JOURNAL OF AERONAUTICAL MATERIALS
ISSN: 1005-5053

Vol. 43, Issue-01, 2023
pp. 459-466

ARTIFICIAL INTELLIGENCE BASED OBJECT OBSERVATION BY USING CAMERA IN AERONAUTICAL SECURITY SURVEILLANCE SYSTEM (CASSS)

Dr Velumani P S
Professor, Department of Computer Science and Electronics, University of Science and Technology Meghalaya, Techno City, Killing Road, Baridua 9th Mile, Ri-Bhoi Meghalaya-793 101, India

Dr. Ajay khunteta
Professor, Faculty of Computer science & Engineering, Poornima University, Jaipur India

Prof. (Dr.) Abdul Matin
Department of Sociology, University of Science and Technology Meghalaya, Techno City, Killing Road, Baridua 9th Mile, Ri-Bhoi, Meghalaya-793 101, India

Abstract:
Machine learning is the sub area in the field of AI which indicates that the system can have the capability to learn the input data from the environment. Once the given training to certain machines it can classify the accurate & in accurate data for the end users. At present human being needs to have more knowledge to program & execute the system as per our requirement. If machine can learn directly which will reduce the work load and entire activity of human will be replaced by machine and all the electro mechanical system will be automated. This leads to the concept of machines become more knowledge and involvement in area will be huge accuracy of data and information calculation. Here the proposed model is a Deep learning model (DLM) to predict the object observation in Aeronautical security Surveillance by using different supervisory models and algorithms. The AI field becomes more and more popular in 21st century. In this the field of computer application, cyber security, automation & mathematical model logics, questions have been raised about the specific concept of Artificial Intelligence .The day when computers are getting trained a new life has been started to the human being and some of the applications like face recognizing in Aeronautical System, Automatic translation & robotic control systems will be prepared with full accuracy. There are different types of machine learning algorithms, feature selection methods; dimension reduction & delay of wireless data will be decided for camera based observation systems.
KEYWORDS: Machine Learning (ML), Firebase Machine Learning's Model (FMLM), Deep Learning and Google Mobile Vision(GMV)

I.INTRODUCTION

An object classification and observation is a straight forward, but some of the differences between object localization and object detection is getting varied, whenever all three tasks may be equally referred to as object recognition pattern algorithm. Image classification involves assigning a classes name to a particular image, whereas object localization involving in one or more objects of an image. Object detection & observation is most challenging task and combines these two tasks are interesting in AI Based Algorithms of interest in the image and assigns them a class label. Together, all of these problems are referred to as object recognition and observation by using Deep Learning and Google Mobile Vision (GMV).

II. METHODOLOGY:

As we know very well the types of learning are supervised & unsupervised learning. The following method is used to develop the synchronization based dialog control method to present the Application layer level in AI developed systems. To understand the object observation which is said that a “person” is considered as an “object” and will be detected by using several analysis methods like face detection, skeleton detection & predestination detachment. The understand of computer vision problem, object detection is able to provide valuable information for semantic understanding of images of videos, related to many applications[6,7,8] including image classification, human behavioral analysis & autonomous driving. Due to the varieties of different angle view of an object, poses, occlusions& lighting conditions, it is difficult to perfectly accomplish object detection with an additional object location task [8].

Google Mobile Vision

Object Detector Settings	
Detection mode	STREAM_MODE (default) | SINGLE_IMAGE_MODE In STREAM_MODE (default), the object detector runs with low latency, but might produce incomplete results (such as unspecified bounding boxes or category labels) on the first few invocations of the detector. Also, in STREAM_MODE, the detector assigns tracking IDs to objects, which we can use to track objects across frames. Use this mode when we want to track objects, or when low latency is important, such as when processing video streams in real time. In SINGLE_IMAGE_MODE, the object detector returns the result after the object's bounding box is determined. If we also enable classification it returns the result after the bounding box and category label are both available. As a consequence, detection latency is potentially higher. Also, in SINGLE_IMAGE_MODE, tracking IDs are not assigned. Use this mode if latency isn't critical and we don't want to deal with partial results.
Detect and track multiple objects	false (default) | true Whether to detect and track up to five objects or only the most prominent object (default).
Classify objects	false (default) | true Whether or not to classify detected objects into coarse categories. When enabled, the object detector classifies objects into the following categories: fashion goods, food, home goods, places, and plants.

Google Mobile Vision (GMV) was deprecated and we are asking developers to migrate to the ML Kit SDK which is its replacement. Migrating to the new SDK ensures we get the best performance, stability and latest features. In addition, ML Kit provides additional ML-powered APIs, not only for Vision, but also Natural Language use cases.

If we are using Mobile Vision’s barcode scanning, text recognition or face detection APIs in wer app today, please migrate to the new ML Kit SDK, by following the ML Kit migration guide for Android and the ML Kit migration guide for iOS.

To create custom image classification models from own training data with AutoML Vision Edge. With ML Kit's image labeling APIs we can detect and extract information about entities in an image across a broad group of categories. The default image labeling model can identify general objects, places, activities, animal species, products, and more ML Kit's base image labeling model is built for general-purpose use. It's trained to recognize 400 categories that describe the most commonly-found objects in photos. The app might need a specialized image classification model that recognizes a narrower number of categories in more detail, such as a model that distinguishes between species of flowers or types of food.

This API lets we tailor to a particular use case by supporting custom image classification models from a wide range of sources. Please refer to. Custom models can be bundled with app or dynamically downloaded from the cloud using Firebase Machine Learning's Model(FMLM) deployment service.

Image Labeling uses bilinear image scaling and stretching to adjust the input image size and aspect ratio so that they fit the requirements of the underlying model. Two techniques will be used. One is Non Adaptive & another is adaptive.

To tackle common challenges, like lighting, focusing some easy-to-use turn-key APIs are used for more custom use-cases .The proposed algorithm is integrate it in the app and deploy it in production.

STAGES OF ALGORITHMS:

1. Dependencies API.
2. Configure the object detector.
3. Prepare the input image.
4. Process the image.
5. Get Information about observed objects.

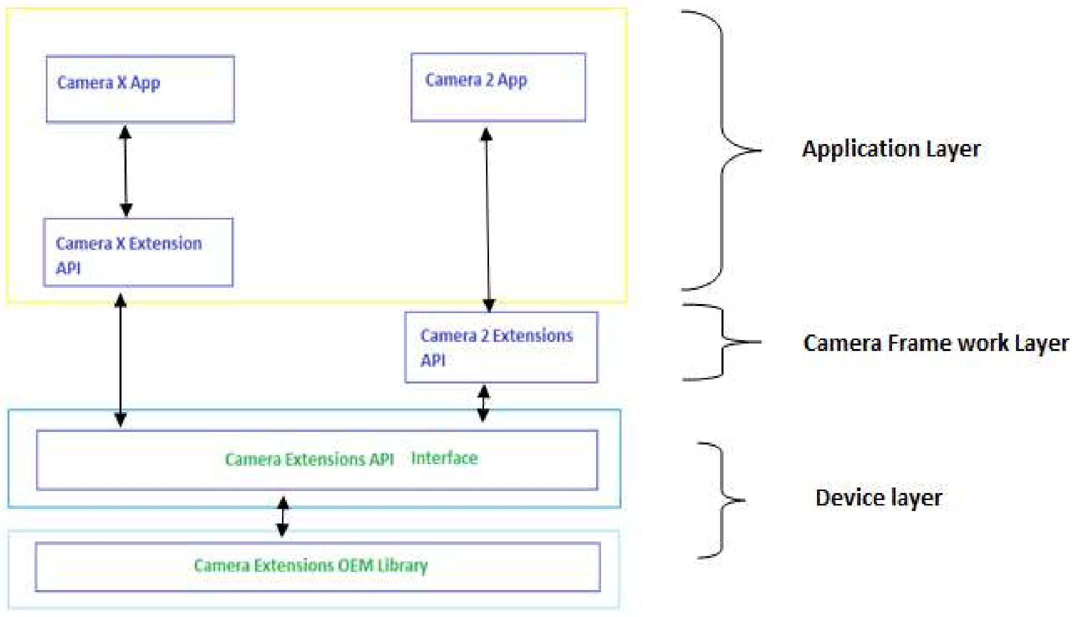


Fig 1: Camera Layer Architecture

Stage 1: Dependencies API

API requires Android API higher levels.

1.In project-level Lbuild.gradlefile, make sure to include Google's Maven repository in both wer buildscript and allprojects sections.

2.Add the dependencies for the ML Kit Android libraries to wer module's app-level gradle file, which is usually app/build.gradle:

Algorithm :

```
dependencies {
  // ...
  implementation 'com.google.mlkit:object-detection'
}
```

Stage-2: Configure the object detector

To detect and track objects, first create an instance of ObjectDetector and optionally specify any detector settings that we want to change from the default.

Configure the object detector for wer use case with an ObjDetectOptions object. We can change the following settings:

The object detection and tracking API is optimized for these two core use cases:

1. Live detection and tracking of the most prominent object in the camera viewfinder.
2. The detection of multiple objects from a static image

Stage 3: Prepare the input image

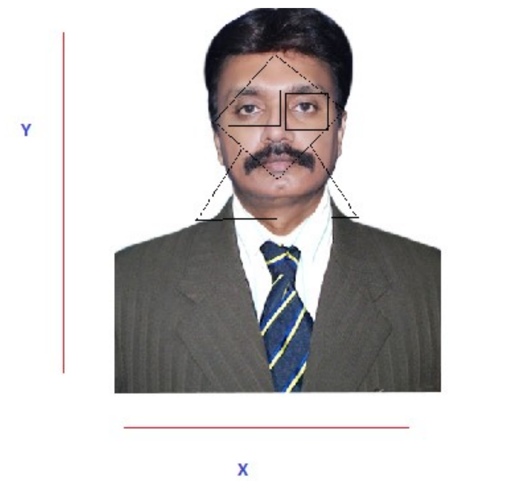
	DATA & PROCESSION ON IMAGE	
	Data: << >> Face:<< >> Gender:<< >> Date:<< >>	Capturing Extracting Comparing Matching

Fig 2: Face detection- Eye, Lip pattern matching.

To detect and track objects, pass images to the Object Detector instance's process () method. The object detector runs directly from a Bitmap, NV21 Byte Buffer or a YUV media. Image.

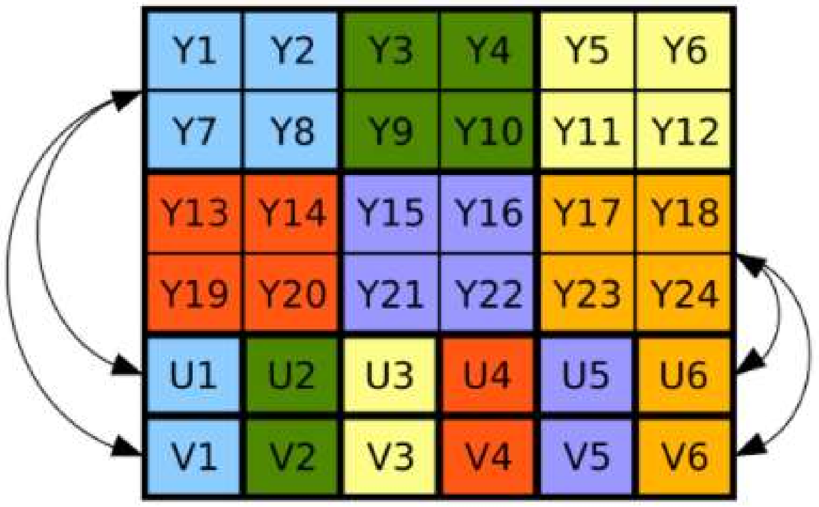


Fig 3: YUV –Single format method

Constructing an Input Image from those sources are recommended if we have direct access to one of them. If we construct an Input Image from other sources, we will handle the conversion internally for we and it might be less efficient. Pixel in Y there is a corresponding value in U/V planes with 2:1 ratio (2 pixel Y correspond to one pixel U/V).
For each frame of video or image in a sequence, do the following:
We can create an Input Image object from different sources, each is explained below:

- Using a media.Image
- To create an InputImage object from a media.Image object, such as when we capture an image from a device's camera, pass the media.Image object and the image's rotation to InputImage.fromMediaImage ().
- If we use the CameraX library, the OnImageCapturedListener and ImageAnalysis.Analyzer classes calculate the rotation value.

In case of camera library not used that gives the image's rotation degree, we can calculate it from the device's rotation degree and the orientation of camera sensor in the device:

Then, pass the media. Image object and the rotation degree value to InputImage.fromMediaImage ():

Bounding box	A Rect that indicates the position of the object in the image.	
Tracking ID	An integer that identifies the object across images. Null in SINGLE_IMAGE_MODE.	
Labels	Label description	The label's text description. It will be one of the String defined in PredefinedCategory.
	Label index	The label's index among all the labels supported by the classi be one of the integer constants defined in PredefinedCategor
	Label confidence	The confidence value of the object classification.

Stage 4. Process the image

Pass the image to the process () method:

Algorithm:

```
objectDetector.process(image)
    .addOnSuccessListener { detectedObjects ->
        // Task completed successfully
        // ...
    }
    .addOnFailureListener { e ->
        // Task failed with an exception
        // ...
    }
```

Stage 5. Get information about detected objects

If the call to process() succeeds, a list of DetectedObjects is passed to the success listener.

Each DetectedObject contains the following properties:

Algorithm :

```
for (detectedObject in detectedObjects) {
  val boundingBox = detectedObject.boundingBox
  val trackingId = detectedObject.trackingId
  for (label in detectedObject.labels) {
    val text = label.text
    if (PredefinedCategory.OC == text) {
      ...
    }
    val index = label.index
    if (PredefinedCategory.OC_INDEX == index) {
      ...
    }
    val confidence = label.confidence
  }
}
```

Note: OC –denotes object classifications of fashion goods, food, home goods, places, and plants.

Ensuring a great user experience

Successful object detection depends on the object's visual complexity. In order to be detected, objects with a small number of visual features might need to take up a larger part of the image. We should provide users with guidance on capturing input that works well with the kind of objects we want to detect.

When we use classification, if we want to detect objects that don't fall cleanly into the supported categories, implementation for special handling for unknown objects is necessary. In such cases sub sets of all objects needs to be observed properly.

III.RESULTS:

In Real time application

- While using the Camera or camera2 API, throttle calls to the detector. If a new video frame becomes available while the detector is running, drop the frame. See the VisionProcessorBase class in the quickstart sample app for an example.
- While using the CameraX API, be sure that backpressure strategy is set to its default value ImageAnalysis.STRATEGY_KEEP_ONLY_LATEST. This guarantees only one image will be delivered for analysis at a time. If more images are produced when the analyzer is busy, they will be dropped automatically and not queued for delivery. Once the image being analyzed is closed by calling ImageProxy.close(), the next latest image will be delivered.

- When we use the output of the detector to overlay graphics on the input image, first get the result from ML Kit, then render the image and overlay in a single step. This renders to the display surface only once for each input frame. See the CameraSourcePreview and GraphicOverlay classes in the quickstart sample app for an example.

IV.CONCLUSION:

In this paper, we have presented, an object observation using the object detector settings & methods by using camera API in Aeronautical Security Surveillance system presented and further processing a base type that provides object-being-observed, observer, and observed epoch properties to the objects that derive from it. The performance Improvement parameter helps to observe the object more clearly. In future it will help to develop generic algorithms for object observation.

V. REFERENCES:

D. Vernon, G. Metta, and G. Sandini, ‘‘A survey of artificial cognitive systems: Implications for the autonomous development of mental capabilities in computational agents,’’ IEEE Trans. Evol. Computer., vol. 11, no. 2,pp. 151–180, Apr. 2007.

D. Kirsh, ‘‘Thinking with external representations,’’ Ai Soc., vol. 25, no. 4,pp. 441–454, 2010

V. Khodadadi et al. Application Of Ants Colony System For Bankruptcy Prediction Of Companies Listed In Tehran Stock Exchange , Business Intelligence Journal, 2010.

Ming-Yuan Leon Li and Peter Miu. A hybrid bankruptcy prediction model with dynamic loadings on accounting-ratio-based and market-based information: A binary quantile regression approach. Empirical Finance, 17:818–833, 2010.

W. Beaver, Financial Ratios As Predictors Of Failure, Journal Of Accounting Research 5: 71-111,1996.

E. Altman,. Financial Ratios, Discriminant Analysis And The Prediction Of Corporate Bankruptcy, The Journal Of Finance 23(4): 589-609,1968.

Evaluation of Machine Learning Algorithms in Artificial Intelligence,International Journal of Computer Science and Mobile Computing,Vol. 4, Issue. 5, May 2015, pg.278 – 286

Smith BM, Yao X, Chen KS, Kirby ED A Larger Social Network Enhances Novel Object Location Memory and Reduces Hippocampal Microgliosis in Aged Mice. Frontiers in Aging Neuroscience. 10 (142), 1–16 (2018).