GrabCut Implementation

- Anmol Agarwal

Directory structure:

IPYNB NOTEBOOKS

- `Assignment3.ipynb`: Has the content corresponding to deliverable 1 (the good and bad examples and ALSO the examples which included user refinement heatmaps)
- `Experiments_<parameter name>.ipynb`: have the images/results used in the experiments which have been used to derive the inferences present in the REPORT content below

JSON and other data

- `ans_data.json`: contains the scores for all images across 10 iterations (with default parameters) and was used to make the table in deliverable 1
- `DIRECTORY ans_images`: contains directories for each image and stores snapshots of the GrabCut pipeline on each image across iterations

Basic stages and my understanding of the Grabcut algorithm

`(I make references to the variables/notation used in this section in the comments in my code)`

- All the pixels in the image are initialised as a part of one of the categories in the Trimap T = { T_F , T_B , T_U } where:
  - T_F = pixels which the user has indicated as being surely Foreground and whose foreground/background status cannot change in the future (unless the user intervenes using the REFINE tool)
  - T_B = pixels which the user has indicated as being surely Background and whose foreground/background status cannot change in the future (unless the user intervenes using the REFINE tool)
  - T_U = pixels whose foreground/background status is still not surely decided and whose status can change based on the likelihoods calculated using the GMMs
- **Variables involved:**
  - Image can be considered as an array
    - z= { z_1, z_2, ……, z_N} where N is the total number of pixels and in our case, Z_i = {red channel component, green channel component, blue channel component}
  - The current status (foreground/background) estimated for the pixel (X):
    - x = { x_1, x_2 , …, x_N } where:
      - if x_i == 0 : pixel is currently estimated to be a part of the background
      - if x_i == 1: pixel is currently estimated to be a part of the foreground
  - Estimated distributions of the foreground and background pixels
    - Here, each class (fg and bg) are modelled separately with each K full-covariance Gaussian Mixture Models (authors suggest K = 5).
    - In order to deal with the GMM tractably, in the optimization framework, an additional vector k = { k_1 ,., k_N } is introduced, with $k_i \in$ { 1,...., K }, assigning, to each pixel, a unique GMM component, one component either from the foreground or the background model, depending on what is the current estimated category of the pixel (ie x_i==0 or x_i==1).
    - The variable w is used to denote the parameters of the Gaussian Mixture Models.
- An energy function E is defined so that its minimum should correspond to a good segmentation, in the sense that it is
  - guided both by the given foreground and background intensity GMMs
  - opacity (ie x) is “coherent”, reflecting a tendency to solidity of objects.
- This energy term is modelled as:
  - $E(x, \omega, z) = U(x, \omega, z) + V(x, z)$

Data Term

- The data term U evaluates the fit of the segmentation x to the pixel data z, given the model ω, and is defined for all pixels in T_U as:

  Given its intensity value, how likely is a pixel to be foreground or background?

$$U(x, \omega, z) = \sum_{pixel \in T_U} -log[h_B(z_i)][x_{pixel} == 0] \ -log[h_F(z_i)][x_{pixel} == 1] + \sum_{pixel \in T_F or T_B} \text{if (assigned category != user-defined category) : large}$$

Using a large constant in the error function is a **frugal way** of enforcing constraints in my implementation. Here, $h_{class}(pixel)$ is the highest likelihood (among all K components of the class) that the pixel belongs to the particular component.

Smoothness Term

- The smoothness term V evaluates whether pixels which are close to each other in terms of location and intensity belong to same category. For a pair of neighbouring pixels: if they belong to the same category, the penalty is zero ELSE the penalty depends on the similarity in their intensity and the extent of the closeness location-wise.

  Given their intensity values, how likely are two neighboring pixels to have two labels?

$$V(x, z) = \gamma \sum_{\text{pixel 1, pixel 2 are neighbours}} [\text{0 if they have same label else 1}] \ * \ dis(\text{pixel 1}, \text{pixel 2})^{-1} \ * \ (e^{-\beta ||z_{\text{pixel 1}} - z_{\text{pixel 2}}||^2})$$

Here,

- β = $(2 \times E[(z_i - z_j)^2])^{-1}$ where E is the expectation operator over all pairs of neighbouring pixels. Modelling this in this manner tends to lower the penalty for assigning different classes to the 2 pixels if their RGB values differs significantly despite them being close to each other location-wise
- dis(·) is the Euclidean distance of neighbouring pixels in 2D space

Problem Formulation

- The problem can now be formulated as:

$$x_{best} = arg\ \underset{x}{min}(\underset{\omega}{min}\ \ E[x, \omega, z])$$

We clearly see that the problem has 2 major steps:

1. Segmentation using graph cuts (Requires having the foreground-background distribution)
2. Foreground-background modelling using unsupervised clustering (Requires having segmentation)

This **cyclic dependency is modelled as** an ITERATIVE algorithm

My Implementation Details (GrabCut - Iterative Optimisation)

Step 1: Initialise Mixture Models based on user annotation

- The trimap labels (T_U, T_B, T_F) for the pixels are determined by using the initial bounding box drawn by the user.
- All pixels outside the box are put in T_B and all pixels inside the box are put in T_U
- Now, for initialisation, all pixels in T_B are assigned an estimated label as BG ($X_i = 0$) and all pixels in T_U are assigned an estimated label as FG ($x_i = 1$). Then, K GMMs are assigned for the FG label using pixels having $x_i == 1$ and K GMMs are assigned for the BG label using pixels having $x_i == 0$.
- Which FG nodes will be used to decide which of the K components for FG label is decided on the basis of K-means clustering.

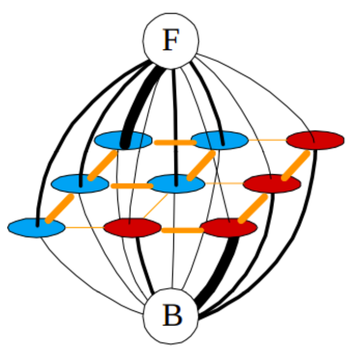
STEP 2: Assign GMM components and Learn GMM parameters

- This step is skipped in case of the 0th iteration as GMMs were already initialized in STEP 1.
- As a result of the previous iterations, due to the partitions created after applying the MIN-CUT algorithm (STEPS 3 and 4), the labels for some of the pixels might have changed.
- For each pixel in FG, we find the component among the K GMM Foreground components which has the highest likelihood of having this pixel (score_samples() function).
- Bullet 3 is repeated for pixels having BG label as well.
- Now, the GMM parameters need to be relearned. This is easily done for each of the $2K$ components by using the data points allotted to that component in BULLETS 3 and 4.

State of the Pixel	Description
pixel $\in T_B$ and $x_{\text{pixel}} = 0$	Surely background
pixel $\in T_F$ and $x_{\text{pixel}} = 1$	Surely foreground
pixel $\in T_U$ and $x_{\text{pixel}} = 0$	Possible background pixel
pixel $\in T_U$ and $x_{\text{pixel}} = 1$	Possible foreground pixel

STEP 3: Formulation of a graph and estimation of segmentation (using min-cut)

- Each pixel = node
- 2 additional nodes (source and sink) corresponding to the FG and BG classes.
- `Edges:`
    - **Type 1: N-links** connect pixels in 4/8 neighbouring pixels around them. The weights are assigned using the smoothness term defined above. The weights remain constant as the algorithm proceeds.
    - **Type 2: T-links** connect pixels to the source (FG) and sink (BG) nodes. If a pixel P is more likely to belong to FG than class BG, then the value for negative likelihood for FG will be lesser than the negative likelihood for BG and vice versa. Hence, pixel node is connected to BG using a weight derived from the negative log-likelihood of the FG class and vice versa. The weights of these edges change across the iterations.



- **Energy optimization equivalent to min cut.**
- **Algorithm to find the best possible labelling:** Hence, MIN-ST cut is applied to the graph to partition the graph into 2 parts such that the SOURCE and SINK belong to different parts. Due to the minimum nature of the cut, it is inferred that all nodes which are in the same partition as FG are likely to be FG pixels and vice versa. The x_i values of the pixels are modified based on which partition they lie in.
- For 2 label based segmentation, the above is solvable by standard flow algorithms and is polynomial time in theory and from my experience, it looked linear in practice.
- Repeat STEP 2

Possible STEP 4: User Refinement

The user has the option to relabel some mislabelled pixels and hence, guide segmentation in the right direction. For instance, if the user has manually labelled a pixel as belonging to BG, then the pixel is put in T_B and it's x_i value is changed to 0. The x_i value for this pixel would NOT be allowed to change in future iterations.

NOTE: EXPERIMENTS ON NEXT PAGE