Fast Rotated Bounding Box Annotations for Object Detection

A thesis submitted to the
College of Graduate and Postdoctoral Studies
in partial pulfillment of the requirements
for the degree of Master of Science
in the Department of Computer Science
University of Saskatchewan
Saskatoon

By
Minhajul Arifin Badhon

Abstract

Traditionally, object detection models use a large amount of annotated data and axis-aligned bounding boxes (AABBs) are often chosen as the image annotation technique for both training and predictions. The purpose of annotating the objects in the images is to indicate the regions of interest with the corresponding labels. Accurate object annotations help the computer vision models to understand the distinct patterns of the image features to recognize and localize different classes of objects. However, AABBs are often a poor fit for elongated object instances. It's also challenging to localize objects with AABBs in densely packed aerial images because of overlapping adjacent bounding boxes. Alternatively, using rectangular annotations that can be oriented diagonally, also known as rotated bounding boxes (RBB), can provide a much tighter fit for elongated objects and reduce the potential bounding box overlap between adjacent objects. However, RBBs are much more time-consuming and tedious to annotate than AABBs for large datasets.

In this work, we propose a novel annotation tool named as FastRoLabelImg (Fast Rotated LabelImg) for producing high-quality RBB annotations with low time and effort. The tool generates accurate RBB proposals for objects of interest as the annotator makes progress through the dataset. It can also adapt available AABBs to generate RBB proposals. Furthermore, a multipoint box drawing system is provided to reduce manual RBB annotation time compared to the existing methods. Across three diverse datasets, we show that the proposal generation methods can achieve a maximum of 88.9% manual workload reduction. We also show that our proposed manual annotation method is twice as fast as the existing system with the same accuracy by conducting a participant study. Lastly, we publish the RBB annotations for two public datasets in order to motivate future research that will contribute in developing more competent object detection algorithms capable of RBB predictions.

Acknowledgements

I would like to express my heartfelt thanks and sincere gratitude to my supervisor Prof. Ian Stavness for his invaluable supervision and guidance that he has so kindly extended to me throughout the entire period of my study. This journey would not be complete without having the enormous support and excellent mentorship from him. I truly appreciate his contribution and efforts towards the completion of my thesis and other research projects throughout the program.

I would also like to take this opportunity to thank Prof. Mark Eramian and Prof. Kevin Stanley. I am grateful for the valuable knowledge I gained from their courses which I was able to leverage to the benefit of my research.

I would also like to thank my lab mates for extending their kindness, friendship and help that have made my journey fun and comfortable.

Finally, I am thankful to my beloved parents, parents-in-law, wife and my friends; their unconditional love and unwavering support are the reasons I have come so far.

To Mom and Dad, my greatest blessings from the Almighty.

Contents

List of Tables

List of Figures

List of Abbreviations

AABB	Axis-aligned Bounding Box
RBB	Rotated Bounding Box
FastRoLabelImg	Fast Rotated LabelImg
GWHD	Global Wheat Head Detection
CARPK	Car Parking
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
SVM	Support Vector Machine
MSE	Mean Squared Error
ROI	Region of Interest
RPN	Region Proposal Network
NMS	Non Maximum Suppression
R-CNN	Region-based Convolutional Neural Network
HRoI	Horizontal Region of Interest
RRoI	Rotated Region of Interest
FIAT	Fast Image Data Annotation Tool
IAD	Intelligent Annotation Dialogues
DOTA	Dataset of Object deTection in Aerial images
IoU	Intersection over Union
ANOVA	Analysis of Variance
RM	Repeated Measures
UI	User Interface
YOLACT	You Only Look At CoefficienTs
SOLO	Segmenting Objects by Locations

1 Introduction

1.1 Motivation

Object detection is an important and well-studied computer vision task. With the emergence of large scale benchmark datasets [34, 60, 86], deep learning based detection algorithms have advanced rapidly [82, 81, 63]. The performance of these detection algorithms depends on the quality of the training data and associated annotations. For detection tasks, images need to be accurately annotated to indicate the location and class of the objects of interest for the algorithms to learn and predict new images. To annotate object locations in the images we can use different techniques that vary in terms of precision and annotation effort. The benchmark datasets primarily contain natural images annotated with axis-aligned bounding boxes (AABB), which do not localize elongated and rotated objects well (Figure 1.1a). This is particularly problematic in aerial image datasets, due to the immense scale variations as well as random rotations of the objects therein (e.g., cars, ships). Also, AABBs fail to isolate the objects properly in crowded scenes as adjacent bounding boxes heavily overlap each other (Figure 1.1b). The inclusion of background features or adjacent objects within an AABB potentially degrades detection performance and precise localization. To overcome this issue, object annotation with rotated bounding boxes (RBB) has been explored for detecting ships [64, 107, 112, 106, 65], vehicles [92], text [108, 114] and other categories of objects [28, 109, 110, 104] and these models showed improved detection performance compared to AABBs. RBBs can capture the true aspect ratio of the objects and contain fewer background pixels. Also, RBBs can localize objects precisely in dense or occluded scenes where objects are very close to each other or possibly overlapping (Figure 1.1). Therefore, object detection algorithms with RBB annotations perform better in remote sensing images compared to the traditional AABB annotations.

Most previous RBB-based object detection methods are based on supervised learning and require a large amount of annotated images with RBBs around objects for training. Though image annotation tools [9, 11] can support automated annotation using pre-trained models, manual annotation often plays a vital role in the image annotation process in case of images with new object classes or to correct inaccurate automatic annotations. However, Creating RBB annotations manually is more tedious and time-consuming than annotating images with AABBs using the current tools. Annotators can create an AABB by specifying any two opposite corner points with two simple mouse clicks. But, they need to perform multiple tasks to create each of the RBBs. These tasks involve creating AABBs as the first step and then iteratively rotate and resize the bounding boxes until they are a good fit (this procedure is referred to as the *current system* in this work). Annotation tools that support directly drawing RBBs are not widely available. In this work, we aim to fill this gap by integrating new RBB annotation features to the existing open-source annotation tool LabelImg [6, 7]. The modified LabelImg tool named as FastRoLabelImg (Fast Rotated LabelImg) will be made publicly available

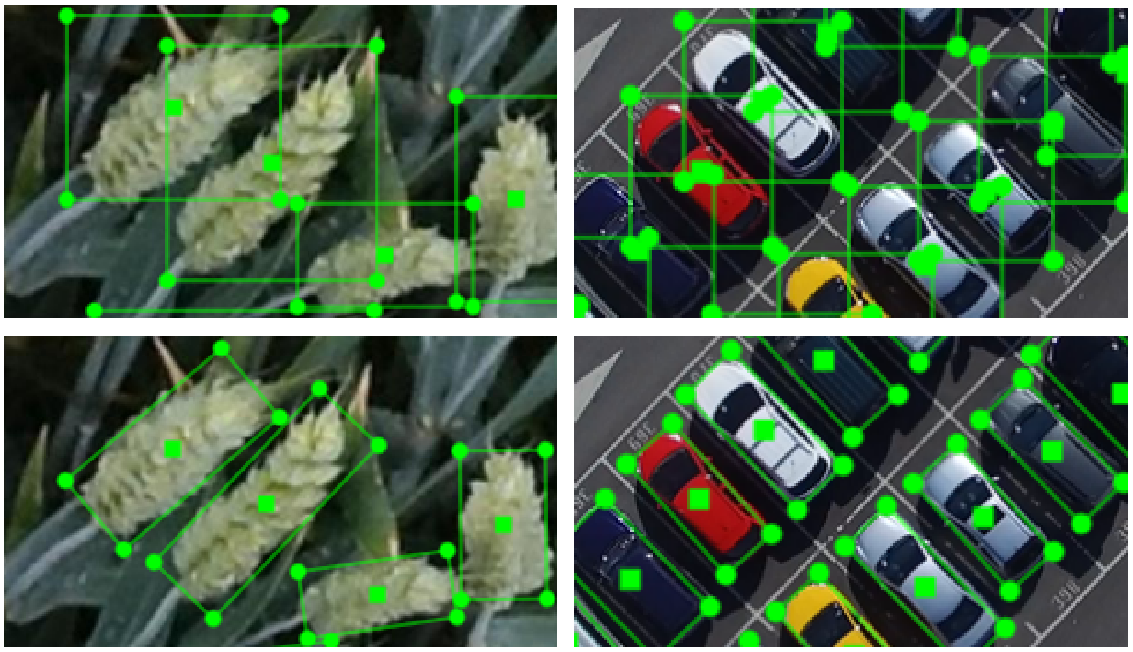

(a) (b)

Figure 1.1: Annotation issues with AABBs. a) AABBs fail to localize elongated and rotated objects well b) AABBs have higher overlap (top) compared to rotated boxes (bottom).

to enable annotators to efficiently annotate large image datasets with RBBs.

1.2 Problem Statement

Though AABBs are cheap and easy to obtain compared to RBBs, the use of such annotations hampers accurate detection of target objects in many cases. Several works have been done to solve this issue with RBB annotations. Researchers are also working continuously to refine the model architectures [109, 110] for even more robust and improved detections with RBBs. Therefore, it is important to prepare large datasets with RBB annotations to either solve a new task using the developed models or improve the existing models to gain higher accuracy. However, obtaining RBB annotations is costly with professional data labeling services [3] and time-consuming with open-source annotation tools [7]. To address this situation, our objective is to develop an open-source image annotation tool with specific features to obtain high-quality RBB annotations with less time and effort.

1.3 Proposed Solution

To reduce annotation workload, our tool automatically provides RBB proposals assisted by computer vision techniques. We use pre-trained models as well as an iteratively trained object detector. The latter model incrementally learns about the dataset as new annotations are available. As the annotator goes through the images in the dataset, the model is continually and seamlessly retrained and provides improved RBB proposals. In this work, we mainly focus on minimizing the annotation time and effort instead of developing a novel detector. When the model is retrained, it doesn't block the annotator from continuous interactions with the tool. On the other hand, the pre-trained model can help to generate RBB proposals for those classes of objects on which the model is already trained on. We also explore ways in which *a priori* AABB annotations can be used to improve the accuracy of RBB proposals, for datasets with existing AABB annotations.

We evaluate the effectiveness of our developed methods using images drawn from the GWHD (Global Wheat Head Detection) dataset [25], CARPK (Car Parking Lot) dataset [47], and Airbus ship detection challenge dataset [10]. The selected images ($\sim$1000) of the GWHD and CARPK dataset are manually annotated using RBBs and publicly made available at `https://doi.org/10.6084/m9.figshare.13014230.v1` to encourage further research in this domain. The proposed RBB proposal generation methods attain a minimum workload reduction of 65.8% for the wheat dataset and a maximum of 88.9% for the ship dataset.

In addition to providing RBB proposals, FastRoLabelImg also includes a multipoint box drawing mechanism to speed up manual bounding box creation. This system simplifies the process of drawing RBBs which is otherwise quite cumbersome in the existing tools. In this system, the annotators can create an RBB with three or four mouse clicks and fewer or no subsequent adjustments. Therefore, we speed up the annotation process by optimizing the manual effort instead of computational complexity. We evaluate our proposed RBB drawing interface through a participant study and show that our two proposed variants are significantly faster than the traditional method and achieve the same level of annotation accuracy.

1.4 Contributions

The contributions of this paper include:

1. We design a multipoint drawing system to accelerate the process of manually creating RBB annotations. The multipoint drawing system provides various configurations which differ based on the number of required mouse clicks. During an annotation session, the annotator can choose the suitable configuration depending on the characteristics of the target objects.

2. We assess the effectiveness of the proposed manual RBB drawing system by conducting a participant study. The result show that one of the configurations of the drawing system is as accurate as the current system while being three times faster.

3. We develop multiple methods based on object detection models and image processing techniques to automatically generate RBB proposals for specific object classes.

4. Our automatic RBB proposal generation methods can integrate pre-existing AABB annotations to improve the proposals. The pre-existing annotations can be used to either directly adapt to RBB proposals or filter out the false-negative proposals.

5. We evaluate the proposed RBB generation methods on subsets of three publicly available datasets (GWHD dataset, CARPK dataset, Airbus ship dataset). The methods achieve over 50% manual workload reduction on all the datasets.

6. We manually annotate subsets of two datasets (GWHD dataset, CARPK dataset) with RBBs and make those publicly available.

1.5 Thesis Organization

This thesis is organized as follows:

- Chapter 2. Background: provides an introduction to the fundamental concepts entailed in this work such as machine learning techniques, building blocks of a convolutional neural network, and popular methods of object detection and segmentation.
- Chapter 3. Literature Review: discusses the related works incorporating human-machine collaboration to minimize the image annotation cost as well as existing bounding box annotation tools and their limitations.
- Chapter 4. Methods: describes our proposed method for efficient image annotation with reduced human effort.
- Chapter 5. FastRoLabelImg Interface: presents the UI components of FastRoLabelImg to interact with our developed features.
- Chapter 6. Data Acquisition: demonstrates the procedures to build three datasets for later experiments.
- Chapter 7. Experiments and Results: outlines the experimental setup, evaluation metrics, quantitative results, and related discussions.
- Chapter 8. Conclusion and Future Work: highlights the contributions of this work with summarized results and outlines some future works to get rid of the current limitations.

2 Background

2.1 Machine Learning

Machine learning is a branch of AI which focuses on developing algorithms that can learn from the given data by capturing the inherent patterns in the data and gain the ability to make reliable educated guesses on unseen data without being explicitly programmed with a defined ruleset. In a traditional algorithm, the computer system is fed with the data and a set of instructions/rules formed by a human expert and upon execution the computer outputs the desired result. On the other hand, a machine learning algorithm derives the set of rules from the given input data and desired results in an automatic fashion as shown in Figure 2.1. But, to achieve that kind of derivation power the modern algorithms need to experience an enormous amount of data. The predictive accuracy of the algorithms also improve with increasing exposure based on suitable tasks-specific performance measurement metrics. The data can be of different forms like numbers, words, images, etc.

Machine learning has been a driving force in bringing unthinkable efficiency in our personal and work lives. For example, machine learning has its usage in self-driving cars [79], product recommendation systems [94], speech recognition [72], language translation [29], health care [46], image recognition [30] and so many other things. In a self-driving car, machine learning is used to detect humans and other objects. Here, after having seen multiple examples of humans and other objects, the algorithm gains the ability to detect similar targets in unseen scenarios.

Machine learning is majorly sub-categorized into three types as shown in Figure 2.2. This section will provide an introduction to one of the types named supervised learning.

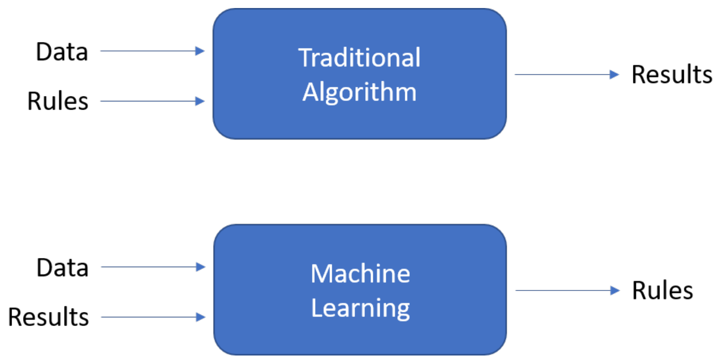


Figure 2.1: Traditional algorithm vs Machine learning algorithm [24].

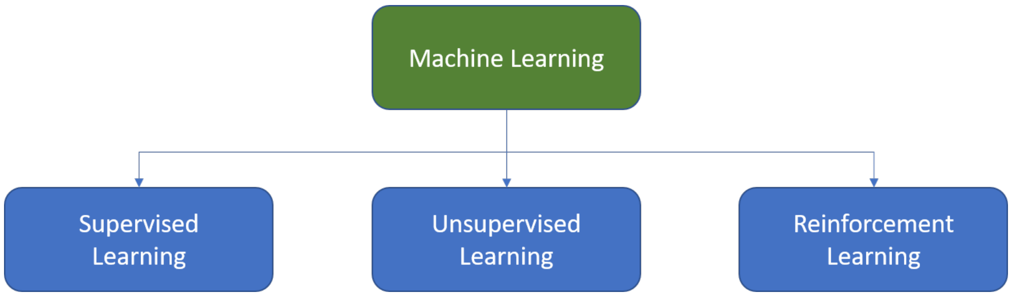


Figure 2.2: Types of machine learning.

2.1.1 Supervised Learning

In supervised learning, as the name suggests, the model is supervised using labeled data during its learning process. Therefore, in this case, each observation in the training dataset is associated with the correct label which guides the model to fine tune itself iteratively. Once the learning step is done, the trained model can assign labels to future observations.

Formally, supervised learning tries to learn a mapping function that can correlate the input data to the output variables. It can be expressed as,

$$Y = f(x) \tag{2.1}$$

Therefore, the model can predict the output label Y when presented with an input data x using the mapping function f that is approximated during the training phase.

Supervised learning problems can be grouped into two subcategories:

- Classification: When the output variable is a category, such as “cat” or “dog” or “happy” and “unhappy”, then it’s called a classification problem. In an image classification problem, a classifier can be used to predict the category of the object present in the image.

- Regression: When the output variable is a continuous number, such as house prices in dollars, then it’s called a regression problem. Therefore, a regressor can be used to determine the coordinates of the corner points of a bounding box that encapsulates an object in an image.

Section 1.4 shows how classifiers and regressors can be used as building blocks for developing an image-based object detection algorithm.

2.2 Learning Techniques

There are many techniques for learning a model from the given data. Two of the most relevant techniques are discussed in this section.

2.2.1 Active Learning

In supervised learning, the machine learning algorithms demand a large amount labeled data to build effective models. But, getting annotations for unlabeled data can easily become very expensive. Active learning is a type of machine learning that reduces this cost by allowing the learner algorithm to interactively request a human annotator to label the most informative samples. Therefore, the learner has the ability to choose the subset of the data that will benefit its accuracy the most. The learning process can be controlled with some stopping criteria such as the total number of iterations or when the model can't improve the accuracy above a certain threshold after an iteration.

There are three categories of active learning:

- Stream-based selective sampling: In this process, the learner is presented with a sample in each iteration and it can either discard or request to label the data based on the informativeness. The informativeness of the data can be determined using a query strategy (e.g. Least confidence, margin sampling).
- Pool-based sampling: This type of active learning is mostly used. Here, the model is first trained with some manually labeled data. Then, the model determines the set of most informative examples and requests the labels.
- Membership query synthesis: In this setting, the model is allowed to create it's own synthetic data and request the annotator to label it.

2.2.2 Online Learning

Traditional machine learning methods assume that the whole data is available before the training. Therefore, the learning of the model happens in a batch mode by minimizing the loss on the batches of training data. But, in the real-world often the system generates/gathers a large amount of data continuously. This tends to make the models trained on fixed data quickly obsolete as the patterns in the data may change over time. Online learning enables us to train a model incrementally in near real-time as new data becomes available sequentially. Moreover, it is useful when the available data is so large that batch learning is computationally infeasible. Online learning needs less processing power and it is more memory efficient. It achieves memory efficiency as the data already used to train the model once doesn't need to be stored for another pass. However, online learning is vulnerable to catastrophic forgetting [68] which refers to the issue of forgetting prior learned information as new information is learned. Incremental learning algorithms can solve this issue by adjusting the model to new data while preserving the current knowledge.

2.3 Convolutional Neural Networks (CNN)

2.3.1 Neural Network

In the human brain, multiple neurons are arranged together to form a network of neurons that can receive information from our senses and transfers the processed information from one neuron to another. The structure of the neural network in machine learning is motivated by this concept of the network of neurons. A neural network is a combination

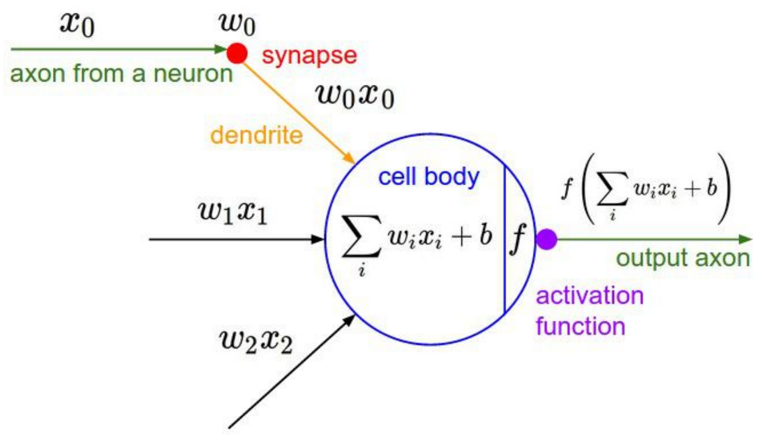


Figure 2.3: Neuron structure with three inputs ($x_i, i = 0, 1, 2$). The output is determined after normalizing the weighted summation (with bias b) of the inputs with an activation function [59].

of similarly connected artificial neurons that can pass data through the network and the flow of data is regulated by weights. A single neuron takes one or more inputs, multiplies the real-valued inputs with the corresponding weights, and sums the multiplication results along with a bias. Then, the result can be normalized with an activation function to obtain the output of the neuron (optional). The structure of a neuron with N inputs is displayed in Figure 2.3 and the performed mathematical operations can be expressed as,

$$y(x, w) = f(\sum_{i=1}^{N} w_ix_i + b) \tag{2.2}$$

where x, w, b represent vectors of inputs, weights, and bias respectively.

In it's simplest form, a neural network has an input layer, a hidden layer, and an output layer. Each layer can have one or more neurons. Also, the network can have multiple hidden layers. Here, the word "hidden" just means not input or output. The neurons in any two adjacent layers are fully pairwise connected. In other words, all the neurons of a layer i are fully connected to all the neurons of the next layer $i + 1$. But, the neurons in the same layer don't have any connection. These connections allow passing the data forward in the network from one layer to another. Such networks are called "feedforward neural networks" because the network doesn't have any cycle. Figure 2.4 illustrates a simple neural network. The left-most layer is the input layer and the right-most layer is the output layer. The layers in between are the hidden layers. The neural network shown in the figure can be depicted as a neural network with three layers. Note that the input layer is not counted as the first layer of the network.

A neural network with at least one hidden layer can approximate any continuous function [22] that can map an input X to an output Y as in $f : X \rightarrow Y$. As each neuron in the network acts as a small computational unit, the function $f^n(.)$ learned by the layer n constitutes different functions and can be expressed as,

$$f(x) = f^n(f^{n-1}(...f^2(f^1(x)))) \tag{2.3}$$

Neural Network learns the mapping function by adjusting the weights and biases iteratively as it goes through the training observations. In supervised learning, each training observation x_i has a known desired output y_i. To quantify

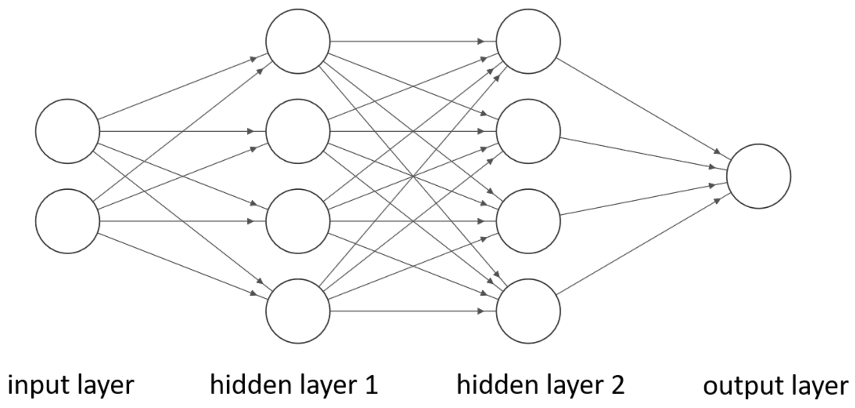


Figure 2.4: A 3-layer neural network.

the performance of the network, we use a cost function. An example of such a cost function(C) is MSE (Mean Squared Error) which can be written as,

$$C(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y_i} - y_i)^2 \tag{2.4}$$

where θ is a vector of all weights and biases in the network, m is the number of training examples, $\hat{y_i}$ is the predicted output for the i^{th} example and y_i is the ground truth output for the i^{th} example. To allow learning for the model, we minimize the cost function using an optimization algorithm (e.g. gradient descent [85]) by finding a set of optimal values for the weights and biases. The gradient descent algorithm uses the gradient of the cost function to update the parameters in the opposite direction of the gradient iteratively. With each step, we move towards the global minima of the cost function. To compute the gradient of the cost function for each of the parameters of the model efficiently, the backpropagation algorithm [45] is used.

To train a model using a neural network, the dataset is usually divided into three partitions: training data, validation data, test data. While the training data is used to train the model, the validation data is used to evaluate the performance of the model at regular intervals to allow potential tuning of the hyperparameters. The network can learn more complex functions with a higher number of hidden layers or neurons. But, this may introduce overfitting to the training data and loose generalization power on new observations. We can use regularization [40] or dropout [36] techniques to handle possible overfitting. At the end of the training, the test data is used to measure how well the trained model performs on unseen observations.

2.3.2 Convolutional Neural Network

Convolutional Neural Network (ConvNet or CNN) is a class of deep learning algorithm that is used in the field of computer vision and natural language processing. In computer vision, CNN takes images as inputs and automatically extracts useful features to enable various image analysis tasks such as image recognition, object detection, semantic segmentation, etc. Like a nerual network, a CNN is also composed of artificial neurons. But, neurons in a CNN may connect to a subset of neurons of the previous layer (local connectivity) unlike full connectivity of neurons of the

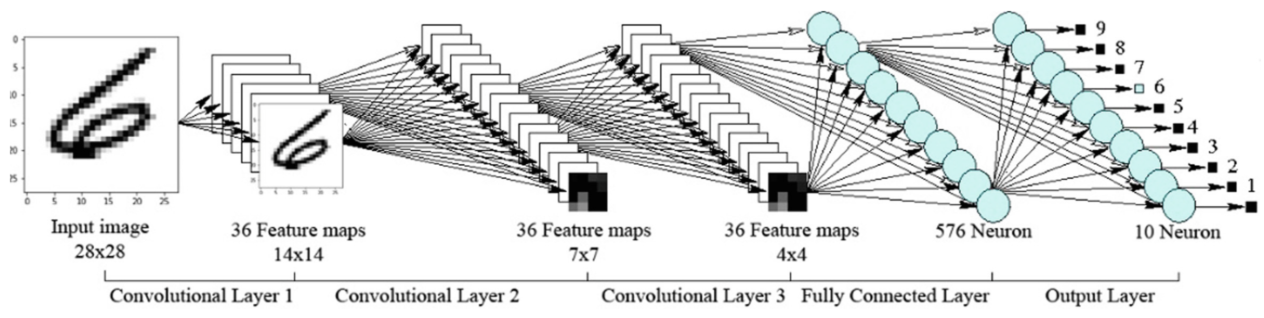


Figure 2.5: A CNN to classify handwriten digits. Reprinted by permission from Springer Nature: Springer Nature, Soft Computing in Data Science, [88] (Exploratory Analysis of MNIST Handwritten Digit for Machine Learning Modelling, Shamsuddin, Mohd Razif and Abdul-Rahman, Shuzlina and Mohamed, Azlinah), Copyright (2019).

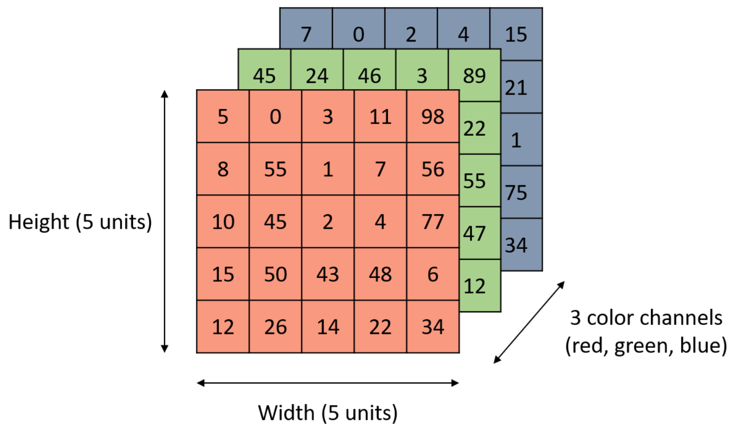


Figure 2.6: Representation of a RGB image of size 5×5.

adjacent layers in a neural network. A CNN performs a series of convolution and pooling operations for hierarchical feature abstraction, then the output of the last pooling layer is flattened and passed through some fully connected layers to obtain the output scores. Figure 2.5 demonstrates the architecture of a CNN to classify handwritten digits.

Computers see an RGB image as a collection of pixel values ranges from 0 to 255 arranged in a three-dimensional matrix. If a RGB image has a size of 28×28, then it can be represented with a matrix of size $28 \times 28 \times 3$. Here, 3 is the number of channels in an RGB image considering the red, green, and blue channels respectively. However, a binary image has only one channel. Representation of a RGB image is shown in Figure 2.6.

It is possible to input an image to a regular multi-layer neural network by flattering the high dimensional array (Figure 2.7). But, the network fails to capture the spatial structures present in the image and it hurts the performance of the model. Also, a regular neural network is not scalable to high-resolution images. For example, For example, if we process an image of size 200×200, a fully connected neuron of the first hidden layer will need to learn 120,000 weights. If the image size is increased to 1000×1000, that neuron will have 3 million weights. Also, typically we want more of such neurons. Therefore, the number of parameters rises rapidly causing the required processing time and

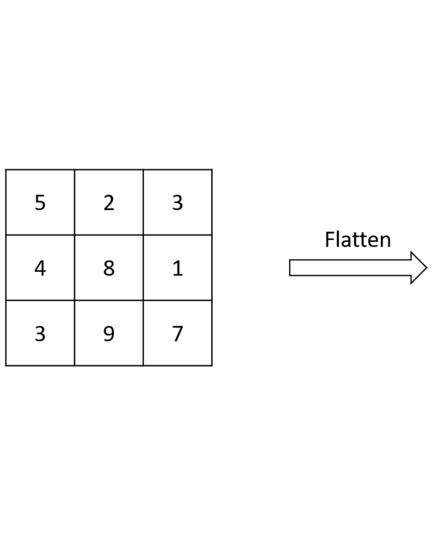


Figure 2.7: Flattening a 3×3 matrix to a 9×1 vector.

computational power to increase aggressively. However, CNN can understand the temporal and spatial dependencies in an image. Besides, it vastly reduces the number of parameters of the network without losing features by allowing local connectivity and sharing of parameters (e.g. weights, biases).

2.3.3 Basic Layers of CNN

A convolutional neural network has four basic components that are used as building blocks of the network. These are the convolutional layer, non-linear activations, pooling layer, and fully connected layer. Now, we briefly describe the operations of these layers:

Convolutional Layer

A convolutional layer performs a linear operation named "Convolution" on the input volume with a filter/kernel to detect local dependencies in the input. A kernel is a matrix of learnable weights. The filter slides over the input and at each spatial location an element-wise product is calculated using the kernel values and the corresponding pixel values of the input and summed up with a bias to get a value in the output (feature map) related to that location. The step size between two consecutive spatial locations is called stride. Before convolution, the input can be padded (e.g. zero-padding adds some rows and columns with zero values around the input) to avoid losing information in the edges. Without padding, the output feature map will be smaller in size. Multiple filters can be used to capture different characteristics of the input and the output feature maps are stacked. If a $N \times N$ input matrix is convolved with a $F \times F$ filter, padding P and stride S, the size $R \times R$ of the output feature map can be calculated as,

$$R = \frac{N + 2P - F}{S} + 1 \tag{2.5}$$

Figure 2.8 illustrates the process of a convolutional layer on a 3×3 input with a 2×2 kernel to obtain a 2×2

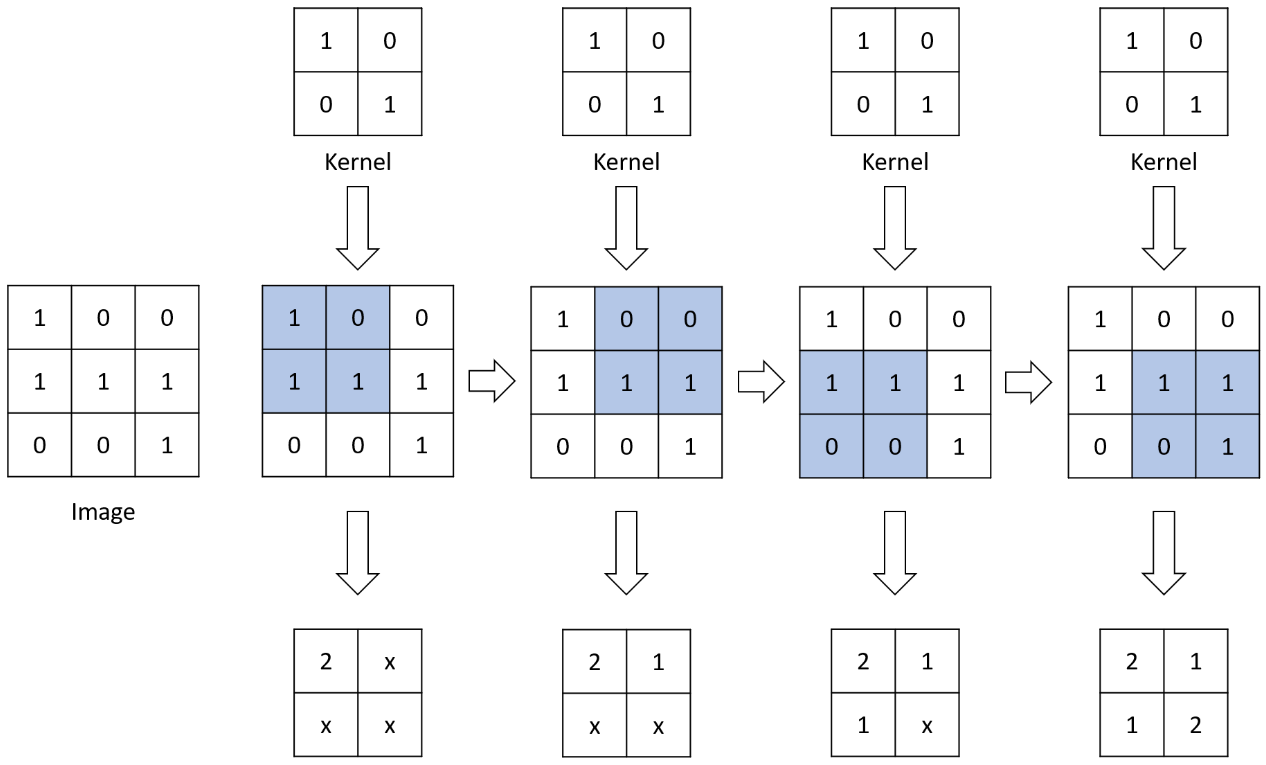


Figure 2.8: A convolutional layer [31].

feature map.

If the input is an RGB image, the filters used in a convolutional layer will also be three dimensional. The number of channels in a filter will be the same as the input. Convolutional layers vastly reduce the number of parameters to be learned in the network as the filters are shared across all the image positions in a convolutional layer and don't depend on the number of inputs. For example, consider an image of shape $200 \times 200 \times 3$. Now, for a fully connected neural network that has 10,000 neurons in the first hidden layer, there will be a total of $120,000$ input neurons $\times 10,000$ hidden neurons $= 1,200,000,000$ parameters to be learned only in the first layer. In contrast, if the same image is presented to a convolutional layer and 32 filters of size 5×5 are applied to extract features, we have only $32 \times (5 \times 5 \times 3 + 1) = 2,432$ parameters to learn including bias.

A sequence of convolutional layers in a CNN creates a hierarchical abstraction of the input image. The earlier layers in the network detect low-level features such as edges, colors, corners etc. from the image. In the successive layers, it starts recognizing more complex patterns like shapes or a combination of several low-level features. Eventually, the deep layers capture high-level features like digits, face parts, and others depending on the image.

Non-linear Activations

As convolution is a linear operation, an activation function is applied to the output of the convolution element-wise to introduce nonlinearity to the model. It allows the model to learn more complex functions that can approximate the real-world data better. Earlier on, non-linear activation functions such as sigmoid or hyperbolic tangent (tanh) were

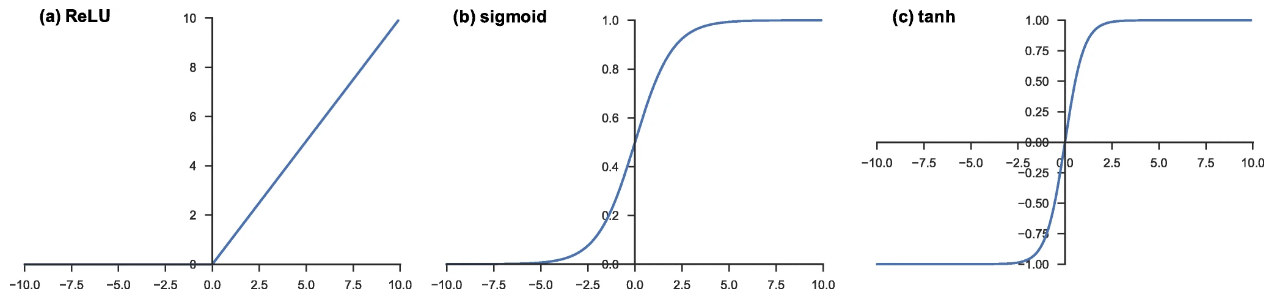


Figure 2.9: Commonly used non-linear activation functions. a) ReLU b) sigmoid c) tanh [103].

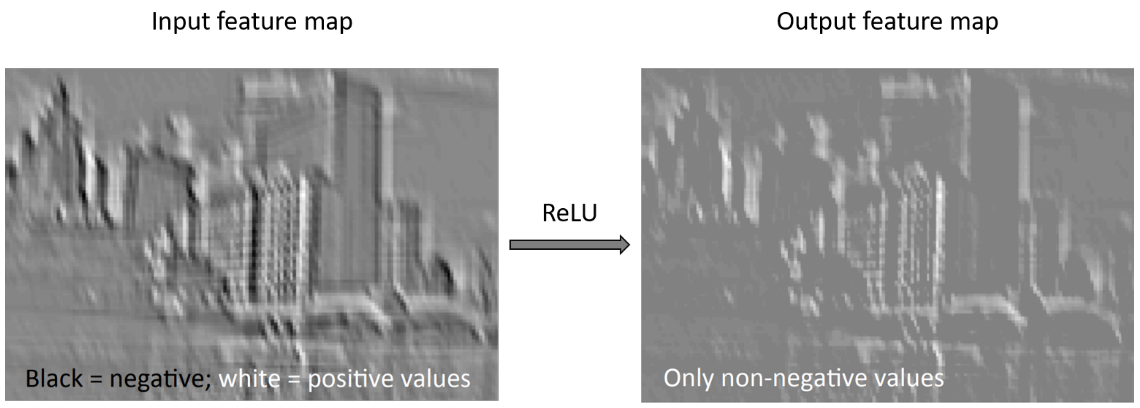


Figure 2.10: ReLU operation [52].

used. The sigmoid function is computed as,

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

and, the tanh function is computed as,

$$f(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{2.7}$$

Though the tanh function works better than the sigmoid function, both face slow gradient descent (the model takes longer to converge) when the input is too small or high. One of the most commonly used activation functions is ReLU (Rectified Linear Unit) which is free of slow convergence issues and computes several times faster [57] than others. It is also robust to the vanishing gradient problem. ReLU can be expressed as,

$$f(x) = max(0, x) \tag{2.8}$$

Therefore, ReLU converts any negative inputs to 0 and just pass the input to the output otherwise. Figure 2.10 shows the impact of adjusting the negative values on an input feature map.

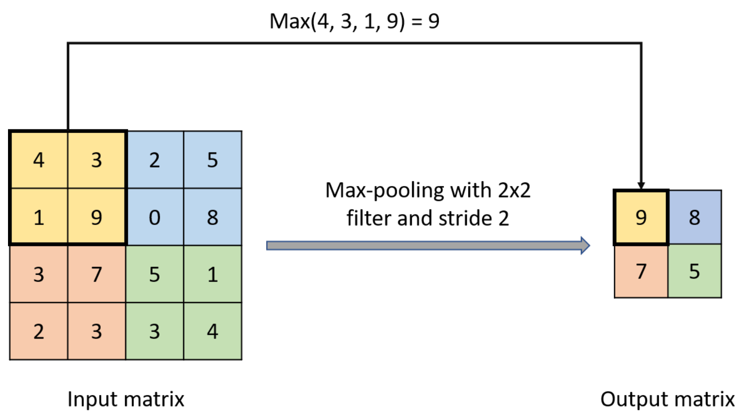


Figure 2.11: Max pooling operation [58].

Pooling Layer

A pooling layer is used to reduce the size of the feature maps without giving up the dominant features. The downsampling operation not only introduces some rotational and positional invariance but also lessens the number of learnable parameters in the subsequent layers. A pooling layer is usually placed after a convolutional layer in the network. There are different types of pooling operations (e.g. max-pooling, average-pooling) based on the retrieval of values from a pooling window of size $N \times N$. Max-pooling takes the maximum value of the pooling windows whereas average-pooling takes the average of the values in the pooling window. Typically, max-pooling is widely used over average-pooling as it can remove noisy activations in addition to the dimensionality reduction. Figure 2.11 shows an example of max-pooling on a 4×4 input using a 2×2 pooling window with a stride of 2.

Fully Connected Layer

The fully connected layer is just like the traditional feed-forward neural network and placed at the end of the CNN. The high-level features extracted by the previous layers are flattened to a column vector to fed to the fully connected layer. For a classification task, it maps the input features to output class scores using softmax as the activation function. Softmax can take a vector of real-valued numbers and normalize it a vector of n probabilities that sum to one. Here, n represents the number of classes in the dataset.

2.4 Object Detection

Object detection is one of the challenging tasks of computer vision. In object detection, the goal is to determine the class and spatial location of objects that belong to a set of predefined categories in images. Usually, the extent of each of the objects is expressed with axis-aligned bounding boxes that fit the objects tightly. The bounding boxes can be of different sizes and aspect ratios. For example, an object detector that is responsible for detecting cats and dogs in the images will output a set of bounding boxes containing either cat or dog for the given image as shown in Figure 2.12.

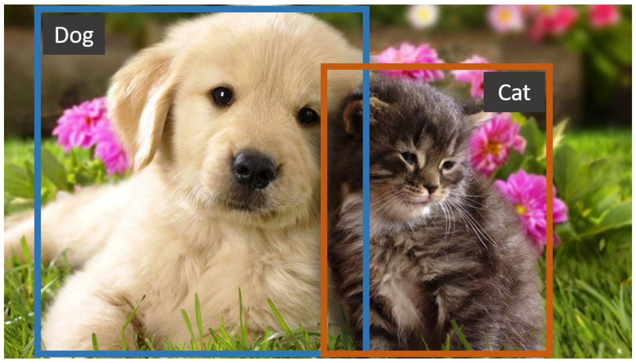


Figure 2.12: Object detection with axis-aligned bounding boxes [4].

Object detection serves as the foundation in solving different complex tasks such as pedestrian detection, counting people or objects, object tracking, face detection, vehicle detection, visual search engine etc.

In traditional algorithms, handcrafted features are used in combination with traditional classifiers such as Support Vector Machine (SVM) [70] and regression models to identify and localize objects of interest in the images. Feature descriptors such as SIFT [66], SURF [14], HOG [23], etc. are used to form the definition of each class of objects and later matched to a new image to figure out the groups of pixels that represent an object class. But, these approaches get increasingly complex with a higher number of classes and involve manual decision-making of what features are useful for different object classes.

In contrast, deep learning-based object detection models utilize Convolutional Neural Networks to learn the distinct patterns that exist for different classes of objects with the expense of a large amount of annotated example images. This type of learned feature representation technique is more flexible compared to the 'programmed' features in traditional methods. In addition, With sufficient computational power and data, these deep learning-based models allow end-to-end learning and higher accuracy than traditional approaches. The automatic feature extraction process, as well as the overall accuracy can be further improved by using rotated bounding boxes to locate the objects in the aerial images.

In the following sub-sections, we discuss the deep learning-based object detection architecture that is used to generate rotated bounding box proposals for objects in the developed image annotation tool. First, we describe the architecture of the popular object detection framework named Faster R-CNN. Subsequently, we briefly discuss the RoI (Region of Interest) Transformer that can be embedded to the Faster R-CNN architecture to allow object detection using rotated bounding boxes.

2.4.1 Faster R-CNN

Faster R-CNN [82] emerged as the third iteration of the R-CNN series of object detectors in 2015. The R-CNN family of detectors first came to the scene with the introduction of region proposal based framework R-CNN [38] by Ross Girshick in 2014. The R-CNN framework uses selective search to generate a predefined number of candidate region proposals which are then warped to a fixed dimension and individually passed through a convolutional neural network to extract respective output feature vectors. In the next stage, category-specific SVM classifiers are used to classify the objects inside the proposed regions and the bounding box regressor is used to adjust the bounding boxes accurately. Training of the R-CNN detector is multistage and costly in terms of space and time. After a year, Fast R-CNN [37] is developed to improve the R-CNN network by allowing to share convolutional computation while extracting features from multiple region proposals. Additionally, Fast R-CNN introduces a region of interest(RoI) pooling layer which enables working with input images of different sizes and produces fixed-size feature vectors to be used by the fully connected layers for classification and regression. Despite all of this progress, these detectors are slow in terms of test-time predictions because of the expensive region proposal generation step that runs on the CPU compared to the GPU accelerated computations of convolutional and fully connected layers. Faster R-CNN brings the concept of convolution-based region proposal network to solve the computational bottleneck and unifies different modules (e.g. feature extraction, proposal generation, bounding box regression, etc.) to permit end-to-end learning.

Briefly, the Faster R-CNN object detection network has a feature extraction network followed by two subnetworks. One of those is a Region Proposal Network (RPN) that can produce candidate proposals as possible regions where the objects of interest may be found. The other one is the region-based convolutional neural network that integrates a classifier to mark the bounding boxes with object classes and a bounding box regressor to convert region proposals to precise box predictions.

Feature Extraction

To obtain robust salient features from the input images, the Faster R-CNN framework makes use of pre-trained convolutional neural networks. These networks are classification networks pre-trained on large datasets like ImageNet [86]. Faster R-CNN takes the output of an intermediate layer of the feature extraction network for future steps. The extracted feature map from an image typically has a much smaller width and height compared to the original images due to the dimensionality reduction capability of the pooling layers and greater depth because of the number of filters being learned in the process. Moreover, the feature map preserves the relative positions of the objects in the x, y dimension while containing the feature related information of the image at its depth.

Specifically, Faster R-CNN uses ZF and VGG pre-trained on ImageNet in its original implementation for the experiments. The architecture of the VGG-16 [91] network is shown in Figure 2.13. In the paper, the output of the conv5/conv5_1 layer is used to obtain the intermediate feature map as marked in the same figure. To reduce the overfitting nature of shallow networks, researchers tend to use deeper networks. Deeper networks have a higher number of convolutional layers which means that there is a higher number of parameters to learn during the training. This allows better learning for the model as at every layer the network learns a new more abstract representation of the input image. In our implementation, we use ResNet [43] as the backbone feature extraction network because of its high capacity. Besides, ResNet solves the problem of vanishing gradients which often occurs for deep networks by using residual skip connections.

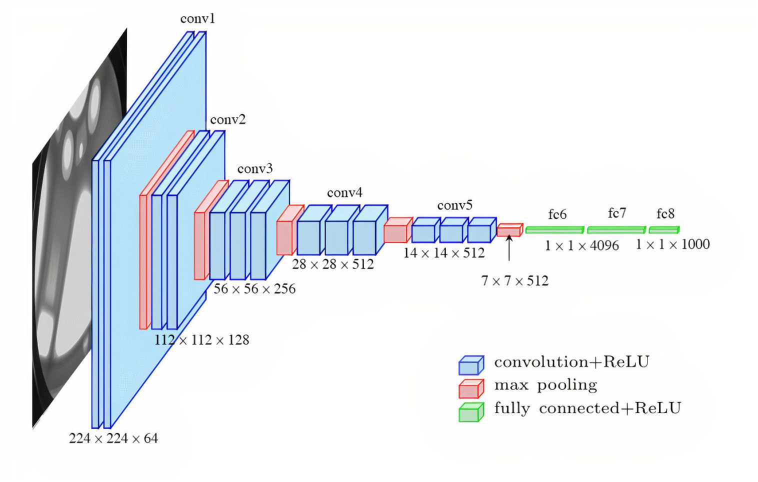


Figure 2.13: VGG architecture as proposed in [91]. Image source [35], copyright ©2017 IEEE.

Anchors

One of the important ideas of the Faster R-CNN is anchors. Anchors help to handle varying numbers of objects of divergent scales and aspect ratios in an image. After getting the convolutional feature map, a sliding window is moved over the image, and anchor centers are positioned at the center of the window. Faster R-CNN uses a sliding window of size 3×3 representing the width and height. A set of K anchors is defined at each point of the feature map. The number of anchors K depends on two configurable parameters which are scale and aspect ratio. These parameters are configured based on the object sizes in the training dataset. In the official implementation, Faster R-CNN uses three different scales and aspect ratios giving rise to a combination of $K = 9$ anchor boxes at each point of the feature map (Figure 2.14). If the width and height of the feature map are W and H respectively, then there will be a total of $N = W \times H \times K$ generated anchors. Anchor centers can be projected back the original image using the subsampling ratio of the feature extractor to present the anchor boxes on the original image. As Faster R-CNN utilizes only convolutional and pooling layers, the subsampling ratio can be found using the width and height of the original image and the downsampled feature map. If the subsampling ratio is r, then the anchor centers are r pixels apart in the original image.

Region Proposal Network

By taking the extracted feature map from the image as input and with the help of the generated anchors, the region proposal network (RPN) outputs a set of region proposals that may contain the objects of interest. As the generated anchors may contain background as well, we need to identify whether an anchor contains an object or not. Also, the anchors may not fit the objects well due to the usage of a limited number of scales and aspect ratios. To solve this, RPN predicts two kinds of output for each anchor. One is an objectness score representing the probability of having an object inside. The other is the offsets required to turn the anchor into a region proposal that fits the target object better.

To obtain the required outputs, the Region proposal network incorporates both classification and regression branches

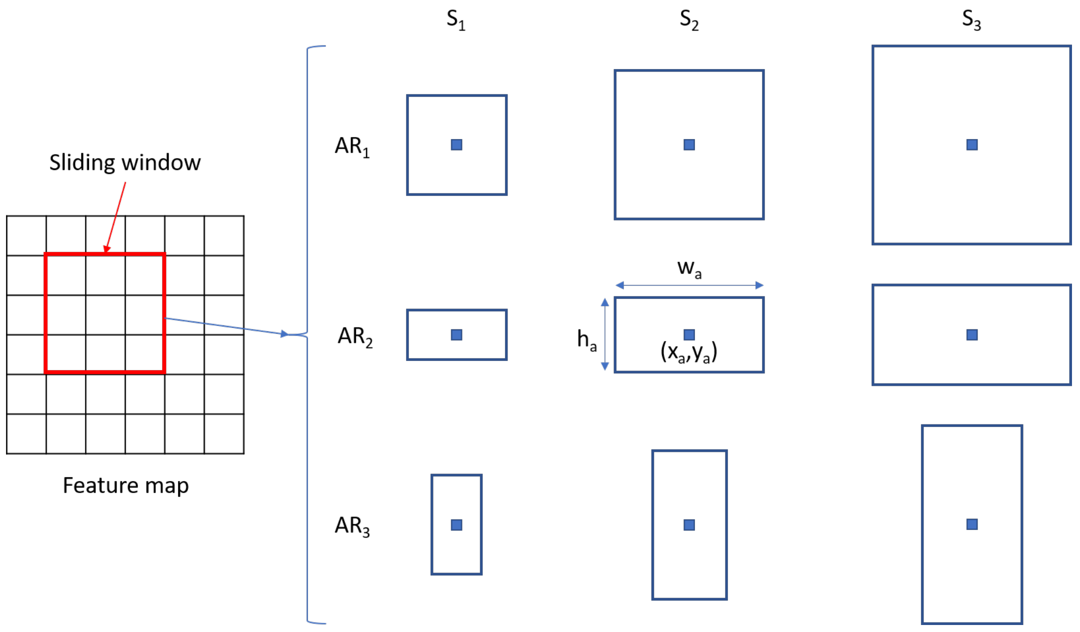


Figure 2.14: Anchor boxes generation. Here, 9 anchor boxes are generated for each sliding window on the feature map. The center, width and height of the anchor boxes are denoted as (x_a, y_a), w_a, and h_a respectively.

and the implementation is fully convolutional. First, it applies a convolution with 3×3 kernel and 512 number of channels. Then, there are two convolutional layers in parallel. Each of the layers uses a kernel of size 1×1 and the number of channels can be derived from the number of anchors (K) at each pixel in the feature map. In the classification layer, there are $2K$ predictions. For each of the anchors, there are a foreground score and a background score. On the other hand, for the regression, there are $4K$ outputs where each anchor gets four predicted offsets that can be denoted as $\Delta x, \Delta y, \Delta w, \Delta h$. Here, Δx and Δy represents the difference between the centers of the anchor and the target ground truth bounding box whereas Δw and Δh mean the difference in width and height respectively between the anchor and the target ground truth bounding box. These offsets can be applied to the width, height, and center of the anchor to get the actual coordinates. The architecture of the RPN is shown in Figure 2.15.

To train the designed RPN, the target class scores and regression offsets are calculated based on the ground truth bounding boxes. An anchor is considered to be a foreground if it has the highest IoU with a ground truth bounding box or the IoU is greater than 0.7. On the contrary, an anchor is labeled as background if the IoU is less than 0.3. Any other anchors are ignored during the training. For regression, transformation parameters (Δ values) are calculated for the foreground proposals using the closest ground truth bounding boxes.

Once the proposals are generated, the proposals are passed through a Non-maximum Suppression (NMS) step to remove the overlapping proposals. In this process, the proposals are sorted by objectness score and at each step, the proposal with the highest score is chosen. Then, any other proposals having an IoU greater than 0.5 with the selected proposal is discarded. These steps are repeated for all the proposals to obtain the final list of region proposals.

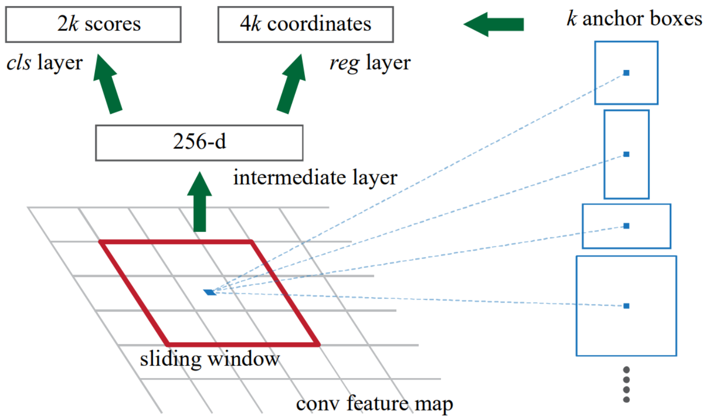


Figure 2.15: Region proposal network. The figure shows convolution for one location on the feature map. For each anchor box, the classification layer outputs two scores (foreground and background score) and the regression layer outputs four predicted offsets to convert the anchor box to a region proposal [82]. Copyright ©2017 IEEE.

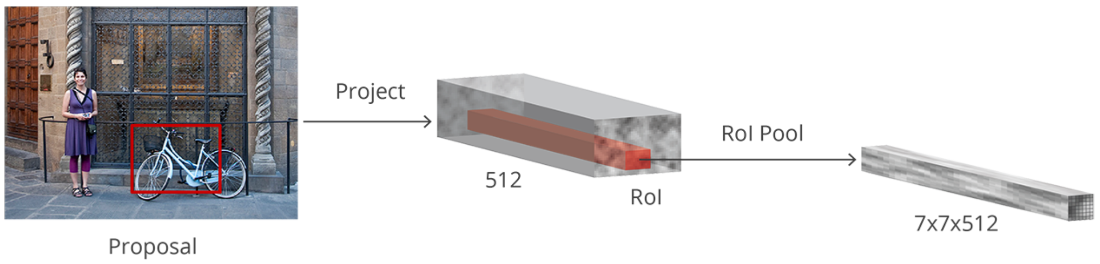


Figure 2.16: RoI pooling layer takes the region proposals and the extracted feature map as inputs, projects the region proposals to the feature map, and outputs a fixed size smaller feature map. [80].

RoI Pooling

The region proposals produced by the RPN can have various sizes. But, the fully connected layers ahead in the network architecture expect fixed-size input. To meet this expectation, the generated proposals are fed through this RoI (Region of Interest) pooling layer. This layer takes the RPN generated region proposals and the feature map created by the feature extraction network as inputs. Then, the proposed region coordinates are mapped to the feature map to get the corresponding relevant region in the map. Afterward, this feature map region is divided into a grid of size $M \times N$. Both M and N are layer parameters and don't depend on the inputs. Lastly, max-pooling is used on the pixels of each grid cell to obtain a fixed size smaller feature map as output. The number of channels remains the same as the input feature map. Figure 2.16 shows an example of the pooling operation.

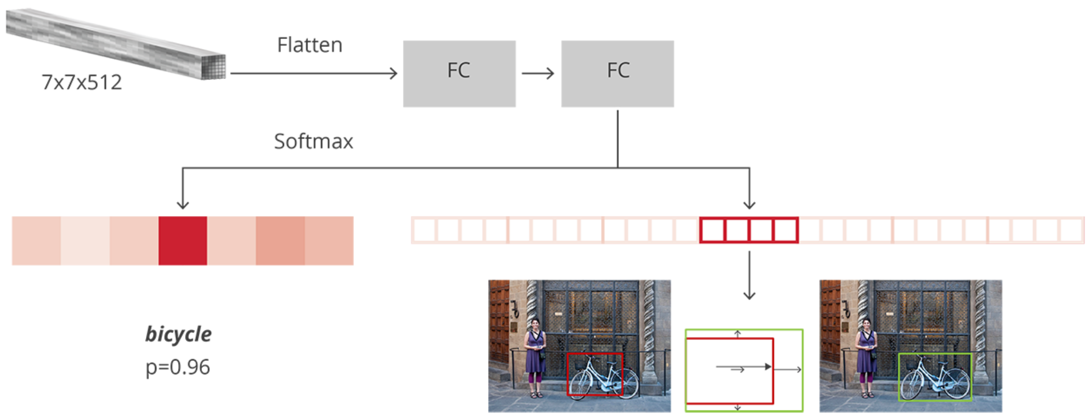


Figure 2.17: R-CNN architecture [80]. For each proposal, the fixed size feature map is flattened and passed through two fully connected layers that output the classification scores and position offsets to obtain the final predictions.

Region-based Convolutional Neural Network (R-CNN)

The region-based convolutional neural network marks the last component of the pipeline. It takes the fixed-size feature map for each proposal given by the RoI pooling layer and flattens it to obtain a one-dimensional feature vector. The final feature vector is used to classify the object inside the proposal and regress the width, height, and center offsets to transform the proposal to accurate predictions for each class. To achieve this, first, the feature vector is linked to two fully-connected layers. The Faster R-CNN implementation uses 4096 neurons in each of these layers along with ReLU activation. At the end of the network, there are two separate fully connected layers for classification and regression:

- A fully-connected layer for classification with N+1 outputs where N represents the number of classes and the background class is considered with the extra 1.

- A fully-connected layer for regression with $4N$ outputs. Here, the output comprises the four position offsets for each class.

The architecture of the R-CNN is illustrated in Figure 2.17 with the example of bicycle detection.

For the training of this network, the classification and regression targets are generated by comparing the region proposals with the available ground truth bounding boxes. Proposals that have IoU greater than 0.5 with any of the ground truth boxes are assigned to the respective class of the matching ground truth. If the IoU is between 0.1 and 0.5, the proposals are marked with background class. For all the proposals that have an assigned class other than background, regression offsets are calculated based on the matched ground-truth bounding box.

Like the RPN, the predicted bounding boxes may overlap. Therefore, after discarding the bounding boxes predicted as background, a class-specific NMS step is applied to the others to obtain the final list of bounding box predictions for an image.

2.4.2 RoI (Region of Interest) Transformer

RoI transformer [28] can be embedded in a two-stage detection framework such as Faster R-CNN to efficiently convert the horizontal region proposals (HRoI) to rotated region proposals (RRoI) and extract rotation-invariant features for accurate regression of rotated bounding boxes per objects. To enable predictions of rotated bounding boxes, the regression branch of the R-CNN is modified to output one additional offset ($\Delta\theta$) representing the difference in rotation. Besides, the height and the width are considered as the short and the long side of the bounding box respectively. The orientation is calculated between the x-axis and the width and falls in the range $[0, \pi]$.

RoI transformer is composed of a rotated RoI learner and a rotated RoI alignment module. Here, we describe how these two modules combine to build an object detector with rotated bounding boxes as predictions.

Rotated RoI Learner

In the case of RPN, horizontal anchors of different sizes are used to derive horizontal region proposals. Therefore, a trivial approach to obtain rotated region proposals would be the usage of rotated anchors as employed in [106, 112, 67]. But, the design of such anchors is limited by the number of directions used and fails to capture all the rotational variances for the objects in the images. This causes a misalignment between the extracted features and the actual objects in the scene. Furthermore, the higher number of directions leads to higher computational complexity.

To avoid a large amount of rotated anchors and alleviate the misalignment issues, the rotated RoI learner is proposed to give a better approximation of rotations. It takes the feature maps of horizontal region proposals as input and uses a fully connected layer to transform those to rotated proposals. For each HRoI, the module predicts five offsets denoted as $\Delta x, \Delta y, \Delta w, \Delta h, \Delta\theta$ that represents the differences of the center, width, height, and rotation between the HRoI and the target ground truth RRoI. The rotational difference ($\Delta\theta$) is in $[0, 2\pi]$. The target ground truth RRoI is found by matching the HRoI with the external rectangles of the ground truth RRoIs. Given the offsets and the input HRoI, the module uses a decoder to determine the parameters $(x_r, y_r, w_r, h_r, \theta_r)$ of the RRoI. The parameters can be used to project the RoI on to the feature maps and extract relevant features in the following layers of the network.

Rotated RoI Align

RoI align [44] solves the problem of quantization associated with RoI pooling. RoI transformer incorporates a rotated version of RoI align to be able to obtain rotation invariant deep features. Therefore, after feature extraction of each RRoI, the warped feature has a rectangular shape as shown in Figure 2.18.

2.5 Image Segmentation

Image segmentation is a technique to divide the image into multiple parts or regions named segments. Each segment represents a salient region (object or parts of the object) of the image and composed of a cluster of pixels. Image segmentation provides a pixel-level accurate understanding of the image unlike object detection with bounding boxes

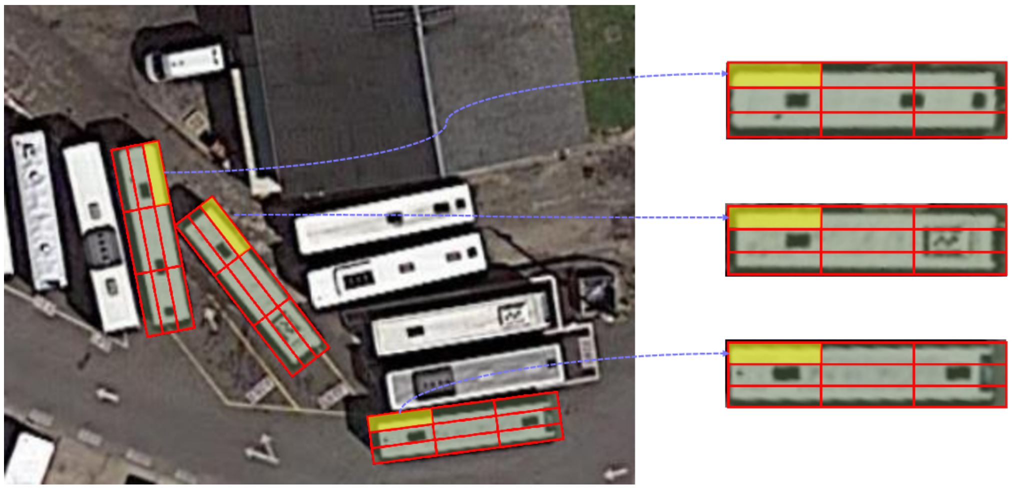
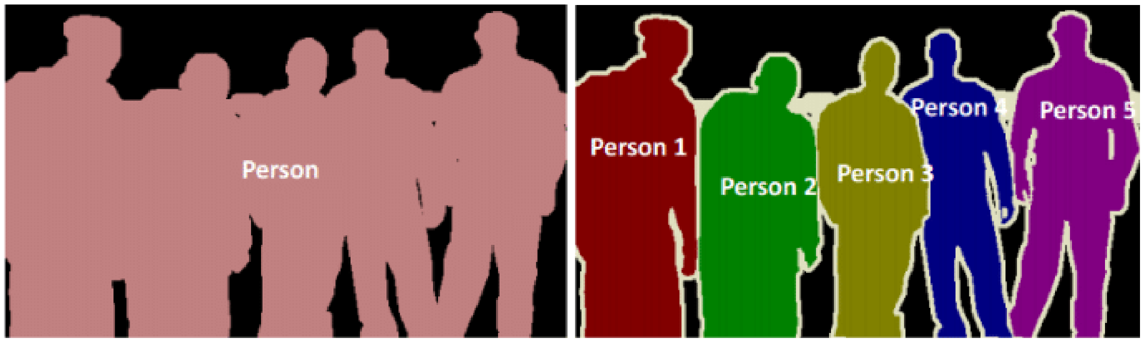


Figure 2.18: Rotated RoI align [28]. Copyright ©2019 IEEE.



(a) Semantic segmentation **(b)** Instance segmentation

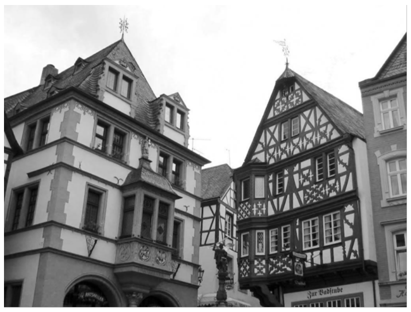
Figure 2.19: Different kinds of image segmentation.

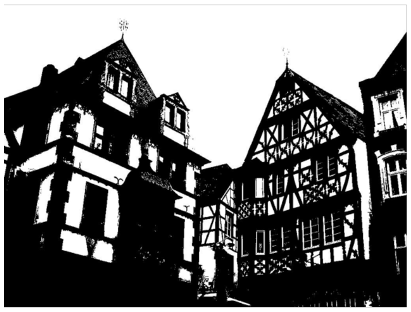
that may contain multiple unrelated pixels. It can be further subdivided into semantic segmentation and instance segmentation.

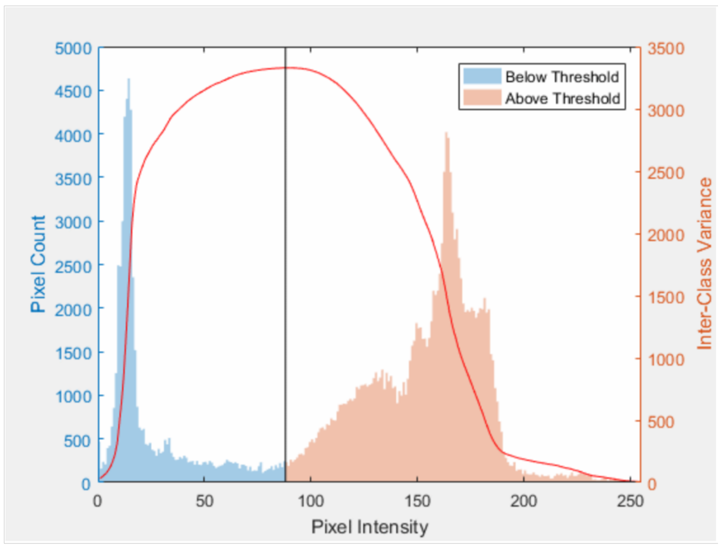
- Semantic segmentation: In semantic segmentation, pixels belonging to objects of the same classes are assigned with a particular label and labels are different across different classes. For example, in Figure 2.19a all the pixels representing persons have the same label.

- Instance segmentation: Instance segmentation gives a unique label to every object in the image. For example, each person has a different color label in Figure 2.19b.

In this work, we have used a traditional region-based segmentation method as well as a deep learning-based instance segmentation method named Mask R-CNN to process the regions of the objects in the images to help generate rotated bounding box proposals.


Original


Thresholded



Histogram

Figure 2.20: Image thresholding using Otsu's method [8].

2.5.1 Region-based Segmentation

The region-based segmentation methods group pixels with similar attributes to form unique regions. If there is a sharp contrast between the foreground and background of a grayscale image, pixel values will be different for them. Therefore, the intensity of the pixels can be used to classify the pixels as foreground or background using a threshold and create a binary image. In the binary image, any pixels having an intensity greater than the threshold are labeled as white (foreground) whereas pixels with lower intensity are labeled as black (background). This type of binarization is recognized as threshold segmentation.

If a single threshold is used globally, it's known as global thresholding. Global thresholding doesn't perform well if the input image has non-uniform illumination. To handle the issue another kind of thresholding technique named local thresholding can be employed. Local thresholding uses different thresholds for different parts of the image based on the local image characteristics.

Otsu's Method

Otsu's method is one of the most popular global thresholding algorithms to perform automatic image thresholding. The method works well if the histogram of the input image has two maxima. Given a grayscale image, the method iteratively tries to find the optimal threshold value by minimizing the within-class variance of the two groups of pixels separated by the candidate threshold values.

Let's consider a image with N pixels and the intensity values in the range $[0, I]$. The method first builds a histogram of the image and computes probabilities for each of the pixel intensities using,

$$P_i = \frac{n_i}{N} \tag{2.9}$$

where n_i represents the frequency of intensity i in the image.

In each iteration, the algorithm tries a threshold value t that divides the pixel intensities into two classes. The variable t can take any value in the range $[0 \dots I]$. Therefore, the pixel intensities of the first class are in $[0 \dots t]$ and the second class are in $[t + 1 \dots I]$, For each of the two classes, class probabilities can be calculated as,

$$w_1(t) = \sum_{i=0}^{t} P_i \tag{2.10}$$

$$w_2(t) = \sum_{i=t+1}^{I} P_i \tag{2.11}$$

Then, the mean and variance of the pixel intensities of each class is calculated to determine the within class variance. The mean can be obtained as,

$$\mu_1(t) = \sum_{i=0}^{t} \frac{iP_i}{w_1(t)} \tag{2.12}$$

$$\mu_2(t) = \sum_{i=t+1}^{I} \frac{iP_i}{w_2(t)} \tag{2.13}$$

and, the variance can be expressed as,

$$\sigma_1^2(t) = \sum_{i=0}^{t} [i - \mu_1(t)]^2 \frac{P_i}{w_1(t)} \tag{2.14}$$

$$\sigma_2^2(t) = \sum_{i=t+1}^{I} [i - \mu_2(t)]^2 \frac{P_i}{w_2(t)} \tag{2.15}$$

The equation to get within class variance $\sigma_w^2(t)$ can be written as,

$$\sigma_w^2(t) = w_1(t)\sigma_1^2(t) + w_2(t)\sigma_2^2(t) \tag{2.16}$$

After trying all possible threshold values, the value of t that minimizes the within-class variance is considered as the optimal value to threshold the image.

The total variance of the image pixel intensities is a sum of within-class variance and between-class variance. As the total variance can be calculated independent of the threshold, the threshold t that minimizes the within-class variance also maximizes the between-class variance. It can be shown that the between-class variance can be computed as,

$$\sigma_b^2(t) = w_1(t)w_2(t)[\mu_1(t) - \mu_2(t)]^2 \tag{2.17}$$

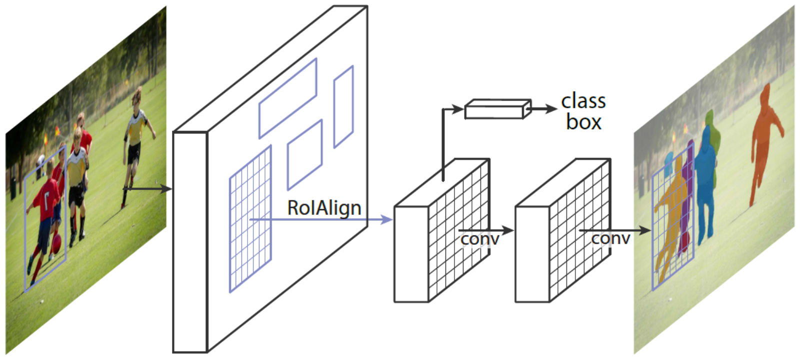


Figure 2.21: The framework of Mask R-CNN [44]. Copyright ©2019 IEEE.

Based on the equation above, the calculation of between-class variance eliminates the need to compute variances. Therefore, instead of minimizing the within class variance, we can maximize the between-class variance to find the optimal threshold value in a similar iterative manner much faster. An example of a thresholded image using between-class maximization is shown in Figure 2.20.

2.5.2 Mask R-CNN

Mask R-CNN is a deep learning-based instance segmentation algorithm. It adapts the Faster R-CNN framework to produce pixel-accurate binary segmentation masks. There are primarily two changes. One is the inclusion of a specialized branch for instance segmentation in the second stage of the framework. And, the other one is the refinement of the pooling layer using RoIAlign.

Mask R-CNN adds a separate branch in parallel to the classification and regression branch. The branch passes each RoI through two more convolutional layers to obtain one segmentation mask per class. The final output mask is determined based on the result of the classification branch. Figure 2.21 shows the architecture of the Mask R-CNN framework for instance segmentation.

RoIAlign significantly improves the mask predictions by eliminating the need for quantization. In the Faster R-CNN framework, quantization is applied during the calculation of cell coordinates by taking the integer parts when RoI is projected on to the feature maps as well as in the RoI pooling step. For example, if the proposed RoI has a size of 430×580 and the downsampling ratio is 16, then on the feature map the width is calculated as $[430/16] = [26.875] = 26$ and the height is calculated as $[580/16] = [36.25] = 36$. Similar rounding also exists in the pooling step for dividing the RoI region into $K \times K$ pooling windows. Therefore, these operations create misalignment issues between the RoI and the extracted features which impact the pixel-accurate mask predictions. To solve this, RoIAlign removes the rounding process and determines the values of the pooling windows using bi-linear interpolation [48] at a few sampled points.

3 Literature Review

In this chapter, we discuss the scope of object annotations with RBBs for object localization and the relevant object annotation tools. We also review different forms of weak supervision techniques such as box verification, point clicks, eye tracking, etc., and human-machine combined efforts to obtain object annotations with reduced time and labor. Lastly, we examine the impact of active learning in minimizing the annotation cost by prioritizing the most informative images.

3.1 Bounding Box Annotation Tools

For automated content analysis, fully supervised computer vision models rely on manual annotation of images. The success of the trained models depends on the accuracy of the gathered annotation. Image annotation acts as a way to direct the model to some relevant properties in the images we are interested in to solve a task. Each image can have one or multiple annotations and different forms of annotations are required to tackle different types of image analysis tasks. For example, an image recognition task needs image-level labels such as category labels or descriptive attribute labels. However, per-object annotations using AABBs, RBBs, or polygons are widely used for data labeling in object detection tasks. An AABB has a rectangular or square shape and fits the contained object as tightly as possible. An RBB additionally integrates the orientation of the fitted object. Therefore, RBBs have several advantages over standard AABBs. RBBs can capture the actual size and aspect ratio of objects of arbitrary orientations. They also provide a tighter fit and thus eliminate a lot of the background pixels within the annotations improving the feature extraction compared to AABBs. Moreover, RBBs enable the trained models to detect densely packed objects without overlapping boxes in the aerial images. Though RBBs provide more accurate representations of the objects, these don't outline the true shape of non-rectangular objects. On the other hand, polygon annotations can precisely define the shape and location of such objects and are supported in many of the annotation tools [33, 93, 99]. However, the process of getting polygon annotations is extremely time-consuming, expensive, and error-prone. According to Google AI Platform Data Labeling Service [3], polygon annotations are 4 times more expensive than AABB annotations whereas RBB annotations are only 1.37 times expensive. Therefore, even though polygon annotations perform better than bounding box annotations for detection tasks, bounding box annotations are primarily used for object detection to make it cost-effective [69]. Also, polygon annotations are primarily used for image segmentation [44] and most of the popular detection algorithms [82, 81, 63] are developed to work with AABB annotations. As a result, for object detection tasks, RBB annotations can provide the right balance among the annotation time, cost and model accuracy. But, the current system of RBB annotations is still complicated and more time-consuming than AABB annotations.

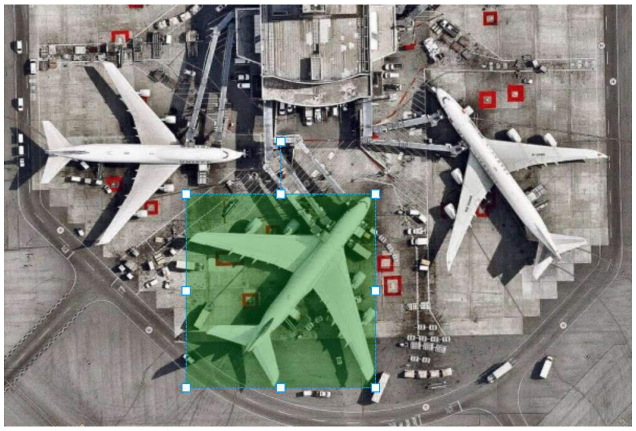
Other forms of annotation techniques also exist in the literature such as polyline annotation for lane detection in self-driving cars [2] and point clicks for object detection and segmentation [75, 76, 15]. Therefore, a lot of annotation tools have been developed over the years to facilitate manual annotation which allow different shapes such as box, polygon, circle, point, etc. to describe the object of interest. In this work, we are interested in annotation tools that allow drawing bounding boxes to localize objects in the images. Particularly, we are interested in RBB annotations and limit our comparison to open-source tools only.

LabelImg [6] enables annotators to create AABB annotations in the images. A modified version of LabelImg [7] also supports drawing of an RBB in three steps. First, an AABB is drawn which is followed by rotating the box in any direction using four keyboard keys presses to match the orientation of the encapsulated object. Finally, the width and height of the box are adjusted again to fit the object. The resulting RBB is expressed in the saved file with the center, width, height, and orientation of the box. This process is labor-intensive and time-consuming when thousand of object instances need to be annotated.

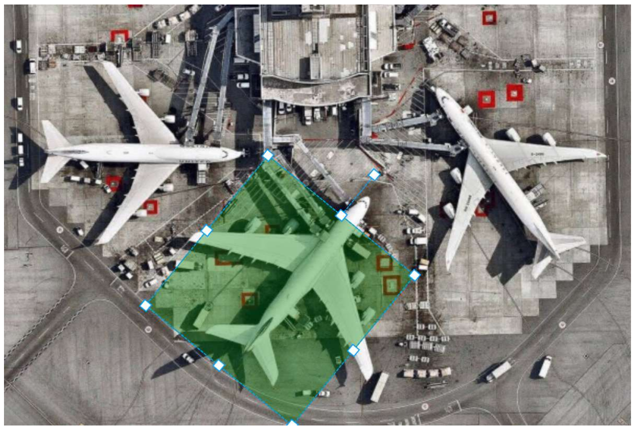
Fast Image Data Annotation Tool (FIAT) [5] allows drawing an RBB around the object in a similar way like modified LabelImg. After drawing the initial AABB, rotation can be applied to the box using multiple keyboard key presses. However, this tool also provides a way to draw an RBB by defining two diagonal points when the boxes have a known fixed aspect ratio. But, as in real-world datasets objects in a single image can have various aspect ratios, the annotation tool fails to deliver a fast labeling interface for rotated bounding box annotation.

Like the above-mentioned tools, Label Studio [93] also offers to create an RBB in multiple steps. But, instead of keyboard key presses, the tool enables the annotator to use mouse drag to apply rotation to the box. The steps to create the initial axis-aligned box and adjust it later are analogous to previously discussed tools. Therefore, the drawing of a rotated annotation is essentially a combination of three different tasks in this process and it may be possible to develop a more direct way for annotating with RBBs. Figure 3.1 shows the three required steps to create an RBB for a plane using Label Studio.

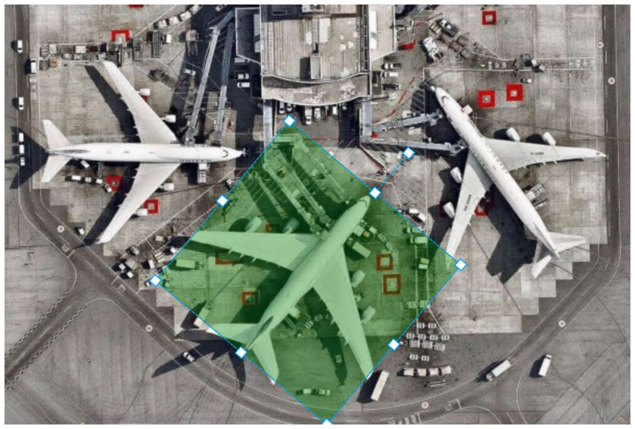
Some of the tools [9, 11] allow pre-trained machine learning models to automate the annotation process but are limited by the number of classes the pre-trained model supports. Also, the models are not iteratively trained on the latest annotated images to generate better proposals for the objects in the images over time. Anno-Mage [9] is a semi-automatic image annotation tool that integrates an one-stage object detector named RetinaNet [61] to produce AABB suggestions on a given image. The object detector is pre-trained on the MS COCO dataset that has objects of 80 classes. Therefore, if a new image to be labeled contains objects of classes other than those 80 or look visually very different than the trained images, the model's ability to provide suggestions degrades. This phenomenon inflates the manual annotation time in the process. CVAT [11] also allows the annotator to choose from a wide range of computer vision models based on the annotation task. The annotators can also upload their models. Again, these models are pre-trained on some other datasets and not capable of serving as a suggestive algorithm for a new dataset with new classes of objects.



(a) Draw AABB



(b) Rotate the box to match the orientation of the target object



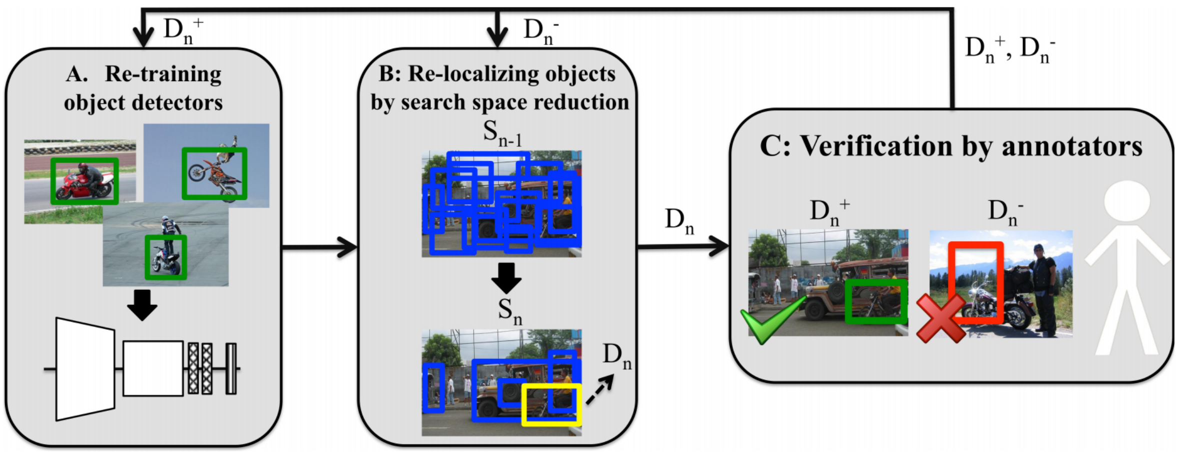
(c) Adjust the box to obtain the final RBB

Figure 3.1: The three steps to create an RBB using Label Studio. a) Draw AABB. b) Rotate it to match the orientation of the target object. c) Adjust the box to obtain the final RBB.

3.2 Weakly Supervised Object Localization

In weakly supervised object localization approaches, image-level labels that describe the classes of objects present in the image are used to train object detectors instead of actual bounding boxes. Though these techniques [17, 39, 27, 42, 51, 113] eliminate the need for bounding box annotations, the trained object detectors are significantly weak in terms of accuracy. Fully supervised object detectors with AABBs have twice the accuracy than the weak detectors.

Alternative forms of cheap human supervision have been explored in recent works. In the box-verification series, Papadopoulos et al. [74] used human verification signals in both retraining an object detector and re-localizing the objects in the images via proposals iteratively as illustrated in Figure 3.2. The authors utilized EdgeBoxes [115] to generate the initial set of proposals. Later, at each iteration, the search space was minimized by eliminating bad proposals with the help of human verification. The trained detector achieved better object localization compared to the previous WSOL approaches as only the correct bounding boxes were used in the training.



D_n^+
D_n^-
D_n^+, D_n^-
A. Re-training object detectors
B: Re-localizing objects by search space reduction
S_{n-1}
S_n
D_n
D_n
C: Verification by annotators
D_n^+
D_n^-

Figure 3.2: The proposed framework for training object detectors using human verification by iterating between A) re-training object detectors, B) re-localizing objects, and C) human box verification [74]. Copyright ©2016 IEEE.

Konyushkova et al. [55] presented an Intelligent Annotation Dialogues (IAD) agent that could determine the most time-efficient sequences of annotation actions to produce bounding box annotations for an image when image-level labels were available. It demonstrated that initially, the majority of the bounding boxes needed to be manually drawn. But, as the detector got stronger, box verification grew in number.

Papadopoulos et al. [73] proposed a novel approach to train object detectors with eye movement data instead of manual bounding-box annotations to reduce the time required to annotate images. In this method, annotators would find the target object in the image, look at it, and press a button. During this process, by tracking the eye movements, information about the eye fixations of the annotators is collected. The fixation data are then employed to automatically predict a bounding box for each image in two stages (Figure 3.3). First, a superpixel classifier trained with images having both fixations and manual bounding boxes is used to predict the initial segmentation of the object using fixation

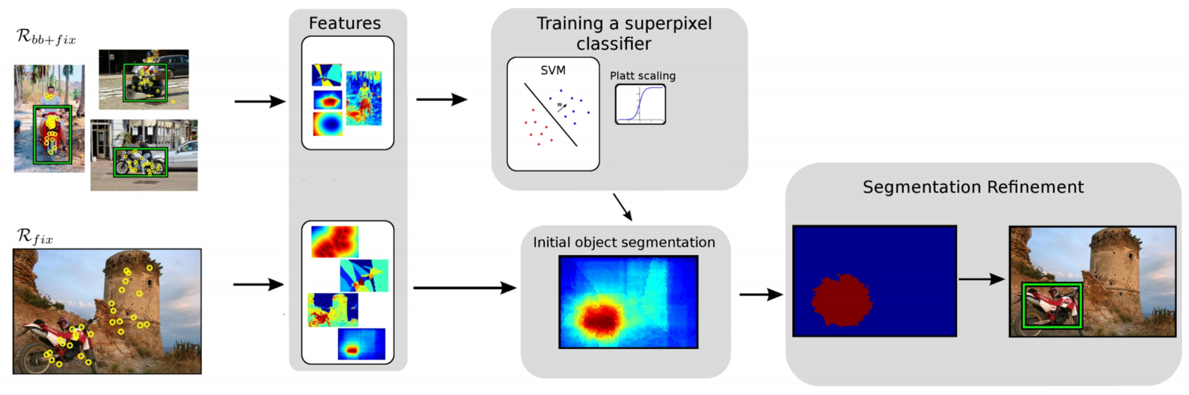


Figure 3.3: The proposed method for predicting bounding boxes from fixations. Reprinted by permission from Springer Nature: Springer Nature, Computer Vision – ECCV 2014, [73] (Training Object Class Detectors from Eye Tracking Data, Papadopoulos et al.), Copyright (2014).

data. Then, the bounding box of the object is found from the rectified initial segmentation.

Papadopoulos et al. [75] developed a scheme called extreme clicking where the annotators were requested to click on four well-defined physical points of an object namely the top, bottom, left-most and right-most points as shown in Figure 3.4. This approach was showed to be five times faster than traditional manual bounding box annotations. The collected four points per object could be used to obtain the axis-aligned bounding boxes, as well as better initializing the GrabCut algorithm to acquire accurate segmentations. The models trained on such annotations achieved comparable accuracy to the models trained with manual annotations.

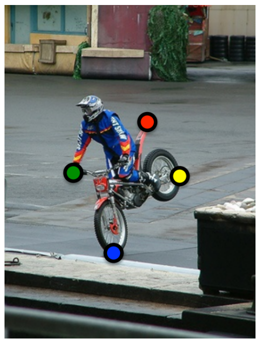


Figure 3.4: Annotating an instance of motorbike with extreme clicking scheme [75]. Copyright ©2016 IEEE.

Papadopoulos et al. [76] presented an annotation scheme employing click that minimizes human annotation effort and reduces total annotation time by 9 to 18 times compared to manually drawn tight object bounding boxes. The researchers asked annotators to click on the center of an imaginary bounding box enclosing an object, where the class of the object in an image is known. These clicks provided an approximation of the position of the center of the whole bounding box. The authors also obtained a more accurate approximation of the center by calculating the average of the click positions of two different annotators using the same object. Moreover, they trained a model that can estimate errors that annotators might introduce while detecting the center point of the object and used that to further refine the

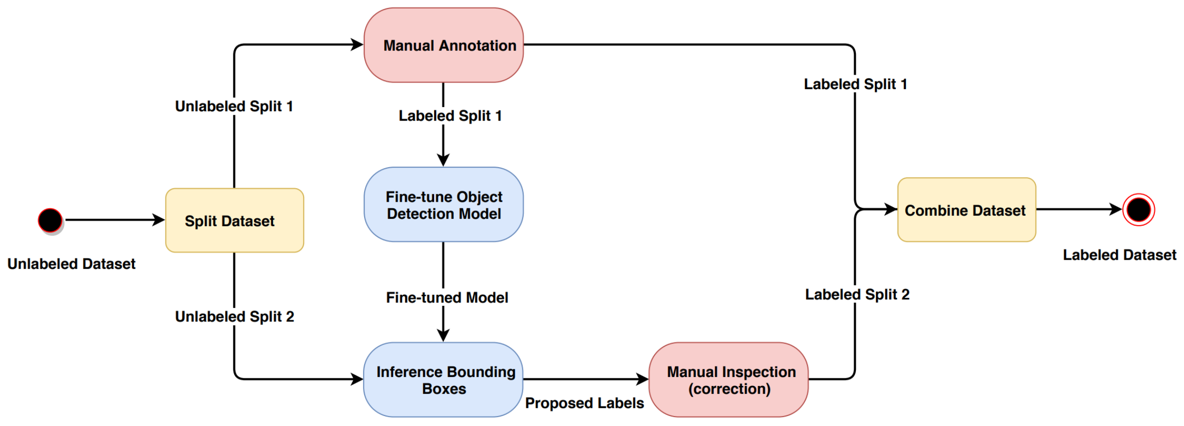


Figure 3.5: The proposed two stage workflow for image dataset annotation [13]. Copyright ©2018 IEEE.

labeling.

Bearman et al. [15] proposed a fast supervision method for image segmentation based on the points provided by the annotators to the objects in the images. Each point was captured at the center of an imaginary bounding box around the object. They demonstrated with experiments that the point-supervision accompanied by an objectness prior in the loss function improved the accuracy of the trained segmentation model than the other forms of image-level supervision. The accuracy improved, even more, when the training started with some images having pixel-level full supervision and followed by point-supervision for the other images. Alternatively, Wang et al. [100] used a single finger touch on each object instead of a point click. Then, the local features (e.g. edge, texture, etc.) to the touched point were leveraged to determine the segmentation.

3.3 Human-machine Collaborative Annotation

Computer vision models pre-trained on large scale benchmark datasets [34, 60, 16] are not strong enough to detect all the object instances in complex scenes. This becomes more evident when the intention is to build a new dataset that contains uncommon classes of objects. To solve the challenging task of getting high-quality object annotations efficiently, various works have combined the responses of computer vision models with human input.

Adhikari et al. [13] proposed a two-stage framework for fast acquisition of bounding box annotations for image datasets to be used in object detection. The researchers divided the dataset into two parts. Then, a Faster RCNN model (pre-trained on MS COCO dataset) was trained using the manually annotated first part of the dataset to generate AABB proposals on the second part. These proposals were further refined by a human annotator with actions like removing the incorrect boxes and drawing new ones. Both of the annotated parts of the dataset were combined to obtain the complete labeled dataset (Figure 3.5). Moreover, the authors devised a method to measure the workload incurred during the labeling process. This method was employed to evaluate the optimal partitioning strategy of the dataset to achieve the maximum workload reduction.

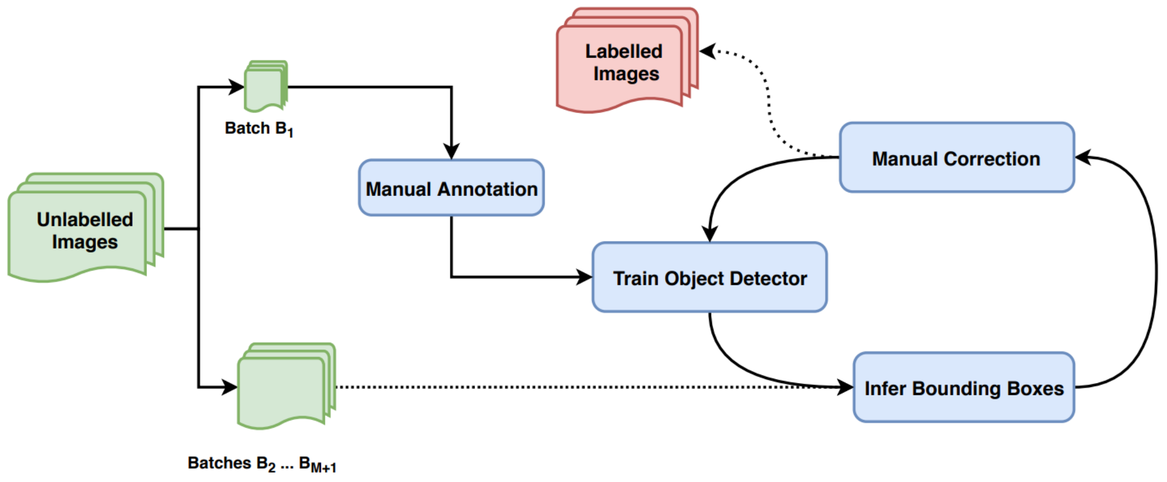


Figure 3.6: The workflow for Iterative bounding box annotation [12]. Copyright ©2021 IEEE.

Adhikari and Huttunen [12] built on the previous approach by iteratively training the model with small batches of labeled images and proposals were generated on the next batch in the line. These proposals assisted a human annotator to efficiently produce bounding box annotations for that next batch and those annotations were used to update the model for the following iteration (Figure 3.6). The focus of this work was not on developing an annotation tool but to compare different ordering of images to train a strong detector as early as possible. Getting a better model in early iterations helps to reduce the manual annotation effort as well as the total time needed. By evaluating on three datasets, the authors showed that the developed iterative approach reduced annotation workload by up to 75% compared to fully manual annotation. They also investigated the catastrophic forgetting [54] behavior of the model that could have a negative impact on the model during incremental learning.

Several other works used human-machine collaborative annotation effort to develop interactive object segmentation [20, 84, 21, 32, 49, 89], interactive video annotation [98, 53] and attribute-based classification [18, 77, 78] methods.

Dutt Jain and Grauman [32] explored different modes of annotation namely bounding box, sloppy contour, and tight polygon as outlined in Figure 3.7 and tried to predict the most suitable input modality for each image to obtain the best possible image segmentations. A bounding box is easy to draw but contains more background pixels. However, a sloppy contour has fewer background pixels but require higher attention and effort from the annotator. Tight polygon is the same as the segmentation and the most expensive one. Therefore, each input modalities involve different annotation time cost by the annotator. In this work, the authors optimized the total annotation time by only requesting an expensive annotation mode when the content of the image demanded it. Multiple classifiers were trained to predict the segmentation difficulty of the annotation modalities.

Boykov and Jolly [20] introduced a new method for generalized interactive segmentation of N-dimensional images by combining soft constraints with user-defined hard constraints. The user could mark a set of pixels of the image to definitely be in the foreground as well as a set of pixels to be in the background. These given pixel labels were considered as hard constraints whereas soft constraints were defined as the region and boundary properties of the

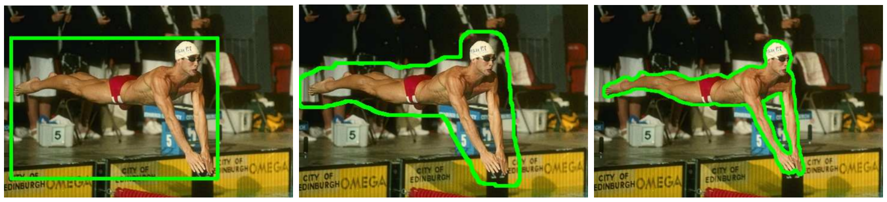

(a) Bounding box (b) Sloppy contour (c) Tight polygon

Figure 3.7: Possible modes of annotation [32]. Copyright ©2013 IEEE.

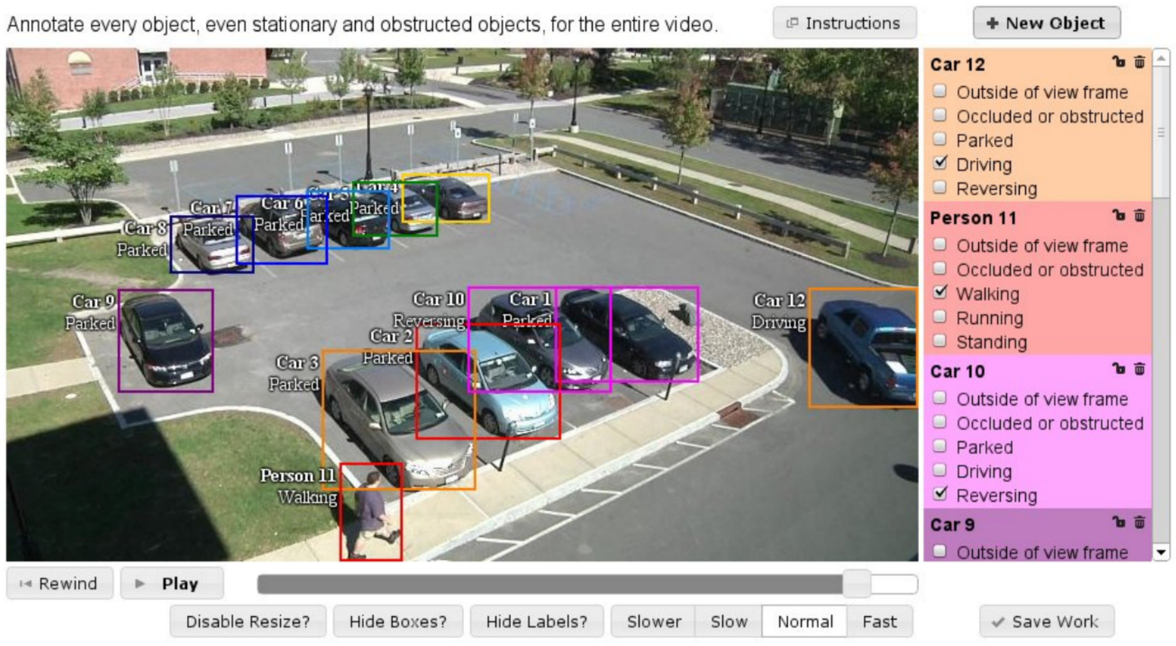


Figure 3.8: User interface for interactive video segmentation. Reprinted by permission from Springer Nature: Springer Nature, International Journal of Computer Vision, [98] (Efficiently Scaling up Crowdsourced Video Annotation, Carl Vondrick et al.), Copyright (2012).

segmentation. The method determined an image segmentation by minimizing a cost function. The authors claimed that the result obtained from this method provided a better balance of boundary and region properties among all segmentations and ensured that the constraints were maintained.

A video annotation platform with an efficient interface was developed by Vondrick et al. [98] that allowed the annotators to annotate every object of interest with high-quality labels in a complex video (Figure 3.8). Additionally, through their experiments, the authors demonstrated the limitations of the existing sub-optimal interfaces and established the need for a simplified and restricted interface.

Khodabandeh et al. [53] proposed a novel algorithm developed by incorporating unsupervised learning with user feedback in order to discover human interactions in video sequences. The algorithm was proven to be effective in obtaining accurate clustering results from unlabeled data and minimal user feedback.

3.4 Active Learning

Active learning based approaches try to train a model while reducing the human annotation effort by only selecting the most informative unlabeled images. The selection is done by the trained model itself. Therefore, active learning can be used to train an object detector with a limited annotation budget.

de Boer et al. [26] presented an annotation tool to train object detectors with active learning. Once a few images were annotated, an object detector was trained. The trained model was used to sort the remaining items based on confidence and the higher confidence detections were shown to the annotator first. This minimized the number of corrections needed from the annotators as in many cases box verification would be enough. The model was iteratively updated as new accurate annotation became available. The paper also experimented with other active learning techniques such as uncertainty or random sampling and concluded that the high-confidence technique worked well with fewer images.

Yao et al. [111] investigated the effectiveness of active learning compared to the incremental learning for interactively annotating object instances of a fixed-size dataset. Incremental learning iteratively trained a model to obtain suggestive annotation on new images that were corrected by the annotator. The annotation cost was modeled as the time taken to complete an image annotation. As incremental learning randomly traversed the images, the authors employed active learning to select the image with the highest predicted annotation cost at each iteration to request annotation. This reduced the total annotation cost as a stronger model was trained with diverse images in fewer iterations and provided better suggestions for the images ahead.

Vijayanarasimhan and Grauman [97] developed an end to end scalable approach that could gather images given a class of interest from the Web using keyword searches and employed active learning to identify the unlabeled data that would improve the model the most. The method acquired labels for the identified data using crowd-sourcing automatically. To use in a large scale setting, the authors proposed a part based detector for SVM classifiers and a hashing scheme. The trained detector needed only one-third of the training data to achieve state-of-the-art performance on six selected categories of the PASCAL VOC [34] dataset. The steps of learning live object detectors with this method are shown in Figure 3.9.

Many of the previous active learning works deal with image classification [50, 56] and region labeling [90, 95, 96]. Joshi et al. [50] proposed a generalization of the margin-based uncertainty measure for multiple classes in the task of image recognition. The uncertainty for each image was calculated by estimating class probabilities and it could be used to select the most informative images in the dataset to label to maximize the accuracy of the classification model with fewer training images. Therefore, it allowed building a multi-class classification model efficiently when a large number of unlabeled data was available.

Kovashka et al. [56] explored the advantages of shared descriptive attribute labels among objects of multiple classes and modeled the object-attribute, attribute-attribute relations. Then, the authors proposed an active learning based approach to query either the category labels or the attribute labels of images that would impact the process of multi-class object category learning the most. This strategy stimulated the training of the learner algorithm than the traditional active learning approaches.

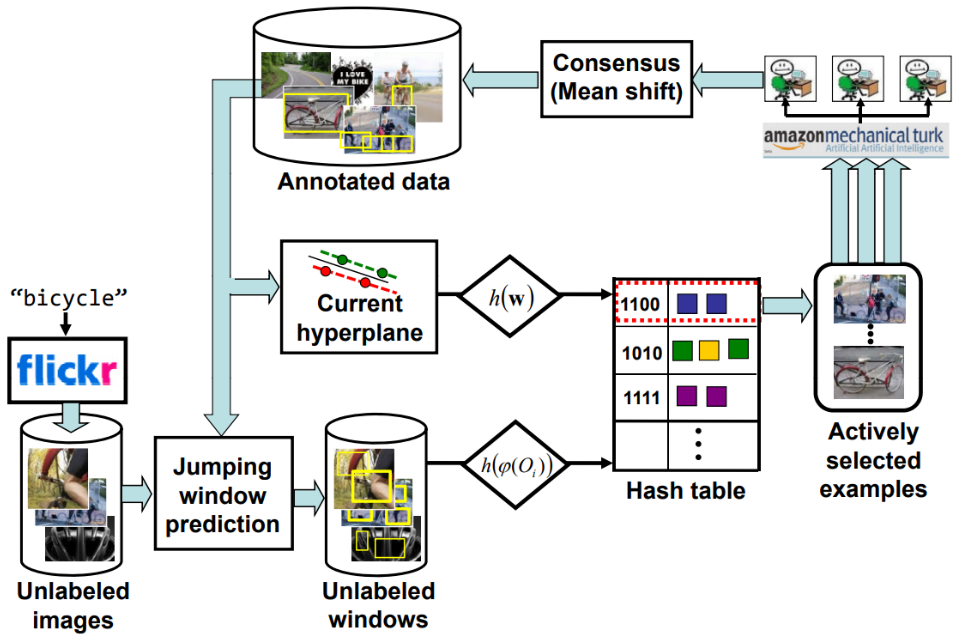


Figure 3.9: Learning object detectors in real-time with crowd-sourcing [97]. Copyright ©2011 IEEE.

Based on the content, some images are difficult to annotate than others. Vijayanarasimhan and Grauman [95] quantified the trade-off between selecting the most useful image for training and the annotation time it required and proposed an active learning strategy to strike a balance between uncertainty and relative annotation cost. It paved the way for learning visual categories with a model while minimizing the annotation effort.

Though active learning minimizes annotation time by reducing the number of images to annotate, the amount of time required to draw bounding boxes on the selected images is notably higher. Moreover, we are interested in the full annotation of all the images in this work.

3.5 Summary

Previous works that were concerned with faster bounding box annotations mainly focus on obtaining AABBs to localize object instances in the images. In contrast, we aim to produce RBB annotations for better localization. In our work, we use computer vision techniques to generate RBB proposals which are then verified by visual inspections or corrected by manual drawing. When the manual drawing is required, we observe that drawing an RBB is not as straight forward as opposed to an AABB. Hence, we design a multipoint drawing system to easily create an RBB from scratch. This minimizes the annotation time substantially in combination with RBB proposals. Our tool is also able to assist in generating high-quality RBB annotations given the respective AABBs.

4 RBB Annotation Methods

The goal of this thesis is to develop methods for annotating image datasets with accurate RBBs that will reduce human labour and annotation time. To achieve this target, we employ several strategies to cover two distinct aspects of RBB annotation; we try to minimize the need for manual RBB annotations while accelerating the process of manual RBB annotations when required. The proposed RBB annotation methods are integrated to an existing open-source annotation tool named LabelImg [6, 7] and the new tool is called FastRoLabelImg.

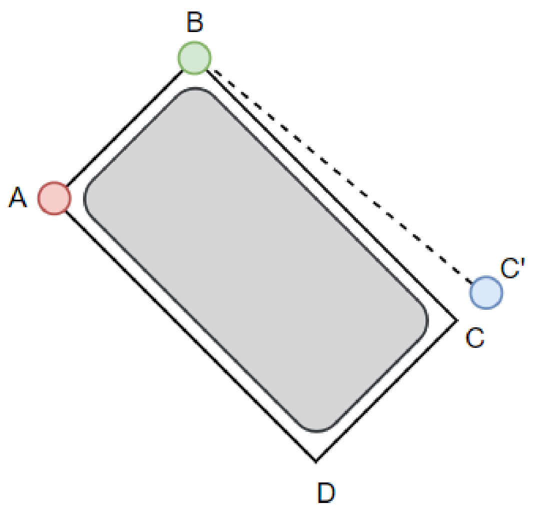
We design a multipoint box drawing system that enables the annotators to draw RBBs efficiently when manual annotation is required. The drawing system has multiple configurations and the annotators are free to choose the suitable configuration based on the appearance of the objects to be annotated.

Furthermore, the number of required manual RBB annotations is reduced by generating high-quality RBB proposals for the objects in the images. These proposals can be visually inspected by an annotator to determine their correctness. Fewer manual RBBs are needed with a higher number of correct proposals. We leverage computer vision models and traditional image processing techniques to produce these RBB proposals either from scratch or by using existing AABBs, if available. The annotators are given the options in the interface to utilize the most appropriate method for the dataset at hand.

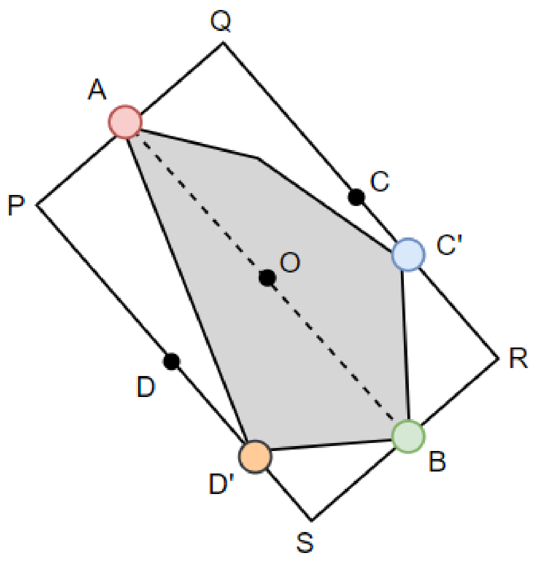
In this chapter, the developed methods for both reducing and accelerating the manual RBB annotation are described in detail.

4.1 Manual RBB Annotation

The purpose of the multipoint drawing system is to facilitate the process of manual RBB creation. Traditionally, 2 points are specified to annotate with AABBs. In that case, an annotator usually specifies the top-left and bottom-right corners of an AABB to create it. With the existing system of LabelImg, once the top-left point is clicked, the annotator needs to hold down the mouse button and drag it to the bottom-right point before leaving. We provide a simplification of this process that doesn't require the annotator to hold down the mouse button and two direct clicks are enough to create an AABB. However, we mainly focus on RBB creation and propose two new configurations: 3-point and 4-point. These configurations allow the annotators to specify a pre-defined set of points in some recommended orders to create RBBs as quickly as possible with fewer adjustments.



(a) 3-point



(b) 4-point

Figure 4.1: Multipoint box drawing system for objects of interest (grey shapes). The colored dots are clicked around the objects in the order: Red⇒ Green⇒Blue⇒Orange.

4.1.1 3-point System

In this approach, the annotator clicks on three corner points around an object instance to create a bounding box of the desired shape and orientation. The 3-point system is intended for creating rectangular bounding boxes of any orientations. Following the Figure 4.1a, if the drawn points are $A(x_1, y_1)$, $B(x_2, y_2)$, and $C'(x_3^{'}, y_3^{'})$ in this order, then first we calculate the perpendicular distance d from C' to the line passing through A and B using,

$$d = \frac{|(x_2 - x_1)(y_1 - y_3^{'}) - (x_1 - x_3^{'})(y_2 - y_1)|}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}} \tag{4.1}$$

Afterward, we determine the coordinates of point $C(x_3, y_3)$ at a perpendicular distance d from B such that the point C lies on the same side of AB as C'. To do this, we calculate the slope (m_{AB}) of the line AB as,

$$m_{AB} = \frac{y_1 - y_2}{x_1 - x_2} \tag{4.2}$$

Then, the slope of the perpendicular line BC will be,

$$m_{BC} = -\frac{1}{m_{AB}} \tag{4.3}$$

We add a small value of 10^{-6} to the denominator to avoid division by 0 where applicable (e.g. Equation 4.2). Using the slopes, the coordinates of point $C(x_3, y_3)$ are given by,

$$x_3 = x_2 \pm \frac{d}{\sqrt{1 + m_{BC}^2}} \tag{4.4}$$

$$y_3 = y_2 \pm m_{BC}\frac{d}{\sqrt{1 + m_{BC}^2}} \tag{4.5}$$

As there are two possible coordinates for C on either side of AB, we choose the coordinate that is closer to C'. We take these additional steps to make sure that the line BC is perpendicular to AB through the point B.

Finally, given the corner points $A(x_1, y_1), B(x_2, y_2)$ and $C(x_3, y_3)$ of the box, we can determine the other corner point $D(x_4, y_4)$ as,

$$x_4 = x_1 + x_3 - x_2 \tag{4.6}$$

$$y_4 = y_1 + y_3 - y_2 \tag{4.7}$$

4.1.2 4-point System

For non-rectangular objects, sometimes it is difficult to envision the corners of the RBBs that fit the object instances. Therefore, the resulting RBBs can be inaccurate and may need multiple attempts or adjustments to get the accurate RBBs making the process of RBB annotation laborious again. To handle ellipsoid objects without well-defined "corners", we provide a second configuration for the RBB drawing system that can make use of four drawn points. These points identify the opposite endpoints of the objects and their thickness instead of the corner points of the intended bounding boxes.

In the 4-point drawing system (Figure 4.1b), if the points drawn are $A(x_a, y_a), B(x_b, y_b), C'(x_{c'}, y_{c'})$, and $D'(x_{d'}, y_{d'})$, we find the center $O(x_o, y_o)$ of the line AB using,

$$x_o = \frac{x_a + x_b}{2} \tag{4.8}$$

$$y_o = \frac{y_a + y_b}{2} \tag{4.9}$$

Then, similar to the 3 points drawing system, we consider points A, O, and C' to calculate the position of the point C using the following equations:

$$x_c = x_o \pm \frac{d_{C'}}{\sqrt{1 + m_{OC}^2}} \tag{4.10}$$

$$y_c = y_o \pm m_{OC} \frac{d_{C'}}{\sqrt{1 + m_{OC}^2}} \tag{4.11}$$

where $d_{C'}$ is the perpendicular distance of C' to the line AB and m_{OC} is the slope of the line OC. Therefore, the coordinates of the point $Q(x_q, y_q)$ can be determined as,

$$x_q = x_a + x_c - x_o \tag{4.12}$$

$$y_q = y_a + y_c - y_o \tag{4.13}$$

Similarly, the coordinates of the point $R(x_r, y_r)$ can be calculated using,

$$x_r = x_b + x_c - x_o \tag{4.14}$$

$$y_r = y_b + y_c - y_o \tag{4.15}$$

Consecutively, we calculate the coordinates of P and S using the points $A, B, O,$ and D'. Thus, we have all the coordinates of the RBB $PQRS$.

4.2 Automatic RBB Proposals

When no pre-existing AABBs are available for an image, we use object detection models capable of predicting RBBs to generate proposals for the specified classes of objects in the image. We integrate two object detection models in FastRoLabelImg.

4.2.1 Pre-trained Detector

The pre-trained detector is based on the Faster R-CNN [82] framework embedded with ROI Transformer [28] to be able to predict the RBBs. The predicted bounding boxes can be denoted as $\{(x_i, y_i), i = 1,2,3,4\}$, where (x_i, y_i) denotes the positions of the box corners in the image. The pre-trained detector is trained on the DOTA [102] dataset which consists of 2,806 aerial images and the contained objects belong to 15 categories including plane, ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, roundabout, soccer ball field and basketball court. Therefore, this detector can support up to 15 categories when used to generate proposals. The predictions can also be filtered by confidence score and non-maximum suppression (NMS) before finalizing the proposals. The annotator can pre-configure the confidence score and overlap threshold for NMS through the interface of the tool.

4.2.2 Iterative Detector

We also use an object detector (Faster R-CNN with ROI Transformer) that is updated iteratively on previously annotated data using vertical re-training. The approach is inspired by batch-mode active learning [87]. In FastRoLabelImg, we use a batch size of 1. Therefore, when a particular image is completely annotated by the annotator and annotations are saved, the detector is re-trained immediately with the recently available RBBs. At each time, when the proposal generation step is requested, the most updated detector is used to generate the proposals for the selected categories. Similar to the pre-trained detector, the predictions can be filtered by a confidence score and NMS. The iterative detector can support up to 1,000 categories of objects. As the labels of the categories are not known beforehand, we use 1,000 placeholder labels to support the number of categories mentioned above when the model is configured initially. Later, when a bounding box appears with a new label, a mapping is created between the original label and a placeholder label

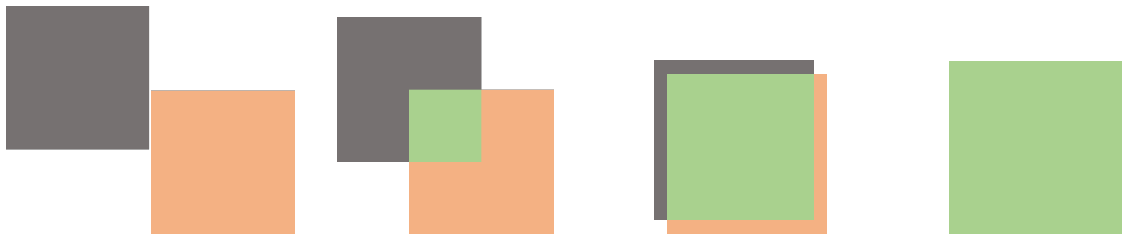

(a) No match (IoU=0). **(b)** Poor match (IoU=0.14). **(c)** Good match (IoU=0.7). **(d)** Perfect match (IoU=1)

Figure 4.2: Different overlapping scenarios between two AABBs using IoU=0.5 as threshold.

to continue the training. Similarly, this mapping is also used to retrieve the predictions of a specified category. The mapping is saved as a JSON file to maintain its state among multiple annotation sessions.

4.3 Using Existing AABB Annotations

When AABBs have already been annotated on a dataset, we leverage this information to improve our initial RBB proposals produced by the object detection models. This approach is discussed as *filtering* RBB proposals with the help of pre-existing AABBs. Moreover, we also develop segmentation-based approaches as an alternative to using object detectors. In this process, the AABB bounded regions are used to apply segmentation-based techniques (Thresholding and Mask R-CNN) to identify the pixels of interest and produce the RBB proposals with additional post-processing. These segmentation-based RBB proposal generation methods are discussed as *adapting* AABBs to RBB proposals in this thesis.

4.3.1 Filtering Proposals with IoU

If RBB proposals are requested for the available AABBs in an image using the object detection models, then we obtain the predictions on the image for the specified categories of objects as the first step. Later, the predicted proposals are matched with the given AABBs based on their overlap. The overlap is measured using the metric Intersection-over-Union (IoU) by considering both of the boxes as polygons. The Intersection-over-Union between two boxes is computed as,

$$Intersection\text{-}over\text{-}Union\ (IoU) = \frac{Area\ of\ Overlap}{Area\ of\ Union} \tag{4.16}$$

The IoU is equal to 1 in the case of a perfect match and the matching worsens as it goes lower. When there is no overlap between two boxes the IoU is zero. Figure 4.2 shows different overlapping scenarios between two AABBs.

The matching process between the proposed RBBs and the existing AABBs filters out many of the false-positive RBB proposals. The IoU threshold for box matching can be tuned by the annotator using the tool to control the quality of the RBB proposals. However, we found that an IoU threshold of 0.5 works well in practice.

4.3.2 Adapting AABBs with Thresholding

We make use of both traditional computer vision techniques as well as a deep learning-based instance segmentation model to serve the purpose of proposal generation from AABBs. Though a deep learning-based model can detect a large number of object classes, the detection is limited by the number of classes the model is pre-trained on. To overcome this issue, a set of assumptions is made about the objects inside the bounding boxes to develop an image processing based approach in support of the segmentation model. Besides, the image processing based approach might construct a better bounding box proposal than the model if the object of interest satisfies the assumptions.

In order to adapt an AABB to an RBB, we assume that the AABBs fit the objects as tightly as possible and the centers of the bounding boxes are on the encapsulated objects. Also, we presume that the histograms of the pixel intensities in the grayscale patches formed by the AABBs will be bimodal.

We define p as a patch image bounded by an AABB b and i as an object instance of class c which is annotated by b. We convert the patch p to a grayscale image and determine a threshold value t using Otsu's method [71] for the patch. We also calculate a mean pixel value m for the middle 10% region of the grayscale patch p. Then, m and t are used to determine the foreground and background pixels of the patch. If m is greater than t, then all the pixels having an intensity value greater than t are marked as foreground (in white color) because m is representative of the color of the foreground object based on the assumption that the centers of the bounding boxes are on the encapsulated objects. Alternatively, if m is less than t, then all the pixels having an intensity value greater than t are marked as background (in black color).

We perform a series of post-processing steps to obtain the final RBB proposal after getting the region of interest. Firstly, we might get one or multiple regions of interest. But, we know that a bounding box should contain one object at most. Therefore, if multiple regions of interest are detected in a patch p, we try to find the best suitable region by weighting each of the regional areas with the distance between the center of the region and the center of the patch. Formally, if there are n regions, then we calculate the weighted area for the i^{th} region as $w_i = area(i)/distance(c_i, c_p)$ where $i = [1 \ldots n]$, c_i is the center of the i^{th} region and c_p is the center of the patch p. Afterward, the region with the maximum weighted area is chosen to apply the post-processing steps.

Subsequently, we find an ellipse that has the same second-order central moments as the selected region of interest [83]. The second-order central moments are defined as the weighted averages of the pixel intensities of the region based on the center of the area. The length of the minor axis of the ellipse is used as the height of the RBB. Furthermore, we extend the major axis on both sides and obtain the two intersecting points with the AABB. The distance between these points is used as the width of the RBB. As for the rotation of the box, the angle between the major axis of the ellipse and the positive x-axis is used. The range of the angle is between 0 to 180 degrees in the pixel coordinate system (positive rotation is clockwise from the x-axis). Lastly, the center of the AABB is considered the center of the RBB.

Finally, we can compute the coordinates of the four corners of the RBB in the image given the width, height, center, and rotation of the box. We proceed by creating an AABB with the determined center, width, and height (width is along the x-axis). Then, we apply a rotation matrix to each of the corner points using matrix multiplication to obtain the rotated corner points $\{(x_i^{'}, y_i^{'}), i = 1, 2, 3, 4\}$. For example, if (x, y) is a corner point that is rotated clockwise by an angle θ, we can get the rotated points $(x^{'}, y^{'})$ using the equation:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{4.17}$$

4.3.3 Adapting AABBs with Mask R-CNN

We employ the popular Mask R-CNN [44] model pre-trained on the MS COCO 2017 [60] dataset to obtain the regions of the objects inside the pre-existing AABBs. Fundamentally, this approach involves two steps: 1) filtering the predictions of the Mask R-CNN model with pre-existing AABBs, and 2) adapting the AABBs using the related segmentation masks to RBB proposals. As the second step is primarily responsible for generating proposals from predicted segmentations, we consider the overall process as the adaption of AABBs. For example, let's assume that we need to determine an RBB proposal for an object of class c, and the object is contained in an AABB b. Then, the model is used to run prediction on the whole image and we obtain segmentation masks as well as respective bounding boxes for objects of class c. Later, we find the predicted AABB that has the maximum overlap with the given bounding box b and utilize the associated segmentation mask to locate each pixel belonging to the encapsulated object in the patch image. Once the region of the object is found, the post-processing steps remain the same as before to obtain the RBB proposal.

4.4 Summary

In this chapter, we explained the process of creating RBBs for each of the available configurations of the multipoint drawing system. The procedures for creating RBB proposals using object detectors are also presented. Furthermore, we discussed the post-processing steps of obtaining proposals using object segmentations extracted from the existing AABBs by the thresholding technique or the Mask R-CNN model.

5 FastRoLabelImg Interface

In this chapter, we discuss the core interactive components of our proposed RBB annotation tool named as FastRoLabelImg. It extends the open-source LabelImg tool and we provide explicit buttons and widgets as well as keyboard shortcut keys to trigger the desired actions.

5.1 Layout and Saved File Format

Figure 5.1 shows the complete interface of the tool. The interface comprises several components. To the left side of the interface, there are multiple quick access buttons. These buttons are organized into three groups from top to bottom. The first group of buttons enables the annotator to open images, save annotations and traverse the list of the images. The second and third groups of buttons provide access to the multipoint box drawing system and proposal generation methods respectively. There are four more panels in the user interface. One of those panels shows a preview of the image to be annotated and allows the annotator to interactively annotate the image. The other panels are dedicated to present the list of image files, box labels of the current image, and proposal settings respectively.

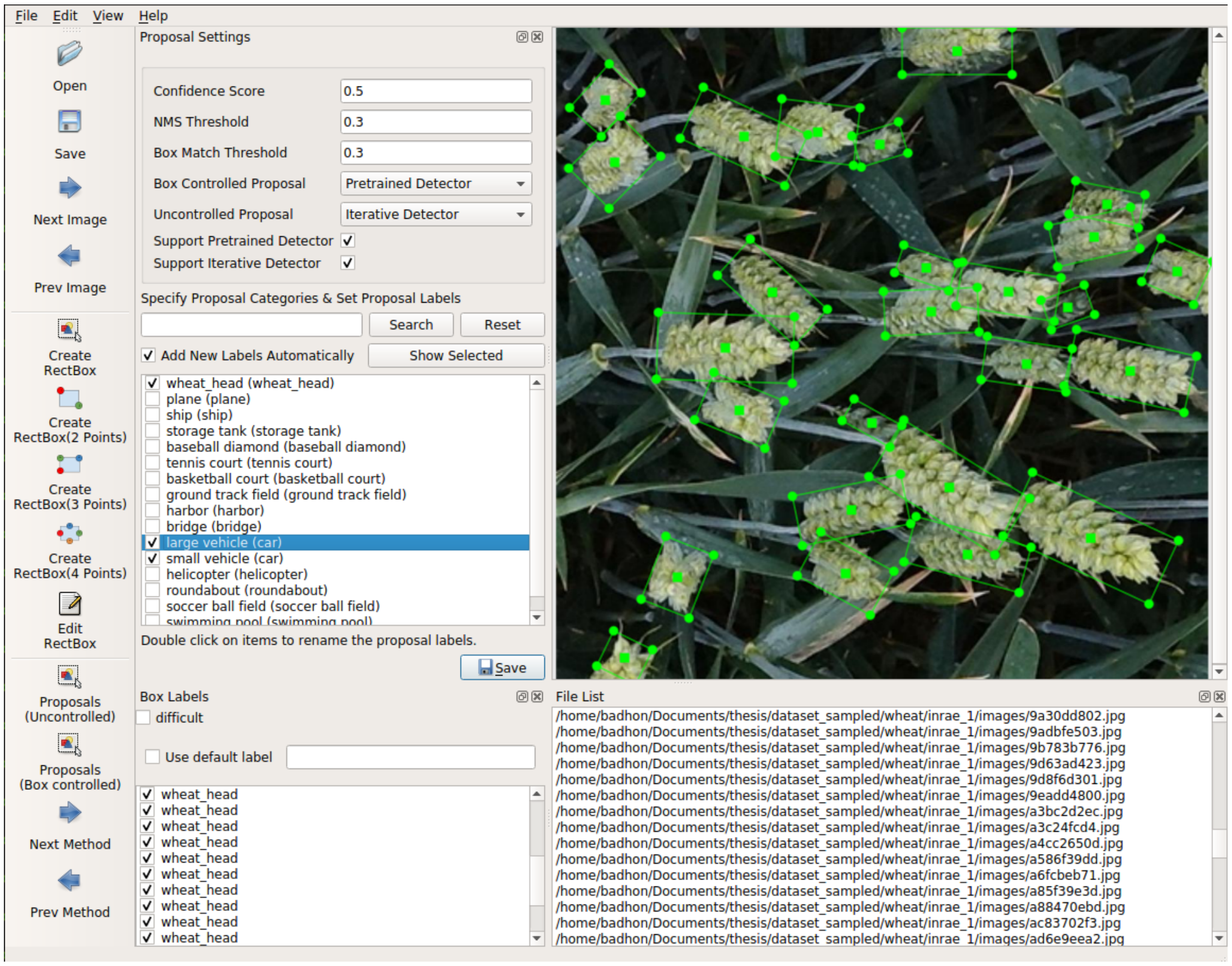
The annotations are saved as XML files in a format similar to PASCAL VOC [34] annotations. To encode RBBs, we save the width, height, center, and rotation of the boxes instead of top-left and bottom-right coordinates as is done for AABBs. The width is considered as the larger side of the bounding box. The rotation is calculated as the angle between the width and the x-axis clockwise in the pixel coordinate system and falls in the range $[0, \pi]$.

In the following sub-sections, we describe the individual control options regarding the proposal generation and the manual RBB box drawing system.

5.2 Proposal Generation

FastRoLabelImg presents a concise user interface to use the RBB proposal generation methods. Before applying one of the methods, the annotator is able to modify the RBB proposal generation through a settings window. Then, the explicit UI buttons or the keyboard keys can be used to try out the integrated proposal generation approaches and the most suitable one can be chosen to carry out the annotation process.

The proposal settings window as illustrated in Figure 5.2a enables the annotator to pre-configure some of the options regarding the computer vision models. These options include the confidence score used by the models, the accepted overlapping threshold for the non-maximum suppression technique, and the IoU threshold to match the proposals with the given AABBs. The annotator can also choose the default methods to be used when the buttons related to proposal



Confidence Score 0.5
NMS Threshold 0.3
Box Match Threshold 0.3
Box Controlled Proposal: Pretrained Detector
Uncontrolled Proposal: Iterative Detector
Support Pretrained Detector ✓
Support Iterative Detector ✓

✓ Add New Labels Automatically

✓ wheat_head (wheat_head)
plane (plane)
ship (ship)
storage tank (storage tank)
baseball diamond (baseball diamond)
tennis court (tennis court)
basketball court (basketball court)
ground track field (ground track field)
harbor (harbor)
bridge (bridge)
✓ large vehicle (car)
✓ small vehicle (car)
helicopter (helicopter)
roundabout (roundabout)
soccer ball field (soccer ball field)
swimming pool (swimming pool)

Double click on items to rename the proposal labels.

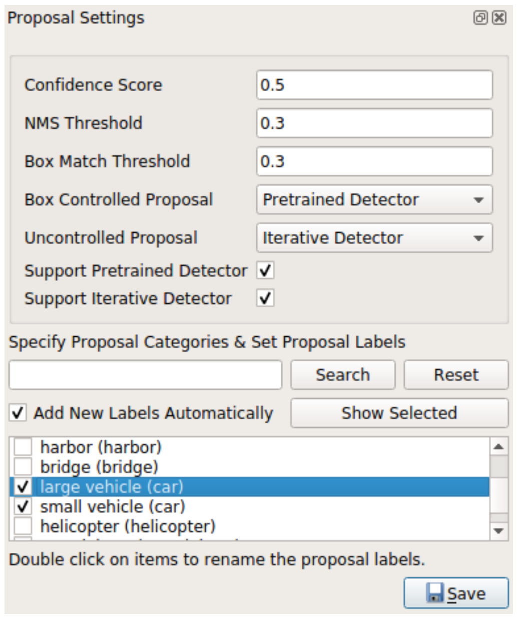
difficult

Use default label

✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head
✓ wheat_head

/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/9a30dd802.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/9adbfe503.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/9b783b776.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/9d63ad423.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/9d8f6d301.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/9eadd4800.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a3bc2d2ec.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a3c24fcd4.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a4cc2650d.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a586f39dd.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a6fcbeb71.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a85f39e3d.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/a88470ebd.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/ac83702f3.jpg
/home/badhon/Documents/thesis/dataset_sampled/wheat/inrae_1/images/ad6e9eea2.jpg

Figure 5.1: Interface of FastRoLabelImg.


Proposal Settings

Confidence Score: 0.5
NMS Threshold: 0.3
Box Match Threshold: 0.3
Box Controlled Proposal: Pretrained Detector
Uncontrolled Proposal: Iterative Detector
Support Pretrained Detector: ✓
Support Iterative Detector: ✓

Specify Proposal Categories & Set Proposal Labels

Search | Reset

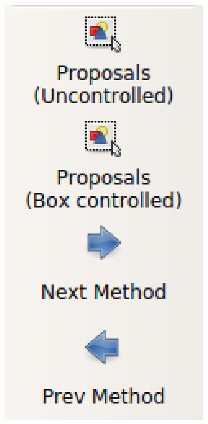
✓ Add New Labels Automatically | Show Selected

- ☐ harbor (harbor)
- ☐ bridge (bridge)
- ✓ large vehicle (car)
- ✓ small vehicle (car)
- ☐ helicopter (helicopter)

Double click on items to rename the proposal labels.

Save

(a) Proposal settings



(b) Proposal generation (UI buttons)

Figure 5.2: UI components related to proposal generation.

generation are clicked. For proposals given the AABBs, the annotator can choose among one of the four methods. These methods are pre-trained detector, iterative detector, Mask R-CNN, and thresholding approach. In contrast, either the pre-trained detector or the iterative detector can be selected as the default method for uncontrolled proposal generation. Support of the thresholding technique can be toggled as well to handle cases where no predictions are found from the models for the given boxes. Finally, the annotator can use the interface to specify the class labels for which predictions will be extracted from the computer vision models. The list shows the available classes of the MS COCO and DOTA datasets. Additionally, any new classes used to re-train the iterative detector can be appended to the list. The annotator can also rename the proposal labels for each of the class labels. For example, the pre-trained detector has two separate class labels to detect vehicles namely "small vehicle" and "large vehicle". Now, if the annotator wants to identify all the vehicles as cars regardless of the size of the vehicles for the newly built dataset, the proposal label for both of these classes can be renamed to "car". The tool configurations can be saved once the annotator has modified those to the task/dataset of interest. The saved configurations are maintained among separate annotation sessions to reduce the overhead of setting those repeatedly.

Multiple buttons are laid out on the user interface to request automatic proposals as shown in Figure 5.2b. The first two buttons can be clicked to generate uncontrolled and existing AABBs controlled proposals respectively with the pre-configured default methods. The next two buttons can be used to traverse the other proposal generation methods if needed. Shortcut keys are also provided to access each of the methods separately as depicted in Table 5.1. The shortcut keys might prove to be more flexible and faster in practice.

Key	Action (Generate proposals using)
Q	Pre-trained detector
W	Iterative detector
Shift + Q	Pre-trained detector for all the boxes or the selected box
Shift + W	Iterative detector for all the boxes or the selected box
E	Mask R-CNN model for all the boxes or the selected box
R	Thresholding approach for all the boxes or the selected box

Table 5.1: Keyboard shortcut keys for proposal generation.

Key	Action
P	Draw AABBs using 2 points
Shift + P	Draw RBBs using 3 points
Cntrl + P	Draw RBBs using 4 points

Table 5.2: Keyboard shortcut keys for manual annotation.

5.3 Manual RBB Annotation

There are three buttons in the tool as shown in Figure 5.3 that can be utilized to initiate each of the drawing mechanisms. Once activated, the annotator needs to follow the recommended ordering of the points to be drawn as mentioned in Figure 4.1. Keyboard shortcuts are extensively used in the LabelImg tool and we have added new shortcuts for initiating a manual RBB annotation as described in Table 5.2.

To create AABBs, the existing system of LabelImg needs the annotator to click on a corner and drag the mouse to the opposite corner. But, our 2-point system removes the need of holding down the mouse button while drawing the AABBs. The first click defines one of the corners of the AABB and as the mouse is moved a preview of that box is shown. With the second mouse click, the opposite corner point is set and the AABB is finalized. Though the annotator can specify any two opposite corners to draw an AABB in both the existing and the 2-point system, the tool always saves the top-left and bottom-right coordinates of the AABB in the output XML file to adhere to the PASCAL VOC format.

The 3-point system involves three mouse clicks following the recommended order as described in Section 4.1.1. This system is useful for rectangular objects such as overhead cars where the corners of the objects can be easily realized and clicked. For non-rectangular objects, the annotator needs to click on the corners of the imaginary bounding box.

Similarly, the 4-point system needs four mouse clicks as described in Section 4.1.2. In contrast to the 3-point system, the 4-point system is useful for ellipsoid shaped objects such as wheat heads where the corners are not easily clickable, but the two endpoints of the object and its thickness can be easily clicked.

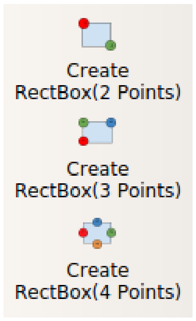


Figure 5.3: Multipoint box drawing system (UI buttons).

5.4 Availability

The source code and the relevant instructions to run FastRoLabelImg will be available at `https://github.com/p2irc/FastRoLabelImg`. The annotation tool is licensed under MIT License [1] that is a free software license and allows reusability of the code.

5.5 Summary

In this chapter, we described the basic layout of FastRoLabelImg and the specific UI components that are added to access the proposed proposal generation as well as the manual RBB annotation methods. We also documented the list of related keyboard shortcut keys. In addition, we discussed how the AABBs and RBBs are parameterized in the output XML files.

6 Data Acquisition and Annotation

We chose three datasets of diverse elongated objects (overhead images of wheat heads, cars, ships) to evaluate our proposed RBB methods. Statistics regarding the RBB annotations in these datasets are summarized in Table 6.1. In this chapter, we describe the data acquisition techniques and annotation procedures, present samples images and outline the observations on each of the datasets by analyzing the properties of the objects in the images.

6.1 Wheat Dataset

The wheat dataset is adapted from the Global Wheat Head Detection (GWHD) dataset [25]. The original dataset is comprised of wheat head images from ten different locations around the world covering a large number of genotypes from Europe, North America, Australia, and Asia, and collected over four years. Wheat head images can be used to predict the size and density of the wheat heads. These early predictions facilitate the farmers to make informed management decisions on their fields. Researchers originally captured 2219 high-resolution RGB images which were then subdivided into small patches of size 1024×1024 to obtain 4,698 RGB images in the final dataset. These images were labeled using an online annotation platform. Therefore, the full dataset consists of 4,698 RGB images with 188,445 wheat heads labeled with AABBs.

For our automatic RBB proposal generation experiments, we have randomly sampled images from four of the dataset subdomains named usask_1, inrae_1, rres_1 and ethz_1 that represent Canada, France, UK, and Switzerland respectively. In our sampled dataset, we have 200 images each from Canada, UK, Switzerland, and 176 images from France. These images originally had AABB ground truths only. However, we need RBB ground truths to test our RBB proposal generation method. Therefore, the selected images were manually annotated by a single annotator with RBBs using RoLabelImg[7] to conduct the experiments. Annotations were done over multiple sessions to control the fatigue of the annotator and maximum focus was given for accurate annotations by following the guidelines of the domain experts. The resulting annotations are summarized in Table 6.2 for each of the dataset subdomains and Figure 6.1 shows some example images with both AABB and RBB ground truths for a side by side comparison. In terms of the number of labeled heads with RBBs, the 200 images from Switzerland have the largest number of annotated wheat heads measuring 13,310 with an average of 67 heads per image. The 176 images from France have the lowest average with 20 heads per image and a total of 3,603 annotated wheat heads. Overall, the wheat dataset has 776 images containing 30,449 RBB annotated wheat heads and on average 39 wheat heads per image. For the participant study that is conducted to evaluate our manual RBB annotation methods, we draw images (3-4 images for each of the 12 participants) from utokyo_1 and utokyo_2 subdomains. These images are similarly manually annotated with RBBs.

Name	No.of images	No. of labeled objects	Avg no. of objects/image
Wheat	776	30,449	39
Car	150	9,330	62
Ship	4,500	5,401	1

Table 6.1: Datasets.

Source	Country	No. of images	No. of labeled heads	Avg no. of heads/image
USask_1	Canada	200	5,831	29
RRes_1	UK	200	7,705	38
ETHZ_1	Switzerland	200	13,310	67
INRAE_1	France	176	3,603	20

Table 6.2: Subdomains of the wheat dataset.

Furthermore, we analyze some of the key properties of the ground truth bounding boxes to have a better sense of the contents of the images. Figure 6.2 shows an analysis of the wheat dataset. The first row of the figure outlines the distribution of the aspect ratio for both ground truth AABBs and RBBs from left to right respectively. We see that the distribution of aspect ratio for the RBBs shifts to the right compared to the other distribution for the AABBs. This indicates that the traditional AABB annotations couldn't fit the objects as tightly as possible in images with many elongated wheat heads because of rotational variance. This idea becomes evident by observing the distribution of rotations for the RBBs. By observing the high number of objects per image from the histogram, we can imagine the possibility of having many densely crowded areas in the images to accommodate all the objects. As shown in Figure 6.1, these densely crowded areas result in occluded wheat heads. Therefore, objects in the images have high density, arbitrary rotations, occlusion in dense areas and, blur issues (observed in some images by inspection). All these factors combine to make the wheat dataset a challenging one to annotate with RBBs from scratch. If our proposed method can reduce the workload on this dataset by accurate RBB proposals and easier manual annotation, it will tend to reduce the RBB annotation time even more on easier datasets.

6.2 Car Dataset

The car dataset is prepared by selecting drone-view images from the CARPK dataset [47]. The original dataset contains cars from four different parking lots. The native dataset has 1,448 images with 89,777 cars and only AABB annotations are available. Many of the images are close to identical as the drone used to collect the images were stalled or moving slowly. To reduce the overlap between images, we have taken each 10$^{\text{th}}$ image of the dataset and a total of 150 images are selected. Similar to the wheat dataset, selected images are manually labeled with RBBs for further experimentation.

As shown in Table 6.1, the sampled car dataset has 9,330 instances of annotated cars and the average number of cars in each of the images is 62. The average is significantly higher than the wheat dataset. Figure 6.3 shows two

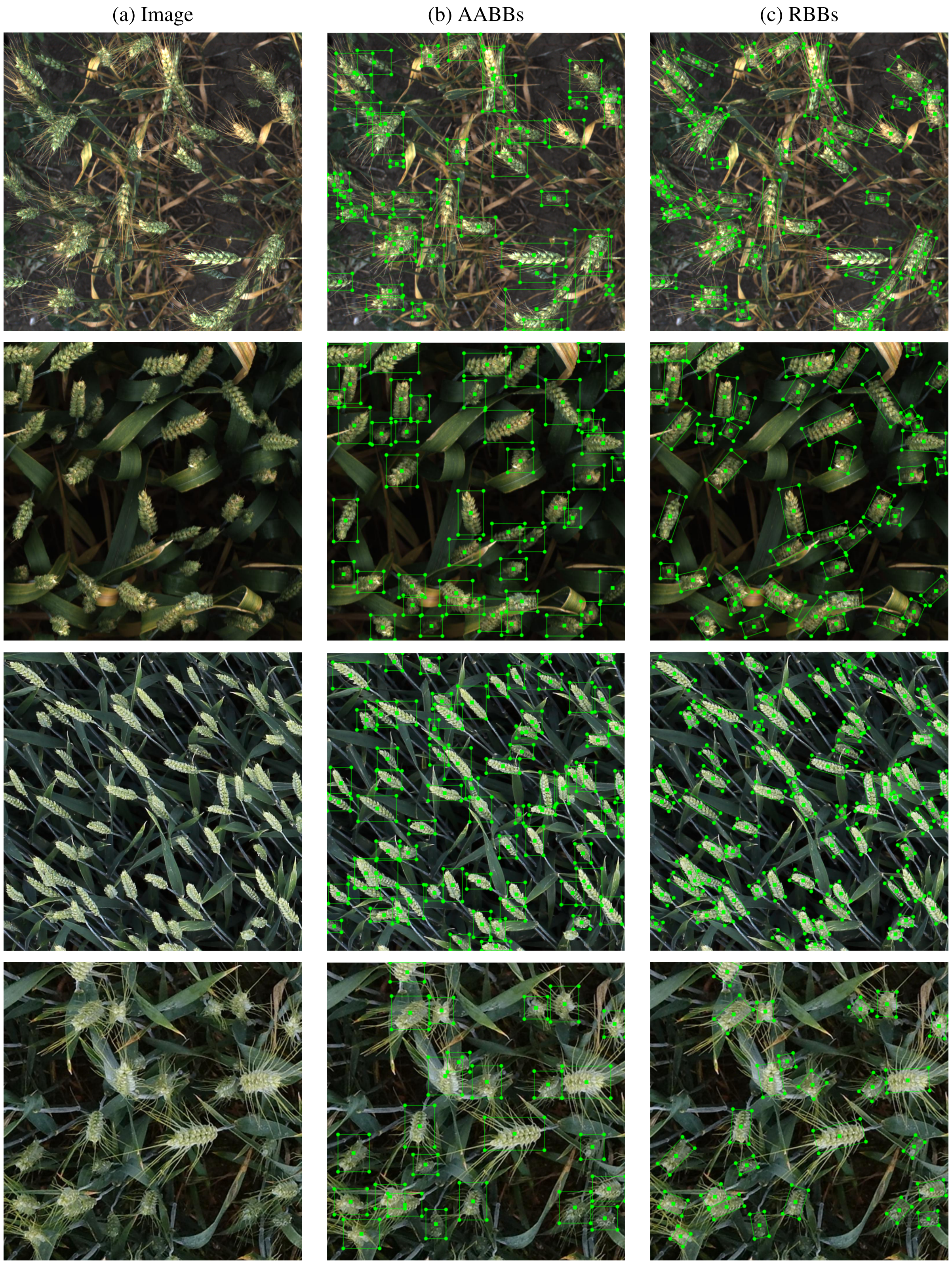


Figure 6.1: Example images of the wheat dataset from four subdomains namely usask_1, inrae_1, rres_1, and ethz_1 respectively (top to bottom). a) Input image. b) Ground truth AABB annotations. c) Ground truth RBB annotations.

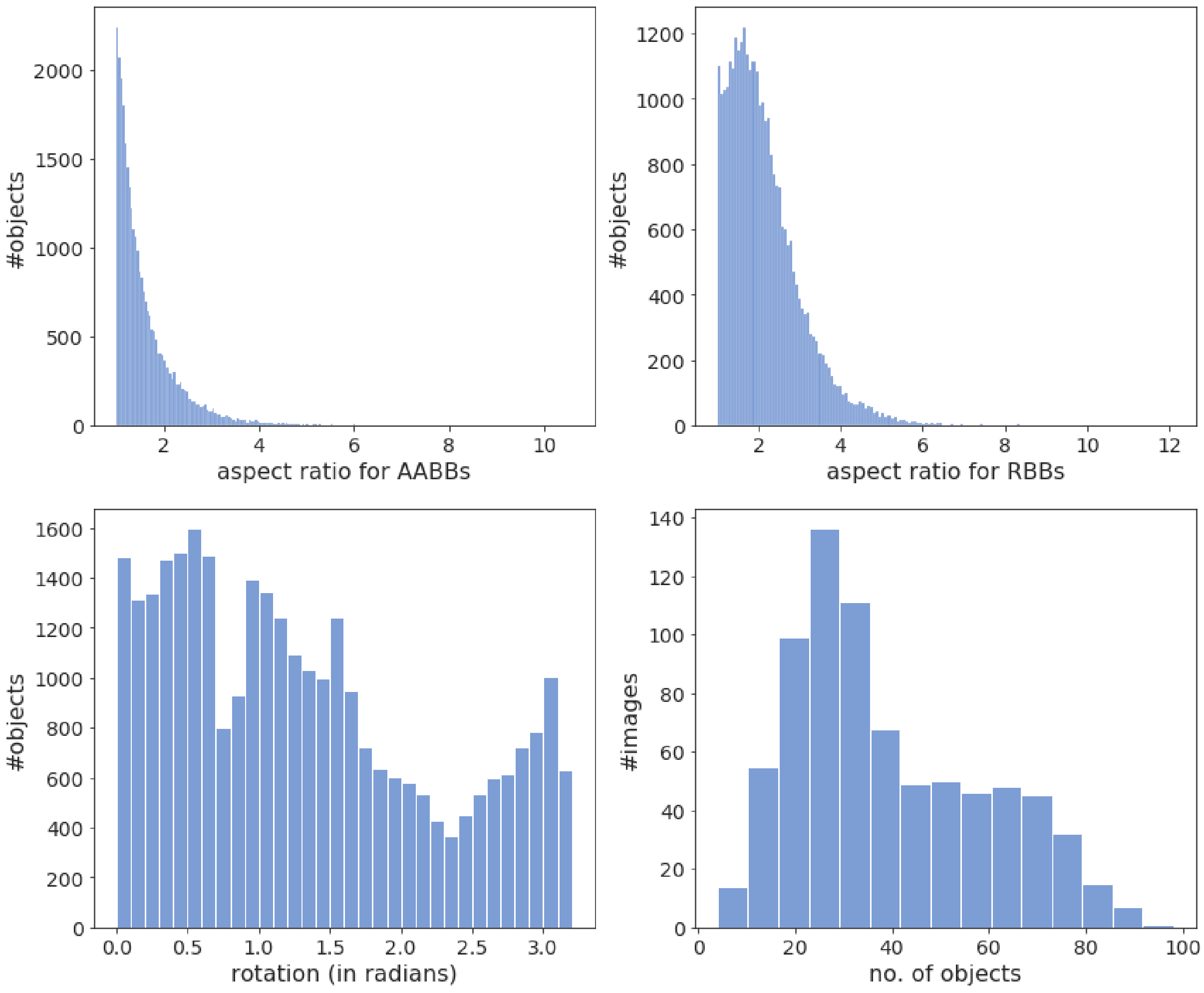


Figure 6.2: Summary of object annotations in the wheat dataset, including the distribution of aspect ratio for AABBs (top-left), aspect ratio for RBBs (top-right), rotations of RBBs (bottom-left), and the number of annotated objects with RBBs (bottom-right).

example images from the car dataset along with their respective AABB ground truths and RBB ground truths. The example images depict that the images in the dataset are densely packed with different kind of cars. Hence, a high average per image is observed.

Similar to the wheat dataset, we analyze some properties of the bounding boxes such as aspect ratio and orientation. We also present the distribution of the number of objects per image. The obtained distributions are illustrated in Figure 6.4. From the two distributions of aspect ratio (top-row), we observe that the aspect ratio for both RBBs and AABBs is mostly around two. Therefore, both of the annotation techniques captured the true aspect ratio of many of the cars. In other words, there are fewer variations in rotations for the cars. However, we do see some changes in the distributions which depict that some cars are now more tightly encapsulated with RBBs than before. By seeing the other two distributions, we develop an understanding that most of the cars are closely aligned either verticaly or horizontally and the number of cars per image largely varies between 25 and 75.

6.3 Ship Dataset

The images of the Airbus Ship Detection Challenge dataset [10] are used to form the ship dataset. The challenge dataset has more than 100k satellite images but only around 25% of the images contain ships. First, we process the dataset by discarding the images with no ships. Then, only the images ($\approx$9k) where the area of each of the ships is more than 1,000 pixels are kept. Finally, we randomly select 4,500 images from those to build the ship dataset. The challenge dataset supplies the segmentation masks of the ships. Therefore, we use the masks to derive initial RBB annotations. Later, accurate RBB boxes are produced by manual corrections.

Table 6.1 shows that the extracted ship dataset has around 5,401 RBB annotated ships in the 4,500 sampled images with an average of 1 ship per image. Some of the examples of this dataset are presented in Figure 6.5 accompanied by its ground truths. These images look vastly different than the other two datasets in terms of the number of instances in the images. Most of the images in the dataset contain 1-4 ships. Although in many of the images the ships are distinctive from the background, there are images where the foreground ships look very similar to the texture of the background sea.

Figure 6.6 shows the statistical properties of the bounding boxes. Based on the distributions, we find that the ships in the images are positioned in a varying number of orientations, and by using the RBB annotations we can localize the ships more accurately than the AABB annotations.

Automatic ship detection allows monitoring incidents like accidents, illegal cargo movement, or fishing and drug trafficking in the open sea. As the industry generates a large number of surveillance images or videos, a fast RBB annotation tool will be useful for training better models rapidly.

6.4 Summary

To investigate the effectiveness of our proposed methods, we construct three diverse datasets. The images in the datasets are obtained from their publicly available counterparts with some pre-processing steps. Then, we incorporate

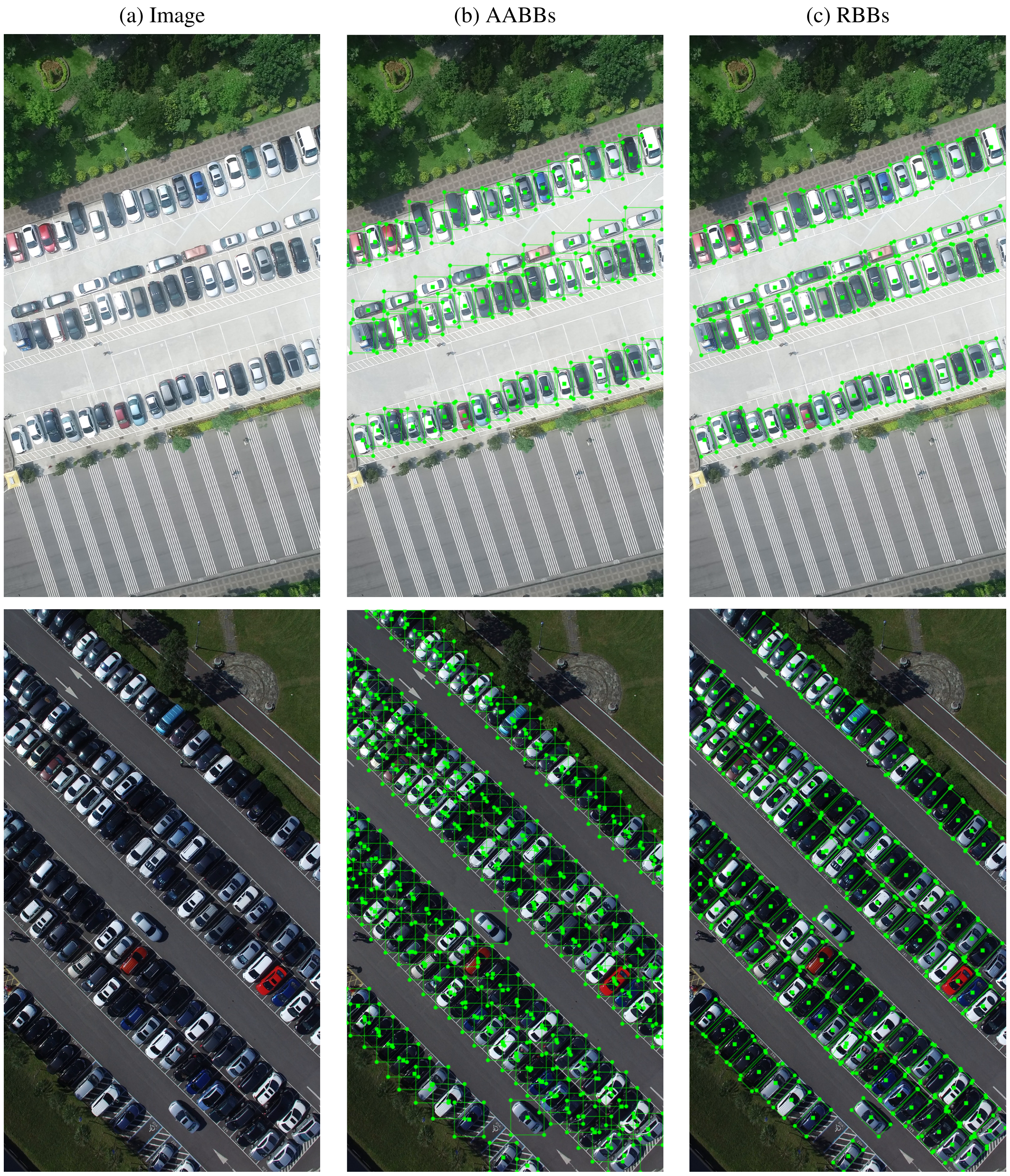


Figure 6.3: Example images of the car dataset. a) Input image. b) Ground truth AABB annotations. c) Ground truth RBB annotations.

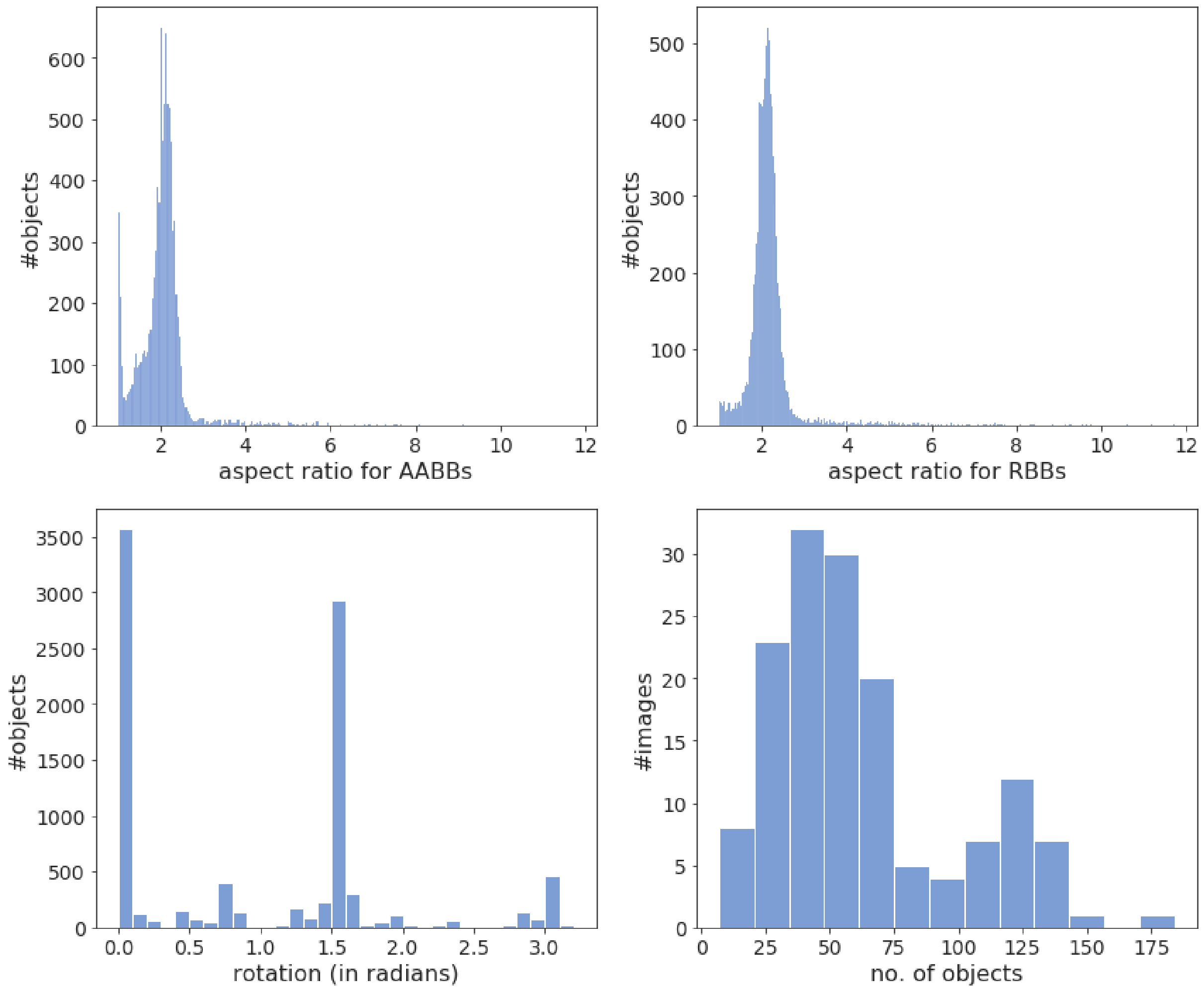


Figure 6.4: Summary of object annotations in the car dataset, including the distribution of aspect ratio for AABBs (top-left), aspect ratio for RBBs (top-right), rotations of RBBs (bottom-left), and the number of annotated objects with RBBs (bottom-right).

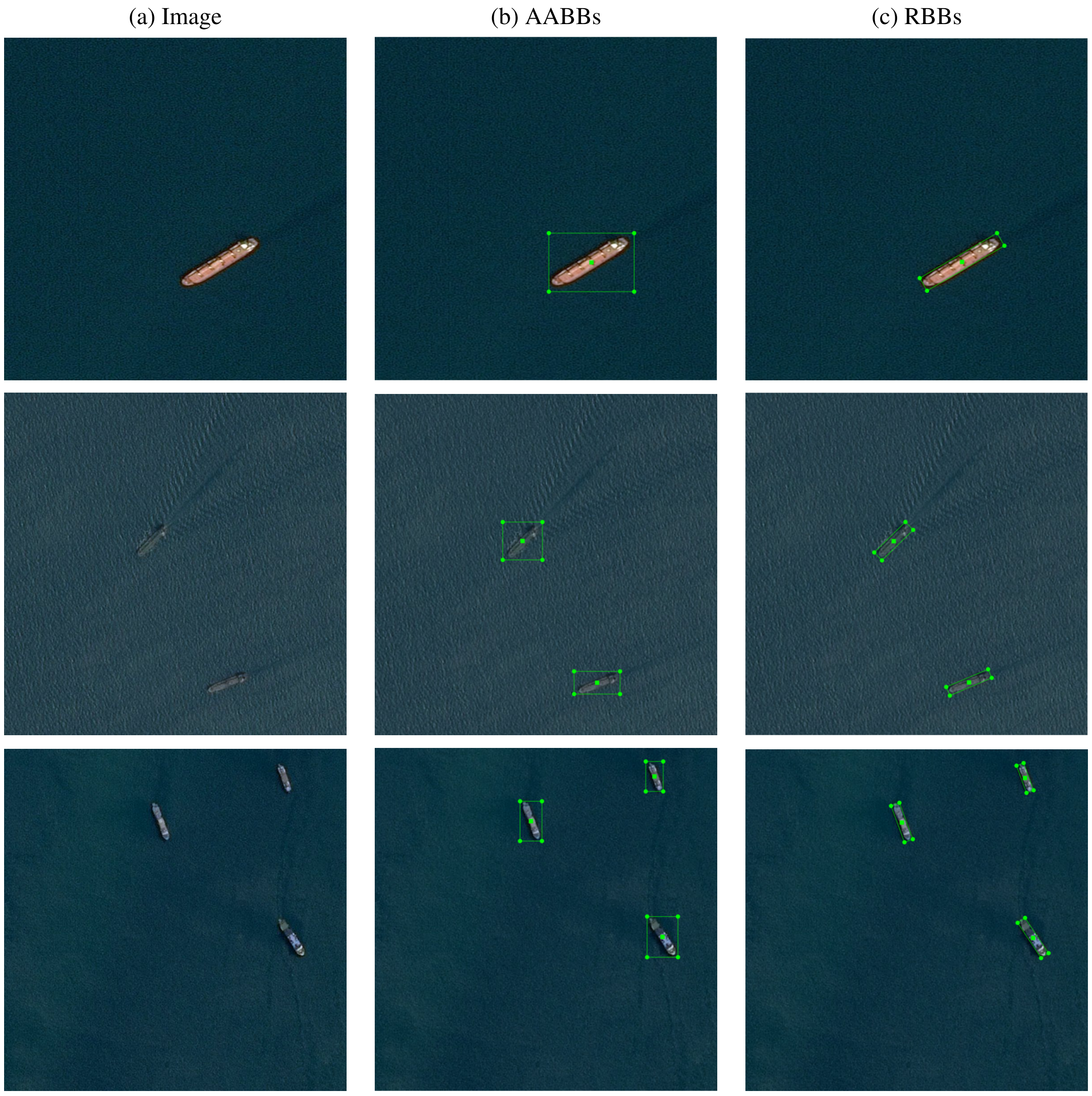


Figure 6.5: Example images of the ship dataset. a) Input image. b) Ground truth AABB annotations. c) Ground truth RBB annotations.

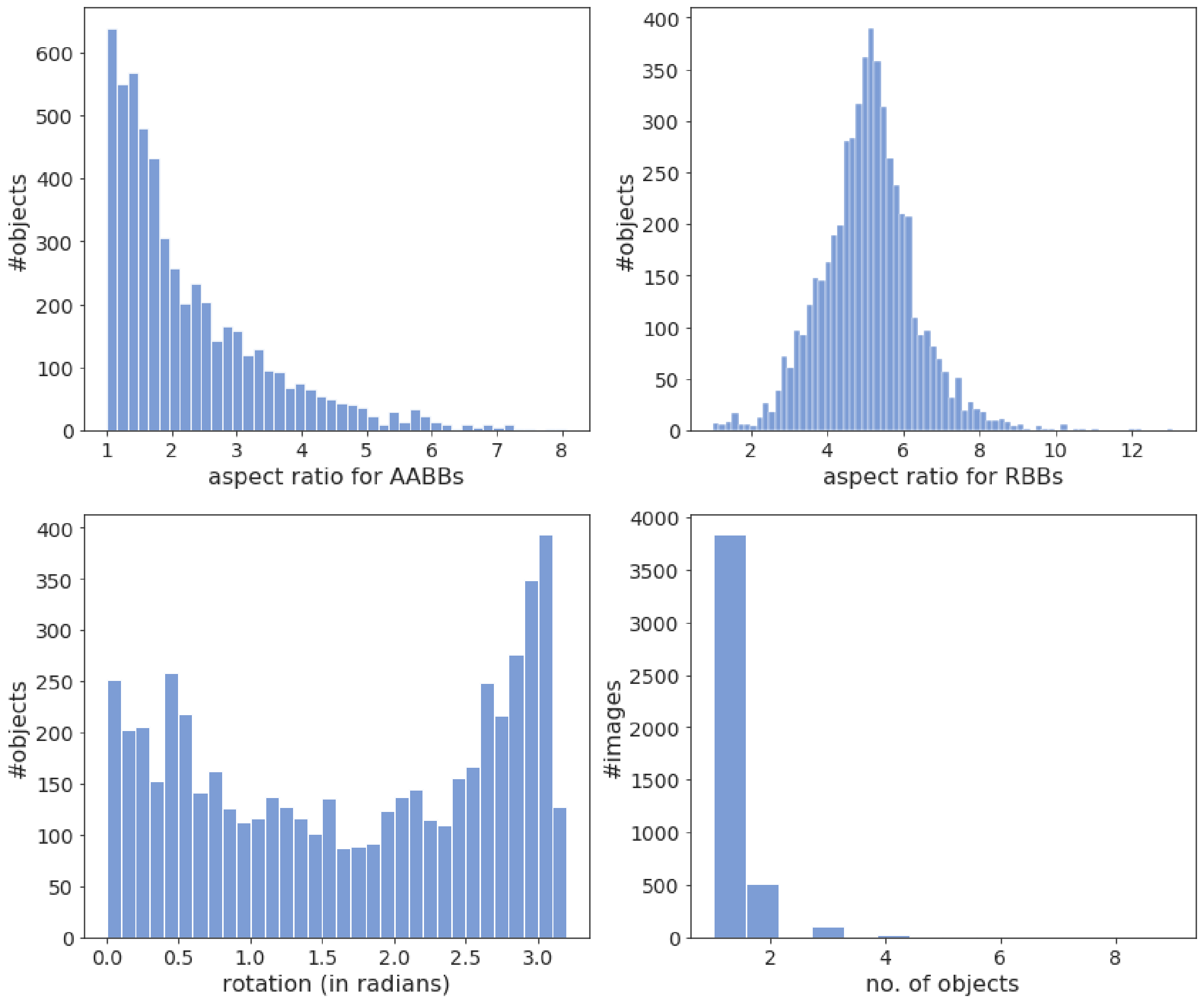


Figure 6.6: Summary of object annotations in the ship dataset, including the distribution of aspect ratio for AABBs (top-left), aspect ratio for RBBs (top-right), rotations of RBBs (bottom-left), and the number of annotated objects with RBBs (bottom-right).

manual RBB annotations with the images to obtain our final datasets. Each of the datasets has both AABB and RBB annotations. By analyzing the image annotations, we observe that the RBBs will be effective to improve localization. Some of the images from the wheat and car datasets are later used to conduct a participant study to evaluate that the proposed multipoint drawing system (Section 7.1). The complete datasets are used to quantify the possible annotation workload reduction our RBB proposal generation method can achieve (Section 7.2).

7 Experiments and Results

In this chapter, we describe two experiments conducted for this thesis. The first one is a participant study to assess the proposed manual RBB annotation method. The second experiment is used to determine the manual workload reduction by the proposal generation methods. For each experiment, we present the data used to run the experiment as well as report and discuss the observed results along with the evaluation procedures.

7.1 Manual RBB Annotation Experiment

To complete the task of annotating an object with an RBB, the current annotation tool requires the annotator to first draw an AABB which can be rotated afterward by repeated key presses. After rotation, the box often needs to be resized again to fit the object. This rotate and resize operations may need to be repeated a number of times to obtain a tight fitting box. On the other hand, the developed 3-point and 4-point systems can be used to directly create a better initial RBB depending on the shape of the object and minimize the number of box adjustment steps. Therefore, we expect that the multipoint drawing system will reduce the time and effort needed to draw an RBB compared to the current system. To quantify the time-effectiveness of the proposed system and investigate the annotation accuracy we conducted a participant study. The design of the participant study and the analysis of the obtained data are described in the following subsections.

7.1.1 Study Methods and Design

The purpose of the study was to compare the speed and accuracy of the proposed RBB drawing methods to the current method. The study used a 3×2 within-participants RM-ANOVA with factors *Annotation Method* (3-point, 4-point, current) and *Object Type* (car, wheat). The car and wheat instances represent rectangular and ellipsoid shaped objects respectively. The dependent variables were annotation time and accuracy. We analyzed the dependent variables separately.

Hypotheses

The four hypotheses for the study are listed in Table 7.1.

Tasks and Data

Each of the participants annotated a set of images using the current, 3-point, and 4-point systems. The images could be of two types with respect to the type of their object instances. The set of images were unique to each participant

ID	Hypothesis
H1	The 3-point and 4-point methods will be faster than the current method
H2	The 3-point and 4-point methods will be more accurate than the current method
H3	The 3-point method will be faster than the 4-point method as it requires fewer clicks
H4	The 4-point method will be more accurate than the 3-point method as the specified points are mostly on the edges of the object

Table 7.1: The hypotheses for the participant study.

but they remained the same across all conditions. As the participants repeatedly annotated the same set of images with different conditions, there could be a potential learning effect on the observed annotation time and accuracy. Therefore, we counterbalanced the order of object types and annotation methods.

We chose a subset of wheat and car images for the study as it wasn't feasible to use all the wheat and car images for annotation by the participants. Annotating too many images would cause long annotation sessions and the increased workload could cause fatigue and negatively impact the quality of study data. Therefore, we used a limited number of wheat and car images for the study. The car images were selected from the car dataset. And, the images containing wheat heads were taken from the GWHD dataset. Particularly, the images were collected randomly from the sub-datasets named UTokyo_1 and UTokyo_2. As the name suggests, the origin of these images was Japan. Our target was to select 50 wheat heads and 50 car instances for each participant. However, we couldn't exactly select 50 wheat heads as the number of instances in the wheat images was random. Therefore, each participant was given 3-4 wheat images with the expected count of wheat heads in the range 45-60. For the car dataset, the average number of cars per image was 62. Therefore, we prepared 3-4 car images per participant to accommodate exactly 50 cars by cropping the original images. We asked the participants to avoid any object instances that were partly visible in the images. For annotating cars, the participants were requested to fit the objects tightly including the mirrors and wheels. For wheat heads, the participants were instructed to fit the objects with RBBs as tightly as possible excluding the awns. The criteria for annotating objects and the degree of annotation accuracy depend on specific tasks or models being developed. Some models [28] enlarge RRoIs during the feature extraction to include more contextual information to improve performance. Being lenient with the degree of annotation accuracy may promote faster annotation, but we chose to get as accurate annotations as possible in our study.

We have also created the ground truth RBBs with the annotation tool named RoLabelImg[7] to measure accuracy of the participants annotations. Figure 7.1 presents some examples of the wheat images used in the study along with their RBB ground truths. And, Figure 7.2 illustrates the variations in the appearances of both kinds of objects.

Procedure

We ran the study remotely using the laptops/desktops of the participants due to the COVID-19 restrictions. The study procedure is depicted in Figure 7.3. For each of the participants, we supplied the images to be annotated and a Windows-compatible installer of FastRoLabelImg. Therefore, we started the study with a device compatibility check.

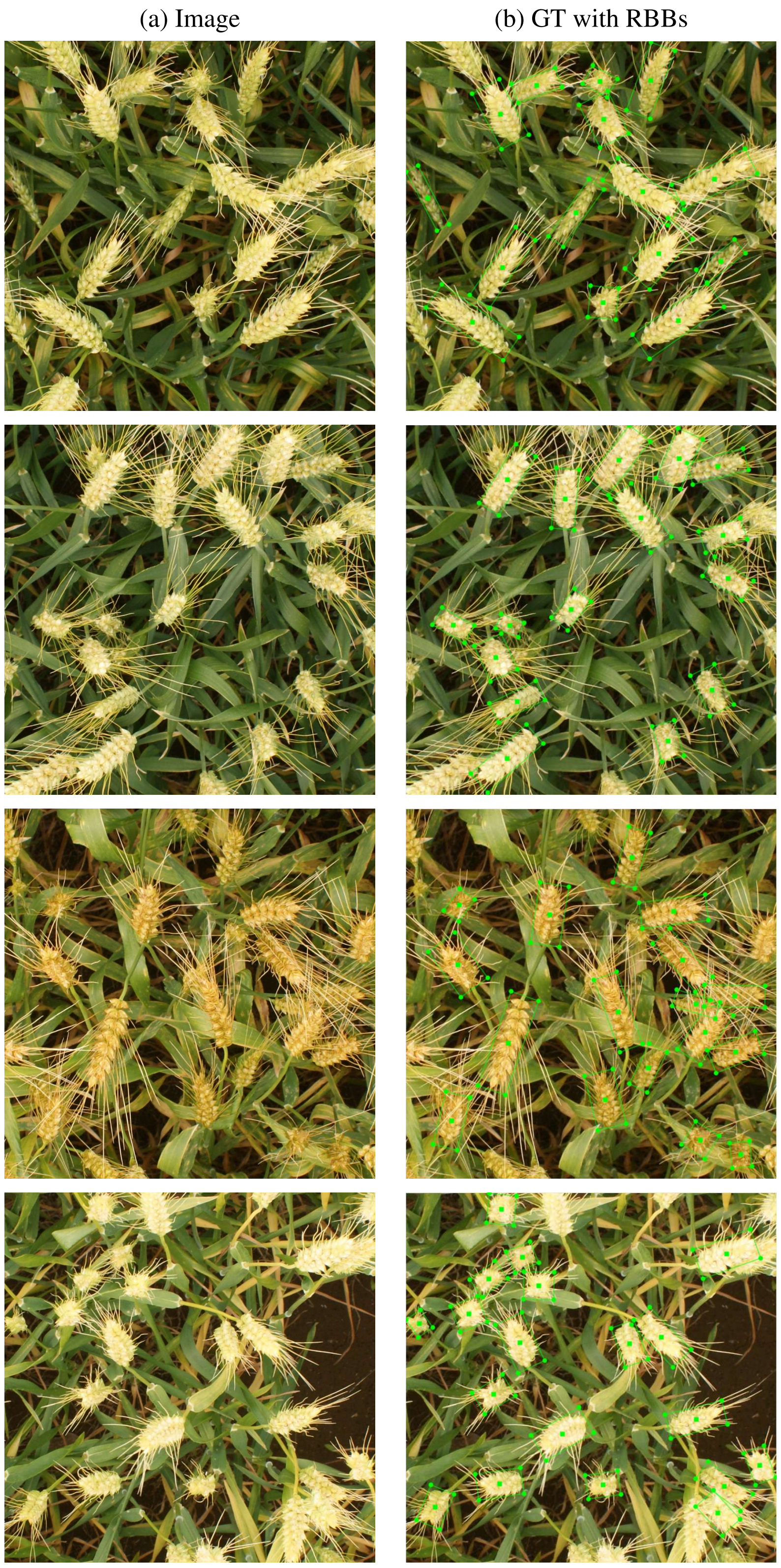
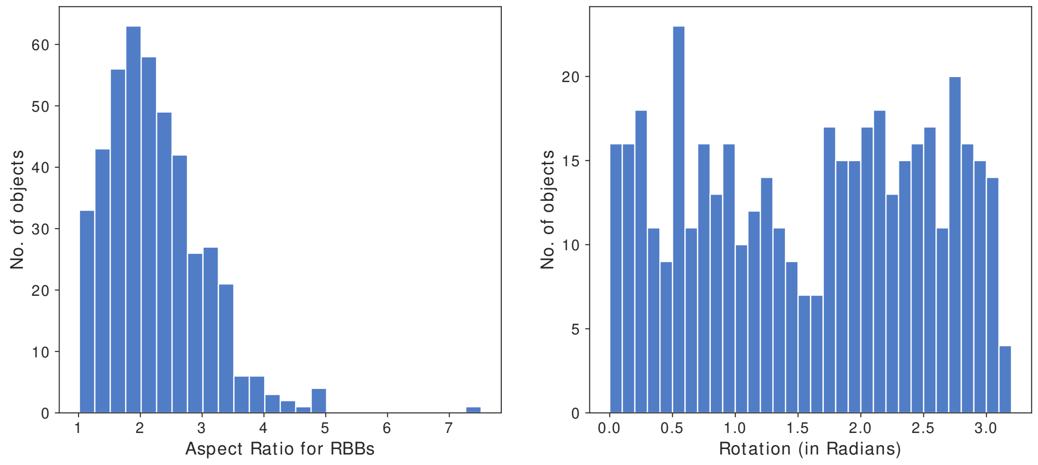
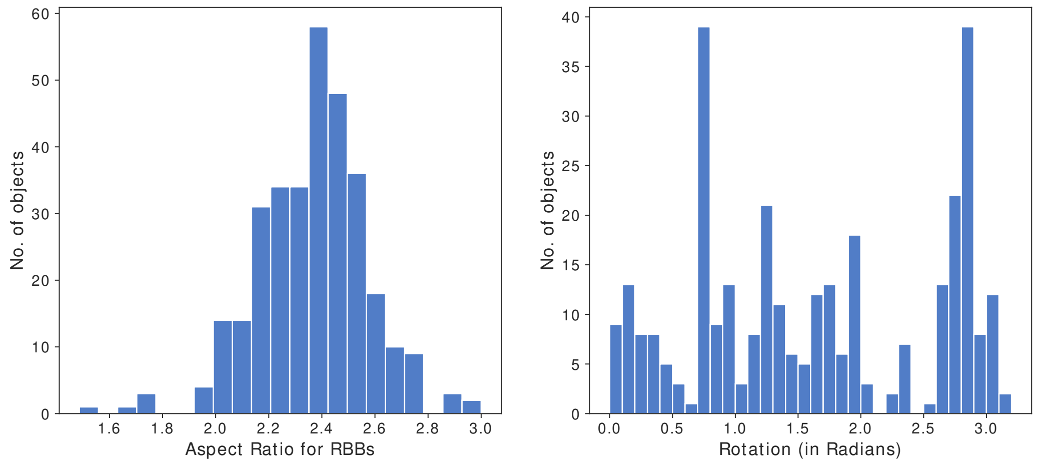


Figure 7.1: Example images of the wheat instances from Tokyo, Japan. a) Image. b) Ground truth with RBBs.



(a) Wheat instances



(b) Car instances

Figure 7.2: Distribution of aspect ratios (left) and rotations (right) of the GT RBBs for a) Wheat instances b) Car instances.

If compatible the participants signed an informed consent form (Figures A.1, A.2, A.3) online. Then, the participants filled up a pre-questionnaire (Figure B.1) which queried them about the level of experience of using computer, mouse, and annotation tool as well as some basic information e.g. gender, age. After completing the form, FastRoLabelImg was installed on their devices. Any required support and guidance were given remotely.

Once the software was installed, the participants were given a demonstration of the interface of the annotation tool and each of the annotation methods. In addition, the criteria of accurate annotations were explained to them with examples for both wheat and car instances. Then, they were given a few minutes to practice the box drawing approaches on some practice wheat and car images. During the practice tasks, necessary feedback was given to the participants on their annotations over video conferences. They were also instructed to complete the annotations as fast and as accurately as possible for all of the methods.

Finally, the participants performed the main annotation tasks. They took part in two separate sessions. Each session involved annotating either wheat or car images with the three conditions repeatedly. The participants had the opportunity to take a break at the end of a session. During the sessions, the RBB annotations were saved to measure the accuracy (using IoU) of each box by comparing those to the ground truth RBBs. The annotation time was also recorded for each of the boxes using the annotation tool internally. The time difference between the starting point of two consecutive boxes was considered as the annotation time of the first box. Also, when the save button is clicked, the time difference between the starting point of the last box and the save operation was recorded as the annotation time of the last box. At the end of every session, the participants completed a user preference questionnaire (Figure B.2). They were asked to rank the annotation methods for the recently concluded object type based on the annotation speed (higher rank represented faster method), difficulty (higher rank represented easier method), and overall preference. In total, the study lasted around 90 minutes for each participant.

Participants

Twelve participants (6 female and 6 male) were recruited from a local university for the participant study. The age range of the participants was 24-31 (mean 27.3). All of them were right-handed. The participants self-reported very high familiarity with computer and mouse. But, they reported a mixed experience of using annotation tools as shown in Figure 7.4. Six of the participants self-reported no experience of using annotation tools and the others had low to high familiarity (4 low, 1 moderate, 1 high).

Apparatus

The experiments were conducted on the personal laptops of the participants due to the remote nature of the study. All the devices had MS Windows 10 operating system and the display sizes were in the range 15-17 inches. The installed annotation tool (FastRoLabelImg) was written in Python (PyQT5). The annotation inputs were received using a USB optical mouse.

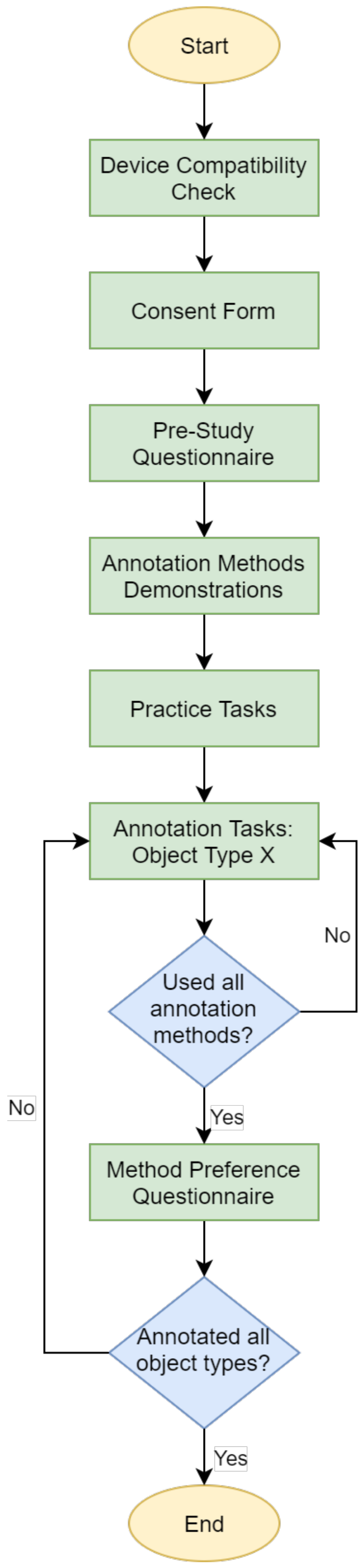

Start
Device Compatibility Check
Consent Form
Pre-Study Questionnaire
Annotation Methods Demonstrations
Practice Tasks
Annotation Tasks: Object Type X
No
Used all annotation methods?
No
Yes
Method Preference Questionnaire
Annotated all object types?
Yes
End

Figure 7.3: Study procedure.

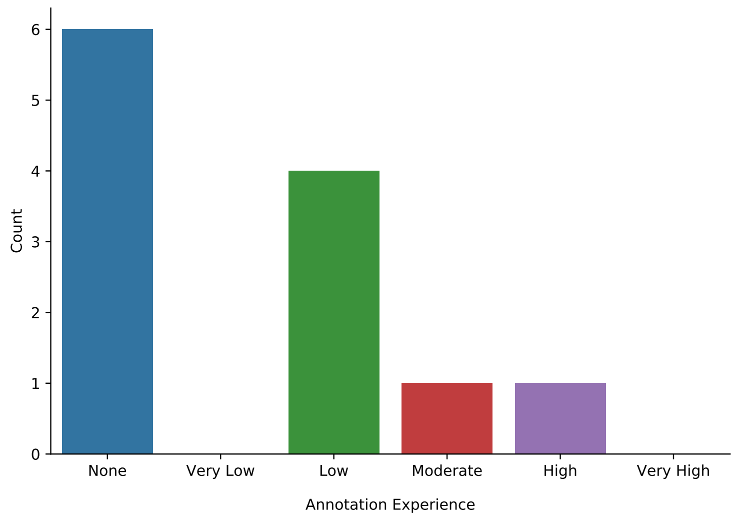


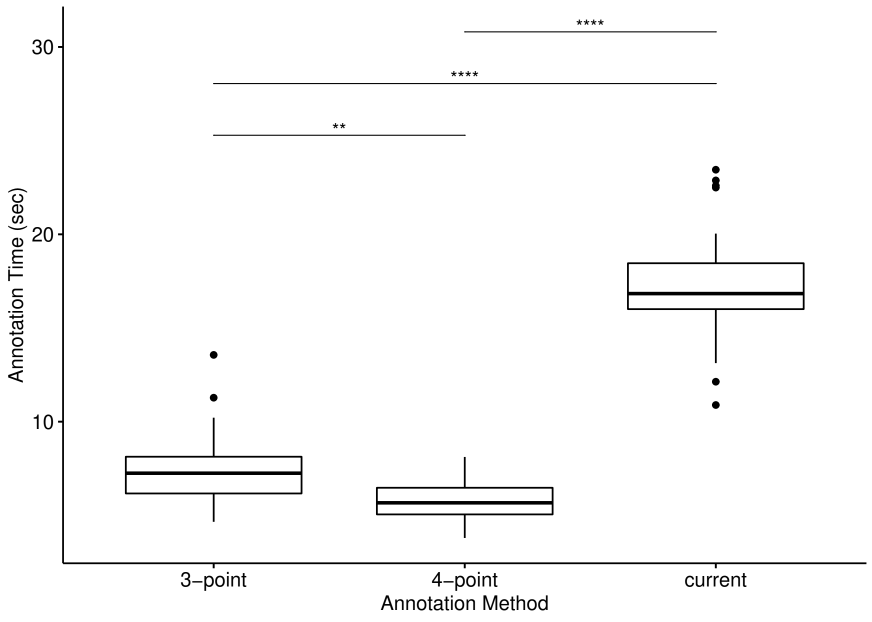
Figure 7.4: Participants experience of using annotation tools.

7.1.2 Results

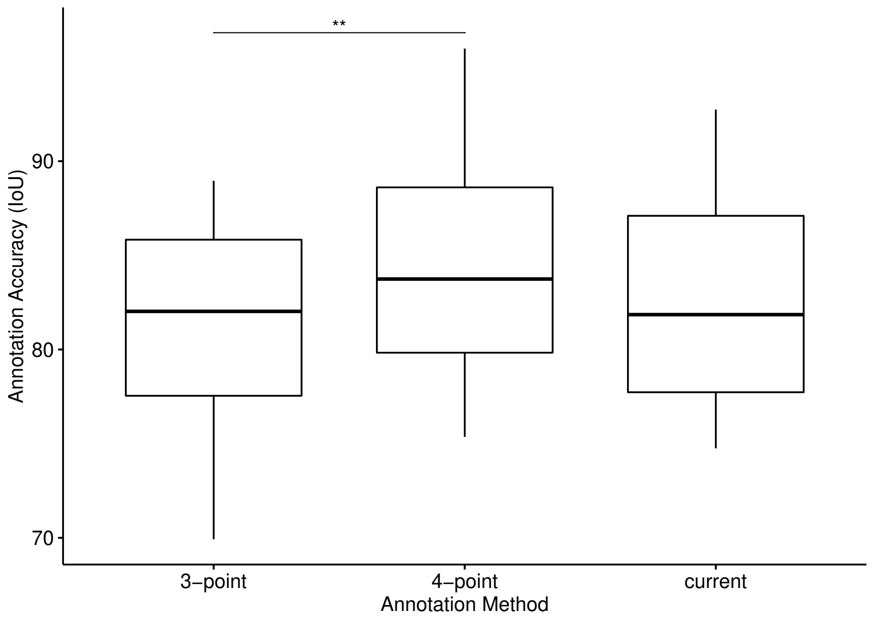
After the data collection phase, we applied some filtering steps to obtain the final set of objects to analyze the effect of annotation method and object type on annotation time and accuracy. To answer our research questions, we needed three RBB annotations with annotation time (using each method) and the ground truth RBB for each of the objects. The ground truth RBBs were required to measure the accuracy of the annotations and create associations among RBBs across the three annotation methods. Therefore, the following operations were performed for each image:

- We discarded all the RBBs that had an IoU less or equal to 0.1 with any of the ground truth RBBs.
- Then, we matched the RBBs obtained from each annotation method (condition) with the ground truths and discarded any object that had missing RBBs for one or more of the methods (due to a participant failing to localize the object with some methods)
- After linking the RBBs across methods, we found the accuracy of the RBBs by calculating IoU with the respective ground truth boxes.

In the end, we got a set of objects having annotation time and accuracy observed by each of the annotation methods. As the last step, for both annotation time and accuracy, we determined outliers within each object type and annotation method for each participant. A trial was considered an outlier and discarded if the measured value was three s.d away from the respective group mean. Out of 1236 trials, 67 trials were discarded based on annotation time and 16 trials were discarded based on annotation accuracy. For annotation time, outliers were caused due to spending an unusually long time while annotating some objects. The reasons behind unusual annotation time could be momentary lack of concentration or getting confused about the annotation specifics in some scenarios.



(a) Annotation time



(b) Annotation accuracy

Figure 7.5: Dispersion of the data through their quartiles and the effect of annotation method on a) Annotation time b) Annotation accuracy. (** : $p \leq 0.01$, **** : $p \leq 0.0001$)

Annotation Time

ANOVA results showed that for annotation time there was no statistically significant interaction $(F(2,22) = 2.24, p = 0.130)$ between *Annotation Method* and *Object Type*, suggesting that the effect of annotation methods does not depend on the object types. The *Annotation Method* had a significant main effect $(F(2,22) = 132.09, p =< 0.0001)$ on annotation time; Greenhouse-Geiser correction was applied for sphericity violation. Post-hoc pairwise t-tests (Bonferroni corrected) revealed significant difference between the annotation methods (all $p < 0.05$). The mean annotation time was 7.54s (s.d. 2.10s) for 3-point, 5.78s (s.d. 1.08s) for 4-point and 17.3s (s.d. 3.32s) for the current method. Therefore, we accept **H1** and must reject **H3**.

The mean annotation time using different conditions and the statistical significance results are summarized in Figure 7.5a.

Annotation Accuracy

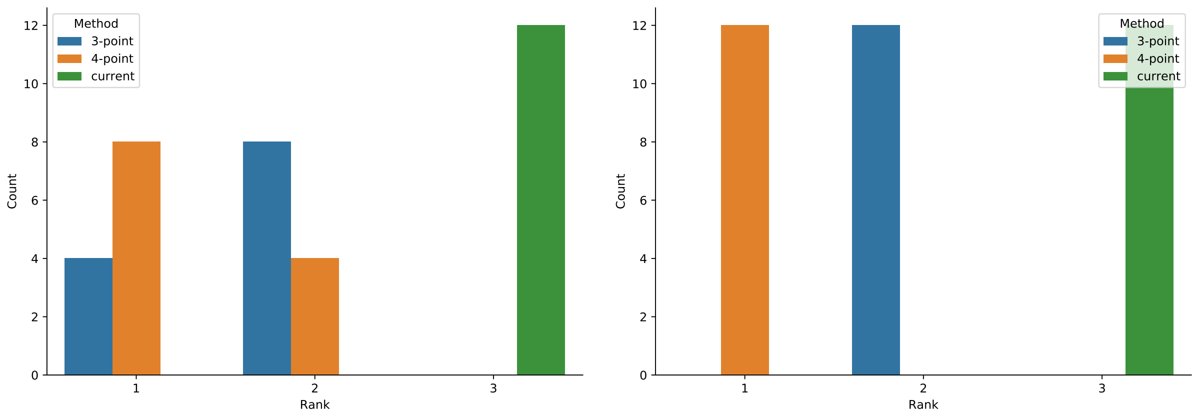
RM-ANOVA results indicated no significant interaction between *Annotation Method* and *Object Type* $(F(2,22) = 4.57, p = 0.15)$ for annotation accuracy, again suggesting that the effect of annotation methods does not depend on the object types. We found that there was a significant main effect of *Annotation Method* $((F(2,22) = 4.44, p = 0.024)$ and *Object Type* $((F(1,12) = 541.39, p =< 0.0001)$ on annotation accuracy. Post-hoc pairwise t-tests (Bonferroni corrected) showed significant difference between the 3-point and 4-point methods ($p = 0.007$) as illustrated in Figure 7.5b. The mean annotation accuracy was 81.5 (s.d. 5.14) for 3-point, 84.3 (s.d. 5.97) for 4-point and 82.4 (s.d. 5.66) for the current method. Therefore, we must reject **H2** and accept **H4**.

Participant Preferences

For each individual participant, the responses for ranking the annotation methods were consistent across categories such as annotation speed, difficulty, and overall preference. For example, if a participant assigned rank i to a method x based on annotation speed, he had given the same ranking to that method based on the other two categories (e.g. perceived difficulty and overall preference). Therefore, we presented the results across the three categories for both object types as Figure 7.6 instead of three separate figures. For annotating car objects, 75% of the participants preferred the 4-point system and the other 25% participants preferred the 3-point system as shown in Figure 7.6a. For wheat instances, all the participants preferred the 4-point system over others (Figure 7.6b). For both types of objects, the current system got the lowest ranking based on the participant preferences.

7.1.3 Discussion

By conducting the study, we found two main results. First, both the 3-point and 4-point systems improved annotation time compared to the current system. According to the mean annotation time, the 3-point system is 2.3 times faster, and the 4-point system is 3 times faster than the current system. Secondly, in terms of annotation accuracy, the 4-point system maintained a high-accuracy similar to the current system while being the fastest irrespective of the shape of



(a) Car instances **(b)** Wheat instances

Figure 7.6: Ranking of annotation methods based on participant preferences (i.e. annotation speed, difficulty and overall preference) for a) Car instances b) Wheat instances.

the objects and highly preferred by the participants. Though the 3-point system was faster than the current system, it lagged behind (by 1.1% according to the mean) in annotation accuracy.

For the 4-point system, the points to be clicked were mostly on the edges of the objects. For example, the wheat instances were identified using two opposite end points and its thickness whereas the cars were indicated by pointing out the center of two opposite sides and the extreme points of the other two sides. Therefore, the participants were able to quickly annotate the instances while maintaining high accuracy. However, the 3-point method involved a higher degree of approximations for ellipsoid shaped objects because distinct object corners are not apparent. It might be one of the reasons for the 3-point system having a slower annotation speed in contrast to the 4-point system. Another possible reason could be the absence of live previews of the RBBs when the points were being clicked. It caused the participants to have a delayed understanding of inaccurate annotations and needed multiple attempts to get accurate RBBs in some cases. But, the 3-point system was still faster than the current system due to fewer steps. Though the current method was slower, it had a higher accuracy than the 3-point method. It suggests that the participants probably spent more time making the annotations as accurate as possible. However, we observed during the study that the participants increasingly compromised the accuracy when they had to stay on an object for a longer period. It might be one of the contributing factors for the lower mean accuracy of the current system compared to the 4-point method but it's not statistically significant ($p = 0.169$).

7.2 Proposal Generation Experiment

The proposed 3-point and 4-point RBB annotation methods can speed up manual RBB annotations. However, it still takes a long time to annotate a large dataset. The proposal generation methods can help by reducing the required number of manual annotations. Therefore, the generated proposals need to be accurate enough so that the respective objects don't demand manual corrections. We assess the efficiency of the proposal generation methods by measuring the percentage of manual workload reduction on three diverse datasets. In the following subsections, we discuss the necessary details of workload estimation with or without proposals and present the experimental results.

7.2.1 Workload Estimation

We estimate the manual workload to evaluate the different strategies of proposal generation. In the absence of proposal generation, the workload is measured as the number of total objects in the images of a dataset because the annotator will only create a new box for each of the objects. We count the objects of interest by the available ground truth RBBs. When proposal generation is used, the workload is modeled as removals of the incorrect proposals (false positives) and additions of the missed ground truth bounding boxes (false negatives). We calculate the number of such boxes per image using the precision and recall metrics following [12] and no actual participants is used in this evaluation process. When pre-existing AABBs are not available, the amount of corrections is determined as

$$\text{\# corrections} = \text{\# additions} + \text{\# removals} \tag{7.1}$$

where,

$$\text{\# additions} = \text{\# objects of interest} \times (1 - \text{recall}) \tag{7.2}$$

$$\text{\# removals} = \text{\# proposals} \times (1 - \text{precision}) \tag{7.3}$$

Also, precision and recall are defined as

$$\text{precision} = \frac{\text{\# correct proposals}}{\text{\# generated proposals}} \tag{7.4}$$

$$\text{recall} = \frac{\text{\# correct proposals}}{\text{\# objects of interest}} \tag{7.5}$$

However, if the proposals are filtered by a set of pre-existing AABBs, the number of false positives is limited by the number of given AABB annotations. Therefore, there can be at most one proposal for each pre-existing AABB and if that proposal is incorrect, a removal operation will always be followed by a box addition. The time taken to remove a incorrect proposal is minimal and constant compared to a box addition. Therefore, we only examine the number of box additions in this case to determine the number of corrections as expressed by

$$\text{\# corrections} = \text{\# additions} \tag{7.6}$$

7.2.2 Experimental Setup

We experiment with the wheat, car, and ship datasets described in the previous chapter. The RBB ground truth annotations are available for all the images of these datasets. We use the RBB ground truths to estimate the manual workload reduction for each dataset separately. Manual workload reduction (%) is determined using the cumulative number of ground truth bounding boxes and the cumulative number of required manual corrections for all the images in a dataset. It can be calculated as,

$$\text{Manual workload reduction (\%)} = \frac{\text{\# GT RBBs} - \text{\# corrections}}{\text{\# GT RBBs}} \times 100 \tag{7.7}$$

We run multiple experiments for each dataset and different proposal generation techniques are used in each of these experiments. We experiment using two detection models: pre-trained and iterative with or without filtering. When filtering is enabled, the detectors make use of the pre-existing AABBs to filter out many of the incorrect proposals, and a maximum of one proposal will be generated for each of the AABBs. We also include segmentation-based proposal generation approaches (Thresholding and Mask R-CNN) in our experiments that allow adapting the pre-existing AABBs to RBB proposals by estimating the area and orientation of the annotated objects. Lastly, we use both the detectors and AABB adaption approaches jointly in two of the experiments for each dataset. In this case, any AABBs for which the detection models can't produce a proposal are adapted to RBB proposals using the segmentation-based methods.

In our experiments, we consider a proposal as correct if the IoU overlap is greater or equal to 0.5 between the ground truth bounding box and the proposed bounding box as well as the difference in the rotations is less than 10 degrees. We configure the other parameters such as confidence score, NMS threshold, and box match IoU threshold to 0.5, 0.1, and 0.2 respectively. The confidence score threshold of 0.5 is often chosen for getting predictions from the object detectors in the literature [34, 60]. The other parameter values are set (by experimenting with the range 0.1 to 0.5) to capture the reduction of the number of manual corrections. However, these are not necessarily the optimal values and may change based on the dataset. We provide the ability to the annotator to adjust these values using the UI to achieve the best possible results. The images in a dataset are iteratively traversed to calculate the number of required corrections per image. For the iterative detector, proposals are generated on the i^{th} image using the model trained on $1 \ldots (i-1)$ images for evaluation. Also, the model is trained for 30 epochs for each of the images to keep a balance between the training time and the prediction accuracy.

7.2.3 Results

The experimental results are shown in Table 7.2 for all of the datasets. In the table, NA represents that a method does not apply to a dataset. For example, the pre-trained detector is not trained on wheat instances. Therefore, we couldn't use that model to experiment on the wheat dataset. Similarly, the Mask R-CNN model is unable to predict wheat or ship instances as it's not trained on those classes of objects.

When no AABBs are available, the pre-trained detector obtains a manual workload reduction of 27.4% and 65% for the car and ship datasets respectively. The iterative detector almost doubles (54.8%) the workload reduction for the car dataset and improves it by 4.1% for the ship dataset. Besides, the iterative detector introduces a 39.8% manual workload reduction for the wheat dataset. Both the pre-trained and iterative detector achieve a better performance when proposals are generated for a set of given AABBs. The performance of the iterative detector increases by at least 10%

Detector	Filtering	Adapting	Wheat	Car	Ship
Pretrained	No	No	NA	27.41	64.95
Iterative	No	No	39.77	54.79	69.09
Pretrained	Yes	No	NA	28.75	75.80
Iterative	Yes	No	63.52	66.72	81.04
No	No	Thresholding	50.53	75.11	67.30
No	No	Mask RCNN	NA	4.13	NA
Pretrained	Yes	Thresholding	NA	80.42	87.59
Iterative	Yes	Thresholding	**65.80**	**81.08**	**88.92**

Table 7.2: Manual workload reduction (%) using different methods on wheat, car and ship dataset. (Higher is better)

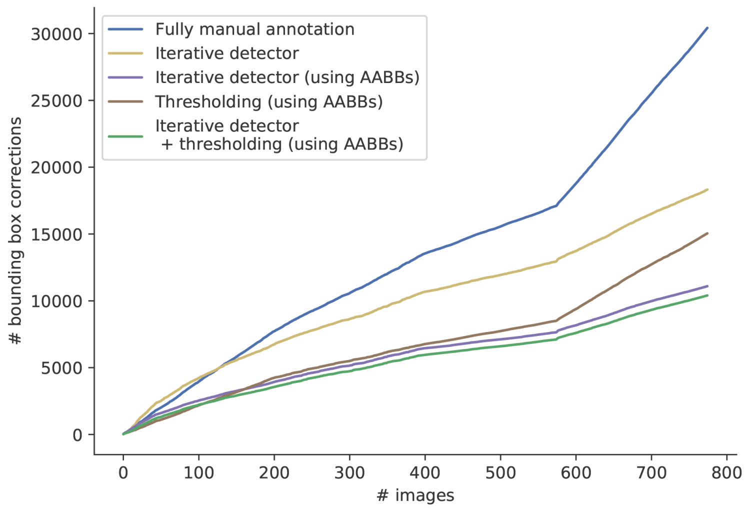


Figure 7.7: Required manual corrections using different methods for the wheat dataset.

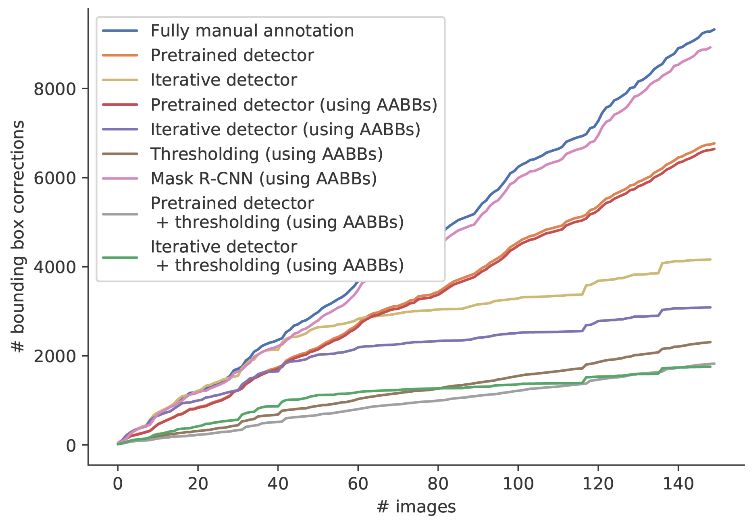


Figure 7.8: Required manual corrections using different methods for the car dataset.

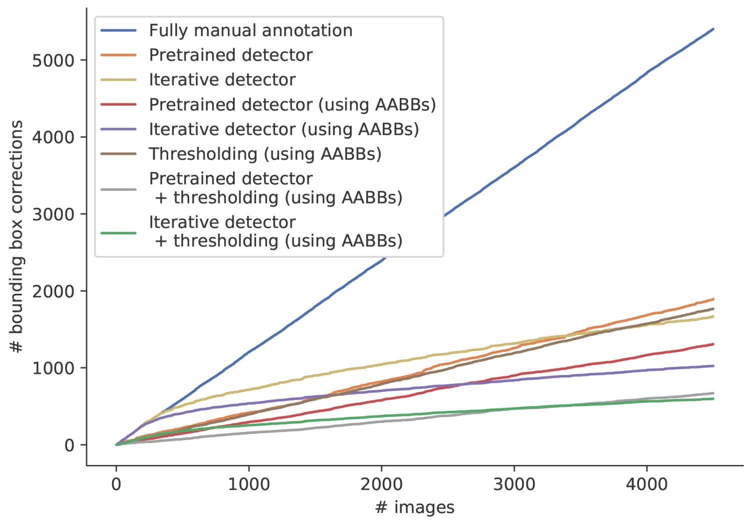


Figure 7.9: Required manual corrections using different methods for the ship dataset.

across all the datasets. Similarly, the pre-trained detector achieves a 10.9% rise in performance for the ship dataset. However, it acquires only a slight increment of close to 1% for the car dataset.

When the available AABBs are adapted with thresholding (in combination with post-processing steps), we reach a higher or similar percentage of workload reduction compared to the object detectors without filtering for all the datasets. The manual workload reduces by 50.5%, 75.1%, and 67.3% for the wheat, car and ship datasets respectively. The approach that involves adapting with Mask R-CNN only attains a 4.1% workload reduction for the car dataset and it's not applicable to the other datasets. Finally, The maximum workload reduction is achieved for any dataset when the iterative detector is employed in combination with the thresholding technique. This approach results in a manual workload reduction of 65.8% for the wheat dataset, 81.1% for the car dataset, and 88.9% for the ship dataset.

Figure 7.7 reveals the impact of different strategies to alleviate the workload throughout the annotation campaign for the wheat dataset. Without any RBB suggestions, the annotator would need to create a total of 30,420 new bounding boxes. But, with the proposals constructed by the iterative detector in absence of AABBs, the number of manual corrections is reduced to 18,321. In presence of AABBs, manual corrections are further reduced to 11,097 bounding boxes. However, it goes to as low as 10,401 giving us a workload reduction of 65.9% when the thresholding approach is applied to generate RBB proposals for those pre-existing AABBs for which the detector couldn't produce any proposals.

Figure 7.8 depicts that the required manual corrections can be as few as 1,756 bounding boxes for the car dataset while the total bounding boxes are 9,330. Lastly, we can reach the maximum workload reduction of 88.9% for the ship dataset as shown in the Figure 7.9 by using the combination of the iterative detector and thresholding approach. Only 598 boxes need to be corrected out of the 5,401 instances in this case.

7.2.4 Discussion

The iterative detector performs better than the pre-trained detector without filtering (i.e. no AABBs are available) because the iterative detector is incrementally trained with the labeled images of the current dataset which is being annotated. Therefore, the training images are similar to the test images and it helps the iterative detector to understand the content of the images even better. We see that both the pre-trained and the iterative detector obtain a higher percentage of workload reduction when a set of AABBs are available for the images of the datasets. This is because the given boxes can be used to filter out many of the false positives which reduces the number of box removal operations previously required.

The proposals that are generated by adapting the available AABBs with thresholding remove a significant number of manual corrections because many of the wheat, car, and ship instances align with the assumptions of the thresholding approach. In contrast, the Mask R-CNN model experienced a low workload reduction because the (street-view) car images it is trained on are very different from the drone-view images of the car dataset. When the thresholding technique is used as the backup proposal generation method with the pre-trained or iterative detector (with filtering), the manual workload reductions are maximized because the thresholding approach can potentially produce proposals for any missed objects by the detectors.

7.3 Summary

We evaluated both the speed and accuracy of the multipoint manual RBB annotation methods and the effectiveness of the RBB proposal generation methods used in FastRoLabelImg. The results demonstrated that the proposed multipoint drawing system is faster than the current RBB drawing system irrespective of the shape of the objects. The results also showed that the 4-point system maximizes the annotation accuracy while minimizing the annotation time. Furthermore, we described the evaluation procedure of the proposal generation methods and reported that the proposed methods can reach a maximum of 88.9% manual workload reduction under our experimental settings.

8 Conclusion and Future Work

In this thesis, we develop an object annotation tool named FastRoLabelImg to speed up the tedious task of RBB annotations by minimizing the number of manual annotations with automatic proposals and providing an efficient manual annotation mechanism at the same time. This chapter reviews the contributions of this thesis, discusses its limitations, and proposes directions for future work.

8.1 Contributions

We provide an all-in-one annotation environment that allows generating rotated bounding box annotations for large image datasets easily and quickly. The contributions of this work can be stated as follows:

1. We propose an easy-to-use multipoint drawing system that is designed to improve manual RBB annotation time and accuracy for objects of different shapes.

2. We observe with a participant study that the available configurations (3-point and 4-point) in the proposed system are faster for manual RBB annotations than the current existing approach. Besides, one of the configurations (4-point) fosters more accurate annotations with little effort.

3. We add an online learning based object detector to FastRoLabelImg that improves over time to generate RBB proposals. We also integrate a pre-trained detector and segmentation based proposal generation methods to support during the earlier iterations of the iterative detector as it is likely to produce inaccurate proposals at the start.

4. When pre-existing AABBs are available, the segmentation based approaches are used to adapt the AABB annotations to RBB proposals. Moreover, both the pre-trained and the iterative detectors can take advantage of the available AABB annotations by limiting the produced proposals only to the pre-existing AABBs based on IoU. FastRoLabelImg also gives the annotators the flexibility to tune the key parameters related to the proposal generation approaches to match with the dataset at hand.

5. The efficacy of the automatic RBB proposal generation methods is tested on three diverse datasets. Although the percentage of workload reduction depends on the size and nature of the dataset, it is evident from the experimental results that our proposal generation methods reduce the annotation time and effort remarkably. The number of manual corrections for the bounding boxes dropped by 65.8%, 81.1%, and 88.9% for the wheat, car, and ship datasets respectively with the help of automatic RBB proposals.

6. The RBB annotations for two of the datasets (wheat and car) are made publicly available to encourage future research in developing object detectors capable of RBB predictions.

8.2 Limitations and Future Work

We have integrated the proposed manual RBB annotation methods as well as the automatic RBB proposal generation methods into FastRoLabelImg. The developed methods are also evaluated with experiments and a participant study. However, the annotation tool/study has a number of limitations that could be addressed as future work.

Among the three datasets used in this thesis, the wheat dataset is built by sampling 776 images from the original GWHD dataset. Those images are annotated with RBB annotations for evaluating our developed methods and also made available to other researchers. The GWHD dataset has more than 4,500 images and we couldn't annotate all the images due to time constraints. However, we can now efficiently annotate the other images with the help of the proposed annotation tool and produce RBB annotations for the complete GWHD dataset. The new RBB annotations will help develop more accurate detection models capable of RBB predictions.

The proposed multipoint drawing system provides two main configurations for creating RBB annotations namely the 3-point and the 4-point systems. Both of these techniques have a unique requirement of position and order of the mouse clicks. It is possible to design many new configurations that the annotator will be able to choose from based on the properties of the objects they are dealing with. For example, the 4-point system can be modified to use three clicks only if the objects of interest have a symmetrical top-down view such as plane, car, etc. In this case, the annotator will need to click on two opposite extreme points and the extent of one of the other two sides as shown in Figure 8.1. Therefore, we can fine-tune new configurations to easily create RBB annotations for various categories of objects. Also, for both 3-point and 4-point systems, the first two points specified are used to decide the orientation of the RBBs and the annotators can't see the finalized RBBs until all the points are clicked. As a result, if the annotation tool can show a live preview of the bounding box when the mouse is being moved by an annotator to click the points, it may be useful to reduce the number of repeated attempts.

Some of the recent works [62, 105] proposed object detection architectures with bounding circles. The bounding circles are effective to detect objects such as orange, tomato, etc. in agricultural images, glomeruli in biomedical images, and others. As future work, we can adapt FastRoLabelImg for creating circle and ellipse representations of objects. To compare the annotation time and accuracy of the two available configurations of the multipoint drawing system to the current RBB drawing method, a participant study is conducted and interpreted using within-participants RM-ANOVA in this work. It is possible to use a set of common images across participants during the data collection phase and do intra- and inter-rater reliability tests for manual RBB annotations as future work.

The automatic proposal generation methods immensely reduce the amount of manual work. Currently, the pre-trained detector used in the process is fixed. But, in the future, we can enable annotators to import their own detector in FastRoLabelImg. It will allow annotators to use any existing models trained with a set of sample images of a custom dataset to annotate the remaining images with better proposals as well as add new images with annotations easily in

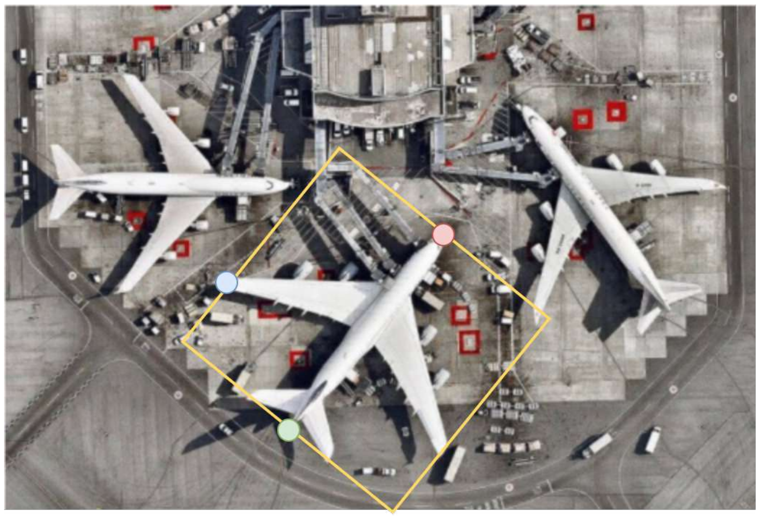


Figure 8.1: The modified 4-point configuration may require 3 clicks for symmetrical objects. The colored dots represent clicks around the object in the order: Red⇒ Green⇒Blue.

the future. Also, we can explore more advanced image processing based segmentation methods to help in creating RBB proposals instead of otsu's method of thresholding. It's also possible that a different deep learning based instance segmentation model (e.g. YOLACT [19], SOLO [101], etc.) may work better. For the online detector, the impact of having an accurate detector as quickly as possible is crucial as the magnitude of manual corrections will be reduced for the remaining images. One way to achieve this can be the use of active learning. With active learning, we can try to find an effective ordering of the images at each step to present the most informative images to the detector in the earlier iterations [41]. It might be possible to obtain a strong detector in fewer iterations in this way.

Once the automatic proposal generation step is requested and fulfilled for an image, the annotator is left with reviewing the RBB proposals by inspection and correct if necessary. However, inspecting object annotations in crowded images is itself a very hard task. So, it is also important to develop a better user interface for inspection. Lastly, the primary focus of this work is to establish the effectiveness of the proposed methods. Therefore, we follow the design pattern of LabelImg to integrate our developed methods in the interface. Further study can be conducted to explore optimal user interface design to operate these new methods for annotating images with RBBs. Providing the tool as open-source we hope that such refinement can be made by the LabelImg community as annotators start to perform RBB annotation as a routine practice.

References

[1] Mit license. https://en.wikipedia.org/wiki/MIT_License. (Accessed on 04/30/2021).

[2] Polyline annotation for lane detection in self-driving cars — by anolytics — medium. https://anolytics.medium.com/polyline-annotation-for-lane-detection-in-self-driving-cars-332db55067b3. (Accessed on 01/26/2021).

[3] Pricing of data labeling service — google cloud. https://cloud.google.com/ai-platform/data-labeling/pricing. (Accessed on 04/08/2021).

[4] dogcat1.jpg (1000×485). https://www.dogtagart.com/sites/default/files/styles/scale_width_1200_webp/public/blog/dogcat1.jpg. (Accessed on 05/03/2021).

[5] Fastannotationtool: A tool using opencv to annotate images for image classification, optical character reading, ..., 2015. URL https://github.com/christopher5106/FastAnnotationTool. (Accessed on 01/25/2021).

[6] Labelimg, 2016. URL https://github.com/tzutalin/labelImg. (Accessed on 09/18/2020).

[7] rolabelimg, 2017. URL https://github.com/cgvict/roLabelImg. (Accessed on 09/18/2020).

[8] Otsu's method - wikipedia. https://en.wikipedia.org/wiki/Otsu%27s_method, March 2018. (Accessed on 01/26/2021).

[9] Anno-mage: A semi automatic image annotation tool, 2018. URL https://github.com/virajmavani/semi-auto-image-annotation-tool. (Accessed on 09/18/2020).

[10] Airbus ship detection challenge, 2018. URL https://www.kaggle.com/c/airbus-ship-detection. (Accessed on 09/18/2020).

[11] Computer vision annotation tool (cvat), 2019. URL https://github.com/openvinotoolkit/cvat. (Accessed on 09/18/2020).

[12] Bishwo Adhikari and Heikki Huttunen. Iterative bounding box annotation for object detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4040–4046. IEEE, 2021.

[13] Bishwo Adhikari, Jukka Peltomaki, Jussi Puura, and Heikki Huttunen. Faster bounding box annotation for object detection in indoor scenes. In *2018 7th European Workshop on Visual Information Processing (EUVIP)*, pages 1–6. IEEE, 2018.

[14] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[15] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. In *European conference on computer vision*, pages 549–565. Springer, 2016.

[16] Alex Berg, Jia Deng, and L Fei-Fei. Large scale visual recognition challenge (ilsvrc), 2010. *URL http://www.image-net. org/challenges/LSVRC*, 3, 2010.

[17] Hakan Bilen and Andrea Vedaldi. Weakly supervised deep detection networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2846–2854, 2016.

[18] Arijit Biswas and Devi Parikh. Simultaneous active learning of classifiers & attributes via relative feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 644–651, 2013.

[19] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9157–9166, 2019.

[20] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *Proceedings eighth IEEE international conference on computer vision. ICCV 2001*, volume 1, pages 105–112. IEEE, 2001.

[21] Lluis Castrejon, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5230–5238, 2017.

[22] George Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[23] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[24] Datalya. Machine learning vs. traditional programming paradigm — data science blog. `https://datalya.com/blog/machine-learning/machine-learning-vs-traditional-programming-paradigm`. (Accessed on 04/22/2021).

[25] Etienne David, Simon Madec, Pouria Sadeghi-Tehran, Helge Aasen, Bangyou Zheng, Shouyang Liu, Norbert Kirchgessner, Goro Ishikawa, Koichi Nagasawa, Minhajul A. Badhon, Curtis Pozniak, Benoit de Solan, Andreas Hund, Scott C. Chapman, Frédéric Baret, Ian Stavness, and Wei Guo. Global Wheat Head Detection (GWHD) Dataset: A Large and Diverse Dataset of High-Resolution RGB-Labelled Images to Develop and Benchmark Wheat Head Detection Methods. *Plant Phenomics*, 2020, Aug 2020. doi: 10.34133/2020/3521852.

[26] Maaike HT de Boer, Henri Bouma, Maarten Kruithof, and Bart Joosten. Rapid annotation tool to train novel concept detectors with active learning. In *MMEDIA 2019: International Conference on Advances in Multimedia, Valencia 24-28 march 2019*, 2019.

[27] Thomas Deselaers, Bogdan Alexe, and Vittorio Ferrari. Localizing objects while learning their appearance. In *European conference on computer vision*, pages 452–466. Springer, 2010.

[28] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning roi transformer for oriented object detection in aerial images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2849–2858, 2019.

[29] Daxiang Dong, Hua Wu, Wei He, Dianhai Yu, and Haifeng Wang. Multi-task learning for multiple language translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1723–1732, 2015.

[30] PN Druzhkov and VD Kustikova. A survey of deep learning methods and software tools for image classification and object detection. *Pattern Recognition and Image Analysis*, 26(1):9–15, 2016.

[31] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint arXiv:1603.07285*, 2016.

[32] Suyog Dutt Jain and Kristen Grauman. Predicting sufficient annotation strength for interactive foreground segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1313–1320, 2013.

[33] Abhishek Dutta, Ankush Gupta, and Andrew Zissermann. Vgg image annotator (via). *URL: http://www. robots. ox. ac. uk/˜ vgg/software/via*, 2016.

[34] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1): 98–136, 2015.

[35] Max Ferguson, Ronay Ak, Yung-Tsun Tina Lee, and Kincho H Law. Automatic localization of casting defects with convolutional neural networks. In *2017 IEEE international conference on big data (big data)*, pages 1726–1735. IEEE, 2017.

[36] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[37] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[38] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 38 (1):142–158, 2015.

[39] Ramazan Gokberk Cinbis, Jakob Verbeek, and Cordelia Schmid. Multi-fold mil training for weakly supervised object localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2409–2416, 2014.

[40] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[41] Elmar Haussmann, Michele Fenzi, Kashyap Chitta, Jan Ivanecky, Hanson Xu, Donna Roy, Akshita Mittel, Nicolas Koumchatzky, Clement Farabet, and Jose M Alvarez. Scalable active learning for object detection. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1430–1435. IEEE, 2020.

[42] Manuel Haußmann, Fred A Hamprecht, and Melih Kandemir. Variational bayesian multiple instance learning with gaussian processes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6570–6579, 2017.

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

[44] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[45] Robert Hecht-Nielsen. Theory of the backpropagation neural network. In *Neural networks for perception*, pages 65–93. Elsevier, 1992.

[46] Geoffrey Hinton. Deep learning—a technology with the potential to transform health care. *Jama*, 320(11): 1101–1102, 2018.

[47] Meng-Ru Hsieh, Yen-Liang Lin, and Winston H Hsu. Drone-based object counting by spatially regularized regional proposal network. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4145–4153, 2017.

[48] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *arXiv preprint arXiv:1506.02025*, 2015.

[49] Suyog Dutt Jain and Kristen Grauman. Click carving: Segmenting objects in video with point clicks. *arXiv preprint arXiv:1607.01115*, 2016.

[50] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379. IEEE, 2009.

[51] Vadim Kantorov, Maxime Oquab, Minsu Cho, and Ivan Laptev. Contextlocnet: Context-aware deep network models for weakly supervised localization. In *European Conference on Computer Vision*, pages 350–365. Springer, 2016.

[52] Ujjwal Karn. An intuitive explanation of convolutional neural networks. `https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`, August 2016. (Accessed on 01/26/2021).

[53] Mehran Khodabandeh, Arash Vahdat, Guang-Tong Zhou, Hossein Hajimirsadeghi, Mehrsan Javan Roshtkhari, Greg Mori, and Stephen Se. Discovering human interactions in videos with limited data labeling. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 9–18, 2015.

[54] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[55] Ksenia Konyushkova, Jasper Uijlings, Christoph H Lampert, and Vittorio Ferrari. Learning intelligent dialogs for bounding box annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9175–9184, 2018.

[56] Adriana Kovashka, Sudheendra Vijayanarasimhan, and Kristen Grauman. Actively selecting annotations among objects and attributes. In *2011 International Conference on Computer Vision*, pages 1403–1410. IEEE, 2011.

[57] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[58] Feifei Li. Cs231n convolutional neural networks for visual recognition. `https://cs231n.github.io/convolutional-networks`. (Accessed on 05/03/2021).

[59] Feifei Li. Cs231n convolutional neural networks for visual recognition. `https://cs231n.github.io/neural-networks-1/`, May 2017. (Accessed on 01/26/2021).

[60] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[61] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[62] Guoxu Liu, Joseph Christian Nouaze, Philippe Lyonel Touko Mbouembe, and Jae Ho Kim. Yolo-tomato: A robust algorithm for tomato detection based on yolov3. *Sensors*, 20(7):2145, 2020.

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

[64] Wenchao Liu, Long Ma, and He Chen. Arbitrary-oriented ship detection framework in optical remote-sensing images. *IEEE Geoscience and Remote Sensing Letters*, 15(6):937–941, 2018.

[65] Zikun Liu, Jingao Hu, Lubin Weng, and Yiping Yang. Rotated region based cnn for ship detection. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 900–904. IEEE, 2017.

[66] David G Lowe. Object recognition from local scale-invariant features. In *Proceedings of the seventh IEEE international conference on computer vision*, volume 2, pages 1150–1157. Ieee, 1999.

[67] Jianqi Ma, Weiyuan Shao, Hao Ye, Li Wang, Hong Wang, Yingbin Zheng, and Xiangyang Xue. Arbitrary-oriented scene text detection via rotation proposals. *IEEE Transactions on Multimedia*, 20(11):3111–3122, 2018.

[68] Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pages 109–165. Elsevier, 1989.

[69] James F Mullen Jr, Franklin R Tanner, and Phil A Sallee. Comparing the effects of annotation type on machine learning detection performance. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.

[70] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

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

[72] Jayashree Padmanabhan and Melvin Jose Johnson Premkumar. Machine learning in automatic speech recognition: A survey. *IETE Technical Review*, 32(4):240–251, 2015.

[73] Dim P Papadopoulos, Alasdair DF Clarke, Frank Keller, and Vittorio Ferrari. Training object class detectors from eye tracking data. In *European conference on computer vision*, pages 361–376. Springer, 2014.

[74] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 854–863, 2016.

[75] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *Proceedings of the IEEE international conference on computer vision*, pages 4930–4939, 2017.

[76] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6374–6383, 2017.

[77] Devi Parikh and Kristen Grauman. Relative attributes. In *2011 International Conference on Computer Vision*, pages 503–510. IEEE, 2011.

[78] Amar Parkash and Devi Parikh. Attributes for classifier feedback. In *European Conference on Computer Vision*, pages 354–368. Springer, 2012.

[79] Qing Rao and Jelena Frtunikj. Deep learning for self-driving cars: Chances and challenges. In *Proceedings of the 1st International Workshop on Software Engineering for AI in Autonomous Systems*, pages 35–38, 2018.

[80] Javier Ray. Faster r-cnn: Down the rabbit hole of modern object detection. `https://tryolabs.com/blog/2018/01/18/faster-r-cnn-down-the-rabbit-hole-of-modern-object-detection/`, January 2018. (Accessed on 01/26/2021).

[81] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[82] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(6):1137–1149, 2016.

[83] Lourena Rocha, Luiz Velho, and Paulo Cezar Pinto Carvalho. Image moments-based structuring and tracking of objects. In *Proceedings. XV Brazilian Symposium on Computer Graphics and Image Processing*, pages 99–105. IEEE, 2002.

[84] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *ACM transactions on graphics (TOG)*, 23(3):309–314, 2004.

[85] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[86] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3):211–252, 2015.

[87] Burr Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[88] Mohd Razif Shamsuddin, Shuzlina Abdul-Rahman, and Azlinah Mohamed. Exploratory analysis of mnist handwritten digit for machine learning modelling. In *International Conference on Soft Computing in Data Science*, pages 134–145. Springer, 2018.

[89] Naveen Shankar Nagaraja, Frank R Schmidt, and Thomas Brox. Video segmentation with just a few strokes. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3235–3243, 2015.

[90] Behjat Siddiquie and Abhinav Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2979–2986. IEEE, 2010.

[91] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[92] Tianyu Tang, Shilin Zhou, Zhipeng Deng, Lin Lei, and Huanxin Zou. Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks. *Remote Sensing*, 9(11):1170, 2017.

[93] Maxim Tkachenko, Mikhail Malyuk, Nikita Shevchenko, Andrey Holmanyuk, and Nikolai Liubimov. Label Studio: Data labeling software, 2020. URL `https://github.com/heartexlabs/label-studio`. Open source software available from https://github.com/heartexlabs/label-studio.

[94] Hessel Tuinhof, Clemens Pirker, and Markus Haltmeier. Image-based fashion product recommendation with deep learning. In *International Conference on Machine Learning, Optimization, and Data Science*, pages 472–481. Springer, 2018.

[95] Sudheendra Vijayanarasimhan and Kristen Grauman. Multi-level active prediction of useful image annotations for recognition. In *Advances in Neural Information Processing Systems*, pages 1705–1712, 2009.

[96] Sudheendra Vijayanarasimhan and Kristen Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2262–2269. IEEE, 2009.

[97] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. In *CVPR 2011*, pages 1449–1456, 2011. doi: 10.1109/CVPR.2011.5995430.

[98] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International journal of computer vision*, 101(1):184–204, 2013.

[99] Kentaro Wada. labelme: Image Polygonal Annotation with Python. `https://github.com/wkentaro/labelme`, 2016.

[100] Tinghuai Wang, Bo Han, and John Collomosse. Touchcut: Fast image and video segmentation using single-touch interaction. *Computer Vision and Image Understanding*, 120:14–30, 2014.

[101] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In *European Conference on Computer Vision*, pages 649–665. Springer, 2020.

[102] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983, 2018.

[103] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[104] Feng Yang, Wentong Li, Haiwei Hu, Wanyi Li, and Peng Wang. Multi-scale feature integrated attention-based rotation network for object detection in vhr aerial images. *Sensors*, 20(6):1686, 2020.

[105] Haichun Yang, Ruining Deng, Yuzhe Lu, Zheyu Zhu, Ye Chen, Joseph T Roland, Le Lu, Bennett A Landman, Agnes B Fogo, and Yuankai Huo. Circlenet: Anchor-free detection with circle representation. *arXiv preprint arXiv:2006.02474*, 2020.

[106] Xue Yang, Hao Sun, Kun Fu, Jirui Yang, Xian Sun, Menglong Yan, and Zhi Guo. Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks. *Remote Sensing*, 10(1):132, 2018.

[107] Xue Yang, Hao Sun, Xian Sun, Menglong Yan, Zhi Guo, and Kun Fu. Position detection and direction prediction for arbitrary-oriented ships via multitask rotation region convolutional neural network. *IEEE Access*, 6:50839–50849, 2018.

[108] Xue Yang, Qingqing Liu, Junchi Yan, Ang Li, Zhiqiang Zhang, and Gang Yu. R3det: Refined single-stage detector with feature refinement for rotating object. *arXiv preprint arXiv:1908.05612*, 2019.

[109] Xue Yang, Jirui Yang, Junchi Yan, Yue Zhang, Tengfei Zhang, Zhi Guo, Xian Sun, and Kun Fu. Scrdet: Towards more robust detection for small, cluttered and rotated objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8232–8241, 2019.

[110] Xue Yang, Junchi Yan, Xiaokang Yang, Jin Tang, Wenlong Liao, and Tao He. Scrdet++: Detecting small, cluttered and rotated objects via instance-level feature denoising and rotation loss smoothing. *arXiv preprint arXiv:2004.13316*, 2020.

[111] Angela Yao, Juergen Gall, Christian Leistner, and Luc Van Gool. Interactive object detection. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3242–3249. IEEE, 2012.

[112] Zenghui Zhang, Weiwei Guo, Shengnan Zhu, and Wenxian Yu. Toward arbitrary-oriented ship detection with rotated region proposal and discrimination networks. *IEEE Geoscience and Remote Sensing Letters*, 15(11): 1745–1749, 2018.

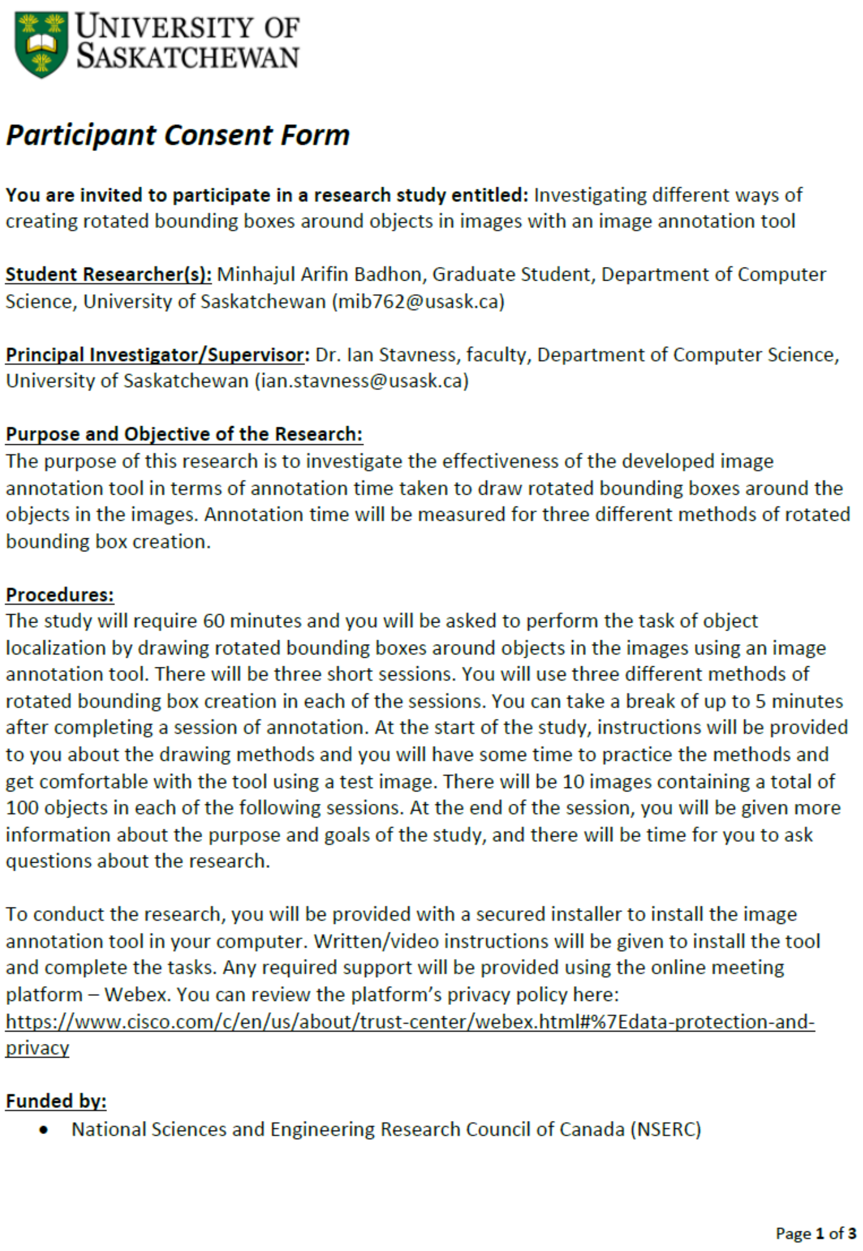
[113] Yi Zhu, Yanzhao Zhou, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Soft proposal networks for weakly supervised object localization. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1841–1850, 2017.

[114] Yixing Zhu, Chixiang Ma, and Jun Du. Rotated cascade r-cnn: A shape robust detector with coordinate regression. *Pattern Recognition*, 96:106964, 2019.

[115] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.

Appendix A

Participant Study - Consent Form



UNIVERSITY OF SASKATCHEWAN

Participant Consent Form

You are invited to participate in a research study entitled: Investigating different ways of creating rotated bounding boxes around objects in images with an image annotation tool

Student Researcher(s): Minhajul Arifin Badhon, Graduate Student, Department of Computer Science, University of Saskatchewan (mib762@usask.ca)

Principal Investigator/Supervisor: Dr. Ian Stavness, faculty, Department of Computer Science, University of Saskatchewan (ian.stavness@usask.ca)

Purpose and Objective of the Research:
The purpose of this research is to investigate the effectiveness of the developed image annotation tool in terms of annotation time taken to draw rotated bounding boxes around the objects in the images. Annotation time will be measured for three different methods of rotated bounding box creation.

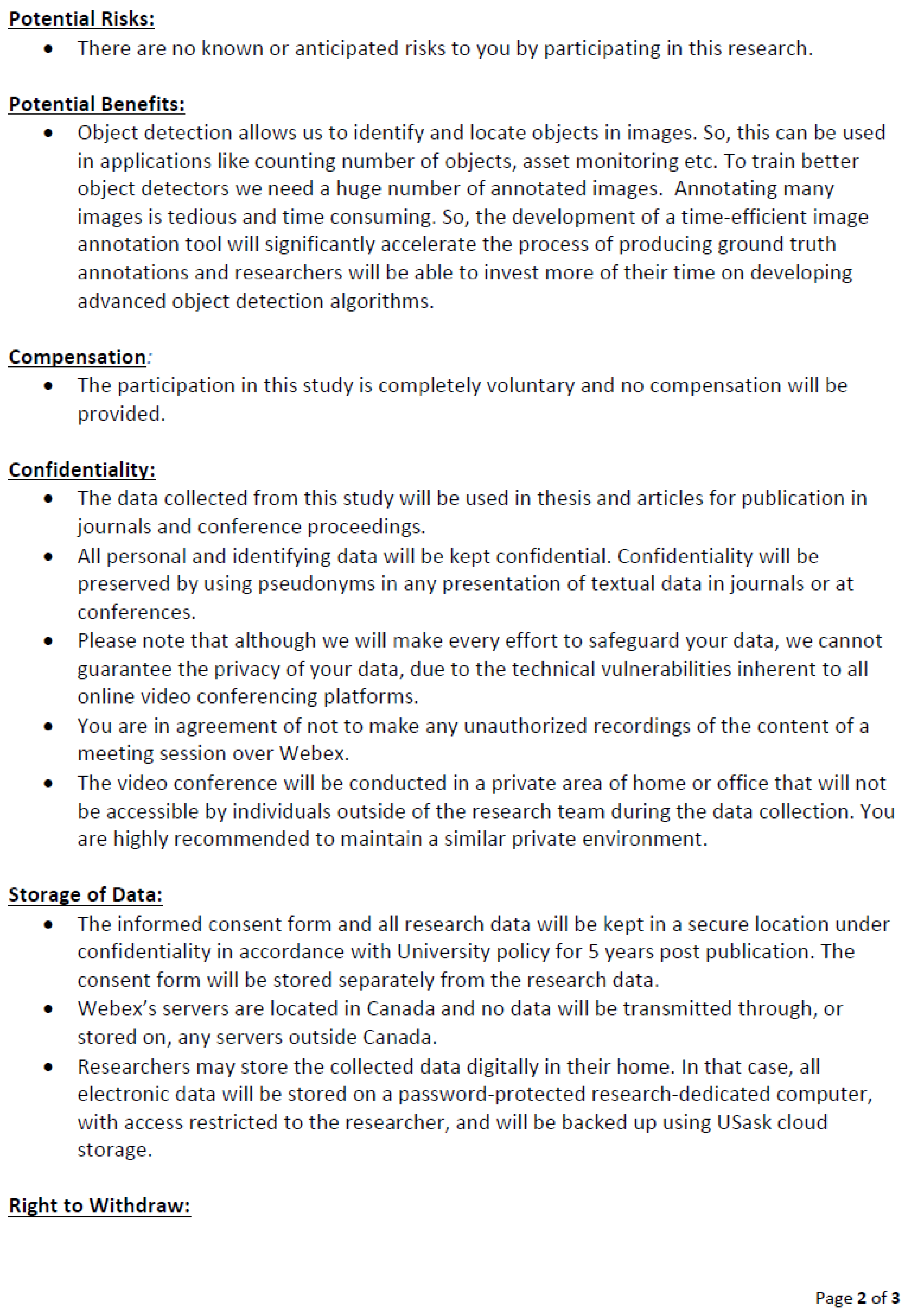
Procedures:
The study will require 60 minutes and you will be asked to perform the task of object localization by drawing rotated bounding boxes around objects in the images using an image annotation tool. There will be three short sessions. You will use three different methods of rotated bounding box creation in each of the sessions. You can take a break of up to 5 minutes after completing a session of annotation. At the start of the study, instructions will be provided to you about the drawing methods and you will have some time to practice the methods and get comfortable with the tool using a test image. There will be 10 images containing a total of 100 objects in each of the following sessions. At the end of the session, you will be given more information about the purpose and goals of the study, and there will be time for you to ask questions about the research.

To conduct the research, you will be provided with a secured installer to install the image annotation tool in your computer. Written/video instructions will be given to install the tool and complete the tasks. Any required support will be provided using the online meeting platform – Webex. You can review the platform's privacy policy here: https://www.cisco.com/c/en/us/about/trust-center/webex.html#%7Edata-protection-and-privacy

Funded by:
- National Sciences and Engineering Research Council of Canada (NSERC)

Page **1** of **3**

Figure A.1: Participant consent form (page 1).



Potential Risks:

- There are no known or anticipated risks to you by participating in this research.

Potential Benefits:

- Object detection allows us to identify and locate objects in images. So, this can be used in applications like counting number of objects, asset monitoring etc. To train better object detectors we need a huge number of annotated images. Annotating many images is tedious and time consuming. So, the development of a time-efficient image annotation tool will significantly accelerate the process of producing ground truth annotations and researchers will be able to invest more of their time on developing advanced object detection algorithms.

Compensation:

- The participation in this study is completely voluntary and no compensation will be provided.

Confidentiality:

- The data collected from this study will be used in thesis and articles for publication in journals and conference proceedings.
- All personal and identifying data will be kept confidential. Confidentiality will be preserved by using pseudonyms in any presentation of textual data in journals or at conferences.
- Please note that although we will make every effort to safeguard your data, we cannot guarantee the privacy of your data, due to the technical vulnerabilities inherent to all online video conferencing platforms.
- You are in agreement of not to make any unauthorized recordings of the content of a meeting session over Webex.
- The video conference will be conducted in a private area of home or office that will not be accessible by individuals outside of the research team during the data collection. You are highly recommended to maintain a similar private environment.

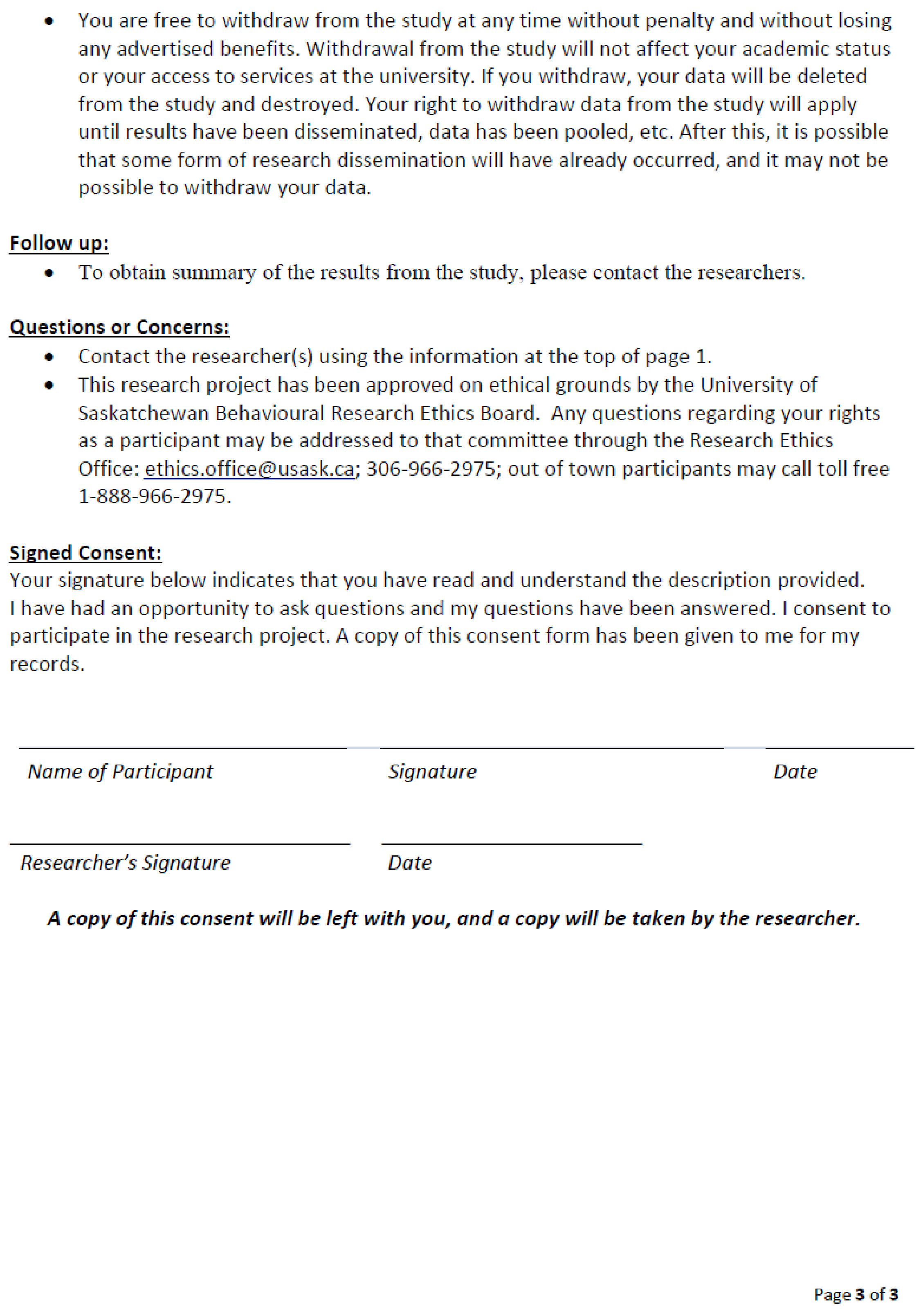
Storage of Data:

- The informed consent form and all research data will be kept in a secure location under confidentiality in accordance with University policy for 5 years post publication. The consent form will be stored separately from the research data.
- Webex's servers are located in Canada and no data will be transmitted through, or stored on, any servers outside Canada.
- Researchers may store the collected data digitally in their home. In that case, all electronic data will be stored on a password-protected research-dedicated computer, with access restricted to the researcher, and will be backed up using USask cloud storage.

Right to Withdraw:

Page **2** of **3**

Figure A.2: Participant consent form (page 2).



- You are free to withdraw from the study at any time without penalty and without losing any advertised benefits. Withdrawal from the study will not affect your academic status or your access to services at the university. If you withdraw, your data will be deleted from the study and destroyed. Your right to withdraw data from the study will apply until results have been disseminated, data has been pooled, etc. After this, it is possible that some form of research dissemination will have already occurred, and it may not be possible to withdraw your data.

Follow up:

- To obtain summary of the results from the study, please contact the researchers.

Questions or Concerns:

- Contact the researcher(s) using the information at the top of page 1.
- This research project has been approved on ethical grounds by the University of Saskatchewan Behavioural Research Ethics Board. Any questions regarding your rights as a participant may be addressed to that committee through the Research Ethics Office: ethics.office@usask.ca; 306-966-2975; out of town participants may call toll free 1-888-966-2975.

Signed Consent:

Your signature below indicates that you have read and understand the description provided. I have had an opportunity to ask questions and my questions have been answered. I consent to participate in the research project. A copy of this consent form has been given to me for my records.

Name of Participant	*Signature*	*Date*
Researcher's Signature	*Date*	

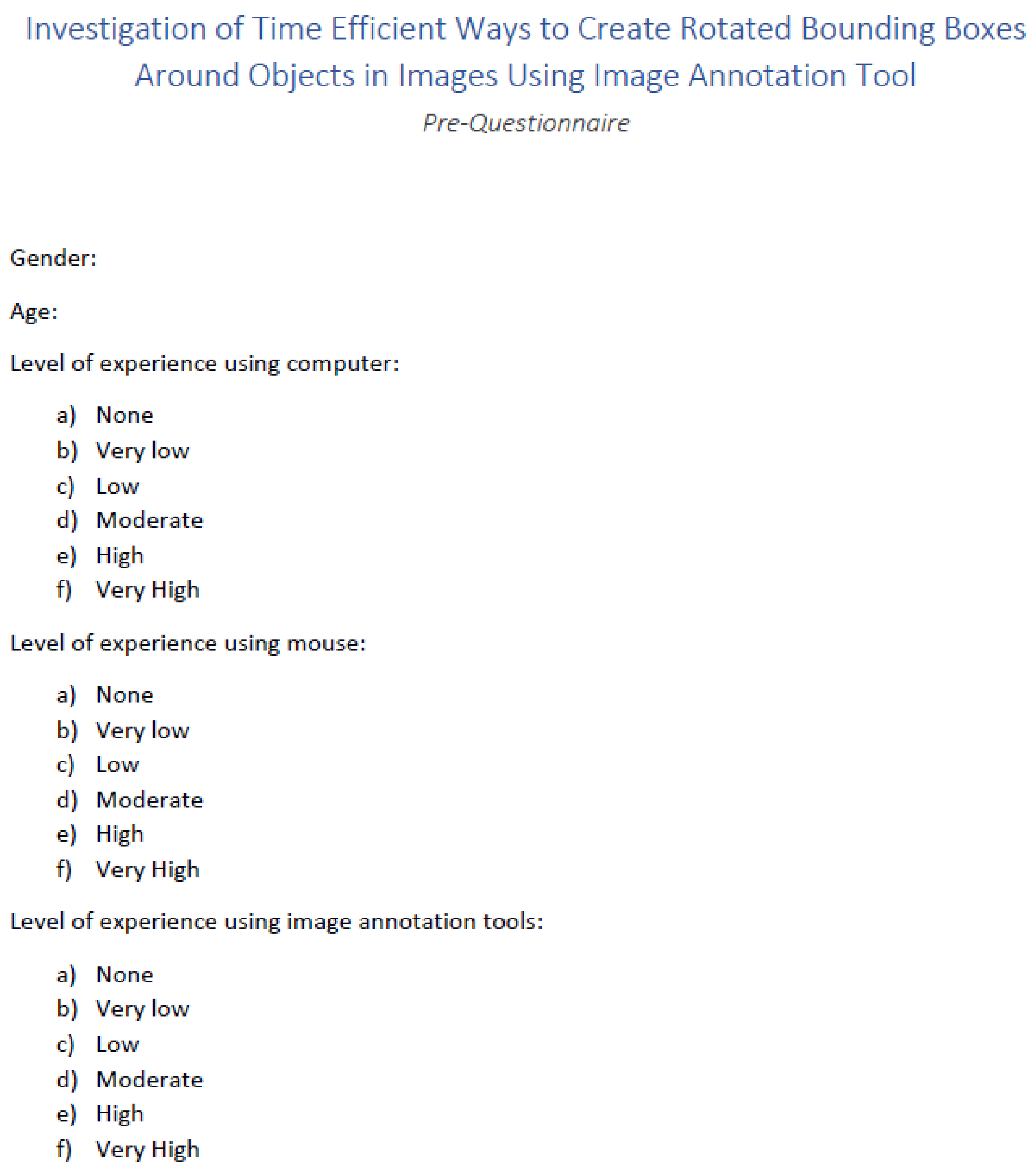
A copy of this consent will be left with you, and a copy will be taken by the researcher.

Page **3** of **3**

Figure A.3: Participant consent form (page 3).

Appendix B

Participant Study - Questionnaires



Investigation of Time Efficient Ways to Create Rotated Bounding Boxes Around Objects in Images Using Image Annotation Tool

Pre-Questionnaire

Gender:

Age:

Level of experience using computer:

a) None
b) Very low
c) Low
d) Moderate
e) High
f) Very High

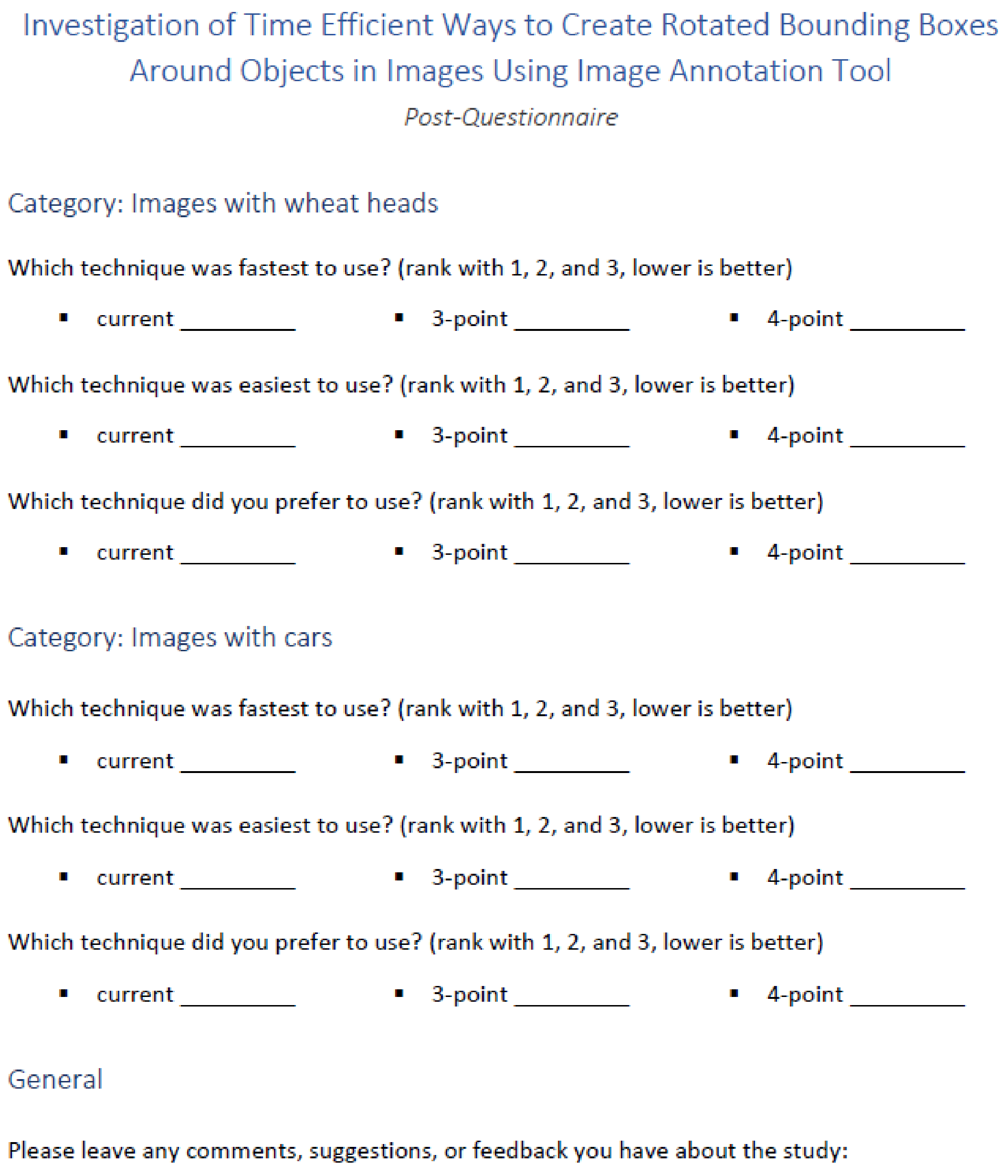
Level of experience using mouse:

a) None
b) Very low
c) Low
d) Moderate
e) High
f) Very High

Level of experience using image annotation tools:

a) None
b) Very low
c) Low
d) Moderate
e) High
f) Very High

Figure B.1: Pre-questionnaire for the study.



Investigation of Time Efficient Ways to Create Rotated Bounding Boxes Around Objects in Images Using Image Annotation Tool

Post-Questionnaire

Category: Images with wheat heads

Which technique was fastest to use? (rank with 1, 2, and 3, lower is better)

- current ________ ▪ 3-point ________ ▪ 4-point ________

Which technique was easiest to use? (rank with 1, 2, and 3, lower is better)

- current ________ ▪ 3-point ________ ▪ 4-point ________

Which technique did you prefer to use? (rank with 1, 2, and 3, lower is better)

- current ________ ▪ 3-point ________ ▪ 4-point ________

Category: Images with cars

Which technique was fastest to use? (rank with 1, 2, and 3, lower is better)

- current ________ ▪ 3-point ________ ▪ 4-point ________

Which technique was easiest to use? (rank with 1, 2, and 3, lower is better)

- current ________ ▪ 3-point ________ ▪ 4-point ________

Which technique did you prefer to use? (rank with 1, 2, and 3, lower is better)

- current ________ ▪ 3-point ________ ▪ 4-point ________

General

Please leave any comments, suggestions, or feedback you have about the study:

Figure B.2: Post-questionnaire for the study.