Microsoft Research Cambridge

SIGGRAPH2004

# GrabCut

## Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother
Vladimir Kolmogorov
Andrew Blake

**Microsoft Research Cambridge-UK**

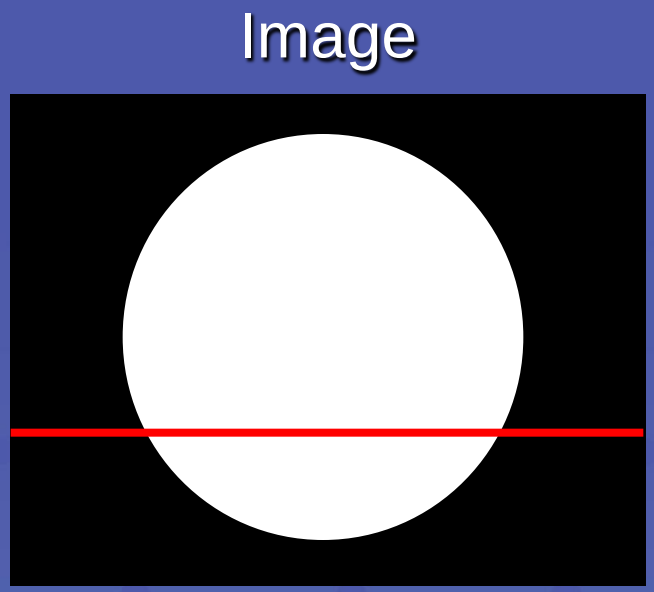
# Photomontage

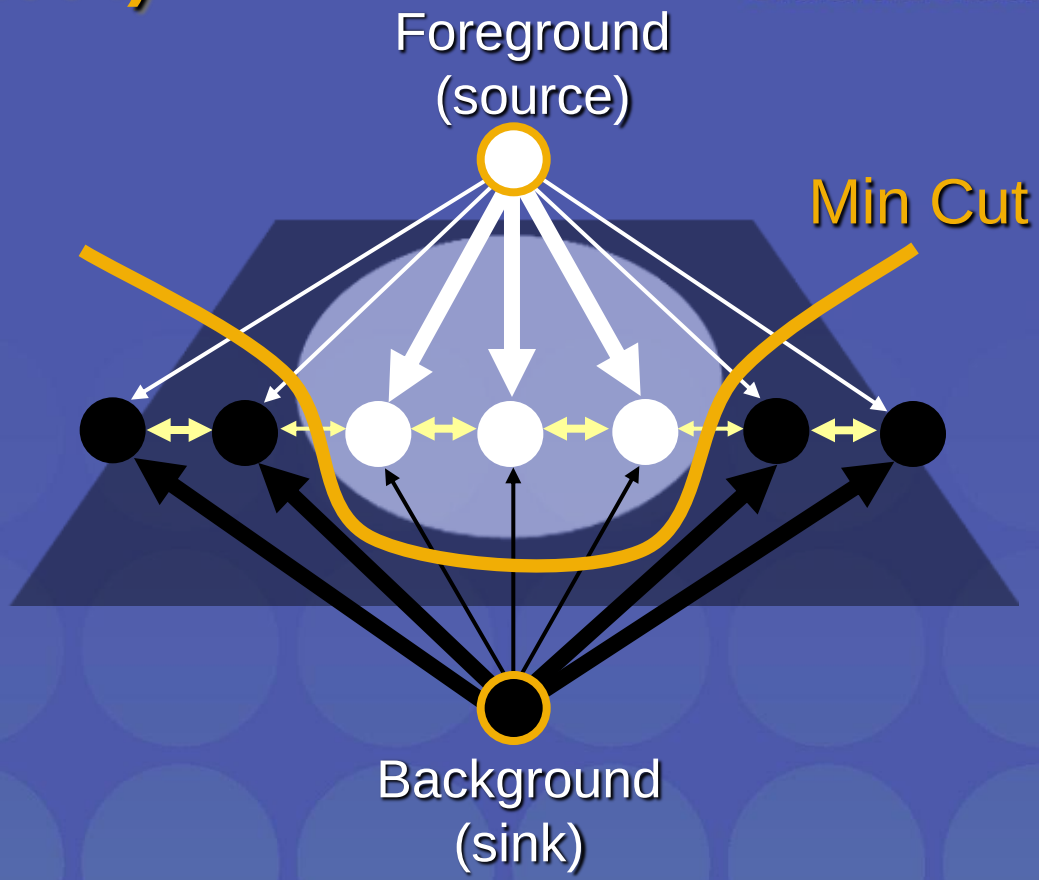
# Problem

Fast &
Accurate ?

# Graph Cuts

## Boykov and Jolly (2001)




***Cut:*** separating source and sink; Energy: collection of edges

***Min Cut:*** Global minimal enegry in polynomial time

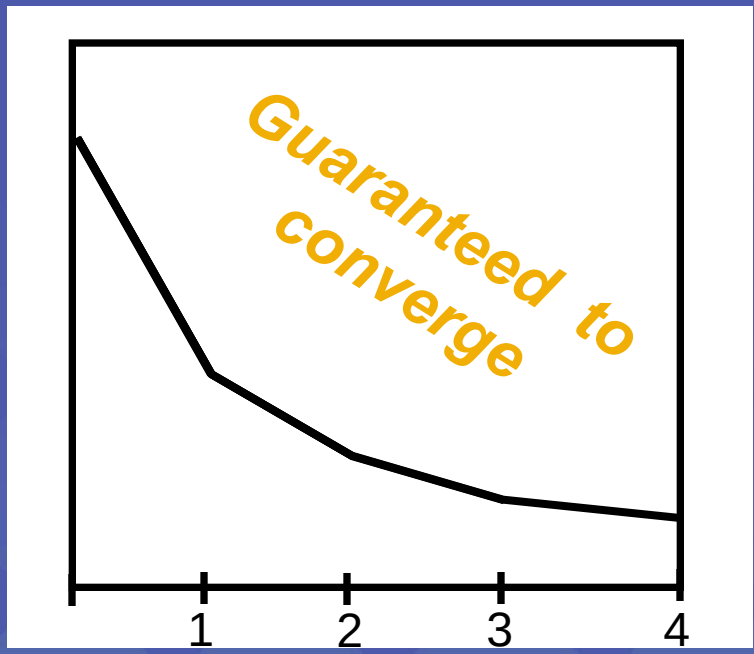
# Iterated Graph Cut


User Initialisation

**K-means for learning colour distributions**

**Graph cuts to infer the segmentation**

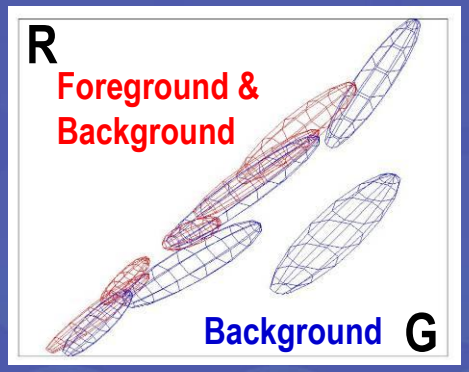
# Iterated Graph Cuts



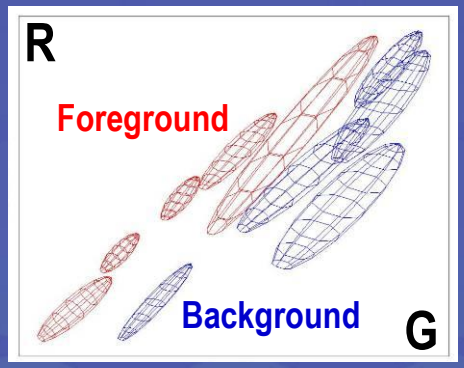

Result

Energy after each Iteration

# Colour Model



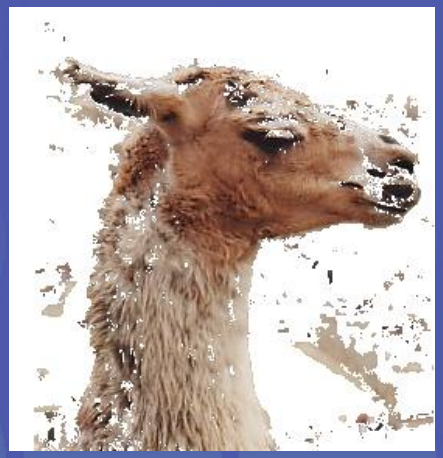

Iterated
graph cut




Gaussian Mixture Model (typically 5-8 components)

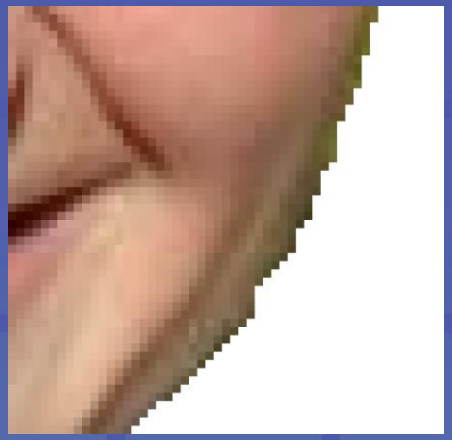
# Coherence Model

An object is a coherent set of pixels:



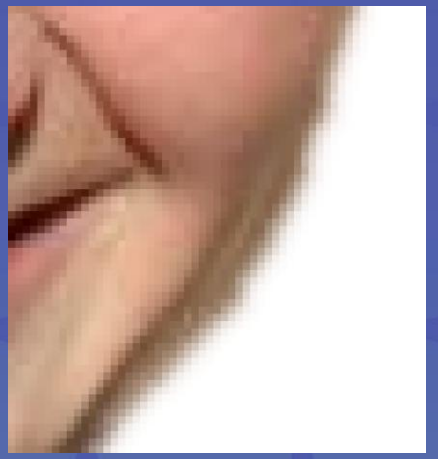
**Blake et al. (2004):** Learn jointly

# Border Matting



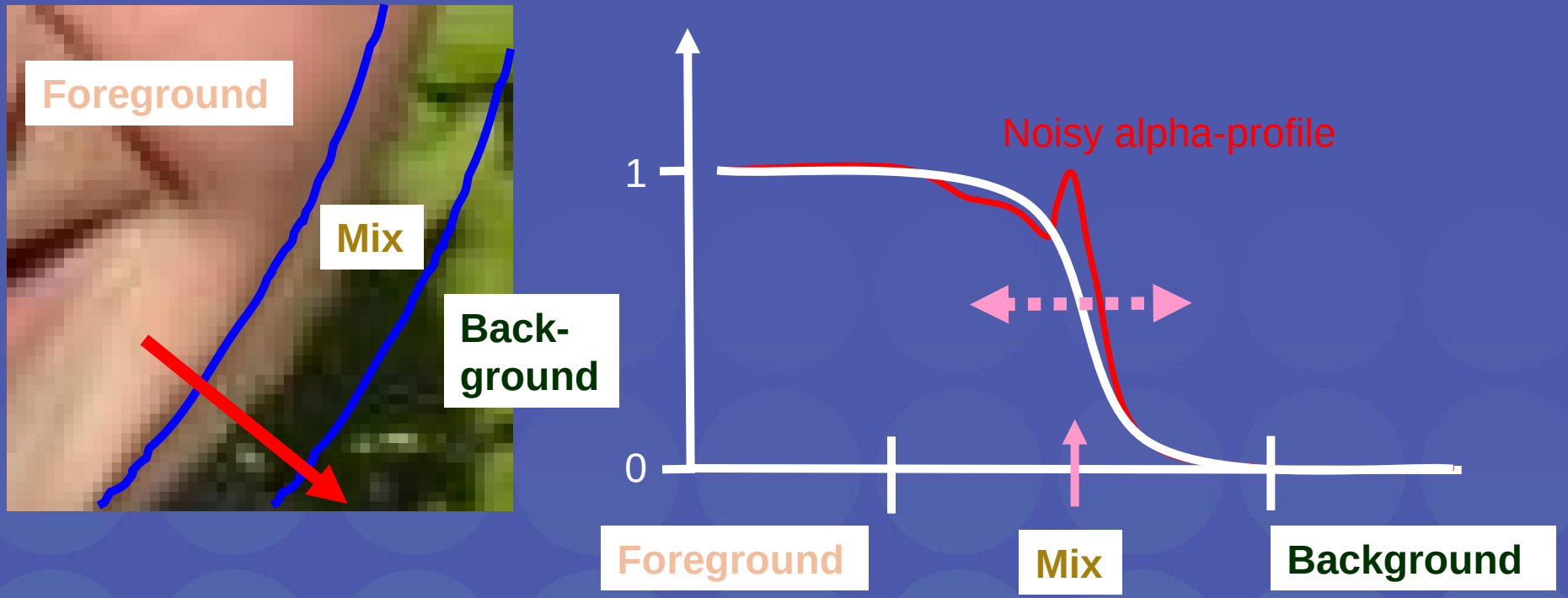
Hard Segmentation

Automatic Trimap



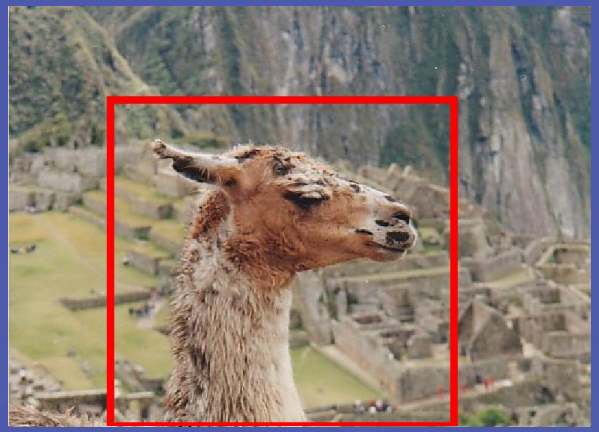
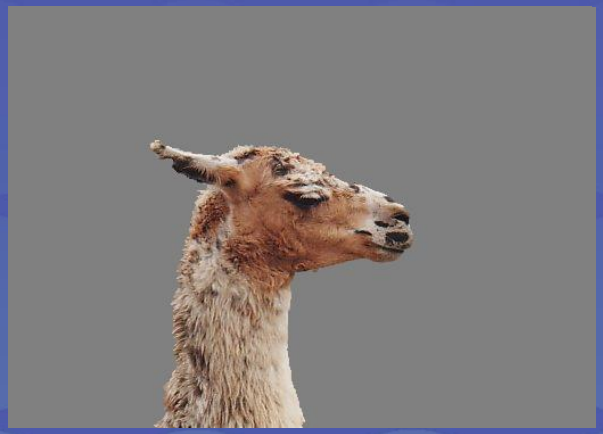
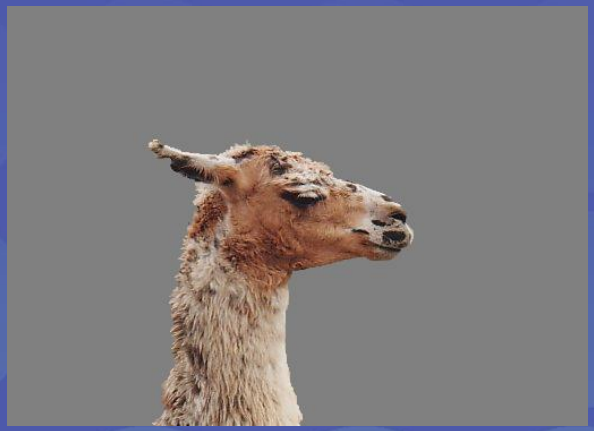
Soft Segmentation

# Border Matting

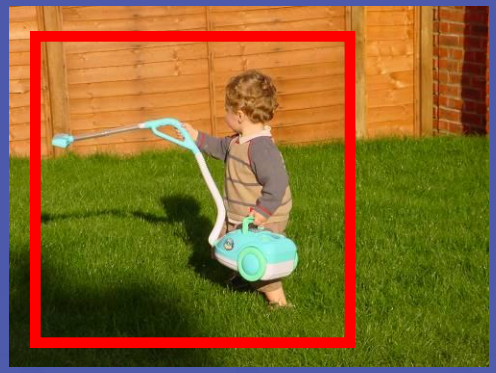
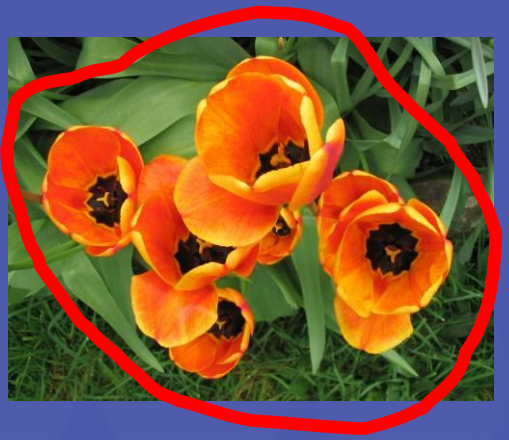



Fit a smooth alpha-profile with parameters

# Comparison


| | Boykov and Jolly (2001) | GrabCut |
|---|---|---|
| User Input |  |  |
| Result |  |  |
| | Error Rate: 0.72% | Error Rate: 0.72% |

# Moderately straightforward examples



… GrabCut completes automatically

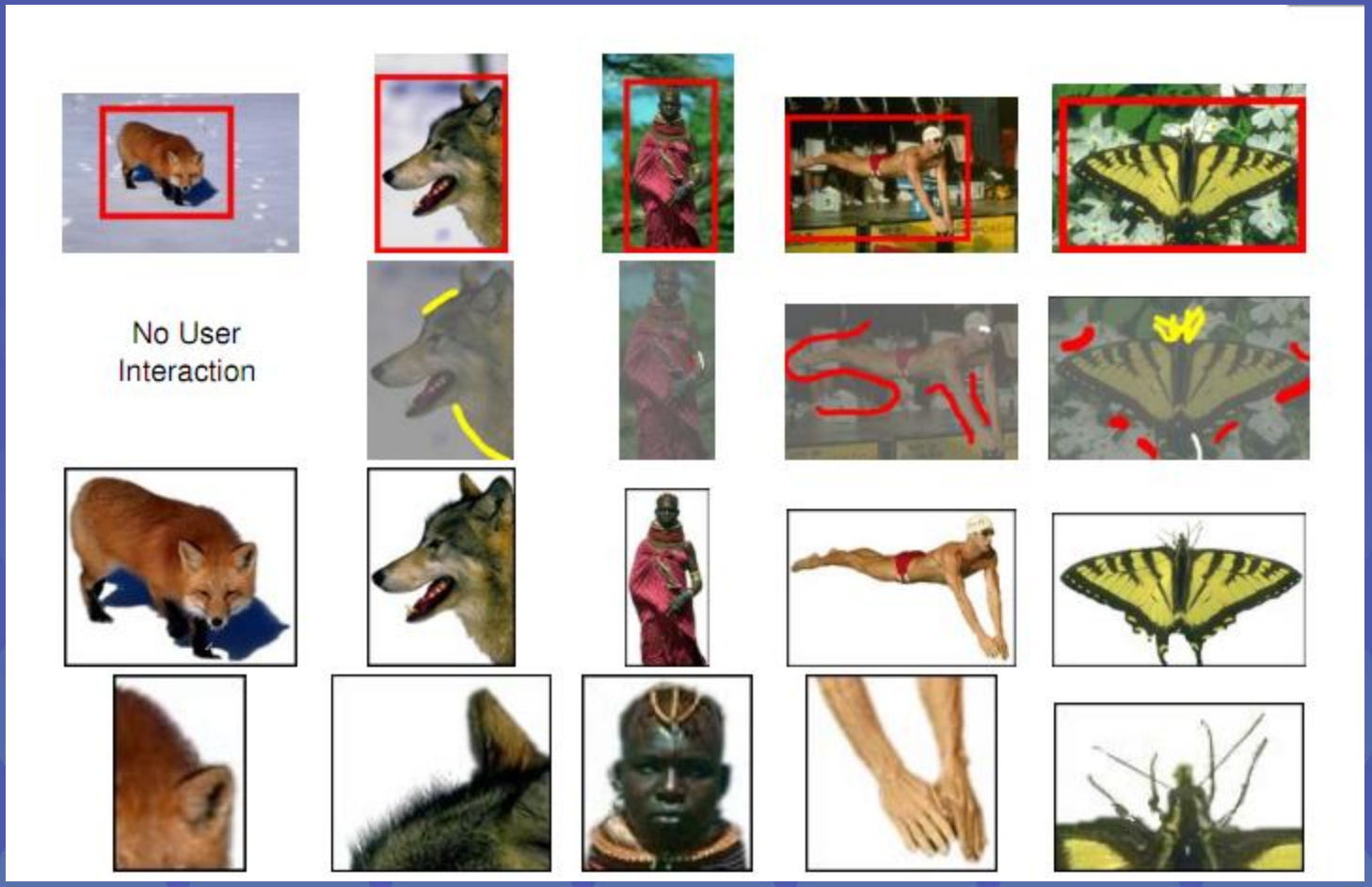
# Difficult Examples

SIGGRAPH2004

Initial Rectangle


Initial Result

# Difficult Examples

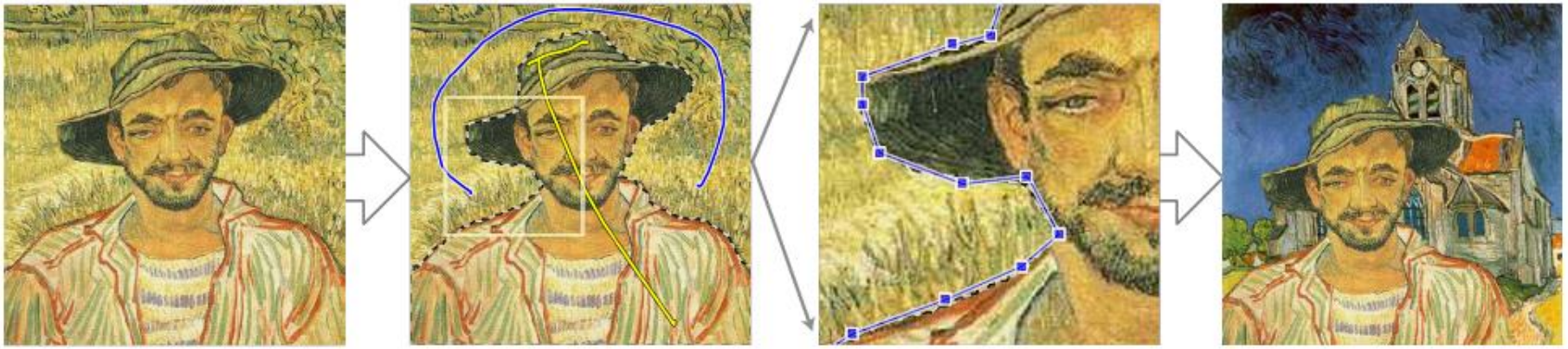


# Lazy Snapping

**Lazy Snapping**

†Yin Li* ‡Jian Sun †Chi-Keung Tang ‡Heung-Yeung Shum

†Hong Kong University of Science and Technology ‡Microsoft Research Asia



(a) Input image (b) Object Marking (c) Boundary editing (d) Output composition

# Lazy Snapping Overview

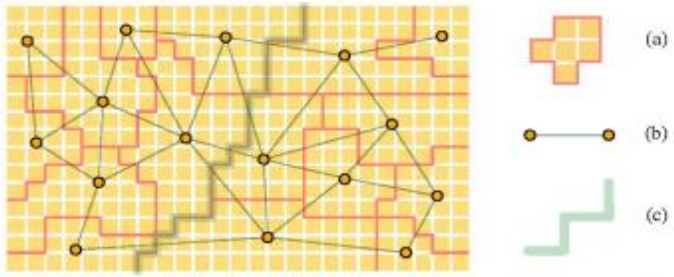
- Main Steps:
  1. Mark strokes as foreground & background (object marking)
  2. Perform Boykov & Jolly style graphcut
  3. Boundary Editing
- Familiar Formulation
  - Minimize Gibbs Energy:

$$E(X) = \sum_{i \in \mathcal{V}} E_1(x_i) + \lambda \sum_{(i,j) \in \mathcal{E}} E_2(x_i, x_j)$$

- $E_1(X)$ – Cluster (K-means), encodes color similarity of node (likelihood energy)
- $E_2(X)$ – encodes energy due to gradient along object boundary (color gradient between two nodes)

# Lazy Snapping—Pre-processing

## Watershed Algorithm



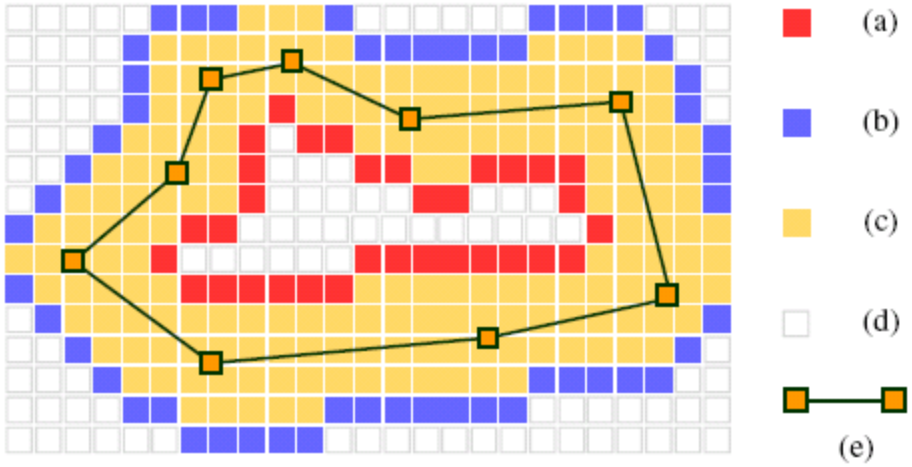

- Picture is presegmented with the Watershed Algorithm
- Graphcut works on segment instead of pixel level
- Segments are connected if one pixel within them is connected
- Reuslts are available almost instantly

# Lazy Snaping – Speedup

| Image | Dimension | Nodes Ratio | Edges Ratio | Lag with Pre-segmentation | Lag without Pre-segmentation |
|---|---|---|---|---|---|
| Boy | $(408, 600)$ | $10.7$ | $16.8$ | $0.12s$ | $0.57s$ |
| Ballet | $(440, 800)$ | $11.4$ | $18.3$ | $0.21s$ | $1.39s$ |
| Twins | $(1024, 768)$ | $20.7$ | $32.5$ | $0.25s$ | $1.82s$ |
| Girl | $(768, 1147)$ | $23.8$ | $37.6$ | $0.22s$ | $2.49s$ |
| Grandpa | $(1147, 768)$ | $19.3$ | $30.5$ | $0.22s$ | $3.56s$ |

# Boundary Editing

- Graph Cut is only step 1
- Allow user to edit boundary directly
- Formulate as Graph Cut
- Allow user to guide boundary regions w/ brush




(a) Foreground seeds $\mathcal{F}$ (b) Background seeds $\mathcal{B}$
(c) Uncertain regions $\mathcal{U}$ (d) Pixels ignored by graph cut
(e) Polygon vertices and lines

(a) (b)

# User Study

- Results
  - Easy of Use – Lazy Snapping 20% less mistakes
  - Speed – Lazy Snapping 60% less time
  - Accuracy – Lazy Snapping 60% less pixels wrong
- Feedback
  - “Almost Magic”
  - “Much Easier”
- Suggestions
  - Get rid of 2 step process
  - Make it easy to switch between graph cut strokes and boundary editing

# More Results

(a) Girl (4/2/12) (b) Ballet (4/7/14) (c) Boy (6/2/13)

(c) Grandpa (4/2/11) (d) Twins (4/4/12)