GrabCut

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

Foreground Extraction: Say what?

We have

We want

Motivation

The GrabCut Approach

User input

GrabCut

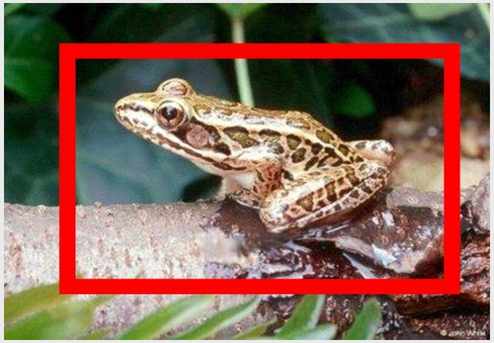
Output

Introduction

Overview

"Grab"

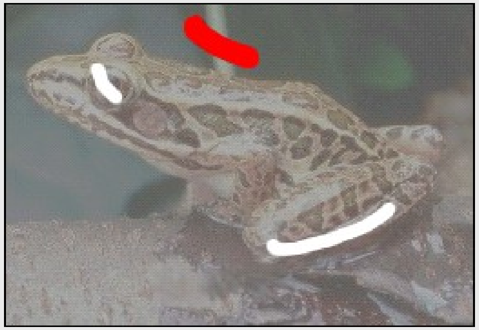
"Cut"



1. User marks object

Iterative Minimization

2. Intermediate Result



3. User adjusts selection

4. Final Result

Introduction

Aspirations

The Goals...

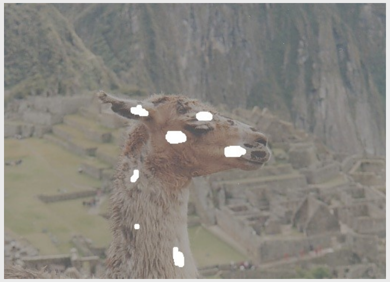
- interactive foreground extraction
- high performance & quality - minimal user input
- usability in non-trivial images

...can only be achieved with

- good user interface *(→ rectangle / lasso)*
- accurate segmentation *(→ iterative graph cuts)*
- convincing alpha values *(→ border matting)*

Previous Approaches

Magic Wand



Thresholding

Input: Seed Pixel(s) + Tolerance

Output: Pixels within tolerance of seed pixels' colors statistics

Previous Approaches

Intelligent Scissors

Energy Minimization
&
Interpolation

Input: Points on segmentation boundary

Output: Pixels within minimum cost contour

Previous Approaches

Graph Cut

Graph Cut

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

Min-Cut

Input: Clue-mark inside and outside region

Output: Pixels within "best" inside region

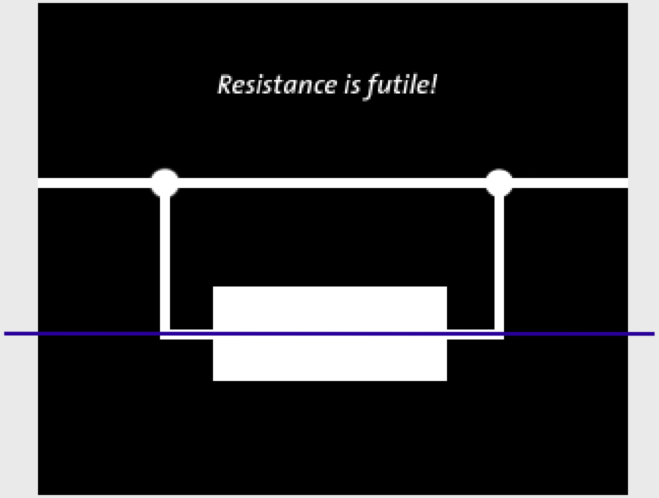
Previous Approaches

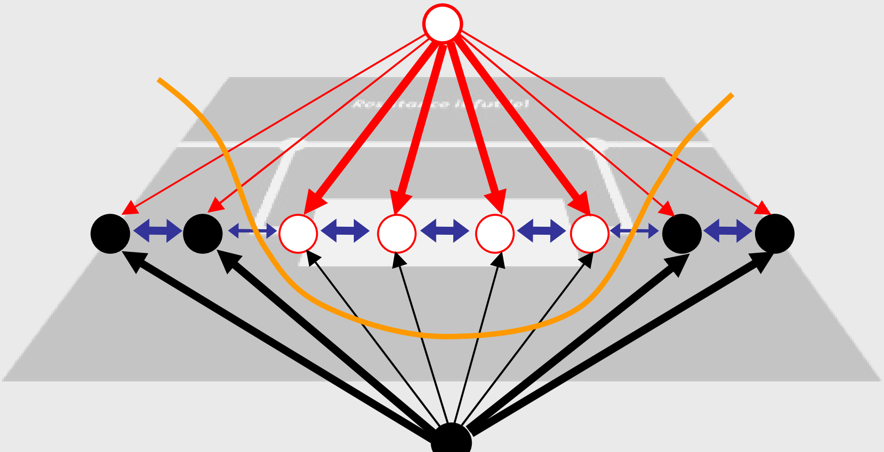
Graph Cut

A little bit more detailed

1. Represent image as graph (weights ~ 1/energy)
2. Energy minimization
3. Find minimum cost path & cut it

$$Cost = \sum(cut\ weights)$$





The Algorithm

Novelties

GrabCut: extension of Graph Cut

- iterative optimization
- incomplete labelling
- simplified user interaction
- border matting

Behind the Scenes

Model

Given

from start	N pixels	
	color array	$\underline{z} = (\underline{z}_1, \ldots, \underline{z}_N)$
after init	initial trimap	$T = \{T_B, T_U, T_F\}$
	initial alpha matte	$\underline{\alpha} = (\alpha_1, \ldots, \alpha_N), \quad \alpha \in \{0,1\}$
	GMM components	$f_1, \ldots, f_K \qquad b_1, \ldots, b_K$
	GMM array	$\underline{k} = (k_1, \ldots, k_n, \ldots, k_N), \quad k_\mathrm{n} \in \{1, \ldots, \mathrm{K}\}$

Behind the Scenes

Model

Wanted

GMM parameters	$\Theta = \{\pi(\alpha,k), \mu(\alpha,k), \Sigma(\alpha,k)\}, \quad \alpha = 0,1;\ k = 1,\ldots,K$
final alpha matte	$\underline{\alpha} = (\alpha_1,\ldots,\alpha_N), \quad \alpha \in [0,1]$

Energy function $E = U + V$

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

V: smoothness term *(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

until convergence

User editing

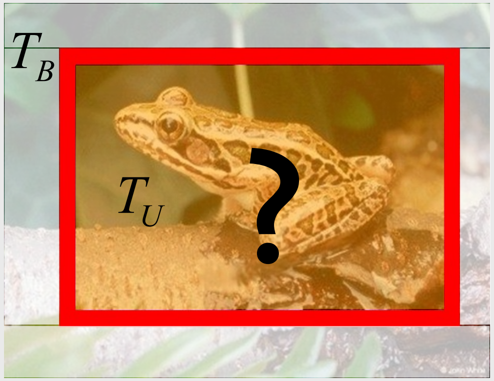
User: fg / bg brush
GrabCut :
- update trimap
- step 3+4 once

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 $\underline{\alpha}$
(alpha value per pixel)

$$\alpha_n = \begin{cases} 0 & \text{if } n \in T_B \\ 1 & \phantom{\text{if }} n \in T_U \end{cases}$$

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. $T_F = \emptyset,\ T_U = T_B$

Alpha Matte $\underline{\alpha}$
(alpha value per pixel)

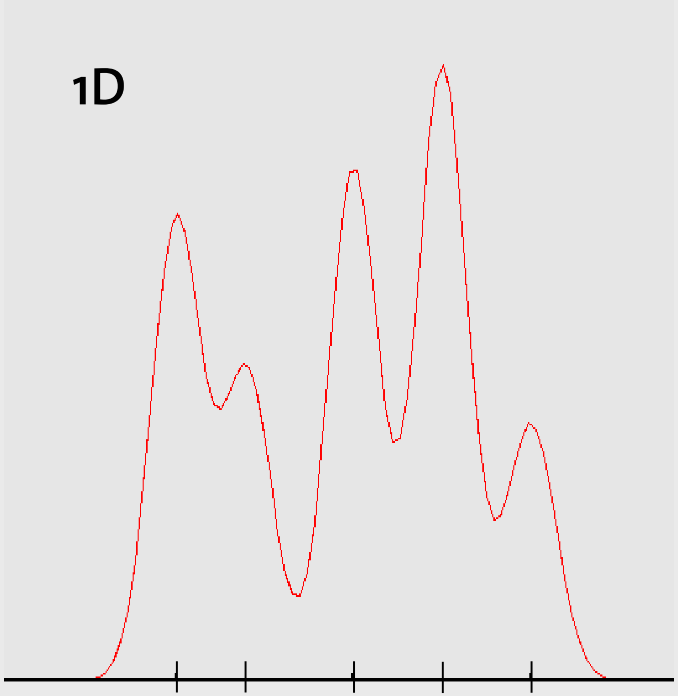
$$\alpha_n = \begin{cases} 0 & \text{if} \quad n \in T_B \\ 1 & \quad\quad n \in T_U \end{cases}$$

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 π, means μ, covariances Σ
- 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

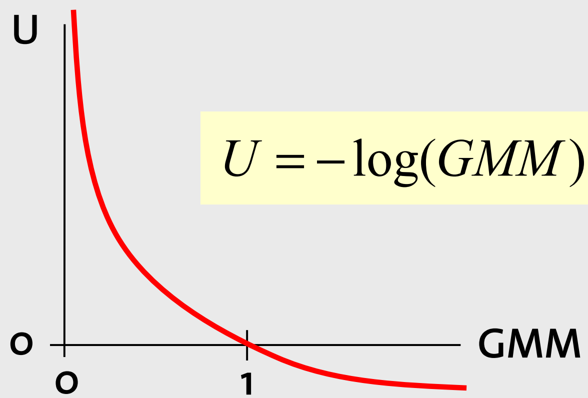
1. Assign GMMs to pixels
2. Adapt GMM parameters
3. Estimate Segmentation
4. Apply border matting

Energy function

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

U: fit of $\underline{\alpha}$ to $\underline{z}$ given GMM *(region information)*

V: smoothness term *(boundary/edge information)*



$$U = -\log(GMM)$$

the better GMMs, the lower U

$$V = \gamma \sum_{(m,n)\in C} [\alpha_n \neq \alpha_m] \underbrace{e^{\left(-\frac{\|z_m - z_n\|^2}{2\sigma^2}\right)}}_{\text{penalize low color difference}}$$

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

}

Behind the Scenes

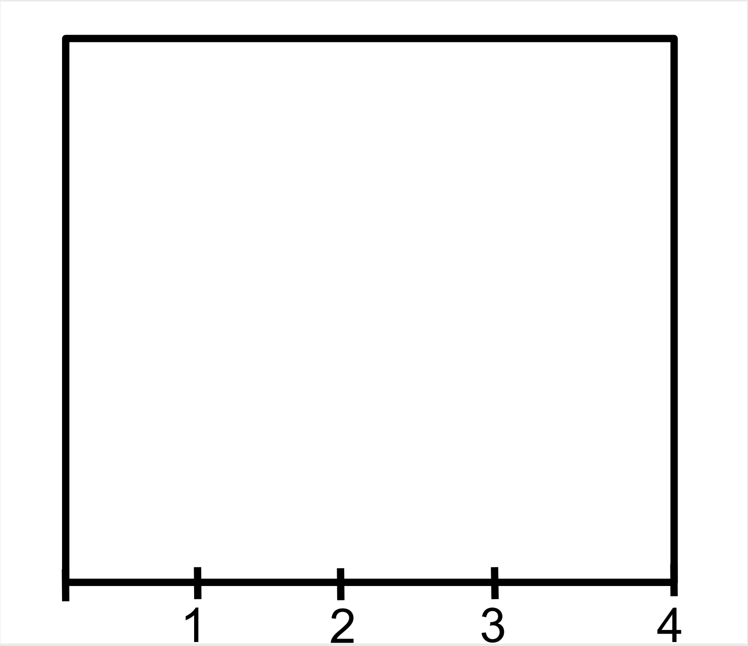
Demo

Iterative Minimization

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

Result

Energy



Behind the Scenes

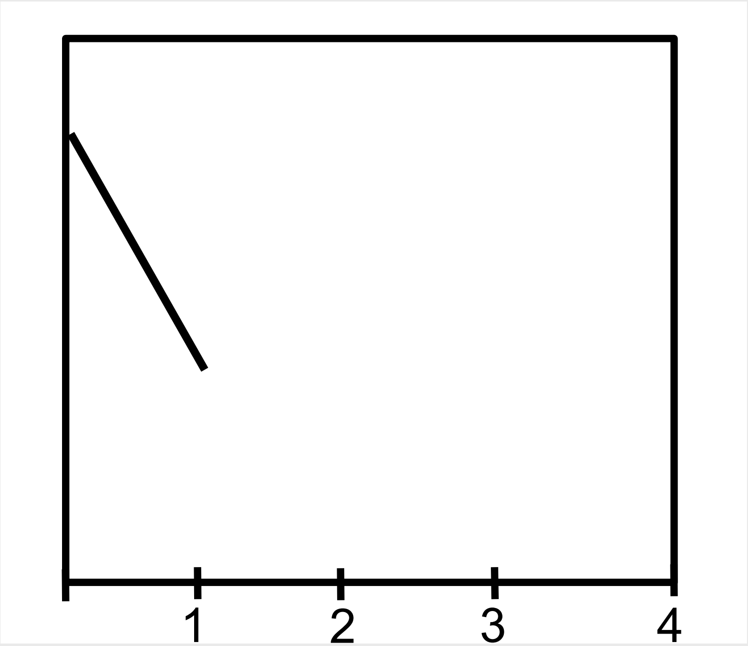
Demo

Iterative Minimization

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

Result

Energy



Behind the Scenes

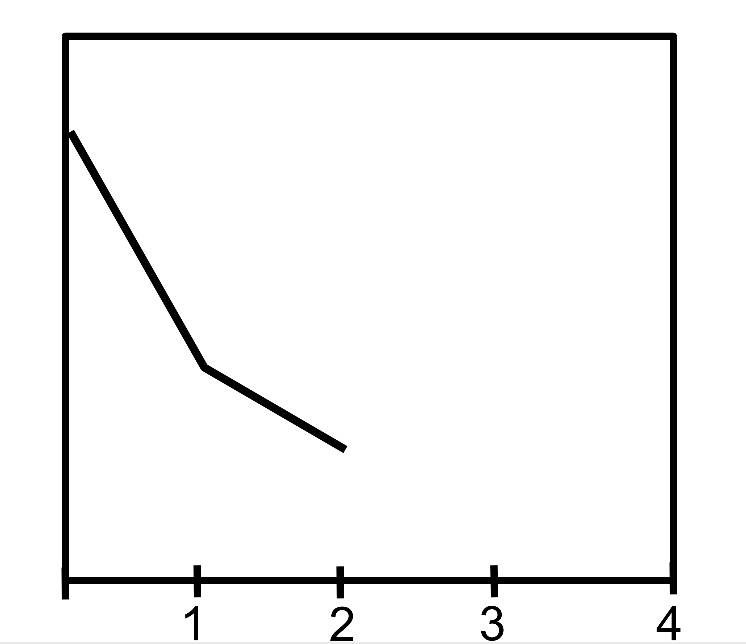
Demo

Iterative Minimization

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

Result

Energy



Behind the Scenes

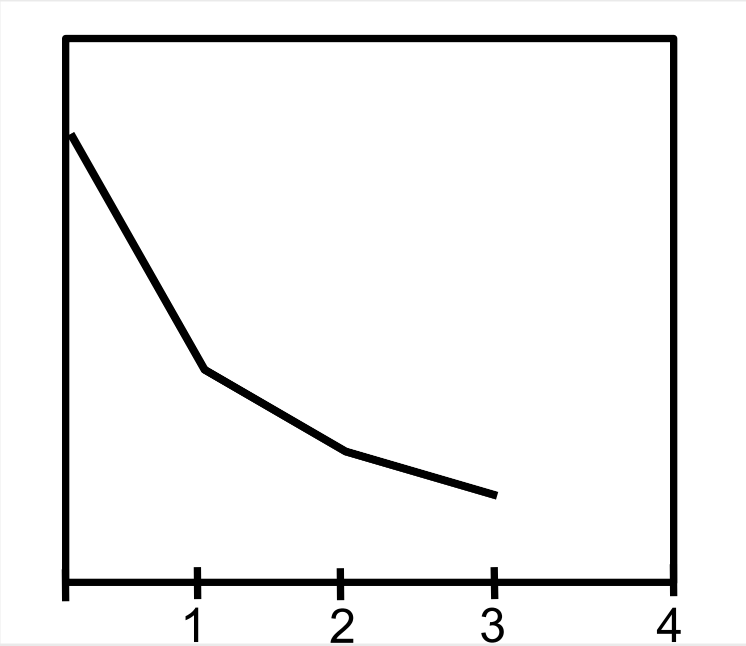
Demo

Iterative Minimization

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

Result

Energy



Behind the Scenes

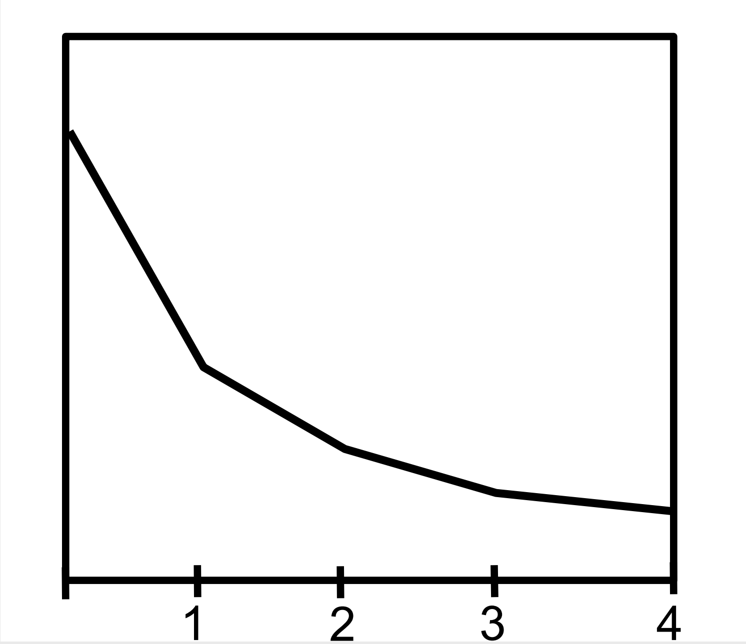
Demo

Iterative Minimization

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

Result

Energy

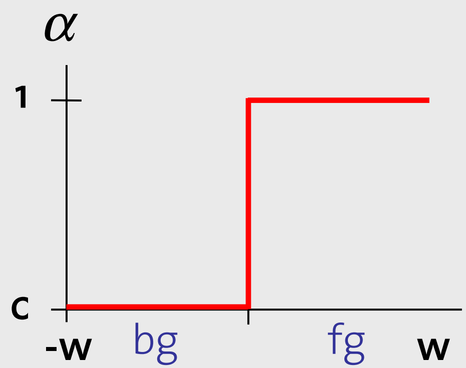


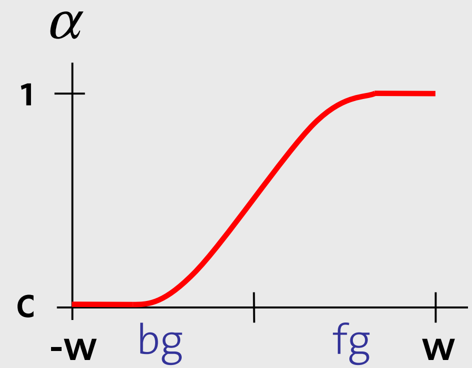
Behind the Scenes

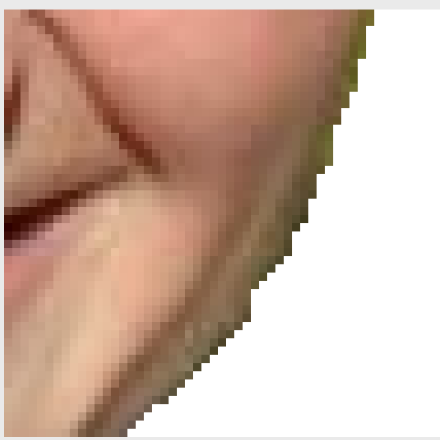
Border Matting

Iterative Minimization

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

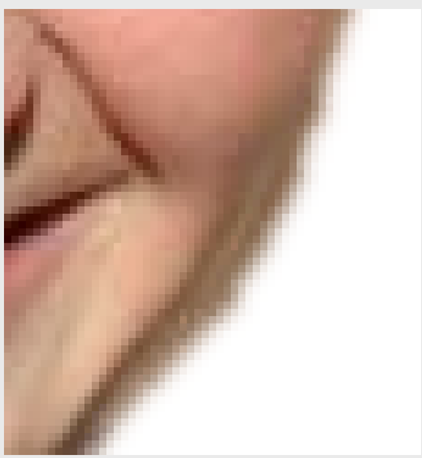






hard segmentation

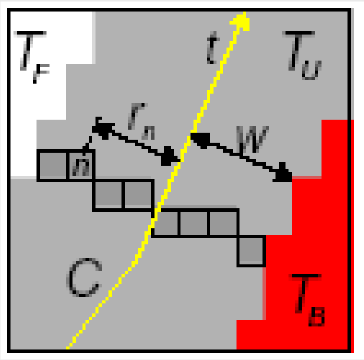
Border Matting

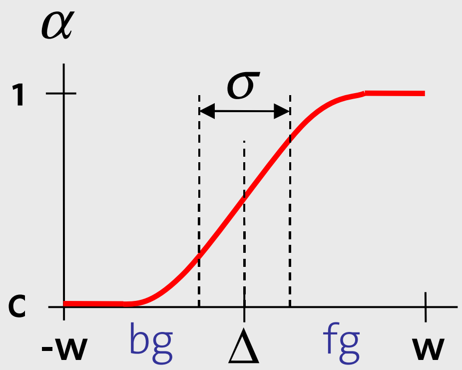


soft segmentation

Iterative Minimization

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





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 Δ and $\boldsymbol{\sigma}$ by energy minimization)

Avoid color bleeding

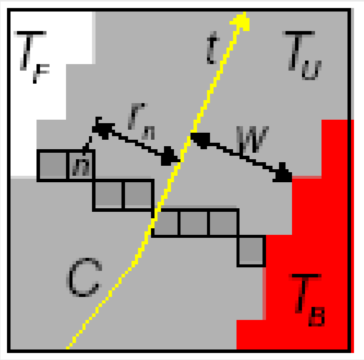
1. estimate foreground color
2. get closest "matching" color from foreground neighborhood

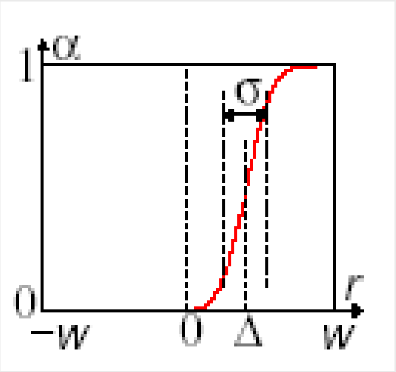
Behind the Scenes

Border Matting

Iterative Minimization

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





Border Matting

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 Δ and σ by energy minimization)

Avoid color bleeding

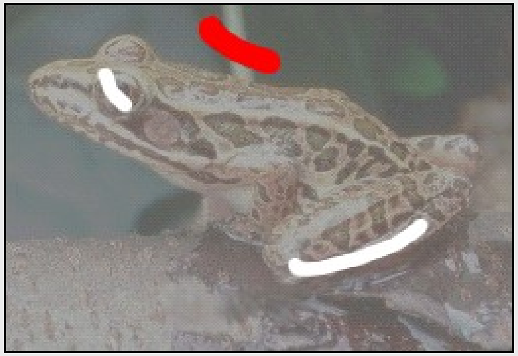
1. estimate foreground color
2. get closest "matching" color from foreground neighborhood

Behind the Scenes

User Editing

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

Results

Problems

Problems

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

Results

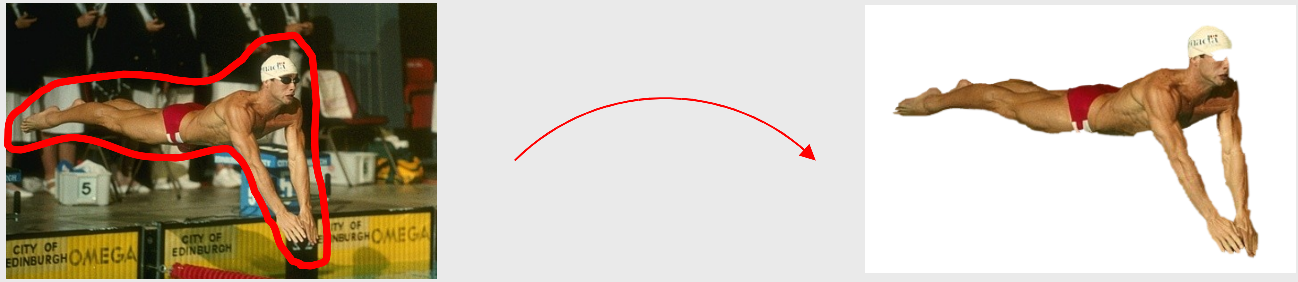
Problems

Problems

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

Problems

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

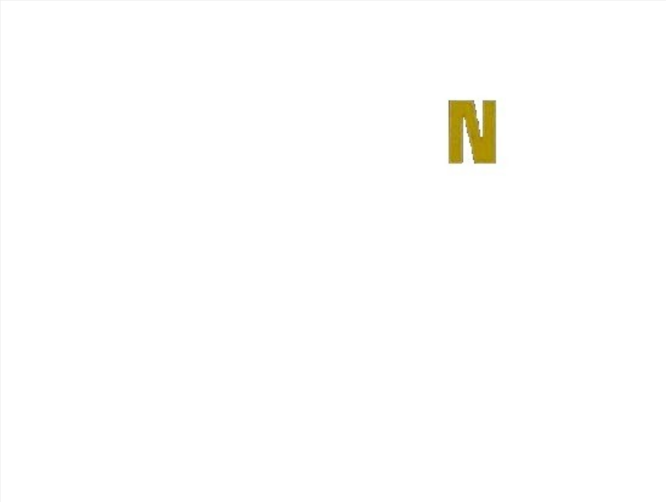


Results

Problems

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

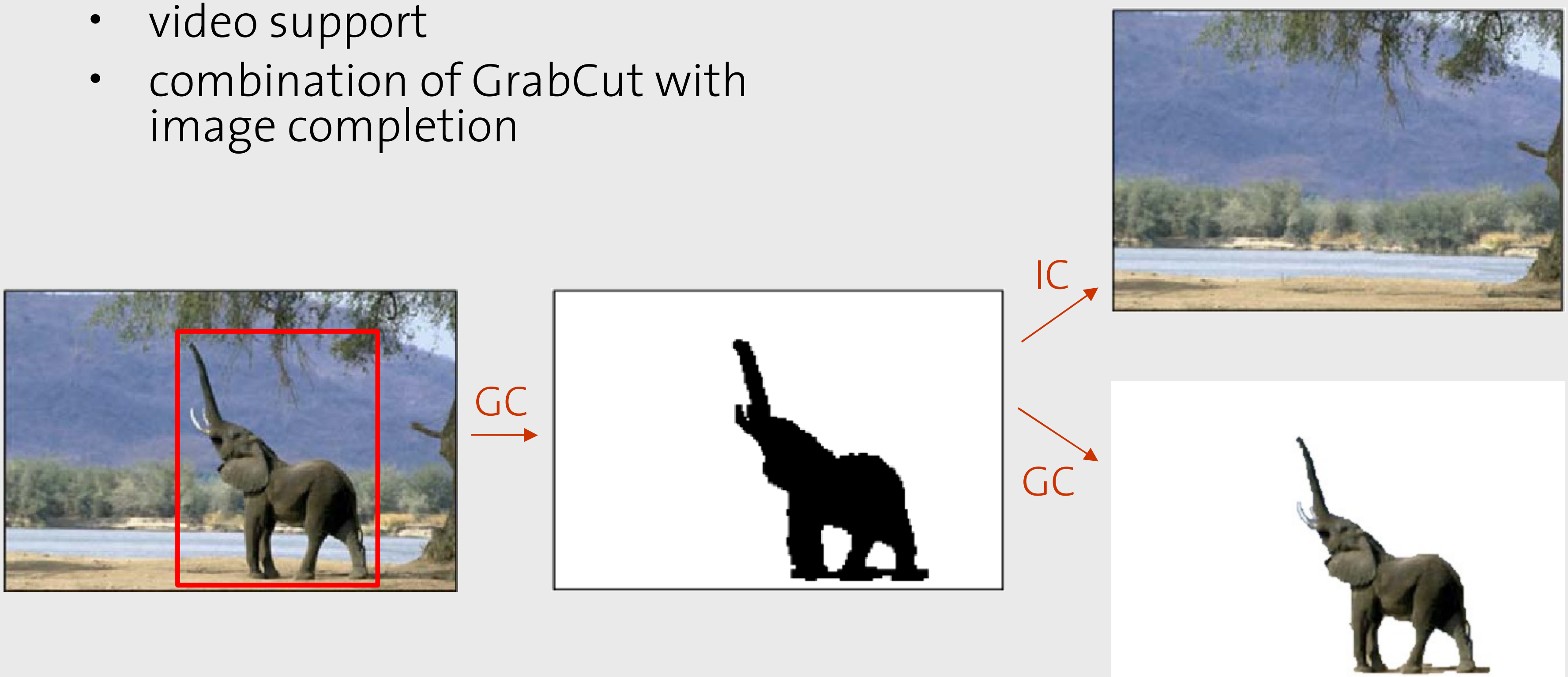
- claims proven (Gimp-plugin by Matthew Marsh, MS Expression)
- rather minimalist explanations

Conclusions

Future Work

Future Work

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



Discussion

Questions? Ideas?