B657 Activity, Belief Propagation

Spring 2017

Let's say you want to separate the foreground object(s) of a photo from the background. In other words, we want to segment the image into two regions. Let's use a semi-supervised approach. Our program will take a color image and a set of "seed points" that indicate where some of the foreground and background pixels in the image are located. These seed points could be quick strokes drawn manually by a human, like in Figure 1. The program will then use these seeds in an MRF model to produce a complete segmentation of the image, partitioning the image into foreground and background (Figure 1).

To get you started, we've created a skeleton program called "segment," which takes an image to be segmented and a second image with seed points. The seed points image is just a blank image with a few red and blue areas, that show the background and foreground seed points, respectively. Run the program like this:

```
./segment input.png seeds.png
```

The program should create an image that's a disparity map, that has only two possible values: some relatively high disparity for foreground pixels that should be seen as close to the viewer, and some relatively low disparity (probably 0) for background that should be far away. The existing skeleton code doesn't do very much – it just computes a random depth map.

1. As a first step towards a better segmentation algorithm, we should at least make use of the seed points provided to the program. The seed points are a set $\mathcal{F} \in I$ of pixels of image I known to be foreground points, and a set $\mathcal{B} \in I$ of points known to be background points. Based on the set of points in $\mathcal{F}$, we can build a simple appearance model of what foreground pixels "look like," for instance by simply estimating means $\mu = [\mu_R \ \mu_G \ \mu_B]^T$ and variances $\Sigma = \text{diag}(\sigma_R, \sigma_G, \sigma_B)$ of the RGB color values in $\mathcal{F}$. Let $L(p)$ denote a binary label assigned to pixel p. Then we can define a function that measures the "cost" of giving a label $L(p)$ to pixel p,

$$D(L(p), I(p)) = \begin{cases} 0 & \text{if } L(p) = 0 \text{ and } p \in \mathcal{B} \\ 0 & \text{if } L(p) = 1 \text{ and } p \in \mathcal{F} \\ \infty & \text{if } L(p) = 0 \text{ and } p \in \mathcal{F} \\ \infty & \text{if } L(p) = 1 \text{ and } p \in \mathcal{B} \\ -\log N(I(p); \mu, \Sigma) & \text{if } L(p) = 1 \text{ and } p \notin \mathcal{F} \text{ and } p \notin \mathcal{B} \\ \beta & \text{if } L(p) = 0 \text{ and } p \notin \mathcal{F} \text{ and } p \notin \mathcal{B}, \end{cases}$$

   where β is a constant and $N(I(p); \mu, \Sigma)$ is a Gaussian probability density function evaluated with the RGB color values at $I(p)$. Intuitively, this says that one pays no cost for assigning a 1 to a pixel in $\mathcal{F}$ or a 0 to a pixel in $\mathcal{B}$, but we pay a very high cost for assigning the wrong label to one of these known pixels. For a pixel not in $\mathcal{B}$ or $\mathcal{F}$, we pay some fixed cost β to assign it to background, and a cost for assigning it to foreground that depends on how similar its color is to the known foreground pixels.

   Implement a function called `naive_segment()` that chooses a most-likely pixel for each pixel given only the above energy function, i.e. computes the labeling,

$$L* = \arg\min_L \sum_{p \in I} D(L(p), I(p)).$$

   To help you test your code, we've given you some sample images and seed files in the images/ directory of the skeleton code archive.

2. The above formulation has a major disadvantage, of course: it doesn't enforce any sort of spatial coherence on the segmentation result. To fix this, let's define a more complicated energy function,

$$E(L, I) = \sum_{p \in I} D(L(p), I(p)) + \alpha \sum_{p \in I} \sum_{q \in \mathcal{N}(p)} V(L(p), L(q))$$

where α is a constant, $\mathcal{N}(p)$ is the set of 4-neighbors of p (i.e. $\mathcal{N}(p) = \{(i-1, j), (i+1, j), (i, j-1), (i, j+1)\}$ for $p = (i, j)$ not on the image boundary), and $V(\cdot, \cdot)$ is a pairwise cost function that penalizes disagreement, e.g. $V(a, b) = 0$ if $a = b$ and 1 otherwise.

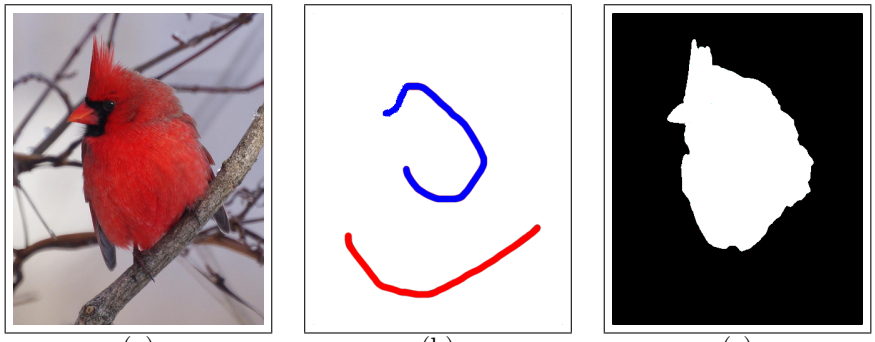
Now to actually perform segmentation, we simply need to minimize the energy; i.e. find $L*$ such that,

$$L* = \arg\min_{L} E(L, I).$$

As we saw in class, this is an MRF energy minimization problem. Implement loopy belief propagation to (approximately) minimize this energy function. Your final program should produce two disparity maps, one for the simple approach of step 1, and one for the MRF of step 2.

What to turn in

Please submit your source code and whatever output images you can to Canvas.



(a) (b) (c)

Figure 1: Sample semi-supervised segmentation problem and results: (a) input image, (b) foreground (blue) and background (red) strokes drawn by a human, indicating roughly where the foreground and background regions of the image are, (c) sample disparity map produced by segmentation.