

SATELLITE IMAGE **SEGMENTATION**



Guided by
Prof. Jigna Patel

Submitted by:
14bce007



DEPARTMENT OF COMPUTER ENGINEERING
Ahmedabad 382481

Satellite Image Segmentation

Major Project

Submitted in fulfillment of the requirements

For the degree of

Bachelor of Technology in Computer Engineering

By: -

Anushka Tak: 14BCE007

Guided by: -

Prof. Jigna Patel

[DEPARTMENT OF COMPUTER ENGINEERING]



DEPARTMENT OF COMPUTER ENGINEERING

Ahmedabad 382481

CERTIFICATE

This is to certify that the Minor project entitled "**Satellite Image Segmentation**" submitted by **Anushka Tak (14BCE007)**, towards the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Engineering of Nirma University is the record of work carried out by her under my supervision and guidance. In my opinion, the submitted work has reached a level required for being accepted for examination. The results embodied in this major project, to the best of my knowledge, haven't been submitted to any other university or institution for award of any degree or diploma.

Prof. Jigna Patel

Department of Computer Engg
Institute of Technology,
Nirma University, Ahmedabad

Dr. Sanjay Garg

Dept of Computer Engineering
Institute of Technology
Nirma University, Ahmedabad

ACKNOWLEDGEMENT

I would like to express my gratitude to my mentors and my colleagues without whose help this project would not have been a success. My mentor, **Prof Jigna Patel** guided me throughout the duration of the project to help me achieve a better final product. With her suggestions to make amendments, I was able to keep the project on track and reach a meaningful conclusion with a better in-depth understanding of the problem statement. Along with the final semester project titled '**Satellite Image Segmentation**', I have completed my internship for 16 weeks in Infosys Ltd, Mysore as a part of training program. This included making a project titled '**Automation of QP Setup Generation**' based on Database Management. I would like this opportunity to thank my mentor, **Nimisha Ajaykumar** to guide me for the former mentioned project. The report includes details of both the projects.

I am also thankful to Dr **Sanjay Garg**, Head, Department of Computer Science & Engineering, Institute of Technology, Nirma University, Ahmedabad for his continual kind words of encouragement and motivation throughout the Major Project. I am also thankful to **Dr Alka Mahajan**, I/c Director, Institute of Technology for his kind support in all respect during my study.

ABSTRACT

Image Segmentation is an extension of the Image Classification problem. The concept needs not only to classify the objects in an image but to do it on a per-pixel level. An image may contain multiple objects of our interest that needs to be recognized and classified. Taking images from the satellite and applying these concepts have enormous applications. Google Earth images help in Land Classification problems, Traffic Control systems, Video Surveillance and many others. Images taken by satellite capture spectrums that are of spatial and temporal resolution. There is a unique branch of study to analyse, process and classify spectral features called Spectral statistics. Processing on these spectral images, extracting their features and making use of them to our benefit is what the main aim of the project is.

The project that I've worked upon in the company consists of Automation of Question Paper Generation for assessments. The employees before joining the company undergo training for a fixed period as a part of company policy and are assessed on the basis of their skills. The stream Infrastructure Management Services, Database Management requires the candidates to perform operations on a database. The scripts created automates operations like Database Creation, Backup, Data Migration, User Creation, Listener Management, Storage Management.

CONTENTS

Certificate	
Acknowledgement	
Abstract	
Table of Contents	
Chapter 1 Introduction	7
1.1 General	
1.2 Objective	
1.3 Scope	
Chapter 2 Literature survey	8
2.1 Watershed Transformation	
2.2 Deep Neural Networks	
2.3 Analysis	
Chapter 3 Algorithms	11
3.1 Analysis	
Chapter 4 Proposed Model	13
Chapter 5 Implementation	13
Chapter 6 Conclusion	18
References	20

Chapter 1

Introduction

1.1 General

Image segmentation is processing an image into multiple partitions based on some level of similarity. The same is done by applying various algorithms that group such pixels that are similar. As the name suggests, the images that are worked upon are captured by images. A satellite produces hyperspectral images which obtain spectrum of each pixel in the image with a purpose of object detection. Satellite Image segmentation includes multiple computer vision processes such as feature extraction, object detection and classification. Based on the application, an algorithm is chosen to segment the image and further classify the pixels.

1.2 Objective

The project aims to emulate the image processing method to acquire the spectrum of every pixel in an image to make our model be able to classify into appropriate classes. The data is ultimately analyzed to distinguish regions based on its features. After Literature Survey, understanding the problem statement and analyzing various methods, I realized deep learning technique will provide the best results for the same requirement since deep learning methods include multi-layer processing with less time and better accuracy performance. Sub sampling layers give better result, by use of CNN and auto-encoders. With the increase number of auto encoders, the accuracy and the clarity of image increases. Similarly increase number of sub sampling too gives the better.

1.3 Scope

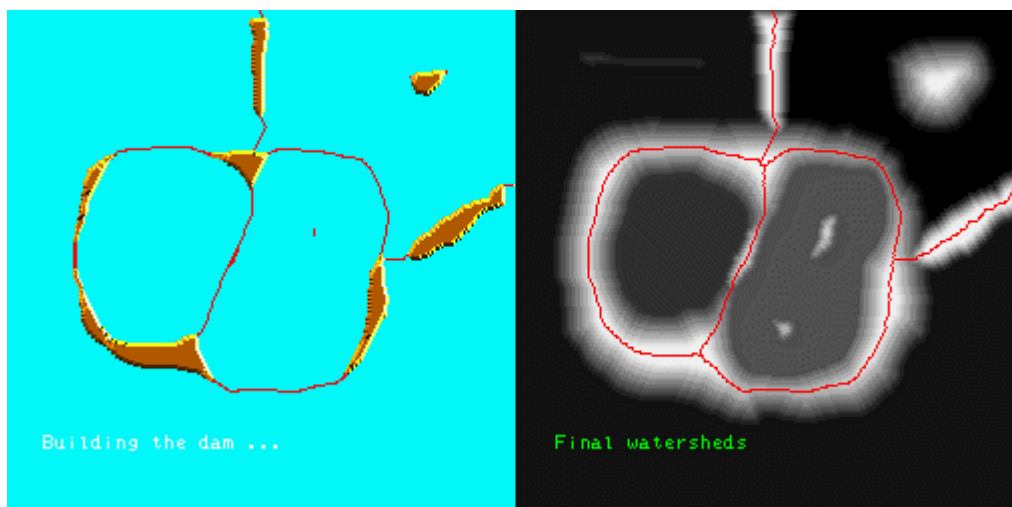
As per the theoretical concepts, the images can be better classified into regions on applying more fine and complex Deep Learning models by changing a few attributes like adding layers and changing weights while training. Once an acceptable accuracy is attained, the model can be put to good use to classify Google Map Images and for land classification problem.

Chapter 2

Literature Survey

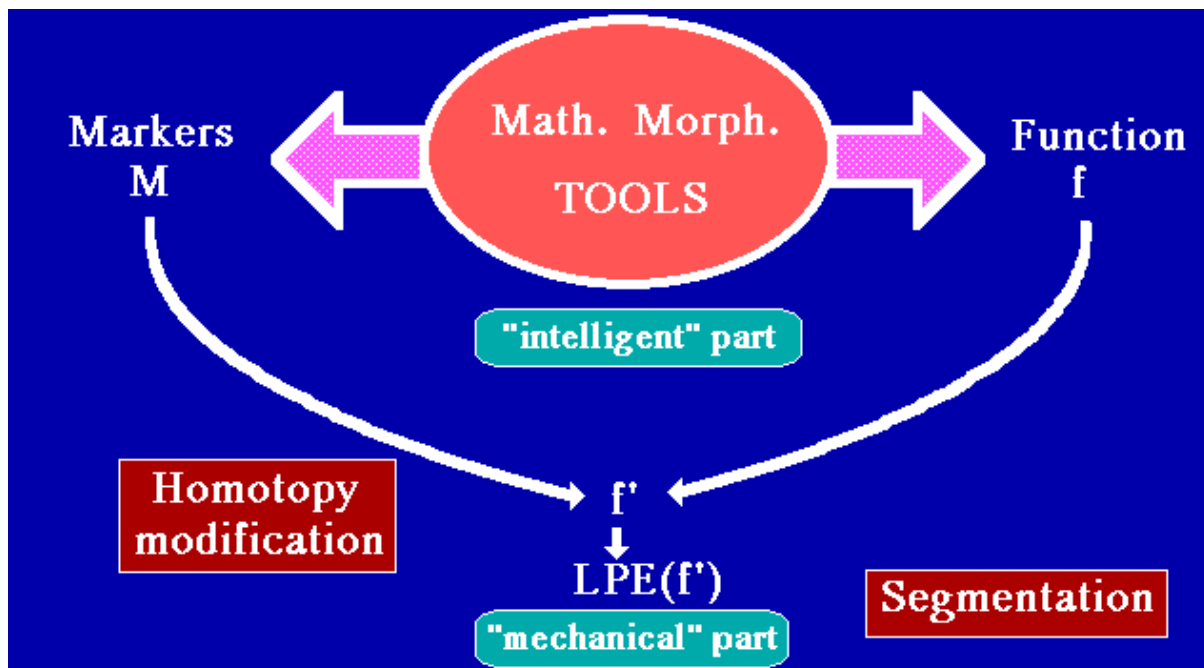
2.1 Watershed Transformation

Any greytone image can be considered as a topographic surface. If we surge this surface from its minima and, on the off chance that we keep the converging of the waters originating from various sources, we segment the picture into two distinct sets: the catchment basins and the watershed lines. We then apply this transformation to the image gradient and the rest should combine according to homogeneity.



However, it is more often than not observed that noise in the gradient image results in over segmentation. This problem could be resolved by tweaking the same algorithms with a few more details. This was Marker-controlled watershed. In this, a previously defined set of markers are referenced while flooding the topographic surface. This prevents over segmentation. Generally, contrast or gradient acts as the segmentation criterion.



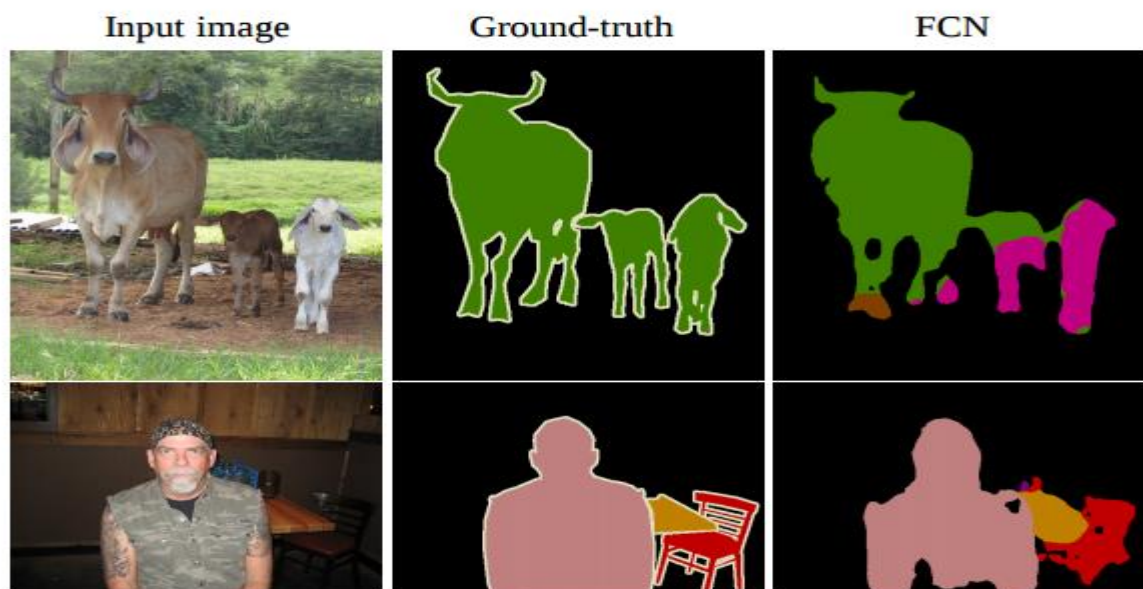


Segmentation Paradigm

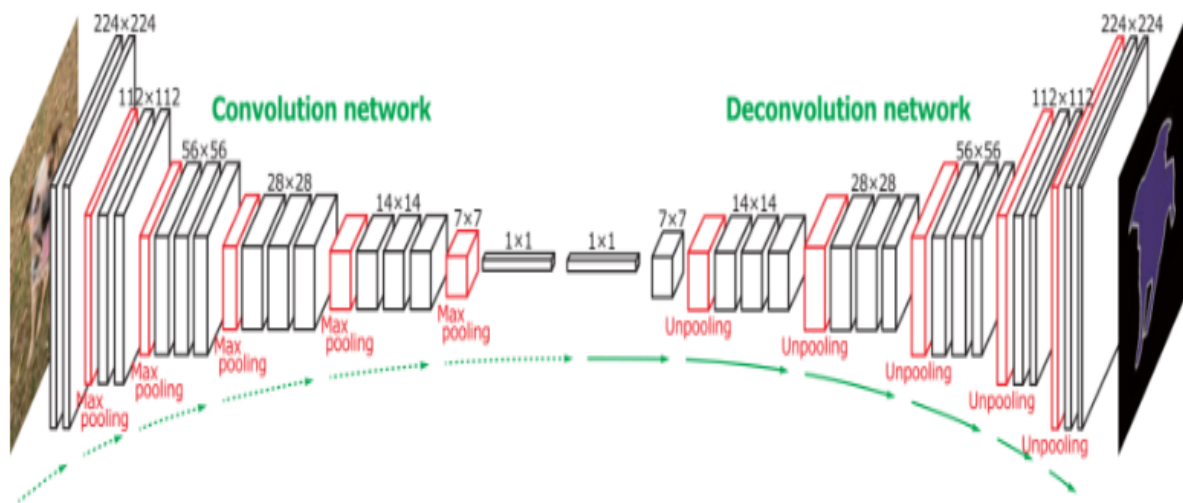
2.2 Deep Learning

An image is a collection of pixels that can serve us useful information. Deep learning is a set of algorithms that uses machine learning techniques to represent the data. As explained earlier, we need pixel level classification and for the same we use convolutional neural networks.

Basically what we want is the image below where every pixel has a label associated with it.

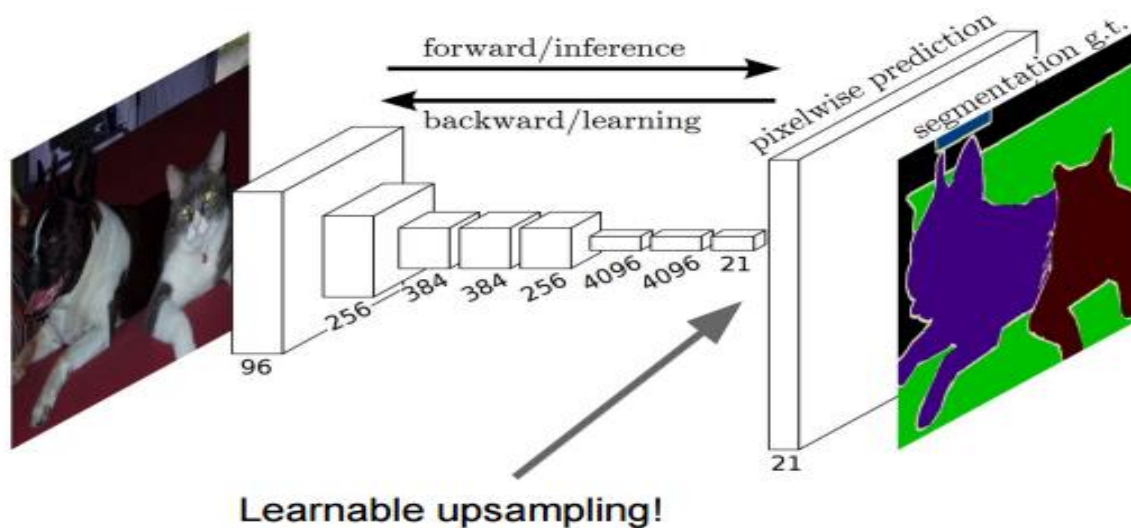


A Fully Convolutional neural network (FCN) is a normal CNN, where the last fully connected layer is substituted by another convolution layer with a large "receptive field". The idea is to capture the global context of the scene.



2.3 Analysis

The ability to predict and therefore to anticipate the future is an important attribute of intelligence. Convolutional Neural Networks prove to be more efficient since they after training on several images can use their own intelligence(learning) to give a more accurate and plausible prediction about the relevant regions in the image to be tested. There is less human intervention and considerable accuracy involved which is what makes this technique state of the art. Therefore, I proceeded with implementing the same in a model.



Chapter 3

Algorithms

3.1 Analysis

1. Watershed Transformation is based on topological interpretation. The results obtained from this algorithm are more stable. It has an edge over others since it detects continuous boundaries. Although the complex calculations of gradients make it less favorable for implementation.

2. Color based Segmentation such as K Means Clustering is a supervised Machine Learning technique which is based on division into homogeneous clusters. What makes it better than others is that the fuzzy algorithm uses partial membership therefore making it more useful for real problems. At the same time, determining membership function poses a problem.

3. Thresholding method such as Otsu's method is based on the histogram peaks of the image to find particular threshold values. This is highly dependent on peaks and spatial details are not considered. At the same time it is the simplest method of image segmentation and one doesn't need to have any prior information.

4. Unsupervised Learning with MRF and Expectation Maximization iteratively estimates the posterior probabilities and distributions of labeling when no training data is available and no estimate of segmentation model can be formed. It is popular for fitting mixture distributions. On one hand it proves to be better than the gradient descent algorithm in case of missing values. On the other, Exact MAP estimates are computationally and otherwise expensive. Its high requirement of storage makes it ideally less favorable for implementation.

5. Simulated Annealing

SA uses change in pixel label over iterations and estimates the difference in energy of each newly formed graph to the initial data. It can deal with arbitrary systems, is easy to code, finds optimal solution. Its shortcomings include it being an heuristic method, which is problem-specific or take advantage of extra information about the system, will often be better than general methods.

The fact that cost function computation may be expensive makes it less preferred.

6. Gradient Descent finds the maximum or minimum of a response surface by following the gradient, either up or down. It is a relatively simple mechanism to determine nearest optimum by few calculations. In case of large dataset or missing values, the same may prove to be little troublesome and inefficient. Another disadvantage it has is it may not be able to provide global optimum everytime. In such cases, Simulated Annealing can overcome the inefficiency.

7. Edge Detection is one of the most basic and essential algorithms. Based on discontinuity detection, it is good for images that have better contrast between

objects. Along with being essential, it is not suitable when an image contains too many edges.

8. Region growing

Based on partitioning image into homogeneous regions, it is more immune to noise and useful when it is easy to define similarity criteria. This is less favorable for implementation because it is an expensive method in terms of time and memory.

Chapter 4-5

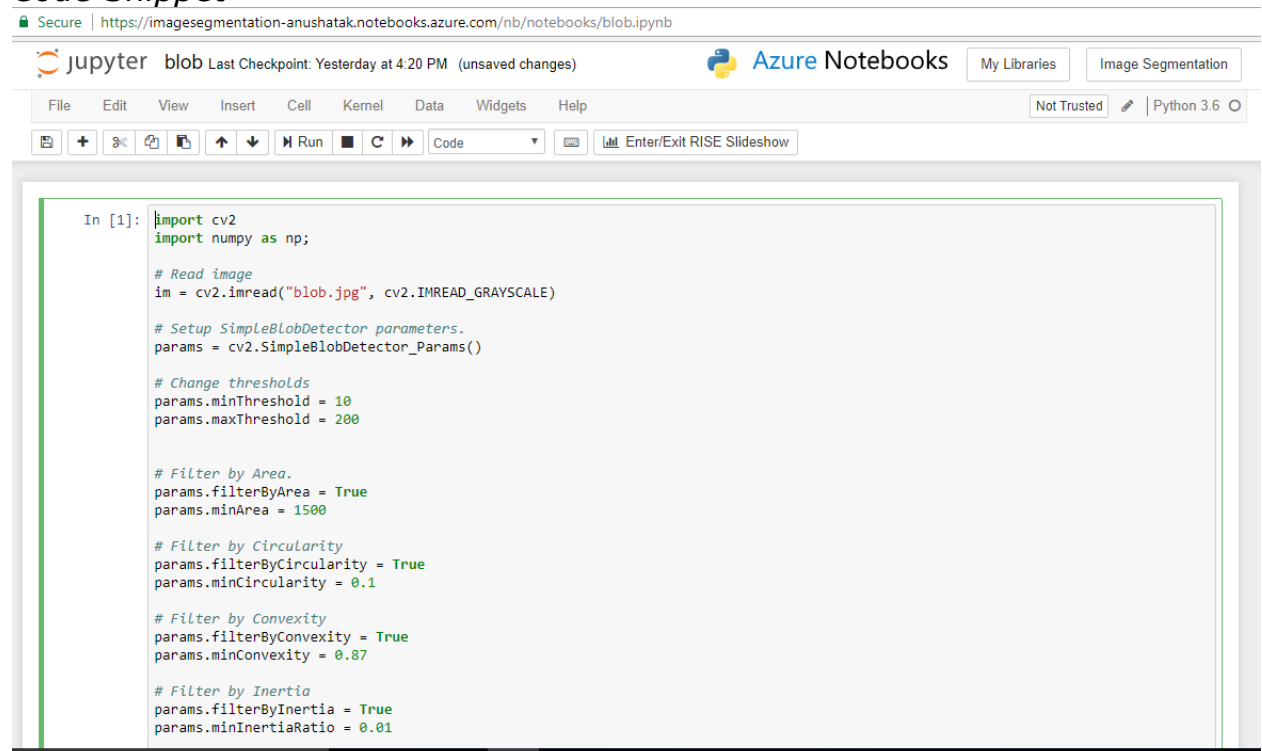
Proposed Model and Implementation

1. Watershed Transformation

Libraries Used

- Opencv and numpy and matplotlib

Code Snippet



The screenshot shows a Jupyter Notebook interface within the Azure Notebooks environment. The notebook is titled 'blob' and shows the last checkpoint from yesterday at 4:20 PM. The code is written in Python and uses OpenCV and NumPy for image processing. The code snippet is as follows:

```
In [1]: import cv2
import numpy as np;

# Read image
im = cv2.imread("blob.jpg", cv2.IMREAD_GRAYSCALE)

# Setup SimpleBlobDetector parameters.
params = cv2.SimpleBlobDetector_Params()

# Change thresholds
params.minThreshold = 10
params.maxThreshold = 200

# Filter by Area.
params.filterByArea = True
params.minArea = 1500

# Filter by Circularity
params.filterByCircularity = True
params.minCircularity = 0.1

# Filter by Convexity
params.filterByConvexity = True
params.minConvexity = 0.87

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.01
```

```
params.minConvexity = 0.07

# Filter by Inertia
params.filterByInertia = True
params.minInertiaRatio = 0.01

# Create a detector with the parameters
ver = (cv2.__version__).split('.')
if int(ver[0]) < 3 :
    detector = cv2.SimpleBlobDetector(params)
else :
    detector = cv2.SimpleBlobDetector_create(params)

# Detect blobs.
keypoints = detector.detect(im)

# Draw detected blobs as red circles.
# cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS ensures
# the size of the circle corresponds to the size of blob

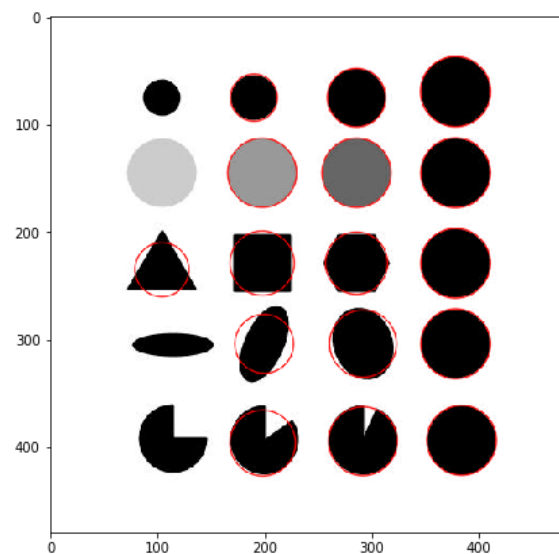
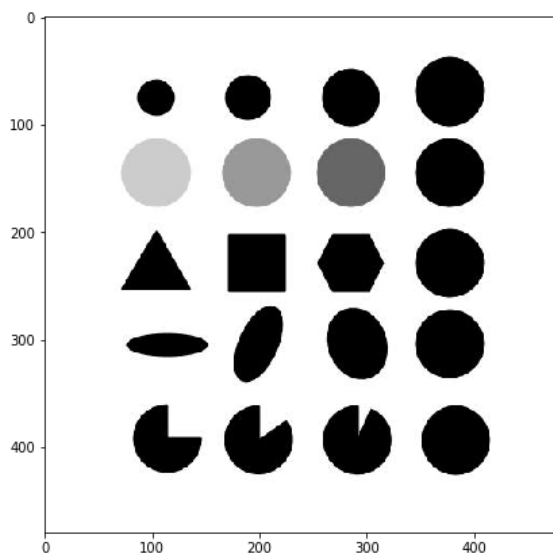
im_with_keypoints = cv2.drawKeypoints(im, keypoints, np.array([]), (0,0,255), cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)

# Show blobs
cv2.imwrite("Keypoints.jpg", im_with_keypoints)
#cv2.waitKey(0)
```

Out[1]: True

In []:

Output



2. SVM

After implementing the fundamental algorithm, to overcome the shortcomings of the algorithm a machine learning technique was used. This was making use of Support Vector Machine.

Feature Extractor Code Snippet

```

def extract_features(directory, feature_directory):
    features_all = list()
    for name in listdir(directory):
        width = 386
        height = 386
        window_size = 32
        # load an image from file
        filename = directory + '/' + name
        image = cv2.imread(filename)
        if image is None:
            continue
        image = cv2.resize(image, (width, height))
        count = 0
        for i in range(1, height - 1, window_size):
            for j in range(1, width - 1, window_size):
                temp = image.copy()
                # features = generate_feature_set(image, (j,i), window_size)
                features = generate_feature_set_surrounding(image, (j,i), window_size)
                cv2.rectangle(temp, (j, i), (j + window_size, i + window_size), (0, 0, 255), 2)
                cv2.imshow("TempImage", temp)
                is_building = 0
                while True:
                    key = cv2.waitKey(0)
                    print(key)
                    if key % 256 == 32:
                        features_filename = feature_directory + "/features_" + str(time.time())
                        with open(features_filename, 'wb') as f:
                            pickle.dump(features_all, f)
                        return
                    elif key % 256 == 27:
                        return
                    elif key % 256 == 112:
                        is_building = 1
                        count += 1
                        break
                    elif key % 256 == 113:
                        break
                print(str(features[:5]) + "---->" + str(is_building))
                features_all += [(row, is_building) for row in features]
        print("The total no. of features: " + str(len(features_all)))
        print("Total no. of positive blocks: " + str(count))
        features_filename = feature_directory + "/features_" + str(time.time())
        with open(features_filename, 'wb') as f:
            pickle.dump(features_all, f)

```

```

def load_features(directory):
    features_all = None

    for name in listdir(directory):
        if name.startswith('.'):
            continue
        print("In load_features")
        filename = directory + "/" + name
        print(filename)
        with open(filename, 'rb') as f:
            features = pickle.load(f)
            features = np.array(features)
        if features.shape[0] > 0:
            if features_all is None:
                features_all = features.copy()
            else:
                features_all = np.concatenate((features_all, features), axis = 0)
    print(np.array(features_all).shape)
    return np.array(features_all)

def segment_image_display_pixel(image, width, height, model):
    prediction_mask = np.zeros((width,height,3))
    for i in range(0, height):
        for j in range(0, width):
            feature = generate_feature_set_pixel(image,(j,i))
            prediction = model.predict([feature])[0]
            prediction_mask[j,i] = [prediction,prediction,prediction]
    prediction_mask_bwimage = prediction_mask * 100

    #cv2.imshow("Segmented",prediction_mask_bwimage)
    prediction_mask = prediction_mask * 255

    prediction_mask = prediction_mask.astype(np.uint8)
    print(prediction_mask)

    print(type(image[0,0,0]))
    print(type(prediction_mask[0,0,0]))

    masked_image = cv2.bitwise_and(image, prediction_mask)

    print(masked_image)

    cv2.imshow("Image",image)
    cv2.imshow("Masked", masked_image)
    key = cv2.waitKey(0)

```



```

def divide_train_test(dataset):
    ratio = 0.8
    SEED = 450
    total_length = len(dataset)
    print("Total length: " + str(total_length))
    train_length = int(ratio * total_length)
    print("Train length: " + str(train_length))
    random.seed(SEED)
    dataset_copy = list(dataset)
    random.shuffle(dataset_copy)
    print(len(dataset_copy[:train_length]))
    return np.array(dataset_copy[:train_length]), np.array(dataset_copy[train_length:])

def SVM_train(train_data, test_data):
    print("Train data shape: " + str(train_data.shape))
    train_features = np.reshape(train_data[:,0], (train_data.shape[0],1))
    train_target = np.reshape(train_data[:,1], (train_data.shape[0],1)).flatten()
    train_features_formatted = np.array([row[0] for row in train_features])
    print(train_features_formatted.shape)
    print(train_target.shape)

    svm_classifier = SVC()
    train_target = train_target.astype('int')
    svm_classifier.fit(train_features_formatted, train_target)
    print("Training done...")
    pickle.dump(svm_classifier, open('model3.sav', 'wb'))
    exit()
    return svm_classifier

```

Chapter 6

Conclusion

6.1 Analysis of Support Vector Machine and Further Scope

To obtain better accuracy of the entire model and classify each pixel correctly, SVM first extracts features of each pixel in the image given and trains the model learning from it. Since manual intervention is only required in training step, this proves to be a more convenient method than watershed algorithm.

For every pixel we use a sliding window protocol to run over the entire image. It extracts features (rgb values) of one window and stores it corresponding to a class value as trained by us (class A building, class B background)

Doing the same for the entire image in multiple sliding windows, we have with us a proper trained and classified classes containing multiple images of buildings and background. On this the model learns and develops the ability to predict in a new image which the building pixel is and which the background.

First we convert all the images into dimensions of squares for easy operations. So we have images of 384*384 dimensions and 32*32 are the dimensions of the sliding window.

While training, we pressed the key 'p' to indicate building class and 'q' to indicate otherwise. After running the sliding window over the entire image, we had rgb values correspond to each window with its label and location. But a better accuracy would be obtained if the window was further classified at the level of each pixel.

So while training the model now treats every pixel in a positive sliding window as a positive example. This will include many false positives and may tamper with the accuracy. But this step was required to make the model finer. Since training of each pixel will be a burdensome task, we take this approach and test on every pixel.

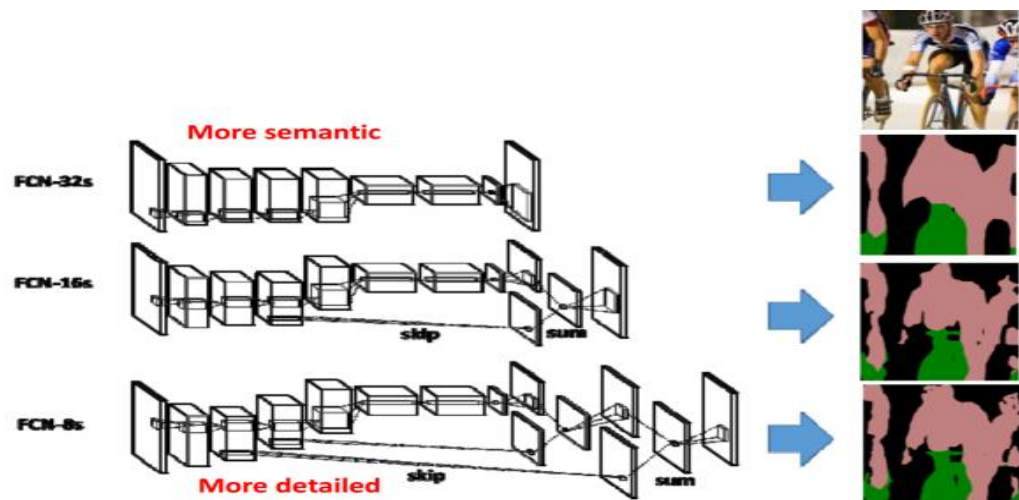
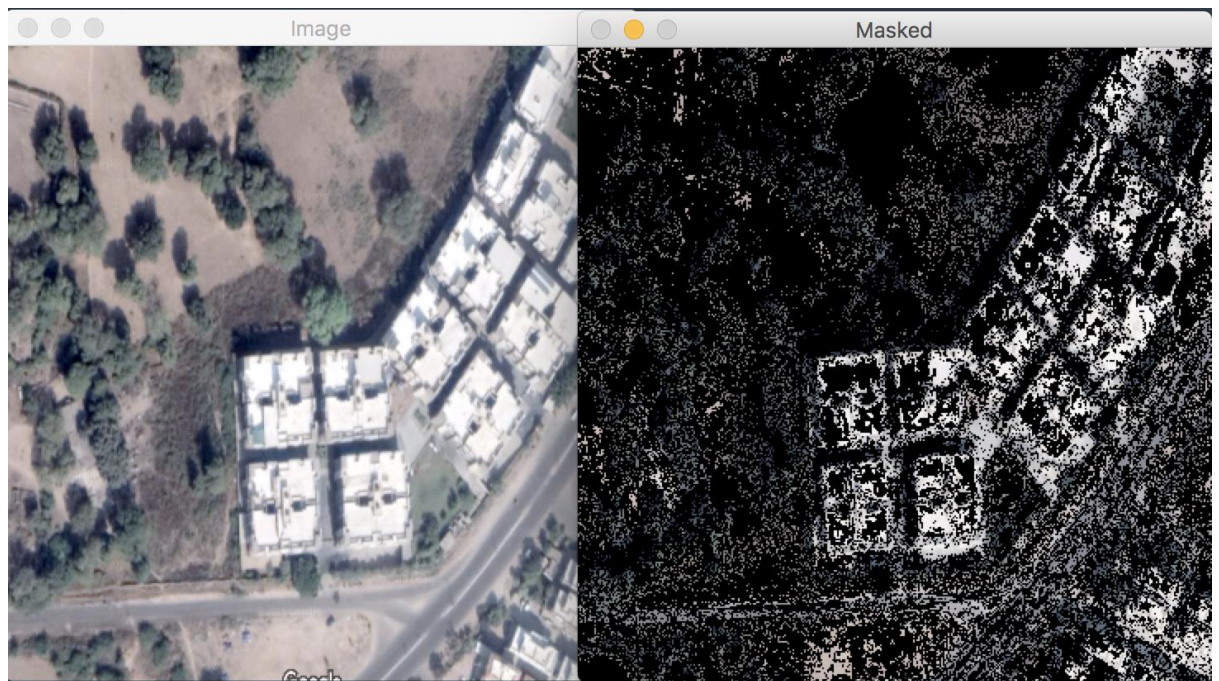
Another reason for a low accuracy is because the images that I have used for training are either jpeg or png format which are unable to capture finer attributes in an image. In large scale projects, to implement the same researchers use hyperspectral images.

In SVM we trained our model to learn on more complex features and predict. That is the same ideology we wanted to apply on our next implementation and hence we made the switch to Neural networks.

It itself will decide what features to learn on and how to classify pixels. This is a shift in our approach from image processing to Machine learning and deep learning

It can be further understood an optimum number of layers used in the neural network will lead to the best accuracy attained.

Fit training model and display image and masked image



A representation of Neural Networks

References

- *Liang-Chieh Chen , George Papandreou, Senior Member, IEEE, Iasonas Kokkinos, Member, IEEE, Kevin Murphy, and Alan L. Yuille "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs" in IEEE Transactions 2004*
- *"Predicting LandUse and Atmospheric Conditions from Amazon Rainforest Satellite Imagery" a Thesis paper by Rohisha Adke, Joe Johnson Stanford University*
- *Pauline Luc, Natalia Neverova, Camille Couprie1, Jakob Verbeek, Yann LeCun "Predicting Deeper into the Future of Semantic Segmentation" ICCV paper*
- *Jonathan Long, Evan Shelhamer, Trevor Darrell "Fully Convolutional Networks for Semantic Segmentation" arXiv.org*

Useful Links

1. <https://medium.com/nanonets/how-to-do-image-segmentation-using-deep-learning-c673cc5862ef>
2. <https://www.datacamp.com/community/tutorials/convolutional-neural-networks-python>
3. <https://blog.goodaudience.com/using-convolutional-neural-networks-for-image-segmentation-a-quick-intro-75bd68779225>