Image Segmentation via Total Variation and Hypothesis Testing Methods

Dennis Sun
Stanford University
Dept. of Statistics
dlsun@stanford.edu

Matthew Ho
Stanford University
Dept. of Electrical Engineering
matthew.ho@stanford.edu

Abstract

We introduce a novel algorithm for image segmentation by revisiting a traditional method, the watershed algorithm, and introducing a hypothesis testing framework to solve the problem of oversegmentation. The hypothesis tests are used to determine whether or not to split a region or merge two neighboring regions. We also introduce a new nonlinear filter which appears to be useful for image segmentation. We show that our methods achieve comparable and even superior performance to many established methods for image segmentation, such as normalized cuts.

1. Introduction

Image segmentation is a fundamental task in computer vision. It is a prerequisite for many higher-order tasks, such as object detection and recognition. Despite this, segmentation is still very much an open problem. In this paper, we begin by describing total variation denoising as an alternative to traditional nonlinear edge-preserving filters, such as median filters. Then we describe our algorithm for successively merging and splitting regions, starting from an oversegmentation, such as one obtained from the watershed algorithm. Our results on the Berkeley Segmentation Dataset follow, and we conclude by discussing potential for future work.

Figure 1 shows some

2. A New Nonlinear Filter for Image Segmentation

Total variation denoising (TVD) is a robust algorithm for reconstructing noisy images.[9] The most famous form of total variation denoising, anisotropic TVD, finds a smoothed image J which solves the criterion:

$$\underset{J}{\text{minimize}} ||I-J||_2^2 + \lambda \sum_{x,y} |J_{x+1,y} - J_{x,y}| + |J_{x,y+1} - J_{x,y}|$$

Figure 1. Buffalo image from the Berkeley Segmentation Dataset, original and smoothed using total variation denoising.

where I is the original image. The L1 penalty on the differences between adjacent pixels encourages sparsity and hence total variation denoising tends to result in images which are piecewise constant. The particular algorithm used in this paper relies on the Split Bregman Method, described in [4]. Figure 1 shows the visual effect of TVD smoothing on an image.

One way to obtain segmentations from the TVD-smoothed image is to take each piecewise constant region as a segment. However, because of imprecision in the optimization algorithm (first-order iterative methods are typically used to solve such problems), the piecewise constant regions may not appear as such. Also, if the intensity varies gradually across the image, then TVD smoothing will not in general return constant regions.

Nevertheless, TVD is useful as a nonlinear filter which preserves edges, a feature of obvious importance in image segmentation[2]. Since linear filters such as the Gaussian are completely characterized by their frequency response, they cannot filter away noise while preserving edges, since both correspond to the high frequency components of an image. Thus, some form of a nonlinear filter is necessary. The computer vision community has traditionally used median filters for this purpose[11]. However, we found that the median filter tended to produce slightly less smooth segmentations as compared with TVD, when combined with our merging and splitting algorithm. Figure 2 compares the two.

We decided to exploit the edge-preserving property of TVD to obtain a preliminary segmentation from the TVD-

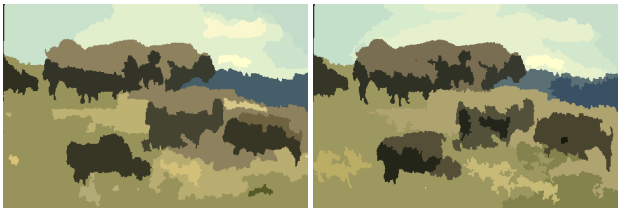

Figure 2. The result of segmenting after smoothing the buffalo using TVD (left) and using median filtering (right). The results are similar, although the TVD result is slightly smoother. This may be because there is less numerical instability with TVD since it does not have the tendency to set adjacent values *exactly* equal to zero. This causes problems when we compute, for example, t-statistics and divide by the variance within each region, as discussed in Section 3.

Figure 3. The top two images show the resulting segmentation after applying k-means directly to the original image and to the TVD smoothed image. The latter is much more visually appealing. To show that arbitrary smoothing will not necessarily solve the problem, the bottom left image depicts the results from applying a Gaussian filter to the image. Finally, the bottom right image depicts the ground truth (24 segments) as labeled by a human.

smoothed image. We first took the gradient of the TVD-smoothed image, then applied the watershed algorithm to the gradient. The watershed algorithm groups together all pixels which would fall into the same "basin". One can imagine flooding the plane of the image and taking each "pool" of water thus obtained as a segment. The intuition for applying watershed to the gradient is that high points in the gradient correspond to edges (since the noise has been filtered away), thus the basins correspond to the segments between edges.

It is important to note that TVD is useful not only in conjunction with the algorithm we are about to propose but with any segmentation algorithm. Figure 3 shows how TVD smoothing can improve the results of k-means clustering as applied to image segmentation.

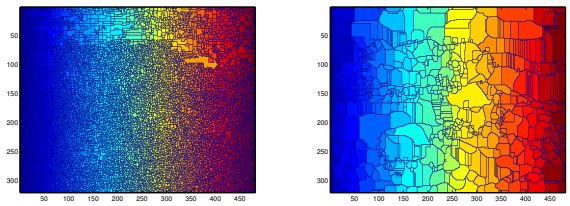

Figure 4. The watershed algorithm as applied to the original and to the TVD smoothed buffalo image. The smoothing helps reduce the number of segments, but the image is still oversegmented.

3. A Hypothesis Testing Approach to Region Merging and Splitting

The watershed algorithm associates one segment with each local minimum in the gradient, and although smoothing helps, the watershed algorithm still produces an oversegmentation, as the image processing community has long been aware. The traditional approach to overcome this problem is to preprocess the image before applying the watershed algorithm. One popular approach is to use region markers, but the downside is that these markers often need to be specified by a human, although algorithms exist to automate the marker selection process.[5]

We focused on *post*processing the watershed segmentation. The first approach we tried was to reduce the number of segmentations by flattening all local minima that were not at least a certain depth (as measured by the difference in intensity between the minimum and the nearest local maximum). However, we found that although flattening shallow minima did initially improve the segmentation, it was not possible to continue the procedure until only the desired number of clusters remained.

We ultimately devised a *merging and splitting* algorithm for reducing the number of segmentations. Although the name is inspired by region merging and splitting, a classic segmentation algorithm, our algorithm is similar only in spirit. Classical splitting and merging refers specifically to an algorithm whereby an image is successively divided into four quadrants, then successively merged when no further segmentations are possible. By contrast, our algorithm is bottom-up, starting with a fine segmentation and successively merging regions and splitting as necessary. The two components of this process are described below.

3.1. Merging

We iterate over each watershed region and compute the "distance" between the region and its neighbors. We then successively merge the two neighboring regions in the image that are the "closest," as measured by this metric. The distance between all relevant regions are recomputed after each merger.

The most obvious metric is the difference in mean intensity between the two regions. The "hypothesis testing" in the title of this section refers to the fact that the t-statistic can also be a useful metric:

$$t_{ij} = \frac{\bar{X}_i - \bar{X}_j}{\sqrt{\frac{s_i^2}{n_i} + \frac{s_j^2}{n_j}}}$$

where $\bar{X}_i$ and s_i refer to the mean and standard deviation of the pixel intensities in region i and n_i refers to the number of pixels. Intuitively, the t-statistic compares the between region difference with the within region difference. The t-statistic will be small if: (1) the difference between the regions is small, (2) the variance within region is large, and (3) the regions are small. In all three cases, it makes sense to favor merging, and the t-statistic takes this into account. A comparison of difference in means and the t-statistic is provided in Section 4.

In our implementation, we used the t-statistic only as a metric, without taking advantage of the hypothesis testing framework. This is because the evaluation benchmarks that we used required us to prespecify a certain number of segmentations, so it makes sense to continue taking the minimum t-statistic and merging until only the desired number of regions remained, regardless of whether the t statistics were statistically significant. However, our method can easily be extended to a real-world situation in which the number of segments is not known in advance. In such cases, we could simply set a significance threshold and stop once the t-statistics all exceed this threshold.

Lastly, it is worth mentioning that we applied our segmentations to multichannel images. Thus, for each region, there are actually a set of (three) means and standard deviations. For the sake of computation, we simply took the root-mean-square of the individual t-statistics for each channel (i.e. the 2-norm of the vector of t-statistics). We found no considerable advantage to using other norms (1-norm, ∞-norm). However, a gain in performance might be achieved by taking into account the multivariate structure of the data. The multivariate generalization of the t-statistic is the Hotelling's two-sample T-squared statistic, given by:

$$T_{ij}^2 = \frac{n_i n_j}{n_i + n_j}(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j)^T \hat{\Sigma}^{-1}(\bar{\mathbf{x}}_i - \bar{\mathbf{x}}_j)$$

Our approach of taking the 2-norm of the marginal t-statistics is essentially equivalent to assuming that the channels are uncorrelated, an assumption which is clearly violated. However, it remains to be seen whether we could achieve a gain in performance by taking correlations into account.

3.2. Splitting

Merging yields satisfactory results, as we show in the following section, but it is a greedy algorithm, so it will inevitably make mergers that make sense locally but not globally. Therefore, it is desirable to have some mechanism for backtracking so that we are not stuck with a merger once it has been made. Figure 5 shows how splitting can improve a segmentation.

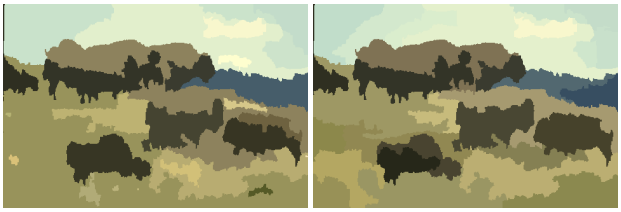

Figure 5. The left image shows the result of applying merging without splitting. The buffalo near the top of the image are joined in one segment with the grass along a very narrow strip. Such connections form because of the greedy nature of the merging algorithm and are broken when we apply a splitting step as well, as the right image shows.

We handle the problem of splitting regions by checking the existing regions every couple of iterations to see if any splits should be made. As with merging, splitting is binary: we only split regions into two at a time. To decide whether a region should be split, we perform k-means clustering on the features of the component watershed segments (from the original watershed segmentation) that make up the region. The reason for using watershed segments as observations rather than the pixels themselves is that individual pixels are noisy and result in oversegmentation. (See Figure 6.) We lose very little by grouping the pixels into watershed segments, since the initial oversegmentation captures every edge in the original image.

We used 5 features: the pixel coordinates (x,y) and the average intensity in each channel (r,g,b). Then we compare the squared error from the model which assumes only a single cluster, with the squared error from the model which assumes two clusters, i.e.

$$RSS_1 = \sum_{i \in R} ||\mathbf{x_i} - \bar{\mathbf{x}}||_2^2$$

$$RSS_2 = \sum_{i \in R_1} ||\mathbf{x_i} - \bar{\mathbf{x}}_\mathbf{1}||_2^2 + \sum_{j \in R_2} ||\mathbf{x_j} - \bar{\mathbf{x}}_\mathbf{2}||_2^2$$

where the mean is over the three channels and the sum is over the pixels in a given region. Note that RSS_2 will always be less than RSS_1. In order to compare whether RSS_2 is *sufficiently* smaller than RSS_1 to justify the two clusters, we appeal to the equivalence between k-means clustering and the EM algorithm for the Gaussian mixture model. We assume that the data come from a Gaussian mixture with two components with means μ_1 and μ_2 and equal variances, and we are testing the hypothesis that $H_0 : \mu_1 = \mu_2$ (in which case the data essentially come from one cluster).

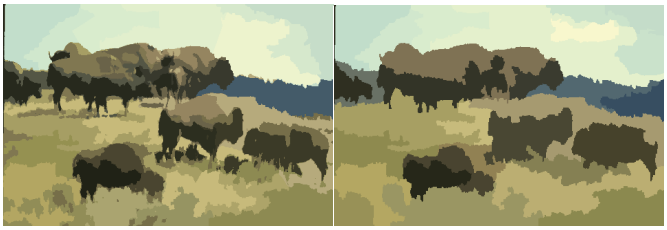

Figure 6. Two segmentations of the buffalo image: on the left, the k-means clustering in the splitting step is performed at a pixel level; on the right, it is performed on watershed regions from the initial segmentation. The latter leads to more contiguous regions because there is less noise in groups of pixels than in individual pixels.

The conventional method for testing whether a more general model (i.e. mixture of two Gaussians) offers a significant improvement over a *nested* model (i.e. one Gaussian, which is just a mixture of two Gaussians with the same mean) is the F-test. The F statistic is:

$$F = \frac{RSS_1 - RSS_2}{RSS_2/(n-2)}$$

which has the F-distribution with $(1, n-2)$ degrees of freedom if $\mathbf{x}_i \sim N(\mu_i, \Sigma)$ and μ_i is estimated by least squares.[6] In the case of the Gaussian mixtures, μ_i is indeed estimated by least squares to be either $\bar{x}_1$ or $\bar{x}_2$, but in order for the statistic to follow the F-distribution, the cluster memberships must be known in advance (i.e. we must know whether $\hat{\mu}_i = \bar{x}_1$ or $\bar{x}_2$). In reality, we estimate the cluster memberships from the data itself, so the statistic does not follow an F-distribution. The literature on the exact and asymptotic distribution is voluminous.[3]

For our purposes, however, a heuristic will suffice. We calculate a psuedo p-value from the F-distribution and compare it with $\alpha/$(# of regions). In our experiments we have taken $\alpha = .05$. The number of regions in the above formula can be thought of as a Bonferroni-type correction which adjusts for the number of tests we conduct at each iteration (we test whether each of the regions needs to be split). The effect of this correction is to gradually relax the criteria for significance as the number of regions decreases, thereby increasing the number of splits during the later stages.

Once we decide to split a region, we scan the image for the smallest region (in terms of number of pixels), delete this region, and set the index of the newly created region to the index of the deleted region. Thus, the splitting step does not change the total number of regions. This rule of thumb has the added benefit of pruning the segmentation of any small, isolated segments that might appear.

3.3. Summary

Our algorithm interweaves the two processes of merging and splitting. The algorithm is summarized in Table 1.

while $nRegionsLeft > k$ **do**
  {Merging}
  $(i, j) \leftarrow \arg\min_{i,j}(t_{ij})$
  merge regions i and j
  delete t_{jk}, t_{kj} for all k and update t_{ik}, t_{ki} for all i
  {Splitting}
  if $nRegionsLeft \mod 5 == 0$ **then**
    for each region **do**
      Cluster (x, y, r, g, b) in region into two clusters
      $F \leftarrow \frac{RSS_1 - RSS_2}{RSS_2/(n-2)}$
      if $1 - fcdf(F, 1, n-2) > \alpha/nRegionsLeft$ **then**
        find smallest region r in image
        merge r with closest region
        assign label r to cluster 2
      end if
    end for
  end if
end while

Table 1. Merging and splitting algorithm

Although we apply the merging and splitting algorithm to a watershed segmentation, our algorithm will work just as well for any preliminary oversegmentation. If an image is undersegmented, then this algorithm can be adapted by reversing the merging and splitting steps.

4. Results

As described in Arbelàez et. al. [1], contour detection and image segmentation are two similar, but different problems, each with their own set of evaluation metrics. The former involves specifying the salient contours of an image and does not guarantee closed regions, while the latter explicity partitions an image into non-overlapping labeled regions.

Our project falls into the latter category. Although we can evaluate a proposed segmentation visually based on perception, it is also desirable to come up with quantitative measures of evaluation.

4.1. GCE and LCE

GCE and LCE, introduced in Martin et. al.[8], are two metrics for quantifying how closely a generated segmentation S_1 matches a ground truth segmentation S_2. Intuitively, they measure how consistent overlapping regions are between S_1 and S_2.

We calculate local refinement error for each pixel:

$$E(S_1, S_2, p_i) = \frac{|R(S_1, p_i) \backslash R(S_2, p_i)|}{|R(S_1, p_i)|}$$

then compute two quantities, GCE (Global Consistency Er-

ror) and LCE (Local Consistency Error).

$$GCE(S_1, S_2) = \frac{1}{N} \min \left\{ \sum_i E(S_1, S_2, p_i), \sum_i E(S_2, S_1, p_i) \right\}$$

$$LCE(S_1, S_2) = \frac{1}{N} \sum_i \min \left\{ E(S_1, S_2, p_i), E(S_2, S_1, p_i) \right\}$$

where N is the number of pixels in the image.

Notice that the LCE score is necessarily at least as good as the GCE score, as LCE allows for refinement in both directions. In addition, they are only meaningful when the number of segments for S_1 and S_2 are approximately the same.

For this project, we take LCE as the descriptive feature.

4.2. Segmentation Covering

Segmentation Covering is another performance metric used to evaluate segmentations [1] [7]. As in GCE and LCE, we are comparing two segmentations of an image, S_1 and S_2

Define the *overlap* between two arbitrary regions $R_1 \in S_1$ and $R_2 \in S_2$ to be:

$$\mathcal{O}(R_1, R_2) = \frac{|R_1 \cap R_2|}{|R_1 \cup R_2|}$$

Then the *Covering* of S_1 by S_2 is given by:

$$\mathcal{C}(S_2 \to S_1) = \frac{1}{N} \sum_{R_1 \in S_1} |R_1| \cdot \max_{R_2 \in S_2} \mathcal{O}(R_1, R_2)$$

where N is the number of pixels in the image.

Thus when comparing a machine based segmentation M with a ground truth human segmentation H, we can return two metrics: $\mathcal{C}(M \to H)$ and $\mathcal{C}(H \to M)$. For this project we take as the descriptive feature the better of the two, i.e.

$$\mathcal{C} = \max \left\{ \mathcal{C}(M \to H), \mathcal{C}(H \to M) \right\}$$

4.3. Evaluation Scheme

We make use of images from the Berkeley Segmentation Data Set [1]. The Data Set consists of 500 images, separated into training (200 images), validation (100 images), and test (200 images) sets. As we do not do any training these distinctions are irrelevant; however, we made the arbitrary decision of running our algorithm and benchmark code on the training and validation sets.

Each image in the Data Set has associated with it a set H of "ground truth" human generated segmentations. We chose to compare our algorithm against the human generated segmentation with the greatest number of segments. In another words, we specify the *nSeg* input parameter to be $\max_i |H_i|$.

After all 300 images were processed, we used the benchmarks described in sections 4.1 and 4.2 for quantitative evaluation. We also ran the standard NCuts algorithm [10] on the same set of images for comparison.

	GCE	LCE	Covering
NCuts	0.289	0.216	0.336
	(0.108)	(0.096)	(0.096)
diff of means	0.177	0.113	0.504
	(0.120)	(0.082)	(0.175)
t test	0.3003	0.198	0.421
	(0.119)	(0.090)	(0.119)

Table 2. Summary of scores obtained by the different segmentation methods on the 300 images. Format: mean score (standard deviation)

4.4. Results and Examples

A summary of the results is shown in Table 2; we provide the mean score as well as the standard deviation in parentheses. As a reminder, we desire a lower LCE score and a higher Covering score.

We noticed that Ncuts tended to yield smooth, "blocky" regions, especially in Figures 7, 8, and 9, due to its tendency to "need" to make cuts, even when unnecessary. We see this most clearly in Figures 8 and 10.

The images do support the relatively high scores that the difference of means achieves. It is the only method that is able to pick out the cow in Figure 7 and the wolf in 9, and is at the same time very good at differentiating foreground from background, as seen in Figures 8 and 10; the skies in both images are clearly one segment. This makes sense: by only taking into account the mean intensity (and color) of a region, it is that much more willing to merge regions that are closer in RGB space, and that much more unwilling to merge regions that are further away, with no regard to the sizes of the regions.

However, difference of means did also fail egregiously for some images, i.e. Figures 11 and 12. We surmise that the eagle's wings in Figure 11 were not different enough from the sky, causing a merge. It is more clear in Figure 12; the woman's white dress and the man's shirt is too close in color space to the white gates in the background, two unncessary merges.

The t statistic metric does particularly well on images like Figure 12, with a high amount of detail and color, as it normalizes out the intensity by the size of the regions. Thus it is reluctant to merge big regions together. This is desirable for some images and undesirable for others; like Ncuts

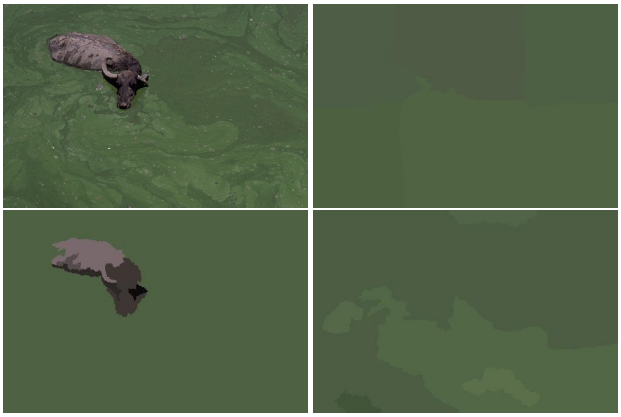

Figure 7. From left to right, top to bottom. Original image, Ncuts, Merging with mean intensity, Merging with t test

Figure 8. From left to right, top to bottom. Original image, Ncuts, Merging with mean intensity, Merging with t test

Figure 9. From left to right, top to bottom. Original image, Ncuts, Merging with mean intensity, Merging with t test

it has difficulty seeing sky and water as one contiguous region.

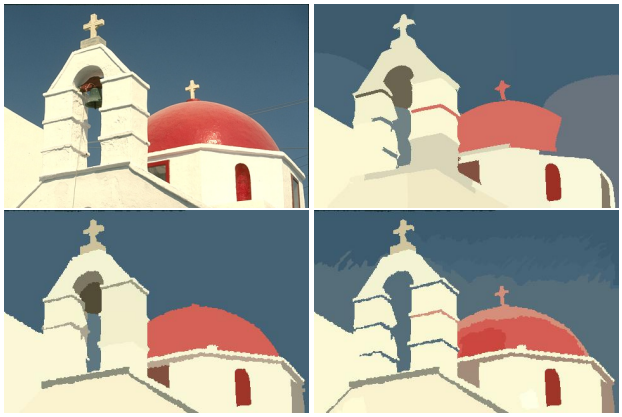

Figure 10. From left to right, top to bottom. Original image, Ncuts, Merging with mean intensity, Merging with t test

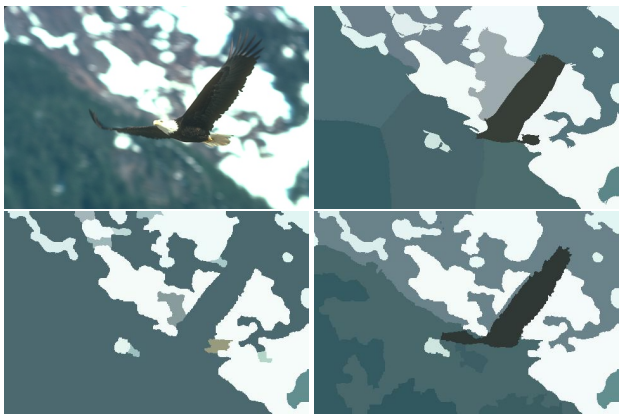

Figure 11. From left to right, top to bottom. Original image, Ncuts, Merging with mean intensity, Merging with t test

Figure 12. From left to right, top to bottom. Original image, Ncuts, Merging with mean intensity, Merging with t test

4.5. Performance

We also collected running time information on the different algorithms. The timing data shown in Table 3 is for total processing time: including running the algorithm and benchmark code.

	Time (secs)
Ncuts	139.7
diff of means	132.8
t test	140.6

Table 3. Algorithm Runtimes

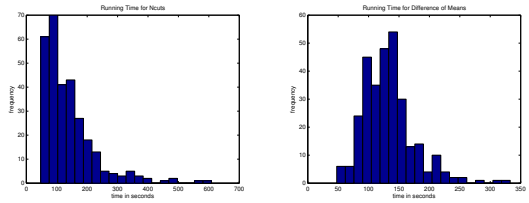

Figure 13. Running Time for Ncuts and Difference of Means

In particular, it was interesting to note the distribution of times for running the Ncuts algorithm. While the times for difference of means and t test metrics followed a normal distribution, the Ncuts distribution is heavily skewed to the right, with a handful of images taking well over 8 minutes. This happens as the Ncuts algorithm involves calculating eigenvalues of matrices the size of the image. This does not scale as well as our algorithm which involves only simple vector manipulations.

5. Further Directions and Conclusion

In summary, this paper makes two contributions to the problem of image segmentation. First, we have demonstrated that total variation denoising can be useful as a nonlinear filter that smooths an image while preserving edges. Second, we have introduced a method for reducing the oversegmentation problem of the watershed algorithm, by successively merging and splitting regions based on hypothesis tests. We have shown that this approach in its naivest form can achieve comparable performance to many established segmentation algorithms, such as normalized cuts.

There is much scope, both theoretical and experimental, to extend this algorithm further. In terms of theory, we can look at more nuanced versions of the hypothesis tests that we used in our implementation. As mentioned in Section 3, Hotelling's T-squared statistic could be used to account for correlations between channels, and a more formal test could be conducted to test for the number of clusters in the Gaussian mixture model. Perhaps k-means may not be optimal for determining splitting at all. Our preliminary results using normalized cuts to determine splitting were not promising, although this may be because we did not have a principled way of deciding whether or not to accept a split.

Also, we performed all our analysis in an unsupervised setting. A few parameters were chosen by hand and tested. This can easily be extended to a supervised setting, whereby parameters are chosen by cross-validation. There are many parameters to tune: the smoothing parameter and the tolerance for total variation denoising, the threshold for deciding a split, number of neighbors to consider (we used 8-connected neighborhoods), etc. The parameters that we chose as a result of our experimentation are almost certainly suboptimal; we could gain performance simply by choosing parameters that are ideal for image processing by cross-validation. To take this one step further, we could also determine different optimal parameters for different *types* of images by correlating image features with performance given certain parameters.

Finally, we processed most of our images in RGB color space. Our preliminary experiments with HSV and L*a*b* color space were disappointing, but ultimately, working in RGB space makes our algorithm sensitive to changes in illumination. It is important to continue to explore alternative color spaces that more closely model human perception.

References

[1] P. Arbelà andez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(5):898 –916, May 2011.

[2] T. F. Chan, S. Osher, and J. Shen. The digital tv filter and nonlinear denoising. *IEEE Trans. Image Process*, 10:231–241, 2001.

[3] J. Chen and P. Li. Hypothesis test for normal mixture models: The em approach. *Vol.*, (arXiv:0908.3428. IMS-AOS-AOS651), Aug 2009.

[4] T. Goldstein and S. Osher. The split bregman method for l1-regularized problems. *SIAM J. Img. Sci.*, 2:323–343, April 2009.

[5] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using MATLAB 2nd Ed.".* Prentice Hall Press, Upper Saddle River, NJ, USA, 2009.

[6] E. L. Lehmann and J. P. Romano. *Testing statistical hypotheses*. Springer Texts in Statistics. Springer, New York, third edition, 2005.

[7] M. R. Maire. *Contour Detection and Image Segmentation*. PhD thesis, EECS Department, University of California, Berkeley, Sep 2009.

[8] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *Proc. 8th Int'l Conf. Computer Vision*, volume 2, pages 416–423, July 2001.

[9] L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60(3):259–268, 1992.

[10] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22:888–905, 1997.

[11] R. Szeliski. Computer vision : Algorithms and applications. *Computer*, 5:832, 2010.