Fine Pose Estimation and Region Proposals from a Single Image

Thesis submitted in partial fulfillment
of the requirements for the degree of

MS in Computer Science
by
Research

by

Sudipto Banerjee
201307554
sudipto.banerjee@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
March 2018

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Fine Pose Estimation and Region Proposals from a Single Image" by Sudipto Banerjee, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Adviser: Prof. Anoop M. Namboodiri

To my Parents and my Advisor

Acknowledgments

I would like to express my sincere gratitude towards my advisor Dr. Anoop M. Namboodiri for his constant support and motivation throughout the duration of my stay in IIIT-H. His immense knowledge in the area of research has really benefited and I believe, will continue to benefit me in the long run. He taught me to think about the bigger goals without leaving out the significance of the minute details. Moreover, I thank him for his freedom with respect to the selection of problem statement, which in turn has helped teach me to look into any task in a more intuitive manner. All in all, his influence led to my personal and my professional development, shaping my attitude towards and research and life.

I would also like to thank all the other professors in CVIT, who have helped in any way possible - Prof. P.J.Narayanan, Prof. C.V.Jawahar, and Prof. Jawanthi Siwaswami. They have collectively managed to maintain an encouraging presence and have always been there for any student who needs any guidance. I am fortunate enough to have my fellow MS by Research friends who had kept me motivated throughout my journey in III-H. Many thanks to Amrisha Vohra, Koustav Mullick, Riddhiman Dasgupta, Koustav Ghoshal, Aniket Singh who always helped me whenever I got stuck in any problem. Many thanks to my Debarshi Dutta, Ayushi Dalmia, Satarupa Guha, Chandan Pradhan. Even though they were not part of CVIT, they have always been of great support and have made my stay in IIIT worth remembering. Without them, my life in IIIT would not have been so fun and promising.

Finally, I would like to thank my parents and my sister, who are and always have been there for me in every aspect of life. Thanks to their continuous support and motivation, it has really been easier for me to accomplish success in every venture. They have always inspired me to achieve bigger and never quit. Last but not the least, many thanks to all the people who have inspired me in any and every way possible, and wasn't acknowledged personally above.

Abstract

Understanding the precise 3D structure of an environment is one of the fundamental goals of computer vision and is challenging due to a variety of factors such as appearance variation, illumination, pose, noise, occlusion and scene clutter. A generic solution to the problem is ill-posed due to the loss of depth information during imaging. In this paper, we consider a specific but common situation, where the scene contains known objects. Given 3D models of a set of known objects and a cluttered scene image, we try to detect these objects in the image, and align 3D models to their images to find their exact pose. We develop an approach that poses this as a 3D-to-2D alignment problem. We also deal with pose estimation of 3D articulated objects in images. We evaluate our proposed method on BigBird dataset and our own tabletop dataset, and present experimental comparisons with state-of-the-art methods.

In order to find the pose of an object, we come up with a hierarchical approach whereby we first an initial estimate of the pose and thereby refine it using a robust algorithm. Obtaining the initial estimate is crucial as the refinement is entirely dependant on it. Estimating the object proposals or region proposals from an image is a well-known but difficult task, as the complexity of the problem intensifies due to the presence of object-object interaction and background clutter. We tackle the problem by coming up with a robust Convolutional Neural Network based method which learns object proposals in a supervised manner. As we need region proposals at object level, we solve the problem of instance-level semantic segmentation, where each pixel in the image is classified into one of the known classes. Moreover, two pixels are labelled differently if they belong to two different instances of the same class. We show quantitative and qualitative comparison of our proposed network models with previous approaches, and show our results on the challenging PASCAL VOC dataset.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Estimating the pose or orientation of objects in scene images is a well-known and challenging problem. Solving the problem in an efficient and accurate manner is of key importance in various computer vision tasks like understanding the 3D structure of the environment and augmented reality. Apart from that, various robotics tasks like robot manipulation, navigation and environment mapping requires knowledge of the 3D space around the robot. Knowing the pose of an object enables any automatic entity to grasp that object. In this thesis, we consider a set of scene images with considerable amount of clutter and background noise. We aim to develop a robust approach whereby we use a single image to estimate the orientation of all the known objects, without the use of any depth information.

One of the major sub problem of this task is detecting the object candidates from the image. In order to do that, we need a powerful segmentation scheme that provides the strong object hypothesis with a high precision. In order to do that, we also look into solving the problem of instance level semantic segmentation using deep neural networks. In instance level segmentation, each pixel has to be identified at instance level. In other words, other than classifying a pixel to one of the classes (like in semantic segmentation), we also need to identify the object instance it belongs to. For example, in an image containing multiple objects of the same class, semantic segmentation fails to classify the instances, although it might succeed in categorizing the pixels with high accuracy. Instance level segmentation assists in such a scenario where along with the category, we also need to obtain the instance where a pixel comes from. Given a powerful algorithm which can do this, we can extract strong object candidates from the image. Once we have the hypothesis, it becomes easier to obtain the pose from them. Figure 1.1 shows overview of the pose estimation problem. We explain the method in detail later in the thesis.

1.1 Pose Estimation

In computer vision and robotics, a typical task requires identifying an object and inferring the orientation or *pose* of that object with respect to a particular coordinate system. This information is crucial in problems such as robotic manipulation, or locomotion. Apart from that, estimating the pose of objects in an image can be used to get the 3D understanding of the scene, which in turn enables one to describe

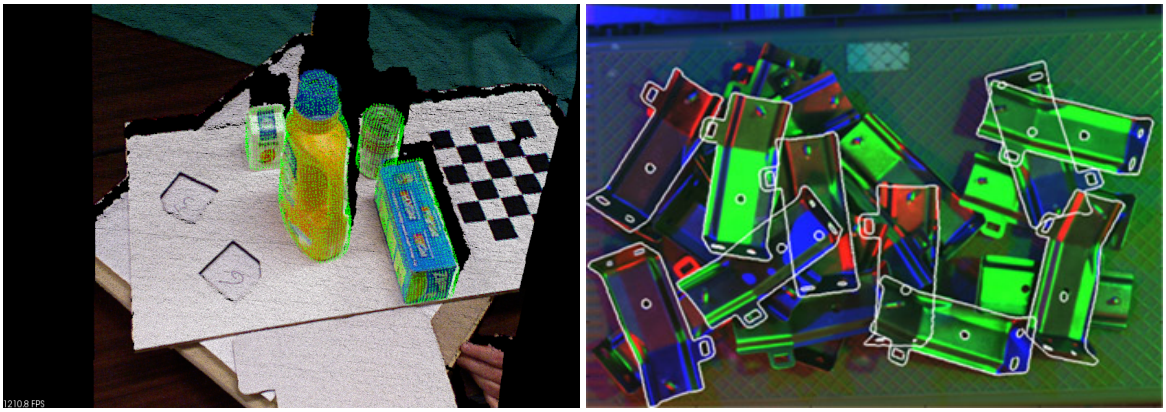


Figure 1.1 Example images showing applications of pose estimation of 3D objects in tabletop scene images. The problem deals with the orientation of known 3D objects from a single image. (Best viewed in color)

the scene in detail. The of *position* and *orientation* of an object is collectively referred to as its pose and is typically represented as its *Rotation* and *Translation* matrices or *Quaternion*. *Exterior orientation* is also used as synonyms to pose. The data from which the pose is inferred can be a single image (without or without *depth* information), collection of images (*Structure from Motion*, depth data or a sequence of images when a camera is moving relative to an object.

Depending on the application and the availability of data, there are three classes of methodologies which can be used to solve the problem of *pose estimation*:

1.1.1 Analytic or geometric approach

Given that the camera is calibrated, the mapping from 3D points in the scene and 2D points in the image is known. If also the geometry of the object is known, it means that the projected image of the object on the camera image is a well-known function of the object's pose. Once a set of *control points* on the object, typically corners or other feature points, has been identified it is then possible to solve the pose transformation from a set of equations which relate the 3D coordinates of the points with their 2D image coordinates. Algorithms that determine the pose of a point cloud with respect to another point cloud are known as *point set registration* algorithms, if the correspondences between points are not already known.

1.1.2 Genetic algorithm

If the pose of an object does not have to be computed in real-time, a genetic algorithm may be used. This approach is robust especially when the images are not perfectly calibrated. In this particular case,

the pose represent the genetic representation and the error between the projection of the object control points with the image is the *fitness function*.

1.1.3 Learning-based

These methods use machine learning-based systems which learn the mapping from 2D image features to transformation matrices. In other words, this means that a sufficiently large set of images of the object, in different poses, must be presented to the system during a learning phase. Once the learning phase is completed, the system should be able to present an estimate of the object's pose given an image of the object. The drawback of this approach is it requires a large set of feature-to-transformation correspondences is required for the model to learn.

In the previous subsections, we saw various techniques for pose estimation. In this thesis, we consider a hierarchical approach where we first compute a set of object proposals which tells us a rough estimate of the object's pose and location in the image along with its contour. Using the region proposals, we aim to refine the rough pose in an efficient manner. It has to be noted that, in order to obtain good refined pose, we need to find good object proposals in the first place. In other words, for each pixel we must assign a class label to it with a high confidence. Moreover, we should solve this problem at instance level, that is, two pixels must be labelled differently if they belong to two objects (even if they are similar objects). This problem is otherwise posed as instance-level semantic segmentation. We give a brief overview of semantic segmentation task in the next subsections.

1.2 Segmentation

Image segmentation is the process of partitioning an image into segments of pixels (also called *superpixels*). The pixels belonging to one segment specifically means that they are similar in one way or the other. The goal of segmentation is to simplify or represent the image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. Specifically, image segmentation assigns a label to every pixel in an image such that pixels with the same label share certain characteristics, such as color, intensity, or texture. Moreover, segments differ from each other with respect to the same characteristics.

Applications of image segmentation spans across a wide range of problems in computer vision. Not only is it crucial in the task of classification and detection but also important in the applications of pose estimation, medical imaging, content-based image retrieval and video surveillance. In the field of medical image processing, segmentation is applied in tasks such as locating tumors and other pathologies [40], measure tissue volumes, surgery planning and intra-surgery navigation, to name a few. Pedestrian detection, face detection, satellite imagery analysis are some of the applications in classical object

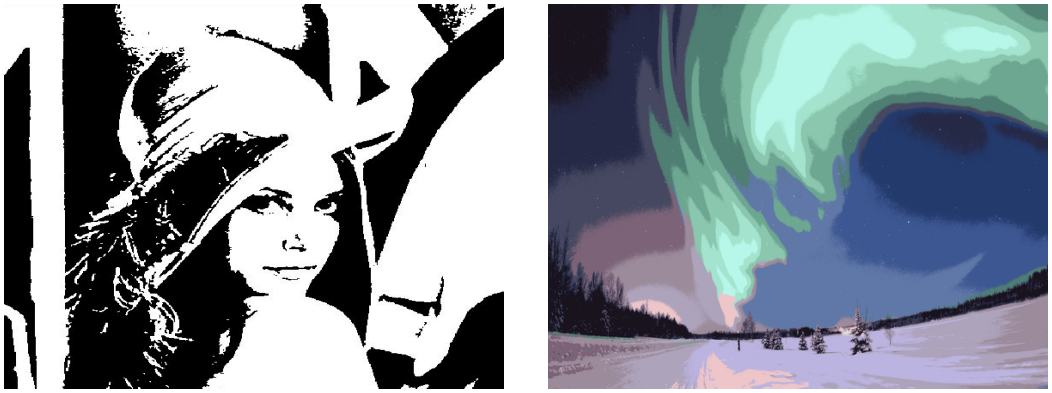


Figure 1.2 (Left) Segmentation by thresholding. Pixel values above a specified value are classified as foreground, whereas others as background. (Right) Segmentation by K-means clustering. The image shows the various segments produced using the algorithm. As is evident from the image, K-means clustering suffers from rough edges. Hence crisp boundaries are not obtained.

detection [76, 43, 12]. Segmentation has been proved to produce good results in the tasks of fingerprint recognition, document and handwriting analysis [48, 47].

Image segmentation has been a part of the computer vision community for a few decades, and hence there have been ample amount of approaches, each with its advantages and disadvantages. Generally these methods are combined with domain specific knowledge to solve. To name a few, here are some of the methods which have become popular in their own virtue:

1.2.1 Thresholding and Clustering

One of the most simple and earliest method for image binarization, *thresholding* simply employs a clipping function (*threshold*) to classify a pixel into either foreground or background. Several popular methods are used to automatically derive the threshold from the image itself like Otsu [68] or clustering, depending on an entropy function. Another widely used method for image segmentation is *K-means clustering*: An iterative algorithm which partitions an image into K (user specified or heuristic) clusters. First K cluster centers are picked and all the pixels are assigned a center, based on certain heuristic (distance or RGB values). Next, the cluster centers are recomputed from the assigned pixels. The steps continue to run iteratively until the total mean error no longer changes. K-means algorithm has proved to be a very effective method for image segmentation in early datasets. Depending on the way the cluster centers are picked in the beginning, there are other methods of clustering as well (cites). Figure 1.2 shows examples of thresholding and K-means clustering.

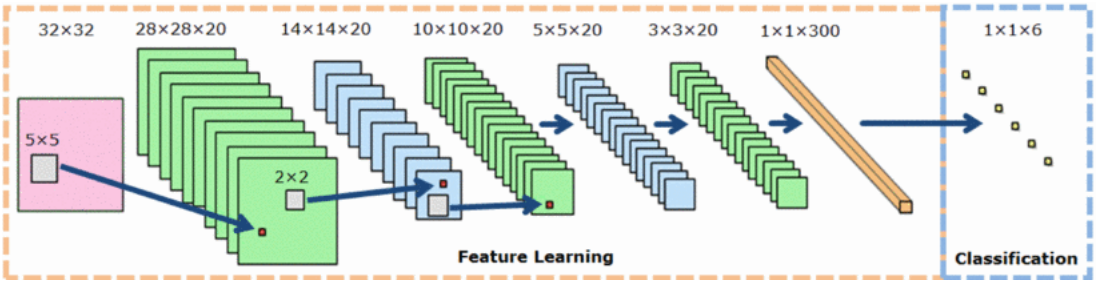


Figure 1.3 Outline of a CNN. Given an input image, filters of size 5×5 are applied on it in the first hidden layer. Likewise, the other feature maps are obtained until a fully connected layer where all neurons are connected to every other neuron. Finally a classifier classifies the 300-dimensional feature into one of the 6 classes.

1.2.2 Graphical Models

Graphical models are an unique way of solving the problem of image segmentation by considering the image as a connected graph of pixels. Technically, an image is represented as $G(V, E)$, where V represents the set of nodes (or pixels) and E denotes the set of edges between them. Thereby, the problem is reduced to partitioning the graph into segments such that the total energy (of some sort) is reduced. Graphical models leverage the features of neighbouring pixels to derive the possibility of a pixel to belong to the same class as them. *Conditional Random Fields* [36] is one of the oldest and widely used graphical method for segmentation which models the image as a set of random variables. Let X_i be the random variable associated to pixel i, which represents the label assigned. The labels can take any values from a pre-defined set $L = \{l_1, l_2, \ldots, l_L\}$. Let $\mathbf{X}$ be the vector formed by the random variables X_1, X_2, ..., X_N, where N is the number of pixels in the image. Given a graph $G = (V, E)$, where $V = \{X1, X2, ..., XN\}$, and a global observation $\mathbf{I}$ (image), the pair $(\mathbf{I}, X)$ can be modelled as a CRF characterized by a Gibbs distribution of the form $P(\mathbf{X} = x|I) = \frac{1}{Z(\mathbf{I})} exp(-E(x|\mathbf{I}))$. Here $E(x)$ is the energy of the configuration $x \in L^N$ and $Z(\mathbf{I})$ is the partitioning function [25]. In fully connected pairwise CRF model of [42], the energy of a label assignment x is given by:

$$E(x) = \sum_i \lambda_u(x_i) + \sum_{i<j} \lambda_p(x_i, x_j) \tag{1.1}$$

where the *unary* energy components $\lambda_u(x_i)$ measure the inverse likelihood (and therefore, the cost) of the pixel i taking the label x_i, and pairwise energy components $\lambda_p(x_i, x_j)$ measure the cost of assigning labels x_i, x_j to pixels i and j simultaneously. In the end, we get label assignment of the pixels which minimize the energy function given in equation 1.1.

1.2.3 Convolution Neural Networks

Artificial neural networks (ANN) are a family of models inspired by biological neural networks. In other words, they are represented to work just like the *neurons* in our brain. Neural network are very powerful in estimating and optimizing any energy function from a large number of typically unknown inputs. Each neuron is represented as a node, and are interconnected by edges. Just like the neurons in the central nervous system, the nodes facilitate the transfer of information within themselves. Usually in an ANN, the goal is to optimize an *objective function* or *loss function* from the inputs and optional groundtruth outputs. The way it does that is to *convolve* or multiply the edge weights (initialized randomly) and the value of the nodes, passing through an *activation function*, and finally getting an error. The error is in turn used to update the edge weights (termed as *backpropagation*). The entire process is repeated until convergence. Apart from the input and output nodes, there might be a layer of nodes in between them (termed as *hidden layer*). The nodes of hidden layer fetch signals from previous input signals and forward by doing the same convolution on them, to the next layer. Neural networks aim to optimize the function just like the classical gradient descent algorithm, where the function tries to reach a global maxima by updating the weights iteratively. Artificial neural networks can be compared to a new born child who tries to recognize a ball from examples of balls.

Convolutional neural networks (CNN) or *ConvNets* are very similar to ordinary neural networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. However, ConvNet architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. Input image is ”convolved” with *kernels* or *filters* of predefined size to obtain *feature maps*, which are in turn convolved in the next layer. Finally, a high dimensional feature is obtained which can then be used as a powerful tool in solving tasks like object detection, recognition, localization and classification. Figure 1.3 shows a typical CNN. CNNs are part of a more evolved machine learning technique called *Deep Learning*. The speciality of *Deep Neural Networks* (DNN) is that there are multiple hidden layers, making the network *deeper*. In a DNN, the filters learn the intricate details from an image. During the early stages of a network, the filters learn generic edges and boundaries. As the network goes deeper, unique characteristics from the image are learned. This proves how invaluable DNNs are useful in classification task. The problem of image segmentation can also be solved using DNNs by classifying each pixel in an image into one of the classes (including background). The reason DNNs have become so powerful and essential is that, we do not have to worry about hand-crafted features and feeding them to a classifier. The network does that itself, by looking for the best feature which minimizes the loss function. We will discuss more about CNNs (and DNNs) and how they solve the problem of image segmentation later.

1.3 Outline

This thesis is organized as follows. In the next chapter, we look into previous works on pose estimation and semantic segmentation. In chapter 3, we describe our proposed method on estimating pose of known objects from a single image of a cluttered scene. In chapter 4, we evaluate our proposed approach on instance specific semantic segmentation to get initial object level hypothesis. The initial object proposals thereafter act as initial seeds for pose estimation. Finally, we conclude the thesis with our contributions and some of the existing methods of our problem. We also make way for any future work that might be relevant with this problem statement.

Chapter 2

Literature Survey

In this chapter, we build a background on the task of 3D pose estimation and semantic segmentation. We study previous works on estimating the pose, given a single (or multiple) images, with (or without) depth information. We also dive into the advantages and disadvantages of each work in detail and show how our proposed approach deals with them. Object proposal techniques being a classical task in computer vision, there have been several works depending on the availability of data and application. To keep the chapter within scope, we only study the works that have been a breakthrough in their own virtue. Most of such works have been built on deep learning framework. We explain each of those approaches in detail and also evaluate cases where they fail.

2.1 3D Pose Estimation

Various approaches for pose estimation of 3D objects in 2D images have come into existence over the past decade. Finding the exact pose of a model in a scene image in an unconstrained environment is not a trivial task. Issues like lighting conditions, background clutter, quality of images, occlusion due to surronding objects, objects of different size, shape and texture and more importantly, availability of 3D models of common day to day objects, add to the complexity of the problem. In the last few years, researchers have attempted to solve the problem of category level object detection, recognition, and pose estimation [32, 1, 16, 18, 22, 31, 37, 50], which not only serves purposes in the field of computer vision, but is also significant in robotics applications like grasping, manipulation, object picking, and navigation.

The problem of object detection requires the labelling of the image pixels associated with the object. Typical Object detector gives the bounding box around the object for segmentation and annotation of the category of the object present in the image. Previous works [74, 31, 78], have emphasized on segmentation of objects in indoor environments, which is suitable for various computer vision and robotics tasks. Richtsfeld *et al.* [74], pre-segment the input image based on surface normals of objects, and estimate the surface patches by fitting NURBS (non-uniform rational B-splines). Their algorithm performs a graphcut over the graph constructed from the surface patches to get the segmentation results.

Hariharan *et al* [32], proposed a CNN based architecture which classify region proposals for simulatneous detection and segmentation of all instances of category in an image. Dai *et al* [16], address the multiple image segmentation using unsupervised learning framework for cosegmentation. They used co-sketch for alligning common objects between multiple images i.e automatic discovery of a codebook of deformable shape template shared by input images. Faktor*et al* [22] also uses composition of co-segments for segmentation of shared objects in multiple images. Li *et al* [50], proposed codemaps, a joint formulation of classification score and the local neighbor-hood of object in the image for semantic segmentation and object retrieval. Gupta*et al* [31], addressed the problem on contour detection by designing general and class specific features computed from RGB-D images for semantic segmenation of indoor images. Dong*et al* [18], used the complementary nature of object detection and segmentation. They proposed a unified framework which enforce the consistency between final detection and segmentation results. Their proposed model integrated both local and global context information to better distinguish the ambiguous samples. A annotation approach for interactive segmentation is proposed by Jain *et al* [37]. Mishra *et al* [59] introduced an algorithm for automatic segmenation of simple object of any size from its background by selecting points inside the object for segmenting an object. They deal with realtive simple objects with regular shape and size and simple texture. Aggarwal *et al* [1], estimated floor objects using a framework which utilizes generic classifier trained from appearance cues as well as floor density estimates, both trained from a variety of indoor images and adapted it results to a specific test image where they integrate appearance, position and geometric cues in an iterative framework for floor segmentation.

Recent works have also explored the task of pose estimation of known objects in scene images. In [52], Lim *et al* have estimated the pose of large scale objects like table, chair, bookcase etc. in common indoor scene images. The paper has also introduced a new dataset where they have exact texture-less 3D CAD models of objects in the scene. Here, they have used local keypoint detectors to find the candidate poses from the training data, and used a global correspondence scheme to find the refined pose of the objects in the image. Joseph *et al*, in [51] introduced a novel approach of localizing objects in an image, and estiamting their fine pose, by leveraging the geometric properties of the CAD model, and appearance information from the scene image. The work of Aubry *et al* [5], involves exemplar part-based 2D-3D alignment of CAD models in indoor scene images. The part-based correspondences has been established by representing the 3D model using a set of view-dependant mid-level visual elements, learned from rendered views. But they have only worked with 3D models of different categories of chairs.

Payet *et al*, in [70] addresses view-invariant object detection and pose estimation from a single image using image contours as basic features. They learn a sparse object model in terms of a few view-dependent shape templates and proposed a new mid-level feature, called bag of boundaries (BOB), aimed at lifting from individual edges toward their more informative summaries for identifying object boundaries amidst the background clutter. Alahari *et al*, in [2] develop a segmentation model incorporating person detection, pose estimation, as well as colour, motion, and disparity cues for pixel-wise segmentation and pose estimation of multiple people in a stereoscopic video.

Researchers have also worked pose estimation of objects in cluttered outdoor scene environments [86]. They have trained a deformable part-based model, which produces a hypothesis about the possible object locations in the scene. The final pose of the object is calculated by minimizing the distance between the object contour, and the projection of the 3D model. Previous works like [57] [56] attempt to estimate the pose of transparent objects in tabletop scenes. For instance, in [57], Lysenkov *et al* have used depth cues from Kinect sensor to recognize and estimate pose of rigid transparent objects. In [56], the authors have used Geometric Hashing [46], to match precomputed silhouettes of an object, with a test silhouette. The pose is further refined, by incorporating DCM [53] cost function into the LM-ICP [24] framework.

2.2 Proposal of Object Hypothesis

There are several works that deal with problems of establishing a hypothesis about object segments from an image. In this subsections we are going to give a brief overview of some of the object proposal techniques which are still being used in solving fundamental tasks like object detection and segmentation. With the recent advancements in Convolutional Neural Networks (CNNs), almost every task in computer vision has been accomplished with excellent accuracy. Deep Neural Networks (DNNs) have also shown tremendous improvements over graphical models [49, 29, 44, 63]. These methods rely on graphical models like Markov Random Field (MRF) or Conditional Random Field (CRF) which basically formulates an image into a graph by assigning nodal value to each pixel and weights to edges to signify relationships between pixels. Other than graphical models, DNNs hae been proven to be more effecient compared to traditional bottom-up hierarchical region proposal approaches like Selective Search [84] and Multiscale Combinatorial Grouping (MCG) [4]. In Selective Search, multiple region proposals are first obtained in order to capture objects at different image conditions and scales. Using groundtruth regions, spurious regions are classified as negative examples and proposals which have considerable overlap with groundtruth are considered as positive examples. An SVM is trained on these examples and for a given test image, an exhaustive search is done based on the trained classifier to get the most reliable region proposals. Whereas in MCG, first a normalized cut algorithm is developed to get initial region proposals. Then an efficient segmenter is proposed which takes the multi-scale information into account and computes reliable region proposals by combining multiple regions belonging to the same object. Most methods also rely on oversegmentation methods like superpixels [62]. For each such superpixel, features are extracted and a linear or non-linear classifier is trained to classify the collection of pixels (superpixels) into one of the classes, including background. The problem with oversegmentation approach is that it depends on the number and size of the superpixels, which might not be known beforehand. Highly textured images might result in a lot of small superpixels and it might be difficult to extract features from such superpixels. To address such problems, people have resorted to training their images on DNNs. Once a network is trained, an image is given as input and per pixel classification score is obtained. The class with the maximum score is considered as the pixel class. The

segmentation task can also be considered as a *tscene parsing* task. We are going to use both these terms interchangeably. Yann *et al.* [3] presented their work on scene parsing for road segmentation where they have learnt features from a DNN and also proposed a novel texture descriptor based on a learned color plane fusion to obtain maximal uniformity in road areas. Farabet *et al.* [23] proposed a DNN based method for scene labelling where they use multiscael CNN trained from raw pixels to extract dense features which encode regions of multiple scales centered around each pixel. Regions with CNN features (R-CNN) is one of the most famous works from Girshick *et al.*, [27] where they have extracted several regions at multiple scales from a generic region proposal method like Selective Search [84], or MCG [4], and classify each such region into classes. For semantic segmentation, the the final softmax layer is replaced with a spatial softmax layer which has the size equal to the size of the downsampled groundtruth. Faster variants of R-CNN have also been introduced: Fast R-CNN [26] and Faster R-CNN [73], which improves the run time by 10 times without trading off the accuracy. Moving onto more works which produce dense pixel labeling, Hariharan *et al.* [33] proposed to concatenate the computed inter-mediate feature maps within the DCNNs for pixel classification, and Dai *et al.* [14] proposed to pool the inter-mediate feature maps by region proposals. In [32], the problem of detection and segmentation was tackled simultaneously by training a network to learn a combination of R-CNN generated features per bounding box proposal and fixed length feature length from region proposals obtained from MCG. Chen *et al.* [10] proposed a relatively simpler yet efficient method for semantic segmentation where dense class predictions are obtained from a trained network using a novel *Hole algorithm*. This addition not only improved the classification accuracy but also reduced the training time. Thereafter, the class probabilties are used as unary potentials in a fully connected CRF [42]. Pairwise potentials are computed as a gaussian weighted function of pixel positions and RGB values. An altogether different approach to image segmentation is introduced by Long *et al.* [55] where they directly applies the DNN to the whole image in sliding window fashion at different scales, replacing the last fully connected layer by a convolution layer. Other than that, Long *et al.*, upsamples the feature maps from the intermediate layers and concatenates the scores to the upper layers to get denser predictions are different scales. Eigen *et al.* [20] refines the prediction results from course to fine by propagating the course results to another DNN. Zheng *et al.* [85] introduces a novel method for image segmentation where they have formulated Conditional Random Fields as Recurrent Neural Networks. Due to the limited capacity of neural networks to delineate visual objects, they use probabilistic graphical models along with CNNs to get a finer segmentation map. They trained the convolutional neural network with the CRF loss function end-to-end and have obtained promising results in the PASCAL VOC 2012 segmentation benchmark dataset. Most of the recent tasks tackling the problem of image segmentation produce dense pixel-level predictions at multiple scales. But most of the time, they fail to properly segment objects which are very small or defocused (as they are far away from the camera), or even very large objects. To tackle this problem, Noh *et al.* [67] came up with a novel proposal which trains a network which is a combination of convolution and deconvolution network. The convolution network learns features and deconvolution layer upsamples the learnt feature maps to the size of the image. Next, spatial softmax is applied to

assign probability scores on a per-pixel basis.

All these works have proposed methods for pixel-level classification for scene parsing. But what we are really interested in is object level scene parsing. In other words, two pixels are labelled differently if they are part of different objects, even if they belong to the same class. There have been several works that deal with this problem using CNNs. Dai *et al.* [15] proposed an instance aware semantic segmentation using multi-task network cascades. The idea is to form a network of cascades for multiple tasks and backpropagating the error through all the cascades. Simply put, the first task is the generic Fast R-CNN where Region of Interests (ROIs) are obtained from pooling. They take the features from this module as an input to the next module which produces masks on a per-object basis. The features from the first and second module goes into the third module which produces the segmentation map on a per-pixel basis. The error produces by the spatial softmax layer is backpropagated through all the modules in an end-to-end manner. In [71] (DeepMask), Pedro *et al.*employed a *VGG* network, which is pre-trained on Imagenet [75] dataset for image classification. In this work, dense predictions are computed on a sliding window basis at multiple scales. The loss function is a weighted sum of *objectness score* and error from semantic segmentation network. Roughly speaking, the network is bifurcated into two modules: one which calculates the probability of the patch belonging to an object and the second which is a normal segmentation network with spatial softmax at the end which gives a pixel-wise probability map. The summed error is backpropagated through the network. Dai *et al.* [13] compares their work with DeepMask by taking advantage of the powerful Fully Connected Network for semantic segmentation and applying it on the problem of instance level segmentation.

Chapter 3

Fine Pose Estimation of Known Objects from a Single Scene Image

In this chapter, we describe out proposed method on estimating pose of known 3D objects from a single scene image. Our problem does not take any depth data into account. This means the exact depth position of a *voxel* in the world coordinate system with respect to the camera is unknown beforehand. Hence, we pose our problem as an optimization problem where we try to minimize the *reprojection error* between the object proposal and its corresponding 3D model at its initial pose. We show that our approach is robust and works with both rigid and *articulated* objects. We show qualitative and quantitative comparison with the state-of-the-art approach: Seeing3DChairs [5], and prove that our method converges with a lower reprojection error. Owing to the unavailability of public datasets that serve our purpose, we test our algorithm on our own dataset: OSCAD. We describe our dataset in detail in later subsections.

3.1 Introduction

The problem of detection, segmentation and recognition of objects in natural and indoor scenes are fundamental to computer vision and has received a significant amount of attention over the past decade. Although researchers have come up with successful solutions for constrained environments [79, 80], detection and segmentation in unconstrained environments is still challenging. Various factors like illumination, background clutter and object-object interaction (*e.g.*occlusion), add to the complexity of the problem in such environments. Apart from object detection and recognition, estimating the position and orientation of objects in an image is of significant interest in tasks like 3D scene understanding and reconstruction. Figure 3.1 illustrates an overview of our problem.

Approaches to segmentation of indoor scenes include modeling the appearance of objects from images as well as their structure in RGBD data. Richtsfeld *et al.* [74], pre-segment the input image based on surface normals of objects, and perform segmentation on a graph over the estimated surface patches from NURBS. Hariharan *et al.* [32], proposed a CNN based architecture that classifies region proposals for simultaneous detection and segmentation of all instances of a category in an image. Aggarwal *et al.* [1] proposes a method specific to estimating floor regions from appearance and geometric cues.

Figure 3.1 We attempt to align 3D CAD models of objects to a single RGB tabletop image. As seen here, our proposed method is invariant to shape, size and texture of objects, and deals with scenes of varying complexity in terms of occlusion and clutter. We also deal with alignment of articulated objects (*e.g.*scissors). (Best viewed in color)

Recent works have also explored the task of pose estimation of known objects in scene images. In [52], Lim *et al.*have estimated the pose of furnitures in common indoor scene images, using user provided correspondences between the test image and candidate poses obtained from training data. Zhu *et al.* [86] attempt to estimate the pose of objects in outdoor scene images, for applications in robotics grasping. The approach of aligning CAD models of objects to their images to estimate object pose has been explored in the context of generic furnitures and chairs was attempted by Lim *et al.* [51] and Aubry *et al.* [5] respectively. They used either geomtric features of 3D models or appearance of rendered exemplars to carry out the alignment. These are closest to our work in their approach, and we compare our results with the latter.

Our major contributions include: 1) An ensemble of shape features that work well for aligning textureless 3D models, 2) A two-stage alignment scheme that is efficient and accurate and 3) An extension of the proposed approach to handle articulated objects. We demonstrate our results on a variety of tabletop objects including transparent ones and scene images with occlusion and background clutter. Note that textureless 3D models are used to generalize our proposed method to objects that are very similar in shape and size, but vary in texture. As a result, using texture or intensity based feature matching technique is not possible. Experimental results show that the proposed method outperforms the state-of-the-art method for pose estimation.

We begin by extracting object entities from the given image. Using the proposals, we prune out the outliers. From the set of *probable* object shapes, we obtain the initial hypothesis (or *pose*) using a trained

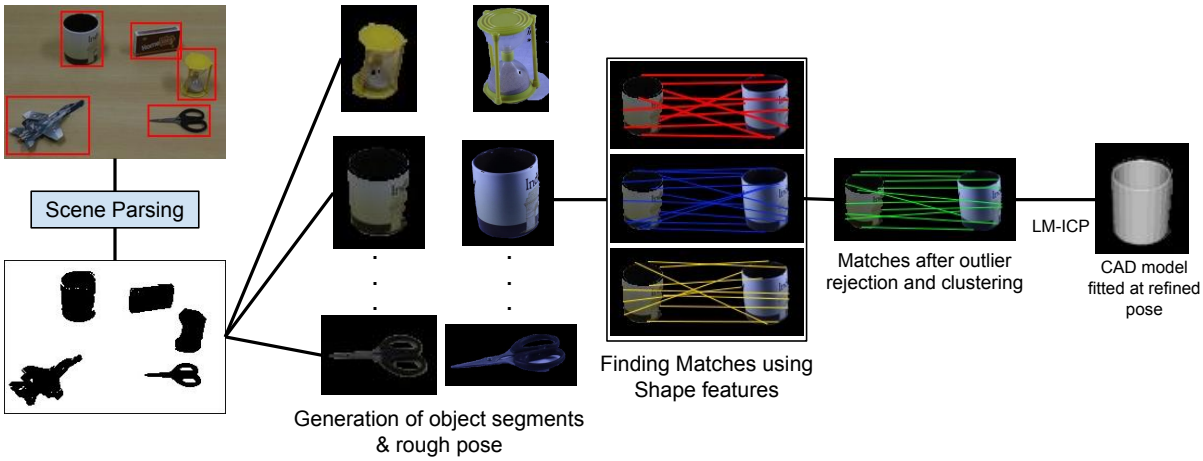


Figure 3.2 Method Outline: Input scene image is parsed to detect and segment objects using appearance and geometric cues. Initial pose is estimated from the closest matching examplar. An ensemble of shape features is used on object contour points, followed by rejection of spurious marches to obtain correspondences. Pose is refined by iteratively minimizing the reprojection error between the model and object.

model of discriminative appearance and geometric cues. Once we have the initial pose information from the object proposals, we compute three types of shape features from both the *object* and *model* contours. Point-to-point correspondences are established by feature matching and spurious matches are removed by a modified version of RANSAC, called PROSAC [11]. From the three sets of features, we obtain the purest matches by clustering similar matches. Finally, we perform Iterative Closest Point with LM optimization(LM-ICP) [24] on the set of 3D-2D correspondences to get the pose of the object. We now describe each of the aforementioned steps in detail.

3.2 Method Overview

Figure 3.2 gives an outline of our proposed method. We work with images of known objects of different shape, size, texture and poses. We parse a given indoor scene image into constituent entities of individual objects present in the scene, using a rough segmentation algorithm. The retrieved entities in the scene with their rough boundaries are used to get the respective matched codebook from the learned exemplars of individual objects. The azimuth and elevation angle captured from the matched image is then used as an initial estimate of the pose. To further refine the estimate, we use an ensemble of shape features to establish matches between object and model silhouettes. We align a 3D model of the object, by iteratively minimizing the reprojection error. We now explain the two-stage scheme of alignment: estimation of entities and rough pose, followed by refinement of the initial hypothesis.

3.2.1 Estimation of Entites and their rough Contours

For each of the object pose image in the training data, an exemplar model is learned based on appearance and geometric cues. Appearance cues capture the local shape of the object, while geometric cues are significant in cluttered scenes, where the object's color and texture are not easily computed, due to occlusions. We learn the appearance cues from the superpixel segmented image of the individual objects using simple linear iterative clustering (SLIC) superpixels. Similar to Aggarwal *et al.* [1], we use color, texture and shape cues of the superpixels and learn a GMM for separating object superpixels from that of background pixels. We learn exemplar SVM as proposed in [58] to learn the global shape and geometric cues extracted, using the HoG and CNN features.

Assuming we have N poses for each of the M models from our dataset, we learn $M \times N$ pose level GMMs and $M \times N$ exemplar SVMs. Using these discriminative classifiers, the probable objects in the test scene image are classified, and the closest possible match from the training data of all possible poses is found. GMMs give the probability estimates which acts as the unary potential, and a standard Potts model is used to get the pairwise potential. A standard implementation of MRF Grabcut [41] is used to iteratively refine the object pose estimation.

3.2.2 Fine Pose Estimation

Given the initial estimate and object classes, our goal is to find the exact pose and fit the corresponding CAD models in the test image. We define a pose by two parameters, *azimuth angle* denoted by θ, and *elevation angle*, denoted by ϕ, which are part of the spherical coordinate system. Using the object class, we fetch the corresponding CAD model and take a snapshot from the initial pose. Taking the viewpoint at the origin, we use the property of Back Face Culling to find the model contour. Assuming we have the normals to all the vertices, we discard all 3D points where the dot product of their normal and camera-to-triangle vector is greater than or equal to zero. Hence, any 3D point X lies on the edge if $(X - P).N \geq 0$, where P is the camera view-point, and N is the normal to X. Model silhouette for a pose $[R|t]$ can be obtained by finding all the 3D points which satisfy the equation:

$$(R.N)^T.(R.X + t) = N^T.(X + R^T t) = 0 \tag{3.1}$$

where R and t are the $Rotation$ matrix and $Translation$ vector respectively.

In order to find matches between the model and object contour, we use shape features which best describe the appearance and geometry of the object. We apply Procruste's shape analysis [19] to bring both contours to the same scale. Correspondences obtained using each of the features individually are not good enough to align the model. Hence, we combine the features to get a reduced set of good matches. The final set of correspondences is then used to find the true pose of the object by iteratively minimizing the reprojection error until convergence. We now explain the approach in detail.

3.2.2.1 Local shape information using a Buffer of HOG features

Our goal is to develop features which are robust enough to capture 3D-to-2D correspondences between the model and object contour properly. Finding accurate matches is crucial to avoid the reprojection error function getting stuck at a local minima in the pose space. It has been observed from [34, 52, 5] that learning an LDA classifier on HOG features produce as good matches as learning a computationally expensive SVM classifier. In a cluttered environment, it is difficult to obtain precise segments from the image. Hence, to make our method robust, we create a buffer of HOG features of K neighbouring points on either side, along the contour. Doing so captures the local shape information with respect to a point. This step prevents redundant matches due to lack of additional neighbouring information. We learn a binary LDA classifier taking one feature as positive set and others as negative set. Assuming each object sample point as an individual class, we obtain N binary classifiers. Each model point is matched to one of the classes which minimizes the classification score. More formally, the matching score between p-th object point and q-th model point is defined as

$$S(p,q) = \min_{p} w_p^T x_q \tag{3.2}$$

where x_q is feature vector of the q-th model point, w_p is the weight vector learned by LDA classifier for object point p. HOG features are computed on 8×8 cell and 2×2 block. Number of orientation bins is fixed at 9. The HOG features are normalized such that $\sum_{i=1}^{|H|} h_i = 1$, where $h_i \in H$, and $|H|$ is the number of points in the contour. Value of K is fixed at 10 in our experiments.

3.2.2.2 Shape context features

Modelling our task as a shape matching problem, we use Mori *et al.*'s [61] work for object detection using shape contexts. The key idea of the feature is to capture the distribution of all points with respect to a reference point. For each point on the contour, a normalized k-bin log-polar histogram of the relative coordinates of the remaining points is computed as,

$$h_i(k) = \#\{q \neq p_i : (q - p_i) \in bin(k)\} \tag{3.3}$$

which is defined as the shape context of p_i. Cost of matching a model point to an object point is taken as a weighted sum of shape context cost and appearance cost:

$$C = \beta C_S + (1 - \beta) C_A \tag{3.4}$$

where C_S is the χ^2 distance between the shape contexts of the point pairs, and C_A is the tangle angle dissimilarity between them. Here, $0 \leq C_S, C_A \leq 1$, and β is the weighting parameter. To find a set of point-to-point correspondences between the two shapes which minimize the total cost, the problem can be expressed as a standard linear assignment program with the objective function

$$\text{minimize} \sum_{p_i \in O} \sum_{q_j \in M} C(i,j) x_{ij} \tag{3.5}$$

subject to constraints, $\sum_{q_j \in M} x_{ij} = 1$ for $p_i \in O$, $\sum_{p_i \in O} x_{ij} = 1$ for $q_j \in M$ and $x_{ij} \geq 0$ for $p_i, q_j \in O, M$. $C(i,j)$ is an $N \times N$ cost matrix, where N is the number of sample points from the contours. The variable x_{ij} is 1 when sample point p_i from model contour M corresponds to sample point q_j from object contour O and 0 otherwise. Modelling this as a bipartite matching problem, Mori *et al* [7] solved the optimization using $Hungarian$ assignment [69]. However, this takes $O(N^3)$ time for matching, which is computationally expensive if the number of samples is high. Hence, we use Jonker-Volgenant algorithm [38], because of its robustness, and efficiency in terms of computation speed.

3.2.2.3 Chamfer Matching

The problem of establishing correspondences across the model and object sample points can be formulated as finding the $closest$ object point, for each model point. Here, the distance metric is the Chamfer Distance [9] between the two points. The problem with previous chamfer matching methods like DCM [53], OCM [77] is that they cannot deal with the local deformations of the object. Hence, they are prone to wrong matches. Hence, following the work of Nguyen *et al.* [66], we formulate the matching task as a maximum of a posteriori (MAP) problem. In other words, for each model point m, finding the corresponding object point o is equivalent to solving the following

$$p(o|m) = \kappa \max_{o} e^{-\alpha d(m,o)} \tag{3.6}$$

where κ and α are positive parameters. $p(o|m)$ is the probability of having an object point o corresponding to model point m and, $d(m,o)$ denotes the distance between m and o given by,

$$d(m,o) = \sqrt{\|m-o\|_2 + \lambda[f(a_m, a_o)]^2} \tag{3.7}$$

where $\|m-o\|$ is the l_2 norm of distance, and $f(m,o)$ is error of orientation of m and o. λ is a weight factor. Similar to [66], Variational Mean Field approach is used to solve the MAP problem. Finally, we obtain a set of point-to-point matches.

3.2.2.4 Ensemble of shape features

The set of correspondences obtained from each of the shape matching methods, might contain outliers. The problem intensifies when the an object is occluded, and we have a partial object mask. Figure 3.3 shows such a scenario. We use a faster variant of RANSAC: Progressive Sample Consensus (PROSAC) [11], to remove the outliers. The use of PROSAC drastically reduces the computation time for obtaining the inliers. We use an ensemble of these shape features to get a reduced set of $pure$ matches, which contain the least number of outliers. In order to do that, we partition the sampled points on the object contour into K clusters. We find that, clustering using k-means algorithm [54] yields satisfactory results. The intuition is that, for a model point, if the shape features are good enough to capture its local appearance, they should all map to object points which are *close* to each other. More precisely,

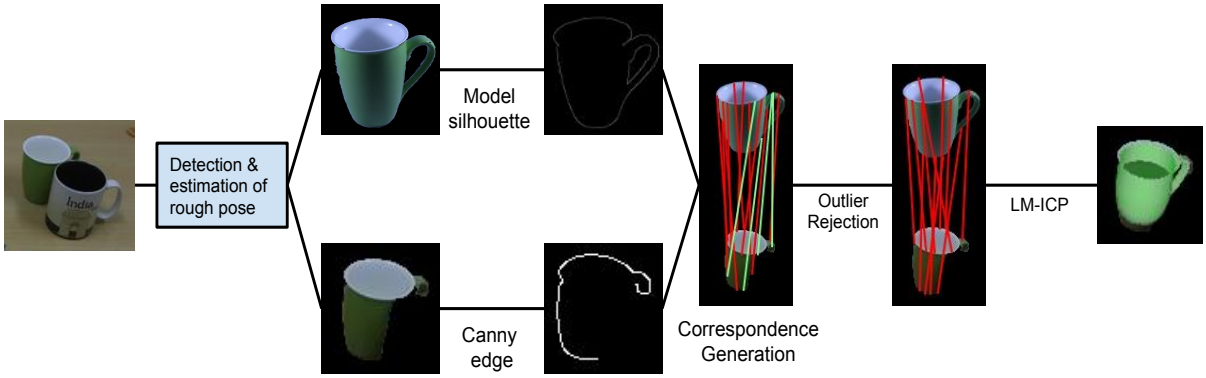


Figure 3.3 Figure shows the pipeline for alignment of an occluded object. Correspondences are generated using ensemble of shape features, from object and model contour. Only a subset of matches are shown here. Candidate matches are acquired by outlier rejection using PROSAC. Incorrect matches are denoted in green. Pose is refined using LM-ICP using inlier matches. (Best viewed in color)

for a model point, we assert that object point matches obtained from the shape features, are similar if they belong to the same cluster. We finally obtain the *common* 3D-to-2D correspondence, where the model point is matched with the object point which has the minimum reprojection error with that model vertex. We observe that using the reduced set of matches, our algorithm converges with significantly less number of iterations with minimum error. Figure 3.4 shows the comparison. We plot the error function with respect to azimuth and elevation angle for an example, and find that it is much smoother, when using ensemble of features. Hence, a lesser likelihood of getting stuck at a local minima. Finally, given a set of pure 3D-to-2D correspondences, we use Iterative Closest Point algorithm using Levenberg-Marquardt optimization (LM-ICP) [24] to obtain the transformation matrix which aligns the model to the object.

3.2.3 Problem with articulated objects

Our proposed approach also deals with articulated models. If we are given a 3D model with predefined articulation points, and the information about their degrees of freedom, we can use this data to deform the model to best fit the model in the 2D image. Aubry *et al.*[6, 5] starts off by sampling a set of cameras around the 3D model. Then, visual discriminative elements are extracted from the synthesized views of the model, and matched with the test image, to obtain the best alignment. One drawback of this approach is that, they cannot handle articulated objects. Any change in the model influenced by the articulation points is hard to be accounted for by the synthesized views, because the model is assumed to be rigid and constant. Moreover, even if another level of complexity is added by synthesizing views by varying the model parameters controlled by the articulation points, it will further increase the computation time of the problem. The proposed method avoids the aforementioned issues, by also taking into

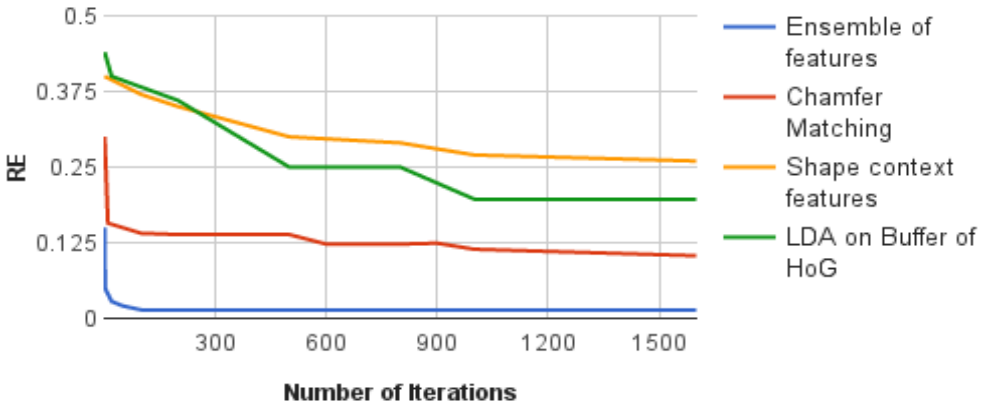


Figure 3.4 Plot showing Reprojection Error (RE) vs number of iterations required to align a model to an object. Note that, using correspondences obtained from ensemble of features, RE reaches convergence with lower number of iterations. (Best viewed in color)

account transformation of the model due to the articulation points. Assuming the deformation caused by each articulation points is represented by its own transformation matrix, we modify the LM-ICP objective function as follows

$$\sum_{i=1}^{N_d} \min_j \epsilon^2(|d_i - KTA_j m_j|) \tag{3.8}$$

$$A_j = \prod_{k=1}^{N_a} (\ x_{jk} T_k + (1 - x_{jk}) I\ ) \tag{3.9}$$

where d_i and m_j are the i-th and j-th data and model point respectively. N_d and N_a are the number of data points and number of possible ways of deformations caused by the articulation points. K is the camera matrix. T is the global alignment transformation matrix. T_k is the transformation caused by the k-th articulation point. I is the identity matrix. x_{jk} is defined as

$$x_{jk} = \begin{cases} 1, & \text{if } k\text{-th articulation point affects } j\text{-th model point} \\ 0, & \text{otherwise} \end{cases} \tag{3.10}$$

where $\epsilon^2(|x|)$ is an error function defined by $\epsilon^2(|x|) = \|x\|^2$. The idea is to find a model point corresponding to each object contour point, such that applying the deformation and the global transformation on the model point minimizes the total reprojection error.

3.3 Dataset Overview

In order to facilitate the experiments on detection, localization and segmentation tasks, over the years researchers have established benchmark image datasets such as Caltech-101 [30], MIT CSAIL object and scene database [83], PASCAL [21], NYU Depth datasets [65], ImageNet [17]. This has led to a significant advancement in robotics applications which requires interation among objects in an environment, such as navigation, visual search, object grasping and object manipulation. To further facilitate research, BigBIRD [82], Washington RGB-D [8] datasets were introduced recently. Objects and scenes in these datasets has hierarchical category structure and target issues related to both object instance and category level recognition. Furthermore, to address problems of 3D reconstruction and fine pose estimation, datasets such as SUN3D [72], Google 3D warehouse [28], IKEA 3D models [60] are released.

In view of the importance of such datasets, we introduce a high quality, large scale RGB-D object dataset comprising of a broad range of household objects used in daily chores Figure 1. The dataset covers a wide spectrum of different categories of objects according to shape, texture and instance, spanning multiple poses for both azimuth and elevation with resepct to camera sensor. In addition, we also provide scene images capturing scenarios ranging from sparsely clutter to densely clutter table top images, under unconstrained environment. We also introduce an additional 3D CAD models and a ground truth from stuctured-light patterns of all the objects in the dataset.

The main motivation behind introducting our own dataset is that most of the benchmark datasets like Caltech-101 and MIT CSAIL object and scene dataset are collection of images from internet sources like Flickr, Google, Picsearch and Bing. Hence, they lack variation in terms of different pose and multiple viewpoints. Also they lack exact viewpoint or pose information, which makes them unusable in our task. Washington RGB-D Dataset consists of relatively simple freestanding categorical objects and most of the objects are of similar shape and size, not covering wide range of irregular objects. BigBIRD, though provided pose information for every image or point cloud, they have not provided table top scene where these objects occur and can be used for scene understanding and robotic manipulation. In addition to this, none of the datasets provide 3D CAD models, except for SUN3D and IKEA 3D models. But they generally have models of large scale objects like chair, table, sofa and other household furnitures. These datasets lack in 3D models for objects of daily chores which occur in table tops and can be used for robotic manipulation applications.

Our dataset currently consists of 200 household items randing across various categories. We provide RGB and depth images for every 10 rotation, for each object. We capture images at five elevation angles of 0, 15, 30, 45 and 60. We capture one RGB and one depth from an angle of 90, because any change in pose of the object around the azimuth angle at an elevation of 90, can be accounted by a simple 2D rotation of the pose. So we have 36 images per elevation angle, and 5 elevation angles per object, and 1 image at 90, giving a total of 181 RGB-D images per object. In our pose estimation task, we acquire the pose information from a single image without the help of any depth cues. Hence, we do not use the depth data. We only work with the RGB images.

	LDA-BHOG	Shape Context	Chamfer Distance	Ensemble of features
RE	0.197	0.271	0.089	0.015
RMSE	0.243	0.336	0.097	0.026
Time taken(in sec)	12.87	10.49	2.46	28.56

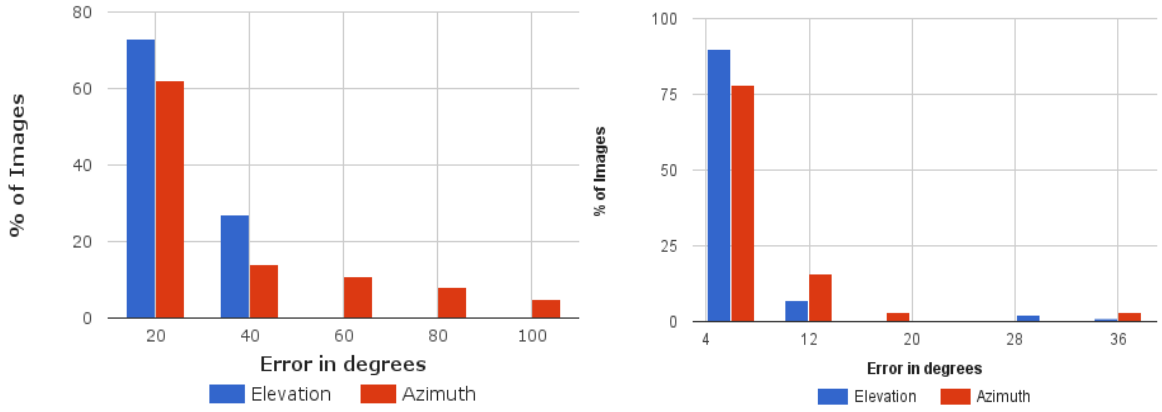
Table 3.1 Table showing the Reprojection Error(RE), Root Mean Square Error(RMSE), and time taken for estimating pose of object with each of the shape features individually, and using ensemble of features. The Chamfer Distance algorithm was implemented in C++.

3.4 Experimental Results and Analysis

In order to evaluate our proposed method, we use 3DMOS: a 3D model, object and scene dataset of tabletop objects. Existing datasets like [45] and [82], have been primarily used for object detection and recognition. Hence, they do not have scene images. For tackling problems like 3D scene understanding and modelling, we require 3D models, as well as scenes. Aubrey *et al.* [5] and Lim *et al.* [52] introduced datasets where they provide object image dataset as well as scene images. However, the main difficulty in using the aforementioned datasets is that [5] work with different classes of chairs only, while [52] consider common indoor objects like bookcase, bed etc. The objects in our dataset are common household items generally found on home or office tabletops. Our dataset contains images of objects captured at regular intervals, and scene images of tabletop objects of varying complexity. We also provide 3D CAD models corresponding to each object.[1] The models have been acquired from Google 3D Warehouse [28]. We run the proposed approach on these scene images and the results are shown in Figure 3.6. We compare our results with state-of-the-art Seeing 3D Chairs [5] (S3DC). We evaluate our proposed method on the BigBird dataset [82] as well as on 3DMOS. We experiment with 75 objects from BigBird, consisting of 600 images per object and 50 objects from 3DMOS, each having 90 images. We also run our approach on cluttered tabletop scene images from our dataset. We also compare our results with Seeing 3D Chairs [5] (S3DC). In S3DC, the problem is solved by matching the closest rendered image based on discriminative features. Hence, the refined pose is accurate to the interval of successive viewpoints (in degrees).

We learn the exemplar model on our training data, and obtain a learned classifier for each object sample. A test image is classified into the object class corresponding to the sample classifier yielding the maximum score. After refinement of the estimate, the Mean Absolute Error (MAE) of the final pose is measured. We show quantitative results of our proposed method in Figure 3.5. As observed, our proposed method is able to provide an initial estimate within 20° of groundtruth for 76% of the test data. Pose is further refined till 6° of groundtruth. On the other hand, S3DC has an average azimuth error of 20° for 87% of their examples. As evident from figure $3.5(c)$, we further refine error till 6° of groundtruth for 83% of our examples. Given the groundtruth pose vector, (c_x, c_y, c_z) and predicted

[1]Objects, CAD models and scene images from our dataset are shown in the supplementary material submitted with this paper.



(a) (b)

Figure 3.5 (a) Error in rough estimate and (b) Error in pose after fine alignment. In order to evaluate the fine alignment error, we use those examples for which the initial estimate is within 20°of groundtruth. (Best viewed in color)

pose vector, (c'_x, c'_y, c'_z), the pose error is taken as the euclidean distance between them. Table 3.2 gives results for some of the objects from our dataset.

	Transparent Bottle		Sand Clock		Pen Stand		Scissors		Spectacles Case	
	Accuracy(%)	MPE	Accuracy(%)	MPE	Accuracy(%)	MPE	Accuracy(%)	MPE	Accuracy(%)	MPE
S3DC	31.11	0.147	83.33	0.086	**91.11**	0.093	6.67	0.4	44.44	**0.016**
Ours	**83.52**	**0.0016**	**86.81**	**0.033**	83.52	**0.002**	**73.63**	**0.086**	**54.65**	0.121

Table 3.2 We compare the classification accuracy and mean pose error (MPE) with S3DC, for some of the objects from our dataset with varying complexity in terms of shape, size and material.

We compare the classification accuracy and mean pose errors with S3DC in table 3.3. Figure 3.8 show the qualitative comparisons. We also evaluate our approach on tabletop scene images. Using the pose of objects from a single RGB scene image, we fit their corresponding CAD models. The method fails when the initial estimate is far away from groundtruth. Failure cases are shown in figure 3.8.

While evaluating our approach on tabletop scene images, we find the pose of objects from a single RGB scene image. Using the pose, we fit CAD models of corresponding objects. Table 3.1 shows the errors and time taken at each step of our pipeline. While calculating the reprojection error and RMSE, the points have been normalized such that $0 \leq p, q \leq 1$, where p and q are model and object points respectively.

We run experiments on scene images of various complexity. In cases where one object is occluded by another, we obtain the segmented mask for each object. After determining the object in front, we subtract the part of mask that blocks the occluded object. Hence, we obtain a partial contour for the blocked object. Spurious matches in this scenario are removed by outlier rejection using PROSAC. Figure 3.3 shows the results of such an alignment. We compare our results with that of Seeing 3D

	BigBird		TableTop	
	Accuracy (%)	MPE	Accuracy (%)	MPE
S3DC	34.5	0.013	45.7	0.044
Ours	**49.7**	**0.008**	**67.3**	**0.021**

Table 3.3 Quantitative comparison of S3DC with our proposed method on BigBird and TableTop dataset. Note that our method outperforms S3DC with respect to classification accuracy and mean pose error. Mean Pose errors are compared for correctly classified examples only.

Chairs [5] (S3DC) in figure 3.8. Note that our method correctly detects and classifies objects and is able to provide a finer alignment.

We also run our experiments on articulated objects. Objects such as $scissors$ have articulation points which influence certain model vertices to rotate or translate with respect to them. Given the articulation points, and their transformation matrices, we modify the LM-ICP framework to account for the error due to local deformations in the object. Figure 3.7 shows the results on articulated objects.

Since our fine alignment is based on shape features, symmetric objects (like $wireless\ mouse$ in Figure 3.8) often fail to fit correctly. Moreover, as the initial estimate is based on context aware segmentation, the approach is unable to find the exact pose if the rough pose is far away from the groundtruth. Figure 3.8 shows some of the failure cases of our method.

Figure 3.6 We show results of our proposed approach. Top row shows the scene images from our dataset. In bottom row, we superimpose the CAD models onto the image in their exact pose. Note that occluded objects are also aligned with minimal error. (Best viewed in color)

3.5 Conclusions

We have developed an ensemble of shape features for the purpose of fine alignment of textureless 3D models of objects to 2D images of real-world objects. We are able achieve this in spite of the clutter

(a) (b) (c)

Figure 3.7 Demonstration of our method on articulated object. (a) Input test image (b) Original model (c) Deformed model superimposed on image. (Best viewed in color)

and occlusion in the scene as well as the lack of texture on the 3D models. The method is shown to be very effective on a dataset of tabletop objects and robust against partial occlusion and comparative results with state of the art are presented.

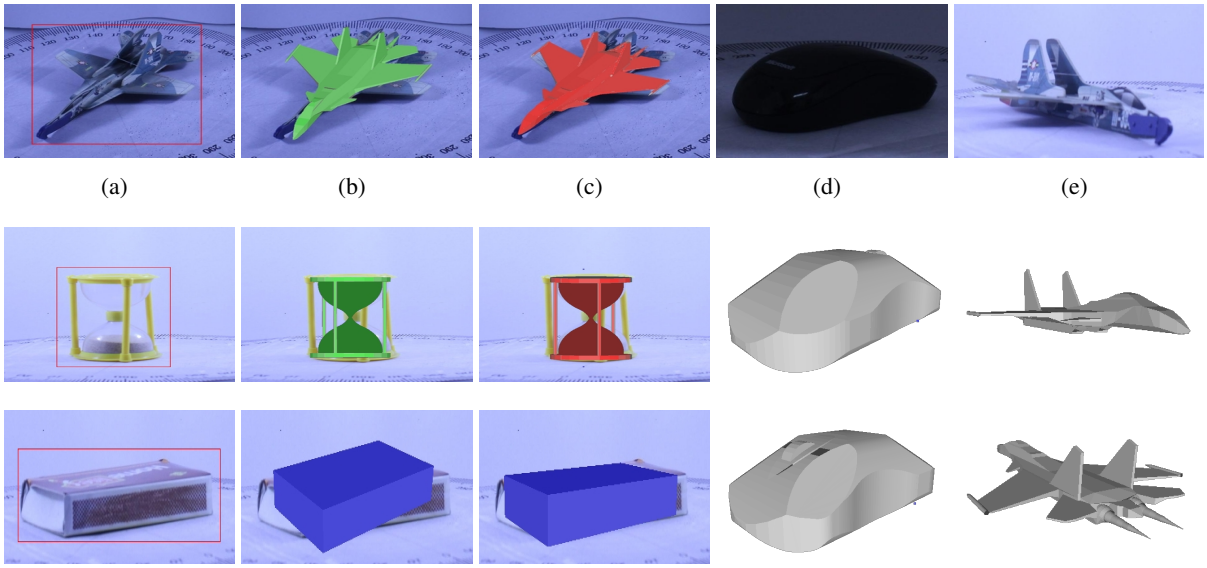


Figure 3.8 (a) Input test images (b) Fine poses obtained from S3DC (c) Fine poses obtained from the proposed approach. Although, S3DC correctly classifies objects, they fail to align the CAD model properly. (d-e) Failure cases. Top row shows input images. Middle row and bottom row shows groundtruth and obtained pose respectively. (Best viewed in color)

Chapter 4

Object Proposals Using Deep Features for Instance Level Semantic Segmentation

4.1 CNN: An Overview

Convolutional Neural Networks (CNN) are very similar to ordinary Neural Networks (NN). But as we have previously mentioned, CNNs are specially built to take advantage of the 2D structure of images. Of course, neural networks can be modified to work accordingly with 3D structure or voxels as well. NNs have proved to be highly efficient in the field of Natural Language Processing (NLP) and data mining. The term *convolution* comes from the fact that parts of images are *convolved* with filters or kernels of predefined size to get some output. Neural networks receive an input (a single vector), and transform it through a serious of *hidden layers*. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous later, and where neurons in a single layer function independently and do no share any connections. The last fully connected layer is called the ”output layer” and in classification settings it represents the class probability scores. The class with maximum score represents the category for that object in the image.

A CNN, on the other hand is comprised of one or more *convolution layers* (often with a subsampling or *pooling* step) and then followed by one of more fully connected layers as in the standard NN architecture. The architecture of CNN is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. One benefit of CNN is that they are easier to train and have fewer parameters than fully connected networks with same number of hidden layers. The reason is that a neuron is connected to few other neurons in the previous layer, which is equivalent to the size of the kernel. More on this later. In this section, we explain the architecture and types of layers in a typical CNN in detail. We also explain the *backpropagation* technique which is how a CNN learns from the examples.

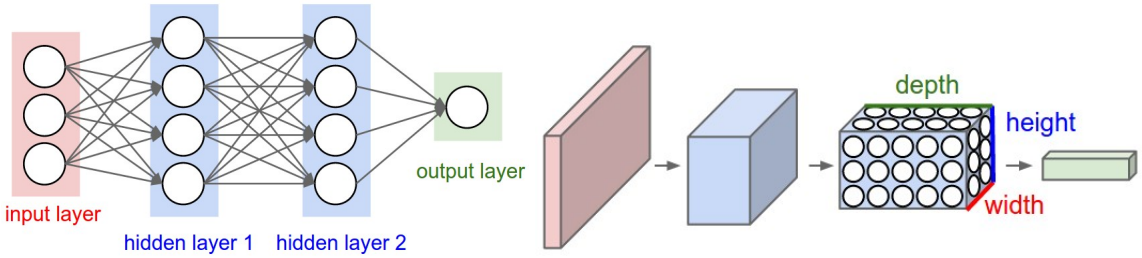


Figure 4.1 Left: A regular 3-layer Neural Network. Right: A ConvNet arranges its neurons in three dimensions (width, height, depth), as visualized in one of the layers. Every layer of a ConvNet transforms the 3D input volume to a 3D output volume of neuron activations. (Image source: Google)

4.1.1 Architecture

Regular NNs don't scale well to full images. Since all neurons are connected to all others, they have a huge number of parameters. For example, in CIFAR-10, images are only of size $32 \times 32 \times 3$, so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 3072 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectible size, *eg* $200 \times 200 \times 3$, would lead to neurons that have $\sim 120,000$ weights. Moreover, we will have more hidden layers and hence more number of neurons, so the parameters add up quickly. This not only overfits the data but also take a lot of time training.

CNNs take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular neural network, the layers in CNN are arranged in 3 dimensions: **width**, **height**, and **depth**. Here *depth* refers to the third dimension of an activation volume. As previously mentioned, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully connected manner. In a classification task, the final output layer has dimensions $1 \times 1 \times C$, where C is the number of classes. Figure4.1 shows the comparison in architecture of a CNN and a neural network.

4.1.2 Types of Layers

As described above, every layer of a CNN transforms a 3-dimensional volume to another through a differentiable function. We give a brief description of the important layers involved in a CNN.

4.1.2.1 Convolution Layer

Convolution layer or simple *Conv* layer is the core building block of a Convolutional Network, and its output volume can be interpreted as holding neurons arranged in 3D volume. The conv layer's parameters consist of a set of learnable filters. Every filter is small spatially (along width and height),

but extends through the full depth of the input volume. During the forward pass, we convolve each filter across the width and height of the input volume, producing a 2-dimensional activation map of that filter. As we slide the filter across the input, we are computing the dot product between the entries of the filter and the input. Intuitively, the network will learn filters that activate when they see some specific type of feature at some spatial position in the input. Stacking these activation maps for all filters along the depth dimension forms the full output volume. Every entry in the output volume can thus also be interpreted as an output of a neuron that looks at only a small region in the input and shares parameters with neurons in the same activation map (since these numbers all result from applying the same filter). When dealing with high-dimensional inputs such as images, as we saw above it is impractical to connect neurons to all neurons in the previous volume. Instead, each neuron connects to only a local region of the input volume. The spatial extent of this connectivity is a hyperparameter called the receptive field of the neuron. The extent of the connectivity along the depth axis is always equal to the depth of the input volume. It is important to note this asymmetry in how we treat the spatial dimensions (width and height) and the depth dimension: The connections are local in space (along width and height), but always full along the entire depth of the input volume.

The output of a conv layer is another 3D volume. Three hyperparameters control the size of the output volume: **depth**, **stride**, and **zero-padding**. First, the *depth* of the output volume is a hyperparameter that we can pick; It controls the number of neurons in the conv layer that connect to the same region of the input volume. This is analogous to a regular neural network, where we have multiple neurons in a hidden layer all looking at the exact same input. We will refer to a set of neurons that are all looking at the same region of the input as a*depth column*. Second, a *stride* is the pre-defined hop with which we slide the filter across the spatial dimension of the input. When the stride is 1, then we will allocate a new depth column of neurons to spatial positions only 1 spatial unit apart. Smaller stride leads to heavily overlapping receptive fields between the columns, and also to large output volumes. Conversely, higher stride mean receptive fields will overlap less and the resulting output volume will have smaller dimensions spatially. Finally, *zero-padding* lets us control the amount of padding that we might add to the spatial border of the input. We compute the spatial size of the output volume as a function of the input volume size (W), the receptive field size of the conv layer (F) (assuming both height and width are same), the applied stride (S). and the amount of zero-padding (P). We can convince ourselves that the output size is $(W - F + 2P)/S + 1$. As an example, let's consider the Krizhevsky *et al.* [43] architecture that won the *ImageNet* [17] challenge in 2012. It takes images of size $227 \times 227 \times 3$ as input. On the first conv layer, receptive field size is $F = 11$, stride $S = 4$ and no zero-padding. The conv layer had a depth of $K = 96$. By the formula, we see that the output volume size was indeed $55 \times 55 \times 96$. All the 96 neurons in each depth column are connected to the same $11 \times 11 \times 3$ region of the input, but with different weights. *Parameter Sharing* is a scheme used in conv layers to control the amount of parameters. In the above architecture, we see that there are approximately ~ 100 million parameters! Assuming that one patch feature is useful to compute at different spatial location in an input and denoting a single 2-dimensional slice of depth as a *depth slice*, we are going to constrain the neurons

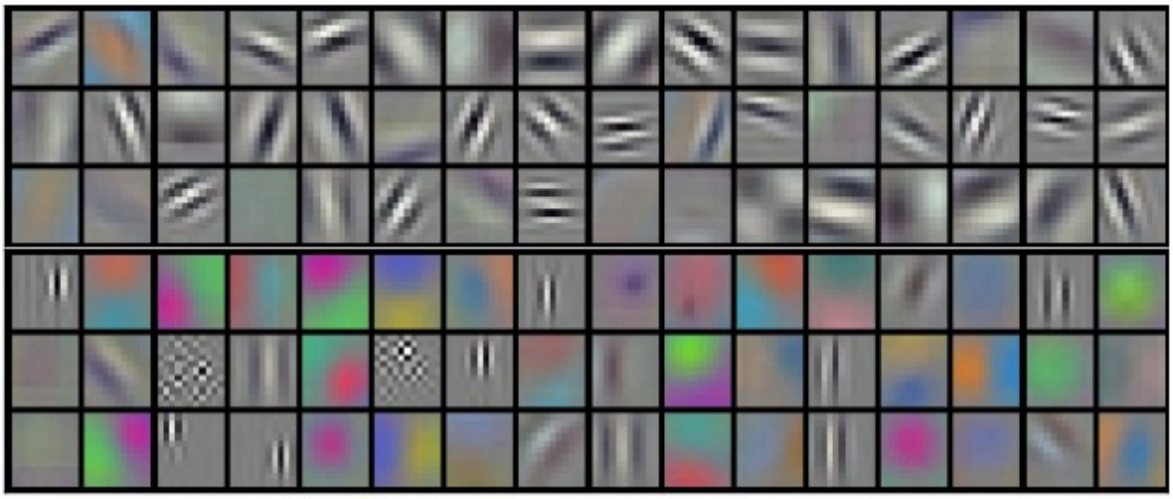


Figure 4.2 Filters learned by Krizhevsky et al. Each of the 96 filters shown here is of size $11 \times 11 \times 3$, and each one is shared by the $55 * 55$ neurons in one depth slice. Note that the filters learn high level features like horizontal and vertical edges, among others. (Image source: Google)

in each depth slice to use the same paramaters. The intuition behind this is, if detecting a horizontal edge is important at some location in the image, it should be useful at some other location as well. As expected, the number comes down to $\sim 35,000$ parameters. Figure4.2 shows the learned filters in the trained Krizhevsky architecture.

4.1.2.2 Rectified Linear Unit (ReLU

Rectified Linear Unit or *ReLU* [64] is an *activation function* which has shown excellent results on Restricted Boltzmann Machines (RBM). An activation function is nothing but a function of inputs. For example, a threshold function can be a simplest activation function. All values below a specified value are clipped. In other words, all neurons which have less value than threshold, are not allowed to fire. ReLU is represented as a function of the inputs: $H(x) = max(0, x)$, where $x = Aw + b$. ReLU have only been recently introduced to tackle the problem of *vanishing gradient*. Vanishing gradient is a common yet difficult problem in gradient-based methods. During backpropagation, the weights of the network are updated with the gradient of the error signals. While using exponential activation functions like *sigmoid function*, the gradient reduces exponentially after each iteration; hence training gets stalled. Since ReLU is a linear function of the weights, the gradient simply is unit. This reduces the chances of vanishing gradients. Not to mention that ReLU is very fast to calculate and relatively simpler to differentiate compared to other known activation functions.

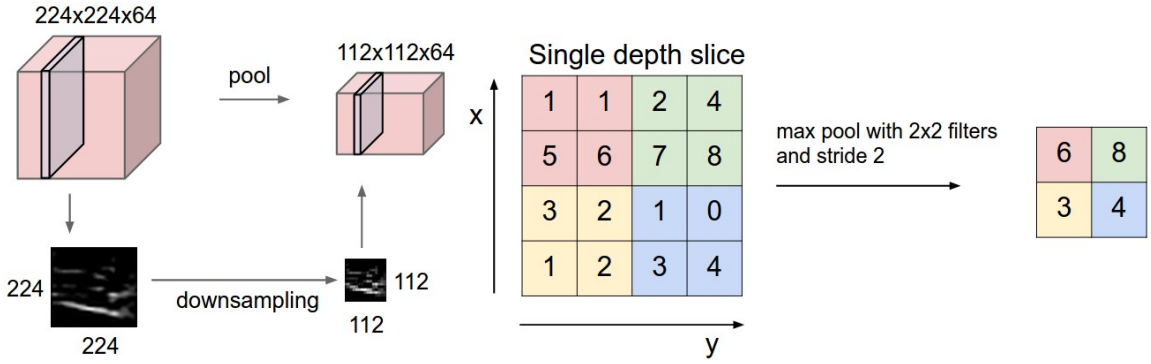


Figure 4.3 Pooling layer downsamples the input volume spatially. Left: The spatial size is reduced by half. Although the number of depth slices remain unchanged. Right: Max pooling operation. If the receptive field is 2×2, the maximum value from a window of size 2×2 is selected. Similar operation follows in average pooling. (Image source: Wikipedia)

4.1.2.3 Pooling Layer

A pooling layer is a relatively simple layer in the CNN architecture. Its function is to progressively reduce the spatial size of the representation to reduce the amount of parameters and computation in the network. Pooling also controls overfitting. It operates independently in each depth slice and resizes it spatially. Generally, a pooling layer works the same way as a conv layer, except that instead of multiplying a kernel with the input, it performs an scalar arithmetic operation on the inputs. As a result, the single, scalar value of a pool represents the entire receptive field. Examples of pooling are: **Max** and **Average**. In *Max* pooling, the maximum element among the inputs is chosen and forwarded the next layer. In *Average* pooling, the average of all input values are forwarded. There are other varieties of pooling, but those are out of the scope of this thesis. Figure4.3 illustrates the pooling operation.

4.1.2.4 Fully Connected Layer

A fully connected layer is just like a standard neural network, except that in CNN, they work in a spatial manner. In this layer, a neurons is connected to all the neurons in the previous layer. Hence, the output is just the matrix multiplication of the inputs and the weights with a bias offset.

4.1.3 Backpropagation

Backpropagation is the technique with which a neural network learns from examples. Think of a neural network as a black box. Given a lot of examples to this black box, it ultimately learns some unique characteristics from those examples. When we say ”learn”, we mean updating the weights or parameters of the network. In other words, the way the signals with which the neurons interact with each other; these signals are constantly being updated after ”seeing” each example. Once the network

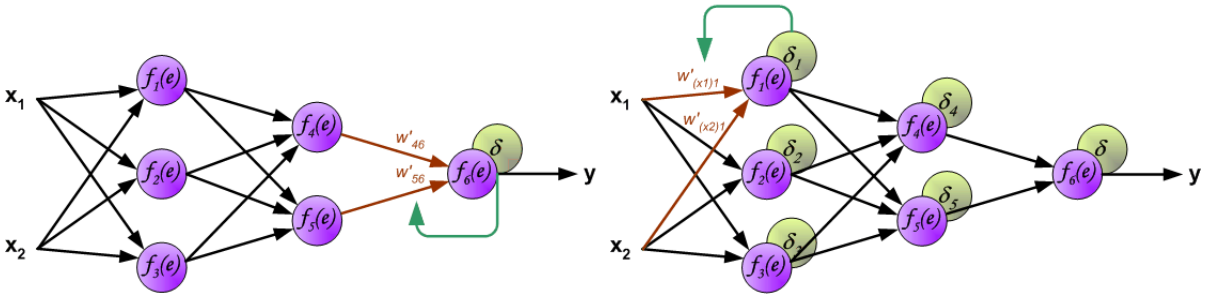


Figure 4.4 Example of backpropagation in a standard neural network. Purple blobs denote the neurons as functions of input example e. Green blobs denote the gradient errors. (Image source: Wikipedia)

stops seeing anything new, the learning stops; the weights converge and there are no more updations of the neuron branches. Backpropagation is the method of updating the weights by differentiating the error function with respect to the inputs. Recall that to minimize a function with respect to an input, we need to find the gradient of the function with respect to the input. We do the same thing in a CNN. Given an input, the gradient of the error function is computed and the weights of the last layer are updated. After that, the error is "backpropagated" back through the network in the same manner, until it reaches the higher layers and finally the first layer. Using the updated weights, a new input is forwarded, and the weights are re-updated in the same manner. It should be noted that backpropagation requires actual groundtruth data to compute the error. Hence backpropagation works in supervised machine learning. In unsupervised learning like K-means algorithm, the error is calculated from the mean distance of examples from the cluster center; no true labels are required as such.

Let us consider the example shown in Figure4.4. We have a 2-dimensional input with an activation function, denoted by e and a single output denoted by y. The gradient of the error $f_6(e)$ with respect to e is $\frac{df_6(e)}{de}$. The weight update rule for backpropagation follows:

$$w_{46}^{'} = w_{46} + \eta\delta\frac{df_6(e)}{de}y \tag{4.1}$$

$$w_{56}^{'} = w_{56} + \eta\delta\frac{df_6(e)}{de}y \tag{4.2}$$

where η is the learning rate and $\delta = y - z$, z being the groundtruth value. Figure4.4 (left) shows the gradient of the error signal with respect e is used to updated the weights of the last layer. In Figure4.4 (right), we see that the error is "backpropagated" to the upper layers, and the weights are updated using the gradients of $f_1(e)$. Hence, we can write:

$$w_{(x_1)_1}^{'} = w_{(x_1)_1} + \eta\delta\frac{df_1(e)}{de}x_1 \tag{4.3}$$

$$w_{(x_2)_1}^{'} = w_{(x_2)_1} + \eta\delta\frac{df_1(e)}{de}x_2 \tag{4.4}$$

4.2 CNN in Multi Instance Semantic Segmentation

Convolutional Neural Networks with the classical backpropagation technique came into existence long back. Since then, researchers have been trying to get their hands dirty with it to solve any and every problem in computer vision, data mining, social network analysis and natural language processing. The major breakthrough of CNN occured when Yann *et al.* [47] used gradient descent to tackle the problem of MNIST digit classification task. Earlier, feature based classifier were used for that, and were either not efficient in terms of accuracy, or failed at classifying difficult examples. With CNN, Yann *et al.*achieved around 95% accuracy, which beat the other state-of-the-art methods by a long margin.

It has to be noted that CNN first started off to be applied to object classification tasks like digit classification, where we have a single label against an image. The labels in this case are 0 through 9. In case of handwriting recognition, the documents are the training examples and the groundtruth labels are the corresponding authors of the documents. Another ground breaking achievement in the computer vision community came through with the introduction of *Imagenet* [17]. Imagenet is an image database organized according to the WordNet hierarchy. Considering each word as a node, the WordNet can be imagined as a tree, where each node represents a meaningful concept and holds a set of other words below it. With over 14 million images in the database, Imagenet is currently the biggest database of images available online for the task of object detection and classification. Krizhevsky *et al.* [43] achieved a test error of 37.5%, which beat the state-of-the-art *Sparse Coding* and *SIFT* based approaches by a long margin. Since then, CNNs have been heavily used in the task of object detection, classification and localization in images and videos [43, 12, 39, 76, 14].

With the recent advancement of CNN in the problem of object recognition, researchers started thinking of applying CNN to classify an image pixel, instead of an object: Semantic segmentation. Intuitively, this works because, the lower level layers capture intricate details about any object in the image. In other words, a feature map in the lower layer actually sees a part of the image. The only difference in architecture is the groundtruth labels now become 2D in nature, with spatial location pixel denoting the label at that pixel. In instance specific semantic segmentation, each pixel is identified not only with its class but also its instance. In other words, all pixels belonging to a particular instance should be labelled. Conversely, two pixels which do not belong to the same class should be labelled differently, irrespective of them belonging to the same category. In the next section, we describe our proposed approach to solve the problem of multi-instance semantic segmentation using deep neural networks.

4.3 Proposed Approach

In this section, we explain our proposed method in detail. To start off, we formulate finding the multiple instances as search for multiple instances in the image. Once we have a proposed hypothesis on the probable object spaces, getting bounding box around them gives us the possible instances of objects. The next step becomes removing the spurious object proposals. We can reduce the number of

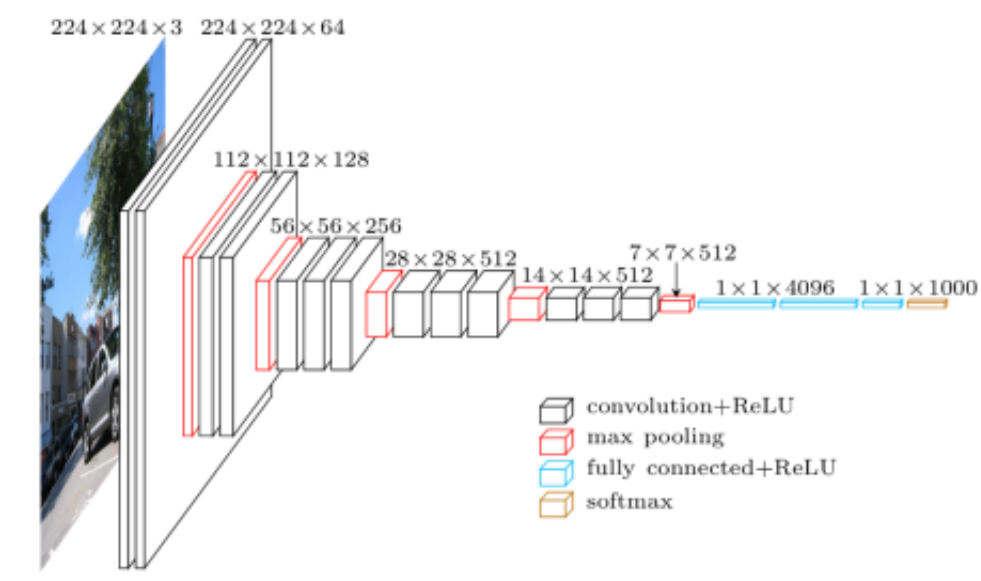


Figure 4.5 Architecture of the VGG-16 network. An input image of size $224 \times 224 \times 3$ is forwarded through the network and 4096-dimensional features are learned through efficient back-propagation along with softmax loss function to obtain 1000 class specific probabilities corresponding to 1000 classes. VGG-16 is trained on 1000 class Imagenet dataset. (Image source: Google)

proposals using heuristics like area of bounding box, number of pixels in the proposal etc. For the rest of the proposals, we need to categorize them into either background or one of the foreground classes. Once we have the detection scheme in place, we can segment the individual instances one by one. Generally, the detection and segmentation modules work as a pipeline, but our method, we try to refine the one with the other. In our deep network, we have one branch computing the object score for an instance, while another branch solving the segmentation problem. We use a loss function which takes into account the losses of both the detection and segmentation branches of network and backpropagates the error as a whole back through the network.

4.3.1 Pre-trained VGG-16 Network

Oxford's *Visual Geometry Group* (VGG) published a paper [81] where they have trained neural network on the Imagenet dataset and achieved amazing results on the test set with an top-5 error rate of 7.5% on the *ILSVRC 2012* validation set. The trained models are made available for researchers to use. Hence, we do not have to re-train our own network from scratch which takes a few days to weeks on any modern GPU based machine, depending on the size and depth of the network. We use the pre-trained model of the *VGG* 16-layer network. Since it is already trained on a huge dataset of images, we can assume that any scene dataset consists of objects that have been "seen" by the network. Hence,

the feature maps of the upper layers need not be re-trained, as they represent higher level features like edges or boundaries. We modify the lower layers of the network according to our approach, and re-train keeping the weights of the upper layers intact. In other words, we allow backpropagation upto a certain layer and block from there below. The network architecture is shown in Figure4.5.

4.3.2 R-CNN, Fast and Faster R-CNN

Region based CNN or R-CNN [27] is a state-of-the-art simple and scalable object detector based on rich features extracted from region proposals. Given an image, region proposal techniques are used to propose multiple instances of segments in a bottom-up approach. R-CNN takes advantage of the power of convolutional neural networks to extract fixed length features. The features are trained on class specific SVM (Support Vector Machines) to classify the regions into either background or one of the foreground classes. General VGG pre-trained network are trained on Imagenet dataset. Hence, the network is trained to learn filters on a sliding window manner across the image, over all the scales of the image. Finally, the fully connected layer recognizes the object in the image as one of the 1000 objects. In other words, Imagenet images have a single object in an image, which makes the VGG trained network to recognize a single object in an image. The use of region proposal techniques facilitates the segmentation of an image into mutliple segments which can in turn be treated as a separate bounding box. Each bounding box is then fed into the VGG trained network and fixed length features are extracted.

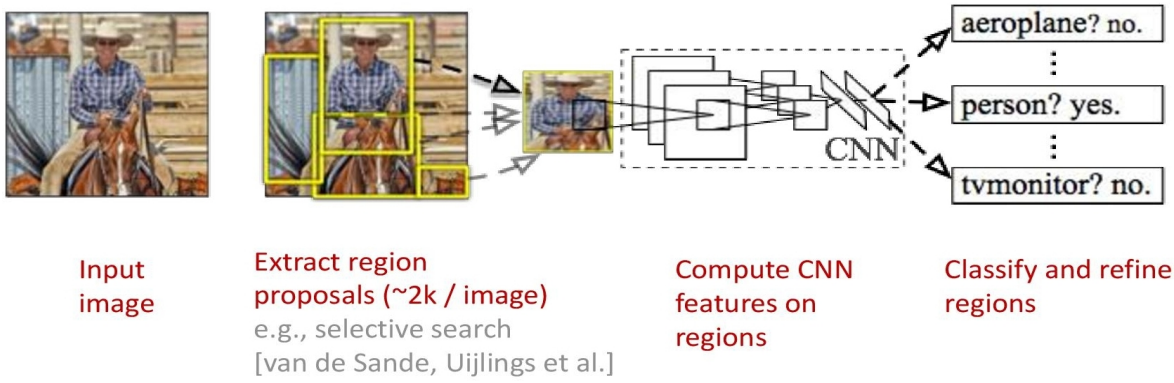


Figure 4.6 Illustration of the R-CNN pipeline. The first stage is extracting regions prposals. R-CNN uses selective search [84] to obtain the prposals. Each proposal bounding box is used to compute CNN features on pre-trained VGG-16 network trained on Imagenet dataset. Finally using the fixed length 4096-dimensional features, the detected regions are classified and refined. (Image source: [27])

Figure4.6 shows the pipeline of the region based object detector. The final feature matrix size of 2000×4096 is classified using an SVM whose weight matrix has the dimensions $4096 \times N$, where N is the number of classes. Clearly, the dimensions are consistent. The region is classified into the

class with the highest score in the weight matrix. But there are two major problems with R-CNN. 1) During test time, generation of 2000 regions from any region proposal method is going to take time, and extraction of features for each bounding box make the process even slower. A typical image takes 47s per image on a GPU, which prohibits the use of R-CNN in real time applications. 2) The bounding boxes from the region proposal method are warped to a size which is consistent with the VGG-16 network (227×227). Hence, aspect ratio and scale of the boxes are not retained (see figure4.7), which renders some misclassification for objects which lose their identity when warped. To tackle these problems, Fast R-CNN was introduced [26].

Fast R-CNN makes some innovative changes to R-CNN architecture to facilitate the forward propagation of variable size bounding boxes. R-CNN is slow because it performs forward pass for each object proposal, without sharing computation. *Spatial Pyramid Pooling* networks (SPPnets) [35] were proposed to speed up R-CNN by sharing computation. Spatial Pyramid Pooling uses the advantage of the fact that convolutional layers can work on variable sized inputs, as they are basically work in a sliding window fashion. But fully connected layer needs the input to be consistent with the number of neurons in the layer. In SPP, an input convolutional feature map of arbitrary size is converted to a fixed length representation, using a number of spatial pools by taking the maximum (or average) activation in a rectangular region. Figure 4.7 shows the overview of SPP. During training time, an SPP layer is added in between the last convolutional layer and the fully connected layer, which converts the variable sized feature maps into fixed length output which is consistent with the fully connected layer.

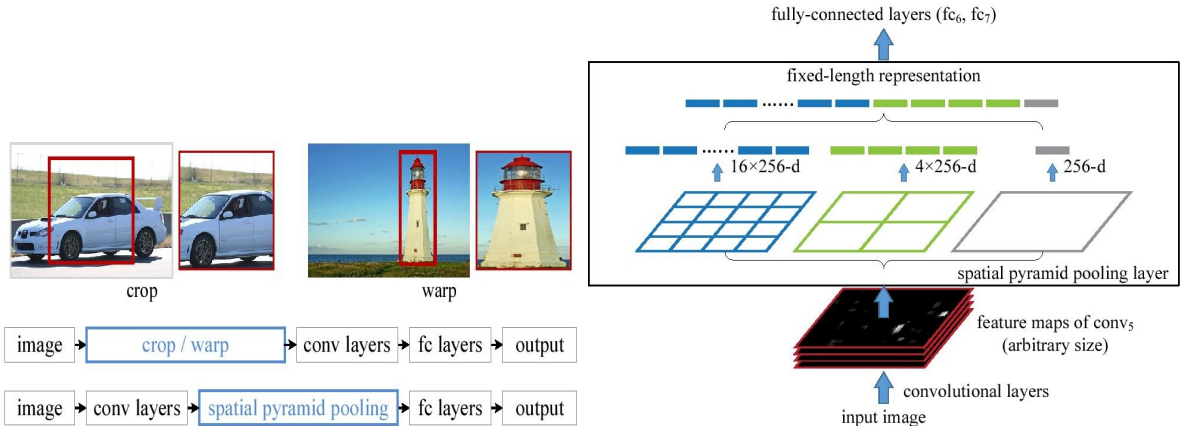


Figure 4.7 Left: Cropping or warping does not retain the aspect ratio and scale of the object. Hence, a spatial pyramid pooling layer is added in between the last conv layer and fully connnected layer. Right: Network structure with a spatial pyramid pooling layer. Here 256 is the number of feature maps of the last convolutional layer ($conv_5$). (Image Source: [35])

The advantage of fast R-CNN is it shares the computation of the feature maps across overlapping ROI (Region of Interest). Hence, network weights of overlapping regions need not be computed more than once. Fast R-CNN uses the groundtruth ROIs as regions proposals, as opposed to R-CNN's selective

search region proposal technique. This enables fast R-CNN to formulate the detection and localization task as a multi-task problem. In other words, the architecture gets two types of information from each ROI: class of the object and the bounding box coordinates. As a result, during training time, the fully connected layer branches off to not only the general softmax regressor (for classification), but also a bounding box regressor, which is represented as a smooth L1 loss function. The arbitrary sized ROIs from the feature maps are converted to fixed length representations using spatial pyramid pooling (which is called ROI pooling layer here). The network thus learns to jointly minimze the two objective functions.

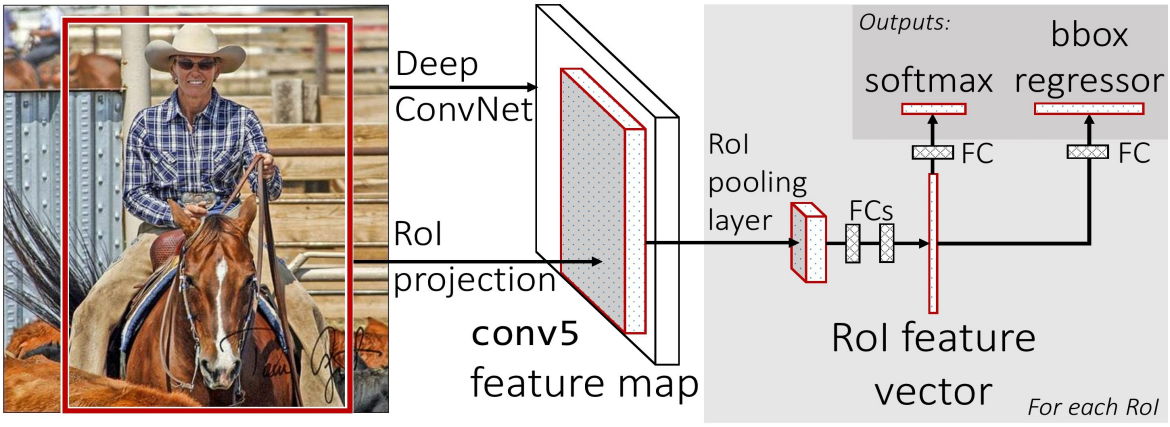


Figure 4.8 Network architecture of fast R-CNN. (Image Source: [26])

Figure 4.8 shows the network structure of fast R-CNN. Note that the network jointly learns to optimize the softmax loss and bounding box loss. There are three inputs to the network. The batch of images (the batch size has been fixed at 1), the ROIs which is a 5-tuple: 4 coordinates and a class id, and the corresponding labels. The softmax regressor layer has 21 outputs as there are 21 classes including background (experiments have been conducted on PASCAL VOC dataset which contains 21 unique objects). Whereas the bounding box regressor has 84 outputs, corresponding to 4 coordinates for each class.

Fast R-CNN is fast but to make it more real time, *Faster R-CNN* was introduced, which makes detection and localization almost real time. In this work, a *region proposal network* (RPN) was introduced which shares full convolutional features with the detection framework, hence enabling nearly cost-free region proposals. RPN is trained end-to-end to generate rich region proposals, which are then used in the fast R-CNN framework. RPN predicts object bounds and objectness score simultaneously. RPN is very efficient compared to using region proposal methods like selective search, MCG etc. owing to the fact that it shares the computation of the wrights at each convolutional layer and hence utilized during training time. It was observed that only 300 proposals per image was generated by RPM which is almost 7 times less compared to nearly 2000 proposals generated in R-CNN. Once the ROIs are generated by

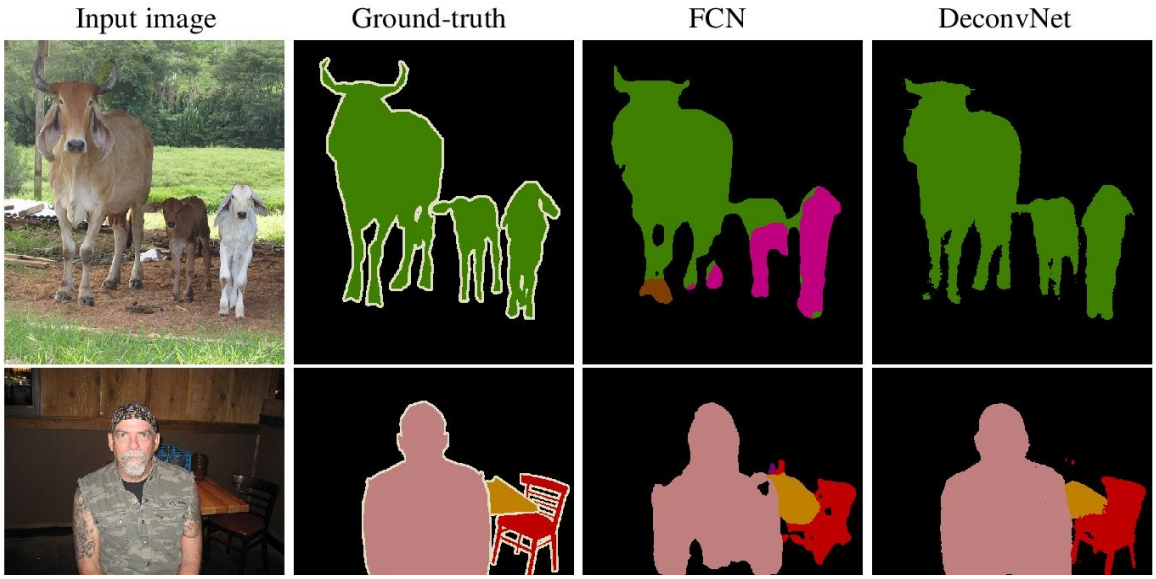


Figure 4.9 Examples where DeconvNet shows better segmentation results than FCN. (Best viewed in color)

RPN, the rest of the framework is based on fast R-CNN. Each ROI is then spatially pooled to create fixed length representations and forwarded to fully connected layers.

4.3.3 Network Overview

In this section, we describe a detailed explanation of some of the network architectures that we have developed. The overall task that we are interested in is optimizing a mutli-task loss function which essentially refines the segmentation task. In order to get the individual object instances, we use either fast R-CNN or region proposal network (RPN) of faster R-CNN. Once we have the instances, we now have control over the pixel-wise classification of each instance bounding box. Optimizing the pixel-wise classification score can be done in a variety of ways, from simple spatial softmax classification loss to end-to-end training of conditional random fields (CRF) to jointly reduce the loss as a mean-field estimation. In the next subsections, we describe each of the above specified networks in detail.

4.3.4 Training Ensemble of Fast R-CNN and Deconvolution Network

As explained before, fast (or faster) R-CNN is specifically designed for learning the classification task. We use the learnt filters from the pre-trained VGG-16 network in order to refine the pixel-wise segmentation task, as well as get the instances as bounding boxes. Apart from changing the network architecture and fine-tuning VGG-16, we also modify the loss function to formulate the segmentation process as a multi-task problem, just like fast R-CNN. The idea is to use the detection of region pro-

posals to improve segmentation framework and vice-versa. Doing so, the network learns to see multiple instances as multiple ROIs obtained from region proposals. To this end, we use the concept of fast R-CNN to obtain ROIs from convolutional feature maps and upsample them to obtain the separate instances of a class. Along with the regressed bounding boxes, we also obtain objectness score for each ROI, which should ideally help in refining the segmentation part of the network. We treat the segmentation architecture as a *deconvolution* network which branches off from the last fully connected layer. At the end of the deconvolution framework, a spatial softmax loss is applied on the 2D feature maps of the last layer, which classifies each pixel into one of the classes, including background. Hence, the network jointly learns to optimize a multi-task loss which is a weighted sum of three losses: box classification score, bounding box L1 loss, and spatial softmax loss. The reason we use deconvolution network is that FCN gives inaccurate results in images which have small objects, or images which have inconsistent labels due to large object size, deconvolution networks have been proved to be robust against them. Figure 4.9 shows examples of images where deconvolution networks produce better results than FCN.

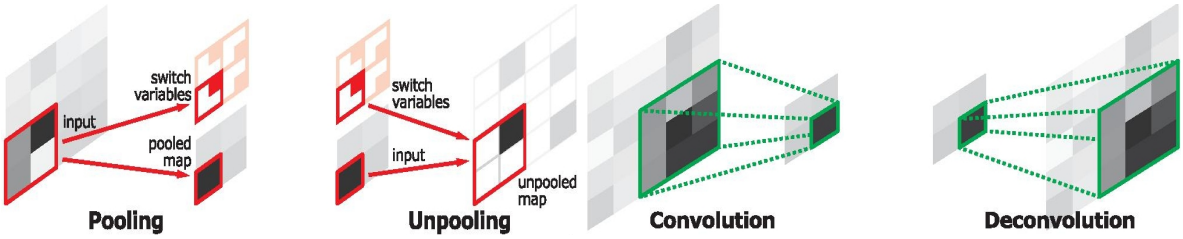


Figure 4.10 Illustration of unpooling and deconvolution operation. (Best viewed in color)

Modelling the segmentation problem in a deconvolution framework has been first seen in the work of Noh *et al.* [67], where a deconvolution network on top of the pre-trained VGG-16 network. A deconvolution network has two key operations: *Unpooling* and *Deconvolution*. Unpooling is nothing but the reverse operation of pooling. In pooling, multiple activations in a spatial window is represented by a single activation value. This not only reduces noisy activations and makes the classification robust, but also loses essential localization information which is required for semantic segmentation. Unpooling tackles this problem by restoring the original size of activations, using *switch variables* which records the location of the forwarded activation in the spatial window.As described in (cite from deconv paper page 3), unpooling is particularly useful to reconstruct the structure of input object. On the other hand, deconvolution layer densifies the sparse output of unpooling layer by learning filters from a single activation. Contrary to a convolution layer which connects weights from mutliple locations in a receptive field to a single activation in the next layer. The output of a deconvolution layer is an enlarged feature map whose size is identical to the one before the unpooling layer. Figure 4.10 shows the unpooling and deconvolution process. Hence, just like convolution layers, the hierarchical structure of deconvolution layers is used to capture the different level of shape details of an object at different stages of the network.

The filters in lower layers learn the overall shape of an object, while the class-specific fine details are encoded in the filters at higher layers, as seen in figure 4.11.

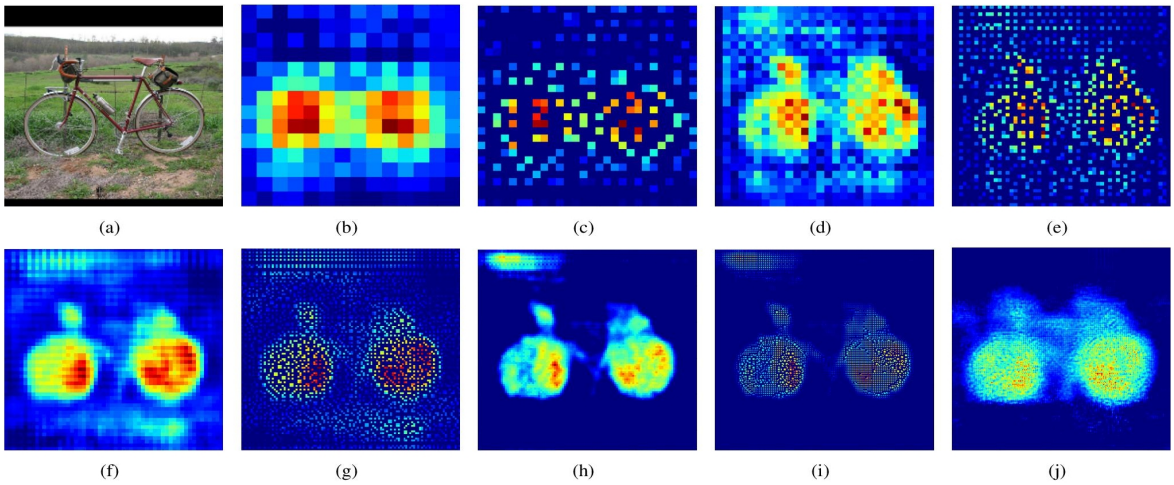


Figure 4.11 Visualization of activation in deconvolution network. Activation maps from (a) to (j) correspond to feature maps from lower to higher layers in the deconvolution network. The figure shows different maps after unpooling and deconvolution layers. As mentioned above, lower layers learn shape of the object, whereas higher layers learn finer class-specific details. (Best viewed in color)

4.3.5 Multi-Scale Feature Learning: Variable Spatial Kernel Size

We use the faster R-CNN network which is pre-trained on PASCAL VOC dataset to get the individual instances of objects during training. The reason is the region proposal network is already trained to learn the multiple objects from the convolutional feature maps and groundtruth bounding boxes. One we have the regions at hand, we remove the upper layers from the trained faster R-CNN network and concatenate a network similar to Deeplab [10]. In between the last convolution layer and fully connected layer, we add a block which we hereafter call *Variable Kernel Network* (VKN). VKN consists of K kernels of variable size which act upon the last convolutional feature maps. In other words, the last convolutional feature map is convolved with multiple kernels of variable sizes and the last feature map is obtained by element-wise sum of the outputs. VKN has been shown to be effective in learning objects at different scales (due to variable sized filters) and hence segmentation results have proved to be better than without the VKN module. Following Chen *et al.* [10], we also add the *hole algorithm* or *atrous algorithm* in the convolutional layers, which not only speeds up the training process but also increases mean IoU.

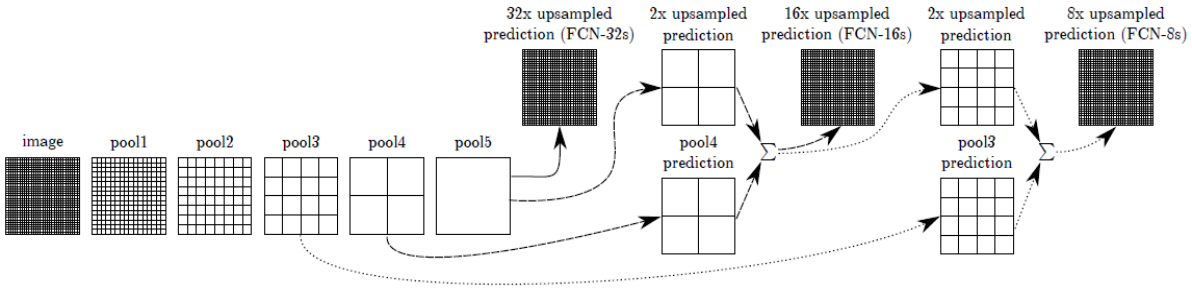


Figure 4.12 Skip Architecture as described in [55]. As shown, pooled features from lower layers are upsampled and element-wise sum is performed on them at upper layers.

4.3.6 Multi-Scale Feature Learning: Skip Architecture

Skip Architecture was first introduced in FCN networks. Fully convolutional networks for semantic segmentation had shown significant improvements in semantic segemtation results and as a result is one of the state-of-the-art methods in this task. Skip architecture is a modification in the neural network which combines learnt feature maps from lower layers to that in higher layers. In other words, semantic information from a deep, course layer is combined with appearance information from a shallow, fine layer to produce finer detailed segmentation results. Specifically, pooled feature maps (feature maps after each pooling layer) are combined together before the fully connected layer. This not only ensures the network learns the classes in different scales but also makes sure the learned filters are more robust to coursity of the objects. In order to combine the pooled feature maps at each scale, the maps need to be upsampled with the same upscale factor. FCN networks has been trained on PASCAL VOC dataset, which makes it easier to fine-tune, as the number of classes remain same, as opposed to VGG-16 network which had been trained on Imagenet dataset. Hence, refining the network paramaters require extensive tuning of learning and decay rate.

Figure 4.12 shows the skip architecture as describe in [55]. We make a slight modification in the network though. In FCN, the pooling makes the network robust by representing a spatial window of activations with a single value. Upsampling that value merely interpolates it to the scaled spatial window around it. Hence, no learning of filters takes place. Hence, we add convolutional layers after each pooling stage and upsample the final convolved features before merging. This ensures not only a more robust feature learning owing to pooling, but also a more courser learning due to efficient back-propagation of the convolution layers, at different spatial scales. Figure (show figure) shows the modified network with skip architecture and convolution layers.

4.3.7 Post-Processing CRF on Unary Potentials from CNN

During test time, dense predictions from the network are computed from the output score map. For each pixel in the spatial map in the last layer, a score is generated for each class. For example, if the size of the output prediction layer is $h \times w \times C$, where C is the number of classes, then the class of the pixel is obtained by $argmax_{c \in C} O$, where O is the output score map. To generalize the above task in a graphical model, we can say that the output score map is in fact the *unary potentials* for each pixel. Unary potentials specify the probability that a pixel belongs to a given class. More specifically, the score needs to be normalized such that the sum is 1. This gives the probability map and hence the unary values for each pixel. During test time, the output scores need to be upsampled to the size of the image. A simple bilinear or trilinear interpolation proves to be good enough for bringing the score map upto the size of the image. After that, a fully-connected CRF [42] (or dense CRF) on the upsampled score map, estimates the class labels using mean-field estimation. The *pairwise potentials* provide a data-dependant smoothing term that encourages assigning similar labels to pixels with similar properties. To recap, the energy of a label assignment x is given by:

$$E(x) = \sum_i \lambda_u(x_i) + \sum_{i<j} \lambda_p(x_i, x_j) \tag{4.5}$$

where $\lambda_u(x_i)$ measures the inverse likelihood (and therefore the cost) of pixel i taking label x_i, and pairwise potentials measures cost of assigning labels x_i and x_j to pixels i and j respectively. The pairwise potentials are modelled as weighted Gaussians, similary to (cite from CRF RNN):

$$\lambda_p(x_i, x_j) = \tau(x_i, x_j) \sum_{m=1}^{M} w^{(m)} k_G^{(m)}(f_i, f_j) \tag{4.6}$$

where each $k_G^{(m)}$ for $m = 1, \ldots, M$ is a Gaussian kernel applied on feature vectors. $w^{(m)}$ is the weight parameter. f_i and f_j are feature vectors of pixel i and j respectively and are derived from their RGB values and spatial location. $\tau(x_i, x_j)$ is called the label compatibility function, which captures the compatibility between different pairs of labels. $\tau(x_i, x_j)$ is 1 if $x_i = x_j$, otherwise 0 (i.e. Potts model). Following the work of (cite krahenbul), we adopt bilateral position and color terms for the kernels as given below.

$$w_1 exp(-\frac{||p_i - p_j||^2}{2\sigma_\alpha^2} - \frac{||I_i - I_j||^2}{2\sigma_\beta^2}) + w_2 exp(\frac{||p_i - p_j||^2}{2\sigma_\gamma^2}) \tag{4.7}$$

where the first kernel depends on the pixel positions (p) and RGB intensities (I) and the second kernel depends on the pixel positions. σ_γ controls the scale of Gaussian kernel and is a hyperparameter. Figure (show figure) shows the fully-connected CRF as a post-processing module after predictions from the network are generated.

4.4 Experiments, Results and Comparisons

We train our network on 12GB Nvidia K40 GPU where it takes close to 16 hours to train the ensemble of fast R-CNN and deconvolution network. We train on the PASCAL VOC 2012 dataset has 20 foregound classes and one background class. The original dataset consists of 1,464 training images, 1,449 validation images and 1,456 testing images. The dataset is augmented by extra annotations provided by Hariharan *et al.* [32], resulting in a total of 10,582 training images. We show performance with respect to mean intersection-over-union (IoU) averaged across 21 classes.

During training we set a batch size of 2 images with minibatch size of 128 subimages or ROIs. In other words, a total of 128 ROIs must be generated per image using region proposal network or using fast R-CNN pipeline. The initial learning rate has been set to 0.001, which we mutliply by 0.1 after every 2000 iterations. For the last classification layer, we use a learning rate of 0.01. We use a momentum of 0.9 and weight decay of 0.0005. The decay multipliers for weights and bias are set to 0.1 and 0 respectively. The weights of the fully connected layers used for softmax classification and bounding-box regression are initialized from zero-mean Gaussian distributions with standard deviations 0.01 and 0.001, respectively. Biases are initialized to 0. Note that we are fine-tuning from pre-trained VGG-16 network which has been trained on Imagenet dataset.

Once the network has been fine-tuned, we perform dense predictions with fully-connected CRF as a post-processing step treating the network output scores as the unary potentials and weighted Gaussian kernels (RGB and location features) as pairwise potentials, as explained before. In order to model equation 4.7, we use the default values of $w_2 = 3$ and $\sigma_\gamma = 3$ and search for the best values of w_1, σ_α, and σ_β by cross-validation on a small set of images from the validation set. We show quantitative and qualitative comparison of our proposed network models with existing state-of-the-art methods. Fast R-CNN + Deconv model is the combination of Fast R-CNN convolution along with learning a deconvolution network for semantic segmentation. VKN and Skip are the network modifications as mentioned in Section 4.3.5 and Section 4.3.6 respectively. We use fully-connected CRF as a post-processing step to refine the region proposals and obtain a better mean IoU. We compare our work with MSRA-CFM [14], SDS [32], FCN-8s [55], TTI-Zoomout [62], R-CNN [27] and Deeplab-CRF (Deeplab's network with post-processing using CRF) [10] and show the mean IoU measurements in Table 4.4.

Figure 4.13 shows the qualitative results of our proposed network model along with some of the other recent methods. Note that the results have been obtained by running the network models on the 'test' set of the PASCAL VOC dataset. The network has bee trained on the 'val' and the augmented 'train' set of the PASCAL VOC 2012 dataset. Also note that, we show the results of our best network model based on Fast R-CNN and Deconvolution with fully-connected CRF based post-processing (FAST R-CNN + Deconv + FCCRF). As observed, we get the multiple instances of the same class using the FAST R-CNN architecture which gives us bounding box around each instance. Following that, we learn deconvolution filters by upsampling the feature maps at each layer and finally computing spatial softmax probabilities in a supervised manner.

	Mean IoU (%)		Mean IoU (%)
Fast R-CNN + Deconv	62.7	MSRA-CFM	61.8
VKN-Hole	59.7	FCN-8s	62.2
VKN-Hole + Skip	61.4	TTI-Zoomout	64.4
Fast R-CNN + Deconv + FCCRF	66.8	R-CNN	47.9
VKN + FCCRF	65.5	SDS	52.6
VKN + Skip + FCCRF	64.3	Deeplab-CRF	66.4

Table 4.1 Qualitative comparison of our proposed network models with some of the previous works on this problem. The networks are trained on PASCAL VOC 2012 'val' set along with augmented 'train' set. The comparison is measured on the 'test' set. Note that the metrics for MSRA-CFM, FCN-8s,TTI-Zoomout and R-CNN are taken directly from their literatur.

Figure 4.13 Qualitative comparison of our best network model (Fast R-CNN + Deconv + FCCRF) with some of the previous works.) Row (1) shows the original image from the PASCAL VOC 2012 'test' set. Row (2) shows the results of SDS [32]. Row (3) shows the results of Deeplab with CRF [10]. Row (4) shows the results from our proposed Fast R-CNN and Deconvolution based network model. Row (5) and (6) shows the class level and object level pixel annotations. (The colors for object level pixel annotations do not have any relevance to the object and just to signify that two pixels are different). (Best viewed in color)

Chapter 5

Conclusions and Future Work

In the first part of the thesis, we explained the problem of pose estimation of known 3D objects from a single image. As observed from the section, the problem is not at all trivial because the absence of depth data, one of the degrees of freedom is basically unknown. Hence we came up with a robust hierarchical method for estimating the orientation of objects in complex and clusttered scene images. To explain in brief, we use any generic segmenter or region proposal method to obtain initial estimates of the objetcs from the images. The training data captures the initial estimates of their positions and orientation. After extracting the seed pose along with the class labels of the objects and their rough regions, we use robust shape features which give us a correspondences between the obtained region proposals and the raw contours from known 3D object from the initial pose. Using the sets of features, we prune out false correspondences by clustering features which are very close to each other signifying that they are very strong matches. Using the final set of strong matches, we use Iterative Closest Point algorithm to estimate the transformation matrix between the rough proposal and the initial estimate. Using the transformation, we bring the initial estiamte closer to the actual groundtruth pose. The qualitative and quantitative comparisons shows that our method outperforms previous state-of-the-art method on pose estimation.

As we have seen in the part of the problem, obtaining region proposals from the image is a very crucial step in the whole process of hierarchical pose estimation. Hence, in the second part of the thesis, we dive into a survey of various region proposal methods based on CNNs. The reason we choose CNN is merely because they have proven to be more effective compared to other hierarchical methods of region proposal methods or probabilistic graphical models. Apart from that, we also build a foundation of the various key concepts which are required to understand the problem of instance-level semantic segmenation using Deep Neural Networks. We are interested in instance-level semantic segmentation as we also want the instance of each pixel other than the class label. This information is crucial in the problem of pose estimation as two different objects of the same class should be proposed as two different regions. We come up with a robust method for instance level segmentation using an ensemble of two methods. In the first phase of the approach, we use a fast CNN based object classifier to obtain

bounding boxes around potential objects. Using the bounding box proposals, we learn a deconvolution network which learns the lower level characteristics of the objects in a supervised manner.

Related Publications

- Banerjee, Sudipto, Sanchit Aggarwal, and Anoop M. Namboodiri. ”Fine Pose Estimation of Known Objects in Cluttered Scene Images.” in Proceedings of Third IAPR Asian Conference on Pattern Recognition (ACPR), 2015

Bibliography

[1] S. Aggarwal, A. M. Namboodiri, and C. Jawahar. Estimating floor regions in cluttered indoor scenes from first person camera view. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 4275–4280. IEEE, 2014.

[2] K. Alahari, G. Seguin, J. Sivic, and I. Laptev. Pose estimation and segmentation of people in 3d movies. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2112–2119. IEEE, 2013.

[3] J. M. Alvarez, T. Gevers, Y. LeCun, and A. M. Lopez. Road scene segmentation from a single image. In *European Conference on Computer Vision*, pages 376–389. Springer, 2012.

[4] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014.

[5] M. Aubry, D. Maturana, A. A. Efros, B. C. Russell, and J. Sivic. Seeing 3d chairs: exemplar part-based 2d-3d alignment using a large dataset of cad models. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 3762–3769. IEEE, 2014.

[6] M. Aubry, B. C. Russell, and J. Sivic. Painting-to-3d model alignment via discriminative visual elements. *ACM Transactions on Graphics (TOG)*, 33(2):14, 2014.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(4):509–522, 2002.

[8] M. Blum, J. T. Springenberg, J. Wülfing, and M. Riedmiller. A learned feature descriptor for object recognition in rgb-d data. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 1298–1303. IEEE, 2012.

[9] G. Borgefors. Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transactions on pattern analysis and machine intelligence*, 10(6):849–865, 1988.

[10] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014.

[11] O. Chum and J. Matas. Matching with prosac-progressive sample consensus. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 220–226. IEEE, 2005.

[12] D. C. Ciresan, U. Meier, J. Masci, L. Maria Gambardella, and J. Schmidhuber. Flexible, high performance convolutional neural networks for image classification. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1237, 2011.

[13] J. Dai, K. He, Y. Li, S. Ren, and J. Sun. Instance-sensitive fully convolutional networks. *arXiv preprint arXiv:1603.08678*, 2016.

[14] J. Dai, K. He, and J. Sun. Convolutional feature masking for joint object and stuff segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3992–4000, 2015.

[15] J. Dai, K. He, and J. Sun. Instance-aware semantic segmentation via multi-task network cascades. *arXiv preprint arXiv:1512.04412*, 2015.

[16] J. Dai, Y. N. Wu, J. Zhou, and S.-C. Zhu. Cosegmentation and cosketch by unsupervised learning. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1305–1312. IEEE, 2013.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[18] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In *Computer Vision–ECCV 2014*. Springer, 2014.

[19] I. L. Dryden and K. V. Mardia. *Statistical shape analysis*, volume 4. J. Wiley Chichester, 1998.

[20] D. Eigen and R. Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2650–2658, 2015.

[21] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010.

[22] A. Faktor and M. Irani. Co-segmentation by composition. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1297–1304. IEEE, 2013.

[23] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[24] A. W. Fitzgibbon. Robust registration of 2d and 3d point sets. *Image and Vision Computing*, 21(13):1145–1153, 2003.

[25] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Transactions on pattern analysis and machine intelligence*, (6):721–741, 1984.

[26] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[27] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, 2014.

[28] Google. 3D warehouse. `http://sketchup.google.com/3dwarehouse`.

[29] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *2009 IEEE 12th international conference on computer vision*, pages 1–8. IEEE, 2009.

[30] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. 2007.

[31] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. *International Journal of Computer Vision*, pages 1–17, 2014.

[32] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *Computer Vision–ECCV 2014*. Springer, 2014.

[33] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 447–456, 2015.

[34] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *Computer Vision–ECCV 2012*, pages 459–472. Springer, 2012.

[35] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *European Conference on Computer Vision*, pages 346–361. Springer, 2014.

[36] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán. Multiscale conditional random fields for image labeling. In *Computer vision and pattern recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE computer society conference on*, volume 2, pages II–695. IEEE, 2004.

[37] S. D. Jain and K. Grauman. Predicting sufficient annotation strength for interactive foreground segmentation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1313–1320. IEEE, 2013.

[38] R. Jonker and A. Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. *Computing*, 38(4):325–340, 1987.

[39] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 1725–1732, 2014.

[40] M. R. Kaus, S. K. Warfield, A. Nabavi, P. M. Black, F. A. Jolesz, and R. Kikinis. Automated segmentation of mr images of brain tumors 1. *Radiology*, 218(2):586–591, 2001.

[41] V. Kolmogorov. Convergent tree-reweighted message passing for energy minimization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 2006.

[42] V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. *Adv. Neural Inf. Process. Syst*, 2011.

[43] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[44] M. P. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 3217–3224. IEEE, 2010.

[45] K. Lai, L. Bo, X. Ren, and D. Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1817–1824. IEEE, 2011.

[46] Y. Lamdan and H. J. Wolfson. Geometric hashing: A general and efficient model-based recognition scheme. 1988.

[47] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[48] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[49] V. Lempitsky, A. Vedaldi, and A. Zisserman. Pylon model for semantic segmentation. In *Advances in neural information processing systems*, pages 1485–1493, 2011.

[50] Z. Li, E. Gavves, K. E. van de Sande, C. G. Snoek, and A. W. Smeulders. Codemaps-segment, classify and search objects locally. In *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 2013.

[51] J. J. Lim, A. Khosla, and A. Torralba. Fpm: Fine pose parts-based model with 3d cad models. In *Computer Vision–ECCV 2014*, pages 478–493. Springer, 2014.

[52] J. J. Lim, H. Pirsiavash, and A. Torralba. Parsing ikea objects: Fine pose estimation. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2992–2999. IEEE, 2013.

[53] M.-Y. Liu, O. Tuzel, A. Veeraraghavan, and R. Chellappa. Fast directional chamfer matching. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1696–1703. IEEE, 2010.

[54] S. Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[55] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[56] I. Lysenkov, V. Eruhimov, and G. Bradski. Recognition and pose estimation of rigid transparent objects with a kinect sensor. *Robotics*, page 273, 2013.

[57] I. Lysenkov and V. Rabaud. Pose estimation of rigid transparent objects in transparent clutter. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 162–169. IEEE, 2013.

[58] T. Malisiewicz, A. Gupta, and A. A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.

[59] A. K. Mishra and Y. Aloimonos. Visual segmentation of simple objects for robots. *Robotics: Science and Systems VII*, 2012.

[60] MIT. IKEA. `http://ikea.csail.mit.edu/`.

[61] G. Mori, S. Belongie, and J. Malik. Efficient shape matching using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1832–1837, 2005.

[62] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3376–3385, 2015.

[63] D. Munoz, J. A. Bagnell, and M. Hebert. Stacked hierarchical labeling. In *European Conference on Computer Vision*, pages 57–70. Springer, 2010.

[64] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814, 2010.

[65] P. K. Nathan Silberman, Derek Hoiem and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012.

[66] D. T. Nguyen. A novel chamfer template matching method using variational mean field. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 2425–2432. IEEE, 2014.

[67] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[68] N. Otsu. A threshold selection method from gray-level histograms. *Automatica*, 11(285-296):23–27, 1975.

[69] C. H. Papadimitriou and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[70] N. Payet and S. Todorovic. From contours to 3d object detection and pose estimation. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 983–990. IEEE, 2011.

[71] P. O. Pinheiro, R. Collobert, and P. Dollar. Learning to segment object candidates. In *Advances in Neural Information Processing Systems*, pages 1990–1998, 2015.

[72] Princeton. Sun 3D. `http://sun3d.cs.princeton.edu/`.

[73] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[74] A. Richtsfeld, T. Morwald, J. Prankl, M. Zillich, and M. Vincze. Segmentation of unknown objects in indoor environments. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4791–4796. IEEE, 2012.

[75] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[76] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[77] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE transactions on pattern analysis and machine intelligence*, 30(7):1270–1281, 2008.

[78] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Computer Vision–ECCV 2012*, pages 746–760. Springer, 2012.

[79] R. Sim and G. Dudek. Learning visual landmarks for pose estimation. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 3, pages 1972–1978. IEEE, 1999.

[80] R. Sim and G. Dudek. Learning environmental features for pose estimation. *Image and Vision Computing*, 19(11):733–739, 2001.

[81] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[82] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel. Bigbird: A large-scale 3d database of object instances. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 509–516. IEEE, 2014.

[83] A. Torralba, K. P. Murphy, W. T. Freeman, and M. A. Rubin. Context-based vision system for place and object recognition. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 273–280. IEEE, 2003.

[84] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *International journal of computer vision*, 2013.

[85] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.

[86] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3936–3943. IEEE, 2014.