Semantic Segmentation of an Image Using Random Forest and Single Histogram Class Model

Mentor: Dr. Amitabha Mukarjee

Rohan Jingar
Mridul Verma
{rohanj,mridulv}@iitk.ac.in

April,2012

Abstract

This work explains and analyzes the implementation of Single Histogram Class Model based on textons. Here we explain the use of textons as a feature for image segmentation. We explore the method proposed by Schroff *et al* [3] for image segmentation. This report explains the method proposed and analyzes the effects of various parameters on the performance. Here we use MSRC data set to compare achieved performance to the original work. We are getting 67.7% accuracy in segmentation on MSRC data set.

Contents

1 Introduction

Object detection and Image classification in an image has been one of the most fascinating and highly applicative problems in AI. Many methods have been proposed mostly using the methods of machine learning by learning the properties of objects. This is a supervised learning problem as number of object classes is finite and predetermined. Humans are highly efficient when it comes to recognizing and classifying objects in an image than the current machines. Humans generally do not process much while doing such tasks as they focus only on the area of interest in the scene. This area is of great interest to many researchers working in the field of AI and image processing.
First of all we need to distinguish between Image Classification, Object detection, Image Segmentation.

Image Classification: Here the problem is to decide whether a specified object is present in the image or not. We may need to return the list of objects present in the image without their locations in the image.

Object Detection: Here we need to localize the objects in the image along with identifying them. To be more precise we may need to produce a bounding box around that object in the image.

Image segmentation: Here we not only detect the location of object but also the region occupied by that object in the image. To be more clear we need to do classification at pixel level i.e. we need to assign each pixel a class label indicating the object class that pixel belongs to.

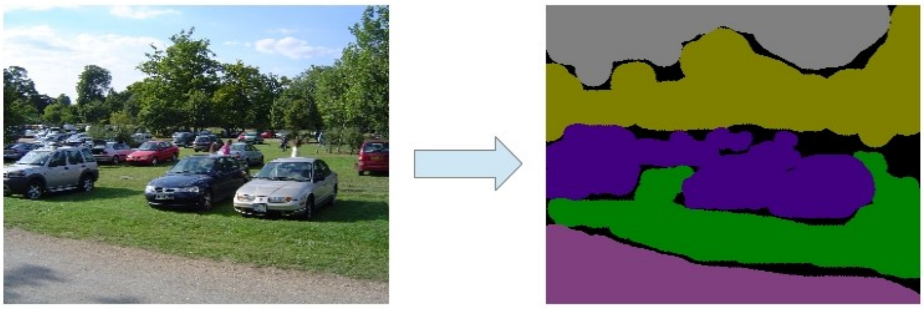


Figure 1: Segmenting the given image. Here the color value assigned to a pixel indicates its class label

It clear that the results for Image Classification and Object Detection can be easily derived from the segmented Image.

2 Data-Set

We used MSRC[1][1] dataset which is a standard dataset and is used in other works also. The dataset has labelled data for 21 object classes. The following object classes are included:
aeroplane, bicycle, bird, boat, body, book, building, car, cat, chair,cow, dog, face, flower, grass, road, sheep, sign, sky, tree, water.
Note: the black region in the groundtruth images between object classes or at boundaries

[1] Microsoft Research Cambridge

represent none of the classes. It represents NULL region and should be ignored while training. We have a total of 591 images out of which 276 are used for training, 256 for testing and 59 are validation images.

3 Random Forest

As from the name it is clear that this classifier contains many trees / decision trees. Random forest is a group of N independent Decision trees. By independent we mean they are trained independently. It will become more clear when we explain the training process of each individual decision tree.

3.1 Decision Tree

Decision trees are decision support tools which are tree-structured. These trees usually have a decision to be made at a node, the result of which decides to which node to move next (right child or left child).
Here we consider only binary decision tree. A general form for node test for such binary tree can be of the form:
First compute $t_p : D \to \mathbb{R}$ then decide as:

$$T = \begin{cases} t_p < \lambda : \text{ go to left child} \\ \text{otherwise go to right child} \end{cases}$$

Here D is the domain of features. In our case it is Texton, RGB, Hog, F17 filtered output.
The node tests will be explained in detail in the subsequent sections.

3.2 Training of N decision Trees

First Some terms should be clear before we proceed towards training:

Bagging: We inject randomness and independence into training by randomly sub-sampling the the training data for each tree. Given a Set of all the training data/points to be used for training of the Random Forest, We first randomly select a subset of it and we use these data points for training of a particular tree. Thus all the trees are independent from each other. It also improves the time required for training.
To further improve the time duration we inject independence at node level also by sub-sampling a set of node tests from a pool P of node tests.

Information gain ΔE: To select the best classifying node test at each node we use the method of information gain. From the set of node tests to be considered for a node we choose that node test as a decision which maximizes ΔE. In simple terms that node test will split the data points in the feature space into regions which mainly contains data points a single class.
Q is the set of data points sub sampled from the training set. a particular node test (t_p, λ) will split it into two disjoint sets Q_{left} and Q_{right}
We define entropy of Q as:
$E(Q) = \Sigma_i \; q_i ln(q_i)$ where $\overrightarrow{q}$ is the class histogram over Q and summation is over all the classes.

$\Delta E = E(Q) - \Sigma_i \frac{|Q_i|}{|Q|} E(Q_i)$ where $i \in$ {left, right}

Consider training of a particular decision tree:

1. We have a set of training data points T.

2. We sub-sample a set Q from T. The size of Q is given to us.

3. We take #nf node tests form pool P of all the node tests.

4. Out of these #nf node tests the one which maximizes ΔE is chosen as a decision for that node.

5. In the above step we have split the set Q into Q_{left} and Q_{right}. So for the left child of current node we use Q_{left} as training data and Q_{right} for right child.

6. And thus we iteratively proceed until: either a Q_i is empty or a threshold value for ΔE is achieved.

3.3 Creating a Pool P of node tests

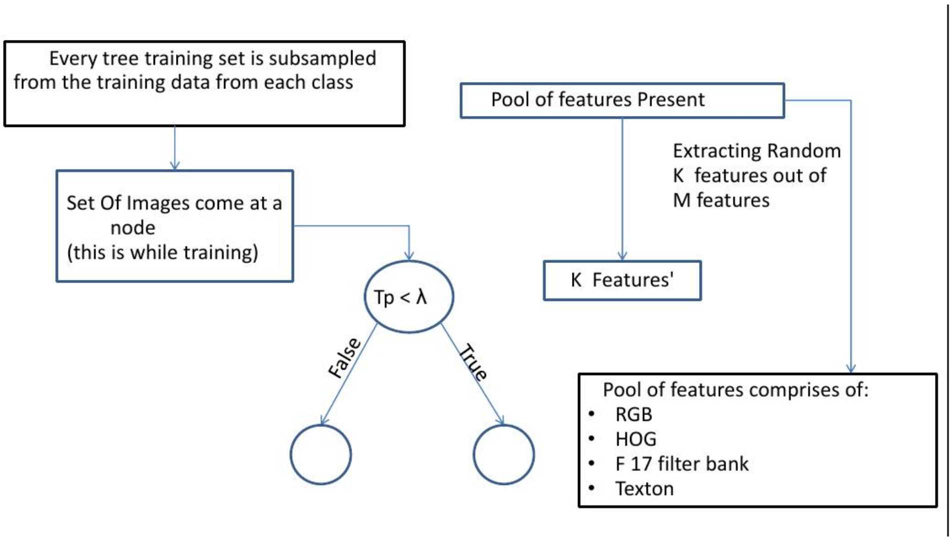

Every tree training set is subsampled from the training data from each class
Pool of features Present
Extracting Random K features out of M features
Set Of Images come at a node (this is while training)
Tp < λ
K Features'
False
True
Pool of features comprises of:
RGB
HOG
F 17 filter bank
Texton

Figure 2: Schematic diagram showing construction of node test

Here we will describe what type of node tests are used for creating an initial pool P of node tests. The following features are used for creating node tests:

- RGB
- F-17
- HOG
- texton

3.3.1 RGB

Node tests for RGB features comprises of the simple differences of the responses computed over one of the three channels / simple sum of the responses computed over one of the channels.

In simple terms there are two types of tests abstest and difftest, where the former denotes the response computed over a rectangle(from any channel) at some offset from a pixel and the later denotes difference of responses computed over two rectangles(from any two channel) at some offsets from a pixel.

3.3.2 F 17

Similar node tests are there for F 17 features. It is a set of 17 filters based on guassian, derivative of guassian and laplacian of guassian as defined in Winn *et al. (2005)* [4]. In this the image is transformed into 17 channels and the node tests comprises of the simple differences of the responses computed over the 17 channels / simple sum of the responses computed over one of the 17 channels.

3.3.3 HOG

Similar tests will be computed over the response of HOG on the given image.

3.3.4 Texton

Textons were first introduced and used in vision by Leung and Malik(1999). In simple language textons are used to represent visual features of an image. We can think of a texton as a pattern appearing in the image at various locations. Actually texton is defined as the smallest repeating feature in an image.

This procedure for computing k textons by using 3x3 window is as follows:

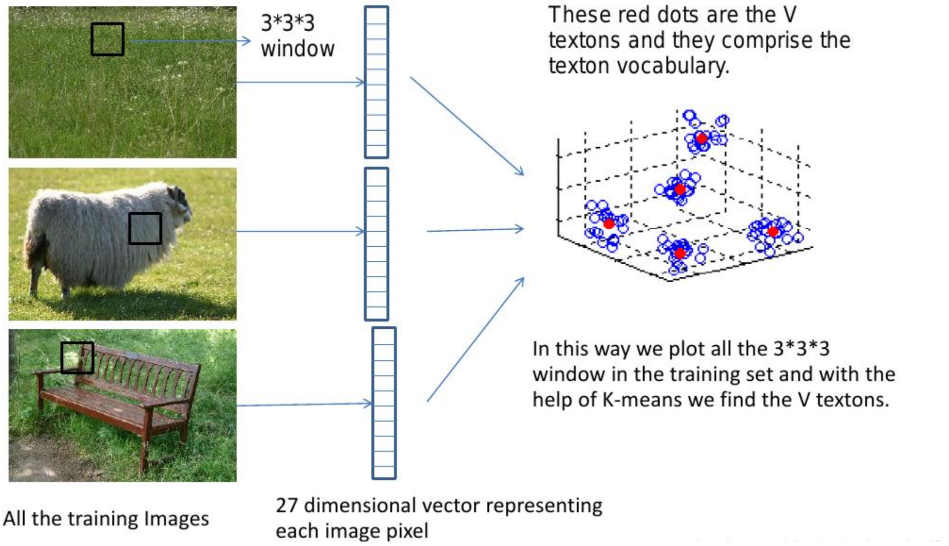

3*3*3
window
These red dots are the V
textons and they comprise the
texton vocabulary.
In this way we plot all the 3*3*3
window in the training set and with the
help of K-means we find the V textons.
All the training Images
27 dimensional vector representing
each image pixel

Figure 3: Schematic diagram showing the procedure to compute textons via k-means clustering

- First we take a 3x3 window around every pixel in every image in the training set.
- Then we plot all of these 3x3x3 (last 3 for the RGB channels) =27 dimensional vector in a 27 dimensional vector space.

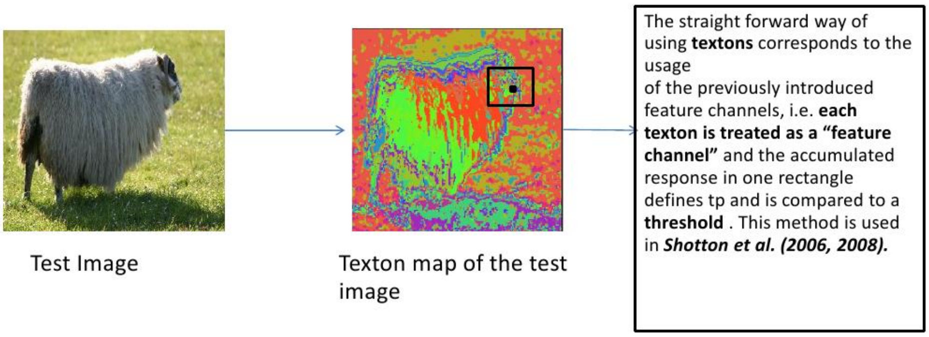
- Then we apply K-means to all these data points to find out the k cluster centers .K-means works as follows:
    - We first assign random k points as cluster centers.
    - Then we find for each point to which cluster center it is nearest.
    - In this way we get K clusters and then we find out the mean of these points in each cluster and assign these points as the new cluster centers.
    - This procedure goes on for a fixed no. of iterations before it converges.
- The K cluster centers that we get are the K textons/ which form our dictionary.

To get a visual representation of the texton map of an image the procedure is:

- For finding out the texton of each pixel, we take a window of size 3x3 around each pixel and then we get the 3x3x3 =27 dimensional vector and find out to which of the K cluster centers it is the closest and we color the pixel accordingly (we have assigned each of the texton a color map).

Node Test In case of textons we have two options:

1. We take a window around a pixel and count the number of pixels for each of the textons inside the window. We consider each texton as a separate channel and compute node test as in the case of RGB. A sample example is illustrated in fig-4.



The straight forward way of using **textons** corresponds to the usage of the previously introduced feature channels, i.e. **each texton is treated as a "feature channel"** and the accumulated response in one rectangle defines tp and is compared to a **threshold** . This method is used in ***Shotton et al. (2006, 2008).***

Figure 4: Schematic diagram for node test using textons

2. We can compute texton histograms and then use these histograms as a feature. This method is explained in detail in the next section.

4 Single Histohram Class Model

Once we have modelled the texton maps of training images, we can use these for training and classification.

We define **exemplars** as regions of the image belonging to a particular class. We can compute histograms over these exemplars. Instead of storing many such histograms of a class and use them in nearest neighbour method, we combine them to form a single histogram for each class. We call such histograms as Single-Histogram Class models SHCM. Schroff *et al.*[2] proposed the following method for computing SHCM.

4.1 Computing SHCM

1. Compute exemplar histograms q_c^i for or all the occurrences of an object class C.

2. Take average of all such q_c^i's and store it as Q_c

This Q_c is the SHCM for class C and we will use this histogram for the classification part instead of the many q_c^i's which would have been very time consuming.

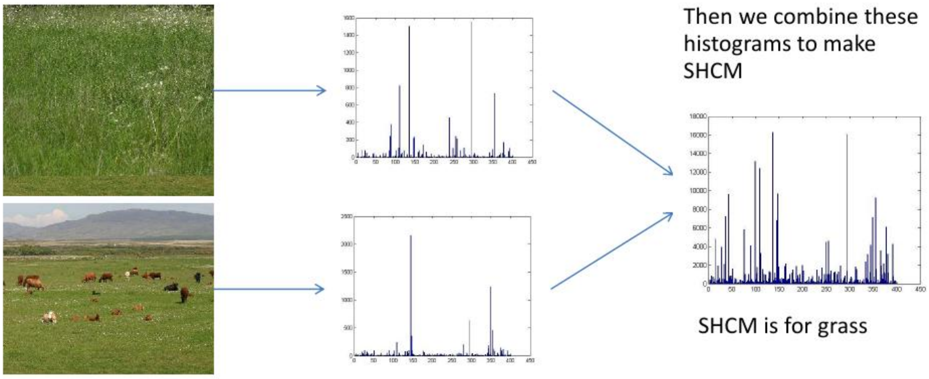


Figure 5: Schematic diagram showing the procedure to compute SHCM for class grass

4.2 KL Divergence

To compare two histograms the method used is KL Divergence. For two histograms h_1 and h_2 the KL Divergence is defined as :
$KL(h_1||h_2) = \Sigma\ h_{1_i}\ ln\ (\frac{h_{1_i}}{h_{2_i}})$ To find the class label of a test pixel p we first compute test histogram h over the context around that pixel and find the class c s.t.:
$c = arg\ min_c\ (KL(h||Q_c))$
and assign this class label to p.

5 Modifications in the code

The code released by Schroff is able to train the classifier based on basic node test in RGB, HOG, F-17 feature space. The code for computing textons, texton-map, and building SHCM for each class is not present in the released version. We have written the following files:-

1. compute_clusters.m

2. computeTextonMap.m

3. nearest.m

4. shcm.m for computing SHCM for a class

5. shcmgt.m for computing SHCM based on exemplar regions

6 To run the code released

Since the code is very memory intensive as it requires all the training data to be stored in memory, it may be needed to set the flag MALLOC_CHECK_ to 1 before starting the matlab. Otherwise matlab will stop working showing some warning. The code may not be able to run on a normal desktop with 4 GB of RAM. We ran the code on a dedicated server with sufficient amount of memory.

For 64 bit machine the standard libc.so.6 is not at the expected location. It may be needed to be linked at the expected location. In our case we had to do link /lib/libc.so.6 with /lib64.

```
$ export MALLOC_CHECK_=1
$ ln -s /lib/libc.so.6 /lib64
```

7 Analysis

7.1 Creating Texton-maps

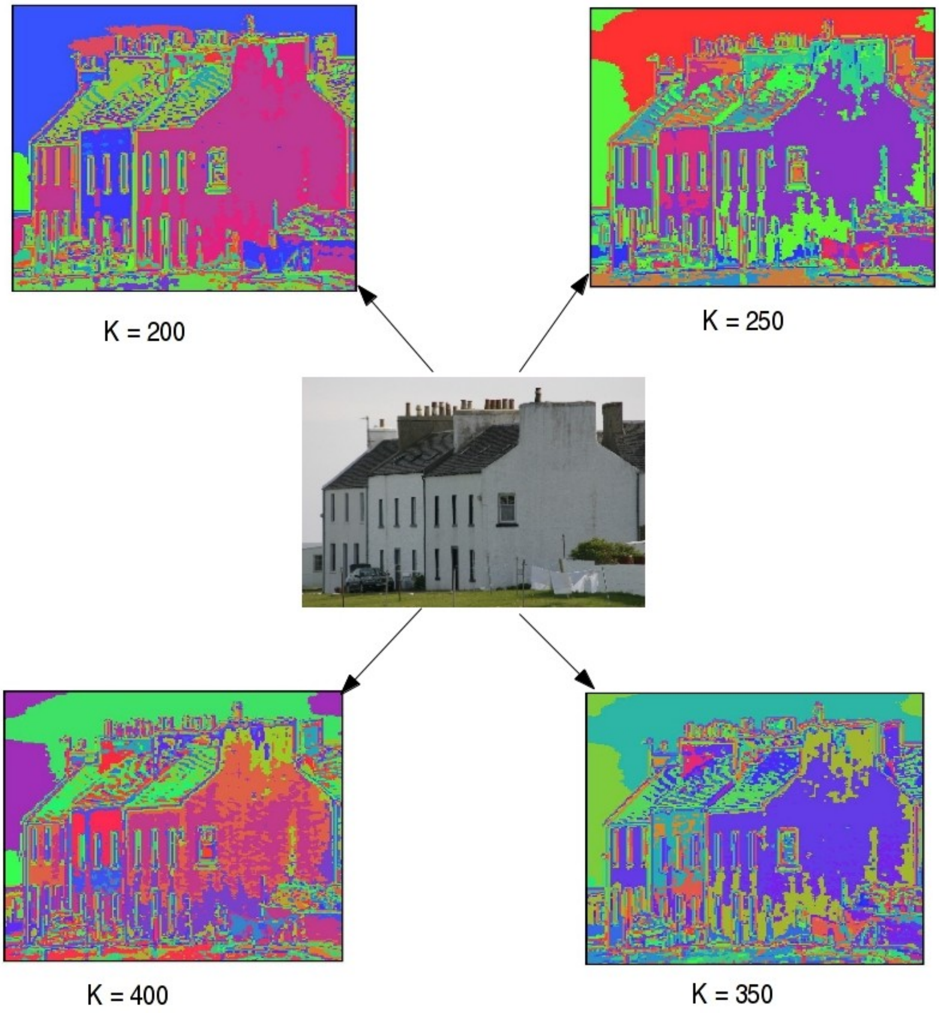


Figure 6: Showing the effects of value of k on the texton map

In fig.4 we have shown the texton maps of an image for various values of k. It can be clearly induced that as we increase the value of k we get a more dense set of visual features of an object.

7.1.1 Single Histogram Class Models:

We took initially 6 classes randomly. They were:

- Class 1: Grass
- Class 2:Tree
- Class 3: Buildings
- Class 5: Cow
- Class 9: Sheep
- Class 10: Flower

Now we found out the maximum of each class:

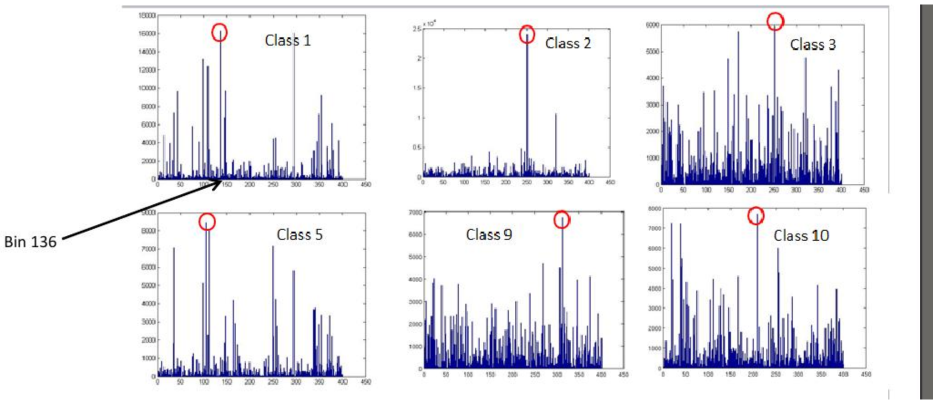


Figure 7: Finding maximum of various class histograms

Then we took the array of each of the class histogram and put them in an 2- D matrix and in this way we got an 6*400 array and then we took the standard deviation of the matrix column wise and we found out that the maximum standard deviation was at the bin 136 and observing the maximums we found out that the grass (class 2) maximum was also at bin 136.

The reason for this type of observation was that we were taking 6 different classes and each of these classes had its discriminating types of features. By this discriminating types of features, we mean there would be some texton which would be present in high amount in one of the classes but would be barely present in other classes, because it is the discriminating feature of that class from the other classes. This thing would cause the standard deviation to be high in these bins (textons). The discriminating texton was texton#136 for the grass class (class2).

We plotted the histogram models in two ways:

- case-1 In first way, we plotted whole of the image histogram of a class and then we combined all of these images of a class.
- case-2 In second way, we did not plot the whole image histogram instead we just plotted the histogram of the ROI(region belonging to that class) in the image.

We took classes 2 and 3, we observed their histogram models according to case 1 and find out that the maximum for both the classes was at bin#251. But when we took the histogram models according to case 2 the maximums were now at different bins.

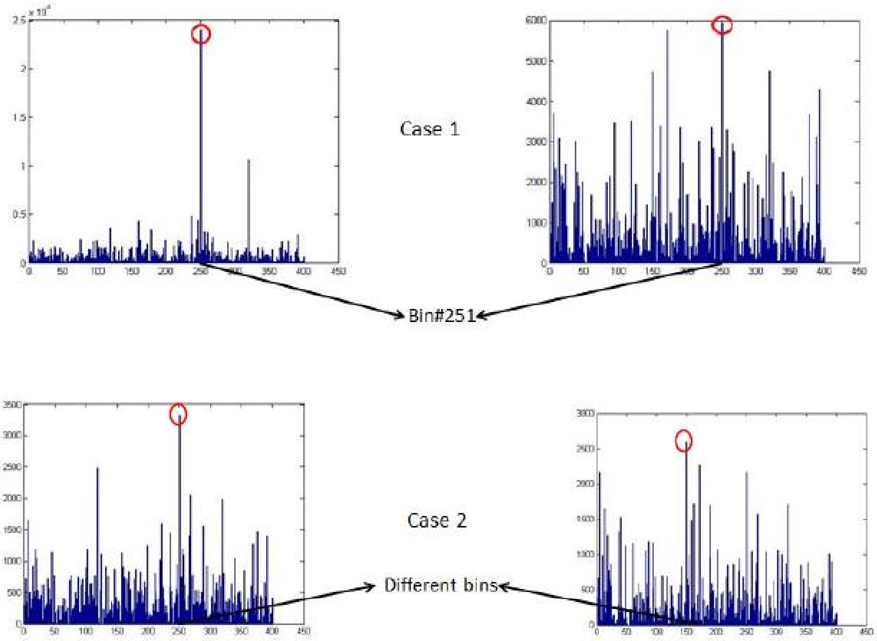


Figure 8: Finding maximum of various class histograms

The reason for such kind of observation can be that in case 1 we are plotting the histogram for whole of the image in which objects like tree and building can occur together. But when we plot according to case 2 ,we plot only that part of the image which is ROI (explicitly belonging to that class) in this the outliers cannot come and hence the maximum would not be the same.

8 Results and Performance

We did training and testing for various parameters of a Random Forest.

No. of Trees	Max. Depth	Pool size $\|P\|$	features/node (#nf)	Performance
10	10	30000	200	59.3 %
20	20	30000	200	63 %
20	20	30000	600	67.1 %
30	20	30000	600	67.7 %

Result images can be found in the appendix.

9 Acknowledgements

We are thankful to F. Schroff for making their code publicly available on their project page and for continuously helping and guiding us in right direction. We are also thankful to Alec's Web Log for providing a matlab script for generating random distinguishable colors. We are grateful to Dr. Amitabha Mukerjee for their valuable feedback and suggestions.

References

[1] A. Criminisi. Microsoft research cambridge object recognition image database. `http://research.microsoft.com/vision/cambridge/recognition/`.

[2] F. Schroff, A. Criminisi, and A. Zisserman. Single histogram class model for image segmentation. In *ICGVIP*, pages 3–5, 2006.

[3] F. Schroff, A. Criminisi, and A. Zisserman. Object class segmentation using random forests. In *Proceedings of the British Machine Vision Conference*, pages 1–8, 2008.

[4] J. Winn, A. Criminisi, and T. Minka. Object categorization by learned universal visual dictionary. In *Proceedings of the 10th International Conference on Computer Vision Beijing*, pages 2–3, 2005.

10 Appendix

The following are the result images for the parameters:

- Number of trees N = 30
- Max Depth of a tree = 20
- Total features per tree $|P| = 30000$
- no. of features drawn from pool #nf= 200
- Percentage pixels correctly classified = 67.67%

In each image below the left, middle and the right column represent original image, ground truth image and segmented image respectively.

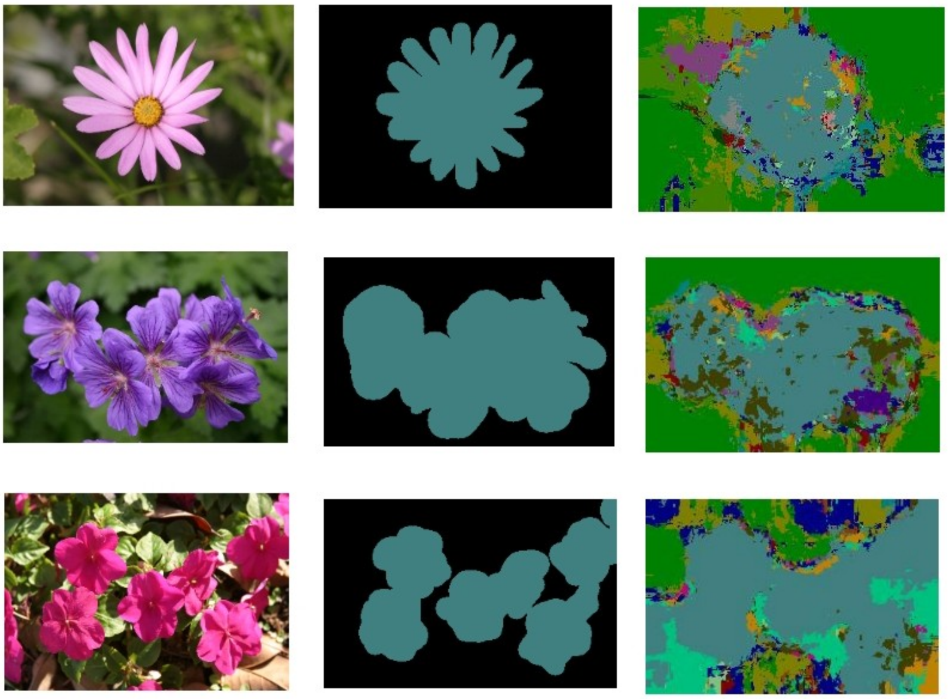


Figure 9: Result images showing segmentation of class flowers

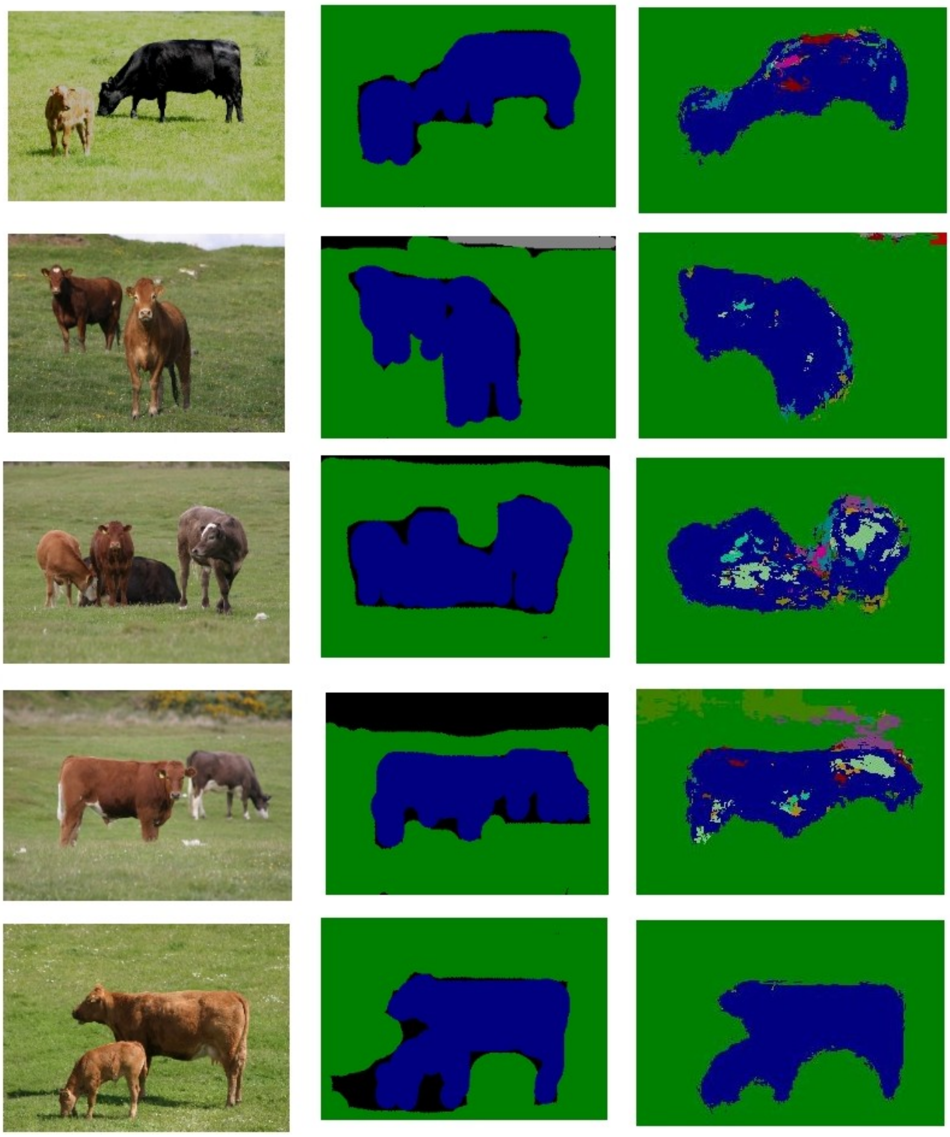


Figure 10: Result images showing segmentation of class grass and cow

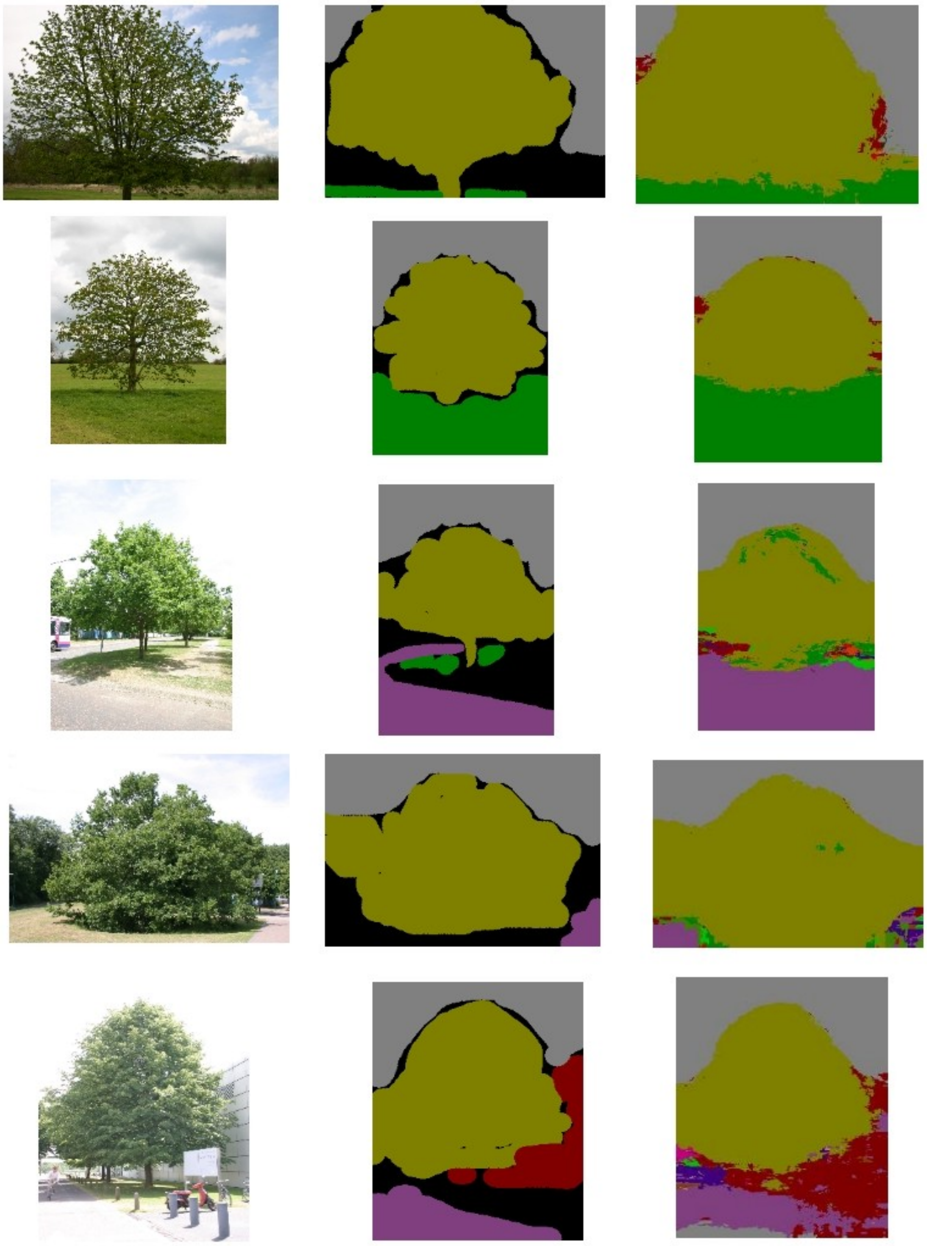


Figure 11: Result images showing segmentation of class tree,road and sky

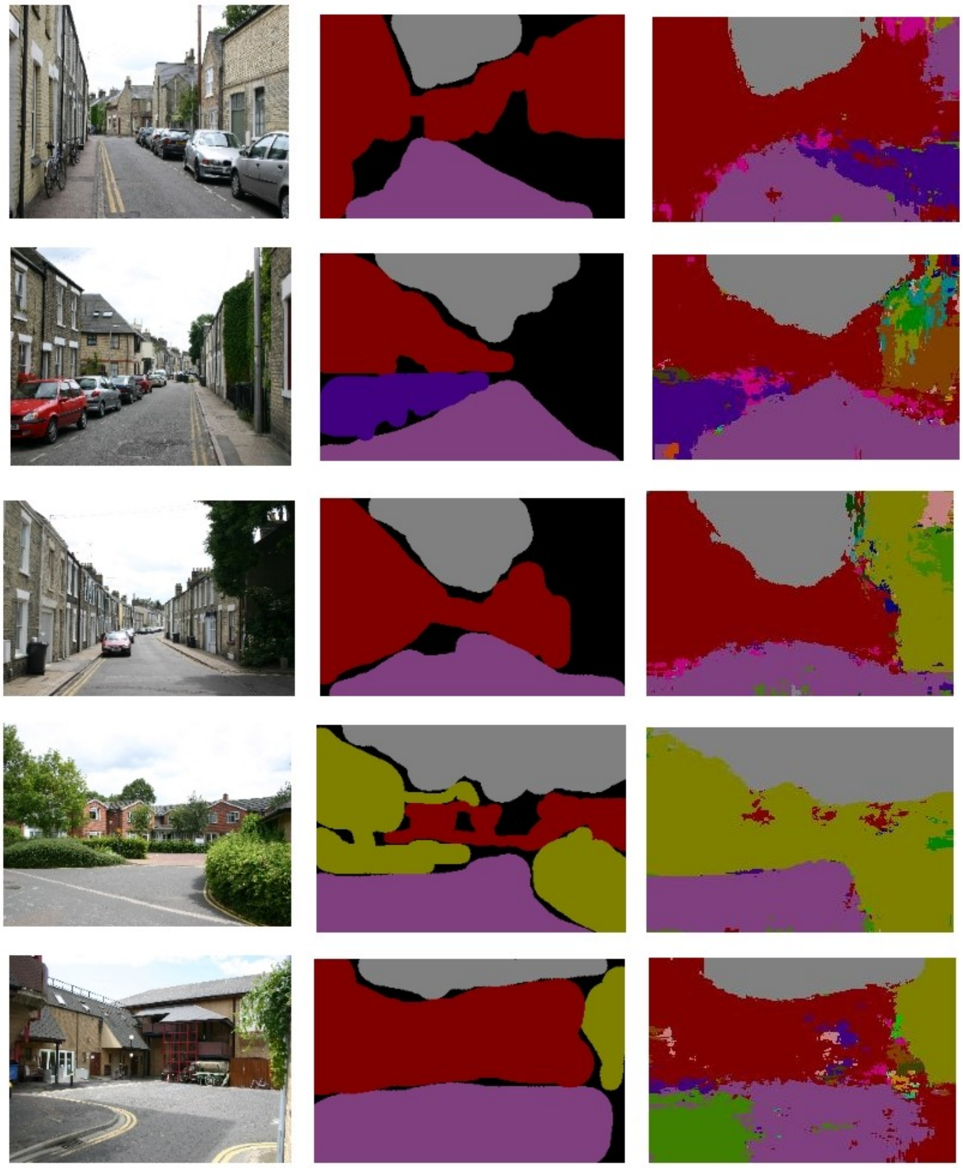


Figure 12: Result images showing segmentation of class building,car,road and sky

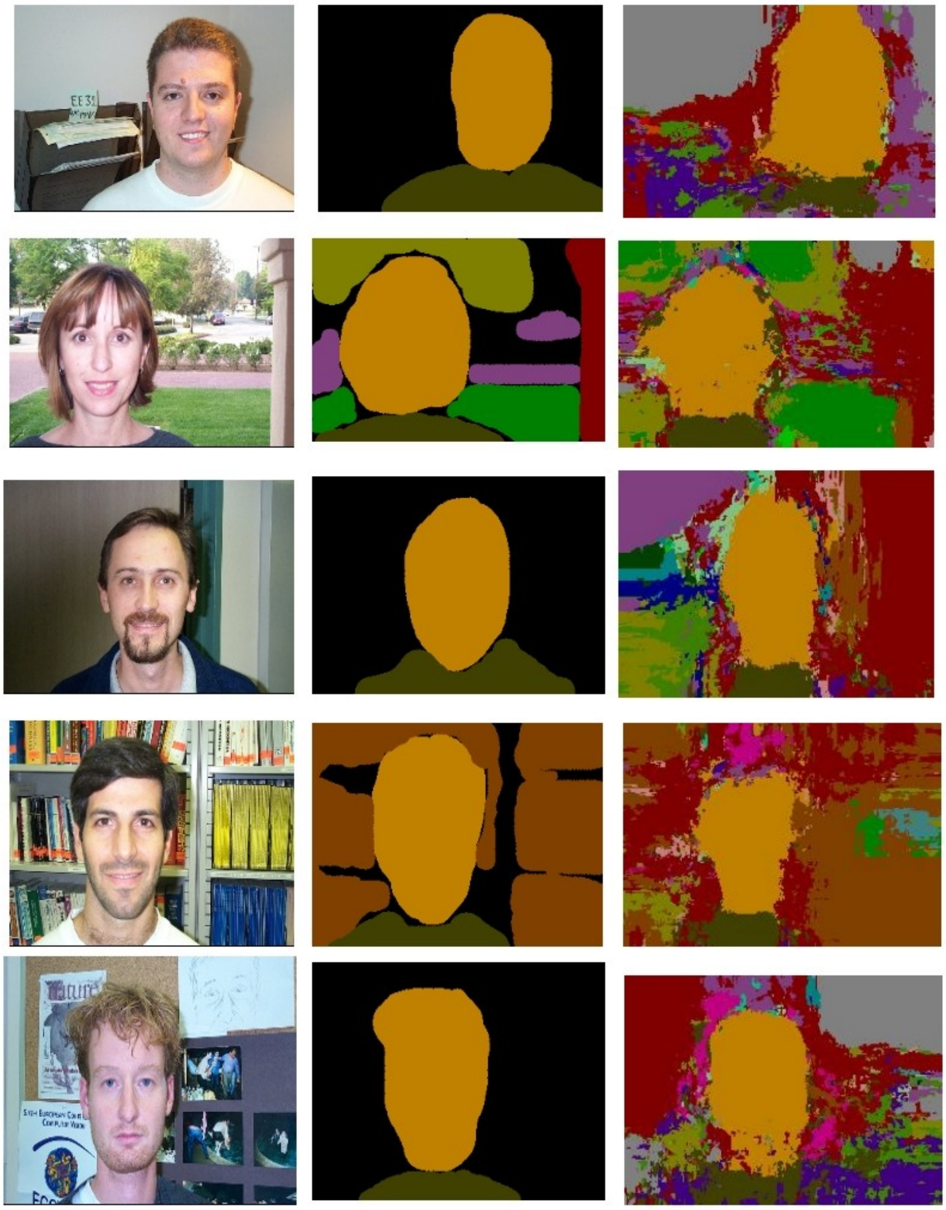


Figure 13: Result images showing segmentation of class face and body

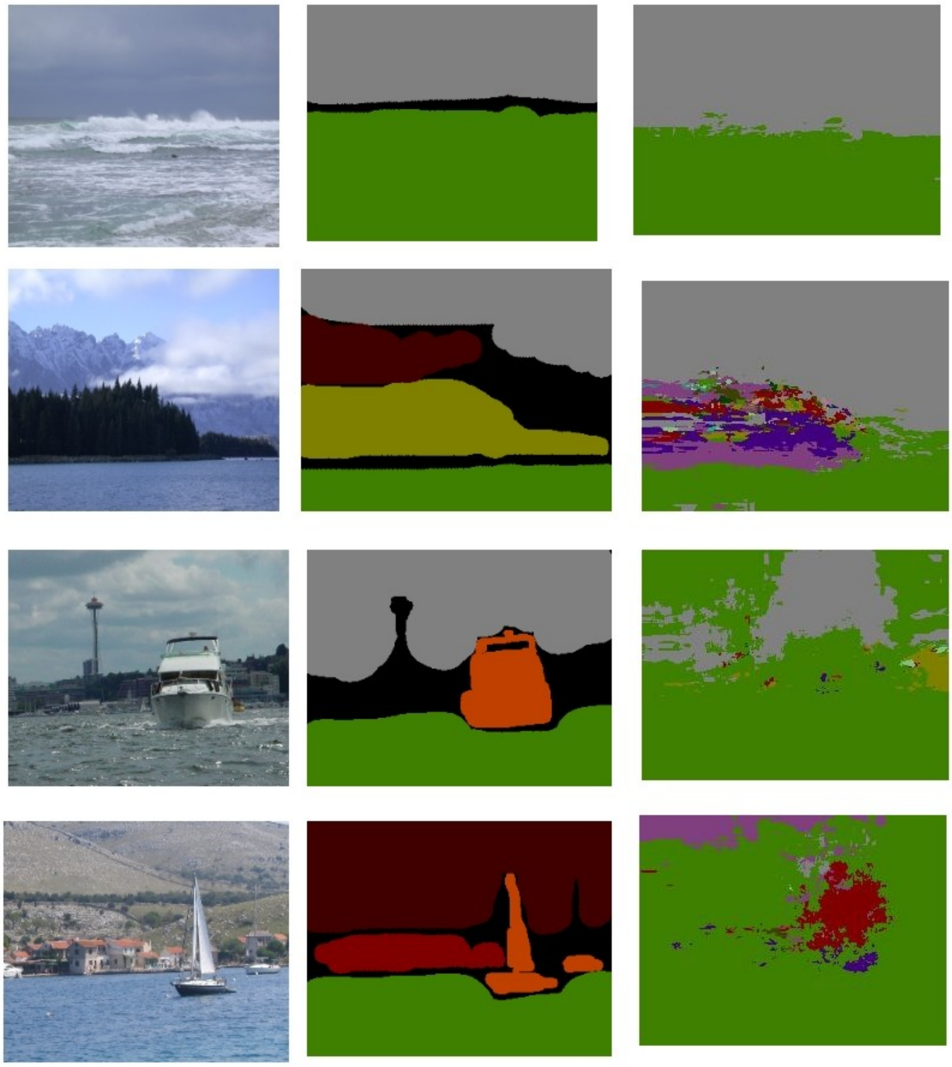


Figure 14: Result images showing segmentation of class boat,water and sky