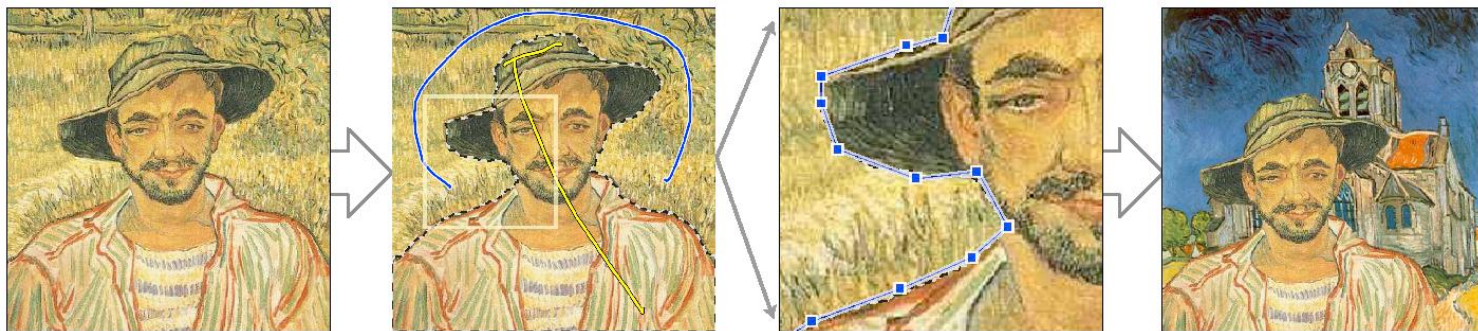


Lazy Snapping

A paper from Siggraph04

by Yin Li, Jian Sun, Chi-Keung Tang, Heung-Yeung Shum,
Microsoft Research Asia



Presented by Gerhard Röthlin



Image Cutout

- Composing a foreground object with an alternative background
- Removing a foreground object from a background, maybe with automatic means (*“Fragment-Based Image Completion”, “Image Completion with Structure Propagation”*)
- Creating a matte for further image enhancements (selective blurring, lighting conditions, ...)



Outline

- Method Overview
- Object Marking
- Boundary Editing
- User Evaluation
- Results
- Further Improvements



Related Work

- Rother,C., Bake,A., Kolmogorov,V. 2004. *Grabcut - interactive foreground extraction using iterated graph cuts*. Presented by Ursina Caluori
- Kwatra,V., Schödl,A., Essa, I., Turk,G., Bobick,A. 2003. *Graphcut textures: Image and video synthesis using graph cuts*, presented by Benjamin Sigg
- Boykov,Y., Jolly,M., 2001. *Interactive graph cuts for optimal boundary & region segmentation of objects in n-d images*
- Boykov,Y., Kolmogorov,V., 2001. *An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision*
- Many more

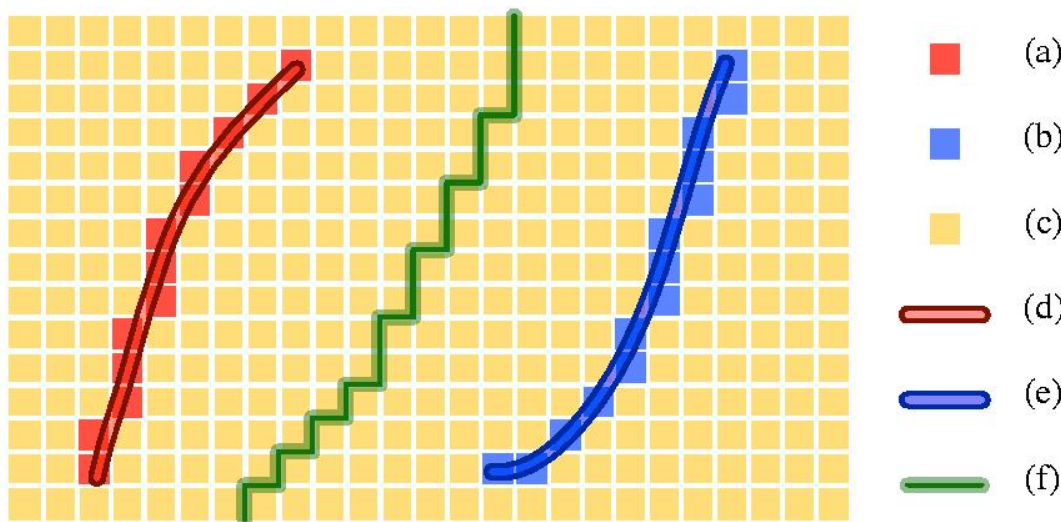


Method Overview

- Goal: Separate foreground from background
- First step: foreground/background selection
 - Only Few strokes are needed
 - Mincut on segment level
- Second step: Boundary correction
 - Mincut on pixel level

Object Marking Step

- Easily segment foreground and background
- Mark seeds in each area
- The GraphCut algorithm solves the labelling problem, on a pixel level





Graph Cut Image Segmentation

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

- Minimize Gibbs Energy
 - $E_1(x_i)$: Likelihood Energy
 - $E_2(x_i, x_j)$: Prior Energy
- X_i : Label of node i .
 - in $\{0: \textit{background}, 1: \textit{foreground}\}$

Likelihood Energy

$$E_1(x_i = 1) = 0 \quad E_1(x_i = 0) = \infty \quad \forall i \in \mathcal{F}$$

$$E_1(x_i = 1) = \infty \quad E_1(x_i = 0) = 0 \quad \forall i \in \mathcal{B}$$

$$E_1(x_i = 1) = \frac{d_i^{\mathcal{F}}}{d_i^{\mathcal{F}} + d_i^{\mathcal{B}}} \quad E_1(x_i = 0) = \frac{d_i^{\mathcal{B}}}{d_i^{\mathcal{F}} + d_i^{\mathcal{B}}} \quad \forall i \in \mathcal{U}$$

- Ensure that seed pixel are in the right region
- Use colour similarity to give an energy for uncertain pixel
 - $d_i^{\mathcal{F}}$: Distance from Front Colour
 - $d_i^{\mathcal{B}}$: Distance from Background Colour

Prior Energy

$$E_2(x_i, x_j) = |x_i - x_j| \cdot g(C_{ij})$$

$$C_{ij} = ||C(i) - C(j)||^2$$

$$g(\epsilon) = \frac{1}{\epsilon + 1}$$

- Penalty Term for boundaries
- Larger if adjacent pixel have similar colour
- Only nonzero if across segmentation boundary

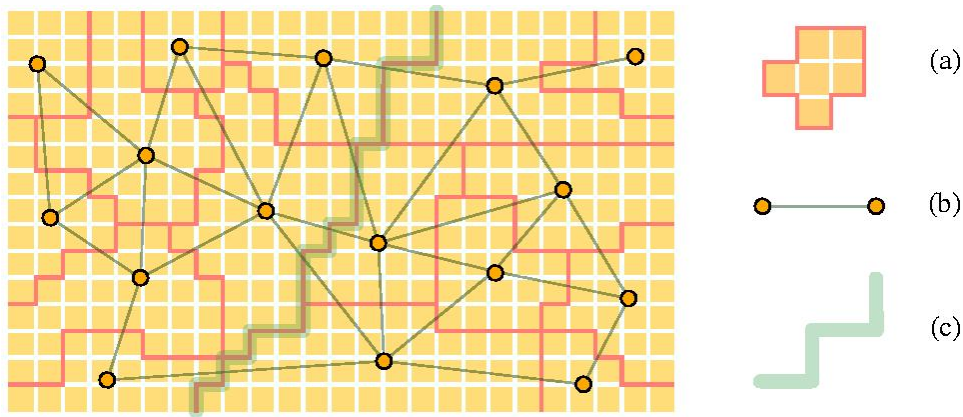


Mincut - Maxflow

- To minimize the Energy, a Maxflow algorithm is used [Boykov and Kolmogorov 2001]
- This optimized method still doesn't provide real-time responses
- The measurements below were calculated on a Centrino 1.5GHz with 512 MB RAM

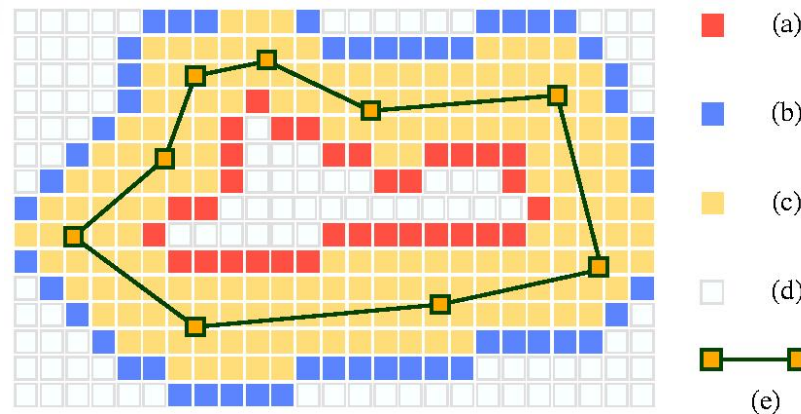
Image	Dimension	Nodes Ratio	Edges Ratio	Lag with Pre-segmentation	Lag without Pre-segmentation
Boy	(408, 600)	10.7	16.8	0.12s	0.57s
Ballet	(440, 800)	11.4	18.3	0.21s	1.39s
Twins	(1024, 768)	20.7	32.5	0.25s	1.82s
Girl	(768, 1147)	23.8	37.6	0.22s	2.49s
Grandpa	(1147, 768)	19.3	30.5	0.22s	3.56s

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
- Results are available almost instantly

Boundary Editing



- Object Marking is good, but there may still be errors
- Low contrast boundaries and other ambiguous areas are problematic
- The user interface should provide similar ease of use as step one



Boundary Editing

- Direct Vertex editing
 - Create new vertices
 - Move existing vertices
- Overriding brush
 - Replace part of the polygon
- After each editing step, the GraphCut algorithm calculates a new segmentation
- Only a small band of pixels around the polygon is uncertain

Boundary Editing

$$E_2(x_i, x_j) = |x_i - x_j| \cdot g \left((1 - \beta) \cdot C_{ij} + \beta \cdot \eta \cdot g(D_{ij}^2) \right)$$

- Likelihood energy is the same as in step one
- Prior energy gets augmented with polygon locations as soft constraints
- Hard constraints are possible





Joining the Parts

- See a movie of the technology in action, 4min, 31sec of image cutout



User Evaluation

- The method was tested on ten inexperienced people and four experts
- Compares Photoshop (Magnetic Lasso) and Lazy Snapping
 - Cut out a series of four pictures as exact as possible
 - Cut out a series of four pictures within 60 seconds each
- Results were taped and compared with the ground truth



User Evaluation Results

- Error ratio of Lazy Snapping was only 20% of the one from Photoshop
- Done in less than 60% of the time spent in Photoshop, but results showed a high standard deviation
- Less error pixels than with Photoshop, and the intermediate result of task two was usable

User Feedback

- Subject found Lazy Snapping to be „much easier“, „almost magic“
- Concerned that it encourages lazyness
- Dissatisfaction with clearly separated two step approach

Results



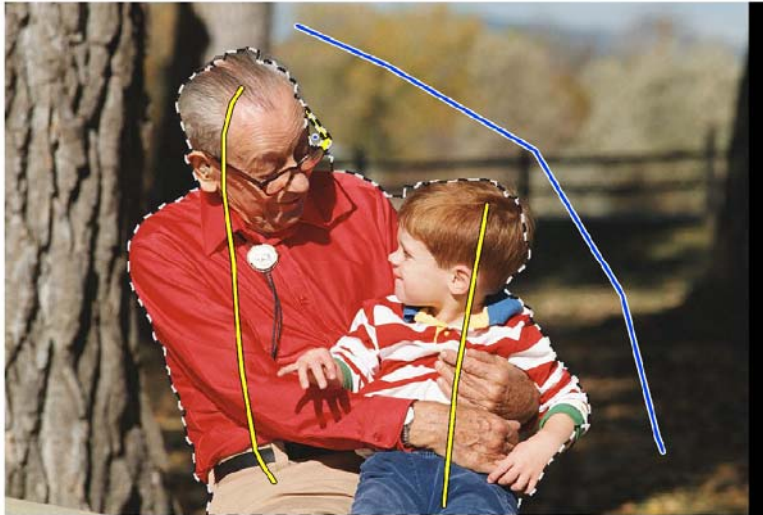
Results



Results



Result



Further Questions

- Is it possible to join both phases, or make it possible to alternate?
- The first phase uses a pre-segmented representation of the picture. How does this influence the cut?
- The likelihood energy uses colour proximity. How does this affect segmentation of greyscale pictures?



Discussion

- Questions?
- Discussion