

Traffic Analysis System

28/05/2021

Software Requirements and Installation

1. OS: Ubuntu 20.04 LTS

2. Python version 3.8

```
apt install python3.8  
apt install python3-pip
```

3. FFmpeg version 4 or newer

```
apt install ffmpeg
```

4. PostgreSQL version 12 or newer

```
apt install postgresql  
create custom user, password, database
```

5. Web server (Apache or Nginx)

6. Python libs from standard repo (pip install -r requirements.txt)

```
ffmpeg-python  
psutil  
psycopg2-binary  
pandas  
paho-mqtt  
norfair  
pandarallel
```

7. Special Python libs

- PyTorch for system CUDA version (<https://pytorch.org/>)
- OpenCV 4
 - apt update
 - apt install libopencv-dev python3-opencv

8. Traffic Analysis System

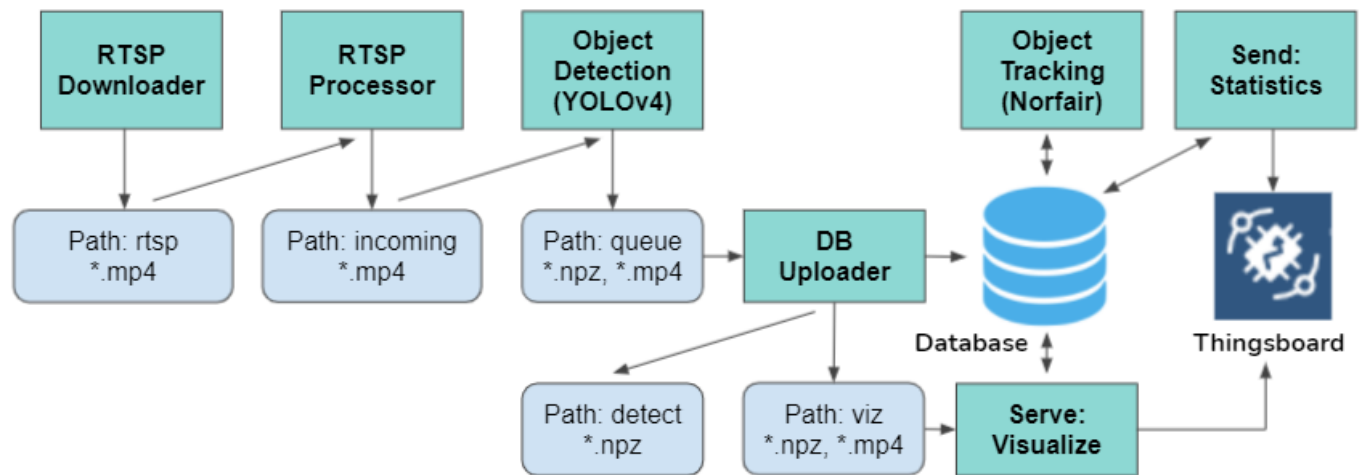
- 8.1. Install software (<https://github.com/DRR-IGI/workflow.git>)
- 8.2. Setup local configuration file (bin/config.ini)
- 8.3. Make data dirs

```
python makedirs.py
```
- 8.3. Setup database

```
psql -d (db_name) < ../sql/schema.sql
```
- 8.4. Start/ Stop the system

```
bin/start.sh  
bin/stop.sh
```

Main Processing Workflow



1. RTSP Downloader

Download segment of RTSP video into path: rtsp

2. RTSP Processor

Check and move complete video segments into path: incoming

3. Object Detection (YOLOv4)

Detect objects in video, write detection results to a npz file. Move complete data into path: queue

4. DB Uploader

- Upload detection results, move the npz file to path: detect
- Generate frame-no and timestamp mapping to support visualization, move the npz file with video into path: viz

5. Object Tracking (Norfair)

In the database, wait for new detection data and create tracking data

6. Serve: Visualize

Periodically, generate visualization and send a trigger to Thingsboard

7. Send: Statistics

Periodically, send statistic to Thingsboard

System Configuration

1. Thingsboard Relationship and Attributes

Asset: server

contains devices: **cctv**

server1

Asset details

?

×

<

Details

Attributes

Latest telemetry

Alarms

Events

>

Outbound relations

Direction

From

▼

+

↺

🔍

<input type="checkbox"/>	Type ↑	To entity type	To entity name		
<input type="checkbox"/>	Contains	Device	cctv_101		
<input type="checkbox"/>	Contains	Device	cctv_102		

Asset admin table				Device admin table	
Entity name ↑	Entity type			Entity name ↑	Entity type
server1	Asset			cctv_101	Device
server2	Asset			cctv_102	Device

Device: cctv

cctv_101

Device details

?

×

Details

Attributes

Latest telemetry

Alarms

Events

Relations

Audit Logs

Shared attributes

Entity attributes scope

Shared attributes

+

↺

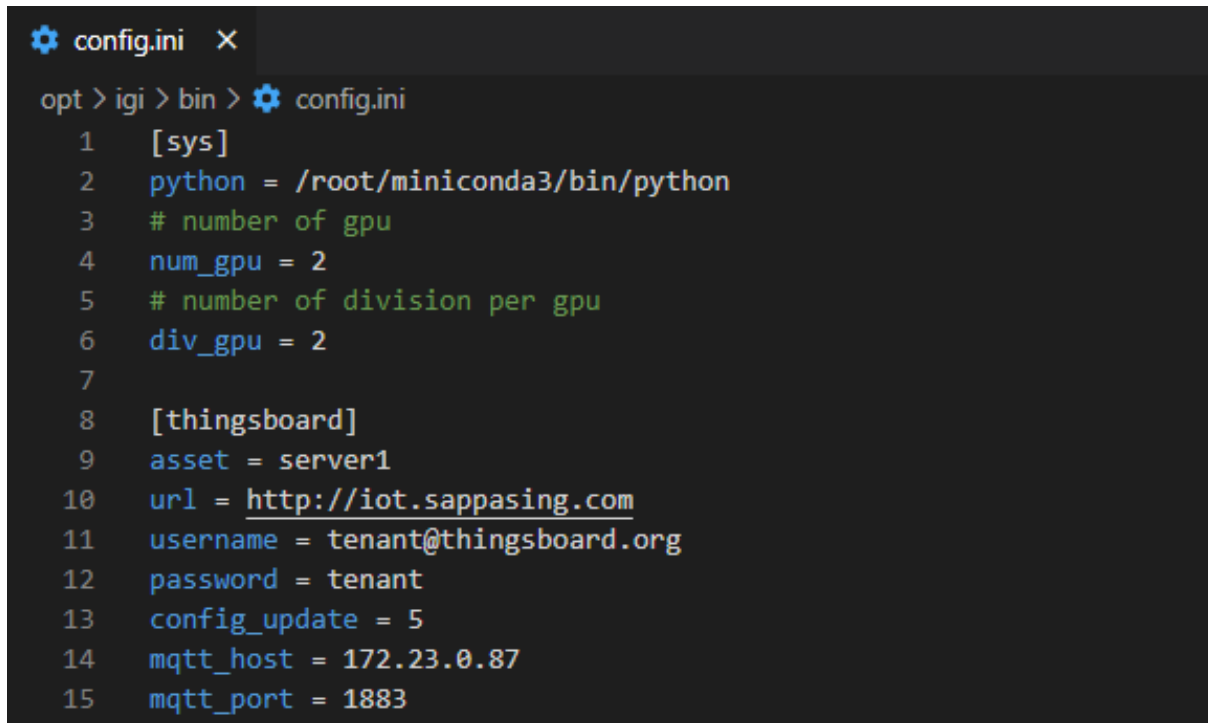
🔍

<input type="checkbox"/>	Last update time	Key ↑	Value	
<input type="checkbox"/>	2021-05-14 13:45:38	cam_param	$\begin{bmatrix} 0.0025 & 0 & -0.88 \\ 0.99298422 & 0.02977351 & 0.11443723 \\ 0.08427805 & 0.50064045 & -0.86154301 \end{bmatrix}$	
<input type="checkbox"/>	2021-05-17 11:21:53	conflict_point	$\{ "point": [[1054, 137], [1116, 251]], "type": "เส้นทึบ" \}$	

Shared Attributes:

Attributes	Description
cam_param	Perspective adjustment matrix
lane_info	Lanes Informaton
rtsp_src	RTSP source (rtsp://host:port/stream)
speed_marker	Zone of LOS checking
conflict_point	Define conflict line that no car should cross
ai_disable	Disable AI processing (if true)

2. Local configuration (config.ini)

A screenshot of a terminal window with a dark background. The title bar shows a gear icon, the text 'config.ini', and a close button 'X'. The terminal prompt is 'opt > igi > bin >'. Below the prompt, the contents of 'config.ini' are displayed, with line numbers 1 through 15 on the left. The configuration includes a '[sys]' section with 'python', 'num_gpu', and 'div_gpu' settings, and a '[thingsboard]' section with 'asset', 'url', 'username', 'password', 'config_update', 'mqtt_host', and 'mqtt_port' settings.

```
opt > igi > bin > config.ini
1  [sys]
2  python = /root/miniconda3/bin/python
3  # number of gpu
4  num_gpu = 2
5  # number of division per gpu
6  div_gpu = 2
7
8  [thingsboard]
9  asset = server1
10 url = http://iot.sappasing.com
11 username = tenant@thingsboard.org
12 password = tenant
13 config_update = 5
14 mqtt_host = 172.23.0.87
15 mqtt_port = 1883
```

3. Database: CCTV Table

Column	Description
id	ID of CCTV (Integer)
device_id	Thingsboard: Device ID
name	Thingsboard: Name of CCTV
access_token	Thingsboard: Access Token
attributes	Thingsboard: CCTV shared attributes
last_update	Attributes last updated
active	Ready for processing (with valid attributes) Required Attributes: 'cam_param', 'lane_info', 'rtsp_src' ** If "ai_disable" = True, CCTV Active = False **

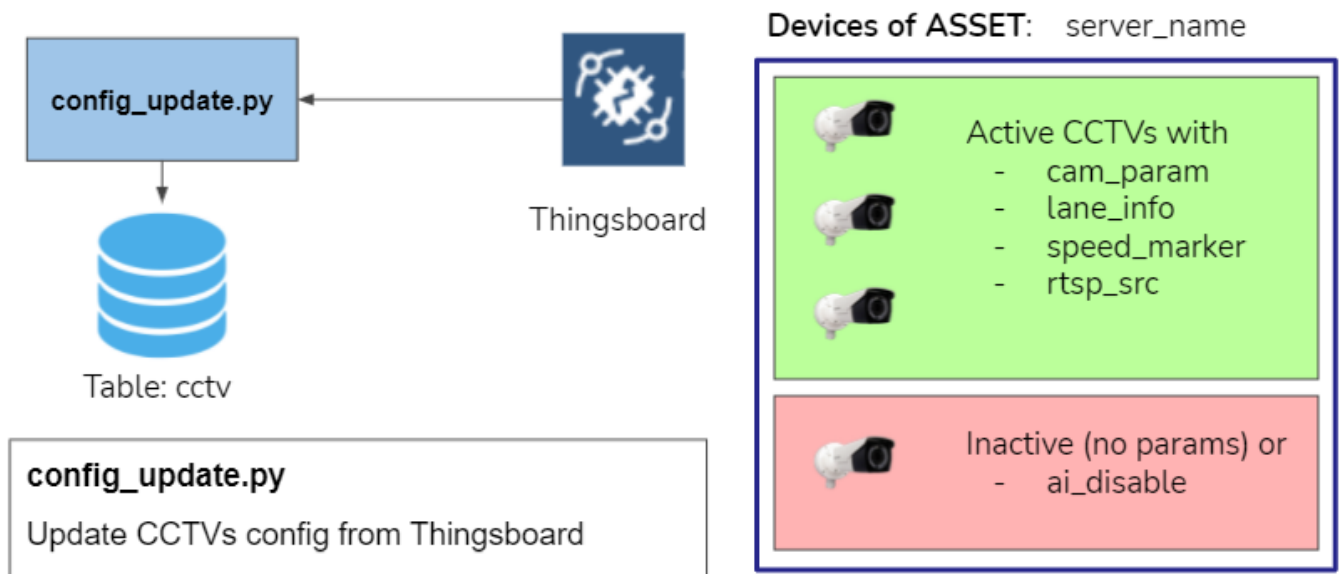
Components

- 1. Config Updater (config_update.py)**
Periodically update CCTV config from Thingsboard to DB
- 2. CCTV Processor (cctv_processor.py)**
Continuously process CCTV data
- 3. Path Monitor (monitor.py)**
Watch and process files in the path

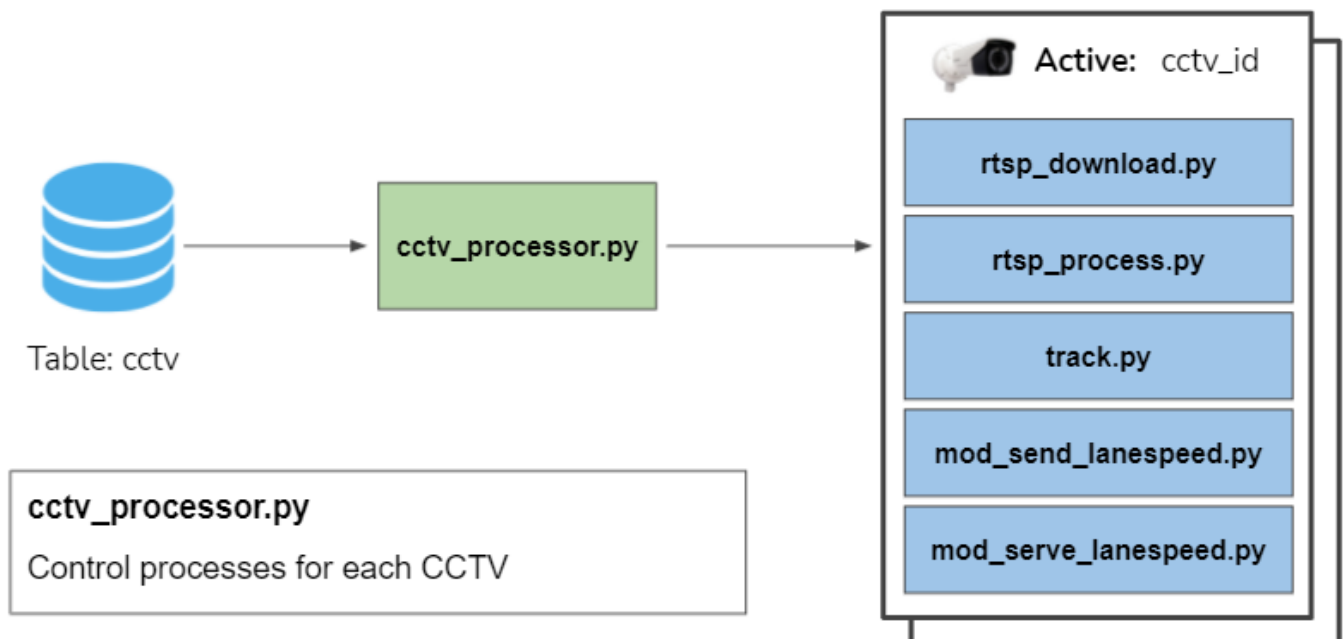
Type of Processing

System processing	Per CCTV processing	Concurrent processing
Config Updater	<div>CCTV Processor</div> <ul style="list-style-type: none">• rtsp_download• rtsp_process• track• mod_send_lanespeed• mod_serve_lanespeed	Path Monitor

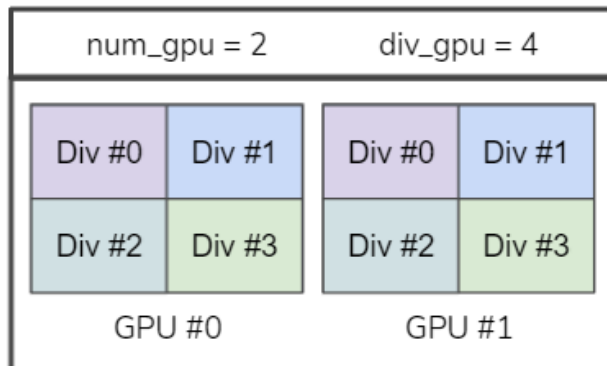
Config Updater



CCTV Processor



Path Monitor



monitor.py

Watch and process files in path

```
monitor.py <path>
├── monitor.py <path> <gpu#> <div#>
├── monitor.py <path> <gpu#> <div#>
├── ...
└── monitor.py <path> <gpu#> <div#>
```

Input Path	Operation	Output Path
incoming	detect.py	queue
queue	upload.py	viz, detect

CCTV Processing Status

Process	Key
RTSP Download / Process	rtsp_ts
Detect / Upload	detect_ts
Track	track_ts
Send Statistic	send_ts
Serve Visualization	serve_ts

cctv_101
Device details

Details Attributes Latest telemetry Alarms Events Relations

Client attributes Entity attributes scope Client attributes

Last update time ↓	Key	Value
2021-05-21 15:05:51	send_ts	1621585123582
2021-05-21 15:05:50	track_ts	1621585064076
2021-05-21 15:05:43	detect_ts	1621585064160
2021-05-21 15:05:05	serve_ts	1621584823582

Programs and command-line arguments

[config_update.py]

Update CCTV config from Thingsboard

```
python config_update.py [--sync sec]
```

[detect_opencv.py]

Object detection - YOLOv4 OpenCV DNN

```
python detect_opencv.py <source_video> [--weights model_weight_file] [--config  
model_config_file] [--output out.npz]
```

[detect.py]

Object Detection - PyTorch YOLOv4

```
python detect.py <source_video> [--weights model_weight_file] [--config  
model_config_file] [--output out.npz] [--device cuda_device]
```

[makedirs.py]

Make all required data dirs

[mod_avg_lanespeed.py]

Calculate average lanespeed and number of cars

```
python mod_avg_lanespeed.py <cctv_id> <start_ts> <stop_ts>
```

[mod_conflict_report.py]

Generate conflict report

```
python mod_conflict_report.py <cctv_id> <start_ts> <stop_ts> <out.csv>
```

[mod_LOS.py]

Generate LOS report

```
python mod_LOS.py <cctv_id> <start_ts> <stop_ts> <time_step> <out.csv>
```

[mod_send_lanespeed.py]

Send statistic data to Thingsboard

```
python mod_send_lanespeed.py <cctv_id> [last_ts]
```

[mod_serve_lanespeed.py]

Serve visualize video and send trigger to Thingsboard

```
python mod_serve_lanespeed.py <cctv_id> [last_ts]
```

[mod_viz_detect.py]

Visualize object detection

```
python mod_viz_detect.py <cctv_id> <start_ts> <stop_ts> <out.mp4>
```

[mod_viz_lanespeed.py]

Visualize Lane speed

```
python mod_viz_lanespeed.py <cctv_id> <start_ts> <stop_ts> <out.mp4>
```

[mod_viz_track.py]

Visualize Object Tracking

```
python mod_viz_track.py <cctv_id> <start_ts> <stop_ts> <out.mp4>
```

[mod_viz_trajectory_conflict.py]

Visualize Lane speed with trajectory view

```
python mod_viz_trajectory_conflict.py <cctv_id> <start_ts> <stop_ts> <out.mp4>
```

[monitor.py]

Monitor and process files in the path

```
python monitor.py <queue|incoming> [<gpu_id> <div_id>]
```

[rtsp_download.py]

Download video from RTSP source

```
python rtsp_download.py <cctv_id>
```

[rtsp_process.py]

Check and move RTSP video to path: incoming

```
python rtsp_process.py <cctv_id>
```

[start.sh]

Start the workflow

[stop.sh]

Stop the workflow

[track.py]

Object tracking with Norfair

```
python track.py <cctv_id> [last_ts]
```

[upload.py]

Upload object detection result to DB

```
python upload.py <input_video>
```