# Image Segmentation Using Markov Random Fields: An Implementation

Jake Callahan, James Griffin, McKay Harward, Taylor Paskett, Isaac Robbins, Mingyan Zhao

April 21, 2020

**Abstract**

The goal of this project was to evaluate current use of Markov random fields in image segmentation, then to implement one of the more advanced methods. In this paper we discuss how we implemented the equations from a paper based on an unsupervised method that used metropolis Hastings algorithm to find the best parameters for the Markov random field as well as the best number of classes. In short, we coded up a hands-off method of segmenting an input image.

## 1 Background and Motivation

Image segmentation is one the the most important problems in machine learning today. Industries like medicine, transportation, and manufacturing have all been benefited by the rapid advances we have made in creating image segmentation systems. One of the most interesting and valuable fields of study is the area focused on unsupervised image segmentation methods. A powerful and accurate unsupervised image segmentation system would have applications in neural photo editing using introspective adversarial networks as well as in image recognition and other computer graphics applications. In this paper, we examine and implement [INSERT NAMES HERE]'s idea for using Markov random fields for unsupervised image segmentation.

The idea for using Markov random fields for image segmentation has been around for a long time; much research has been done in this area, mostly focusing on supervised learning methods. Most of this research centers around segmenting images based on pretraining a Markov random field on a class, and then estimating which pixels in a given image are most likely to be of that class. While this is useful, the more hands-off the approach the easier it is to apply to new scenarios.

Unfortunately, research regarding hands-off applications is virtually nonexistent. Other than [INSERT NAMES HERE]'s research, which was published in 2000, we could find no other papers examining the use of Markov random fields for unsupervised segmentation. The main drawback of this unsupervised method is that it trades an initial investment of creating pertained Markov random fields for a computationally expensive algorithm using metropolis Hastings. Thus, it might be possible that limitations in computing power kept further research in this area from being valuable. We believe that with the advances in computing power over the last 20 years it is now valuable to revisit these methods in order to determine if applications are possible in the current world of machine learning.

## 2 Data

The algorithm we outline below trains itself on each image it operates on. As such, we didn't need (or want) a large dataset to use in this project. Instead, we curated a small corpus of images by hand that we believed would be useful in providing us information about how well our algorithm was training and making predictions. The full set of those images can be found here: [INSERT LINK TO ALL IMAGES WE USED]

## 3 Methods

The algorithm can be outlined in 6 steps:

1. Random initialize each Markov random field for a randomly drawn number of classes
2. Segment the image based on these MRFs
3. Sample noise parameters $\mu$ and $\sigma$
4. Sample MRF parameters $\beta_0$ and $\beta_1$
5. Sample number of classes
6. Repeat steps (2) - (5) until convergence

The sampling method follows the same basic pattern: sample new parameters from their prior distributions, calculate an acceptance probability (or likelihood) for those parameters, and then use the Metropolis-Hastings algorithm to choose to accept/reject those parameters.

On the surface, this algorithm is simple and was defined in [INSERT NAMES HERE]'s paper. However, many of the equation definitions and notational choices were unclear and required extensive interpretation. Further, some of the equations as defined in the text suffered from underflow/overflow errors and thus required considerable reworking to be suitable for implementation. We define now:

## 3.1 Equation Definitions

In the following equations, we use the following notation: $c$ denotes each segmentation class, $T_t$ denotes the temperature value at time $t$, and $x_s$ and $y_s$ denote the label and value of pixel $s$, respectively (note that this is the reverse of how machine learning data is traditionally notated).

### 3.1.1 Noise

The noise parameters $\mu$ and $\sigma$ affect [INSERT HOW NOISE AFFECTS HERE]. They are sampled using Metropolis-Hastings with the following likelihood:

$$\begin{aligned} p\left(\mu_c, \sigma_c | Y, X\right) &\propto \prod_{s:x_s=c} p\left(y_s|\mu_c,\sigma_c\right) p_r\left(\mu_c\right) p_r\left(\sigma_c\right) \\ &= \frac{1}{\sigma_c\left(2\pi\sigma_c^2 T_t\right)^{n_c}} \exp\left\{-\frac{1}{2T_t}\sum_{s:x_s=c}\left(\frac{y_s-\mu_c}{\sigma_c}\right)^2\right\} \end{aligned} \tag{1}$$

Here, $n_c$ denotes the number of pixels labeled with class $c$.

### 3.1.2 MRF

The noise parameters $\mu$ and $\sigma$ are sampled using Metropolis-Hastings with the following probability:

$$\begin{aligned} & p\left(\beta_c^{(0)}, c \in \Lambda | X\right) \\ &\quad = p\left(X|\beta_c^{(0)}, c\in\Lambda\right) p_r\left(\beta_c^{(0)}, c\in\Lambda\right) \\ &\quad = \prod\nolimits_{(c\in\Lambda,\psi_\eta)} \left(\frac{\exp\left(-\frac{1}{T_t}\left[\beta_c^{(0)}+\beta^{(1)}V(c,\eta)\right]\right)}{\sum_{i\in\Lambda}\exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)}+\beta^{(1)}V(i,\eta)\right]\right)}\right)^{n_{(c,n)}} \\ &\quad = \prod\nolimits_{(c\in\Lambda)} \left(\frac{\exp\left(-\frac{1}{T_t}\left[\beta_c^{(0)}\right]\right)}{\sum_{i\in\Lambda}\exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)}\right]\right)}\right)^{n_c} \\ &\quad \times \prod\nolimits_{(c\in\Lambda,\psi\eta)} \left(\frac{\exp\left(-\frac{1}{T_t}\left[\beta^{(1)}V(c,\eta)\right]\right)\sum_{i\in\Lambda}\exp\left(-\frac{1}{T_t}\left[\beta_i^{(0)}\right]\right)}{\sum_{i\in\Lambda}\exp\Big(-\frac{1}{T_t}\left[\beta_i^{(0)}+\beta^{(1)}V(i,\eta)\right]}\right)^{n_{(c,n)}} \end{aligned} \tag{2}$$

### 3.1.3 Number of classes

The number of classes are sampled with likelihood

$$\begin{aligned} &\frac{p\left(X=x^+,\psi^+,k^+|Y=y\right)}{p(X=x,\psi,k|Y=y)}\frac{1}{p_\beta\left(u_1\right)p_\beta\left(u_2\right)p_\beta\left(u_3\right)} \\ &\quad\times\frac{\partial\left(\Psi_{c1},\Psi_{c2}\right)}{p(\text{segmentation})}\left|\frac{1}{\partial\left(\Psi_c,u_1,u_2,u_3\right)}\right| \end{aligned} \tag{3}$$

## 4 Results

## 5 Analysis

Our methods turned out to be fairly effective. They don't par with current state of the art technologies but they definitely do well for something done by a few undergraduate students in a single project, not research. The model we've created does a fairly decent job of segmenting images on par with what the original authors achieved, though sometimes in a few extra iterations. Looking at how it works, I do think it would work in a GAN for neural photo editing to maybe train the GAN to focus itself more closely on one area while keeping the rest of the image the same, but even with faster computers, it seems it still takes too long for any practical applications. Given enough computing power it can be used.

## 6 Conclusion

The initial intent of getting a completely unsupervised method of image segmentation using Markov Random Fields was successful and could probably be improved upon for efficiency. If given the opportunity to try image segmentation again, I would probably opt to use a neural network as they tend to require an initial investment of training time but can then be used to segment an image very quickly. Something we could do is have more informed priors for each parameters. We could also implement for color images (may be?)