MANIFOLD, DEEP AND ADVERSARIAL LEARNING FOR VISUAL OBJECT DETECTION

by

Qiaosong Wang

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

2019

MANIFOLD, DEEP AND ADVERSARIAL LEARNING FOR VISUAL OBJECT DETECTION

by

Qiaosong Wang

Approved: _____
Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: _____
Levi T. Thompson, Ph.D.
Dean of the College of Engineering

Approved: _____
Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Christopher E. Rasmussen, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chandra Kambhamettu, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Li Liao, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Paul Huang, Ph.D.
Member of dissertation committee

ACKNOWLEDGEMENTS

I would like to give tremendous and heartfelt thanks to my advisor, Professor Christopher Rasmussen, for giving me the opportunity to work on exciting research projects during the past few years. Throughout my studies, he has always been supportive and spent an enormous amount of time guiding me through various research problems.

I am fortunate enough to work with my co-advisor, Professor Jingyi Yu, who provides enormous support and insightful suggestions on a few research topics. Also, thank Professor Yu for helping me with a few funding sources that made my Ph.D. work possible.

Thanks to Professor Chandra Kambhamettu, Professor Li Liao, and Professor Paul Huang for serving on my dissertation committee and providing invaluable feedback and comments.

I owe my deepest thanks to my parents for their support. Also, thank all my friends for the wonderful time I spent with you at UD.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

In recent years, emerging technologies such as deep learning have become critical in enabling robots to interact with complex real-world environments. This dissertation is focused on developing new algorithms to improve object detection for mobile robots. We start by exploring unsupervised algorithms to detect class-agnostic object saliency using a novel graph model. Next, we enable detection of class-specific objects and extend it to tracking. Furthermore, we demonstrate a joint learning scheme for simultaneous detection and fine-grained classification. Finally, we present a framework to perform 3D visual object detection on monocular images.

Firstly, we propose an unsupervised class-agnostic object detection approach by exploiting novel graph structure and background priors. The input image is represented as an undirected graph with superpixels as nodes. Feature vectors are extracted from each node to cover regional color, contrast and texture information. A novel graph model is proposed to effectively capture local and global saliency cues. To obtain more accurate results, we optimize the saliency map by using a robust background measure. Comprehensive evaluations on benchmark datasets indicate that our algorithm universally surpasses state-of-the-art unsupervised solutions and performs favorably against supervised approaches.

Secondly, we show a deep visual object detector without using object proposals. We modify a generic object detection network to train deep neural networks (DNN) models for bird and nest categories and extend it to enforce temporal continuity for tracking. The system shows satisfactory speed and accuracy for both detection and tracking. We also contribute a new dataset for nest detection. The proposed detector is well-suited for environmental robotic applications which demands real-time performance.

Next, we demonstrate a unified framework to detect and classify fine-grained objects. To evaluate performance, we have created a new benchmark for fine-grained recognition.

Experiments show that our approach performs favorably against competitive methods. Our network structure provides more desirable characteristics for practical computer vision applications and reaches a good balance between the model size, computational complexity, and accuracy.

Moreover, we extend our work to RGB-D datasets and begin by introducing an algorithm to produce 3D shapes of a scene on a mobile device. The algorithm leverages stereo cameras to generate a full resolution depth map of the scene, recording 3D geometry information. Quantitative analysis showed that this new 3D imaging algorithm consistently outperformed the existing methods. We also show a novel scheme for rendering dynamic Depth of Field (DoF) effects based on the generated depth map.

Lastly, we develop a framework to detect and classify 3D objects from monocular images. Experiments show that our approach performs favorably against competitive methods trained on LiDAR data. Our method leverages generative adversarial networks (GANs) to perform monocular depth estimation. The GAN approach is more flexible in terms of extending to other computer vision tasks. Also, we integrate both visual and structural cues into the feature map representation, which distinguishes our method from those purely operating on LiDAR data, and those who learn depth from a monocular image but still perform detection on the pseudo LiDAR data (ignoring visual information). Our system can be used to add visual intelligence to smart vehicles, which is particularly useful for improving camera-based advanced driver-assistance systems (ADAS) for L3 level autonomy. Also, our system could be used as a supplementary or fall-back option to LiDAR sensors.

Chapter 1

INTRODUCTION

Visual object detection is a complex task which deals with localizing, counting, and recognizing semantic objects of certain classes. Object detection has applications in many areas, including video analysis, face recognition, image retrieval, autonomous driving, robotic grasping, and so on. The human ability to rapidly detect natural objects in a scene has long been studied in neuroscience and cognitive psychology, but this task is particularly challenging for computers.

The literature of visual object detection is vast. One of the earliest visual attention models is proposed by Itti *et al*. in 1998 to detect attentive regions in scenes [58]. Inspired by this model, a series of work [100, 17, 65] are published to investigate salient locations in natural images, which is later termed *Fixation Prediction*. The second milestone came at 2007, when Liu *et al*. defined the problem of *Salient Object Detection*, which aims at detecting objects by rectangles in an image [88]. This sparks interest in research to extend the problem to *Saliency Map Prediction* [135, 44, 109, 141, 151, 78] , which uses a probability map instead of a binary rectangular map to mark salient objects. Since then, various low-level features have been shown to be effective for saliency object detection, such as color contrast, edge density [109], backgroundness [141, 151], objectness [21, 63], focus [63], etc. Also, multiple benchmark datasets [150, 32, 24, 88, 15, 64] are proposed and a series of supervised approaches [65, 88, 63, 79] show great success by learning visual knowledge from ground truth annotations. Efforts have also made for *Object Proposal Prediction* [11, 19] and *Object Subtizing* [110, 135, 160]. Recently, it has been shown that deep convolutional neural network (CNN) architectures such as [71, 124] are able to generate highly efficient abstract image representations for computer vision tasks. When combined with object proposals

[113, 38], deep classification networks could be transformed into detection frameworks locating and classifying object instances at the same time. More recently, detection systems have adopted region proposal networks to learn ground-truth locations [89, 108, 105, 106], enabling a large speed-up. Modern detection algorithms exhibit high detection accuracy, low-computational cost and real-time performance.

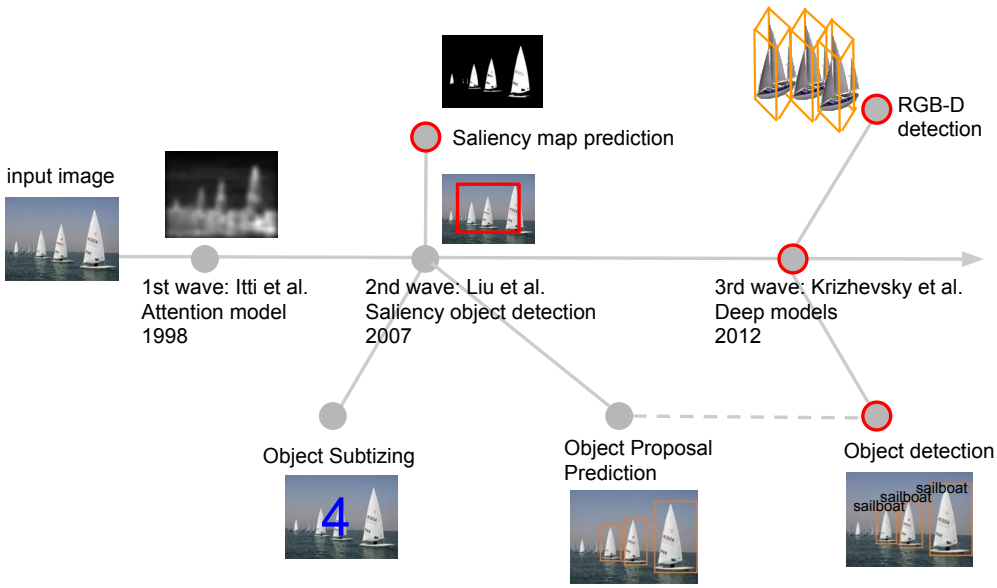


Figure 1.1: A brief chronicle of visual object detection. The first wave begins with Itti's [58] visual attention model. The second wave is started when Liu *et al* [88] defined the problem of Salient Object Detection. The third wave is represented by the series of publications based on the deep convolutional neural network (CNN) architectures [71, 124]. The topics discussed in this dissertation are marked in red.

In this dissertation, we focus on developing algorithms to improve object detection accuracy. We exploit 3 different approaches, from manifold learning to deep and adversarial learning. Firstly, we introduce an unsupervised, bottom-up method to detect class-agnostic object regions in an image. Next, we move to supervised learning and show a method to detect class-specific objects with extension to tracking on video frames using convolutional neural networks. We make further developments to this network to enable joint learning of object detection and fine-grained classification. Finally, we move to RGB + depth data for richer representation. We begin by showing a method to acquire and refine depth maps on

a mobile device. Then, we introduce a framework for detecting 3D objects on monocular images.

Depending on the perspective, our approach to visual object detection can be viewed as incremental development from unsupervised (Chapter 2) to supervised (Chapter 3, 4, 6) and from 2D (Chapter 2, 3, 4) to 3D (Chapter 5, 6). An overview of the topics covered in this dissertation can be found at Fig. 1.2. We make reference to previously published work here: [139, 136, 137, 138].

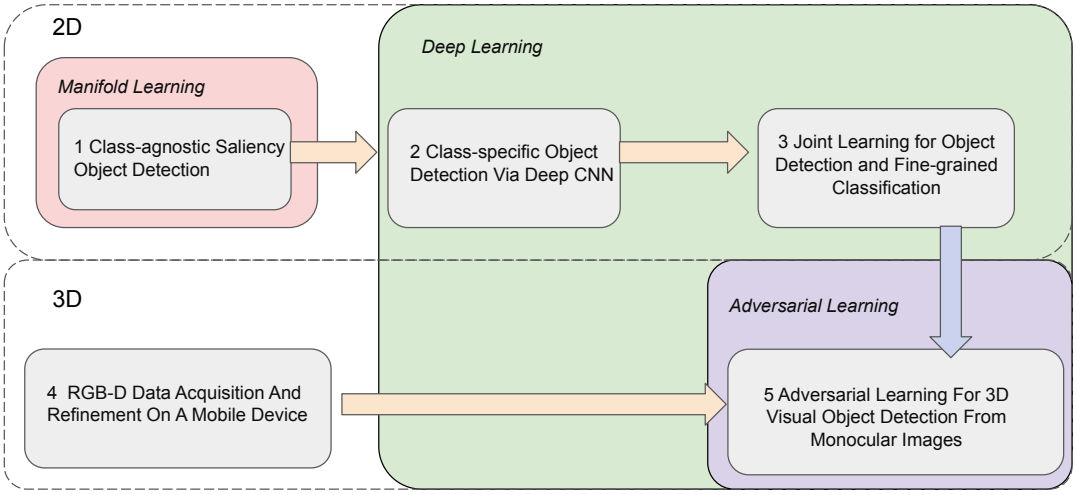


Figure 1.2: Overview of topics covered in this dissertation. Our work shows incremental development from unsupervised to supervised approaches, and from two-dimensional to three-dimensional space. Each block in this diagram shows the research work of a chapter in this dissertation.

1.1 Statement

This dissertation focuses on exploring graph-based manifold ranking, deep convolutional neural networks and generative adversarial networks to improve mean Average Precision (mAP) scores on 2D and 3D object detection benchmark datasets.

1.2 Contributions

A Novel Graph Model for Manifold-Ranking We present a novel graph model for salient object detection. Our graph model incorporates local and global contrast and naturally enforces the background connectivity constraint. The proposed feature distance metrics effectively and efficiently combine local color and texture cues to represent the intrinsic manifold structure. The proposed model achieves state-of-the-art results.

Detection and Tracking of Class-Specific Objects We have presented a deep CNN system specialized for bird and nest detection and tracking that exhibits excellent accuracy and speed. The proposed system modifies the loss term on the existing YOLO framework and introduces a way to integrate detection with a real-time tracker to improve tracking performance.

Joint Learning for Object Detection and Fine-grained Classification We propose an approach for joint object detection and fine-grained classification. Despite recent achievements in both fine-grained classification and object detection, few works have demonstrated datasets or solutions to simultaneously handle both tasks. We make two contributions to this problem. Firstly, we construct a fine-grained classification and detection benchmark. Secondly, we show an end-to-end convolutional neural network (CNN) architecture to detect and classify fine-grained objects. Experimental results verify that our network perform favorably against alternatives.

Acquisition and Refinement of Depth Maps on a Mobile Device We implemented a system to calculate and refine depth maps on an off-the-shelf tablet with dual cameras. The system can be easily ported to other mobile devices with limited computational power. We conducted extensive experiments to obtain the optimal combination of methods and parameters.

We experimented with Graph Cuts for depth map calculation, Joint Bilateral Upsampling for depth map refinement and the system is capable of working with a variety of scene structures and illumination conditions. Quantitative evaluations show that the proposed method can be used to improve existing stereo matching results. We also proposed a scheme for rendering dynamic Depth of Field (DoF) effects using the output depth map.

Adversarial Learning for 3D Visual Object Detection from Monocular Images We propose a novel approach to predict accurate 3D bounding box locations on monocular images. We first train a generative adversarial network (GAN) to perform monocular depth estimation. The ground truth training depth data is obtained via depth completion on LiDAR scans. Next, we combine both depth and appearance data into a birds-eye-view representation with height, density and grayscale intensity as the three feature channels. Finally, We train a convolutional neural network (CNN) on our feature map leveraging bounding boxes annotated on corresponding LiDAR scans. Experiments show that our method performs favorably against baselines.

1.3 Blueprint of the Dissertation

The rest of this dissertation is organized as follows.

Chapter 2 discusses an unsupervised approach for class-agonistic salient object detection. The proposed approach is based on a novel graph model and background priors.

In Chapter 3, we introduce a method for visual object detection via deep convolutional neural networks. We also extend this framework to enforce temporal continuity for tracking.

In Chapter 4, we demonstrate a joint framework to detect and classify fine-grained objects. We also created a new benchmark to evaluate this task which we term *fine-grained recognition*.

In Chapter 5, we move from 2D to the 3D domain. To begin with, we introduce an algorithm to capture accurate 3D data on a low-cost mobile device. We also show how to render DoF effects based on captured depth maps.

In Chapter 6, we develop a novel framework for detecting and classifying 3D objects from monocular images. We show depth map prediction via adversarial learning and propose a scheme for feature map generation. Our trained detector only requires a monocular image as input and has its unique advantage comparing to LiDAR-based methods.

Chapter 7 draws a conclusion to this dissertation and points out potential future directions.

Chapter 2

CLASS-AGNOSTIC SALIENT OBJECT DETECTION VIA A NOVEL GRAPH MODEL

In this chaper, we propose an unsupervised class-agnostic salient object detection approach by exploiting novel graph structure and background priors. The input image is represented as an undirected graph with superpixels as nodes. Feature vectors are extracted from each node to cover regional color, contrast and texture information. A novel graph model is proposed to effectively capture local and global saliency cues. To obtain more accurate saliency estimations, we optimize the saliency map by using a robust background measure. Comprehensive evaluations on benchmark datasets indicate that our algorithm universally surpasses state-of-the-art unsupervised solutions and performs favorably against supervised approaches.

2.1 Introduction

Humans are able to rapidly identify the visually distinctive objects in a scene. This fundamental capability has long been studied in neuroscience and cognitive psychology. In the computer vision community, researchers focus on similar tasks to determine regions that attract attention from a human perception system. The selected regions contain finer details of interest and can be used for extraction of intermediate and higher level information. Therefore, a fast and robust saliency detection algorithm can benefit various other vision tasks.

The literature of saliency map estimation is vast. However, most existing approaches can be categorized into unsupervised (typically bottom-up) [44, 109, 141, 151, 78] and supervised (typically bottom-up, but more recent approaches are a combination of top-down and bottom-up) [65, 88, 63, 79] approaches.

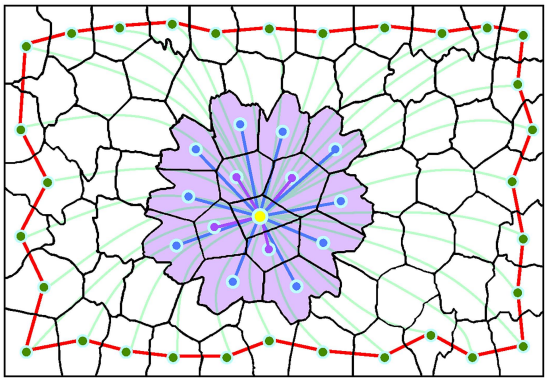


Figure 2.1: Our novel graph structure with superpixels as nodes. The purple and blue lines represent connections to first and second order neighbors, respectively. The green lines indicate that each node is connected to the boundary nodes on four sides of the image. The red lines show that the all boundary nodes are connected among themselves. See Sec. 2.3.1 for details.

While supervised approaches are able to automatically integrate multiple features and in general achieve better performance than unsupervised methods, it is still expensive to perform the training process, especially data collection. Also, compared to traditional special-purpose object detectors (e.g. pedestrian detection) where objects under the same class share some consistency, the salient objects from two images are often found vastly different in terms of visual appearance, especially when the object can be anything. Furthermore, the process of generating pixel-wise ground truth annotations itself is expensive and labor-intensive, and sometimes may even be impossible considering the scale of today's massive long-tailed visual repositories. This is typically the case in large e-commerce scenarios. A fast saliency technique can be an essential preprocessing step for background removal or object/product detection and recognition in large ecommerce applications.

In this chapter, we propose an unsupervised bottom-up saliency estimation approach. Our method is based on the remarkable success of the spectral graph theory. We focus on the core elements of spectral clustering algorithms. Specifically, we introduce a new graph model which captures local/global contrast and effectively utilizes the boundary prior. Inspired by ISOMAP manifold learning [127], we introduce geodesic distance to calculate the weight matrix. This constraint maximally enforces the background connectivity prior.

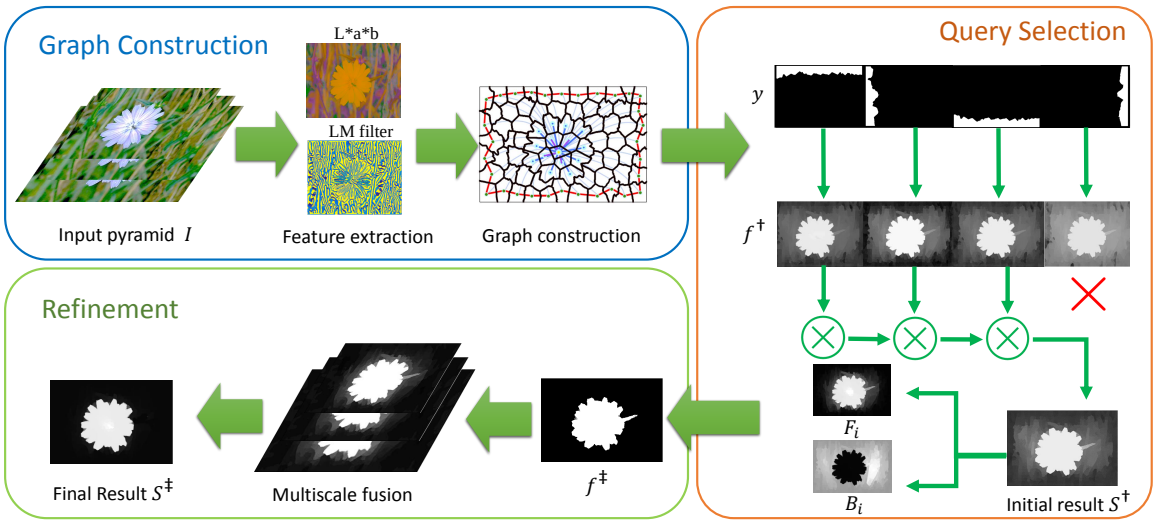


Figure 2.2: Pipeline of the proposed algorithm, divided into three parts: Graph Construction (Sec. 2.3), Query Selection (Sec. 2.4.1) and Refinement (Sec. 2.4.2). Nodes in the graph are superpixels. Weights are based on color and texture features. Groups of background seeds are selected for initial saliency based on their influence via the graph structure. Inconsistent groups of background seeds are eliminated. This estimated saliency map is refined by passing it again through the system. This process is repeated at multiple scales and results are fused.

Furthermore, we exploit boundary prior for selecting seeds to perform an initial background query. The resulting saliency map is further used to generate seeds to perform another query to obtain the final saliency map. As we will demonstrate empirically, the proposed method universally outperforms state-of-the-art unsupervised methods (e.g. GMR [151]) by a large margin, and in some cases even excels supervised methods (e.g. DRFI [62]). Our claim is that the proposed graph model provides more desirable characteristics for saliency detection and achieves unprecedented balance between computational complexity and accuracy.

2.2 Related Work

The core of our work is closely related to graph-based manifold ranking as in [151], geodesic distance as in [141], boundary prior sampling as in [78] and multi-scale fusion as in [150].

Supervised vs. Unsupervised Unsupervised methods [44, 109, 141, 151, 78] aim

at separating salient objects by extracting cues from the input image only. To date, various low-level features have been shown to be effective for saliency detection, such as color contrast, edge density [109], backgroundness [141, 151], objectness [21, 63], focus [63], etc. By eliminating the requirement of training, unsupervised methods can be easily integrated into various applications. In contrast, supervised approaches [65, 88, 63, 79] acquire visual knowledge from ground truth annotations. Recent advances in deep learning show promising results on benchmark datasets [79]. However, it is expensive to collect the hand-labeled images and set up the learning framework.

Graph-based Models Graph-based approaches have gained great popularity due to the simplicity and efficiency of graph algorithms. Harel et al. [44] proposed the graph based visual saliency (GBVS), a graph-based saliency model with multiple features to extract saliency information. Chang et al. [21] present a computational framework by constructing a graphical model to fuse objectness and regional saliency. Yang et al. [151] rank the similarity of superpixels with foreground or background seeds via graph-based manifold ranking. This method is further improved by Li et al. to generate pixel-wise saliency maps via regularized random walks ranking [78].

Center vs. Background Prior Recently, more and more bottom-up methods prefer to use the image boundary as the background seeds. This *boundary prior* is more general than previously used *center prior*, which assumes that the saliency object tend to appear near the image center [65, 88]. Wei et al. [141] define the saliency of a region to be the length of its shortest path to the virtual background node. In [171], a robust background measure is proposed to characterize the spatial layout of an image region with respect to the boundary regions.

2.3 Graph Construction

Our approach is based on building an undirected weighted graph for superpixels. We first segment the input image I into n superpixels $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ via the Simple Linear

Iterative Clustering (SLIC) [7] algorithm. For each superpixel s, we extract color and texture information to form a regional feature descriptor r. A metric is proposed to calculate the edge weight between two given descriptors. Next, we construct a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ (see Fig. 2.1) where $\mathcal{V}$ is a set of nodes corresponding to superpixels $\mathcal{S}$, and edges $\mathcal{E}$ are constructed using the proposed graph model. $\mathcal{E}$ is quantified by a weight matrix $W = [w_{ij}]_{n\times n}$ where the weights are calculated using distances between extracted feature descriptors. In Sec. 2.3.1, we describe our newly proposed graph model and in Sec. 2.3.2 we show how to extract regional features and calculate the weight matrix W.

2.3.1 Proposed Graph Model

Given a set of superpixels $\mathcal{S}$, we start by building a k-regular graph where each node is only connected to its immediate neighbors. We define the adjacency matrix of the initial graph $\mathcal{G}$ to be $A = [a_{ij}]_{n\times n}$. If $a_{ij} = 1$, then the nodes s_i and s_j are adjacent, otherwise $a_{ij} = 0$. As $\mathcal{G}$ is undirected we require $a_{ij} = a_{ji}$. $\mathcal{B} \in \mathcal{S}$ denotes a set of boundary nodes containing $|\mathcal{B}|$ superpixels on the four borders of the input image. For robust purposes, we only choose to use three borders, and the selection of borders is described in Sec. 2.4.1. We subsequently add edges to the initial graph $\mathcal{G}$ to build a new graph model with the following rules: 1) Each node is connected to both its immediate neighbors and 2-hop neighbors; 2) We add edges to connect each node to boundary nodes on the four sides of the image. The weight for each edge is divided by the number of boundary nodes; 3) Any pair of nodes on the image boundary is considered to be connected. We denote the above three rules by

R_1, R_2 and R_3, and the final edge set $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2, \mathcal{E}_3\}$ can be obtained as:

$$
\begin{aligned}
R_1 : \mathcal{E}_1 &= \{(s_i, s_j) | s_i, s_j \in \mathcal{S}, a_{ij} = 1\} \\
&\cup \{(s_i, s_k) | s_k \in \mathcal{S}, a_{kj} = 1\}, \\
w_{ij} &= weight(r_i, r_j). \\
R_2 : \mathcal{E}_2 &= \{(s_i, s_j) | s_i \in \mathcal{S}, s_j \in \mathcal{B}\}, \\
w_{ij} &= weight(r_i, r_j)/|\mathcal{B}|. \\
R_3 : \mathcal{E}_3 &= \{(s_i, s_j) | s_i, s_j \in \mathcal{B}\}, \\
w_{ij} &= weight(r_i, r_j).
\end{aligned}
\tag{2.1}
$$

The structure of our graph model is shown in Fig. 2.1. Since neighboring superpixels are more likely to be visually similar, R_1 enables us to effectively utilize local neighborhood relationships between the superpixel nodes. R_2 connects each node to all boundary nodes, enforcing the global contrast constraint. Since the number of boundary superpixels may be large, we average the edge weights, making the total contribution of boundary nodes equivalent to only one single superpixel. R_3 enforces the graph to be a closed-loop. Combined with R_2 which connects each superpixel to boundary nodes. R_3 further reduces the geodesic distance of two similar superpixels.

Figure 2.3: The effect of our graph model described in Sec. 2.3.1. From left to right: input image, result using the graph structure proposed by [151], result obtained using our graph model. Our model performs better since it encodes background consistency, global contrast and local contrast.

Figure 2.4: Examples where geodesic distance generate more accurate results. From left to right: input image, results without enforcing the geodesic distance constraints, results with geodesic constraints. Geodesic distance avoids missing parts due to color bleeding.

2.3.2 Feature Extraction

In this section, we detail the process of extracting feature descriptors from each superpixel. This process is crucial to the estimation of the final saliency map as the edge weights are calculated by comparing the feature descriptors of two nodes. A good feature descriptor should exhibit high contrast between salient and non-salient regions. In our work, we mainly adopt two kinds of features: color and texture. For color features, we consider mean color values and color histograms in the CIELAB [53] color space for each superpixel. For texture features, we use responses from the Leung-Malik (LM) filter bank [77]. Let $v^{lab}, h^{lab}, h^{tex}$ be the mean L*a*b* color, L*a*b* histogram and max LM response histogram of superpixel s, we define the distance between two superpixels as:

$$\begin{aligned} dist(r_i, r_j) =& \lambda_1 ||v_i^{lab} - v_j^{lab}|| + \lambda_2 \, \chi^2(h_i^{lab}, h_j^{lab}) \\ &+ \lambda_3 \, \chi^2(h_i^{tex}, h_j^{tex}). \end{aligned} \tag{2.2}$$

where $r = (v, h^{lab}, h^{tex})$ is the combined feature descriptor for superpixel s, λ_1, λ_2 and λ_3 are weighting parameters, $\chi^2(h_1, h_2) = \sum_{i=1}^{K} \frac{2(h_1(i) - h_2(i))^2}{h_1(i) + h_2(i)}$ is the chi-squared distance between histograms h_1 and h_2 with K being the number of bins. The edge weights can be

obtained by the Gaussian similarity function:

$$weight(r_i, r_j) = \begin{cases} exp(-dist(r_i, r_j)/\sigma^2) & \text{if } a_{ij} = 1, \\ \min_{\rho_1=r_i,\rho_2=r_{i+1},\ldots,\rho_m=r_j} & \\ \sum_{\rho=1}^{m-1} weight(\rho_k, \rho_{k+1}) & \text{if } a_{ij} = 0. \end{cases} \tag{2.3}$$

where σ is a constant. In the above equation, the second condition considers the shortest path between nodes i, j. As can be seen from Eq.(2.2), our approach is completely based on intrinsic cues of the input image. Without any prior knowledge of size of the salient object, we adopt the L-layer Gaussian pyramid for robustness. The lth-level pyramid I^l is obtained as:

$$I^l(x, y) = \sum_{s=-2}^{2} \sum_{t=-2}^{2} \omega(s, t) I^{l-1}(2x + s, 2y + t), l \geqslant 1. \tag{2.4}$$

where I^0 is the original image, $\omega(s, t)$ is a Gaussian weighting function (identical at all levels). The number of superpixels $n^l = |\mathcal{S}^l|$ for the l-th level pyramid I^l is set as:

$$n^l = \frac{n^{l-1}}{2^{2(l-1)}} \tag{2.5}$$

Next, we extract multiscale features r^l and build weight matrices W^l for each level. The final saliency estimation is conducted on each level independently and the output saliency map is combined using results from all levels (see Sec. 2.5.2 for details).

2.4 Background Priors

Given the weighted graph, we can take either foreground or background nodes as queries [151]. The resulting saliency map is calculated based on its relevance to the queries. Our algorithm is based on background priors, which consists of two parts: the *boundary prior* and the *connectivity prior*. The first prior is based on the observation that the salient object seldom touches the image borders. Compared to the *center prior* [65, 88] which

Algorithm 1 Visual Saliency via Novel Graph Model and Background Priors

procedure INPUT(**Input image I and related parameters**)

1. Apply SLIC [7] and separate input image I into n superpixels $\mathcal{S} = \{s_1, s_2, ..., s_n\}$, establish graph structure with Eq. (2.1).

2. Calculate W and D using Eq. (2.2) and Eq. (2.3).

3. Select three borders as query seeds as described in Sec. 2.4.1 and obtain query vector $y = [y_1, y_2, ..., y_n]^T$.

4. Acquire initial saliency estimation $S^\dagger$ using Eq. (2.7), Eq. (2.8) and Eq. (2.9).

5. Optimize $S^\dagger$ using Eq. (2.10) and re-apply Eq. (2.7) to obtain the foreground estimation. Apply Eq. (2.4) and average results across different levels to obtain final saliency map $S^\ddagger$.

Output: A saliency map $S^\ddagger$ with the same size as the input image

end procedure

assumes that the salient object always stays at the center of an image, the *boundary prior* is more robust, which is validated on several public datasets [141]. In our work, we choose three out of four borders as background seeds to perform queries [78]. This is because the foreground object may completely occupy one border of an image, which is commonly seen in portrait photos. Therefore, eliminating one border which tends to have a very distinct appearance generates more accurate results. The second prior is based on the insight that background regions are usually large and homogeneous. Therefore, the superpixels in the background can be easily connected to each other. This prior is also applicable for images with a shallow depth of field, where the background region is out of focus. The rest of this section is organized as follows: Sec. 2.4.1 elaborates the detailed steps of the initial background query and Sec. 2.4.2 illustrates a refinement scheme based on the *connectivity prior*.

Table 2.1: Ablation study on adding different components to the baseline GMR [151] algorithm (Sec. 2.5.2). All results correspond to ECSSD. GF = guided filter [48], RTV = texture smoothing using relative total variation [149], EBR = erroneous boundary removal [78], RPCA = robust PCA [18], LAB = CIELAB color [53], HIST = L*a*b* histogram, LM = Leung-Malik filter bank [77], LBP = local binary patterns [98], AVE = simple averaging, HS = hierarchical saliency [150], GMR = graph based manifold ranking [151], BC = boundary connection, GEO = geodesic distance. Methods included in the final pipeline are marked in bold.

Evaluation	Preprocessing		Sampling		Features				Scaling		Graph		
	GF	RTV	**EBR**	RPCA	**LAB**	**HIST**	**LM**	LBP	**AVE**	HS	GMR	**BC**	**GEO**
Precision	0.712	0.716	0.725	0.755	0.731	0.725	0.718	0.614	0.727	0.734	0.731	**0.771**	0.743
Recall	**0.729**	0.712	0.723	0.646	0.575	0.631	0.682	0.577	0.710	0.716	0.569	0.626	0.618
F-Measure	0.713	0.716	0.725	0.745	0.715	0.716	0.715	0.610	0.725	0.733	0.714	**0.756**	0.730
Runtime (s)	0.136	2.237	0.011	4.782	0.025	0.031	0.094	0.047	0.002	0.129	0.258	0.327	0.538

2.4.1 Query via the Boundary Prior

To provide more accurate saliency estimations, we first compare the four borders of the image and remove one with the most distinctive color distribution. We combine boundary superpixels together to form a single region, and use Eq. (2.2) to compute the distance of any two of the four regions $\{\mathcal{B}_{top}, \mathcal{B}_{bottom}, \mathcal{B}_{left}, \mathcal{B}_{right}\}$. The resulting 4×4 matrix is summed column-wise, and the maximum column corresponds to the boundary to be removed.

Once the query boundaries are obtained, we can label the corresponding superpixels to be background. More formally, we build a query vector $y = [y_1, y_2, ..., y_n]^T$, where $y_i = 1$ if s_i belongs one of the four query boundaries, otherwise $y_i = 0$. Given the weight matrix $W = [w_{ij}]_{n \times n}$ computed in Sec. 2.3.2, we can obtain the degree matrix $D = diag(d_1, d_2, ..., d_n)$, where $d_i = \sum_j w_{ij}$. Let f be the ranking function assigning rank values $f = [f_1, f_2, ..., f_n]^T$ which could be obtained by solving the following minimization problem:

$$f^{\dagger} = arg\min_{f} \frac{1}{2} \left(\sum_{ij=1}^{n} w_{ij} \left\| \frac{f_i}{\sqrt{d_i}} - \frac{f_j}{\sqrt{d_j}} \right\|^2 + \mu \sum_{i=1}^{n} \|f_i - y_i\|^2 \right). \tag{2.6}$$

where μ is a controlling parameter. The optimized solution is given in [168] as:

$$f^{\dagger} = (D - \frac{W}{\mu + 1})^{-1} y. \tag{2.7}$$

Three ranking results $f^{\dagger}(b)$ will be achieved after applying Eq. (2.7), where b corresponds one of the three borders. Since the ranking results show the background relevance of each node, we still need to calculate their complement values to obtain the foreground-based saliency:

$$S_i(b) = 1 - f_i^{\dagger}(b), i = 1, 2, ..., n. \tag{2.8}$$

The results are then put into element-wise multiplication to calculate the saliency map:

$$S^{\dagger} = \prod_{b} S_i(b). \tag{2.9}$$

2.4.2 Refinement

In this section, we seek to optimize the result from the previous section. The optimized result will be used as foreground query by applying Eq. (2.7) again. The cost function is designed to assign 1 to salient region value and 0 to background region. The optimized result is then obtained by minimizing the following cost function [171]:

$$f^{\ddagger} = arg \min_{f} \left(\sum_{i=1}^{n} F_i (f_i - 1)^2 + \sum_{i=1}^{n} B_i f_i^2 + \sum_{i,j} w_{ij} (f_i - f_j)^2 \right). \tag{2.10}$$

Where F_i and B_i are foreground and background probabilities, $F_i > mean(S_i)$ and $B_i < mean(S_i)$. The three terms are all squared errors and the optimal result is computed by least-square. The newly obtained f is a binary indicator vector and can be used as seed for foreground queries. By re-applying Eq. (2.7), we obtain the final saliency map $S^{\ddagger} = (D - \frac{W}{\mu + 1})^{-1} f^{\ddagger}$.

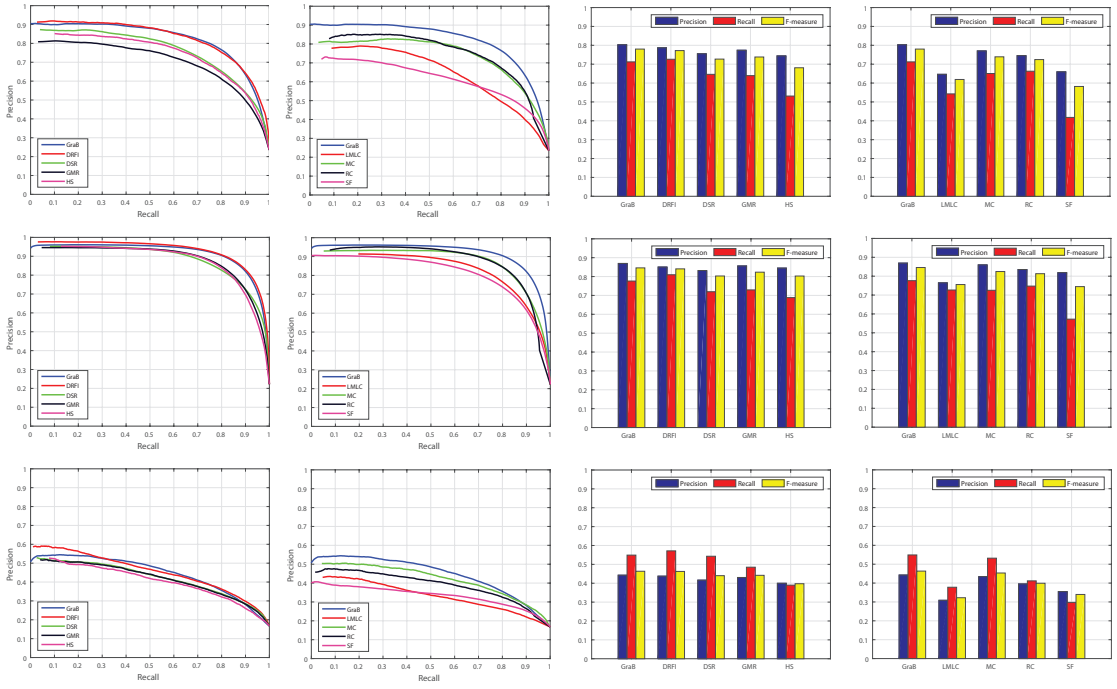


Figure 2.5: Quantitative PR-curve and F-measure evaluation of 9 approaches on 3 datasets. The rows from top to bottom correspond to ECSSD, THUS10K and JuddDB, respectively. Clearly, our approach excels all other unsupervised approaches and performs favorably against a powerful supervised approach (DRFI).

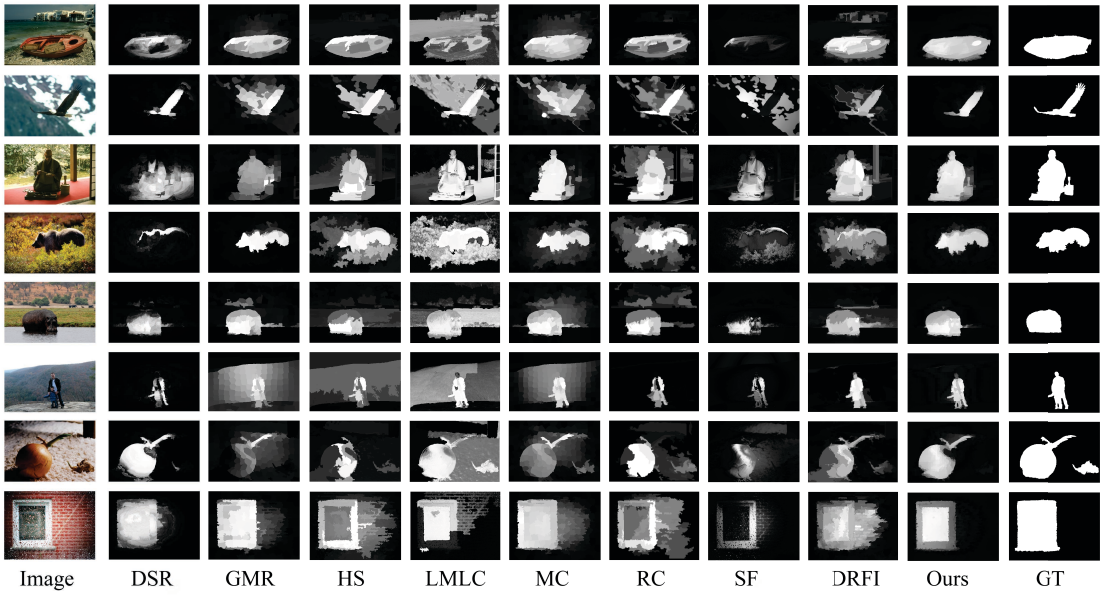


Figure 2.6: Qualitative evaluation. DRFI is one of the best supervised approaches. All other approaches shown here are unsupervised. Our model is closely related to GMR, but gives much better performance. See Sec. 2.5.3 for details.

2.5 Experiments

2.5.1 Parameter Setup

We set the number of superpixels n to be 200 in all experiments. λ_1, λ_2 and λ_3 in Eq. (2.2) are set to 0.25, 0.45 and 0.3, respectively. In our experiment, we use a 3 level pyramid, hence $l = 3$ in Eq. (2.4). The constant σ in Eq. (2.3) and μ in Eq. (2.6) are empirically chosen and $\sigma^2 = 0.1$, $1/(\mu + 1) = 0.99$. Our method is implemented using Matlab on a machine with Intel Core i5-2500K 3.3 GHz CPU and 16GB RAM. The mean execution time is around 800 ms to process a typical 400×300 image.

2.5.2 Ablation Studies

We start by modifying the GMR framework proposed by [151]. We experiment different design options among five categories: preprocessing, sampling, features, scaling and graph structure. The individual components are added to the original GMR framework and quantitative evaluations are conducted on the entire ECSSD dataset (Fig. 2.5 and Fig. 2.8).

Preprocessing The input images are often composed of objects at various scales with diverse texture details. Therefore, it is important to remove detrimental or unwanted content. We choose two edge-preserving filters for testing: guided image filtering [48] and imaging smoothing via relative total variation [149]. The first method performs edge-preserving smoothing while the second method extracts important structure from texture based on inherent variation and relative total variation measures. Quantitative evaluations suggest that both methods are able to improve the saliency detection results with similar performance.

Sampling Our method estimates saliency by using boundary superpixels as queries. If the foreground object touches one or more boundaries of the image, then the query results may be problematic. Therefore, it is important to smartly choose boundary superpixels as seeds. We tested two schemes for sampling boundary superpixels: erroneous boundary removal and robust principle component analysis. The details of the first method is illustrated in Sec. 2.4.1. The second method is based on the recently proposed rank minimization model [18]. We randomly sample 25% of all superpixels on each border, and repeat this step n times. This results in $4n$ set of query seeds. For each set we apply Eq. (2.7) to estimate saliency values for all superpixels. We unroll each resulting image into a vector and stack them into a matrix P. The low rank matrix A can be recovered from the corrupted data matrix $P = A + E$ by solving the following convex optimization problem:

$$\min_{A,E} ||A||_* + \lambda ||E||_1. \tag{2.11}$$

where $||\cdot||_*$ denotes the nuclear norm, $||\cdot||_1$ denotes the sum of absolute values of matrix entries, λ is a positive weighting parameter and E is a sparse error matrix. In our experiment, we set $n = 5$ and perform the query 20 times for each image to get the initial saliency map $S^{\dagger}$. Evaluation on the complete ECSSD dataset shows that RPCA achieves better precision than erroneous boundary removal.

Features As stated in Sec. 2.3.2, we associate each superpixel with a feature vector to calculate the weight matrix W. A good feature descriptor should exhibit high contrast between salient and non-salient regions. In our experiment, we mainly test four different

features: mean L*a*b* value [53], L*a*b* histogram, responses from the LM filter bank and local binary patterns (LBP) [98]. Among these features, the mean L*a*b* value is shown to be effective in [141, 151, 171]. According to Jiang et al. [62], the L*a*b* histogram is the most important regional feature in their feature integration framework. We are able to achieve satisfactory precision using the first two features. The LM filter response gives better overall recall. LBP feature seems to be not as effective as LM texture features in our case. Therefore, we linearly combine the first three features together to form the final feature vector.

Scaling In the saliency detection literature, hierarchical models are often adopted for robustness purpose [150, 62]. Our first experiment is to build an image pyramid, apply our algorithm to each layer and simply average all maps (Sec. 2.3.1). We subsequently test the approach proposed in [150]. This method differs from naive multi-layer fusion by selecting optimal weights for each region using hierarchical inference. Due to the proposed tree structure, the saliency inference can efficiently be conducted using belief propagation.

Graph Structure We use the model proposed by [151] as a baseline to test variations on the graph structure. The reference model enforces rule R_1 and R_3 in Sec. 2.1 and adopts Euclidean distance as the weighting metric. We conduct experiments on both graph structures (Sec. 2.3.1) and distance metrics (Sec. 2.3.2). Quantitative evaluations show a major performance improvement compared to other methodologies.

Combination We have presented 5 different strategies to facilitate more accurate saliency estimation. However, it is difficult to test all permutations and analyze the interactions between different methods. Therefore, how to optimally combine these methods still remains non-trivial. For example, the use of guided filter and multiscale averaging alone improves the recall scores. However, when combined together the performance drops slightly. Also, we choose not to use RPCA-based boundary sampling and belief-propagation based multi-layer fusion due to speed-accuracy tradeoffs. In our final model we choose not to perform any texture smoothing and employ the multiscale averaging scheme due to its simplicity and efficacy. The color histogram based erroneous boundary removal scheme is used for generating the initial queries. The methods we choose to include in the final pipeline are

marked in bold in Table 2.1. At the core of our algorithm is the newly proposed graph model and geodesic distance metric as they offer significant performance improvements.

2.5.3 Comparison with State-of-the-Art

Datasets In the experiments, we qualitatively and quantitatively compare the proposed approach with eight state-of-the-art approaches, including DRFI [62], DSR [82], GMR [151], HS[150], LMLC [148], MC [61], RC [23], SF [102]. It is important to note that besides DRFI, all other methods are unsupervised. The evaluation is conducted on three challenging datasets: ECSSD, THUS10K and JuddDB. The Extended Complex Scene Saliency Dataset (ECSSD) [150] contains 1000 semantically meaningful but structurally complex images from the BSD dataset [12], PASCAL VOC [32] and the Internet. The binary masks for the salient objects are produced by 5 subjects. THUS10K [24] contains 10000 images with pixel-level ground-truth labelings from the large dataset (20,000+ images) proposed by Liu et al. [88]. The JuddDB dataset [15] is created from the MIT saliency benchmark [64], mainly for checking generality of salient object detection models over real-world scenes with multiple objects and complex background. Additionally, we compared with all saliency object segmentation methods mentioned in [83] and [153] on the PASCAL-S dataset, including CPMC+GBVS [83], CPMC+PatchCut [153], GBVS+PatchCut [153], RC [23], SF [102], PCAS [94] and FT [8]. The PASCAL-S is proposed to avoid the dataset design bias, where the image selection process deliberately emphasizes the concept of saliency [83].

Evaluation We follow the canonical precision-recall curve and F-measure methodologies to evaluate the performance of our algorithm using the toolbox provided by [81]. The PR-curve and F-measure comparisons are shown in Fig. 2.5. Specifically, the PR curve is obtained by binarizing the saliency map using varying thresholds from 0 to 255, as mentioned in [8, 23, 102, 114]. F-measure is obtained using the metric proposed by [8]:

$$F_\beta = \frac{(1+\beta^2)Precision \times Recall}{\beta^2 Precision + Recall} \tag{2.12}$$

Here, the precision and recall rates binarized using an adaptive threshold determined as two

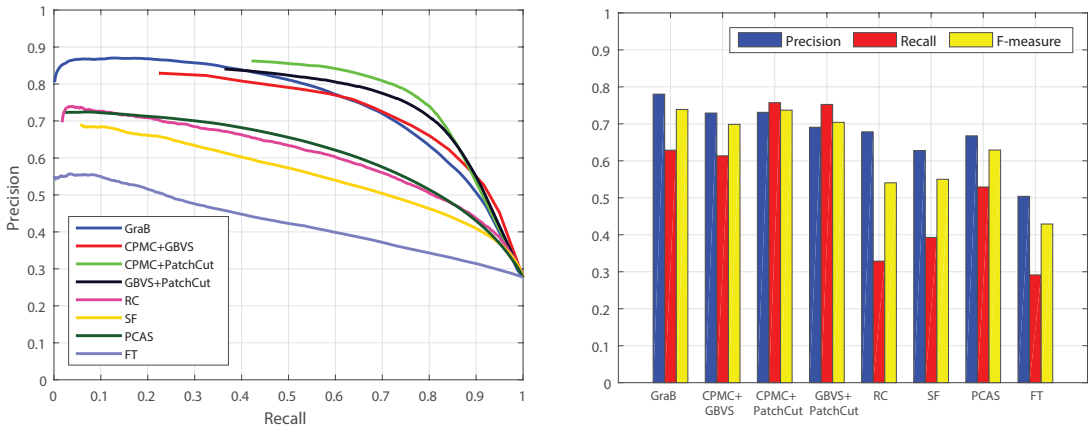


Figure 2.7: Quantitative PR-curve and F-measure evaluation of 7 methods on the PASCAL-S dataset. Note that our method achieves similar or better F-measure as more compute expensive methods.

times the mean saliency of a given image. We set β to 0.3 to emphasize the precision[8, 151, 78].

As can be seen in Fig. 2.5, our method significantly outperforms all seven unsupervised methods by a large margin. Specifically, our method achieved an improvement of 6% in comparison with the baseline GMR model on the challenging ECSSD dataset. Also, our method is highly competitive when compared to DRFI on all three datasets. It is worth noting that DRFI takes around 24 hours for training and 10 seconds for testing given a typical 400×300 image [62], whereas our method is fully unsupervised and only takes 800 milliseconds to process a similar image. Furthermore, DRFI takes 2500 images for training and extracts more than 20 different features, while our method is purely based on the input image and only uses 3 simple features. In other words, our method is much more efficient than DRFI yet still capable of maintaining competitive accuracy. The efficacy of our graph model is self-evident.

Quantitative evaluations on PASCAL-S [83] (Fig. 2.7) show that our method achieves higher precision, recall and F-measure scores compared to the state-of-the-art CPMC+GBVS algorithm presented in [83]. Also, our method performs favorably against the more recent PatchCut method [153] and clearly above all other saliency algorithms. Again, our method is

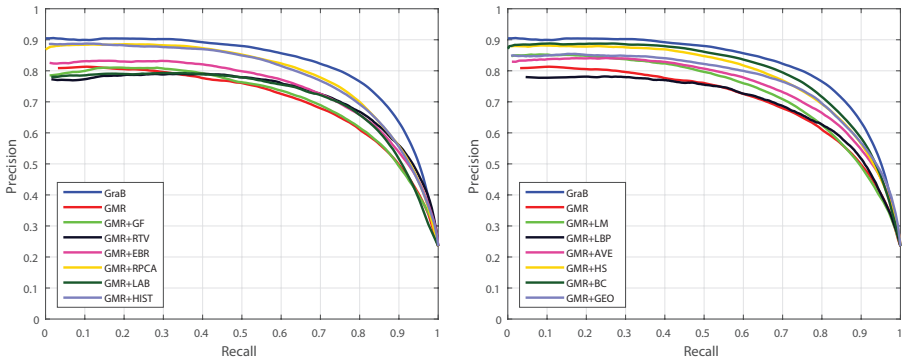


Figure 2.8: Quantitative PR-curve on different design options mentioned in Sec. 2.5.2. The baseline method (GMR) and final combined method (GraB) are added to both figures for comparison.

training-free and performs much faster than CPMC+GBVS and PatchCut. (CPMC+GBVS takes around 30s to process a 400×300 image, according to our experiment; PatchCut takes around 10s for segmenting a 200×200 image, as reported by [153]. Both methods require extra training/example data).

Our evaluation does not include some of the latest deep-learning methods. The crux of our method is to propose a novel heuristic model which is able to achieve similar performance to supervised methods like DRFI or CPMC+GBVS without preparing expensive training data. This provides simplicity and easy-to-use generality in many practical applications where computing power is limited and ground truth annotations are very expensive or impossible to acquire.

Fig. 2.6 shows a few saliency maps for qualitative evaluation. We note that the proposed algorithm uniformly highlights the salient regions and preserves fine object boundaries than other methods.

2.6 Conclusion

We present a novel unsupervised saliency estimation method based on a novel graph model and background priors. Our graph model incorporates local and global contrast and naturally enforces the background connectivity constraint. The proposed feature distance

metrics effectively and efficiently combines local color and texture cues to represent the intrinsic manifold structure. We further optimize the background seeds by exploiting a boundary query and refinement scheme, achieving state-of-the-art results. Our future work includes theoretical analysis on the proposed graph model and its potential towards building better clustering algorithms. Also, we would like to accelerate our algorithm via parallel computing, as large-scale spectral clustering has been trivially accomplished in high-performance graphics hardware [123].

Chapter 3

DETECTION AND TRACKING OF CLASS-SPECIFIC OBJECTS VIA DEEP CONVOLUTIONAL NEURAL NETWORKS

In this chapter we present a visual object detector based on a deep convolutional neural network that quickly outputs bounding box hypotheses. We modify the network for better performance, specialize it for a robotic application involving bird and nest categories (including the creation of a new dataset for the latter), and extend it to enforce temporal continuity for tracking. The system exhibits very competitive detection accuracy and speed, as well as robust, high-speed tracking on several difficult sequences

3.1 Introduction

Visual object detection is a complex task which entails recognizing, localizing, and counting objects within an image. The human ability to rapidly detect natural objects in a scene has long been studied in neuroscience and cognitive psychology [129], but this task is particularly challenging for computers. Until recently, the best-performing detectors for objects such as people and cars used combinations of handcrafted image features such as histograms of oriented gradients [26, 43].

Our motivation in this chapter is not general object detection, but rather to rapidly and accurately detect and track *birds* and *bird nests* in forest scenes for a environmental robotic application. "Bird" is a category in the well-known PASCAL VOC dataset [31], a widely-used benchmark in visual category classification, detection, and segmentation. However, there is very little previous work on visual bird tracking or bird nest detection: [112] applies morphological analysis to analyze overhead images of poultry, [120] using saliency methods on visible-wavelength and infrared images to find ground nests in agricultural fields, and [144] uses shape analysis to find nests as outliers on power poles adjacent to high-speed rail.

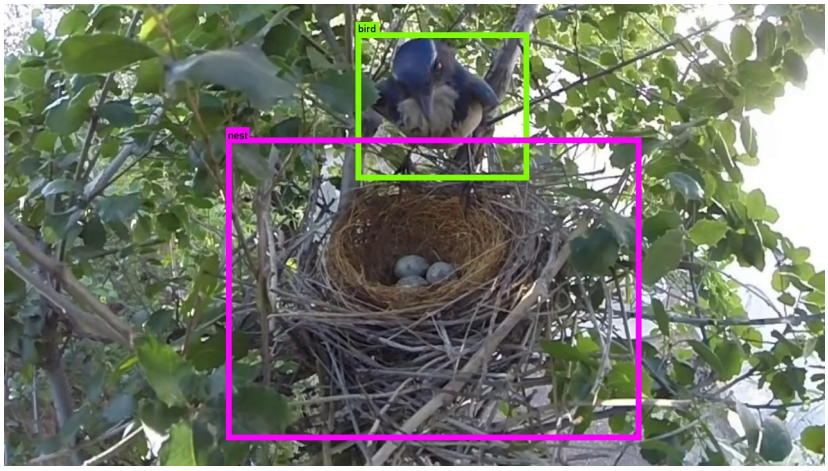


Figure 3.1: Sample bird and nest detections from a test video sequence

In the last few years, standard detection pipelines have been dramatically outperformed by *deep learning* representations. Deep convolutional neural network (CNN) architectures such as [71, 124] are able to generate high-level image representations that are effective for a variety of tasks. However, most CNN-based object detectors operate either in a sliding window fashion [113] or by generating object "proposals" separately [131, 172] and then evaluating these hypotheses. These approaches can achieve good results on difficult detection benchmarks, but are typically fairly slow and not well-suited to real-time deployment when compared to very fast and accurate special-purpose detectors (such as for pedestrians [13] and traffic signs [95]) built with other machine learning methods.

More recently, it has been shown that current CNNs have sufficient power to represent geometric information for localizing objects, opening the possibility of building state-of-the-art object detectors that rely exclusively on CNNs free of proposal generation schemes [105, 108, 75]. In such approaches, the network is trained end-to-end to predict both the appearance and geometric information of an object. At test time, given an input image, the entire network is only evaluated once instead of evaluating at different locations and scales of the image, enabling a large speed-up.

Inspired by these examples, in Sec. 2 we build on the general-purpose "YOLO" detection network [105], which exhibits excellent accuracy and runs at up to 150 Hz by directly

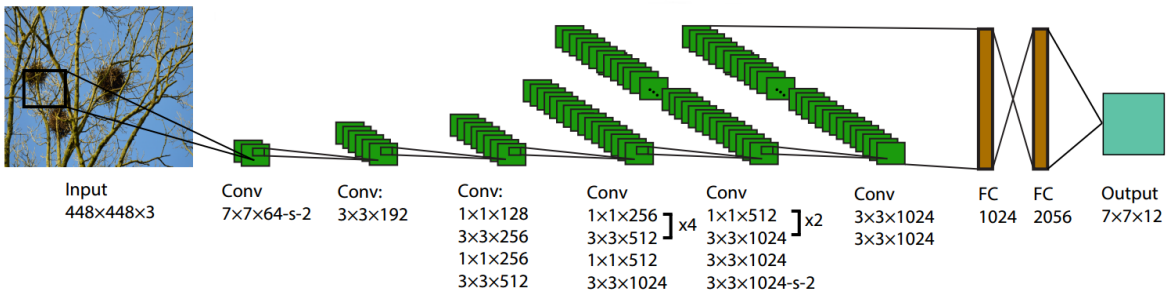


Figure 3.2: $\text{YOLO}_{\text{B+N}}$ detection pipeline (here $S = 7$, $B = 2$, and $C = 2$)

outputting detection bounding boxes with confidences. We improve upon the original network by making several modifications, and specialize it by training on only our two classes "bird" and "nest." To this end, we contribution a new dataset for nest detection, described in Sec. 4.1.

The speed of the detector permits it to be integrated into a real-time tracker. One advantage of a deep CNN tracker vs. most standard template-based trackers [145] is "automatic" initialization: because it has an *a priori* class concept, it can find the object(s) itself, and refind it/them if occlusions or mistracking occurs. There has been some recent work on applying deep learning techniques to visual tracking, or so-called "deep tracking" [134, 80, 133], but these are still relatively slow. In Sec. 3 we extend the baseline YOLO detector to improve the temporal smoothness of the localization estimate while retaining robustness to object appearance and pose changes.

3.2 Detection

We adapt the 24-layer YOLO network [105] for detection tasks, which we term $\mathbf{YOLO_{B+N}}$ ("YOLO Birds + Nests"). $\text{YOLO}_{\text{B+N}}$ has approximately the same architecture as the GoogLeNet proposed by [124], except that the inception modules are replaced by 1×1 reduction layers $+ 3 \times 3$ convolutional layers. The full network structure is shown in Fig. 3.2: it takes a raw input image, resizes it to 448×448, and outputs the size and location of bounding boxes for all C classes.

The resized input image is divided into an $S \times S$ grid of cells, each of which contains information on B hypothetical object bounding boxes. Each bounding box is parametrized by a 5-D vector $[x, y, w, h, P(Obj)]$, where $P(Obj) = 1$ if the center of any ground-truth

object bounding box is inside the cell and $P(Obj) = 0$ otherwise. Each grid cell also includes a conditional class probability: $\Pr(c \mid Obj)$, where $c \in \{C\}$. Accordingly, the class-specific confidence is given by: $P(c) = Pr(c \mid Obj)P(Obj)$. For each presented image, the output layer of the network is an $S \times S \times (B * 5 + C)$ tensor. Non-maximal suppression is used to remove duplicate detections, followed by thresholding on $P(c)$.

Our modifications are as follows. First, during the training stage, except the final layer which uses a linear activation function ϕ, all other layers in $\mathrm{YOLO_{B+N}}$ use *softplus* activation [28]: $\phi_{\mathrm{YOLO}_{B+N}}(x) = ln(1 + e^x)$. This gives a smoother approximation than the leaky activation function in the original implementation in [105]. Second, in [105] a term in the network loss function containing the square roots of the bounding box width and height is used to address the fact that small deviations in large boxes should weigh less than in small boxes.

We got better results using normalized coordinates to equally weigh errors between large and small boxes:

$$\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} 1_{ij}^{obj} (\frac{w_i - \hat{w}_i}{\hat{w}_i})^2 + (\frac{h_i - \hat{h}_i}{\hat{h}_i})^2 \tag{3.1}$$

3.3 Tracking

A "naive" $\mathrm{YOLO_{B+N}}$ tracker consists of running the detector on each successive frame independently. As seen in Table 3.1, this approach surpasses a number of recent trackers benchmarked in [145]. However, it still misses detections in isolated frames, and the localization is a little noisy, suggesting the introduction of a temporal filter.

We accomplish this by running $\mathrm{YOLO_{B+N}}$ and a template-based tracker simultaneously, combining them to create a hybrid detector-tracker which we call $\mathbf{TrackYOLO_{B+N}}$. For the single-object tracker here, we use the very fast kernelized correlation filter (KCF) [51] (coded as "CSK" in [145]), which maintains a trained linear classifier for all frames since last initialization. The two threads are combined as follows:

- When $\mathrm{YOLO_{B+N}}$ first detects an object, KCF is initialized using the highest-probability detected bounding box

- Let $B^t_{\mathrm{YOLO_{B+N}}}$ and B^t_{KCF} denote the output bounding boxes at time t of the base detector and the template tracker, respectively, where $B = [x, y, w, h]$

- Let $\Delta(t) = ||B^t_{\mathrm{YOLO_{B+N}}} - B^t_{KCF}||$ be a "disagreement" measure for each frame that $\mathrm{YOLO_{B+N}}$ has at least one detection, and define a threshold $\epsilon = 0.5 \times \max(W/S, H/S)$ where $W \times H$ are the image dimensions and S is the YOLO detection grid size

- If $\Delta(t) < \epsilon$, the hybrid tracking solution is a linear combination:
  $B^t_{\mathrm{TrackYOLO_{B+N}}} = \lambda_1 B^t_{\mathrm{YOLO_{B+N}}} + \lambda_2 B^t_{KCF}$.
  Else, clear the training buffer of KCF and fall back to the detector:
  $B^t_{\mathrm{TrackYOLO_{B+N}}} = B^t_{\mathrm{YOLO_{B+N}}}$

- Finally, if $\mathrm{YOLO_{B+N}}$ does not detect an object, the template tracker alone is used:
  $B^t_{\mathrm{TrackYOLO_{B+N}}} = B^t_{KCF}$

3.4 Experiments

3.4.1 Dataset

Bird nests We collected 114 images from the web, each containing at least one nest from a variety of species, for a total of 169 nest instances[1]. A wide range of scales were included, from close-ups to very distant views, and image resolutions ranged from 500×333 pixels to 4000×3000. Some examples (with results overlaid) can be seen in Fig. 3.5(a).

Birds We used the *bird* object category from the 2012 PASCAL VOC dataset [31], a widely-used benchmark in visual category classification, detection, and segmentation. VOC 2012 has 20 classes; there are 765 images containing birds in the *trainval* portion of the data, with 1,165 bird instances present. Samples are in Fig. 3.5(b).

Tracking Neither birds nor nests are in standard tracking benchmarks [145], so we prepared several image sequences from YouTube videos; 3 are presented here. For each, we manually chose ground truth bounding boxes every 10 frames and linearly interpolated them to generate annotations for all frames. The first sequence has 540 frames at 1920×1080 resolution.

[1] Full nest dataset available here: http://nameless.cis.udel.edu/data/nests

It is taken from a ground-based camera and the dominant motion is a zoom-in on a distant nest. The second sequence has 310 frames at 1280×720, and is from a drone flying around a tree containing a large bird nest. The third sequence has 564 frames at 854×450, and is a pan to follow a single bird flapping in front of a complex background. Samples from these sequences can be seen in Fig. 3.5(c)-(e).

3.4.2 Detection

We set the following parameters for $\mathrm{YOLO_{B+N}}$ training: batch size = 64, momentum = 0.6, decay = 0.001, learning rate = 0.0001, iterations = 5,000. For robustness, we perform data perturbation during training via random scaling and translations of up to 30% of the original image size and random adjustment of the exposure and saturation of the image by up to a factor of 2 in the HSV color space. Our model is pre-trained on the 1000-class ImageNet classification training set [71] and fine-tuned on the VOC 2012 trainval set containing only bird images and half of the nest dataset. For testing, detection threshold on P(c)$= 0.2$, and the correctness threshold on *Intersection over Union* (IoU) $= 0.5$.

Results are summarized in Table 3.1 in terms of mean Average Precision (mAP) [31] and time in seconds to process each image. For a baseline comparison (denoted "ImageNet-CNN"), we used the Caffe reference network [60] with approximately the same architecture as [71] and selective search to generate 4,000 object proposals per image. The network was trained and tested on the nest and bird data separately. mAP on both categories was quite low, and processing time very long. For a more competitive comparison, we refer to the PASCAL VOC 2012 detection task submissions [5]. At the time of submission, the leader using only PASCAL VOC data is "DenseBox", a VGG16-like CNN which performs end-to-end object detection [55]. DenseBox's mAP on the "bird" category is well below ours, and it is fairly slow and thus is not suitable for tracking.

When external training data is allowed, the current VOC 2012 detection leader is "ResNet", based on a residual network with a depth of over 100 layers [46]. Its "bird" mAP is the *only* submission higher than that of $\mathrm{YOLO_{B+N}}$, but at a cost of considerably more processing time. However, this number is not directly comparable to ours. All of the

Table 3.1: mAP %, time on detection, tracking datasets

	Nest	Bird	Ground nest	UAV nest	Flying bird	s/im
$\mathbf{YOLO_{B+N}}$	97.9	77.2	36.8	62.8	27.8	0.07
ImageNet-CNN	34.5	18.2				32.0
DenseBox [55]		28.8				≥ 2
YOLO [105]		57.7				0.07
ResNet [46]		84.8				≥ 2
$\mathbf{TrackYOLO_{B+N}}$			63.8	45.6	77.4	0.07
KCF ("CSK") [51]			15.9	54.6	74.3	0.001
CT [161]			19.9	59.8	81.4	0.03
SCM [166]			17.5	6.0	75.3	9.61

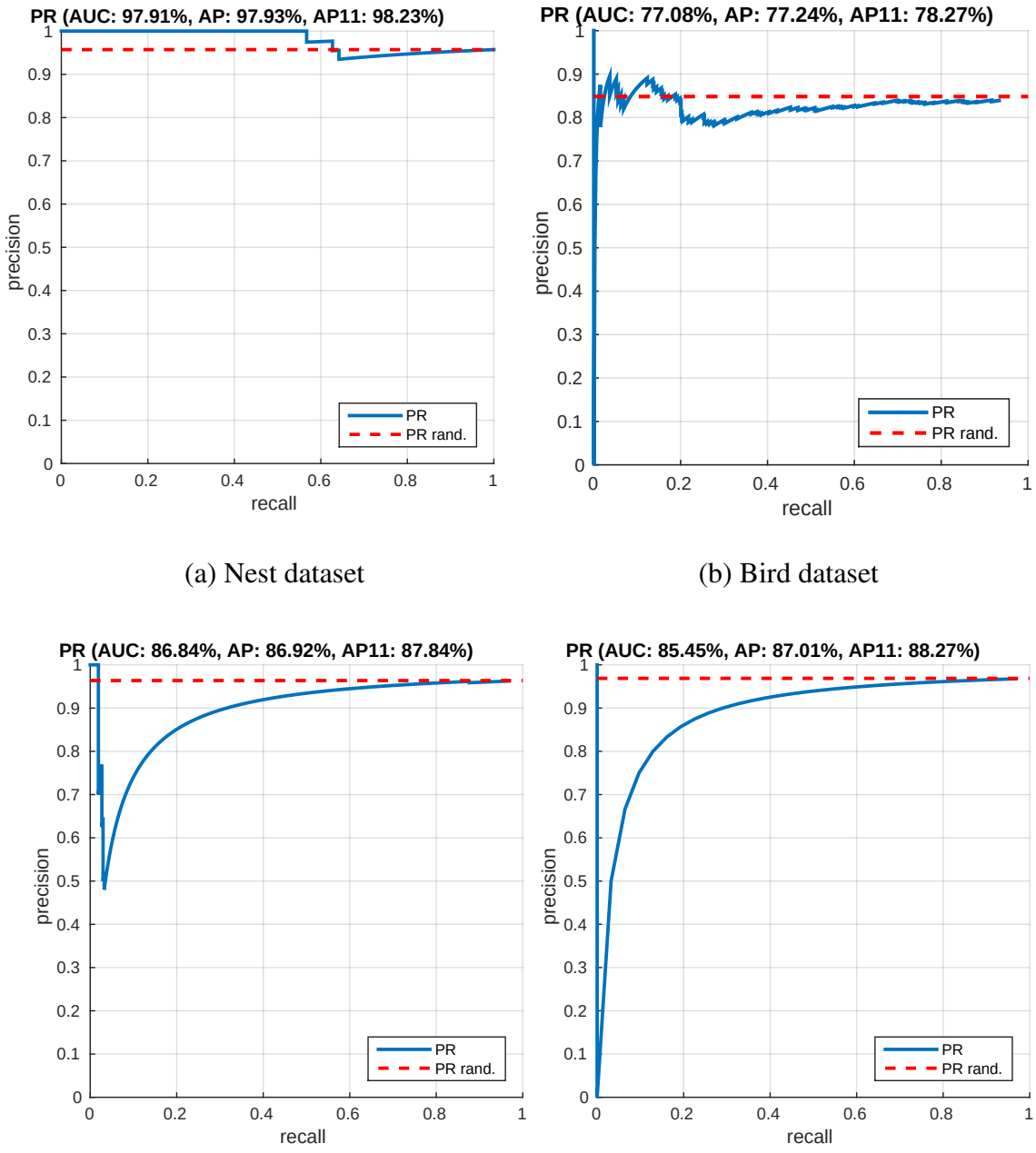
detectors submitted to [5] are attempting a harder task in that they are trained for $C = 20$ classes rather than $C = 2$ as we do. To capture the difference in difficulty, we note the lower mAP for the original YOLO [105], also using external training data.

We were only able to directly compare ImageNet-CNN to $\mathrm{YOLO_{B+N}}$ on the "nest" category, but we obtained a higher mAP for it than *any* algorithm on any other category in the VOC dataset. This may be because nests are rigid objects with relatively less appearance variation than other categories.

3.4.3 Tracking

Table 3.1 also shows tracking results for $\mathrm{YOLO_{B+N}}$ and $\mathrm{TrackYOLO_{B+N}}$ ($\epsilon = 60$ and $\lambda_1 = \lambda_2 = 0.5$) as compared to several trackers benchmarked in [145] (others were measured, but left out for space reasons). The comparison trackers and $\mathrm{TrackYOLO_{B+N}}$ were started on the ground truth bounding box in the first frame, whereas $\mathrm{YOLO_{B+N}}$ has to find the object by itself.

In all of the sequences, both YOLO-based trackers found and followed the object throughout the sequence, as seen in Figure 3.5(c)-(e). We observe that the ground nest sequence was most difficult for the comparison trackers, most likely because of its extreme scale change. $\mathrm{TrackYOLO_{B+N}}$ provided the most improvement on the flying bird sequence, because $\mathrm{YOLO_{B+N}}$ did not reliably detect the bird in certain phases of its flapping cycle.



(a) Nest dataset

(b) Bird dataset

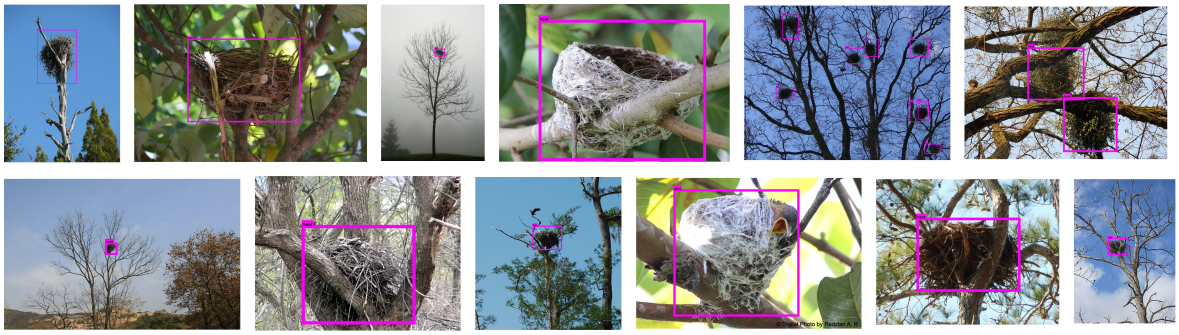
(c) Ground cam nest sequence

(d) UAV nest sequence

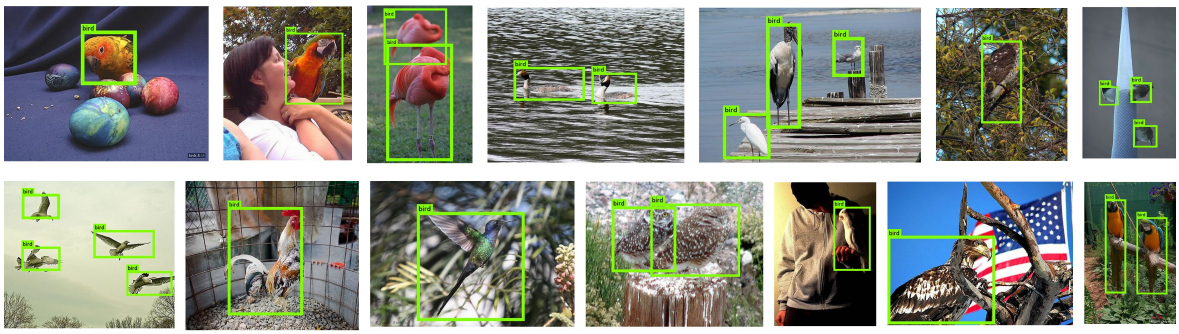
Figure 3.3: Quantitative PR-curves for different datasets

3.5 Conclusion

In this chapter we have presented a deep CNN system specialized for bird and nest detection and tracking that exhibits excellent accuracy and speed. Current work focuses on incorporating scene context (sky/ground/tree segmentations) into the detection process,



(a) Our nest dataset



(b) PASCAL VOC 2012 bird dataset

Figure 3.4: (a-b) Sample detection results of $\mathrm{YOLO}_{\mathrm{B+N}}$.

125 162 235 520

(c) Ground camera nest video sequence with frame numbers

10 87 210 312

(d) UAV nest video sequence

5 164 318 476

(e) Flying bird video sequence

Figure 3.5: (c)-(e) Sample tracking results, where blue bounding box is output of hybrid tracker $\mathrm{TrackYOLO}_{\mathrm{B+N}}$, yellow is naive tracker $\mathrm{YOLO}_{\mathrm{B+N}}$, and red is KCF [51] alone (initialized manually)

bringing more online learning into the tracking process without impacting speed severely, and extending the tracking to multi-object/class scenarios.

Chapter 4

JOINT LEARNING FOR OBJECT DETECTION AND FINE-GRAINED CLASSIFICATION

In this Chaper we propose a joint learning approach for object detection and fine-grained classification. Fine-grained classification is a challenging problem due to subtle differences between intra-class categories. In practice, fine-grained classification is often used in conjunction with object detection algorithms to locate and identify object categories. Despite recent achievements in both fine-grained classification and object detection, few works have demonstrated datasets or solutions to simultaneously handle both tasks. We make two contributions to this problem. Firstly, we construct a fine-grained classification and detection benchmark. Secondly, we show an end-to-end convolutional neural networks (CNNs) architecture to detect and classify fine-grained objects. Experimental results verify that our networks perform favorably against alternatives.

4.1 Introduction

Locating and identifying objects is important for many computer vision applications. For example, one could develop automated computer vision software to process live surveillance videos and recognize brand and model of a vehicle for identifying traffic violations. For environmental monitoring, locating and recognizing wildlife could generate statistics to help protecting endangered species. However, even for car enthusiasts or bird-watching experts, it is still difficult to identify a specific car model or bird species accurately.

Recent advances in deep learning have shown promising results for image classification and detection. PASCAL VOC [31] and ImageNet [71] are widely used to evaluate classification/detection performance. Both datasets provide annotations for generic object

classification and detection. In contrast to generic object classification, fine-grained classification aims at identifying objects within the same fine-grained category (or subcategory). Stanford Cars [70] and Caltech-UCSD Birds 200 (CUB-200) [142] are the two most popular benchmarks for evaluating such tasks. Comparing to the generic classification task, images in fine-grained datasets usually exhibit small inter-class and large intra-class variations in visual appearances. The small inter-class variation is due to the natural of fine-grained classification task, where all objects belonging to the same category share similar appearances. The large intra-class variation is introduced by the dataset, where objects are often presented in close-up photos with a combination of pose, viewpoint, illumination and background changes (see Fig. 4.1). A simple change in the camera perspective may lead to dramatic visual differences which could easily fool a neural network model trained on generic classification datasets.

While fine-grained datasets composed of close-ups are often geometrically warped comparing to photos from generic classification datasets, it enables learning of features more robust to variations in camera perspective and pose. Humans are able to rapidly identify the model of a car from key visual features such as the shape of the taillight or logo [154]. However, CNNs struggle on learning the fine details because it only learns the object appearance and lacks understanding of keypoint location, pose and geometry [59]. Recent work has shown that CNNs have remarkable capabilities to learn geometry-related information in convolutional layers [167, 99], but final fully-connected layers weaken this ability and tend to keep only category-level information. Based on this finding, researchers propose several two-stage object detectors [108, 47, 84] to use shared convolutional layers to find object proposals and output final predictions.

This leads to an open question: can we utilize a single network to learn both object-level localization information as well as stable fine-grained features invariant to imaging conditions and pose deformations? We address this question by creating a unified framework to perform *fine-grained object recognition*, subsuming the problems of object detection and fine-grained classification. Although many deep learning methods have been proposed for each task, we are not aware of any framework that directly train on both fine-grained classification and generic object detection datasets. By learning both spatial locations and

Figure 4.1: Sample images from the PASCAL VOC and CUB-200-2011 datasets. Compared to generic object detection datasets, fine-grained detection datasets include more close-up photos to include details.

intra-class diversities of an object, we enable the network to produce quality feature vectors with high distinctiveness. Additionally, as most convolutional layers in our network are shared for both tasks, we introduce very little computational overhead to achieve combined goals of efficiency and accuracy.

The rest of this chapter is organized as follows. Section 4.2 discusses related work. Section 4.3 details our overall framework based on the proposed method. We show experimental results in Section 4.4 and conclude in Section 4.5.

4.2 Related Work

Visual Object Classification: Visual object classification aims at predicting class labels from a given image. Deep convolutional neural networks (DNNs) have led to tremendous success in visual recognition [71, 50]. Deep networks naturally enforce features from different levels and can be trained easily in an end-to-end fashion to reach high accuracy. Highway Networks [119] were the first to introduce bypassing paths to tackle the gradient vanishing problem. ResNet further utilizes identity mapping as bypassing paths and achieved remarkable performance in benchmark datasets such as ImageNet and Microsoft COCO [50]. Experiments have shown that ResNet is able to converge on as many as 1000 layers. In the mean time, researchers also attempt to make the network wider. [126] proposes wider residual blocks to achieve better accuracy.

Fine-grained Classification: As a direct extension of the visual object classification problem, fine-grained recognition is the process of identifying objects within the same category. It can also be viewed as classifying subcategories. Fine-grained categorization has been extensively investigated and various fine-grained classification datasets are proposed. These studies include bird species classification [155, 69] and recognition of car brand, year and model [70, 154]. [162] was among the first to employ deep object detection networks for fine-grained classification. [56] leverages a convolutional networks to locate multiple object parts and a two-stream classification network to encode object and part level information. Spatial transformer networks [59] explicitly allow spatial manipulation of training data, giving neural networks the ability to actively transform the region of interest. Other research directions include attention models and feature pooling. Attention models aim at focusing the network on only a few distinctive image parts/keypoints [72, 165] while feature pooling collects second-order or higher-order statistics to form a more distinctive feature vector for better classification results [86, 67].

Object Detection: Object detection aims at finding locations of object instances in a scene. Recent work shows CNN has sufficient power to learning geometric representations to predict both the class label and geometric information of an object [167, 99, 108, 47]. Modern

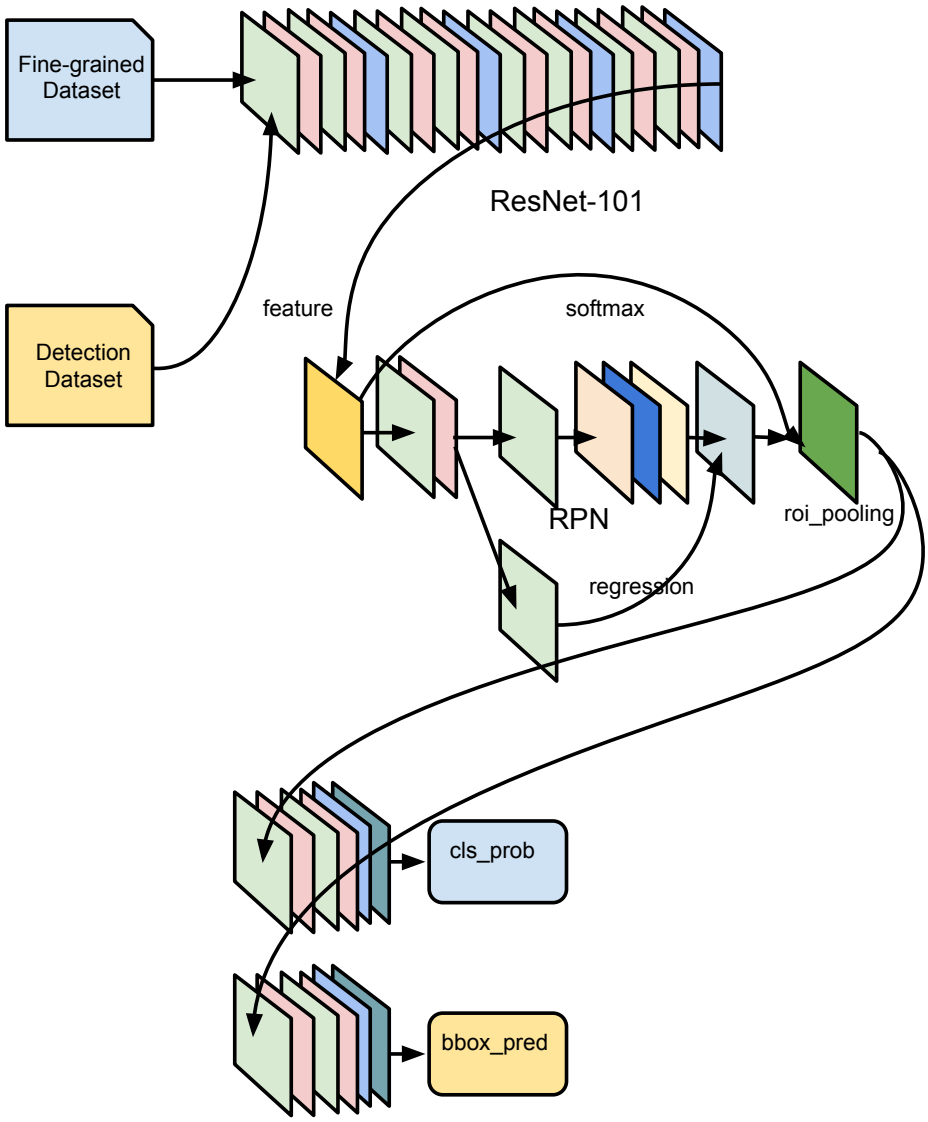


Figure 4.2: Architecture of our network. Our network is based on Faster-RCNN and branches after ROI Pooling layer for fine-grained classification.

detectors can be categorized as one-stage or two-stage frameworks. The one-stage detection framework [89, 106] is free of object proposals and can be trained in an end-to-end fashion. At test time, the entire network is only evaluated once, achieving real-time performance. The two-stage detection framework [108, 47], on the other hand, includes a class-agnostic region proposal generator and a classifier. Generally speaking, one-stage frameworks exhibit better efficiency and run at a higher frame rate, while two-stage frameworks are more accurate and are capable of locating smaller objects. Also, there is a growing interest in converting these frameworks into more compact versions for real-time/embedded systems [84].

Weakly-supervised Object Localization: The recent progress of deep learning is largely due to advances in high-power computing hardware and the availability of large-scale, high-quality annotated datasets. However, the annotation of ground truth labels is expensive and labor-intensive, and sometimes even impossible considering the scale of todays massive visual data. Therefore, it is important to develop unsupervised or weakly-supervised approaches to enable continuous learning. Researchers have investigated weakly-supervised object localization by studying maximal activations in the network layers [99, 167]. Our work attempts to learn from both detection and classification benchmarks without full annotation.

4.3 Approach

4.3.1 Network Architecture

In [108], Ren *et al.* designed the Faster-RCNN network which is composed of a Region Proposal Network (RPN) and a backbone CNNs. To improve efficiency, they also combined the RPN and the CNNs into a single network with a large number of shared layers. Faster-RCNN exhibits excellent tradeoff of speed and accuracy. It runs at near real-time speed (5fps) on a single GPU, while achieving state-of-the-art detection accuracy. Although Faster-RCNN gives perfect results on detection benchmarks, it can not be directly applied to the task of learning both detection and fine-grained classification. Therefore, we build upon the Faster-RCNN network architecture and made a few modifications. Firstly, we use ResNet-101 [50] as the backbone CNNs as it is deeper and gives better accuracy compared to

Figure 4.3: Randomly selected sample images from our FGR-4K benchmark. Compared to the Stanford Cars [70] dataset, our benchmark contains more real-life images with complex background.

the ZF or VGG16 network used in the original paper. Next, we made a new branch after the ROI Pooling layer for predicting fine-grained categories. Therefore, apart from the original RPN stream which predicts bounding box coordinates, the additional steam simultaneously predicts fine-grained class labels. Our network structure is illustrated in Fig. 4.2.

4.3.2 Fine-grained Recognition Benchmark

Existing fine-grained classification benchmarks such as Stanford Cars [70] only contain one object per image. Also, a large portion of the dataset is composed of stock images with clean/white background. To provide a fair technical benchmark for evaluating fine-grained classification and object detection performance, We introduce a new dataset called FGR-4K (Fine-grained Recognition 4K). We plan to open-source this dataset for research purposes. Our dataset is annotated using the same labels from the Stanford Cars dataset (see Fig. 4.3). This dataset is constructed for testing only, as training and validation data could

Algorithm 2 Training Strategy

Parameters: $\mathcal{M} \leftarrow$ CNNs Model
$\mathcal{D} \leftarrow$ Dataset

procedure TRAININGSTRATEGY($\mathcal{D}_{fg+det}$)
    $\mathcal{M}_{fg} \leftarrow$ LEARNINGPOLICY($\mathcal{M}_{ImageNet}, \mathcal{D}_{fg}$)
    $\mathcal{M}_{det} \leftarrow$ LEARNINGPOLICY($\mathcal{M}_{ImageNet}, \mathcal{D}_{det}$)
    $\mathcal{M}_{joint} \leftarrow$ `netsurgery`($\mathcal{M}_{fg}, \mathcal{M}_{det}$)
    $\mathcal{M}_{joint} \leftarrow$ LEARNINGPOLICY($\mathcal{M}_{joint}, \mathcal{D}_{fg+det}$)
    return $\mathcal{M}_{joint}$
end procedure

be obtained from Stanford Cars or PASCAL VOC car class. Our dataset provides 3818 images crawled from Google image search API. These images are filtered to include only those permitted for commercial reuse. We also run automated deduplication, white background detection and text detection algorithms to remove images not suitable for annotation. The deduplication is done by removing images with the same SHA256 checksum on RGB values. White background is identified by converting the image to a binary map using 33% above its median pixel value as the threshold. If white pixels occupy more than 50% of the image then we will remove it from the candidate set. We use the open source text detection library [4] to remove images detected with any words or letters. After the automatic filtering, we manually clean up the dataset to choose only real-life images with complex background. Next, we send the candidate set to human annotators to draw rectangles around all car objects that fall into the given 196 fine-grained categories. Compared to the Stanford Cars dataset, our benchmark contains more real-life images with complex backgrounds which are challenging for both fine-grained classification and object detection tasks.

4.3.3 Training Strategy

Because of the high diversity in fine-grained classes, we use on-the-fly data augmentation during training. The augmentation includes random cropping, resizing and rotation up to 30 degrees. We also included random smooth filtering and JPEG compression varying

from 50% to 90% quality. The online augmentation is only applied to training while the validation/testing accuracy is still reported on the original dataset. Due to the difficulty in training initial weights for the whole network, we first train a ResNet model on the fine-grained dataset and a Faster-RCNN model on the detection dataset separately. Note that these two networks are using the same ResNet-101 backbone and the only difference between them is the RPN network inserted between Res4b and Res5a layers. Once both models converge, we freeze all layers in the detection model by setting learning rate to 0 for all bottom layers below the ROI Pooling layer. Next, we append all top layers above res4b from the fine-grained model to the detection model as a separate branch and start finetuning the entire network. We use MSRA initialization [49] and SGD solver for optimization.

During training, we define the maximum number of iterations as max_iter and iterations per step as $step\_size$. The learning rate will drop by a factor of 10 (*i.e.* multiplied by lr_decay where $lr\_decay = 0.1$). Once the step size is reached, the learning rate decreases by 10. The accuracy jumps when learning rate changes. This is because the solver has been optimizing at a certain learning rate for a certain number of iterations to find the local optimum. The weight of the whole model stabilizes for the duration of a consistent learning rate. After the learning rate reduces, it is easier for the neural networks to capture fine details and increase accuracy. Assuming the initial warm-up learning rate to be lr_init, when we reach the max_iter the learning rate will be $lr\_init \times lr\_decay^{\max\_iter/step\_size}$. When the accuracy saturates, we will only fine-tune softmax layers to obtain the joint model. We illustrate the details of our training strategy in Algo. 2.

4.4 Experiments

4.4.1 Fine-grained Classification

We use Caltech-UCSD Birds-200-2011 (CUB-200-2011) [142] and Stanford Cars dataset [70] for fine-grained classification experiments. The CUB-200-2011 dataset contains 200 fine-grained bird categories with 11788 images. The Stanford Cars dataset contains 16185 images with 196 classes of cars including year, make and model. Details of the datasets are shown in Table. 4.2. For both datasets, We first fine-tune on the ImageNet

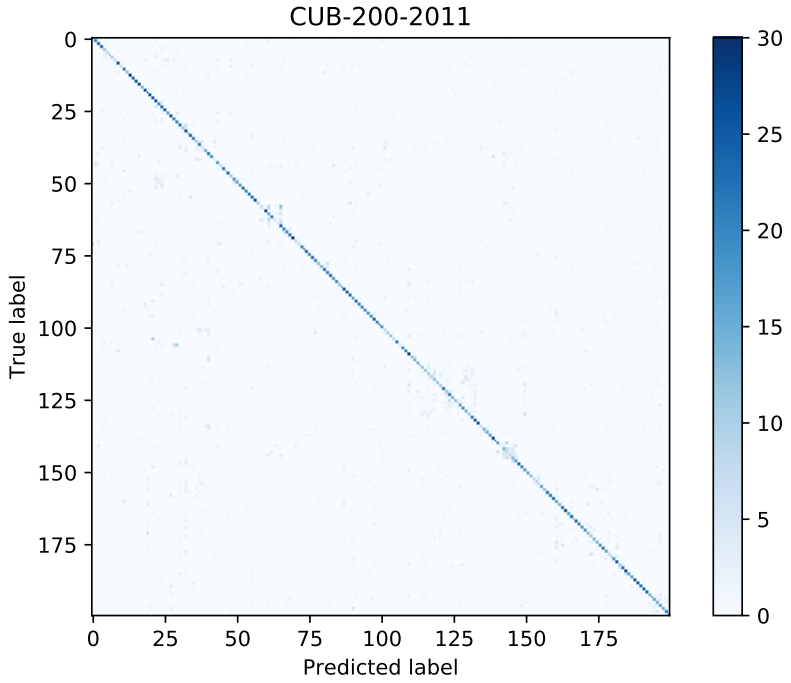
Dataset	Image	Class	Train	Val	Object
CUB-200 [142]	11788	200	5994	5794	11788
Stanford Cars [69]	16185	196	8144	8041	16185
VOC Bird [31]	765	1	395	370	1119
VOC Car [31]	1161	1	590	571	2017
FGR-4K	3818	196	N/A	N/A	4008

Table 4.1: Dataset Statistics. The PASCAL VOC Bird and Car dataset are derived from the PASCAL VOC 0712 dataset with only Bird or Car class with all other categories removed. Our FGR-4K dataset contains 3818 images with 196 labels inherited from the Stanford Cars dataset. Our dataset aims at providing a technical benchmark for testing purposes only.

CUB-200	Top-1	Stanford Cars	Top-1
Xiao *et al.* [146]	69.7	Krause *et al.* [69]	92.8
Simon *et al.* [115]	81.0	Lin *et al.* [86]	91.3
Kong *et al.* [67]	84.2	Zhang *et al.* [163]	88.4
Liu *et al.* [90]	85.4	Xie *et al.* [147]	86.3
Ours Initial	81.0	Ours Initial	90.1
Ours Joint	72.6	Ours Joint	86.2

Methods	Train	Test	Bird	Car
Faster-RCNN (VGG16) [108]	VOC 0712	VOC07	70.9	84.7
Faster-RCNN (ResNet50) [108]	VOC 0712	VOC12	74.3	75.9
SSD300 [89]	VOC 0712	VOC07	70.5	76.1
YOLO [106]	VOC 0712	VOC12	57.7	55.9
YOLOv2 544 [107]	VOC 0712	VOC12	74.8	76.5
Ours (ResNet101)	VOC 0712	VOC07	72.0	75.2
Ours (ResNet101)	VOC 0712	VOC12	77.9	78.0

Table 4.2: Results on Fine-grained Classification and Object Detection benchmarks. Our methods handles both tasks at the same time, while performs favorably against alternative methods designed for each task.



10:

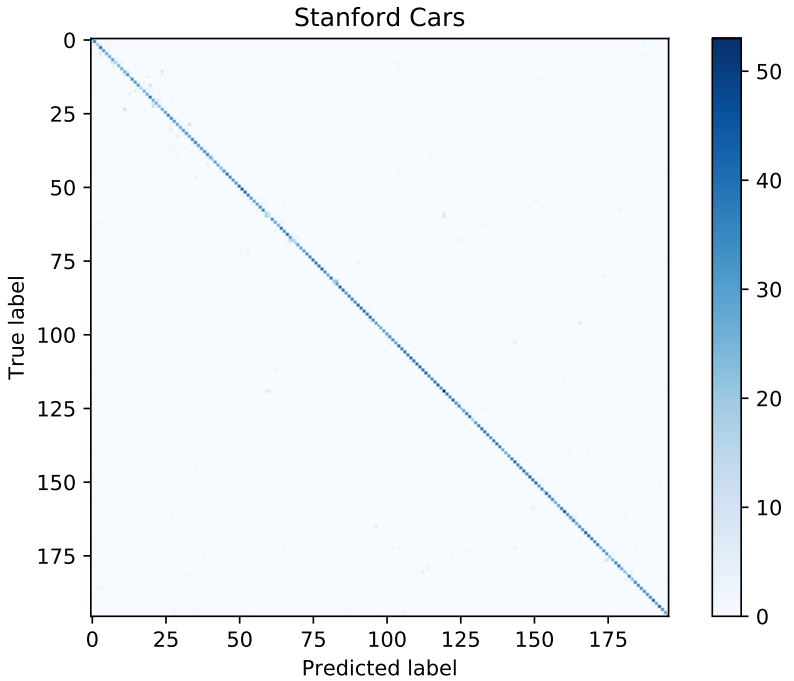


Figure 4.4: Confusion matrix on the CUB-200 and Stanford Cars datasets. The vertical axis shows the groundtruth labels while the horizontal axis shows the predicted labels. Note that our model makes more false positive predictions on the CUB-200 dataset compared to the Stanford Cars dataset. This is because CUB-200 contains non-rigid transformations with larger intra-category variance on object pose and appearance.

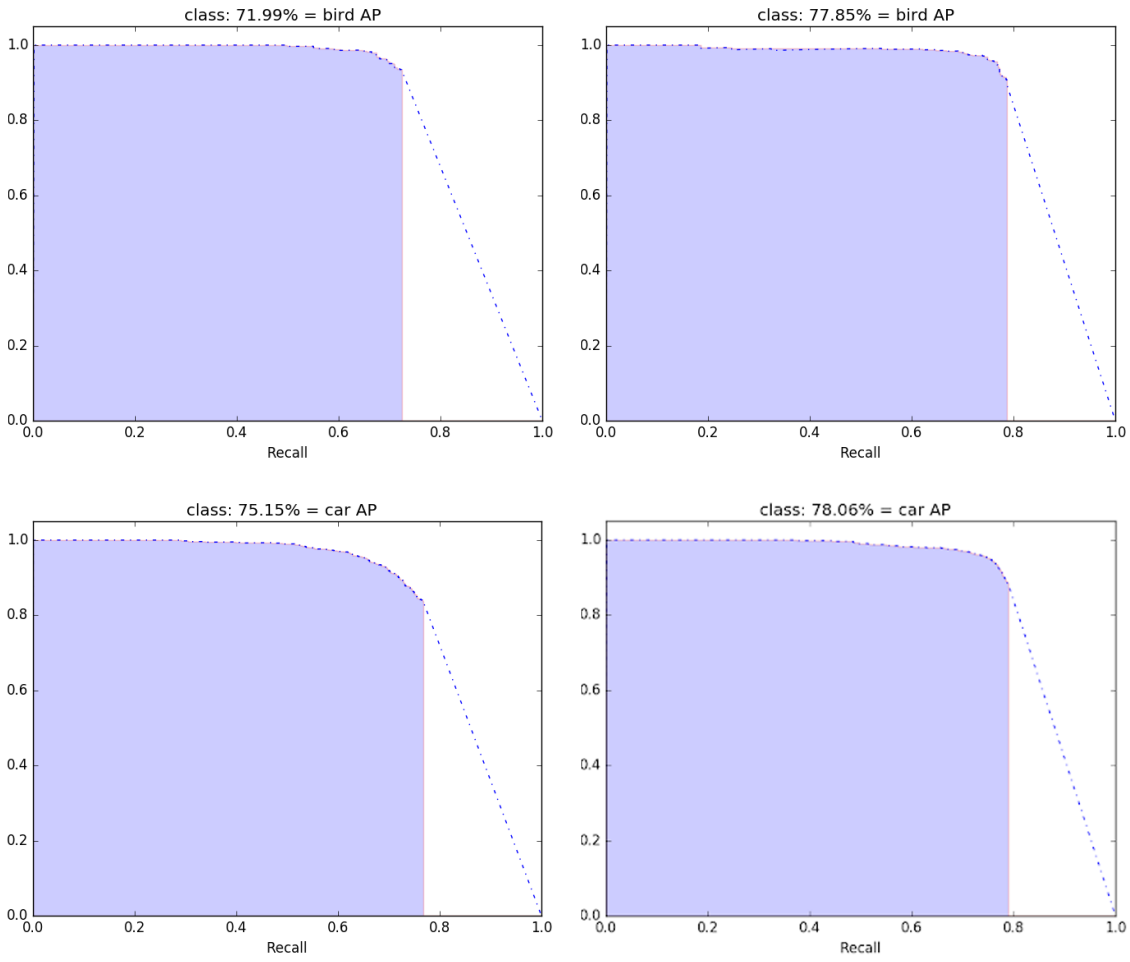
ResNet-101 model. The first round is trained from scratch with random weight initialization. The model is trained with the base learning rate of 0.01, gamma of 0.5, momentum of 0.9 and weight decay of 0.0001. After the first round reaches maximum number of iterations, we replace all bottom layers except the softmax layer with pre-trained ImageNet weights. The accuracy of the initial model trained from scratch is 66.0% and the one fine-tuned on ImageNet is 77.8 %. Next, we start fine-tuning this model with the training scheme mentioned in Sect. 4.3.3. Once the training accuracy is saturated, we fix all bottom layer weights and only fine-tune the softmax layer. The final accuracy is 81.0%. A comparison to related work is shown in Table 4.2.

4.4.2 Object Detection

We build our approach based on the state-of-the-art Faster-RCNN [108] detection framework. We start training the modified Faster-RCNN network explained in Sect .4.3 with ResNet-101 backend. The weights of ResNet-101 is initialized with pre-trained ImageNet weights. We train our model with base learning rate of 0.001, gamma of 0.1, momentum of 0.9 and weight decay of 0.0005. The training is performed on the combined PASCAL VOC 07+12 dataset on a single class (car or bird). After 1000K iterations the tested mAP is 71.99% on VOC 07 and 77.85% on VOC 12 for the bird class. We repeat the same procedure for the car class on VOC 07 and 12 and the final mAP for the car class is 75.15% and 78.06%, respectively. Quantitative evaluations can be found at Table. 4.2. Note that our accuracy is reported on images only containing the bird or car class with all other annotations removed, while the accuracy reported by all other algorithms still consider classes other than bird or car. The precision-recall curves are shown in Fig. 4.5. Qualitative results are shown in Fig. 4.9.

4.4.3 Joint Training

Now that we obtained models for both detection and fine-grained classification, we start merging the models for joint training. During inference time, the network will produce bounding box locations in an image, as well as fine-grained class labels for each bounding



(a) VOC07 Bird (b) VOC12 Bird (c) VOC07 Car (d) VOC12 Car

Figure 4.5: Quantitative PR-curves for Bird and Car class on PASCAL VOC07 and PASCAL VOC12 validation datasets.

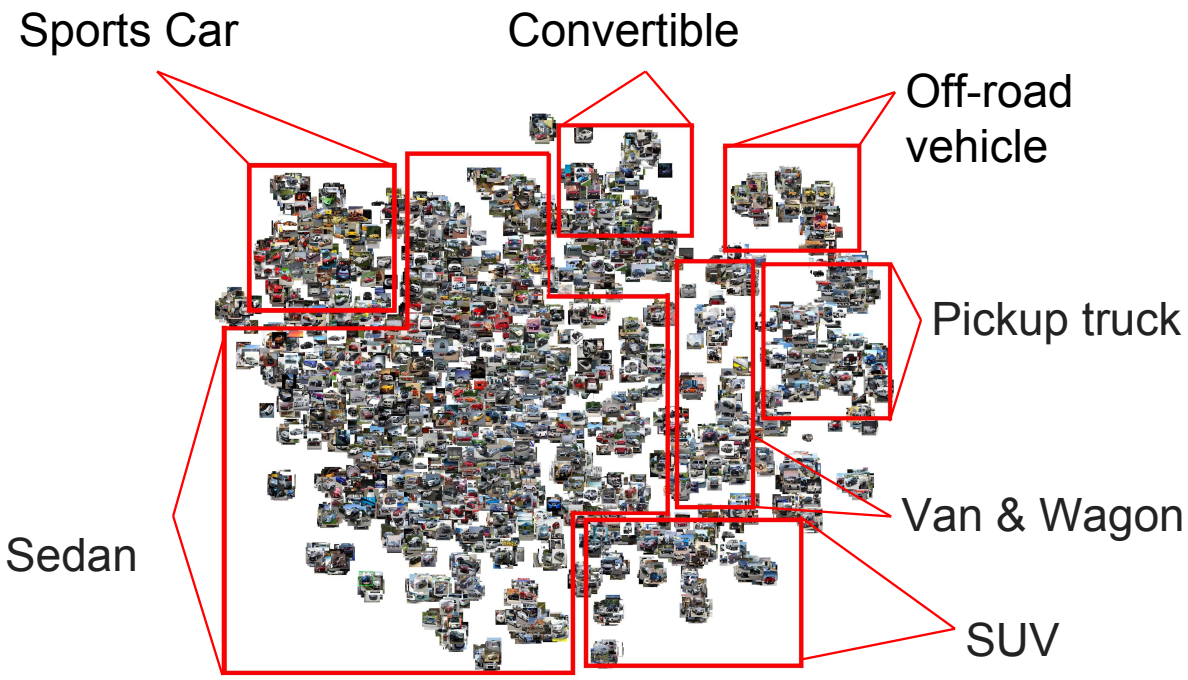


Figure 4.6: t-SNE visualization [92] of features extracted from the joint model on Stanford Cars test set (Best viewed zoomed in and in color). The similarities are calculated purely based on visual feature embeddings. This illustrates that our joint model is able to preserve fine-grained semantics information after dimensionality reduction.

box. We freeze all layers in the detection model by setting learning rate to 0 for all bottom layers below the roi_pool5 layer. Next, we append all top layers above res4b from the fine-grained classification model to roi_pool5 layer in the detection model. Because of weight discrepancies between the original bottom layers and the new bottom layers trained on the object detection dataset, the test accuracy of the joint model on CUB-200-2011 drops from 81.0% to 67.5%. After joint training for another round (1000K iterations, freezing bottom layers) the fine-grained classification accuracy goes up to 72.6%. We performed the same net-surgery and training procedures for the car class. The final classification accuracy for cars is 86.2%. The detection accuracy stays the same since we are only training the fine-grained branch with all other weights fixed. We found that this training schedule leads to the best results comparing to alternative approaches such as adjusting weights for all layers. The whole training process is illustrated in Algorithm 2. Next, we apply our joint model to the FGR-4K benchmark dataset. We show the precision-recall curve for the best and worst performing 30 categories in Fig. 4.7. This is done by varying the IoU threshold from 0.5 to 0.9 and recalculate mAP scores for all classes.

We also compare our model with the commercial vehicle recognition service provided by Sighthound [3] on the FGR-4K dataset. Since Sighthound API only returns vehicle brand and model, we remove the year info from both the groundtruth and predicted results on the FGR-4K dataset for comparison. The final average mAP of our method is 62.59% while the average mAP of Sighthound is 56.88%. We show True /False predictions for each fine-grained class in Fig. 4.8. Note that the production Sighthound model is possibly trained on a enlarged dataset which includes more vehicle models than the FGR-4K dataset.

4.4.4 Analysis

For the fine-grained classification experiments, we show the confusion matrix on the CUB-200 and Stanford Cars datasets in Fig. 4.4. As shown in Fig. 4.4, the CUB-200 contains objects with non-rigid transformations and is more challenging compared to the Stanford Cars dataset. Also, according to Table. 4.2, there is a larger gap between the classification accuracy of our model and attention-based models [90, 69] on CUB-200

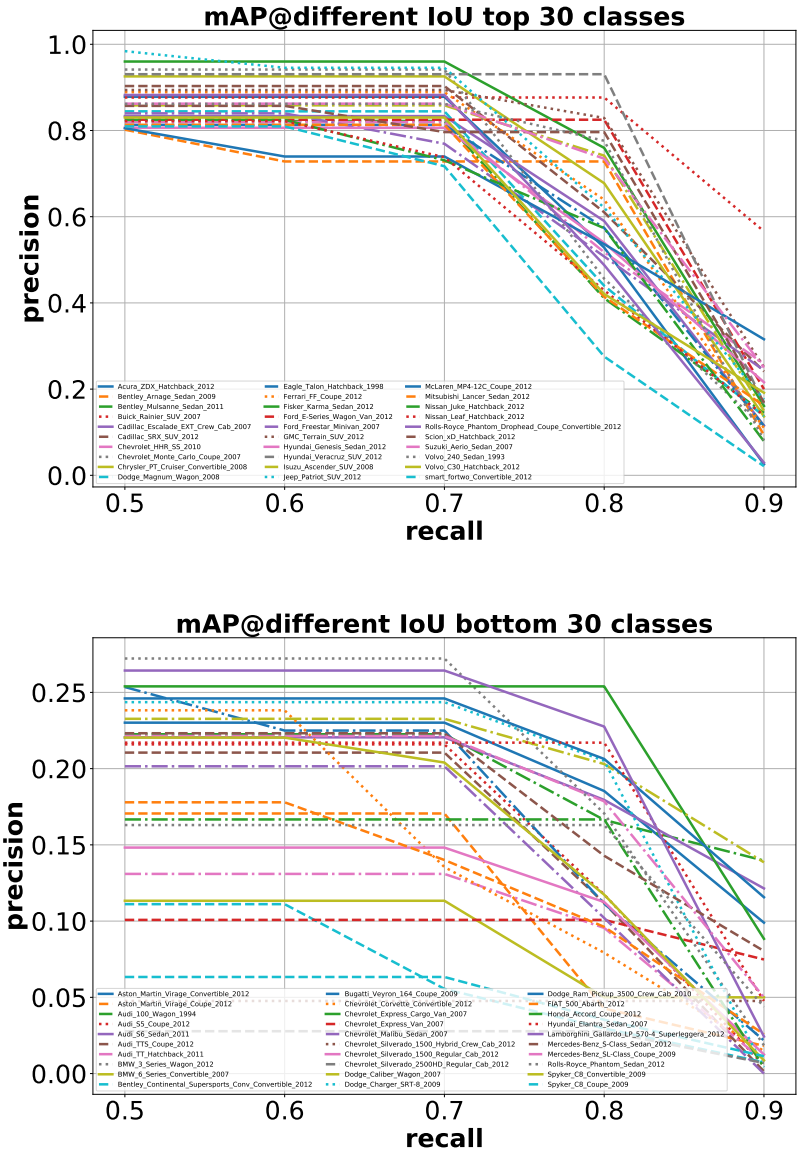


Figure 4.7: Precision-recall curve of our joint model on the FGR-4K dataset (Best viewed electronically). Left: PR curve on the top-performing 30 classes. Right: PR curve on the lowest-performing 30 classes. Note that our model is robust to stricter IoU criterias and the performance starts to degrade when IoU is increased to more than 0.7.

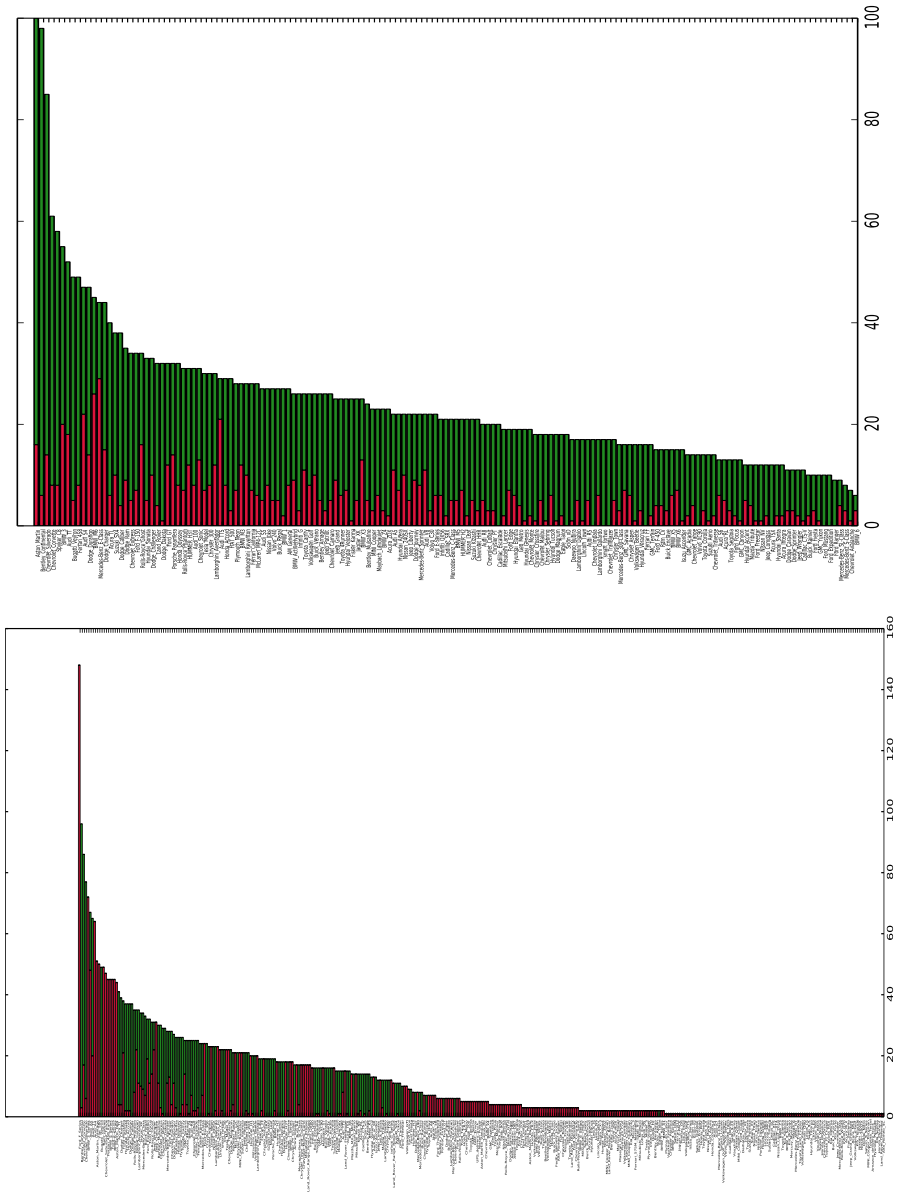


Figure 4.8: mAP and TP/FP scores of our joint model on the FGR-4K dataset (Best viewed electronically). Left: True (green)/False (red) predictions using our approach. Right: True (green)/False (red) predictions obtained by Sighthound Cloud API for vehicle recognition [3]. The average mAP of our approach is 62.59% while the average mAP of the Sighthound Cloud API is 56.88 %. Note that our model predicts less false alarms comparing to the Sighthound production model.

.

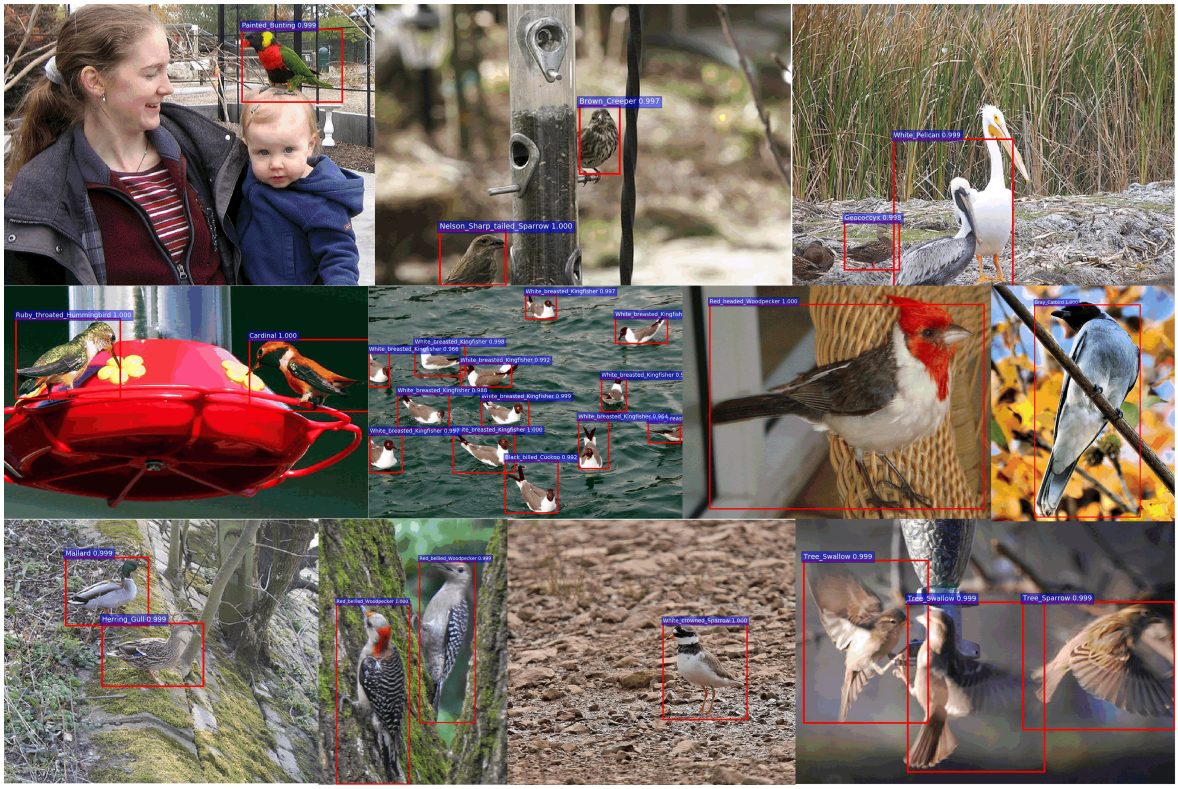
compared to Stanford Cars. Furthermore, our joint model suffers more accuracy degradation on the CUB-200 dataset compared to the Stanford Cars dataset. Despite these limitations, our model is able to learn from two vastly distinctive datasets and demonstrate competitive performance compared to methods developed for each task. We apply t-SNE visualization [92] to features extracted from the joint model on Stanford Cars test set and visualize the embeddings in Fig. 4.6. This illustrates that our joint model is able to preserve fine-grained semantics information in the high-dimensional space. The t-SNE visualization also indicates that our learnt features are able to capture visual similarities but is less sensitive to pose variations. For the FGR-4K dataset, we evaluate the detector performance against varying IoU thresholds in Fig. 4.7. As can be seen from the figure, the mAP per category only starts decreasing when IoU is more than 0.7. This shows that our detector is robust to stricter evaluation criterias, which is generally more desirable for real life vision applications. We also notice that the best-performing class labels are mostly composed of visually distinctive car models from different manufacturers, while the lowest-performing classes are more often from the same manufacturer with similar car-model names. This implies that the current joint network is good at detecting objects but is still having difficulties capturing small partial details within the object. Therefore, in the future we could consider adding attention-based models or feature pooling methods to better handle small details and improve accuracy.

We show qualitative results in Fig. 4.9 (a) by running forward-inference on this joint model. Note that our model is able to predict fine-grained class labels not present in the PASCAL VOC dataset. As shown in Fig. 4.9 (a), our joint network predicts accurate bounding box locations for all bird objects in an image. In addition, our network is good at recognizing subtle color (e.g. Red headed woodpecker in row 2 column 3, Gray Catbird in row 2 column 4, Red bellied woodpecker in row 3 column 2 and White crowned sparrow in row 3 column 3) and shape variations (e.g. difference between Mallard and Herring Gull in row 3 column 1). In Fig. 4.9 (b), the model is able to recognize subtle differences between two similar-looking cars with the same color (e.g. ”Jaguar XK XKR 2012” in row 2 column 1 vs. ”BMW M6 Convertible 2010” in row 3 column 4) and partial object with occlusion (e.g. ”Mercedes Benz SL Class Sedan 2012” in row 1 column 5). For object classes not present

in the Stanford Cars dataset, the network is able to assign a label with a closest visual match (e.g. ”Dodge Challenger SRT8 2011” in row 1 column 3).

4.5 Conclusion

In this chapter we have presented a framework to detect and classify fine-grained objects. To evaluate performance, we have created a new benchmark for fine-grained recognition. Experiments show that our approach performs favorably against competitive methods. In summary, our network structure provides more desirable characteristics for practical computer vision applications and reaches good balance between the model size, computational complexity and accuracy. Our system can be used to add visual intelligence to mobile devices. This feature is particularly useful for ornithologists or car enthusiasts who wish to identify or search for a particular object of interest. In the future, we plan to leverage post-training quantization techniques to compress our joint model and enable fast forward-inference on mobile apps.



(a) Results on PASCAL VOC 0712 bird class

(b) Results on PASCAL VOC 0712 car class

Figure 4.9: Qualitative results on PASCAL VOC 0712 Bird and Car classes [31]. The labels are learnt from CUB-200-2011 [142] and Stanford Cars dataset [70], respectively.

Chapter 5

RGB-D DATA ACQUISITION AND REFINEMENT ON A MOBILE DEVICE

From this chapter on we aim at moving from 2D to 3D space. To begin with, we introduce an algorithm to capture accurate 3D data on a low-cost mobile device. We calibrate the stereo cameras and rectify the stereo image pairs through FCam API, then generate a low-res disparity map using graph cuts stereo matching and subsequently upsample it via joint bilateral upsampling.

Further, we show a method for realistic depth of field (DoF) rendering using the captured high resolution disparity map. We generate a synthetic light field by warping the raw color image to nearby viewpoints, according to the corresponding values in the high-resolution disparity map. Next, we render dynamic DoF effect on the tablet screen with light field rendering. The user can easily capture and generate desired DoF effects with arbitrary aperture sizes or focal depths using the tablet only, with no additional hardware or software required. The system has been examined in a variety of environments with satisfactory results.

5.1 Introduction

Recent advances in commodity depth cameras enable the user to capture 3D scene geometry. This brings huge potential for mobile robotic applications. However, due to cost and manufacturing constraints, it is still difficult and expensive to integrate 3D sensors into small hand-held devices like smartphones or tablets.

To address this problem, we develop a an algorithm to calculate and refine depthmaps on mobile devices with dual-cameras. Because our algorithm works on regular stereo camera systems, it can be directly applied to existing consumer products such as 3D-enabled mobile phones, tablets and portable game consoles. We also consider the current status of mobile

computing devices in our software system design and make it less platform-dependent by using common libraries such as OpenCV, OpenGL ES and FCam API. We start by using two cameras provided by the NVIDIA Tegra 3 prototype tablet to capture stereo image pairs. We subsequently recover high-resolution disparity maps of the scene through Graph Cuts (GC) [66] and then refine the depthmap using joint bilateral upsampling. We implement our algorithm on the NVIDIA Tegra 3 prototype tablet under the FCam architecture [9]. Experiments show that our system produces accurate depthmaps in both indoor and outdoor scenes with various depth ranges.

Also, we demonstrate that DoF effects can be rendered using the proposed approach on low-cost mobile devices. We first capture stereo image pairs by using the FCam API, then apply the Graph Cuts stereo matching algorithm to obtain low-resolution disparity maps. Next, we take raw color images as guide images and upsample the low-resolution disparity maps via joint bilateral upsampling. Once the high-resolution disparity maps are generated, we can synthesize light fields by warping the raw color images from the original viewing position to nearby viewpoints. We then render dynamic DoF effects by using the synthetic light fields and visualize the results on the tablet screen. We evaluate a variety of real-time stereo matching and edge-preserving upsampling algorithms for the tablet platform. Experimental results show that our approach provides a good tradeoff between expected depth-recovering quality and running time. All the aforementioned processing algorithms are implemented to the Android operating system and tested on the Tegra 3 T30 prototype tablet. The user can easily install the software, capture and generate desired DoF effects using the tablet only, with no additional hardware or software required. The system has been tested in a variety of environments with satisfactory results.

5.2 Related Work

Since conventional imaging systems are only two-dimensional, a variety of methods have been developed for capturing and storing light fields in a 2D form. Ives and Lippmann [87] were the first to propose a prototype camera to capture light fields. The Stanford multi-camera array [143] is composed of 128 synchronized CMOS firewire cameras and streams

captured data to 4 PC hosts for processing. Because of the excessive data volume, DoF effects are rendered offline. The MIT light field camera array [152] uses 64 usb webcams and is capable of performing real time rendering of DoF effects. However, these camera systems are bulky and hard to build. Recently, Ng [97] has introduced a new camera design by placing a microlens array in front of the sensor with distance equals microlens focal length, wherein each microlens captures a perspective view in the scene from a slightly different position. However, the spatial resolution near the microlens array plane is close to the number of microlenses. To overcome this limitation, Lumsdaine and Georgiev [36] introduced the focused plenoptic camera which trades angular resolution for spatial resolution. An alternative approach is to integrate light-modulating masks to conventional cameras and multiplex the radiance in the frequency domain [132]. This design enables the camera sensor to capture both spatial and angular frequency components, but reduces light efficiency.

As rapid research and development provide great opportunities, hand-held plenoptic camera has been proven practical and quickly progressed into markets. The Lytro camera [91, 37] is the first implementation of a consumer level plenoptic camera. Recently, Pelican Imaging [101] announced a 16-lens mobile plenoptic camera system and scheduled to implement it to new smartphones in 2014.

Our work is inspired by the algorithms proposed by Yu *et al.* [159, 158, 157]. However, the system proposed in these two papers is bulky, expensive and the algorithm is highly dependent on the GPU performance, making it hard to transfer the proposed method to small handheld devices such as cellphones and compact size cameras. The system used by Yu *et al.* [159] is composed of a desktop workstation and a customized stereo camera system. The desktop is equipped with a 3.2 GHz Intel Core i7 970 6-core CPU and a NVIDIA Geforce GTX 480 Graphic Card with 1.5 GB memory. Actually, very few laptops on the market can reach the same level of performance, let alone tablets or cellphones. Also, this system connects to two Point Grey Flea 2 cameras via a Firewire link. The retail price for two Flea cameras is around 1500 dollars and the camera itself requires external power source, professional software for functionalities such as auto exposure, white balancing and stereo synchronization, which is almost impractical for general users without a computer

vision background. In addition, most scenes in this paper are indoor scenes with controlled lighting, and the user is required to tune different parameters on a GUI in order to obtain a good-looking disparity map in different scenes. In contrast, our software system works directly on an off-the-shelf tablet which costs less than 400 dollars. Since our algorithm is implemented under the Android operating system using highly optimized CPU-only functions from OpenCV4Android SDK, it can be easily ported to other handheld Android devices with limited computational power. Besides, we conducted extensive experiments to obtain parameters that generate optimal results. Therefore, it's easy to install and use our software, no hardware setup or parameter adjustment is required. Furthermore, our system uses Graph Cuts [16] instead of Belief Propagation (BP) [122] for stereo matching, and is tested working under complex illumination conditions. According to tests carried out by M. Tappen *et al.* [125], Graph Cuts generates smoother solutions compared to BP and consistently performs better than BP in all quality metrics for the Middlebury [111] Tsukuba benchmark image pair. To conclude, we made the following contributions:

- We implemented the entire system on an off-the-shelf Android tablet using highly optimized CPU-only functions from OpenCV4Android SDK. The system can be easily ported to other mobile photography devices with limited computational power.
- We conducted extensive experiments to obtain the optimal combination of methods and parameters under the Tegra 3 T30 prototype device. As a result, there is no need for parameter adjustment and it is easy for the user to install and use our application.
- We experimented with Graph Cuts for disparity map calculation and the system is capable of working with a variety of scene structures and illumination conditions.

5.3 Overview

5.4 Programmable Stereo Camera

5.4.1 Development Environment

The Tegra 3 T30 prototype tablet is equipped with a 1.5 GHz quad-core ARM Cortex-A9 CPU and a 520 MHz GPU. It has three sensors. The rear main sensor and secondary

sensor are identical with a 6 centimeter baseline. The third sensor is on the same side of the multi-touch screen facing the user. The raw image resolution is 640×360(16:9).

Our software is running under Android 4.1(Jelly Bean) operating system. We use the Tegra Android Developer Pack (TADP) for building and debugging the application. This software toolkit integrates Android SDK features to Eclipse IDE by using the Android Development Tools (ADT) Plugin. ADT extends the capabilities of Eclipse and enables the user to design graphic UI, debug the application using SDK tools, and deploy APK files to physical or virtual devices. Since typical Android applications are written in Java and compiled for the Dalvik Virtual Machine, there's another toolset called Android Native Development Kit (NDT) for the user to implement part of the application in native code languages such as C and C++. However, using the NDT brings certain drawbacks. Firstly, the developer has to use the NDT to compile native code which hardly integrates with the Java code, so the complexity of the application is increased. Besides, using native code on Android system generally does not result in a noticeable improvement in performance. For our application, since we need to use the FCam API for capturing stereo pairs, OpenCV and OpenGL ES for image processing and visualization, we implemented most of the code in C++ and run the code inside the Android application by using the Java Native Interface (JNI). The JNI is a vendor-neutral interface that permits the Java code to interact with underlying native code or load dynamic shared libraries. By using the TADP, our workflow is greatly simplified. We first send commands to the camera using the FCam API, then convert raw stereo image pairs to *cv::Mat* format and use OpenCV for rectification, stereo matching, joint bilateral upsampling and DoF rendering. The final results are visualized on the screen using OpenGL ES.

5.4.2 The FCam API

Many computational photography applications follow the general pattern of capturing multiple images with changing parameters and combine them into a new picture. However, implementing these algorithms on a consumer level tablet has been hampered by a number

of factors. One fundamental impediment is the lack of open software architecture for controlling the camera parameters. The Frankencamera [9] proposed by Adams *et al.* is the first architecture to address this problem. Two implementations of this concept are a custom-built F2 camera and a Nokia N900 smartphone running a modified software stack to include the FCam API. Alejandro *et al.* extended the implementation of FCam API to support multiple cameras [130] and enables the NVIDIA Tegra 3 prototype tablet to trigger stereo captures.

5.4.3 Calibration, Synchronization and Autofocus

Since the two sensors are not perfectly aligned, we calibrated the stereo pair using a planar checker board pattern outlined by Zhang [164]. We computed the calibration parameters and saved them to the tablet hard drive as a configuration file. Once the user starts the application, it automatically loads the calibration parameters to memory for real-time rectification. This reduces distortion caused by the optical lens and improves stereo matching results. Even though we obtained rectified image pairs, we still need to synchronize the sensors since we cannot rectify over time for dynamic scenes. The main mechanism for synchronizing multiple sensors in FCam API is by extending the basic *sensor* component to a *sensor array* [130]. A new abstract class called *SynchronizationObject* is also derived from the *Device* class with members *release* and *wait* for software synchronization. When the request queue for the camera sensors is being processed, if a *wait* is found and a certain condition is not satisfied, the *sensor* will halt until this condition is satisfied. On the other hand, if a *release* is found and the condition is satisfied, the halted *sensor* will be allowed to proceed. The FCam API also provides classes such as *Fence*, *MultiSensor*, *MultiShot*, *MultiImage* and *MultiFrame* for the user to control the stereo sensor with desired request parameters.

In our application, we use the rear main camera to continuously detect the best focusing position and send updates to the other sensor. Firstly, we ask the rear main lens to start sweeping the lens. We then get each frame with its focusing location. Next, we sum up all the values of the sharpness map attached to the frame and send updates to the autofocus function. The autofocus routine will move the lens in a relatively slower speed to refine

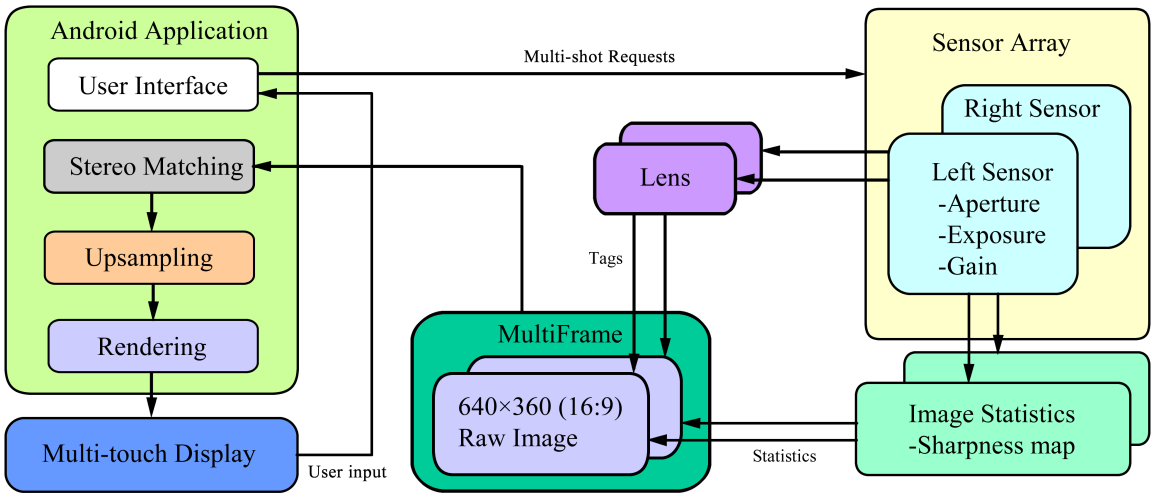


Figure 5.1: This diagram shows how our application interacts with the camera system. Our application accepts user input from the multi-touch screen, sends multi-shot requests to the sensors with desired parameters and then transfers the raw stereo image pairs to the stereo matching module. We then upsample the low-resolution disparity map and synthesize a light field image array. Finally, we render DoF effects on the screen of the tablet. We compute the best focal plane by using image statistics information tagged with the raw image frame.

the best focal depth. Once this process is done, we trigger a stereo capture with identical aperture, exposure and gain parameters for both sensors. The overall image quality is satisfactory, considering the fact that the size of the sensor is very small and the cost is much lower than research stereo camera systems such as Pointgreys Bumblebee. Figure 5.1 shows how our software system interacts with the imaging hardware.

5.5 Disparity Map Generation

Computing depth information from stereo camera systems is one of the core problems in computer vision. Stereo algorithms based on local correspondences [54, 35] are usually fast, but introduces inaccurate boundaries or even bleeding artifacts. Global stereo estimation methods such as Graph Cuts (GC) [16] and Belief Propagation (BP) [122] have shown good results on complex scenes with occlusions, textureless regions and large depth changes [111]. However, running these algorithms on full-resolution (1 mega pixel) image

pairs is still expensive and hence impractical for mobile devices. Therefore, we first downsample the raw input image pair and recover a low-resolution disparity map via GC. Next, we take each raw color image as the guidance image and upsample the disparity map via joint bilateral upsampling [68].

5.5.1 Graph Cuts Stereo Matching

In order to efficiently generate a high-resolution disparity map with detailed information about the scene, we propose a two-step approach. We first recover a low resolution disparity map on downsampled image pairs with the size of 160×90. Given the low resolution image pairs, the goal is to find labeling of pixels indicating their disparities. Suppose $f(p)$ is the label of pixel p; $D_p(x)$ is the data term, reflecting how well pixel p fits its counterpart pixel p shifted by x in the other image; $V_{p,q}(y,z)$ is the smoothness term indicating the penalty of assigning disparity y to pixel p and disparity z to pixel q; N is the set of neighboring pixels, the correspondence problem can be formulated as minimizing the following energy function:

$$E(f) = arg\min_{f} \left(\sum_{p \in P} D_p\left(f\left(p\right)\right) + \sum_{\{p,q\} \in N} V_{p,q}\left(f\left(p\right), f\left(q\right)\right) \right) \tag{5.1}$$

The local minimization of function (1) can be efficiently approximated using the alpha-expansion presented in [16]. In our implementation, we set the number of disparities to be 16 and run the algorithm for 5 iterations. If the algorithm cannot find a valid alpha expansion that decreases the overall energy function value, it may also terminate in less than 5 iterations. The performance of GC on the Tegra 3 tablet platform can be found at Table 5.1.

To evaluate our scheme, we performed experiments on various stereo image datasets. The stereo matching methods used here are block matching (BM), semi-global block matching (SGBM) [54], efficient large-scale stereo (ELAS) [35] and Graph Cuts (GC) [16]. Table 5.1 shows the running time of these algorithms on the Tegra 3 tablet and Figure 5.2 shows the calculated disparity map results. According to our experiments, BM is faster than SGBM and ELAS on any given dataset, but requires an adequate choice of the window size. Smaller window sizes may lead to a larger bad pixel percentage while bigger window sizes may

Table 5.1: Comparing running time (ms) of different stereo matching methods on the Tegra 3 tablet, using the Middlebury Cones dataset. The longer edge is set to 160 pixels and the number of disparities is set to 16.

Data sets	BM	SGBM	ELAS	GC
Tsukuba	15	28	51	189
Venus	13	30	97	234
Cones	19	42	124	321

cause inaccuracy problems on the boundary. Besides, the overall accuracy of disparity values generated by BM is not very high. As can be seen from Figure 5.2, we can still identify the curved surface area of the cones from results generated by SGBM and ELAS, but the same area looks almost flat in BM. SGBM and ELAS are two very popular stereo matching algorithms with near real-time performance. According to our experiments on the tablet, they are very similar to each other in terms of running time and accuracy. From Table 5.1 and Figure 5.2 we can see that ELAS generates better boundaries than SGBM on the cones dataset, but takes more processing time and produces more border bleeding artifacts. The GC gives smooth transitions between regions of different disparity values. According to Table 5.2, the GC algorithm outperforms other algorithms in most of the quality metrics on the Middlebury datasets. For our application, since the quality of upsampled result is highly dependent on the precision of boundary values in low resolution disparity maps, we choose to use GC rather than other methods which runs faster. Another reason is that we are running the GC algorithm on low-resolution imagery. According to Table 5.1, the running time is around 250 ms, which is still acceptable compared to ELAS (around 100 ms). In return, noisy and invalid object boundaries are well optimized and the resulting disparity map is ideal for refinement filters such as joint bilateral upsampling.

Table 5.2: Evaluation of different stereo matching methods on the Middlebury stereo datasets cite in bad pixel percentage (%). The method shown in the last row applies 5 iterations of Joint Bilateral Upsampling to the downsampled results (half of the original size) of GC, using the full resolution color image as guidance image. The resolutions of the four datasets (Tsukuba, Venus, Teddy, Cones) are 384×288, 434×383, 450×375, 450×375, respectively. If not specified, raw image size of each individual dataset will be the same for the remainder of this paper. Nonocc: bad pixel percentage in non-occluded regions; All: bad pixel percentage in all regions; Disc: bad pixel percentage in regions near depth discontinuities.

.

	Tsukuba			Venus			Teddy			Cones		
	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
BM	10.3	11.9	21.5	12.4	13.9	21.6	16.7	23.1	27.3	7.46	17.2	23.8
SGBM	3.26	3.96	12.8	1.00	1.57	11.3	6.02	12.2	16.3	3.06	9.75	8.90
ELAS	3.96	5.42	17.9	1.82	2.78	20.9	7.92	14.5	22.8	6.81	14.9	17.2
GC	1.94	4.12	9.39	1.79	3.44	8.75	16.5	25.0	24.9	7.70	18.2	15.3
Proposed	1.01	2.83	5.42	0.18	0.59	1.99	6.57	11.2	15.1	3.06	9.70	8.92

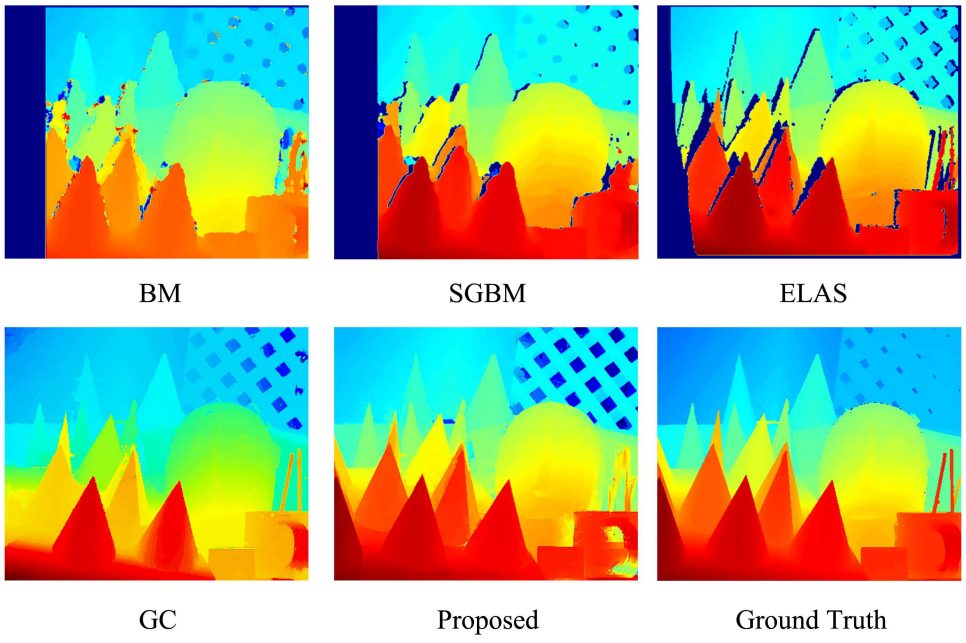


Figure 5.2: Comparison of our approach and other popular stereo matching algorithms.

5.5.2 Joint Bilateral Upsampling

Because the stereo matching process is performed on low resolution stereo image pairs, the result disparity map cannot be directly used for DoF synthesis. We need to upsample the disparity map while keeping important edge information.

Bilateral filtering proposed by C. Tomasi *et al.* [128] is a simple, non-iterative scheme for edge preserving smoothing which uses both a spatial kernel and a range kernel. However, for low signal-to-noise ratio (SNR) images, this algorithm cannot keep the edge information very well. A variant called joint bilateral filter introduced by J. Kopf [68] *et al.* addresses this problem by adding the original RGB image as a guidance image. More formally, let p and q be two pixels on the full resolution color image I; $p_{\downarrow}$ and $q_{\downarrow}$ denote the corresponding coordinates in the low resolution disparity map $D^{'}$; f is the spatial filter kernel, g is the range filter kernel, W is the spatial support of kernel f, and K_p is the normalizing factor. The upsampled solution D_p can be obtained as:

$$D_p = \frac{1}{K_p} \sum_{q_{\downarrow} \in W} D^{'}_{q\downarrow} f\left(\|p_{\downarrow} - q_{\downarrow}\|\right) g\left(\|I_p - I_q\|\right) \tag{5.2}$$

This method uses RGB values from the color image to create the range filter kernel and combines high frequency components from the color image and low frequency components from the disparity map. As a result, color edges are integrated with depth edges in the final upsampled disparity map. Since depth discontinuities typically correspond with color edges, this method can remove small noises. However, it may bring some unwanted effects. Firstly, blurring and aliasing effects caused by the optical lens are transferred to the disparity map. Besides, the filtering process may change disparity values in occlusion boundaries according to high frequency components in the color image, and thus causing the disparity map to be inaccurate. We address this problem by iteratively refining the disparity map after the upsampling process is done. As a result, the output image of the previous stage becomes the input of the next stage.

Figure 5.3 shows results after different number of iterations. The initial disparity map (see Fig. 5.3 (a)) is noisy and inaccurate because it is generated on low resolution image pairs. However, if too many iterations are applied to the input image (Fig. 5.3 (d)), the boundaries of the cup handle starts to bleed into the background, which is a result of

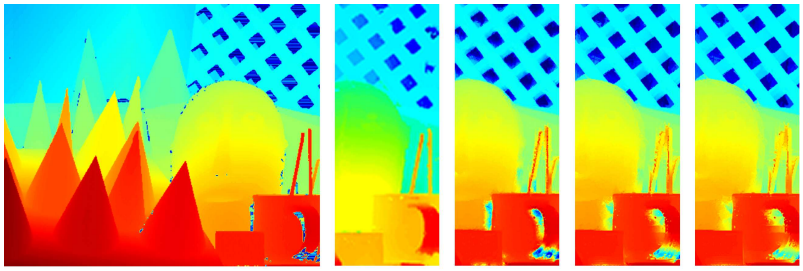


Figure 5.3: Comparison of results using different number of iterations. (a),(b),(c),(d) are using 0,5,10,20 iterations respectively.

over-smoothing. Also, more iterations add to the complexity and processing overhead of the entire application. According to Figure 5.4, the quality of the disparity map can be improved during the first 5 or 6 iterations. This is because Joint Bilateral Upsampling can preserve edges while removing noises in the disparity map. However, if the refining process contains too many iterations, then the disparities of one side of edges starts to bleed into the other side, causing the bad pixel percentage to go up, especially in regions near depth discontinuities (refer to the increase of disc values in Figure 5.4). Therefore, a compromise number of iterations must be chosen. In our application, the number is set to 5. Since the Middlebury datasets contain both simple scenes like Venus and complex scenes such as Teddy and Cones, we assume that 5 iterations should return good results under a variety of scene structures. Generally, it takes around 40 milliseconds to finish the 5 iteration steps on the tablet. Figure 5.5 illustrates the detailed view of our result compared to other standard upsampling methods. Because DoF effects are most apparent around depth edges, it is very important to recover detailed boundaries in the high resolution disparity map. According to Table 5.3, our method outperforms other methods in all quality metrics and generates better boundary regions (refer to disc values in Table 5.3) by using the fine details from the high resolution color image.

Table 5.3: Evaluation of various upsampling methods on the Middlebury stereo datasets in bad pixel percentage (%). We run these methods on downsampled ground truth data (half of the original size), and then try to recover the disparity maps at original size and measure the error percentage. Nonocc: bad pixel percentage in non-occluded regions; All: bad pixel percentage in all regions; Disc: bad pixel percentage in regions near depth discontinuities.

	Tsukuba			Venus			Teddy			Cones		
	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc	nonocc	all	disc
Nearest Neighbor	5.55	6.65	18.3	0.47	1.02	6.56	8.65	9.77	28.2	7.98	9.62	23.7
Bicubic	4.97	5.69	18.7	0.67	0.93	9.32	4.89	5.61	17.8	6.81	7.59	20.6
Bilateral	4.59	5.04	10.8	0.41	0.60	5.75	4.52	5.12	16.3	6.85	8.41	20.5
Proposed	3.08	3.34	7.54	0.25	0.33	3.47	2.41	2.89	8.76	3.45	3.96	10.5

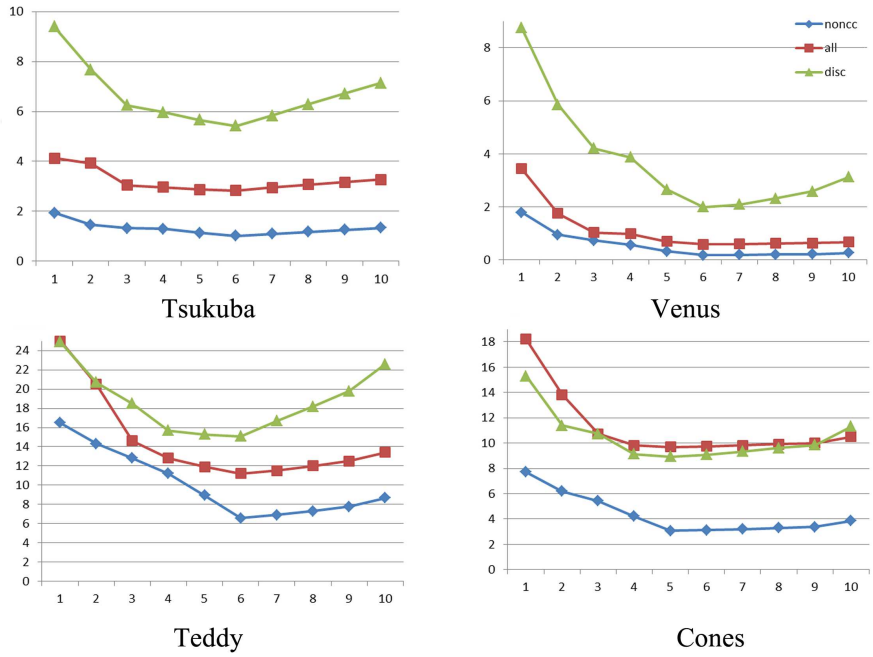


Figure 5.4: Evaluation of the disparity maps using different number of Joint Bilateral Upsampling iterations on the Middlebury stereo dataset. The horizontal axis shows the number of iterations and the vertical axis shows the bad pixel percentage.

5.6 Depth of Field Rendering

Once we obtained the high resolution disparity map, we can proceed to synthesize dynamic DoF effects. Previous studies suggested that real time DoF effects can be obtained

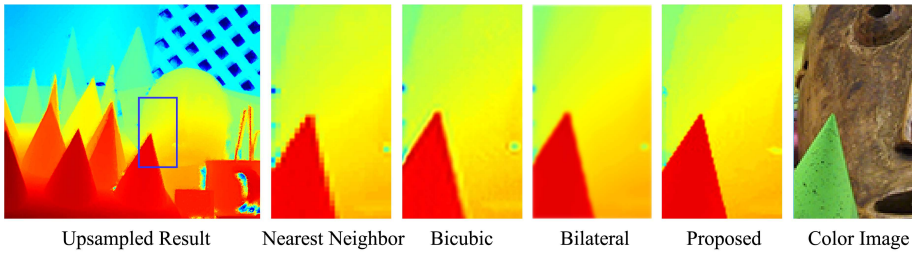


Figure 5.5: Comparison of our approach and other upsampling algorithms on the Middlebury cones dataset.

by applying a spatially varying blur on the color image and use the disparity value to determine the size of the blur kernel [73, 74]. However, this method suffers from strong intensity leakage and boundary bleeding artifacts. Other methods such as distributed ray tracing [25] and accumulation buffer [42] give more accurate results. However, these methods are computationally expensive and therefore can only provide a limited frame rate.

5.6.1 Synthesized Light Field Generation

In this paper, we use a similar approach to [157] by generating a synthetic light field on the fly. The main idea is to get the light field image array by warping the raw color image to nearby viewpoints according to corresponding values in the upsampled high resolution disparity map. The light field array can then be used to represent rays in the scene. Each ray in the light field can be indexed by an integer 4-tuple (s, t, u, v), where (s, t) is the image index and (u, v) is the pixel index within an image. Next, we set the rear main camera as the reference camera and use the high resolution color image and disparity map for reference view R_{00}. We then compute all rays passing through a spatial point X with shifted disparity γ from the reference view. Suppose X is projected to pixel (u_0, v_0) in the reference camera, we can compute its image pixel coordinate in any other light field camera view R_{st} as:

$$(u, v) = (u_0, v_0) + (s, t) \cdot \gamma \tag{5.3}$$

However, this algorithm may introduce holes in warped views, and this artifact becomes more severe when the synthesized baseline increases. To resolve this issue, we start from the boundary of the holes and iteratively take nearby pixels to fill the holes. Note that this module is only used for generating pleasing individual views for the user to interactively

shift the perspective. In the final rendering process, missing rays are simply discarded and the filled pixels are not used. Figure 5.6 shows warped views of an indoor scene using the aforementioned warping and hole-filling algorithms.

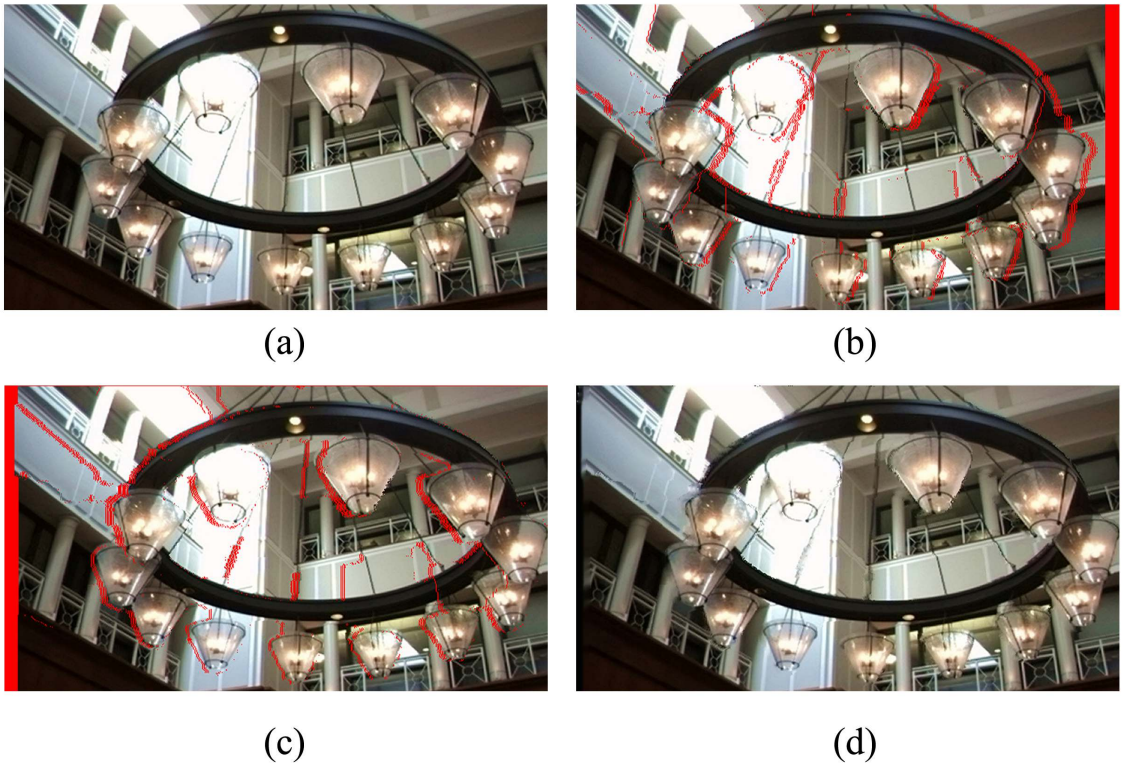


Figure 5.6: Synthesized light field view, missing pixels are marked in red. (a) input image (b) warped left side view (c) warped right side view (d) result image using our hole-filing algorithm, taking (c) as the input.

Since the image formed by a thin lens is proportional to the irradiance at pixel a [121], if we use $L_{out}(s,t,u,v)$ to represent out-of-lens light field and $L_{in}(s,t,u,v)$ to represent in-lens light field, the pixels in this image can be obtained as a weighted integral of all incoming radiance through the lens:

$$a(x,y) \simeq \sum_{(s,t)} L_{in}(s,t,u,v) \cdot \cos^4 \phi \tag{5.4}$$

To compute the out-of-lens light field, we simply remap the pixel $a(x,y)$ to pixel $(u_0,v_0) = (w-x, h-y)$ in the reference view R_{00}. Therefore, we can focus at any scene

depth with corresponding disparity γ_f by finding the pixel index in camera R_{st} using equation 5.3. Since the direction for each ray is $(s, t, 1)$, we can approximate the attenuation term $\cos^4 \phi$ as $\frac{1}{(s^2+t^2+1)^2}$, and the irradiance at a can be calculated as:

$$a(x, y) \simeq \sum_{(s,t)} \frac{L_{out}(s, t, u_0 + s \cdot \gamma_f, v_0 + t \cdot \gamma_f)}{(s^2 + t^2 + 1)^2} \tag{5.5}$$

Figure 5.7 shows details of the rendered image by using different sizes of the synthesized Light Field array. Since aliasing artifacts are related to scene depth and sampling frequency [20], we can reduce aliasing in the rendered image by increasing the size of the synthesized Light Field array.

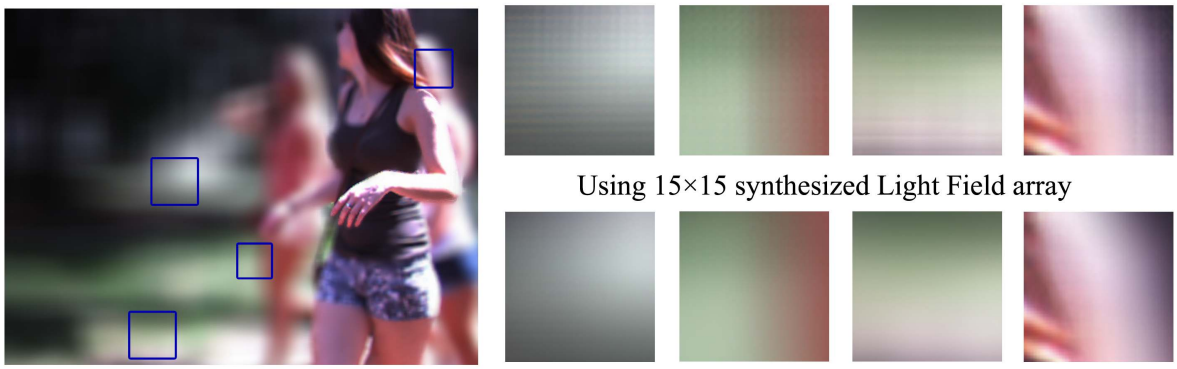


Figure 5.7: Comparing rendering results with different sizes of the synthesized Light Field array.

5.6.2 Comparison of our method of single-image blurring

Reducing boundary artifacts is very important as DoF effects are apparent near the occlusion boundaries. Comparing with single-image blurring methods [73, 74], our light field based analysis is good at reducing two types of boundary artifacts: the *boundary discontinuity* and *intensity leakage* artifacts. We summarize four types of boundary artifacts and analyze them separately. A detailed illustration of the four cases can be found at Fig. 5.8. In practice, the four cases can occur at the same time within a single scene.

Our analysis is based on the real world scene shown in Fig. 5.8. Consider a woman in a black dress walking in front of a white building. When we conduct the DoF analysis,

the camera is either focused at the foreground (the woman) or the background (the building). For Fig. 5.8 (a, b), we assume the camera to be focused at the background and for Fig. 5.8 (c, d), we assume that the camera is focused at the foreground. For each case, a comparison of results using different methods is shown at the right side of the images.

Now consider the first two cases shown in Fig. 5.8 (a, b). Suppose P_b is a point on the background building and its image I_b in the camera is right next to the foreground as shown in Fig. 9 (a). The ground truth result should blend both the foreground and background points for calculating I_b to make the transition natural and smooth. However, single image blurring methods would consider P_b in focus and directly use its color as the value of I_b. This will result in a *boundary discontinuity* artifact because of the abrupt jump between foreground and background pixel values. Our method, however, takes advantage of the synthesized light field and attempts to use rays originating from both the foreground and background to calculate pixel value of I_b, and hence generates correct results for this scenario. Similarly, for a foreground point P_f shown in Fig. 5.8 (b), the ground truth result should blend its neighboring foreground pixels and a single in-focus background point. Single-image blurring methods will use a large kernel to blend a group of foreground and background pixels, producing the *intensity leakage* artifact. In contrast, our method only takes rays needed to get the value of P_f and is free of intensity leakage artifacts. However, due to occlusion, some background pixels may be missing. In this case, our method will blend foreground rays and accessible background rays together. Since the missing rays only occupy a small portion of all background rays, our method produces reasonable approximations.

For the other two cases (Fig. 5.8 (c,d)), assume that the camera is focused at the foreground. As shown in Fig. 5.8 (c), the ground truth result should only blend background pixels. However, because of the blur kernel, the single-image blurring method blends both foreground and background pixels and thus causing intensity leakage problems. Our method, on the other hand, only attempts to blend background rays. Similar to the previous case, some rays are occluded by the foreground. We simply discard these rays and by blending existing rays together, we are able to reach reasonable approximations of the ground truth. For the last case, consider a point P_f on the foreground shown in Fig.5.8 (d). Since this pixel is

considered to be in focus, the single image blurring method will directly use its color and produce the correct result. Our method collects all rays coming from P_f and these rays are all accessible. Therefore, our method is also able to get the correct result.

Fig. 5.9 shows results of our method and single image blurring on an outdoor scene. As mentioned before, our method reduces artifacts on boundary regions compared to single image blurring approaches. In fact, our method will not cause any intensity leakage problems. When examining the single image blurring result (Fig. 5.9 (a)), it is very easy to find intensity leakage artifacts along the boundary, whereas our technique prevents such leakage (Fig. 5.9 (c)). Also, our method provides smooth transitions from the handbag strips to the background (Fig. 5.9 (d)) while single image blurring method exhibits multiple discontinuous jumps in intensity values.

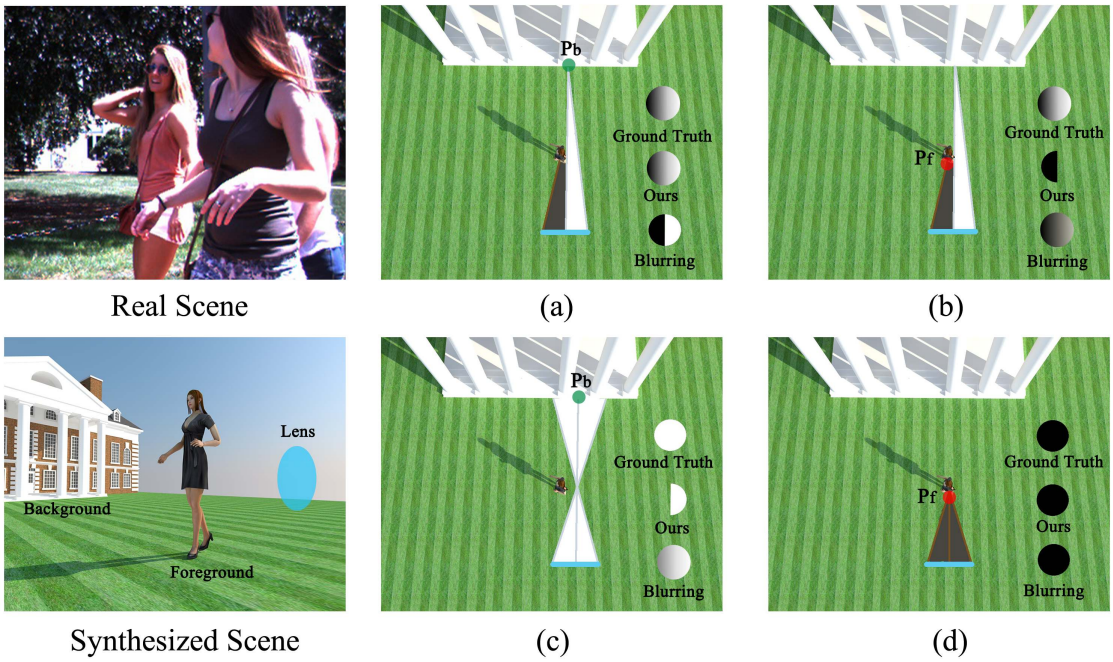


Figure 5.8: Causes of different boundary artifacts. See Section 6.2 for details.

5.7 Results and Analysis

We conducted extensive experiments on both indoor and outdoor scenes. Figure 5.10 and 5.11 shows the results generated by our system under different scene structures and illumination conditions. Scene 1 and 2 demonstrate our system's ability of handling real-time dynamic scenes; Scene 3 shows the result on an outdoor scene with strong illumination and shadows; Scene 4 displays the result on an indoor scene with transparent and textureless regions.

The processing speed of different frames vary from less than a second to several hundred seconds, depending on parameters such as number of stereo matching iterations, number of bilateral upsampling iterations and the size of the synthesized light field array. The user can keep taking pictures while the processing takes place in the background. Considering the performance of current mobile device processors, rendering real-time DoF effects on HD video streams is still not practical. However, this does not prevent users from taking consecutive video frames and render them offline, as can be seen in scene 1 and 2 of Figure

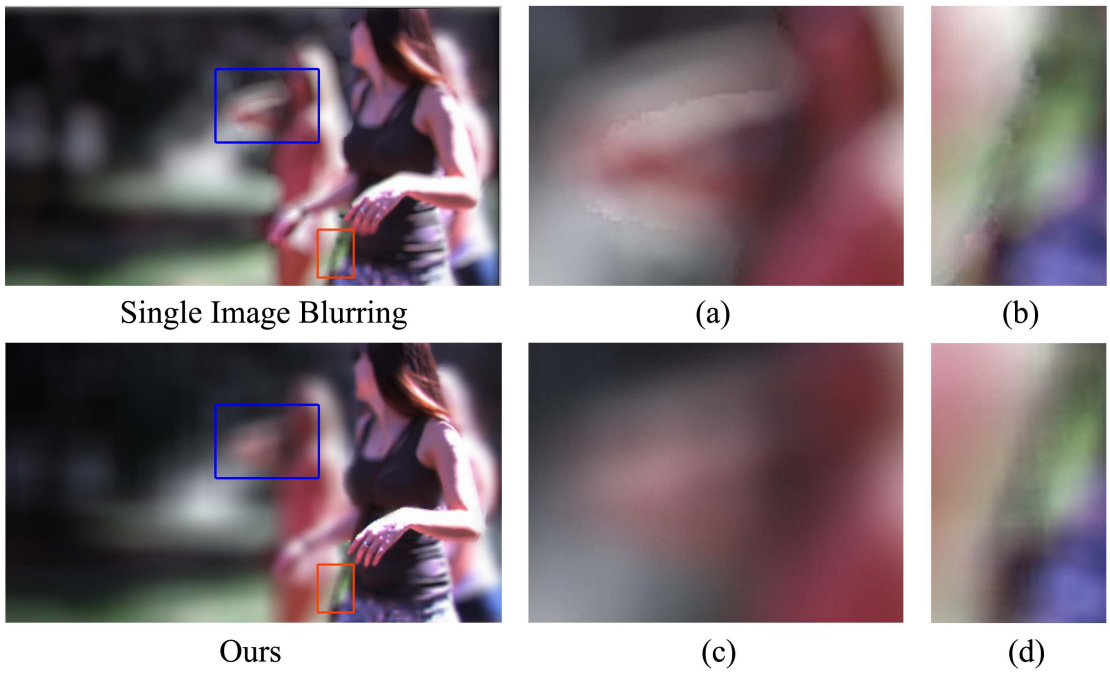


Figure 5.9: Comparison between our method and single image blurring. Single image blurring methods suffer from intensity leakage (a) and boundary discontinuity (b) artifacts. Our method (c,d) reduces these artifacts.

5.10. Also, since in general the stereo cameras on mobile devices have a small baseline, the disparity values of pixels in the downsampled images have certain max/min thresholds. We can reduce the number of disparity labels in the Graph Cuts algorithm and further improve processing speed without introducing much performance penalty.

We first demonstrate our system in dynamic outdoor scenes. Figure 5.10 shows results of two frames from the same video sequence. Since we currently do not have any auto-exposure or High Dynamic Range (HDR) modules implemented, some parts of the photo are over-exposed. As shown in the photograph, many texture details are lost in the over-exposed regions, making it challenging for the stereo matching algorithm to recover accurate disparity values. Moreover, the background lawn contains noticeable shadows and large portions of the building wall are textureless. This adds to the difficulty of finding pixel to pixel correspondences. Notwithstanding, our algorithm generates visually good-looking disparity maps. The edges of the woman's hand and arm are preserved when they are in

focus and objects outside of the focal plane are blurred smoothly.

Figure 5.11 scene 3 displays a scene of two women walking in front of a parking lot. Typically the working range of the tablet sensor is from half a meter to five meters. As a result, the cars in the parking lot are already approaching the maximum working distance of the sensor. This, however, does not affect the overall refocusing result as the cars with similar disparity values are either all in focus (Fig. 5.11 row 2, column 2) or blurred (Fig. 5.11 row 2, column 1). The sidewalk in front of the parking lot has a lot of textureless areas, making it difficult to achieve coherent disparity values. As a result, the left and right part of the sidewalk are blurred slightly differently although they are on the same plane (Fig. 5.11 row 2, column 2). Also, because the women in scene 3 are farther away from the camera compared to the women in scene 1 and 2, the boundaries of women in scene 3 are coarser and fine details on the bodies are lost. Therefore, foregrounds in scene 3 are more uniformly blurred compared to scene 1 and 2.

Indoor scenes have controllable environments and undoubtedly aid the performance of our system. For example, most structures from an indoor scene are within the working range of our system and typically indoor lighting won't cause problems such as over-exposure or shadows. Scene 4 of Figure 5.11 shows results on an indoor scene with transparent objects and textureless regions. Since our algorithm effectively fills holes and corrects bad pixels on the disparity map by using the guide color image, the resulting disparity map looks clean and disparity edges of the chandelier are well preserved (Fig. 5.11 row 3, column 2). The upper left part of the wall surface is over-exposed and the light bulb in the foreground almost merged into the background. However, the disparity map still recovers edges correctly. As can be seen in Figure 5.11 row 4, column 2, the defocus blur fades correctly from the out-of-focus light bulb regions into the in-focus wall regions, despite the fact that they are both white and do not have clear boundaries in-between.

The discussion here is based on our own captured data and it is hard to evaluate rendered results because of the lack of ground truth. To address this problem, we conducted subjective rating tests with twenty people. Among these people, ten have a computer vision

or graphics background and the remaining have no expertise in the related field. For convenience and clarity, the rating is done on a 0-9 scale for measuring the quality of rendered results. We define the rating as follows: 0 (Not Acceptable), 1 (Acceptable), 3 (Good, but needs improvement), 5 (Satisfactory), 7 (Very good) and 9 (Excellent). The test results can be found at Table 5.4. The average rating of the non-expert group is 8.1 and the average rating from the experts is 5.3. Therefore, the overall quality of the rendered results can be concluded as satisfactory.

According to Table 5.2, our method returns the best disparity map results in terms of overall bad pixels percentage. Also, our system correctly handles complex scene structures with real world illumination conditions. Last but not least, according to result images in Figure 5.7, we reduce aliasing artifacts in out-of-focus regions by blending multiple synthesized light field views together.

Table 5.4: Results of subjective quality rating tests.

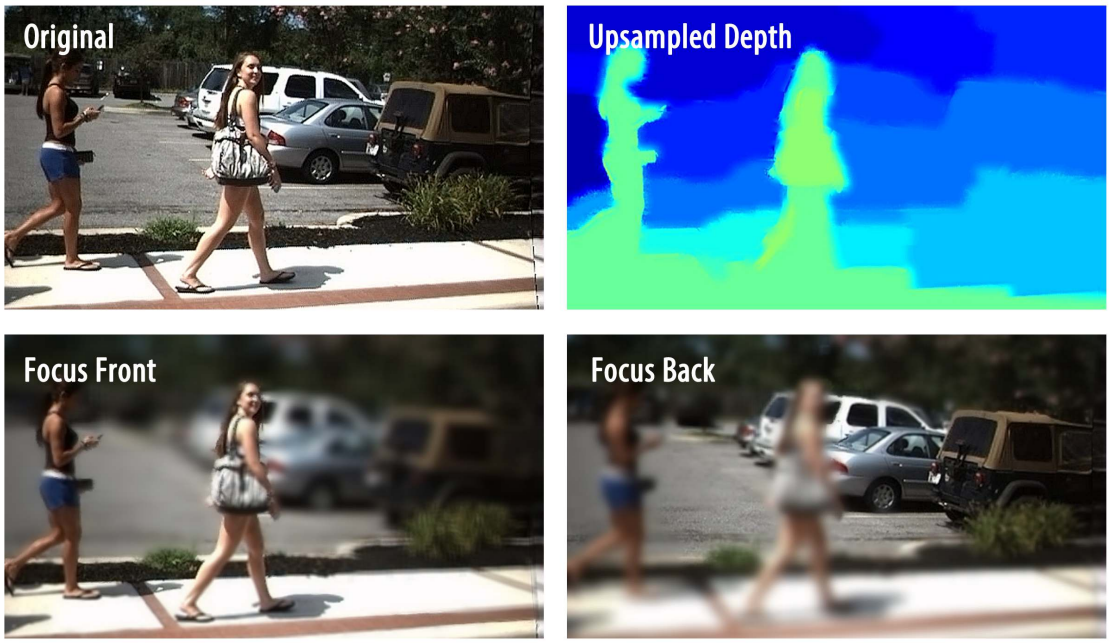
User ♯	1	2	3	4	5	6	7	8	9	10	Average
Non-experts	7	9	9	8	9	8	7	7	9	8	8.1
Experts	5	3	7	6	8	7	1	5	5	6	5.3

Finally, to demonstrate that our algorithm is also capable of generating high quality DoF effects using high resolution stereo input, we leverage mobile devices Fujifilm FinePix Real 3D camera to capture a set of stereo images and generate shallow DoF images with refocus capabilities at 6MP resolution, as shown in Figure 5.12 and 5.13. Current light field cameras are not capable of generating such high resolution images. Figure 5.12 shows the scene of a person playing with a skateboard. Our algorithm is able to preserve most of the depth discontinuities in the scene, such as the edges of the hand, the skateboard and the leg. Note that the background between the legs is marked as the foreground, leaving artifacts in the final rendering. This is due to the unsuccessful depth estimation of the Graph Cut algorithm and our current depth upsampling is largely relying on the initial estimation. In the future, we plan to employ depth error correction into our upsampling scheme. Figure 5.13 shows a scene of a sculpture in a shopping mall. Dispite the complex occlusion conditions

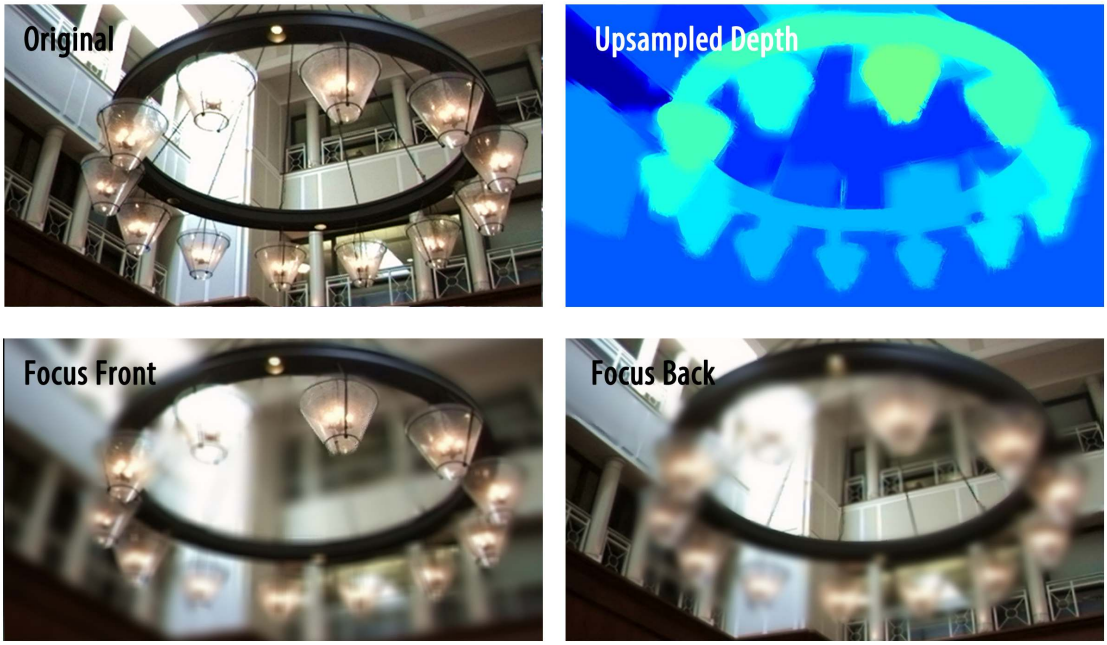
Scene 1

Scene 2

Figure 5.10: Input disparity map and rendered images of our system on two frames from the same stereo video sequence.



Scene 3



Scene 4

Figure 5.11: Input disparity map and rendered images of our system on two real scenes with the same arrangements as Figure 11.

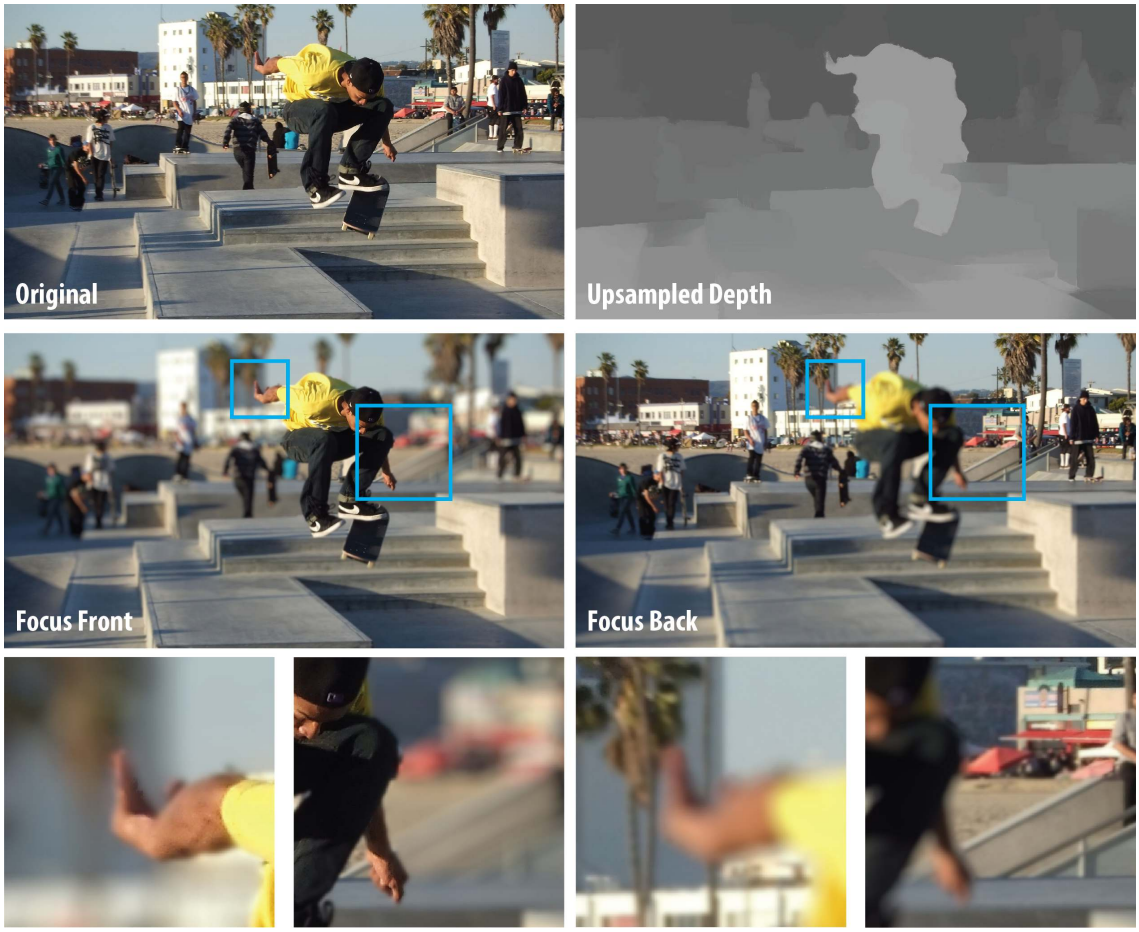


Figure 5.12: Our result on a skateboard scene at 6MP captured by Fujifilm FinePix Real 3D camera. Courtesy of Design-Design [27].

in the scene, our algorithm is still able to synthesize shallow DoF effects with little artifacts, such as fussy edges on the stairs.

5.8 Conclusion

In this chapter, we demonstrate that accurate, high-resolution depthmaps can be obtained using low-cost stereo vision sensors on mobile devices. We first capture stereo image pairs by using the FCam API, then apply the Graph Cuts stereo matching algorithm to obtain low-resolution disparity maps. Next, we take raw color images as guide images and

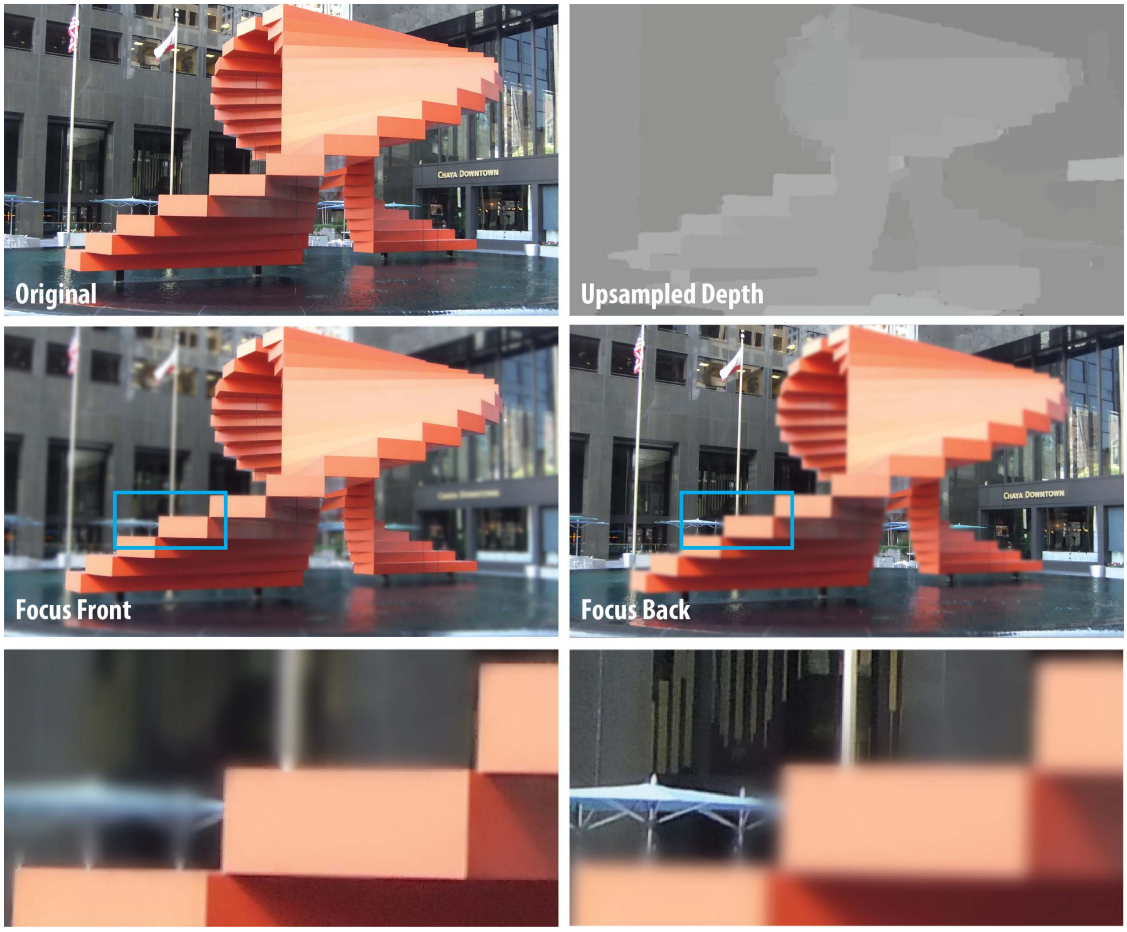


Figure 5.13: Our result on a sculpture scene at 6MP captured by Fujifilm FinePix Real 3D camera. Courtesy of Design-Design [27].

upsample the low-resolution disparity maps via joint bilateral upsampling. We evaluate a variety of real-time stereo matching and edge-preserving upsampling algorithms for the tablet platform. Experimental results show that our approach provides a good tradeoff between expected depth-recovering quality and running time. All the aforementioned processing algorithms are implemented to the Android operating system and tested on the Tegra 3 T30 prototype tablet. The user can easily install the software, capture and generate desired DoF effects using the tablet only, with no additional hardware or software required. The system has been tested in a variety of environments with satisfactory results. Also, we generate the synthesized light field by using a disparity warping scheme and render high quality DoF effects. Our system efficiently renders dynamic DoF effects with arbitrary aperture sizes and focal lengths in a variety of indoor and outdoor scenes.

Chapter 6

ADVERSARIAL LEARNING FOR 3D VISUAL OBJECT DETECTION FROM MONOCULAR IMAGES

In this chapter, we propose a novel approach to predict accurate 3D bounding box locations on monocular images. We first train a generative adversarial network (GAN) to perform monocular depth estimation. The ground truth training depth data is obtained via depth completion on LiDAR scans. Next, we combine both depth and appearance data into a birds-eye-view representation with height, density and grayscale intensity as the three feature channels. Finally, We train a convolutional neural network (CNN) on our feature map leveraging bounding boxes annotated on corresponding LiDAR scans. Experiments show that our method performs favorably against baselines.

6.1 Introduction

In the past few years, new types of LiDAR (Light Detection And Ranging) sensors have been developed for autonomous vehicles. These sensors provide an accurate 3D perception of the surrounding environment in real-time. As a result, several LiDAR-based classification, detection, and segmentation datasets are made available to public [33, 34]. LiDAR is popular and advantageous compared to traditional stereo or multi-camera ranging devices for a variety of reasons. Firstly, LiDAR is able to give accurate measurements invariant of the ego car distance, while camera based ranging algorithms typically give a degraded performance on distant objects. This is because the object size reduces quadratically with distance to the camera for most imaging sensors. Secondly, LiDAR is an active time-of-flight (ToF) sensing device which works on a variety of objects including specular/metallic surfaces and textureless regions. Also, depending on the wavelength, LiDAR devices have certain levels of see-through capability on transparent objects (e.g. cloud, rain, snow). On

the contrary, computer vision algorithms operating on camera sensors will start to fail when reflective/textureless/transparent regions increase. Finally, most LiDAR devices give 360-degree surrounding scans and immediate reading of orientation and distance to the object, whereas camera sensors usually have limited field-of-view (FOV) and multi-camera calibration issues, plus additional computation overheads to produce depth maps from raw input images. In the 2007 DARPA Urban Challenge, a team [96] finished in the second place using LiDAR alone with no camera sensors involved. Despite its advantages, there are a few major drawbacks of LiDAR sensors. Firstly, they are typically bulky and expensive for wide use and deployment. Secondly, even top-of-the-line LiDAR sensors only provide 64 or 128 sparse scanlines across the 3D space, while camera sensors operate at a much higher resolution (typically ranging from 5 to 20 megapixels). Finally, LiDAR signals are inherently limited to spatial information and do not provide what cameras can typically see, such as words on the traffic sign, color, and pattern of the vehicle, etc. Therefore, it is still important to keep the camera sensors as a supplementary/fall-back option.

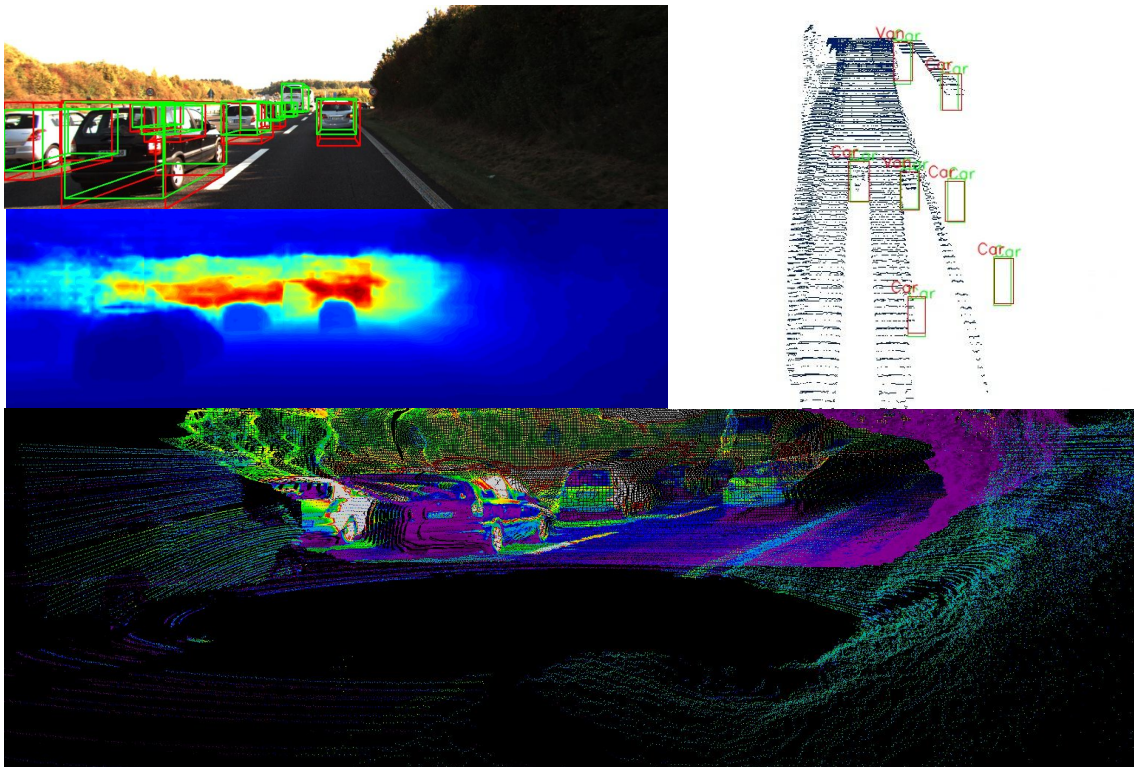


Figure 6.1: Sample output and intermediate results from our pipeline (Best viewed electronically). Top left: Our predicted 3D bounding boxes (red) *vs.* ground-truth annotations on the KITTI dataset (green). Top middle: predicted depth map. Top right: 2D detection results on our depth map projected to the birds-eye-view (BEV) map. Bottom: Our transformed point cloud aligned with LiDAR scanlines. The intensity values on our point cloud are calculated using grayscale intensity values from the input RGB image.

Intuitively, depth data provide more useful descriptions of spatial information, while appearance data provide more visual cues to identify objects into different categories. Therefore, when combining semantically rich appearance data with depth information, one can improve the performance of both locating and categorizing objects in an image. Early research attempts to combine simple depth cues with image features for richer representation [117]. However, due to difficulty in propagating gradients in the model, simply stacking features from different modalities could not give satisfactory performance. Gupta *et al.* proposed to use horizontal disparity, height above ground, and angle with the direction of gravity to form another 3 channel image for training [41]. Due to difficulty on training these type of feature maps, it is a common practice to finetune on existing models trained on RGB images [118].

However, it is questionable whether this way of inter-model fusion is reasonable as depth features seldom resemble shape, color, and appearance from the visible light spectrum. Lenz *et al.* proposes to learn features from RGB and depth images separately and then fuse at a higher level [76]. This method is termed *Late Fusion* by Eitel *et al.* [29]. Most work on 3D detection, on the other hand, are either purely based on LiDAR data [10, 116, 104] or simply use visual cues to supplement LiDAR data [103, 85].

In this chapter, we propose a novel approach to leverage both depth and visual cues for 3D object detection on monocular images. At the core of our technique is to integrate appearance and structural cues for better object detection. Our method contains three stages. Firstly, we use an unpaired image to image translation network to learn bi-directional transformations from RGB images to depth maps. Secondly, we calculate height, density and grayscale intensity as 3 feature channels and project the feature map to a birds-eye-view representation. Finally, we take advantage of 3D bounding box annotations on LiDAR data and train our object detection model on the feature map. The rest of this chapter is organized as follows. In Section 6.2 we discuss related work. In Section 6.3 we demonstrate different components of the proposed method. We show experimental results and analysis in Section 6.4, 6.5 and draw conclusions in Section 6.6.

6.2 Related Work

Dual Learning The idea of using forward and backward consistency to improve training has a long history [14]. Recently, He *et al.* [45] proposed the concept of dual learning to improve the performance of machine translation systems. The proposed mechanism can be viewed as a two-agent communication game. The two agents may not be able to translate one language to another, but are still able to evaluate and collectively improve the quality of the two translation models by going through the full forward-backward translation cycle. This procedure can be performed by an arbitrary number of rounds until the two models are fully converged. This idea inspired conditional GANs based cross-domain translation tasks [170, 156] and improves performance in image-based depth/shape estimation tasks. [39, 169]. We choose to use GAN for depth prediction because it allows unpaired image

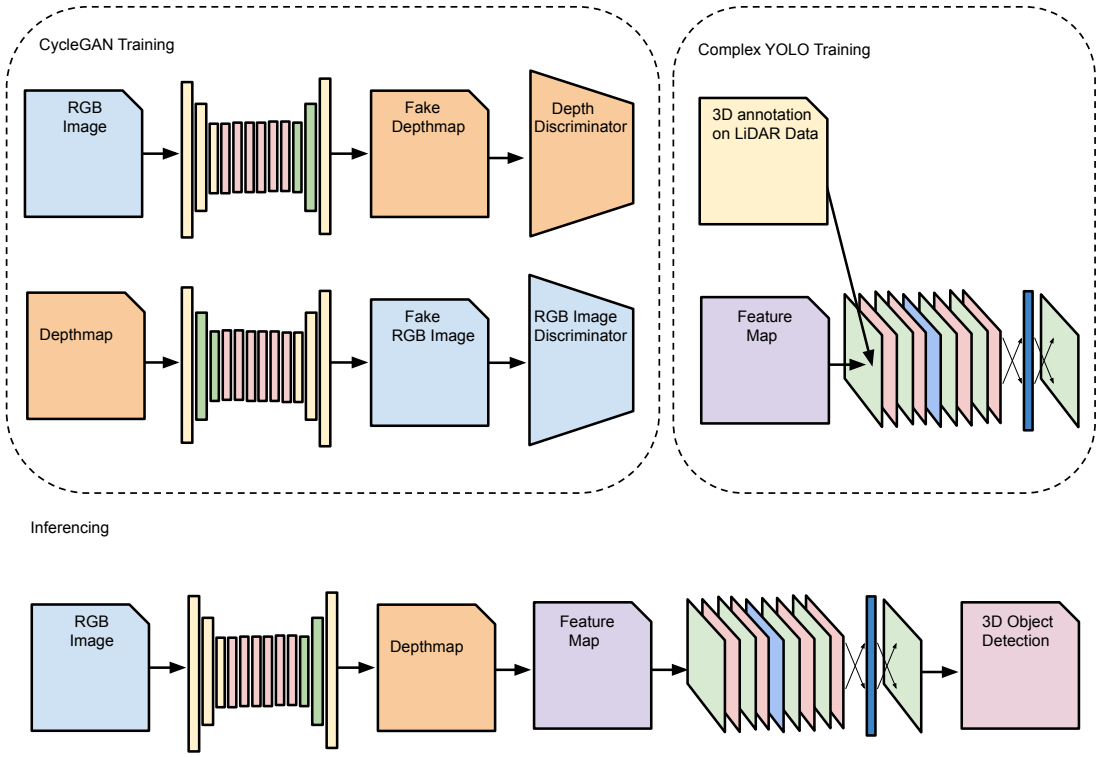


Figure 6.2: Architecture of our network. Top left: our CycleGAN based depth prediction network. Top right: our 3D detection network based on Complex YOLO. Bottom: our network for inference. Note that for training our method requires both monocular images and aligned LiDAR scans. However, for inference we only need monocular images to predict 3D object locations and categories. See Section. 6.3 for details.

domain transfer, while other depth prediction models are usually dependent on LiDAR input guidance.

Image-to-Image Translation The idea of learning from a pair of images and then apply the model at inference time to produce an analogous target image from the input image dates back to [52]. More recently, Isola *et al.* [57] proposed a method which exploits conditional adversarial networks as a unified framework for image to image translation. It uses the L1 loss function to enforce generated synthetic images to be similar to ground truth training images while letting GANs to only hallucinate high-frequency details in the image. This is because the L1 loss can already guarantee similarity at low frequencies. Therefore, instead of processing the whole image, the discriminator only attempts to classify if a $N \times N$ image patch is correct or not. This method produces remarkable results on a variety of tasks, including photographs from sketches, automatic colonization of black and white images, raw images to label maps, thermal to visible light images, and so on.

3D Object Detection on LiDAR data Recent advance in sensor and computing technology enables 3D object detection on structural data. Due to the difficulty in processing large-scale point cloud data, most works preprocess the raw input data into either voxels or birds-eye-view maps (BEV). Chen *et al.* converts LiDAR data to a BEV representation for 3D object detection in the road scene [22]. Liang *et al.* develop a 3D object detector that reasons in BEV space and integrates visual cues by learning to project camera-based features into the BEV space [85]. YOLO3D [10] extends the 2D YOLOv2 object detector [107] to the BEV map and achieves real-time performance on the KITTI dataset. Complex-YOLO [116] also operates on the BEV map by running an E-RPN that estimates object orientations by both imaginary and real numbers.

3D Object Detection on RGB Images More recent publication [140] introduces the concept of *pseudo LiDAR*, arguing that by converting the image-based depth maps to a representation that closely mimics the LiDAR signal, one could obtain state-of-the-art results on stereo vision based 3D object detection. Our method is along the lines of performing 3D object

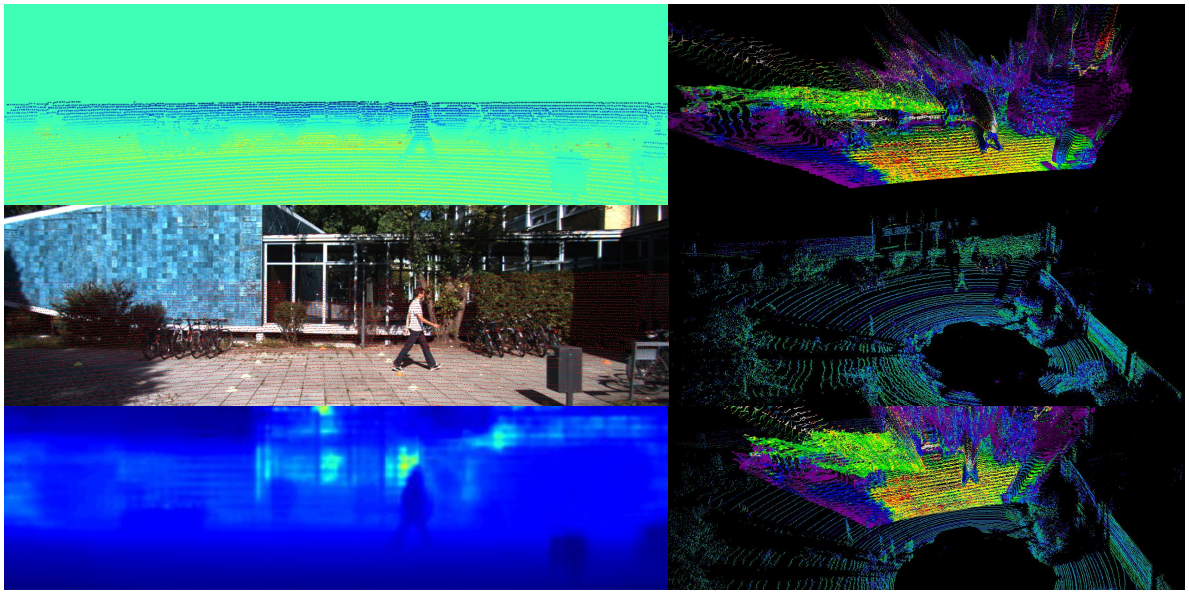


Figure 6.3: Bidirectional transforms between LiDAR and camera coordinates (Best viewed electronically). Top left: LiDAR scans provided by the KITTI dataset projected to the camera imaging plane, color-coded by depth. Middle left: LiDAR scans projected to the corresponding RGB image. Bottom left: predicted depth map with one-to-one mappings to the input image. Top right: predicted depth map transformed to the LiDAR coordinates, color-coded by one channel grayscale intensity. Middle right: LiDAR scan color-coded by intensity/reflectivity. Bottom right: our transformed point cloud aligned with the LiDAR scan. Note the LiDAR has a much wider field of view (FOV).

detection on RGB images. However, our method differs from the pseudo-LiDAR approach in a few aspects. Firstly, our detection is performed on a feature map consists of height, density, and grayscale intensity information. This feature map combines both depth and visual cues and is not intended to mimic the LiDAR signal. Secondly, our method leverages unpaired adversarial learning to predict the depth map, eliminating the need for collecting pairwise-aligned RGB and depth data, thus making it much easier to apply to use cases other than autonomous driving (*e.g.* indoor scenes, close-up scenes, top-down surveillance videos, etc.)

6.3 Approach

6.3.1 Depth Estimation

We adopt the CycleGAN [170] for depth estimation from monocular images. We use the sparse to dense [93] depth completion results on KITTI LiDAR scans as groudtruth for training. The learning objective contains 2 terms: an adversarial loss and a cycle consistency loss:

$$\begin{aligned} &\mathcal{L}(G, F, D_X, D_Y) = \\ &\quad \gamma_{adv}\mathcal{L}_{adv}(G, D_X, D_Y, X, Y) + \gamma_{cyc}\mathcal{L}_{cyc}(G, F) \end{aligned} \tag{6.1}$$

Where γ_{adv}, γ_{cyc} are the hyper-parameters to adjust loss on each term and are empirically set during the experiment. $\{x_i \in X\}_{i=1}^{N}$ and $\{y_i \in Y\}_{i=1}^{N}$ are N training images from the RGB dataset and depth dataset, respectively. G and F are mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$ to transform RGB images to depth maps or vice versa. D_X and D_Y are adversarial discriminators to distinguish between real images $\{x\}$ and synthetic images $\{F(y)\}$, or $\{y\}$ with $\{F(x)\}$. The cycle consistency loss is defined as:

$$\begin{aligned} \mathcal{L}_{cyc}(G, F) =& \mathbb{E}_x \sim p_{data}(x)[\|F(G(x)) - x\|] \\ &+ \mathbb{E}_y \sim p_{data}(y)[\|G(F(y)) - y\|] \end{aligned} \tag{6.2}$$

It can be viewed as translating an RGB image into a depth image and then translate it back to compare with the original using L1 norm. Based on the cycle consistency loss, two discriminators D_X and D_Y are introduced to calculate the adversarial loss [40]. This term enforces the distribution of translated images to be as close to the training images as possible. The adversarial loss is defined as:

$$\begin{aligned} &\mathcal{L}_{adv}(G, F, D_X, D_Y, X, Y) = \\ &= \mathbb{E}_y \sim p_{data}(y)[logD_Y(y)] + \mathbb{E}_x \sim p_{data}(x)[logD_X(x)] \\ &+ \mathbb{E}_x \sim p_{data}(x)[log(1 - D_Y G(x))] \\ &+ \mathbb{E}_y \sim p_{data}(y)[log(1 - D_X G(y))] \end{aligned} \tag{6.3}$$

By using the above two loss terms we aim to minimize adversarial discriminator errors of both visual and structural cues, as well as L1 error of predicted images *vs.* original images.

6.3.2 Feature Map Generation

Once we obtained the trained GAN model for depth prediction, we would like to transform the depth map from camera coordinate system to the LiDAR coordinate system for alignment with ground-truth bounding box annotations. In order to do this, we first transform the depth map to the rectified (rotated) camera coordinate system:

$$\begin{aligned} z_{rect} &= D(u, v) \\ x_{rect} &= \frac{(u - c_u) \times z_{rect}}{f_u} + b_x \\ y_{rect} &= \frac{(v - c_v) \times z_{rect}}{f_v} + b_y \end{aligned} \tag{6.4}$$

Where $(x_{rect}, y_{rect}, z_{rect})$ is the 3D point coordinate in the rectified camera coordinates. (u, v) denotes a pixel location in the predicted depth map. (c_u, c_v) is the pixel location corresponding to the imaging center, f_u, f_v are the horizontal and vertical focal length and b_x, b_y are the baselines with respect to reference camera. The camera intrinsic can be obtained from the projection matrix provided by [33]:

$$\mathbf{P}_{rect} = \begin{pmatrix} f_u & 0 & c_u & -f_u b_x \\ 0 & f_v & c_v & -f_v b_y \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{6.5}$$

Next, we transform the 3D point cloud from rectified camera coordinates to reference camera coordinates and then to the LiDAR coordinates by calling the KITTI utility library[2]. Let $\mathbf{T}_{cam}^{velo}$ be the 4×4 transformation matrix from the camera coordinate system to the LiDAR coordinates, $\mathbf{R}_{rect}$ be the 4×4 rectifying rotation matrix converted from Cartesian to homogeneous coordinates by adding a fourth zero row and setting $\mathbf{R}_{rect}(4, 4) = 1$, $\mathcal{P}_{velo}$ and

$\mathcal{P}_{rect} \in \mathbb{R}^3$ be the 3D point coordinates in the LiDAR and rectified camera coordinates, we can write the transformation as:

$$\mathcal{P}_{velo} = \mathbf{T}_{cam}^{velo} \, \mathbf{R}_{rect}^{-1} \, \mathcal{P}_{rect} \tag{6.6}$$

Note that we also store the RGB value and index of each point in another table to obtain the RGBXYZ representation of $\mathcal{P}_{velo}$. Next, we perform preprocessing in a fashion similar to Complex YOLO [116] to transform $\mathcal{P}_{velo}$ into the BEV feature map. The only difference between Complex YOLO and our method is that we are using grayscale intensity (visual) as the blue channel of the image while Complex YOLO sets the blue channel to LiDAR intensity (reflectivity). More formally, let $\mathcal{S}$ be the mapping function to map each point in $\mathcal{P}_{velo}$ to a grid cell $\mathcal{S}_{bev}$ [116], we can formulate the transformation as:

$$\begin{aligned} f_g\left(\mathcal{S}_{bev}^j\right) &= \max\left(\mathcal{P}_{velo \to bev}^i \cdot [0,0,1]^T\right) \\ f_b\left(\mathcal{S}_{bev}^j\right) &= \max\left(I\left(\mathcal{P}_{velo \to bev}^i\right)\right) \\ f_r\left(\mathcal{S}_{bev}^j\right) &= \min(1.0, \log(\left|\mathcal{P}_{velo \to bev}^i\right| + 1)/64) \end{aligned} \tag{6.7}$$

Where f is the resulting 3-channel feature map. f_g, f_b, f_r denotes height map, grayscale intensity map and density map, respectively. I is the grayscale intensity of $\mathcal{P}_{velo}$ calculated from the RGB values. $\mathcal{P}_{velo \to bev}$ denotes the 3D points mapped to the grid cell $\mathcal{S}_{bev}$. To this end, we have constructed the feature map which is aligned with LiDAR 3D object bounding box annotations ready for training. We visualize the height, density and intensity maps of both LiDAR data and predicted depth data in Fig. 6.5. We also show the alignment of LiDAR data *vs.* our transformed point cloud in Fig. 6.3. As can be seen from Fig. 6.3, the LiDAR scanlines are accurately projected onto the camera coordinates. Also, the transformed point cloud is well-aligned with LiDAR data. It is worth-noting that the field of view (FOV) of LiDAR is much larger than the camera. This is reflected in both Fig. 6.3 (row 1 column 2 *vs.* row 2 column 2) and Fig. 6.5 (first row *vs.* second row). Therefore, unlike other methods, during mAP evaluation we only compare with ground-truth bounding box annotations that falls within the camera FOV.

6.3.3 3D Object Detection

We follow the Complex YOLO model architecture put forth by [116] to train the 3D object detector. This detector takes the BEV feature map mentioned in Section. 6.3.2 as input, and extends the YOLOv2 detector [107] by a complex angle regression and a Euler region proposal networks (E-RPN). The E-RPN is a direct extension of the region proposal networks (RPN) proposed by Ren *et al.* [108]. Specifically, consider (x, y, w, l, ϕ) as a vector describing 2D locations, size and orientations of 3D objects in the BEV coordinates, the parameterizations of the 5 coordinates can be obtained as [108, 116]:

$$\begin{aligned} b_x &= \sigma\left(t_x\right) + c_x \\ b_y &= \sigma\left(t_x\right) + c_y \\ b_w &= p_w e^{t_w} \\ b_l &= p_l e^{t_l} \\ b_\phi &= \arg\left(|z| e^{i b_\phi}\right) = \arctan_2\left(t_{Im}, t_{Re}\right) \end{aligned} \tag{6.8}$$

The loss function of Complex YOLO is defined as a multi-part loss. The first part is the YOLOv2 loss [107] and the second part is an Euler regression loss:

$$\begin{aligned} L_{YOLO} &= \lambda_{coord} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} \left[\left(x_i - \hat{x}_i\right)^2 + \left(y_i - \hat{y}_i\right)^2\right] \\ &+ \lambda_{coord} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2\right. \\ &\left. + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2\right] + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \\ &+ \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} \left(C_i - \hat{C}_i\right)^2 \\ &+ \lambda_{noobj} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{noobj} \left(C_i - \hat{C}_i\right)^2 \\ &+ \Sigma_{i=0}^{S^2} 1_{i}^{obj} \Sigma_{c \in classes} \left(p_i(c) - \hat{p}_i(c)\right)^2 \\ L_{Euler} &= \lambda_{\text{coord}} \Sigma_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} \left| e^{i b_\phi} - e^{i \hat{b}_\phi} \right| \\ L_{\text{Total}} &= L_{\text{YOLO}} + L_{\text{Euler}} \\ L_{\text{Euler}} &= \lambda_{\text{coord}} \sum_{i=0}^{S^2} \Sigma_{j=0}^{B} 1_{ij}^{obj} \left| e^{i b_\phi} - e^{i \hat{b}_\phi} \right| \end{aligned} \tag{6.9, 6.10}$$

Method	Train	Test	Car	Tram	Truck	Van
MV3D [22]	LiDAR+RGB	LiDAR+RGB	86.02	N/A	N/A	N/A
C-YOLO [116]	LiDAR	LiDAR	85.89	N/A	N/A	N/A
Ours	LiDAR+RGB	RGB	78.78	58.70	80.00	74.14

Table 6.1: Quantitative mAP results. Note that we only evaluate within the visible range of the predicted depth map/point cloud, whereas all other methods evaluate on the full LiDAR scan. Also, our method and ComplexYOLO report scores on the random test split while MV3D evaluate on the test set. Both MV3D and ComplexYOLO scores are reported under the easy category of the BEV evaluation task. See Section. 6.4 for details.

According to the authors, the Euler loss leads to a closed-form space eliminating singularities. This leads to state-of-the-arts results on the KITTI 3D object detection dataset, while achieving real-time performance on the embedded NVIDIA TX2 platform [116].

6.4 Experiments

We use the KITTI 3D object detection benchmark suite for training and evaluation. The KITTI 3D object detection benchmark consists of 7481 training images as well as the corresponding point clouds, training labels and camera calibration files. We first use the supervised model [93] provided by the author to obtain depth maps for all training images. We subsequently perform a random train (60%) / validation (25%) / test (15%) split and use the code provided by the author [170] for training the CycleGAN model. The model is trained from scratch with random weight initialization. We set base learning rate = 0.0002, gamma = 0.5, momentum = 0.5. We train for 200 epochs and use the CycleGAN model for constructing BEV feature maps.

Next, we modify the Complex YOLO framework by constructing feature maps on-the-fly during training. We still use the same train/validation sets and construct the BEV feature map for every image. We run CycleGAN inference on every image to obtain the depth map, then follow the transformations outlined in Section. 6.3.2 to obtain the 3-channel feature map as input for the Complex YOLO framework. Our implementation is based on the open source code provided by [1]. The ground-truth labels are obtained by converting

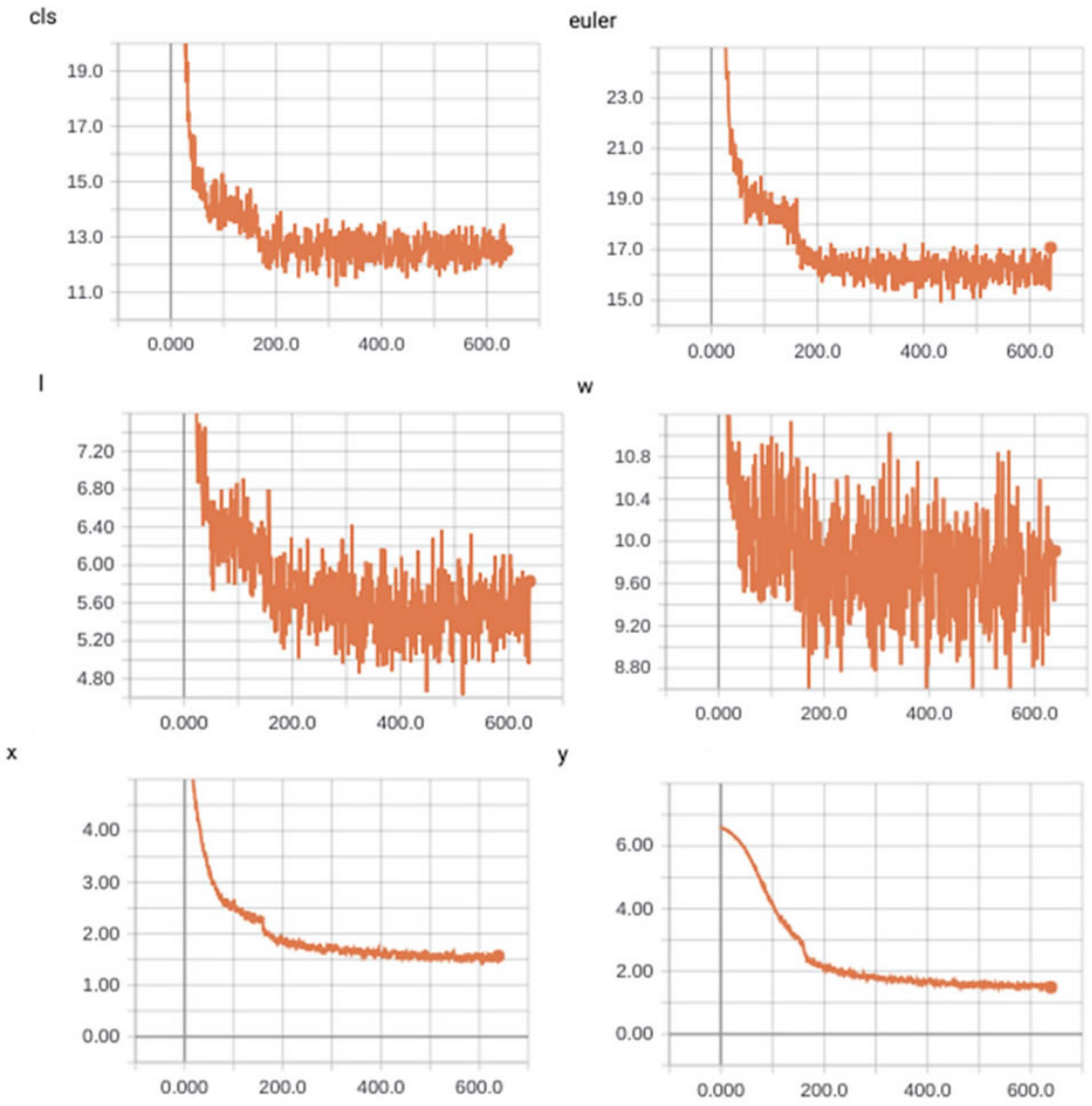


Figure 6.4: Training loss visualization using TensorBoard [6]. Top: training loss on class labels, Euler region proposals Middle and Bottom: training loss on object length, object width, horizontal and vertical locations. See Section. 6.4 for details.

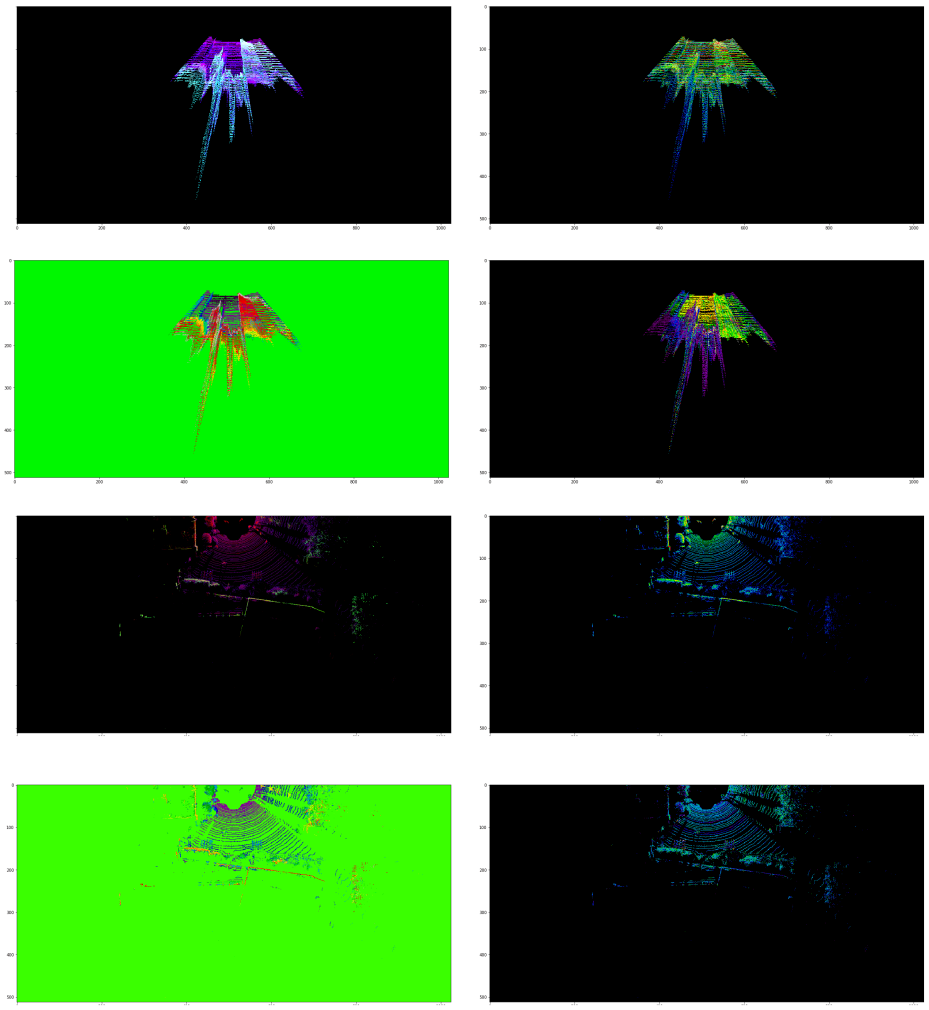


Figure 6.5: Feature map visualization (Best viewed electronically). Top two rows: Our combined BEV feature map, density map, height map and grayscale intensity map. Bottom two rows: Feature map, density map, height map and intensity map on corresponding LiDAR scans used by Complex YOLO [116]. See Section. 6.3.2 for details.

3D bounding box labels to 2D bounding boxes in the BEV coordinates. We set the base learning rate = 0.0001, gamma = 0.5, momentum = 0.9 and batch size = 32. We train for 700 epochs with four NVIDIA Titan V GPUs. The training losses mentioned in Section 6.3.3 are visualized in Fig. 6.4 using TensorBoard [6] until the 600th step ($\sim$ 4 epochs).

Similar to [116] which perform PASCAL VOC [32] style mean Average Precision (mAP) evaluation on its own test set split, we run mAP on 2D bounding boxes in the BEV space. There is one difference from our evaluation scheme *vs.* LiDAR-based methods: since LiDAR data has much wider FOV, we only evaluate against ground-truth bounding boxes that fall within our camera FOV. This means that we are not considering ground-truth labels that are too far away, or outside of our camera view frustum. Also, since our predicted depth map is not able to capture fine structures of small objects like LiDARs do, we only evaluate on four categories including car, tram, truck and van. As shown in Fig. 6.6, these four (out of seven) categories consist of more than 85% of objects in the KITTI 3D object detection dataset. We also vary the IoU threshold from 0.5 to 0.9 with a 0.1 interval and recalculate the mAP scores. We show the mAP scores at different IoU thresholds in Fig. 6.6. Compared to existing methods, our framework is one of the few that directly performs inference on RGB images. We compare with MV3D [22] and Complex YOLO [116] results in Table 6.1. The scores of MV3D and Complex YOLO are adopted from the original paper in the BEV category with easy difficulty. Easy difficulty is defined according to the bounding box height and occlusion/truncation levels. In general, the easy task corresponds to cars within 30 meters of the ego-car distance, according to [140]. Note that the effective range of our transformed point cloud is shorter than this 30-meter range. Also, the Complex YOLO scores are reported on the test split (similar to our evaluation) whereas the MV3D reports on the KITTI test set.

6.5 Discussion

We show quantitative PR-curve evaluations in Fig. 6.6 and compare with other methods in Table 6.1. As can be seen from Fig. 4.7, our method achieves satisfactory results on car, truck, van and tram categories, and the car category demonstrates the highest mAP

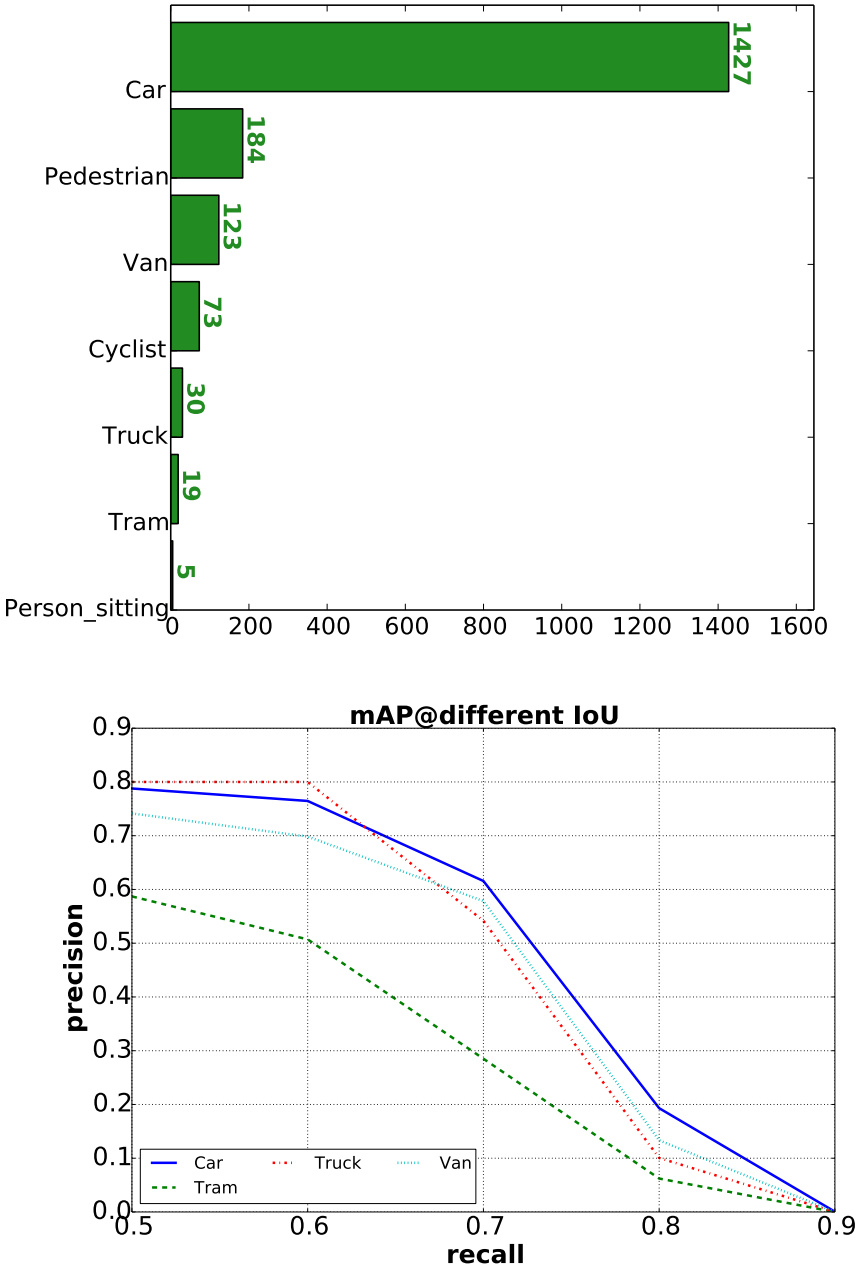


Figure 6.6: Dataset statistics and precision-recall curve. Left: Number of objects per class in the KITTI dataset. Right: mean Average Precision (mAP) values on the car, truck, van and tram classes across varying Intersection-over-Union (IoU) values. Note that the performance of our model is robust to stricter IoU criterias and the performance only begins to significantly degrade when IoU is bigger than 0.6.

scores across varying IoUs. This may due to the fact that cars are more common in real-life scenes and thus easier to recognize. Also, because the categories in the KITTI dataset is highly imbalanced, it is possible that the car class is over-represented and the classifier is biased towards this single class. In the future, we plan to test our approach on an evenly sampled 3D object detection dataset with more diverse examples. According to Fig. 6.6, the mAP starts to dramatically decrease only when the IoU value is more than 0.6. This shows that our detector is robust to stricter evaluation criteria, which is generally more desirable for complex real-life scenes. Also, according to Table 6.1, our method is competitive when compared to other LiDAR-based methods, even though our network only uses RGB images as input to perform forward inference. We visualize the qualitative results in Fig. 6.7. Our approach works well in cluttered scenes (*e.g.* row 1 and 2). It might be difficult for appearance-based methods to separate vehicles parked closely together (row 2 column 1), but our method makes accurate depth predictions and the BEV map (row 2 column 2) makes it much easier to learn the relative locations of the vehicles. However, for small objects and thin structures (*e.g.* pedestrian in row 3 and 4), our network is not able to capture, as the predicted depth maps are not as accurate as LiDAR scans. Also, our method takes both structural (*e.g.* height) and visual cues for inference. For example, in the last row, the closest and farthest objects are wrongly classified as vans while the middle object is correctly classified as a car. This is because our feature map also contains the height map. The SUV and MPV in the front and back are taller than the sedan in the middle, which possibly leads to the wrong classification result. In general, Fig. 6.7 demonstrates that the bounding box predictions (structural) are more accurate than class predictions (visual appearance). This implies that our network is good at localizing objects but is still having difficulties learning visual features of an object. This can also be observed in Fig. 6.4, where the classification loss curve shows more oscillations than bounding box coordinates (x and y). Although the learning objective is designed to minimize both classification and localization errors, it is interesting to see what roles the structural and visual cues play, and when one overwhelms the other. In the future, we plan to train and test on more datasets and visualize neuron activation heatmaps in each channel (height, density and color intensity).

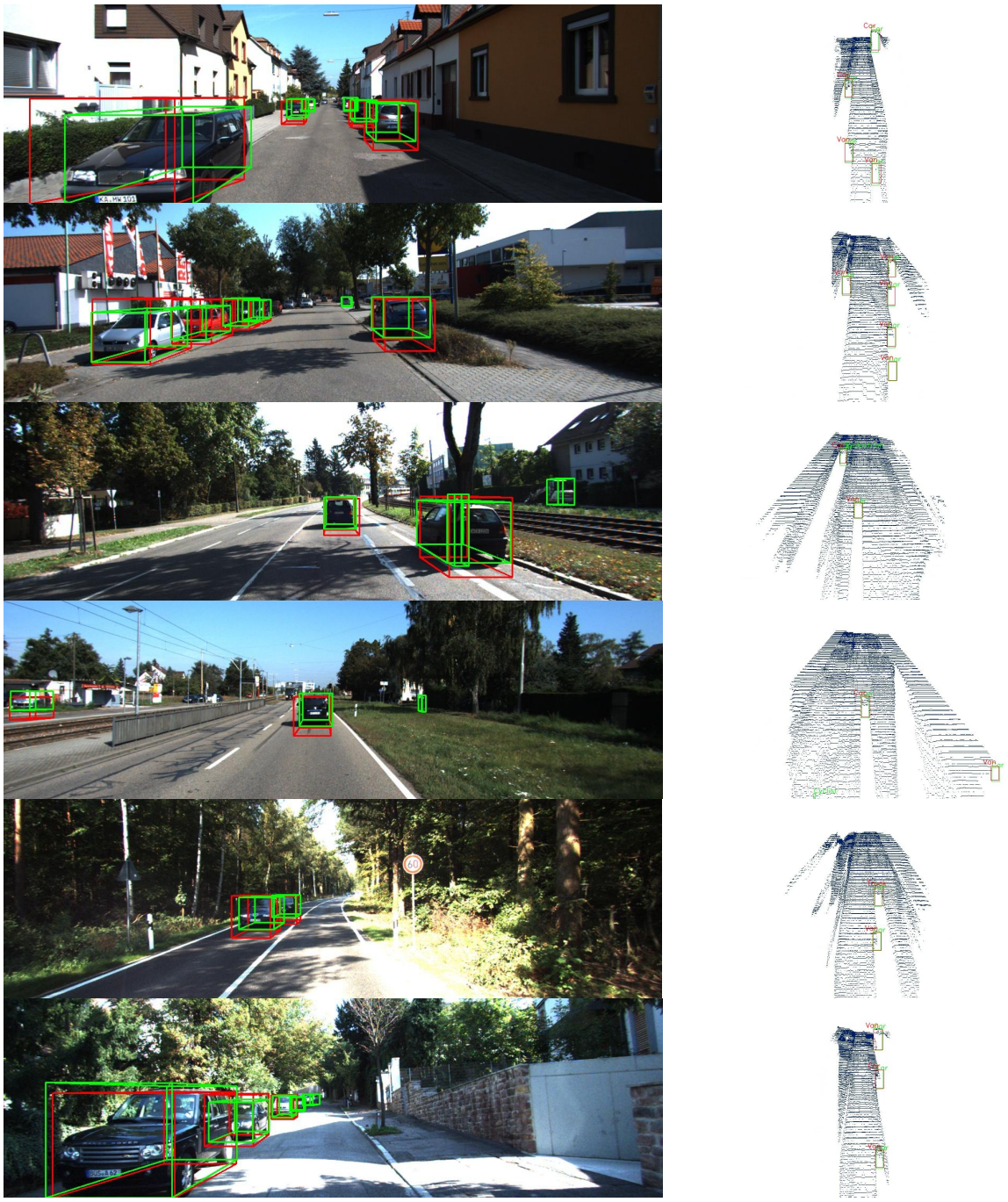


Figure 6.7: Qualitative results on the KITTI dataset. Left: our 3D bounding box predictions (red) *vs.* ground-truth (green) annotations projected to the camera imaging plane. Right: our 2D bounding box predictions (red) on the BEV map vs ground-truth (green) annotations. Note that the camera optical axis is facing down on the BEV map for better visualization. See Section. 6.5 for details.

6.6 Conclusion

In this chapter, we have presented a framework to detect and classify 3D objects from monocular images. Experiments show that our approach performs favorably against competitive methods trained on LiDAR data. Our method leverages generative adversarial networks to perform monocular depth estimation. The training groudtruth are obtained by completing LiDAR scans. The GAN approach is more flexible in terms of extending to other computer vision tasks. On the contrary, traditional monocular depth prediction networks are heavily dependent on pair-wise color-to-depth alignment and LiDAR input. Also, we integrate both visual and structural cues into the feature map representation, which distinguishes our method from those purely operating on LiDAR data, and those who learn depth from a monocular image but still perform detection on the pseudo LiDAR data (ignoring visual information). Our system can be used to add visual intelligence to smart vehicles, which is particularly useful for improving camera-based advanced driver-assistance systems (ADAS) for L3 level autonomy. Also, our system could be used as a supplementary or fall-back option to LiDAR sensors. In the future, we plan to include spatiotemporal data to improve both depth prediction (*e.g.* optical flow) and object detection (*e.g.* YOLO4D [30]).

Chapter 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this dissertation, we aim at investigating various aspects of the visual object detection problem. Firstly, we investigate an unsupervised manifold learning approach for detecting class-agnostic salient objects. The detection result is a probability map showing the pixel level objectness. Then, we focus on a deep learning based method for class-specific object detection. We modify existing neural networks architecture and extend it for tracking in motion sequences. Our network is improved to output both bounding boxes and class labels. Further, we extend to joint learning enabling both object detection and fine-grained classification. Our work enables joint prediction of bounding box coordinates and fine-grained class labels, improving the previous inter-class level categorization to a more fine-grained intra-class level categorization. Next, we target at moving from 2D object detection to 3D object detection. We begin with a method to capture depth data on low-cost mobile devices. Experimental results show that our method provides a good tradeoff between expected depth-recovering quality and computational efficiency. Finally, we proposed an approach leveraging both adversarial and deep learning to directly predict 3D bounding boxes and class labels on 2D images.

Our research on the object detection problem can be viewed as a multi-dimensional incremental development from unsupervised (Chapter 2) to supervised (Chapter 3, Chapter 4, 6) and from 2D (Chapter 2, 3, 4) to 3D (Chapter 5). We briefly summarize each chapter as follows:

In Chapter 2, we present a novel unsupervised class-agnostic salient object detection method based on a novel graph model and background priors. Our graph model incorporates

local and global contrast and naturally enforces the background connectivity constraint. The proposed feature distance metrics effectively and efficiently combine local color and texture cues to represent the intrinsic manifold structure. We further optimize the background seeds by exploiting a boundary query and refinement scheme, achieving state-of-the-art results. Our proposed graph model exhibits the potential towards building better clustering algorithms.

In Chapter 3, we introduce a method for visual object detection via deep convolutional neural networks. We also extend this framework to enforce temporal continuity for tracking. We modify the network for better performance, specialize it for a robotic application involving bird and nest categories. We also proposed a new dataset for the nest category. The system exhibits very competitive detection accuracy and speed, as well as robust, high-speed tracking on several difficult sequences.

In Chapter 4, we demonstrate a joint framework to detect and classify fine-grained objects. We also created a new benchmark to evaluate this task which we term *fine-grained recognition*, subsuming the problems of object detection and fine-grained classification. Although many deep learning methods have been proposed for each task, we are not aware of any framework that directly train on both fine-grained classification and generic object detection datasets. By learning both spatial locations and intra-class diversities of an object, we enable the network to produce quality feature vectors with high distinctiveness. Additionally, as most convolutional layers in our network are shared for both tasks, we introduce very little computational overhead to achieve combined goals of efficiency and accuracy.

In Chapter 5, we move from 2D to the 3D domain. To begin with, we introduce an algorithm to capture accurate 3D data on a low-cost mobile device. We first capture stereo image pairs on a low-cost tablet, then apply the Graph Cuts stereo matching algorithm to obtain low-resolution disparity maps. Next, we take raw color images as guide images to upsample the low-resolution disparity maps into high-resolution ones via joint bilateral upsampling. We evaluate a variety of real-time stereo matching and edge-preserving upsampling algorithms for the tablet platform. Experimental results show that our approach provides a good tradeoff between expected depth-recovering quality and computational efficiency. We

also show how to render DoF effects based on captured depth maps and why the proposed scheme is advantageous compared to image-based rendering.

In Chapter 6, we propose a novel approach to leverage both depth and visual cues for 3D object detection on monocular images. At the core of our technique is to integrate appearance and structural cues for better object detection. Our method contains three stages. Firstly, we use an unpaired image to image translation network to learn bi-directional transformations from RGB images to depth maps. Secondly, we calculate height, density and grayscale intensity as 3 feature channels and project the feature map to a birds-eye-view representation. Finally, we take advantage of 3D bounding box annotations on LiDAR data and train our object detection model on the feature map. Experiments show that our approach performs favorably against competitive methods trained on LiDAR data.

7.2 Future Work

Visual object detection is one of the most fundamental yet most challenging problems in computer vision. Despite tremendous success in object detection over the past few years with the advent of deep learning, much more research and development still remain to be done to match or exceed human performance. The work presented in the previous chapters point out a few directions for future research. We briefly summarize a few of them as follows:

Computational Complexity Chapter 2, 3 and 5 explicitly investigated computational efficiency issues. For practical robotic applications, real-time or lightweight approaches are much more desirable. Similar to the application scenario presented in Chapter 6, a real-time object detector with high accuracy is a critical step for an autonomous vehicle to understand the environment with adequate precision for vehicle operation. In the future, we would like to work on model compression, binarization and distillation techniques to improve computational efficiency while maintaining high detection accuracy. We are also interested in applying deep learning models to mobile or embedded devices such as phones, tablets, IoT devices, autonomous vehicle computing platforms, etc.

Feature Robustness As mentioned in Chapter 4, humans are able to rapidly identify the model of a car from key visual features. However, CNNs struggle on learning a robust feature representation of objects because it learns pose and geometry information in the convolution layers, but only keeps the category level info in the final dense layers. Since most of the state-of-the-art object detectors heavily depend on the CNNs as the main feature extractor, they are inherently limited to learning only partial semantic features. Learning robust geometry-related features invariant to pose, scale and deformations are critical towards improving reliability, usability, and applicability of existing deep learning models. In the future, we plan to conduct research on attention-based models and feature pooling to improve feature quality.

Weakly Supervised Learning Some object detectors might work at small scale, but would be difficult to operate at the industry level. Also, real-world images or semantic classes are diverse, dynamic and constantly changing. For example, the model trained on the car dataset mentioned in Chapter 4 may not work well for new car models. For another example, images and depth data used for training in the KITTI dataset mentioned in Chapter 6 are captured in Europe, and might not represent the city and road structures in the US. This volatile nature of training samples brings an extra layer of difficulty in designing a visual object detector. Therefore, we believe that it is possible to maintain high accuracy based on existing models while eliminating the need for data collection by leveraging weakly supervised learning approaches.

Scene Context Reasoning As pointed out by Chapter 3, a possible future research direction would be to incorporate scene context (sky/ground/tree segmentations) into the detection process. Also, the crux of the graph model proposed by Chapter 2 is to incorporate background contextual information to aid foreground detection. We see associating objects with scene structures as a future research area to work on. This could potentially be done by leveraging knowledge graph or multi-task learning approaches. One could also jointly train a model on image embeddings along with text embeddings. This multi-modal approach

could leverage strong priors in language logic and structures and push the state-of-the-art in visual object detection.

Spatiotemporal Reasoning In both Chapter 3 and 6 we mentioned that motion cues could be an important feature to learn in addition to visual and structural cues. Detection and discrimination of motion are one of the most fundamental abilities of humans. We think most of the work presented in this dissertation have direct extensions to the spatiotemporal domain. Also, learning spatiotemporal information could potentially reduce the complexity of visual object detectors. Instead of running forward inference on every frame, one could perform object detection on only a few keyframes and track the object locations in the intermediate frames. Also, the recurrences of objects in multiple frames could enable motion forecasting, which helps to narrow down the region where visual object detectors would scan and generate proposals.

BIBLIOGRAPHY

[1] Complex yolo with uncertainty. https://github.com/wl5/complex_yolo_3d. Accessed: 4/10/2019.

[2] pykitti open source utility library. https://github.com/utiasSTARS/pykitti. Accessed: 4/10/2019.

[3] Sighthound cloud api for vehicle recognition. https://www.sighthound.com/products/cloud. Accessed: 3/10/2019.

[4] Tesseract open source ocr engine. https://github.com/tesseract-ocr/tesseract. Accessed: 3/10/2019.

[5] Pascal voc challenge performance evaluation and download server – detection. http://host.robots.ox.ac.uk:8080/leaderboard/displaylb.php?challengeid=11&compid=4, January 31, 2016.

[6] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[7] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Susstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *PAMI*, 34(11):2274–2282, 2012.

[8] Ravi Achanta, Sheila Hemami, Francisco Estrada, and Sabine Susstrunk. Frequency-tuned salient region detection. In *CVPR*, pages 1597–1604. IEEE, 2009.

[9] Andrew Adams, David E Jacobs, Jennifer Dolson, Marius Tico, Kari Pulli, Eino-Ville Talvala, Boris Ajdin, Daniel Vaquero, Hendrik Lensch, Mark Horowitz, et al. The frankencamera: an experimental platform for computational photography. *ACM Transactions on Graphics (TOG)*, 29(4):29, 2010.

[10] Waleed Ali, Sherif Abdelkarim, Mahmoud Zidan, Mohamed Zahran, and Ahmad El Sallab. Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[11] P. Arbelaez, J. Pont-Tuset, J. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Computer Vision and Pattern Recognition (CVPR)*, pages 328–335. IEEE, 2014.

[12] Pablo Arbelaez, Charless Fowlkes, and David Martin. The berkeley segmentation dataset and benchmark. *see http://www. eecs. berkeley. edu/Research/Projects/CS/vision/bsds*, 2007.

[13] R. Benenson, M. Mathias, R. Timofte, and L. Van Gool. Pedestrian detection at 100 frames per second. In *Computer Vision and Pattern Recognition (CVPR)*. 2012.

[14] Avrim Blum and Tom Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the eleventh annual conference on Computational learning theory*, pages 92–100. ACM, 1998.

[15] Ali Borji. What is a salient object? a dataset and a baseline model for salient object detection. *TIP*, 24(2):742–756, 2015.

[16] 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.

[17] Neil Bruce and John Tsotsos. Saliency based on information maximization. In *Advances in neural information processing systems*, pages 155–162, 2006.

[18] Emmanuel J Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *JACM*, 58(3):11, 2011.

[19] Joao Carreira and Cristian Sminchisescu. Cpmc: Automatic object segmentation using constrained parametric min-cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1312–1328, 2012.

[20] Jin-Xiang Chai, Xin Tong, Shing-Chow Chan, and Heung-Yeung Shum. Plenoptic sampling. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 307–318. ACM Press/Addison-Wesley Publishing Co., 2000.

[21] Kai-Yueh Chang, Tyng-Luh Liu, Hwann-Tzong Chen, and Shang-Hong Lai. Fusing generic objectness and visual saliency for salient object detection. In *ICCV*, pages 914–921. IEEE, 2011.

[22] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[23] Ming Cheng, Niloy J Mitra, Xumin Huang, Philip HS Torr, and Song Hu. Global contrast based salient region detection. *PAMI*, 37(3):569–582, 2015.

[24] Ming-Ming Cheng, Niloy J Mitra, Xiaolei Huang, Philip HS Torr, and Shi-Min Hu. Salient object detection and segmentation. *Image*, 2(3):9, 2011.

[25] Robert L Cook, Thomas Porter, and Loren Carpenter. Distributed ray tracing. In *ACM SIGGRAPH Computer Graphics*, volume 18, pages 137–145. ACM, 1984.

[26] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2005.

[27] Design-Design. http://www.design-design.co.uk/sample-mpo-3d-images-for-television-display.

[28] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia. Incorporating second-order functional knowledge for better option pricing. *Advances in Neural Information Processing Systems (NIPS)*, pages 472–478, 2001.

[29] Andreas Eitel, Jost Tobias Springenberg, Luciano Spinello, Martin Riedmiller, and Wolfram Burgard. Multimodal deep learning for robust rgb-d object recognition. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 681–687. IEEE, 2015.

[30] Ahmad El Sallab, Ibrahim Sobh, Mahmoud Zidan, Mohamed Zahran, and Sherif Abdelkarim. Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds. 2018.

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

[32] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[33] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[34] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[35] Andreas Geiger, Martin Roser, and Raquel Urtasun. Efficient large-scale stereo matching. In *Computer Vision–ACCV 2010*, pages 25–38. Springer, 2011.

[36] Todor Georgiev and Andrew Lumsdaine. Focused plenoptic camera and rendering. *Journal of Electronic Imaging*, 19(2):021106–021106, 2010.

[37] Todor Georgiev, Zhan Yu, Andrew Lumsdaine, and Sergio Goma. Lytro camera technology: theory, algorithms, performance analysis. In *IS&T/SPIE Electronic Imaging*, pages 86671J–86671J. International Society for Optics and Photonics, 2013.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38(1):142–158, 2016.

[39] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *CVPR*, volume 2, page 7, 2017.

[40] 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.

[41] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentation. In *European Conference on Computer Vision*, pages 345–360. Springer, 2014.

[42] Paul Haeberli and Kurt Akeley. The accumulation buffer: hardware support for high-quality rendering. In *ACM SIGGRAPH Computer Graphics*, volume 24, pages 309–318. ACM, 1990.

[43] F. Han, Y. Shan, R. Cekander, H. Sawhney, and R. Kumar. A twostage approach to people and vehicle detection with hog-based svm. In *Proceedings of the Performance Metrics for Intelligent Systems Workshop*. 2006.

[44] Jonathan Harel, Christof Koch, and Pietro Perona. Graph-based visual saliency. In *NIPS*, pages 545–552, 2006.

[45] Di He, Yingce Xia, Tao Qin, Liwei Wang, Nenghai Yu, Tieyan Liu, and Wei-Ying Ma. Dual learning for machine translation. In *Advances in Neural Information Processing Systems*, pages 820–828, 2016.

[46] K. He, X. Zhang, R. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[47] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, pages 2980–2988. IEEE, 2017.

[48] Kaiming He, Jian Sun, and Xiaoou Tang. Guided image filtering. *PAMI*, 35(6):1397–1409, 2013.

[49] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[50] 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.

[51] J. Henriques, R. Caseiro, P. Martins, and J. Batista. Exploiting the circulant structure of tracking-by-detection with kernels. In *European Conference on Computer Vision (ECCV)*, 2012.

[52] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. Image analogies. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 2001.

[53] Bernhard Hill, Th Roger, and Friedrich Wilhelm Vorhagen. Comparative analysis of the quantization of color spaces on the basis of the cielab color-difference formula. *ACM TOG*, 16(2):109–154, 1997.

[54] Heiko Hirschmuller. Accurate and efficient stereo processing by semi-global matching and mutual information. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2, pages 807–814. IEEE, 2005.

[55] L. Huang, Y. Yang, Y. Deng, and Y. Yu. Densebox: Unifying landmark localization with end to end object detection. *arXiv preprint arXiv:1509.04874*, 2015.

[56] Shaoli Huang, Zhe Xu, Dacheng Tao, and Ya Zhang. Part-stacked cnn for fine-grained visual categorization. In *CVPR*, pages 1173–1182, 2016.

[57] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

[58] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259, 1998.

[59] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015.

[60] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the ACM International Conference on Multimedia*, pages 675–678. ACM, 2014.

[61] Bowen Jiang, Lihe Zhang, Huchuan Lu, Chuan Yang, and Ming-Hsuan Yang. Saliency detection via absorbing markov chain. In *ICCV*, pages 1665–1672. IEEE, 2013.

[62] Huaizu Jiang, Jingdong Wang, Zejian Yuan, Yang Wu, Nanning Zheng, and Shipeng Li. Salient object detection: A discriminative regional feature integration approach. In *CVPR*, 2013.

[63] Peng Jiang, Haibin Ling, Jingyi Yu, and Jingliang Peng. Salient region detection by ufo: Uniqueness, focusness and objectness. In *ICCV*, pages 1976–1983. IEEE, 2013.

[64] Tilke Judd, Frédo Durand, and Antonio Torralba. A benchmark of computational models of saliency to predict human fixations. In *MIT Technical Report*, 2012.

[65] Tilke Judd, Krista Ehinger, Frédo Durand, and Antonio Torralba. Learning to predict where humans look. In *ICCV*, pages 2106–2113. IEEE, 2009.

[66] Vladimir Kolmogorov and Ramin Zabih. Computing visual correspondence with occlusions using graph cuts. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 2, pages 508–515. IEEE, 2001.

[67] Shu Kong and Charless Fowlkes. Low-rank bilinear pooling for fine-grained classification. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 7025–7034. IEEE, 2017.

[68] Johannes Kopf, Michael F Cohen, Dani Lischinski, and Matt Uyttendaele. Joint bilateral upsampling. *ACM Trans. Graph.*, 26(3):96, 2007.

[69] Jonathan Krause, Hailin Jin, Jianchao Yang, and Li Fei-Fei. Fine-grained recognition without part annotations. In *CVPR*, pages 5546–5555, 2015.

[70] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*, Sydney, Australia, 2013.

[71] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems (NIPS)*, pages 1097–1105, 2012.

[72] Michael Lam, Behrooz Mahasseni, and Sinisa Todorovic. Fine-grained recognition as hsnet search for informative image parts. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6497–6506. IEEE, 2017.

[73] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. Depth-of-field rendering with multiview synthesis. In *ACM Transactions on Graphics (TOG)*, volume 28, page 134. ACM, 2009.

[74] Sungkil Lee, Elmar Eisemann, and Hans-Peter Seidel. Real-time lens blur effects and focus control. *ACM Transactions on Graphics (TOG)*, 29(4):65, 2010.

[75] K. Lenc and A. Vedaldi. R-cnn minus r. *arXiv preprint arXiv:1506.06981*, 2015.

[76] Ian Lenz, Honglak Lee, and Ashutosh Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[77] Thomas Leung and Jitendra Malik. Representing and recognizing the visual appearance of materials using three-dimensional textons. *IJCV*, 43(1):29–44, 2001.

[78] Changyang Li, Yuchen Yuan, Weidong Cai, Yong Xia, and David Dagan Feng. Robust saliency detection via regularized random walks ranking. In *CVPR*. IEEE, 2015.

[79] Guanbin Li and Yizhou Yu. Visual saliency based on multiscale deep features. 2015.

[80] H. Li, Y. Li, and F. Porikli. Deeptrack: Learning discriminative feature representations by convolutional neural networks for visual tracking. In *British Machine Vision Conference (BMVC)*, 2014.

[81] Xi Li, Yao Li, Chunhua Shen, Anthony Dick, and Anton Hengel. Contextual hypergraph modeling for salient object detection. In *ICCV*, pages 3328–3335, 2013.

[82] Xiaohui Li, Huchuan Lu, Lihe Zhang, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via dense and sparse reconstruction. In *ICCV*, pages 2976–2983. IEEE, 2013.

[83] Yin Li, Xiaodi Hou, Christian Koch, James M Rehg, and Alan L Yuille. The secrets of salient object segmentation. In *CVPR*, pages 280–287. IEEE, 2014.

[84] Zeming Li, Chao Peng, Gang Yu, Xiangyu Zhang, Yangdong Deng, and Jian Sun. Light-head r-cnn: In defense of two-stage object detector. *arXiv preprint arXiv:1711.07264*, 2017.

[85] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018.

[86] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. Bilinear cnn models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

[87] Gabriel Lippmann. La photographie intgrale. *Comptes-Rendus,Acadmie des Sciences*, 146:446–451, 1908.

[88] Tie Liu, Zejian Yuan, Jian Sun, Jingdong Wang, Nanning Zheng, Xiaoou Tang, and Heung-Yeung Shum. Learning to detect a salient object. *PAMI*, 33(2):353–367, 2011.

[89] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[90] Xiao Liu, Jiang Wang, Shilei Wen, Errui Ding, and Yuanqing Lin. Localizing by describing: Attribute-guided attention localization for fine-grained recognition. In *AAAI*, pages 4190–4196, 2017.

[91] Lytro. www.lytro.com.

[92] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[93] Fangchang Mal and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018.

[94] Ran Margolin, Avishay Tal, and Lihi Zelnik-Manor. What makes a patch distinct? In *CVPR*, pages 1139–1146. IEEE, 2013.

[95] M. Mathias, R. Timofte, R. Benenson, and L. Van Gool. Traffic sign recognition how far are we from the solution? In *Int. Joint Conf. on Neural Networks*. 2013.

[96] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, et al. Junior: The stanford entry in the urban challenge. *Journal of field Robotics*, 25(9):569–597, 2008.

[97] Ren Ng, Marc Levoy, Mathieu Brédif, Gene Duval, Mark Horowitz, and Pat Hanrahan. Light field photography with a hand-held plenoptic camera. *Computer Science Technical Report CSTR*, 2(11), 2005.

[98] Timo Ojala, Matti Pietikäinen, and David Harwood. A comparative study of texture measures with classification based on featured distributions. *Pattern Recognition*, 29(1):51–59, 1996.

[99] Maxime Oquab, Léon Bottou, Ivan Laptev, and Josef Sivic. Is object localization for free?-weakly-supervised learning with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 685–694, 2015.

[100] Derrick Parkhurst, Klinton Law, and Ernst Niebur. Modeling the role of salience in the allocation of overt visual attention. *Vision research*, 42(1):107–123, 2002.

[101] PelicanImaging. www.pelicanimaging.com.

[102] Federico Perazzi, Philipp Krähenbühl, Yael Pritch, and Alexander Hornung. Saliency filters: Contrast based filtering for salient region detection. In *CVPR*, pages 733–740. IEEE, 2012.

[103] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[104] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[105] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *arXiv preprint arXiv:1506.02640*, 2015.

[106] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[107] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. *arXiv preprint*, 2017.

[108] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 91–99, 2015.

[109] Paul L Rosin. A simple method for detecting salient regions. *Pattern Recognition*, 42(11):2363–2371, 2009.

[110] Christian Scharfenberger, Steven L Waslander, John S Zelek, and David A Clausi. Existence detection of objects in images for robot vision using saliency histogram features. In *Computer and Robot Vision (CRV), 2013 International Conference on*, pages 75–82. IEEE, 2013.

[111] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International journal of computer vision*, 47(1-3):7–42, 2002.

[112] D. Sergeant, R. Boyle, and M. Forbes. Computer visual tracking of poultry. *Computers and Electronics in Agriculture*, 21(1), 1998.

[113] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[114] Xiaohui Shen and Ying Wu. A unified approach to salient object detection via low rank matrix recovery. In *CVPR*, pages 853–860. IEEE, 2012.

[115] Marcel Simon and Erik Rodner. Neural activation constellations: Unsupervised part model discovery with convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1143–1151, 2015.

[116] Martin Simony, Stefan Milzy, Karl Amendey, and Horst-Michael Gross. Complex-yolo: an euler-region-proposal for real-time 3d object detection on point clouds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

[117] Richard Socher, Brody Huval, Bharath Bath, Christopher D Manning, and Andrew Y Ng. Convolutional-recursive deep learning for 3d object classification. In *Advances in Neural Information Processing Systems*, pages 656–664, 2012.

[118] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, volume 5, page 6, 2015.

[119] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[120] K. Steen, O. Therkildsen, O. Green, and H. Karstoft. Detection of bird nests during mechanical weeding by incremental background modeling and visual saliency. *Sensors*, 3(15), 2015.

[121] Leslie Stroebel, John Compton, Ira Current, and Richard Zakia. *Photographic Materials and Processes*. Focal Press, Boston/London, 1986.

[122] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):787–800, 2003.

[123] Narayanan Sundaram and Kurt Keutzer. Long term video segmentation through pixel level spectral clustering on gpus. In *ICCV Workshops*, pages 475–482. IEEE, 2011.

[124] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR)*, 2015.

[125] Marshall F Tappen and William T Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. In *Ninth IEEE International Conference on Computer Vision*, pages 900–906. IEEE, 2003.

[126] Sasha Targ, Diogo Almeida, and Kevin Lyman. Resnet in resnet: generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

[127] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.

[128] Carlo Tomasi and Roberto Manduchi. Bilateral filtering for gray and color images. In *Sixth International Conference on Computer Vision, 1998.*, pages 839–846. IEEE, 1998.

[129] A. Treisman and G. Gelade. A feature-integration theory of attention. *Cognitive psychology*, 12(1):97–136, 1980.

[130] Alejandro Troccoli, Dawid Pajak, and Kari Pulli. Fcam for multiple cameras. In *IS&T/SPIE Electronic Imaging*, pages 830404–830404. International Society for Optics and Photonics, 2012.

[131] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[132] Ashok Veeraraghavan, Ramesh Raskar, Amit Agrawal, Ankit Mohan, and Jack Tumblin. Dappled photography: Mask enhanced cameras for heterodyned light fields and coded aperture refocusing. *ACM Transactions on Graphics*, 26(3):69, 2007.

[133] L. Wang, W. Ouyang, X. Wang, and H. Lu. Visual tracking with fully convolutional networks. In *International Conference on Computer Vision (ICCV)*. 2015.

[134] N. Wang and D. Yeung. Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems (NIPS)*. 2013.

[135] Peng Wang, Jingdong Wang, Gang Zeng, Jie Feng, Hongbin Zha, and Shipeng Li. Salient object detection for searched web images via global saliency. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3194–3201. IEEE, 2012.

[136] Qiaosong Wang, Christopher Rasmussen, and Chunbo Song. Fast, deep detection and tracking of birds and nests. In *International Symposium on Visual Computing*, pages 146–155. Springer, 2016.

[137] Qiaosong Wang, Zhan Yu, Christopher Rasmussen, and Jingyi Yu. Stereo vision based depth of field rendering on a mobile device. In *Proc. of SPIE Vol*, volume 9023, pages 902307–1.

[138] Qiaosong Wang, Zhan Yu, Christopher Rasmussen, and Jingyi Yu. Stereo vision–based depth of field rendering on a mobile device. *Journal of Electronic Imaging*, 23(2):023009, 2014.

[139] Qiaosong Wang, Wen Zheng, and Robinson Piramuthu. Grab: Visual saliency via novel graph model and background priors. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 535–543, 2016.

[140] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. *arXiv preprint arXiv:1812.07179*, 2018.

[141] Yichen Wei, Fang Wen, Wangjiang Zhu, and Jian Sun. Geodesic saliency using background priors. In *ECCV*, pages 29–42. Springer, 2012.

[142] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology, 2010.

[143] Bennett Wilburn, Neel Joshi, Vaibhav Vaish, Eino-Ville Talvala, Emilio Antunez, Adam Barth, Andrew Adams, Mark Horowitz, and Marc Levoy. High performance imaging using large camera arrays. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 765–776. ACM, 2005.

[144] X. Wu, P. Yuan, Q. Peng, C. Ngo, and J. He. Detection of bird nests in overhead catenary system images for high-speed rail. *Pattern Recognition*, 51:242–254, 2016.

[145] Y. Wu, J. Lim, and M. Yang. Online object tracking: A benchmark. In *Computer Vision and Pattern Recognition (CVPR)*, 2013.

[146] Tianjun Xiao, Yichong Xu, Kuiyuan Yang, Jiaxing Zhang, Yuxin Peng, and Zheng Zhang. The application of two-level attention models in deep convolutional neural network for fine-grained image classification. In *CVPR*, pages 842–850, 2015.

[147] Saining Xie, Tianbao Yang, Xiaoyu Wang, and Yuanqing Lin. Hyper-class augmented and regularized deep learning for fine-grained image classification. In *CVPR*, pages 2645–2654, 2015.

[148] Yulin Xie, Huchuan Lu, and Ming-Hsuan Yang. Bayesian saliency via low and mid level cues. *TIP*, 22(5):1689–1698, 2013.

[149] Li Xu, Qiong Yan, Yang Xia, and Jiaya Jia. Structure extraction from texture via relative total variation. *ACM TOG*, 31(6):139, 2012.

[150] Qiong Yan, Li Xu, Jianping Shi, and Jiaya Jia. Hierarchical saliency detection. In *CVPR*, June 2013.

[151] Chuan Yang, Lihe Zhang, Huchuan Lu, Xiang Ruan, and Ming-Hsuan Yang. Saliency detection via graph-based manifold ranking. In *CVPR*, pages 3166–3173. IEEE, 2013.

[152] Jason C Yang, Matthew Everett, Chris Buehler, and Leonard McMillan. A real-time distributed light field camera. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 77–86. Eurographics Association, 2002.

[153] Jimei Yang, B. Price, S. Cohen, Zhe Lin, and Ming-Hsuan Yang. Patchcut: Data-driven object segmentation via local shape transfer. In *CVPR*, pages 1770–1778. IEEE, 2015.

[154] Linjie Yang, Ping Luo, Chen Change Loy, and Xiaoou Tang. A large-scale car dataset for fine-grained categorization and verification. In *CVPR*, pages 3973–3981, 2015.

[155] Bangpeng Yao, Aditya Khosla, and Li Fei-Fei. Combining randomization and discrimination for fine-grained image categorization. In *CVPR*. IEEE, 2011.

[156] Zili Yi, Hao (Richard) Zhang, Ping Tan, and Minglun Gong. Dualgan: Unsupervised dual learning for image-to-image translation. In *ICCV*, pages 2868–2876, 2017.

[157] Xuan Yu, Rui Wang, and Jingyi Yu. Real-time depth of field rendering via dynamic light field generation and filtering. In *Computer Graphics Forum*, volume 29, pages 2099–2107. Wiley Online Library, 2010.

[158] Zhan Yu, Christopher Thorpe, Xuan Yu, Scott Grauer-Gray, Feng Li, and Jingyi Yu. Dynamic depth of field on live video streams: A stereo solution. 2011.

[159] Zhan Yu, Xuan Yu, Christopher Thorpe, Scott Grauer-Gray, Feng Li, and Jingyi Yu. Racking focus and tracking focus on live video streams: a stereo solution. *The Visual Computer*, pages 1–14, 2013.

[160] Jianming Zhang, Shugao Ma, Mehrnoosh Sameki, Stan Sclaroff, Margrit Betke, Zhe Lin, Xiaohui Shen, Brian Price, and Radomir Mech. Salient object subitizing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4045–4054, 2015.

[161] K. Zhang, L. Zhang, and M. Yang. Real-time compressive tracking. In *European Conference on Computer Vision (ECCV)*, 2012.

[162] Ning Zhang, Jeff Donahue, Ross Girshick, and Trevor Darrell. Part-based r-cnns for fine-grained category detection. In *ECCV*, pages 834–849. Springer, 2014.

[163] Xiaofan Zhang, Feng Zhou, Yuanqing Lin, and Shaoting Zhang. Embedding label structures for fine-grained feature representation. In *CVPR*, pages 1114–1123, 2016.

[164] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[165] Heliang Zheng, Jianlong Fu, Tao Mei, and Jiebo Luo. Learning multi-attention convolutional neural network for fine-grained image recognition. In *Int. Conf. on Computer Vision*, volume 6, 2017.

[166] W. Zhong, H. Lu, and M. Yang. Robust object tracking via sparsity-based collaborative model. In *Computer Vision and Pattern Recognition (CVPR)*. 2012.

[167] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.

[168] Dengyong Zhou, Jason Weston, Arthur Gretton, Olivier Bousquet, and Bernhard Schölkopf. Ranking on data manifolds. *NIPS*, 16:169–176, 2004.

[169] Tinghui Zhou, Philipp Krahenbuhl, Mathieu Aubry, Qixing Huang, and Alexei A Efros. Learning dense correspondence via 3d-guided cycle consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 117–126, 2016.

[170] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

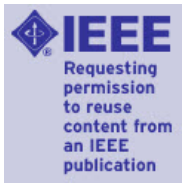
[171] Wangjiang Zhu, Shuang Liang, Yichen Wei, and Jian Sun. Saliency optimization from robust background detection. In *CVPR*, pages 2814–2821. IEEE, 2014.

[172] C. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *European Conference on Computer Vision (ECCV)*, pages 391–405. Springer, 2014.

Appendix A

PERMISSIONS

The following are reuse permissions for some materials in this paper.



Title:	GraB: Visual Saliency via Novel Graph Model and Background Priors
Conference Proceedings:	2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)
Author:	Qiaosong Wang
Publisher:	IEEE
Date:	June 2016

Thesis / Dissertation Reuse

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:

Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1) In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.
2) In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.
3) If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1) The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]
2) Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.
3) In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to http://www.ieee.org/publications_standards/publications/rights/rights_link.html to learn how to obtain a License from RightsLink.

If applicable, University Microfilms and/or ProQuest Library, or the Archives of Canada may supply single copies of the dissertation.

BACK CLOSE WINDOW

Order Completed

Thank you for your order.

This Agreement between Qiaosong Wang ("You") and Springer Nature ("Springer Nature") consists of your license details and the terms and conditions provided by Springer Nature and Copyright Clearance Center.

Your confirmation email will contain your order number for future reference.

printable details

License Number	4579070993343
License date	Apr 30, 2019
Licensed Content Publisher	Springer Nature
Licensed Content Publication	Springer eBook
Licensed Content Title	Fast, Deep Detection and Tracking of Birds and Nests
Licensed Content Author	Qiaosong Wang, Christopher Rasmussen, Chunbo Song
Licensed Content Date	Jan 1, 2016
Type of Use	Thesis/Dissertation
Requestor type	non-commercial (non-profit)
Format	print and electronic
Portion	full article/chapter
Will you be translating?	no
Circulation/distribution	>50,000
Author of this Springer Nature content	yes
Title	Manifold, Deep and Adversarial Learning for Visual Object Detection
Institution name	University of Delaware
Expected presentation date	Jun 2019
Requestor Location	

Total	0.00 USD

ORDER MORE CLOSE WINDOW

G Suite

Qiaosong Wang <qiaosong@udel.edu>

Permission for reuse in dissertation

3 messages

Qiaosong Wang <qiaosong@udel.edu>
To: reprint_permission@spie.org

Dear Sir/Madam,

I request the following two papers to be reused in my dissertation (I am the author of both papers):

- Title and author: Stereo Vision based Depth of Field Rendering on a Mobile Device
  Volume, issue, and page numbers: Proc. of SPIE Vol 9023, page 902307--1 year 2014
- What you would like to reproduce: main methods proposed in this paper
- Where you will republish the requested material: University of Delaware PhD Dissertation
-

- Title and author: Stereo Vision based Depth of Field Rendering on a Mobile Device
  Volume, issue, and page numbers: Journal of Electronic Imaging volume 23 number 2 pages 023009 year 2014
- What you would like to reproduce: main methods proposed in this paper
- Where you will republish the requested material: University of Delaware PhD Dissertation
-

Thank you,
Qiaosong Wang
--
Best Regards,
Qiaosong Wang
http://qiaosongwang.com

Katie Sinclair <katies@spie.org>
To: Qiaosong Wang <qiaosong@udel.edu>

Dear Mr. Wang,

Thank you for seeking permission from SPIE to reprint material from our publications. SPIE shares the copyright with you, so as author you retain the right to reproduce your paper in part or in whole.

Publisher's permission is hereby granted under the following conditions:

(1) the material to be used has appeared in our publication without credit or acknowledgment to another source; and

(2) you credit the original SPIE publication. Include the authors' names, title of paper, volume title, SPIE volume number, and year of publication in your credit statement.

Best of luck with your dissertation! Please let us know if we can be of further assistance.

Best,

Katie Sinclair

Editorial Assistant, Publications

SPIE

+1 360 685 5436

katies@spie.org

Pascal VOC Challenges 2005-2012

The VOC series of challenges has now finished. We are very grateful to the hundreds of participants that have taken part in the challenges over the years. The PASCAL VOC Evaluation Server will continue to run.

> ***Mark Everingham***
>
> *It is with great sadness that we report that Mark Everingham died in 2012. Mark was the key member of the VOC project, and it would have been impossible without his selfless contributions. The VOC workshop at ECCV 2012 was dedicated to Mark's memory. A tribute web page has been set up, and an appreciation of Mark's life and work published.*

Details of each of the challenges can be found on the corresponding challenge page:

- The VOC2012 Challenge
- The VOC2011 Challenge
- The VOC2010 Challenge
- The VOC2009 Challenge
- The VOC2008 Challenge
- The VOC2007 Challenge
- The VOC2006 Challenge
- The VOC2005 Challenge

Further details of the challenges may be found in the sections below:

- Best Practice (Recommendations on using the training and test data)
- History and Background of the VOC Challenge
- The PASCAL Object Recognition Database Collection
- Publications relating to the VOC Challenge
- Organizers
- Support

Best Practice

The VOC challenge encourages two types of participation: (i) methods which are trained using only the provided "trainval" (training + validation) data; (ii) methods built or trained using any data except the provided test data, for example commercial systems. In both cases the *test* data must be used strictly for reporting of results alone - it must not be used in any way to train or tune systems, for example by runing multiple parameter choices and reporting the best results obtained.

If using the training data we provide as part of the challenge development kit, all development, e.g. feature selection and parameter tuning, must use the "trainval" (training + validation) set alone. One way is to divide the set into training and validation sets (as suggested in the development kit). Other schemes e.g. *n*-fold cross-validation are equally valid. The tuned algorithms should then be run only *once* on the test data.

In VOC2007 we made all annotations available (i.e. for training, validation and test data) but since then we have not made the test annotations available. Instead, results on the test data are submitted to an evaluation server.

Since algorithms should only be run *once* on the test data we strongly discourage multiple submissions to the server (and indeed the number of submissions for the same algorithm is strictly controlled), as the evaluation server should not be used for parameter tuning.

We encourage you to publish test results always on the latest release of the challenge, using the output of the evaluation server. If you wish to compare methods or design choices e.g. subsets of features, then there are two options: (i) use the entire VOC2007 data, where all annotations are available; (ii) report cross-validation results using the latest "trainval" set alone.

Policy on email address requirements when registering for the evaluation server

In line with the Best Practice procedures (above) we restrict the number of times that the test data can be processed by the evaluation server. To prevent any abuses of this restriction an institutional email address is required when registering for the evaluation server. This aims to prevent one user registering multiple times under different emails. Institutional emails include academic ones, such as name@university.ac.uk, and corporate ones, but not personal ones, such as name@gmail.com or name@123.com.

Database Rights

The VOC data includes images obtained from the "flickr" website. Use of these images must respect the corresponding terms of use:

- "flickr" terms of use

Find free-to-use images

When you do a Google Search, you can filter your results to find images, videos, or text that you have permission to use. To do this, use an Advanced Search filter called "usage rights" that lets you know when you can use, share, or modify something you find online.

Find images, text, and videos you can reuse

1. Go to Advanced Image Search for images or Advanced Search for anything else.
2. In the "all these words" box, type what you want to search.
3. In the "Usage rights" section, use the drop-down to choose what kind of license you want the content to have.
4. Select **Advanced Search**.

Note: Before reusing content, make sure that its license is legitimate and check the exact terms of reuse. For example, the license might require that you give credit to the image creator when you use the image. Google can't tell if the license label is legitimate, so we don't know if the content is lawfully licensed.

Types of usage rights

- **Free to use or share:** Allows you to copy or redistribute its content if the content remains unchanged.
- **Free to use share or modify:** Allows you to copy, modify, or redistribute in ways specified in the license.
- **Commercially:** If you want content for commercial use, be sure to select an option that includes the word "commercially."

How usage rights work

Usage rights help you find content that you can use above and beyond fair use . Site owners can use licenses to let you know if and how content on their sites can be reused.

The usage rights filter in Advanced Search shows you content either labeled with a Creative Commons or similar license, or is in the public domain. For images, the usage rights filter also shows you images labeled with the GNU Free Documentation license.

Report incorrect usage rights

If you find content with the wrong usage rights in the search results, let us know in the Google Search Forum .

The KITTI Vision Benchmark Suite

A project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago

Karlsruhe Institute of Technology

home | setup | stereo | flow | sceneflow | depth | odometry | object | tracking | road | semantics | raw data | submit results

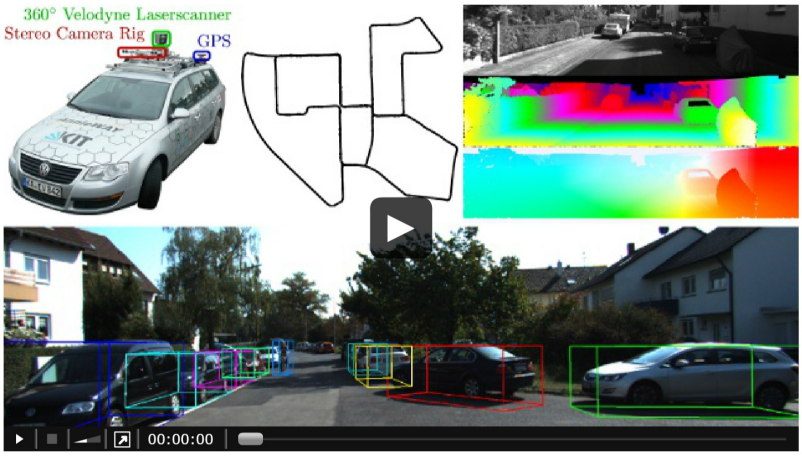
Andreas Geiger (MPI Tübingen) | Philip Lenz (KIT) | Christoph Stiller (KIT) | Raquel Urtasun (University of Toronto)

Welcome to the KITTI Vision Benchmark Suite!

We take advantage of our autonomous driving platform Annieway to develop novel challenging real-world computer vision benchmarks. Our tasks of interest are: stereo, optical flow, visual odometry, 3D object detection and 3D tracking. For this purpose, we equipped a standard station wagon with two high-resolution color and grayscale video cameras. Accurate ground truth is provided by a Velodyne laser scanner and a GPS localization system. Our datsets are captured by driving around the mid-size city of Karlsruhe, in rural areas and on highways. Up to 15 cars and 30 pedestrians are visible per image. Besides providing all data in raw format, we extract benchmarks for each task. For each of our benchmarks, we also provide an evaluation metric and this evaluation website. Preliminary experiments show that methods ranking high on established benchmarks such as Middlebury perform below average when being moved outside the laboratory to the real world. Our goal is to reduce this bias and complement existing benchmarks by providing real-world benchmarks with novel difficulties to the community.

Share

To get started, grab a cup of your favorite beverage and watch our video trailer (5 minutes):



This video: in high-resolution (720 MB) at youtube

Copyright