Interactive Foreground Extraction using Iterated Graph Cuts

Carsten Rother - Vladimir Kolmogorov - Andrew Blake

Siggraph '04
Outline

Introduction & Motivation

Previous Approaches
• Magic Wand
• Intelligent Scissors
• Graph Cut

The Algorithm
• Novelties
• Behind the Scenes

Results

Conclusions
Motivation

Forefront Extraction: Say what?

We have

We want
Motivation
The GrabCut Approach

Introduction | Previous Approaches | The Algorithm | Results | Conclusions
Introduction
Overview

1. User marks object

"Grab"

Iterative Minimization

"Cut"

2. Intermediate Result

3. User adjusts selection

4. Final Result
The Goals...
- interactive foreground extraction
- high performance & quality - minimal user input
- usability in non-trivial images

...can only be achieved with
- good user interface (rectangle / lasso)
- accurate segmentation (iterative graph cuts)
- convincing alpha values (border matting)
Previous Approaches

Magic Wand

**Input:** Seed Pixel(s) + Tolerance

**Output:** Pixels within tolerance of seed pixels' colors statistics

**Thresholding**
Previous Approaches

Intelligent Scissors

**Input:** Points on segmentation boundary

**Output:** Pixels within minimum cost contour

Energy Minimization & Interpolation
Graph Cut

- foundation of GrabCut → pay attention 😊
- uses boundary and region information
- segmentation: min-cut by energy minimization

Input: Clue-mark inside and outside region

Output: Pixels within "best" inside region
Previous Approaches

Graph Cut

A little bit more detailed
1. Represent image as graph (weights $\sim 1/\text{energy}$)
2. Energy minimization
3. Find minimum cost path & cut it

$\text{Cost} = \sum \text{(cut weights)}$

Source (Foreground)

Sink (Background)
The Algorithm
Novelties

**GrabCut:** extension of Graph Cut
- iterative optimization
- incomplete labelling
- simplified user interaction
- border matting
### Behind the Scenes

#### Model

**Given**

<table>
<thead>
<tr>
<th>from start</th>
<th>N pixels</th>
</tr>
</thead>
<tbody>
<tr>
<td>color array</td>
<td>$\mathbf{z} = (z_1, \ldots, z_N)$</td>
</tr>
<tr>
<td>initial trimap</td>
<td>$T = {T_B, T_U, T_F}$</td>
</tr>
<tr>
<td>initial alpha matte</td>
<td>$\alpha = (\alpha_1, \ldots, \alpha_N), \quad \alpha \in {0,1}$</td>
</tr>
<tr>
<td>GMM components</td>
<td>$f_1, \ldots, f_K, \quad b_1, \ldots, b_K$</td>
</tr>
<tr>
<td>GMM array</td>
<td>$k = (k_1, \ldots, k_n, \ldots, k_N), \quad k_n \in {1, \ldots, K}$</td>
</tr>
</tbody>
</table>

**after init**
Behind the Scenes

Model

Wanted

<table>
<thead>
<tr>
<th>GMM parameters</th>
<th>( \Theta = {\pi(\alpha, k), \mu(\alpha, k), \Sigma(\alpha, k)}, ; \alpha = 0, 1; ; k = 1, \ldots, K )</th>
</tr>
</thead>
<tbody>
<tr>
<td>final alpha matte</td>
<td>( \underline{\alpha} = (\alpha_1, \ldots, \alpha_N), ; \alpha \in [0, 1] )</td>
</tr>
</tbody>
</table>

Energy function \( E = U + V \)

\( U \): fit of \( \underline{\alpha} \) to \( z \) given GMM \((\text{color/region information})\)

\( V \): smoothness term \((\text{boundary/edge information})\)
Behind the Scenes

Outline

- Initialization
  - Trimap
  - Alpha matte
  - GMMs

- Iterative Minimization
  1. Assign GMMs to pixels
  2. Adapt GMM parameters
  3. Estimate Segmentation
  4. Apply border matting

- User editing
  User: fg / bg brush
  GrabCut:
  - update trimap
  - step 3+4 once

until convergence
Behind the Scenes
Trimap & Alpha Matte

Initialization

- **Trimap**
- **Alpha matte**
- **GMMs**

**Trimap** $T = \{T_B, T_U, T_F\}$
(background, unknown and foreground region)

1. user indicates background region $T_B$
   (hard constraints!)
2. $T_F = \emptyset$, $T_U = \overline{T_B}$

**Alpha Matte** $\alpha$
(alpha value per pixel)

$$
\alpha_n = \begin{cases} 
0 & \text{if } n \in T_B \\
1 & \text{if } n \in T_U 
\end{cases}
$$
## Behind the Scenes

### Trimap & Alpha Matte

<table>
<thead>
<tr>
<th>Initialization</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Trimap</td>
</tr>
<tr>
<td>• Alpha matte</td>
</tr>
<tr>
<td>• GMMs</td>
</tr>
</tbody>
</table>

#### Trimap

\[ T = \{ T_B, T_U, T_F \} \]

(Background, unknown and foreground region)

1. User indicates background region \( T_B \) (hard constraints!)
2. \( T_F = \emptyset \), \( T_U = \overline{T_B} \)

#### Alpha Matte

\[ \alpha \]

(Alpha value per pixel)

\[ \alpha_n = \begin{cases} 
0 & \text{if } n \in T_B \\
1 & \text{if } n \in T_U 
\end{cases} \]
Behind the Scenes
Gaussian Mixture Models

Initialization
- Trimap
- Alpha matte
- GMMs

Gaussian Mixture Model (GMM)
- approximation of color distribution
- weighted sum of K gaussians
- parameters: weights $\pi$, means $\mu$, covariances $\Sigma$
- in color space

In GrabCut
- $K = 5$
- 1 foreground GMM, 1 background GMM

1. fit bg GMM to colors of pixels with $\alpha = 0$
2. fit fg GMM to colors of pixels with $\alpha = 1$
Behind the Scenes
Energy function revisited

**Iterative Minimization**
1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

**Energy function**

\[
E(\alpha, k, \Theta, z) = U(\alpha, k, \Theta, z) + V(\alpha, z)
\]

- **U**: fit of \( \alpha \) to \( z \) given GMM (region information)
- **V**: smoothness term (boundary/edge information)

\[
U = - \log(GMM)
\]

the better GMMs, the lower U

\[
V = \gamma \sum_{(m,n) \in C} [\alpha_n \neq \alpha_m] e^{-\frac{||z_m - z_n||^2}{2\sigma^2}}
\]

penalize low color difference

neighboring pixels at segmentation boundary with similar color get punished
Behind the Scenes
Computation

Iterative Minimization

1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

Iterative Minimization

while (!converged) {
  1. Fill $k$: for each pixel, find best GMM component
  2. Find best GMM parameters $\Theta(\alpha, k)$
  3. Perform Graph Cut
     1. Minimize energy
        $\min_{\{\alpha_n: n \in T_U\}} \min_k E(\alpha, k, \Theta, z)$
     2. Cut $\rightarrow$ adjust $\alpha$
}

Computation
Behind the Scenes

Demo

Iterative Minimization
1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

Result

Energy

1. 2. 3. 4.
Behind the Scenes

Demo

**Iterative Minimization**

1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

**Result**

**Energy**

1 2 3 4
Behind the Scenes

Demo

Iterative Minimization

1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

Result

Energy

Introduction | Previous Approaches | The Algorithm | Results | Conclusions
Behind the Scenes

Demo

Iterative Minimization
1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

Result

Energy

---

Introduction | Previous Approaches | The Algorithm | Results | Conclusions
Behind the Scenes
Demo

**Iterative Minimization**
1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

**Result**

**Energy**

1. 2. 3. 4.
Behind the Scenes

Border Matting

Iterative Minimization
1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. **Apply border matting**

Border Matting

hard segmentation

soft segmentation
Border Matting
1. obtain C from hard segmentation, define w (here w=4)
2. \( \forall \text{pixel} \in T_U \): assign t(n)
3. go along C, find best transition for every t
   (i.e. find \( \Delta \) and \( \sigma \) by energy minimization)

Avoid color bleeding
1. estimate foreground color
2. get closest "matching" color from foreground neighborhood
**Border Matting**

1. obtain C from hard segmentation, define w (here w=4)
2. \( \forall \text{pixel} \in T_U : \) assign \( t(n) \)
3. go along C, find best transition for every \( t \) (i.e. find \( \Delta \) and \( \sigma \) by energy minimization)

**Avoid color bleeding**

1. estimate foreground color
2. get closest "matching" color from foreground neighborhood

---

**Iterative Minimization**

1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting
User editing

User: fg / bg brush
GrabCut:  
• update trimap  
• step 3+4 once

User (fore- / background brush)  
adjust hard constraints

GrabCut  
• update trimap  
• estimate new segmentation (& apply border matting)
Results

Performance

**GrabCut**
- Target rectangle: 450 x 300 pixels
- 2.5 GHz CPU, 512 MB RAM
- Initial segmentation: 0.9 sec
- after each brush stroke: 0.12 sec

**GrabCut vs. Graph Cut**
- quality: perform comparably
- time: Graph Cut probably faster
- GrabCut fewer user interactions
Problems

- camouflage & low contrast
- thin structures
- inadequate bg representation
- several objects
Problems

• camouflage & low contrast
• thin structures
• inadequate bg representation
• several objects
Problems

• camouflage & low contrast
• thin structures
• inadequate bg representation
• several objects
Problems

• camouflage & low contrast
• thin structures
• inadequate bg representation
• several objects
Conclusions

My 2 Cents

The Algorithm
• fast
• good results in many cases
• nice ideas

The Paper
• claims proven (Gimp-plugin by Matthew Marsh, MS Expression)
• rather minimalist explanations
Future Work

- 3D support
- video support
- combination of GrabCut with image completion
Discussion

Questions? Ideas?