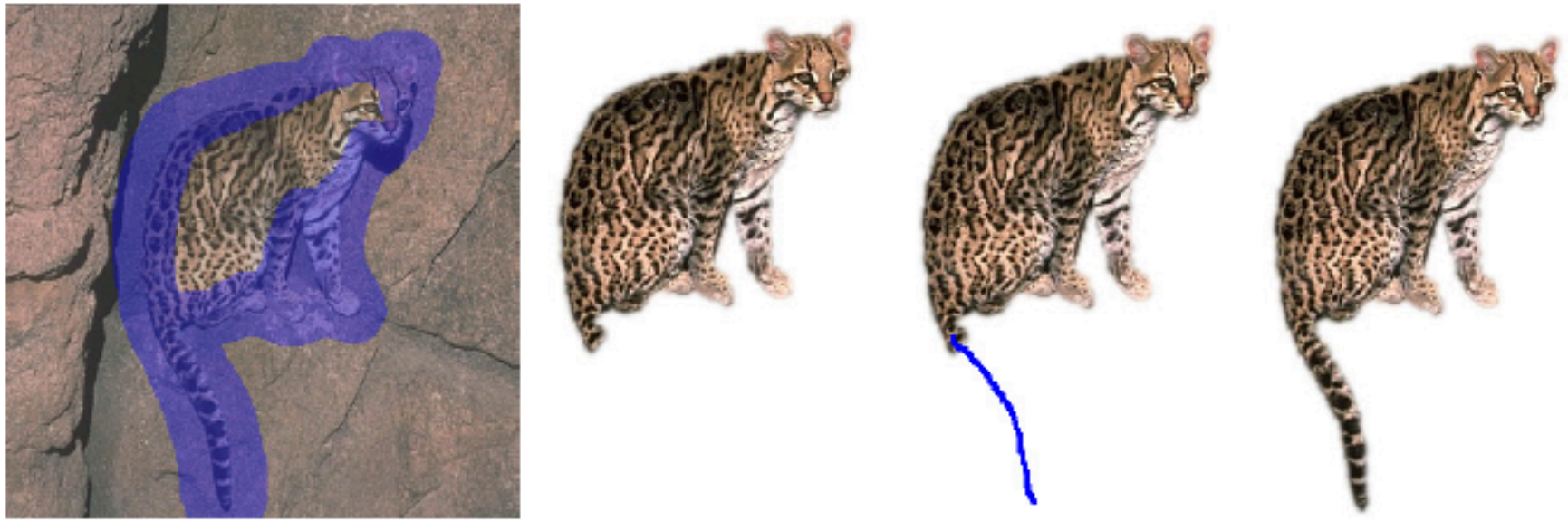
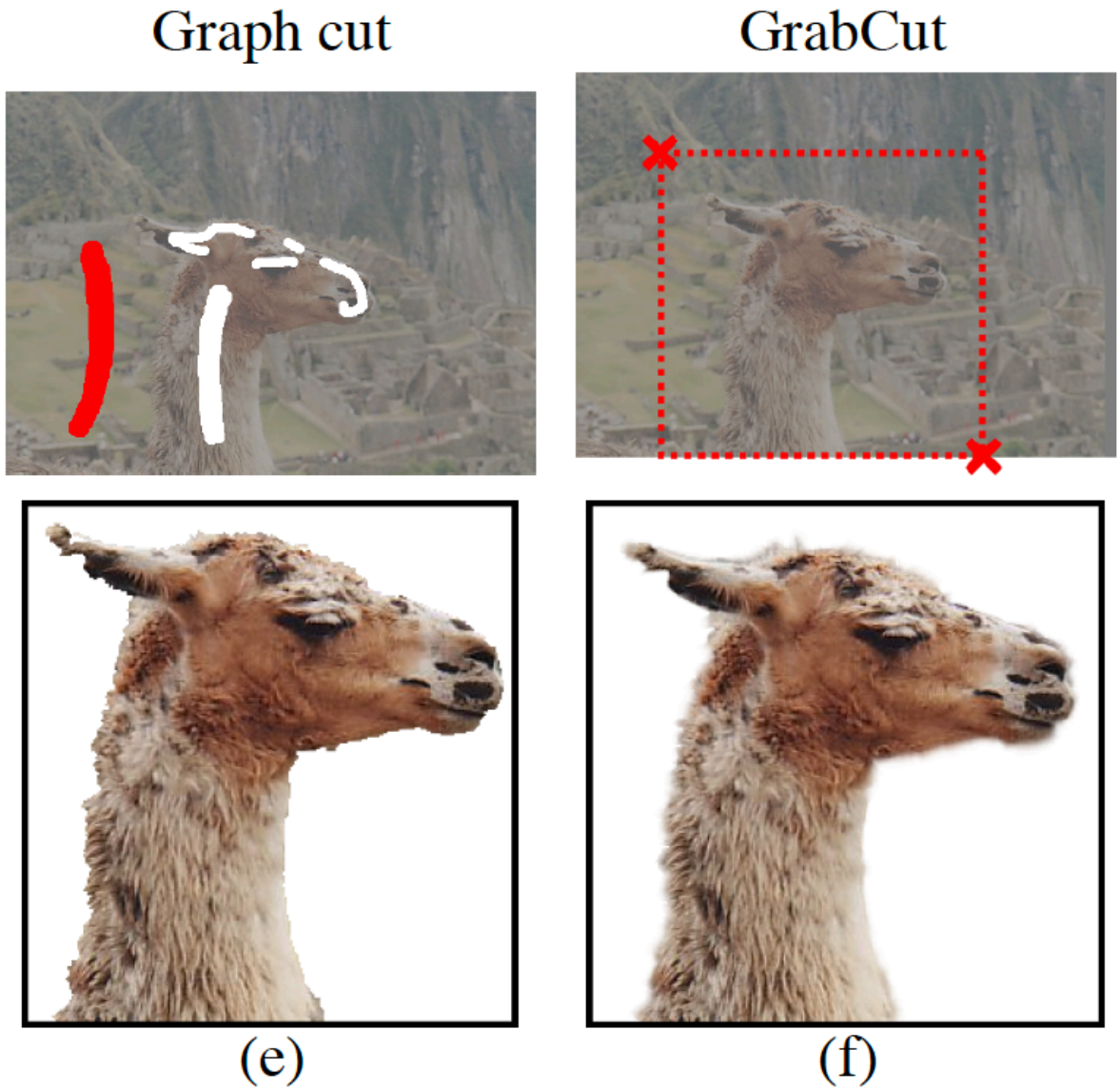


(a) (b) (c) (d)

Fig. 1. Illustrating the GMMRF algorithm for interactive segmentation. The user draws a fat pen trail enclosing the object boundary (a), marked in blue. This defines the "trimap" with foreground/background/unclassified labels. The GMMRF algorithm produces (b). Missing parts of the object can be added efficiently: the user roughly applies a foreground brush (c), marked in blue, and the GMMRF method adds the whole region (d).

Blake et al, 04 (on web page - in homeworks)

Foreground/background from markup



Rother et al, 04

Segmentation framework

Parameters

Pixel values

Minimize: $\mathbf{E}(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z})$.

Smoothness term: quadratic in alpha

Data term: linear in alpha

Subject to:

$$\alpha_n = \begin{cases} 0 & \text{pixel } n \text{ background} \\ 1 & \text{pixel } n \text{ foreground} \end{cases}$$

Segmentation framework

Probability of obtaining z_n given alpha_n

$$U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = \sum_n -\log h(z_n; \alpha_n)\,.$$

The smoothness term can be written as

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} dis(m,n)^{-1}\,[\alpha_n \neq \alpha_m] \exp -\beta (z_m - z_n)^2$$

Segmentation framework

$$U(\underline{\alpha}, \underline{\theta}, \mathbf{z}) = \sum_n -\log h(z_n; \alpha_n) .$$

The smoothness term can be written as

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} dis(m,n)^{-1} \, [\alpha_n \neq \alpha_m] \exp -\beta (z_m - z_n)$$

distance between pixel locations

Contrast between pixels

Notice: it is better to agree than to disagree
(cost goes down)

Key issues

- Impute probability model from markup
- Given model, compute solution

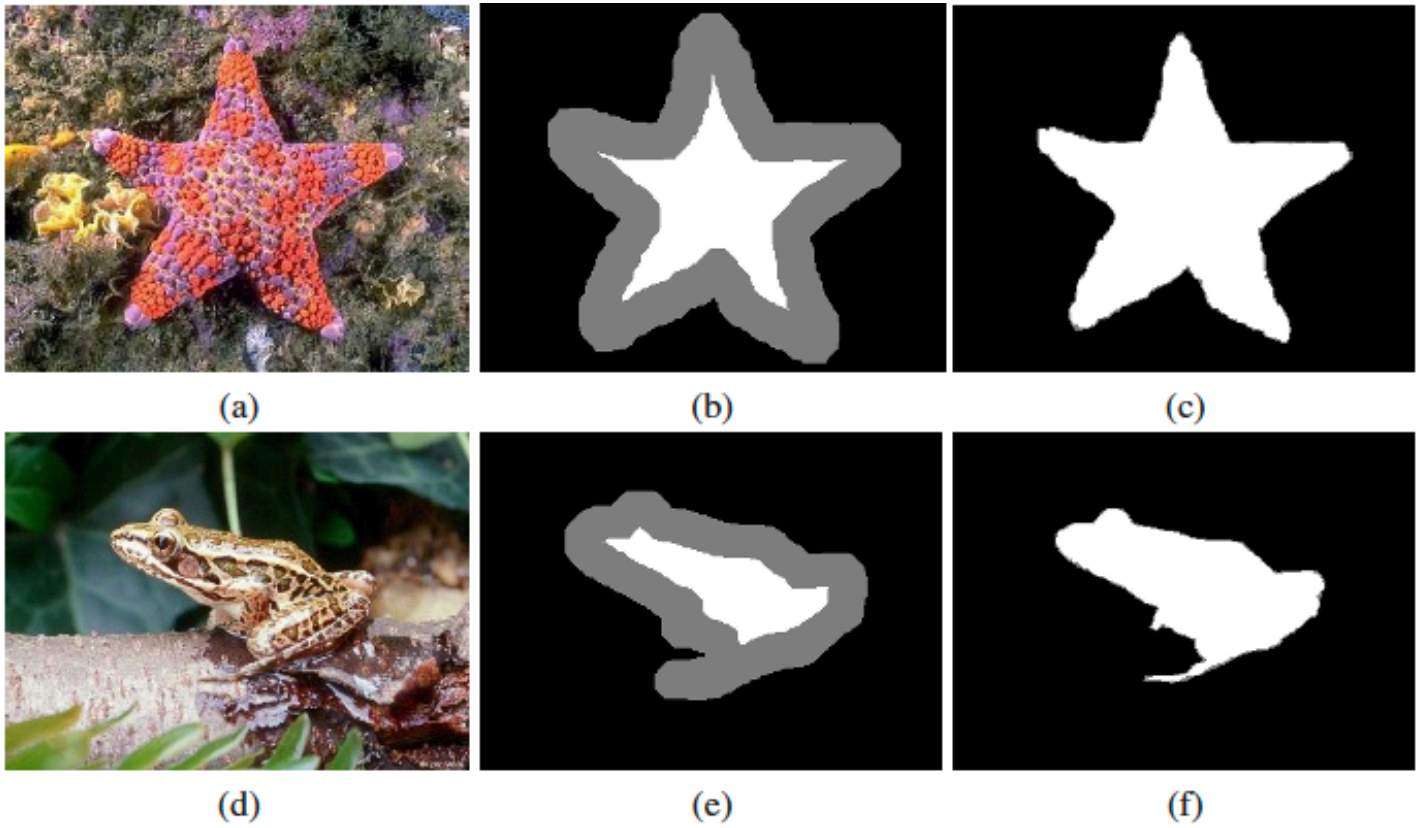


Fig. 2. (a,d) Two images from the test database. (b,e) User defined trimap with foreground (white), background (black) and unclassified (grey). (c,f) Expert trimap which classifies pixels into foreground (white), background (black) and unknown (grey); unknown here refers to pixels too close to the object boundary for the expert to classify, including mixed pixels.

Blake et al, 04 (on web page - in homeworks)

Simplest probability model

- Markup is a tri-map
  - build model for foreground, background
    - histogram; single gaussian for color; mixture of gaussians for color
- now for each pixel, the data term is

$$\alpha_n \left[-\log p(z_n|\text{foreground}) \right] + (1 - \alpha_n) \left[-\log p(z_n|\text{background}) \right]$$

Smoothness term

$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2 .$$

Which is:

$$\sum_{m,n\in C} (\alpha_n(1-\alpha_m) + (1-\alpha_n)\alpha_m) \left[\gamma \exp\left(-\beta \parallel z_m - z_n \parallel^2\right)\right]$$

Which is:

$$(\text{linear term in } \alpha) - \sum_{m,n\in C} (\alpha_n \alpha_m) \left[\gamma \exp\left(-\beta \parallel z_m - z_n \parallel^2\right)\right]$$

Underlying optimization problem

positive

Minimize:

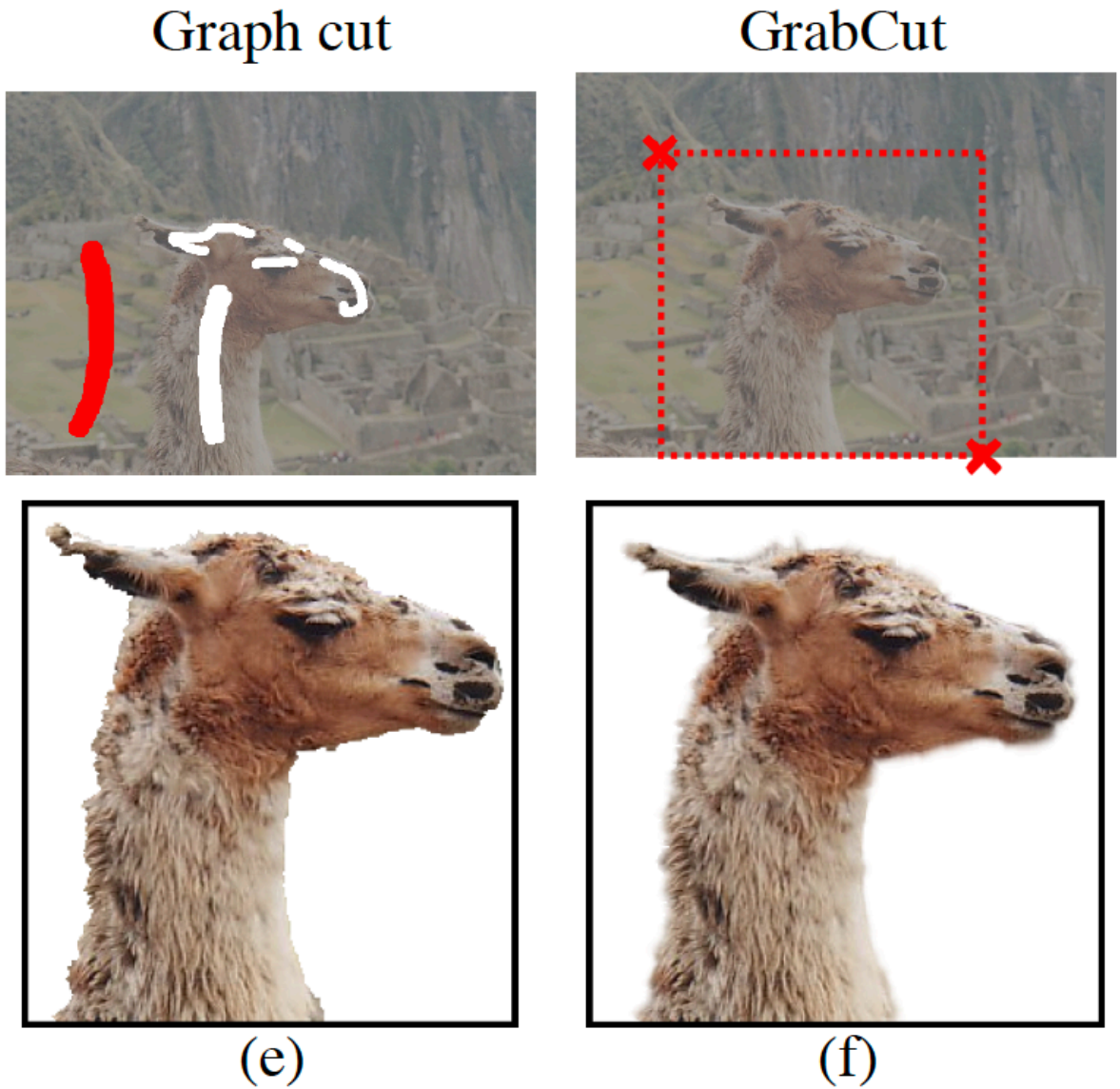
$$(\text{linear term in } \alpha) - \sum_{m,n \in C} (\alpha_n \alpha_m) \, [w_{m,n}]$$

Subject to:

$$\alpha_n = \begin{cases} 0 & \text{pixel } n \text{ background} \\ 1 & \text{pixel } n \text{ foreground} \end{cases}$$

This, by earlier results, is a max-flow/min-cut problem
(steps: turn it into a linear program; notice that inconvenient constraints are not active, so that soln will be at integer point; OR directly)

Foreground/background from markup



Rother et al, 04

More interesting probability model

- Have 2k normal distributions
  - k each: for foreground, background
  - in color or some other feature vector
  - parameters: mean, covariance
- At each pixel, have an integer
  - assigns pixel to a normal $f_n \in \{1, 2, ..., k\}$
- which gives

$$\alpha_n \left[-\log p(z_n | \text{foreground}, f_n) \right] + (1 - \alpha_n) \left[-\log p(z_n | \text{background}, f_n) \right]$$

Steps:

- Iterate:
  - Estimate f_n for each pixel
    - given alpha_n, mean, covariance for each normal
      - use normal with largest likelihood
  - Estimate mean, covariance for each normal
    - given f_n, alpha_n
  - Estimate alpha_n for each pixel
    - by cutting graph
- Until energy does not decline