CS664 Lecture #2: Texture synthesis and its mathematical basis

Some slides taken from:

- **Alyosha Efros and Thomas Leung**
  http://www.cs.berkeley.edu/~efros/research/EfrosLeung.html
- **John Malone-Lee and Nigel Smart**
  http://www.cs.bris.ac.uk/Teaching/Resources/COMS30124/Lectures/Part_I.pdf
- **David Lowe et al.**
  www.cs.ubc.ca/~lowe/425/slides/11-Classifiers.ppt

Last lecture we saw:

- Vision is hard
  - Problems are ill-defined (“up for grabs”)
- Even simple problems like edge detection are remarkably complex
- The human visual system ignores a lot of things that are actually present in the raw data
  - Such as color changes
- Segmentation is even harder than edge detection

Limits of segmentation

- Classical segmentation algorithms only handle intensities
  - Typically perform poorly on highly-textured regions (e.g., a plaid shirt)
  - Example images are often carefully chosen

Generalization

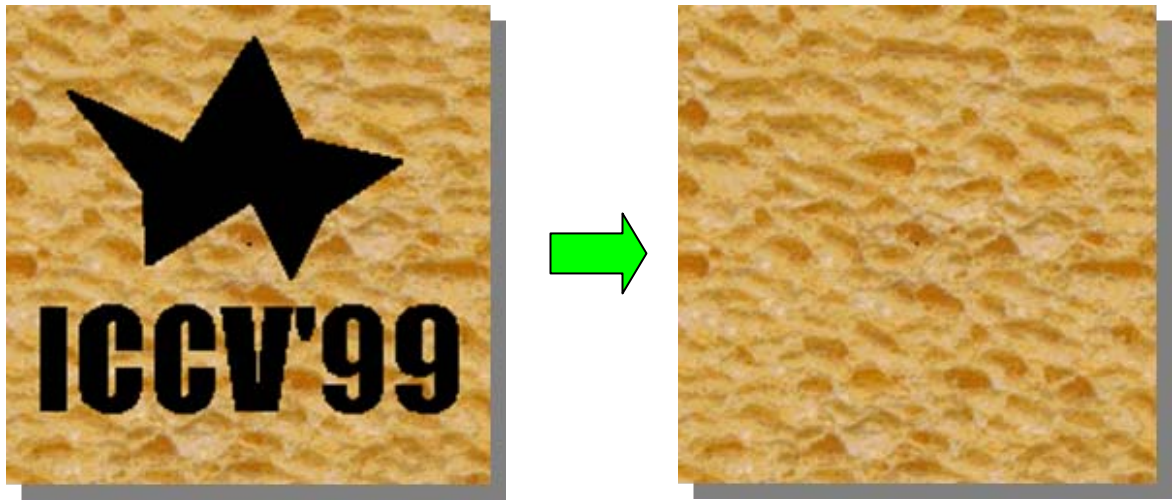
- Can generalize the notion of affinity to handle texture
  - Similarity in terms of orientation or scale
- It is possible to compute at each pixel a local vector that determines for a given θ the evidence at that pixel for an edge at orientation θ
  - Can also depend on scale
  - How much do you need to blur the image before this feature vanishes?

Texture

- A particularly badly-defined problem
  - Texture is what makes something look like itself independent of lighting, scale and position
    - "Giving a name to your confusion"
  - Various local spatial statistics can be used
- We will look at this in the context of a very cool application
  - Image inpainting, or texture synthesis
  - How to delete someone from a scene?

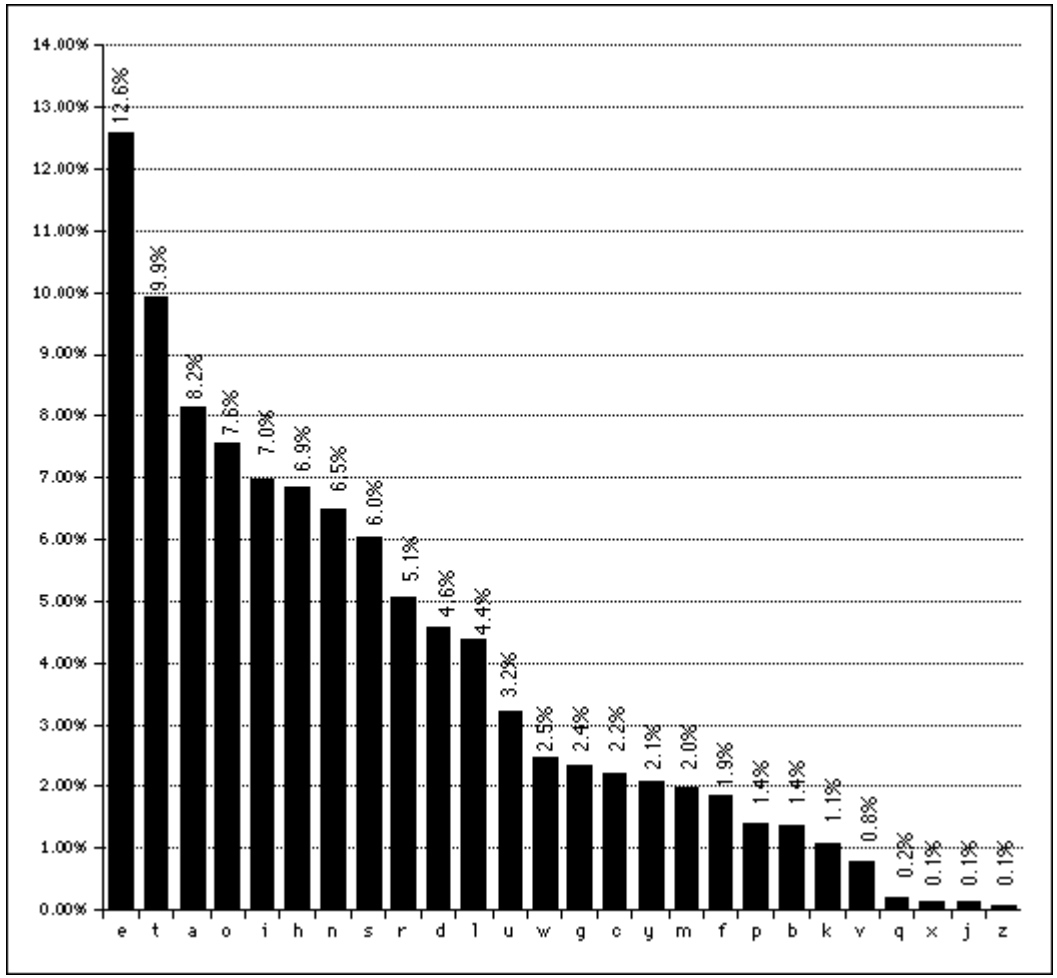
Our 1st vision paper

- Efros & Leung, “Texture Synthesis by Non-parametric Sampling”, ICCV ’99
  - http://www.cs.berkeley.edu/~efros/research/EfrosLeung.html



- Idea: draw on analogy to Shannon’s work on generating “English-like” sentences

Letter frequencies



Taken from:

Alice in Wonderland

http://www.mathmaniacs.org/lessons/02-textcomp/alicehistogram.html

E	12.6%
T	9.9%
A	8.2%
O	7.6%

Estimation and sampling

- We can estimate a **probability density** from the histogram
  - Informally, this is the relative frequencies of the letters (in the limit)
- Can use this to generate pseudo-English
  - Pick random $x \in [0,1]$ uniformly
  - If $x < 0.126$, output 'e'
  - Else if $x < 0.126+0.099$ output 't'
  - Else if $x < 0.126+0.099+0.082$ output 'a', etc.
- Sampling from a distribution/density

E	12.6%
T	9.9%
A	8.2%
O	7.6%

Generalizing this so it works

- Different **pairs** can be common (“th”), rare (“yz”) or forbidden (“qx”)
- Most common bigrams:
  - TH, HE, IN, ER, AN, RE, ED, ON, etc…
- Most common trigrams:
  - THE, ING, AND, HER, ERE, ENT, THA, NTH, WAS, ETH, FOR
- Compute distribution from a large text
- Can do this for words as well as letters!

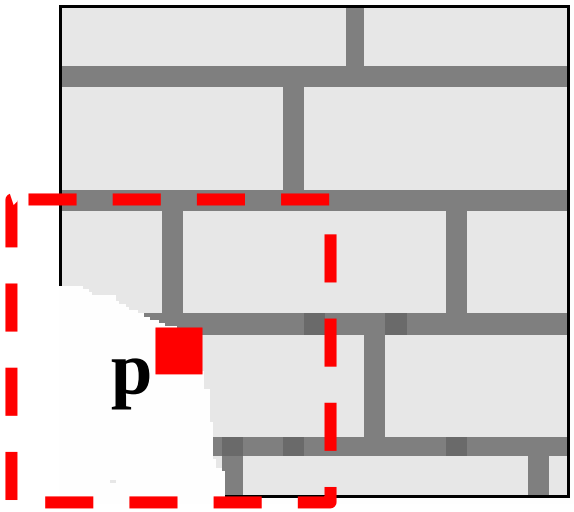
Text from trigrams

- Start with a common bigram
  - Given the preceding two words, there is a distribution over the next word
  - Sample from that, and continue

WE NEED TO EAT CAKE

- Results (Mark V. Shaney, alt.singles):
  - *"As I've commented before, really relating to someone involves standing next to impossible."*
  - *"I spent an interesting evening recently with a grain of salt"*

Try this for texture



- Given the intensities in the red box, we have a distribution over intensities of **p**
  - Computed from the rest of the image
- Sample from this to generate intensity of **p**, then do it again

Sampling issues and solutions

- Exact match is probably not present
  - So they find the best match using SSD error
  - Sample from all matches within some distance of that match
    - up to 10% greater SSD error
- Need to emphasize local structure
  - Weight the SSD by a gaussian
- Some pixels have unknown values
  - Don't count them in the match

Results

Effects of window size

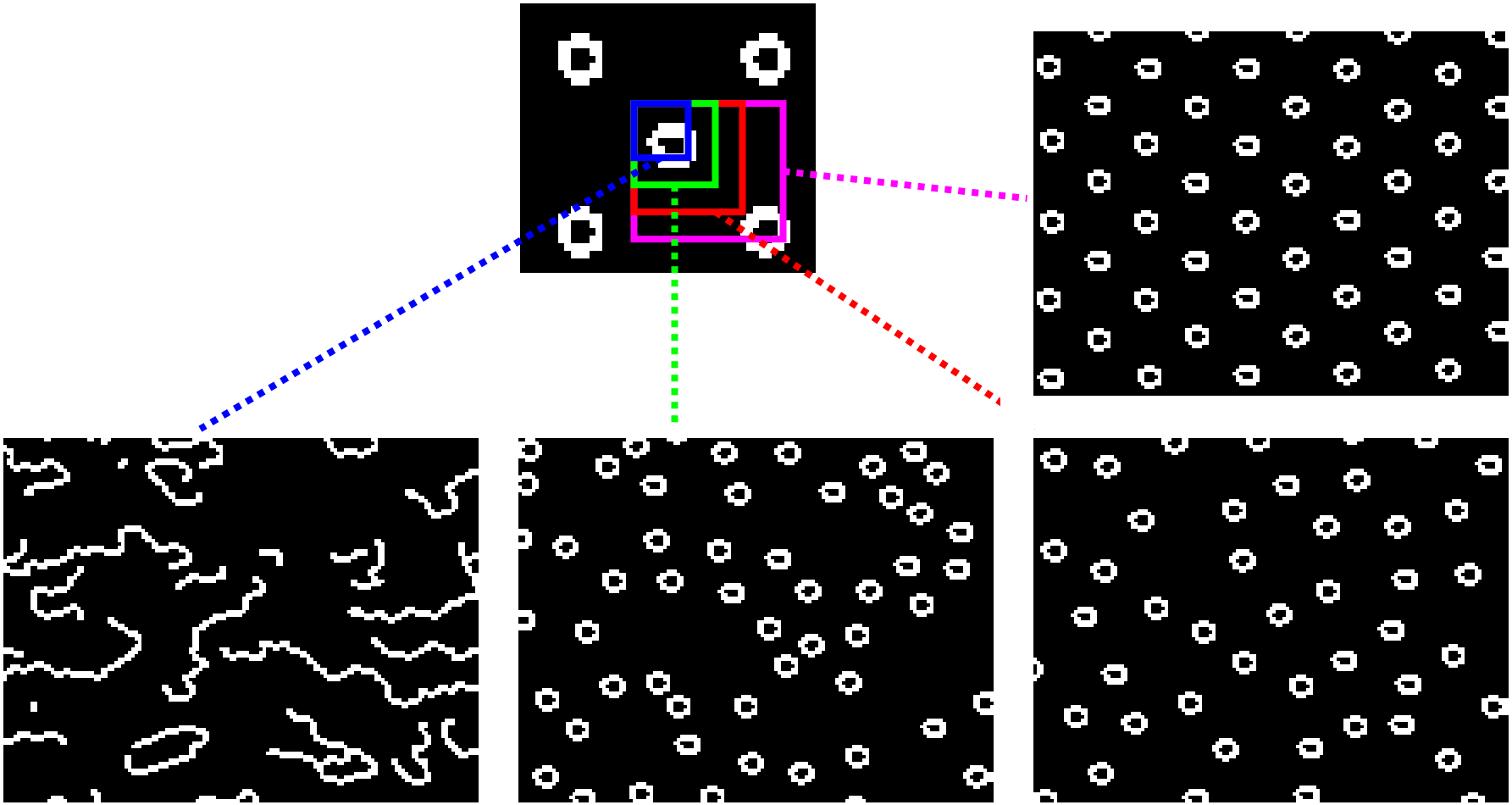


Image extrapolation

Mathematical basis

- Efros and Leung's paper draws on:
  - Density estimation
    - Specifically, non-parametric density estimation
  - Sampling from a distribution
  - Markov chains
  - Markov Random Fields
- We could easily spend a semester on these topics (don't worry, we won't...)
  - But we will cover each of them, and describe some of their applications to computer vision

Density estimation

- Compute a density given some data drawn from an underlying distribution
- Suppose that we know the true distribution is a bell curve with center μ and width σ
  - How can we estimate μ and σ?

$$\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i$$

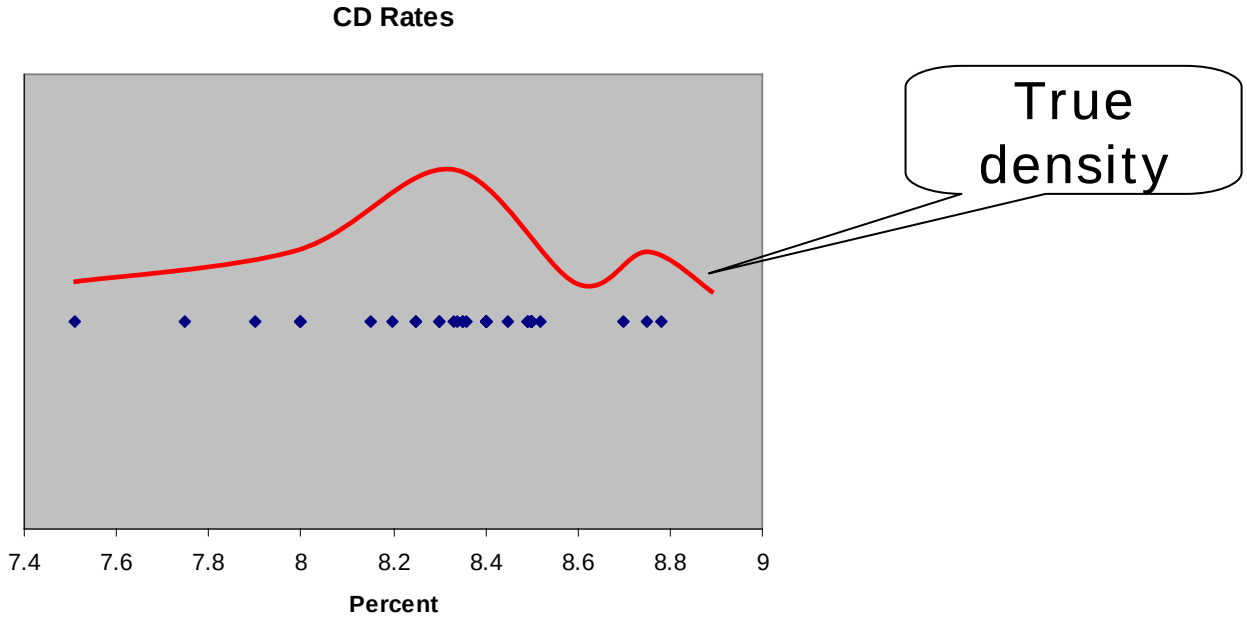
  - Later on we'll see these are "optimal" estimates for μ and σ

Non-parametric case

- We knew the underlying distribution
  - All we needed was to estimate the parameters
  - Obviously, this gives bad results when the true distribution isn't what we think it is
  - Non-gaussian distributions are in general rare, and hard to handle
    - But they occur a **lot** in computer vision!
  - Another reason why vision is hard
    - Even when you know the problem definition, it tends to be quite challenging

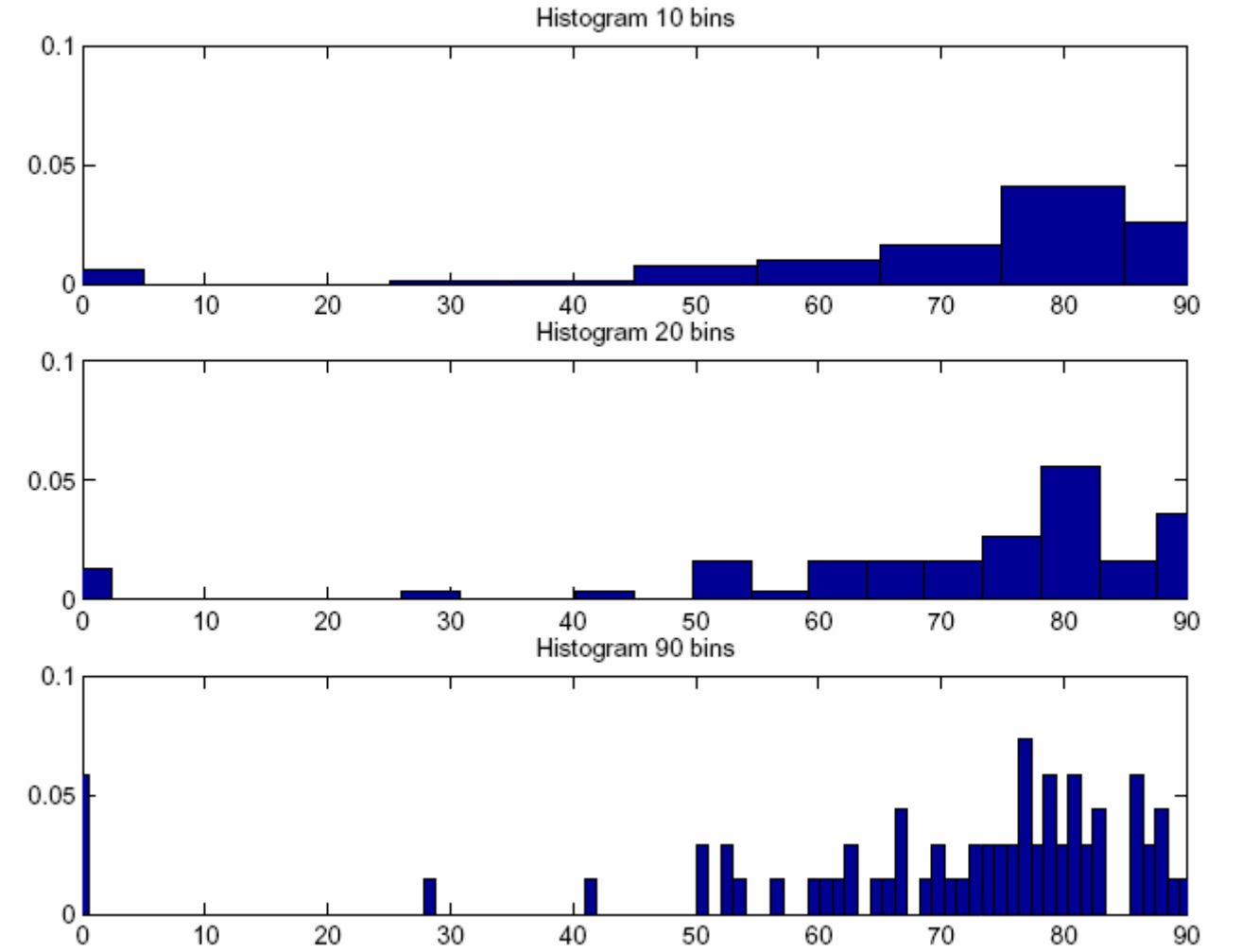
Is there a free lunch?

- Suppose we simply plot the data points
  - Assume 1-D for the moment



  - How to compute density from data?

Histogram representation



Bin-size tradeoffs

- Fewer, larger bins give a smoother but less accurate answer
  - Tend to avoid “gaps” (i.e., places where the density is declared to be 0)
- More, smaller bins have the opposite property
- If you don’t know anything in advance, there’s no way to predict bin size
  - Also, note that this method isn’t a great idea in high dimensions