COMPUTER VISION FOR PLANT AND ANIMAL INVENTORY

A Dissertation Presented to

the Faculty of the Graduate School

at the University of Missouri

In Partial Fulfillment

of the Requirements for the Degree

Doctor of Philosophy

---

by

GUANG CHEN

Dr. Yi Shang, Advisor

DEC 2021

The undersigned, appointed by the Dean of the Graduate School, have examined the thesis entitled:

COMPUTER VISION FOR
PLANT AND ANIMAL INVENTORY

presented by Guang Chen,
a candidate for the degree of Doctor of Philosophy and hereby certify that, in their opinion, it is worthy of acceptance.

Professor Yi Shang

Professor Jianlin Cheng

Professor Praveen K. Edara

Professor Yunxin Zhao

ACKNOWLEDGMENTS

I would like to express my special appreciation to my advisor Dr. Yi Shang for his guidance, suggestions, supervision and support during my Ph.D. program. I got enlightened by his Artificial Intelligence lessons. I've also benefited a lot from the research programs conducted by him. He is foresighted and bought GPUs for the lab very early before more and more people realized that deep learning would be the most powerful tool for a variety of research topics. And the TITAN X in the lab became the ideal starting point of my deep learning research.

I would like to thank my committee members Dr. Jianlin Cheng, Dr. Praveen K. Edara and Dr. Yunxin Zhao, for providing scientific guidance, advice and feedback during my Ph.D. program at Mizzou.

I would like to thank my labmates in the Distributed and Intelligent Computing Lab at Mizzou. They gave me a happy time at school. Especial thanks to Peng Sun. He is my trustable teammate and helped me a lot. He also looked after my belongings every time I left for an internship.

I would like to thank Joel Sartwell from the Missouri Department of Conservation. He is nice and patient. The cooperation with him is joyful.

I would like to thank my parents for their financial support and encouragement for three decades. It is very lucky for me to have them. They made me focus on my career and never worry about the living expenses before I got my first job.

Finally, I would like to thank my wife, Yini Duanmu. She accompanies me wherever I go, from Beijing to California. She brings me not only care and compassion, but also ideas and techniques. She even helped me implement convolutional neural networks. That's so amazing!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

The population, composition, and spatial distribution of the plants and animals in certain regions are always important data for natural resource management, conservation and farming. The traditional ways to acquire such data require human participation. The procedure of data processing by human is usually cumbersome, expensive and time-consuming. Hence the algorithms for automatic animal and plant inventory show their worth and become a hot topic.

We propose a series of computer vision methods for automated plant and animal inventory, to recognize, localize, categorize, track and count different objects of interest, including vegetation, trees, fishes and livestock animals. We make use of different sensors, hardware platforms, neural network architectures and pipelines to deal with the varied properties and challenges of these objects.

(1) For vegetation analysis, we propose a fast multistage method to estimate the coverage. The reference board is localized based on its edge and texture features. And then a K-means color model of the board is generated. Finally, the vegetation is segmented at pixel level using the color model. The proposed method is robust to lighting condition changes. (2) For tree counting in aerial images, we propose a novel method called density transformer, or DENT, to learn and predict the density of the trees at different positions. DENT uses an efficient multi-receptive field network to extract visual features from different positions. A transformer encoder is applied to filter and transfer useful contextual information across different spatial positions. DENT significantly outperformed the existing state-of-art CNN detectors and regressors on both the dataset built by ourselves and an existing cross-site dataset. (3) We propose a framework of fish classification system using boat cameras. The framework contains two branches. A branch extracts the contextual information from the whole image. The other branch localizes all the individual fish and normalizes their poses.

The classification results from the two branches are weighted based on the clearness of the image and the familiarness of the context. Our system achieved the top 1% rank in the competition of The Nature Conservancy Fisheries Monitoring. (4) We also propose a video-based pig counting algorithm using an inspection robot. We adopt a novel bottom-up keypoint tracking method and a novel spatial-aware temporal response filtering method to count the pigs. The proposed approach outperformed the other methods and even human competitors in the experiments.

Chapter 1

Introduction

1.1 Problem and Motivation

The population and composition of the animals and plants at a certain place are essential information for the studies of ecology, conservation and farming. Imagine the following scenarios: (1) The scientists in Missouri are learning the vegetation structures and expect to measure the density of grass at several sites. (2) Some forests in California are damaged by wildfire in recent years. To estimate the loss, the natural resource specialists expect to compare the changes of the tree count in different regions. (3) The sea fish in the Western and Central Pacific is being threatened by illegal and unregulated fishing practices. The conservationists expect to monitor the species and number of fish caught by the fishing boats. Traditionally, to fetch such information, human laborers are sent to the study fields to measure a set of samples and record the observation. (4) Several pig farms in Jilin Province of China periodically reallocate pig houses to the pigs according to their status, including their age, weight, health condition, and feeding amount. The owners of the pig farm expect to track the pigs and monitor the number of pigs in each region on daily basis.

However, the traditional way for the plant and animal inventory has serious drawbacks as follow: Firstly, in some situations, the showing up of human should be limited or avoided. Possible reasons include: Intrusive inspections may cause disturbance to some local ecosystem; The environmental exposure at the study fields may cause health risk to human; The contact between human and livestock may cause hygiene issues to both of them; Some places are difficult, dangerous or expensive for human to reach, such as the mountain tops of high altitude. Secondly, manual data processing by human is typically cumbersome. This means expensive labor costs and that the time-consuming procedures hinder the managers of the natural resources or farms from making rapid decisions.

The development of hardware and software in recent decades makes efficient automatic or computer-aided plant and animal inventory possible. On one hand, a lot of hardwares have been more powerful and affordable, including cameras, drones, chips for embedded systems and GPUs. These hardware help the tasks in different ways. High-resolution cameras take clear images; And fisheye lenses have wide fields of view. Drones can be used to fly over the study area for aerial imagery. Embedded systems can be deployed on a robot walking in the study area. GPUs accelerate the parallel computation of the models. On the other hand, the breakthrough of computer vision and deep learning algorithms encourages the emergence of artificial intelligence software. The technology of AI is advancing towards maturity in some areas like object classification, localization and tracking, and even outperforming human on some specific tasks. The design of efficient systems and algorithms for plant and animal inventory is worth exploring and also the goal of this dissertation.

Depending on objects of interest and environmental conditions, the technical challenges for computer vision-based automatic inventory systems can be task-specific. For example, the objects may be numerous, crowded and even moving fast and randomly. Day/night cycles, seasons and weathers affect the lighting conditions. The

objects may have different poses, shapes and appearances, and are in different categories. When the data are processed offline, powerful hardware can be used. But for online embedded systems, the computational resources limit the complexity of the models. In this sense, different strategies should be adopted to solve the problems.

1.2 Involved works and publications

We will cover four typical kinds of objects: vegetation, trees, fishes and livestocks in this dissertation. The challenges and the proposed approaches will be introduced. We provide a brief overview as follows:

A) Vegetation Coverage Estimation from Images

- Camera: conventional camera
- Platform: offline
- Input: images
- Properties of objects: dense
- Techniques: detection and segmentation
- My publication: G. Chen, Y. Liu, N. Wergeles, Y. Shang, J. Sartwell, T. Thompson and A. Lewandowski. "Digital Image Vegetation Analysis with Machine Learning", In Proceedings of the 2017 International Conference on Robotics and Artificial Intelligence (ICRAI 2017), Association for Computing Machinery (ACM), New York, NY, USA, 6–10. DOI: 10.1145/3175603.3175611

In this work, we propose a computer vision-based approach to analyze the wild vegetation coverage. The vegetation coverage is estimated by the percentage of a reference

board covered by grass. Since the wild environment is uncontrolled, the grass and the reference board show different colors. The goal of the proposed approach is to accurately detect the reference board and segment the board versus grass. We adopt handcrafted features to localize the board and generate region proposals for local patches that are not covered by the grass. And then the segmentation is formulated as an one-class classification problem. A color model is generated based on the region proposals and predicts the pixel-level segmentation result. There are two main advantages of this approach. Firstly, because it automatically infers the color of the reference board at test time, it is robust to the lighting condition change. Secondly, it requires only a small set of training data and no pixel-level annotation is needed.

B) Tree Counting and Density Estimation Using Aerial Imagery

- Camera: Aerial camera
- Platform: offline
- Input: images
- Properties of objects: random pose, sometimes blurry and unclear in dark
- Techniques: counting by density estimation
- My publication: G. Chen and Y. Shang, "Transformer for Tree Counting in Aerial Images", *submitted to* Remote Sensing. 2021

In this work we propose an efficient density-based approach called density transformer, or DENT, for tree counting in aerial images. The model consists of a Multi-Receptive Field convolutional network (Multi-RF CNN), a transformer encoder, a Density Map Generator (DMG) and a tree counter. The Multi-RF CNN simutaneously extracts visual features from concentric receptive fields in different sizes. The transformer

models the interaction of the visual features extracted from different positions, filtering and transfering the useful contextual information across the features. The DMG generates the density map showing the spatial distribution of the trees. And the tree counter predicts the number of trees in the input image. We also collect the fully labeled dataset called Yosemite Tree Dataset. To the best of our knowledge, it is the largest public available common benchmark dataset of tree counting. We compare it with the famous and popular state-of-art methods on Yosemite Tree Dataset and an external cross-site dataset as well. DENT significantly outperforms almost all the other methods.

C) Fish Detection and Classification Using Boat Camera

- Camera: boat camera
- Platform: offline
- Input: images
- Properties of objects: numerous, dense
- Techniques: counting by density estimation
- My publication: G. Chen, P. Sun and Y. Shang, "Automatic Fish Classification System Using Deep Learning," 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI), 2017, pp. 24-29, DOI: 10.1109/ICTAI.2017.00016.

The goal of this work is to detect which species of fish appear on a fishing boat via the images captured from boat cameras of various angles. We propose an automatic fisheries classification system involving both instance-level classification and image-level classification. The instance-level classifier localizes the fish ignoring the other objects such as working crew and fishing gears. Since a captured fish can be placed

at a random position on boat and with a random pose, the instance-level classifier only focuses on the local region of the fish and normalize the pose to achieve better accuracy. In some circumstances, the individual fish can hardly be recognized. For example, when images are taken at night, they may have inaccurate colors. The camera can also be affected by splashed water or rain and output blurry images. In this case, the backup solution is to make a prediction based on the contextual information and prior knowledge. The image-level classifier learns to link the categorical information of fish and the whole scenario in the training set. At test time, a mechanism is designed to estimate how familiar the current scenario is and to balance the two classifiers. The proposed method achieved top 0.7% rank on the final leaderboard in the worldwide competition of "The Nature Conservancy Fisheries Monitoring" hosted by Kaggle.

D) Livestock Detection and Counting Using Inspection Robot

- Camera: fisheye lens on robot
- Platform: online, edge computing on embedded system
- Input: videos
- Properties of objects: deformable, crowded, moving
- Techniques: bottom-up detection, counting by tracking
- My publication: G. Chen, S. Shen, L. Wen, S. Luo and L. Bo, "Efficient Pig Counting in Crowds with Keypoints Tracking and Spatial-aware Temporal Response Filtering", 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 10052-10058, DOI: 10.1109/ICRA40945.2020.9197211.

This work is an edge computing robot system for counting pigs. Today, a typical pig house is too large for a single camera to cover, hence counting pigs using cameras

is a non-trivial task. We developed an inspection robot walking on the rail installed on the ceiling of the pig house. The robot scans the pig house using a fisheye lens. The number of the pigs is inferred based on the captured video. The inference model is deployed on the embedded system of the robot. To accurately localize the pigs we proposed a bottom-up keypoint detection method to deal with the crowded and deformable bodies of pigs, outperforming the existing state-of-art top-down detectors: The locations and the affinity fields are predicted using a fully convolutional network. Based on the locations and affinity, the keypoints are spatially associated. So that the keypoints belonging to the same instances are grouped. An then the instances are tracked across different frames. We also propose a Spatial-aware Temporal Response Filtering procedure to handle the missing instances, false alarms and the trajectories of the pigs. Finally, the proposed approach is compared with human volunteers. Our approach significantly outperform the human volunteers in term of counting accuracy.

Each kind of object will be introduced in an individual chapter in the rest of this dissertation.

1.3 Contributions

The main contributions of the works reported in this dissertation include:

1. A Digital Image Vegetation Analysis algorithm that automatically estimates the vegetation coverage. And it is robust to lighting condition change.

    - We propose a method to infer the position of the reference board directly from the HOG feature maps.
    - We propose a LBP+SVM procedure to generate coarse reference board segmentation proposals.
    - We propose a K-means based method to generate color model from the ex-

amplar regions of reference board at test time and refine the segmentation result to pixel level.

2. A novel density transformer, or DENT, significantly outperforming the other mainstream state-of-art methods for tree counting tasks.

    - We propose a surprisingly simple but efficient strategy to extract visual features from multiple receptive fields and improve the feature representation. The multiple receptive fields are implemented by jump connections from intermediate layers of a backbone CNN to the final layer.
    - We propose a transformer-based method to share contextual information across visual features corresponding to different positions.
    - We propose two decoders to decode the hidden states of the transformer to generate tree density maps and tree counts. The two decoders output results in different granularity with different computational costs but similar counting accuracy. Such design provides the user choices based on their time budget.

3. An automatic fish classification system robust to the random pose of fish, extreme lighting condition change and wet camera, achieving the top 1% solutions among the "The Nature Conservancy Fisheries Monitoring" competition.

    - We propose a two-branch framework to classify fish using the local region and context respectively.
    - We propose a detection, pose normalization and classification workflow to improve the classfication accuracy.
    - We propose an adaptive prediction average mechanism to estimate which branch works better and reweight the results from the two branches. The

estimation is made according to the detection scores and the familiarity of the context.

4. An inspection robot system for livestock counting, with a novel algorithm robust to the livestock and camera movement. The proposed method outperforms human competitors in pig counting experiments.

    - We propose a efficient bottom-up keypoint detection framework to detect crowded and overlapping livestocks.
    - We propose a fast CNN consisting of depthwise separable convolutional blocks to generate the heatmaps and the offset vector maps for keypoints. And it is efficient enough to run on a conventional embedded system.
    - We propose a novel spatial-aware temporal response filtering method to suppress the false positives and tracking failure and predict the count of the livestock.

Chapter 2

Vegetation Coverage Estimation from Images

2.1 Introduction

Digital image vegetation analysis (DIVA) is becoming an indispensable ground-based, large-scale, non-destructive technique for measuring vegetation structure [3]. DIVA provides a simple method for assessing landscape scale vegetation heterogeneity and can serve as a method of assessing wildlife habitat suitability with ramifications for determining habitat management success. DIVA techniques are intended to replace the laborious cut and weigh process and have been shown to produce surprisingly accurate results compared to other traditional visual obstructive methods [3, 4]. When combined with automated computer processes DIVA has the potential to become the principle method for evaluating vegetation characteristics in terrestrial ecosystems. Unfortunately, one major impediment to this goal is the lack of a standardized process of dealing with the inherent error in DIVA; i.e. variations in camera resolutions, lighting conditions and even ambient wind speeds can have a direct impact on DIVA results. A difference in variability will occur by location, investigator, time of day

Figure 2.1: The pipeline of the proposed method.

and even from image to image [5, 6]. In those cases where DIVA is semi-automated through software, these errors will be compounded by the use of arbitrary contrast thresholds and/or sets of highly variable user-specified pixel parameters [3]. The most basic uncertainty with DIVA is the misclassification of pixel values as vegetation as the relative contrast of the vegetation decreases against a known backdrop (photo-board). This misclassification or blurring places a finite limit on the detectability of the vegetation that can be visualized and it reduces the contrast of small features in the image.

In this work, we explore to identify the structure of the vegetation with automatic machine learning approach. The vegetation's structure is learnt by analyzing the coverage of vegetation on the blackboard. The algorithm can learn the vegetation from the training images. During the testing phase, the algorithm can predict the coverage of vegetation in digital images without human intervention. We developed our pipeline with two stages (Figure 2.1). In the first stage, we localize blackboard from the whole image. Our approach can find the pixel location of blackboard, as well as its width and height. In the second stage, given the blackboard region location, we further segment the image pixel of vegetation from blackboard.

The proposed approach is evaluated on a dataset collected in a real world field study. The experimental results demonstrate that the proposed method is robust to the color instability caused by the variation of lighting condition. On the test set, the success rate of the localization of blackboard is 93%. The error rate of coverage

estimation is 3%.

2.2 Related works

In literature, researchers are focusing on using different methods to collect and process vegetation data. In terms of collecting data, there are visual obstruction methods [7, 8] and imagery methods [3, 4]. Visual obstruction methods are common and are introduced first to quantify vegetation structures. Robel Pole [7] is one of the standard methods for visual obstruction. The author used a linear regression-based analysis to prove that the relationship between the mean of visual obstruction measurements and the weight of clipped vegetation are highly correlated. Coverboard [8] is used to read the vegetation density and showed it can be used to measure vegetation on any scale. However, visual obstruction methods mainly rely on a human observer. There is the potential for mistakes, based on different human criteria. It is also the least desirable method because of the amount of effort required to process which cannot be automated by a computer.

Recently, more research has shifted their focus on digital image based measurements. Previous DIVA techniques fall generally into 3 categories: arbitrary threshold classifications, human-based, and photo-training selection [3]. However, none is based off a machine learning approach, to the best of our knowledge. In [4], an image thresholding based vegetation measurement is proposed. In this arbitrary threshold method, the image is converted into a binary black and white image. Any pixel over a pre-defined and arbitrary threshold value is assigned one (vegetation) and zero (photo-board) otherwise. This method is highly affected by variations in light from both temporal and environmental factors such as shadowing and/or highlighting.

For the human-based selection [9] it is more accurate than arbitrary threshold methods since this is a fully supervised process. However, this method is based on

human criteria to determine what is vegetation, it may increase the error rate of detection. It is also time consuming compared to arbitrary threshold method.

The photo-training technique [10] is a semi-automatic albeit arbitrarily-defined training process. It required users to specify the parameters first, then the algorithm can run automatically. However, if the user specified values are only partially correct, in identifying vegetation, which is generally the case with thousands of images with significant temporal and environmental differences, then the error will likely be perpetuated to the full set of digital images. As a result, all the three categories approaches cannot run automatically, and require human intervention or decisions. Therefore, all three methods increase detection error rate and have a slow processing time.

In this work, we propose automatic localization and segmentation algorithms to improve all three detectability factors and significantly reduce arbitrarily defined criteria as well as human influences in variability. Our novelty includes: (1) Our approach is fully automatic and does not require any human involved actions. (2) Our approach not only takes into account color difference between vegetation area and backdrop, but also considers shape and texture information, which is more robust to lighting and weather changes. (3) Our approach is fast. It only takes 2-3 seconds to process one image and a few minutes to train hundreds of images on a single common CPU.

2.3 Proposed method

The goal of the proposed method is to design an efficient approach to segment the vegetation region on the blackboard from the rest of the image. To achieve the goal, the blackboard is localized on the image. And then an adaptive color model is applied to predict whether each pixel in the area of blackboard is covered by vegetation or not. We also expect to deal with several problems: First, human labeling is expensive, so a

learning algorithm which does not requires a large amount of labeled data is preferred. Second, the environmental conditions such as weather and lighting vary, hence any preset colors are not reliable to recognize the blackboard/vegetation.

2.3.1 Blackboard Localization

The work in this work is under an assumption that at least three edges (the top, left, and right edges) of the blackboard are visible in the image and the blackboard is standing upright. Following a heuristic that the gradient on the horizontal edge should be vertical and the gradient on the vertical edges should be horizontal. Histogram of Oriented Gradient (HOG) [11] features are utilized to localize the blackboard without supervised learning (and without labeled data). Instead of applying a sliding window like the traditional way to detect objects, the three edges can be detected separately to achieve a fast processing speed.

The blackboard localization approach is as follows:

(1) Convert the input image into gray-scale. We find separately processing each channel of image in RGB or HSV space in the following procedures improve the performance a little. But a gray-scale version of the input image works well.

(2) Compute the gradient values. The most common method is to apply the 1-D centered point discrete derivative mask in one or both of the horizontal and vertical directions. An easy way to acquire the gradient map is to do a convolution between the image and two filter $[-1, 0, 1]$ and $[-1, 0, 1]^\top$.

(3) Divide the image into small cells. In our program, the size of each cell is 12 pixels by 12 pixels.

(4) Each pixel within the cell casts a weighted vote for an orientation-based histogram constructed on the values found in the gradient computation. In our program, the number of orientation bins is 12.

(5) Concatenate the HOG features for each cell to get the HOG feature map for

the image. This feature map has a lower resolution and 12 channels. Each channel is corresponding to an arrange of orientation. Denote the channel i of the feature map as C_i and assume the C_0, C_3, C_6, C_9 represent the upward, leftward, downward and rightward components of the gradient map respectively. Then $C_3 + C_9$ indicates the if there is a strong vertical edge at each position. Ideally, the summation of the columns has two peaks which shows where the vertical side of the blackboard is. Hence in this work, the top 2 maxima are chosen. Similarly, the horizontal edge of the blackboard can also be localized.

(6) Since the aspect ratio of the blackboard is known, the outline of the entire blackboard can be inferred.

2.3.2 Segmentation at raw level

In the previous step, we get the location of the blackboard. The sub-image in the area of blackboard is divided into small cells. since the pattern for the blackboard and the vegetation are visually different. For the training set, we separate the blackboard and the vegetation. We can use a Local Binary Pattern (LBP) [10] to get the features of the blackboard and vegetation. We use these features to train a Support Vector Machine (SVM) and use this classifier to roughly separate the blackboard and the vegetation in the testing set. The higher the score the cell receives, the probability is higher the pixel is from the blackboard rather than the vegetation.

We use the following methods to build a color model for the corresponding picture:

1) The sub-image is cropped from the bounding box. We resize the sub-image to 600 by 900 pixels.

2) Divide the sub-image into small cells. In our program, the size of each cell is 60 by 60 pixels. Get the Local Binary Pattern (LBP) feature vector for each cell.

3) Train a SVM to classify the blackboard and the vegetation. Use the LBP feature vector for the images of the training set to train a SVM.

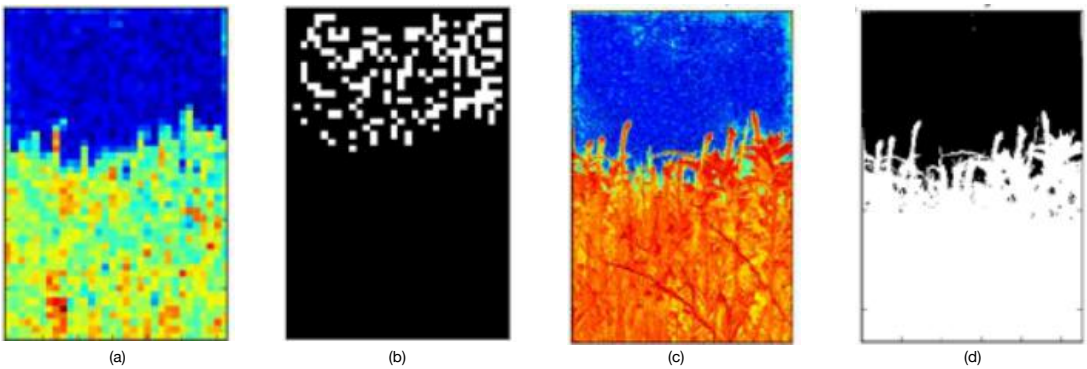


Figure 2.2: (a) The response map of LBP+SVM. (b) The top scored cells for blackboard. (c) The distance between each pixel to the K-means color model. (d) The segmentation result.

4) Use the SVM created in the previous step to classify the cells of the input image and get a score for each cell (A example is shown in Figure 2.2(a)). The score indicates how likely the part of blackboard in the cell is covered by vegetation.

2.3.3 Segmentation at pixel level

The RGB colors in the lowest-scored cells (Figure 2.2(b)) are clustered by using a K-Means model with 8 clusters. This color model is used to approximately simulate the distribution of the color of the blackboard. Only the lowest-scored cells are used to build the color model to make sure that the color model is not strongly intervened by pixel of vegetation. Given a pixel, the probability that it is covered by vegetation is measured by the distance of the color of the pixel to the nearest cluster center of the K-Means model (Figure 2.2(c)). By adjusting the parameters of the K-Means Color Model, we can get a better detection. In our program, we set the threshold as 27. This means if the distance of the color of this pixel is less than 27 to the nearest cluster, we say the pixel is blackboard, otherwise it is from vegetation. An example of the final segmentation result is shown in Figure 2.2(d).

2.3.4 Compute the Coverage of the Vegetation

We can calculate the number of the vegetation pixels because we already identify the pixels from the blackboard in the previous step. The total number of pixels in the entire blackboard area is also known. The difference of these two are the number of pixels from the vegetation. The steps for this procedure are outlined below:

1) In the sub-image, we scanned each line of pixels to compute the number of pixels N we detected in the vegetation. Since the size of the sub image is 600 x 900 pixels, we can get the coverage for each line which is equal to the number of pixels divided by 600, e.g. $N/600$. 2) The total coverage is equal to the number of pixels detected from vegetation X divided by the total amount of pixels in the sub-image, e.g. $N/(600 \times 900)$. Finally, we can calculate the coverage of the vegetation as shown in Figure 2.3. We plot the vegetation coverage overlay on the top of original image. We then plot the distribution of the vegetation coverage in terms of the height. In the distribution figure shown in Figure 2.3, the vertical axis is the height of the blackboard and the horizontal axis is the coverage. We calculate the total vegetation coverage percentage over the entire blackboard and display the results on the plot, which is located slightly above the x-axis, e.g. Coverage: 55.23%.

Figure 2.3: An example of calculated vegetation coverage.

2.4 Experiments

2.4.1 Dataset

We evaluate our proposed method on the dataset collected by the Missouri Department of Conservation (MDC) (See examples in Figure 2.4). This dataset contains 1,260 vegetation coverboard images. The resolution of the images are 5184×3456. The real size of the blackboard is 1m×1.5m. The picture is taken from 5 meters away from the board, therefore the blackboard will approximately be the same size in each image. In our experiment, we use 80% of the data as training and use 20% of the data to test the performance of our approach. During the training stage, we extract features and train the SVM model. During the testing phase, we evaluate our trained model on the new images to get the accuracy.

Figure 2.4: Example images for vegetation coverage estimation taken in Missouri

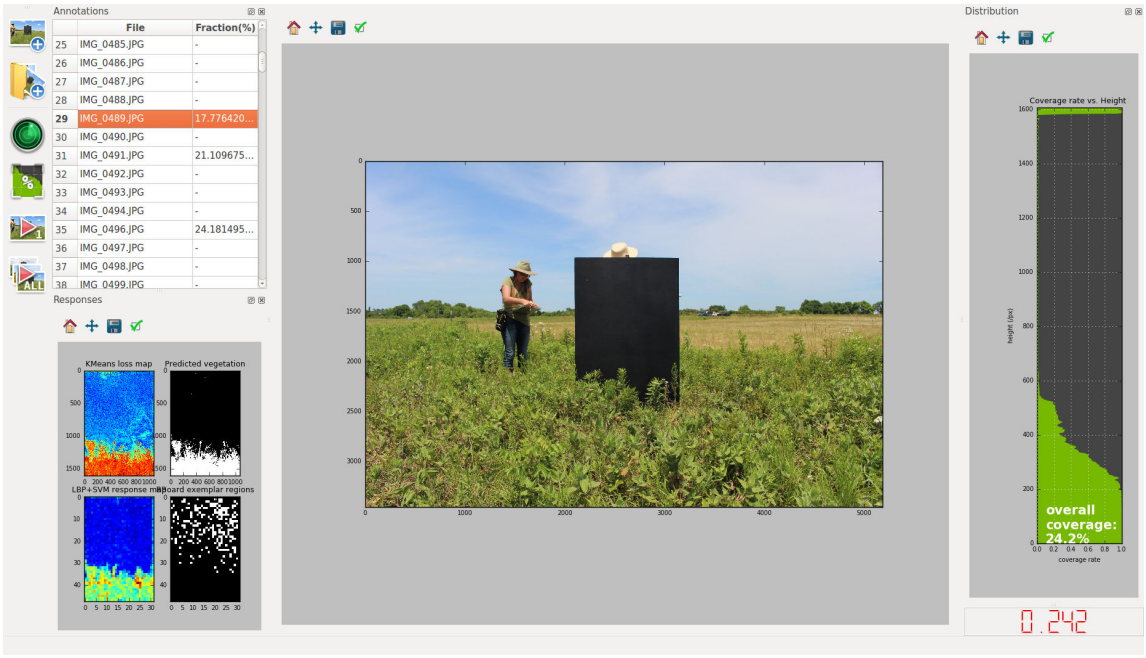


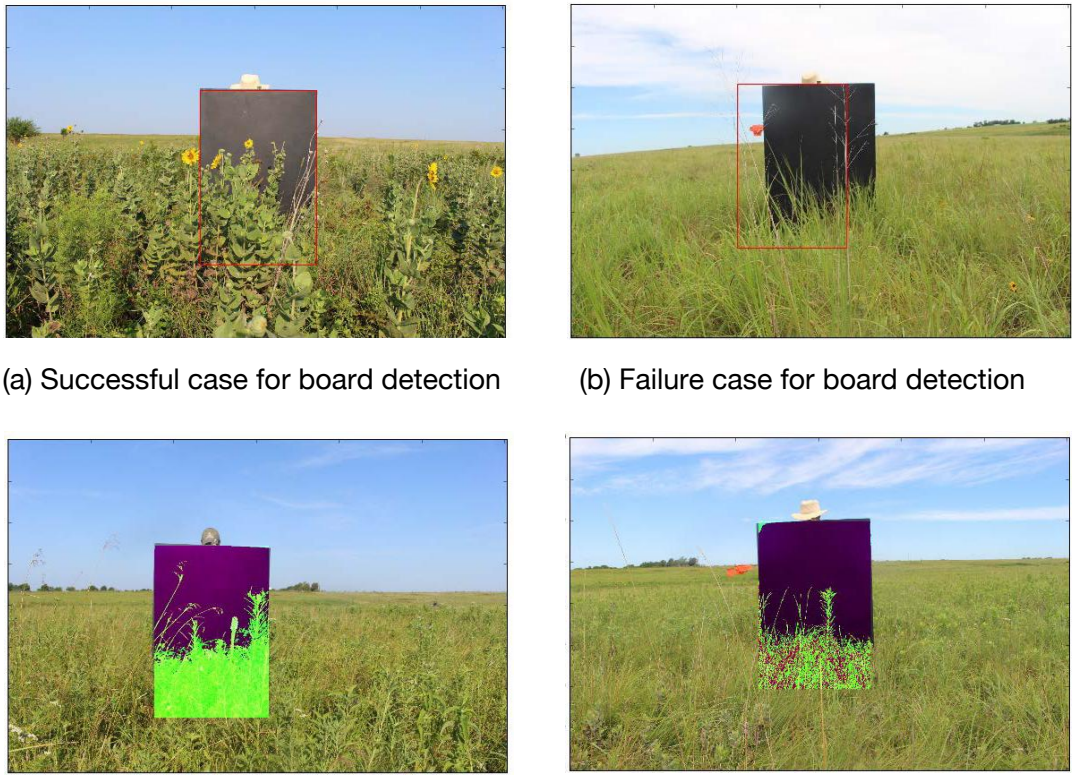
Figure 2.5: The graphical user interface of the software using the proposed method

2.4.2 Result

We developed a software with graphical user interface based on PyQt for batch processing the images (Figure 2.5). We report the success rate for the blackboard detection and mean absolute error (MAE) for the vegetation coverage analysis.

Results of Blackboard Localization

For our blackboard localization stage, we tested 200 images and the success rate is 93%. Examples of correctly detected blackboard are shown in Figure 2.6(a). A success case for board detection is defined as our detected bounding box can correctly and fully cover the blackboard in the image. We also demonstrate an example of wrongly detected blackboards in Figure 2.6(b). We find the reason for the blackboard to be detected incorrectly is due to the large number of vertical gradient generated by an object standing upright. For example, in Figure 2.6(b), there is a red balloon on the left background of the blackboard.



(a) Successful case for board detection

(b) Failure case for board detection

(c) Successful case for segmentation

(d) Failure case for segmentation

Figure 2.6: Example results of the proposed method

Results of Vegetation Coverage Analysis

For the grass detection stage and coverage computation, after analyzing our result the estimated mean absolute error rate is approximately 3%. We also show a success case and a failure case in Figure 2.6(c) and (d). In the failure case, a part of the grass has very similar color with the reflection of the board. This part of grass are misclassified.

2.5 Conclusion

In this work we proposed an approach for digital image vegetation analysis based on handcrafted features and traditional machine learning methods. Experimental results are reported to shows its performance. The two-stage segmentation method proposed in this work effectively predict the coverage and the density of vegetation even if only limited amount of labeled training data is available. It is worth noting that in recent years algorithms based on deep learning show their state-of-art performance in many computer vision tasks. In our future work, we will apply deep neural networks to reinforce or replace the handcrafted features used in the proposed method. What's more, since training a deep neural network in a supervised fashion usually requires a large amount of labeled data and hand-labeling is expensive, we will also investigate if the proposed method can be used to regularize the training or generate pseudo labels as the targets to supervise the training.

Chapter 3

Tree Counting and Density Estimation Using Aerial Imagery

3.1 Introduction

The density and distribution of forest trees is important information for ecologists to understand the ecosystem in certain regions. For example, the environmental effect of deforestation or forest fires may be estimated based on the number of lost trees and their location. In the recent decades, forest trees are often counted with the help of aerial imagery. Since manually counting the trees from images can still be time consuming, automatic tree counting algorithms have been developed to lower the time cost. With the breakthrough of deep learning in the recent decade, deep neural networks (DNNs) made unprecedented progress in computer vision tasks such as image classification [12, 13, 14, 15, 16] and object detection [17, 18, 19, 20, 21, 22]. DNNs also become widely popular for object counting. A fashion of object counting methods using DNNs is detection based, i.e. to localize each individual object of interest first and then get the total number. So far this is the mainstream of the published tree counting methods [23, 24, 25, 26, 27, 28, 29]. Another fashion for object

counting is to regress the density of objects in the image using DNNs. The success is reported in a growing number of works for crowd (people) counting [30, 31, 32, 33, 34]. However, the competitiveness of density based algorithms for tree counting are not sufficiently explored as they are applied in much fewer published works with limited comparative evaluation [35, 36].

We would like to exploit DNN regressors and propose a method for tree counting called density transformer or DENT, which consists of a multi-receptive field (Multi-RF) convolutional neural network (CNN), a transformer and two heads: Density Map Generator (DMG) and tree counter. The Multi-RF CNN extracts visual features from images with multiple receptive fields of different sizes simutatinously, perceiving the patterns of both the local patch and the concentric context. The transformer models the pair-wise relations between the visual features and filters the contextual visual information sharing across different positions using attention mechanism. The two heads, the DMG and the tree counter, parallelly decode the hidden states of the transformer to generate the tree density map at different granularity levels. If a relatively coarse tree map already meets the demand, the DMG can be detached after training to save the inference time. The whole model of DENT is end-to-end trainable. An example input image and the tree density map generated using DENT is illustrated in Figure 3.1.

We also found few benchmark dataset publicly available for tree counting tasks. To the best of our knowledge, the existing works report their performance tested on either private data or a small subset (<10k trees) of dataset made for other tasks [27, 37]. The lack of a common benchmark makes the fair comparison across different methods infeasible. Hence we release an fully labeled dataset called Yosemite Tree Dataset, which has a $\sim$10km^2 rectangular study area with $\sim$100k trees whose coordinates are annotated. It is suitable for evaluating not only the performance of tree counting algorithms but also the counting error versus the area of region of interest.

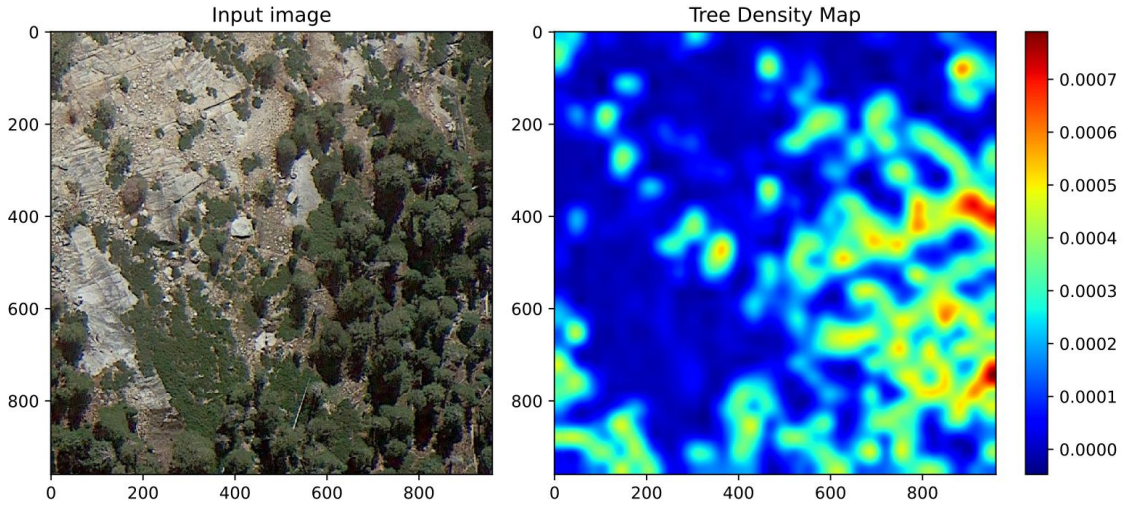


Figure 3.1: An example tree density map generated using DENT (with interpolation)

To demonstrate the effectiveness of DENT, we compare DENT with the existing state-of-art methods of different types, including fully convolutional networks regressors and detectors. The methods are evaluated on the Yosemite Tree Dataset and the cross-site NeonTreeEvaluation [27] Dataset. On both of them DENT achieves very competitive results in the experiments and significantly outperforms most of the other methods.

The main contributions of this work include two parts. The first part is the novel end-to-end approach for tree counting, using a efficient multi-receptive field CNN architecture for visual feature representation, a transformer for modeling the pairwise interaction between the visual features, and two heads for outputs at different granularity and time costs. The second part is the Yosemite Tree Dataset as common benchmark for tree counting.

3.2 Related works

3.2.1 Transformers

Transformers [38] are attention-based deep learning models. They are initially proposed in the area of natural language processing (NLP). The input of a transformer is an embedding sequence. Pair-wise interaction between any two elements of the sequence is formulated by the transformers. The output corresponding to an element is the aggregated from all the elements of the sequence with different weights depend on their relationship. In this work, we adopt transformer to enhance the CNN features, by selectively transfering contextual information among different elements.

3.2.2 Density estimation

Learning density maps using deep CNNs is a trend of crowd counting. On this trend, the counting task is formulated as a regression program. The CNNs are trained to predict the density distribution over the input image. But the location of each individual object is not explicitly predicted. When the object is crowded, the representation of density map is relatively robust. In the existing works, different network architectures are tried. MCNN [30] uses a multi-column network with different filter sizes for object at different scales. The features from all the columns are fused to predict the crowd density map. SwitchCNN [31] has an additional classifier to predict and switch to the best column for the given image. CSRNet [32] generates high-resolution density map. It is composed of a front-end CNN for feature extraction and an back-end CNN for map generation. It uses dilated convolution instead of pooling or transposed convolution to reduce the computational complexity. CANNet [34] encodes contextual information at different scale by subtract local average from the feature maps.

For tree counting tasks, an AlexNet [12] regressor is applied in [35]. In the work

of [36], AlexNet [12], VGGNet [13] and a UNet [39] are evaluated and compared; And the UNet achieves the best performance. In this work, we follow the paradigm of the density estimation problem and fomulate tree counting as regression problem.

3.2.3 Object detection

The purpose of object detection is to localize each object of interest in the image. Traditional detectors explicitly use a sliding window of predefined size to scan each position of the image [40, 41, 42, 43, 44]. These early works usually extract hand-crafted features such as HOG [42] and SIFT [45]. These feature are finally fed to a classifier such as a support vector machine (SVM) or a neural network. Modern detectors make use of the powerful features from deep convolutional neural networks (CNNs) pretrained on large-scale classification datasets [12]. These detectors adopt different strategies to generate bounding boxes for objects using CNNs. Faster-RCNN [17], RetinaNet [19] and YOLO [20] predefine a set of anchors and formulate the detection into two sub-problems: classification of the subimage in each anchor and regression of the offset between the ground truth box and the anchor. CenterNet [21] treats the center of an object as a keypoint and regress the width and height. RetinaNet, YOLO and CenterNet infer the results in one shot. In contrast, Faster-RCNN recomputes the features for classification after the generation of region proposals.

Tree counting by detection So far, most of the published works of tree counting algorithms are based on detection. These methods can be categorized into tree groups:

1. Explicitly using sliding windows. The very early works in [46, 47, 48, 49] synthesize the expected appearance of trees and generate a template based on the prior knowledge. The likelihood of the existence of a tree in a sliding window is estimated by the correlation between the tree template and the image patch in the window. However, the templates oversimplify the diverse appearance of trees in real world. Later works use hand-crafted features plus classifiers. For example, a feature

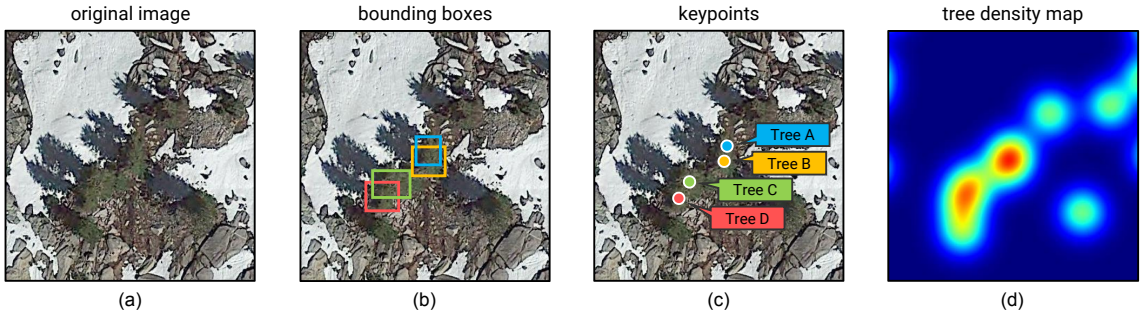


Figure 3.2: Comparison among different types of annotations for tree counting algorithms. (a) The original image. Some tree crowns are overlapping. It is difficult for human to detemine the borders of the trees. But from the shadows we can roughly localize each of them. (b) Bounding boxes, suppose to contain the whole objects using rectangles. (c) Keypoints, usually represented in 2D coordinates. (d) Tree density map, a heatmap indicating the spatial distribution of trees.

descriptor using circular autocorrelation is designed to detect the shape of palm tree in [50]. The goal in [51] is also to detect palm tree, but the descriptor used is HOG [42]. While [24] and [52] use CNNs to recognize palm trees in sliding window to learn features automatically. TS-CNNs [53] has two sliding windows of different sizes, each has a AlexNet classifier. One is to recognize the pattern of trees, the other one is to suppress the false positives according to the spatial distribution of the surrounding objects.

2. Fully convolutional classifiers are equivalent of sliding window CNN classifiers but with better computational efficiency. U-Net [39] and DenseNet [54] are used to predict the confidence maps of tree in [55] and [56]. The peaks on the confidence maps are considered as the final prediction.

3. Modern CNN detectors like Faster RCNN [17], SSD [18], RetinaNet [19] and YOLOv3 [20] have state-of-art localization performance in general object detection tasks. These approaches are also applied for tree detection in [25, 26, 27, 28, 29].

Counting trees in aerial images using detectors is straightforward but with some disadvantages, especially when the trees are dense and crowded. Firstly the representation of overlapping trees may be ambiguous for detectors at test time. A

typical detector usually outputs an excessive number of initial boxes and applys Non-Maximum Suppression (NMS) to select the best ones. The basic idea of NMS is to pair-wisely check the Intersection over Union (IoU) of every two proposal boxes, and remove the one with lower detection score when their IoU is higher than a preset threshold (typically 0.45 or 0.50). For tree counting, it is often the case that two correct boxes have high IoU. An example case is shown in Figure 3.2(b). In this case it is very likely that the NMS procedure will remove either of the blue box and the yellow box and cause an underestimation of tree count. Secondly, the threshold for the detection score directly effects the predicted tree count. Deliberately tuning the threshold requires extra effort. Thirdly, bounding boxes are relatively expensive to label. The labelers need to determine the width and the height of the boxes. It is often difficult when the trees are overlapping.

3.3 Methodology

The architecture of the DENT model is illustrated in Figure 3.3. It contains four main components: a Multi-Receptive Field convolutional network (Multi-RF CNN) to compute a feature map over an input image, a transformer encoder to model the interaction of features extracted from different positions, a Density Map Generator (DMG) to predict the density of the trees and a counter to regress the number of trees in the image.

Starting from a RGB aerial image $I \in \mathbb{R}^{3\times H_0\times W_0}$, the Multi-RF CNN generates a low resolution feature map $f_{\text{CNN}} \in \mathbb{R}^{C\times H\times W}$, where C is the number of output channels, and in this work $H = \frac{H_0}{32}$ and $W = \frac{W_0}{32}$. The feature map is projected using a trainable linear transform to generate $f_{\text{visual}} \in \mathbb{R}^{d_{\text{model}}\times H\times W}$, where d_{model} is the dimension of the hidden states of the transformer encoder. For convenient, it can also be reshaped and represented in a sequence form: $f_{\text{visual}} = [f_0, f_1, ..., f_L]$ where

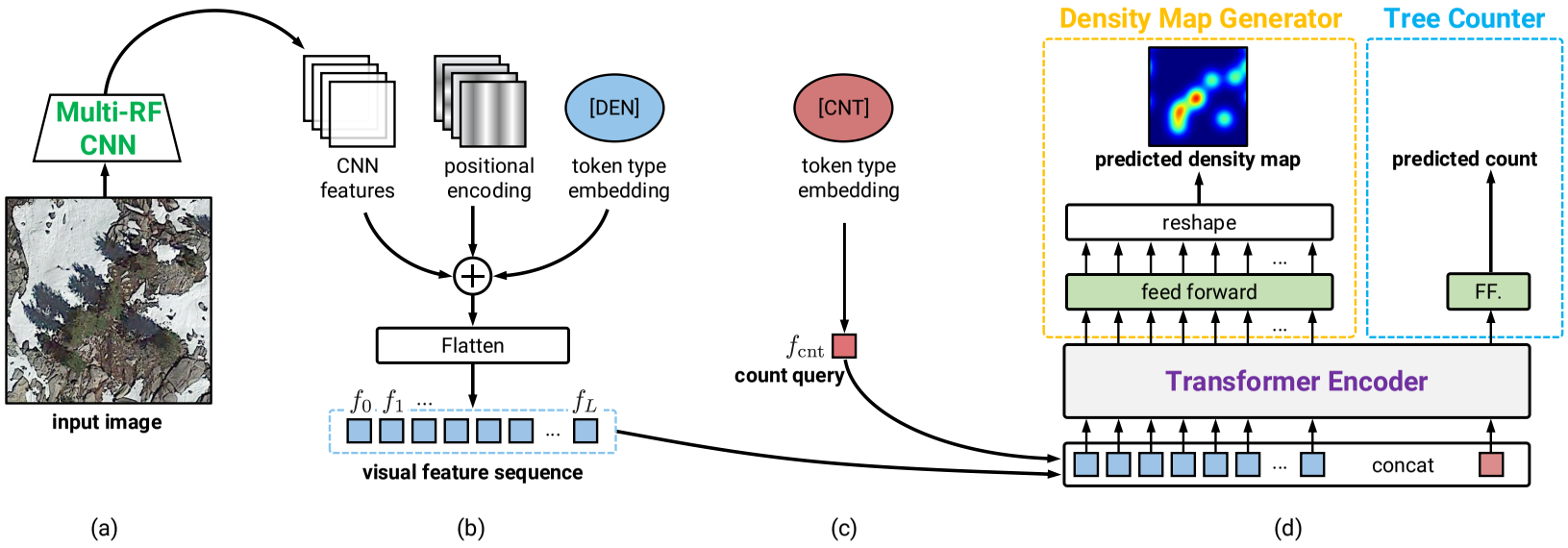


Figure 3.3: The architecture of the proposed **DENT** Network. (a) Visual features are extracted using a Multi-Receptive Field convolutional network (Multi-RF CNN). (b) Positional encoding and token type embedding are applied to the visual features. The visual features are flattened as a sequence $[f_0, f_1, ..., f_L]$. (c) f_{cnt}, the embedding of `CNT` token, works as a count query. (d) The visual feature sequence and the count query are concatenated as the input of the transformer encoder. On top of the transformer encoder there are two heads: A Density Map Generator (DMG) predicts the density of objects at different positions. A counter predicts the count of the objects in the whole input image.

$L = HW$ and $f_i \in \mathbb{R}^{d_{\text{model}}}$. Since each f_i is corresponding to a certain position p_i in the image, we use it to estimate the tree density at p_i. We also use a special embedding $f_{\text{cnt}} \in \mathbb{R}^{d_{\text{model}}}$ to query the number of trees in the image. The transformer encoder selectively transfers the information across $f_0 \sim f_L$ and f_{cnt}. The final hidden state of the transformer are decoded by the DMG and the tree counter. And then the DMG generates a density map $D \in \mathbb{R}^{H \times W}$. Meanwhile, the tree counter outputs the number of trees $\hat{z} \in \mathbb{R}$. The details of the components are discussed in the following sections.

3.3.1 Multi-Receptive Field network

Inspired by the *macula* of human retina, we extract feature representation from each position of image using multiple receptive fields, based on the intuitive assumptions: A wide receptive field of CNN covers a large area of image containing rich contextual infomation. On the other hand, a narrow one focuses on the details in a small region of interest without distracted by the surrounding objects.

Early works in MCNN [30] and SwitchCNN [31] control the receptive fields by desiging multi-column networks with different convolutional kernel sizes. We argue that such strategy has limitations: Firstly, using these methods it is not easy to implement a small receptive field on much deeper networks, because generally the receptive field is enlarged quickly with the depth of network increased. Modern deep networks usually has large receptive fields. For example, a VGG16[13] has a receptive field of 212×212 while a ResNet50 [14] has a receptive field of 483×483 [57]. Secondly, the widely used pretrained off-the-shelf models cannot be reused. Searching for the optimal architecture and pretraining takes extra effort. To avoid these limitations, We use off-the-shelf network as a backbone and add jump connections to its early layers to implement small receptive fields.

We proposed Multi-Receptive Field convolutional network (Multi-RF CNN) as

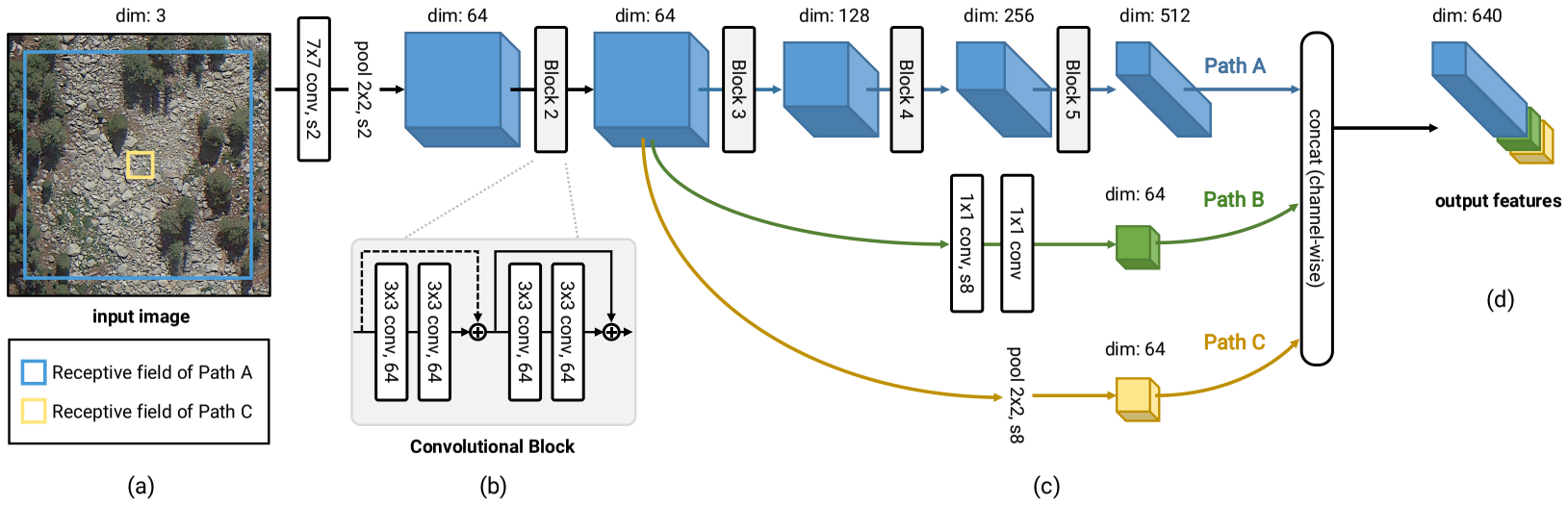


Figure 3.4: Architecture of the proposed Multi-Receptive Field network. This CNN has three different paths to output feature maps, each has its own receptive field. (a) The receptive fields of Path A and C are visualized as boxes. The receptive field of Path B is similar to A's and omitted in this figure. (b) The backbone is built using residual convolutional block as shown. Each block contains four convolutional layers having the same number of output dimensions (channels). There stride of the first convolutional layer could be either 1 (Block 2) or 2 (Block 3∼5). There are two jump connections in each block. The first jump connection (visualized in dotted arrow) project the input features to a higher dimension space if necessary. (c) The three paths diverge from Block 2. Path B contains two 1x1 conv. layers. Path C contains only an average pooling layer. (d) The output feature maps are concatenated along the channel axis to compose the final output.

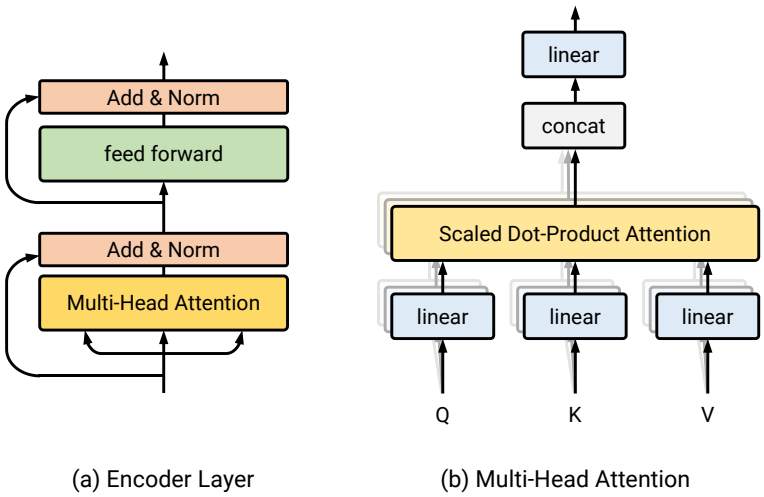


Figure 3.5: (a) The architecture of an transformer encoder layer. It contains a multi-head attention sublayer and a feed forward sublayer. Each sublayer has a residual connection and the output is processed by layer normalization [1]. (b) The architecture of the multi-head attention sublayer. Matrices Q, K and V are projected using multiple groups of linear projections parallelly. In each group, the projected Q, K and V generate ouput using scaled dot-product attention. The attention outputs from different groups are concatenated and linearly projected as the final output. All the linear projections mentioned are learned at training phase.

depicted in Figure 3.4. Specificially, the network contains a vanilla ResNet18 and two extra paths added on the convolutional Block 2. We refer the original path of ResNet18 from Block 2 (i.e. Block 3∼5) as Path A. Path B consists of two 1×1 convolutional layers. Path C is simply an average pooling layer. The stride of Path B and C is 8. The receptive fields of the three paths are naturally different, as 466×466, 43×43 and 47×47 respectively. Offsets are also applied on the input of Path B and C to ensure that the output feature maps from the three paths are center-aligned. These feature maps are concatenated along the channel axis as the final output. Although the architecture of Multi-RF CNN is surprisingly simple, we observe that it outperforms the vanilla ResNet18 in our experiments.

3.3.2 Transformer encoder

We exploit the self-attention mechanism of transformer [38] to model two types of interactions: those between the visual features extracted at different positions, and those between the visual features and the counting query. In this section we introduce the transformer encoder and discuss the two types of interactions.

Architecture

We use only the encoder part of a standard transformer. The encoder contains a group of stacked encoder layers. By default, number of encoder layers is 2 in this work. Each encoder layer (Figure 3.5(a)) has identical structure but its unshared weights. The attention mechanism takes effect in the multi-head attention sublayer (Figure 3.5(b)), where the core function is *scaled dot-product attention*. Given a query matrix $Q \in \mathbb{R}^{L_q \times d_k}$, a key matrix $K \in \mathbb{R}^{L_k \times d_k}$ and a value matrix $K \in \mathbb{R}^{L_k \times d_v}$, the scaled dot-product attention is defined as following:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{3.1}$$

The multi-head attention can be defined as:

$$\text{MultiHead}(Q, K, V) = \text{concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{3.2}$$

where h is the total number of heads, and

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{3.3}$$

where the projection matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{\text{model}}}$ are learnable at training stage. We will omit the other details about transformer, since the encoder we used is almost the same with the original. We refer the readers to [38] for the details.

Interaction between visual features

Contextual information is essential for density estimation. It can be extracted by convolutional networks in their receptive fields as discussed in Section 3.3.1. The interaction in a convolutional network occur only between the convolutional kernels and the previous-layer feature maps. As a supplement, we exploit the self-attention mechanism to realize the pair-wise interaction between features at different positions. The attention score for feature vector f_i on another feature f_j can be roughly defined as

$$a_{ij} = f_i W^Q (f_j W^K)^\top / \sqrt{d_k} \tag{3.4}$$

The contextual information collected by f_i can be defined as

$$o_i = \text{softmax}(a_{i\cdot})(f_{\text{visual}} W^V)^\top \tag{3.5}$$

Equation 3.4 and 3.5 are equivalent with an individual head in the multi-head attention mechanism when $Q = K = V = f_{\text{visual}}$.

However, Equation 3.4 and 3.5 is permutation-invariant and any positional information is ignored. Hence we add a 2D version of positional encodings [38, 58, 22] to the visual features before feeding the transformer encoder:

$$\begin{aligned}
PE(x,y)_{4i+0} &= \sin(x/10000^{4i/d_{\text{model}}}) \\
PE(x,y)_{4i+1} &= \cos(x/10000^{4i/d_{\text{model}}}) \\
PE(x,y)_{4i+2} &= \sin(y/10000^{4i/d_{\text{model}}}) \\
PE(x,y)_{4i+3} &= \cos(y/10000^{4i/d_{\text{model}}})
\end{aligned} \tag{3.6}$$

where (x, y) is the 2D position on the feature maps and i is the dimension.

Interaction between visual features and counting query

Inspired by the [CLS] token used in BERT [59], we also introduce a token [CNT] appended to the end of the input sequence of transformer encoder (Figure 3.3(c)). The corresponding token type embedding is f_{cnt}. Hence the input of the transformer encoder is $[f_0, f_1, f_2, ...f_L, f_{\text{cnt}}]$. The hidden state of the transformer corresponding to the [CNT] token represents the aggregate embedding of the sequence and serves as a global context for tree counting. In contrast, each visual feature vector is corresponding to a patch of the image and used to estimate the local tree density. For convenience sake, these visual feature vectors are also refered as [DEN] tokens in this work. To differentiate these two types of tokens, we also apply an token type embedding $f_{\text{den}}$for the DEN tokens (Figure 3.3(b)). The application of f_{den} can be seen as a in-place self-add operation: $f_i \mathrel{+}= f_{\text{den}}$. Specifically, $f_{\text{cnt}}, f_{\text{den}} \in \mathbb{R}^{d_{\text{model}}}$. And both f_{cnt} and f_{den} are learnable parameters at training time. The usage of the two token type embeddings are inspired by [59], where segment embeddings are used for different sentences, and [60], where token type embeddings are used for visual features versus textual features.

3.3.3 Density Map Generator (DMG)

The Density Map Generator is a fully connected feed forward network followed by a reshape operation. The feed forward network takes the final hidden state of transformer corresponding to each [DEN] token to predict the tree density. The output sequence for all [DEN] tokens is reshaped into a 2D map, which is the predicted tree density map.

Tree density map A tree density map (Figure 3.2(d)) represents the spatial distribution of trees in the image. The ground truth tree density map can be generated from the keypoint annotations of the trees (Figure 3.2(c)). Given an image I, denote $p_i = (x_i, y_i)$ is the location of the ith tree and z is the tree count. The original

annotation map is generated as

$$A(p) = \sum_{i=1}^{z} \delta(p - p_i) \tag{3.7}$$

where δ is the delta function. Following the works for crowd counting [32, 34, 30, 31], the ground truth tree density map D^{gt} is generated from the annotation map convolved by a Gaussian kernel:[1]

$$D^{\text{gt}} = A * G_\sigma \tag{3.8}$$

where $G_\sigma(x)$ is a 2D Gaussian kernel with standard deviation σ:

$$G_\sigma(p) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{\|p\|^2}{2\sigma^2}\right) \tag{3.9}$$

Denote the predicted density map is $D(p; I, \theta)$, where θ stands for the parameters of DENT. The loss of DMG is Mean Squared Error (*MSE*):

$$L_{\text{DMG}} = \frac{1}{BHW} \sum_{p} \sum_{i=1}^{B} \left(D(p; I_i, \theta) - D_i^{\text{gt}}\right)^2 \tag{3.10}$$

where B is the batch size; H, W are the height and width of the density map.

Density-based counting At test stage, the estimated counts of the trees $\hat{z}_R$ in a region of interest R is given by the integral of the tree density map:

$$\hat{z}_R = \sum_{p \in R} D(p; I, \theta) \tag{3.11}$$

[1] In practice, the model learns an $H \times W$ tree density map, which is a sum-pooled version of the $H_0 \times W_0$ density map.

because when $R^2 \gg \sigma^2$, we have

$$z_R = \sum_{p \in R} A(p) \approx \sum_{x \in R} D^{\text{gt}}(p) \tag{3.12}$$

3.3.4 Tree counter

The tree counter of DENT is a feed forward network that decode the transformer output corresponding to the `[CLS]` token. The target of the network is the tree count normalized by the number of the `[DEN]` tokens, i.e. the average of the density map:

$$\frac{z}{L} = \frac{z}{HW} \tag{3.13}$$

This network is also trained using *MSE* loss. We found the normalization helps the imbalance of losses for the tree counter and DMG. Denoting $c(I, \theta)$ as the output of the tree counter, the loss of the tree counter is

$$L_{\text{CNT}} = \frac{1}{B} \sum_{i=1}^{B} \left(c(I_i, \theta) - \frac{z_i}{L} \right)^2 \tag{3.14}$$

The predicted tree number is

$$\hat{z} = c(I_i, \theta) L \tag{3.15}$$

The tree counter is a relatively lightweight head of DENT compared with DMG. Since the tree counter gives predicted tree count for each $H \times W$ area in the study area, the predictions over the whole study area can also be seen as a coarse density map. If a more refined density map is not demanded, the DMG can be pruned after training. And then the computational complexity of the Dot-Product Attention in the top encoder layer is reduced from $O(L^2 \cdot d_{\text{model}})$ to $O(L \cdot d_{\text{model}})$, because the interaction between `[DEN]` tokens in that layer is no longer needed. Examples of the

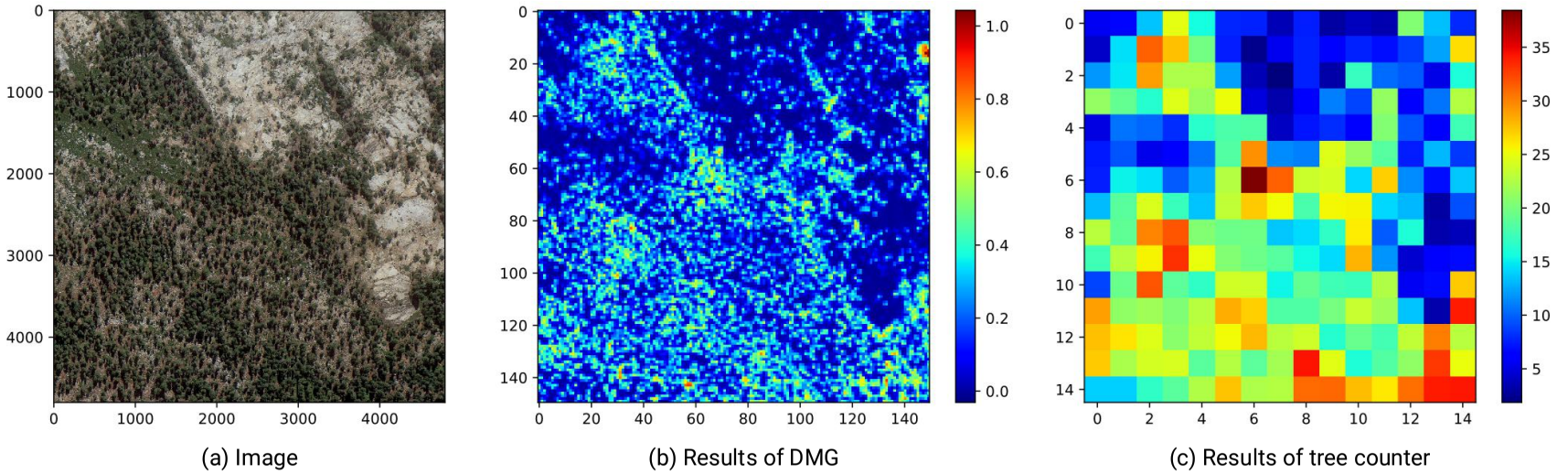


Figure 3.6: A comparison between the granularities of the DMG and the tree counter. (a) An example image from a 4800×4800 region, which is 566m×566m in real world. (b) The corresponding 150×150 desity map generated by DMG. (c) The corresponding 15×15 coarser density map generated using the tree counter.

density maps generated by a DMG and a tree counter are shown for comparison in Figure 3.6.

3.4 Datasets

3.4.1 Yosemite Tree Dataset

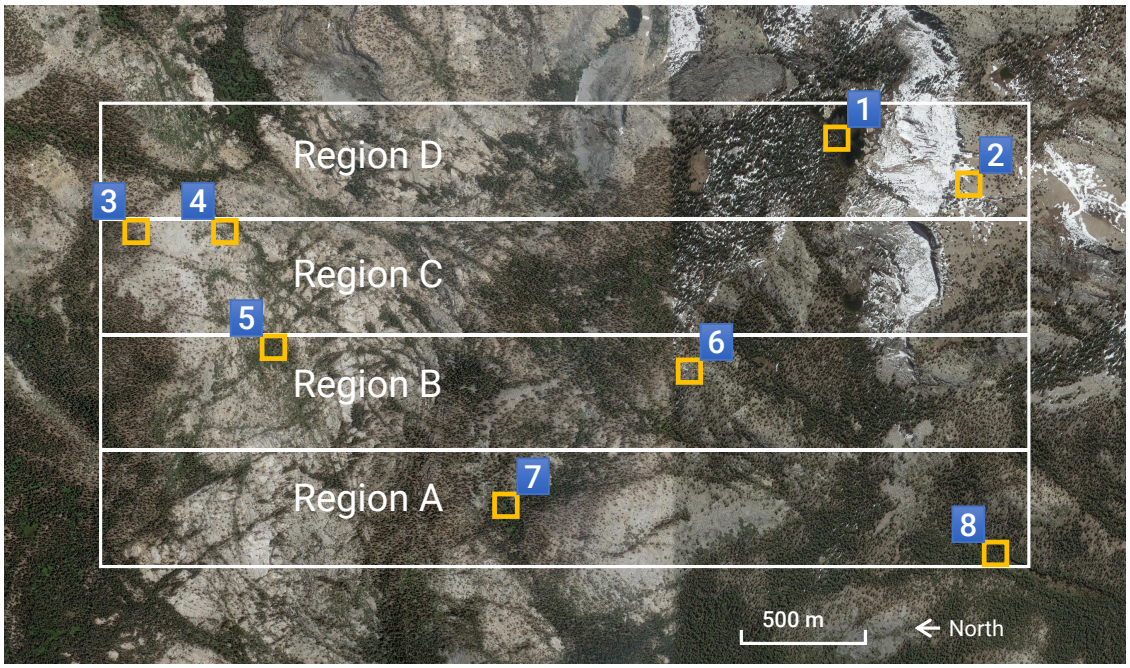
We choose a rectangular study area in the Yosemite national park and build a benchmark dataset for tree counting based on RGB aerial images. (Figure 3.7) The images are collected via Google Maps at 11.8cm/pixel resolution and stitched together. The study area is 2262.5m×4525.1m in the real world and 19200×38400 pixels in the image. Inside the study area, the position of each individual tree is manually labeled. The total number of labeled trees is 98,949.

We split the study area into four regions A, B, C and D of the same size. (Figure 3.7(a)) Region B and D are used as training set, and Region A and C as test set. To evaluate the accuracy of different tree counting algorithms, we further divide the study area into small non-overlapping square blocks. The counting errors in different blocks are supposed to calculated seperately. And the statistics of the errors are used as the metrics. Different block sizes can be used to analyze the accuracy versus the size of region of interest, for example 960×960 (Figure 3.7(b)) and 4800×4800.

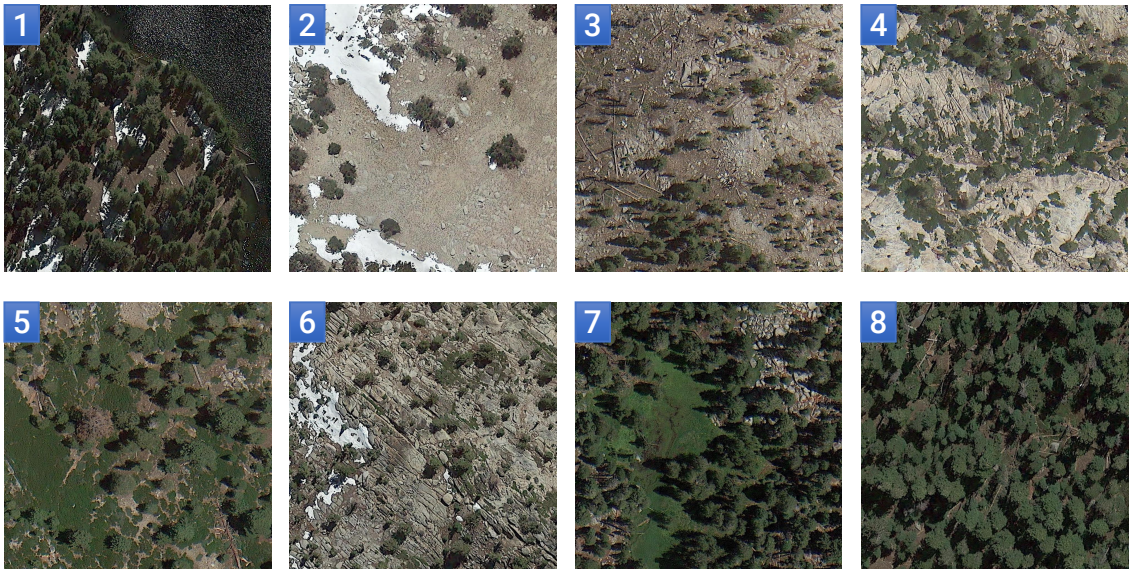
To better demonstrate ground truth distribution of the tree counts versus the block size, histograms are shown in Figure 3.8.

3.4.2 NeonTreeEvaluation Dataset

We also evaluate the models using NeonTreeEvaluation Dataset [2], which is collected from 22 sites across the United States by multiple types of sensors. The forest types vary in different sites. (Examples are shown in Figure 3.9.) In this work, we only



(a)



(b)

Figure 3.7: (a) The study area of the Yosemite Tree Dataset, centered at Latitude 37.854, Longitude -119.548. The study area consists of four rectangular regions A, B, C and D of the same size. Each region is an 565.6m×4525.1m subarea, corresponding to 4800×38400 pixels in the dataset. (b) Example images cropped from different locations of the dataset. These examples show the variance of the land covers, the directions of light, and the sizes and the shapes of the trees. The actual size of each example shown is 113.1m×113.1m, corresponding to 960×960 pixels in the dataset.

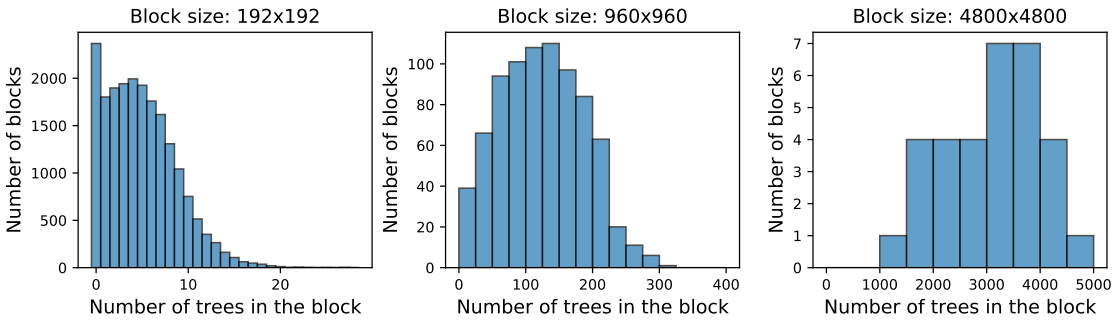


Figure 3.8: Histograms of tree counts of Yosemite Tree Dataset.

use the fully labeled RGB data, as following: (1) A test set of 194 images containing 6,634 annotated trees. The size of each image is 400×400 pixels and corresponding to a 40m×40m region in the real word. (2) A training set including 15 much larger images, containing 17,790 annotated trees. We crop them into 3,395 400×400 training images as consistent with the test images.

3.5 Experiments

3.5.1 Evaluation metric

By following the works for crowd density estimation, we evaluate different methods for tree counting using Mean absolute error (MAE) and Root Mean Squared Error ($RMSE$), which are defined as following:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|z_i - \hat{z}_i|\,,\ RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(z_i - \hat{z}_i)^2} \tag{3.16}$$

where N is the total number of blocks of in test set, z_i denotes the true number of trees in the ith block, and $\hat{z}_i$ is the predicted number of trees in the ith block inferred by algorithms. For the NeonTreeEvaluation Dataset, a block is simply a test image. For the Yosemite Tree Dataset, we set the block size to 960×960 and 4800×4800 and

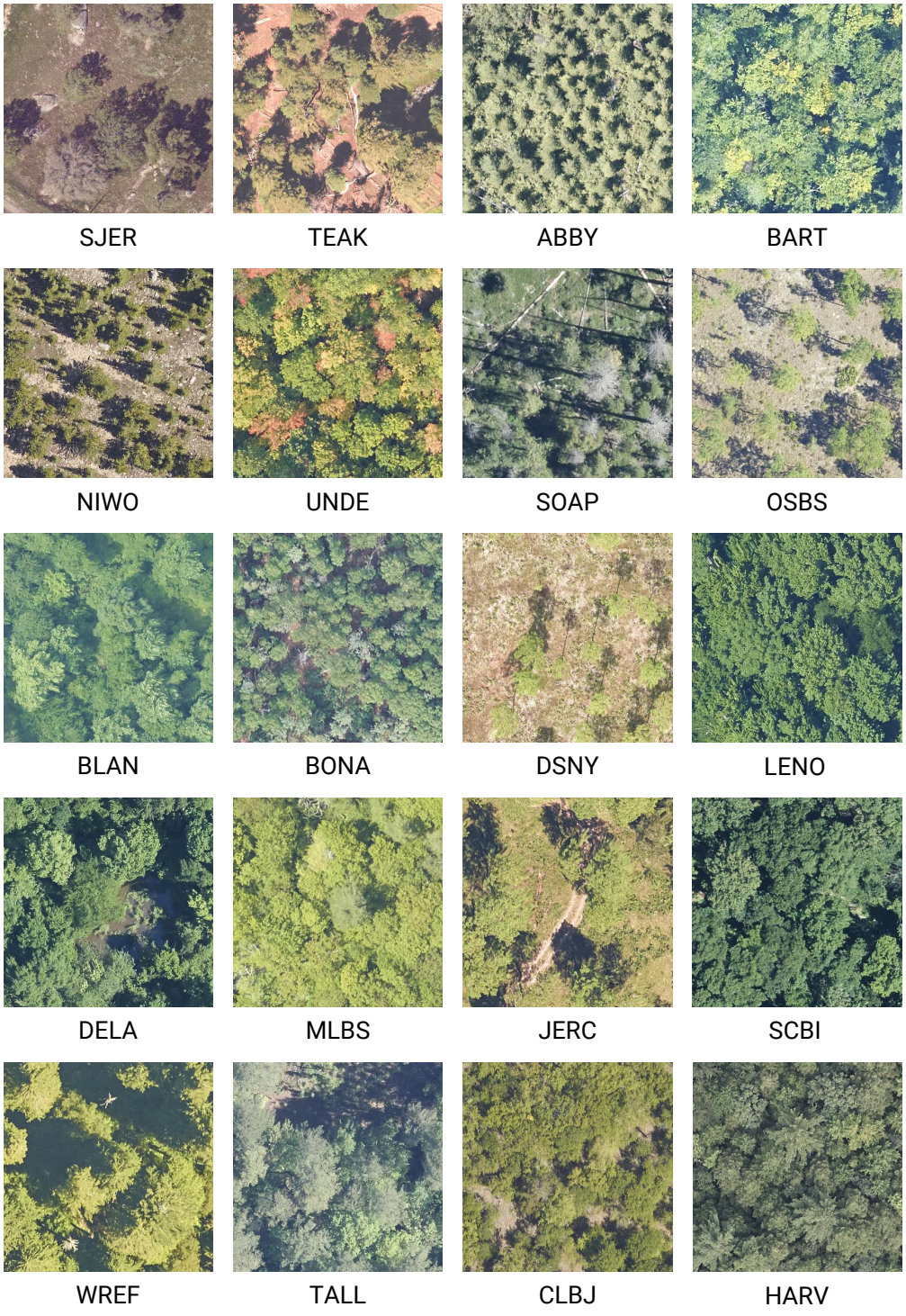


Figure 3.9: Examples of the test images in NeonTreeEvaluation Dataset [2]. The four-letter captions under the images are abbreviations of the site names. The forest types vary across different sites.

Table 3.1: Counting errors of different approaches on Yosemite Tree Dataset

	block size: 960×960 113m×113m in real world				block size: 4800×4800 566m×566m in real world			
	Region A		Region C		Region A		Region C	
Method	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*	*MAE*	*RMSE*
UNet [39]	16.3	20.7	12.9	17.7	318.5	367.0	203.8	228.0
MCNN [30]	19.7	25.3	16.8	21.0	311.0	371.1	283.3	378.0
MCNN (End-to-end) [30]	21.8	27.6	18.4	22.7	388.2	453.6	239.4	286.5
SwitchCNN [31]	17.2	22.2	14.8	18.5	271.1	317.9	175.7	212.2
SegNet [61]	12.7	17.0	15.9	19.4	270.6	299.7	209.8	228.5
CSRNet [32]	20.9	26.3	19.1	24.6	287.0	364.7	295.3	301.3
SANet [33]	18.4	23.5	17.6	22.1	272.1	344.6	285.6	297.9
CANNet [34]	10.8	13.8	12.0	**16.2**	**122.6**	161.1	130.2	159.5
Faster-RCNN-Res50 [17]	13.9	18.1	15.0	20.0	260.2	269.7	237.0	278.0
Faster-RCNN-Res101 [17]	13.4	17.4	15.9	20.9	235.9	256.6	240.6	285.2
RetinaNet-Res50 [19]	14.3	18.1	15.0	18.6	224.1	248.7	187.5	240.0
RetinaNet-Res101 [19]	16.0	20.2	16.2	21.1	290.7	317.2	233.2	301.8
YOLOv3 [20]	17.3	22.6	15.6	20.1	353.2	383.6	256.9	286.9
CenterNet-DLA34 [21]	14.9	20.7	14.6	19.0	344.9	398.0	250.0	299.9
CenterNet-Res50 [21]	13.7	17.5	13.7	17.4	311.1	335.3	237.9	257.8
CenterNet-Res101 [21]	12.1	16.2	13.4	17.2	237.6	271.4	212.0	241.4
DENT-DMG	**10.7**	**13.7**	**11.9**	16.5	148.7	163.9	**123.9**	**158.3**
DENT-CNT	10.7	13.7	12.0	16.6	140.6	**154.4**	133.7	169.3

report the results.

3.5.2 Comparison to state-of-art methods

We compare DENT with the state of the art methods of different fashions, including density-based methods and detection-based methods. The tested density-based methods include fully convolutional networks originally designed for segmentation and crowd counting. The tested detection-based methods include one-stage and two-stage, anchor-based and anchor-free detectors.

The results are shown in Table 3.1 and 3.2. The two heads of DENT, i.e. the DMG and the tree counter, archieve closed performance. On the Yosemite Dataset,

Table 3.2: Counting errors of different approaches on NeonTreeEvaluation Dataset

Method	*MAE*	*RMSE*
UNet [39]	34.7	56.4
MCNN [30]	14.7	24.7
MCNN (End-to-end) [30]	15.5	25.7
SwitchCNN [31]	15.2	25.1
SegNet [61]	28.9	47.5
CSRNet [32]	33.9	52.2
SANet [33]	18.4	30.1
CANNet [34]	14.6	23.1
Faster-RCNN-Res50 [17]	11.1	15.7
Faster-RCNN-Res101 [17]	11.9	18.2
RetinaNet-Res50 [19]	10.9	15.9
RetinaNet-Res101 [19]	12.0	16.8
YOLOv3 [20]	15.2	31.8
CenterNet-DLA34 [21]	10.2	17.2
CenterNet-Res50 [21]	13.0	23.5
CenterNet-Res101 [21]	12.5	20.4
DENT-DMG	**7.5**	12.3
DENT-CNT	7.6	**12.2**

they are nearly on par with CANNet and outperform the other state of arts methods in terms of *MAE* and *RMSE* for every test region and block size setting. On the cross-site NeonTreeEvaluation Dataset, they significantly outperforms all the other methods.

3.5.3 Technical details

We implent DENT using PyTorch [62]. The DMG is based on ResNet18 in PyTorch model zoo. The bert encoder is based on BERT model released in Hugging Face[2] model zoo. We set $d_{\text{model}=512}$ and $h = 8$ for multi-head attention. The dimension of the intermediate layer in feed forward networks is 2048. The standard deviation of Gaussian kernel for density map generation is $\sigma = 15$.

On the Yosemite Tree Dataset, we crop 320×320 subimages from the study areas for training and testing. While on the NeonTreeEvaluation Dataset, as the test set are officially provided as 400×400 images, we crop 400×400 subimages for training from the large training images. Since the downsample rate of the whole DENT is 32, we pad the input images with zero values to 416×416 in both training phase and test phase. The batch sizes we used to train DENT on Yosemite Tree Dataset and NeonTreeEvaluation Dataset are 48 and 32 respectively. Except those mentioned above, we use same setting to train DENT on the two datasets.

Pretraining and initialization The ResNet in Multi-RF network is pretrained on the ImageNet dataset [63, 64]. All the other components of DENT are learned from scratch. All the parameters of the transformer are initialized with Xavier [65]. The initial query count embedding is a fixed zero vector. The token type embeddings are initialized using normal distribution.

Loss The total loss during training is the weighted sum of the losses of the DMG and

[2]https://huggingface.co/

tree counter:

$$L = L_{\text{DMG}} + \lambda L_{\text{CNT}} \tag{3.17}$$

where λ is a weighting factor to balance the losses of the two heads. In our experiments we use $\lambda = 1$ by default.

Optimizer We use Adam [66] to minimize the loss for totally 300 epochs without weight decay. The initial learning rate is 10^{-5} for the first 100 epoches. And then we apply a learning rate decay by a factor of 0.5 for every 50 epochs. We also apply gradient clipping to stablize the training. The max norm of the gradients is set to 0.1.

Regularization and Data Augmentation For reducing overfitting, dropout and random-flip are applied. Specifically, a dropout of 0.1 is added before each Add&Norm layer in the transformer encoder. The training images along with the target tree density map are randomly flipped horizontally and/or vertically.

3.5.4 Ablation study

To evaluate the effects of the Multi-RF CNN and the transformer layers, ablation experiments are done on the test set of Yosemite Tree Dataset for 960×960 blocks. The results are provided in Table 3.3. We start from a ResNet18 without transformer. The output is projected to a single channel linearly using a 1×1 convolutional layer. Interestingly this baseline already archieves lower errors compared with some existing methods (Table. 3.1). After Adding two extra paths to the ResNet18 to get a Multi-RF network, the counting errors are lowered (The third row in Table. 3.1). Adding two transformer layers as encoder make performance gain on both ResNet18 and Multi-RF network. We also try different number of transformer layers. Two layers work best in our experiments. More layers worsen the results and take longer training time to converge.

Table 3.3: Comparison of models with different CNNs and number of transformer layers on Yosemite Tree Dataset for block size 960×960. The models are tested on the union of Region A and Region C. When #transformer layers = 0, the CNN features are linearly projected to the predicted density map, otherwise the density map is generated by DMG.

visual feature extractor	#transformer layers	*MAE*	*RMSE*
ResNet18	0	13.4	17.7
ResNet18	2	12.8	16.9
Multi-RF	0	13.0	17.0
Multi-RF	1	12.0	16.5
Multi-RF	2	**11.3**	**15.2**
Multi-RF	3	11.8	16.7

3.5.5 Inference time

To demonstrate the computational efficiency of DENT we test it on the whole 19200×38400 study area and report the inference time. The tests are done with a single NVIDIA Tesla V100 SXM2 GPU with CUDA 11.3. Every neural layer runs in native PyTorch with batch size=1 in the default FP32 precision. We run 10 times for each case and report the average. The inference time of our basic implementation is 47.8 seconds.

Faster version Due to the shift-invariance of convolution, when a study area is scanned by the Multi-RF CNN, the size of the scan window (input size) does not effect the final feature map.[3] We adopt a two-stage inference mode to improve the GPU utilization and lower the time cost: At the first stage, the backbone takes in a larger input image[4] and generates a larger feature map. At the second stage, the transformer along with the DMG and the tree counter scans the feature map using its original input size. We test this strategy with a 4800×4800 input size for the

[3]This is true only when every layer in the backbone has padding size=0. And beware that if padding size=0 is used at test time, it should be used at training time as well to avoid accuracy drop.

[4]The resolution is still 11.8cm/pixel, but each input image covers a larger area in the real world.

Multi-RF CNN, the inference time is shorten to 16.0 seconds. When the DMG is pruned as discussed in Section 3.3.4, the inference time can be furthur shortened to 11.5 seconds. Even further improvement is possible with tricks like batch processing and low precision inference but beyond the scope of this work.

3.6 Conclusion and future work

We presented a deep neural regressor, DENT, based on CNN and transformer for tree counting in aerial images. We built a large benchmark dataset, Yosemite Tree Dataset, to evaluate different tree counting algorithms. We also used an existing cross-site dataset to test the robustness of the methods. Our approach achieved competitive results and outperformed the state-of-art methods. Ablation study further supported the effectiveness of the design. With the advancement of drones, aerial imagery is becoming more and more affordable. However, due to the limited visual field, the captured photos need to be stitched to create the whole picture of a large study field. This procedure can be laborious. For this reason, an accurate video-based tree counting algorithm would be more automatic and appealing. The emerging applications of video-based density estimation methods for crowd counting inspired us. We will explore video-based tree counting algorithms in the future work.

Chapter 4

Fish Detection and Classification Using Boat Camera

4.1 Introduction

The species composition and distribution of fish are important biological data for fisheries research. Traditionally the counting of different types of fish is processed manually, which is time-consuming. With the development of computer technology, computer vision-based fish detectors and classifiers are applied to enhance the efficiency.

In this work, we propose a system combining two strategies together: one is to use context information to overcome the unclarity of fish; the other one is to detect localize and align fish first before classifying to extract discriminative features. The system is expected to balance the two strategies automatically. Our research aims at designing a harsh environments robust solution for fish classification based on deep learning. The query images taken in different work environments can be sent to a remote server. Then the remote server launches the proposed system to localize and classify the fish in the images (Fig. 4.1), and then all the results are stored on hard

Figure 4.1: Examples of fish detection and classification in different environments

disks.

In the world-wide competition of "The Nature Conservancy Fisheries Monitoring" hosted by Kaggle, our model achieved top 0.7% rank on the final leaderboard among solutions submitted by 2,293 different competitive teams.

4.2 Related Works

4.2.1 Fish Classification

Traditional computer vision-based fish detection and classification tasks are usually dealt with using hand engineered features. Fish is assumed to be lying on a semi-opaque conveyor with backlight underneath[67]. After edge detection a grid is drawn over the fish body. With 10 equidistant width measured along the length, average color is drawn from each grid element to form a feature vector. The classification is based on principal component analysis (PCA). Concetto et al. [68] propose a method

for fish classification in underwater environment based on not only shape but also texture. The appearance of fish in different 3d views is modeled using affine transformation. Mehdi et al. [69] apply a Haar detectors to localize the snout and tail of a fish and PCA to model the prior knowledge of shape. To our best knowledge, most of the related works [70] [67] [68] [69] are based on low level hand-crafted features and applied in elaborately controlled circumstances. Hence their usage is limited.

4.2.2 Deep Convolutional Neural Networks

In recent years, deep learning with convolutional neural networks (CNNs) has proven to be very effective at various perception tasks. On ImageNet [63] classification task, CNN has produced state of art results, reaching as high as 97.75% accuracy on ILSVRC 2017 [71] which has already been better than human ability on image classification (94.9% accuracy). On detection problem of ImageNet, CNN provided 73.14% mean average precision (mAP) for 85 categories on ILSVRC 2017. Some advanced neural networks have been proposed during the ImageNet competition, like VGGNet [13], GoogleNet (Inception) [16], and ResNet [14]. Thus, object detection and image classification problem has been achieved a series of breakthroughs using deep convolutional neural network.

4.3 Dataset

We test the proposed system in Kaggle *The Nature Conservancy Fisheries Monitoring* Competition[1]. (Both the competition itself and the provided dataset are refered as NCFM for short below.) The objective of the competition is to classify the provided images into 8 categories: *Albacore tuna*, *Bigeye tuna*, *Dolphinfish*, *Opah*, *Shark*, *Yellowfin tuna*, *Others* and *No fish*. Competitors are required to train a model on the

[1]https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring

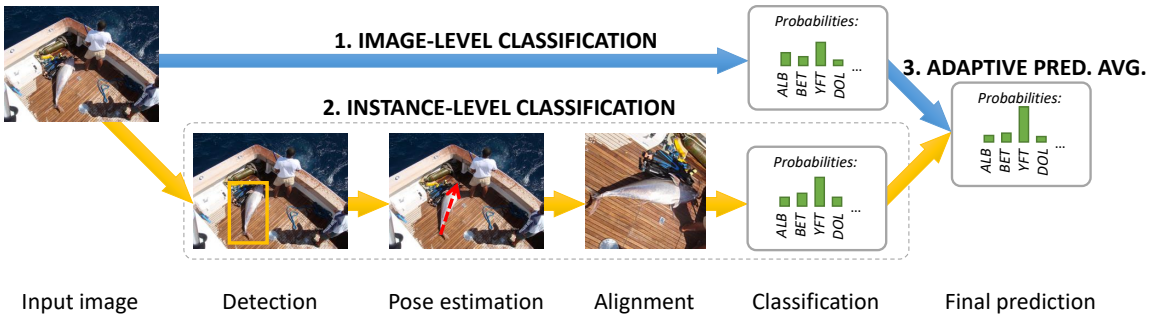


Figure 4.2: Data flow in the proposed system. (*blue arrows*) The image-level classification branch makes prediction according to all the objects in the scenario. (*yellow arrows*) Instance-level classification branch focuses on fish individuals. Fish are detected and aligned in order to be classified. At last, the importance of the two branches is estimated; and then the predictions made by the two branches are reweighted and averaged as the final prediction.

provided training set and submit prediction on test sets to the online evaluation system. The evaluation results are shown on a leaderboard. To avoid leaderboard probing and reflect the overall performance of the solutions from competitors, accuracy is invisible to the competitors. From the feedback of the system only cross-entropy loss can be seen, as the only criterion for ranking:

$$loss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log\left(p_{ij}\right), \tag{4.1}$$

where N is the number of images in the test set, M is the number of possible classes, y_{ij} is 1 if image i belongs to class j and 0 otherwise, and p_{ij} is the predicted probability that image i belongs to class j.

There are two stages in the competition. At each stage competitors are given a test set. At stage 1 competitors can modify and tune their models and choose a most satisfying one. At stage 2 the chosen models are demanded to be fixed and no parameter-tuning is allowed. The test sets on these two stages are not overlapped. The final rank of each competitor team is determined only by their performance on the test set of stage 2.

There are 3777 images in training set, 1000 images in test set 1 and 12138 images in test set 2. All the images are taken by cameras which are set up on different fishing boats. The position and view angle of each camera is fixed. Compared with VOC2007[72], Imagenet [63] and MSCOCO [73], the images in NCFM have much more complicated scenarios, consisting of workers walking around, fishing gears and containers everywhere. The fish can be scattered on the bottom of boats, in the fish net, held by people or in the sea water. They are usually far from the cameras and may be partially occluded. So the lighting varies seriously in different images since they can be taken in any time in day and in different weathers. Sometimes rain or splashed sea water covers the camera or makes it lose focus. Hence NCFM dataset have strong representativeness as they expose the problems that an automatic fishery classification system may encounter in real working environments. We evaluate the performance of proposed method on NCFM dataset and report the cross-entropy loss.

Table 4.1: Number of images in NCFM dataset

Subset	***Training***	***Test - Stage 1***	***Test - Stage 2***
#images	3777	1000	12138

4.4 Proposed System

Our proposed fishery classification system has two branches (Fig. 4.2), one is for image-level classification and the other is for instance-level classification. The prediction made by these two branches are fused together as the final prediction.

4.4.1 Instance-Level Classification

Directly using the whole image as input may lead to a heavy burden on training of CNN classifiers. Because firstly the abundant objects appear in images unrelated with fish categorization may distract the learning direction. Secondly the dataset can be easily over-fitted especially when the sample size of training set is not large enough. For example, a classifier may unexpectedly learn the boat colors disregarding the features of fish. Thirdly to keep the resolution of image good enough in order to preserve the appearance details of fish, either the CNN classifiers must have a large input size, which consumes a lot of memory and computational time; or the images have to be scaled to a small size and then much information of fish appearance is lost. For these reasons, focusing on fish instances rather than distributing the receptive field over the whole image alleviates the burden of classifier on both training phase and test phase.

The variation of the poses and scales of fish also encumbers CNNs. Since CNNs have to learn redundant filters to recognize objects in different poses and scales. If they can be resized to similar scales and rotated to similar azimuth angles, the filter may be learned more effectively to describe discriminative features.

To address the problems mentioned above, the instance-level classification branch of the proposed system is consisted of four cascade modules: detection, pose estimation, alignment and classification:

Detection

We expect to localize the fish in the images using bounding boxes:

$$B = \{(s_b, x_{0b}, y_{0b}, x_{1b}, y_{1b})\} \tag{4.2}$$

where B is a set of bounding boxes for a given image; b is the index of a certain bounding box; s_b is the detection score of the bounding box b, representing the confidence that there is a fish in the box; (x_{0b}, y_{0b}) and (x_{1b}, y_{1b}) are the coordinates of the left-top vertex and the right-bottom vertex of the bounding box b.

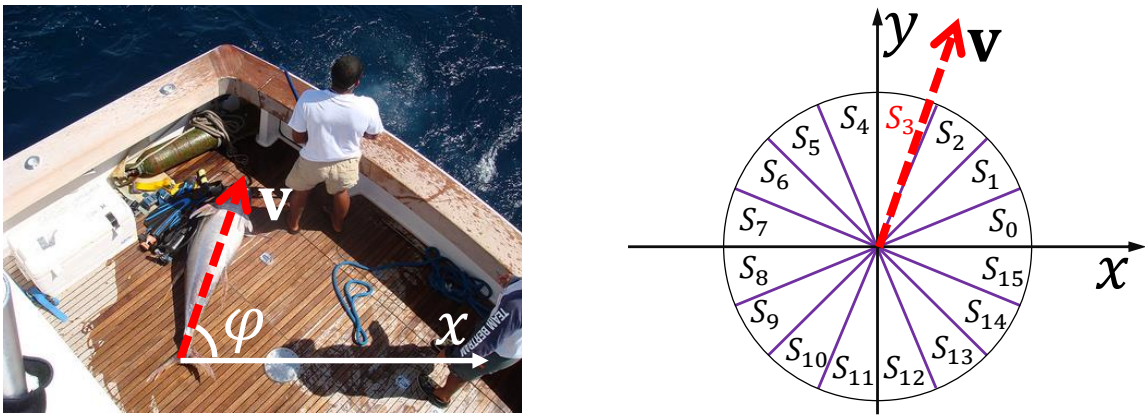
Two state-of-art methods SSD [18] and YOLOv2[74] are chosen as the detectors in the current work. The performance is discussed in 4.5.1.

Pose Estimation & Alignment

Alignment is a widely used procedure to deal with pose variation of objects such as face [75] [76] [77] and bird [78] [79] in fine-grained classification tasks. We follow a simple alignment strategy: all detected fish are rotated by appropriate angles so that they can have the same orientation (e.g. lying horizontally and facing to the right).

At training phase the rotation angle can be computed simply based on the key point labels (coordinates of fish heads and tails). At testing phase, the poses of fish are unknown. So the orientation of fish need to be estimated first. There are multiple approaches to train CNNs that can predict the pose of an fish. One is to applies CNNs as regressors to predict keypoints of object like DeepPose [80]. Another is to formulate the pose estimation problem as a classification task. We find that the latter one works better. A possible reason is that DeepPose was originally evaluated on FLIC Dataset [81] and Leeds Sports Dataset [82]. The images in these two datasets were taken in Hollywood movie studios and stadiums, where the photography conditions such as illumination and focus are well controlled. While the images in NCFM were taken in harsh environments; in this case a neural network which roughly classifies the orientation of fish may be more robust than those who try to output the exact positions of keypoints.

Specifically, denote the orientation of fish as a vector that $\mathbf{v}$ from the tail to its



(a) Azimuth angle of fish

(b) Orientation and sectors

Figure 4.3: Pose estimation of fish: (a) The orientation of fish is represented as a vector **v** from tail to head. (b) XY-plan is equally divided into sectors. **v** belongs to one of the sectors.

head (Fig. 4.3):

$$\mathbf{v} = (v_x, v_y) = (x_{head} - x_{tail}, y_{head} - y_{tail}) \tag{4.3}$$

where (x_{head}, y_{head}) is the position of fish head and (x_{tail}, y_{tail}) is the position of fish tail. In the polar coordinate system, if the angle between 0 and 2π is equally divided into K sectors:

$$S_k = \left[\frac{2\pi k}{K}, \frac{2\pi(k+1)}{K}\right) \tag{4.4}$$

$$k \in \{0, 1, 2, ..., K-1\} \tag{4.5}$$

The azimuth angle φ of fish is defined as:

$$\varphi = atan2(v_y, v_x) \tag{4.6}$$

A fish belongs sector S_k if:

$$k = floor\left(\frac{K\varphi}{2\pi}\right) \tag{4.7}$$

If the predicted sector of fish is k, we use the approximated predicted azimuth angle:

$$\varphi^* = \frac{2\pi(k+0.5)}{K} \tag{4.8}$$

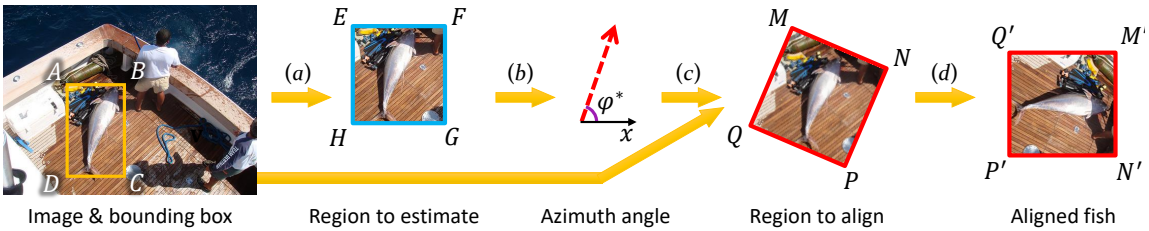


Figure 4.4: Pose estimation and alignment: (a) given an image and a bounding box $ABCD$, a square subimage $EFGH$ is cropped. The side length of cropped subimage is as same as the longer side of the bounding box. (b) The azimuth angle of fish in subimage $EFGH$ is then estimated as φ^*. (c) A region $MNPQ$ with azimuth angle φ^* containing the fish is selected. (d) The subimage in $MNPQ$ is cropped and rotated so that the fish lies horizontally. The rotated subimage is denoted as $M'N'P'Q'$ (called aligned fish in this work) and is used as input of classifiers for instance classification.

In practice, we apply a VGG16 [13] as the pose classifier and choose $K = 16$ to predict k.

Once a fish instance is detected and the pose is estimated, a square sub-image I_b is cropped out as the aligned fish. (See Fig. 4.4. I_b is the subimage in square $M'N'P'Q'$. And see Fig. 4.5 for more details.) The side length l of I_b is set to be proportional to the diagonal length of box b:

$$l = (1+\rho)\|\mathbf{v}\| \tag{4.9}$$

where ρ is a padding factor to control the room between the fish and the edge of the sub-image. In our experiments, ρ is set to 0 at all test phases and randomly sampled from $[0, 1/7]$ at training phase.

Instance Classification

We apply a CNN to learn conditional probability functions F':

$$F'(c, I_b) \approx P(c|fish) \tag{4.10}$$

where I_b is an aligned fish; P stands for probability, c stands for a possible fish type, i.e. *Albacore tuna*, *Bigeye tuna*, *Dolphinfish* etc.

Because there may be zero, one or multiple fish in an image, the prediction for the whole image is:

$$F(c, I) = \begin{cases} 1 - s_{max}, & if\ c = \text{“}No\ fish\text{”} \\ \dfrac{s_{max} \sum_b s_b F'(c, I_b)}{\sum_b s_b}, & otherwise \end{cases} \tag{4.11}$$

where

$$s_{max} = \max_b s_b \tag{4.12}$$

4.4.2 Image-Level Classification

Ideally, fish types can be determined by the appearance of fish instances. However, when the image is too blurry or dark for an intelligent agent to clearly see the fish, or if the training data cannot well cover the appearance of fish in different environmental conditions, the prior knowledge plays a important role to recognize the fish using context information. For example, it is reasonable to assume that some boats catches some types of fish more than other types. For this purpose, similarly with 4.4.1 we apply a CNN to learn the conditional probability distribution of fish type c given an image I:

$$G(c, I) \approx P(c|boat, daylight, weather...) \tag{4.13}$$

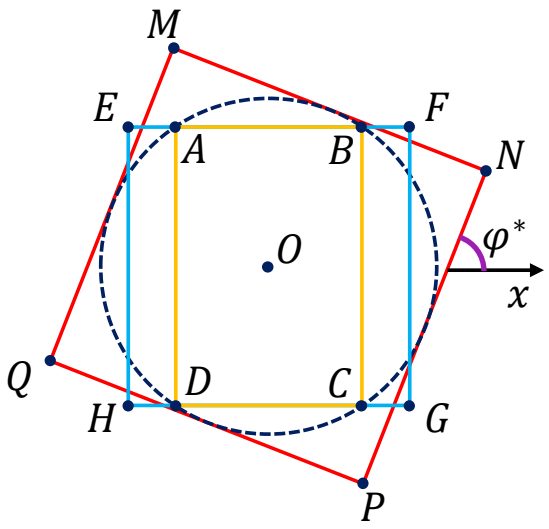


Figure 4.5: The spatial relation among the preliminary results (shown in Fig. 4.4) in pose estimation module and alignment module. Rectangle $ABCD$, square $EFGH$ and square $MNPQ$ have the same centroid O. Circle O is both the circumscribed circle of rectangle $ABCD$ and the inscribed circle of square $MNPQ$.

The CNN takes the whole image as input and outputs the predicted probability of each possible fish type.

4.4.3 Adaptive Prediction Average

For real-world application, it cannot be ensured that all the environmental conditions also appear in training set. For example, the actual working environment may be on a new boat which has not been seen before (has not been seen in training set). In such cases, (4.13) may be hardly learned and may output unexpected results. To suppress this side effect, the results of instance-level and image-level are reweighted in the form as following:

$$H(c, I) = (1 - w(I))F(c, I) + w(I)G(c, I) \tag{4.14}$$

where $H(c, I)$ denote the final prediction function which gives the predicted probability that the fish in image I belong(s) to class c; and $w(I)$ is an function that adaptively reweights the instance-level prediction and image-level prediction. Following the heuristic that:

- The more likely the boat is a new boat, the larger weight should be assigned to the image-level prediction and vice versa.
- The more clearly the fish can be seen, the larger weight should be assigned to the instance-level prediction and vice versa.

we take the familiarity of boat and the clarity into account to choose $w(I)$.

It's difficult to formulate the familiar boat/new boat estimation as a supervised classification task, because there are only familiar boats in training set. An idea is to label all the boats in the training set, and randomly choose some boats as pseudo new boats to train a classifier. However, labeling work is time-costing and not preferred in this work. Instead we directly measure the familiarity of a scenario by the distance in feature space between a test image and its nearest neighbor in training set. Specifically, the distance d is defined as:

$$d = \min_i(\|f(I) - f(I_i^{train})\|) \tag{4.15}$$

where I_i^{train} is the i-th training image; function $f(I)$ denotes the feature of image I. In this work it is extracted from the last convolutional layer of a CaffeNet [12] pretrained on ImageNet. The smaller d is, the more similar the test image is with some training images.

As the detection scores stand for the confidence of detectors that there are fish in the bounding boxes, they imply how obviously the object in boxes are fish. We choose the their maximum s_{max} to estimate the clarity of fish in a test image.

For convenience, we assume $w(I)$ can be decomposed into two terms respectively relied on s_{max} and d and independent with each other:

$$w(I) = w_s(s_{max})w_d(d) \tag{4.16}$$

The functions $w_s(s_{max})$ and $w_d(d)$ are manually tuned and the final choice is discussed in 4.5.5.

4.5 Experimental Results

4.5.1 Detection

In our experiments, we choose 512×512 input size for SSD and 544×544 for YOLOv2. Since labeling the test data is prohibited in the competition, we validate the detection module on a small subset containing 500 images randomly sampled from the training set. SSD achieves 0.853 mean average precision and YOLOv2 achieves 0.738 mean average precision. The precision-recall curves are shown in Fig. 4.6. The union of the

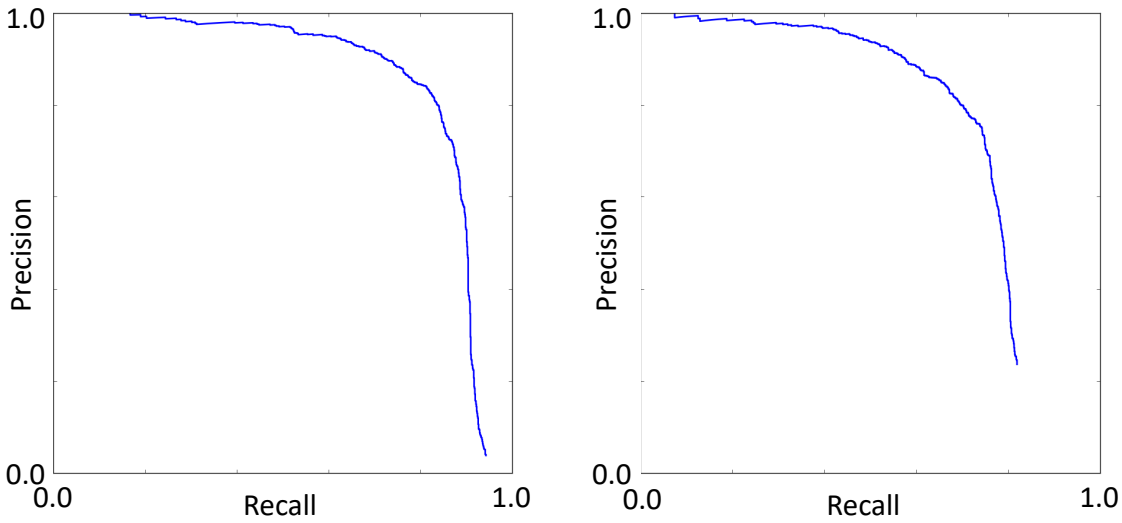


Figure 4.6: Precision-Recall curves of detectors on a subset of 500 images sampled from training set: (*left*) SSD; (*right*) YOLOv2.

two sets of bounding boxes is used as the final output of the detection module. Inter-detector Non Maximum Suppression (NMS) is tested. However, the inter-detector NMS worsen the final classification performance. A possible reason is that the scores of bounding boxes proposed by different detectors are lack of comparability. For example SSD and YOLO have different hard-mining policies to deal with the problem of imbalance samples, which makes them assign different detection scores to an object.

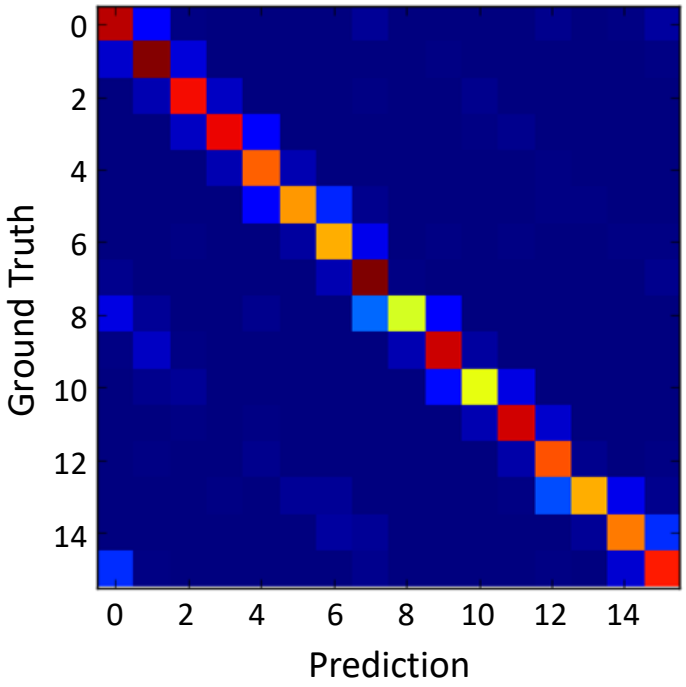


Figure 4.7: Confusion matrix for pose estimation on validation set

4.5.2 Pose Estimation & Alignment

The VGG16 pose classifier is evaluated on the same subset of training set as mentioned in 4.5.1. It achieves 0.80 top-1 error rate and 0.96 top-2 error rate. The confusion matrix is shown in Fig. 4.7.

4.5.3 Alignment & Instance Classification

For instance classification, we follow a simple strategy: train multiple networks and get the average of their prediction. We evaluated three different CNN structures for instance classification. They are ResNet50 [14], VGG16 [13] and Inception V3 [16]. To prevent overfitting, random cropping and scale jittering are applied at training phase. We also find using external data from ImageNet helps to decrease loss. The results are listed in Table 4.2.

Table 4.2: Performance of Instance Classification on Test Set 1

Network	**Quantity**	**Trained on external data**	**Loss**
VGG16	1	*No*	1.137
VGG16	32	*No*	1.017
VGG16 (unaligned fish)	32	*No*	1.284
VGG16	8	*Yes*	0.942
ResNet50	32	*No*	0.968
ResNet50	8	*Yes*	0.921
Inception V3	32	*No*	0.886
Inception V3	22	*Yes*	0.853

4.5.4 Image-Level Classification

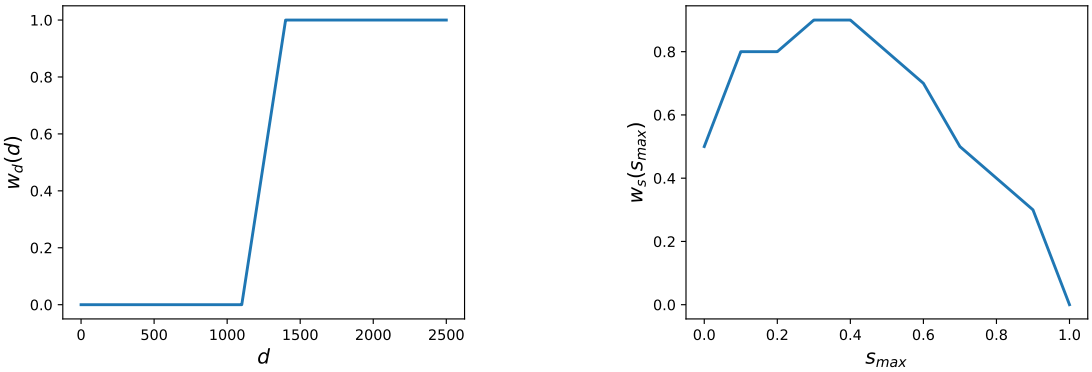
We modify the input size of VGG16 to 640×360. To decrease the computational complexity, all the fully connected layers are discarded and three convolutional layers are added followed by a global average pooling layer (GAP). All the new added convolutional layers have 128 channels and 3×3 kernel size. In this way the original VGG16 is modified to a fully convolutional network (FCN). In our experiments, a single model of such FCN achieves 1.129 cross entropy on test set 1. Averaging 16 different models achieves 0.937 cross entropy while averaging 256 different models achieves 0.918 cross entropy. Further enlarging the input size doesn't improve the results, hence we keep the input size of the FCNs 640×360 in the final model.

Table 4.3: The Effects of Different Choices of w

$w(I)$	0.00	0.25	0.50	0.75	1.00	Adaptive
$loss$	1.017	0.873	0.632	0.725	0.937	**0.604**

4.5.5 Adaptive Prediction Average

To observe the effect of different choices of $w(I)$, we use the average of prediction made by 32 VGG16 instance classifiers and the average of prediction made by 16 FCNs for image-level classification. Functions $w_d(d)$ and $w_s(s_{max})$ are manually tuned as shown in Fig. 4.8. We observes simply averaging the results of instance-level classification and image-level classification outperforms either one. And adaptive $w(I)$ outperforms all the constant $w(I)$. The results are listed in Table. 4.3



(a) $w_d(d)$ in the final model.

(b) $w_s(s_{max})$ in the final model.

Figure 4.8: The final choice of $w_d(d)$ and $w_s(s_{max})$.

4.5.6 Overall Performance

The configuration of the final submitted model is listed in Table 4.4. It achieves 0.578 cross entropy (rank: 17th) on test set at stage 1 and 1.387 (rank: 16th) at stage 2. The main reason for the gap between the loss at two stages is because most of the boats in test set 1 also appear in training set, while the boats in test set 2 rarely

appear in training set. Hence at stage 2, less context information can be used to make inference. We also note that although the loss increases at stage 2, our rank is actually slightly improved, which implies that our method is robust to new boat among all the competitors.

Table 4.4: Networks in the Final Model

Level	**Job**	**Network**	**Quantity**
Instance-Level	Detectors	SSD	1
		YOLO	1
	Pose Estimator	VGG16	1
	Classifiers	Inception V3*	22
		Inception V3	32
		VGG16*	8
		VGG16	32
Image-Level	Classifier	FCN(VGG16) + GAP	256
Adaptive Average	Descriptor	CaffeNet	1

*Trained using external data from ImageNet.

4.6 Conclusion

This work introduces a fish classification system which can extract discrimination features from fish instances and also make use of the context information from the scenario around. It is robust to different environmental conditions since it can balance the inference made based on fish instances and context information. Experimental results show that the system is an effective solution for practical applications related to fish classification. The whole system combined several state of art CNN models in object recognition and image classification domain and it can be run automatically from end to end to generate the confidence score of each class for each input image. Other advanced CNN models can also be integrated in the system base on needs to generate promising results.

Chapter 5

Livestock Detection and Counting Using Inspection Robot

5.1 Introduction

Frequently counting the number of pigs in grouping houses is a critical management task for large-scale pig farming facilities. On one hand, pigs are often moved into different barns at distinct growth stages or grouped into separate large pens by size. Farmers need to know how many pigs are in each large pens. On the other hand, comparing the counting result with the actual number of pigs enables the early detection of unexpected events, e.g., missing pigs. However, walking around the pig barns to count a large number of pigs is costly in labor. Thus, automated pig counting and monitoring using computer vision techniques is a promising way to support intensive pig farming management, while reducing cost.

In recent years, various computer vision algorithms have been widely adopted to support various developments of agriculture and farming automation, such as cattle gait tacking [83], pig weight estimation [84] and fruit counting [85]. Despite of these exciting progresses, pig counting remains a very challenging task, due to large pig

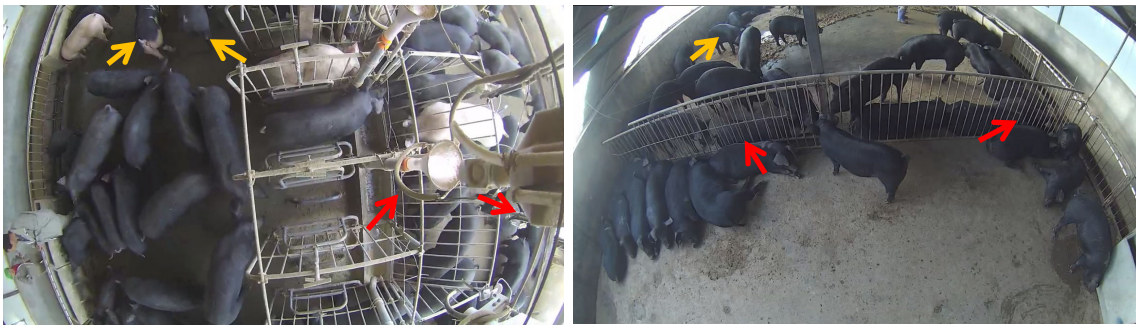


Figure 5.1: Illustrations of pig counting challenges in large grouping houses. The top-down view images are captured by our inspection robot with a fisheye camerat. Red arrows point to examples of pig overlapping and occlusion. Yellow arrows show cases where pigs are moving in or out of camera field of view.

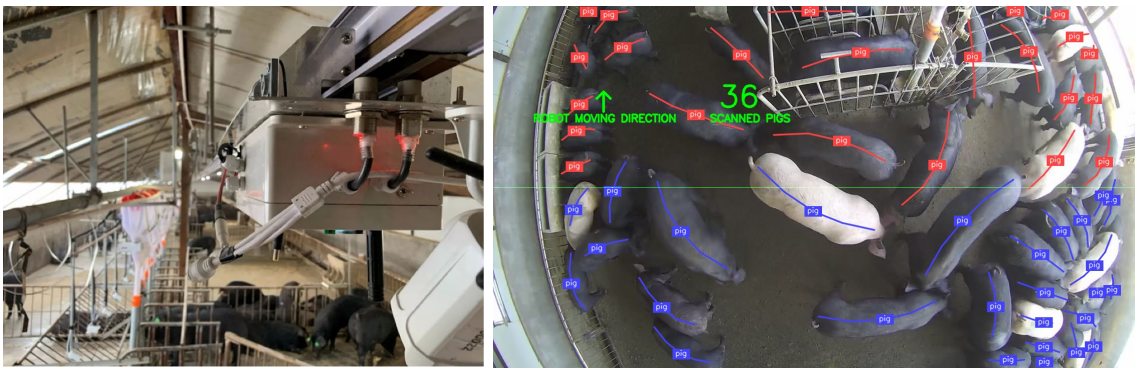


Figure 5.2: Illustrations of our pig counting system. (a) the installed inspection robots with rails and fisheye cameras for pig counting. (b) a single video frame with detected pig skeletons using our counting algorithm.

movements, high group density, overlapping, occlusion and camera perspective, as illustrated in Fig. 5.1. Few works in literature studied the development of automated pig counting system. Existing works [86] only handled pig counting problem in a single image. Nonetheless, as shown in Fig. 5.1, the field of view of a single image is only restricted to a small region and it is impossible to monitor a large pig grouping house. Furthermore, it could not deal with the cases that pigs frequently enter into or exist from the camera view. Towards overcoming these challenges, we presented an novel automated counting algorithm with an inspection robot and monocular fisheye camera. Fig. 5.2a showed two pictures of our inspection robot with a fisheye camera installed on the roof rail in our experimental pig grouping houses. Fig. 5.2b visualized a single video frame with detected pig skeletons output using our pig counting pipeline.

The main contributions of this work are summarized as follows. 1) The sensor configuration is presented, which is suitable for pig counting in large-scale grouping house. 2) A novel bottom-up detection method is proposed to identify pigs, while addressing detection challenges due to overlapping, occlusion and deformation of body shapes. 3) A novel online spatial-aware temporal response filtering (STRF) method is designed to suppress false positives caused by tracking failures or pig movements. 4) An efficient algorithm of the counting pipeline is designed and deployed to an embedded system, which achieves high speed performance.

5.2 Related works

Counting in the crowd is an important, but challenging task due to severe occlusion, perspective distortions, complex illumination and diverse distribution of target sizes [87]. Recently, deep-learning-based methods [87, 88] have been developed to estimate single image density map for crowd counting. Sindagi et. al. [88] developed a contex-

tual pyramid convolutional neural network (CNN) for crowd density map estimation. Both global and local contexts were employed in the network to achieve better accuracy. Shen et. al. [87] proposed an adversarial cross-scale consistency pursuit method to improve the estimation consistency and reduce the averaging effect in [88]. These methods formulate the counting problem as density map estimation, thus having the advantage to handle server occlusion and perspective distortions. However, density-map-based methods lost the detailed individual information and discarded the accurate location information for each single target. Therefore, it loses the ability to associate targets across time, and is not suitable for video-based counting.

Recently, researchers in agriculture presented many works towards tackling counting problems in various scenarios. Tian et. al. [86] counted pigs in a single image using a CNN-based method for pig density map estimation. Similar as [88], this method is not suitable for video-based counting problem. As a single image only have a small field of view (as shown in Fig. 5.1), it cannot be used for pig counting in large grouping houses. Liu et. al. [85, 89] developed a fruit counting pipeline using a monocular camera. Individual fruits are first segmented using a CNN-based method, and then tracked by a Kalman Filter corrected Kanade-Lucas-Tomasi (KLT) tracker. A structure from motion (SfM) algorithm was utilized to get the relative 3D location and size estimate to reject outliers and double counted fruit tracks. This method is only suitable for rigid shape and stationary target counting task, and does not work for moving livestock counting cases. Hodgson et. al. [90] demonstrated that images collected by unmanned aerial vehicles (UAV) could help wildlife monitoring and counting. Rivas et. al. [91] studied cattle detection from aerial view photos. Counting based on aerial view is promising, but could hardly be used for indoor livestock counting scenes without developing algorithms to handle severe occlusion (e.g. caused by indoor building structures or perspective distortions), overlapping, double counted tracking trajectories due to entering into or existing from camera view.

Different from previous approaches, we presented a novel video-based pig counting system for large pig grouping houses. The developed counting pipeline overcame dense detection challenges (e.g. overlapping or occlusion) by a novel bottom-up pig body parts detection and association algorithm. A STRF method was developed to obtain the counting number by reducing the counting error caused by tracking failures or pig movements.

5.3 Approach

In this work, we presented an efficient pig counting system for large pig grouping houses. Fig. 5.3 demonstrates the entire algorithm pipeline. In our counting system, the camera moved from one side of the pig grouping house roof till the other end of roof and scanned the whole house with top-down view. A whole single counting pass scanned the house once by the camera. As summarized in the Fig. 5.3, subsequently, we detected multiple pig body keypoints, associated them to localize each individual pig, tracked pigs cross frames and obtained counting results using STRF method.

5.3.1 Sensor configuration

An inspection robot, which can move back and forth along a rail installed on the roof of the pig house, was used for pig counting data acquisition and processing from top-down view (Fig. 5.2a). Several sensors used for different applications were installed inside the robot, including a monocular fisheye camera for pig counting, a RGB-D camera for pig weight estimation, gas and temperature sensors for environmental control etc. Inside the inspection robot, an embedded system with RockChip RK3399 multi-core ARM processor was used for processing data from cameras and running the pig counting algorithm.

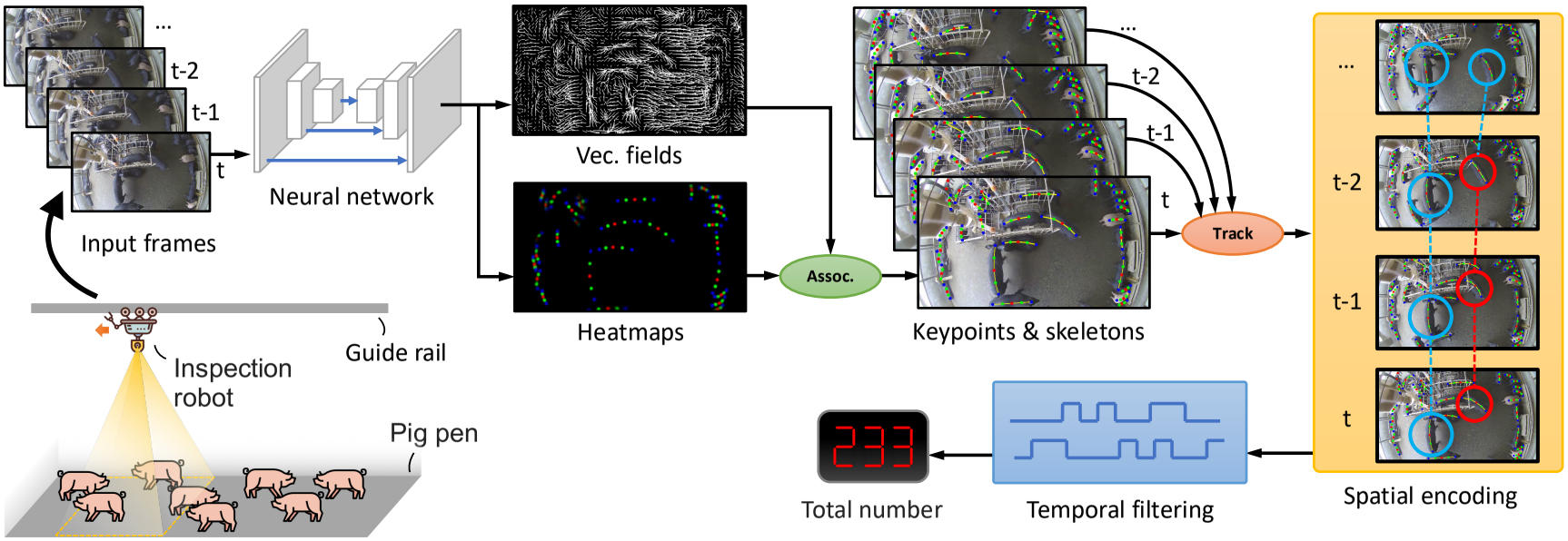


Figure 5.3: Pig counting pipeline. The inspection robot moved from one side of the pig house roof to the other end to scan the whole region. A proposed bottom-up detection CNN model was first applied on each video frame to obtain the keypoints and skeletons of all pig candidates. An on-line tracking algorithm was then used to generate the temporal associations across frames. Lastly, STRF, including spatial encoding and temporal filtering, was used to generate the final count.

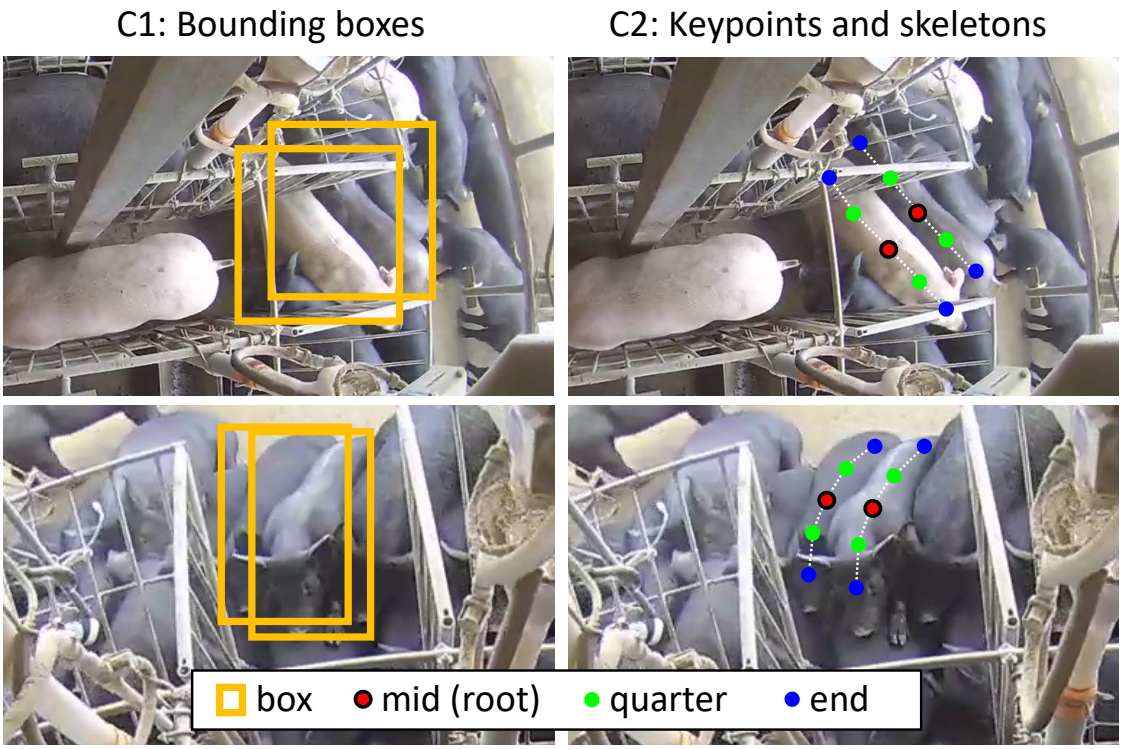


Figure 5.4: Illustration of top-down bounding boxes v.s. bottom-up keypoints for pigs detection. Column 1 (C1): bounding boxes had very high overlap ratios for adjacent pigs. Column 2 (C2): body parts keypoints for adjacent pigs. In this work, five keypoints are defined: one middle body part keypoint, two body end keypoints and two quarter body keypoints.

5.3.2 Bottom-up detection

The first step was to detect pig candidates in each video frame. Traditionally, top-down object detectors, such as faster RCNN[17], SSD [18] and YOLOv3 [20], have been widely used. These methods first proposed locations of detection candidates using bounding boxes, and then classified each box to be the real target or not. Non-maximum suppression (NMS) are employed as a post-processing method to significantly reduce false positive candidates by removing the bounding boxes that have high overlap ratios (intersection over union) with each other. Nonetheless, using bounding boxes to localize the pigs is sub-optimal in this application. The deformable long oval pig shapes are very challenging for bounding-box-based approaches in crowded scene. As shown in Fig. 5.4C1, the bounding boxes around two adjacent pigs have

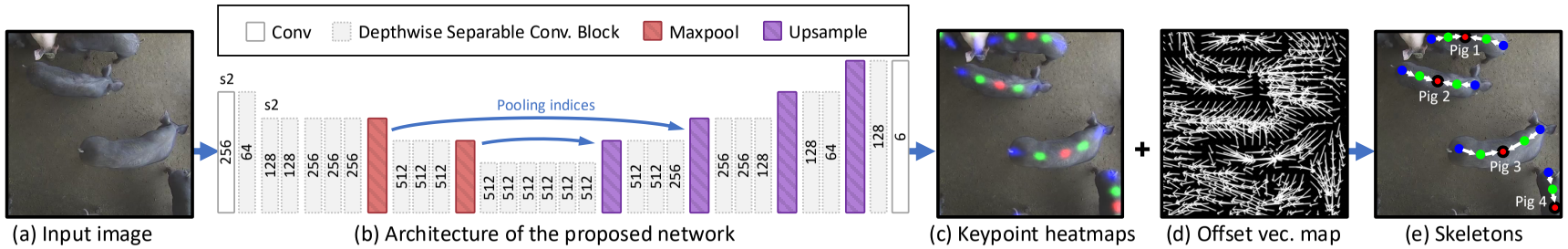


Figure 5.5: The proposed bottom-up keypoints detection CNN architecture. The network contains two regular convolutional layer at the two ends and 24 depthwise separeble conv. blocks in-between. In (b) the numbers indicate the number of output channels; And $s2$ means a stride of 2. All the conv. layers except the pointwise conv layers have kernel size 3×3.

very high overlap ratio, whose ambiguous nature tends to confuse the neural network training. Moreover for inference, the NMS post-processing step would enforce the detector to only select one bounding box for these high overlapping cases, resulting in false negatives. Compared with bounding boxes, the pig skeletons defined by keypoints are more suitable for differentiating pigs in the crowd as shown in Fig. 5.4C2. In this work, we defined five pig body keypoints, including one *mid point* (red), two *quarter points* (green) and two *end points* (blue); and tree-structured pig skeletons connecting the adjacent keypoints.

Inspired by [92], we presented an efficient bottom-up detection approach (Fig. 5.5) to overcome aforementioned limitations. This method is consisted of a keypoints detection step and a keypoints association step. These steps were based on a deep convolutional encoder-decoder network. The network output two different kinds of maps: 1) Four keypoint heatmaps and 2) an offset vector field. (Fig. 5.5d) Each heatmap provided information to classify each pixel into to one of the keypoints or background class. The offset vector field indicated the relative positional relationships between the adjacent keypoints, which helped the system group the keypoints and identify which pig instance that these keypoints belonged to.

Keypoint detection The goal is to detect all visible keypoints belonging to each single pig in the input. For this purpose, we applied a fully convolutional network to produce heatmaps with four channels (three channels for each keypoint type and one for background), which had the same size with the input image. This heatmap prediction was then formulated as a per-pixel multi-class classification problem. For each pixel location, the neural network learned to predict if it belonged to one of the keypoint type or background. We followed [93] to generate classification targets. Let $\mathcal{D}_R(y) = \{x : \|x - y\| < R\}$ be a circular region centered at position y with radius R. We denoted $k_{i,c}$ as the i-th keypoint of type c. All pixels of $\mathcal{D}_R(k_{i,c})$ had the same class label c. In this work, R was set to be 5. Cross-entropy loss was employed for

this task. At testing stage, the local maxima of the heatmaps were chosen as the predicted keypoints.

Keypoint association Due to instance-agnostic nature of the predicted keypoints on heatmaps, one unique instance ID had to be assigned for each detected keypoint so that we "connect the dots" belonging to the same individual instance. For this purpose, we added to our neural network a separate two channel outputs of offset field indicating the displacement from a given keypoint to its parent in the skeleton (Fig. 5.4C2). Here we denoted $F(k_{i,c})$ as the parent node of keypoint $k_{i,c}$. If $x \in \mathcal{D}_R(k_{i,c})$, the target offset $V(x)$ was vector starting from x. If $k_{i,c}$ itself is a root node, i.e. $c = mid$, $V(x)$ ended at $k_{i,c}$; Otherwise $V(x)$ ended at $F(k_{i,c})$.

Let us denote the offset field predicted by the network as $U(x)$. In order to supervise the training, the regression loss for offset field was defined as

$$L_r = \sum_x (1 - G_0(x)) \left\| U(x) - V(x) \right\|^2, \tag{5.1}$$

where $G_0(x)$ was the binary background mask used for ignoring the regression loss at the background pixels, where the offset vector were undefined.

At testing stage, an iterative greedy algorithm was adopt to associate the predicted keypoints. We alternatively searched the best candidate parent node for all the predicted keypoints, and removed the surplus keypoints from their candidate children list, until no better hypothesis could be found. The best candidate parent node was defined as the keypoint which was in the correct class and match the predicted offset vector best. The euclidean distance between the predicted offset and the actual offset was used to measure the match.

Architecture of the network We proposed an architecture (Fig. 5.5b) for the network. Depthwise separable convolutions [94] were used as the basic building blocks to reduce the computational cost. Following [61], we used location-withheld maxpool-

ing to improve the localization accuracy, which preserved indices at the max pooling layers of the encoder and passed them to the corresponding up-sampling layers of the decoder.

5.3.3 Keypoints tracking

In order to count pigs across video frames, an efficient on-line tracking method was employed to associate pig keypoints temporally. This method took the grouped pig keypoints for single frames as input, and then assigned a unique identification number (id) to each pig across frames. This problem was formulated as a bipartite graph matching based energy maximization problem. The estimated pig candidates C^t at frame t were then associated with the previous pig candidates C^{t-1} at frame $t-1$ by bipartite graph matching.

$$\begin{aligned} \widehat{s} = \arg\max_{s} & \sum_{C_i^t \in C^t} \sum_{C_j^{t-1} \in C^{t-1}} \Psi_{C_i^t, C_j^{t-1}} \times s_{C_i^t, C_j^{t-1}} \\ s.t. \quad & \forall C_j^{t-1} \in C^{t-1}, \sum_{C_i^t \in C^t} s_{C_i^t, C_j^{t-1}} \in \{0,1\}, \\ & \forall C_i^t \in C^t, \sum_{C_j^{t-1} \in C^{t-1}} s_{C_i^t, C_j^{t-1}} \in \{0,1\}, \end{aligned} \tag{5.2}$$

where C_j^{t-1} was the j^{th} pig candidate in C^{t-1} and C_i^t was the i^{th} pig candidate in C^t. $s_{C_i^t, C_j^{t-1}} \in \{0,1\}$ was a binary variable and indicates if C_j^{t-1} and C_i^t were associated. The potential $\Psi_{C_i^t, C_j^{t-1}}$ represented the similarity measurements between C_j^{t-1} and C_i^t.

$$\Psi_{C_i^t, C_j^{t-1}} = \lambda_1 \Psi^A_{C_i^t, C_j^{t-1}} + \lambda_2 \Psi^L_{C_i^t, C_j^{t-1}}, \tag{5.3}$$

where $\Psi^A_{C_i^t, C_j^{t-1}}$ represented the keypoints appearance similarities between candidates. And $\Psi^L_{C_i^t, C_j^{t-1}}$ implied the spatial similarities. λ_1 and λ_2 were hyper-parameters to

balance the contributions of the two terms.

The spatial similarities was calculated as the l_2 distance between the propagated C_j^{t-1} spatial location and encoded C_i^t center location. $\Psi^L_{C_i^t, C_j^{t-1}} = \left\| P(L(C_j^{t-1})) - L(C_i^t) \right\|^2$. The appearance similarity was calculated as the the l_2 distance across all keypoints embedded deep features between C_i^t and C_j^{t-1}.

$$\Psi^L_{C_i^t, C_j^{t-1}} = \sum_{n=1}^{5} \lambda_n^L \left\| K^n_{C_j^{t-1}} - K^n_{C_i^t} \right\|^2, \tag{5.4}$$

where K^n represented the n^{th} keypoint deep appearance feature obtained from convolution layer before the last upsampling layer of our keypoints CNN. λ_n^L were the hyper-parameters balancing the weights.

The aforementioned bipartite graph matching problem was solved using Hungarian method.

5.3.4 Spatial-aware temporal response filtering

Traditionally, video-based counting methods [85, 89] counted the number of unique tracklet ID as the final counting results. These methods were suitable for the cases, where the target objects were stationary and object occlusion was very rare. In the large-scale pig counting scenario, however, pigs moved fast in different directions, and the same pig will often walked out of the camera view and came back again. In addition, the indoor building structures (e.g. the feeding machine) would sometimes block large part of the camera view causing severe occlusions. Occlusions across long frames will cause tracking failure, and break trajectory of one single object into two or more. In these cases, counting the number of unique tracklet IDs would suffer from large false positive errors. To overcome these limitations, we represented a novel spatial-aware temporal response filtering (STRF) method to perform on-line counting, while minimizing the false positives.

The STRF took the tracking trajectories for all previous frames as input, and output the final counting number. It consisted of two steps: 1) spatial encoding; and 2) temporal response filtering. The spatial encoding stage processed each video frame independently, and each detected pig candidate in the frame was assigned a code number based on their spatial locations. The temporal response filtering stage examined each candidate's trajectory across time and obtained a count number, $count_i \in \{0, 1, -1\}$, for this single candidate. The final counting result was the sum of all count number for all candidates: $\sum_{i=0}^{N} count_i$.

As shown in Fig. 5.6a, the spatial encoding stage divided one image frame into activated zone and deactivated zone by an activity scanning line. This scanning line was stationary in a single frame, but served to scan the whole pig house moving with the inspection robot. For all detected pig candidates, activity codes will be assigned based on which activity zone these pigs were in. In our work, pigs in activated zone were assigned code value 0, and pigs in deactivated zone were assigned code value 1. Deactivated zone indicated that all candidates inside have already been counted by the algorithm; and the candidates in activated zone would be counted when the activity scanning line scanned through them.

In the temporal response filtering step, lists of spatial codes in temporal order were generated for each trajectory. One trajectory had one list of spatial codes, and each element of the list corresponded to a time point. Fig. 5.6c illustrated one example of one single pig trajectory from time point $t - 6$ to time point t, where the blue color represented code 1 and the red color represented code 0. As it was shown, the generated temporal code was [0, 0, 0, 0, 1, 1, 1] from $t - 6$ to t. The final count for this trajectory $count_i$ was obtained as the sum of the first order difference of the temporal codes. In this case, the count would be 1, which indicated that this pig was scanned once (from deactivated zone into activated zone) and the total count should be added by 1. Similarly, Fig. 5.6d showed a pig trajectory with code [1, 0, 0, 0, 1,

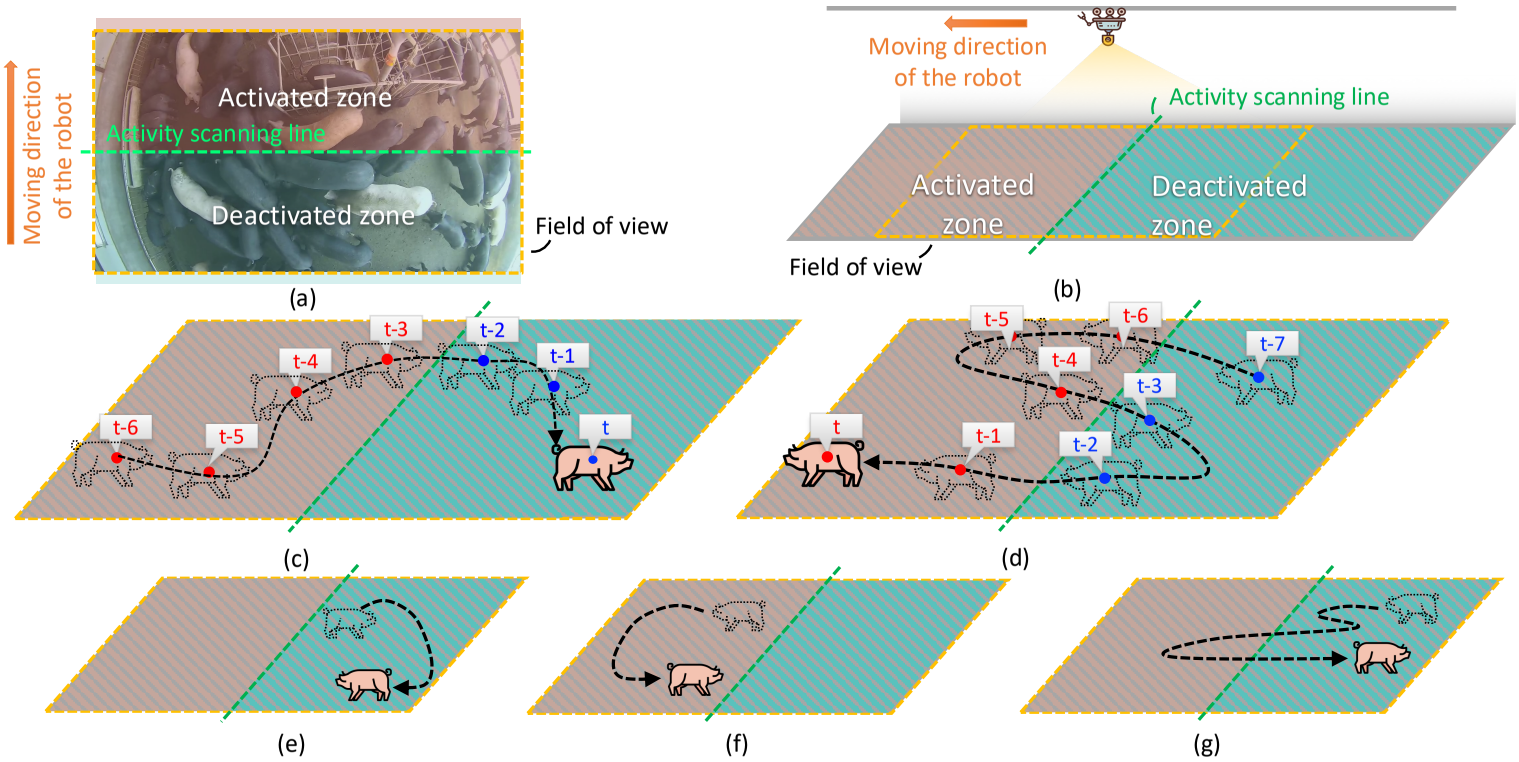


Figure 5.6: Illustration of STRF method. (a) spatial encoding defined activated zone and deactivated zone, and an activity scanning line. (b) Activity scanning line moved with the camera to scan the whole pig house. (c) An example of pig tracking trajectory with count 1. (c) An example of count -1 pig tracking trajectory. (d)(e)(f) Examples of count 0 tracking trajectory.

1, 0, 0] and sum of the the first order difference inferred that the count was -1. This meant that this pig, which has been counted before, moved from scanned zone to to-be scanned zone. Thus, the total count should minus 1. This design enabled the algorithm to avoid false positives counting caused by pig movements into/out camera view. Fig. 5.6e-g showed examples when the pig trajectory count was 0. Fig. 5.6e-f represented pig trajectories that never went across the scanning line. Fig. 5.6g represented cases where the trajectory started and ended in the same activity zone. These examples demonstrated that STRF would not be influenced by the tracking failures (e.g. broke one trajectory into several cased by occlusion) that happened only in one single zone. In this study, a low-pass filter with window size of 5 was applied before the first order differential calculation. This low-pass filtering step was designed to avoid the trajectory jitter near the activity scanning line. The final counting result for the whole video also added the number of detected candidates in deactivated zone of the beginning frame and the number of detected candidates in activated zone of the ending frame.

5.4 Experiments

We collected 51 videos by inspection robots installed in pig grouping houses of two different pig farming corporations. All videos were originally recorded at 1280×720 resolution with frame rate of 25 f/s. For this study, we first resized the video frame to 360×640, and then cropped them to 352×640. All experiments in this work used this resolution. Each video (pig house) had 120∼250 pigs. The length of the videos ranged from 2 minutes to 4 minutes. We randomly split these videos into three subsets, 21 for training, 5 for validation and 25 for testing. The ground truth were provided by workers, who counted the pigs inside the grouping houses when the videos were recorded. We report the error of pig counting using Mean Absolute Percentage Error

(MAPE) and Mean Absolute Error (MAE).

5.4.1 Comparison with human reader

To demonstrate the effectiveness of out method, we compared the performance of our counting system with human readers on test dataset. There were three readers for this study. The readers were required to provide count results by watching the same top-down view videos as the input of the algorithm. There were no time limits for the reading process, and the readers were allowed to pause, rewind, replay the video and took notes for unlimited times. Each reader estimated the pig counts for all the videos in the test datasets. The counting error for both the proposed method and human reader were evaluated using mean absolute percentage error (MAPE) and mean absolute error (MAE). The three readers have MAPE of 11.0%, 17.4%, and 15.9%; and MAE of 12.6, 26.3 and 25.2, respectively. The average time that the human readers have spent on per video is around 1.5 hours. In contrast, our method had MAPE of 2.67%, and MAE of 3.32, which significantly outperformed the human readers.

5.4.2 Ablation study

To validate our proposed CNN architecture for keypoints detection, we compared our method with UNet [95] and stacked Hourglass network [96] using the same train, validation and test datasets. Both methods were modified to fit our pixel-level keypoints detection pipelines. Following [97], the cropping operators was removed from UNet and 7 UNet-submodules were used. The Stacked Hourglass network tested had two hourglass stacked. The Percent of Detected Joints (PDJ) [80] was used as the evaluation metric. One keypoint was considered as detected if the distance between the predicted keypoint and the ground truth was smaller than a fraction of the total

Table 5.1: Comparison of keypoint detection results.

Metric	PDJ@0.1			PDJ@0.2			FLOPs	#Param.
Keypoint type	mid	quarter	end	mid	quarter	end		
UNet [95]	0.938	0.935	0.895	0.971	0.980	0.953	23G	42M
Hourglass [96]	0.913	0.905	0.866	0.954	0.968	0.932	23G	3.6M
Ours	**0.962**	**0.964**	**0.934**	**0.991**	**0.992**	**0.978**	**15G**	**3.3M**

Table 5.2: Comparison of pig counting

Method	MAPE	MAE
SSD [18] + Tracking	327%	412
YOLOv3 [20] + Tracking	247%	368
Proposed Keypoint Detector + Tracking	152%	191
SSD [18] + Tracking + STRF	10.1%	12.2
YOLOv3 [20] + Tracking + STRF	5.35%	7.00
Proposed Keypoint Detector + Tracking + STRF	**2.67**%	**3.32**

length of the skeleton of the pig. As shown in Table 5.1, our method achieved better keypoints detection accuracy for all 5 body parts with significantly less computation cost and smaller parameter size.

We also compared our bottom-up detection method with SSD [18] and YOLOv3 [20] using top-down bounding boxes detection metric: mean average precision with 0.5 IOU (mAP@0.5). The proposed bottom-up approach did not directly output bounding boxes of pig. Thus, we used keypoints/skeleton bounding boxes instead. It should be noted that the keypoints bounding boxes are more strict and harder to predict, and our network was never trained for the bounding boxes detection task. SSD achieved 73.3% mAP while YOLOv3 achieves 79.7% mAP. Our method had 84.3% mAP. Although more challenging, our method showed better performance. It should be noted that a large part of the detection failures happened around image boudaries where large fisheye distortion and image cutoff happened. Due to the design of STRF methods, most of the failures will not influence the final counting result.

To evaluate the effectiveness of the STRF method, we compared the counting results with and without STRF using our detection method, SSD and YOLOv3,

resepctively. Table 5.2 showed that the MAPE and MAE are significantly small when using STRF. And our method achieved better performance with/without STRF compared with SSD or YOLOv3.

5.4.3 Runtime analysis

We analyzed the runtime performance of our method using the test dataset. On desktop computer, it achieved 3.42 frames per second (FPS) running speed with a Intel i7-6850K CPU and 32GB DDR4 2133MHz Memory. When accelerated by a single NVIDIA GeForce GTX 1080Ti GPU, it achieved 82.6 FPS. The proposed counting algorithm has also been deployed on two different edge computing devices. It achieved 0.625 FPS on a Firefly-RK3399 platform, which had a 2GB Memory and a Rockchip RK3399 CPU. On NVIDIA Jetson Nano platform, it achieved 3.19 FPS with a 4GB memory, a quad-core ARM A57 CPU and 128 CUDA cores.

5.5 Conclusion

In this work, we presented a hardware configuration and novel efficient algorithm for pig counting in large grouping houses. An inspection robot with a monocular fisheye camera was installed on the roof with rails, along which the root could move back and forth to collect top-down view videos. A novel efficient bottom-up CNN detection approach was developed to first detect pigs from the crowd. Second, a online tracking method was employed to associate pig ID temporally. A novel STRF method was proposed to calculate the final pig counts, while significantly avoid false positive counting due to tracking failure or large pig movements. The low computation cost design significantly reduce the computation time and model size. This counting algorithm has been deployed in edge computing device of the inspection robot, and achieved counting accuracy superior to human readers.

Chapter 6

Conclusion

In this dissertation, we formulated the animal and plant inventory as computer vision problems under different circumstances and proposed four novel approaches to detect, recognize, track and count the objects of interest. We evaluated our approaches on the data collected from the real world. The experimental results supported the advantage of the approaches.

In Chapter 2 we introduced our approach for vegetation coverage estimation. The usage of handcrafted features like HOG and LBP alleviates the burden of labeling work. Without any training data, the detector achieves a 93% success rate. The segmentation model generates color model for the cover board at test time. Such design is robust to the lighting condition change and achieves 3% mean absolute error rate in the test set collected by the Missouri Department of Conservation.

In Chapter 3 we proposed density transformer or DENT, which is a density-based algorithm for tree counting in aerial images. The visual features are extracted by a novel Multi-Receptive Field network, which has multiple concentric fields of view, focusing on the local patch and wide context respectively. The visual features and a count query embedding are encoded using a transformer encoder, where the useful information is exchanged between the features based on their pair-wise relationship.

Finally, the hidden states of the transformer are decoded by a Density Map Generator and a tree counter. They generate the predicted tree density map and tree count. We used the Yosemite Tree Dataset built by ourselves and the cross-site NeonTreeEvaluation Dataset to evaluate DENT. Although the mainstream of the existing tree counting algorithms is detection-based, our DENT significantly outperformed all the other state-of-art detectors in the experiments. DENT also outperformed the other density-based algorithms.

In Chapter 4 we introduced our approach for fish classification using cameras on fishing boats. The proposed system uses two branches to make predictions. The instance-level branch focus on the individual fish. It localizes the fish and normalizes the pose of the fish. The pose-normalized fish is categorized by CNN classifiers. The image-level branch makes predictions based on all the context in the images in case that the individual fish is difficult to recognize. The predictions from the two branches are adaptively weighted averaged. The weights are determined by how familiar the context is. We evaluated this approach in the "The Nature Conservancy Fisheries Monitoring" competition and archieved top 0.7% rank among 2293 solutions.

In Chapter 5 we introduced our video-based pig counting system. The hardware consists of an on-rail robot carrying a fish-eye lens with a top-down view and an ARM chip for edge computing. The algorithm consists of an efficient fully convolutional network for bottom-up keypoint detection. From the perspective of the algorithm, we proposed an efficient bottom-up keypoint detection approach to localize the pigs. The detected pigs are tracked using bipartite graph matching. We also proposed a Spatial-aware Temporal Response Filtering algorithm to suppress the missing pigs and false alarms. The proposed approach not only outperformed the other methods but also the human volunteers.

In sum, the proposed methods and their performance opened up new prospects of the computer vision applications in plant and animal inventory.

Bibliography

[1] Ba, J., Kiros, J.R., Hinton, G.E.: Layer normalization. ArXiv **abs/1607.06450** (2016)

[2] Weinstein, B.G., Marconi, S., Bohlman, S., Zare, A., White, E.: Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks. Remote Sensing **11**(11) (2019)

[3] Jorgensen, C.F., Stutzman, R.J., Anderson, L.C., Decker, S.E., Powell, L.A., Schacht, W.H., Fontaine, J.J.: Choosing a diva: a comparison of emerging digital imagery vegetation analysis techniques. Applied Vegetation Science **16**(4) (2013) 552–560

[4] Limb, R., Hickman, K., Engle, D., Norland, J., Fuhlendorf, S.: Digital photography: Reduced investigator variation in visual obstruction measurements for southern tallgrass prairie. Rangeland Ecology & Management **60** (09 2007) 548–552

[5] Leis, S.A., Morrison, L.W.: Field test of digital photography biomass estimation technique in tallgrass prairie (2011)

[6] Morrison, L.W.: Observer error in vegetation surveys: a review. Journal of Plant Ecology **9**(4) (12 2015) 367–379

[7] Robel, R.J., Briggs, J.N., Dayton, A.D., Hulbert, L.C.: Relationships between visual obstruction measurements and weight of grassland vegetation. Journal of Range Management **23** (1970) 295–297

[8] Nudds, T.D.: Quantifying the vegetative structure of wildlife cover. Wildlife Society Bulletin (1973-2006) **5**(3) (1977) 113–117

[9] Cagney, J., Cox, S.E., Booth, D.T.: Comparison of point intercept and image analysis for monitoring rangeland transects (2011)

[10] Booth, D.T., Cox, S.E., Johnson, D.E.: Detection-threshold calibration and other factors influencing digital measurements of ground cover (2005)

[11] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Conference on Computer Vision and Pattern Recognition. (2005) 886–893

[12] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Advances in neural information processing systems **25** (2012) 1097–1105

[13] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)

[14] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778

[15] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2015) 1–9

[16] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 2818–2826

[17] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. arXiv preprint arXiv:1506.01497 (2015)

[18] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: European conference on computer vision, Springer (2016) 21–37

[19] Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. (2017) 2980–2988

[20] Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767 (2018)

[21] Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. ArXiv **abs/1904.07850** (2019)

[22] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In Vedaldi, A., Bischof, H., Brox, T., Frahm, J.M., eds.: Computer Vision – ECCV 2020, Cham, Springer International Publishing (2020) 213–229

[23] Mubin, N.A., Nadarajoo, E., Shafri, H.Z.M., Hamedianfar, A.: Young and mature oil palm tree detection and counting using convolutional neural network deep learning method. International Journal of Remote Sensing **40**(19) (2019) 7500–7515

[24] Li, W., Fu, H., Yu, L., Cracknell, A.: Deep learning based oil palm tree detection and counting for high-resolution remote sensing images. Remote Sensing **9**(1) (2017) 22

[25] Xia, M., Li, W., Fu, H., Yu, L., Dong, R., Zheng, J.: Fast and robust detection of oil palm trees using high-resolution remote sensing images. In: Automatic Target Recognition XXIX. Volume 10988., International Society for Optics and Photonics (2019) 109880C

[26] Machefer, M., Lemarchand, F., Bonnefond, V., Hitchins, A., Sidiropoulos, P.: Mask r-cnn refitting strategy for plant counting and sizing in uav imagery. Remote Sensing **12**(18) (2020) 3015

[27] Weinstein, B.G., Marconi, S., Bohlman, S., Zare, A., White, E.: Individual tree-crown detection in rgb imagery using semi-supervised deep learning neural networks. Remote Sensing **11**(11) (2019) 1309

[28] Roslan, Z., Awang, Z., Husen, M.N., Ismail, R., Hamzah, R.: Deep learning for tree crown detection in tropical forest. In: 2020 14th International Conference on Ubiquitous Information Management and Communication (IMCOM), IEEE (2020) 1–7

[29] Zheng, J., Li, W., Xia, M., Dong, R., Fu, H., Yuan, S.: Large-scale oil palm tree detection from high-resolution remote sensing images using faster-rcnn. In: IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium, IEEE (2019) 1422–1425

[30] Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 589–597

[31] Babu Sam, D., Surya, S., Venkatesh Babu, R.: Switching convolutional neural network for crowd counting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 5744–5752

[32] Li, Y., Zhang, X., Chen, D.: Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 1091–1100

[33] Cao, X., Wang, Z., Zhao, Y., Su, F.: Scale aggregation network for accurate and efficient crowd counting. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 734–750

[34] Liu, W., Salzmann, M., Fua, P.: Context-aware crowd counting. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2019) 5099–5108

[35] Djerriri, K., Ghabi, M., Karoui, M.S., Adjoudj, R.: Palm trees counting in remote sensing imagery using regression convolutional neural network. In: IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium, IEEE (2018) 2627–2630

[36] Yao, L., Liu, T., Qin, J., Lu, N., Zhou, C.: Tree counting with high spatial-resolution satellite imagery based on deep neural networks. Ecological Indicators **125** (2021) 107591

[37] Weinstein, B.G., Marconi, S., Bohlman, S.A., Zare, A., White, E.P.: Cross-site learning in deep learning rgb tree crown detection. Ecological Informatics **56** (2020) 101061

[38] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In Guyon, I., Luxburg,

U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R., eds.: Advances in Neural Information Processing Systems. Volume 30., Curran Associates, Inc. (2017)

[39] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical image computing and computer-assisted intervention, Springer (2015) 234–241

[40] Rowley, H.A., Baluja, S., Kanade, T., et al.: Human face detection in visual scenes. Citeseer (1995)

[41] Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001. Volume 1., IEEE (2001) I–I

[42] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Volume 1., Ieee (2005) 886–893

[43] Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence **32**(9) (2009) 1627–1645

[44] Harzallah, H., Jurie, F., Schmid, C.: Combining efficient object localization and image classification. In: 2009 IEEE 12th international conference on computer vision, IEEE (2009) 237–244

[45] Lowe, D.: Object recognition from local scale-invariant features. In: Proceedings of the Seventh IEEE International Conference on Computer Vision. Volume 2. (1999) 1150–1157 vol.2

[46] Pollock, R.: The automatic recognition of individual trees in aerial images of forests based on a synthetic tree crown image model. PhD thesis, University of British Columbia (1996)

[47] Larsen, M., Rudemo, M.: Using ray-traced templates to find individual trees in aerial photographs. In: Proceedings of the Scandinavian Conference on Image Analysis. Volume 2. (1997) 1007–1014

[48] Vibha, L., Shenoy, P.D., Venugopal, K., Patnaik, L.: Robust technique for segmentation and counting of trees from remotely sensed data. In: 2009 IEEE International Advance Computing Conference, IEEE (2009) 1437–1442

[49] Hung, C., Bryson, M., Sukkarieh, S.: Vision-based shadow-aided tree crown detection and classification algorithm using imagery from an unmanned airborne vehicle. In: 34th International Symposium for Remote Sensing of the Environment. (2011)

[50] Manandhar, A., Hoegner, L., Stilla, U.: Palm tree detection using circular autocorrelation of polar shape matrix. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences (2016) 465–472

[51] Wang, Y., Zhu, X., Wu, B.: Automatic detection of individual oil palm trees from uav images using hog features and an svm classifier. International Journal of Remote Sensing **40**(19) (2019) 7356–7370

[52] Li, W., Fu, H., Yu, L.: Deep convolutional neural network based large-scale oil palm tree detection for high-resolution remote sensing images. In: 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), IEEE (2017) 846–849

[53] Li, W., Dong, R., Fu, H., Yu, L.: Large-scale oil palm tree detection from high-resolution satellite images using two-stage convolutional neural networks. Remote Sensing **11**(1) (2019) 11

[54] Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely connected convolutional networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2017) 4700–4708

[55] Freudenberg, M., Nölke, N., Agostini, A., Urban, K., Wörgötter, F., Kleinn, C.: Large scale palm tree detection in high resolution satellite images using u-net. Remote Sensing **11**(3) (2019) 312

[56] Miyoshi, G.T., Arruda, M.d.S., Osco, L.P., Marcato Junior, J., Gonçalves, D.N., Imai, N.N., Tommaselli, A.M.G., Honkavaara, E., Gonçalves, W.N.: A novel deep learning method to identify single tree species in uav-based hyperspectral images. Remote Sensing **12**(8) (2020) 1294

[57] Araujo, A., Norris, W., Sim, J.: Computing receptive fields of convolutional neural networks. Distill **4**(11) (2019) e21

[58] Parmar, N.J., Vaswani, A., Uszkoreit, J., Kaiser, L., Shazeer, N., Ku, A., Tran, D.: Image transformer. In: International Conference on Machine Learning (ICML). (2018)

[59] Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[60] Lei, J., Wang, L., Shen, Y., Yu, D., Berg, T.L., Bansal, M.: Mart: Memory-augmented recurrent transformer for coherent video paragraph captioning. arXiv preprint arXiv:2005.05402 (2020)

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

[62] Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch. (2017)

[63] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: 2009 IEEE conference on computer vision and pattern recognition, Ieee (2009) 248–255

[64] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3) (2015) 211–252

[65] Glorot, X., Bengio, Y.: Understanding the difficulty of training deep feedforward neural networks. In: Proceedings of the thirteenth international conference on artificial intelligence and statistics, JMLR Workshop and Conference Proceedings (2010) 249–256

[66] Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)

[67] White, D., Svellingen, C., Strachan, N.: Automated measurement of species and length of fish by computer vision. Fisheries Research **80**(2) (2006) 203–210

[68] Spampinato, C., Giordano, D., Di Salvo, R., Chen-Burger, Y.H.J., Fisher, R.B., Nadarajan, G.: Automatic fish classification for underwater species behavior understanding. In: Proceedings of the first ACM international workshop on

Analysis and retrieval of tracked events and motion in imagery streams, ACM (2010) 45–50

[69] Ravanbakhsh, M., Shortis, M.R., Shafait, F., Mian, A., Harvey, E.S., Seager, J.W.: Automated fish detection in underwater images using shape-based level sets. The Photogrammetric Record **30**(149) (2015) 46–62

[70] Svellingen, C., Totland, B., White, D., Øvredal, J.T.: Automatic species recognition, length measurement and weight determination, using the catchmeter computer vision system, ICES (2006)

[71] : Large scale visual recognition challenge 2017 (2017)

[72] Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results. http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html

[73] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755

[74] Redmon, J., Farhadi, A.: Yolo9000: better, faster, stronger. arXiv preprint arXiv:1612.08242 (2016)

[75] Kazemi, V., Sullivan, J.: One millisecond face alignment with an ensemble of regression trees. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 1867–1874

[76] Ren, S., Cao, X., Wei, Y., Sun, J.: Face alignment at 3000 fps via regressing local binary features. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2014) 1685–1692

[77] Cao, X., Wei, Y., Wen, F., Sun, J.: Face alignment by explicit shape regression. International Journal of Computer Vision **107**(2) (2014) 177–190

[78] Gavves, E., Fernando, B., Snoek, C.G., Smeulders, A.W., Tuytelaars, T.: Fine-grained categorization by alignments. In: Proceedings of the IEEE International Conference on Computer Vision. (2013) 1713–1720

[79] Krause, J., Jin, H., Yang, J., Fei-Fei, L.: Fine-grained recognition without part annotations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2015) 5546–5555

[80] Toshev, A., Szegedy, C.: Deeppose: Human pose estimation via deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2014) 1653–1660

[81] Sapp, B., Taskar, B.: Modec: Multimodal decomposable models for human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2013) 3674–3681

[82] Johnson, S., Everingham, M.: Clustered pose and nonlinear appearance models for human pose estimation. (2010)

[83] Gardenier, J., Underwood, J., Clark, C.: Object detection for cattle gait tracking. In: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE (2018) 2206–2213

[84] Pezzuolo, A., Guarino, M., Sartori, L., González, L.A., Marinello, F.: On-barn pig weight estimation based on body measurements by a kinect v1 depth camera. Computers and Electronics in Agriculture **148** (2018) 29–36

[85] Liu, X., Chen, S.W., Liu, C., Shivakumar, S.S., Das, J., Taylor, C.J., Underwood, J., Kumar, V.: Monocular camera based fruit counting and mapping

with semantic data association. IEEE Robotics and Automation Letters **4**(3) (2019) 2296–2303

[86] Tian, M., Guo, H., Chen, H., Wang, Q., Long, C., Ma, Y.: Automated pig counting using deep learning. Computers and Electronics in Agriculture **163** (2019)

[87] Shen, Z., Xu, Y., Ni, B., Wang, M., Hu, J., Yang, X.: Crowd counting via adversarial cross-scale consistency pursuit. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2018) 5245–5254

[88] Sindagi, V.A., Patel, V.M.: Generating high-quality crowd density maps using contextual pyramid cnns. In: Proceedings of the IEEE International Conference on Computer Vision. (2017) 1861–1870

[89] Liu, X., Chen, S.W., Aditya, S., Sivakumar, N., Dcunha, S., Qu, C., Taylor, C.J., Das, J., Kumar, V.: Robust fruit counting: Combining deep learning, tracking, and structure from motion. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE (2018) 1045–1052

[90] Hodgson, J.C., Baylis, S.M., Mott, R., Herrod, A., Clarke, R.H.: Precision wildlife monitoring using unmanned aerial vehicles. Scientific reports **6** (2016) 22574

[91] Rivas, A., Chamoso, P., González-Briones, A., Corchado, J.: Detection of cattle using drones and convolutional neural networks. Sensors **18**(7) (2018) 2048

[92] Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2017) 7291–7299

[93] Papandreou, G., Zhu, T., Chen, L.C., Gidaris, S., Tompson, J., Murphy, K.: Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In: Proceedings of the European Conference on Computer Vision (ECCV). (2018) 269–286

[94] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. arXiv preprint arXiv:1704.04861 (2017)

[95] Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: MICCAI. (2015) 234–241

[96] Newell, A., Yang, K., Deng, J.: Stacked hourglass networks for human pose estimation. In: European conference on computer vision, Springer (2016) 483–499

[97] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition. (2017)

[98] Olofsson, K., Wallerman, J., Holmgren, J., Olsson, H.: Tree species discrimination using z/i dmc imagery and template matching of single trees. Scandinavian Journal of Forest Research **21**(S7) (2006) 106–110

[99] Hassaan, O., Nasir, A.K., Roth, H., Khan, M.F.: Precision forestry: trees counting in urban areas using visible imagery based on an unmanned aerial vehicle. IFAC-PapersOnLine **49**(16) (2016) 16–21

[100] Bazi, Y., Malek, S., Alajlan, N., AlHichri, H.: An automatic approach for palm tree counting in uav images. In: 2014 IEEE Geoscience and Remote Sensing Symposium, IEEE (2014) 537–540

[101] Aliero, M.M., Mukhtar, N., Al-Doksi, J.: The usefulness of unmanned airborne vehicle (uav) imagery for automated palm oil tree counting. Journal of Forestry. Researchjournali **1**(1) (2014)

[102] Rizky, A.P., Solahudin, M., et al.: Analysis of aerial photo for estimating tree numbers in oil palm plantation. In: IOP Conference Series: Earth and Environmental Science. Volume 284., IOP Publishing (2019) 012003

[103] Santoso, H., Tani, H., Wang, X.: A simple method for detection and counting of oil palm trees using high-resolution multispectral satellite imagery. International journal of remote sensing **37**(21) (2016) 5122–5134

[104] Kattenborn, T., Sperlich, M., Bataua, K., Koch, B.: Automatic single tree detection in plantations using uav-based photogrammetric point clouds. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences **40**(3) (2014) 139

[105] Khan, S., Gupta, P.K.: Comparitive study of tree counting algorithms in dense and sparse vegetative regions. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences (2018)

[106] Cheang, E.K., Cheang, T.K., Tay, Y.H.: Using convolutional neural networks to count palm trees in satellite images. arXiv preprint arXiv:1701.06462 (2017)

[107] Rezaee, M., Zhang, Y., Mishra, R., Tong, F., Tong, H.: Using a vgg-16 network for individual tree species detection with an object-based approach. In: 2018 10th IAPR Workshop on Pattern Recognition in Remote Sensing (PRRS), IEEE (2018) 1–7

[108] Zortea, M., Nery, M., Ruga, B., Carvalho, L.B., Bastos, A.C.: Oil-palm tree detection in aerial images combining deep learning classifiers. In: IGARSS 2018-

2018 IEEE International Geoscience and Remote Sensing Symposium, IEEE (2018) 657–660

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

VITA

Guang Chen was born in Liaoning Province, China. He received his B.S. Degree at the China University of Mining and Technology in 2011. He received his M.S. Degree at the University of Missouri, Columbia, MO 65211, USA in 2015. And then he started to pursue a Ph.D. Degree in Computer Science at the University of Missouri. His research interests involve Computer Vision and Deep Learning, especially object detection and video understanding. During the Ph.D. program, he has four papers published and a paper under review. During the Ph.D. program, he also have three internships: at the DevTech group of NVIDIA, Beijing, China in 2017; at the Imaging Science group of Tempus Labs, Chicago, Illinois, USA in 2018; and at the Computer Vision group of JD.COM Silicon Valley Research Center, Mountain View, California, USA in 2019. And then he became a regular research scientist at JD.COM in August 2019. In June 2021, he started to work at the Computer Vision group of the Intelligent Creation team at ByteDance, Mountain View, California, USA. He will continue his job at ByteDance after graduation from the University of Missouri.