1 Introduction

In computer vision, “Image Segmentation” is a process of partitioning an image into several pixel groups. It is typically used to locate objects and boundaries in an image. Particularly, “Binary Separation” - how to partition an image into two segments: “foreground” and “background” is of special interest.

Earlier techniques are either based on boundary features [Kass et al., 1988] or manual sketches [Boykov and Lea, 2006]. In this project, we propose a new approach to partition “foreground” and “background” making use of distance data of real objects in the image. To be specific, when an image is taken by camera, there exists a one-to-one mapping from pixels in the image to real objects. For each image to be processed, we have additional data telling the distances between the camera and the real objects corresponding to some pixels in the image.

It is challenging to use the distance data. Although the distance data is helpful, yet it is not complete. A typical distance data set only covers approximately 10% pixels. It is necessary to develop techniques that infer the states of other pixels based on this sparse information. Besides, the data could contain some errors. For example, some pixel may be mapped to a wrong real object resulting in a wrong distance number. Therefore, the algorithm should include procedures that help fix the inherent errors of data.

We plan to use the distance data to first generate the initial states of pixels, i.e, whether it belongs to “foreground” or not. Based on the initial states, we’ll iteratively refine the segmentation following the modified Expectation Maximization process. To do the refinement, we’ll use Markov Random Fields to calculate the log-likelihood of a pixel, taking the information of the 4 pixels around it into account.

We believe our approach is more accurate than previous work. The additional distance data is a good indicator about which group a pixel belongs to, thus giving us confidence in the accuracy of segmentation, compared to earlier techniques which only use the image information alone. Moreover, our approach is flexible in that it does not need manual annotation. Finally, the segmentation decision is not likely to be misled by errors in distance data, since it will do iterative refinement based on the pixel’s local neighbor information.

2 Related Work

As a common problem in computer vision, segmentation has been well studied in the past. There is a large literature on segmentation and clustering, among which two main lines of methods are proposed previously, feature space clustering (e.g. [Comaniciu and Meer, 1997], [Comaniciu and Meer, 1999]) and graph-based approach (e.g. [Shi and Malik, 1997], [Wu and Leahy, 1993]). Besides these methods, which only relies on the image information, researchers have proposed new segmentation algorithm that use rough manual labelling to do more accurate segmentation. We will introduce these three methods briefly.
In feature space clustering, local features, such as low pass filter response, SIFT [Lowe, 2004], etc., are firstly extracted from the given image. For color images, color information is often included. Then those features are concatenated to form a vector as a descriptor for each pixel. In order to segment the image, we might seek a clustering of feature vectors observed in the image. A compact region of image with distinct feature would be expected to have a corresponding high density area in the feature space [Boykov and Lea, 2006].

Another branch of image segmentation algorithms is graph-based method. Instead of consider each pixel independently, we form a graph with each vertex representing an individual pixel and assign weights to edges according to some similarity measure. Then, the segments are determined by graph cuts.

A recent new approach by Boykov and Lea [2006] relies on manual sketches on the original image. It requires a user to draw strokes of different color in the foreground and background area of the image. Then, the algorithm solves segmentation problem based on this extra knowledge.

3 Data

We got our data from Google street view team and preprocessed it for use in our project. The data was collected by a car equipped with 8 cameras and 3 laser scanners. Each of the laser scanner can scan a 180 degree 2D plane at each time. Two of the laser scanners scanned vertically and the third one scanned horizontally. As the car moved, the car positions were recorded by global positioning system. Those positions were adjusted by the scans of the horizontal laser scanner using SLAM (Simultaneous localization and mapping) techniques to get the best precision. The 3D position of each laser scan point can be estimated based on the relative positions of the car and laser scanners and therefore a 3D point cloud can be built from the scans of the vertical laser scanners. At the same time, the eight cameras were taking pictures as the car moved. Although the cameras were well calibrated, the images were taken by rolling shutters and there could exist errors in terms of image projection model if we assume each image were taken in a pinhole model.

Given the 3D point cloud and images, we project the points onto the images with the estimated point positions, camera poses and projection matrices. Then we can get images like Figure 1.

![Figure 1: An image with 3D points projected into it.](image)

In Figure 1, the black dots are the projections of the 3D points on this image. As we can see, due to all kinds of possible errors mentioned above, the 3D projections and the images are not well
aligned and we can’t segment the images directly based on the depth of the 3D scan points. In this project, we want to find the contour between the background (the sky area) and foreground (all the objects appears in the image, including roads and buildings). The models and computing issue will be discussed in the methods section.

4 Methods

Based on the previous work, we merge several ideas in our project. First, to classify a pixel, we can first train a Gaussian mixture model to summarize the variation of pixels in the background and foreground separately. In another way to see this, we are getting an approximate distance from a certain pixel value to the cluster it belongs to and therefore we can get a cost to assign a certain pixel to the foreground or background. On the other hand, in the Gaussian mixture model, we are assigning an latent variable to each pixel. Due to the isolation between adjacent pixels in the Gaussian model, it is easy to imagine that the output of the binary classification will be very noisy in the sense that the adjacent pixels will have different labels although they have similar colors and belong to the same object visually. To solve this problem, we use MRF model to constraint the relation between the labels of adjacent pixels. Also, we use CRF model to adapt this relation to the difference of the pixel colors and therefore the model becomes more flexible. Also, we explore various ways to utilize our laser data. In the following subsections, we will discuss parts of our model and elaborate how we use the scan data.

4.1 Gaussian Mixture Model

The foreground and background shall have different color characteristics. We use Gaussian Mixture Model to summarize the colors of background and foreground. We train the two models separately, which means given a estimate of foreground and background pixels, we don’t change the label of the pixels and only aim to find a more accurate model to describe the given labels. We use standard EM algorithm to train the models.

4.2 Conditional Random Field

To solve the discontinuity problem in only using distance to the closest Gaussian mixture model as a cost, as what has been down in the literature, we add a connection cost between adjacent pixel labels to encourage identical labeling. Our proposed model is

\[ E(c, d, k, \theta, z) = U(c, k, \theta, z) + V(c, d, z) \] (1)

\[ U(c, k, \theta, z) = \sum_{i=0}^{i<N} D(c_i, k_i, \theta_i, z_i) \] (2)

\[ D(c_i, k_i, \theta_i, z_i) = -\log \pi_{k_i}^{z_i} + (c_i - \mu_{k_i}^{z_i})^T (\Sigma_{k_i}^{z_i})^{-1} (c_i - \mu_{k_i}^{z_i}) \] (3)

\[ V(c, d, z) = \gamma \sum_{(i,j) \in N} [z_i \neq z_j] e^{-\beta d_i \|c_i - c_j\|^2} \] (4)

\( c \) is the set of pixel colors. \( d \) is how likely a given pixel is on the boundary from scan. It will be discussed further later in this section. \( k \) is assigned GMM cluster for each pixel. \( z \) is latent variables labeling foreground and background where \( z = 0 \) means foreground and \( z = 1 \) background. \( \pi_{k}^{z} \) is mixture weighting coefficients. \( \mu_{k}^{z} \) is the mean of the \( k \)th Gaussian component. \( \Sigma_{k}^{z} \) is the covariance matrix of the \( k \)th Gaussian component. \( \gamma, \beta \) are parameters to control the connection cost. \( \theta \) is the set of parameters of the Gaussian mixture model and therefore \( \theta = \{ \pi, \mu, \Sigma \} \). \( N \) is set of all the neighbor pixels and we don’t use diagonal neighbor in the experiments. \( N \) is the number of pixels in the image. \( E \) is the energy function representing the cost function we are going to minimize. It can be decomposed into two parts. \( U \) is the cost of unary term. As discussed in the last subsection, it is the Mahalanobis distance to the closest Gaussian cluster of its labeled Gaussian mixture model, adjusted by the mixture weighting coefficient of that cluster. Note that we didn’t use soft assigned
cluster in the model, as was done commonly in Gaussian mixture model. Rother et al. [2004] said that in their experiment, the soft assignment doesn’t improve the results greatly but will make the optimization harder. $V$ is the connection cost between adjacent pixels. The cost is adjusted by the additional information including the pixel color difference and the scan information. The idea is that if the color difference is large or the pixels are likely on the boundary of different segmentations, the inconsistent labeling cost will be small. Otherwise, if two adjacent pixels have very similar colors, they are more likely to have the same labeling. In our case, we consider the color image instead of gray image and so the different difference of pixel colors are measured by $L^2$ norm.

It can be shown [Geman and Geman, 1984, Li, 2001] that the energy function we defined to represent the CRF corresponds to a Gibbs distribution uniquely. So by minimizing the energy function, we are actually seeking the Maximum Likelihood solution to the Gibbs distribution. The graphical representation of our model in color is shown in Figure 2.

![Graphical Representation of Model](image)

Figure 2: The graphical representation of our model in color

It is worth pointing out that when $d$ is always 1. The model is called GrabCut as in [Rother et al., 2004]. When $\beta$ is 0, which means the inconsistent cost between adjacent pixels is constant, the model is called Ising model. We have done experiments on those models and the results are shown in Section 5. We call our model LaserCut in the experiment section.

### 4.3 Inference

As mentioned, we use EM algorithm to estimate the Gaussian mixtures in given the pixel labels. At the end of each iteration, we minimize the energy function (1) to update the pixel labels. There are several ways to infer the labels of pixels based on our model. The traditional way is Monte Carlo Markov Chain method [Bishop, 2006]. There are also some other inference methods proposed during the last several decades [Szeliski et al., 2008]. However, most of them are approximate methods and don’t guarantee an globally optimal solution. Boykov et al. [2001] and Boykov and Funka-Lea [2006] argues that for the binary segmentation, if the energy function has the submodular property, the optimal solution can be got by minimum graph cut. Fortunately, when we constructing our energy function, we make sure that the submodular property is satisfied. Therefore, we solve Maximum flow, the dual problem of minimum cut, to minimize the energy function and update the pixel labels at the end of each EM iteration.
4.4 Use of Scan Data

As you can see in Figure 1, the scan points are not scattered in the image evenly in the foreground and therefore it is hard to extract some robust statistics to describe the foreground or the boundary. One idea is using bins around a pixel to describe its feature. To be specific, given the position of a pixel, we consider all the scan points within certain radius. The circle is then divided in angular direction to get the bins. The feature value of the pixel is how many pizza shape bins contain scan points. Although it is a nice way to describe the scan point description around a pixel, it is hard to find efficient way to calculate it for every pixel.

We later realized that using bins is equivalent to downsampling the image. What’s more, the region of most interest is the boundary between the foreground and background. Therefore, we propose a new methods to extract the scan information from the scan points. First, we downsample the image and then apply Sobel filter [Szeliski, 2010] to extract the edge-like features. The result image is resized back to the original image size. As each pixel now has a estimate how likely it is on the boundary, we use this information in our model, represented by $d$ in Equation (1). The result of this process is shown in Figure 3(a).

To initialize the labels, we do a distance transform [Borgefors, 1986, Felzenszwalb and Huttenlocher, 2004a] on the scan point image, which is a black image with white points showing the projection of the scan points. The result of distance transform is shown in Figure 3(b). Based on certain threshold, we can get a region estimated as background. Since this region will be refined in the algorithm, the threshold used doesn’t matter much.

Figure 3: The use of the scan points. (a) Boundary of the Scan Points. (b) Distance Transform of Scan Point image.

5 Evaluation

In general, evaluating segmentation result is hard due to the absence of precise quantitative measurement without ground truth. In addition, there is another important issue centers around the use of energy to compare energy minimization algorithms. The goal in computer vision is not to compute the lowest energy but the most accurate one, besides computing the global minimum was shown to be NP-complete in general [Boykov et al., 2001]. In the paper by Szeliski et al. [2008], a quantitative comparison between lowest energy achieved by different energy minimization algorithms and
the energy calculated from ground truth provides experimental proof for this argument. Although
one can argue that we could manually draw the boundaries for each image we used in this project,
due to high resolution (1936 x 2592) and limited size of sample labeling from different users we can
account for given the time, the accuracy and unbiasedness of the ground truth cannot be guaranteed.

As an alternative, we decided to conduct a comparison study among the GrabCut model, LaserCut
model and Ising Model, which is a widely used MRF model in image segmentation.

We first evaluate GrabCut. From Figure 4(a), we can see that cars and windows sharing the similar
color as the sky would be segmented as the background. The reason for this is that the distance
information is not used in the later refinement. As mentioned in the previous section, we only use
the distance transform image derived from laser point projections to initialize MRF, and never use
the distance information in the following refinement. It leads to that every edges across the image
are considered identical, so the color feature dominates the later iteration.

<table>
<thead>
<tr>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
</table>

![Figure 4: Segmentation result with (a) GrabCut and (b) LaserCut.](image)

Compared to GrabCut, LaserCut produces more satisfactory result. In Figure 4(b), white cars
in the front are labelled as foreground. This is because the color difference terms are no longer
uniformly weighted in the energy function in LaserCut, but may vary according to the distance data.
To be concrete, we assume that foreground objects got scanned more often than background and the
natural boundaries would occur at the transition between dense and sparse areas of projected LiDAR
points. Based on this assumption, we use simple edge detection algorithm (i.e. Sobel operator) to
extract possible boundaries between foreground and background. Then we assign higher weights to
the edges on the possible boundaries, lower weights to the other regions. In this way, for regions
within the potential boundaries, the influence of color feature will be undermined and certain color
difference is tolerated in the model.

To further evaluate our model, we use Ising model to process the same image. Ising model is a widely
used MRF model, in which color difference between neighboring pixels is completely ignored by
assigning the same cost for connectivity to all edges in the MRF graph. The result is shown in
Figure 5. We observe that Ising model is not as accurate as LaserCut, since more sky pixels are not
labeled as background. In Ising model, in order to avoid labelling white cars as “background”, we
have to tune threshold in distance transformation to relatively high, which means more pixels are
considered “close” to the foreground. As we only used Gaussian mixture model to measure color
feature, GMM intends to label part of the white sky as the foreground, since there are white cars in
the foreground.
We think LaserCut performs best among the three models. Compared to GrabCut model, LaserCut successfully classified the cars with similar color as the background as part of the foreground. This observation suggests the effectiveness of adaptively using extra information acquired from LiDAR scan. And compared to Ising model, LaserCut could provide more complicated and accurate boundary between foreground and background, which indicates the effectiveness of separation on color differences when close to the boundary.

However, the result of LaserCut is not perfect. For instance, some leaves of the tree is mislabelled in Figure 4(b). In fact, it is generally hard to determine the exact boundary of such complicated foreground object, however high resolution the image is. Moreover, given such an image, the true pixel boundary is rather subjective, and different people may have different opinions.

### 6 Lessons Learned and Future Work

Through careful study on the resulting segmentation of our approach, we learnt several things about the algorithm. For one thing, color feature helps to produce a good approximation to the true boundary between foreground and background. In Figure 4(b), the curve of “background” is very close to the truth boundary between green leaves and sky. From the distance data, we can see there are not many “leaves” pixels with distance, therefore we conjecture that the color feature plays an important role in delimiting the boundary.

For the other thing, the algorithm achieves balance between the influence of distance data, which is used to generate initial states, and the influence of color features, which is used to refine the segmentation. For example, in Figure 4(b), although the white cars have similar color to that of sky, they are not considered as background. Another example is that some sky pixels in the image are classified as background, despite the dataset suggests that they have finite distance to the camera. These two examples show that we are able to use color feature to fix minor errors in the distance data while still maintaining most of status generated by the distance data.

Our future work will focus on the following two aspects. First, the background pixels in our experiment images have very similar color. This fact greatly helps segmentation as we can put much weight on the color feature to infer the labels of pixels. In the future, we want to incorporate images
with more colorful background. To do segmentation on such images, we may have to introduce new factors to model background and foreground.

The other aspect is to do the general image segmentation. That is, segment the image into multi-layer. At present, we only use whether the pixel has a finite distance number or not. As we take the actual distance value into account, we can segment the images into several layers, with pixels in the same layers have close distance value.

7 More Results

We also did a comparison study on other images to further evaluate our algorithm as in Figure 6. As we argued before, our algorithm would usually produce a more complicated yet cleaner boundary between foreground and background over the other two.

![Figure 6: Comparison study on other images.](image)

References


