COMPUTER SCIENCE 664: Machine Vision

Instructor: Ramin Zabih

TA: Gurmeet Singh

Course Mechanics

- Signup sheet
  - NetID, Probability(credit)
- http://www.cs.cornell.edu/courses/cs664/
  - Not a substitute for attending class
- Short quizzes every few weeks
  - First 10 minutes of class (i.e.: be on time…)
- One or two programming projects
  - Done in pairs
- 1-page summary of any vision paper
- Final project (done individually)

Final Project

- Most of the course grade comes from this
  - It is actually possible to get an F in CS664
- You need to pick a project by midway through the term, and work hard on it
  - Write-up due during final exam period
  - Proposal due by November 1
- Must be original research
  - i.e., non-trivial idea that has never been done
- Not required to solve a vision problem
  - Document idea's strengths and weaknesses

Topics and techniques

- Mathematical tools such as:
  - Statistics (especially estimation)
  - Differential geometry
  - Linear programming
- Algorithmic techniques such as:
  - Dynamic programming
  - Graph algorithms
- No proofs required, but some coding
- Quizzes to test basic comprehension

What is computer vision about?

- The **content** of images
  - How many people are present?
  - How far away from the camera are they?
  - Did the camera move? If so, how?
- Often, questions about the 3D scene
  - Not uniquely determine by an image!
- Sometimes, issues involving odd and novel imaging devices
- Boundaries are unclear, especially with Image Processing and with Graphics

Computer vision algorithms

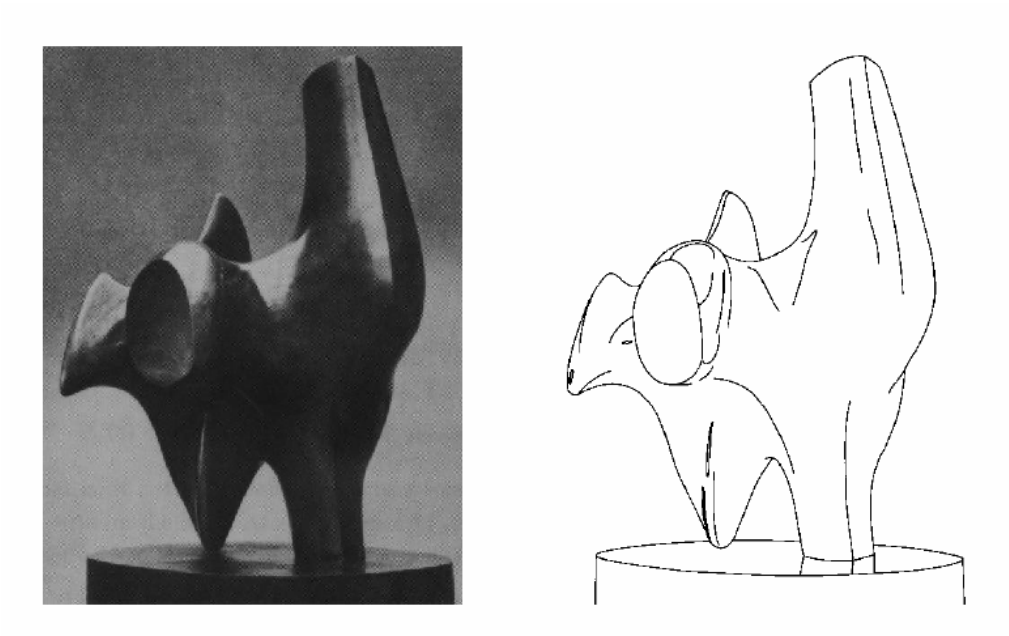
- Input is one or more 2D arrays of intensities (8-bit numbers for grayscale)
- How can such a simple data structure merit an entire field?
  - Very easily, as you will see…
  - Note: CS664 is unusually algorithmic
    - Part of why there is no textbook
- Vision poses unusual algorithmic challenges

First algorithm: edge detection

- An algorithm solves a *problem*
- CS insists on cleanly separating the two
  - Some people in vision think Marr invented this
- Classical example: sorting problem (defined by input and output) vs. algorithm (i.e. quicksort)
  - You can easily tell if the output is right
  - Most of the issue is efficiency and generality

Edge detection problem?

- OK, what is the problem definition?
  - Binary image where 1 = ‘edge’
  - Intuition: compute something like a binary “artist’s sketch” of the scene



Why solve this?

- Sub-problem for other vision problems
  - E.g.: ovoid groups of edges are possible faces
  - Track the motion of edges over time
- Or you might have a 3D wire-frame model of some object, like a car or a stapler
  - You could then find that object's pose and position using edges
- Another motivation: the human visual system does something like edge detection fairly early

Problem definition issues?

- None of the intuitions give you a precise problem definition
- In fact, there is no precise problem definition for edge detection
  - or for almost any other vision problem
- The way that edge detectors usually work is by detecting “big changes”
  - Works real well for synthetic images!

Consequences

- Caricature of many vision papers:
  - Compute binary images somehow
  - Give intuitions that it should be ``edge-like'
  - Show 1-2 example images
- 40 years of progress in vision:
  - Much more complex math
    - Proofs of algorithm optimality, for some formal definition of an edge
    - Note: proofs usually only apply in 1D
  - Slightly better experimental comparisons
    - But still largely subjective

Problems are undefined

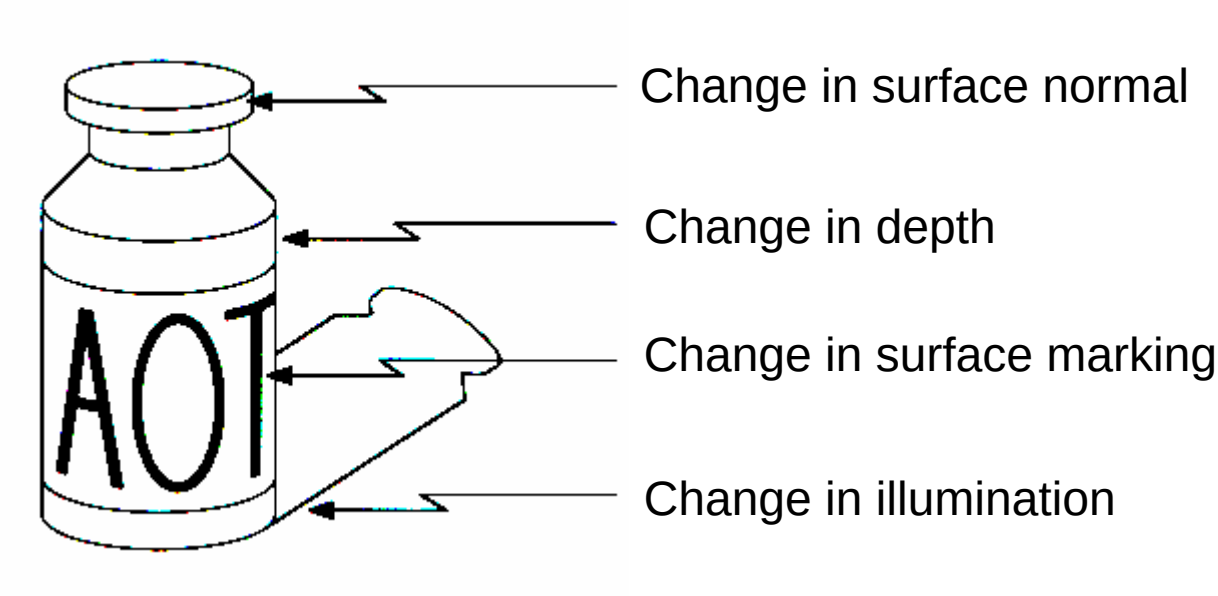
- Hard to tell if your edge detector is working or not
  - Having an application on top isn't a panacea
- This is part of what makes vision "fun"
  - You can define your own problems
  - A slight tweak on a problem definition can lead to a new class of math being applicable

Problems are subtly hard

- Even edge detection is much harder than it appears to be
  - People think calculus is hard, vision is easy
  - It's the opposite for computers
    - maybe for people too, judging by neuron count
- The "important" edges are often very subtle when you look at the actual intensities

What causes edges?

- Many things — which ones matter?



Importance of assumptions

- People clearly uses a lot of knowledge that isn't in the image to understand it
  - True, but not helpful
- No one really knows how to represent such information or to use it effectively
  - 1960's vision tried ad hoc methods
    - "black blob in center = telephone"
    - Powerful if true, but not robust of general
  - Since then, we use very general information
    - I.e., edge pixels should be sparse lines/curves
    - Less powerful, but more reliable

Assumptions in vision

- No one really knows how to represent such information, or to use it effectively
  - Persistent bugaboo of the field
- Many applications require the ability to gracefully handle the unexpected
  - Which means, weak assumptions only
    - Statistics of natural images
  - First time your robot sub sees a platypus?
- A few exceptions
  - medical imaging
  - OCR (not considered to be vision)

Noise

- Another major reason vision is hard
  - Cameras produce noisy (unstable) images
  - Even 1/30 of a second apart in static scenes
    - Take a close look at an HDTV sometime
- Edge detectors are unstable
  - will not produce the same output from what appears to a human to be identical images
- The algorithms essentially threshold the local evidence for an edge
  - When evidence is near to threshold, noise can push it over or under

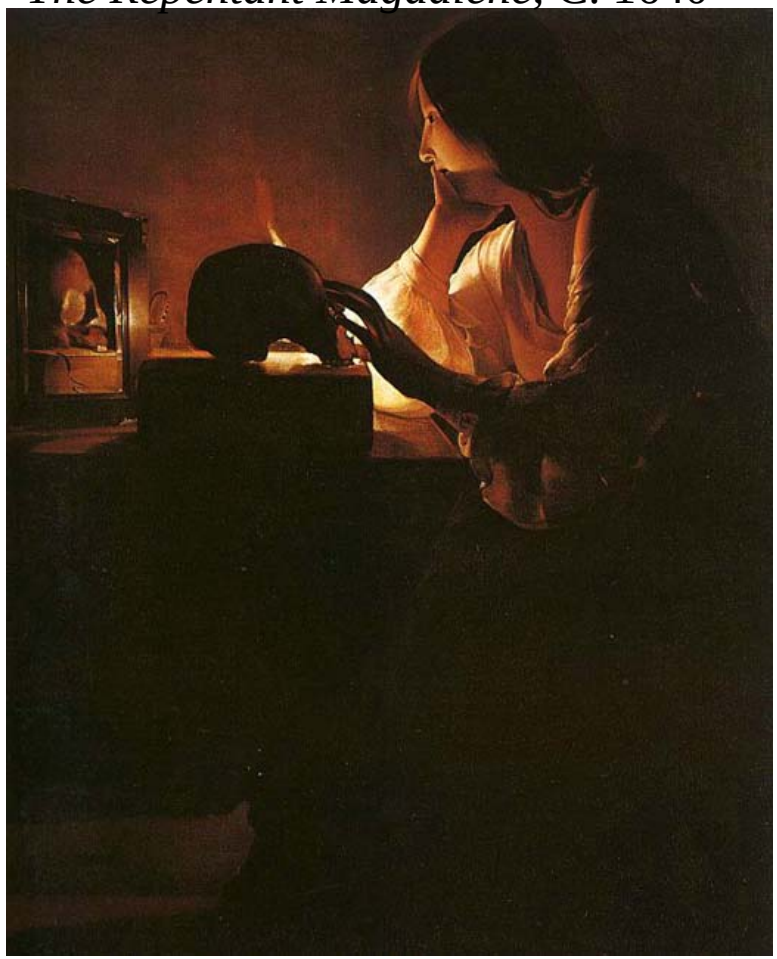
Noise isn't trivial

- Not just a matter of better cameras
  - Though they would help
- As a rule, edge detectors have a *scale* parameter that controls how many edges they will produce
  - We'll discuss this later, when we talk about some actual edge detectors
- Need algorithms that will ignore changes that people just don't see
  - Example: color constancy

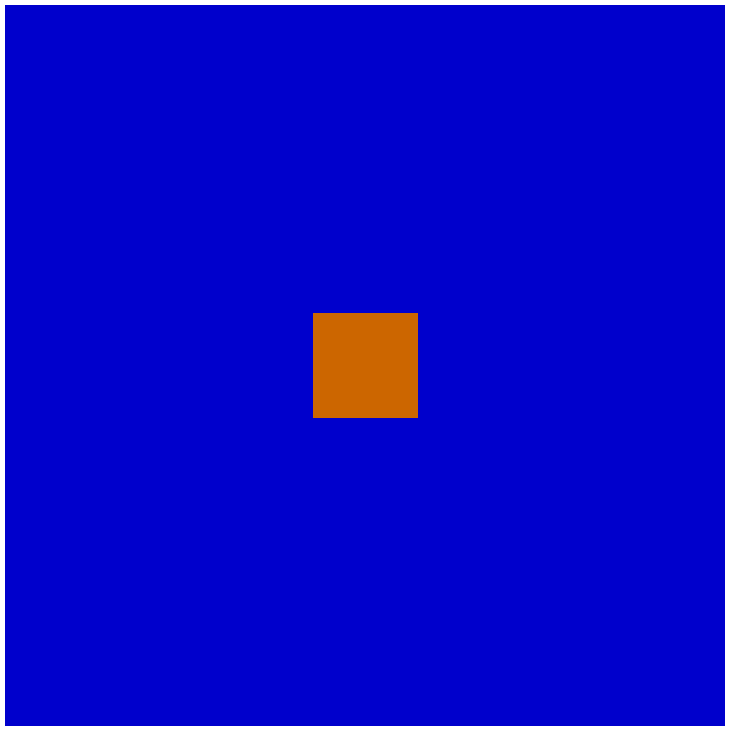
Color Constancy Examples

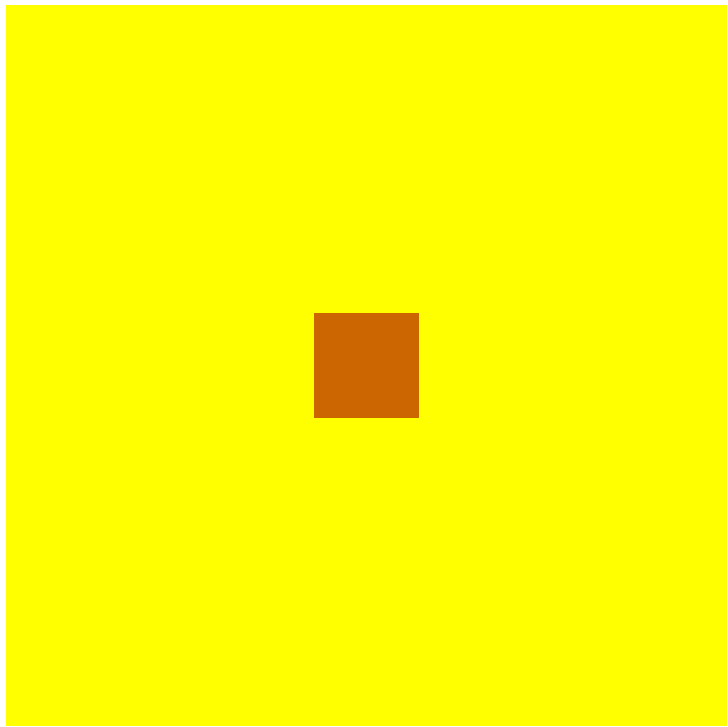
- All color constancy examples c/o Ian Horswill, Northwestern

Georges de la tour,
The Repentant Magdalene, C. 1640

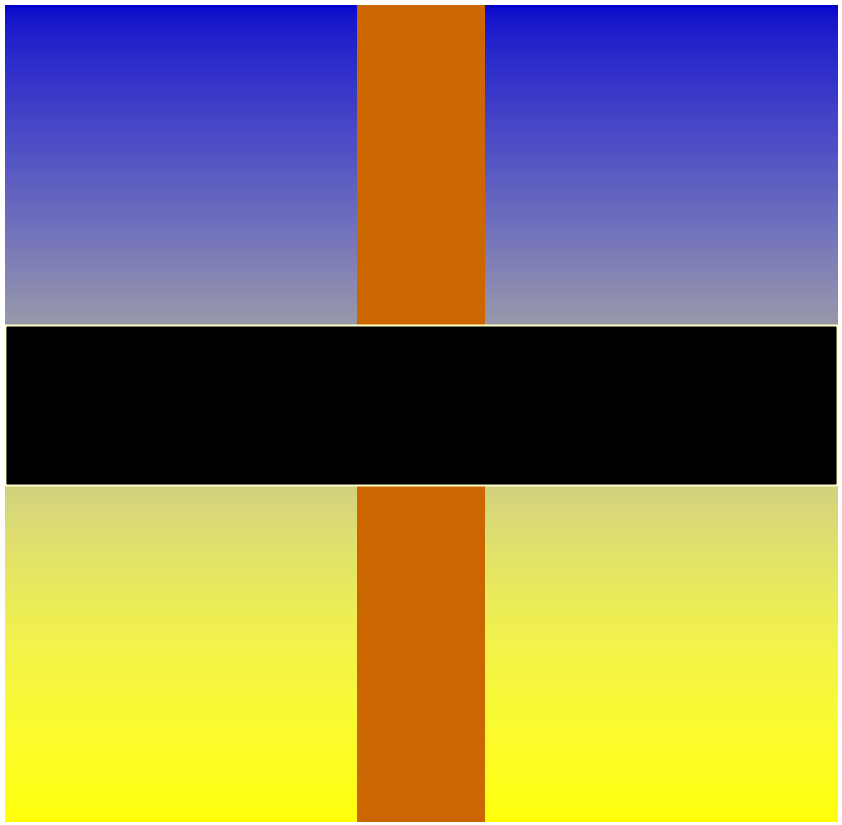


Failures of Color Constancy





More interesting failure



Problem: image segmentation

- Like edge detection, this isn't really a problem in the CS sense
  - Goal is to divide the image into meaningful pieces (usually for further processing)
  - The type of the output is a partition, i.e. a labeling of each pixel with numbers 1...n
    - One number per connected region
    - Ideally, these correspond to "objects"
- Would be great if it worked!
  - Like cold fusion and teleportation

Segmentation is important

- A huge number of potential applications depend on segmentation
  - Colorize movies easily
  - Smart Photoshop, allowing you to place a person or an object in another scene
  - Many Hollywood special effects, or realistic-looking video game design
- But is there a right answer?
  - How many objects are there in this room?

Segmentation algorithms

- Some resemblance to edge detection
  - Look for big changes in intensity
  - Produce regions rather than edges
    - But, some edge detectors do this also
  - Some kind of scale parameter
- Tend to use information farther away in the image
- There are some nice mathematical formulations of the segmentation problem
  - Hot topic: graph partitioning

Graph–based segmentation

- Create a graph from the pixels
  - Edges connect a pixel to its neighbors
    - 4- or 8-connected, or more
  - Find minimum sum-of-edge weights partition
- Edge weight is “affinity” (similarity)
  - Non-negative [why?]
  - High for similar pixels [why?]

$$w_{p,q} = K - |I(p) - I(q)|$$

How to write a vision paper

- Find a graph partitioning algorithm
  - Ideally, a new one
  - Or one that has never been applied to vision
- Run it on a few examples
- Popular approaches involves cuts and/or spectral methods
- Good example of how the field works
  - Ill-defined problem with some intuitions
  - Algorithms based on sophisticated math