Object Detection Streaming and Data Management on Web Browsers

Le-Anh Tran
leanhtran@mju.ac.kr
Myongji University, South Korea

***Abstract* – This project presents an object-detection-based surveillance system applying YOLOv3 engine and streaming the detection result on a web browser with 2 cameras (2 channels), simultaneously the detection information is stored in a database. The application consists of three main components: (1) YOLOv3 object detection module, (2) a database for storing detection results including images and information, and (3) a web template for observation and querying data from the database. This application applied Django, an open-source and flexible web framework written in Python, for communicating between the object detection module, the database, and the web browser. The experimental results proved that the application can work stably and can be applied to surveillance system at companies, factories, or public locations such as bus station or airport.**

Keywords – object detection, streaming, django, database management, web framework.

I. INTRODUCTION

Surveillance systems are systems used for the purpose of observing an area. They usually consist of cameras connected to a recording device or local server and may be watched by a security officer. In the last decades, cameras and recording equipment used to be relatively expensive and required human personnel to monitor camera footage, but nowadays analysis of footage has been made easier by automated software. The captured frames can be stored and organized in a searchable database. Besides, along with the development of artificial intelligence (AI) techniques, most useful and basic information in the footage is also stored in the database with the respective frame. Due to the affordability, surveillance cameras currently are inexpensive and applicable to security systems in houses, companies, and factories [1] or autonomous driver-assistance systems [5].

Our system is an object-detection-based surveillance web application (applied YOLOv3 engine [2]) that streams the detection result on a web browser with 2 cameras (2 channels), simultaneously the detection information is stored in a database. The application was built using Django framework [3] and allows the user to be able to observe the detection outcome, searching functionality help the user filter and query data in the database based on the fields of date time, camera id, and object classes.

This report is organized as follows. Section 2 firstly describes the workflow of the system, then dives deeply into introducing each module of the system including object detection engine, Django framework, and database management. Section 3 discusses the result of the project. Finally, the conclusions of the project are presented in section 4.

II. THE SYSTEM

A. System Overview

Image frame is captured by cameras then is passed through an object detection module to yield the detection. On the one hand, the object detection result is stored in a database including the frame and its respective detected object information (date time, camera id, object class, bounding box coordinate), on the other hand, the image with bounding boxes are drawn on is displayed on the web template. The web application has also a searching functionality for filtering the data by date time, camera id, and object classes. Figure 1 depicts the diagram of the application.

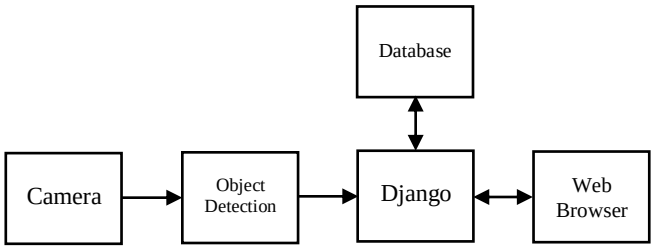


Figure 1. Application Diagram.

B. Object Detection Engine

In recent years, the rapid development of deep learning has drawn more attention to object detection issue. Nowadays, many object detection algorithms have been deployed in a variety of forms and have been applied in various applications. In the project, the use of object detection for surveillance application is discussed.

Object detection in images means not only identify what kind of object is included but also localize it inside the image (obtain the coordinates of the "bounding box" containing the object). The most state-of-the-art object detection algorithms are SSD, R-FCN, RetinaNet, and YOLO. As figure 2, showing YOLOv3 performance

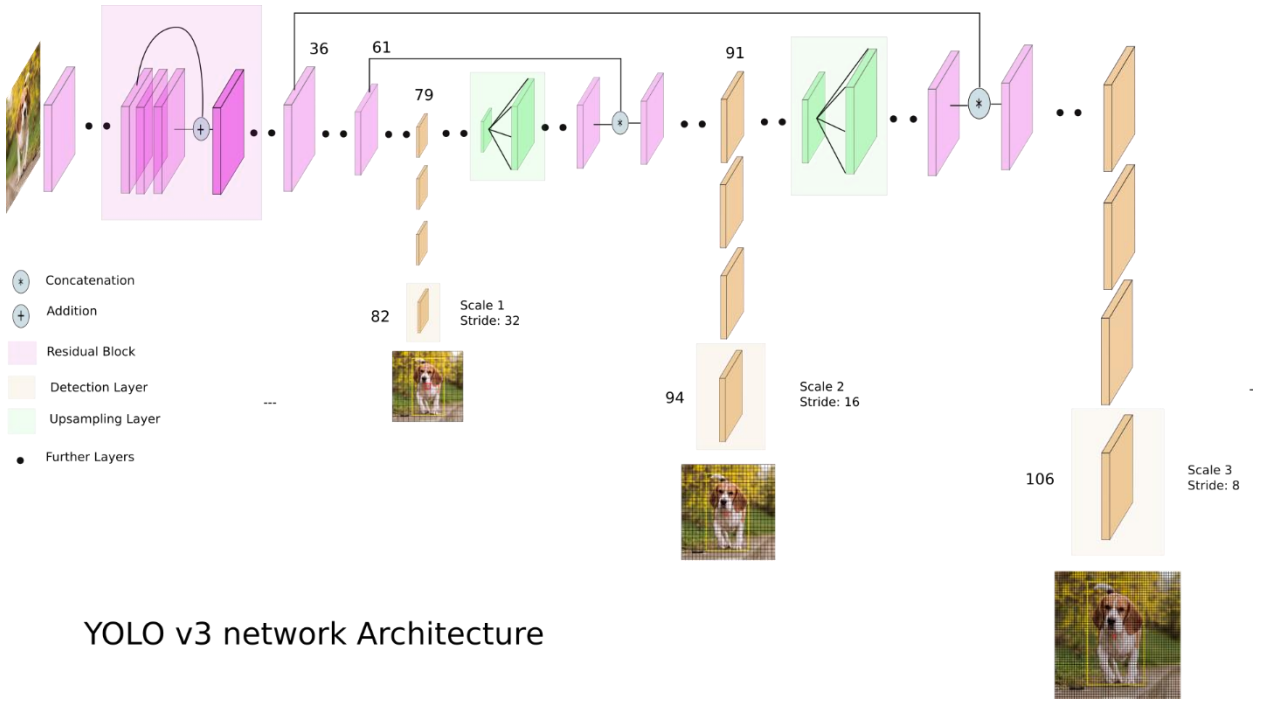


Figure 3. YOLO v3 network architecture (Image source: Ayoosh Kathuria - https://towardsdatascience.com/@ayoosh).

compared to other object detection algorithms, YOLOv3 has the fastest processing time while its Mean Average Precision (mAP) is also favorable, so it was carried out to be deployed in the application. Figure 3 illustrates YOLOv3 network architecture [2].

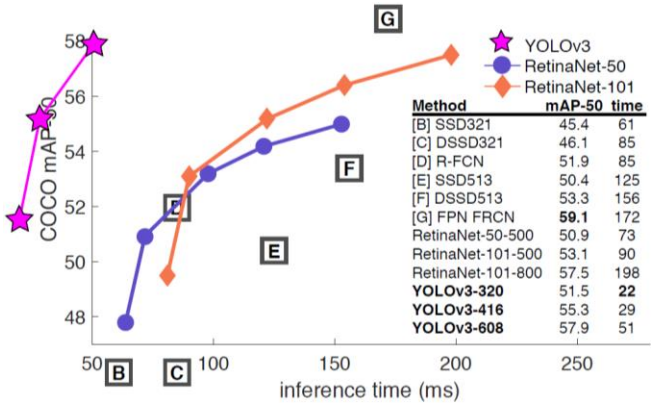


Figure 2. Object Detection Algorithms Performance [2].

For feature extraction, YOLOv3 uses a new network, Darknet-53, which is much more powerful than Darknet-19 but still more efficient than ResNet-101 or ResNet-152 and with 80 class predictions [2].

C. Django Framework

Django is a free and open-source web framework, written in Python, which follows the model–view–controller (MVC) architectural pattern [3].

• URL: A URL mapper is used to redirect HTTP requests to the appropriate view based on the request URL. The URL mapper can also match particular patterns of strings or digits that appear in an URL, and pass these to a view function as data.

• Model: A model is the single, definitive source of information about your data. It contains the essential fields and behaviors of the data you stored. Generally, each model maps to a single database table.

a) Each model is a Python class that subclasses django.db.models.Model.

b) Each attribute of the model represents a database field.

c) With all of this, Django gives an automatically generated database-access API.

• View: A view function, or view for short, is simply a Python function that takes a Web request and returns a Web response. This response can be the HTML contents of a Web page, or a redirect, or a 404 error, or an XML document, or an image, etc.

• Template: Being a web framework, Django needs a convenient way to generate HTML dynamically. The most common approach relies on templates. A template contains the static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

Simple Django's workflow description: Request from the web page is detected by paths in urls.py, urls.py tries to match the paths with appropriate functions in views.py,

views.py processes the request and reads/writes data in the database if needed via models.py, then calls a template and responses to the web page. The workflow is summarized in figure 4.

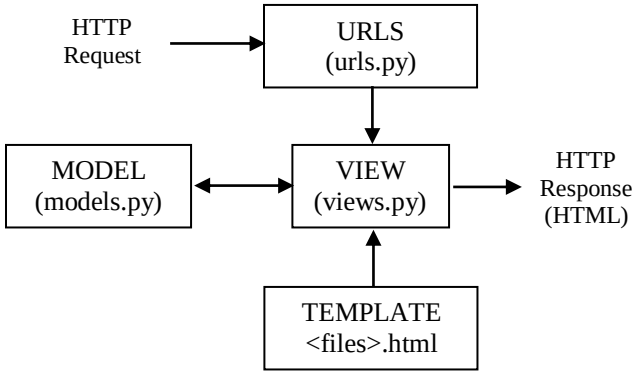


Figure 4. Django's Workflow.

D. Database Management

MySql is an open-source database management system that operates on a client-server model and is used to create and manage databases in the form of managing the relationship between them [4]. Even if it was initially limited, it is now compatible with many operating systems such as Linux, macOS, Microsoft Windows, and Ubuntu.

Description of the tables in the MySql database: In the project, we created 2 tables: *webcam_infor_storage* and *webcam_image_storage* for storing the object detection information and image frame data, respectively.

a) Table 1: *webcam_infor_storage*

This table contains a list of the object detection information including date time, camera id, detected object class, and bounding box coordinate. This table has 8 fields:

id: frame id (INT)

date_time: the date time of the captured frame (DATETIME)

cam_id: each camera has a particular id (INT)

object_class: name of the detected object (VARCHAR)

x, y, w, h: the left-top corner coordinate of the bounding box (x, y) and its width, height (w, h) (INT)

b) Table 2: *webcam_image_storage*

This table stores image data as blob data and basic information of the frame including date time, and camera id. This table has 4 fields:

id: frame id (INT)

date_time: the date time of the captured frame (DATETIME)

cam_id: each camera has a particular id (INT)

image: stored image as blob data (BLOB)

Table 1. The table *webcam_infor_storage* in mysql database.

id	date_time	cam_id	object_class	x	y	w	h
1	2019-07-01 12:00:00.123456	0	person	100	100	150	250
2	2019-07-01 12:00:01.456789	1	cell phone	200	200	50	100
…	…	…	…	…	…	…	…

Table 2. The table *webcam_image_storage* in mysql database.

id	date_time	cam_id	image
1	2019-07-01 12:00:00.123456	0	<blob data>
2	2019-07-01 12:00:01.456789	1	<blob data>
…	…	…	…

III. EXPERIMENTAL RESULTS

The system was built and run in Ubuntu 18.04, including 2 cameras for 2 channels, one is the laptop built-in camera and the other one is Logitech USB camera, captured image is resized to the resolution of 1024x768 pixels. The web template has 4 main tabs, tab Home is the home page of the application in which the object detection outcome of camera 1 is set to default to display, thus tab Camera 1 and tab Home have almost the same contents, tab Camera 2 is for displaying the object detection outcome from camera 2, tab Database is the most important part, which displays data and helps user query data from the database, it has 2 main buttons, *Display 50 latest data* refreshes and displays the 50 newest data from the database, *Filter* returns a dashboard for filtering and querying data from the database by date time, camera id, and object classes.

The following pictures illustrate those main parts of the application: figure 5 shows the web template for displaying object detection outcome of camera 1, figure 6 depicts the table *webcam_infor_storage* in the database, figure 7 indicates that the data is queried from the database then is displayed on a web browser, and figure 8 illustrates the searching functionality for filtering data by date time, camera id, and object classes.

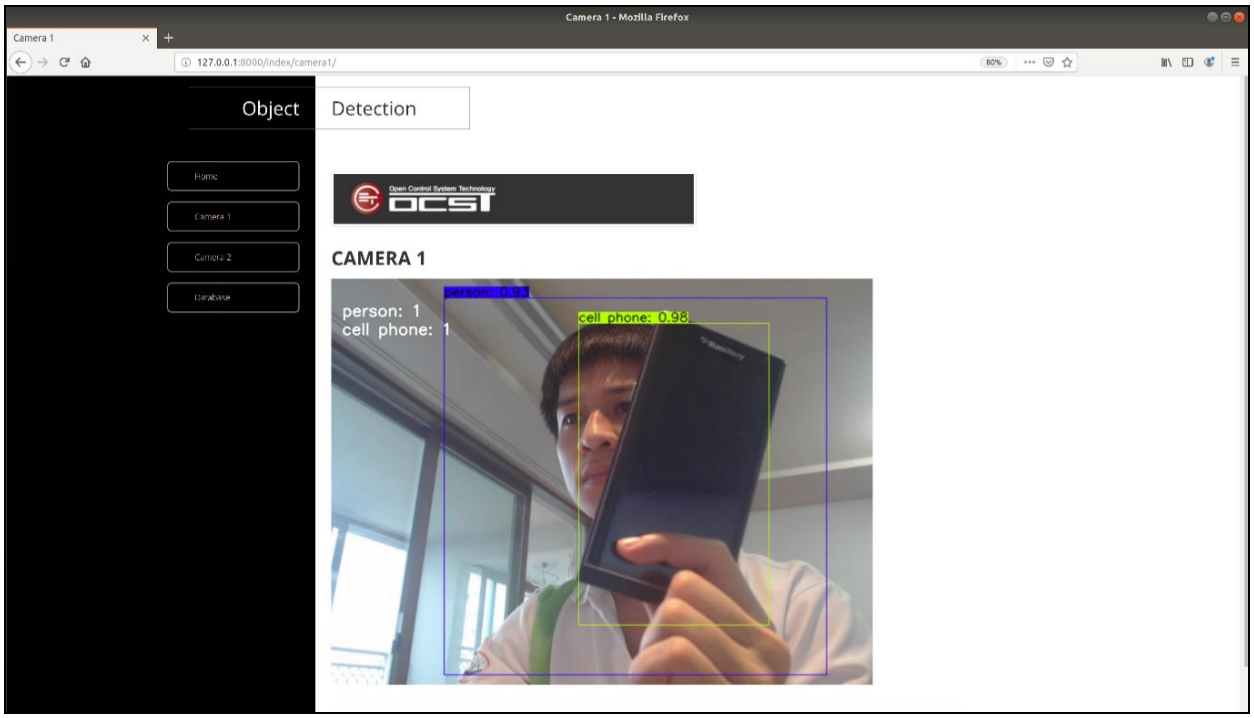
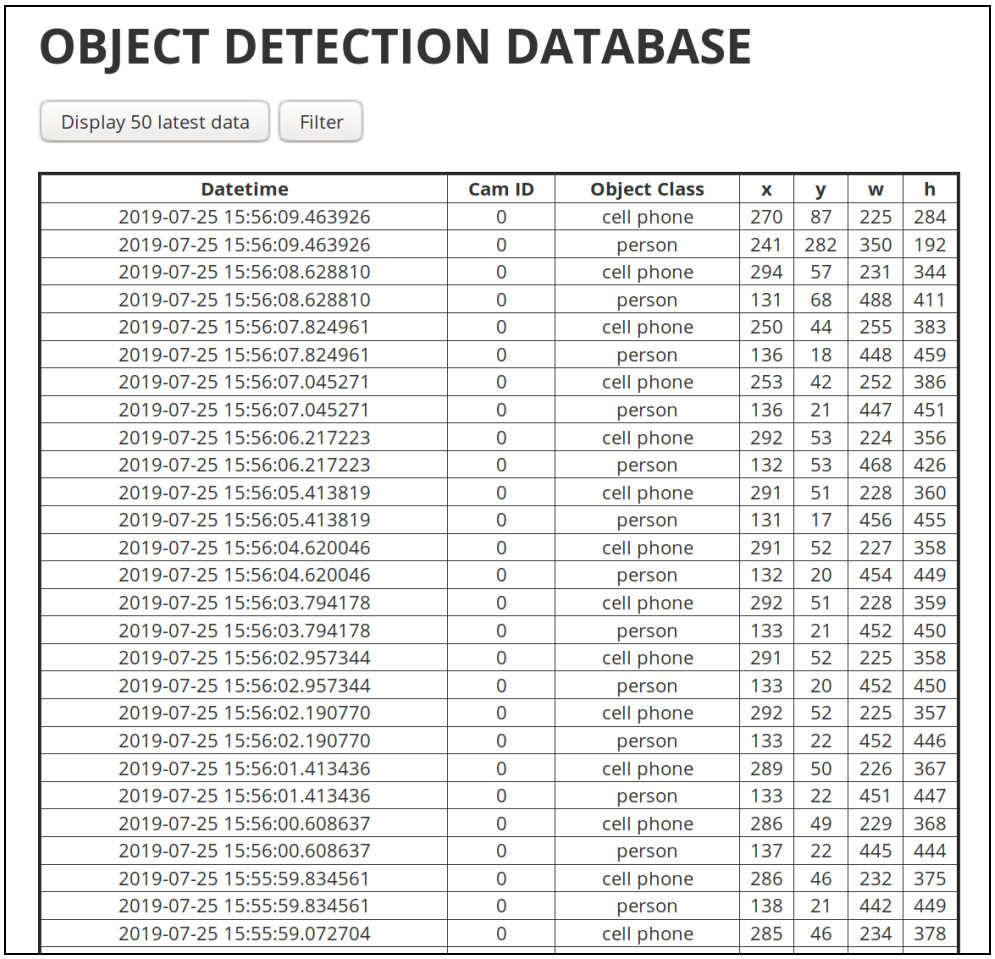


Figure 5. Observation on web template (page of camera 1).

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

#	id	date_time	cam_id	object_class	x	y	w	h
1	1691	2019-07-25 15:56:09.463926	0	person	241	282	350	192
2	1692	2019-07-25 15:56:09.463926	0	cell phone	270	87	225	284
3	1689	2019-07-25 15:56:08.628810	0	person	131	68	488	411
4	1690	2019-07-25 15:56:08.628810	0	cell phone	294	57	231	344
5	1687	2019-07-25 15:56:07.824961	0	person	136	18	448	459
6	1688	2019-07-25 15:56:07.824961	0	cell phone	250	44	255	383
7	1685	2019-07-25 15:56:07.045271	0	person	136	21	447	451
8	1686	2019-07-25 15:56:07.045271	0	cell phone	253	42	252	386
9	1683	2019-07-25 15:56:06.217223	0	person	132	53	468	426
10	1684	2019-07-25 15:56:06.217223	0	cell phone	292	53	224	356
11	1681	2019-07-25 15:56:05.413819	0	person	131	17	456	455
12	1682	2019-07-25 15:56:05.413819	0	cell phone	291	51	228	360
13	1679	2019-07-25 15:56:04.620046	0	person	132	20	454	449
14	1680	2019-07-25 15:56:04.620046	0	cell phone	291	52	227	358
15	1677	2019-07-25 15:56:03.794178	0	person	133	21	452	450
16	1678	2019-07-25 15:56:03.794178	0	cell phone	292	51	228	359
17	1675	2019-07-25 15:56:02.957344	0	person	133	20	452	450
18	1676	2019-07-25 15:56:02.957344	0	cell phone	291	52	225	358
19	1673	2019-07-25 15:56:02.190770	0	person	133	22	452	446
20	1674	2019-07-25 15:56:02.190770	0	cell phone	292	52	225	357

Figure 6. Table *webcam_infor_storage* in the database.



Datetime	Cam ID	Object Class	x	y	w	h
2019-07-25 15:56:09.463926	0	cell phone	270	87	225	284
2019-07-25 15:56:09.463926	0	person	241	282	350	192
2019-07-25 15:56:08.628810	0	cell phone	294	57	231	344
2019-07-25 15:56:08.628810	0	person	131	68	488	411
2019-07-25 15:56:07.824961	0	cell phone	250	44	255	383
2019-07-25 15:56:07.824961	0	person	136	18	448	459
2019-07-25 15:56:07.045271	0	cell phone	253	42	252	386
2019-07-25 15:56:07.045271	0	person	136	21	447	451
2019-07-25 15:56:06.217223	0	cell phone	292	53	224	356
2019-07-25 15:56:06.217223	0	person	132	53	468	426
2019-07-25 15:56:05.413819	0	cell phone	291	51	228	360
2019-07-25 15:56:05.413819	0	person	131	17	456	455
2019-07-25 15:56:04.620046	0	cell phone	291	52	227	358
2019-07-25 15:56:04.620046	0	person	132	20	454	449
2019-07-25 15:56:03.794178	0	cell phone	292	51	228	359
2019-07-25 15:56:03.794178	0	person	133	21	452	450
2019-07-25 15:56:02.957344	0	cell phone	291	52	225	358
2019-07-25 15:56:02.957344	0	person	133	20	452	450
2019-07-25 15:56:02.190770	0	cell phone	292	52	225	357
2019-07-25 15:56:02.190770	0	person	133	22	452	446
2019-07-25 15:56:01.413436	0	cell phone	289	50	226	367
2019-07-25 15:56:01.413436	0	person	133	22	451	447
2019-07-25 15:56:00.608637	0	cell phone	286	49	229	368
2019-07-25 15:56:00.608637	0	person	137	22	445	444
2019-07-25 15:55:59.834561	0	cell phone	286	46	232	375
2019-07-25 15:55:59.834561	0	person	138	21	442	449
2019-07-25 15:55:59.072704	0	cell phone	285	46	234	378

Figure 7. Querying data from database and display it on web template.

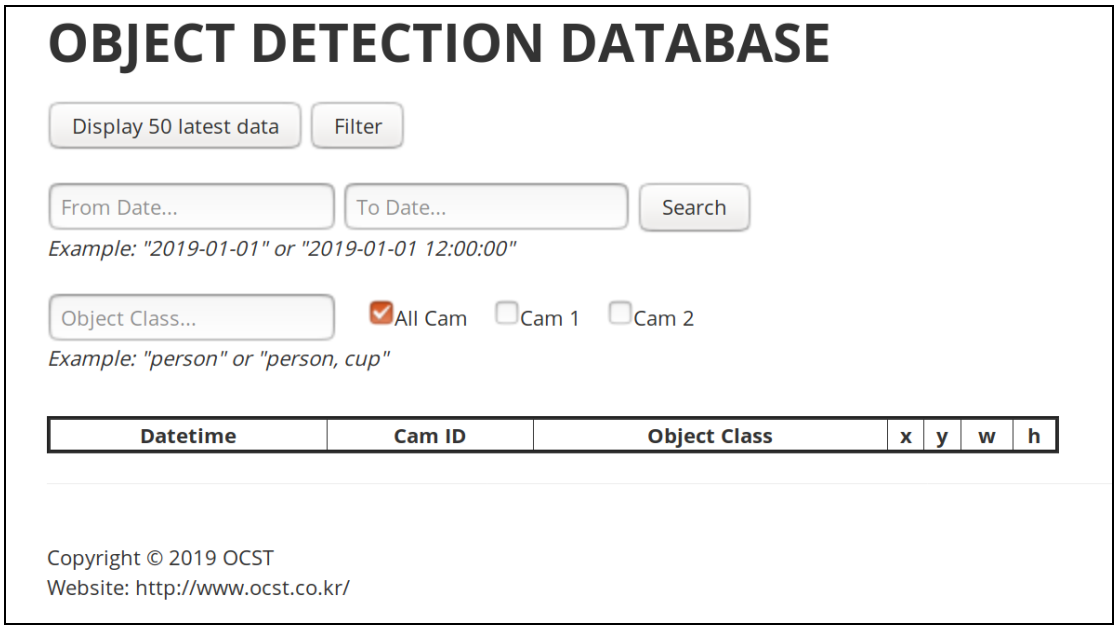


Figure 8. Searching functionality.

IV. USAGE

A. Prerequisites

The system has been run locally. Besides, some packages and installations are required, below are some essential requirements:

- Ubuntu 18.04
- Virtual environment
- Python 3.5
- Django 2.2
- MySQL 5.6

B. Important Folders and Files

The project was built and run in virtual environment with necessary packages, in which, some important folders and files are noted as below:

- manage.py: main of the project
- statics/css: web template written in css
- statics/images: images for web template
- stream/settings.py: settings for database connection, linking to static files, etc.
- stream/urls.py: forward request to functions in views.py
- webcam/views.py: main controller/processor
- webcam/models.py: configure connection to database
- webcam/templates: html files for web display

C. Usage

Create a virtual environment (named *my_env*), the main purpose of virtual environments is to create an isolated environment for Python projects:

```
python3 -m venv my_env
```

Activate virtual environment by running the command:

```
source my_env/bin/activate
```

Navigate terminal to the project, run the command:

```
python3 mamage.py runserver
```

Open any web browser and navigate to the URL:

http://127.0.0.1:8000/index/

Note: Running the project in the first time on a new computer may yield error caused by missing packages or incorrect version of packages, if so, install or update those packages then run the project again.

V. CONCLUSIONS

This report mainly introduces the process and structure of an object-detection-based surveillance web application (applied YOLOv3 engine) that streams the detection result on a web browser with 2 cameras (2 channels), simultaneously the detection information is stored in a database. In terms of web design, this application applied the Django, which is an open-source, flexible web framework, written in Python. In terms of database management, the data is stored and managed in MySql database, an open-source relational database management system. We establish communication between the object detection module and the database, on the web browser, users can query and filter data from the database by using a searching functionality, which can be used to filter data by date time, camera id, and object classes.

The application now can operate stably and is being improved to be more flexible and optimized. Hopefully, the application can be applied to surveillance system at companies, factories, or public locations such as bus station or airport.

REFERENCES

[1] Kinjal A Joshi, and Darshak G. Thakore, "A Survey on Moving Object Detection and Tracking in Video Surveillance System," *International Journal of Soft Computing and Engineering (IJSCE)*, Volume-2, Issue-3, July 2012, pp. 44-48.

[2] Joseph Redmon, and Ali Farhadi, "Yolov3: An incremental improvement," CoRR, vol. abs/1804.02767, 2018.

[3] Chuanhong Zho, and Qi Fei, "Warehouse Management System Development Base on Open-source Web Framework," in *2016 International Conference on Industrial Informatics - Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII 2016)*, Wuhan, China, Dec. 3-4, 2016, pp. 65-68.

[4] Jitesh Shetty, and Jafar Adibi, "The Enron Email Dataset Database Schema and Brief Statistical Report", Los Angeles, CA, 2004.

[5] Le-Anh Tran, and My-Ha Le, "Robust U-Net-based Road Lane Markings Detection for Autonomous Driving," *International Conference on System Science and Engineering (ICSSE 2019)*, 62-66. doi:10.1109/ICSSE.2019.8823532.