# “GrabCut” — Interactive Foreground Extraction using Iterated Graph Cuts

*By Carsten Rother, Vladimir Kolmogorov and Andrew Blake at Microsoft Research Cambridge, UK*

*SIGGRAPH 2004*

Slides by David Anthony Torres

Computer Science and Engineering — University of California at San Diego

# Object Extraction

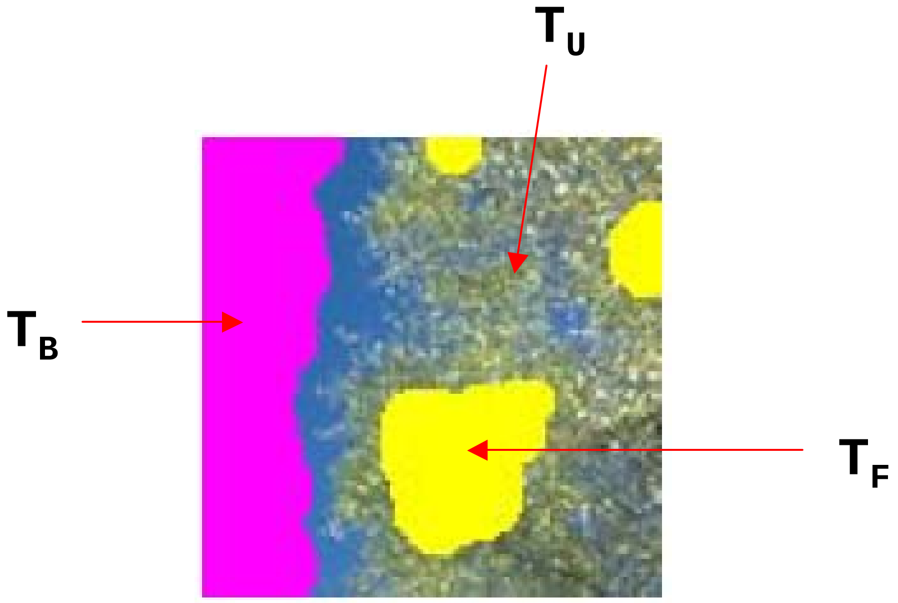
- Deal with efficient, interactive extraction of a foreground object from a complex background.
- Goal is to produce a "good" automatic extraction with as little user interaction as possible.
- Performance is measured by
  - Accurate segmentation of the object.
  - Subjectively convincing extraction when faced with blur, transparency.
  - Free of color bleeding in from the background.

## Object Extraction

- Useful because, historically, extraction of objects from images requires a lot of user interaction.
- Interactive segmentation tools have been developed:
  - ➢ Magic Wand (in Adobe Photoshop)
  - ➢ Intelligent Scissors (a.k.a. Live Wire, Magnetic Lasso)
  - ➢ Bayes Matting and Knockout
  - ➢ Graph Cut
- This paper extends the Graph Cut algorithm so lets look at that first:

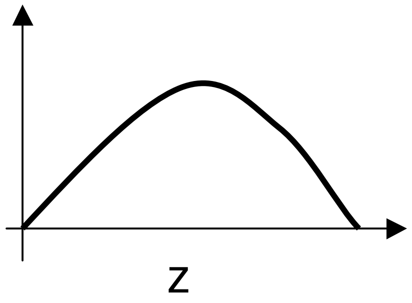
# GraphCut for a Monochrome Image

- User provides a trimap $T = \{ T_F, T_B, T_U \}$ which partitions the image into 3 regions: foreground, background, unknown.



- The image is an array $\mathbf{z} = (z_1, \ldots z_N)$ of grey values indexed by the single index $n$.
- The segmentation of the image is an alpha-channel, or, a series of opacity values $\boldsymbol{\alpha}=(\alpha_1,\ldots, \alpha_N)$ at each pixel with $0 \leq \alpha_n \leq 1$.
- The parameter $\boldsymbol{\theta}$ describes the foreground/background grey-level distributions. i.e. a pair of histogram of gray values:

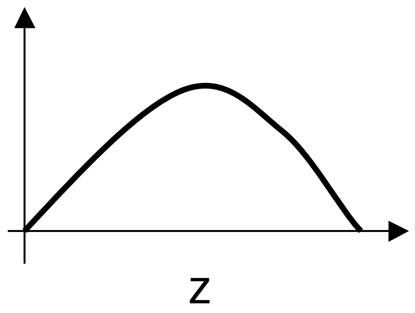
$$\theta = \{h(z;\alpha), \alpha = 0,1\} \Longleftrightarrow$$



- Note that these histograms are directly assembled from the trimaps $T_B$ and $T_F$

- Re-pose the segmentation task:
  - ➢The segmentation task is to infer the unknown opacity values **α** from image **z** and the model **θ**.

$$\theta = \{h(z;\alpha), \alpha = 0,1\} \quad \Longleftrightarrow$$



## Segmentation by Energy Minimization

- An energy function **E** is defined so that its minimum corresponds to a good segmentation.
- This is captured by a “Gibbs” energy of the form:

$$\mathrm{E}(\boldsymbol{\alpha},\boldsymbol{\theta},\mathbf{z}) = \mathbf{U}(\boldsymbol{\alpha},\boldsymbol{\theta},\mathbf{z}) + \mathbf{V}(\boldsymbol{\alpha},\mathbf{z})$$

## E(**α**,**θ**,**z**) = **U**(**α**,**θ**,**z**) + **V**(**α**,**z**)

- U evaluates the fit of the opacity α to the data **z**
  - ➢ i.e. it gives a good score (low score) if α looks like it's consistent with the histogram.

$$U(\alpha,\theta,z)=\sum_n -\log h(z_n;\alpha_n)$$

- V is a smoothness term which penalizes if there is too much disparity between neighboring pixel values.

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

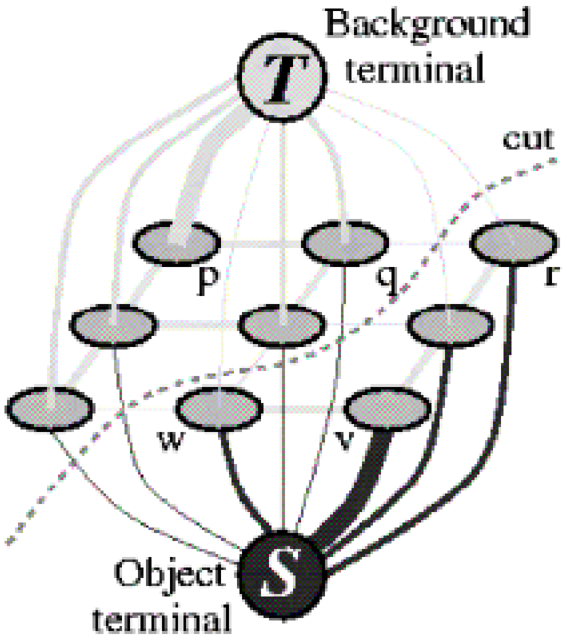
## $\mathrm{E}(\boldsymbol{\alpha},\boldsymbol{\theta},\mathbf{z}) = \mathbf{U}(\boldsymbol{\alpha},\boldsymbol{\theta},\mathbf{z}) + \mathbf{V}(\boldsymbol{\alpha},\mathbf{z})$

- Given the energy model we can obtain a segmentation by finding

$$\alpha = \arg\min_{\alpha} E(\alpha,\theta)$$

- Which can be solved using a minimum cut algorithm which gives you a hard segmentation, $\boldsymbol{\alpha} = \{0,1\}$, of the object.

# Min-Cut



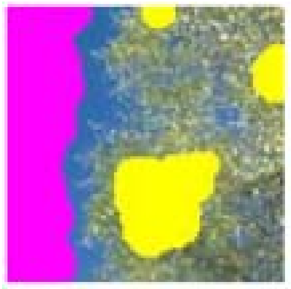
•Edge weights are labeled with U( ) and V ( )

## How GrabCut adds to Graph Cut

- The monochrome image model is replaced for color by a Gaussian Mixture Model (GMM) in place of histograms.
- One shot min-cut solution is replaced by an iterative procedure that alternates between estimation and parameter learning
- Allow for incomplete labeling, i.e. the user need only specify the background trimap $T_B$ (and implicitly the unknown map $T_U$)

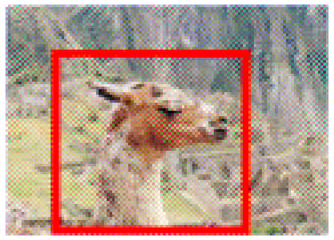
- This amounts to one less user interaction step that was required in previous versions.

From this …



[Specifying foreground and background]

To this …



[Specifying background only]

## Adding the Color Model

- Each pixel $z_n$ is now in RGB color space
- Color space histograms are impractical so we use a Gaussian Mixture Model (GMM)
  - 2 Full-covariance Gaussian mixtures with K components (K ~ 5).
  - One for foreground, one for background.
- Add to our model a vector $\boldsymbol{k} = \{ k_1 \ldots k_N \}$, with $k_i$ in $\{1 \ldots K\}$
- $k_i$ assigns the pixel $z_i$ to a unique GMM component (Either from F.G. or B.G. as α dictates)

# New Energy Model

- Must incorporate ***k*** into our model:

$$\mathbf{E}(\boldsymbol{\alpha},\mathbf{k},\boldsymbol{\theta},\mathbf{z}) = \mathbf{U}(\boldsymbol{\alpha},\mathbf{k},\boldsymbol{\theta},\mathbf{z}) + \mathbf{V}(\boldsymbol{\alpha},\mathbf{z})$$

where

$$\mathbf{U}(\boldsymbol{\alpha},\mathbf{k},\boldsymbol{\theta},\mathbf{z}) = \sum_n D(\alpha_n,k_n,\theta,z_n)$$

- $D(\alpha_n,k_n,\theta_n,z_n) = -\log p(z_n \mid \alpha_n,k_n,\theta) - \log \pi(\alpha_n,k_n)$
- Where $\pi(\cdot)$ is a set of mixture weights which satisfy the constraint:

$$D(\alpha_n,k_n,\underline{\theta},z_n) = -\log \pi(\alpha_n,k_n) + \frac{1}{2}\log\det\Sigma(\alpha_n,k_n) + \frac{1}{2}[z_n - \mu(\alpha_n,k_n)]^\top \Sigma(\alpha_n,k_n)^{-1}[z_n - \mu(\alpha_n,k_n)].$$

# New Energy Model

- Our θ becomes

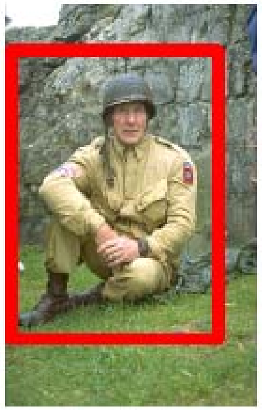
$$\theta=\{\underset{\underset{\text{weights}}{\uparrow}}{\pi(\alpha,k)},\ \underset{\underset{\text{means}}{\uparrow}}{\mu(\alpha,k)},\ \underset{\underset{\text{cov.}}{\uparrow}}{\Sigma(\alpha,k)},\ \underset{\underset{\text{fg/bg.}}{\uparrow}}{\alpha=0,1},\ \underset{\underset{\text{mixture component}}{\uparrow}}{k=1\ldots K}\}$$

- Total of 2K Gaussian components

## And now … the Algorithm:

**Initialisation**

- User initialises trimap $T$ by supplying only $T_B$. The foreground is set to $T_F = \emptyset$; $T_U = \overline{T}_B$, complement of the background.
- Initialise $\alpha_n = 0$ for $n \in T_B$ and $\alpha_n = 1$ for $n \in T_U$.
- Background and foreground GMMs initialised from sets $\alpha_n = 0$ and $\alpha_n = 1$ respectively.



# Iterative Minimization

1. *Assign GMM components to pixels:* for each $n$ in $T_U$,

$$k_n := \arg\min_{k_n} D_n(\alpha_n, k_n, \theta, z_n).$$

- find $k_n$ by iterating through all values 1…K. (K is small)

# Iterative Minimization

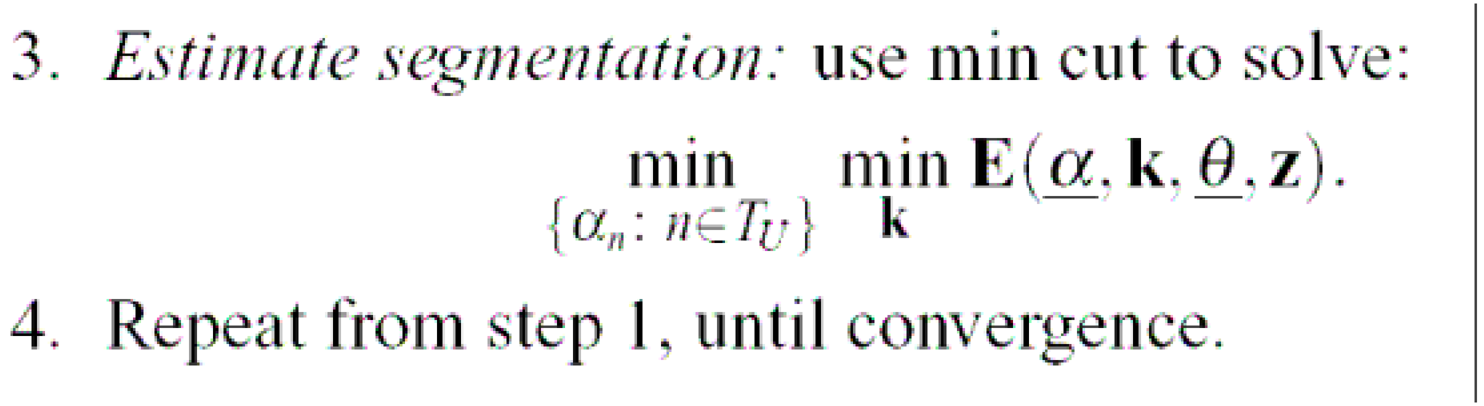
2. *Learn GMM parameters* from data **z**:

$$\underline{\theta} := \arg\min_{\underline{\theta}} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$$

• For example, to find the Gaussian parameters for a component *k* in the foreground:

- ➢Find the set of pixels $Z_k$ assigned to *k* by the ***k***-vector.
- ➢Find μ,Σ, in the standard fashion.
- ➢Update the weights at $\pi(\alpha,k) := |Z_k| / \sum_k |Z_k|$

# Iterative Minimization

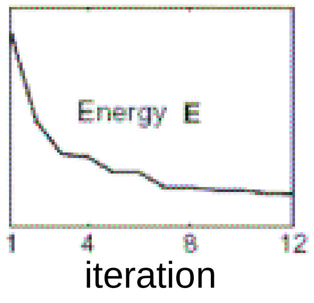


3. *Estimate segmentation:* use min cut to solve:

$$\min_{\{\alpha_n :\, n \in T_U\}} \min_{\mathbf{k}} \mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}).$$

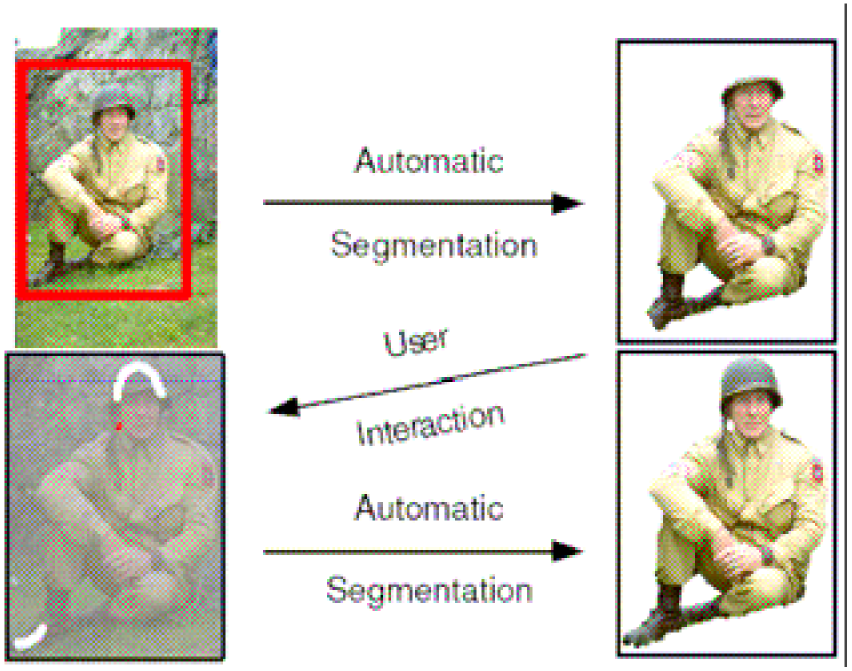
4. Repeat from step 1, until convergence.

- Now that *k* and *θ* are known, we can solve for the opacity values using a minimum-cut algorithm, and reapply the min-cut until convergence.
- Each iteration eats away at the unknown region and continues to minimize the energy function.

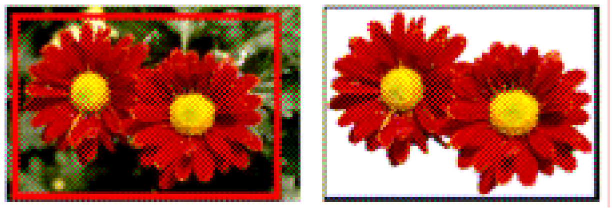


# User Editing

- If the segmentation is imprecise, the user can constrain pixels to the foreground/background
- Then we run min-cut (or the entire procedure) again to produce a final segmentation.

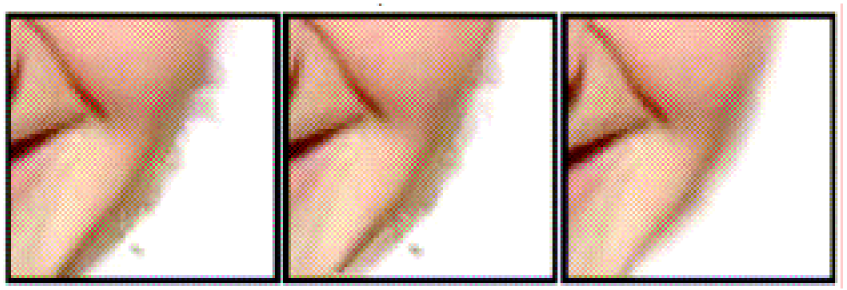


# Summary of Algorithm



- Start with $T_B$ and $T_U$ as inputs.
- We use the input to initialize the foreground and background GMMs
- Using the GMM's we solve an energy minimization problem via min-cut, which corresponds to an initial segmentation of the object, $T'_B$ and $T'_U$.
- Repeat the procedure with $T'_B$ and $T'_U$ as inputs until convergence ($T_F = T'_U$).
- User interaction then repeat either min-cut or all steps.

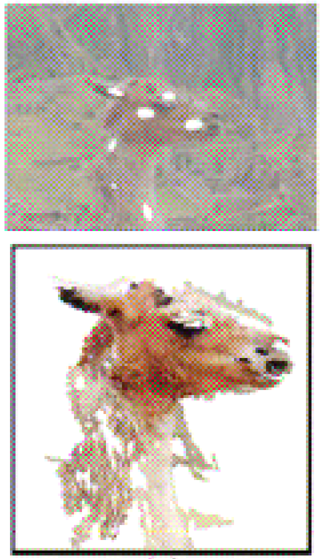
## Border Matting



- After the selection is made, the authors also implement a border matting tool.
- For time purposes I wont go into details.
- Basic idea is to use another energy-minimization solution to estimate the appropriate value of pixels along the boundaries of the object.

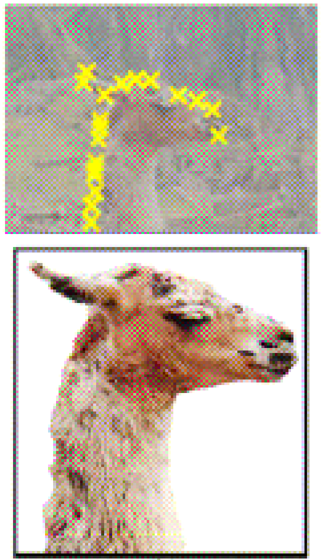
## Results

Magic Wand



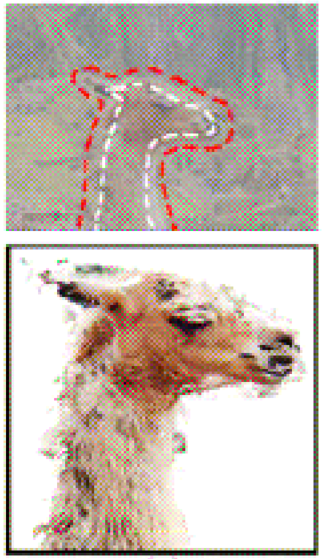
(a)

Intelligent Scissors



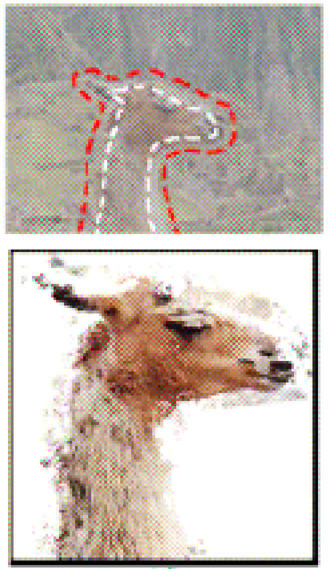
(b)

Bayes Matte



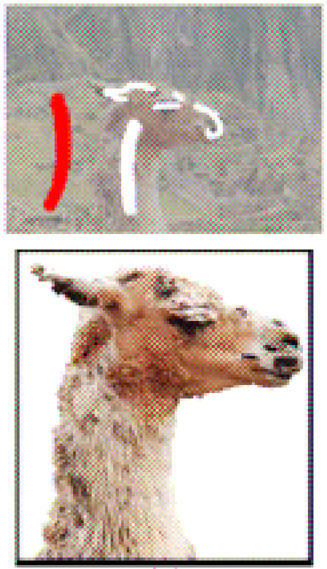
(c)

Knockout 2



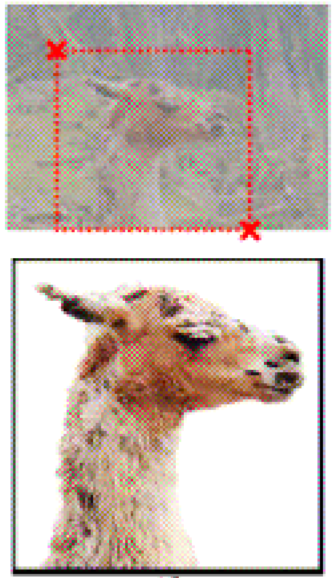
(d)

Graph cut



(e)

GrabCut



(f)

No User Interaction