Image Segmentation via MCMC

1. PROBLEM

Image segmentation in computer vision is the process of grouping parts of an image into related segments. There are numerous potential applications of image segmentation including face and fingerprint detection as well as a variety of medical imaging uses. As this technology becomes more accurate and reliable, it could be used to identify tumors or to help guide surgery with less errors than humans.

Image segmentation is an essential and, often, an initial task in image analysis that is used to distinguish objects of interest from the rest of the image [Lucas 2010]. However, image segmentation is a very challenging and long standing problem. Images are incredibly complex and include a variety of objects at different scales. These objects may also be occluded by other objects and the appearance of the objects can be significantly changed by lighting conditions. Since image segmentation is a well-researched area, many solutions to these problems have been proposed, each with advantages and disadvantages [Dellaert 2007].

2. RELATED WORK

K-means is an algorithm used for clustering N data points into k disjoint subsets, and can be used for image segmentation. This algorithm is simple, fast, and easy to implement, but can also be skewed by outliers. Also, choosing a K value that generates the most accurate results can be difficult. [Hoiem 2010]. Our method automatically optimizes regions.

Region growing methods exploit the fact that pixels which are close together will most likely have similar grey values. One example is the Watershed algorithm, where an image is interpreted as a topographic surface where the gray-levels of the image represent the altitudes, and water flows down the image into regions [Lucas 2010]. This algorithm is only as good as the soft boundaries and may not be able to choose a variety of regions for multiple segmentations. Additionally, These methods do not incorporate top-down information [Hoiem 2010], but our method does.

Graph-cut methods treat image pixels as verticies in a graph and attempt to cut (segment) the image into regions via min-cut algorithms, and can provide very fast inferencing. However, these methods are not always applicable to the problem, require unary terms, and are sensitive to noise [Hoiem 2010]. Our method handles noise and clutter by using multiple salient features.

3. APPROACH

Images within our implementation are represented as Markov networks. To segment an image using Markov networks, or Markov Random Fields (MRF), one approach is to represent the image pixels by a graph of nodes with probabilities and priors being the pixel values. A common method of inference in MRF is Gibbs sampling.

Fig. 1. Original color input image from the BSDS database

Fig. 2. Grayscale segmented output image from running our DDMCMC algorithm on the color input image

Gibbs sampling is an MCMC method that approximates values from the Markov network to find pixels that are similar [Tu and Zhu 2002].

In our MCMC image segmentation implementation, we use a color image as input for the algorithm. A sample input image can be seen in figure 1. The image is first converted to grayscale, and the grayscale values of each pixel are thresholded into several regions. The result of this process looks like a much noisier version of the image in figure 2. These grayscale regions of similarly thresholded pixels are then used as the priors of the MRF graph. The Markov network graph is then constructed. Next, for a given number of iterations, the algorithm loops through all pixels in both the grayscale region image and the color image to calculate the energy at the current pixel. This calculation can be represented as follows for each pixel:

```
classEnergy(pixel)=gibbsEnergy(pixel)+globalAvg(pixel)
```

Fig. 3. Binary contour image of the grayscale output image from our algorithm, used for comparison against the human segmented image

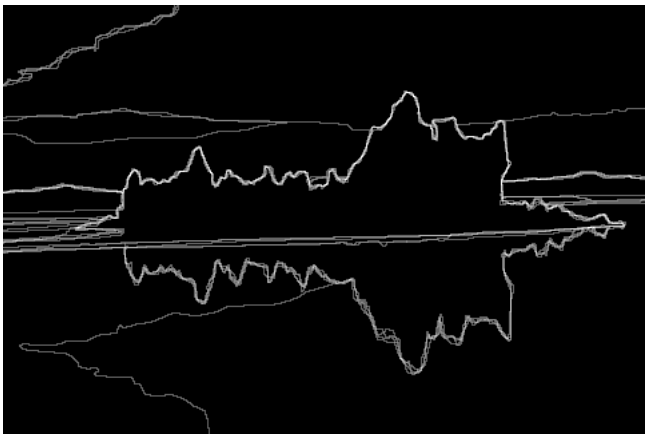

Fig. 4. Human segmented image from the BSDS database; used as the reference for segmentations of the original color image

The Gibbs energy is based on color image information and the grayscale image segment priors. Each pixels RGB values are converted into the Onta I1I2I3 colorspace during the Gibbs sampling process. In the current implementation, the sampling and classification is done by looping through all the pixels linearly, instead of randomly, though multiplie iterations of sampling can influence classification. The segment classes may be changed for a pixel based on the Gibbs energy of the pixels neighbors. This process helps smooth the pre-segmented grayscale image into better regions. The algorithm outputs a grayscale image, as shown in figure 2, consisting of many regions where each region is color coded. Finally, for the evaluation of the algorithm in the BSDS Benchmark, a black and white image is created showing only the outline of each boundary, as seen in figure 3.

4. EVALUATION

Our algorithm was evaluated against The Berkeley Segmentation Dataset and Benchmark [Berkeley 2005] program. The BSDS is a set of Matlab scripts that compare any segmentation algorithms output images against well-defined images that have been hand segmented into regions and labeled by humans. An example of a hand-segmented image can be seen in figure 4.

The benchmark measures two values, precision and recall, to produce a precision-recall curve for the input algorithm. Precision is defined as the probability that a machine-generated boundary pixel is a true boundary pixel. This provides a measure of how much noise is in the output of the detector. Recall is defined as the probability that a true boundary pixel is detected. Recall provides a measure of how much of the ground truth is detected. Here, ground truth is the hand-segmented images created by humans. A precision-recall curve is then created from these values and shows the inherent trade-off between the two. This is the trade-off between misses and false positives as the detector threshold changes.

As shown in figure 5, our algorithms precision and recall were 58% and 41%, respectively. The algorithm was set to output 4 segments using 2 iterations of Gibbs sampling with a weighting factor $\beta = 0.25$.

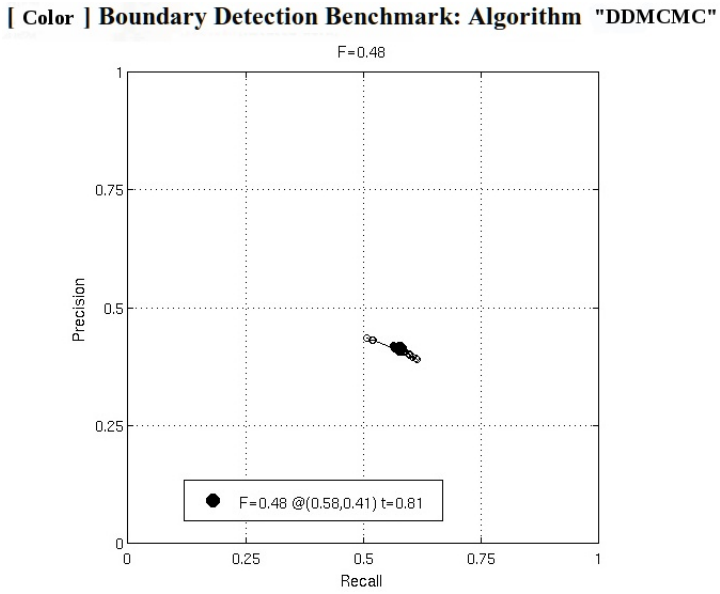

Fig. 5. F-measure of DDMCMC, from BSDS Benchmark

Additionally, the BSDS computes the F-measure, which is the harmonic mean of precision and recall. The F-measure is defined at all points on the precision-recall curve. The maximum F-measure value across an algorithm's precision-recall curve is used as its summary statistic which allows different algorithms to be quantitatively compared [Berkeley 2005].

Our algorithm's global average F-measure was $F = 0.48$, for the same configuration as mentioned previously.

Fig. 6. Original color input image from the BSDS database

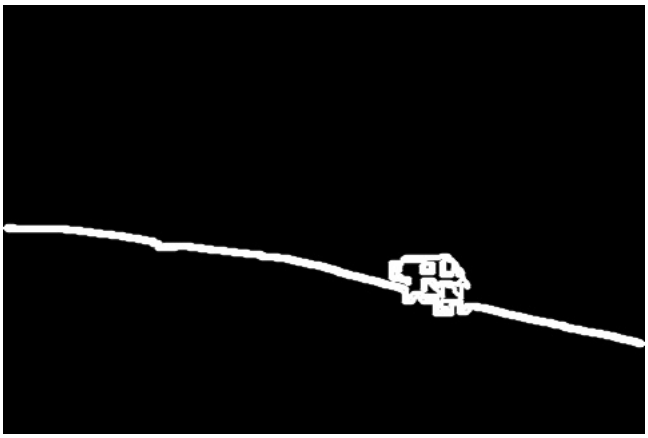

Fig. 7. Binary contour image of the grayscale output image from our algorithm, used for comparison against the human segmented image

5. DISCUSSION

5.1 Analysis

The F-measure is the harmonic mean of the precision and recall score of the BSDS Benchmark, and ranges from [0, 1], where a higher F-measure indicates results that are more accurate than a lower F-measure.

The overall F-measure for our implementation was $F = 0.48$ which has the same accuracy as the segmentation induced by scale-invariance technique from the Berkley Segmentation Benchmark results. The best F-measure is from the Global Probability of Boundary technique which produced a value of $F = 0.70$ [Berkeley 2005]. There are nine other algorithms listed that produced more accurate results than our implementation. Many of them combine several image preprocessing steps such as edge detection, mean color regions, contour maps, gradient extraction, and color-space shifts, to use in a weighted heuristic for classifying segment regions [Berkeley 2005].

Our best F-measure for a single image was $F = 0.88$, as seen in figures 6 and 7, in contrast to the best known score in the Berkeley Database, $F = 0.92$, produced by the Ultrametric Contour Maps (UCM) technique [Berkeley 2005]. The UCM algorithm defines ultrametric distances for boundary extraction by integrating local contour cues along the regions' boundaries and combining this information with intra-region attributes [Berkeley 2005]. In the results of this particular image, there are only two algorithms (out of the top 10) that produced a better F-value than our implementation: UCM and min-cover approaches [Berkeley 2005].

5.2 Future Work

In the future, we would like to implement a few improvements on our current work. Some techniques incorporate image preprocessing, such as the use of edge detection, before the image is sampled. Edge detection could improve our results because this process would much more clearly define the regions. Our algorithm could also be evaluated while scaling the image to a few different sizes. The results from each size could be combined to produce a better final segmentation.

The segmentation result can be significantly altered by changing the number of segments that the algorithm is trying to identify, so finding the correct number of segments can be very important in finding a good solution [Dellaert 2007]. We would like to conduct more research on finding the optimal number of segments. Finally, the number of iterations of Gibbs sampling can have an effect on the accuracy of the solution. It would be interesting to do an analysis of the results obtained by varying the number of iterations.

While working on this problem, we have learned that computer vision is a very challenging area of computer science. This project focused on image segmentation, which is only trying to break an image up in to regions that are associated with one another. This is a task that is so simple for humans to accomplish, but very challenging for computers. It seems like there is a limit to how well image segmentation can work without the computer knowing much more about the objects that are in the picture. However, teaching a computer about objects that may exist within an image makes image segmentation an even more challenging task.

REFERENCES

BERKELEY, U. 2005. The berkeley segmentation dataset and benchmark. *Online Dataset*.

DELLAERT, F. 2007. Part 3: Mcmc tutorial : Monte carlo methods. *Slides from Monte Carlo Methods in Vision and Robotics*.

HOIEM. 2010. Segmentation and grouping. *Slides from CS 543: Computer Vision, University of Illinois*.

LUCAS. 2010. Image segmentation. *Technische Universitaat Maunchen*.

TU AND ZHU. 2002. Image segmentation by data-driven markov chain monte carlo. *PAMI*.