CTS Surveyor User Guide

User Guide

January 7, 2019

Developer

Caerus Technical Solutions, Inc.

Contents

Revision History

Version	Date	Name	Description
1.1.0.1	12/15/18	Release management	Initial Document
1.2.0.1	01/07/19	Release management	Updated for version 1.2.0.1
3.0.0.1	04/05/19	Release management	Updated for version 3.0.0.1
4.0.0.1	08/28/19	Release management	Update for version 4.0.0.1

1 Introduction

The CaerusTech Surveyor application is designed to monitor people walking by the camera and perform age and gender recognition of people's faces. This document will provide guidance on how to use the software from the application developer's perspective.

2 Start Application

After unzipping the contents of the distributable package, go to the root directory and run *start_surveyor.bat* for the demo application. This will start the surveyor server which will initialize the environment and will open the camera. By default, it will present a camera feed window for visualization: detected people, faces and their ages and genders, and some basic statistics will be shown along with the output

3 Config

Root directory contains *config.ini* file where some configuration parameters are defined and can be changed. Please note that there are more configurable parameters can be found in the *config.ini* than are listed below, but at the moment it is not advisable for the user to change those parameters due to the unpredictable results of those changes. The following table defines the changable parameters:

Parameter	Type	Description
DEFAULT_READ_FACES	boolean	Determines whether the application will detect faces by default after startup: if set to False, the application will not be detecting faces until this feature is enabled through the control API
DEFAULT_COUNT_PEOPLE	boolean	Determines whether the application is to detect and count people after startup: if set to False, the application will not be detecting people until this feature is enabled through the API
WAIT_BETWEEN_FRAMES	double	This is a value in seconds that defines how long the application needs to wait between the frames. This feature is designed to reduce CPU load. For example, setting it to 0.25 will make the application sleep for 250 ms between each frame
CAMERA_WIDTH	numeric	Horizontal resolution of the image from camera
CAMERA_HEIGHT	numeric	Vertical resolution of the image from camera
VISUAL_OUTPUT	boolean	If set to True, the application will present a window with the feed coming from camera where faces and people will be indicated; this feature can be use for testing and demo purposes. In production, this setting is expected to be set to False
MINIMUM_FACE_SIZE	numeric	This value represents a minimum height/width of a face in pixels when the facial detection and classification is to be triggered. The larger the value, the closer the person needs to be to the camera before the detection will be run, and thus more reliable facial classification can be made
FACE_FINDER	string	CaerustechSurveyor is equipped to either use haar cascade face or deep learning-based detector. Set it to *"FaceFinderCV"* to use haar cascade classifier or *"FaceFinderDL"* to use deep-

		learning face finder.
FACE_DETECTION_CONFIDENCE	double	This value is used when FACE_DETECTION_STRATEGY is set to *FaceFinderDL* and defines when to detect face based on the confidence level that it is indeed a face
DAYS_TO_KEEP	numeric	A number of days for how long statistical data will be present in the Surveyor's internal database
LOGGING_LEVEL	numeric	Numeric value that represent the logging level
WAIT_BETWEEN_CLASSIFICATIONS	double	This value determines how long the face classification thread should wait between the classification attempts, in seconds. The less the value, the faster the face can be classified but the more of the CPU will be used
RECORD_PEOPLE	boolean	If set to "True", the Surveyor will start a recording whenever a human presence is detected, and will continue recording until no humans are present in front of the camera
SAVE_CLASSIFIED_FACE	boolean	If set to "True", the Surveyor will save the cropped face after the facial classification is complete

4 API

4.1 Event notification

A client can subscribe to the application via web socket at "ws://localhost:6789/". Whenever a facial recognition or a person detection event occurs, the client will receive an event in the following format:

```
{
  "events": [
    {
      "event": "event_type",
```

```
      event-specific data
    }
  ]
}
```

The following table defines event types with which a client application needs to be concerned:

Event	Event Type	Example

Connected to Web Socket	connected	<pre>{
		  "events": [
		    {
		      "event": "connected"
		    }
		  ]
		}</pre>

People Update	people_update	<pre>{
		  "events": [
		    {
		      "people": [
		        {
		          "rectangle": {
		            "x": 152,
		            "y": 72,
		            "height": 417,
		            "width": 616
		          },
		          "id": 86
		        }
		      ],
		      "event": "people_update"
		    }
		  ]
		}</pre>

4.2 Control API's

It is possible to enable and disable individual features as client application's developers see fit. For example, when the facial detection/classification is not required, it can be disabled so that CPU load could be reduced. These commands can be executed by issuing HTTP GET requests to http://localhost:5000/ , with the following options:

- /enable_feature?feature=face
- /enable_feature?feature=people
- /disable_feature?feature=face
- /disable_feature?feature=people

If the request was executed successfully, the client will get back the following response:
{"result":"success"}

Statistical API's

At any point of time, a client application can retrieve people data through Surveyor's statistical API's by issuing GTTP GET to http://localhost:5000/ , where the following endpoints are available:

4.2.1 List all observed people

Endpoint	/stats
Purpose	Used to retrieve all detected individual classification results for the requested period of time
Request	/stats?start=YYYYMMDDThhmmss&end= YYYYMMDDThhmmss
Request example	/stats?start=20190207T000000&end=20190408T235959

Response	JSON array of the person elements, as shown below:
	
	`[`
	`  {`
	`  "id": 114,`
	`  "arrived": "2019-03-25 18:58:17.347425",`
	`  "left": "2019-03-25 18:58:31.554653"`
	`  }`
	`]`

4.2.2 Get hourly counts

API	/hourly_stats
Purpose	Used to retrieve counts by the hour for each class
Request	/hourly_stats? YYYYMMDDThhmmss&end= YYYYMMDDThhmmss
Request example	/hourly_stats?start=20190207T000000&end=20190407T235959

Response	JSON array of class count elements for every hour of the requested time range, as shown below:
	
	`[`
	`  {`
	`  "count": [`
	`  7`
	`  ],`

	<pre>"start": "2019-08-01 00:00:00",
	"end": "2019-08-01 00:59:59"
	}
	]</pre>

4.2.3 Get total counts

API	/total_stats
Purpose	Used to retrieve total counts for each class for the time range
Request	/ total_stats? YYYYMMDDThhmmss&end= YYYYMMDDThhmmss
Request example	/ total_stats?start=20190207T000000&end=20190407T235959

Response	JSON message with counts, as shown below:
	<pre>{
	"count":[12]
	}</pre>

4.2.4 Get face stats

API	/face_stats
Purpose	Used to retrieve a list of all faces detected and classified in the given time frame
Request	/ face_stats? YYYYMMDDThhmmss&end= YYYYMMDDThhmmss
Request example	/ face_stats?start=20190207T000000&end=20190407T235959

Response	JSON message with counts, as shown below:
	<pre>[
	{
	"id": 504,
	"age_lower": 42,
	"age_higher": 44,
	"gender": "male",
	"detected_time": "2019-08-26 07:43:11.676694"
	}
	]</pre>

4.2.5 Get face stats by the hour

API	/hourly_face_stats
Purpose	Returns face classification result counts by the hour
Request	/ hourly_face_stats? YYYYMMDDThhmmss&end= YYYYMMDDThhmmss
Request example	/ hourly_face_stats?start=20190207T000000&end=20190407T235959

Response	JSON message with counts, as shown below:
	
	[
	  {
	  "male_count": 32,
	  "female_count": 17,
	  "child_teen_count": 5,
	  "young_adult_count": 14,
	  "adult_count": 18,
	  "middle_aged_count": 9,
	  "senior_count": 3,
	  "start": "2019-08-29 08:00:00",
	  "end": "2019-08-29 08:59:59"
	  }
	]

4.2.6 Get total face counts

API	/total_face_stats
Purpose	Returns face classification result counts for the given timeframe
Request	/ total_face_stats? YYYYMMDDThhmmss&end= YYYYMMDDThhmmss
Request example	/ total_face_stats?start=20190207T000000&end=20190407T235959

Response	JSON message with counts, as shown below:
	
	{
	  "male_count": 32,
	  "female_count": 17,
	  "child_teen_count": 5,
	  "young_adult_count": 14,
	  "adult_count": 18,
	  "middle_aged_count": 9,
	  "senior_count": 3
	}