

A
Minor Project
On
“Object Detection Web Application”



Submitted to
CHHATTISGARH SWAMI VIVEKANAND TECHNICAL UNIVERSITY
BHILAI (INDIA)

In partial fulfilment of requirement for the award of degree

Of
Bachelor of Technology
In
INFORMATION TECHNOLOGY
by

Sourav Kumar Singh



DEPARTMENT OF INFORMATION TECHNOLOGY
SHRI SHANKARACHARYA TECHNICAL CAMPUS
JUNWANI BHILAI (C.G) 490020

SESSION:2022-2023

DECLARATION

We the undersigned solemnly declare that the report of the project work entitled **“OBJECT DETECTION WEB APPLICATION”** is based on our own work carried out during the course of our study under the supervision of **Anil Mandle**.

We assert that the statements made and the conclusions drawn are an outcome of the project work. We further declare that to the best of our knowledge and belief that the report does not contain any part of any work which has been submitted for the award of any other degree/diploma/certificate in this university or any other university.

(Signature of candidate)
Sourav Kumar Singh

301403319109

CERTIFICATE

This is to certify that the report of the project submitted is an outcome of the project work entitled “OBJECT DETECTION WEB APPLICATION” carried out by

Sourav Kumar Singh

Roll No.: 301403319109

Enrollment No. : BH5028

under my guidance and supervision for the award of Degree in Bachelor of Technology in **INFORMATION TECHNOLOGY** of Chhattisgarh Swami Vivekanand Technical University, Bhilai (C.G.), India.

To the best of my knowledge the report –

- i) Embodies the work of the candidate him/herself,
- ii) Has duly been completed,
- iii) Fulfills the requirement of the Ordinance relating to the BE degree of the University and
- iv) Is up to the desired standard for the purpose of which is submitted.

(Signature of the Guide)

Mr. Anil Mandle

Department of Information Technology

(Head of Department)

Dr. Dolly Shukla

Department of Information Technology

CERTIFICATE BY THE EXAMINERS

This is to certify that the project work entitled

“OBJECT DETECTION WEB APPLICATION”

Submitted by

Sourav Kumar Singh

Roll No.: 301403319109

Enrollment No. : BH5028

has been examined by the undersigned as a part of the examination for the award of Bachelor of Technology degree in Information Technology of Chhattisgarh Swami Vivekanand Technical University, Bhilai.

Internal Examiner
Date:

External Examiner
Date:

ACKNOWLEDGEMENT

I have great pleasure in the submission of this project report entitled OBJECT DETECTION WEB APPLICATION for Shri Shankaracharya Technical Campus in partial fulfillment of the degree of B.Tech. Information Technology. While Submitting this Project report, I take this opportunity to thank those directly or indirectly related to project work.

I would like to thank my guide Ms. Madhuri Gupta who has provided the opportunity and organizing project for me. Without his active co-operation and guidance, it would have become very difficult to complete task in time.

I would like to express sincere thanks and gratitude to P.B Deshmukh (Director), Dolly Shukla(HOD of Information Technology).

While Submission of the project, I also like to thank Mr. Anil Mandle (Project Coordinator), and the staff of Shri Shankaracharya Technical Campus for their continuous help and guidance throughout the course of the project.

Acknowledgement is due to our parents, family members, friends and all those persons who have helped us directly or indirectly in the successful completion of the project work.



Sourav Kumar Singh

TABLE OF CONTENTS

Chapter 1	Introduction	1-4
	1.1 Introduction	1-2
	1.2 Background Subtraction	3
	1.3 Template Matching	4
Chapter 2	Literature Survey	5-6
Chapter 3	Working	7-11
	3.1 Objectives	8
	3.2 Set up and Requirement	9
	3.3 TensorFlow Architecture	10-11
Chapter 4	Methodology	12-17
	4.1 Local Implementation	13
	4.2 Launch VM instance	14
	4.3 Install API Library	15
	4.4 Launch Web Application	16
	4.5 Test Web Application	17
Chapter 5	Result & Discussion	18-20
	5.1 Result	19
	5.2 Discussion	20
Chapter 6	Conclusion & future Enhancements	21-25
	6.1 Conclusion	22
	6.2 Future Enhancement	23-25

CHAPTER 1

INTRODUCTION

1. INTRODUCTION :

A few years ago, the creation of the software and hardware image processing systems was mainly limited to the development of the user interface, which most of the programmers of each firm were engaged in. The situation has been significantly changed with the advent of the Windows operating system when the majority of the developers switched to solving the problems of image processing itself. However, this has not yet led to the cardinal progress in solving typical tasks of recognizing faces, car numbers, road signs, analysing remote and medical images, etc. Each of these "eternal" problems is solved by trial and error by the efforts of numerous groups of the engineers and scientists. As modern technical solutions are turn out to be excessively expensive, the task of automating the creation of the software tools for solving intellectual problems is formulated and intensively solved abroad. In the field of image processing, the required tool kit should be supporting the analysis and recognition of images of previously unknown content and ensure the effective development of applications by ordinary programmers. Just as the Windows toolkit supports the creation of interfaces for solving various applied problems. Object recognition is to describe a collection of related computer vision tasks that involve activities like identifying objects in digital photographs. Image classification involves activities such as predicting the class of one object in an image.

Object localization is refers to identifying the location of one or more objects in an image and drawing an abounding box around their extent. Object detection does the work of combines these two tasks and localizes and classifies one or more objects in an image. When a user or practitioner refers to the term "object recognition", they often mean "object detection". It may be challenging for beginners to distinguish between different related computer vision tasks.

So, we can distinguish between these three computer vision tasks with this example: Image Classification: This is done by Predict the type or class of an object in an image. Input: An image which consists of a single object, such as a photograph.

Output: A class label (e.g. one or more integers that are mapped to class labels).

Object Localization: This is done through, Locate the presence of objects in an image and indicate their location with a bounding box.

Input: An image which 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).

Object Detection: This is done through, Locate the presence of objects with a bounding box and types or classes of the located objects in an image.

Input: An image which 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 extension 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, we can understand that object recognition refers to a suite of challenging computer vision tasks. For example, image classification is simply straight forward, 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 detect obstacles with little conscious thought. The availability of large sets of data, faster GPUs, and better algorithms ,we can now easily train computers to detect and classify multiple objects within an image with high accuracy. We need to understand terms such as object detection, object localization, loss function for object detection and localization, and finally explore an object detection algorithm known as “You only look once” (YOLO).

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 box around 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, is a family of techniques for 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 designed for speed and real-time use.

1.1 BACKGROUND SUBTRACTION

The background subtraction method by Horprasert et al (1999), was able to cope with local illumination changes, such as shadows and highlights, even global illumination changes. In this method, the background model was statistically modelled on each pixel. Computational colour models include the brightness distortion and the chromaticity distortion which was used to distinguish shading background from the ordinary background or moving foreground objects. The background and foreground subtraction method used the following approach. A pixel was modelled by a 4-tuple $[E_i, s_i, a_i, b_i]$, where E_i - a vector with expected colour value, s_i - a vector with the standard deviation of colour value, a_i - the variation of the brightness distortion and b_i was the variation of the chromaticity distortion of the i th pixel.

In the next step, the difference between the background image and the current image was evaluated. Each pixel was finally classified into four categories: original background, shaded background or shadow, highlighted background and moving foreground object. Liyuan Li et al (2003), contributed a method for detecting foreground objects in non-stationary complex environments containing moving background objects. A Bayes decision rule was used for classification of background and foreground changes based on inter-frame colour co-occurrence statistics. An approach to store and fast retrieve colour co-occurrence statistics was also established.

In this method, foreground objects were detected in two steps. First, both the foreground and the background changes are extracted using background subtraction and temporal differencing. The frequent background changes were then recognized using the Bayes decision rule based on the learned colour co-occurrence statistics. Both short-term and long-term strategies to learn the frequent background changes were used. An algorithm focused on obtaining the stationary foreground regions as said by Álvaro Bayona et al (2010), which was useful for applications like the detection of abandoned/stolen objects and parked vehicles.

This algorithm mainly used two steps. Firstly, a sub-sampling scheme based on background subtraction techniques was implemented to obtain stationary foreground regions. This detects foreground changes at different time instants in the same pixel locations. This was done by using a Gaussian distribution function. Secondly, some modifications were introduced on this base algorithm such as thresholding the previously computed subtraction. The main purpose of this algorithm was reducing the amount of stationary foreground detected.

1.2 TEMPLATE MATCHING

Template Matching is the technique of finding small parts of an image which match a template image. It slides the template from the top left to the bottom right of the image and compares for the best match with the template. The template dimension should be equal to the reference image or smaller than the reference image. It recognizes the segment with the highest correlation as the target. Given an image S and an image T , where the dimension of S was both larger than T , output whether S contains a subset image I where I and T are suitably similar in pattern and if such I exists, output the location of I in S as in Hager and Bellhumeur (1998). Schweitzer et al (2011), derived an algorithm which used both upper and lower bound to detect 'k' best matches. Euclidean distance and Walsh transform kernels are used to calculate match measure.

The positive things included the usage of priority queue improved quality of decision as to which bound-improved and when good matches exist inherent cost was dominant and it improved performance. But there were constraints like the absence of good matches that lead to queue cost and the arithmetic operation cost was higher.

The proposed methods don't use queue thereby avoiding the queue cost rather used template matching. Visual tracking methods can be roughly categorized in two ways namely, the feature-based and region-based method as proposed by Ken Ito and Shigeyuki Sakane (2001). The feature-based approach estimates the 3D pose of a target object to fit the image features the edges, given a 3D geometrical model of an object.

This method requires much computational cost. Region-based can be classified into two categories namely, parametric method and view-based method. The parametric method assumes a parametric model of the images in the target image and calculates optimal fitting of the model to pixel data in a region. The view-based method was used to find the best match of a region in a search area given the reference template.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY:

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, Hsiao-Ping Tsai 2011, Nicolas Papadakis and Aure lie Bugeau 2010) attempted for various approaches in object tracking. The nature of the techniques largely depends on the application domain. Some of the research works which made the evolution to proposed work in the field of object tracking are depicted as follows.

CHAPTER 3

WORKING

3.1 Objectives:

This is a basic lab designed to familiarize you with TensorFlow applications. When you are finished, you should be able to:

- Create a virtual machine (VM) using [Compute Engine](#).
- Install the [Object Detection API](#) library.
- Install and launch an object detection web application.
- Test the web application with uploaded images.


3.2 Setup and Requirements:

How to start your lab and sign in to the Google Cloud Console

1. Click the **Start Lab** button. If you need to pay for the lab, a pop-up opens for you to select your payment method. On the left is the **Lab Details** panel with the following:
 - The **Open Google Console** button
 - Time remaining
 - The temporary credentials that you must use for this lab
 - Other information, if needed, to step through this lab
2. Click **Open Google Console**. The lab spins up resources, and then opens another tab that shows the **Sign in** page.
3. Click through the subsequent pages:
 - Accept the terms and conditions.
 - Do not add recovery options or two-factor authentication (because this is a temporary account).
 - Do not sign up for free trials.

Activate Cloud Shell

Cloud Shell is a virtual machine that is loaded with development tools. It offers a persistent 5GB home directory and runs on the Google Cloud. Cloud Shell provides command-line access to your Google Cloud resources.

1. Click **Activate Cloud Shell**  at the top of the Google Cloud console.
2. Click **Continue**.

It takes a few moments to provision and connect to the environment. When you are connected, you are already authenticated, and the project is set to your **PROJECT_ID**. The output contains a line that declares the **PROJECT_ID** for this session `gcloud` is the command-line tool for Google Cloud. It comes pre-installed on Cloud Shell and supports tab-completion.

3. (Optional) You can list the active account name with this command:

```
gcloud auth list
```

3.3 TensorFlow architecture overview

The object detection application uses the following components:

- [TensorFlow 2.x](#): An open source machine learning library developed by researchers and engineers within Google's Machine Intelligence research organization. TensorFlow 2.0 runs on multiple computers to distribute the training workloads.
- [Object Detection API](#): An open source framework built on top of TensorFlow 2.x that makes it easy to construct, train, and deploy object detection models.
- [Pre-trained object detection models](#): The Object Detection API provides pre-trained object detection models for users running inference jobs. Users are not required to train models from scratch.

TensorFlow is at present the most popular software library. There are several real-world applications of deep learning that makes TensorFlow popular. Being an Open-Source library for deep learning and machine learning, TensorFlow finds a role to play in text-

based applications, image recognition, voice search, and many more. DeepFace, Facebook's image recognition system, uses TensorFlow for image recognition. It is used by Apple's Siri for voice recognition. Every Google app that you use has made good use of TensorFlow to make your experience better.

The TensorFlow Object Detection API is an open-source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models.

- There are already pre-trained models in their framework which are referred to as Model Zoo.
- It includes a collection of pre-trained models trained on various datasets such as the
 - COCO (Common Objects in Context) dataset,
 - the KITTI dataset,
 - and the Open Images Dataset.

As you may see below there are various models available so what is different in these models. These various models have different architecture and thus provide different accuracies but there is a trade-off between speed of execution and the accuracy in placing bounding boxes.

TensorFlow bundles together Machine Learning and Deep Learning models and algorithms. It uses Python as a convenient front-end and runs it efficiently in optimized C++.

TensorFlow allows developers to create a graph of computations to perform. Each node in the graph represents a mathematical operation and each connection represents data. Hence, instead of dealing with low-details like figuring out proper ways to hitch the output of one function to the input of another, the developer can focus on the overall logic of the application.

The deep learning artificial intelligence research team at Google, Google Brain, in the year 2015 developed TensorFlow for Google's internal use. This Open-Source Software library is used by the research team to perform several important tasks.

TensorFlow is at present the most popular software library. There are several real-world applications of deep learning that makes TensorFlow popular. Being an Open-Source library for deep learning and machine learning, TensorFlow finds a role to play in text-based applications, image recognition, voice search, and many more. Deep Face, Facebook's image recognition system, uses TensorFlow for image recognition. It is used by Apple's Siri for voice recognition. Every Google app that you use has made good use of TensorFlow to make your experience better.

Here mAP (mean average precision) is the product of precision and recall on detecting bounding boxes. It's a good combined measure for how sensitive the network is to objects of interest and how well it avoids false alarms. The higher the mAP score, the more accurate the network is but that comes at the cost of execution speed which we want to avoid here.

As my PC is a low-end machine with not much processing power, I am using the model `ssd_mobilenet_v1_coco` which is trained on COCO dataset. This model has decent mAP score and less execution time. Also, the COCO is a dataset of 300k images of 90 most commonly found objects so the model can recognise 90 objects.

CHAPTER 4
METHODOLOGY

4.1 Local implementation:

The following diagram shows how this Qwiklab is implemented. The web application is deployed to a VM instance running on Compute Engine.



Figure 4.1 : Implementation

When the client uploads an image to the application, the application runs the inference job locally. The pre-trained model returns the labels of detected objects, and the image coordinates of the corresponding objects. Using these values, the application generates new images populated with rectangles around the detected objects. Separate images are generated for each object category, allowing the client to discriminate between selected objects

You can use five pre-trained models with the Object Detection API. They are trained with [the COCO dataset](#) and are capable of detecting general objects in 80 categories.

The **COCO mAP** column shows the model's accuracy index. Higher numbers indicate better accuracy. As speed increases, accuracy decreases.

Model name	Speed	COCO mAP
ssd_mobilenet_v1_coco	fast	21
ssd_inception_v2_coco	fast	24

rfcn_resnet101_coco	medium	30
faster_rcnn_resnet101_coco	medium	32
faster_rcnn_inception_resnet_v2_atrous_coco	slow	37

.

4.2 Launch a VM instance:

Open the navigation menu and select **Compute Engine > VM Instances** to go to the VM Instances page.

Click the **Create Instance** button.

Create your VM with these settings:

- In the **Machine type** section, click **Custom** from the dropdown.
- **Cores** - replace **2** with **4**.
- **Memory** - replace **16** with **8**.
- In the **Firewall** section, select **Allow HTTP traffic**.

Click the **Networking, disks, security, management, sole-tenancy** dropdown, then click the **Networking** tab.

- Click on the **default** row in the **Network interfaces** section.
- In the **External IP** dropdown, select **Create IP address** from to assign a **static IP** address.
- Type staticip in the **Name** field, then click **Reserve**.

Click the **Create** button to create the instance and wait for the instance to be created.

SSH into the instance:

Next, click the **SSH** button for your instance to open an SSH terminal.

In the SSH shell, enter the following command to switch to the root user:

sudo -i

4.3 Install the Object Detection API library

Install the prerequisite packages:

```
apt-get update
```

```
apt-get install -y protobuf-compiler python3-pil python3-lxml python3-pip python3-dev git
```

```
pip3 install --upgrade pip
```

```
pip3 install Flask==2.1.2 WTFORMS==3.0.1 Flask_WTF==1.0.1 Werkzeug==2.0.3  
itsdangerous==2.1.2 jinja2==3.1.2 protobuf~=3.19.0
```

```
pip3 install tensorflow==2.9.0
```

```
cd /opt
```

```
git clone https://github.com/tensorflow/models
```

```
cd models/research
```

```
protoc object_detection/protos/*.proto --python_out=.
```

```
mkdir -p /opt/graph_def
```

```
cd /tmp
```

```
for model in \
```

```
  ssd_mobilenet_v1_coco_11_06_2017 \
```

```
  ssd_inception_v2_coco_11_06_2017 \
```

```
  rfcn_resnet101_coco_11_06_2017 \
```

```
  faster_rcnn_resnet101_coco_11_06_2017 \
```

```
  faster_rcnn_inception_resnet_v2_atrous_coco_11_06_2017
```

```
do \
```

```
  curl -OL http://download.tensorflow.org/models/object\_detection/\$model.tar.gz
```

```
  tar -xzf $model.tar.gz $model/frozen_inference_graph.pb
```

```
cp -a $model /opt/graph_def/
```

done

```
ln -sf /opt/graph_def/faster_rcnn_resnet101_coco_11_06_2017/frozen_inference_graph.pb  
/opt/graph_def/frozen_inference_graph.pb
```

4.4 Install and launch the web application

```
cd $HOME
```

```
git clone https://github.com/GoogleCloudPlatform/tensorflow-object-detection-example
```

```
cp -a tensorflow-object-detection-example/object_detection_app_p3 /opt/
```

```
chmod u+x /opt/object_detection_app_p3/app.py
```

Create the object detection Service

```
cp /opt/object_detection_app_p3/object-detection.service /etc/systemd/system/
```

```
systemctl daemon-reload
```

```
systemctl enable object-detection
```

```
systemctl start object-detection
```

```
systemctl status object-detection
```

object-detection.service - Object Detection API Demo

Loaded: loaded (/opt/object_detection_app/object-detection.service; linked)

Active: active (running) since Wed 2017-06-21 05:34:10 UTC; 22s ago

Process: 7122 ExecStop=/bin/kill -TERM \$MAINPID (code=exited, status=0/SUCCESS)

Main PID: 7125 (app.py)

CGroup: /system.slice/object-detection.service

└─7125 /usr/bin/python /opt/object_detection_app/app.py

Jun 21 05:34:10 object-detection systemd[1]: Started Object Detection API Demo.

Jun 21 05:34:26 object-detection app.py[7125]: 2017-06-2105:34:26.518736: W tensorflow/core/platform/cpu_fe...ons.

Jun 21 05:34:26 object-detection app.py[7125]: 2017-06-2105:34:26.518790: W tensorflow/core/platform/cpu_fe...ons.

Jun 21 05:34:26 object-detection app.py[7125]: 2017-06-2105:34:26.518795: W tensorflow/core/platform/cpu_fe...ons.

*Jun 21 05:34:26 object-detection app.py[7125]: * Running on http://0.0.0.0:80/ (Press CTRL+C to quit)*

Hint: Some lines were ellipsized, use -l to show in full

4.5 Test the Web Application

Return to the Google Cloud console tab.

Click the external ip-address link next to your VM instance to open a browser window to the example application.

Log in with the following credentials:

- Username - username
- Password - passw0rd

Click **Choose File** and select an image file from your computer to analyse.

Click **Upload** to test.

When you upload an image file with a **JPEG**, **JPG**, or **PNG** extension, the application shows the result of the object detection inference, as shown in the following image. The inference might take up to 30 seconds, depending on the image.

CHAPTER 5
RESULT
AND
DISCUSSIONS

Object Detection API

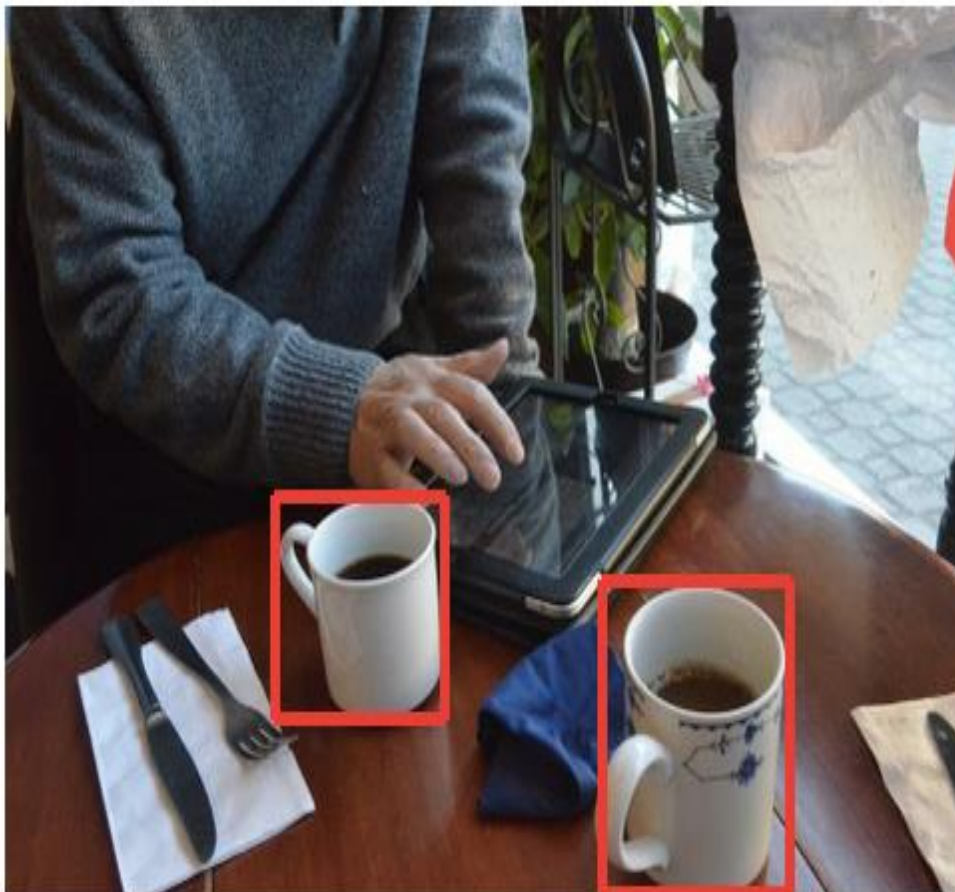
Upload a color photo file.

File extension should be: jpeg, jpg, png

Choose File No file chosen

Upload

cup



[original](#)
[fork](#)
[cup](#)
[dining table](#)
[person](#)
[knife](#)

Figure 5.1 : Web Application

The object names detected by the model are shown to the right of the image, in the application window. Click an object name to display rectangles surrounding the corresponding objects in the image. The rectangle thickness increases with object identification confidence.

In the above image, "fork", "cup", "dining table", "person", and "knife", are detected. After clicking "**cup**", rectangles display around all detected cups in the image. Click "**original**" to see the original image.

Image AI provides many more features useful for customization and production capable deployments for object detection tasks. Some of the features supported are:

- Adjusting Minimum Probability: By default, objects detected with a probability percentage of less than 50 will not be shown or reported. You can increase this value for high certainty cases or reduce the value for cases where all possible objects are needed to be detected.
- Custom Objects Detection: Using a provided Custom Object class, you can tell the detection class to report detections on one or a few number of unique objects.
- Detection Speeds: You can reduce the time it takes to detect an image by setting the speed of detection speed to "fast", "faster" and "fastest".
- Input Types: You can specify and parse in file path to an image, Numpy array or file stream of an image as the input image.

CHAPTER 6
CONCLUSION AND
FUTURE
ENHANCEMENTS

6.1 CONCLUSION:

By using this thesis and based on experimental results we are able to detect object more precisely and identify the objects individually with exact location of an object in the picture in x,y axis. This paper also provides experimental results on different methods for object detection and identification and compares each method for their efficiencies.

This project starts on generic object detection pipelines which provide base architectures for other related tasks. With the help of this the three other common tasks, namely object detection, face detection and pedestrian detection, can be accomplished. Authors accomplished this by combining two things: Object detection with deep learning and OpenCV and Efficient, threaded video streams with OpenCV. The camera sensor noise and lightening condition can change the result as it can create problem in recognizing the object.

6.2 FUTURE ENHANCEMENTS

The object recognition system can be applied in the area of surveillance system, face recognition, fault detection, character recognition etc. The objective of this thesis is to develop an object recognition system to recognize the 2D and 3D objects in the image. The performance of the object recognition system depends on the features used and the classifier employed for recognition. This research work attempts to propose a novel feature extraction method for extracting global features and obtaining local features from the region of interest. Also the research work attempts to hybrid the traditional classifiers to recognize the object. The object recognition system developed in this research was tested with the benchmark datasets like COIL100, Caltech 101, ETH80 and MNIST. The object recognition system is implemented in MATLAB 7.5

It is important to mention the difficulties observed during the experimentation of the object recognition system due to several features present in the image. The research work suggests that the image is to be preprocessed and reduced to a size of 128 x 128. The proposed feature extraction method helps to select the important feature. To improve the efficiency of the classifier, the number of features should be less in number. Specifically, the contributions towards this research work are as follows,

- An object recognition system is developed, that recognizes the two-dimensional and three-dimensional objects.
- The feature extracted is sufficient for recognizing the object and marking the location of the object. x The proposed classifier is able to recognize the object in less computational cost.
- The proposed global feature extraction requires less time, compared to the traditional feature extraction method.
- The performance of the SVM-kNN is greater and promising when compared with the BPN and SVM.

The performance of the One-against-One classifier is efficient.

The methods presented for feature extraction and recognition are common and can be applied to any application that is relevant to object recognition.

The proposed object recognition method combines the state-of-art classifier SVM and k-

NN to recognize the objects in the image. The multiclass SVM is used to hybridize with the k-NN for the recognition. The feature extraction method proposed in this research work is efficient and provides unique information for the classifier.

The image is segmented into 16 parts, from each part the Hu's Moment invariant is computed and it is converted into Eigen component. The local feature of the image is obtained by using the Hessian-Laplace detector. This helps to obtain the objects feature easily and mark the object location without much difficulty.

As a scope for future enhancement.

- Features either the local or global used for recognition can be increased, to increase the efficiency of the object recognition system.
- Geometric properties of the image can be included in the feature vector for recognition. 150
- Using unsupervised classifier instead of a supervised classifier for recognition of the object.
- The proposed object recognition system uses grey-scale image and discards the color information. The colour information in the image can be used for recognition of the object. Colour based object recognition plays vital role in Robotics

Although the visual tracking algorithm proposed here is robust in many of the conditions, it can be made more robust by eliminating some of the limitations as listed below:

- In the Single Visual tracking, the size of the template remains fixed for tracking. If the size of the object reduces with the time, the background becomes more dominant than the object being tracked. In this case the object may not be tracked.
- Fully occluded object cannot be tracked and considered as a new object in the next frame.
- Foreground object extraction depends on the binary segmentation which is carried out by applying threshold techniques. So blob extraction and tracking depends on the threshold value.
- Splitting and merging cannot be handled very well in all conditions using the single camera due to the loss of information of a 3D object projection in 2D images.

To make the system fully automatic and also to overcome the above limitations, in future, multi-view tracking can be implemented using multiple cameras. Multi view tracking has the obvious advantage over single view tracking because of wide

coverage range with different viewing angles for the objects to be tracked.

In this thesis, an effort has been made to develop an algorithm to provide the base for future applications such as listed below.

- In this research work, the object Identification and Visual Tracking has been done through the use of ordinary camera. The concept is well extendable in applications like Intelligent Robots, Automatic Guided Vehicles, Enhancement of Security Systems to detect the suspicious behaviour along with detection of weapons, identify the suspicious movements of enemies on borders with the help of night vision cameras and many such applications.
- In the proposed method, background subtraction technique has been used that is simple and fast. This technique is applicable where there is no movement of camera. For robotic application or automated vehicle assistance system, due to the movement of camera, backgrounds are continuously changing leading to implementation of some different segmentation techniques like single Gaussian mixture or multiple Gaussian mixture models.
- Object identification task with motion estimation needs to be fast enough to be implemented for the real time system. Still there is a scope for developing faster algorithms for object identification. Such algorithms can be implemented using FPGA or CPLD for fast execution.

REFERENCES

1. Agarwal, S., Awan, A., and Roth, D. (2004). Learning to detect objects in images via a sparse, part-based representation. *IEEE Trans. Pattern Anal. Mach. Intell.* 26,1475–1490.doi:10.1109/TPAMI.2004.108
2. Alexe, B., Deselaers, T., and Ferrari, V. (2010). “What is an object?,” in *ComputerVision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (San Francisco,CA: IEEE), 73–80. doi:10.1109/CVPR.2010.5540226
3. Aloimonos, J., Weiss, I., and Bandyopadhyay, A. (1988). Active vision. *Int. J.Comput. Vis.* 1, 333–356. doi:10.1007/BF00133571
4. Andreopoulos, A., and Tsotsos, J. K. (2013). 50 years of object recognition: directions forward. *Comput. Vis. Image Underst.* 117, 827–891. doi:10.1016/j.cviu.2013.04.005
5. Azizpour, H., and Laptev, I. (2012). “Object detection using strongly-superviseddeformable part models,” in *Computer Vision-ECCV 2012* (Florence: Springer),836–849.
6. Azzopardi, G., and Petkov, N. (2013). Trainable cosfire filters for keypoint detectionand pattern recognition. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 490–503.doi:10.1109/TPAMI.2012.106
7. Azzopardi, G., and Petkov, N. (2014). Ventral-stream-like shape representation:from pixel intensity values to trainable object-selective cosfire models. *Front.Comput. Neurosci.* 8:80.doi:10.3389/fncom.2014.00080
8. Benbouzid, D., Busa-Fekete, R., and Kegl, B. (2012). “Fast classification using sparsedecision dags,” in *Proceedings of the 29th International Conference on MachineLearning (ICML-12), ICML ‘12*, eds J. Langford and J. Pineau (New York, NY:Omnipress), 951–958.
9. Bengio, Y. (2012). “Deep learning of representations for unsupervised and transferlearning,” in *ICML Unsupervised and Transfer Learning, Volume 27 of JMLRProceedings*, eds I. Guyon, G. Dror, V. Lemaire, G. W. Taylor, and D. L. Silver(Bellevue: JMLR.Org), 17–36.
10. Bourdev, L. D., Maji, S., Brox, T., and Malik, J. (2010). “Detecting peopleusing mutually consistent poselet activations,” in *Computer Vision – ECCV2010 – 11th European Conference on Computer Vision, Heraklion, Crete, Greece,September 5-11, 2010, Proceedings, Part VI, Volume 6316 of Lecture Notes inComputer*

ScienceK. Daniilidis, P. Maragos, and N. Paragios (Heraklion:Springer), 168–181.

Bourdev, L. D., and Malik, J. (2009). “Poselets: body part detectors trained using 3d human pose annotations,” in IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 – October 4, 2009 (Kyoto: IEEE), 1365–1372.

- i. Correa, M., Hermosilla, G., Verschae, R., and Ruiz-del-Solar, J. (2012). Human detection and identification by robots using thermal and visual information in domestic environments. *J. Intell. Robot Syst.* 66, 223–243. doi:10.1007/s10846-011-9612-2
- ii. Dalal, N., and Triggs, B. (2005). “Histograms of oriented gradients for human detection,” in Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, Vol. 1 (San Diego, CA: IEEE), 886–893. doi:10.1109/CVPR.2005.177
- iii. Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). “Scalable object detection using deep neural networks,” in Computer Vision and Pattern Recognition Frontiers in Robotics and AI www.frontiersin.org November 2015.