# GrabCut

## Interactive Image Foreground Extraction Using Graph Cuts

**Team 39**
Aditya Harikrish
Anirudh Kaushik
Pratyay Suvarnapathaki
Yash Mehan

# GrabCut takes advantage of the graph-like structure of an image to interactively extract a foreground object in a complex environment.

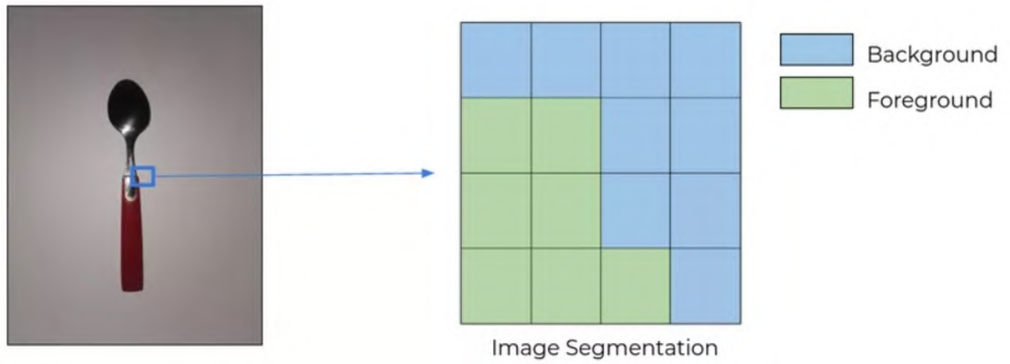
---

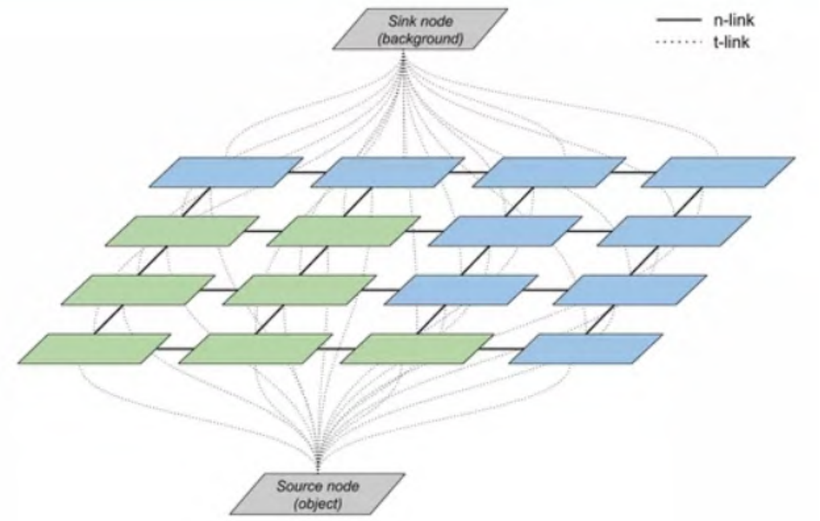
High-level details of the Reference Paper

- Prior to this, tools used texture (color) or edge information (contrast) to 'subtract' backgrounds.
- GrabCut uses an iterative graph cut algorithm that improved the graph cut segmentation SOTA at the time.
- Novel improvements made in the aspects of user-interactivity and border-matting.

# The GrabCut Algorithm

# Graph Formulation




- Segmenting an image means attributing a label to each of its pixel. In the case of GrabCut and other foreground extraction algorithms, each pixel is either labeled as being in the foreground or the background of the image.

- To obtain the segmentation, GrabCut takes advantage of the graph-like structure of an image. Energy functions can be defined to evaluate how good a certain graph cut is in terms of foreground-background separation.

- Lastly, GrabCut is an iterative algorithm. The initial output segmentation is used to re-estimate graph weights and then refine the image segmentation again, resulting in the GrabCut iterative loop.

# Energy Minimization

- Energy functions evaluate the quality of a given segmentation. By design, we chose a function such that its local minima corresponds to a good segmentation. The algorithm then tries to find a cut that minimises this function.

- For a greyscale image, 'local' and 'global' segmentation quality can be modelled as such:

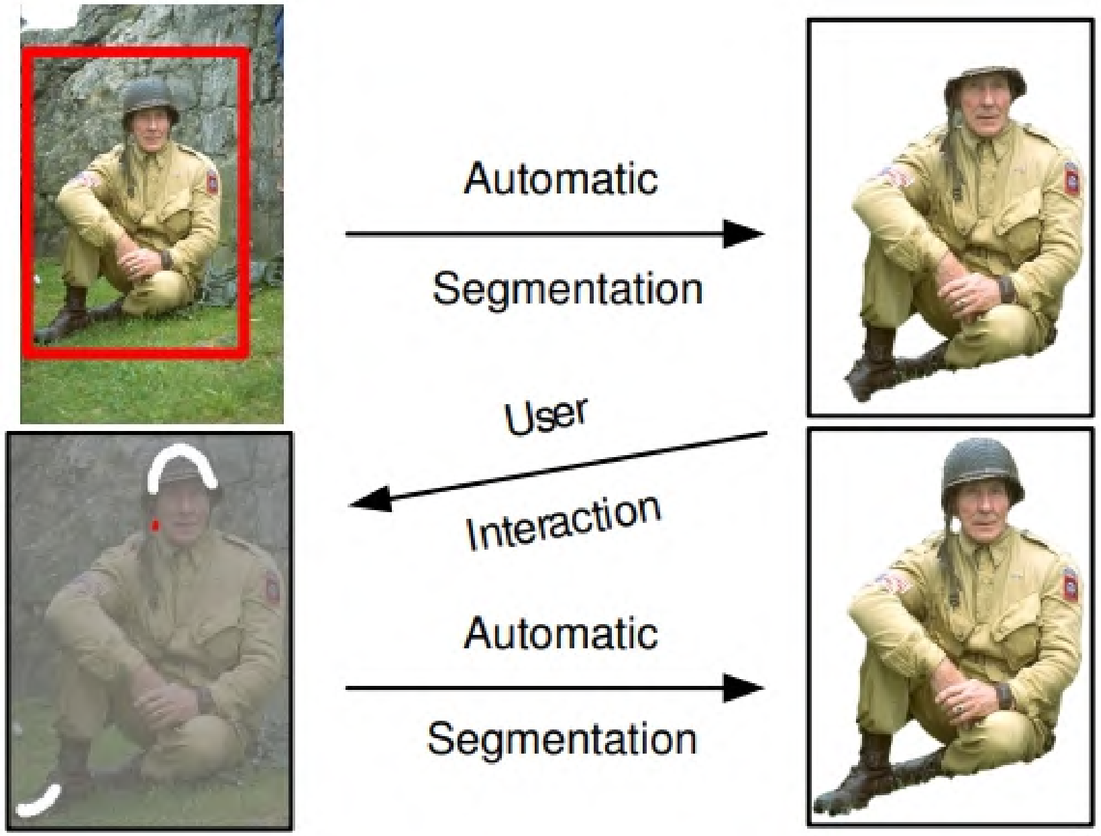
$$V(\underline{\alpha}, \mathbf{z}) = \gamma \sum_{(m,n)\in \mathbf{C}} [\alpha_n \neq \alpha_m] \exp -\beta \|z_m - z_n\|^2 \qquad U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = \sum_n D(\alpha_n, k_n, \underline{\theta}, z_n)$$

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

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

- Our reference paper's novelty lies in using GMMs to perform this task on coloured images. The crux is estimating the 'theta' and 'alpha', iteratively.

# The Iterative 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 minimisation**

1. *Assign GMM components to pixels:* for each $n$ in $T_U$,
$$k_n := \arg\min_{k_n} D_n(\alpha_n, k_n, \theta, z_n).$$
2. *Learn GMM parameters* from data $\mathbf{z}$:
$$\underline{\theta} := \arg\min_{\underline{\theta}} U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z})$$
3. *Estimate segmentation:* use min cut to solve:
$$\min_{\{\alpha_n:\ n \in T_U\}} \min_{\mathbf{k}} \mathrm{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}).$$
4. Repeat from step 1, until convergence.
5. Apply border matting (section 4).

**User editing**

- *Edit:* fix some pixels either to $\alpha_n = 0$ (background brush) or $\alpha_n = 1$ (foreground brush); update trimap $T$ accordingly. Perform step 3 above, just once.
- *Refine operation:* [optional] perform entire iterative minimisation algorithm.

# Flowchart

**Step 1**
User input passed by drawing the bounding box.

**Step 2**
Formulating the energy function.

**Step 3**
Constructing the Graph (modelling the minimization of the energy function as a flow problem).

**Step 4**
Run maxflow to estimate the mincut.

**Step 5**
Segmentation obtained from mincut gives us the foreground and background.
Theta and alpha are estimated iteratively.

# Our Progress

| Week 1 | Week 2 | Week 3 |
| --- | --- | --- |
| Developing an understanding of the paper. | Implementing the basic functionality, that is, modelling the energy function and formulation of graph construction.<br><br>Basic interactivity using OpenCV windowing too. | Fleshing out the user interaction component and GMMs, running tests and preparing deliverables. |

# Thank You