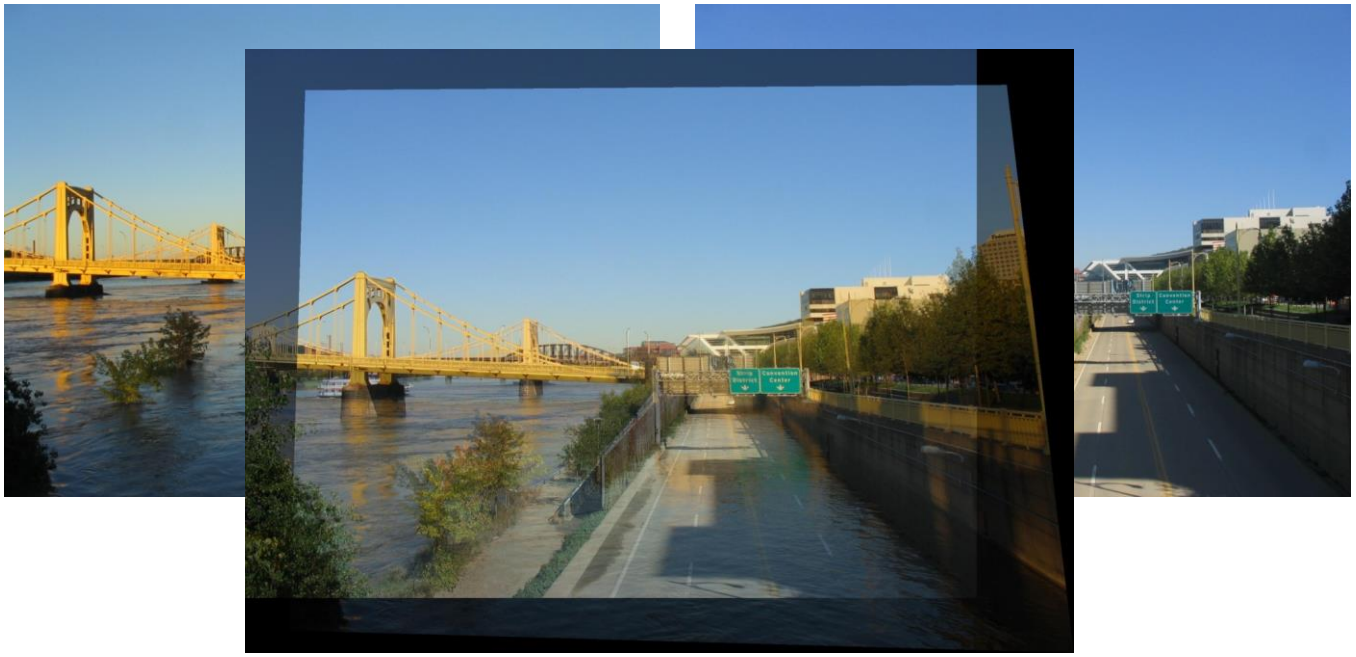
Idea #2: Align, then cross-disolve



- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Image credit: Alyosha Efros

Idea #2: Align, then cross-disolve



- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Image credit: Alyosha Efros

Idea #2: Align, then cross-disolve

- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Image credit: Alyosha Efros

Idea #2: Align, then cross-disolve

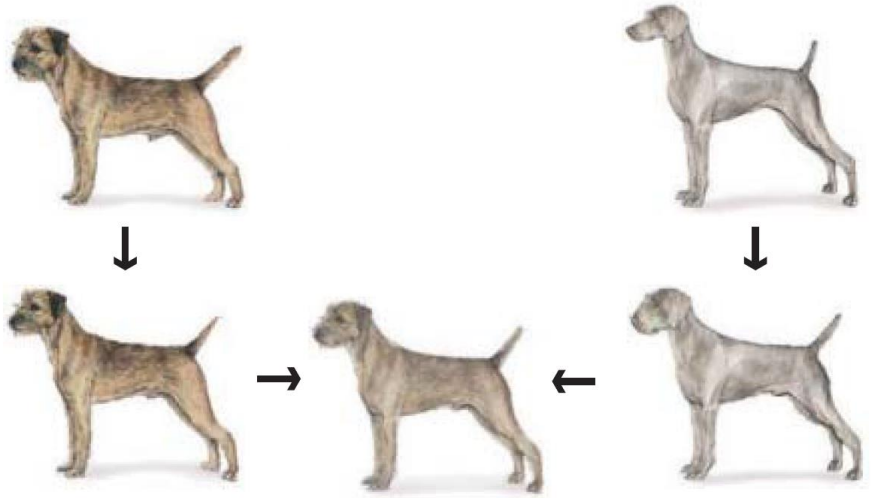
- Align first, then cross-dissolve
  - Alignment using global warp – picture still valid

Image credit: Alyosha Efros

Dog Averaging

- What to do?
  - Cross-dissolve doesn't work
  - Global alignment doesn't work
    - Cannot be done with a global transformation (e.g. affine)
  - Any ideas?
- Feature matching!
  - Nose to nose, tail to tail, etc.
  - This is a local (non-parametric) warp

Idea #3: Local warp, then cross-dissolve



- **Morphing procedure**
- *For every frame t*,

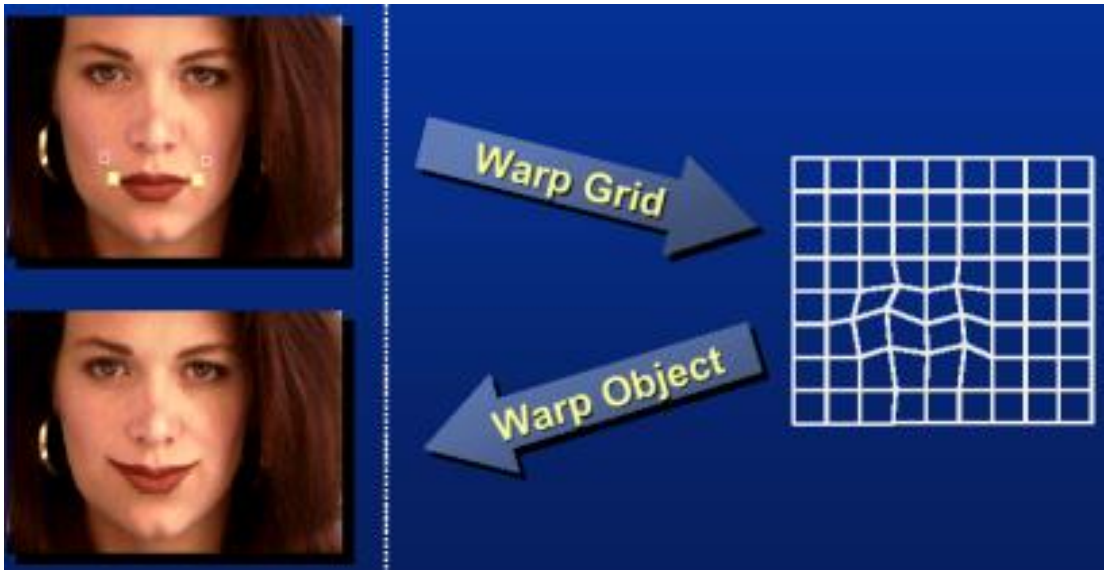
1. Find the average shape (the “mean dog”☺)
   - local warping
2. Find the average color
   - Cross-dissolve the warped images

Local (non-parametric) Image Warping

- Need to specify a more detailed warp function
  - Global warps were functions of a few (2, 4, 8) parameters
  - Non-parametric warps $u(x, y)$ and $v(x, y)$ can be defined independently for every single location (x, y)!
  - Once we know vector field (u, v) we can easily warp each pixel (use backward warping with interpolation)

Image Warping – non-parametric

- Move control points to specify a spline warp
- Spline produces a smooth vector field

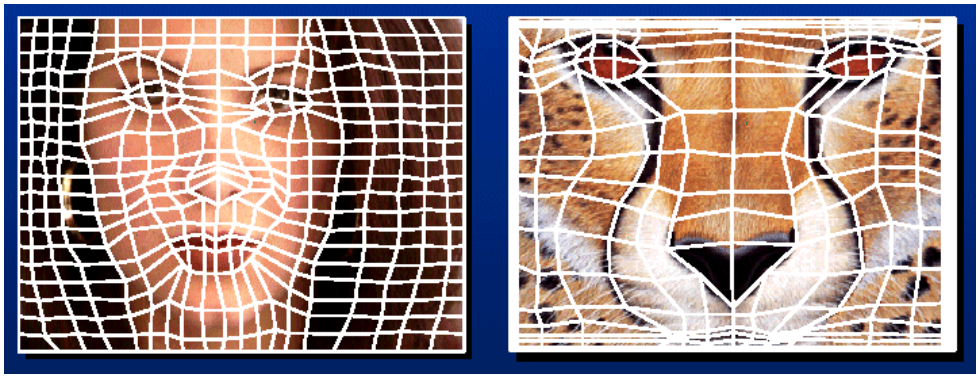


Warp specification - dense

- How can we specify the warp?

  Specify corresponding *spline control points*

  - *interpolate* to a complete warping function



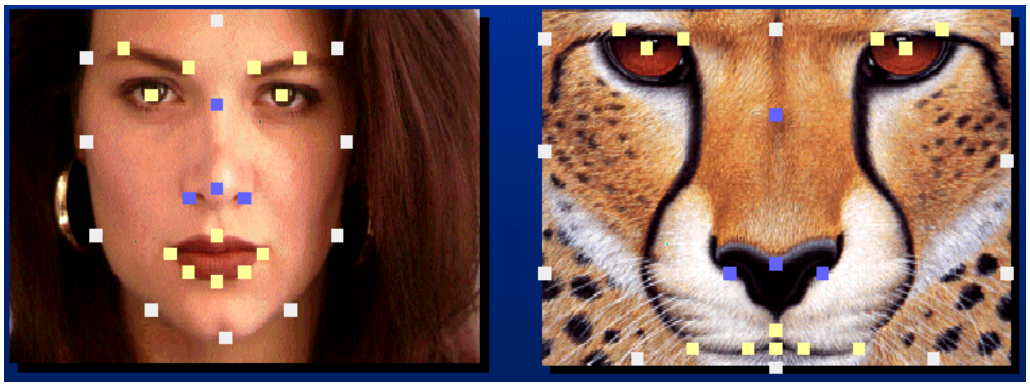
But we want to specify only a few points, not a grid

Warp specification - sparse

- How can we specify the warp?

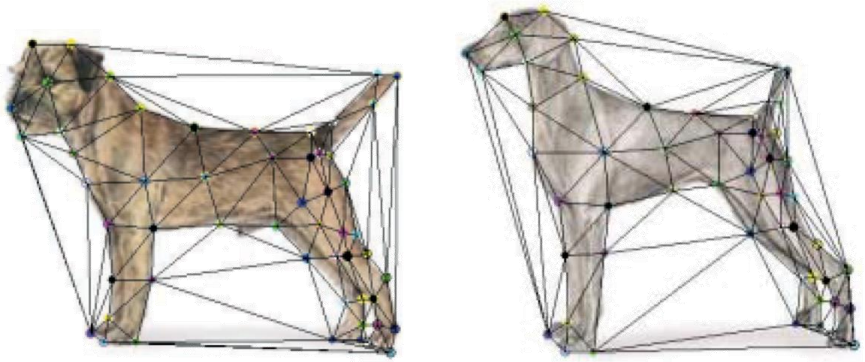
  Specify corresponding *points*

  - *interpolate* to a complete warping function
  - How do we do it?



How do we go from feature points to pixels?

Triangular Mesh

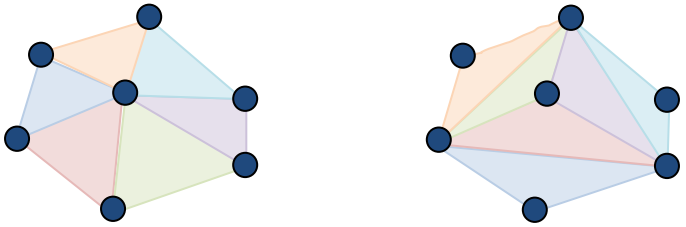


1. Input correspondences at key feature points
2. Define a triangular mesh over the points
   - Same mesh (triangulation) in both images!
   - Now we have triangle-to-triangle correspondences
3. Warp each triangle separately from source to destination
   - Affine warp with three corresponding points (just like take-home question)

Triangulations

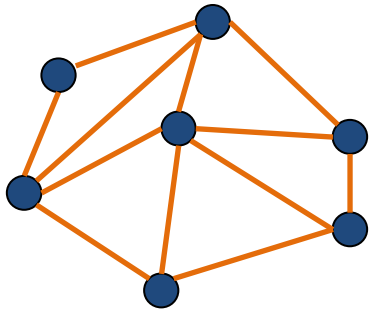
•A *triangulation* of set of points in the plane is a *partition* of the convex hull to triangles whose vertices are the points, and do not contain other points.

•There are an exponential number of triangulations of a point set.



An $O(n^3)$ Triangulation Algorithm

- Repeat until impossible:
  - Select two sites.
  - If the edge connecting them does not intersect previous edges, keep it.

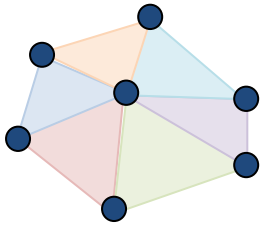


“Quality” Triangulations

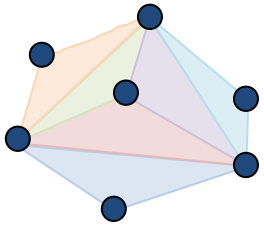
• Let $\alpha(T_i) = (\alpha_{i1}, \ \alpha_{i2}, .., \ \alpha_{i3})$ be the vector of angles in the triangulation T in increasing order:

– A triangulation T_1 is “better” than T_2 if the smallest angle of T_1 is larger than the smallest angle of T_2

– Delaunay triangulation is the “best” (maximizes the smallest angles)



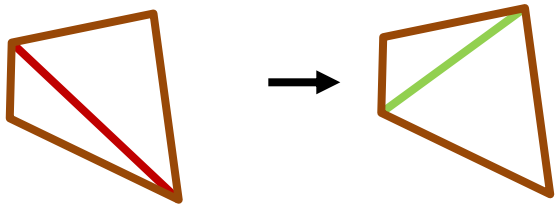
good



bad

Improving a Triangulation

In any convex quadrangle, an *edge flip* is possible. If this flip *improves* the triangulation locally, it also improves the global triangulation.

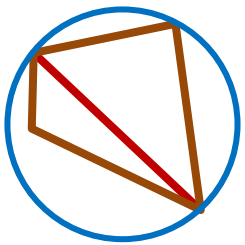


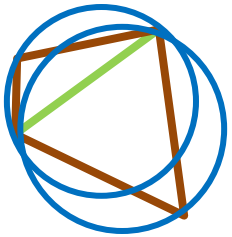
If an edge flip improves the triangulation, the first edge is called "*illegal*".

Illegal Edges

An edge pq is "illegal" iff one of its opposite vertices is inside the circle defined by the other three vertices (see Thale's theorem)

- A triangle is Delaunay iff no other points are inside the circle through the triangle's vertices
- The Delaunay triangulation is not unique if more than three nearby points are co-circular
- The Delaunay triangulation does not exist if three nearby points are colinear





Naïve Delaunay Algorithm

Start with an arbitrary triangulation.

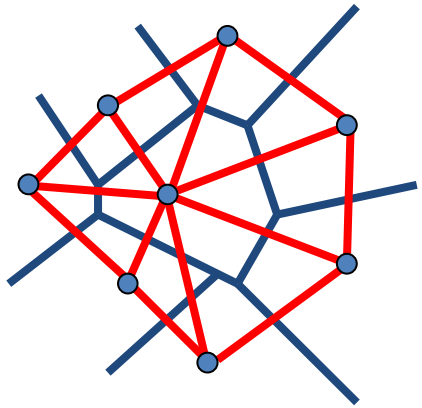
Flip any illegal edge until no more exist.

Could take a long time to terminate.

Delaunay Triangulation by Duality

Draw the dual to the Voronoi diagram by connecting each two neighboring sites in the Voronoi diagram.

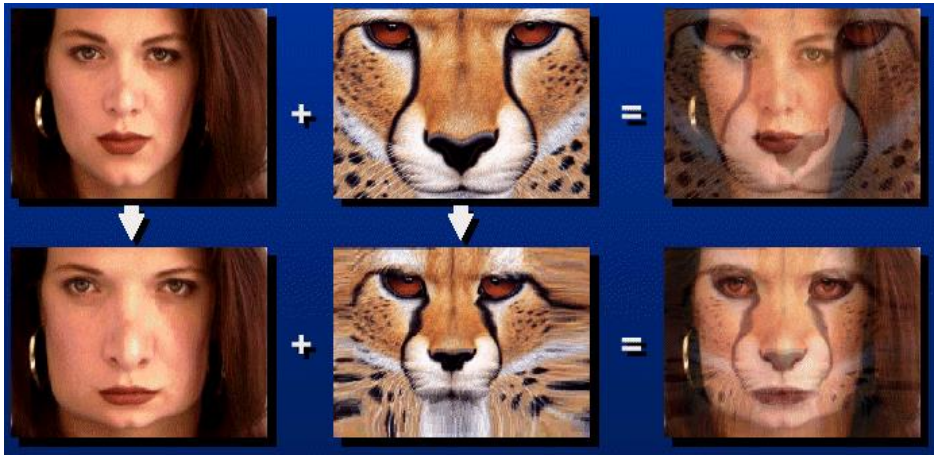
The DT may be constructed in $O(nlogn)$ time.



Demos:
http://www.cs.cornell.edu/home/chew/Delaunay.html
http://alexbeutel.com/webgl/voronoi.html

Image Morphing

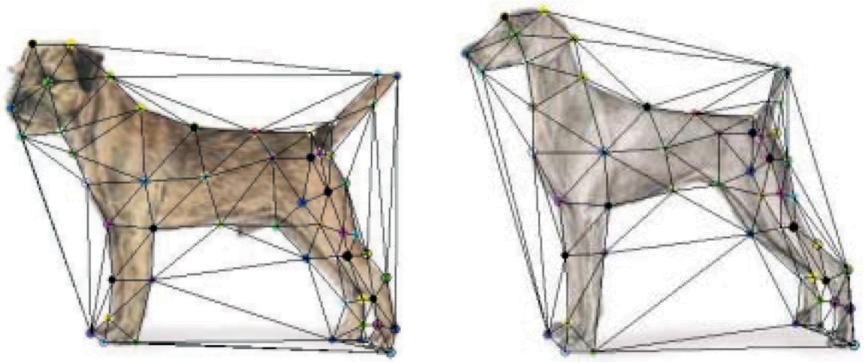
- How do we create a morphing sequence?
  1. Create an intermediate shape (by interpolation)
  2. Warp both images towards it
  3. Cross-dissolve the colors in the newly warped images



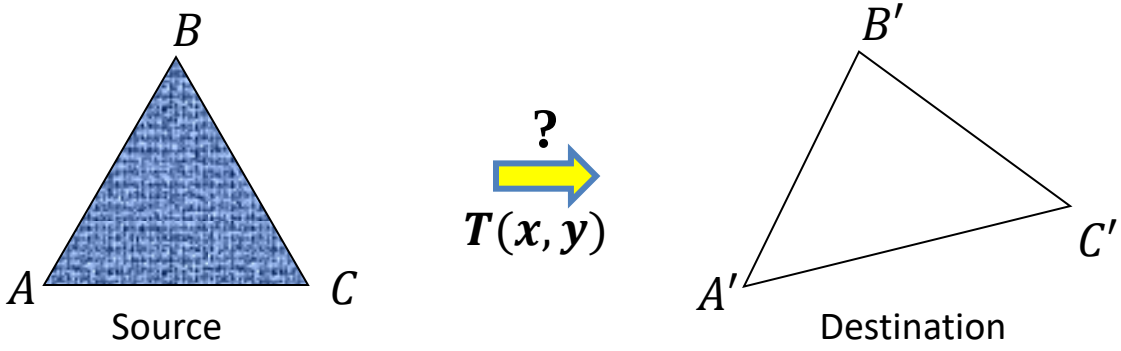
Warp interpolation

How do we create an intermediate shape at time t?

- Assume $t \in [0,1]$
- Simple linear interpolation of each feature pair
  - $t * p_0 + (1 - t) * p_1$ for corresponding features p_0 and p_1

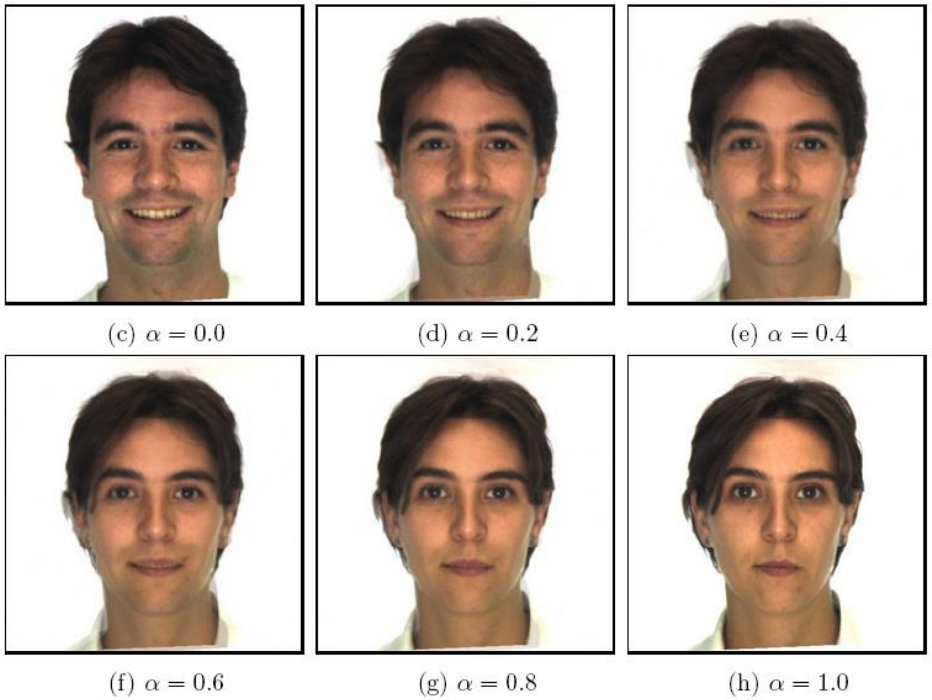


Suppose we have two triangles: ABC and $A'B'C'$. What transformation will map A to A', B to B', and C to C'? How can we get the parameters?



Morphing & Matting

- Extract foreground first to avoid artifacts in the background



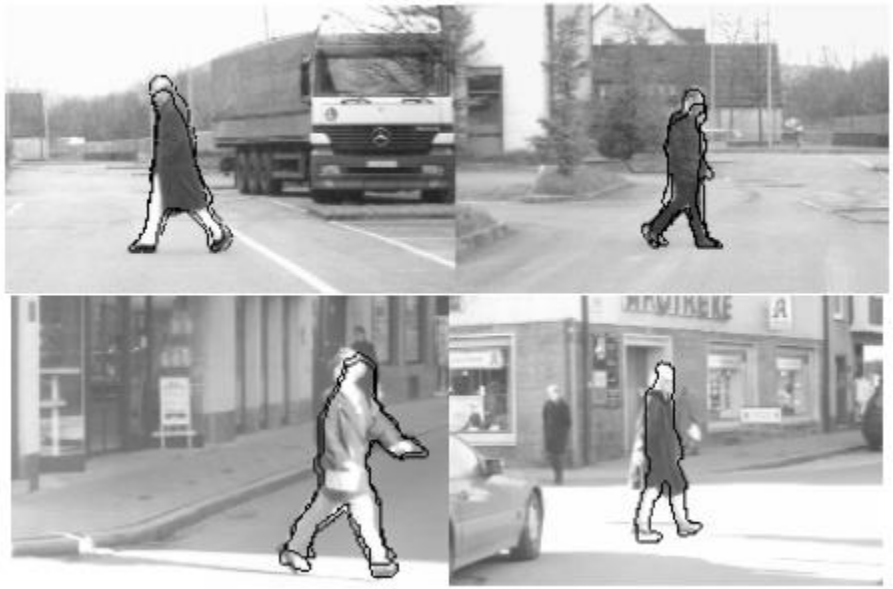
Slide: Durand and Freeman

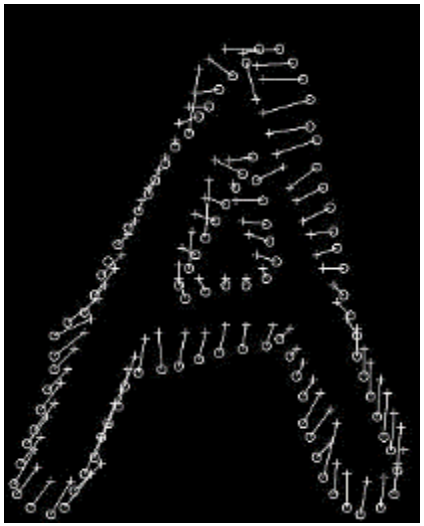
Summary of Morphing

1. Define corresponding points

2. Define triangulation on points
   - Use same triangulation for both images

3. For each $t = 0 : step : 1$
   a. Compute the average shape (weighted average of points)
   b. For each triangle in the average shape
      - Get the affine projection to the corresponding triangles in each image
      - For each pixel in the triangle, find the corresponding points in each image and set value to weighted average (optionally use interpolation)
   c. Save the image as the next frame of the sequence





Shape Matching

Shape Matching

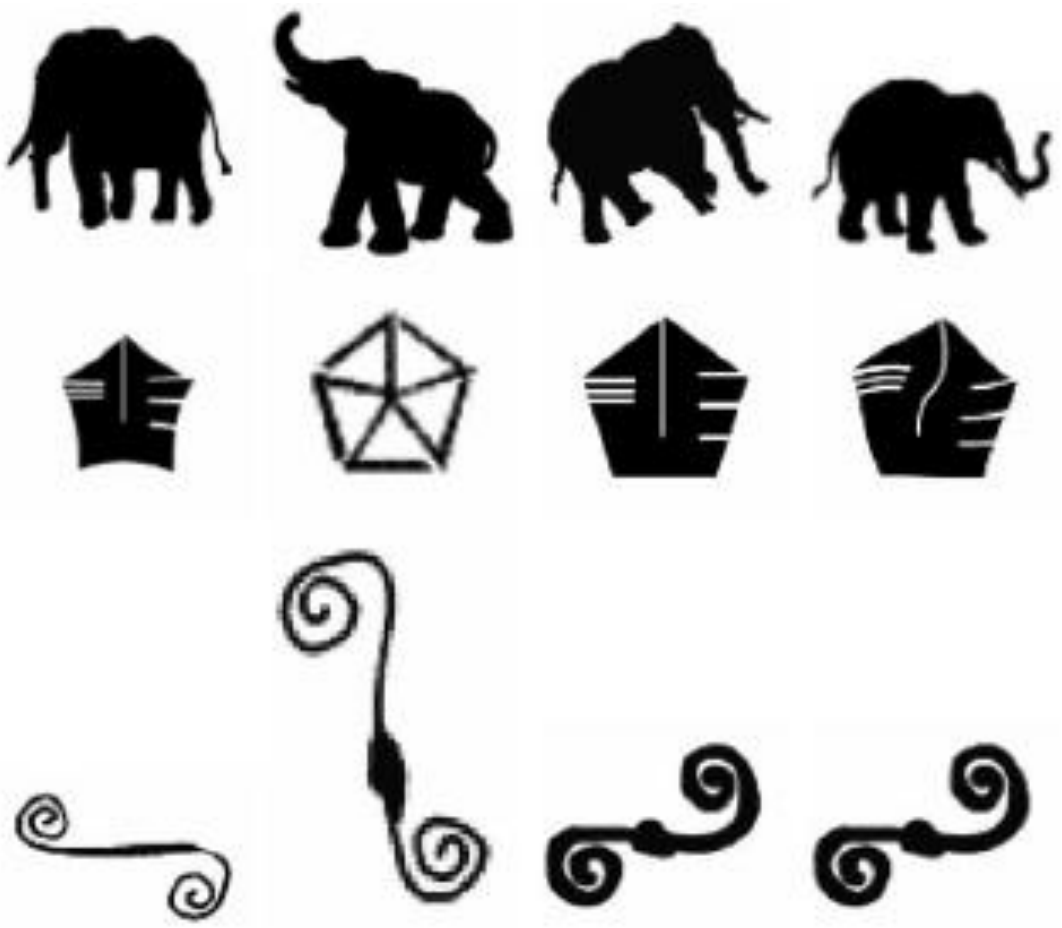


Fig. 11. Examples of shapes in the MPEG7 database for three different categories.

Questions

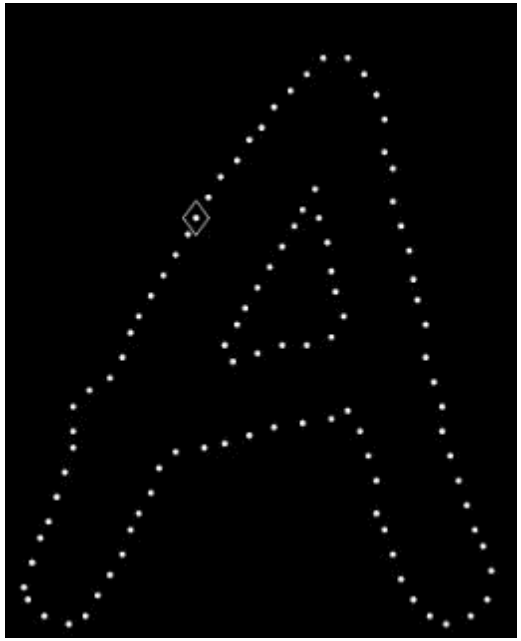
- What features?
- How to compare shapes?

Challenging!

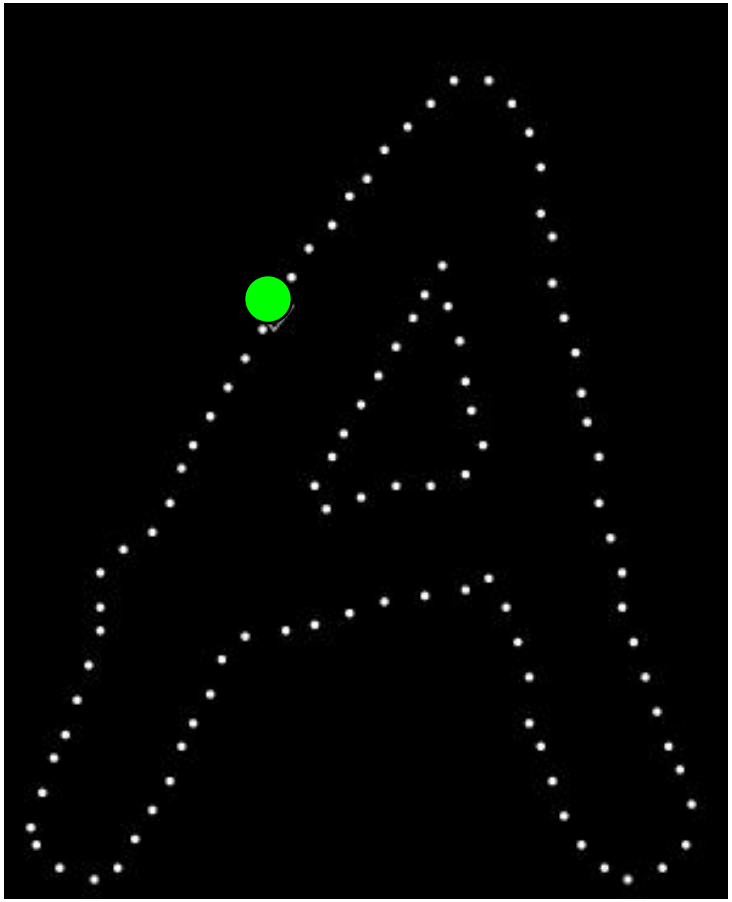
Fig. 1. Examples of two handwritten digits. In terms of pixel-to-pixel comparisons, these two images are quite different, but to the human observer, the shapes appear to be similar.

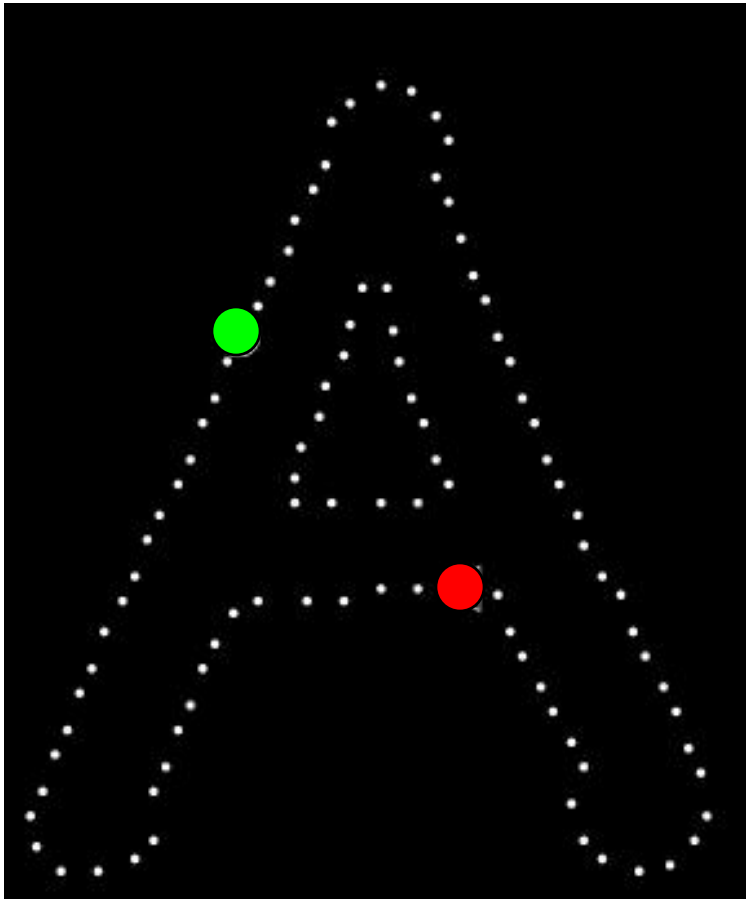
Figure from Belongie et al.

- What limitations might we have using only edge points to represent a shape?
- How descriptive is a point?



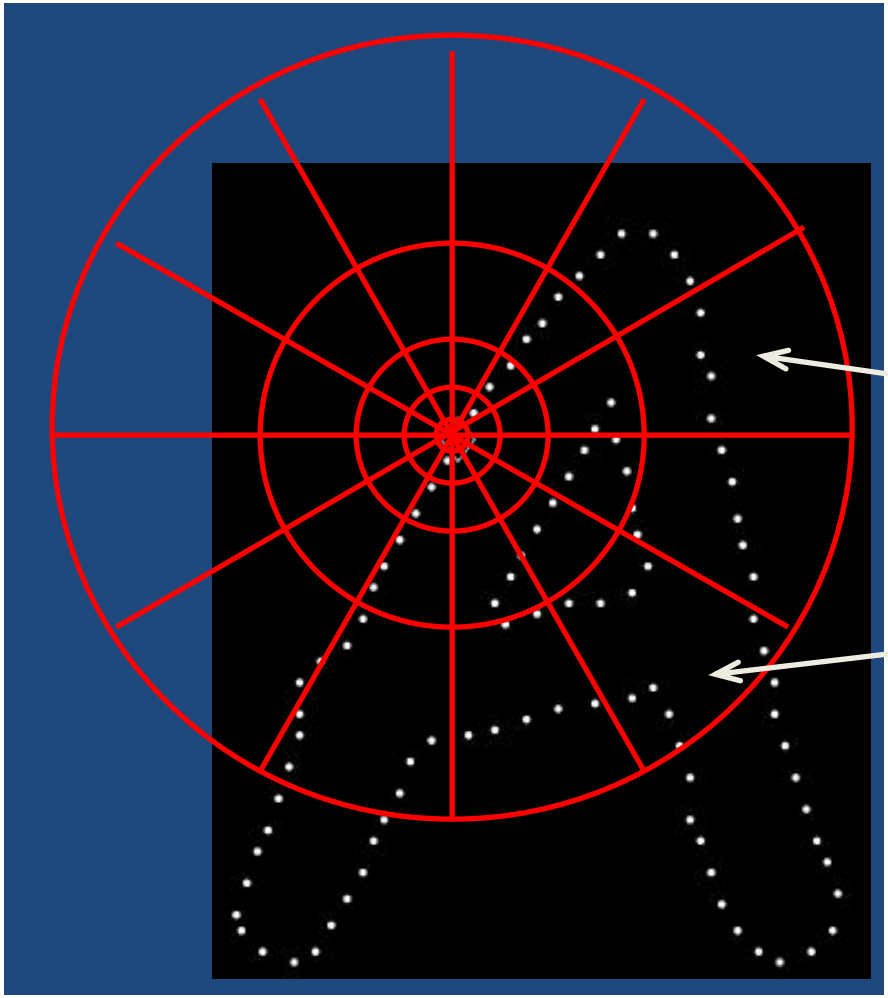
Comparing Shapes





What points on these two sampled contours are most similar? How do you know?

Shape Context Descriptor



Count the number of points inside each bin, e.g.:

Count = 4

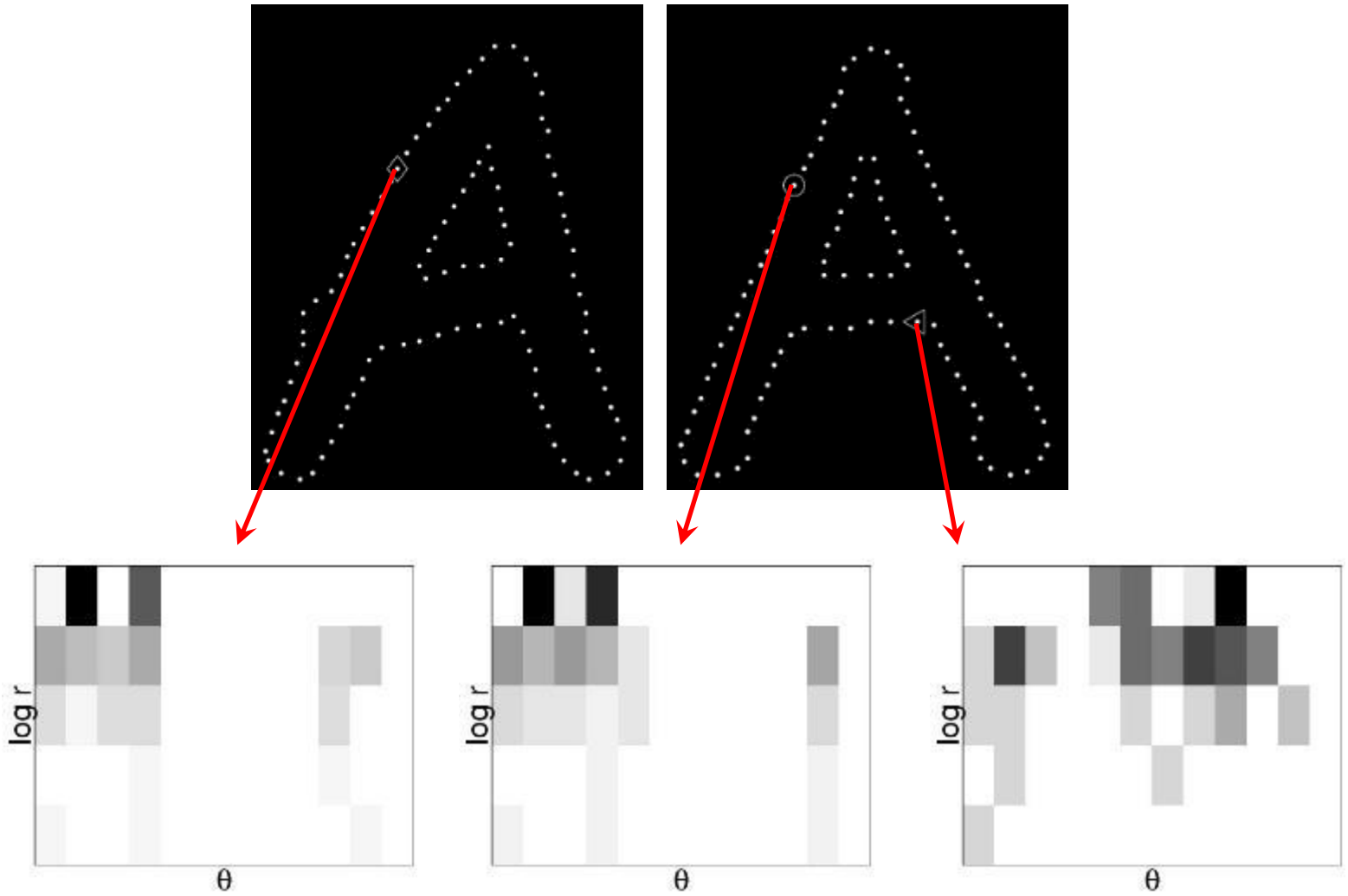
⋮

Count = 10

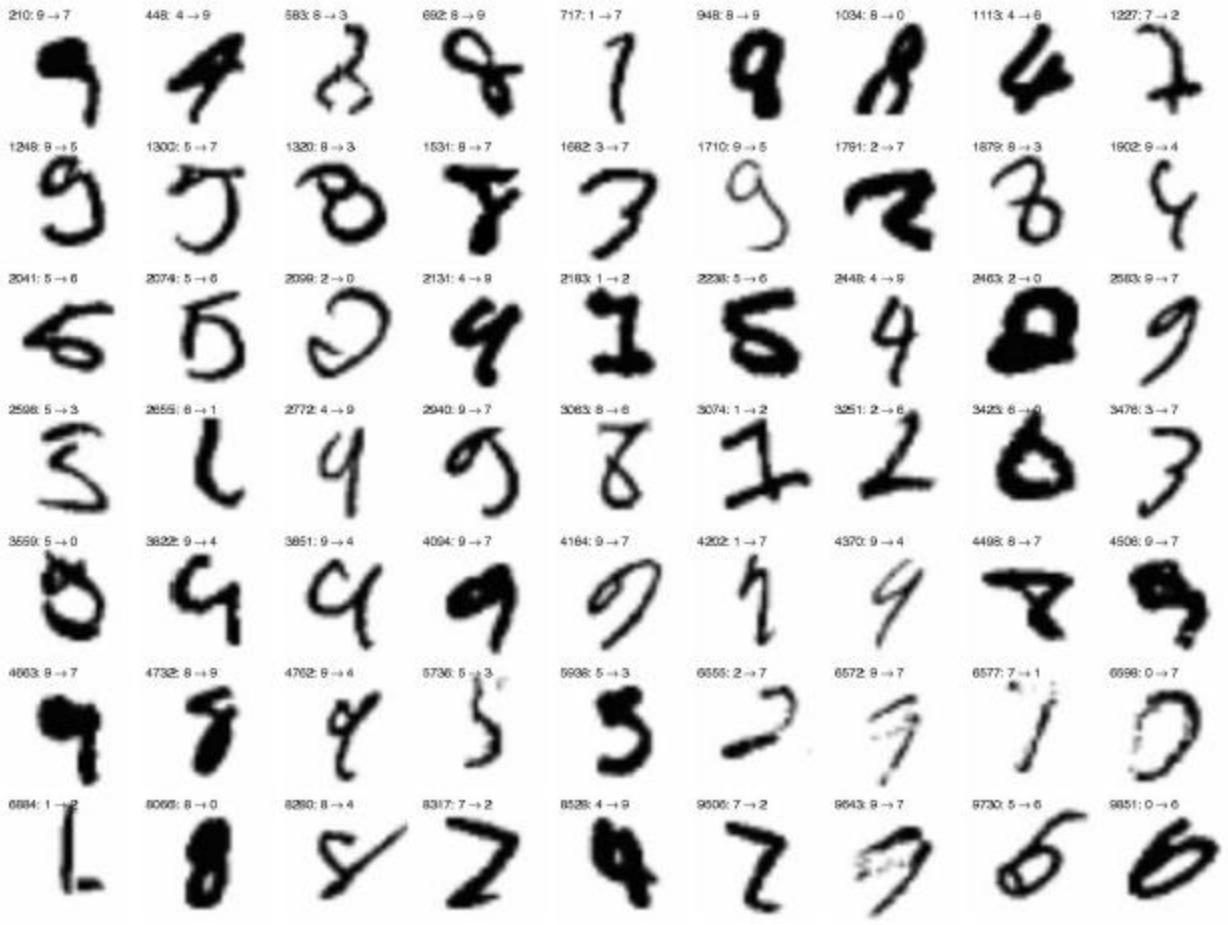
Compact representation of distribution of points relative to each point

Shape context slides from Belongie et al.

Shape Context Descriptor



Shape context matching with handwritten digits



Only errors made out of 10,000 test examples

Shape matching application: CAPTCHA's

CAPTCHA:

"*Completely Automated Public Turing Test To Tell Computers and Humans Apart*"

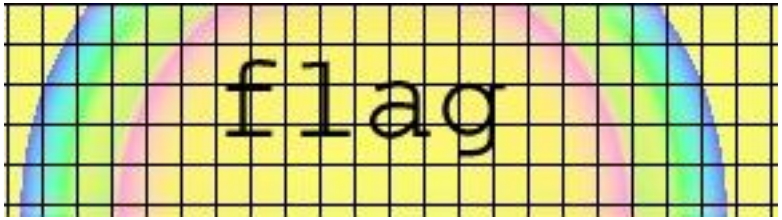
www.captcha.net

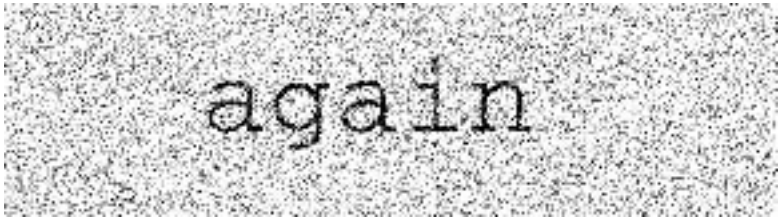
Luis von Ahn, Manuel Blum, Nicholas Hopper, and John Langford

CMU 2000

Shape matching application: breaking a visual CAPTCHA

- Use shape matching to recognize characters, words in spite of clutter, warping, etc.

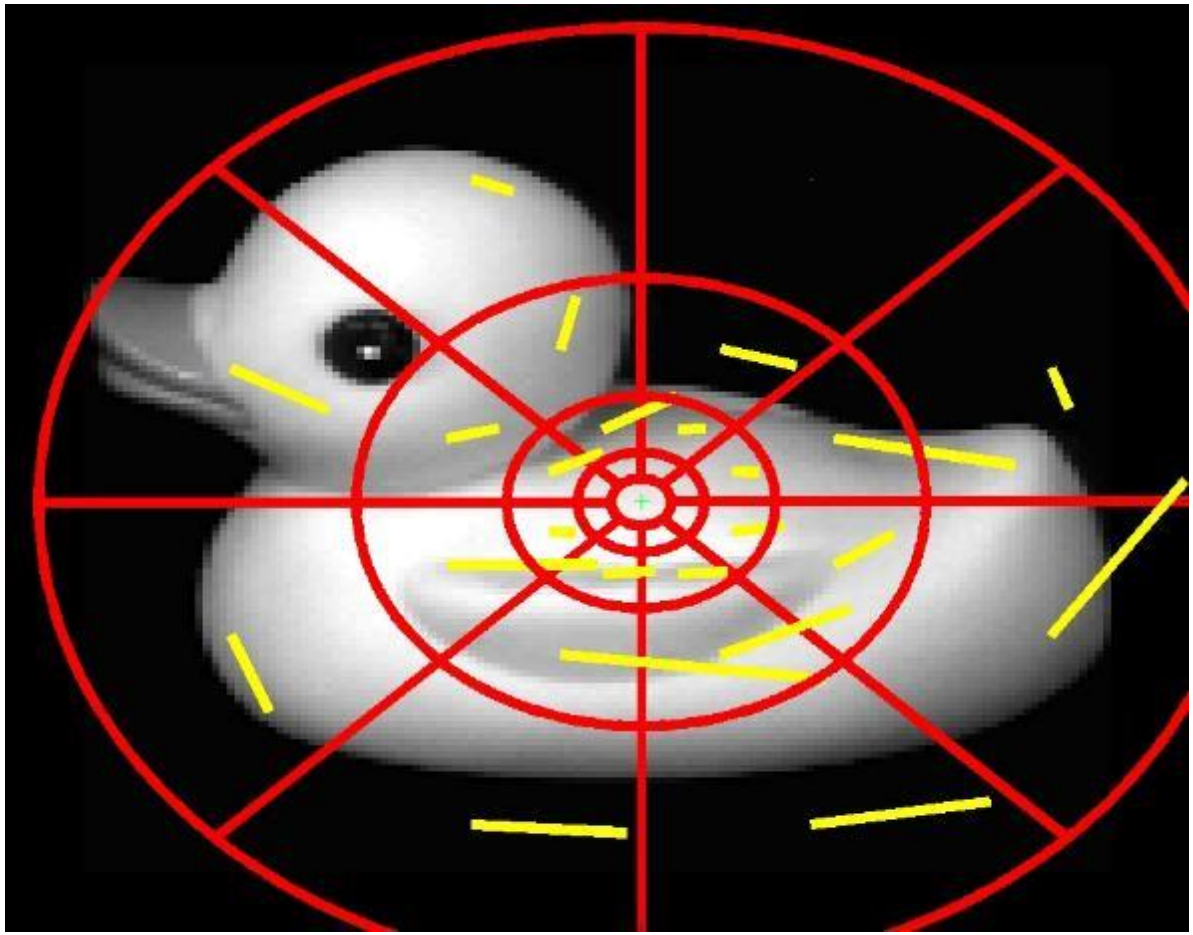




G. Mori and J. Malik, “Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA”, CVPR 2003

Features: Generalized Shape Contexts

- Can put more than just point counts in bins
  - Oriented Energy
  - Colour info
  - Optical flow



Fast Pruning: Representative Shape Contexts

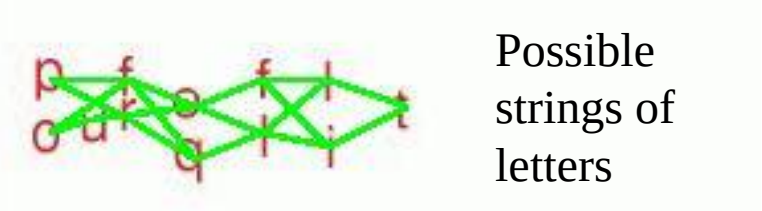
- Pick *k* points in the image at random
  - Compare to all shape contexts for all known letters
  - Vote for closely matching letters
- Keep all letters with scores under threshold

Algorithm A: bottom-up

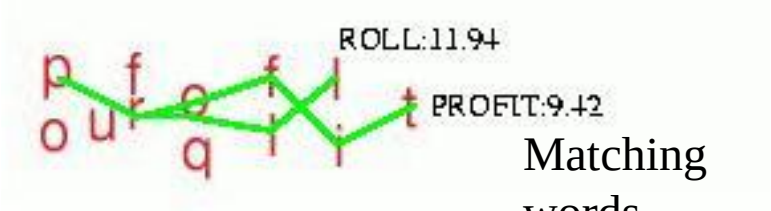
- Look for letters
  - Representative Shape Contexts
- Find pairs of letters that are "consistent"
  - Letters nearby in space
- Search for valid words
- Give scores to the words

Input

Locations of possible letters



Possible strings of letters



Matching words

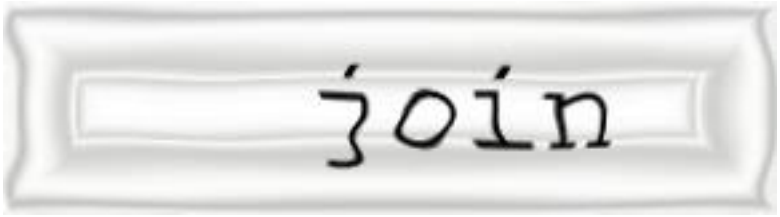
EZ-Gimpy Results with Algorithm A

- 158 of 191 images correctly identified: 83%
  - Running time: ~10 sec. per image (MATLAB, 1 Ghz P3)

horse

spade

smile



join

canvas

here

Gimpy

- Multiple words, task is to find 3 words in the image

- Clutter is other objects, not texture

Algorithm B: Letters are not enough

- Hard to distinguish single letters with so much clutter
- Find words instead of letters
  - Use long range info over entire word
  - Stretch shape contexts into ellipses

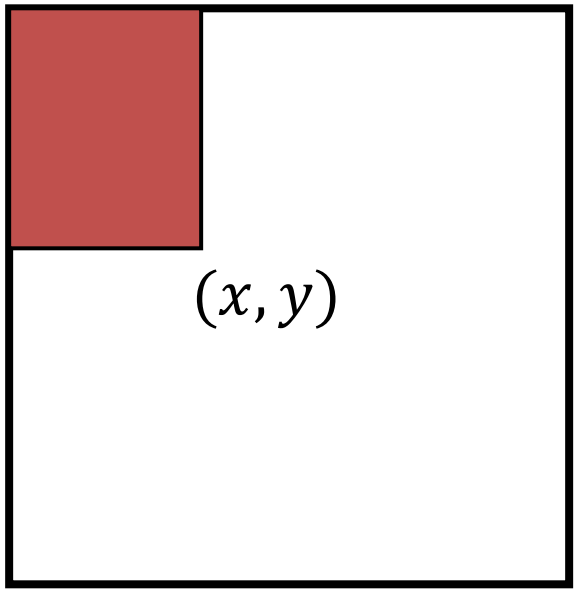
- Search problem becomes huge
  - # of words 600 vs. # of letters 26
  - Prune set of words using opening/closing bigrams

Integral Images

Fast computation of convolution with box filters with integral images

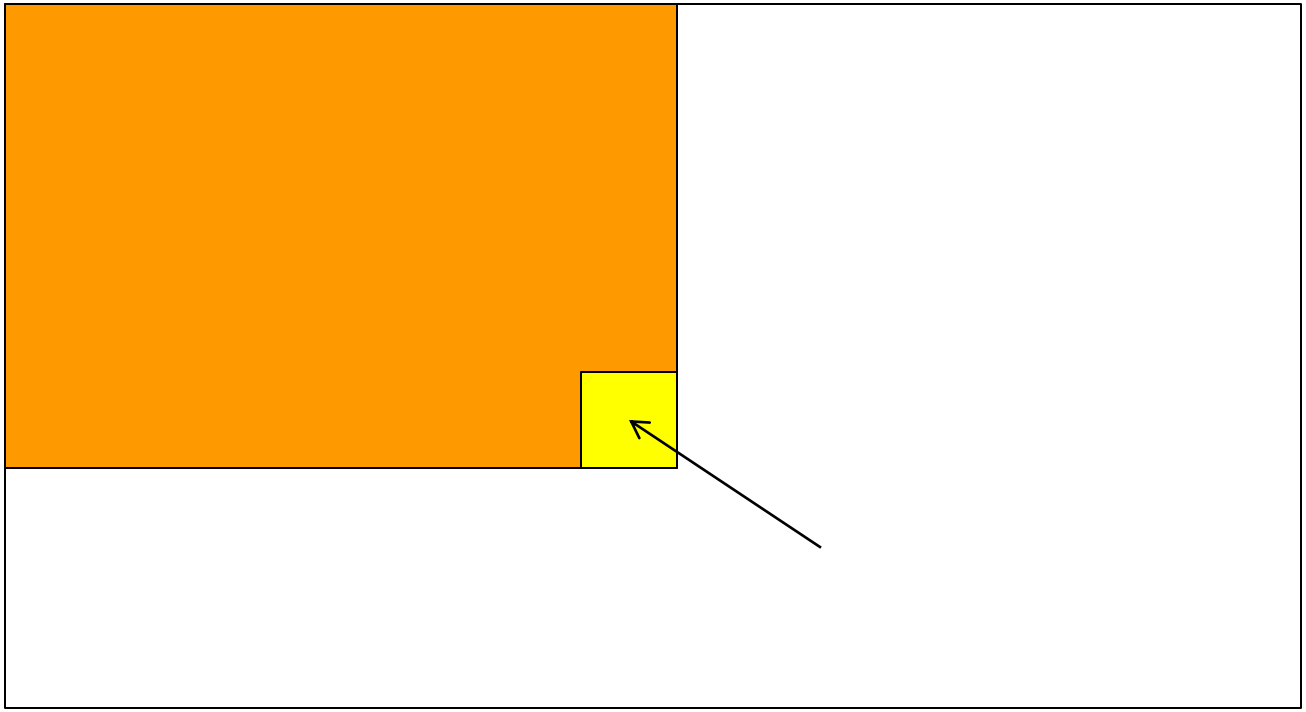
- The *integral image* computes a value at each pixel (x, y) that is the sum of the pixel values above and to the left of (x, y), inclusive.

- This can quickly be computed in one pass through the image.



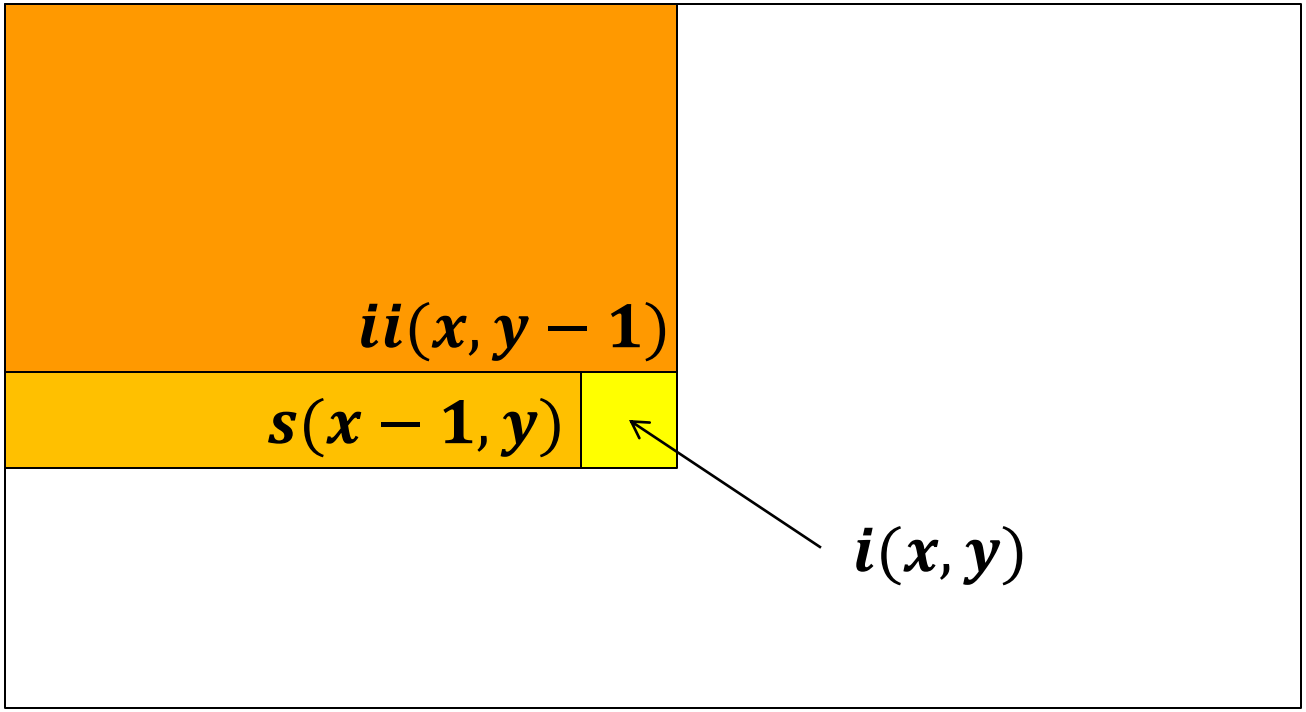
Slide: S. Lazebnik

Computing the integral image



Slide: S. Lazebnik

Computing the integral image



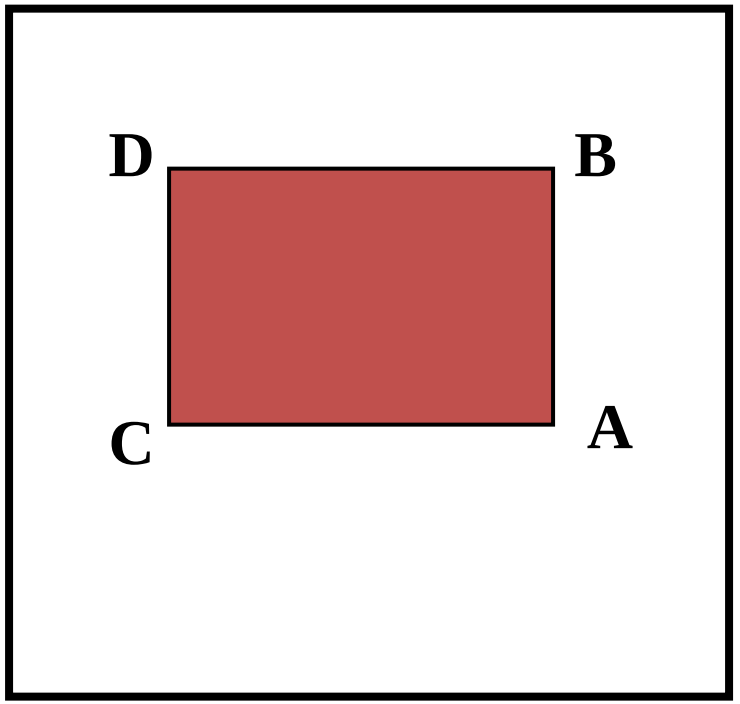
- Cumulative row sum: $s(x, y) = s(x-1, y) + i(x, y)$
- Integral image: $ii(x, y) = ii(x, y-1) + s(x, y)$
- In MATLAB: $ii = cumsum(cumsum(double(i)), 2)$
- In Python: $ii = cumsum(cumsum(double(i), 0), 1)$

Slide: S. Lazebnik

Computing sum within a rectangle

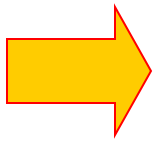
- Let A, B, C, and D be the values of the integral image at the corners of a rectangle.
- Then the sum of original image values within the rectangle can be computed as:

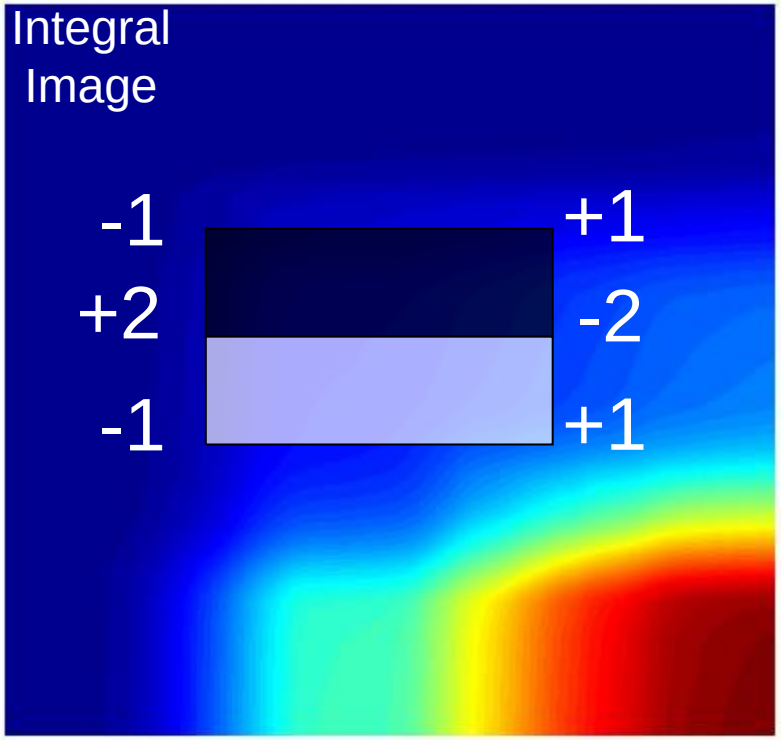
  sum = A – B – C + D
- Only 3 additions are required for any size of rectangle!



Slide: S. Lazebnik

Example





Slide: S. Lazebnik