Project Report on

Conditional Random Fields in Image Segmentation

Submitted in partial fulfillment of requirements of

MATH F266 - STUDY PROJECT

Under the guidance of :

Mr. Bijil Prakash

Sharan R Y 2016B4A70295G

Ahan M R 2016B4A30487G

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE,

PILANI - K.K. BIRLA GOA CAMPUS

Acknowledgements

We are pleased to acknowledge Mr. Bijil Prakash for his invaluable guidance during the course of this project work and without his guidance, this project would have been an uphill task.

We are also grateful to our friends and many other resources which constantly helped us regarding some issues which we faced and were successfully able to fix them.

Jupyter Notebook for Python which greatly helped us in writing the visualization and implementation part of the algorithm.

We thank Hugo Larochelle, for his explanation of the mathematics behind CRFs, his Deep Learning course, which enabled us to understand the top-tier papers.

Last but not the least, Digital Image Processing by Rafael C. Gonzalez and Pattern Recognition by Earl Rose and Richard Johnsonbaugh for being a constant source of information for smooth development of this project.

April 2018

Sharan R Y

Ahan M R

Table of Contents

Introduction

Image processing in the present day is concerned with automatic detection and classification of objects or events. Image Analysis and Processing is an important part of the process and the algorithms used during the same are robust and state-of-the-art.

Recent developments in the field of Computer Vision, gave us inspiration to work with this project, working on understanding and trying to improve the accuracy of results in many applications of Image recognition and processing. We chose Conditional Random Fields because it's considered an important architecture because of the probability dependencies it holds which related the information together, thus adding more insight to the convoluted data.

There has been considerable amount of research concerning CRFs for image processing over the last decade, we aspired to understand these developments and make our contributions if any. We show the superiority of CRF based models over various traditional Image processing methods.

We set out to understand the mathematical design of the method and how exactly does it do better than other models and its shortcomings along with a short summary on the related papers and inspiration and values added by these papers which continuously contributed to our journey.

Lastly, we summarize our learnings.

Background

Image segmentation using Conditional Random Fields(CRFs) is an important image processing technique which has gotten attention in the theoretical research and practical application due to its unique probabilistic approach.

These transformations can be applied directly onto the output with ease but sometimes, it will require few changes to coarse the algorithms to model the data in such a way that it works mainly in extracting the useful information from the data.

By studying CRFs we hoped to get in depth understanding of the problems faced by image processing community and the approaches researches have taken in pursuit to fix them.

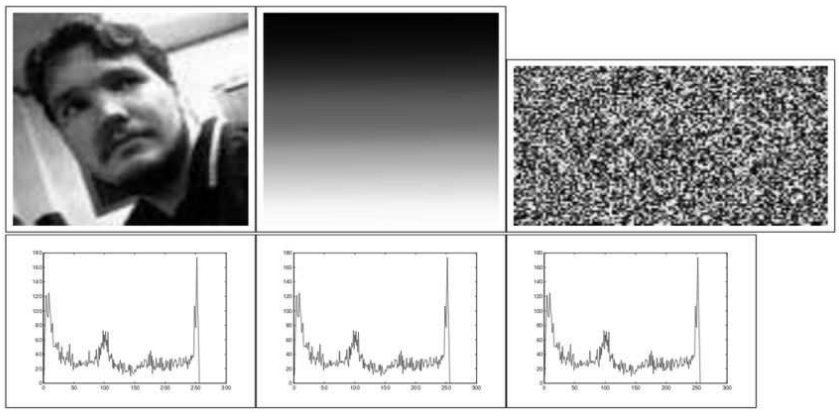
Our aim during the course of this project will be to bring out the motivation, ideas and the importance of CRFs in order to understand their workings.

What is Image Segmentation?

Image Segmentation is a very important idea in the field of Image Processing. It involves labelling the image upto the pixel-level based on which object they belong to. This is done by calculating the probability that a particular pixel belongs to a particular class.

The picture above is a simple example of image segmentation. Here the 2-D picture is divided into 3 parts based on which object they belong to, i.e. the horse, the rider along with the background.

Traditional Techniques



In the above image we can see how three completely different images have identical intensity histograms.

Traditional Image Segmentation algorithms like Otsu's threshold and Fuzzy c-means are very simply to apply and aren't very computationally intensive. These techniques mainly use data intensity to segment the images and they do not consider the spatial structure of the

images. This technique of using intensity histograms is not always reliable as shown in the above figure.

Present Day Techniques

The Main Ideas

The main ideas used in present day techniques are:

- Fully Convolutional Models (FCN)
- Decoder Variants
- Integrating Context Knowledge

State Of The Art

The current state-of-the-art model for semantic segmentation, Fully Convolutional Network (FCN) by Long et al., used mainly the idea of Fully Convolutional Models. They modified existing well developed CNNs like AlexNet, GoogleNet or VGGNet, replacing their fully connected layers with convolutional layers to get spatial segmentation maps instead of classifications as they were originally intended to.

Shortcomings

Spatially Invariant: Does not consider relations between widely placed pixels and hence is not globally aware.

Efficiency: Cannot currently be used for real time segmentation applications. Takes way longer than the currently used framerate in visual devices.

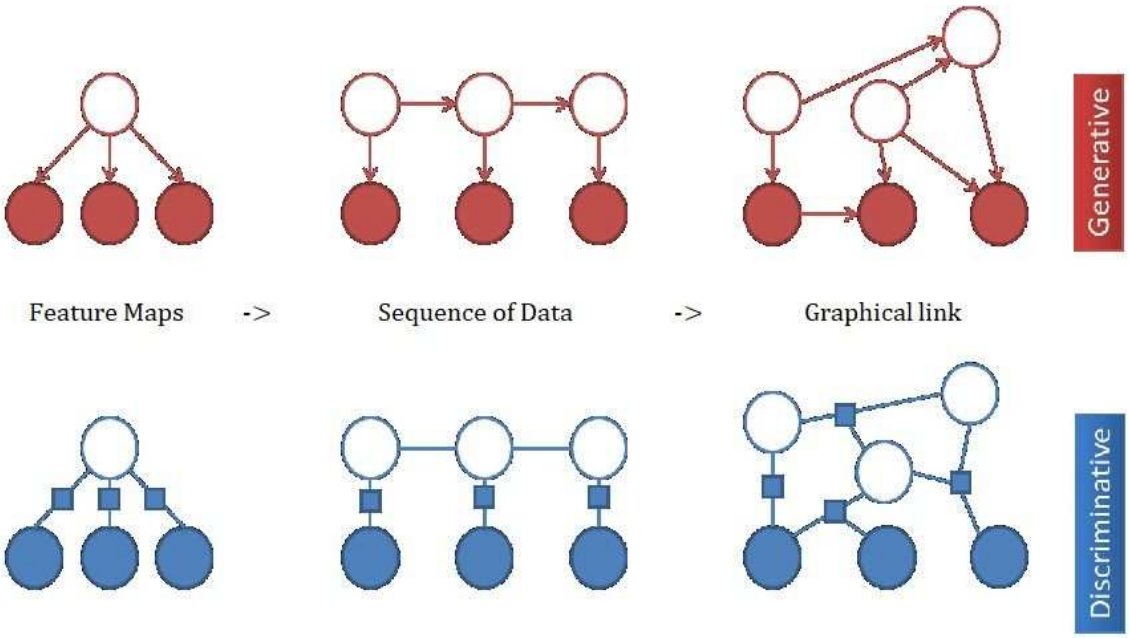
No Instance Awareness: Inability to use the previous images' context to segment the current image.

Conditional Random Fields

Introduction:

Conditional Random Fields(CRFs) are a class of statistical modelling techniques which are mainly used in the pattern recognition and Image segmentation.

They are an undirected discriminative models used for predicting the spatial relationships and sequences. They use contextual information from previous labels, thus increasing the chances of an accurate prediction. CRFs are discriminative probabilistic models and their underlying principle is that they apply Logistic Regression on sequential inputs to get the output.



Definition Let G be a factor graph over X and Y. Then (X, Y) is a *conditional random field* if for any value $\mathbf{x}$ of X, the distribution $p(\mathbf{y}|\mathbf{x})$ factorizes according to G.

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \prod_{a=1}^{A} \Psi_a(\mathbf{y}_a, \mathbf{x}_a).$$

The distribution P(y|x) given above is a conditional distribution for CRFs in general where X is the input data and y is the output generated. Z(x) is a normalizing constant and is used as a conditioning factor and simplifies the graphical models.

What Are Discriminative Classifiers?

There are two kinds of models for classification that are used, Discriminative classifiers and Generative classifiers. Discriminative Classifiers as the name states discriminates or plots a decision boundary, in the given vector space, between different classes. Conditional Random Fields(CRFs) are a type of Discriminative Classifiers, that particularly work on the relationship between sequence of input data and class and use conditional probability to discriminate.

Generative models, on a contrary, trains to model classes probabilistically, and it uses joint probability distributions. For a comparison, Naive Bayes, a probabilistic classifier, is a Generative algorithm, and Logistic Regression, which is a classifier based on Maximum Likelihood estimation, is a discriminative model.

Logistic Function is used for calculating the probability that the given input or sequence of inputs belong to a particular class.

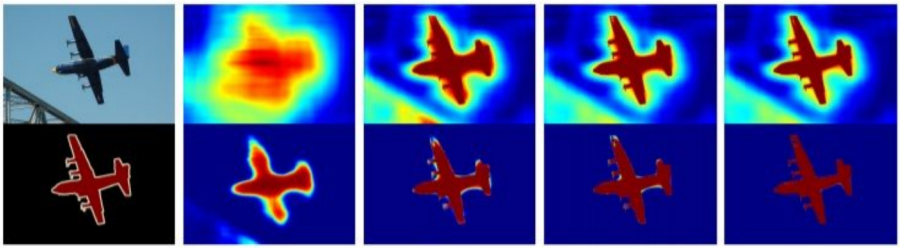
$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^{x}}{1+e^{x}}$$

CRFs In Image Segmentation

This ability to use sequential data image and construct a conditional probability has proved useful in the task of image segmentation. In the case of Image Segmentation this is done by constructing joint probability of n-number of class assignments given n-number of pixels. This enables the model to become context aware.

The CRFs have been really useful in the post processing stages as they are used to refine the output of the segmentation system. It captures the details of each pixel and relates to the data around it.

It enables the combination of lower level image information, i.e the interconnection of pixels and produces an output of multi pixel class systems producing and analysing scores per pixel class.



A visual demonstration of how CRFs help as post processing tool. Here, (a) is the input image along with the ground truth segmentation. (b) is after passing through a CNN. (c) is after one pass through a CRF, (d) is after 2 passes and (e) after 3 passes.

One important feature available in the usage of Conditional random fields is mainly that it captures the dependencies of two different pixels distant from each other which is not possible in Convolutional Neural Networks(CNNs). It encodes known dependencies and relationships between pixel and its neighbourhood and has been proven to be helpful in constructing spatial and/or locale awareness.

Convolutional Neural Networks(CNNs), due to efficiency with image related tasks acquired by training with large amounts of data, are used for image processing in the recent years . The resultant output is a classification maps that has a acceptable accuracy in spite of not considering the dependencies on the neighbourhood of each pixel. However, they tend to produce segmentation with rough edges. They also fail to relate between pixels which are distant from each other. This results in improperly segmented images leading to reduced accuracy.

Because of this particular foreseen circumstance, we use Conditional random fields(CRFs) and Markov random fields(MRFs) as a post processing step to bring significant improvements by modelling the important characteristics geometrically such as the shape of the image, the regional connectivity with other pixels and the spatial-contextual information between many pre formed segmented data.

Hence, the CRFs along with Convolution Neural networks(CNNs) can be used for training and modelling the spatial interpretation and extraction of data for better fine-tuned features and thus building upon the power of CNNs and improving the quality of image segmentation.

How CRFs work?

A vanilla CRF's energy function has two main components:

$$E(\mathbf{x}) = \sum_i \psi_u(x_i) + \sum_{i<j} \psi_p(x_i, x_j),$$

Unary Energy Components $\psi_u(x_i)$: This calculates the inverse-likelihood/cost of assigning x_i label to the ith pixel. This unary energy is obtained from a CNN, which is responsible for predicting the label of the given pixel without considering any factors like consistency, smoothness.

Pairwise energy Components $\psi_p(x_i, x_j)$ **:** Responsible for providing pairwise pixel relations which results in smoothening and consistency, the properties which unary energy components failed to provide. Where $\psi_p(x_i, x_j)$ is computed using the relation:

$$\psi_p(x_i, x_j) = \mu(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k_G^{(m)}(\mathbf{f}_i, \mathbf{f}_j),$$

Here $k_G^{(m)}$ is the Gaussian kernel applied with learned weights, on the feature vectors of the pixels. These feature vectors are derived using RGB data and feature information of the each pixel i or j. This is multiplied by $\mu(x_i, x_j)$, the compatibility function, used to measure the compatibility between each pair of pixels as the name implies.

Different Image Segmentation models that make use of CRFs

Here is brief summaries of the different models we came across in the research papers we read. (In the order they were originally published)

Object Class Segmentation Using Boosted Conditional Random Fields - 2010

Superpixel based framework for object class segmentation. Apart from standard unary potentials the CRF is made up of contrast sensitive pairwise energy function to characterize interactions between neighbouring pixels along with a higher order criterion to enforce labelling consistency.

Dataset: Standard MSRC database.

Effective Semantic Pixel labelling with Convolutional Networks and Conditional Random Fields - 2015

Per pixel class probabilities using CNNs, hand crafted features and CRFs. The CRFs have been primarily used for smoothening of the segments while respecting the edges present in the imagery.

Confirms that CNNs can perform well in the task of semantic labelling of aerial imagery. It was found that CRF smoothening had a negative effect on accuracy but did help in labelling visually by removing speckles from the classified segments.

They reason that the accuracy with CRFs decreased because they were applied at a pixel level. They alo go on to speculate that the accuracy will improve if we apply CRF at a superpixel level using higher order potentials.

Dataset: ISPRS 2D semantic labelling challenge dataset

Higher Order Conditional Random Fields in Deep Neural Networks - 2016

Recognizes previous papers embedding CRFs in CNNs and training them end-to-end but lacking higher order potentials. Reminds that previous research has found higher order potential to improve accuracy.

They employ two different kinds of higher order potential in an attempt to improve the accuracy along with the already widely used unary and binary potentials.

$$E(\mathbf{x}) = \sum_i \psi_i^U(x_i) + \sum_{i<j} \psi_{ij}^P(x_i, x_j) \; + \sum_d \psi_d^{\mathrm{Det}}(\mathbf{x}_d) + \sum_s \psi_s^{\mathrm{SP}}(\mathbf{x}_s),$$

Higher Order Energies from Object Detection: x_d is formed by the pixels belonging to the d^{th} object detection. It is the energies calculated among all the pixels assigned to a particular object. We speculate that this reduces the chances of speckles appearing in the results

Higher Order Energies base on Superpixels: x_s is formed by pixels belonging to s^{th} superpixel. It is the energies calculated among all the pixels assigned to a particular superpixel. This provides superpixel label consistency.

They trained the CRF model end-to-end with the CNN model. They credit the use of higher order potentials in CRFs for achieving state-of-the-art accuracy at the time.

Dataset: PASCAL VOC

High-Resolution Image Classification Integrating Spectral-Spatial-Location Cues by Conditional Random Fields - 2016

A CRF based model on aerial imagery dataset
The proposed algorithm integrates:
Spectral: Using unary potentials to get basic information of the different classes of landmasses.
Spatial contextual: Using pairwise potentials to establish neighbouring interactions between pixels resulting in spatial smoothing.
Spacial location: Using higher order potentials to fix the problem of lancovers which get easily confused with other land cover types.

For high resolution images. Each of these potentials have different focus and can hence provide information from varying perspectives.

Dataset: Aerial Imagery from Pavia Center ROSIS Dataset

Conditional Random Field and Deep Feature Learning for Hyperspectral Image Segmentation - 2017

Using unary and pairwise potentials on hyperspectral remote sensing image segmentation. This is case of 3D segmentation, where the patches are three dimensional cubes. It clso indes the deconvolutional model which helps in improving segmentation masks.

They have integrated CRF and CNN into a common framework where parameters of CRFs are calculated using CNNs making it a Deep CRF. The CRF-CNN architecture's output is improved by deconvolutional block inside the CRF pairwise potential calculations.

They have also introduced a new dataset.

Summary

We have seen that CRF has the capability to use the spatial contextual information other wise not made use of by CNNs. The main reason for this capability of CRFs is their pairwise potentials. This helps smoothen the segments. But as we saw in the earlier papers this may not increase the accuracy. In order for CRF to positively affect the accuracy, we need to make use of higher order potentials. This can be done is many ways. Using energies among the pixels of a superpixel and/or detected object is one of the ways.

We have also seen that CRFs have been, more often than not, used along with a convolutional network. The initial coarse segmentation is being done with the CNN. The smoothening, removal of speckles, making classification consistent within a superpixel, etc., is being done using the CRF. The CRF is included either as a separate model or has been embedded into later stages of the CNN and trained end-to-end.

Code

April 24, 2018

1 Implementation of Image Segmentation using Conditional Random Fields:

1.1 Part 1- Image segmentation and formation of Super Pixels:

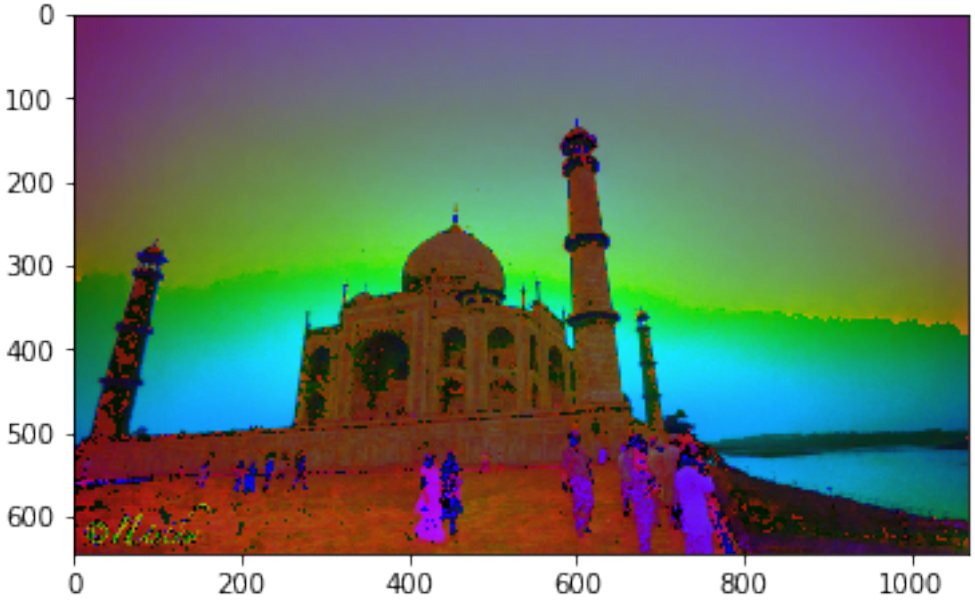
We import the basic libraries necessary for our project using Python, which includes NumPy, Matplotlib, Python imaging Library.

```
In [180]: import skimage
          skimage.__version__

Out[180]: '0.13.1'

In [181]: import numpy as np

          %matplotlib inline
          import matplotlib.pyplot as plt #Importing libraries for our project, scikit image i

In [182]: from PIL import Image
          import cv2
          #Importing opencv library to input any real life picture
          myImage = Image.open("tajmahal1.jpg");
          #myImage.show();
          tajmahal=myImage
          im = cv2.imread("tajmahal1.jpg")
          tajmahal = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
          tajmahal1 = cv2.cvtColor(im, cv2.COLOR_BGR2HLS)
          #above code is mainly for converting any data to black and white data and analyzing
          #ngcm= greycomatrix(im, [1], [0], 256, symmetric=False, normed=True)

In [183]: from skimage import data
          #tajmahal = data.moon()
```

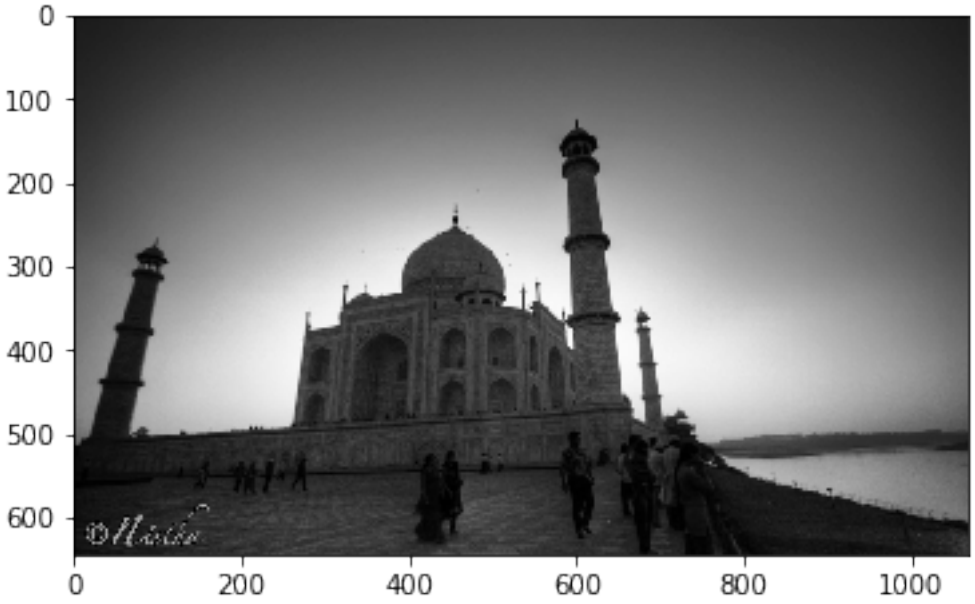
Using Matplotlib, we convert the image into a Grayscale Image and hence, work on this data, as Matplotlib now converts it into an array which will be used directly as an input for pre-processing. So we do so for any kind of Image given as input.

```
In [184]: plt.imshow(tajmahal1, cmap='gray')
```

```
Out[184]: <matplotlib.image.AxesImage at 0x7fd9b5ae6ac8>
```

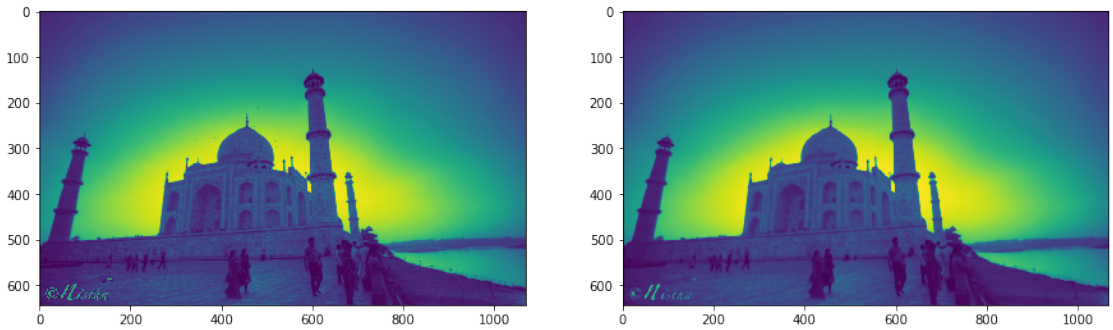



```
In [185]: plt.imshow(tajmahal, cmap='gray')

Out[185]: <matplotlib.image.AxesImage at 0x7fd9b5be0b70>
```



We use the Selenium filter, a morphological feature which returns a structuring element where elements of neighbourhood are 0 and 1.

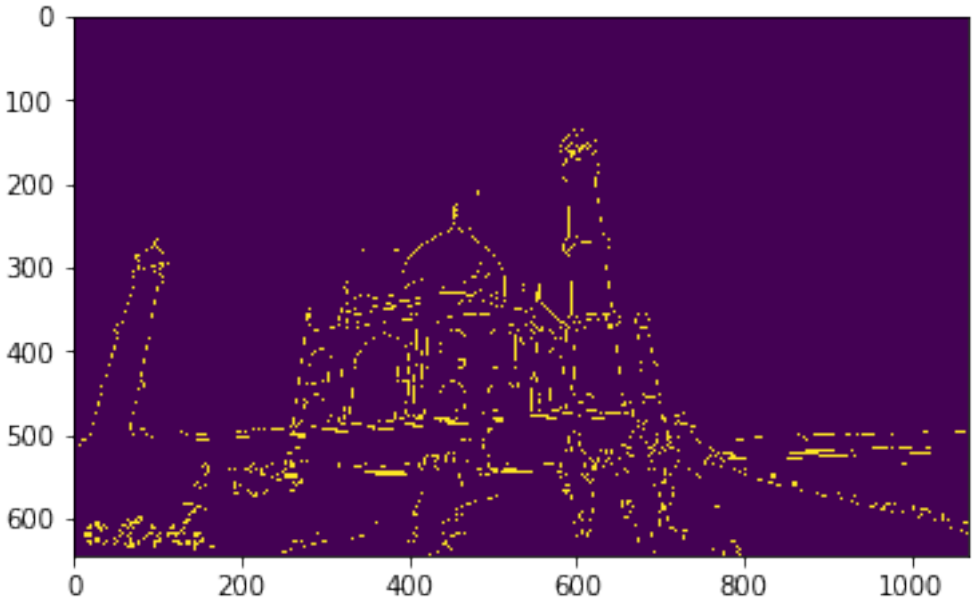
```
In [186]: from skimage import filters

          tajmahal_denoised = filters.median(tajmahal, selem=np.ones((5, 5)))
          #selem is structuring element
          f, (ax0, ax1) = plt.subplots(1, 2, figsize=(15, 5))
          ax0.imshow(tajmahal)
          ax1.imshow(tajmahal_denoised);
```



We now plot the required boundary points using the value of standard deviation which helps in deciding the threshold of the data needed out for pre-processing which can be automated directly using machine learning, but here, we have shown for values to show the different feature maps for classification.

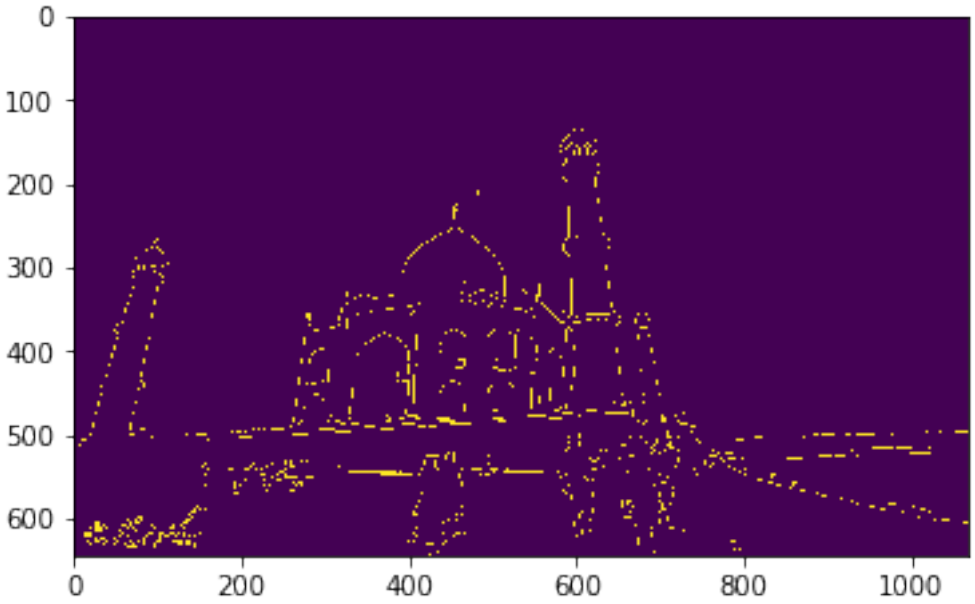
```
In [187]: from skimage import feature
          edges = skimage.feature.canny(tajmahal, sigma=01.2)

          plt.imshow(edges)
```

```
Out[187]: <matplotlib.image.AxesImage at 0x7fd9b5c85860>
```

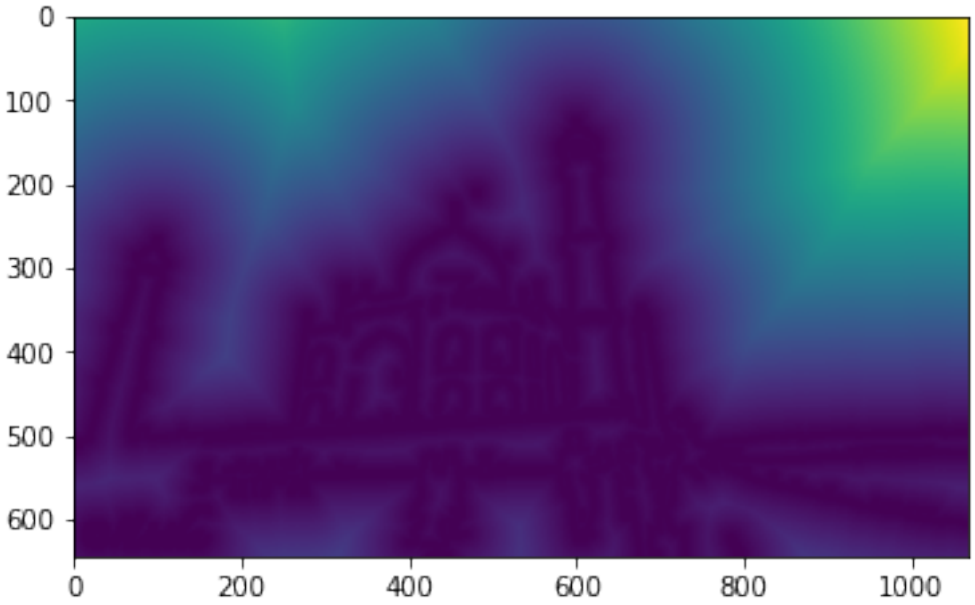


In [188]:
```
from skimage import feature
edges = skimage.feature.canny(tajmahal, sigma=1.5)

plt.imshow(edges)
```

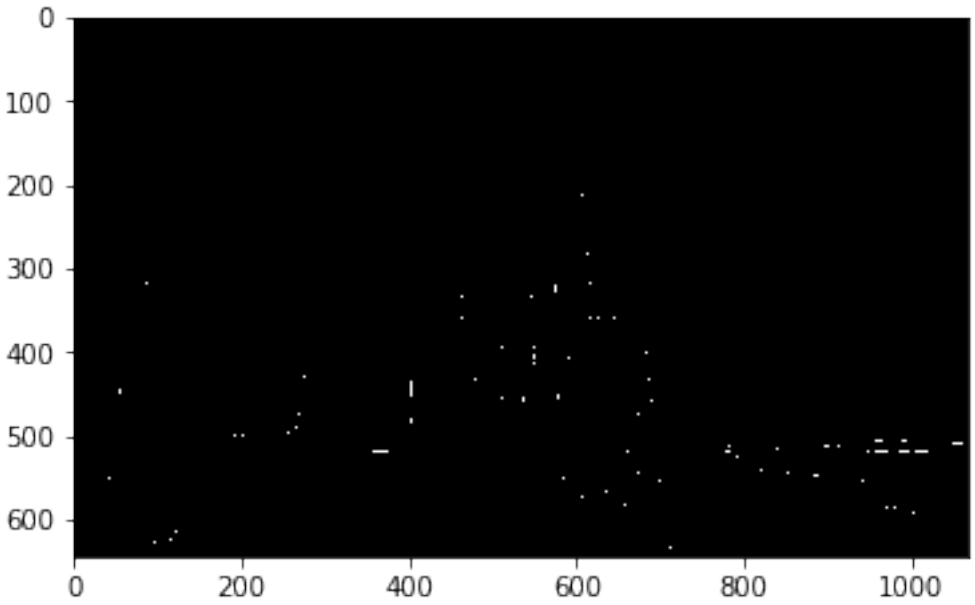
Out[188]: <matplotlib.image.AxesImage at 0x7fd9b5cb9c88>



In [189]:
```
from scipy.ndimage import distance_transform_edt
dt = distance_transform_edt(~edges)

plt.imshow(dt);
```



In [190]:
```
local_max = feature.peak_local_max(dt, indices=False, min_distance=5)
plt.imshow(local_max, cmap='gray');
```

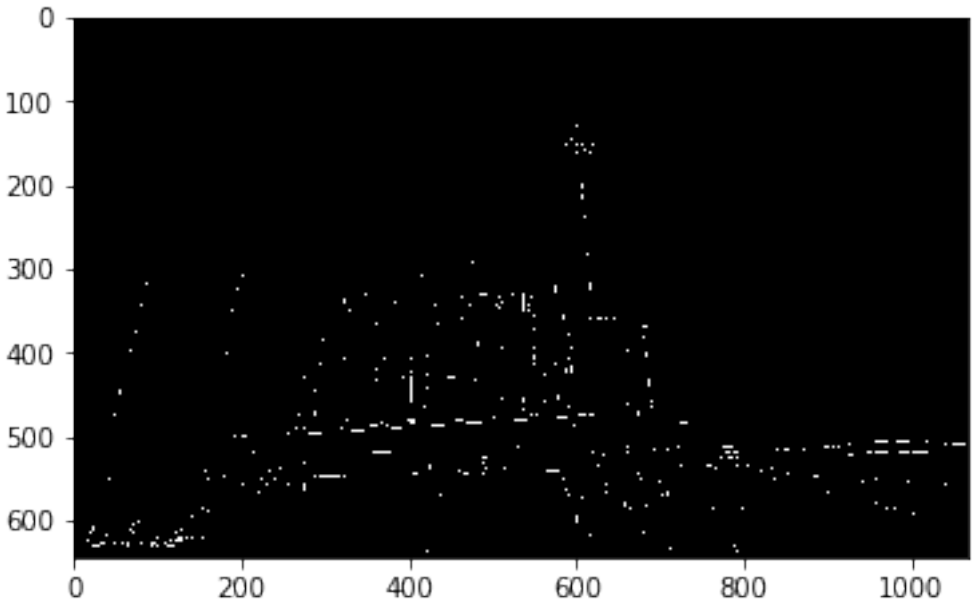


In [191]:
```
peak_idx = feature.peak_local_max(dt, indices=True, min_distance=5)
peak_idx[:]
```

```
Out[191]: array([[636, 682],
                 [635, 682],
                 [634, 682],
                 ...,
                 [142, 603],
                 [142, 602],
                 [142, 601]])
```

We get the set of maxima points which is used for clustering of data points using the distance metric of d(x,y)=||x-y|| and hence, creates a feature map, which is classified based on color maps alloted and segments the data directly into 2 colors which splits the data into 2 segments.

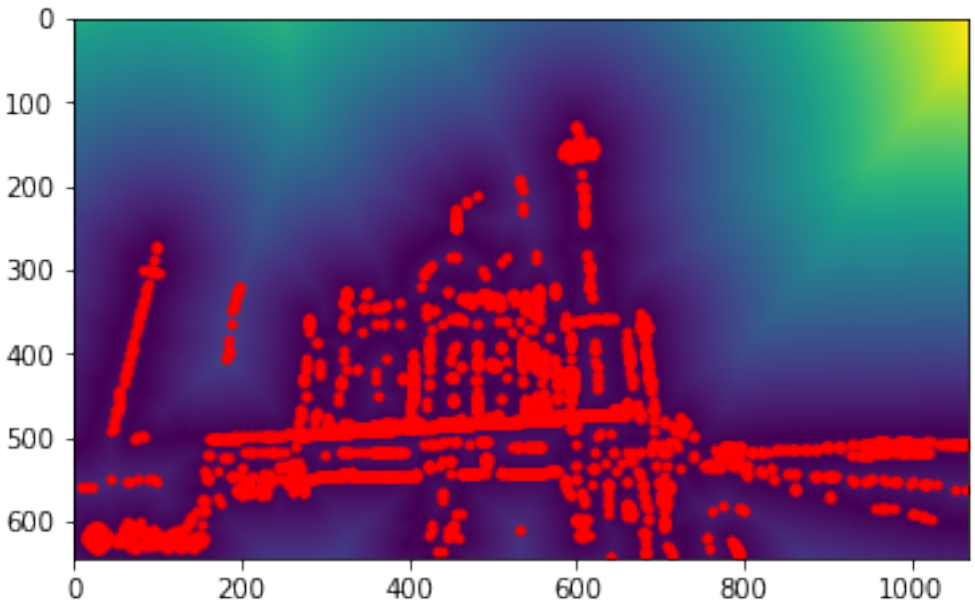
```
In [192]: local_max = feature.peak_local_max(dt, indices=False, min_distance=1)
          plt.imshow(local_max, cmap='gray');
```




```
In [193]: peak_idx = feature.peak_local_max(dt, indices=True, min_distance=2)
          peak_idx[:]
```

```
Out[193]: array([[640, 794],
                 [640, 714],
                 [639, 794],
                 ...,
                 [129, 600],
                 [128, 600],
                 [127, 600]])
```

```
In [194]: plt.plot(peak_idx[:,1], peak_idx[:,0], 'r.')
          plt.plot(peak_idx[1], peak_idx[0], 'r.')
          plt.imshow(dt);
```

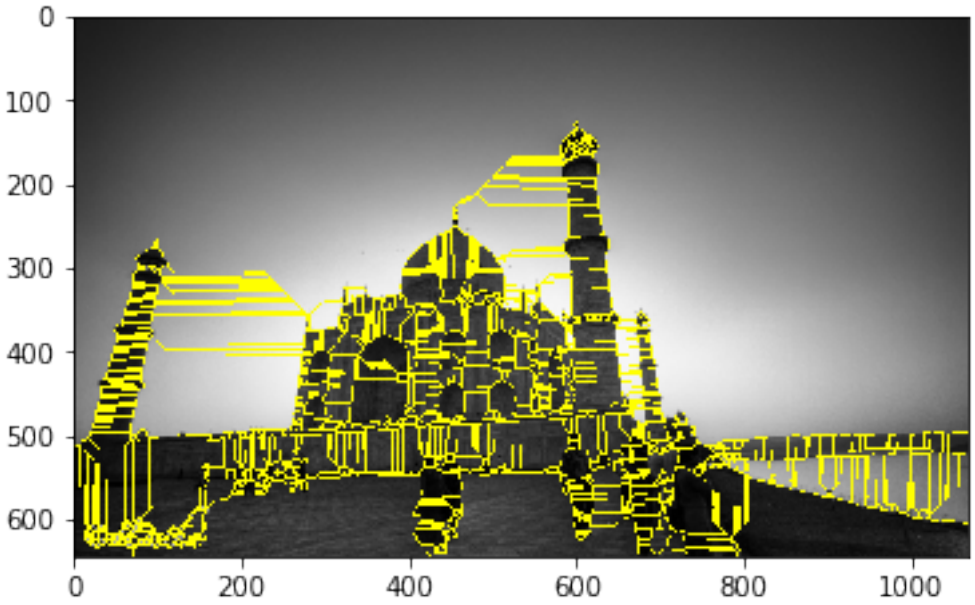



```
In [195]: from skimage import measure

          markers = measure.label(local_max)
```

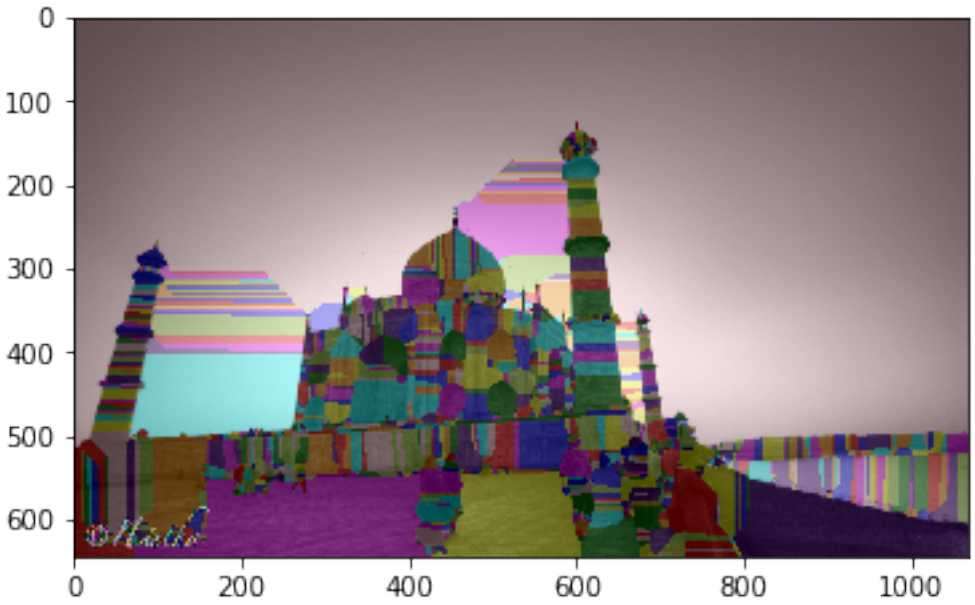
This is a set of super pixels formed by the feature classification and here is where the color maps are alloted based on the pixel density of the colors, which is alloted based on the array we initially created using Matplotlib library.

1.1.1 Formation of Super Pixels:

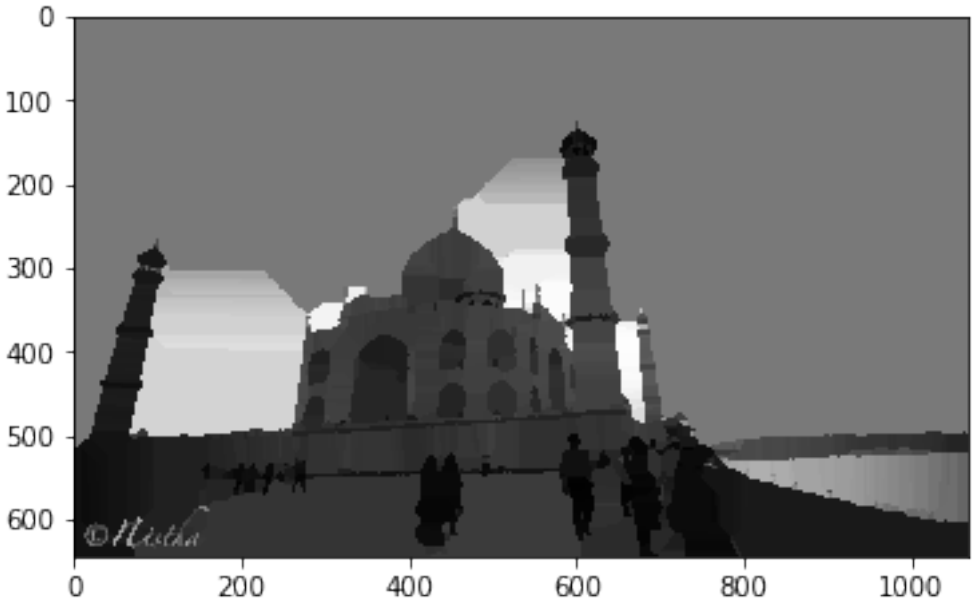
```
In [196]: from skimage import morphology, segmentation

          labels = morphology.watershed(-dt, markers)
          plt.imshow(segmentation.mark_boundaries(tajmahal, labels));
```



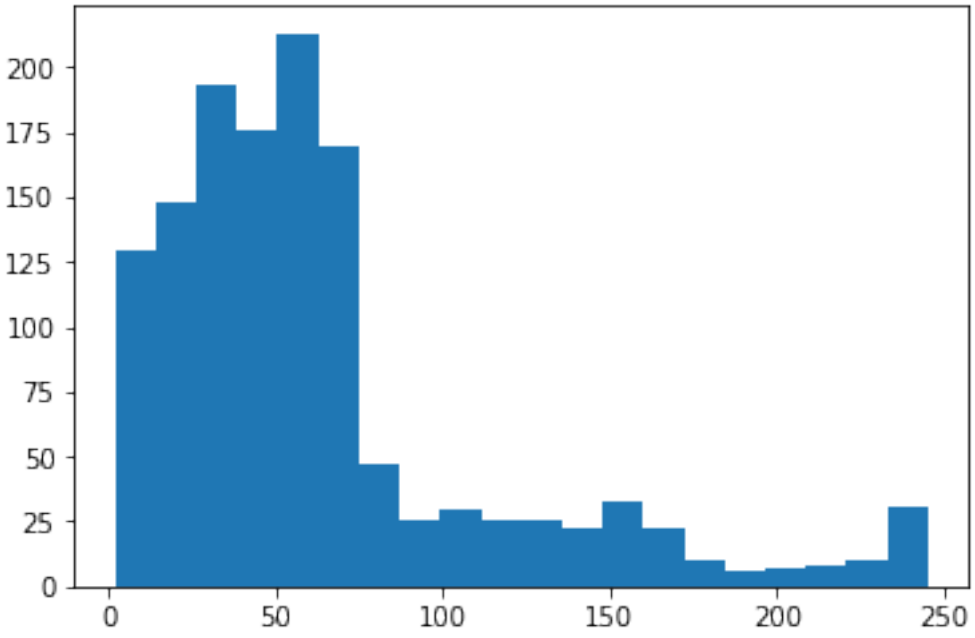

```
In [197]: from skimage import color
          plt.imshow(color.label2rgb(labels, image=tajmahal));
```




```
In [198]: plt.imshow(color.label2rgb(labels, image=tajmahal, kind='avg'), cmap='gray');
```




```
In [199]: regions = measure.regionprops(labels, intensity_image=tajmahal)

In [200]: #print(regions)

In [201]: region_means = [r.mean_intensity for r in regions]
          plt.hist(region_means, bins=20);
```



```
In [202]: from sklearn.cluster import KMeans
          model = KMeans(n_clusters=2)

          region_means = np.array(region_means).reshape(-1, 1)
          model.fit(np.array(region_means).reshape(-1, 1))
          print(model.cluster_centers_)
```

```
[[ 44.4911453 ]
 [166.03060593]]
```

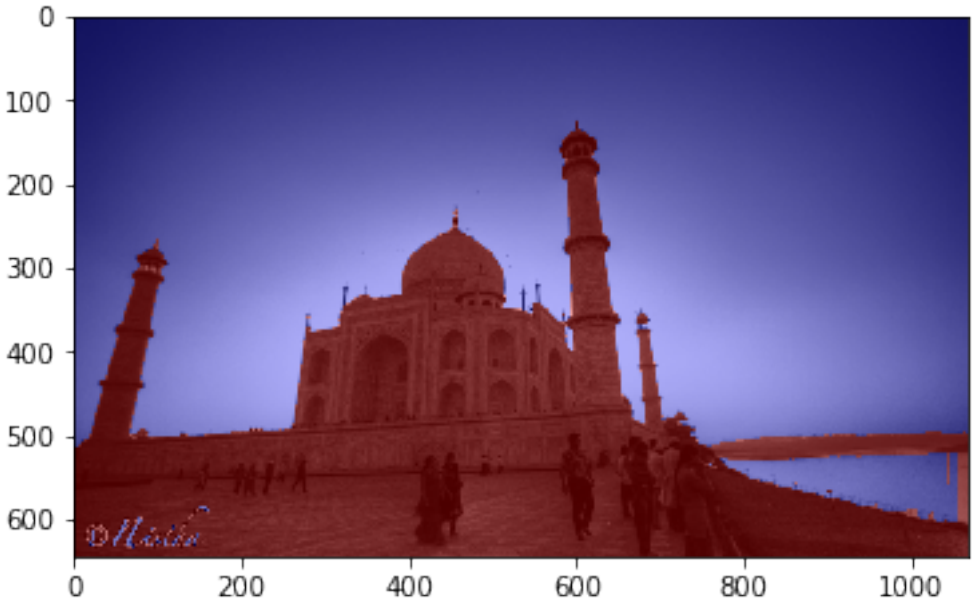
```
In [203]: bg_fg_labels = model.predict(region_means)
          bg_fg_labels
```

```
Out[203]: array([0, 0, 0, ..., 0, 0, 0], dtype=int32)
```

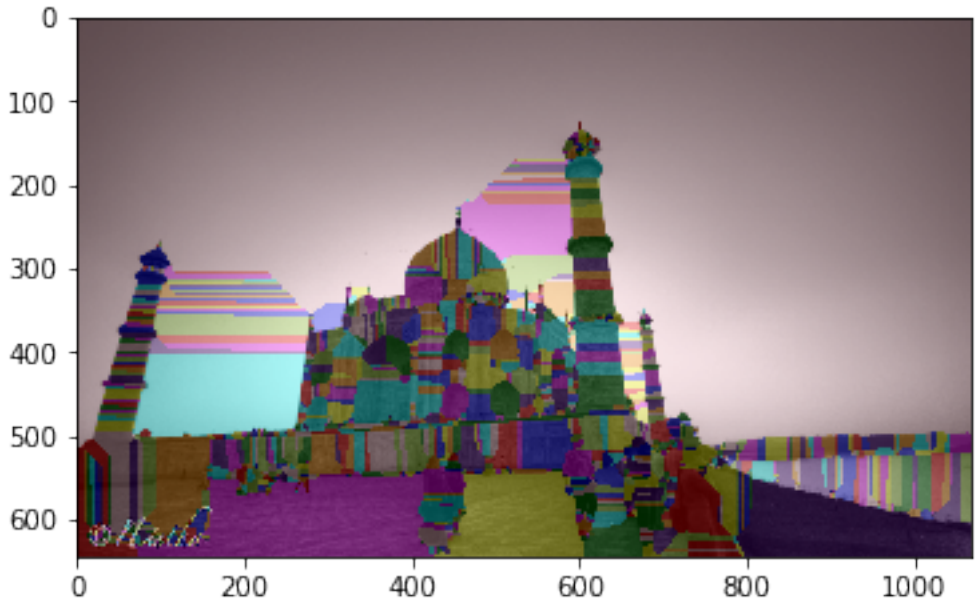
We created a n-tuple which helps in storing the coordinates of the region and hence, decides the bounds for the clear classification.

```
In [204]: classified_labels = labels.copy()
          for bg_fg, region in zip(bg_fg_labels, regions):
              classified_labels[tuple(region.coords.T)] = bg_fg
```

```
In [205]: plt.imshow(color.label2rgb(classified_labels, image=tajmahal));
```



```
In [206]: plt.imshow(color.label2rgb(classified_labels, image=tajmahal),cmap='gray');
          from skimage import color
          plt.imshow(color.label2rgb(labels, image=tajmahal));
```



Hence, we get the superpixels in the end, and use this as input CRFs. Further steps have been described below.

1.2 Part 2: Using the superpixels to segment the image

Getting the required libraries

```
In [4]: !pip install pystruct
        # !wget http://www.ais.uni-bonn.de/download/pascal_preprocessed/data_train.pickle
        # !wget http://www.ais.uni-bonn.de/download/pascal_preprocessed/data_val.pickle
        ! pip install cvxopt
        !pip install pyqpbo
```

```
Requirement already satisfied: pystruct in /usr/local/lib/python2.7/dist-packages
Requirement already satisfied: ad3 in /usr/local/lib/python2.7/dist-packages (from pystruct)
You are using pip version 9.0.3, however version 10.0.1 is available.You should consider upgrad
Requirement already satisfied: cvxopt in /usr/local/lib/python2.7/dist-packages
You are using pip version 9.0.3, however version 10.0.1 is available.You should consider upgrad
Requirement already satisfied: pyqpbo in /usr/local/lib/python2.7/dist-packages
You are using pip version 9.0.3, however version 10.0.1 is available.You should consider upgrad
```

Importing the required libraries.

```
In [0]: import numpy as np
        try:
            import cPickle as pickle
        except ImportError:
            import pickle

        from pystruct import learners
        import pystruct.models as crfs
        from pystruct.utils import SaveLogger
```

Loading preprocessed *superpixels* (what Part 1 aimed to do) for the CRFs to work with.

```
In [0]: data_train = pickle.load(open("data_train.pickle"))
```

```
In [0]: C = 0.01 #Regularization parameter
        n_states = 21 #
```

```
In [8]: print("number of samples: %s" % len(data_train['X'])) # no. of samples being trained o
        class_weights = 1. / np.bincount(np.hstack(data_train['Y']))
        class_weights *= 21. / np.sum(class_weights)
        print(class_weights) # showing normalized class weights
```

```
number of samples: 964
[0.01347196 1.20156934 3.20418491 1.03334963 1.20859606 1.37779951
 0.57249287 0.57249287 0.37817004 0.93515804 1.375507   0.77695461
 0.67594416 1.01184787 0.87572003 0.23247461 1.75515861 1.03205956
 0.96125547 0.7964159  1.00937693]
```

Initializing CRF model imported from pystruct library. This will unary and binary potentials among the superpixels which are then used to segment the image.

1.2.1 Model Definition:

```
In [0]: model = crfs.EdgeFeatureGraphCRF(inference_method='qpbo',
                                         class_weight=class_weights,
                                         symmetric_edge_features=[0, 1],
                                         antisymmetric_edge_features=[2])
```

1.2.2 Solver Definition:

Defining NSlackSSVM model which helps in training the earlier defined model. And then fitting the model.

```
In [ ]: experiment_name = "edge_features_one_slack_trainval_%f" % C

        ssvm = learners.NSlackSSVM(
            model, verbose=1, C=C, max_iter=10, n_jobs=-1,
            tol=0.0001, show_loss_every=5,
            logger=SaveLogger(experiment_name + ".pickle", save_every=100),
            inactive_threshold=1e-3, inactive_window=30, batch_size=100)
        ssvm.fit(data_train['X'], data_train['Y'])
```

1.2.3 Validation score

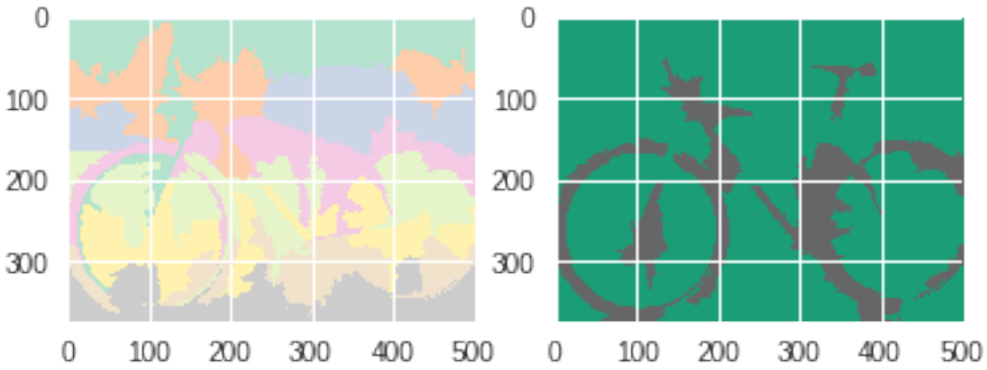
Loading validation dataset(superpixels). Calculating the score of the model. Please keep in mind that we have only partially trained the model due to constraints.

```
In [11]: data_val = pickle.load(open("data_val.pickle"))
         y_pred = ssvm.predict(data_val['X'])
         temp = y_pred
         # we throw away void superpixels and flatten everything
         y_pred, y_true = np.hstack(y_pred), np.hstack(data_val['Y'])
         y_pred = y_pred[y_true != 255]
         y_true = y_true[y_true != 255]

         print("Score on validation set: %f" % np.mean(y_true == y_pred))
```

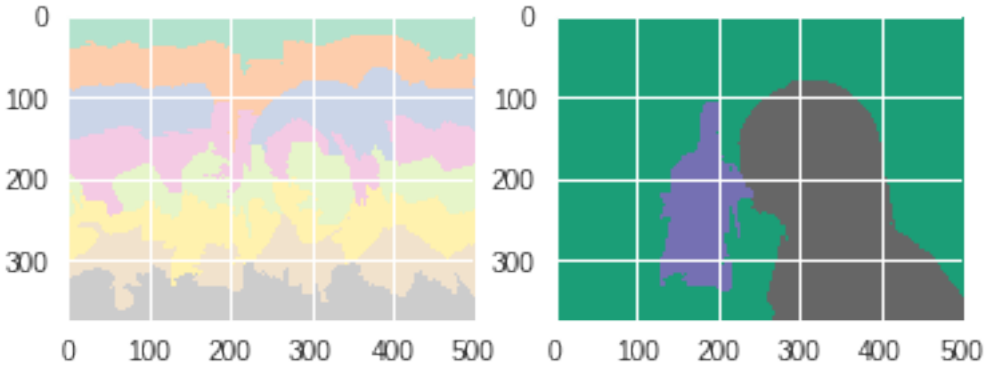
```
Score on validation set: 0.773596
```

1.2.4 Plotting results

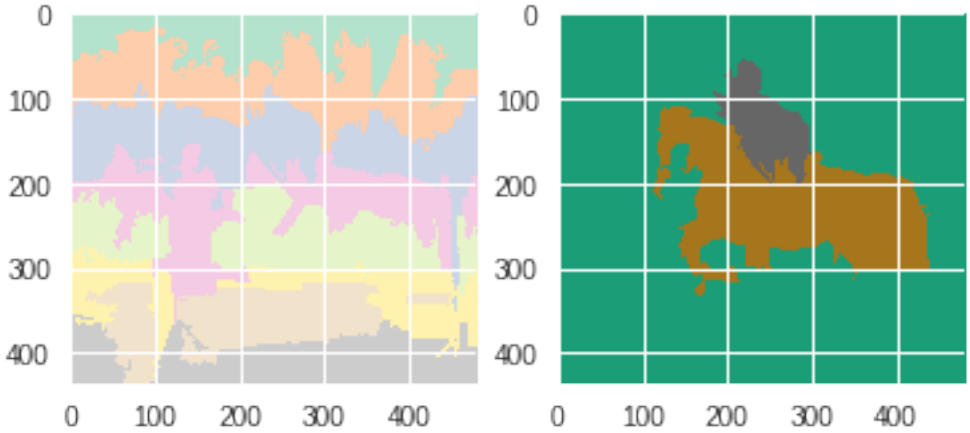
```
In [12]: %matplotlib inline
         from PIL import Image
         import numpy as np
         from matplotlib.pyplot import imshow, show
         import matplotlib.pyplot as plt
         import matplotlib.cm as cm
         from collections import OrderedDict
         cmaps = OrderedDict()
         for i in [209, 9, 23, 24, 31, 35, 28, 49, 50, 59, 64, 74, 80, 138, 198,  222, 223, 238
           plt.subplot(1, 2, 1)
           imshow(data_val['superpixels'][i], cmap='Pastel2')
           img2 = np.zeros(data_val['superpixels'][i].shape)
           for a in range(data_val['superpixels'][i].shape[0]):
             for b in range(data_val['superpixels'][i].shape[1]):
               img2[a,b] = data_val['Y'][i][data_val['superpixels'][i][a,b]]
           plt.subplot(1, 2, 2)
           imshow(img2, cmap='Dark2',)
           show()
         #   print(data_val['Y'][i])
           print(data_val['file_names'][i])
           print (i)
```



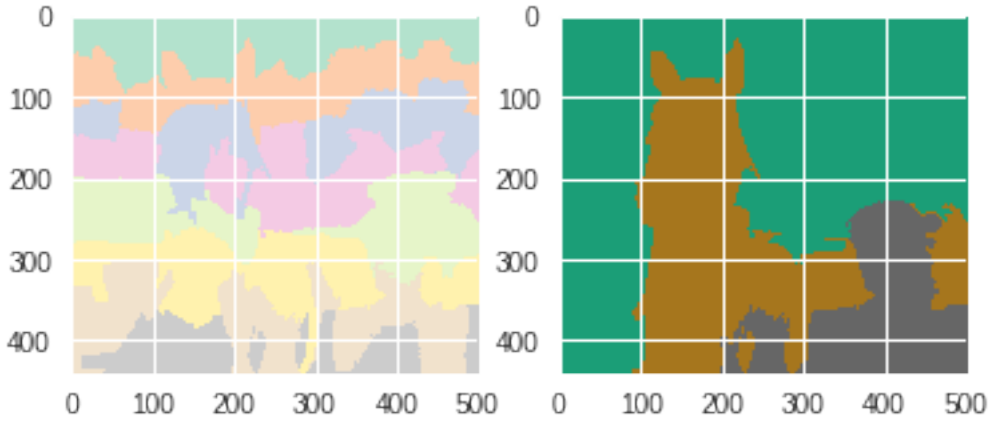
2007_006449
209



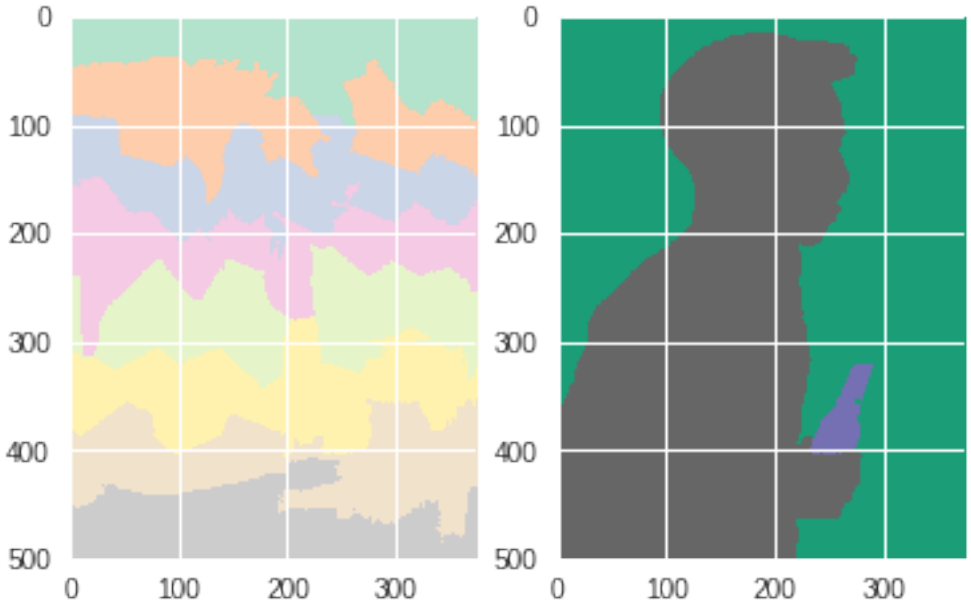
2007_000346
9



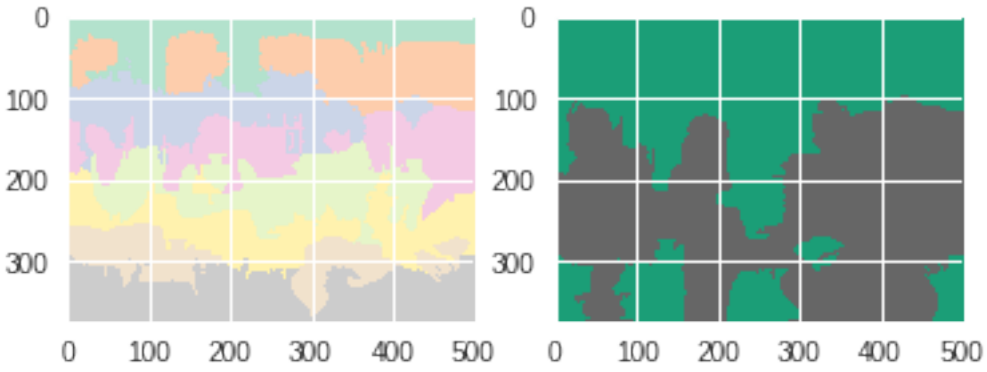
2007_000783
23



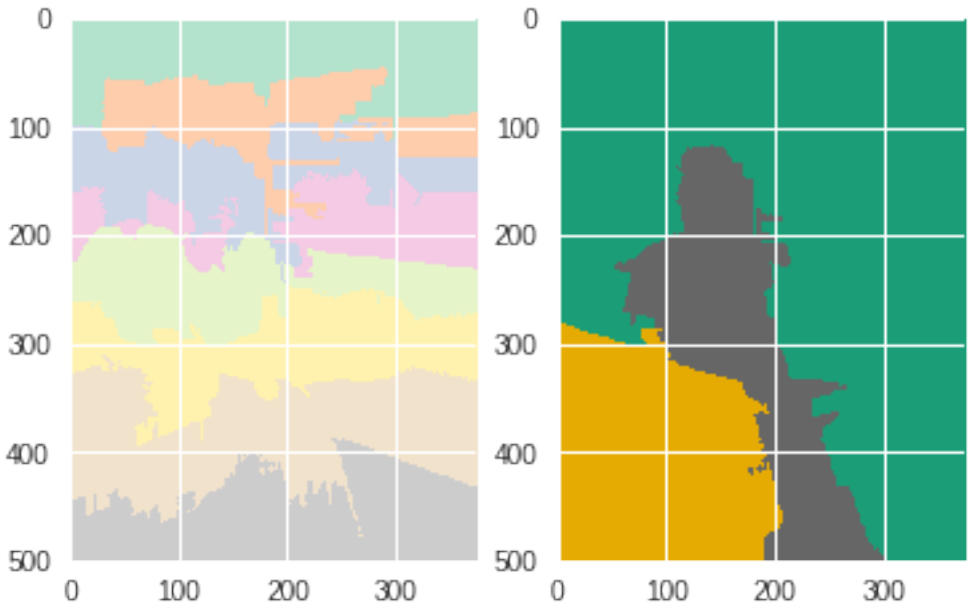
2007_000799
24



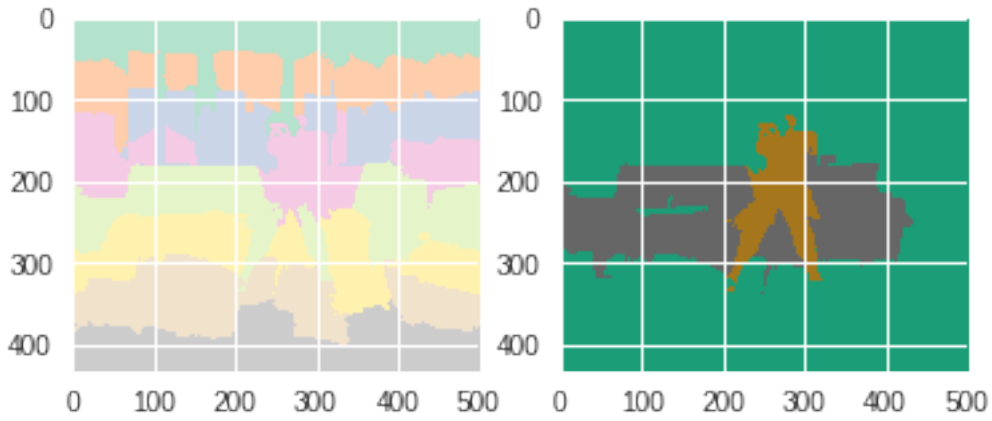
2007_000999
31



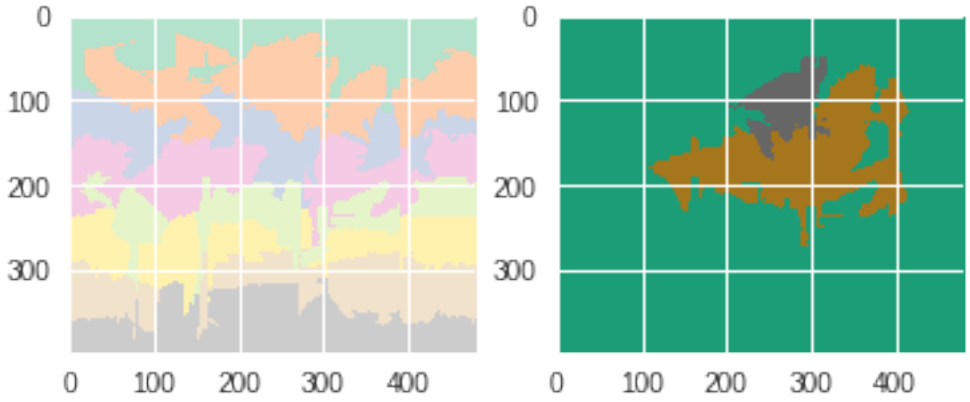
2007_001284
35



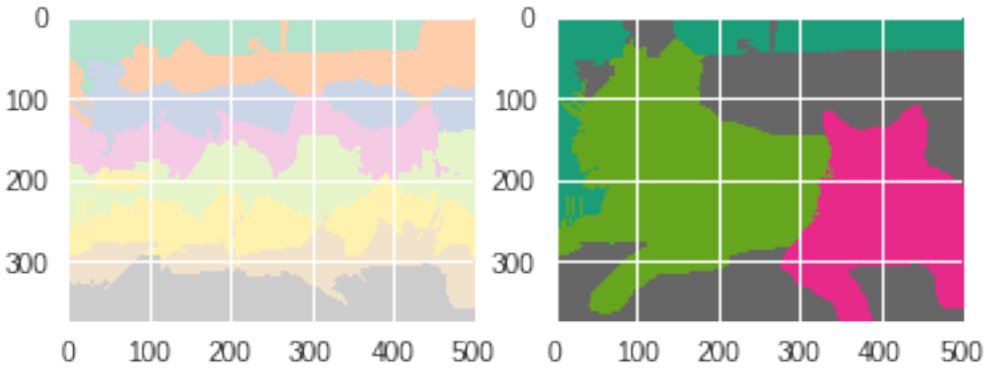
2007_000847
28



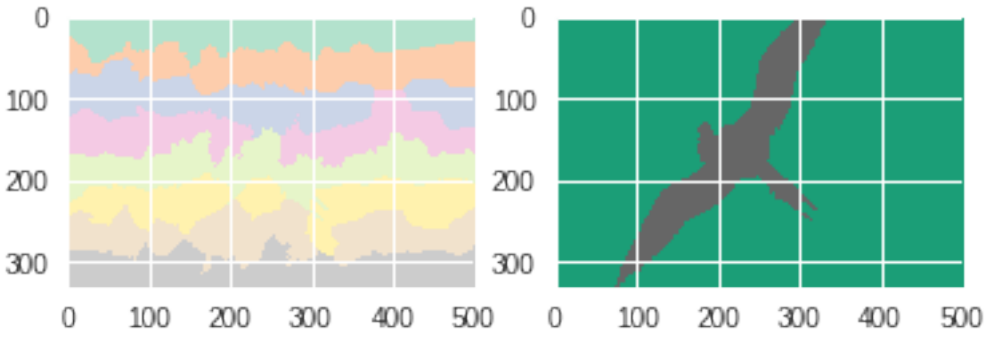
2007_001585
49



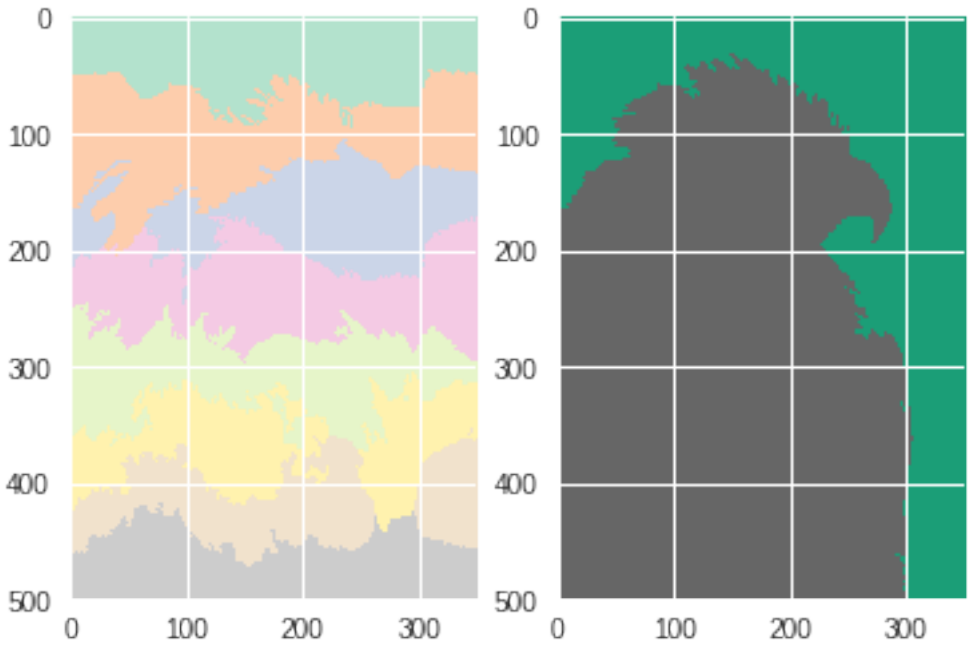
2007_001586
50



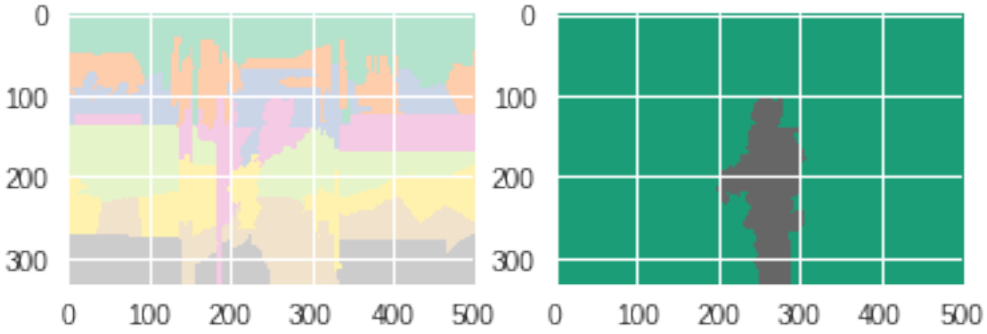
2007_001763
59



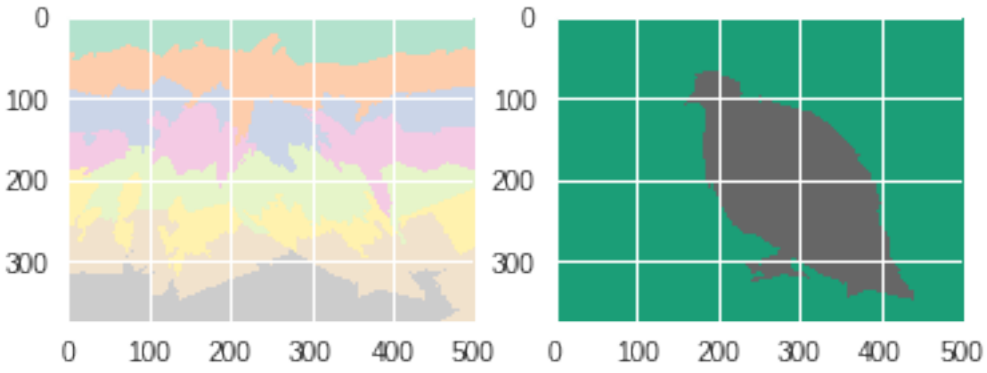
2007_002094
64



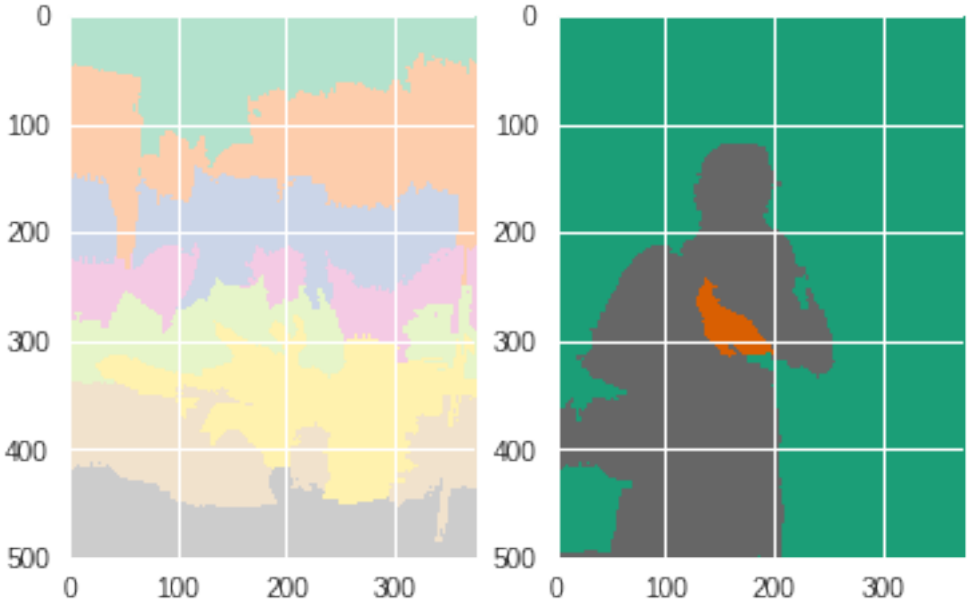
2007_002400
74



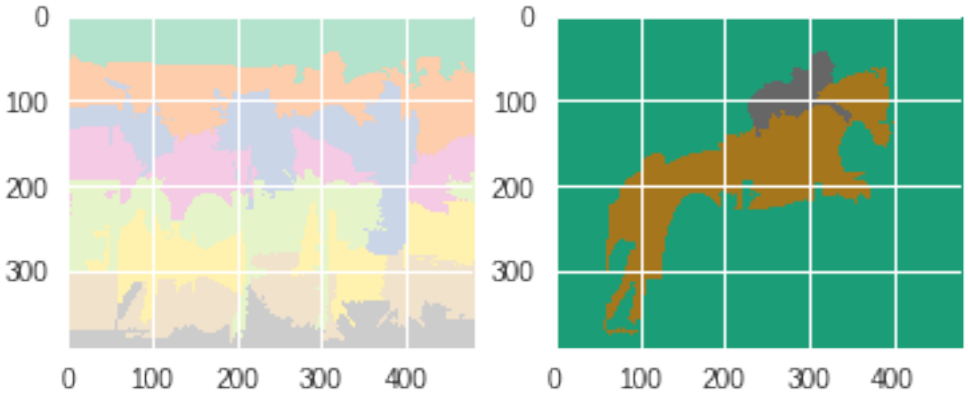
2007_002539
80



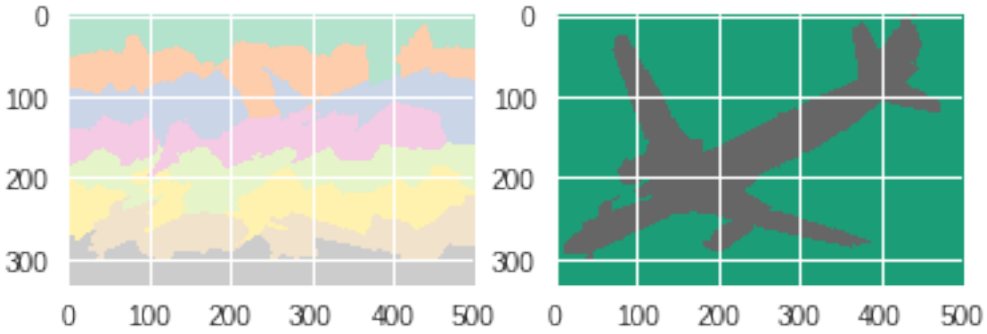
2007_004143
138



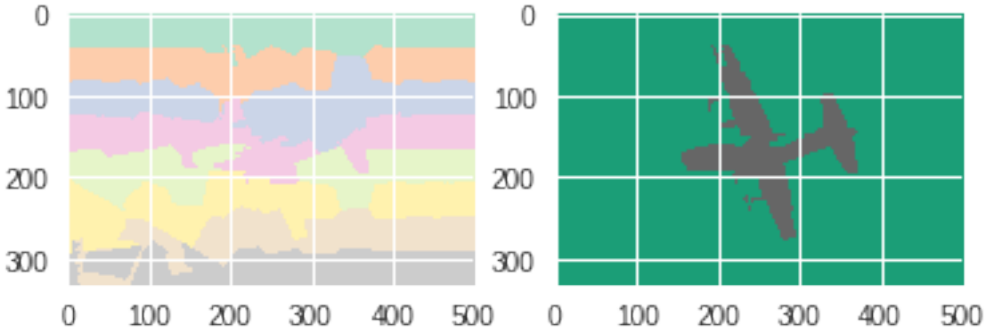
2007_006076
198



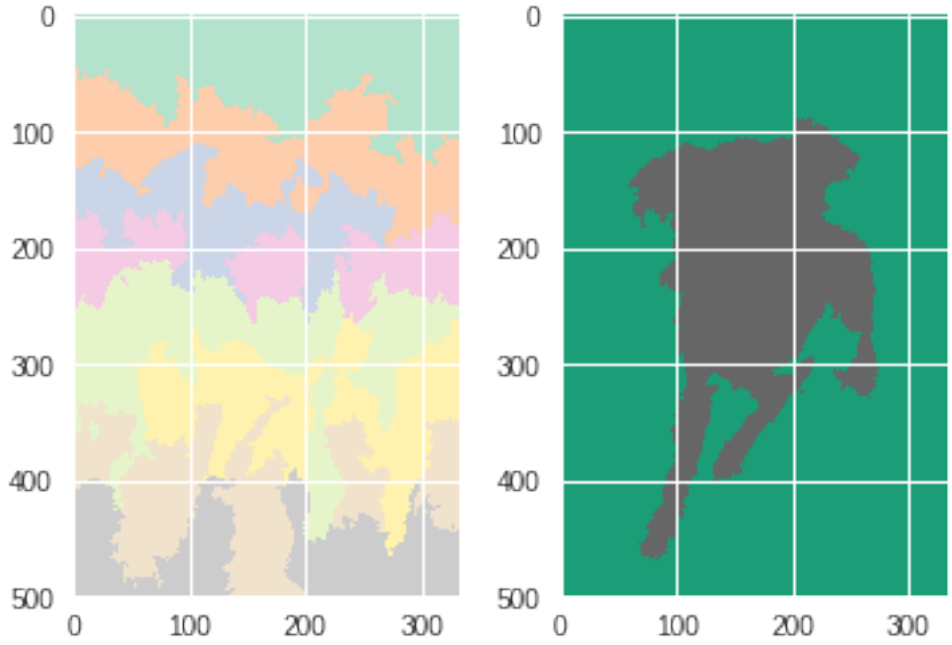
2007_006866
222



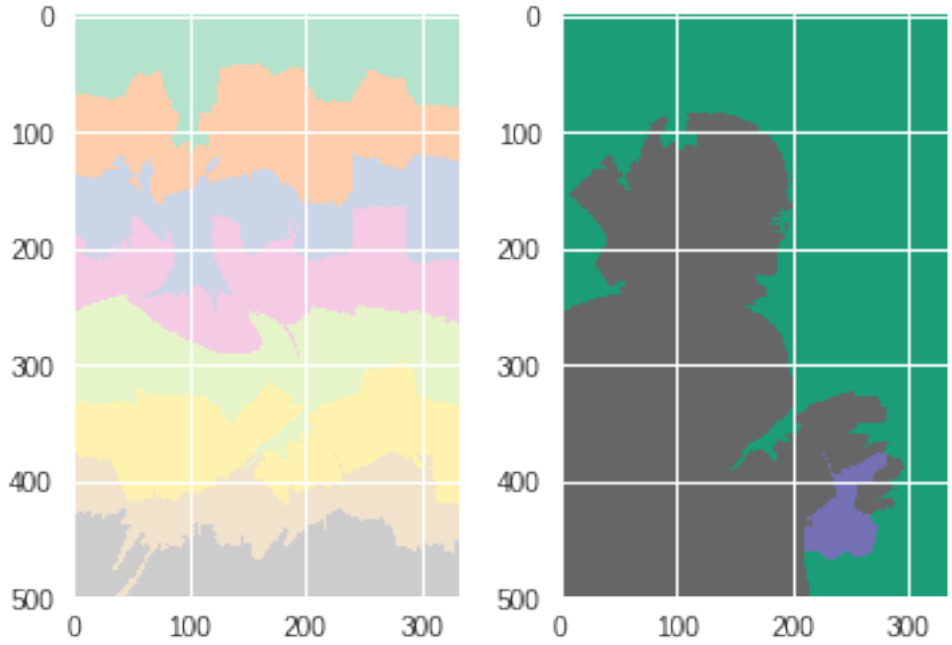
2007_006946
223



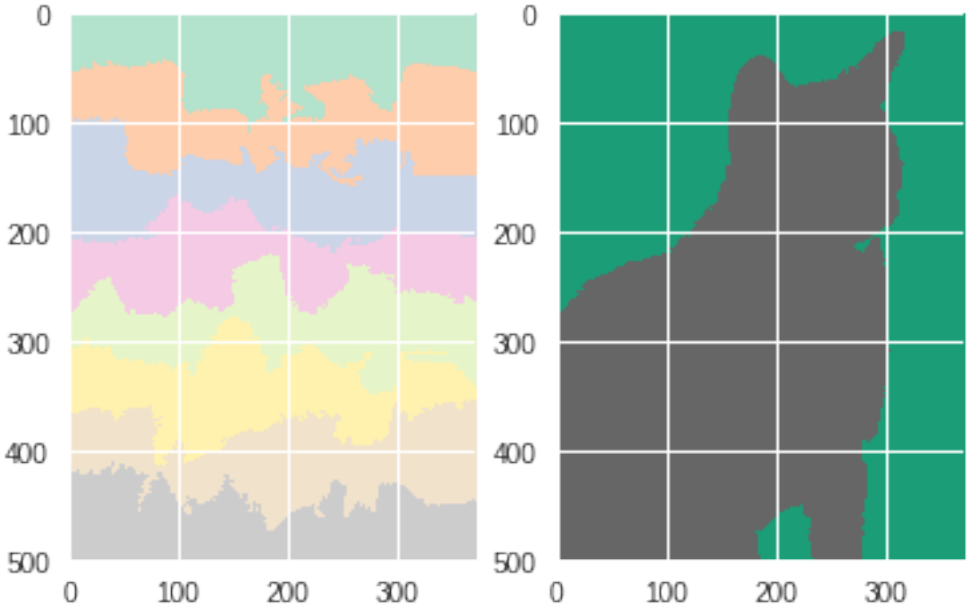
2007_007470
238



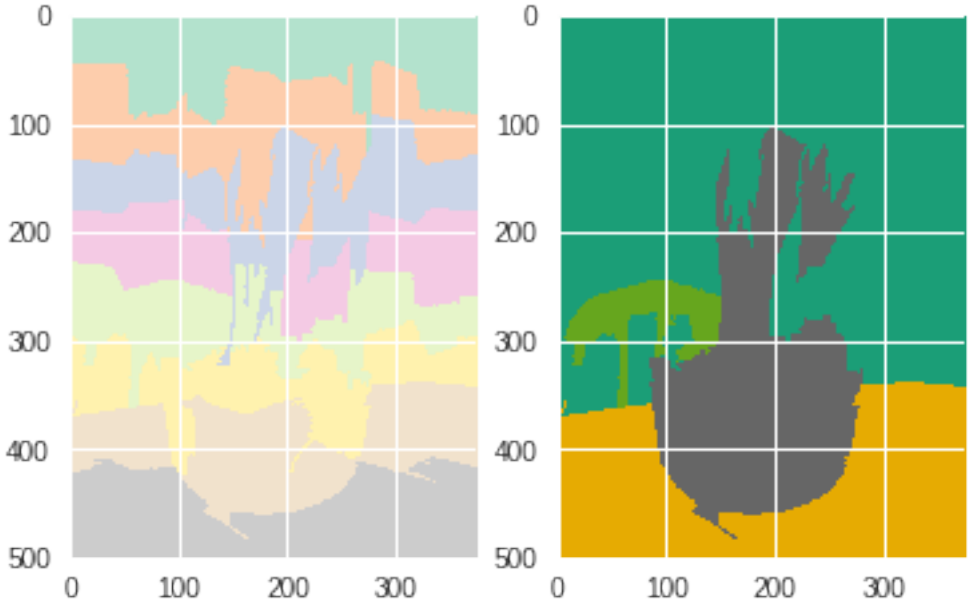
2007_008222
261



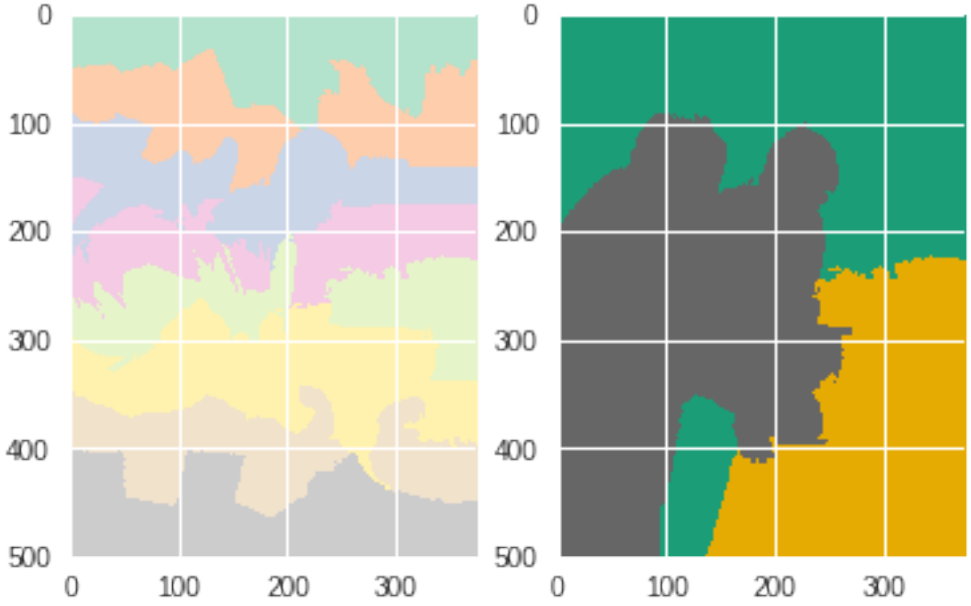
2007_009654
305



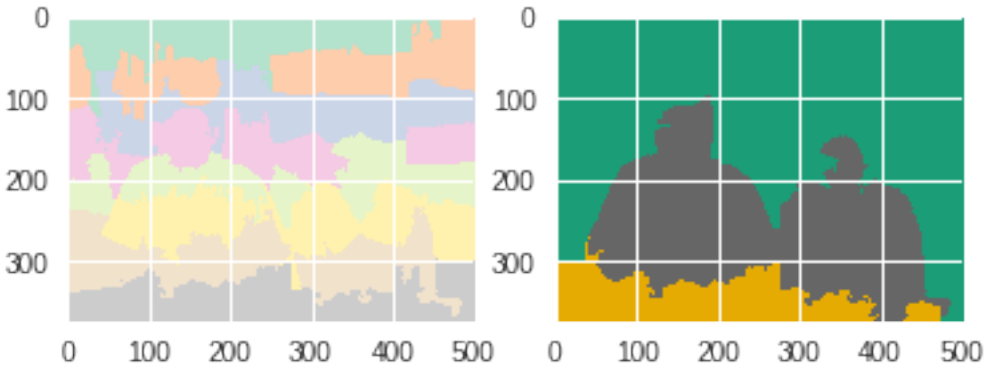
2007_009794
314



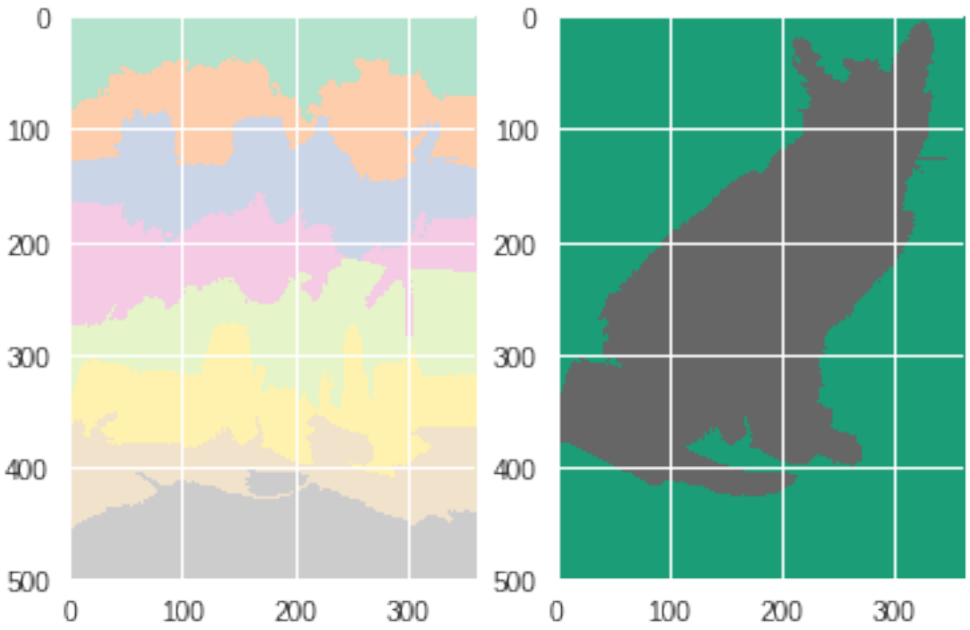
2008_002379
404



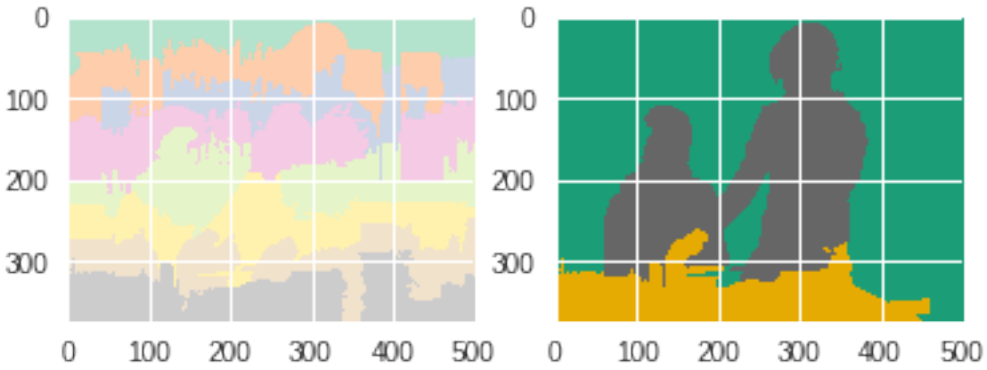
2008_004140
438



2008_006108
485



2010_000163
757



2010_000174
758

Here the first images are the superpixel inputs we gave to the CRF to segment. The CRF after calculating unary and pairwaise potentials predicted the segmentation of the image. You can see in the second image that the CRF is successfully able to detect objects in the image.

Due to the consistency in the segmentation we can come to the conlusion that the model has successfully learned to segment given the superpixels. The superpixels are calculated using traiditional methods described in **Part 1**

Glossary

Superpixels: Images patches which are constructed keeping in the intensity edges of an image. They are often better aligned than rectangular patches.

Convolutional Neural Networks(CNNs): A special variety of neural nets which makes use of learned filters in order represent images.

References

[1] Comparison of Traditional Image Segmentation Techniques and Geostatistical Threshold. https://arxiv.org/pdf/1704.06857.pdf

[2] A review of deep learning models for semantic segmentation. https://matthew.kerwin.net.au/my/honours-thesis.pdf

[3] CRFs and deep feature learning for hyperspectral image segmentation. https://arxiv.org/pdf/1711.04483.pdf

[4] Deep Learning Course by Hugo Larochelle http://www.dmi.usherb.ca/~larocheh/cours/ift725_A2014/contenu.html

[5] An Introduction to Conditional Random Fields By Charles Sutton and Andrew McCallum http://homepages.inf.ed.ac.uk/csutton/publications/crftut-fnt.pdf

[6] Differences in Naive Bayes and Logistic Regression. https://www.cs.cmu.edu/~tom/mlbook/NBayesLogReg.pdf

[7] Conditional Random Fields as Recurrent Neural Networks https://arxiv.org/pdf/1502.03240.pdf

[8] Object class segmentation using boosted conditional random fields http://service.tsi.telecom-paristech.fr/cgi-bin/valipub_download.cgi?dId=203

[9] Effective Semantic Pixel labelling with Convolutional Networks and Conditional Random Fields http://www.cvlibs.net/projects/autonomous_vision_survey/literature/Paisitkriangkrai2015CVPRWORK.pdf

[10] Higher Order Conditional Random Fields in Deep Neural Networks https://arxiv.org/pdf/1511.08119.pdf

[11] High-Resolution Image Classification Integrating Spectral-Spatial-Location Cues by Conditional Random Fields

http://ieeexplore.ieee.org/document/7486129/

[12] Conditional Random Field and Deep Feature Learning for Hyperspectral Image Segmentation
https://arxiv.org/pdf/1711.04483.pdf