Segmentation

CS 450: Introduction to Digital Signal and Image Processing

Recurring Theme in Applied CS

- Cast a specific problem as a more general optimization problem:
  - An objective function that characterizes a good solution (depends on the problem)
  - A method for finding the optimal solution (or close to it)

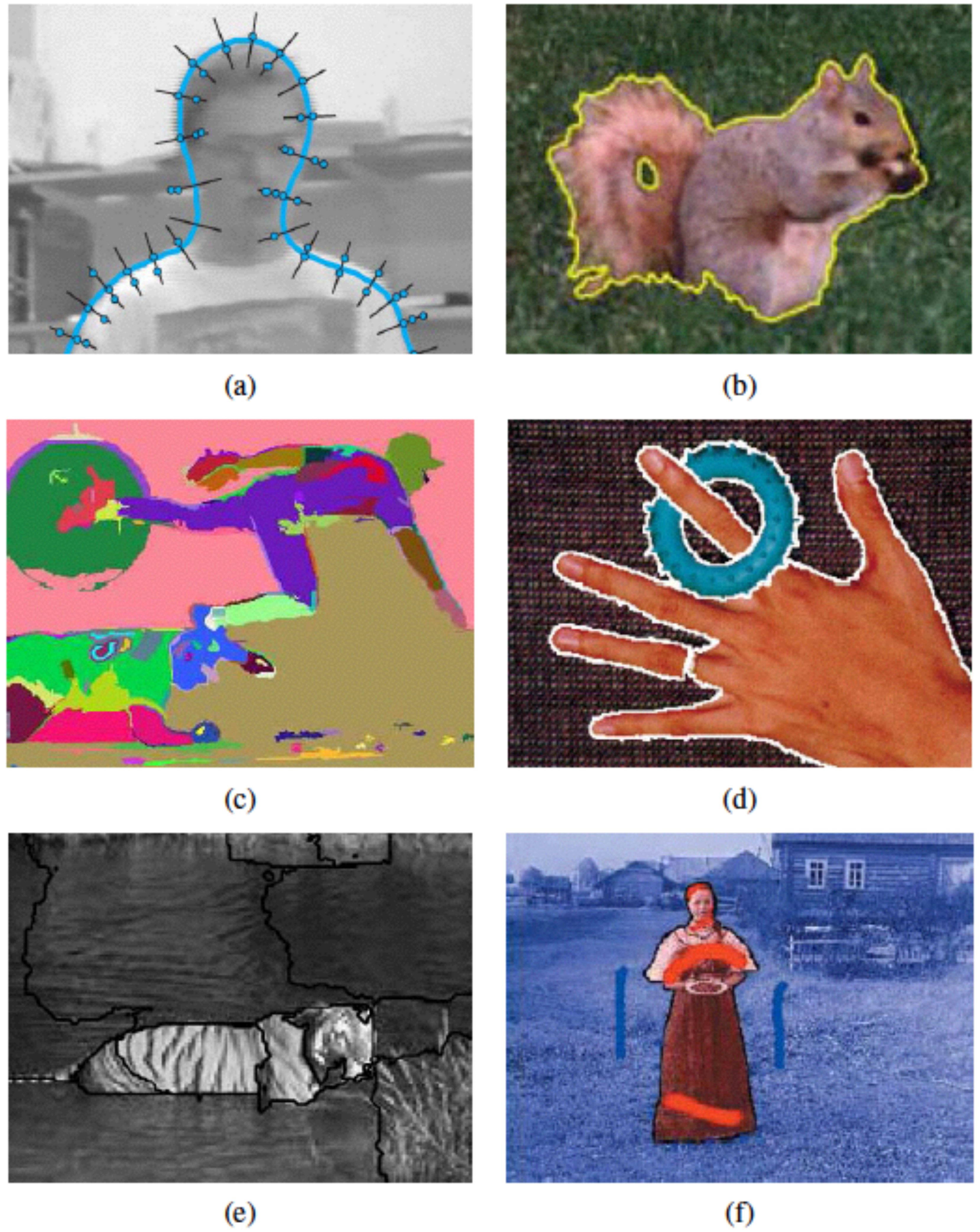
Common Form of Objective Functions

- More broadly:
  - One term that penalizes solutions that don't agree with the data
  - One term that penalizes solutions that are *a priori* unlikely
  - A weight that controls the relative importance of these terms

$$\mathrm{data}(\mathbf{f}, \mathbf{g}) + \lambda \; \mathrm{prior}(\mathbf{f})$$

Segmentation

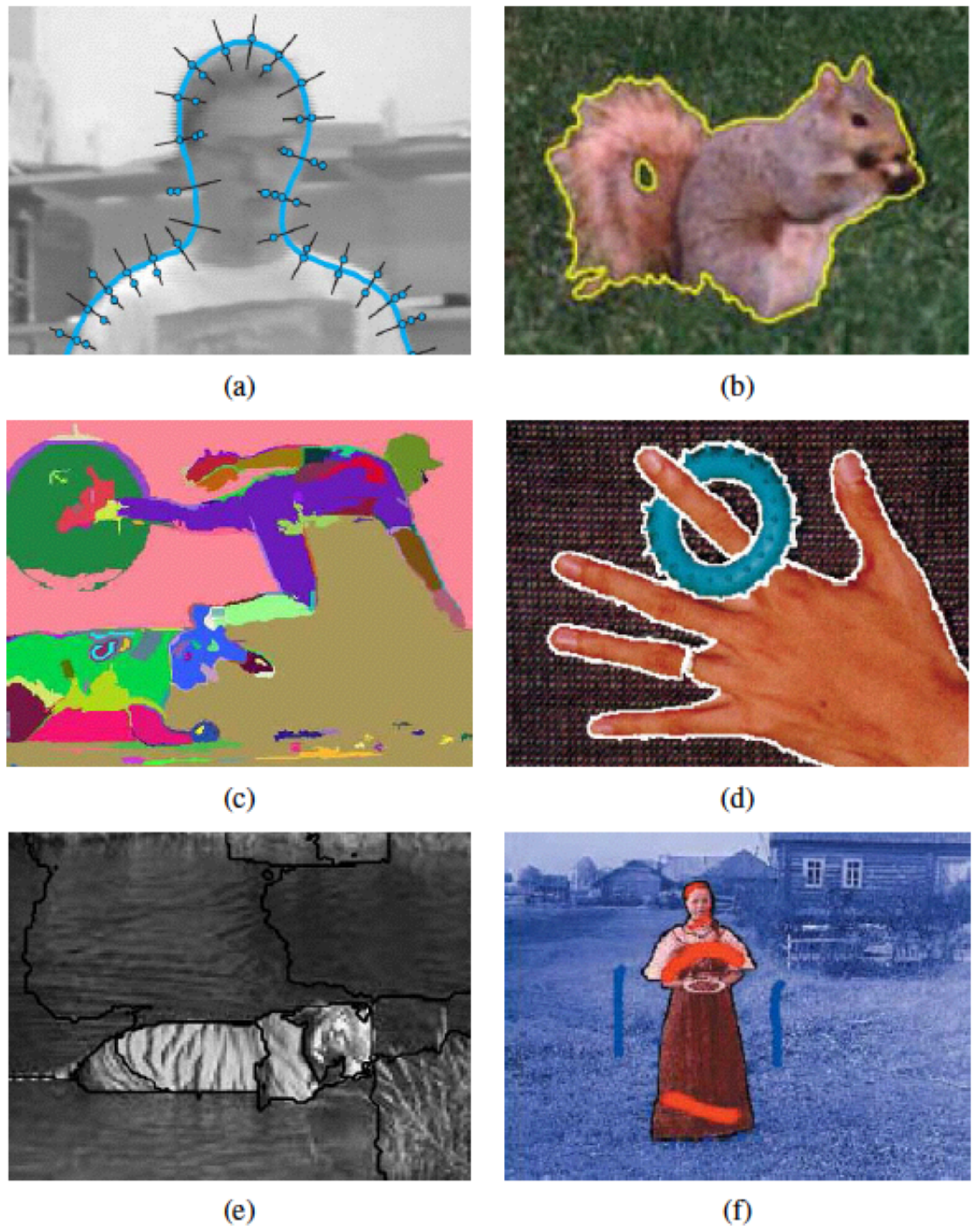
- *Segmentation* is the labeling of each pixel as part of an object region
    - Most often a complete partitioning
    - Often foreground/background, could be multiple regions
- How much user interaction?
    - Supervised = user indicates something about intended result
    - Unsupervised = no user input



Next lab!

General Approaches

- Edge-based
  - Edge detection
  - Linking, fitting, or other method for finding a continuous contour
- Region-based
  - Pixel grouping or clustering
  - Pixel-to-pixel similarity (“affinity”)
- Hybrid: edge + regions

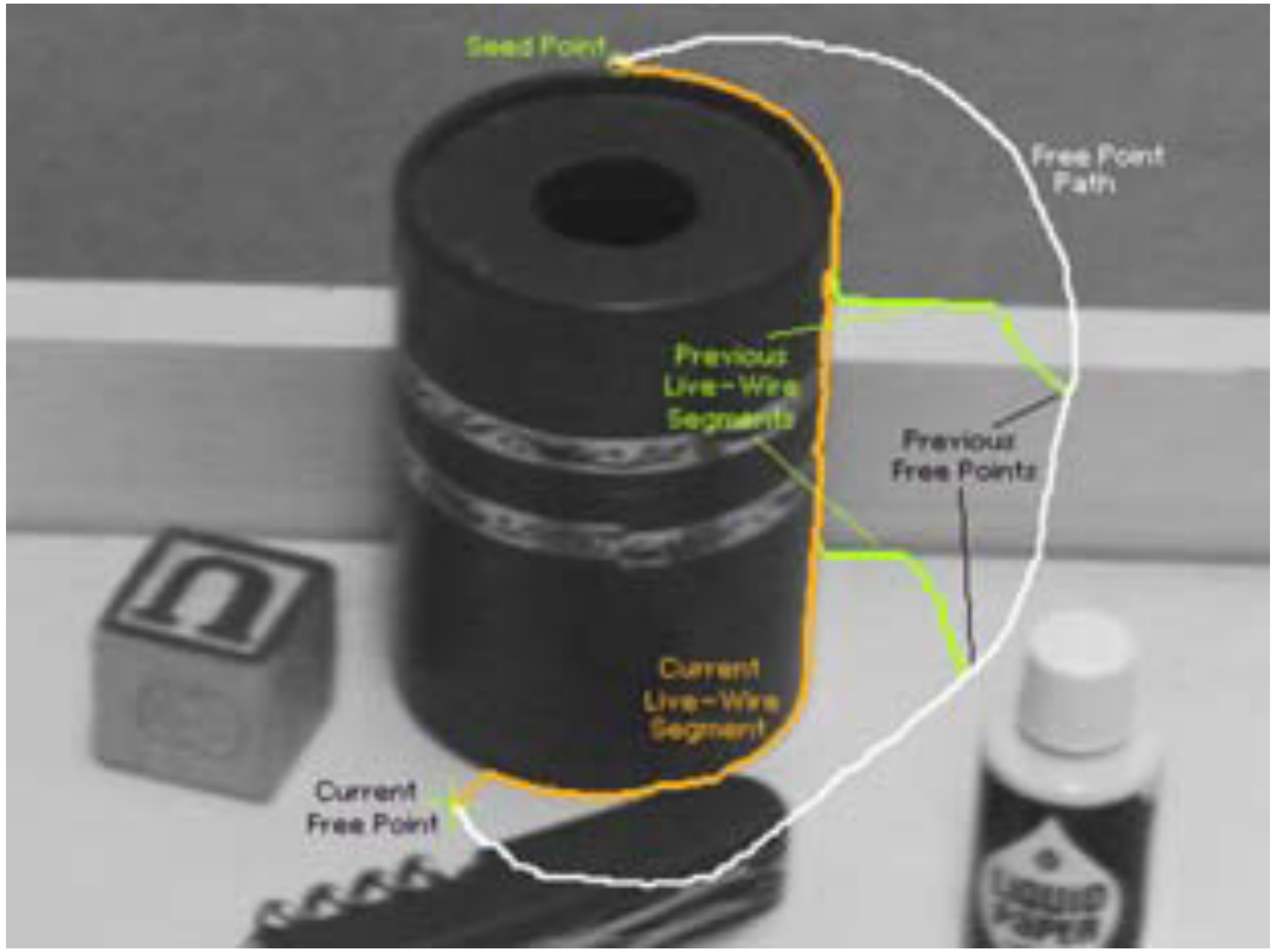


Edge-Based

- Bottom-up linking
- Top-down fitting
  (e.g., Hough transform)
- Optimization-based
  - Contour should fit edges well
  - Contour should be smooth or
    fit other priors

Path-Based Optimization

- Goal is to maximize fit of the contour to image edges
- Can frame optimization as a least-cost path through a graph
- Example: Intelligent Scissors (Mortensen & Barrett, 1995)
- Photoshop's "magnetic lasso"

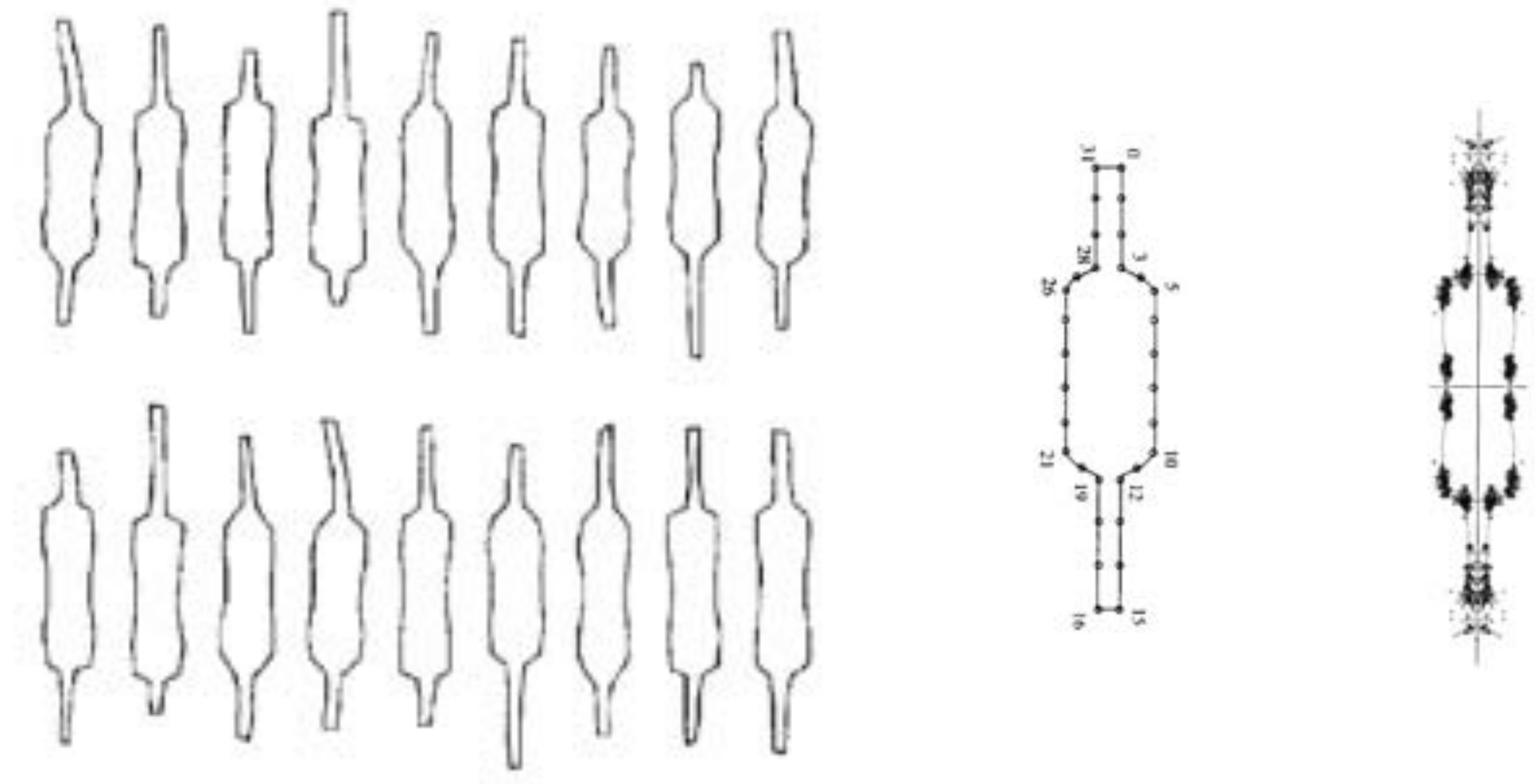


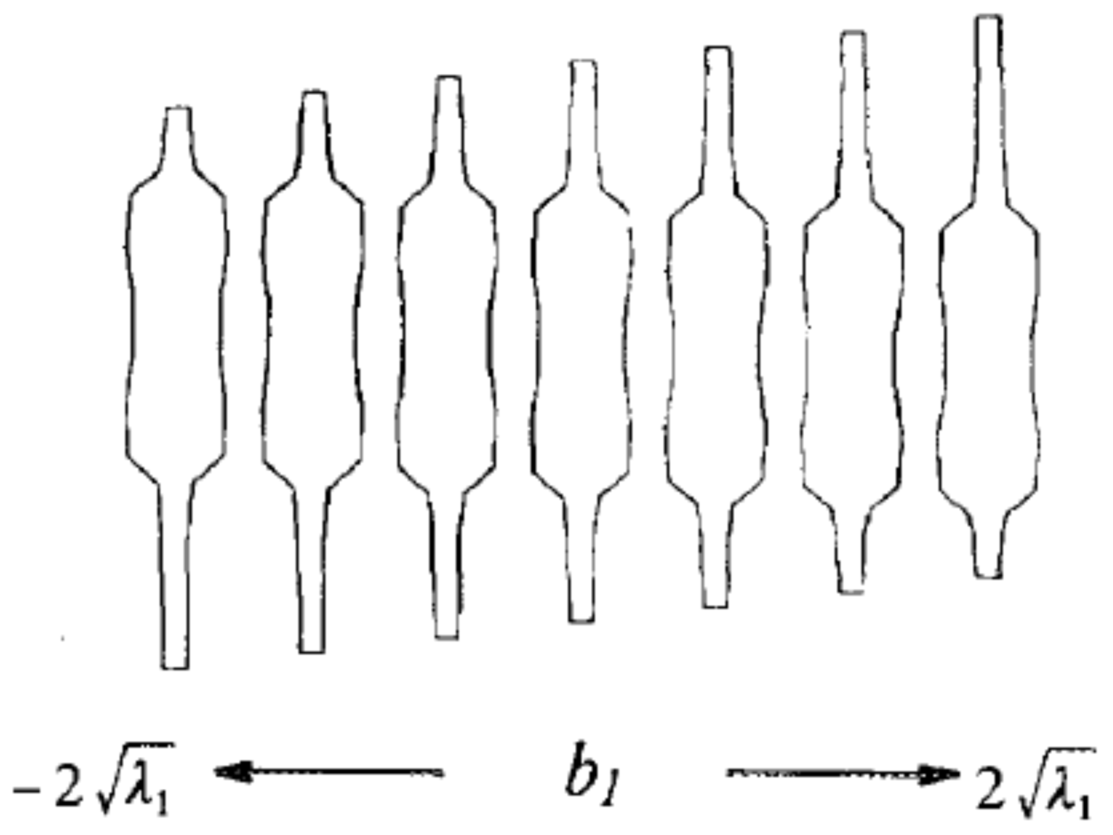
Bayesian-Like Optimization

- Optimization-based
  - Contour should fit edges well
  - Contour should be smooth
- Most common is *deformable models* or *active contours*
  - Start with an initial approximation (user, previous frame, etc.)
  - Iteratively tweak to improve fit to both edges and priors

Shape Priors

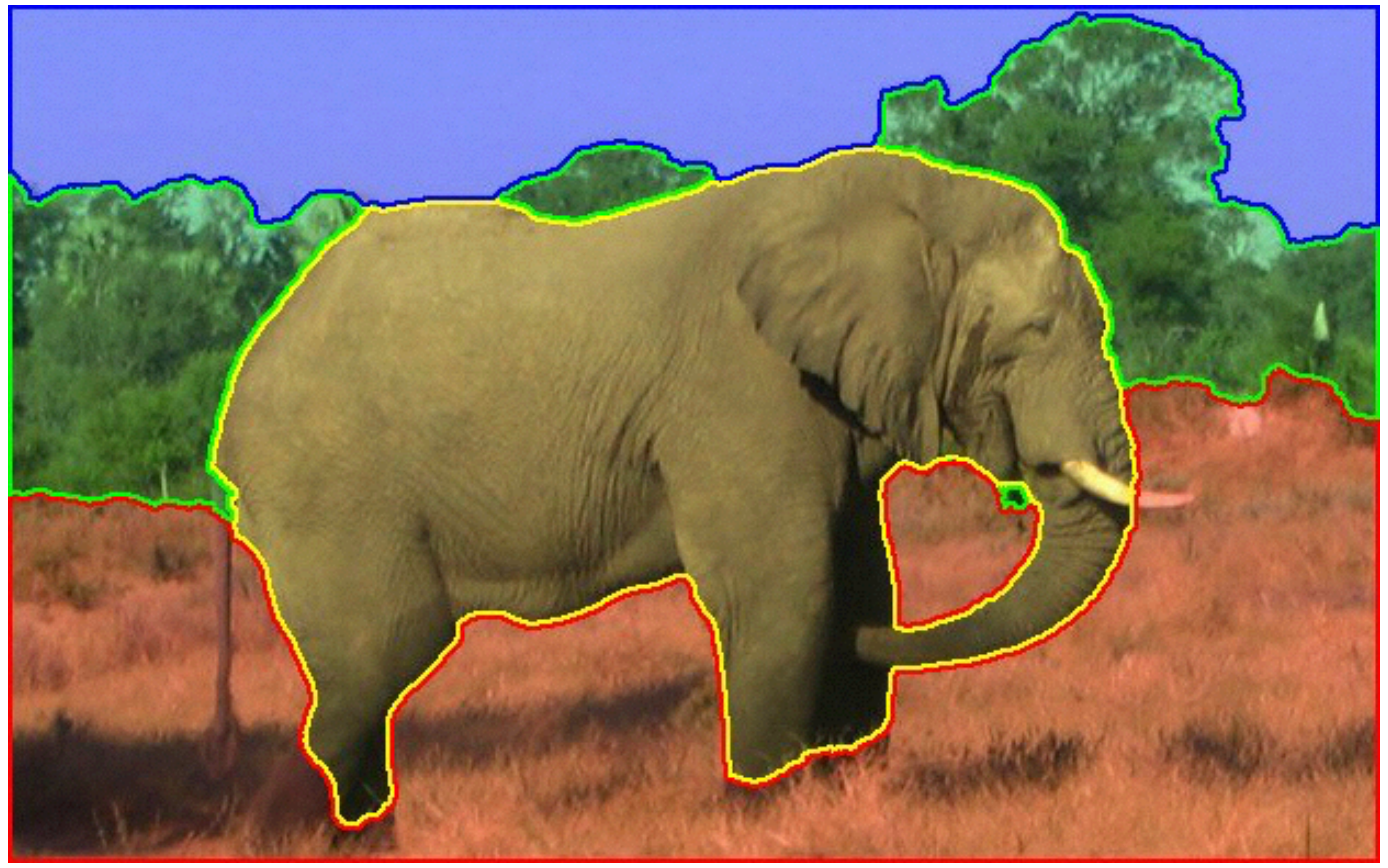
- Besides simple smoothness priors, you can use *shape priors*
- Penalizes solutions that don't fit an approximately known shape
- Can be rigid or flexible
- Can be based on statistical variance

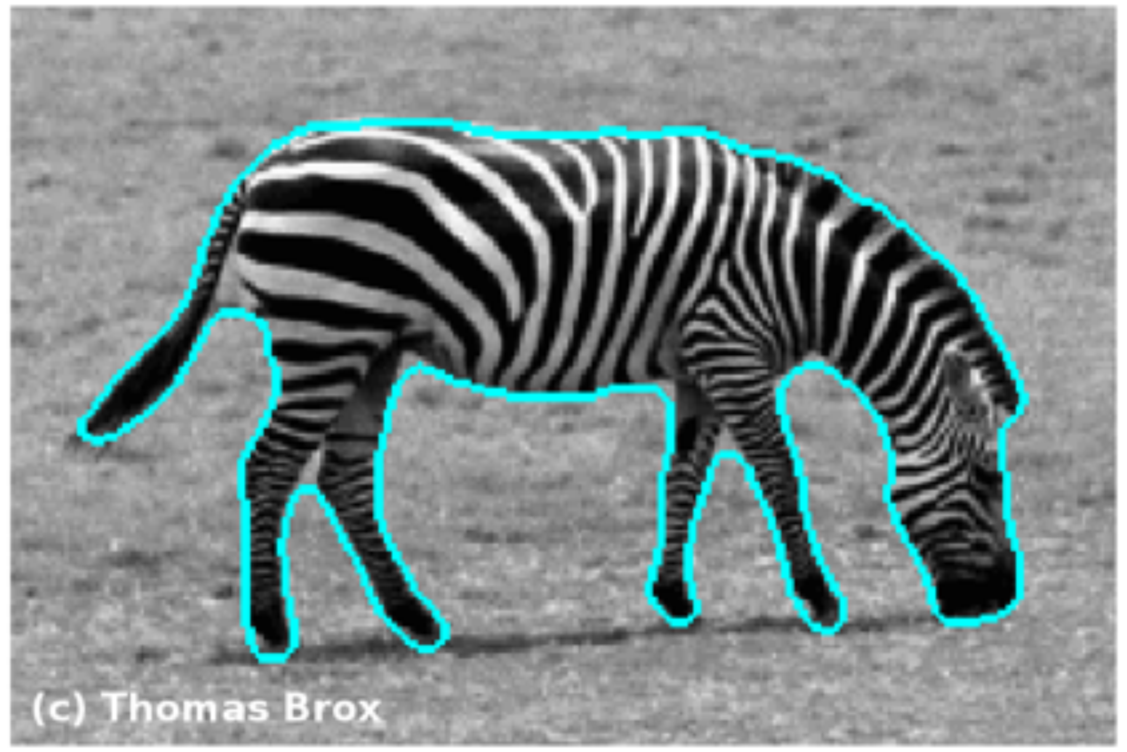




Region-Based

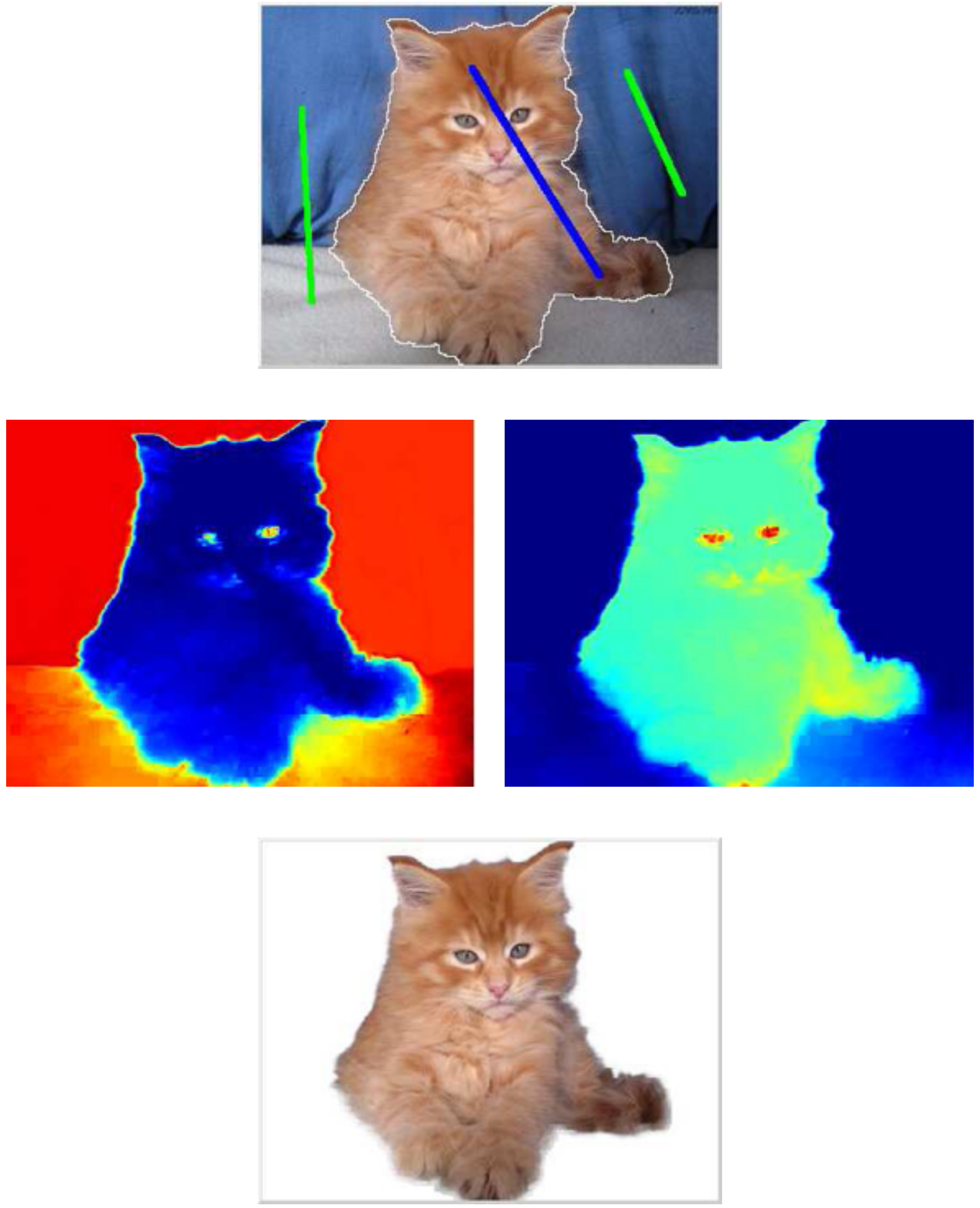
- Core idea: group pixels based on common properties (“affinity”)
- Color, texture, motion, etc.

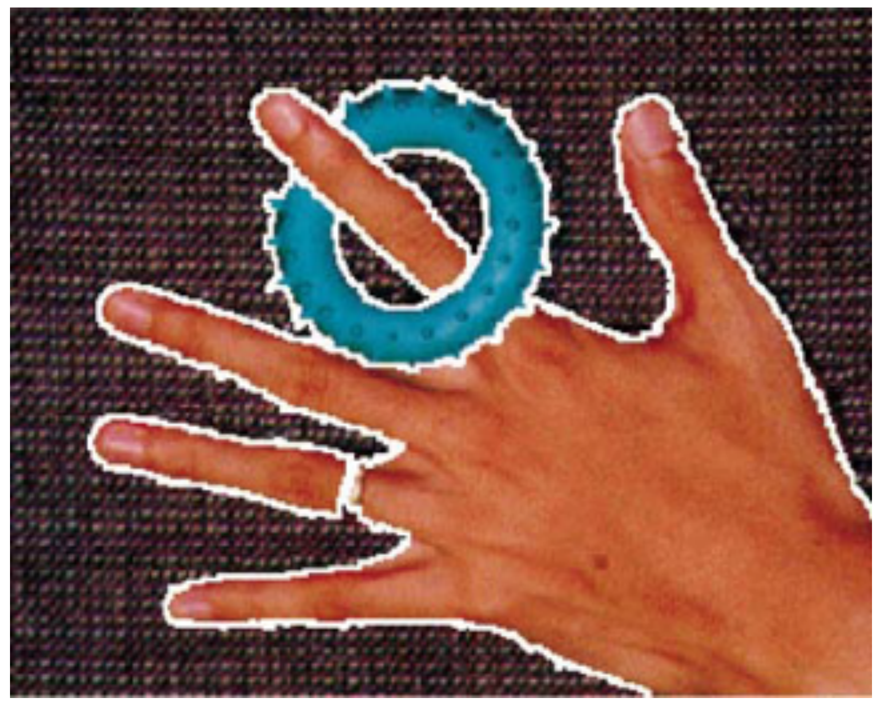




Region-Based

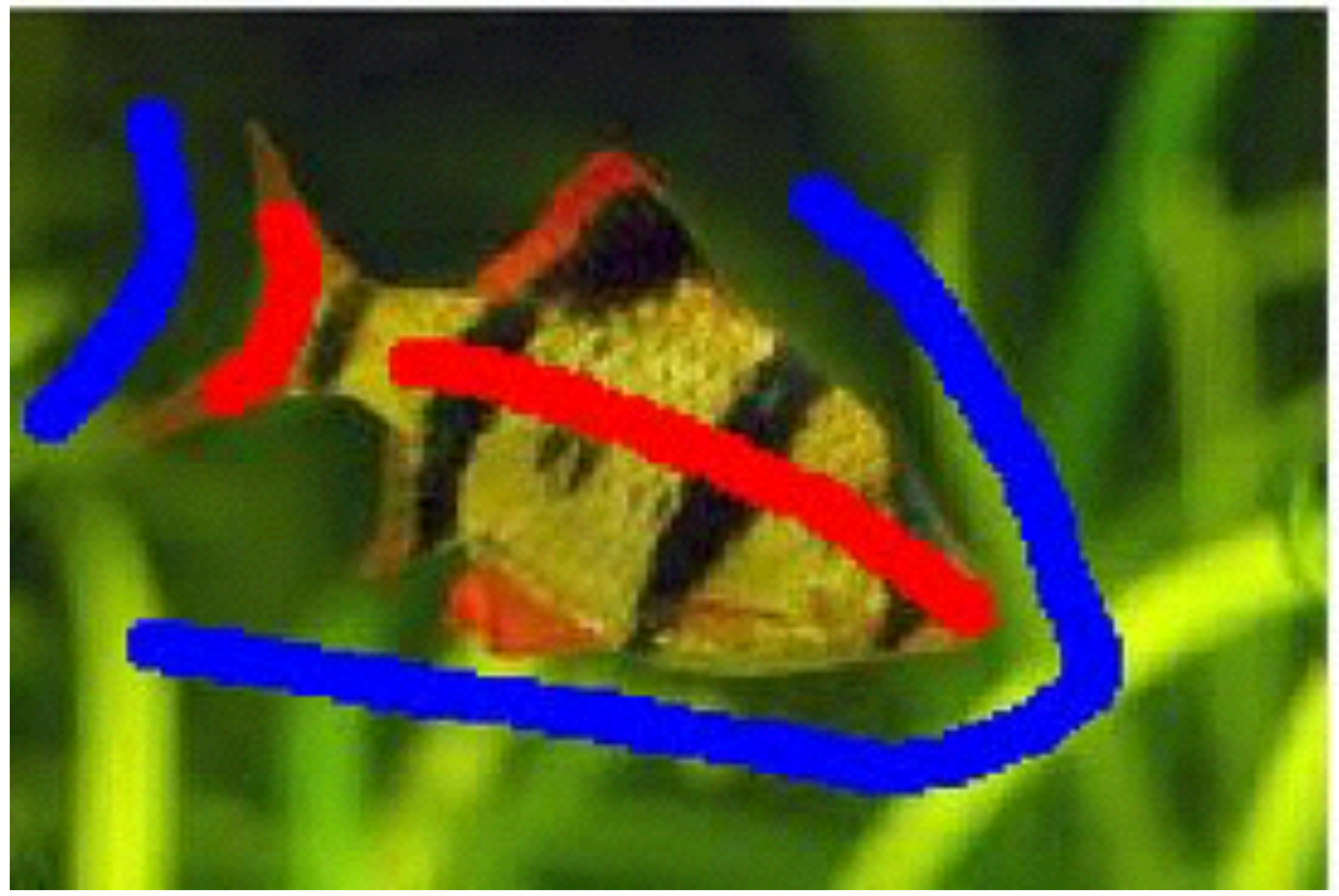
- Region growing
- Gradient watersheds
- Distance-based approaches
  - Magic wand
  - Intelligent Paint
  - Geodesics
- Random walks
- Graph spectral approaches
- Clustering
  - Mean-shift clustering





Graph-Cut Segmentation

- User selected “strokes” to indicate/seed regions
- Bayesian-like cost function
    - Likelihood: labeled pixels should look like the seeds for their region
    - Prior: similar neighbors should go in similar regions (smoothness)
- Turn into an *optimal graph cut* problem



Graph-Cut Segmentation

- Let
  - P = set of pixels
  - L = set of links between neighbors
  - A = a vector of pixel label assignments
- Minimize objective function E(A):
  - Region term penalizes labeling pixels that don't look like the seeds
  - Boundary term penalizes labeling similar neighbors with different labels

$$E(A) = R(A) + \lambda B(A)$$

$$R(A) = \sum_{p \in P} R(p)$$

$$B(A) = \sum_{(p,q) \in L, p \neq q} B(p,q)$$

Region Term

- Use user-specified seed pixels as examples of the colors in the foreground / background
- One way is to use the negative of the log-likelihood of pixels from the seeded regions looking like the pixel p
- Can use other forms, but the key is to penalize labeling pixels as part of a region when they don't look like the seeds for that region

$$R(p) = -\ln P(I_p \mid A_p)$$

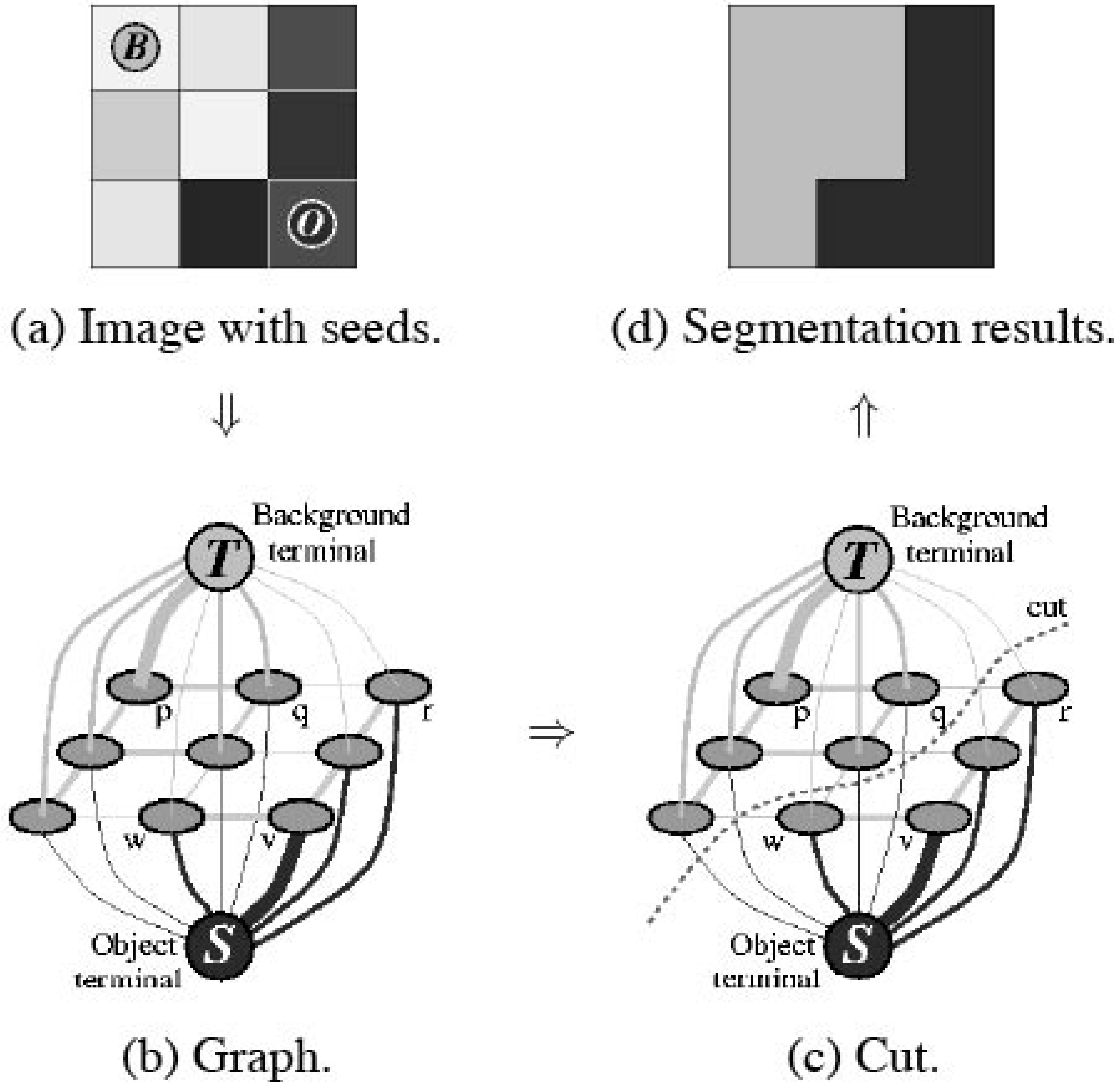
Boundary Term

- Penalize labeling adjacent similar pixels with different labels
- Most common: use a decreasing function of the difference between the two pixels' colors

$$B(p,q) = \begin{cases} e^{-\|I_p - I_q\|^2 / 2\sigma^2} & \text{if } A_p \neq A_q \\ 0 & \text{otherwise} \end{cases}$$

Optimization

- A node for each pixel
- Additional terminal node for each region
- Arcs between neighboring pixels (n-links)
- Arcs between terminals and every pixel (t-links)
- Cost to cut n-link is based on pixel similarity *B(p,q)*
- Cost to cut t-link is based on swapped *R(p)*
- Find the *minimum-cost cut* that separates the two terminal nodes



Next Lab

- Manually specify seed pixel regions (no interactive method needed)
- You build the graph
- You compute the t-link costs based on similarity to the seed pixels
- You compute the n-link costs based on similarity between neighbors
- Use an existing implementation of a graph-cut algorithm (PyMaxflow) to solve for the minimum-cost cut
- Read out the labels and then create a binary image for display