Deep Neural Networks for Semantic Segmentation

by

Abhishake Kumar Bojja
B.Tech., Indian Institute of Technology (Indian School of Mines) Dhanbad, 2015

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

Master of Science

in the Department of Computer Science
University of Victoria

Deep Neural Networks for Semantic Segmentation

by

Abhishake Kumar Bojja
B.Tech., Indian Institute of Technology (Indian School of Mines) Dhanbad, 2015

Supervisory Committee

Dr. Kwang Moo Yi, Co-Supervisor
(Department of Computer Science)

Dr. Andrea Tagliasacchi, Co-Supervisor
(University of Victoria, Google Research and University of Toronto)

Supervisory Committee

Dr. Kwang Moo Yi, Co-Supervisor
(Department of Computer Science)

Dr. Andrea Tagliasacchi, Co-Supervisor
(University of Victoria, Google Research and University of Toronto)

ABSTRACT

Segmenting image into multiple meaningful regions is an essential task in Computer Vision. Deep Learning has been highly successful for segmentation, benefiting from the availability of the annotated datasets and deep neural network architectures. However, depth-based hand segmentation, an important application area of semantic segmentation, has yet to benefit from rich and large datasets. In addition, while deep methods provide robust solutions, they are often not efficient enough for low-powered devices. In this thesis, we focus on these two problems. To tackle the problem of lack of rich data, we propose an automatic method for generating high-quality annotations and introduce a large scale hand segmentation dataset. By exploiting the visual cues given by an RGBD sensor and a pair of colored gloves, we automatically generate dense annotations for two-hand segmentation. Our automatic annotation method lowers the cost/complexity of creating high-quality datasets and makes it easy to expand the dataset in the future. To reduce the computational requirement and allow real-time segmentation on low power devices, we propose a new representation and architecture for deep networks that predict segmentation maps based on Voronoi Diagrams. Voronoi Diagrams split space into discrete regions based on proximity to a set of points making them a powerful representation of regions, which we can then use to represent our segmentation outcomes. Specifically, we propose to estimate the location and class for these sets of points, which are then rasterized into an image. Notably, we use a differentiable definition of the Voronoi Diagram based on the softmax operator, enabling its use as a decoder layer in an end-to-end trainable network. As

rasterization can take place at any given resolution, our method especially excels at rendering high-resolution segmentation maps, given a low-resolution image. We believe that our new HandSeg dataset will open new frontiers in Hand Segmentation research, and our cost-effective automatic annotation pipeline can benefit other relevant labeling tasks. Our newly proposed segmentation network enables high-quality segmentation representations that are not practically possible on low power devices using existing approaches.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my advisors Dr. Kwang Moo Yi and Dr. Andrea Tagliasacchi, for the continuous support of my M.Sc study and related research, for their patience, motivation, and immense knowledge. Their guidance helped me in all the time of research and writing this thesis. I could not have imagined having better advisors and mentors for my M.Sc study.

Besides my advisors, I would like to thank Dr. Madeleine McPherson for being on the committee and providing insightful comments and encouragement.

I would also like to thank Motion Metrics and Nuance Communications for providing me an opportunity to work as a Machine Learning Engineer intern, where I got practical experience in the field.

I thank my fellow labmates for the stimulating discussions, and for all the fun we have had in the last two years. I want to thank Daniel Rebain for his valuable insights on my project and for his help in reviewing my thesis. I also would thank Weiwei for being a great labmate.

I especially thank Sri Raghu Malireddi for being a great friend and a mentor. I am grateful for his guidance towards completing my Master's journey and starting my professional career. I want to thank my friend, Patibandla Brahmendra Sravan Kumar, for introducing Machine Learning to me and paving a path for my career.

Also, I thank my friends Karan Tongay, Sai Prakash Reddy Konda, Sunil Kumar, Sharoff at the University of Victoria for being part of a great journey. I am grateful to Deepak Kumar and Prashanti Priya Angara for filling my life outside the lab and made me feel like a home away from home.

Last but not least, I would like to thank my family: my parents and my sister for supporting me spiritually throughout writing this thesis and my life in general.

DEDICATION

This thesis is dedicated to all my well-wishers - parents, friends and my mentors.

Chapter 1

Introduction

Semantic segmentation is a critical task in Computer Vision, where we assign a label for every pixel in the input image, and this label represents the category of the object to which the pixel belongs to. It is one of the high-level tasks of computer vision, which gives a complete understanding of a given scene [23]. Scene understanding has become very important due to its emerging applications in the fields of Autonomous Driving [18], Virtual Reality [91], image search engines [83] and Human-Computer Interaction [57]. In the past, this problem was tackled using traditional Computer Vision and Machine Learning techniques [59, 54, 16, 51, 33, 8, 62, 77, 49]. However, the recent success of Deep Learning techniques for Computer Vision shows that deep neural networks outperform traditional methods in the task of semantic segmentation with higher accuracy and efficiency. The success of deep learning is much due to the availability of high computational power, annotated datasets, and the deep neural network architectures to solve different tasks. Therefore, having rich large scale labeled datasets and developing efficient deep neural network architectures have been an ongoing endeavour since the introduction of Deep Learning [45, 15, 38, 26, 42].

In this thesis, we focus mainly on two contributions regarding the task of semantic segmentation. In the first part of the thesis, we discuss automating the process of creating large-scale datasets for depth-based hand segmentation dataset and propose a method to annotate automatically with reduced human effort. In the second part, we focus on a novel deep neural network architecture for semantic segmentation on resource constraint devices using Voronoi Diagrams.

1.1 Automating the creation of the hands dataset

Hand gestures are a natural way for humans to interact with the surrounding environment, and as such, many researchers have focused on obtaining accurate hand poses [17, 78]. Recently, as depth cameras have become more accurate and affordable [34, 21], significant progress has been achieved towards this objective [79, 57, 76]. In many cases, the first step in obtaining certain hands poses is to find *where* the hand is in the image, preferably as accurately and robustly as possible. In *hand segmentation*, the detection happens at pixel-level accuracy.

Several heuristic approaches for simplifying the task of hand segmentation have been proposed [57, 79, 72]. Such methods are suitable for small laboratory experiments but do not have the requisite robustness to operate in the full range of real-world interactions. One could learn a hand segmenter from a dataset of annotated depth images. However, as we will show, the limited size and quality of the datasets, which are currently available result in segmenters that typically overfit to the training data, and do not generalize well to the scenarios not seen during training. Due to the small size of available datasets, the application of real-time hand segmentation has received less attention from deep learning.

Therefore, a central goal is to capture a significantly *large* dataset loaded with annotations of superior quality ground truth. To accomplish this, we suggest an automated procedure for creating high-quality hand segmentation annotations from depth data per pixel, and implementing a large-scale dataset that we collected and annotated using the method proposed. As shown in Figure 1.1, our data capture setup consists of an RGBD camera and a colored gloves pair. We obtain this dataset from multiple users, and each user wears the gloves and perform hand motions before the camera. To generate annotations with superior quality ground truth, we then use the color and depth channels of the images from the camera feed with minimal intervention of an annotator.

Note that the only additional equipment necessary for data acquisition is a colored gloves pair, compared to the sophisticated setups used for hand capture (magnetic sensors [90] or optical IR markers [25]). Moreover, the quality of the dataset is much better than the ones that use motion capture sensors, as these methods require an additional heuristics to generate pixel-wise annotations for training a hand segmenter [86]. To the best of our knowledge, our dataset is the only one that provides both qual-

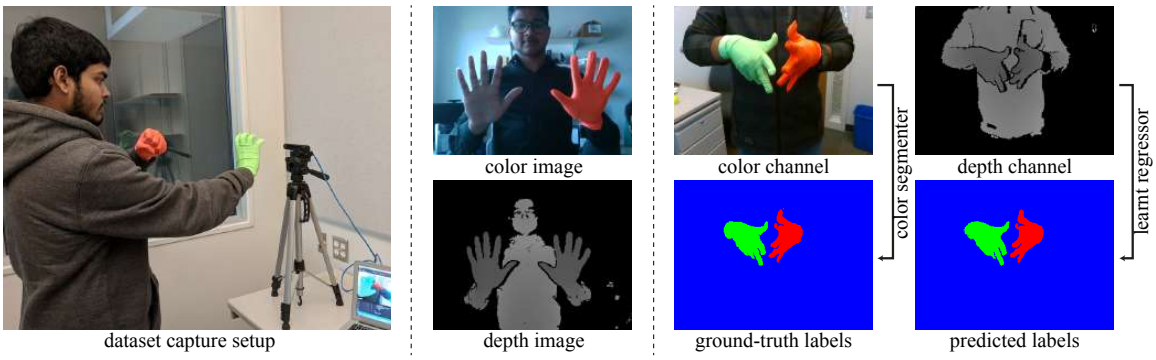

Figure 1.1: Proposed data capture and automatic annotation framework. **(Left)** Our dataset is constructed by recording a user performing hand movements wearing a pair of brightly colored gloves in front of a depth camera. To the best of our knowledge, our dataset is the first *two-hand* dataset for hand segmentation. **(Middle)** The use of tight colored gloves provide a *quasi* non-invasive automatic annotation system, as the signal-to-noise ratio of a conventional depth sensor is not sufficiently high to distinguish between gloved and bare hands. **(Right)** Color images that are aligned with the depth images are exploited to compute ground truth labels without user intervention. We then quickly filter out the few wrongly labeled images through human inspection. We can subsequently use these input-label pairs to train a depth-based semantic segmenter.

ity and quantity of higher magnitude (see Table 1.1). We also provide an in-depth analysis of the effect of using our dataset on multiple neural network architectures for hand segmentation, as well as traditional Random Forests due to their computational efficiency. We empirically find that using *strided [transposed-]convolutions* in place of [un]pooling layers, and the use of skip-connections is essential for achieving high accuracy. This further enables efficient forward-passes within $\approx 5ms$ on an NVIDIA Geforce GTX1080 Ti, making our approach suitable for real-time applications. We discuss our dataset and automatic annotation pipeline in more detail in Chapter 3.

1.2 Voronoi Diagrams as segmentation representations

It is ubiquitous to solve the problem of semantic segmentation on a pixel-by-pixel basis. In pixel-based segmentation, the classification is done at the pixel level. For each pixel in the input image, a label is predicted. Some methods [55, 66, 4, 43, 13]

Table 1.1: Existing and proposed datasets for *exocentric* hand segmentation from depth imagery. Our dataset is the only real dataset that distinguishes the two hands. Furthermore, our capture setup does not require expensive sensors as in the other two real datasets; see text for more details.

Dataset	Annotations	#Frames	#Subj	Hand	Sensor Type	Resolution
Freiburg [96]	synthetic	43,986	20	left/right	Unreal Engine	320 × 320
NYU [81]	automatic	6,736	2	left	Kinect v1	640 × 480
HandNet [86]	heuristic	212,928	10	left	RealSense SR300	320 × 240
Proposed	automatic	210,000	13	left/right	RealSense SR300	640 × 480

predict the label map at the input resolution. In contrast, other methods [45, 94, 91] predicts the label map at a smaller resolution and then upsample the output with a bilinear upsampler to get the output at the desired input resolution. If the input is a high-resolution image, predicting each pixel is very inefficient, particularly for semantic segmentation task, where the pixels belonging to the same object share the same labels. Moreover, making predictions on deep neural networks require a lot of resources for high-resolution inputs making the current methods [45, 55, 66, 4, 94, 43, 91, 13] inefficient for semantic segmentation on low power devices. We approach this problem in a novel way with the help of Voronoi Diagrams.

Voronoi Diagrams partitions the entire space into discrete regions based on a set of points. Based on this definition of the Voronoi Diagram, they are a very natural way to represent the segmentation label map, which is also an image divided into regions based on object categories. We therefore use Voronoi Diagrams to generate semantic labels. Specifically, we propose a Neural Network module which we call the Voronoi Decoder. We use a differentiable definition of a Voronoi Diagram based on the softmax operator and use it as a module in an end-to-end trainable network. In other words, we form an end-to-end trainable encoder-decoder network, where an encoder estimates the location and class for a set of points, and the Voronoi Decoder, which acts as a decoder layer, rasterizes an image from the encoder information.

Our method brings multiple benefits. The proposed Voronoi Decoder module is simple and can be used on top of any popular semantic segmentation networks to render a segmentation label map. In a traditional semantic segmentation problem, one estimates the label pixels for every pixel in the input image. However, in our case, we

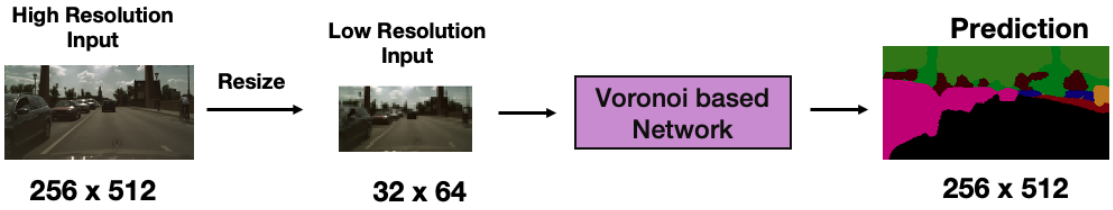

Figure 1.2: Proposed voronoi based network architecture. Our network can produce a high resolution map 256×512 given a small resolution input 32×64.

estimate the location and class information of those points. As rasterization can take place at any given resolution, our method especially excels at rendering high-resolution segmentation maps, given a low-resolution image as shown in Figure 1.2. Therefore, it reduces the computational requirements for obtaining high-resolution segmentation maps significantly. This helps to produce segmentation in real-time on low power devices. We evaluate our method on Cityscapes dataset [14] and compare our method with OCNET [91], one of the best semantic segmentation network. We show that our results are better than OCNET, both qualitatively and quantitatively. We will discuss our method in more detail in Chapter 4.

1.3 Key contributions

In summary, the following are the main contributions in this thesis:

- **Hand Segmentation Dataset**: To support the research of depth-based hand segmentation, we introduce a large scale hands dataset with separate annotations for both the hands. Our dataset is the first-ever dual hand dataset rich in quantity as well as quality. We even evaluated our dataset on multiple segmentation networks and proposed to use a lightweight network for real-time hand segmentation. We also open-sourced our dataset to support the research of hand segmentation.

- **Automatic labeling method:** To ease the process of dataset creation for depth-based hand segmentation, we introduce a method to label the dataset automatically, and create a large scale dataset.

- **Novel method for real-time segmentation on low power devices:** To solve real-time semantic segmentation on resource constraint devices, we propose a new deep neural network architecture based on Voronoi Diagrams. This method suits well for resource-constrained environments and can produce high-resolution segmentation maps given a low-resolution input.

1.4 Overview

The rest of this thesis document is as follows:

Chapter 1 gives a brief introduction about the datasets for hand segmentation and algorithms for semantic segmentation.

Chapter 2 gives a brief overview of the related work on the datasets of hand segmentation and the procedure for generating those datasets. We also discuss the existing methods for semantic segmentation and introduce literature related to Voronoi Diagrams.

Chapter 3 briefly describes our method to annotate the hand segmentation dataset automatically. We will report the performance of different semantic segmentation networks and compare them with our proposed network for hand segmentation.

Chapter 4 discusses our new method for real-time semantic segmentation. We train and evaluate our new model on the Cityscapes dataset and report the results. We even show how our method can work for image generation task.

Chapter 5 provides some final remarks.

Chapter 2

Related Work

2.1 Semantic Segmentation

Recent advances in deep learning methods, particularly convolutional neural networks (CNN), demonstrated that they could be applied to solve the problem of semantic segmentation of objects and scenes can be successfully, and the current state-of-the-art model employs CNN. Long et al. [46] proposed Fully Convolutional Neural Networks (FCNN) for semantic segmentation, where an input of arbitrary size is encoded into a low dimensional latent space using FCNNs and then decoded to original size with bilinear upsampling. Here, they perform segmentation by classifying every local region in the image as a coarse label map from the network, and a simple deconvolution is employed using bilinear upsampling for pixel-level labeling. Due to the fixed size receptive field, the system can work only to a single semantic scale within the image, and if the object in the picture is larger or smaller than the receptive field, then it can be mislabelled. As an extension to this work, Noh et al. [56] employed the Encoder-Decoder strategy and named as DeconvNet, where the model learns to encode the input image to lower dimension embeddings using FCNNs and decodes to the original size using a learned deconvolution network, which comprises deconvolution and un-pooling layers. The model predicts pixel-wise class labels and thus predict segmentation masks. DeconvNet [56] is a large model with many parameters, mainly due to the use of two fully connected layers between Encoder and Decoder, and so, it is inefficient to train. SegNet [2] consists of an encoder-decoder network followed by a pixel-wise classification layer. In FCNN, DeconvNet, SegNet, the Encoder consists of a series of convolutions

and max-pooling operations. As opposed to FCNN, DeconvNet and SegNet use a learnt decoder for generating segmented masks. Both DecovNet and SegNet employs an un-pooling operation that inverts the max-pooling in the Encoder and upsamples the feature maps via memorized max-pooling indices in the corresponding encoder layer. UNet [67] is also an encoder-decoder network with additional skip connections, which carry intermediate feature representations from encoder layers to the decoder layers. Due to the skip connections, it provides additional information from Encoder to the Decoder, which helps to outperform FCNN, DeconvNet, and SegNet. Our network for hand segmentation is based on this UNet architecture.

Most of the recent works on semantic segmentation address the issues, where the output feature maps resolution of the model is smaller than input resolution [91]. Researchers in OCNet[91], PSPNet[94], DeepLabV3[13] use Dilated Convolutions to solve this problem. However, still, the output feature maps size is not equal to the input size. OCNet employs a bilinear upsampler to produce the final output segmentation map. As the bilinear upsampler approximates the pixels while going from lower resolution to higher resolution, the boundaries of the segmentation may not be sharp and reduces the segmentation performance. Our work suggests replacing the bilinear upsampler with a Voronoi Decoder, which can rasterize the image at any resolution without any issue with the boundaries.

In the following sections, we discuss existing research methods, datasets, and deep neural architectures for one of the areas of computer vision, depth-based hand segmentation, and discuss the new deep networks for semantic segmentation. Then we discuss Voronoi Diagrams, which is the basis of our new architecture representation.

2.2 Hand Segmentation

In this section, we discuss the research methods, datasets, and techniques researchers employ to solve the problem of hand segmentation.

2.2.1 Different Approaches to Hand Segmentation

The impressive work of Oikonomidis et al. [58], uses a 3D hand model and performs hand tracking using Particle Swarm Optimization. They approached the problem using skin color segmentation and also assumes that the user wears long sleeves shirt

with only the hand, which exposes in the image. Romero et al. [65] segments hands in HSV space by thresholding skin color. Tzionas et al. use Gaussian mixture models for segmentation based on skin color [82]. In [85], Wang et al. use the color histogram of the first frame in modeling a probabilistic method for segmenting hands. But, these skin color-based segmentation models cannot work in environments involving objects in skin color, other body parts, different skin tones, and different lighting conditions. So using a color glove is a better alternative [84].

In [48], Melax et al. uses dynamics simulation of a 3D hand model for hand tracking and assumes that a camera can track everything which lies within its field of view by making use of short-range depth sensors. Whereas, in [57], Oberweger et al. require only hand to be present in the frame, and it should be close to the camera. Several depth map based methods [64, 31, 32], use a black wristband to segment hands. In [79], Tagliasacchi et al. uses a colored wristband, measure the region of interest as point cloud sets attached to the wrist. In contrast, [72] uses a wrist position of a full-body tracker to identify the region of interest. Using a wristband is inconvenient, although it makes the problem simple and effective. Furthermore, since these methods use connected components for segmentation purposes, they cannot segment hands interacting with objects. So, all these methods come with some inherent assumptions and do not work if the premise fails.

2.2.2 Datasets for Hand Segmentation

As datasets play a vital role in any Computer Vision task, there are some valuable datasets contributed by the research community for hand segmentation task. Some researchers [9, 5, 19, 50, 96] provided color image-based datasets, where the input is a color image. Buehler et al. and Bambachet al. [9, 5] provided pixel-level manually annotated ground truth for respectively ≈ 500 and $\approx 15k$ color images. Annotating segmentation masks manually from color images is extremely labor-intensive, which makes it even difficult to collect large-scale datasets. Some methods [81, 96, 75, 90] used this approach to create a depth-based hand datasets. The dataset size of [81] is only $\approx 7k$. Zimmermann et al. [96] dataset is synthetic, and its size is $\approx 44k$, which is less to train deep neural networks. Moreover, as the dataset is synthetic, the trained models on this data may not generalize well to the real data [71, 29, 97].

2.2.3 Neural network architectures for Hand Segmentation

Learned encoder-decoder architectures have been shown to perform well on semantic segmentation [93, 41, 61, 60], but when fast inference time is essential, random forests are an excellent alternative due to their easy parallelization [37, 73]. In human pose estimation applications, Shotton et al. [74] inferred body part labels via random forests, which was later adopted for hand localization from depth images by Tompson et al. [81], and color images by Zimmermann and Brox [96]. Recently, [80] employed a convolutional neural network to estimate two-hand segmentation masks for hand tracking. In multi-view setups, effective segmentation provides a strong cue for effective tracking [44], and the two tasks can even be coupled into a single optimization problem [35]. Predicted segmentation masks can be noisy and/or coarse, and post-processing is typically employed to remove outliers by regularizing the segmentation [12]. A recent approach by Kolkin et al. [36] accounts for the severity of mis-labeling by a loss encoding their spatial distribution, but this method has yet to be generalized to a multi-label classification scenario like ours. Relevant to our hand segmentation work is also the recent R-CNN series of works, of which the instance segmentation work by He et al. [27] represents the latest installment. While combining bounding box localization with dense segmentation could be effective, it is however unclear to which extent such networks could be adapted to demanding real-time applications such as hand tracking.

2.3 Recent Networks for Semantic Segmentation

2.3.1 Topology Maintained Structure Encoding

In mathematics, Topology is the analysis of the properties that are retained by surface deformations, twists, and stretches. It is important to preserve the boundaries and global contours of the objects while we perform segmentation. A key part of designing a neural network for semantic segmentation is its encoder. As mentioned in 2.1, Encoder-Decoder networks are popular for solving semantic segmentation networks. Convolutional neural networks are the commonly used Encoder networks that do a great job of extracting high-level information. Still, they mostly fail to maintain the topological properties in the image like connection structures and global contours [22]. Fang et al. [22] proposes a Voronoi Diagram Encoder, which improves the extraction

Figure 2.1: This image is taken from Fang et al.[22] In the first col (a), the initial state of Voronoi edges is presented by yellow lines; then the authors detect potential edges of the input image and fit Voronoi edges to boundaries (b); used a labeling algorithm is proposed to merge the cells in the same object(hole) and remove redundant edges (c).

of topological properties in a Convolution neural network and is based on a convex set distance. They used this encoder for image generation tasks using Generative Adversarial Neural networks [24]. The results in the paper are compelling and show the capability of Voronoi Diagrams in maintaining the structures of objects. An example result form the paper is shown in Figure 2.1. However, Voronoi based encoder networks are to be tested on other computer vision tasks like image segmentation, where object boundaries play a key role in the success of the algorithm.

2.3.2 Context based Networks

Context

Context plays a vital role in many computer vision tasks. It helps to understand more about the input scene. It exists in various forms like geometric context, scene context, 3D context, and is chosen according to the problem one deals with. For semantic segmentation, context is a set of pixels belonging to the same class-label. Recent works PSPNet [94], OCNet, ParseNet [43] uses context and it plays an important role for these networks. Here we detail more about the Object Context Module used in

OCNet and our network.

Object Context Module

Object context is a set of pixels belonging to the same class-label. An object context module helps to cluster the pixels belonging to the same object category by using representations of pixels. The success of the OCNet network in giving a state-of-the-art performance in semantic segmentation task is due to the object context module. The module consists of two stages, namely object context estimation and object context aggregation.

Object context estimation. In this stage, each pixel p in the object is represented in a vector form $\mathbf{x}_p$ and similar pixels are extracted by correlating a pixel with every other pixel. Object context map w_p is inspired from self-attention [92] and is defined as follows in [91]:

$$w_{pi} = \frac{e^{(f_q(\mathbf{x}_p)^\top f_k(\mathbf{x}_i))}}{\sum_{i=1}^{W\times H} e^{(f_q(\mathbf{x}_p)^\top f_k(\mathbf{x}_i))}} \ , \tag{2.1}$$

here $\mathbf{x}_p$ and $\mathbf{x}_i$ are the representation vectors of the pixels p and i respectively.

The representation vector $\mathbf{x} \in \mathbb{R}^{C\times N}$, where C is the number of channels and N is the number of feature locations. $f(\cdot)$ is a transformation function applied to representation vectors. $f_q(\cdot)$ is the query transform function and $f_k(\cdot)$ is the key transform function, which are used to calculate the attention as in [92], where $\boldsymbol{f_q}(\mathbf{x}) = \boldsymbol{W_q}\mathbf{x}, \boldsymbol{f_k}(\mathbf{x}) = \boldsymbol{W_k}\mathbf{x}$. $\boldsymbol{W_q} \in \mathbb{R}^{\bar{C}\times C}$, $\boldsymbol{W_k} \in \mathbb{R}^{\bar{C}\times C}$ are the learned weight matrices, which are implemented as 1×1 convolutions. The query and key transform functions project the representation vectors into the common space. As we don't know the ideal transform function, we let the network learn the function during training.

The equation 2.1 is a Softmax function, W and H are the width and height of the input feature map for the object context module.

Object context aggregation. The object context of a pixel p is then constructed by combing weighted representations of all other pixels and is defined as follows:

$$\mathbf{c}_p = \sum_{i=1}^{W\times H} w_{pi}\phi(\mathbf{x}_i) \ , \tag{2.2}$$

Here $\phi(\cdot)$ is the value transform function and $\phi(\mathbf{x}) = \boldsymbol{W_h}\mathbf{x}$. $\boldsymbol{W_h} \in \mathbb{R}^{\bar{C}\times C}$ is a learned

weight matrix, which is implemented as 1×1 convolutions. $\phi(\cdot)$ provides the weight to the pixels based on the similarity factor of pixel i with pixel p. w_{pi} is the object context map defined in the equation 2.1.

2.4 Auto Encoders

Auto encoders [39, 70, 28] are a family of neural networks that try to reproduce the input. It follows an Encoder-Decoder network architecture. The Encoder **E** takes an input image $\mathcal{I}_n$ to produce a latent representation (lambda), which is further passed to a Decoder **D** to reproduce the input $\mathcal{I}_n$. And then, the entire network is trained through back propagation algorithm by minimizing the following reconstruction loss.

$$\mathcal{L}_{\text{rec}} = \sum_{n} \mathbb{E}\left[\|\mathcal{I}_n - (\mathbf{D}\left(\mathbf{E}\left(\mathcal{I}_n\right)\right))\|_2\right] \ , \tag{2.3}$$

where $\mathcal{I}_n$ is the input and ground truth image. $\mathbb{E}$ is the Expectation and $\|.\|_2$ is the L2 (Mean Square Error) loss.

Generally, **E** and **D** are feed-forward networks. Auto Encoders are also used in Dimensionality reduction as the network **E** represents the entire input with the latent representation.

2.5 Voronoi Diagrams

As defined in [10], a Voronoi Diagram is the partition of a plane X into n different regions, and each region contains a generator. The generator within the region is the nearest generator for any given point within the region. As an example, we can see in Figure 2.2, the plane is divided into 8 Voronoi regions, and each contains a Voronoi generator. The Voronoi generator is represented by P_i, where i is the index of the i^{th} region and $i \in [1, 8]$. $V(P_i)$ represents the Voronoi Region, that contains every point closest to P_i. The Voronoi region, $V(P_k)$, associated with the site P_k is the set of all points in X whose distance to P_k is not greater than their distance to the other sites P_j, where j is any index different from k. A Voronoi Edge separates any two Voronoi Generators at exactly halfway, and a Voronoi vertex is the intersection of Voronoi Edges.

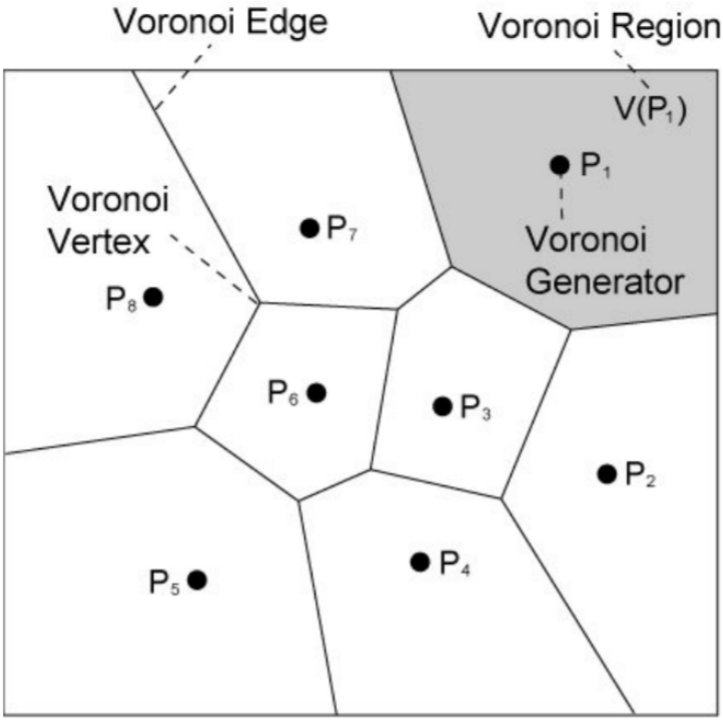

Figure 2.2: Decoding Voronoi Diagram: [10]

2.5.1 Approximating Functions on an image with Restricted Voronoi Diagrams

Nivoliers et al. [53] introduces a method to compute an approximation function of a 3D mesh/image by dividing the mesh/image into different regions/samples, which is a Voronoi Diagram and then optimize both the positions of the samples and the associated approximation values. Here the optimization is done on a single image/mesh basis, which is a significant limitation.

2.5.2 Voronoi based geometric approach for Classification

Polianskii et al. [63] introduces an optimization free geometric approach to the classification problem using Voronoi Diagrams. As Voronoi cell decompositions can be used for the classification task, this method provides additional information about the Voronoi cell boundaries, which is generally omitted in performing classification using Voronoi Cells due to the heavy computation involved in computing Voronoi

diagrams. The authors propose a Monte-Carlo integration-based approach, which computes a weighted integral over the boundaries of Voronoi cells and thus requires less computation than before. This additional geometric information helps to improve the classification performance.

2.5.3 Voronoi based Image Descriptor

The authors in [11] proposed new image descriptors using Voronoi Diagrams termed as Voronoi-based vector of locally aggregated descriptors (VLAD) for the region of interest (RoI) image retrieval. To retrieve an RoI of an image, in addition to the content-based descriptor, the authors propose to use a Voronoi based spatial partitioning for each image in the dataset. This addition makes their method to be descriptor agnostic and gives a competitive performance to the current best practices to solve this task.

2.5.4 Voronoi Diagrams for resampling 3D mesh

Alliez et al. [1] use centroidal Voronoi Diagrams, where the Voronoi generators are present at the centroid of Voronoi regions, for remeshing of triangulated surface mesh. Triangle mesh is the most commonly used 3D geometrical data structure in Computer Graphics to model any object. The modeling of an object is done by a uniform sampling of the given object. But, some of the parts of the object are undersampled or oversampled due to the uneven surfaces. So, resampling of the object is done for better modeling of the object. The authors use the initial sampling to form a weighted Voronoi diagram to create a perfect mesh.

Chapter 3

An Automatically Labeled Dataset for Hand Segmentation from Depth Images

In this chapter, we explain our approach for generating premium quality annotations automatically for depth-based hand segmentation, and introduce our large scale hand segmentation dataset. The following chapter is adapted from our published work [7].

We first discuss the data acquisition process and then explain our automatic annotation method for generating labels in Section 3.1. We then discuss various machine learning models that were applied for solving the problem of segmentation in Section 3.2.1. We next present evaluation of those models on our generated and other publicly available hand segmentation datasets in Section 3.2.2.

3.1 Data acquisition and automatic annotation

For scalable annotation with minimal human interaction, we rely on the synchronized color/depth input of an RGBD device, and a pair of skin-tight brightly colored gloves. As shown in 1.1(middle), this allows a (quasi) non-invasive and cost effective setup, where we can automatically determine ground-truth labels at pixel level. As the gloves fit the user's hand tightly, minimal geometric aberration to the depth map occurs, while the consistent color of the glove can be used to extract the hand ROI via color segmentation. After an initial color calibration session, we ask the user to perform

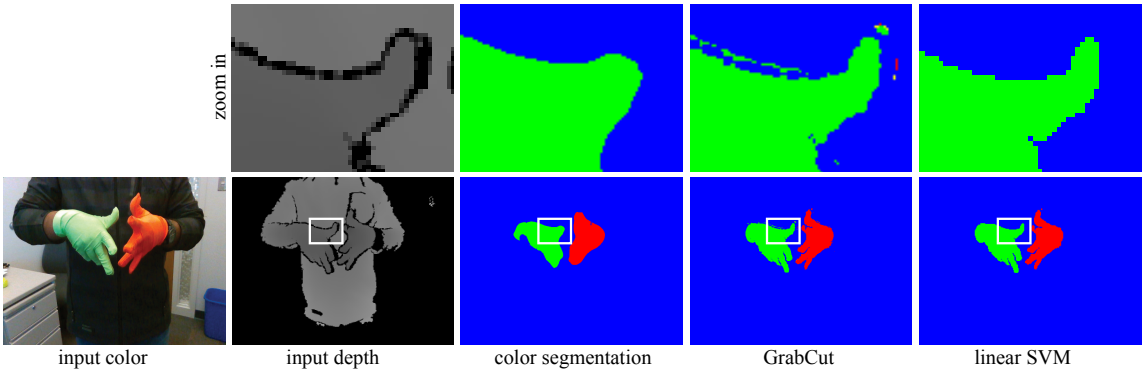

Figure 3.1: Our automatic annotation pipeline. **(Bottom)** input images, the output of each segmentation step. **(Top)** Zoom in to highlight segmentation accuracy. We employ the color image to create ground truth annotations for depth. We first segment the color image via HSV thresholding, then perform GrabCut [69] to obtain better segmentation. We finally train a per-image linear support vector machine (SVM) with RGB, HSV, XYZ, Lab color spaces, as well as image coordinates as input cues to further refine the annotation. Note how the segmentation becomes more accurate as each step is performed. Through this three-stage process we are able to obtain highly accurate ground-truth annotations without manual annotations. See text for details.

a few motions similar to [47], and record sequences of (depth,color) image pairs at a constant 48Hz rate with an Intel RealSense SR300. The users have performed different poses for different hands and even provided some complex cases, where fingers of both hands are overlapping each other. We further move the camera during the capture process to enrich the dataset with various viewpoints. We then execute a color segmentation to generate masks with a very small false positive rate; we finally quickly discard contiguous frames containing erroneous labels via manual inspection of video – this task is *significantly* simpler than manually labeling individual images. In our process we roughly drop 10% of the automatically labeled images, selected conservatively to avoid any wrong label in the dataset. The final HandSeg dataset consists of 210,000 frames collected from 13 users. The dataset contains annotations for both left and right hands, and each frame is of 640 $\times$ 480 resolution.

3.1.1 Automatic label generation

As illustrated in Figure 3.1, we perform color segmentation through a three-stage procedure. The quality of the labeling is enhanced at each stage of the pipeline.

Initial color segmentation. We first perform color space thresholding to obtain a rough segmentation S_r and S_l of the two hands, where r and l denote right and left hands respectively. We will denote both S_r and S_l together as S_*. Specifically, to obtain S_*, we threshold on the HSV color space after smoothing the input image with a Gaussian kernel (with standard deviation $\sigma = 30$) to remove noise. The threshold values we use for our experiments are minimum and maximum values of $[3, 160, 100]$–$[15, 255, 255]$ for left hand, and $[28, 35, 100]$–$[70, 200, 255]$ for right hand, where [H, S, V] denotes the HSV values and $H \in [0, 180]$ while $S, V \in [0, 255]$.

Refinement through GrabCut [69]. As the initial segmentation is coarse due to the initial Gaussian filtering, we further apply GrabCut [69], followed by a linear support vector machine (SVM) classifier [20] to get a more fine-grained segmentation map; see Figure 3.1. We determine the seed points for GrabCut by first finding the bounding boxes R_* for each hand, and then using all points of S_* that are inside R_*; to be robust to noise, we enlarge R_* by 10 percent. At this stage, some of the labels are still inaccurate, especially near the boundaries of the hands.

Refinement through linear SVM [20]. To further enhance the labels, we exploit the high distinctiveness of the glove's color, and train a linear SVM classifier *per image* with a large enough margin, and use the positives that are classified as positives during training as ground-truth labels. Note that this classifier is simply a per-image refinement process that automatically sets-up the per-image thresholds for a simple colour-based thresholding system, based on the GrabCut results. For robust performance we use RGB, HSV, XYZ and Lab color values as well as image coordinates as cues to linear SVM. We also empirically set the hyper-parameter $C = 900$ (margin strength).

3.2 Experiments

3.2.1 Learning to segment hands

Similar to [47, 7], we used the Random Forests, Fully Convolutional Neural Network, DeconvNet, SegNet architectures to evaluate our dataset. Our architectures are shown in the Figure 3.2.

HandSeg network architecture. Our architecture is a U-Net like structure that

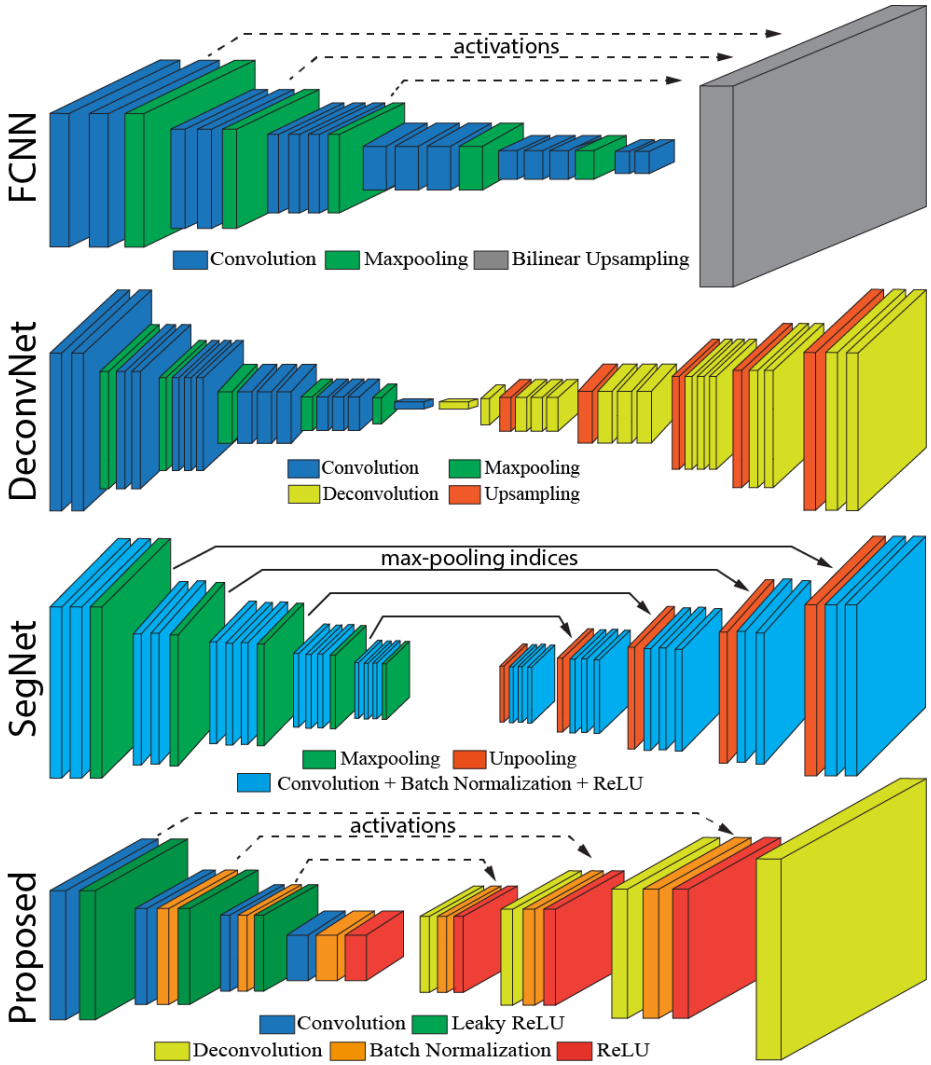

Figure 3.2: Semantic segmentation CNN architectures. Image taken from [47, 6]

containing an encoder, decoder and skip connections from intermediate layers of encoder to decoder, which help to improve the sharpness in the predictions. Pooling layers are helpful in classification tasks but destroy the spatial structure which is important for the segmentation task. Thus, the max-pooling/unpooling layers in the encoder/network layer are replaced with stride-2 convolution/deconvolution layers.

3.2.2 Evaluation

We quantitatively evaluate our dataset with various methods from three different angles. In Section 3.2.4, we evaluate how different methods perform on our data in

Table 3.1: Runtime of each segmentation method. Ours is the fastest to train and test amongst compared deep architectures.

	Random Forests	FCN	DeconvNet	SegNet	**Proposed**
Train time	3h	149h	57h	83h	29h
Test time	1ms	41ms	16ms	30ms	5ms

terms of mean Intersection over Union (mIoU), as well as their runtime both during training and testing. In Section 3.2.5, we show the generalization capabilities of several datasets, including ours. In all our experiments, the dataset was split randomly in a 8:1:1 ratio to form train, validation, and test sets.

3.2.3 Evaluation metrics

In our multi-label classification problem, each pixel can be classified as {left, right, background}. Within each class, we can have *true-positives* (TP), *false-positives* (FP) and *false-negatives* (FN). Given such a categorization, we use the *Intersection over Union*, defined as IoU $= |TP| \,/\, (|TP| + |FP| + |FN|)$, for quantitative evaluation. As in [95], to aggregate results for multiple classes, we use the class-wise average among classes, that is, mean IoU (mIoU). This is to account for the imbalance in the number of pixels for each class. IoU measures the number of pixels common between the ground truth labels and the prediction labels divided by the total number of pixels present in both the labels. It is simply defined as:

$$IoU = \frac{\text{ground truth} \cap \text{prediction}}{\text{ground truth} \cup \text{prediction}} \,. \tag{3.1}$$

3.2.4 Segmenting with different architectures

In Figure 3.3, we compare the different learning approaches in terms of accuracy, and their runtime in Table 3.1. For the runtime experiments, all deep networks were run on a single NVIDIA Geforce GTX 1080 Ti graphics card. In these experiments, we did not distinguish between left and right hands, as DeconvNet and SegNet completely failed due to the class imbalance between left hand, right hand, and background labels. Although Random Forests are clearly the fastest to train and to infer on,

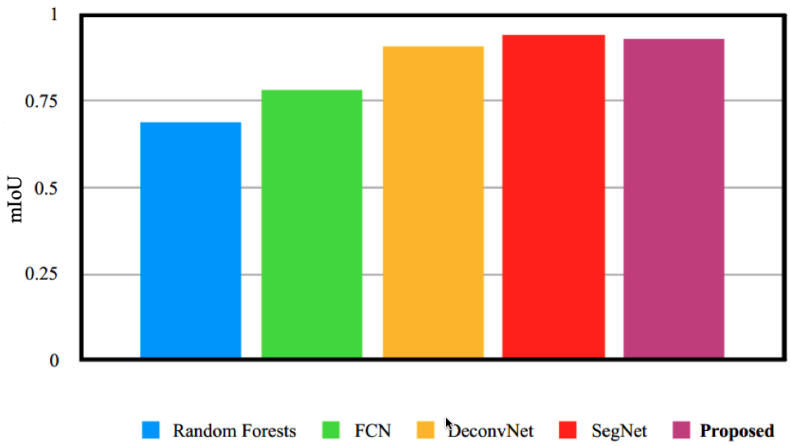

Figure 3.3: Performance of different segmentation methods on our dataset in terms of mIoU (higher is better). Evaluation is performed on a two-class setup, where left and right hands are not distinguished. Otherwise, DeconvNet and SegNet fail to learn. However, the proposed network is able to achieve state of the art in any case.

they perform poorly when compared to deep networks. Due to its simple upsampling scheme, FCN(32s) performs the worst among the evaluated networks. Thanks to its learned decoder network, DeconvNet and SegNet obtain much better results. However, their architectures are too computationally complex, resulting in a runtime that is not suitable for real-time tracking applications when considering that segmentation is typically a pre-processing step for a sophisticated vision pipeline. Our proposed architecture not only on par with the best performing method in terms of accuracy, but it is also fast to forward-propagate, running at $\approx$ 200fps.

3.2.5 Cross-dataset evaluation

We test our baseline network by training/testing on all possible combinations of the datasets in Table 1.1 (with the exception of NYU which is captured using a deprecated sensor). We also include BigHands [90] as a dataset for training, which is a hand pose dataset composed of more than a million images. However, as this dataset is originally intended for hand pose estimation, it does not have per pixel labels. We therefore apply GrabCut [69] with the hands' joint locations as seed points to obtain a rough segmentation label. As these labels are not perfect, this dataset cannot be used for testing. As not all datasets distinguish left and right hands, we perform evaluations for the two-class (hands vs. background) as well as the three-class (left

Table 3.2: Generalization performance across datasets for the **three-class setup**, in terms of mIoU. For BigHands, we use data augmentation to generate both left and right hand labels. Segmenter trained on our dataset, *HandSeg*, performs best in terms of generalization.

Train / Test	**HandSeg** (Ours)	Freiburg	BigHands
HandSeg (Ours)	0.877	0.437	0.492
Freiburg	0.574	0.870	0.408

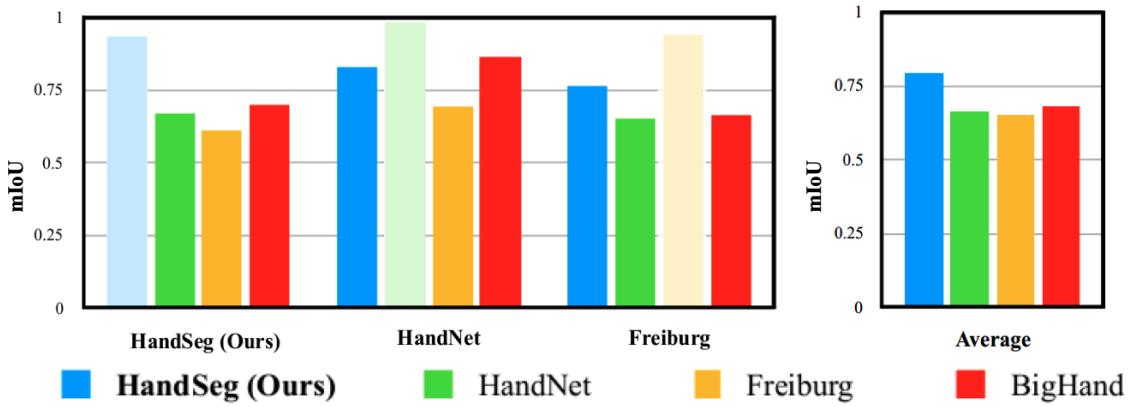

Figure 3.4: Generalization performance across datasets for the **two-class setup**, in terms of mIoU. The dataset used for training is color-coded by the legend at the bottom, and the results are grouped by each test set. The washed-out colors denote the case when trained and tested on the same dataset. On the right, we show the average performance of segmenters trained on each dataset, when tested on other datasets excluding the one used for training. Note that the segmenter trained on our dataset, *HandSeg*, generalizes best on average and on Freiburg, and is on par with the best generalizing dataset on HandNet.

vs. right vs. background) scenario. We summarize the results in Figure 3.4 and in Table 3.2, respectively.

As shown in Figure 3.4, when left and right hands are not distinguished, the segmenter trained with our dataset generalizes better (in average) than when trained with other datasets. Furthermore, as shown in the results on each testing dataset, the segmenter trained on our dataset performs either the best or comparably to the best method, while simultaneously generalizing to unseen datasets.

In Table 3.2, we show the case when the two hands are distinguished. This three-class case is harder than the two-class case above, as the segmenter now has to distinguish between left and right hands. As shown, the segmenter trained with our dataset generalizes better to Freiburg, than the one trained with Freiburg on ours. Considering that the test performance on both datasets is similar, this shows the better generalization capability of our dataset. Furthermore, as the BigHands dataset only features a single hand, data augmentation needs to be applied for the three-class setup, which is the result shown in Table 3.2. The poor numbers clearly demonstrate the need of a hand segmentation dataset that distinguishes left/right hands.

3.2.6 Qualitative evaluation

In Figure 3.5, we provide qualitative segmentation results on our novel dataset. Here, we show results of the proposed architecture, FCN, and the Random Forest. We excluded SegNet and DeconvNet, as for the three-class experiments, these two network architectures failed to deliver any meaningful results on our dataset and converged to a trivial solution, that is, all pixels considered as background. Note how the proposed architecture shows the best performance.

Figure 3.6 shows challenging frames where our network does not deliver perfect results. Sample #1 illustrates how the network can still segment the hands of multiple persons, although it was trained on frames containing a single individual. This reveals the generalization capabilities of our network, which did not only learn to segment *one/two* regions, but also learned a latent *shape-space* for human hands. Sample #2 shows a person holding a cup, while Sample #3 and Sample #4 have the hand lying flat on the body. These scenarios are difficult, as the network has never seen a hand interacting with objects. Although not perfect, the network successfully segments the hands in Samples #3 and #4, but fails on the cup for Sample #2. Accuracy could be improved by accounting for the additional information in the color channel, or by learning the appearance of the object via training examples.

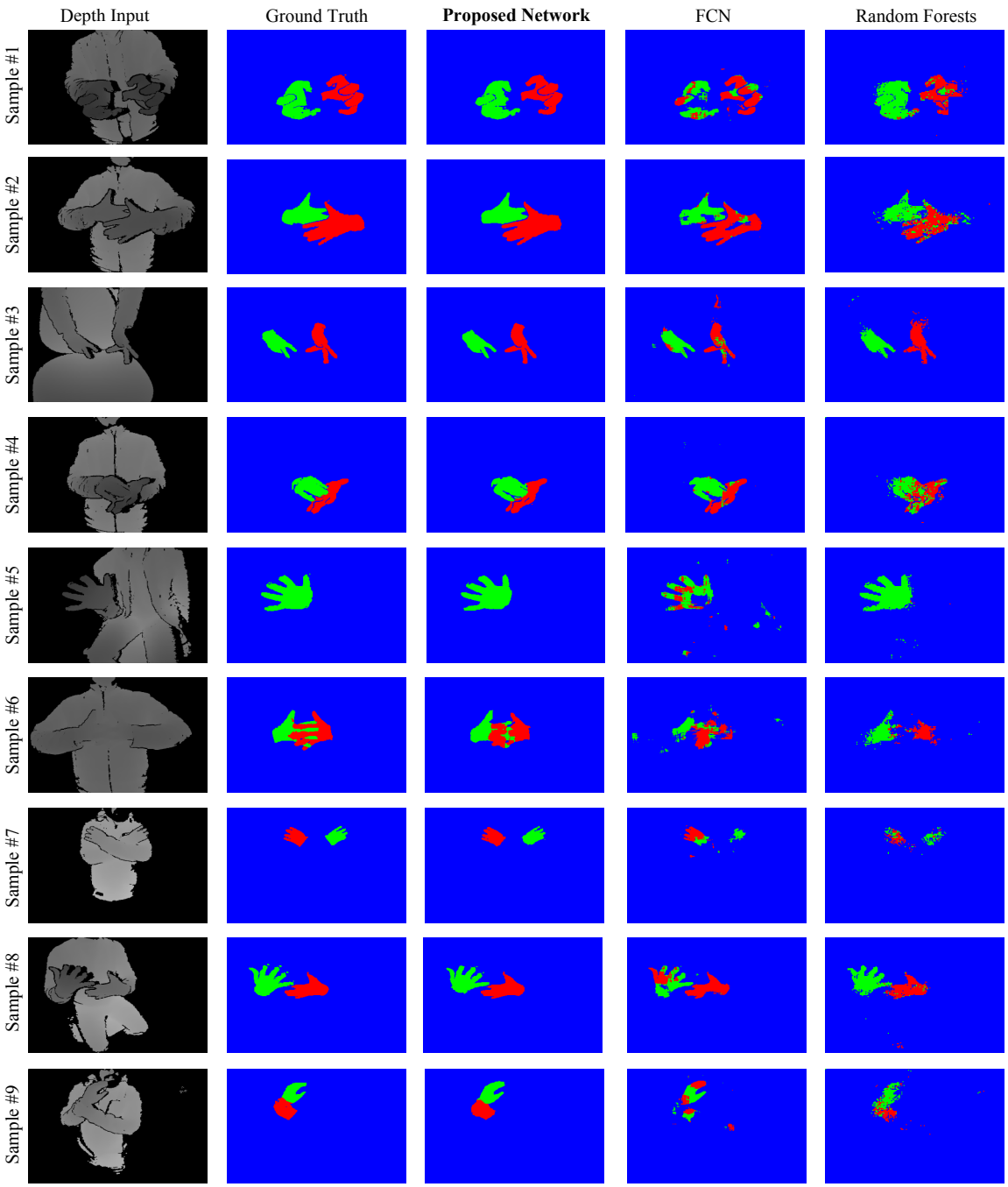

Figure 3.5: Qualitative examples. We illustrate a few examples of hand segmentation performance on our dataset for the proposed network, FCN, and Random Forests. We exclude SegNet and DeconvNet here as they converge to estimating all pixel as the background for this setup. Note how the proposed network gives accurate segmentation for diverse poses, including when the hands are interacting as shown in Sample #4. Sample #6 shows a failure case of our network when there's extreme interaction between the two hands. Still, our architecture performs better than the compared ones, giving relatively accurate segmentation.

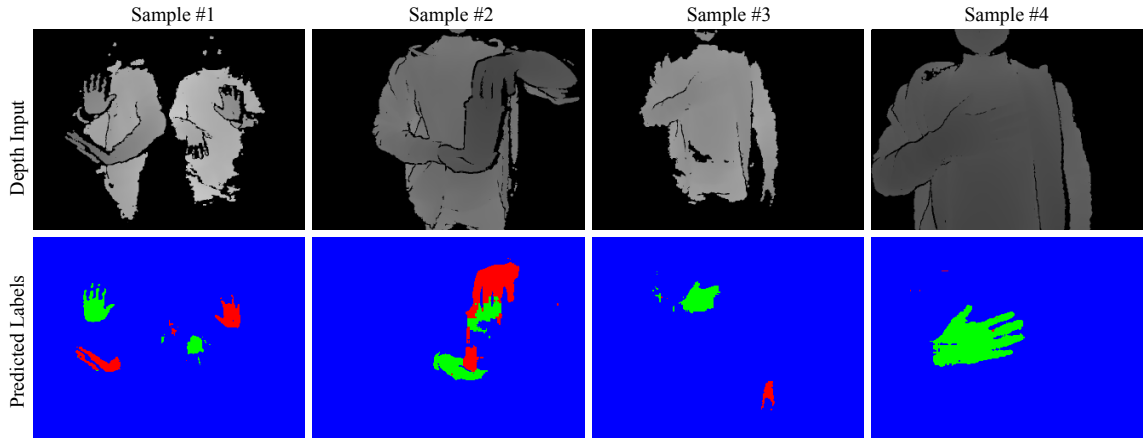

Figure 3.6: A selection of segmentation failure cases. Due to the challenging nature of these examples, our segmenter does not return perfect results. Note that in Sample #1, our network is able to segment all four hands, although it was never trained with more than a single person in the field of view. In Sample #2, our network show error on the cup, as the network never saw hands interacting with objects.

Chapter 4

Segmentation using Voronoi Diagrams

In this chapter, we explain our novel method for semantic segmentation using Voronoi Diagrams. We first discuss our Voronoi Decoder module in Section 4.1. Next, present the results by applying our method to the MNIST Image generation task in section 4.2. We then give our end-to-end deep neural network and then explain the evaluation of our network and other best segmentation network on Cityscapes Dataset and show the results in section 4.3.

4.1 The Voronoi Decoder

The core contribution of our method is the Voronoi decoder. Similar to [53] and [88], we want to learn a family of functions that maps every pixel in a 2D image to a constant value of a Voronoi region in a Voronoi diagram. The method first represents the image with a given number of Voronoi generators and their corresponding constant values. Then, the Voronoi function (Voronoi Decoder) renders a 2D image.

The Voronoi Decoder is a function that can be queried at any given location x of an input. Let us say we have n random Voronoi generators/points in a 2D plane with the point set

$$\mathbf{P} = \left\{ \mathbf{p}_n \in \mathbb{R}^2 \right\}_n \ ,$$

where each point $\mathbf{p}_n$ which maps to a corresponding constant value λ_n belonging to a

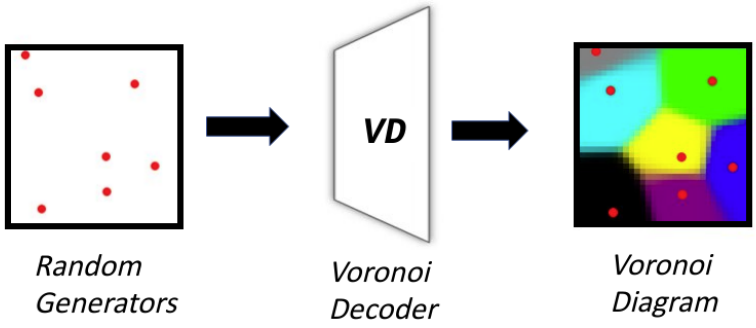

Figure 4.1: Rendering Voronoi Diagram from points using Voronoi Decoder (VD)

color set C, where $\mathbf{C} = \left(\lambda_n \in \mathbb{R}^l\right)_n$ and l is the number of different colors. We define the Voronoi function as:

$$\mathcal{V}(\mathbf{x}|\mathbf{C},\mathbf{P}) = \mathbf{C}\left[\arg\min_n \left\{\|\mathbf{x}-\mathbf{p}_n\|_2\right\}\right]. \tag{4.1}$$

Algorithm: For sampling x, we first create a two-dimensional mesh grid with a shape similar to a 2D plane. Then, we calculate the distances from each pixel x, in the mesh grid to every generator and find the k nearest generators. Let the k distances be given as

$$D_k(\mathbf{x}|\mathbf{P}) = \|\mathbf{x}-\mathbf{p}_k\|_2 \ ,$$

The *argmin* from the equation 4.1 makes the Voronoi function non differentiable because it makes sharp boundaries for the Voronoi edges. We smoothen the Voronoi edges by replacing *argmin* with *soft-argmin* with the help of vector **S**.

$$\mathbf{S}_k(\mathbf{x}|\mathbf{P},\theta) = e^{-\theta D_k(\mathbf{x})}/\Sigma_k e^{-\theta D_k(\mathbf{x})} \ , \tag{4.2}$$

where $\theta \in \mathbb{R}^+$ is a temperature parameter, which helps to control the smoothness in Voronoi edges. So the Voronoi function in the equation 4.1 now becomes

$$\mathcal{V}(\mathbf{x}|\mathbf{C},\mathbf{P},\theta) = \mathbf{C}\cdot\mathbf{S}(\mathbf{x}|\mathbf{P},\theta) \ , \tag{4.3}$$

The output of softmax (S) is multiplied with the colors (C) to generate a Voronoi diagram, as shown in Figure 4.1. In the output diagram, we can observe that all the pixels belonging to a single Voronoi region are represented with smooth transitions at the region edges. The Voronoi Decoder can render the output image at any resolution

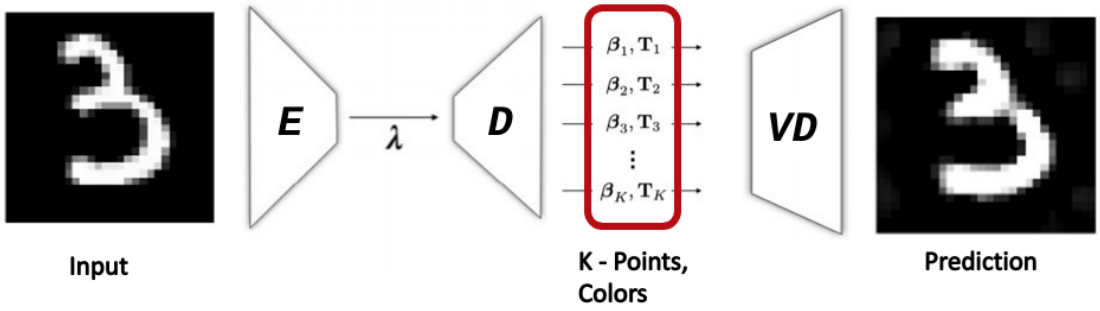

Figure 4.2: MNIST Image Generation using Voronoi based Auto Encoder Network. Here E is Encoder, λ is Latent Space Representation, D is Decoder and VD is Voronoi Decoder. k is the number of parameter pairs produced by D and (β_i, T_i) corresponds to i^{th} parameter pair. β represents (x, y) coordinate and T represents (r, g, b) color information.

without much loss of information and thus supports super-resolution.

4.2 MNIST Image Generation Task

First, we discuss the MNIST Image generation task, where we apply our neural network to the MNIST dataset. Here we treat our proposed neural network as an Auto-Encoder, which takes an input image and generates an output image the same as the input.

4.2.1 Data

MNIST [40] dataset contains handwritten digit images. The dataset contains 60,000 training images and 10,000 test images. In our Image generation task, we use the same input image as a ground truth image while training. The input and prediction image resolution is 28×28.

4.2.2 Architecture

As shown in Figure 4.2, the Voronoi based Auto Encoder contains three networks, namely Encoder (E), Decoder (D), and Voronoi Decoder (VD). The Encoder network

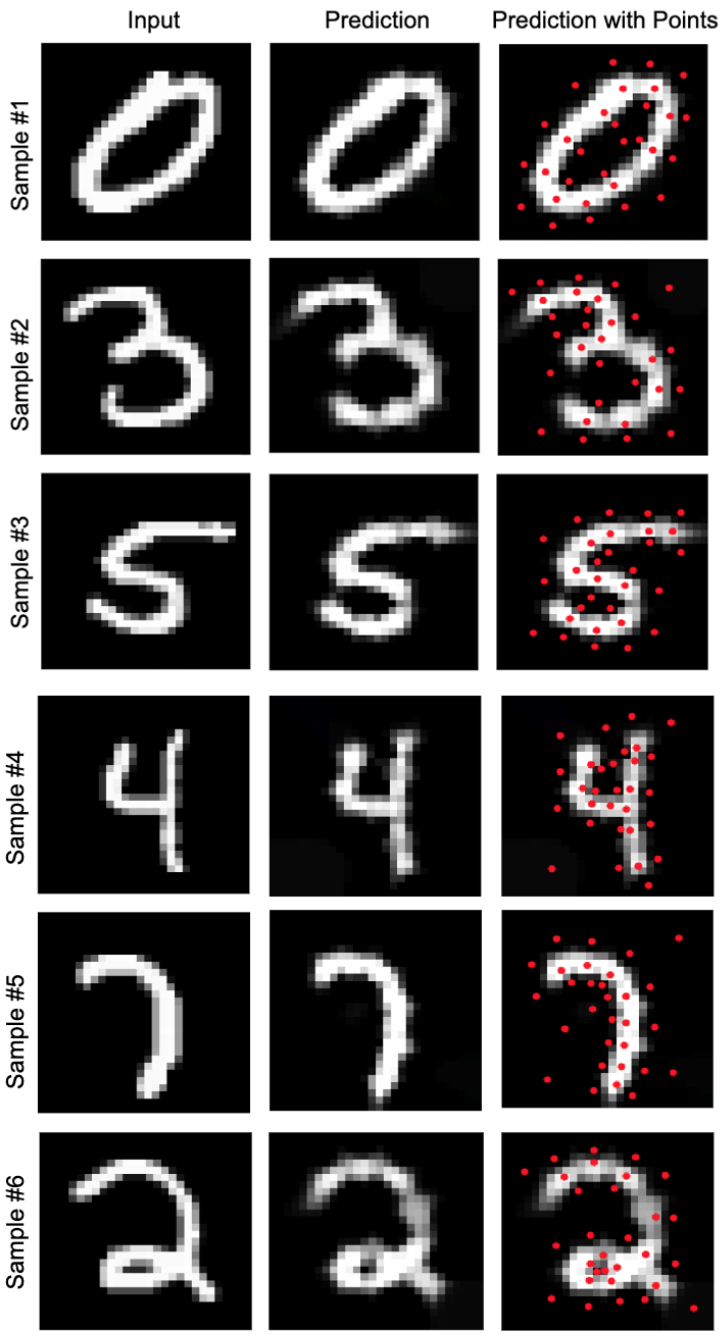

Figure 4.3: Image generation results from our proposed Voronoi based network. The left image is the input and the middle image is the prediction. The point coordinates from the decoder layer are overlapped on the prediction and are displayed on the right.

(E) takes an input image and produces an underlying representation of λ. Then the Decoder generates a collection of k Voronoi parameters containing points and their corresponding colors. Then the Voronoi Decoder (VD) renders the prediction image

at a given resolution using the method described in section 4.1.

The Encoder (E) network is a convolutional network with four blocks, where each block contains a series of Convolution, Group Norm, Leaky Relu layers, and a ResNet identity network [26]. The Decoder (D) is a multi-layer perceptron (MLP) network which contains four blocks, where each block includes a fully connected layer followed by a group norm and a leaky relu layer.

4.2.3 Loss Function

The entire network is trained by minimizing the following reconstruction loss.

$$\mathcal{L}_{\text{rec}} = \sum_{n} \mathbb{E}_{\mathbf{x}\in[0,1]^D} \left[\left\| \mathcal{I}_n - \mathbf{VD}\left(\mathbf{D}\left(\mathbf{E}\left(\mathcal{I}_n\right)\right)\right) \right\|_2 \right] , \tag{4.4}$$

where $\mathcal{I}_n$ is the input and ground truth image. $\|.\|_2$ is the L2 loss and $\mathbb{E}$ is the expectation on L2 loss over $\mathbf{x} \in [0,1]^D$.

We observe that some points lie outside the predicted image and are not utilized for rendering the prediction. Thus, we apply regularization to these points so that they are well spread, and at the same time to be inside the image. Specifically, we define auxiliary boundary points across the boundary of the image and use Laplacian smoothing loss to avoid predicting points that lie outside the boundary defined by the auxiliary points.

The boundary points are chosen along the perimeter of the image, with $\lceil \sqrt{n} \rceil$ points on each border of the image, with n being the number of predicted points from the decoder network. This provides equal spacing of the boundary points as the grid of predicted points and the equal spacing balances the Laplacian loss at the boundaries. Since the prediction image in our case is a square, the total number of boundary points is equal to $4\lceil \sqrt{n} \rceil$.

We define the Laplacian smoothing loss as:

$$\mathcal{L}_{\text{lap}} = \frac{1}{n} \sum_{i=1}^{n} \left\| p_i - \frac{1}{N} \sum_{j=1}^{N} p_j \right\|_2 , \tag{4.5}$$

where n is the number of points, N is the number of adjacent points to p_i, p_j is the position of the j-th adjacent point to p_i in the Delaunay triangulation of the points.

Notice how this loss makes the predicted points shrink towards the auxiliary points and by doing so on all points including the borders, it tries to spread points across the image.

To summarize, the total loss combining both reconstruction and Laplacian is defined as:

$$\mathcal{L}_{\text{total}} = \lambda_{rec}\mathcal{L}_{\text{rec}} + \lambda_{lap}\mathcal{L}_{\text{lap}} \ , \tag{4.6}$$

where λ_{rec} and λ_{lap} are the hyperparameters used to balance the total loss, which we set empirically.

4.2.4 Training Settings

The dimension of the latent space representation λ in our experiments is 128. The number of points is 30, and the value of θ is 30. The hyperparameters λ_{rec} and λ_{lap} values are 1.0 and 0.01 respectively. The evaluation metric we used for this task is the mean square error, which was defined in Equation 4.4.

4.2.5 Results

Image generation results from the MNIST network are shown in Figure 4.3. For a given input image (on the left), we visualize the prediction (middle), prediction with points (on the right) overlapping on it. We can see the predictions (middle) are similar to the input images (on the left). From the right column, We can observe that the points are mostly concentrated at the digit pixels. For the *sample*#5, We can see that the points are distributed along the boundaries and in the middle of digit 7. With this example, we can understand that the model learned that most of the information is at the boundaries of white pixels and utilized all of its points to distribute around the digit 7 and interpolates to the nearest values while rendering the final digit 7. In this case, we use 30 points.

Intuitively, the points should be distributed around the object boundaries, which are the high-frequency areas. For the low-frequency areas, where all the neighbors share the same pixels, a few points are enough to represent the pixels in those object area and use interpolation to render the final values for those object regions. Our method

follows adaptive subdivision [87] in computer graphics, which is used to render the render high-resolution images by considering information only at the high-frequency areas. So, this means our method can represent all the similar pixels in an object with a few numbers of points and colors. In this way, we can represent an entire high-resolution image with some points and colors, thus can save a lot of memory in storing the images. These points and color values can be passed to a learned Voronoi Decoder to render the final output image.

4.3 Cityscapes Segmentation

4.3.1 Segmentation Task

Semantic segmentation is the task of labelling images at pixel level. It has a lot of applications ranging from scene understanding, inferring relationships among objects for autonomous driving. Progress in deep learning through the deep neural networks such as FCN [46], UNet [68], SegNet [3], ParseNet [43], DeepLabv3 [13], and OCNet [91] made a lot of progress in solving the task.

4.3.2 Cityscapes Dataset

We use the Cityscapes dataset [14] for semantic segmentation task, to understand information from a scene at the pixel level. The dataset contains 30 classes, and only 19 classes of them are used for scene parsing evaluation. The dataset includes 2,975 training, 500 validation, and 1,525 test images with high-quality pixel-level annotations. For the test dataset, labels are not publicly available. So, for our experiments, we use the validation set as the test set.

4.3.3 Input Data

For Training and evaluating our method, we use an image resolution of 256×512. Since our main aim is to develop the method for resource-constrained devices, the input to the neural network model is 32×64, but the rendered output is of original size 256×512. We evaluate our results with the given input and output image sizes when we compare them with other methods.

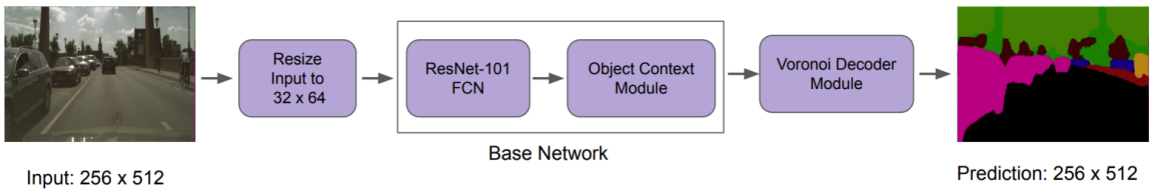

Figure 4.4: Our Overall Network architecture.

Since the input size for the model is very small in our case, we observed that if the number of pixels belonging to a particular class-label is very low, then it is challenging for the model to learn it. So, we chose an optimal threshold value for the number of pixels to consider the class label for Training. If the number of pixels for any class-label is less than the threshold value, then they are ignored. The threshold value in our case is 1 % of input size (32×64) for the model.

4.3.4 Network Architecture

We consider OCNET [91] network, which gives the state-of-the-art performance on the Cityscapes dataset and was discussed in section 2.3.2. The OCNET network uses an object context pooling network on top of a fully convolutional Resnet-101 [26] network to extract context information from the input pixels and assign a single class-label to all the pixels having a similar context. Our overall network architecture is shown in the Figure 4.4. It consists of a base network combining both backbone network and Object Context network and a Voronoi Decoder module. The base network is borrowed from OCNET and the Voronoi Decoder module defined in 4.1 – our main contribution – is appended at the end to decode the feature map at any resolution.

Backbone Network

The fully convolutional ResNet-101 [26] network, which was pre-trained on the ImageNet [15] dataset, is used as a backbone to our network. It takes an input image I of size $h \times w$ and produces a feature map S. The final two blocks of the network are replaced with dilated convolutions [89], with dilation rates of 2, 4 respectively. A dimensionality reduction module (a 3×3 convolution) is employed at the end to reduce the channels of the feature maps output from 2048 to 512.

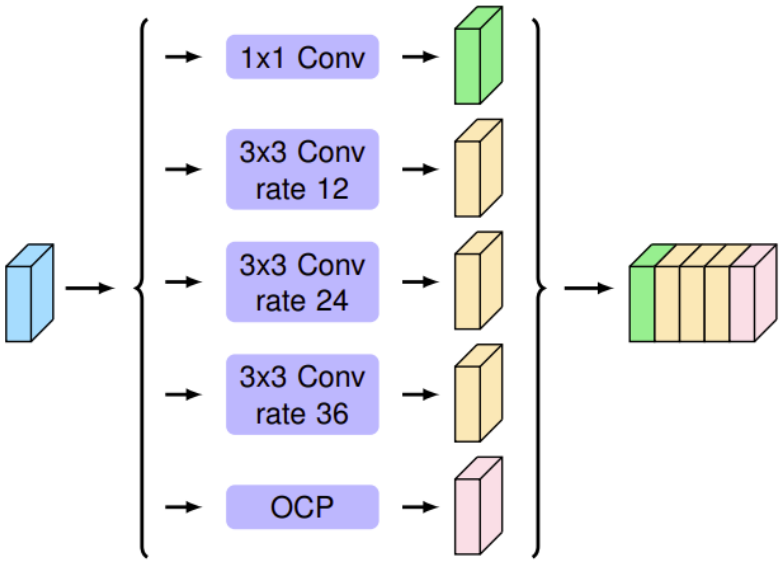

Figure 4.5: Object Context Network Module. Image taken from [91].

Object Context Network

The object-context module extracts the context information of objects from the input feature maps S to produce a feature map $\bar{S}$. The object context network we used is as shown in Figure 4.5. The output feature map from the backbone network is passed through five different modules, and the responses from each module are concatenated as a final output. The first module is a 1×1 convolution block. The next three modules are 3×3 dilated convolution with different dilation rates (12, 24, 36) respectively. The final module is an object context module. All the modules produce an output with 512 channels. The outputs from all the five modules are concatenated to produce an output feature maps with 2560 channels. The channels are further reduced to 512 by employing 1×1 convolutions.

Voronoi Decoder Module

Before feeding the feature map into the Voronoi Decoder Module, we employ a dimension reduction module (a 1×1 convolution) to reduce the channels of the feature maps output $\bar{S}$ from the base network defined in section 4.3.4 from 512 to $k = (n_c + 2)$, where $n_c = 19$, which is the number of labels in the cityscapes dataset used for our experimentation purposes and 2 is to represent (x, y) coordinates, which

are considered as offset points. These offset points are added to the meshgrid of the Voronoi Decoder. So, now the final feature map $\bar{V}$ contains both positional coordinate and color information. The Voronoi decoder module takes $\bar{V}$ with coordinates and colors as channel information as input and renders an output segmentation map of desired output size.

4.3.5 Loss function

We employed cross entropy loss (L_{ce}) on the final output of our model which is defined as below.

$$L_{ce}(y, p) = -\sum_{i} y_i \log(p_i) \ , \tag{4.7}$$

and

$$p_i = \frac{e^{a_i}}{\sum_{k=1}^{N} e_k^a} \ . \tag{4.8}$$

Here, the logits from the model's output a is passed through softmax function p_i and y is the ground truth label.

Our dataset has 19 semantic classes, and the number of pixels for each class is not the same in the entire dataset. Training directly on this dataset leads to a class imbalance problem, where the learned network performs better on the high-frequency semantic class rather than on the lower frequency class. Thus, we use class-balanced cross-entropy loss, where a higher weight is given to low-frequency classes and a lower weight to high-frequency classes.

As in Section 4.2.3, without regularization points are not spread and focus on a small portion of the image. For this task, we found that the method in Section 4.2.3 does not work well. We therefore use an alternative formulation, where we regularize by computing the offset of the estimated triangulation points from a regular mesh grid. In other words, we add penalty from deviating from a pixel-like representation.

In more detail, we first construct an auxiliary mesh grid, G of size $\sqrt{n}$ x $\sqrt{n}$ with n being the number of points covering the whole prediction image. We always choose n as a perfect square to make $\sqrt{n}$ an integer. We then predict the positions of points, and obtain P from the base network as offsets from this grid. Notice how we set the number of points in P, so that we can easily form a mesh grid for G. We then use an

L2 regularizer defined as:

$$\mathcal{L}_{\text{reg}} = \frac{1}{n}\sum_{i=1}^{n} \|p_i - g_i\|_2 \ , \tag{4.9}$$

where n is the number of points, g_i, and p_i are the k-th points on the grid G, and the predictions P. $\|.\|_2$ is the L2 loss.

As before in Section 4.2.3, our total loss function is defined as:

$$\mathcal{L}_{\text{total}} = \lambda_{ce}\mathcal{L}_{\text{ce}} + \lambda_{reg}\mathcal{L}_{\text{reg}} \ , \tag{4.10}$$

where λ_{ce} and λ_{reg} are the hyperparameters used to balance the total loss as before.

4.3.6 Training Settings

Similar to OCNET [91], we employ poly learning rate policy with an initial learning rate as 0.01 and weight decay as 0.0005. The number of points in our case is 100. We set the hyperparameters λ_{ce} and λ_{reg} to 1.0 and 0.01 respectively. The input to our model is (32×64) and the base network's (section: 4.3.4) output is (16×32). The final segmentation output is (256×512). For data augmentation, we flipped image and label pair horizontally.

Theta

In Equation 4.3, we mentioned that the purpose of theta (θ) is to control the smoothness in the Voronoi edges. To see the role of theta visually, we pass the same random generators through the Voronoi Decoder and plot the generated Voronoi Diagrams by varying θ from 0 to 90 in increments of 10. The resultant plots are shown in Figure 4.6. We can observe that when θ is small (between 0 to 30.0), the boundaries are smoother, and as it goes to 90.0, the boundaries become sharper. If the Voronoi edges are smoother, the colors are blurred, and there is a loss of information. But, if the edges are sharper, it makes the Voronoi Decoder non-differentiable. So, we need to choose the value of θ somewhere in between. From our experimentation, we found that θ's optimal value is 30.0. If θ's value is more than 30.0, the Voronoi edges becomes sharper and makes the gradients flow difficult, and if the value is smaller than 30.0, the edges are smoother which makes the network produce blurred output.

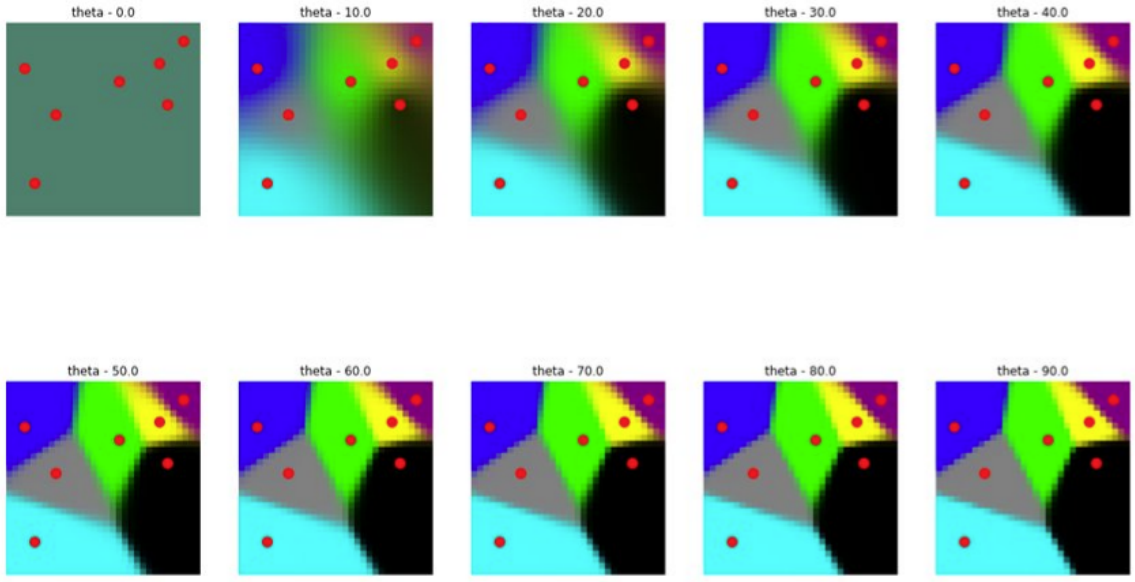

Figure 4.6: Role of theta (θ). The Voronoi edges are smoother when theta is less (First row in the image) and the boundaries become sharper with rise of theta (second row)

4.3.7 Evaluation Metrics

The evaluation metric in our experiments is mean Intersection Over Union (mIoU) as defined in the section 3.2.2.

4.3.8 Baselines

We compare our Voronoi based proposed network with the OCNET network. The architecture of the OCNET network used in our experimentation is as follows. The input image $\bar{I}$ is fed to the base network to produce an output feature map $\bar{S}$, which is of size (16×32). Then $\bar{S}$ passed through a bilinear upsampler layer, which upsamples the input to (256×512) segmentation map with nc channels.

4.3.9 Quantitative and Qualitative Results

On the Cityscapes test dataset, quantitative results are shown in Table 4.1. We can see that our model quantitatively achieves a better performance than OCNET. The

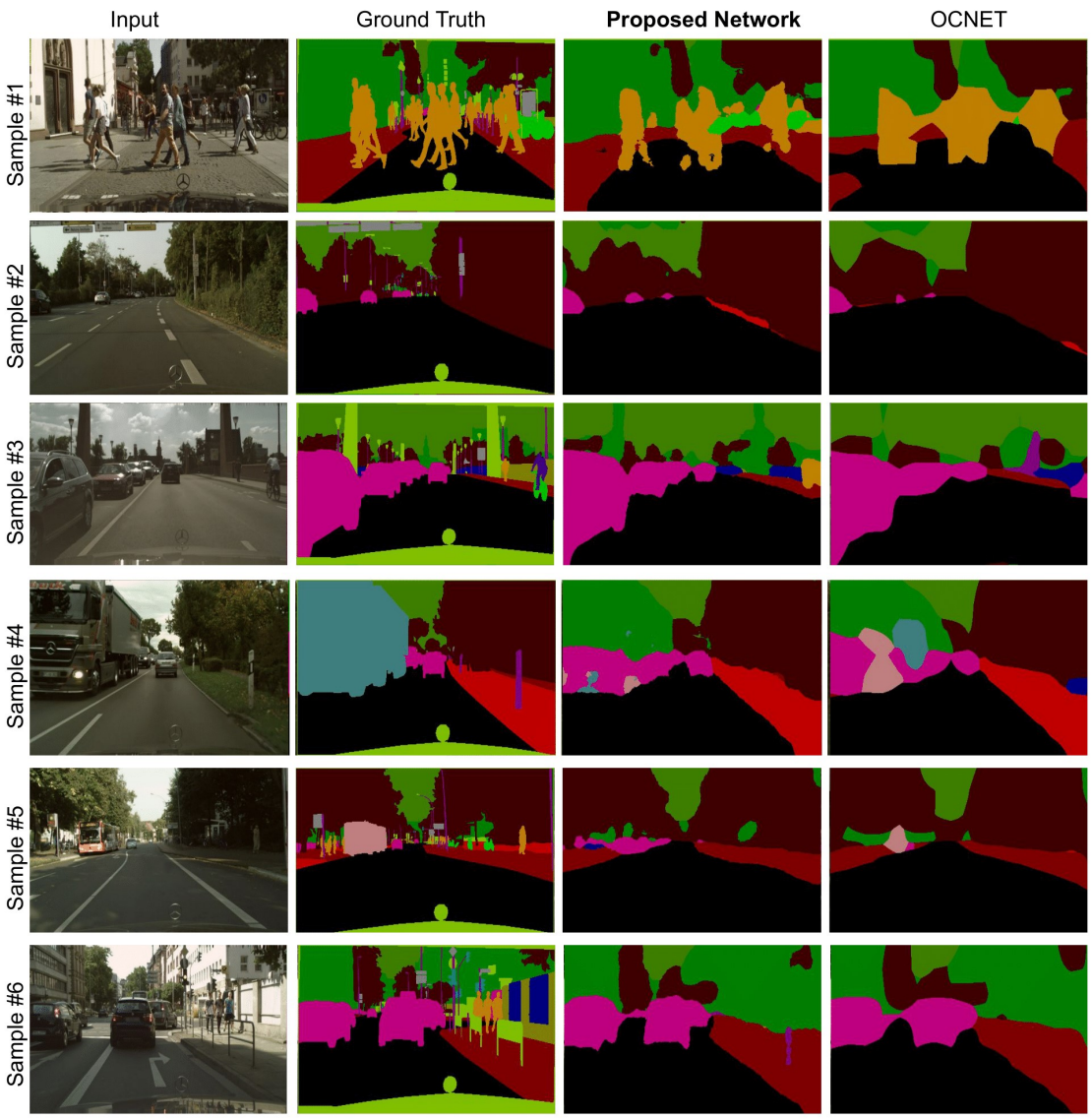

Figure 4.7: Qualitative examples: We illustrate a few examples of semantic segmentation performance on cityscapes dataset for the proposed network and OCNET. From left, the first two columns contain the Cityscapes dataset's Input and the Ground Truth, the last two columns contains the results from our proposed network and the OCNET. We can observe that the proposed network gives better and accurate segmentation than the OCNET.

qualitative results are shown in Figure 4.7. We can observe that the model was able to segment all the objects in the input image. Since the input resolution for the model

Table 4.1: Quantitative results comparing the performance of our network and OCNET on the Cityscapes test dataset.

Method	mIoU (%)
VoroSeg (Ours)	33.69
OCNET	30.99

Table 4.2: Super Resolution: This table presents the rendered segmentation predictions at different resolutions with their corresponding mIoUs.

Output Resolution	mIoU (%)
32×64	32.97
64×128	33.08
128×256	33.11
256×512	33.69

is (32×64), which is very small, the model is not able to learn fine details well. But, it does a better job of identifying large objects.

Rendering at Higher Resolution

As we mentioned in section 4.1 our model supports super resolution, we visually demonstrate the same in Figure 4.8. We use the model trained in the above mentioned training settings in section 4.3.6 and render the output at different resolutions like 32 $\times$ 64, 64 $\times$ 128, 128 $\times$ 256, and 256 $\times$ 512 by using the input in Sample #3 of Figure 4.7 . Quantitative results in Table 4.2 show that our model maintains the same mIoU at different resolutions. In Figure 4.8, the bottom image is the segmentation map rendered at 256 $\times$ 512 resolution. For a selected region with bounding box (yellow color), we present the rendered images at different resolutions, which are displayed in the top row. We can clearly see that there is a visual improvement in the output with increase in resolution (left to right) in the image.

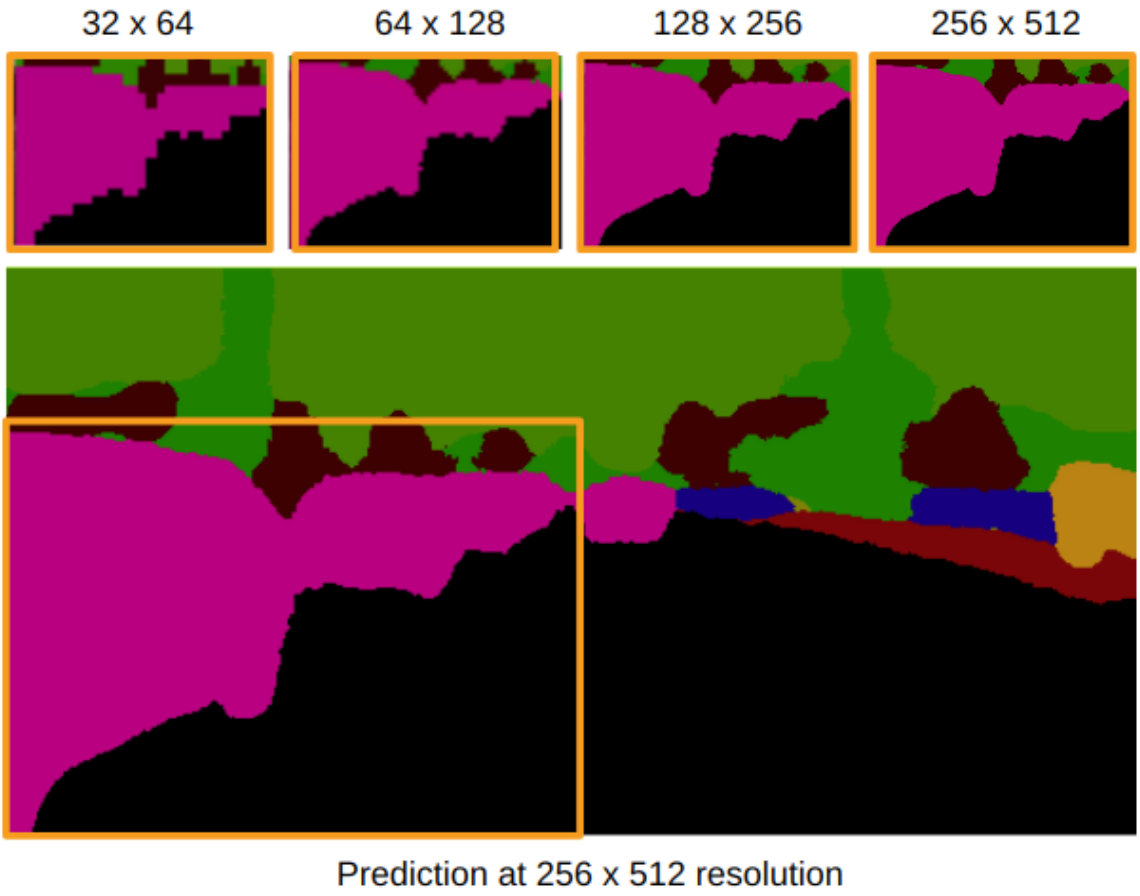

Figure 4.8: Rendering at Higher Resolutions: We pass the input image at sample #3 of Figure 4.7 to the proposed network and rasterize the prediction at different resolutions. The bottom image is the prediction rendered at 256×512 resolution. In the first row, we display the rendered images at various resolutions (increasing from left to right) for a selected region with the bounding box.

Chapter 5

Conclusions

In this thesis, we have focused on Semantic Segmentation, an essential task of Computer Vision, and explored creating datasets and deep neural architectures for semantic segmentation.

Regarding dataset creation, we have proposed an automatic annotation method for easily creating hand segmentation datasets with an RGBD camera, and have introduced a new high-quality dataset for hand segmentation that is significantly larger than what is currently available. Our annotation method requires minimal human interaction and is highly cost-effective. With the proposed method, we have created a dataset that contains high-accuracy dense pixel annotations, significant pose variations, and many different subjects. Our results show that the new dataset, HandSeg, allows training of segmenters that are more general than the ones trained with existing datasets. Precisely, our high-quality depth-based hand segmentation dataset consists of 210,000 frames with dual hand annotations from 13 subjects. We used the Intel RealSense SR30 sensor at a resolution of 640×480 the data. The dataset is made publicly available and is available at the website link https://gfx.uvic.ca/pubs/2019/bojja2019handseg/page.md.

We have proposed a segmentation network that is faster than existing baselines and provides superior mIoU accuracy. While these results are encouraging, our dataset opens new frontiers for investigation, such as the effectiveness of spatially-aware losses [36], the use of efficient quantized networks [30], or its use for weak-supervision of discriminative hand tracking [52].

Regarding the new method for real-time semantic segmentation on low power devices, we proposed a novel representation and architecture for deep networks for

semantic segmentation using Voronoi Diagrams. We used a differentiable definition of the Voronoi Diagram based on the softmax operator, enabling its use as a decoder layer in an end-to-end trainable network. We evaluated our proposed network architecture on the Cityscapes dataset and compared the results with a state of the art segmentation network. We also showed that this architecture is on par with the state of the art performance in semantic segmentation on low-resolution input images. We showed how our method could rasterize the segmentation maps at different resolutions, thus supporting super-resolution and giving the ability for the resource constraint devices to produce high-resolution feature maps with few resources.

Further, we have shown how our proposed network performs on other tasks like Image generation and hand segmentation.

Bibliography

[1] Pierre Alliez, Éric Colin De Verdière, Olivier Devillers, and Martin Isenburg. Centroidal voronoi diagrams for isotropic surface remeshing. *Graphical models*, 67(3):204–231, 2005.

[2] Vijay Badrinarayanan, Ankur Handa, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for robust semantic pixel-wise labelling. *arXiv preprint arXiv:1505.07293*, 2015.

[3] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495, 2017.

[4] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(12):2481–2495, 2017.

[5] Sven Bambach, Stefan Lee, David J. Crandall, and Chen Yu. Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions. In *International Conference on Computer Vision*, 2015.

[6] Abhishake Kumar Bojja, Franziska Mueller, Sri Raghu Malireddi, Markus Oberweger, Vincent Lepetit, Christian Theobalt, Kwang Moo Yi, and Andrea Tagliasacchi. Handseg: An automatically labeled dataset for hand segmentation from depth images. *arXiv*, 2018.

[7] Abhishake Kumar Bojja, Franziska Mueller, Sri Raghu Malireddi, Markus Oberweger, Vincent Lepetit, Christian Theobalt, Kwang Moo Yi, and Andrea Tagliasacchi. Handseg: An automatically labeled dataset for hand segmentation from depth images. In *2019 16th Conference on Computer and Robot Vision (CRV)*, pages 151–158. IEEE, 2019.

[8] Yuri Boykov, Olga Veksler, and Ramin Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11):1222–1239, 2001.

[9] P. Buehler, M. Everingham, D. P. Huttenlocher, and A. Zisserman. Long term arm and hand tracking for continuous sign language tv broadcasts. In *British Machine Vision Conference*, 2008.

[10] Jared Burns. Centroidal voronoi tessellations, 2009.

[11] Aaron Chadha and Yiannis Andreopoulos. Voronoi-based compact image descriptors: Efficient region-of-interest retrieval with vlad and deep-learning-based descriptors. *IEEE Transactions on Multimedia*, 19(7):1596–1608, 2017.

[12] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. In *Proceedings of International Conference on Learning Representations*, 2015.

[13] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

[14] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[15] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[16] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k-means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015.

[17] Ali Erol, George Bebis, Mircea Nicolescu, Richard D Boyle, and Xander Twombly. Vision-based hand pose estimation: A review. In *Computer Vision and Image Understanding*, 2007.

[18] Andreas Ess, Tobias Mueller, Helmut Grabner, and Luc J Van Gool. Segmentation-based urban traffic scene understanding. In *BMVC*, volume 1, page 2. Citeseer, 2009.

[19] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results, 2012.

[20] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9(Aug):1871–1874, 2008.

[21] Sean Ryan Fanello, Julien Valentin, Christoph Rhemann, Adarsh Kowdle, Vladimir Tankovich, and Shahram Izadi. Ultrastereo: Efficient learning-based matching for active stereo systems. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[22] Qing Fang. Topology maintained structure encoding. *arXiv preprint arXiv:1906.10823*, 2019.

[23] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation. *arXiv preprint arXiv:1704.06857*, 2017.

[24] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[25] Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D. Twigg, and Kenrick Kin. Online optical marker-based hand tracking with deep labels. *ACM TOG*, 2018.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *International Conference on Computer Vision*, 2017.

[28] Geoffrey E Hinton and Richard S Zemel. Autoencoders, minimum description length and helmholtz free energy. In *Advances in neural information processing systems*, pages 3–10, 1994.

[29] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A. Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation, 2017.

[30] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[31] Byeongkeun Kang, Yeejin Lee, and Truong Q Nguyen. Efficient hand articulations tracking using adaptive hand model and depth map. In *International Symposium on Visual Computing*, pages 586–598. Springer, 2015.

[32] Byeongkeun Kang, Subarna Tripathi, and Truong Q Nguyen. Real-time sign language fingerspelling recognition using convolutional neural networks from depth map. In *Pattern Recognition (ACPR), 2015 3rd IAPR Asian Conference on*, pages 136–140. IEEE, 2015.

[33] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes: Active contour models. *International journal of computer vision*, 1(4):321–331, 1988.

[34] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. *arXiv*, 2017.

[35] Pushmeet Kohli, Jonathan Rihan, Matthieu Bray, and Philip H. S. Torr. Simultaneous segmentation and pose estimation of humans using dynamic graph cuts. *International Journal of Computer Vision*, 2008.

[36] Nicholas Kolkin, Gregory Shakhnarovich, and Eli Shechtman. Training deep networks to be spatially sensitive. In *International Conference on Computer Vision*, 2017.

[37] Peter Kontschieder, Samuel Rota Bul, Horst Bischof, and Marcello Pelillo. Structured class-labels in random forests for semantic image labelling. In *International Conference on Computer Vision*, 2011.

[38] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[39] Yann LeCun. Modeles connexionnistes de lapprentissage (connectionist learning models). *0*, 1987.

[40] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL `http://yann.lecun.com/exdb/mnist/`.

[41] Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[42] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, 2014.

[43] Wei Liu, Andrew Rabinovich, and Alexander C Berg. Parsenet: Looking wider to see better. *arXiv preprint arXiv:1506.04579*, 2015.

[44] Yebin Liu, Juergen Gall, Carsten Stoll, Qionghai Dai, Hans-Peter Seidel, and Christian Theobalt. Markerless motion capture of multiple characters using multiview image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2013.

[45] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[46] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[47] Sri Raghu Malireddi. *Systematic generation of datasets and benchmarks for modern computer vision*. PhD thesis, 2019.

[48] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of Graphics Interface*, 2013.

[49] Shervin Minaee and Yao Wang. An admm approach to masked signal decomposition using subspace representation. *IEEE Transactions on Image Processing*, 28 (7):3192–3204, 2019.

[50] A. Mittal, A. Zisserman, and P. H. S. Torr. Hand detection using multiple proposals. In *British Machine Vision Conference*, 2011.

[51] Laurent Najman and Michel Schmitt. Watershed of a continuous function. 1994.

[52] Natalia Neverova, Christian Wolf, Florian Nebout, and Graham W. Taylor. Hand pose estimation through semi-supervised and weakly-supervised learning. *Computer Vision and Image Understanding*, 2017.

[53] Vincent Nivoliers and Bruno Lévy. Approximating functions on a mesh with restricted voronoi diagrams. In *Proceedings of the Eleventh Eurographics/ACMSIGGRAPH Symposium on Geometry Processing*, pages 83–92. Eurographics Association, 2013.

[54] Richard Nock and Frank Nielsen. Statistical region merging. *IEEE Transactions on pattern analysis and machine intelligence*, 26(11):1452–1458, 2004.

[55] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *International Conference on Computer Vision*, 2015.

[56] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[57] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. Hands deep in deep learning for hand pose estimation. In *Proceedings of Computer Vision Winter Workshop*, 2015.

[58] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. Efficient model-based 3d tracking of hand articulations using kinect. In *BmVC*, volume 1, page 3, 2011.

[59] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[60] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. In *arXiv*, 2016.

[61] Pedro O. Pinheiro, Tsung-Yi Lin, Ronan Collobert, and Piotr Dollr. Learning to refine object segments. In *European Conference on Computer Vision*, 2016.

[62] Nils Plath, Marc Toussaint, and Shinichi Nakajima. Multi-class image segmentation using conditional random fields and global classification. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 817–824, 2009.

[63] Vladislav Polianskii and Florian T Pokorny. Voronoi boundary classification: A high-dimensional geometric approach via weighted monte carlo integration. In *International Conference on Machine Learning*, pages 5162–5170, 2019.

[64] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. Realtime and robust hand tracking from depth. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1106–1113, 2014.

[65] Javier Romero, Hedvig Kjellström, and Danica Kragic. Hands in action: real-time 3d reconstruction of hands in interaction with objects. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 458–463. IEEE, 2010.

[66] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 234–241. Springer, 2015.

[67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-Assisted Intervention MICCAI 2015*, page 234241, 2015. ISSN 1611-3349. doi: 10.1007/978-3-319-24574-4_28. URL `http://dx.doi.org/10.1007/978-3-319-24574-4_28`.

[68] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on*

Medical image computing and computer-assisted intervention, pages 234–241. Springer, 2015.

[69] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *ACM Transactions Of Graphics*, 2004.

[70] DE Rumelhart, GE Hinton, and RJ Williams. Learning internal representations by error propagation. parallel distributed processing. vol 1: Foundations, 1986.

[71] Swami Sankaranarayanan, Yogesh Balaji, Arpit Jain, Ser Nam Lim, and Rama Chellappa. Learning from synthetic data: Addressing domain shift for semantic segmentation, 2017.

[72] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, 2015.

[73] Jamie Shotton, Matthew Johnson, and Roberto Cipolla. Semantic texton forests for image categorization and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[74] Jamie Shotton, Toby Sharp, Alex Kipman, Andrew Fitzgibbon, Mark Finocchio, Andrew Blake, Mat Cook, and Richard Moore. Real-time human pose recognition in parts from single depth images. *Comm. ACM*, 2013.

[75] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. Interactive markerless articulated hand motion tracking using RGB and depth data. In *International Conference on Computer Vision*, 2013.

[76] Srinath Sridhar, Franziska Mueller, Antti Oulasvirta, and Christian Theobalt. Fast and robust hand tracking using detection-guided optimization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[77] J-L Starck, Michael Elad, and David L Donoho. Image decomposition via the combination of sparse representations and a variational approach. *IEEE transactions on image processing*, 14(10):1570–1582, 2005.

[78] James S Supancic, Gregory Rogez, Yi Yang, Jamie Shotton, and Deva Ramanan. Depth-based hand pose estimation: data, methods, and challenges. In *International Conference on Computer Vision*, pages 1868–1876. IEEE, 2015.

[79] Andrea Tagliasacchi, Matthias Schroeder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. Robust articulated-icp for real-time hand tracking. *Computer Graphics Forum (CGF) (Proc. SGP)*, 2015.

[80] Jonathan Taylor, Vladimir Tankovich, Danhang Tang, Cem Keskin, David Kim, Philip Davidson, Adarsh Kowdle, and Shahram Izadi. Articulated distance fields for ultra-fast tracking of hands interacting. *ACM Transactions on Graphics (TOG)*, 36(6):244, 2017.

[81] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions Of Graphics*, 2014.

[82] Dimitrios Tzionas and Juergen Gall. 3d object reconstruction from hand-object interactions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 729–737, 2015.

[83] Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 157–166. ACM, 2014.

[84] Robert Y Wang and Jovan Popović. Real-time hand-tracking with a color glove. In *ACM transactions on graphics (TOG)*, volume 28, page 63. ACM, 2009.

[85] Yangang Wang, Jianyuan Min, Jianjie Zhang, Yebin Liu, Feng Xu, Qionghai Dai, and Jinxiang Chai. Video-based hand manipulation capture through composite motion control. *ACM Transactions on Graphics (TOG)*, 32(4):43, 2013.

[86] Aaron Wetzler, Ron Slossberg, and Ron Kimmel. Rule of thumb: Deep derotation for improved fingertip detection. In *British Machine Vision Conference*, 2015.

[87] Turner Whitted. An improved illumination model for shaded display. In *ACM Siggraph 2005 Courses*, pages 4–es. 2005.

[88] Francis Williams, Daniele Panozzo, Kwang Moo Yi, and Andrea Tagliasacchi. Voronoinet: General functional approximators with local support. *arXiv preprint arXiv:1912.03629*, 2019.

[89] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015.

[90] Shanxin Yuan, Qi Ye, Bjorn Stenger, Siddhand Jain, and Tae-Kyun Kim. Bighand2.2m benchmark: Hand pose dataset and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[91] Yuhui Yuan and Jingdong Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018.

[92] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. *arXiv preprint arXiv:1805.08318*, 2018.

[93] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[94] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[95] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[96] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *International Conference on Computer Vision*, 2017.

[97] Yang Zou, Zhiding Yu, B. V. K. Vijaya Kumar, and Jinsong Wang. Unsupervised domain adaptation for semantic segmentation via class-balanced self-training. *Lecture Notes in Computer Science*, page 297313, 2018. ISSN 1611-3349. doi: 10.1007/978-3-030-01219-9_18. URL `http://dx.doi.org/10.1007/978-3-030-01219-9_18`.