DIGITAL DETECTION OF LABORATORIES INSIDE CAMPUS

A PROJECT REPORT

Submitted by

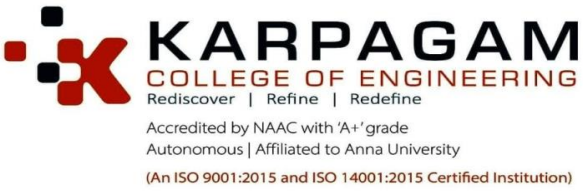
NAVEEN A	**19F136**
PRABHA A	**19F138**
PRIYA S	**19F141**
SUBIKSHA N	**19F152**

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY



ANNA UNIVERSITY, CHENNAI

APRIL 2023

KARPAGAM COLLEGE OF ENGINEERING

(AUTONOMOUS)

COIMBATORE – 641 032

DIGITAL DETECTION OF LABORATORIES INSIDE CAMPUS

Bonafide record of work done by

NAVEEN A	**19F136**
PRABHA A	**19F138**
PRIYA S	**19F141**
SUBIKSHA N	**19F152**

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

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

Of Anna University, Chennai

April 2023

...……	……..…….
Ms. B.P. SREEJA M.Tech.(Ph.D)	**Dr. P.MURUGESWARI M.Tech.,Ph.D.**
Faculty guide	Head of the Department

Certified that the candidate was examined in the viva-voce examination held on

…….

……..	……..
(Internal Examiner)	(External Examiner)

ACKNOWLEDGEMENT

We express our sincere thanks to Karpagam educational and charitable trust for providing necessary facilities to bring out the project successfully. We felt great to record our thanks to the chairman Dr. R. VASANTHAKUMAR, B.E.,(Hons),D.Sc. for all his support and ray of strengthening hope extended.

It is a moment of immense pride for us to reveal our profound thanks to our respected principal, Dr. P. KARTHIGAIKUMAR, Ph.D. who happens to be a striving force in all our endeavours.

We express our sincere thanks to Dr. P. MURUGESWARI, M.Tech., Ph.D. Head of the Department of Information Technology for providing an opportunity to work on this project. Her valuable suggestions helped us a lot to do this project.

A word of thanks would not be sufficient for the work of our project guide Ms.B.P.SREEJA, M.Tech., (Ph.D), Department of Information Technology whose efforts and inspiration lead us through every trying circumstance.

We would also like to recollect the courage and enthusiasm that was inculcated in us by our project co-ordinators Dr. S. SARAVANAN, M.E, Ph.D., & Ms. A. SUGANYA, M.E.,(Ph.D)., Department of Information Technology for valuable guidance and support through the tenure of our project.

We sincerely express our gratitude to all the faculty members of the Department of Information Technology for the encouragement we received throughout the semester.

ABSTRACT

The abundant spatial and contextual information provided by the advanced remote sensing technology has facilitated subsequent automatic interpretation of the optical remote sensing images. In this project, a novel and effective geospatial object detection framework is proposed by combining weakly supervised learning and high-level feature learning. First, the deep Boltzmann machine is adopted to infer the spatial and structural information encoded in the low-level and middle-level features to effectively describe objects in optical RSIs. Then, a novel WSL approach is presented to object detection where the training sets require only binary labels indicating whether an image contains the target object or not. Based on the learned high-level features, it jointly integrates saliency, intra class compactness, and interclass separability in a Bayesian framework to initialize a set of training examples from weakly labelled images and start iterative learning of the object detector. A novel evaluation criterion is also developed to detect model drift and cease the iterative learning. Comprehensive experiments on three optical RSI data sets have demonstrated the efficacy of the proposed approach in benchmarking with several state-of-the-art supervised-learning-based object detection approaches. The enhanced remote sensing technology's wealth of spatial and contextual data made it easier to automatically understand the optical remote sensing images. This project proposes a unique and efficient weakly supervised learning and high-level feature learning framework for geographic item recognition. To accurately represent objects in optical RSIs, the deep Boltzmann machine is first used to infer the spatial and structural information encoded in the low-level and middle-level features. Then an innovative WSL method for object detection is provided, where the training sets just need binary labels to indicate whether a picture contains the target object or not saliency, intra class compactness, and interclass are collaboratively integrated in a Bayesian framework based on the acquired high-level characteristics to initialise a set of training examples from weakly labelled pictures and begin iterative learning of the object detector. In order to stop iterative learning and identify model drift, a novel assessment criterion is also devised. Extensive tests using three optical RSI data sets have shown the effectiveness of the suggested strategy in comparison to a number of cutting-edge supervised-learning-based object recognition methods.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO
	ABSTRACT	**iv**
	LIST OF FIGURES	**vii**
	LIST OF ABBREVIATION	**viii**
1	**INTRODUCTION**	**01**
	1.1 INTRODUCTION TO DOMAIN	03
	1.2 MACHINE LEARNING	04
	1.3 OBJECT DETECTION	05
2	**LITERATURE SURVEY**	**07**
3	**SYSTEM OVERVIEW**	**10**
	3.1 RELATED WORK	12
	3.2 EXISTING WORK	12
	3.2.1 HIGH-LEVEL FEATURE REPRESENTATION	12
	3.2.1.1 LOW-LEVEL DESCRIPTOR EXTRACTION	13
	3.2.1.2 MIDDLE-LEVEL FEATURE GENERATION	13
	3.2.1.3 HIGH-LEVEL FEATURE LEARNING	14
	3.2.2 WEB BASED OBJECT DETECTION	15
	3.2.2.1 SALIENCY	17
	3.2.2.2 INTRA CLASS COMPACTNESS	18
	3.2.2.3 INTRACLASS SEPARABILITY	18
	3.2.2.4 PRIOR PROBABILITY	19
	3.2.2.5 IMPLEMENTATION DETAILS	20
	3.3 PROPOSED METHODOLOGY	21
4	**SYSTEM DESIGN AND WORKING**	**24**
	4.1 SYSTEM ARCHITECTURE	24
	4.2 SYSTEM REQUIREMENTS	24

	4.2.1 SOFTWARE REQUIREMENTS	24
	4.2.2 HARDWARE REQUIREMENTS	25
	4.3 MODULE DESCRIPTION	25
	4.3.1 PROBLEM STATEMENT	25
	4.3.2 ALGORITHM FORMULATION	25
	4.3.3 IMPLEMENTATION PHASE	25
	4.3.4 REALTIME INTERACTIVE KIT	25
5	**IMPLEMENTATION**	**26**
	5.1 SOURCE CODE	26
	5.2 RESULT	57
6	**CONCLUSION**	**59**
7	**FUTURE SCOPE**	**60**
	REFERENCES	**61**

LIST OF FIGURES

FIGURE	DESCRIPTION	PAGE NO.
1.1	Artificial Intelligence	03
1.2	Object Detection	06
3.1	Flowchart of WSL-based object detection	10
3.2	Iterative training of airport detector	12
3.3	Learning Processes for DBM	15
3.4	Illustration of saliency calculation	17
3.5	Evaluation of the proposed Bayesian framework	22
3.6	High-level feature representation of the image patch	23
4.1	System Architecture	24
5.1	Examples of laboratory detection	58

LIST OF ABBREVIATIONS

S.NO.	ACRONYMS	ABBREVIATIONS
1.	AI	Artificial Intelligence
2.	RSI	Remote Sensing Images
3.	ML	Machine Learning
4.	YOLO	You Only Look Once
5.	RCNN	Region-Based Convolutional Neutral Networks
6.	DBM	Deep Boltzmann Machine
7.	SIFT	Scale-Invariant Feature Transform
8.	DL	Detector Learning

CHAPTER 1
INTRODUCTION

The rapid development of remote sensing technologies has rendered many satellite and aerial sensors to provide optical imagery with high spatial resolution, facilitating a wide range of applications such as disaster control, land planning, urban monitoring, and traffic planning. In these applications, the automatic detection of natural or man-made objects is a fundamental task and has received increasing research interest. Early attempts detected objects in optical remote sensing images in an unsupervised manner, which often started from generating regions of interest by grouping pixels into clusters and then detected objects of interest based on the shape and spectral information. Afterward, many supervised learning methods have been adopted to learn the object model effectively with the help of prior information obtained from training examples. By heavily relying on the human-labeled training examples, which are statistically representative of the classification problem to solve, the supervised learning methods can achieve more promising performance than the unsupervised approaches.

The second problem is that the rich information contained in the optical RSIs with the high spatial resolution has more details of objects, whereas feature descriptors used byexisting object detectors are still insufficiently powerful to characterize the structural information of the objects. The limited understanding of the spatial and structural patterns of objects in optical RSIs leads to a tremendous semantic gap in the object detection task. It can be observed that man-made facilities, such as airplanes, vehicles, and airports, always have intrinsic structural properties with specific semantic concepts, which has an obvious difference from the background areas in optical RSIs. Consequently, building high-level structural features is a promising way for the object detection task. Tackle the manual annotation problem for object detection in optical RSIs by proposing a weakly supervised learning framework. As one of the most cost effective learning approaches, WSL only requires a weak label for the training images to specify whether the image contains the object of interest or not. To this end, unlike conventional supervised learning approaches, which rely on manually labeled bounding boxes for training object detector, accurate locations and sizes of the target objects are not needed in the WSL framework. Object detection using WSL tends to solve localization of the objects of interest in each positive training image and object detector training using automatic annotations simultaneously. In practice, WSL is implemented as follows.

Given the weak label only indicating whether a certain category of object is contained in an image or not, an initial annotation is first obtained automatically, based on which, a detector is trained. The trained detector is then used as the annotator to refine the annotation, whereas the detector is iteratively trained using refined annotations until the model drift is detected. In this project, propose a Bayesian framework by jointly exploring saliency, intra class compactness, and interclass separability to initialize a training example set. Afterward, it was propose a novel detector evaluation method, which is able to cease the iterative learning process when the detector starts to drift to bad results, and thus, it is obtain final object detector with satisfactory performance. To tackle the problem of insufficiently powerful feature descriptors, then explore the spatial and structural information within image patches via high-level feature learning. Unlikeexisting works to extract structural features solely based on human design the proposed approach derives high level features by applying unsupervised representation learning approach, where spatial and structural patterns from the low level and middle-level features can be automatically captured. Here, adopt DBM to learn high level feature because it has been demonstrated to have the potential of learning useful distributed feature representations and become a promising way in solving object and speech recognition problems. By studying millions of instances, an image recognition tool can learn to recognise and describe items in photographs, just as a chat bot can learn to make lifelike exchanges with people when fed examples of text chats.

This area of Artificial Intelligence programming focuses on gathering data and formulating rules for how to transform the data into useful knowledge. Algorithms are sets of rules that give computing devices detailed instructions on how to carry out a certain task. The recent advance of remote sensing technology has led to the explosive growth of satellite and aerial images in both quantity and quality. It brings about two increasingly serious problems for the object detection task in optical RSIs. First, supervised-learning-based object detection approaches often require a large number of training data with manual annotation of labeling a bounding box around each object to be detected. In addition, manual annotation is also difficult for the man-made objects such as airplane and car, where the coverage of target object appears to be very small, particularly when complex textures are contained in the image background. As a result, it is difficult to achieve accurate annotation on such small regions. Moreover, the manual annotations may tend to be less accurate and unreliable when the targets are occluded or camouflaged. As a result, it is a great interest in training object detectors with weak supervision for large-scale optical satellite and aerial image data sets.

1.1 DOMAIN INTRODUCTION

Artificial Intelligence is defined as the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. The intelligence demonstrated by machines is known as Artificial Intelligence. Artificial Intelligence has grown to be very popular in today's world. It is the simulation of natural intelligence in machinesthat are programmed to learn and mimic the actions of humans. These machines are able to learnwith experience and perform human-like tasks.

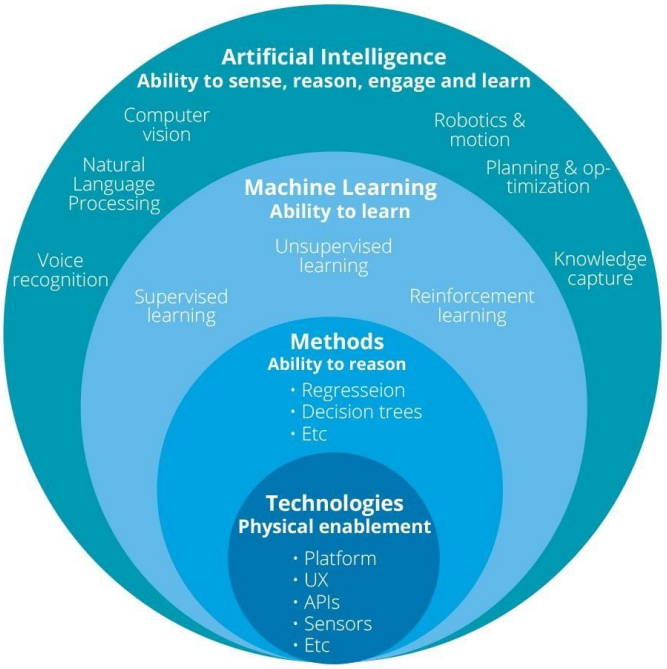


Fig. No. 1.1 Artificial Intelligence

In general, AI systems work by ingesting large amounts of labeled training data, analyzing the data for correlations and patterns, and using these patterns to make predictions about future states. In this way, a chat bot that is fed examples of text chats can learn to produce lifelike exchanges with people, or an image recognition tool can learn to identify and describe objects in images by reviewing millions of examples. This aspect of AI programming focuses on acquiring data and creating rules for how to turn the data into actionable information. The rules, which are called algorithms, provide computing devices with step-by-step instructions for how to complete a specific task.

Computational statistics, which focuses on making predictions with computers, is closely related to machine learning. Methods, concepts, and tools for mathematical optimization Areas where machine learning is applied. A subfield of Machine Learning, data mining focuses on unsupervised learning for exploratory data analysis. Machine learning is also known as "predictive analytics" when it is applied to various business issues. The idea of Machine Learning has been around for a while. Artificial Intelligence and computer gaming pioneer Arthur Samuel, a computer scientist at IBM, is credited with coining the term "machine learning." Samuel created a checkers game for the computer. The more it was used, the more the program learned from its mistakes and used algorithms to forecast outcomes.

1.2 MACHINE LEARNING

Machine learning is the scientific study of algorithms and statistical models thatcomputer systems use to perform a specific task without using explicit instructions, relying on patterns and inference instead. It is seen as a subset of artificial intelligence. Machine learning algorithms build a mathematical model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to perform the task. Machine learning algorithms are used in a wide variety of applications, such as email filtering and computer vision, where it is difficult or infeasible to develop a conventional algorithm for effectively performing the task. Machine learning is closely related to computational statistics, which focuses on making predictions using computers. The study of mathematical optimization delivers methods, theory and application domains to the field of machine learning. Data mining is a field of study within machine learning, and focuses on exploratory data analysis through unsupervised learning. In its application across business problems, machine learning is also referred to as predictive analytics.

Machine Learning is the study of statistical models and methods that computers employ to carry out specified tasks without explicit instructions by relying on patterns and inference rather than data. It is considered to be a part of Artificial Intelligence. Machine Learning algorithms create a mathematical model from sample data, referred to as "training data," in order to make predictions or judgements without being expressly programmed to do so. Machine Learning algorithms are utilized in a wide range of applications, including email filtering and computer vision, when it is challenging or impossible to create a traditional algorithm that can efficiently complete the task.

Machine learning as a concept has been around for quite some time. The term “machine learning” was coined by Arthur Samuel, a computer scientist at IBM and a pioneer in AI and computer gaming. Samuel designed a computer program for playing checkers. The more the programplayed, the more it learned from experience, using algorithms to make predictions.

1.3 OBJECT DETECTION

Object detection is a computer technology related to computer vision and image processing that deals with detecting instances of semantic objects of a certain class in digital images and videos. Well-researched domains of object detection include face detection and pedestrian detection. Object detection has applications in many areas of computer vision, including image retrieval and video surveillance.

The “Single-object localization” is a simpler version of the more broadly defined “Object Localization,” constraining the localization tasks to objects of one type within an image, which may assume is an easier task. The performance of a model for image classification is evaluated using the mean classification error across the predicted class labels. The performance of a model for single-object localization is evaluated using the distance between the expected and predicted bounding box for the expected class. Whereas the performance of a model for object recognition is evaluated using the precision and recall across each of the best matching bounding boxes for the known objects in the image. Lab recognition is to describe a collection of related computer vision tasks that involve activitieslike identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image. Object localization refers to identifying the location ofone or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combining these two tasks and localizing and classifying one or more objects in an image. When a user or practitioner refers to the term “object recognition“, it often mean “object detection“. It may be challenging for beginners to distinguish between different related computer vision tasks.

Input: An image that consists of one or more objects, such as a photograph.

Output: One or more bounding boxes, e.g. defined by a point, width, and height, and a class label for each bounding box.

One of the further extensions to this breakdown of computer vision tasks is object segmentation, also called “object instance segmentation” or “semantic segmentation,” where instances of recognized objects are indicated by highlighting the specific pixels of the object instead of a coarse bounding box. From this breakdown, understand that object recognition refers to a suite of challenging computer vision tasks.

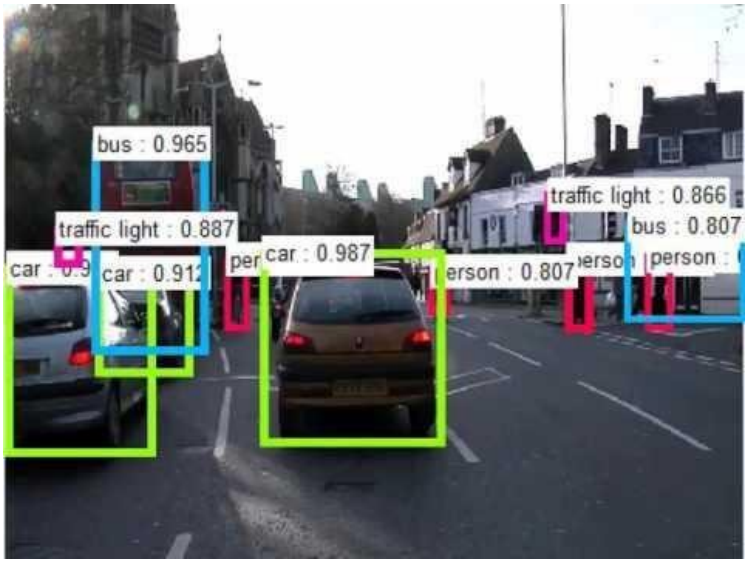


Fig. No. 1.2 Object detection

For example, image classification is simply straightforward, but the differences between object localization and object detection can be confusing, especially when all three tasks may be just as equally referred to as object recognition. Humans can detect and identify objects present in an image. The human visual system is fast and accurate and can also perform complex tasks like identifying multiple objects and detecting obstacles with little conscious thought. With the availability of large sets of data, faster GPUs, and better algorithms, so now easily train computers to detect and classify multiple objects within an image with high accuracy. Understand terms such as object detection, object localization, and loss function for object detection and localization, and finally, explore an object detection algorithm known as “You only look once”.

Image classification also involves assigning a class label to an image, whereas object localization involves drawing a bounding box around one or more objects in an image. Object detection is always more challenging and combines these two tasks and draws a bounding boxaround each object of interest in the image and assigns them a class label. Together, all these problems are referred to as object recognition. Object recognition refers to a collection of related tasks for identifying objects in digital photographs. Region-based Convolutional Neural Networks, or R-CNNs, are a family of techniquesfor addressing object localization and recognition tasks, designed for model performance. You Only Look Once, or YOLO is known as the second family of techniques for object recognition designedfor speed and real-time use

CHAPTER 2

LITERATURE SURVEY

Liu et al., suggested YOLO-You Only Look Once switching the focus on speed from accuracy. YOLO works in near real time and maintains reasonable accuracy. It is followed by further versions YOLOv2, YOLOv3, YOLOv4 and YOLOv5 which perform better in specific areas. E.g. YOLOv3 works accurately on smaller objects, but performs worse on larger objects etc. The different versions of YOLO use CNNs of various sizes and return bounding boxes around objects in the frame. It uses a single feed forward propagation across the network to detect the objects and multiple independent logistic classifiers instead of soft max for class prediction, which are trained with binary cross-entropy loss. In various fields, there is a necessity to detect the target object and also track them effectively while handling occlusions and other included complexities. Many researchers Almeida and Guting 2004, Nicolas Papadakis and Aure lie Bureau 2010 attempted various approaches to object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to the proposed work.

B. Alexe et al., suggested it is necessary in many fields to identify the target item and track it efficiently while managing occlusions and other added complexity. Several approaches of object tracking were tested by several researchers. The application domain heavily influences the approaches' character. The following is a representation of some of the research projects in the field of object tracking that led to the proposed work.

Wojke et al., suggested in Deep SORT - a modification of Simple Online Real time Tracking algorithm with deep metric learning. By using a cosine-based metric, it was able to track objects over longer durations overcoming occlusions and switches in identity. The past few decades have seen a great deal of research into optical RSIs' ability to detect objects or targets. For instance, using the improved parallel-beam Radon transform and the ridge let transform, Li et al. developed an algorithm for straight road edge recognition using optical RSIs.

Hsiao-Ping Tsai et al., suggested in certain applications like robotics, it is required to grasp object, and for that purpose an object needs to be identified. To achieve this is the authors uses Detectron2 R-CNN to make the mask for the object. It used the Cornell Grasping dataset to predict an optimal rectangle for grasping a particular object. Owing to the incredibly small sizes of the subjects in contrast to the background and the steep angle of inclination of the sensor, the Detectron2 effectively masks the subject used extracted shape and context data from the segmented image to identify inshore ships in optical satellite images.

M. I. H. Azhar et.al., suggested in the author proposed S-Siam framework in real time environment. In this paper it is mentioned that the tracking of a particular object is lost when the camera jitters specially when the object is small in size and moving very fast. The experiments were conducted on VOT2016, VOT2018 and VOT2019 datasets and achieve an EAO score of 0.449 and give 10% improvement when compared with other trackers. By using morphological filters to remove road lines and histogram representation to distinguish vehicle targets from background, Liu et al. demonstrated reliable automatic vehicle detection in Quick Bird satellite.

J. Leitl off et al., suggested that these techniques are all used in an unsupervised manner. In a straightforward context are efficient in identifying the designed object category. Many methods began to frame object detection as a classification problem as machine learning techniques advanced. In these methods, a group of features that can describe the objects are extracted first. The classification process is then carried out utilising the extracted features and predefined classifiers. Han et al., for instance, suggested using visual saliency modelling and discriminative learning of sparse coding to detect geographical objects of several classes.

Cheng et al., suggested in the multiple object tracking is used in urban traffic environment. The detection is done using YOLOv3 and then the tracking is done using Deep Sort algorithms. Urban Tracker dataset is used for experimentation and achieved a precision of 0.8989 and accuracy of 0.4265, trained deformable part-based mixture models using the histogram of oriented gradients feature and latent support vector machines. Our earlier work in Zhang et al. utilised WSL as a first step and heuristically merged it with saliency-based self-adaptive segmentation, a negative mining technique, and a negative evaluation mechanism for target detection in RSIs. This study can be greatly improved but does so by omitting certain crucial facts and lacking a moral framework. Liu et al. presented YOLO-You Only Look Once switching the focus on speed from accuracy.

F. H. K. Zaman et al., suggested that YOLO works in near real time and maintains reasonable accuracy. It is followed by further versions YOLOv2, YOLOv3, YOLOv4 and YOLOv5 which perform better in specific areas. E.g. YOLOv3 works accurately on smaller objects, but performs worse on larger objects etc. The different versions of YOLO use CNNs of various sizes and return bounding boxes around objects in the frame. It uses a single feed forward propagation across the network to detect the objects and multiple independent logistic classifiers instead of soft max for class prediction, which are trained with binary cross-entropy loss using deep metric learning, Wojke et al. modified the Simple Online Real time Tracking algorithm to create Deep SORT. It overcame occlusions and switches in identity by utilising a cosine-based metric to follow objects for longer periods of time. In certain applications like robotics, it is required to grasp object, and for that purpose an object needs to be identified.

J. Jin et al., suggested, to achieve this, in, the authors uses Detectron2 R-CNN to make the mask for the object. Thus the Cornell Grasping dataset to predict an optimal rectangle for grasping a particular object. Owing to the incredibly small sizes of the subjects in contrast to the background and the steep angle of inclination of the sensor, the Detectron2 effectively masks the subject. In a real-time scenario, the author presented the S-Siam framework. In this work, it is stated that when the camera jitters, tracking of a specific item is lost, particularly when the object is small and moving quickly. With an EAO score of 0.449 and a 10% improvement over previous trackers, the experiments on the VOT2016, VOT2018, and VOT2019 datasets were successful.

H. Hashim et al., suggested, in a scenario with metropolitan traffic, multiple object tracking is used. YOLOv3-based algorithms are used for detection, and Deep Sort-based techniques are used for tracking. A precision of 0.8989 and an accuracy of 0.4265 were obtained using the Urban Tracker dataset for the experiment.The Deep Sort framework is utilised for crowd surveillance in a real-time context. YOLOv3 is used to detect people, and Deep SORT is used to analyse each frame of the observed individual to anticipate their mobility. The employed three different versions of YOLOv3—YOLOv3 Tiny, YOLOv3 Custom—each of which varied in weight, file size, and object class.

S. Guan et al., suggested in Siamese Network and Optical Flow which is obtained from Kalman filter for realtime object tracking system. The efficiency of their method is tested on MOT13 bench mark. It also claim that the performance of their algorithm is much better than DeepSort Algorithm. The Siamese Network and Optical Flow that the author presented for a real-time object tracking system are derived from the Kalman filter. Their approach's effectiveness is evaluated using the MOT13 benchmark. It also assert that their method performs considerably better than the Deep Sort Algorithm.

CHAPTER 3

SYSTEM OVERVIEW

3.1 RELATED WORK

Object or target detection in optical RSIs has been extensively studied in the past decades. For example, Li et al. developed an algorithm for straight road edge detection from optical RSIs based on the ridgelet transform with the revised parallel-beam Radon transform.[10] Liu et al. detected inshore ships in optical satellite images by using shape and context information that is extracted inthe segmented image. Liu et al. presented robust automatic vehicle detection in Quick Bird satellite images by applying morphological filters for road line removing and histogram representation for separating vehicle targets from background. All these methods are performed in an unsupervised manner. Thus effective for detecting the designed object category in simple scenario. With the advancement of machine learning techniques, many approaches started to cast object detection as a classification problem. In these approaches, a set of features that can characterize the objects is extracted first. Then, classification is performed using the extracted features and predefined classifiers. For example, [15] Han et al. proposed to detect multiple class geospatial objects based on visual saliency modeling and discriminative learning of sparse coding. [4] Cheng et al. used histogram of oriented gradients to train deformable part-based mixture models for each object category. Based on the prior information obtained from a large number of human-labeled training examples, the supervised-learning-based approaches normally can achieve better performance. However, collection of large-scale training examples isoften difficult and very time consuming.

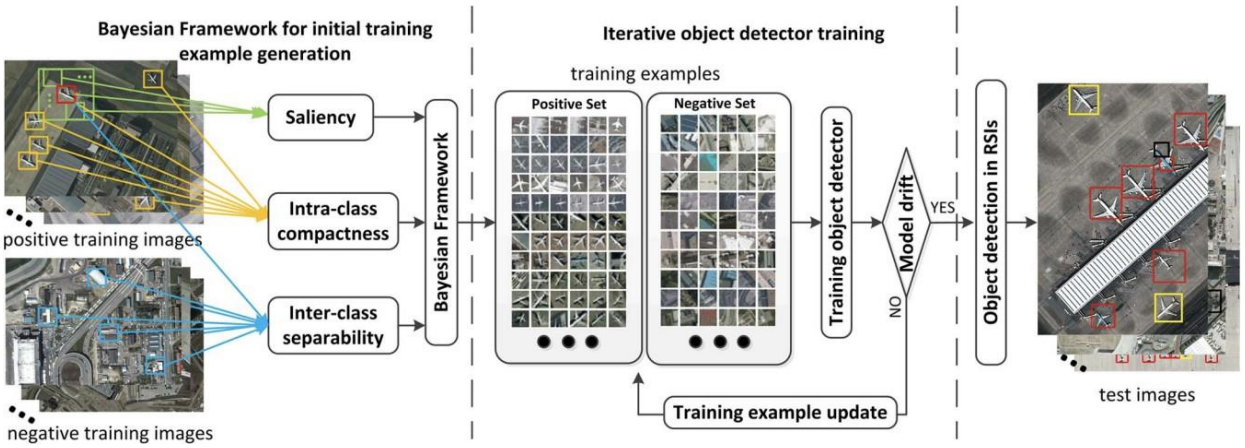


Fig. No. 3.1 Flowchart of WSL-based object detection.

A few efforts have been performed to alleviate the work of human annotation. One interesting idea is to adopt the semi supervised learning model. Such methods apply a self-learning or an active learning scheme where machine learning algorithms can automatically pick the most informative unlabeled examples based on a limited set of available labeled examples. Then, these picked unlabeled examples are combined with the initial labeled examples for the training of object detector or classifier. Specifically, [17] Liao et al. proposed a semi supervised local discriminant analysis method for feature extraction in hyperspectral RSI. Dopido et al. adapted active learning methods to semi supervised learning for hyperspectral image classification. Jun and Ghosh presented a semi supervised spatially adaptive mixture model to identify land covers from hyperspectral images. Although semi supervised learning methods can considerably reduce the labor of human annotation, and still inevitably require a number of precise and concrete labeled training examples where each object is manually labeled by a bounding box in positive training images. WSL is desirable to further reduce the human labor significantly, where the training set needs only binary labels indicating whether an image contains the object of interest. Although a few WSL approaches have been appliedto natural scene image analysis , those existing methods cannot be directly used to the field of RSI analysis have insufficient capability to handle the challenges in RSIs, which contain large-scale complex background and a number of target objects with arbitrary orientation. As an initial effort, in our previous work in [7]Zhang et al., WSL was adopted and heuristically combined with saliency-based self adaptive segmentation, negative mining algorithm, and negative evaluation mechanism for target detection in RSIs. This work lacks a principled framework and ignores some important information, which, thus, can be largely improved.

The fundamental flaw in Siam Mask, the cutting-edge single object tracking and segmentation algorithm, is addressed in this study. Since a bounding box must be manually constructed around the object that wants to be tracked, Siam Mask requires some semi-supervision. It's not always possible to do this though. Or even in the best case scenario slows down the pipeline. The object is automatically detected using Detectron2 and YOLO, and then it is tracked using Siam Mask, which allows us to get around this limitation The major goal of this project is to provide an effective method for end-to-end object identification and tracking, which may then be applied to other applications, such as self-driving automobiles, etc. For time and detection efficiency, examined various methods utilising the most advanced technologies available today. The two techniques' performance on various kinds of datasets was one of the secondary goals. Observe that the object detection provided by YOLO is more accurate and useful. In contrast, Detectron2 offers a faster detection rate than YOLO, hastening the detection and tracking process as a whole.

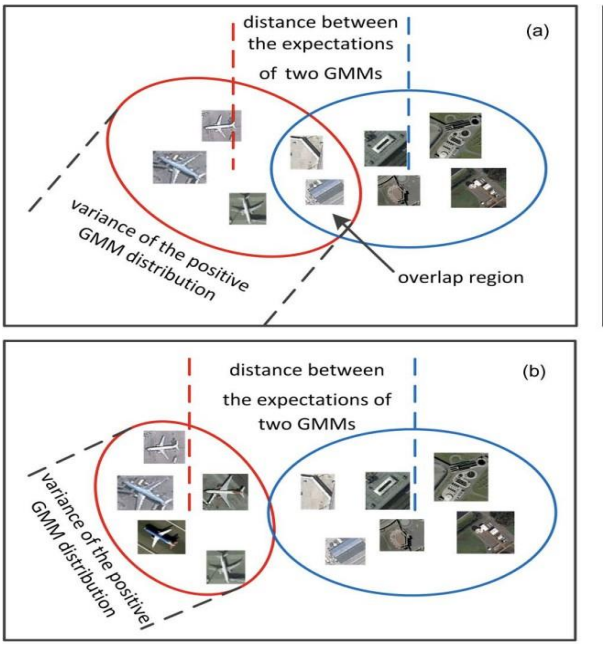


Fig. No. 3.2 Iterative training of airport detector

3.2 EXISTING WORK

Object detection in optical RSIs is a challenge, and provide a novel approach to solve it in this project. Project being considered. Due to its innovation in two crucial areas, it stands apart from prior works. To begin with, this learning strategy is not entirely or partially supervised, unlike most others. With the help of a WSL framework created in this study, training data annotation may be completed with a lot less effort and still produce excellent results. To learn high-level features unsupervised, then secondly created a deep network. This descriptor performs better in RSIs at encapsulating the specifics of an object's structural makeup. Consequently, it can improve object detection performance even further. sets. These three areas will be the focus of our upcoming endeavour. [16] W. Liao, An initially enlarge it to incorporate the collaborative learning of a number of various object detectors. Second, in order to reliably detect objects, then combine spatial information with the extensive spectrum information provided by RSIs. Third, transfer learning will be used to further improve the WSL framework.

3.2.1 HIGH – LEVEL FEATURE REPRESENTATION

The performance of the existing feature descriptions in RSI analysis is still far from satisfactory. The main issue lies in the insufficiency in extracting features using only the pixel based spectral

information, which ignores the contextual spatial information and thus fails to capture the more important structural pattern of the object. With the advancement of the remote sensing technology, optical satellite and aerial imagery with high spatial resolution makes capturing spatial and structural information possible. Nowadays, accurate interpretation of optical RSI relies on effective spatial feature representation to capture the most structural and informative property of the regions in each image. A number of such approaches have started to explore the spatial information by applying some low-level descriptors and gray-level co-occurrence matrice or middle-level features such as bag of word to represent image patches. Although to some extent these human-designed features can improve the classification and detection accuracies in optical RSIs suffer from several problems. Specifically, the low-level descriptors only catch limited local spatial geometric characteristics, which cannot be directly used to describe the structural contents of image patches. The middle-level features are usually extracted based on the statistic property of the low-level descriptors to capture the structural information of the spatial region. However, it cannot provide enough strong description and generalization ability for object detection in complex backgrounds.

To tackle these problems, build high-level feature representation via DBM to capture the spatial and structural patterns encoded in the low-level and middle-level features. DBM is one type of neural networks with deep architecture that learns feature representation in an unsupervised manner and demonstrated to be promising for building high-level feature descriptors. Therefore use it to map the middle level features to the high-level representation that is highly accurate in characterizing different scenes or objects in optical RSIs. Specifically, the extraction of high-level feature representation is carried out in three main stages, i.e., low level descriptor extraction, middle-level feature generation, and high-level feature learning.

3.2.1.1 LOW-LEVEL DESCRIPTOR EXTRACTION

Use low-level features to characterize the local region of each key point in image patches. Due to its ability to handle variations in terms of intensity, rotation, scale, and affine projection, the SIFT descriptor [27] is adopted in the proposed algorithm as the low-level descriptor to detect and describe the key points. According to existing work the SIFT descriptor has been demonstrated to outperform a set of existing descriptors, which are also widely used in analyzing RSIs.

3.2.1.2 MIDDLE-LEVEL FEATURE GENERATION

To alleviate the unrecoverable loss of discriminative information, apply the locality-constrained linear coding model to encode the local descriptors into image patch representation. Specifically, all the extracted low-level descriptors are clustered to generate a codebook by using the K-means method. Let

D = [d1, d2,..., dN] denote a set of N extracted low-level descriptors in one image patch. Given a codebook CB = [cb1, cb2,..., cbM] with M entries, LLC converts each descriptor into a M-dimensional code to generate the image patch representation by the following three steps.

1) For each input low-level descriptor dn, n ∈ [1, N], its five nearest neighbors in CB are used as the local bases LBn to form a local coordinate system .

2) The local code ˜cn is obtained by solving an objective function Then, the full code cn is generated, which is an M × 1 vector with five nonzero elements whose values correspond to ˜cn.

$$\min \sum_{n=1}^{N} ||\mathbf{d}_n - \tilde{\mathbf{c}}_n \cdot \mathbf{LB}_n||^2 \text{ s.t. } \sum_{n=1}^{N} \tilde{\mathbf{c}}_n = 1. \qquad (1)$$

3) The final middle-level image patch representation is yielded by max pooling all the generated codes within the patch.

3.2.1.3 HIGH-LEVEL FEATURE LEARNING

A DBM is a neural network with deep structure constructed by stacking multiple restricted Boltzmann machines. In our framework, a three-layered DBM is adopted to learn high-level representations by capturing the structural and spatial patterns from middle-level features in an unsupervised manner. It contains a visible layer v ∈ {0, 1}M and two hidden layers h1 ∈ {0, 1}H1 and h2 ∈ {0, 1}H2 , where H1 and H2 indicate the numbers of units of the first and second hidden layers, respectively. The energy of the state {v, h1, h2} is
defined as

$$E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \Theta) = -\mathbf{v}^T \mathbf{W}^1 \mathbf{h}^1 - \mathbf{h}^{1T} \mathbf{W}^2 \mathbf{h}^2 \qquad (2)$$

where Θ = {W1,W2} are the model parameters, denoting visible-to-hidden and hidden-to-hidden symmetric interaction
terms. The probability that the model assigns to a visible vector v is given by the Boltzmann distribution

$$\Pr(\mathbf{v}; \Theta) = \frac{1}{Z(\Theta)} \sum_{\mathbf{h}^1, \mathbf{h}^2} \exp\left(-E(\mathbf{v}, \mathbf{h}^1, \mathbf{h}^2; \Theta)\right) \qquad (3)$$

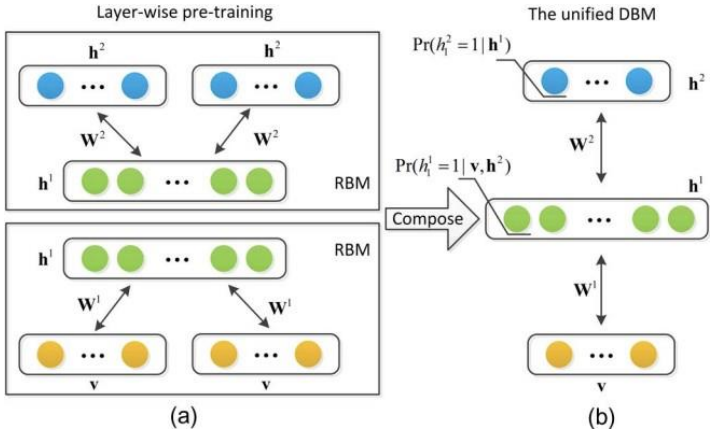


Fig. No. 3.3 Learning processes for DBM

The conditional distributions over the visible units and the two sets of hidden units are given by

$$\Pr(h_i^1 = 1|\mathbf{v}, \mathbf{h}^2) = \text{sigm}\left(\sum_{m=1}^{M} W_{mi}^1 v_m + \sum_{j=1}^{H_2} W_{ij}^2 h_m^2\right) \quad (4)$$

$$\Pr(h_j^2 = 1|\mathbf{h}^1) = \text{sigm}\left(\sum_{i=1}^{H_1} W_{mj}^2 h_i^1\right) \quad (5)$$

$$\Pr(v_m = 1|\mathbf{h}^1) = \text{sigm}\left(\sum_{i=1}^{H_1} W_{mi}^1 h_i^1\right) \quad (6)$$

where sigma is a sigmoid function. Given a set of training data, learning of DBM is a process to determine the related model parameters Θ = {W1,W2}. Although exact maximum-likelihood estimation of these parameters is intractable, efficient approximate learning of DBMs can be carried out by using mean-field inference together with the Markov chain Monte Carlo algorithms [10].Furthermore, the entire model can be efficiently pre trained in a layer-by-layer unsupervised manner by minimizing the energy function in each individual RBM model . Composing the RBM models afterward forms a unified DBM model, which can be used to extract high-level feature representation. In the proposed algorithm, all the middle-level features extracted from the image patches in training images are used as the input data to train DBM, where the second hidden layer is used to build the final high-level feature representation for each image patch.

3.2.2 WSL-BASED OBJECT DETECTION

By applying sliding windows as preprocessing, the training images are divided into many patches. Thus, the patch-level training data X+ = {x+ p|p ∈ [1, P]} and X− = x− q |q ∈ [1, Q]} can be generated from the positive training images and the negative training images, respectively. Our first task is to select potential target object patches from X+ to generate the initial positive training set X+

0 . Typically, three different information cues, namely, saliency, intra class compactness, and interclass separability [20] , are used to initialize the positive training examples. Based on the assumption that the object to be detected is one kind of foreground objects in the image, saliency information ensures that the selected positive example is a foreground region. It acquires generic knowledge about the sizes and locations of the objects. The intra class compactness enforces the selected positive examples to be visually similar to each other, whereas the interclass separability ensures that all selected positive examples are different from negative examples. In this project, a novel Bayesian framework is proposed to combine these three types of information simultaneously to initialize the positive example training set as follows. Let y+ p denote whether an image patch x+ p belongs to one specified object. According to Bayes' rule

$$\Pr\left(y_p^+ = 1|x_p^+\right) = \frac{\Pr\left(x_p^+|y_p^+ = 1\right)\Pr\left(y_p^+ = 1\right)}{\Pr\left(x_p^+\right)} \tag{7}$$

$$\begin{aligned}\Pr\left(y_p^+ = 1|x_p^+\right) &= 1 - \Pr\left(y_p^+ = 0|x_p^+\right) \\ &= 1 - \frac{\Pr\left(x_p^+|y_p^+ = 0\right)\Pr\left(y_p^+ = 0\right)}{\Pr\left(x_p^+\right)}.\end{aligned} \tag{8}$$

After adding the preceding two equations and omitting the constant term,

$$\begin{aligned}\Pr\left(y_p^+ = 1|x_p^+\right) \propto \underbrace{\frac{1}{\Pr\left(x_p^+\right)}}_{\text{Saliency}}\Big[&\underbrace{\Pr\left(x_p^+|y_p^+ = 1\right)}_{\substack{\text{Intra-class}\\ \text{compactness}}} \\ \times \underbrace{\Pr\left(y_p^+ = 1\right)}_{\substack{\text{Prior}\\ \text{Probability}}} - &\underbrace{\Pr\left(x_p^+|y_p^+ = 0\right)}_{\substack{\text{Inter-class}\\ \text{separability}}}\underbrace{\Pr\left(y_p^+ = 0\right)}_{\substack{\text{Prior}\\ \text{Probability}}}\Big].\end{aligned} \tag{9}$$

In the information theory, −log Pr(x+ p), which is the log form of 1/Pr(x+ p), is known as the self-information of the random variable x+ p Self-information increases when the probability of a patch decreases. In other words, patches discriminative from surroundings are more informative and thus more likely to be objects. Therefore, the term of 1/Pr(x+ p) in (9) is associated with the saliency information. The term Pr(x+ p|y+ p = 1) indicates the likelihood that favors image patches sharing the similar characteristic with the class of target object. Hence, it can be considered as the metric of intra class compactness. Similarly, Pr(x+ p|y+ p = 0) reflects the distinctness of image patches in positive and negative images; thus, it corresponds to the metric of interclass separability. Finally, the remaining two prior probabilities Pr(y+ p = 1) and Pr(y+ p = 0) are treated as the weights of the intra class and interclass metrics, respectively.

3.2.2.1 SALIENCY

Assume that objects to be detected are normally one kind of foreground objects, our objective then becomes quantifying how likely each image patch is a foreground object. Foreground objects are generally informative and salient from the surrounding background, as shown in In computer vision, saliency detection technique can be used to estimate the saliency for each image patch. In recent years, it is also employed for the analysis in the domain of

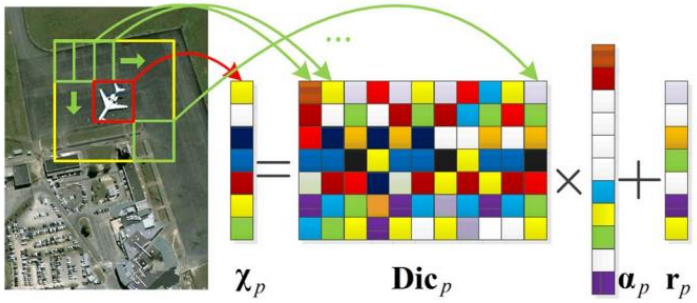


Fig. No. 3.4 Illustration of saliency calculation

remote sensing . Inspired by adopt sparse coding theory to calculate saliency based on the raw pixels to reveal the structural difference between an image patch and its surrounding. For each image patch x+ p it is sparsely coded with its adjacent half overlapped surrounding patches patches indicated by green frames in

$$\chi_p \approx \mathbf{Dic}_p \boldsymbol{\alpha}_p \qquad (10)$$

where χp are the raw pixels within x+ p, whereas Dicp and αp indicate the dictionary constructed by all surrounding patches and the sparse codes, respectively. The rationale behind is to represent χp approximately by its surrounding patches. According to the coding sparseness ||αp||0 and the coding residual rp = χp − Dicpαp indicates the saliency of the image patch x+ p with respect to its surrounding. Therefore, estimate the saliency by

$$1/\Pr\left(x_p^+\right) = ||\boldsymbol{\alpha}_p||_0 \cdot ||\mathbf{r}_p||_1. \qquad (11)$$

3.2.2.2 INTRA CLASS COMPACTNESS

Termed as Pr(x+ p|y+ p = 1) the intra class compactness metric aims to constrain the similarity between positive examples. As positive examples of a specific object category should be visually similar, use the Gaussian mixture model to estimate the probability distribution of all positive examples. Then,

Pr(x+ p|y+p = 1) measures how likely each image patch is a positive example. Image patches with large Pr(x+ p|y+ p = 1) may be selected as positive examples.

Use the high-level feature f + p to represent each image patch x+ p as this feature can handle the variations in scale and orientation and capture the spatial and structural patterns of each image patch. As patterns learned by DBM are approximately independent, the joint probability is simplified to the product of probability of each hidden unit's response

$$\Pr\left(x_p^+|y_p^+=1\right)=\prod_{j=1}^{H_2}\Pr\left(\left[\mathbf{f}_p^+\right]_j|y_p^+=1\right) \qquad (12)$$

where [f + p]j indicates the j th -dimensional value of f + p , and H2 indicates the dimensionality of f + p . The distribution of each hidden unit's response is estimated using GMM with adaptive component K+ j=1,...,H2 by

$$\Pr\left(\left[\mathbf{f}_p^+\right]_j|y_p^+=1\right)=\sum_{k=1}^{K_j^+}\pi_{jk}^+N\left(\left[\mathbf{f}_p^+\right]_j|\mu_{jk}^+,\sigma_{jk}^{2+}\right) \qquad (13)$$

where π+ jk, µ+ jk, and σ2+ jk are parameters of the GMM in the kth component for the jth-dimensional feature. All parametersare inferred based on object candidates in X˜ + by using the expectation-maximization algorithm and Bayesian inference . Here, X˜ + denotes the set of object candidates and will be described in Section V-A5.

3.2.2.3 INTRA CLASS SEPARABILITY

The interclass separability metric is to enforce that the selected positive examples are dissimilar to negative examples. In WSL, the most confident information comes from the negative training images because it definitely do not contain the target. It is also reasonable to believe that

the positive examples containing target objects should be different from the negative image patches in the negative images. Consequently, then collect a large number of negative image patches to estimate the probability distribution of negative examples via a GMM. Then, formulate the interclass metric as the likelihood term Pr(x+ p|y+ p = 0), which reflects the probability of a certain image patch appearing in negative training

images. The high probability of the appearance in negative images would lead to low interclass difference and separability.

Similar to Pr(x+ p|y+ p = 1), Pr(x+ p|y+ p = 0) can be decided based on the high-level feature by

$$\Pr\left(x_p^+|y_p^+=0\right)=\prod_{j=1}^{H_2}\Pr\left(\left[\mathbf{f}_p^+\right]_j|y_p^+=0\right) \tag{14}$$

$$\Pr\left(\left[\mathbf{f}_p^+\right]_j|y_p^+=0\right)=\sum_{k=1}^{K_j^-}\pi_{jk}^- N\left(\left[\mathbf{f}_p^+\right]_j|\mu_{jk}^-,\sigma_{jk}^{2-}\right) \tag{15}$$

where parameters π−jk, μ−jk, σ2−jk , and K−j are inferred by GMM based on all the negative image patches.

3.2.2.4 PRIOR PROBABILITY

Pr(y+p = 1) and Pr(y+p = 0) are two prior terms in the proposed Bayesian framework. According to [24], Bayesian methods would result in poor performance when inappropriate choices of prior are applied without any prior belief. Therefore, inspired by [30], define the prior

terms to reflect the prior belief. Our prior belief is that Pr(y+p =0) should be high when the content of certain image patch x+ p has small distance to the negative image patches in X and Pr(y+p = 1) should become high when the content of x+pis close to the object candidates in X˜ +. Hence, simply adopt the nearest neighbor distance to estimate these prior probabilities as

$$\Pr\left(y_p^+=0\right)=\exp\left\{-||x_p^+-\mathrm{Nn}\left(x_p^+\right)||_1\right\} \tag{16}$$

$$\Pr\left(y_p^+=1\right)=\exp\left\{-||x_p^+-\mathrm{Np}\left(x_p^+\right)||_1\right\} \tag{17}$$

where || · ||1 is the L1 norm. Same as in [20], Nn(x+ p) and Np(x+ p) refer to the nearest neighbors of x+ p in X− and X˜ + in terms of the high level feature, respectively. Finally, these two prior terms are

used as the weights of the interclass and intra class metrics in order to reflect the prior probability that an image patch belongs to the positive and negative training examples, respectively.

3.2.2.5 IMPLEMENTATION DETAILS

The post probability Pr(y+ p = 1|x+p) is estimated by integrating the saliency,intra class, and interclass metrics. Note that, before calculating the intra class compactness metric, X˜ + needs to be available. The work [21] proposed an exhaustive searching strategy to generate one object candidate for each image. However, it lacks accuracy and efficiency for the large-scale RSIs, particularly when it contains multiple target objects located at quite scattered positions. To tackle this challenging problem, practice implement our work in two stages. In the first stage, calculate Pr(y+p = 1|x+ p) approximately by only using the saliency and interclass separability metrics to generate X˜ +. As initially Pr(x+ p|y+ p = 1) and Pr(y+ p = 0) are unknown, omit them by following [30] and [31], which is equivalent to assuming a uniform likelihood distribution for the unspecified object category

$$\Pr\left(y_p^+ = 1|x_p^+\right) \propto \frac{1}{\Pr\left(x_p^+\right)}\left[1 - \Pr\left(x_p^+|y_p^+ = 0\right)\Pr\left(y_p^+ = 0\right)\right]. \tag{18}$$

Hence, X˜ + can be further determined by choosing a probability threshold τ , i.e.,

$$\tilde{X}^+ = \left\{x_p^+ | \Pr\left(y_p^+ = 1|x_p^+\right) \geq \tau\right\}. \tag{19}$$

Once X˜ + is obtained, fully implement the proposed Bayesian framework in the second stage, where all the three types of information are explored and integrated for calculating

Pr(y+p = 1|x+p) by

Similar to the first stage, a threshold τ is chosen to generate the initial positive training set X+0 by

$$X_0^+ = \left\{x_p^+ | \Pr\left(y_p^+ = 1|x_p^+\right) \geq \tau\right\}. \tag{20}$$

By considering the fact that imbalanced positive and negative training data may reduce the performance of the object detector, follow the previous work of [24] to generate the initial negative training set X− 0 by random under sampling of X− to the same size as X+0 .

3.3 PROPOSED METHODOLOGY

The goal of the proposed work is to find target items belonging to the same class in the testing images, given a training optical RSI set with a weak label merely indicating whether a particular type of object is contained in an image or not. due to the fact that these pictures are usually fairly large a simple method of processing is to breakdown the images that are complex and have several things of interest. Sliding windows are used to divide images into small patches, and it is then determined whether the object is present in each piece is fascinating. To accommodate the fluctuation in item sizes, use the multiscale sliding window approach. Two main elements make up the suggested object detection framework: independent feature object detection using learning and WSL. The component's flowchart for feature learning. Extract the low-level and middle-level features from the image patches in order to capture the spatial information, and then use a DBM to learn the hidden patterns of the middle-level features in order to abstract more structural and semantic information and produce the desired high-level feature. The WSL-based object detection component shown in has two stages: training and testing, which are based on the acquired high-level features. Finding out how to use an object detector is the goal of the training phase. During testing, the learnt object detector is used to find things in a sample image.

Iterative object detector training and initialising the training example are the two main processes in the training stage. To estimate the likelihood that a picture patch would contain the object of interest, a Bayesian technique is suggested for the first step. This approach integrates three types of saliency, intra class compactness, and interclass information. The bootstrapping method serves as our model as iteratively train the object detector after initialising the training instances. The positive training set is improved by using the detector as an annotator in each iteration, and the detector is then retrained.

Given a training optical RSI set with a weak label only indicating whether a certain category of object is contained in an image or not, the objective of the proposed work is to detect targetobjects of the same class within the testing images. Because these images generally have a very largescale and contain multiple objects of interest, a straightforward way of processing is to decompose the images into small patches by sliding windows and then predict whether each patch contains the object of interest. Adopt the multiscale sliding window scheme to handle the variation of the object size. The proposed object detection framework consists of two major components: unsupervised feature learning and WSL based object detection. The flowchart of the feature learning component. In order to obtain a more structural and semantic representation of the image patches, extract the low-level and middle-level features to capture the spatial information and then use DBM

to learn the hidden patterns of the middle-level features, which can abstract more structural and semantic information and lead to the desiredhigh-level feature. Based on the obtained high-level features, the component of WSL-based object detection shown in contains two stages: training and testing. The objective of the training stage is to learn an object detector. In the testing stage, the learned object detector is applied to detect objects ina given testing image.

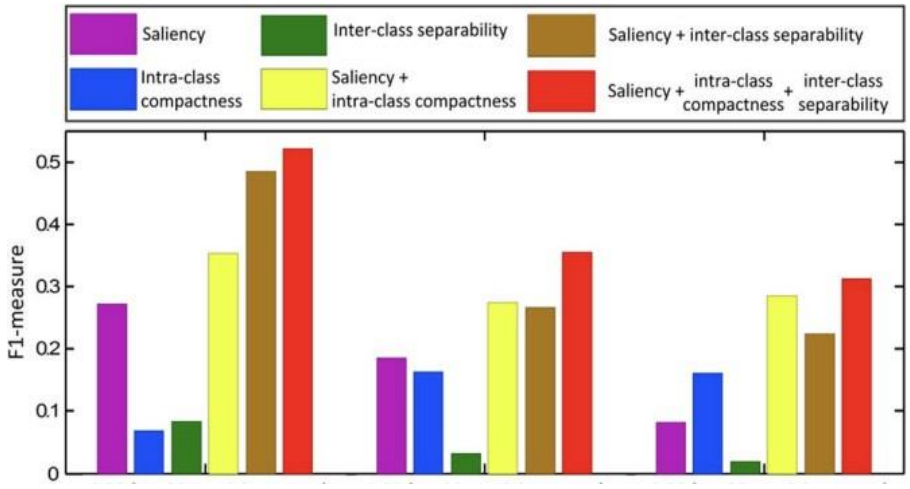


Fig. No. 3.5 Evaluation of the proposed Bayesian framework

The training stage includes two major steps: training example initialization and iterative object detector training. For the first step, a Bayesian approach is proposed to integrate three kinds of important information of saliency, intra class compactness, and interclass separability, which estimates the probability of an image patch being the object of interest. After initializing the training examples, so inspired by the bootstrapping method to train the object detector in an iterative process. In each iteration, the detector is utilized as an annotator to refine the positive training set, which is then used to retrain the detector. Thus, both the training examples and the object detector could be gradually updated to be more precise and strong. Afterward, a novel detector evaluation method is proposed to detect the model drift and stop the iterative process automatically for obtaining the final object detector.

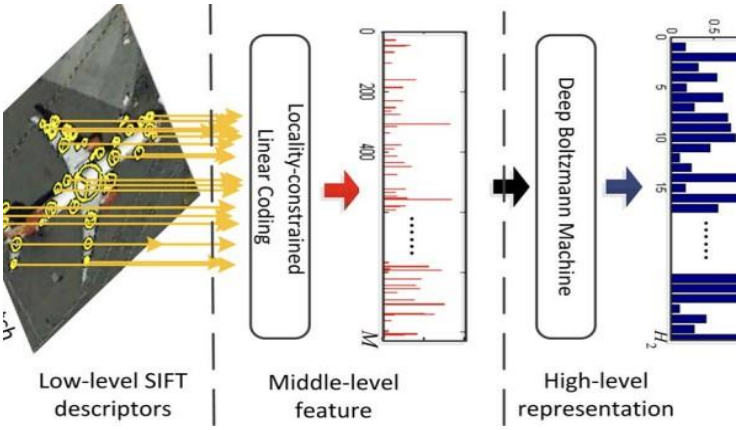


Fig. No. 3.6 High-level feature representation of the image patch.

The performance of the existing feature descriptions in RSI analysis is still far from satisfactory. The main issue lies in the insufficiency in extracting features using only the pixel based spectral information, which ignores the contextual spatial information and thus fails to capture the more important structural pattern of the object. With the advancement of the remote sensing technology, optical satellite and aerial imagery with high spatial resolution makes capturing spatial and structural information possible. Nowadays, accurate interpretation of optical RSI relies on effective spatial feature representation to capture the most structural and informative property of the regions in each image. A number of such approaches have started to explore the spatial information by applying some low-level descriptors, HOG, and gray-level co-occurrence matrice or middle-level features to represent image patches. Although to some extent these human-designed features can improve the classification and detection accuracies in optical RSIs suffer from several problems. Specifically, the low-level descriptors only catch limited local spatial geometric characteristics, which cannot be directly used to describe the structural contents of image patches. The middle-level features are usually extracted based on the statistic property of the low-level descriptors to capture the structural information of the spatial region. However, it cannot provide enough strong description and generalization ability for object detection in complex backgrounds.

CHAPTER 4
SYSTEM DESIGN AND WORKING

4.1 SYSTEM ARCHITECTURE

By contrasting the proposed weakly supervised object detector with various supervised-learning-based approaches and one existing WSL-based method, assessed the performance of the suggested detector. The proposed strategy was first contrasted with the WSL-based method. used the identical experimental circumstances for comparison, including the same feature representation created by DBM, the same sliding window design, and the same testing image.

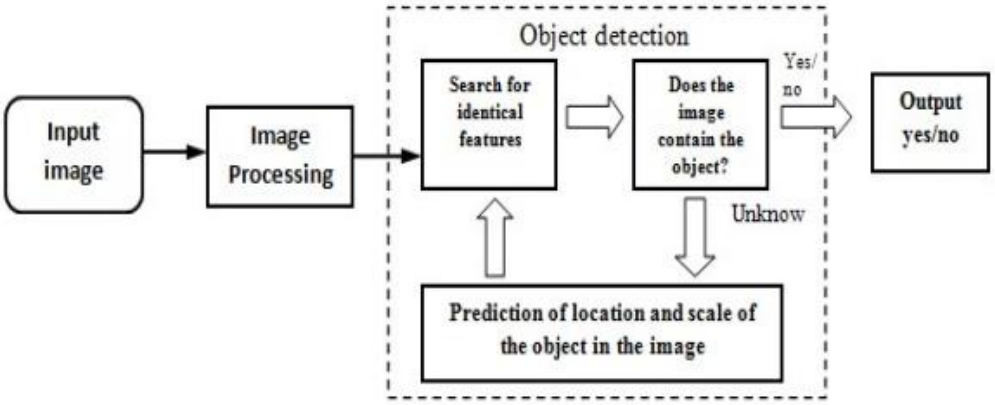


Fig. No. 4.1 System Architecture

4.2 SYSTEM REQUIREMENTS

4.2.1 SOFTWARE REQUIREMENTS

- Windows 10 or more
- Front End : Android
- Back End : JAVA
- Tenser flow

4.2.2 HARDWARE REQUIREMENTS

- Processor : Intel i7 core
- Hard Disk : 2TB
- RAM : 16 Gb or more

4.3 MODULES DESCRIPTION

4.3.1Problem statement & Requirement Identification

In phase one the problem statement is fine-tuned to suit the identification of the laboratories in college is to be implemented. Definition of the issue and Necessity.

4.3.2 Algorithm Formulation

In order to identify the specified colleges' laboratories, an algorithm has been developed.

4.3.3 Implementation Phase

The project is currently being implemented using the hardware and software components.

4.3.4 Real-time interactive kit for project submission

The final real-time implementation and demonstration and testing and comparative analysis report are to be submitted for the approval of the project. The project must have project approval before submitting the final report on real-time implementation, demonstration, testing, and comparative analysis.

CHAPTER 5

IMPLEMENTATION

5.1 SOURCE CODE

```
/*
 * Copyright 2020 Google LLC. All rights reserved.
 *
 * Licensed under the Apache License, Version 2.0 the "License";
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.google.mlkit.vision.demo;

import static java.lang.Math.max;
import static java.lang.Math.min;

import android.content.Context;
import android.graphics.Canvas;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.util.AttributeSet;
import android.view.View;
import com.google.common.base.Preconditions;
import com.google.common.primitives.Ints;
import java.util.ArrayList;
```

```
import java.util.List;

/**
* A view which renders a series of custom graphics to be overlayed on top of an associated
preview
* The creator can add graphics objects, update the objects, and remove
* them, triggering the appropriate drawing and invalidation within the view.
* <p>Supports scaling and mirroring of the graphics relative the camera's preview properties.
The
* idea is that detection items are expressed in terms of an image size, but need to be scaled up to
* the full view size, and also mirrored in the case of the front-facing camera
* <p>Associated {@link Graphic} items should use the following methods to convert to view
* coordinates for the graphics that are drawn:
* <ol>
*   <li>{@link Graphic#scale(float)} adjusts the size of the supplied value from the image scale
to
*       the view scale.
*   <li>{@link Graphic#translateX(float)} and {@link Graphic#translateY(float)} adjust the
*       coordinate from the image's coordinate system to the view coordinate system.
* </ol>
*/
public class GraphicOverlay extends View {
private final Object lock = new Object();
private final List<Graphic> graphics = new ArrayList<>();
// Matrix for transforming from image coordinates to overlay view coordinates.
private final Matrix transformationMatrix = new Matrix();

private int imageWidth;
private int imageHeight;
// The factor of overlay View size to image size. Anything in the image coordinates need to be
// scaled by this amount to fit with the area of overlay View.
private float scaleFactor = 1.0f;
// The number of horizontal pixels needed to be cropped on each side to fit the image with the
// area of overlay View after scaling.
```

```
private float postScaleWidthOffset;
// The number of vertical pixels needed to be cropped on each side to fit the image with the
// area of overlay View after scaling.
private float postScaleHeightOffset;
private boolean isImageFlipped;
private boolean needUpdateTransformation = true;
/**
* Base class for a custom graphics object to be rendered within the graphic overlay. Subclass
* this and implement the {@link Graphic#draw(Canvas)} method to define the graphics element. Add
* instances to the overlay using {@link GraphicOverlay#add(Graphic)}.
*/
 public abstract static class Graphic {
 private GraphicOverlay overlay;

 public Graphic(GraphicOverlay overlay) {
 this.overlay = overlay;
 }

 /**
* Draw the graphic on the supplied canvas. Drawing should use the following methods to convert
 * to view coordinates for the graphics that are drawn:
 *
 * <ol>
 *   <li>{@link Graphic#scale(float)} adjusts the size of the supplied value from the image
 *       scale to the view scale.
 *   <li>{@link Graphic#translateX(float)} and {@link Graphic#translateY(float)} adjust the
 *       coordinate from the image's coordinate system to the view coordinate system.
 * </ol>
 *
 * @param canvas drawing canvas
 */
  public abstract void draw(Canvas canvas);
```

```
protected void drawRect(
Canvas canvas, float left, float top, float right, float bottom, Paint paint) {
canvas.drawRect(left, top, right, bottom, paint);
}

 protected void drawText(Canvas canvas, String text, float x, float y, Paint paint) {
 canvas.drawText(text, x, y, paint);
 }

 /** Adjusts the supplied value from the image scale to the view scale. */
 public float scale(float imagePixel) {
 return imagePixel * overlay.scaleFactor;
 }

 /** Returns the application context of the app. */
 public Context getApplicationContext() {
 return overlay.getContext().getApplicationContext();
 }

 public boolean isImageFlipped() {
 return overlay.isImageFlipped;
 }

 /**
 * Adjusts the x coordinate from the image's coordinate system to the view coordinate system.
 */
 public float translateX(float x) {
 if (overlay.isImageFlipped) {
 return overlay.getWidth() - (scale(x) - overlay.postScaleWidthOffset);
 } else {
 return scale(x) - overlay.postScaleWidthOffset;
 }
 }
```

```
    /**
    * Adjusts the y coordinate from the image's coordinate system to the view coordinate system.
    */
    public float translateY(float y) {
    return scale(y) - overlay.postScaleHeightOffset;
    }
/**
* Returns a {@link Matrix} for transforming from image coordinates to overlay view
coordinates.
*/
public Matrix getTransformationMatrix() {
return overlay.transformationMatrix;
}

 public void postInvalidate() {
 overlay.postInvalidate();
 }

  /**
 * Given the {@code zInImagePixel}, update the color for the passed in {@code paint}. The
color will be
 * more red if the {@code zInImagePixel} is smaller, or more blue ish vice versa. This is
 * useful to visualize the z value of landmarks via color for features like Pose and Face Mesh.
 * @param paint the paint to update color with
 * @param canvas the canvas used to draw with paint
 * @param visualizeZ if true, paint color will be changed.
 * @param rescaleZForVisualization if true, re-scale the z value with zMin and zMax to make
 *     color more distinguishable
 * @param zInImagePixel the z value used to update the paint color
 * @param zMin min value of all z values going to be passed in
 * @param zMax max value of all z values going to be passed in
 */
  public void updatePaintColorByZValue(
```

```
Paint paint,
Canvas canvas,
 boolean visualizeZ,
 boolean rescaleZForVisualization,
 float zInImagePixel,
 float zMin,
 float zMax) {
 if (!visualizeZ) {
 return;
   }

   // When visualizeZ is true, sets up the paint to different colors based on z values.
   // Gets the range of z value.
   float zLowerBoundInScreenPixel;
   float zUpperBoundInScreenPixel;

    if (rescaleZForVisualization) {
    zLowerBoundInScreenPixel = min(-0.001f, scale(zMin));
    zUpperBoundInScreenPixel = max(0.001f, scale(zMax));
    } else {
    // By default, assume the range of z value in screen pixel is [-canvasWidth, canvasWidth].
    float defaultRangeFactor = 1f;
    zLowerBoundInScreenPixel = -defaultRangeFactor * canvas.getWidth();
    zUpperBoundInScreenPixel = defaultRangeFactor * canvas.getWidth();
    }

     float zInScreenPixel = scale(zInImagePixel);

      if (zInScreenPixel < 0) {
      // Sets up the paint to be red if the item is in front of the z origin.
      // Maps values within [zLowerBoundInScreenPixel, 0) to [255, 0) and use it to control the
      // color. The larger the value is, the more red it will be.
      int v = (int) (zInScreenPixel / zLowerBoundInScreenPixel * 255);
      v = Ints.constrainToRange(v, 0, 255);
```

```
        paint.setARGB(255, 255, 255 - v, 255 - v);
        } else {
        // Sets up the paint to be blue if the item is behind the z origin.
        // Maps values within [0, zUpperBoundInScreenPixel] to [0, 255] and use it to control the
        // color. The larger the value is, the more blue it will be.
        int v = (int) (zInScreenPixel / zUpperBoundInScreenPixel * 255);
        v = Ints.constrainToRange(v, 0, 255);
        paint.setARGB(255, 255 - v, 255 - v, 255);
        }
        }
        }
public GraphicOverlay(Context context, AttributeSet attrs) {
super(context, attrs);
addOnLayoutChangeListener(
 (view, left, top, right, bottom, oldLeft, oldTop, oldRight, oldBottom) ->
 needUpdateTransformation = true);
 }

/** Removes all graphics from the overlay. */
public void clear() {
synchronized (lock) {
graphics.clear();
}
postInvalidate();
}

/** Adds a graphic to the overlay. */
public void add(Graphic graphic) {
synchronized (lock) {
graphics.add(graphic);
}
}

/** Removes a graphic from the overlay. */
```

```
public void remove(Graphic graphic) {
synchronized (lock) {
graphics.remove(graphic);
}
postInvalidate();
}

/**
 * Sets the source information of the image being processed by detectors, including size and
 * whether it is flipped, which informs how to transform image coordinates later.
 *
 * @param imageWidth the width of the image sent to ML Kit detectors
 * @param imageHeight the height of the image sent to ML Kit detectors
 * @param isFlipped whether the image is flipped. Should set it to true when the image is from
the
 *     front camera.
 */
public void setImageSourceInfo(int imageWidth, int imageHeight, boolean isFlipped) {
Preconditions.checkState(imageWidth > 0, "image width must be positive");
Preconditions.checkState(imageHeight > 0, "image height must be positive");
synchronized (lock) {
this.imageWidth = imageWidth;
 this.imageHeight = imageHeight;
 this.isImageFlipped = isFlipped;
 needUpdateTransformation = true;
 }
  postInvalidate();
 }

public int getImageWidth() {
return imageWidth;
}

public int getImageHeight() {
```

```
return imageHeight;
}

private void updateTransformationIfNeeded() {
if (!needUpdateTransformation || imageWidth <= 0 || imageHeight <= 0) {
return;
}
float viewAspectRatio = (float) getWidth() / getHeight();
float imageAspectRatio = (float) imageWidth / imageHeight;
postScaleWidthOffset = 0;
postScaleHeightOffset = 0;
if (viewAspectRatio > imageAspectRatio) {
// The image needs to be vertically cropped to be displayed in this view.
scaleFactor = (float) getWidth() / imageWidth;
postScaleHeightOffset = ((float) getWidth() / imageAspectRatio - getHeight()) / 2;
  } else {
  // The image needs to be horizontally cropped to be displayed in this view.
  scaleFactor = (float) getHeight() / imageHeight;
  postScaleWidthOffset = ((float) getHeight() * imageAspectRatio - getWidth()) / 2;
  }

  transformationMatrix.reset();
  transformationMatrix.setScale(scaleFactor, scaleFactor);
  transformationMatrix.postTranslate(-postScaleWidthOffset, -postScaleHeightOffset);

  if (isImageFlipped) {
  transformationMatrix.postScale(-1f, 1f, getWidth() / 2f, getHeight() / 2f);
  }

  needUpdateTransformation = false;
  }

/** Draws the overlay with its associated graphic objects. */
@Override
```

```
protected void onDraw(Canvas canvas) {
super.onDraw(canvas);

synchronized (lock) {
updateTransformationIfNeeded();

for (Graphic graphic : graphics) {
graphic.draw(canvas);
}
}
}
}
```

```
/*
 * Copyright 2020 Google LLC. All rights reserved.
 *
 * Licensed under the Apache License, Version 2.0 (the "License");
 * you may not use this file except in compliance with the License.
 * You may obtain a copy of the License at
 *
 *     http://www.apache.org/licenses/LICENSE-2.0
 *
 * Unless required by applicable law or agreed to in writing, software
 * distributed under the License is distributed on an "AS IS" BASIS,
 * WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 * See the License for the specific language governing permissions and
 * limitations under the License.
 */

package com.google.mlkit.vision.demo;

import android.Manifest;
```

```
import android.annotation.SuppressLint;
import android.app.Activity;
import android.content.Context;
import android.graphics.ImageFormat;
import android.graphics.SurfaceTexture;
import android.hardware.Camera;
import android.hardware.Camera.CameraInfo;
import android.util.Log;
import android.view.Surface;
import android.view.SurfaceHolder;
import android.view.WindowManager;
import androidx.annotation.Nullable;
import androidx.annotation.RequiresPermission;
import com.google.android.gms.common.images.Size;
import com.google.mlkit.vision.demo.preference.PreferenceUtils;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.util.ArrayList;
import java.util.IdentityHashMap;
import java.util.List;

/**
 * Manages the camera and allows UI updates on top of it (e.g. overlaying extra Graphics or
 * displaying extra information). This receives preview frames from the camera at a specified
rate,
 * sending those frames to child classes' detectors / classifiers as fast as it is able to process.
 */
public class CameraSource {
@SuppressLint("InlinedApi")
public static final int CAMERA_FACING_BACK = CameraInfo.CAMERA_FACING_BACK;

@SuppressLint("InlinedApi")
public static final int CAMERA_FACING_FRONT =
CameraInfo.CAMERA_FACING_FRONT;
```

```
public static final int IMAGE_FORMAT = ImageFormat.NV21;
public static final int DEFAULT_REQUESTED_CAMERA_PREVIEW_WIDTH = 480;
public static final int DEFAULT_REQUESTED_CAMERA_PREVIEW_HEIGHT = 360;

private static final String TAG = "MIDemoApp:CameraSource";

/**
* The dummy surface texture must be assigned a chosen name. Since we never use an OpenGL context,
* we can choose any ID we want here. The dummy surface texture is not a crazy hack - it is
* actually how the camera team recommends using the camera without a preview.
*/
private static final int DUMMY_TEXTURE_NAME = 100;

/**
* If the absolute difference between a preview size aspect ratio and a picture size aspect ratio
* is less than this tolerance and are considered to be the same aspect ratio.
*/
private static final float ASPECT_RATIO_TOLERANCE = 0.01f;

protected Activity activity;

private Camera camera;

private int facing = CAMERA_FACING_BACK;

/** Rotation of the device, and thus the associated preview images captured from the device. */
private int rotationDegrees;

private Size previewSize;

private static final float REQUESTED_FPS = 30.0f;
private static final boolean REQUESTED_AUTO_FOCUS = true;
```

```
 // This instance needs to be held onto to avoid GC of its underlying resources. Even though it
 // isn't used outside of the method that creates it, it still must have hard references maintained
 // to it.
 private SurfaceTexture dummySurfaceTexture;

 private final GraphicOverlay graphicOverlay;

 /**
  * Dedicated thread and associated runnable for calling into the detector with frames, as the
  * frames become available from the camera.
  */
 private Thread processingThread;

 private final FrameProcessingRunnable processingRunnable;
 private final Object processorLock = new Object();

 private VisionImageProcessor frameProcessor;

 /**
  * Map to convert between a byte array, received from the camera, and its associated byte buffer.
  * We use byte buffers internally because this is a more efficient way to call into native code
  * later (avoids a potential copy).
* <p><b>Note:</b> uses IdentityHashMap here instead of HashMap because the behavior of an
array's
* equals, hashCode and toString methods is both useless and unexpected. IdentityHashMap
enforces
* identity ('==') check on the keys.
*/
private final IdentityHashMap<byte[], ByteBuffer> bytesToByteBuffer = new
IdentityHashMap<>();

 public CameraSource(Activity activity, GraphicOverlay overlay) {
 this.activity = activity;
```

```
 graphicOverlay = overlay;
   graphicOverlay.clear();
   processingRunnable = new FrameProcessingRunnable();
  }

  //
==========================================================================
=========================
  // Public
  //
==========================================================================
=========================

  /** Stops the camera and releases the resources of the camera and underlying detector. */
  public void release() {
   synchronized (processorLock) {
     stop();
     cleanScreen();

     if (frameProcessor != null) {
       frameProcessor.stop();
     }
   }
  }

  /**
   * Opens the camera and starts sending preview frames to the underlying detector. The preview
   * frames are not displayed.
   *
   * @throws IOException if the camera's preview texture or display could not be initialized
   */
  @RequiresPermission(Manifest.permission.CAMERA)
  public synchronized CameraSource start() throws IOException {
   if (camera != null) {
```

```
    return this;
  }

  camera = createCamera();
  dummySurfaceTexture = new SurfaceTexture(DUMMY_TEXTURE_NAME);
  camera.setPreviewTexture(dummySurfaceTexture);
  camera.startPreview();

  processingThread = new Thread(processingRunnable);
  processingRunnable.setActive(true);
  processingThread.start();
  return this;
}

/**
 * Opens the camera and starts sending preview frames to the underlying detector. The supplied
 * surface holder is used for the preview so frames can be displayed to the user.
 *
 * @param surfaceHolder the surface holder to use for the preview frames
 * @throws IOException if the supplied surface holder could not be used as the preview display
 */
@RequiresPermission(Manifest.permission.CAMERA)
public synchronized CameraSource start(SurfaceHolder surfaceHolder) throws IOException {
  if (camera != null) {
    return this;
  }

  camera = createCamera();
  camera.setPreviewDisplay(surfaceHolder);
  camera.startPreview();

  processingThread = new Thread(processingRunnable);
  processingRunnable.setActive(true);
  processingThread.start();
```

```
    return this;
  }

  /**
   * Closes the camera and stops sending frames to the underlying frame detector.
   *
   * <p>This camera source may be restarted again by calling {@link #start()} or {@link
   * #start(SurfaceHolder)}.
   *
   * <p>Call {@link #release()} instead to completely shut down this camera source and release the
   * resources of the underlying detector.
   */
  public synchronized void stop() {
    processingRunnable.setActive(false);
    if (processingThread != null) {
      try {
        // Wait for the thread to complete to ensure that we can't have multiple threads
        // executing at the same time (i.e., which would happen if we called start too
        // quickly after stop).
        processingThread.join();
      } catch (InterruptedException e) {
        Log.d(TAG, "Frame processing thread interrupted on release.");
      }
      processingThread = null;
    }

    if (camera != null) {
      camera.stopPreview();
      camera.setPreviewCallbackWithBuffer(null);
      try {
        camera.setPreviewTexture(null);
        dummySurfaceTexture = null;
        camera.setPreviewDisplay(null);
```

```
    } catch (Exception e) {
      Log.e(TAG, "Failed to clear camera preview: " + e);
    }
    camera.release();
    camera = null;
  }

  // Release the reference to any image buffers, since these will no longer be in use.
  bytesToByteBuffer.clear();
}

/** Changes the facing of the camera. */
public synchronized void setFacing(int facing) {
  if ((facing != CAMERA_FACING_BACK) && (facing != CAMERA_FACING_FRONT)) {
    throw new IllegalArgumentException("Invalid camera: " + facing);
  }
  this.facing = facing;
}

/** Returns the preview size that is currently in use by the underlying camera. */
public Size getPreviewSize() {
  return previewSize;
}

/**
 * Returns the selected camera; one of {@link #CAMERA_FACING_BACK} or {@link
 * #CAMERA_FACING_FRONT}.
 */
public int getCameraFacing() {
  return facing;
}

/**
 * Opens the camera and applies the user settings.
```

```
 *
 * @throws IOException if camera cannot be found or preview cannot be processed
 */
@SuppressLint("InlinedApi")
private Camera createCamera() throws IOException {
  int requestedCameraId = getIdForRequestedCamera(facing);
  if (requestedCameraId == -1) {
    throw new IOException("Could not find requested camera.");
  }
  Camera camera = Camera.open(requestedCameraId);

  SizePair sizePair = PreferenceUtils.getCameraPreviewSizePair(activity, requestedCameraId);
  if (sizePair == null) {
    sizePair =
        selectSizePair(
            camera,
            DEFAULT_REQUESTED_CAMERA_PREVIEW_WIDTH,
            DEFAULT_REQUESTED_CAMERA_PREVIEW_HEIGHT);
  }

  if (sizePair == null) {
    throw new IOException("Could not find suitable preview size.");
  }

  previewSize = sizePair.preview;
  Log.v(TAG, "Camera preview size: " + previewSize);

  int[] previewFpsRange = selectPreviewFpsRange(camera, REQUESTED_FPS);
  if (previewFpsRange == null) {
    throw new IOException("Could not find suitable preview frames per second range.");
  }

  Camera.Parameters parameters = camera.getParameters();
```

```
Size pictureSize = sizePair.picture;
if (pictureSize != null) {
  Log.v(TAG, "Camera picture size: " + pictureSize);
  parameters.setPictureSize(pictureSize.getWidth(), pictureSize.getHeight());
}
parameters.setPreviewSize(previewSize.getWidth(), previewSize.getHeight());
parameters.setPreviewFpsRange(
    previewFpsRange[Camera.Parameters.PREVIEW_FPS_MIN_INDEX],
    previewFpsRange[Camera.Parameters.PREVIEW_FPS_MAX_INDEX]);
// Use YV12 so that we can exercise YV12->NV21 auto-conversion logic for OCR detection
parameters.setPreviewFormat(IMAGE_FORMAT);

setRotation(camera, parameters, requestedCameraId);

if (REQUESTED_AUTO_FOCUS) {
  if (parameters
      .getSupportedFocusModes()
      .contains(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO)) {
    parameters.setFocusMode(Camera.Parameters.FOCUS_MODE_CONTINUOUS_VIDEO);
  } else {
    Log.i(TAG, "Camera auto focus is not supported on this device.");
  }
}

camera.setParameters(parameters);

// Four frame buffers are needed for working with the camera:
//
//   one for the frame that is currently being executed upon in doing detection
//   one for the next pending frame to process immediately upon completing detection
//   two for the frames that the camera uses to populate future preview images
//
// Through trial and error it appears that two free buffers, in addition to the two buffers
// used in this code, are needed for the camera to work properly.  Perhaps the camera has
```

```
  // one thread for acquiring images, and another thread for calling into user code.  If only
  // three buffers are used, then the camera will spew thousands of warning messages when
  // detection takes a non-trivial amount of time.
  camera.setPreviewCallbackWithBuffer(new CameraPreviewCallback());
  camera.addCallbackBuffer(createPreviewBuffer(previewSize));
  camera.addCallbackBuffer(createPreviewBuffer(previewSize));
  camera.addCallbackBuffer(createPreviewBuffer(previewSize));
  camera.addCallbackBuffer(createPreviewBuffer(previewSize));

  return camera;
}

/**
 * Gets the id for the camera specified by the direction it is facing. Returns -1 if no such
 * camera was found.
 *
 * @param facing the desired camera (front-facing or rear-facing)
 */
private static int getIdForRequestedCamera(int facing) {
  CameraInfo cameraInfo = new CameraInfo();
  for (int i = 0; i < Camera.getNumberOfCameras(); ++i) {
    Camera.getCameraInfo(i, cameraInfo);
    if (cameraInfo.facing == facing) {
      return i;
    }
  }
  return -1;
}

/**
 * Selects the most suitable preview and picture size, given the desired width and height.
 *
 * <p>Even though we only need to find the preview size, it's necessary to find both the preview
 * size and the picture size of the camera together, because these need to have the same aspect
```

```
 * ratio. On some hardware, if you would only set the preview size, you will get a distorted
 * image.
 *
 * @param camera the camera to select a preview size from
 * @param desiredWidth the desired width of the camera preview frames
 * @param desiredHeight the desired height of the camera preview frames
 * @return the selected preview and picture size pair
 */
public static SizePair selectSizePair(Camera camera, int desiredWidth, int desiredHeight) {
  List<SizePair> validPreviewSizes = generateValidPreviewSizeList(camera);

  // The method for selecting the best size is to minimize the sum of the differences between
  // the desired values and the actual values for width and height.  This is certainly not the
  // only way to select the best size, but it provides a decent tradeoff between using the
  // closest aspect ratio vs. using the closest pixel area.
  SizePair selectedPair = null;
  int minDiff = Integer.MAX_VALUE;
  for (SizePair sizePair : validPreviewSizes) {
    Size size = sizePair.preview;
    int diff =
        Math.abs(size.getWidth() - desiredWidth) + Math.abs(size.getHeight() - desiredHeight);
    if (diff < minDiff) {
      selectedPair = sizePair;
      minDiff = diff;
    }
  }

  return selectedPair;
}

/**
 * Stores a preview size and a corresponding same-aspect-ratio picture size. To avoid distorted
 * preview images on some devices, the picture size must be set to a size that is the same aspect
 * ratio as the preview size or the preview may end up being distorted. If the picture size is
```

```
 * null, then there is no picture size with the same aspect ratio as the preview size.
 */
public static class SizePair {
  public final Size preview;
  @Nullable public final Size picture;

  SizePair(Camera.Size previewSize, @Nullable Camera.Size pictureSize) {
    preview = new Size(previewSize.width, previewSize.height);
    picture = pictureSize != null ? new Size(pictureSize.width, pictureSize.height) : null;
  }

  public SizePair(Size previewSize, @Nullable Size pictureSize) {
    preview = previewSize;
    picture = pictureSize;
  }
}

/**
 * Generates a list of acceptable preview sizes. Preview sizes are not acceptable if there is not
 * a corresponding picture size of the same aspect ratio. If there is a corresponding picture size
 * of the same aspect ratio, the picture size is paired up with the preview size.
 *
 * <p>This is necessary because even if we don't use still pictures, the still picture size must
 * be set to a size that is the same aspect ratio as the preview size we choose. Otherwise, the
 * preview images may be distorted on some devices.
 */
public static List<SizePair> generateValidPreviewSizeList(Camera camera) {
  Camera.Parameters parameters = camera.getParameters();
  List<Camera.Size> supportedPreviewSizes = parameters.getSupportedPreviewSizes();
  List<Camera.Size> supportedPictureSizes = parameters.getSupportedPictureSizes();
  List<SizePair> validPreviewSizes = new ArrayList<>();
  for (Camera.Size previewSize : supportedPreviewSizes) {
    float previewAspectRatio = (float) previewSize.width / (float) previewSize.height;
```

```
      for (Camera.Size pictureSize : supportedPictureSizes) {
        float pictureAspectRatio = (float) pictureSize.width / (float) pictureSize.height;
        if (Math.abs(previewAspectRatio - pictureAspectRatio) <
ASPECT_RATIO_TOLERANCE) {
          validPreviewSizes.add(new SizePair(previewSize, pictureSize));
          break;
        }
      }
    }

    // If there are no picture sizes with the same aspect ratio as any preview sizes, allow all
        // still account for it.
    if (validPreviewSizes.size() == 0) {
      Log.w(TAG, "No preview sizes have a corresponding same-aspect-ratio picture size");
      for (Camera.Size previewSize : supportedPreviewSizes) {
        .
        validPreviewSizes.add(new SizePair(previewSize, null));
      }
    }

    return validPreviewSizes;
  }

  /**
   * Selects the most suitable preview frames per second range, given the desired frames per
second.
   *
   * @param camera the camera to select a frames per second range from
   * @param desiredPreviewFps the desired frames per second for the camera preview frames
   * @return the selected preview frames per second range
   */
  @SuppressLint("InlinedApi")
  private static int[] selectPreviewFpsRange(Camera camera, float desiredPreviewFps) {
```

```
    // The camera API uses integers scaled by a factor of 1000 instead of floating-point frame
    // rates.
    int desiredPreviewFpsScaled = (int) (desiredPreviewFps * 1000.0f);

    // Selects a range with whose upper bound is as close as possible to the desired fps while its
    // lower bound is as small as possible to properly expose frames in low light conditions. Note
    // that this may select a range that the desired value is outside of. For example, if the
    // desired frame rate is 30.5, the range (30, 30) is probably more desirable than (30, 40).
    int[] selectedFpsRange = null;
    int minUpperBoundDiff = Integer.MAX_VALUE;
    int minLowerBound = Integer.MAX_VALUE;
    List<int[]> previewFpsRangeList = camera.getParameters().getSupportedPreviewFpsRange();
    for (int[] range : previewFpsRangeList) {
      int upperBoundDiff =
          Math.abs(desiredPreviewFpsScaled -
range[Camera.Parameters.PREVIEW_FPS_MAX_INDEX]);
      int lowerBound = range[Camera.Parameters.PREVIEW_FPS_MIN_INDEX];
      if (upperBoundDiff <= minUpperBoundDiff && lowerBound <= minLowerBound) {
        selectedFpsRange = range;
        minUpperBoundDiff = upperBoundDiff;
        minLowerBound = lowerBound;
      }
    }
    return selectedFpsRange;
  }

  /**
   * Calculates the correct rotation for the given camera id and sets the rotation in the
   * parameters. It also sets the camera's display orientation and rotation.
   *
   * @param parameters the camera parameters for which to set the rotation
   * @param cameraId the camera id to set rotation based on
   */
  private void setRotation(Camera camera, Camera.Parameters parameters, int cameraId) {
```

```
    WindowManager windowManager = (WindowManager)
activity.getSystemService(Context.WINDOW_SERVICE);
    int degrees = 0;
    int rotation = windowManager.getDefaultDisplay().getRotation();
    switch (rotation) {
      case Surface.ROTATION_0:
        degrees = 0;
        break;
      case Surface.ROTATION_90:
        degrees = 90;
        break;
      case Surface.ROTATION_180:
        degrees = 180;
        break;
      case Surface.ROTATION_270:
        degrees = 270;
        break;
      default:
        Log.e(TAG, "Bad rotation value: " + rotation);
    }

    CameraInfo cameraInfo = new CameraInfo();
    Camera.getCameraInfo(cameraId, cameraInfo);

    int displayAngle;
    if (cameraInfo.facing == CameraInfo.CAMERA_FACING_FRONT) {
      this.rotationDegrees = (cameraInfo.orientation + degrees) % 360;
      displayAngle = (360 - this.rotationDegrees) % 360; // compensate for it being mirrored
    } else { // back-facing
      this.rotationDegrees = (cameraInfo.orientation - degrees + 360) % 360;
      displayAngle = this.rotationDegrees;
    }
    Log.d(TAG, "Display rotation is: " + rotation);
    Log.d(TAG, "Camera face is: " + cameraInfo.facing);
```

```
  Log.d(TAG, "Camera rotation is: " + cameraInfo.orientation);
  // This value should be one of the degrees that ImageMetadata accepts: 0, 90, 180 or 270.
  Log.d(TAG, "RotationDegrees is: " + this.rotationDegrees);

  camera.setDisplayOrientation(displayAngle);
  parameters.setRotation(this.rotationDegrees);
}

/**
 * Creates one buffer for the camera preview callback. The size of the buffer is based off of the
 * camera preview size and the format of the camera image.
 *
 * @return a new preview buffer of the appropriate size for the current camera settings
 */
@SuppressLint("InlinedApi")
private byte[] createPreviewBuffer(Size previewSize) {
  int bitsPerPixel = ImageFormat.getBitsPerPixel(IMAGE_FORMAT);
  long sizeInBits = (long) previewSize.getHeight() * previewSize.getWidth() * bitsPerPixel;
  int bufferSize = (int) Math.ceil(sizeInBits / 8.0d) + 1;

  // Creating the byte array this way and wrapping it, as opposed to using .allocate(),
  // should guarantee that there will be an array to work with.
  byte[] byteArray = new byte[bufferSize];
  ByteBuffer buffer = ByteBuffer.wrap(byteArray);
  if (!buffer.hasArray() || (buffer.array() != byteArray)) {
    // I don't think that this will ever happen.  But if it does, then we wouldn't be
    // passing the preview content to the underlying detector later.
    throw new IllegalStateException("Failed to create valid buffer for camera source.");
  }

  bytesToByteBuffer.put(byteArray, buffer);
  return byteArray;
}
```

```
  //
================================================================================
========================
  // Frame processing
  //
================================================================================
========================

  /** Called when the camera has a new preview frame. */
  private class CameraPreviewCallback implements Camera.PreviewCallback {
    @Override
    public void onPreviewFrame(byte[] data, Camera camera) {
      processingRunnable.setNextFrame(data, camera);
    }
  }

  public void setMachineLearningFrameProcessor(VisionImageProcessor processor) {
    synchronized (processorLock) {
      cleanScreen();
      if (frameProcessor != null) {
        frameProcessor.stop();
      }
      frameProcessor = processor;
    }
  }

  /**
   * This runnable controls access to the underlying receiver, calling it to process frames when
   * available from the camera. This is designed to run detection on frames as fast as possible
   * (i.e., without unnecessary context switching or waiting on the next frame).
   *
   * <p>While detection is running on a frame, new frames may be received from the camera. As
these
   * frames come in, the most recent frame is held onto as pending. As soon as detection and its
```

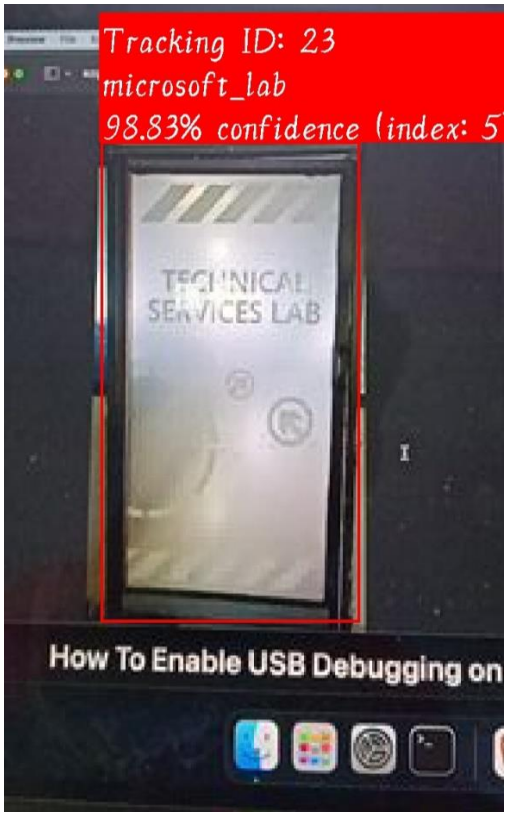
```
   * associated processing is done for the previous frame, detection on the mostly recently
received
   * frame will immediately start on the same thread.
   */
  private class FrameProcessingRunnable implements Runnable {

    // This lock guards all of the member variables below.
    private final Object lock = new Object();
    private boolean active = true;

    // These pending variables hold the state associated with the new frame awaiting processing.
    private ByteBuffer pendingFrameData;

    FrameProcessingRunnable() {}

    /** Marks the runnable as active/not active. Signals any blocked threads to continue. */
    void setActive(boolean active) {
      synchronized (lock) {
        this.active = active;
        lock.notifyAll();
      }
    }

    /**
     * Sets the frame data received from the camera. This adds the previous unused frame buffer
(if
     * present) back to the camera, and keeps a pending reference to the frame data for future use.
     */
    @SuppressWarnings("ByteBufferBackingArray")
    void setNextFrame(byte[] data, Camera camera) {
      synchronized (lock) {
        if (pendingFrameData != null) {
          camera.addCallbackBuffer(pendingFrameData.array());
          pendingFrameData = null;
```

```
      }

      if (!bytesToByteBuffer.containsKey(data)) {
        Log.d(
            TAG,
            "Skipping frame. Could not find ByteBuffer associated with the image "
                + "data from the camera.");
        return;
      }

      pendingFrameData = bytesToByteBuffer.get(data);

      // Notify the processor thread if it is waiting on the next frame (see below).
      lock.notifyAll();
    }
  }

  /**
   * As long as the processing thread is active, this executes detection on frames continuously.
   * The next pending frame is either immediately available or hasn't been received yet. Once it
   * is available, to transfer the frame info to local variables and run detection on that frame.
   * It immediately loops back for the next frame without pausing.
   *
   * <p>If detection takes longer than the time in between new frames from the camera, this will
   * mean that this loop will run without ever waiting on a frame, avoiding any context switching
   * or frame acquisition time latency.
   *
   * <p>If you find that this is using more CPU than you'd like, you should probably decrease
the
   * FPS setting above to allow for some idle time in between frames.
   */
  @SuppressLint("InlinedApi")
  @SuppressWarnings({"GuardedBy", "ByteBufferBackingArray"})
  @Override
```

```
public void run() {
  ByteBuffer data;

  while (true) {
    synchronized (lock) {
      while (active && (pendingFrameData == null)) {
        try {
          lock.wait();
        } catch (InterruptedException e) {
          Log.d(TAG, "Frame processing loop terminated.", e);
          return;
        }
      }

      if (!active) {
        // Exit the loop once this camera source is stopped or released.  We check
        // this here, immediately after the wait() above, to handle the case where
        // setActive(false) had been called, triggering the termination of this
        // loop.
        return;
      }

      // Hold onto the frame data locally, so that we can use this for detection
      // below.  Need to clear pendingFrameData to ensure that this buffer isn't
      // recycled back to the camera before  done using that data.
      data = pendingFrameData;
      pendingFrameData = null;
    }

    // The code below needs to run outside of synchronization, because this will allow
    // the camera to add pending frame(s) while running detection on the current
    // frame.

    try {
```

```
        synchronized (processorLock) {
          frameProcessor.processByteBuffer(
              data,
              new FrameMetadata.Builder()
                  .setWidth(previewSize.getWidth())
                  .setHeight(previewSize.getHeight())
                  .setRotation(rotationDegrees)
                  .build(),
              graphicOverlay);
        }
      } catch (Exception t) {
        Log.e(TAG, "Exception thrown from receiver.", t);
      } finally {
        camera.addCallbackBuffer(data.array());
      }
    }
  }

  /** Cleans up graphicOverlay and child classes can do their cleanups as well . */
  private void cleanScreen() {
    graphicOverlay.clear();
  }
}
```

5.2 RESULT


Tracking ID: 23
microsoft_lab
98.83% confidence (index: 5
How To Enable USB Debugging on


Tracking ID: 32
oracle_lab
98.44% confidence (index:
How To Enable USB Debugging on R

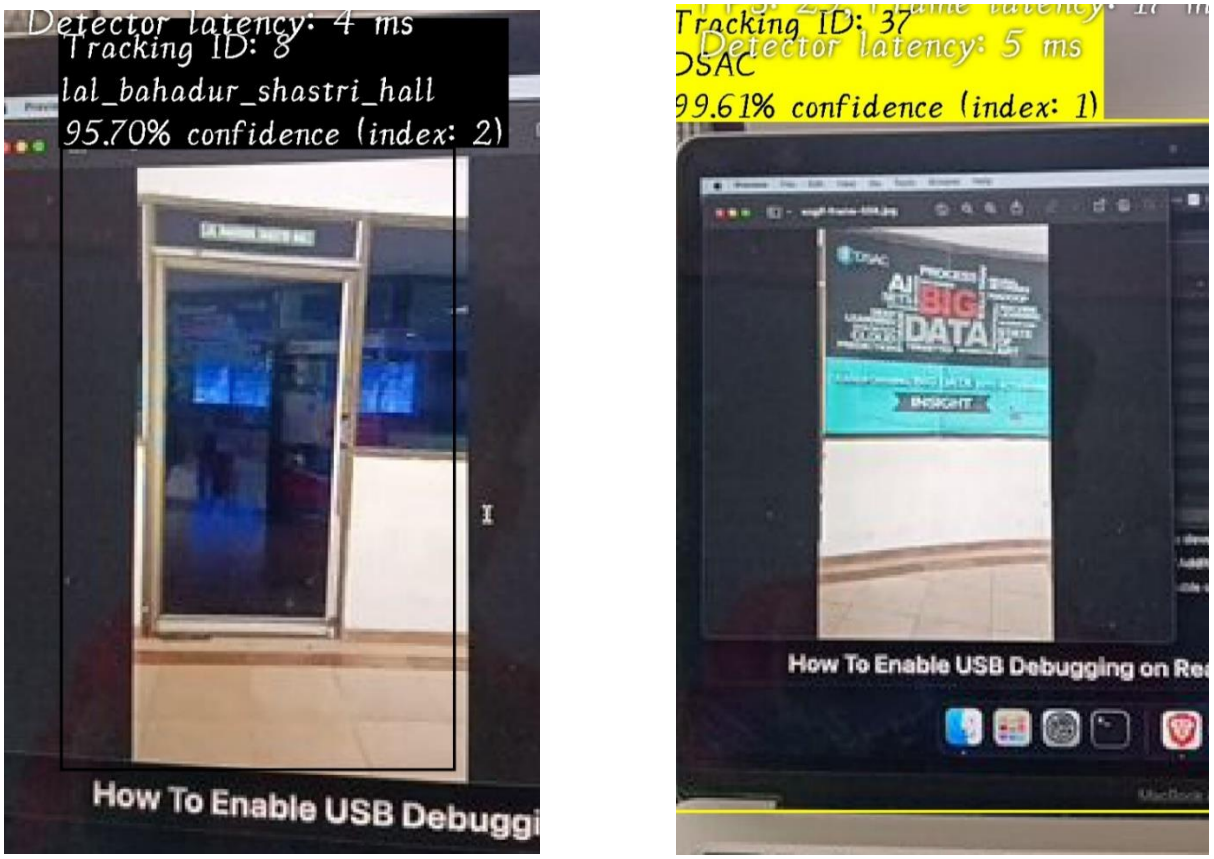


Fig. No. 5.1 Examples of laboratory detection

DESCRIPTION:

In this output it shows the four different places those are microsoft lab, Oracle lab, lal bahadur shastri hall, DSAC inside the campus. We integrated our ML model into the application and use the camera in the app to detect these places.

CHAPTER 6

CONCLUSION

In this project, need to proposed a novel framework to tackle the problem of object detection in optical RSIs. The novelties that distinguish the proposed work from previous works lie in two major aspects. First, instead of using traditional supervised or semi supervised learning methodology, this project developed a WSL framework that can substantially reduce the human labor of annotating training data while achieving the outstanding performance. Second, need to developed a deep network to learn high-level features in an unsupervised manner, which offers a more powerful descriptor to capture the structural information of objects in RSIs. It thus can improve the object detection performance further. Experiments on three different RSI data sets have demonstrated the effectiveness of the proposed work. Our future work will focus on three directions.

It is clear from the combined findings of the three data sets that the supervised baseline method performs better on these datasets because of the potent high-level feature representation created by DBM. The proposed WSL algorithm outperforms the previous WSL-based target detection method and approaches the fully supervised baseline method in terms of detection performance. This is made possible by the Bayesian framework's ability to produce accurate initial training examples and the iterative training scheme's ability to gradually improve the object detector. Additionally, it appears that the weakly supervised detector outperforms the other two state-of-the-art supervised approaches when the high-level feature representation and the suggested WSL framework are combined.

By contrasting the proposed weakly supervised object detector with various supervised-learning-based approaches and one existing WSL-based method, assessed the performance of the suggested detector. The proposed strategy was first contrasted with the WSL-based method to used the identical experimental circumstances for comparison, including the same feature representation created by DBM, the same sliding window design, and the same testing image.

CHAPTER 7

FUTURE SCOPE

In this project currently we use the static data set, which means the data will not change anytime. In future we will move this project into the cloud. Whenever we try to figure out any place, it will scan and add to the dataset on the cloud. Because of this process the accuracy of the model will increase. So we can easily identify any places easily. Now we only have to detect few places inside the campus. In future we will increase our data set and include each and every place inside campus to detect everything.

REFERENCES

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once:Unified, real-time object detection,’’ in Proc. IEEE Conf. Comput. Vis.Pattern Recognit. (CVPR), Jun. 2016, pp. 779–788.

[2] N. Wojke, A. Bewley, and D. Paulus,‘‘Simple online and realtime tracking with a deep association metric,’’ in Proc. IEEE Int. Conf. Image Process.(ICIP), Sep. 2017, pp. 3645–3649, doi: 10.1109/ICIP.2017.8296962

[3] F. H. Zunjani, S. Sen, H. Shekhar, A. Powale, D. Godnaik, and G. C. Nandi, ‘‘Intent-based object grasping by a robot using deep learning,’’ in Proc. IEEE 8th Int. Advance Comput. Conf. (IACC), Dec. 2018,pp.246251,doi:10.1109/IADCC.2018.8692134

[4] Z. Wang, Z. Zhao, and F. Su, ‘‘Real-time tracking with stabilizedframe,’’ in Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW), Jun. 2020, pp. 4431–4438, doi: 10.1109/CVPRW50498.2020.00522.

[5] Z. Y. Chan and S. A. Suandi, ‘‘City tracker: Multiple object tracking inurban mixed traffic scenes,’’ in Proc. IEEE Int. Conf. Signal Image Process. Appl. (ICSIPA), Sep. 2019, pp. 335–339, doi: 10.1109/ICSIPA45851.2019.8977783.

[6] M. I. H. Azhar, F. H. K. Zaman, N. M. Tahir, and H. Hashim, ‘‘People tracking system using DeepSORT,’’ in Proc. 10th IEEE Int. Conf.Control Syst., Comput. Eng.(ICCSCE),Aug.2020,pp.137-141,doi:10.1109/ICCSCE50387.2020.9204956.

[7] J. Jin, X. Li, X. Li, and S. Guan, ‘‘Online multi-object tracking with Siamese network and optical flow,’’ in Proc. IEEE 5th Int. Conf. Image,Vis. Comput. (ICIVC), Jul. 2020, pp. 193–198, doi: 10.1109/ICIVC50857.2020.9177480.

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, ‘‘You only look once:Unified, real-time object detection,’’ in Proc. IEEE Conf. Comput. Vis.Pattern Recognit. (CVPR), Jun. 2016, pp. 779–788.

[9] D. G. Lowe, ‘‘Distinctive image features from scale-invariant keypoints,’’ Int. J. Comput. Vis., vol. 60, no. 2, 2004, pp. 91–110,doi:10.1023/B:VISI.0000029664.99615.94.

[10] H. Bay, T. Tuytelaars, and L. V. Gool, ‘‘SURF: Speeded up robust features,’’ in Proc. 9th Eur. Conf. Comput. Vis., May 2006, pp. 404–417.

[11] N. Cristianini and E. Ricci, ‘‘Support vector machines,’’ in Encyclopedia of Algorithms, M. Y. Kao, Ed. Boston, MA, USA: Springer, 2008, doi:10.1007/978-0-387-30162-4_415.

[12] S. Kustrin and R. Beresford, ‘‘Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research,’’J. Pharmaceutical Biomed. Anal., vol. 22, no. 5, pp. 717–727, 2000, doi:10.1016/S0731-7085(99)00272-1.

[13] L. Shao, D. Wu, and X. Li, “Learning deep and wide: A spectral method for learning deep networks,” IEEE Trans. Neural Netw. Learn. Syst., vol. 25, no. 12, Dec. 2014, pp. 2303–2308.

[14] X. Bai, H. Zhang, and J. Zhou, “VHR object detection based on structural feature extraction and query expansion,” IEEE Trans. Geosci. Remote Sens., vol. 52, no. 10, Oct. 2014, pp. 1–13.

[15] I. Dópido et al., “Semisupervised self-learning for hyperspectral image classification,” IEEE Trans. Geosci. Remote Sens., vol. 51, no. 7, Jul. , pp. 4032–4044

[16] W. Liao, A. Pizurica, P. Scheunders, W. Philips, and Y. Pi, “Semisupervised local discriminant analysis for feature extraction in hyperspectral images,” IEEE Trans. Geosci. Remote Sens., vol. 51, no. 1, Jan. 2013, pp. 184–198.

[17] E. Pasolli, F. Melgani, N. Alajlan, and N. Conci, “Optical image classification: A groundtruth design framework,” IEEE Trans. Geosci. Remote Sens., vol. 51, no. 6, Jun. 2013, pp. 3580–3597.

[18] F. Zheng et al., “A semi-supervised approach for dimensionality reduction with distributional similarity,” Neurocomputing, vol. 103, Mar. 2013, pp. 210–221.

[19] G. Jun and J. Ghosh, “Semisupervised learning of hyperspectral data with unknown landcover classes,” IEEE Trans. Geosci. Remote Sens., vol. 51, no. 1, Jan. 2013, pp. 273–282.

[20] B. Alexe, T. Deselaers, and V. Ferrari, “Measuring the objectness of image windows,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 34, no. 11, Nov. 2012, pp. 2189–2202.

[21] T. Deselaers, B. Alexe, and V. Ferrari, “Weakly supervised localization and learning with generic knowledge” Int. J. Comput. Vis., vol. 100, no. 3, Dec. 2012, pp. 275–293.

[22] P. Siva, C. Russell, and T. Xiang, “In defense of negative mining for annotating weakly labeled data,” in Proc. Eur. Conf. Comput. Vis., pp. 594–608.

[23] P. Siva and T. Xiang, “Weakly supervised object detector learning with model drift detection,” in Proc. IEEE Int. Conf. Comput. Vis., 2011, pp. 343–350.

[24] F. Zhu, and L. Shao, “Weakly-supervised cross-domain dictionary learning for visual recognition” Int. J. Comput. Vis., vol. 109, no. 1–2, Aug. , pp. 42–59

[25] D. Zhang et al., “Weakly supervised learning for target detection in remote sensing images,” IEEE Geosci. Remote Sens. Lett., vol. 12, no. 4, Apr. 2015. 19, pp. 701–705 .

[26] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, “Object detection with discriminatively trained part-based models,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 32, no. 9, Sep. 2010, pp. 1627–1645.

[27] B. Sirmacek and C. Unsalan, “Urban-area and building detection using SIFT keypoints and graph theory,” IEEE Trans. Geosci. Remote Sens., vol. 47, no. 4, Apr. 2009, pp. 1156–1167.

[28] S. Xu, T. Fang, D. Li, and S.Wang, “Object classification of aerial images with bag-ofvisual words,” IEEE Geosci. Remote Sens. Lett., vol. 7, no. 2, Apr. 2010, pp. 366–370.

[29] D. G. Lowe, “Distinctive image features from scale-invariant keypoints” Int. J. Comput. Vis., vol. 60, no. 2, Nov. 2004, pp. 91–110.

[30] Y. Yang and S. Newsam, “Geographic image retrieval using local invariant features,” IEEE Trans. Geosci. Remote Sens., vol. 51, no. 2, Feb. 2013, pp. 818–832.