VMSS 2.0

New Generation of Aupera Video Machine Learning Streaming Server

Quick Start User Guide

Document Revision: 2.1.0

1 RUNNING AUPERA VMSS2.0 ON VMACCEL

In this section, we will introduce the steps to sign-up for the VMAccel demo account, access Aupera Video Machine-learning Streaming Server 2.0 (VMSS2.0) instance and launch custom RTSP streams for tasks. Users can go through these steps to open the door to using the functionalities of Aupera VMSS2.0 AI pipelines in later sections.

1.1 Launching Aupera VMSS2.0 instance on VCK5000

Please sign up for a demo account at: https://www.vmaccel.com/vmssdemo

After completing the sign-up form, you will receive an email with your demo credentials. Please follow the instructions in the email to log into VMAccel. Please note that this account will allow you to evaluate VMSS2.0 for free for 5 hours.

NOTE: The trial accounts allow 5 hours of free evaluation of VMSS2.0. The trial accounts are currently configured to automatically delete any instances when a user logs out. You may use a total of 5 hours of runtime, and this could be extended over several days. Your account will be locked once you have depleted 5 hours of runtime.

For each user, a VMAccel instance will be automatically launched. Once you login, under *Project->Compute* you should see your instance by selecting "*Instances*" in the left-hand side bar. You should be able to see the instance that was just created for you being spawned as shown in Figure 1.1.

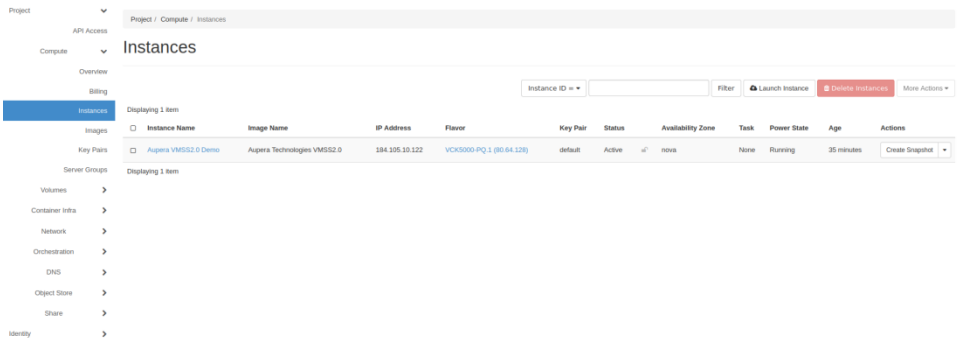

Figure 1.1. VMAccel instances main page

After some time, the status of the instance will change to Active. At this point, the installation of the VMSS2.0 server and the client on the instance begin. Please allow about 3 minutes for this process to finish after you see the status has changed to Active.

VMSS 2.0 consists of two major modules, Aupera Video AI Client (AVAC) and Aupera Video AI Server (AVAS). AVAC allows users to use a user-friendly GUI to connect to AVAS. Additionally, more advanced users may work with AVAS via the command line directly. More detailed information would be introduced in later chapters.

NOTE 1: We may use Aupera Video AI Client (AVAC) and Aupera's VMSS2.0 AI Client interchangeably throughout this document. We may also use Aupera Video AI Server (AVAS) and Aupera's VMSS2.0 AI Server interchangeably throughout this document.

NOTE 2: To run VMSS2.0 pipelines, you can either use Aupera's VMSS2.0 Web Client (as described in **Section 2**) or access the VMSS2.0 server (as described in **Section 3**) using the command line.

NOTE 3: To facilitate testing, the VMAccel instance will automatically start three RTSP streams:

rtsp://<vmaccel_instance_ip_address>:8554/retail
rtsp://<vmaccel_instance_ip_address>:8554/car
rtsp://<vmaccel_instance_ip_address>:8554/crowd

vmaccel_instance_ip_address is the IP address that is shown in the image above under IP Address when you are in *Project->Compute->Instances*. The first stream contains a crowded scene of people passing by and is the most useful for testing head, person, and other human related detections. We use this video for benchmarking our crowd applications. The second stream contains objects usually encountered in a retail scenario. We use this stream for throughput benchmarking.

Additionally, AVAC, the VMSS2.0 Web Client, can connect to any RTSP streams that are broadcast on open ports.

2 RUNNING AUPERA VMSS2.0 PIPELINES THROUGH WEB CLIENT

In this section, we will describe the steps to run the Aupera VMSS2.0 pipelines through the VMSS2.0 Web Client. It is mainly divided into three subsections: accessing the Aupera VMSS2.0 Web Client, running the crowd flow application, and launching custom pipelines. Users can use RTSP streams to run different tasks through the Web Client interface and easily view the task running results.

2.1 Accessing VMSS2.0 Web Client

After launching an Aupera VMSS2.0 instance on VMAccel, you can access the web client by copying the IP address of your instance into your browser on your local machine; and adding the port 3004. For example, in the screenshot below, the IP address of the instance is 184.105.10.164 which means that the web client can be accessed by typing http://184.105.10.164:3004/ in your browser.

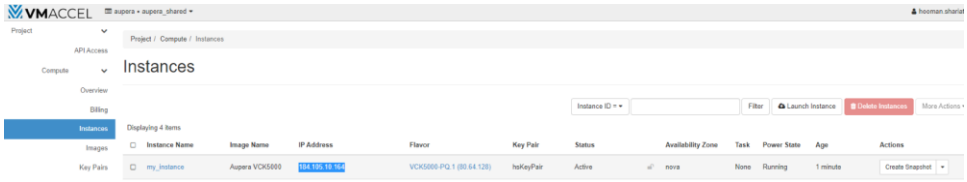

Figure 2.2. VMAccel instances page with instance IP address highlighted

NOTE: Since the relevant ports of your VMAccel instance are open to the public, you do not need to use the browser of the VMAccel instance (accessible through VNC) to launch the Web Client. You can use the browser on any machine with an internet connection.

2.2 Add Camera to Access Applications

A. Click ***Add Camera*** to add at least one camera to access the Application functions.

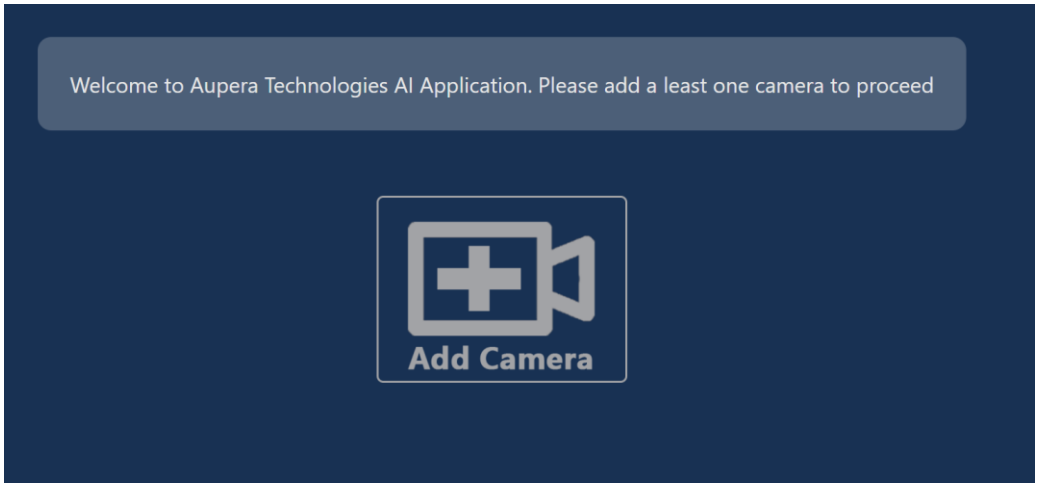

Figure 2.3. Aupera web application page

B. Enter a ***Name*** (any arbitrary name can be used) and ***URL*** (***S/N*** is not required). **Make sure that RTSP URL is correct.** You can right-click on the added camera to access the camera configuration shown in Figure 2.4.

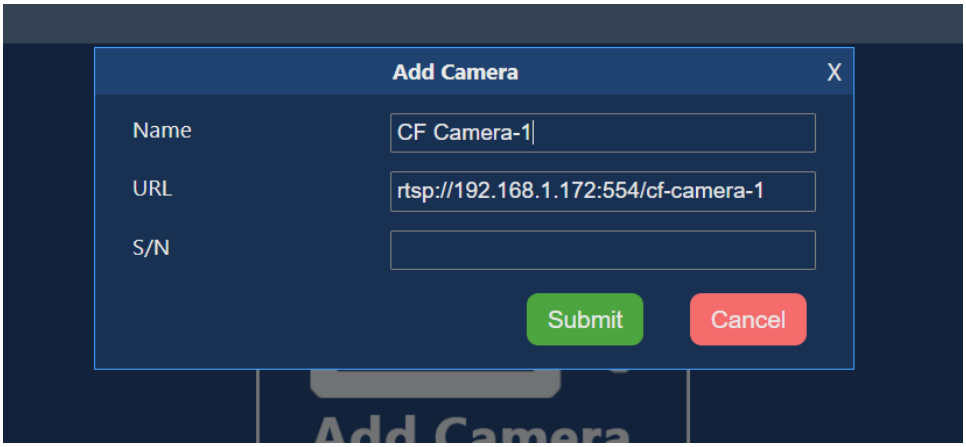

Figure 2.4. Aupera web application page – add camera

NOTE: You can use either your own publicly available RTSP stream (perhaps the one you launched using the instructions in Section 2.2); or it can be one of the three RTSP streams that automatically started up when you launched your VMAccel instance.

For the latter you can use one of the following addresses:

rtsp://<vmaccel_instance_ip_address>:8554/retail
rtsp://<vmaccel_instance_ip_address>:8554/car
rtsp://<vmaccel_instance_ip_address>:8554/crowd

Just make sure to replace the **vmaccel_instance_ip_address** with the IP address of the VMAccel instance you just launched. You can find this IP address by clicking on the left-hand sidebar select “Instances” you can see the instance you just created being spawned as shown in Figure 2.5.

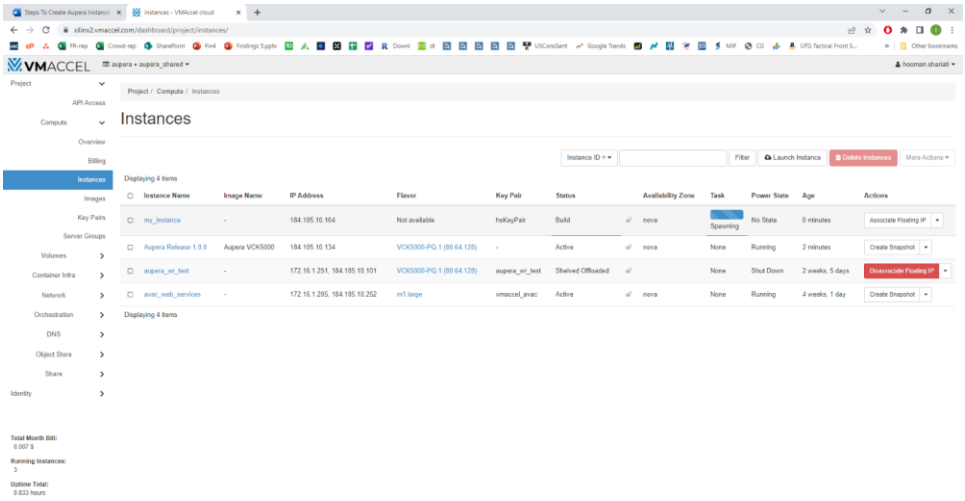

Figure 2.5. VMAccel instances page – preparing a new instance

Again, the first stream shows a scene with people (used for crowd, person, and person attributed applications) while the second stream shows a scene with retail objects.

2.3 Running Aupera's Crowd Flow Application

A. Click ***AI Tasks Hub***, then click ***Add New Task*** button, then there will be a drop down to select a camera

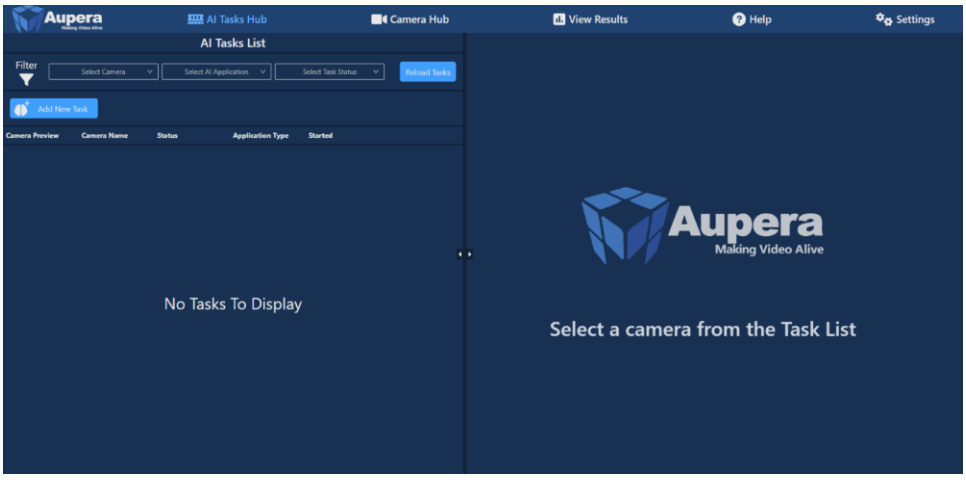

Figure 2.6. Aupera web application page – AI Tasks Hub Page

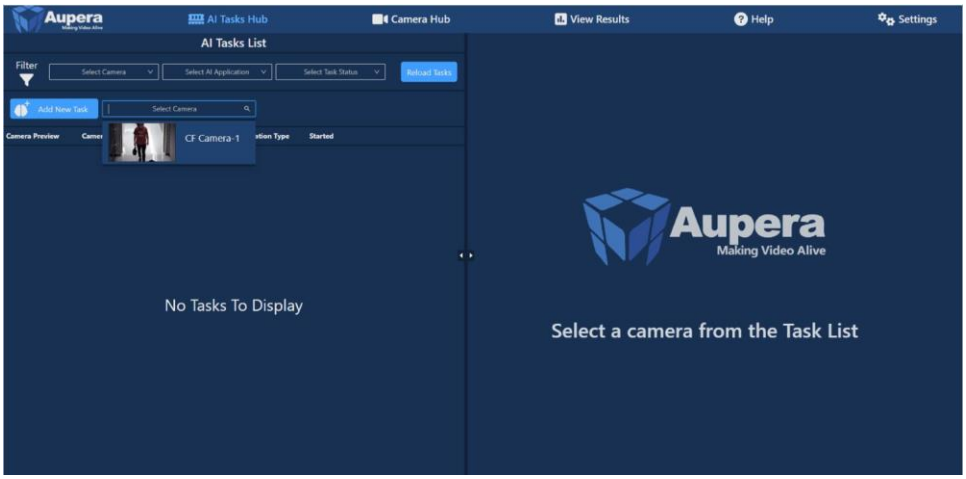

Figure 2.7 Aupera web application page – AI Tasks Hub Page, Add New Task button clicked

B. After selecting a Camera, another drop down will appear to select an application. Select Crowd Flow to open Crowd Flow application set up controls.

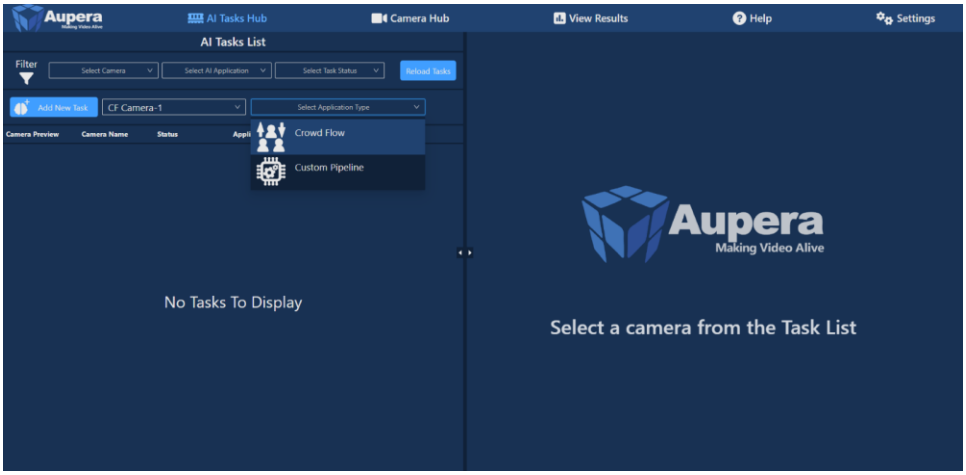

Figure 2.8 Aupera web application page – AI Tasks Hub Page, Camera Selected

C. After selecting Crowd Flow, the Crowd Flow Task Setup window will appear.

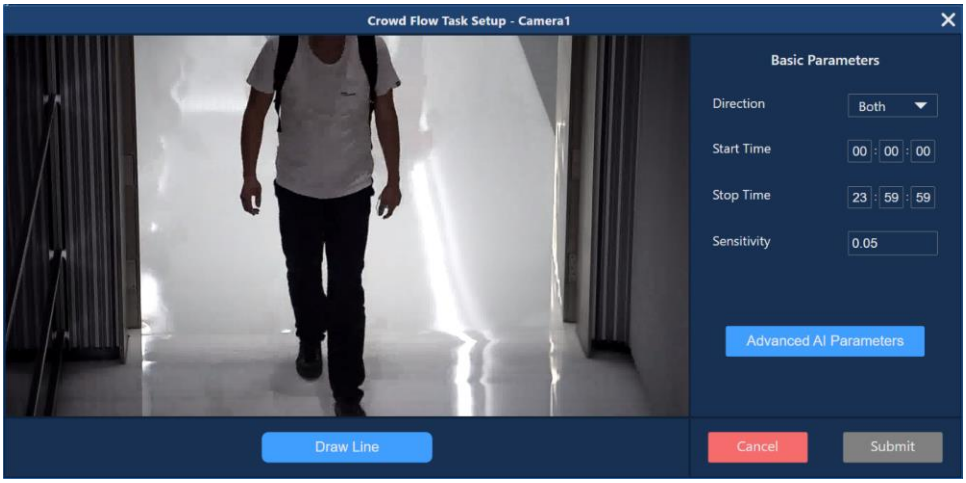

Figure 2.9. Aupera web application page – crowd flow task creation

D. You need to draw a line to indicate a border for people crossing. When a person's head crosses the line, IN/OUT count will reflect this event. These border lines may consist of up to a maximum of 14 segments, a less segmented line provides more accurate results.

1. To start drawing, click the ***Draw Line*** button, the cursor will change to a cross.

2. **Left-click** and **hold the mouse button** on the place where you would like to start the line.

3. Drag the Line to the place where you want to finish the first segment, then release the mouse button.

4. Move a mouse to the end of the next segment to complete it.

5. To finish drawing, click the **Right mouse button**, unfinished segment will be deleted.

After the Line is drawn, "Draw Line" will change to "Redraw Line". Click it if you want to delete the Line drawn and start drawing from the beginning. It is recommended to draw U shaped lines as shown in Figure 2.9 to capture people who may move parallel to the line and around it.

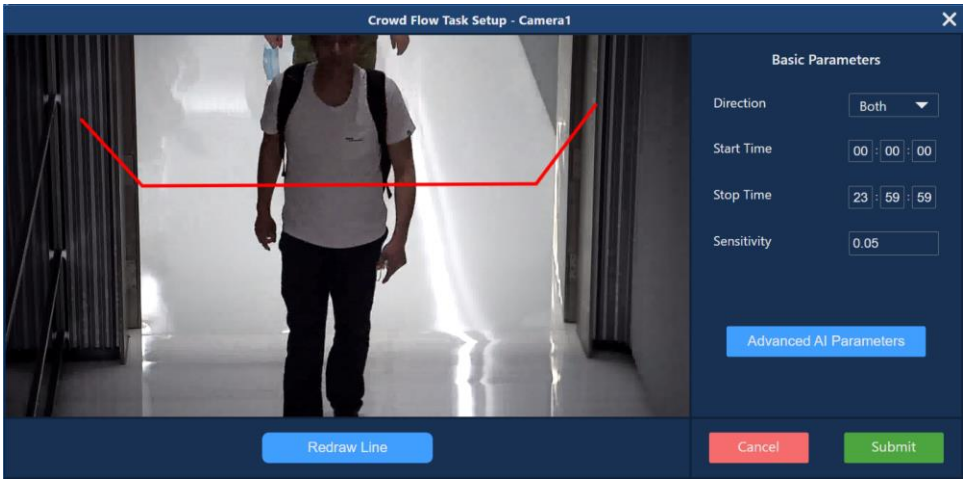

Figure 2.10. Aupera web application page – crowd flow task lines drawing example

E. Changing Basic Parameters is not required to start the task, you can keep default values.

1. ***Direction*** – Count people going “In”, “Out” (“Entering” / “Exiting”) or both directions

2. ***Start/Stop Time*** – counting will be started and stopped at the given time every day;

3. It is recommended to set ***Sensitivity*** to the default value of 0.05;

F. Advanced Parameters can significantly affect the results for a particular task, it is recommended to not change those until recommended by Aupera.

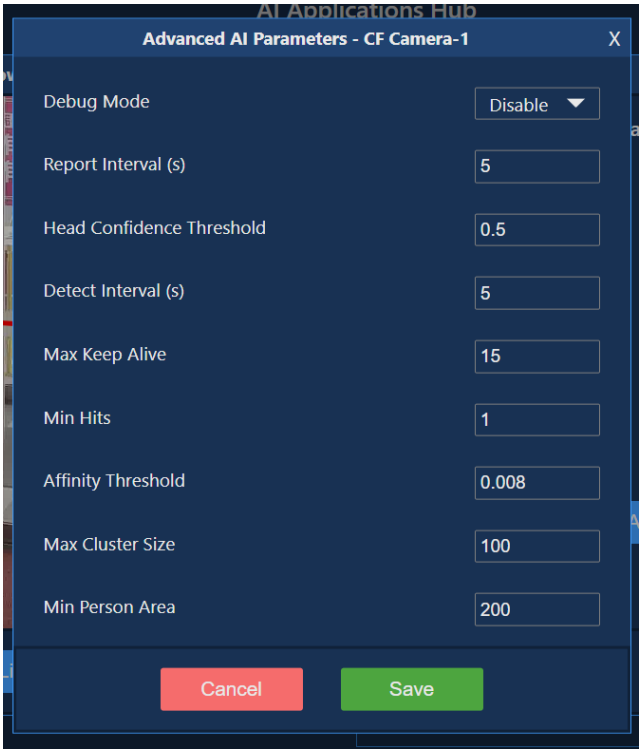

Figure 2.11. Aupera web application page – crowd flow task advanced AI parameters

G. To start the task, click the ***Submit*** button. After that, a pop-up message will notify you that the task was successfully launched.

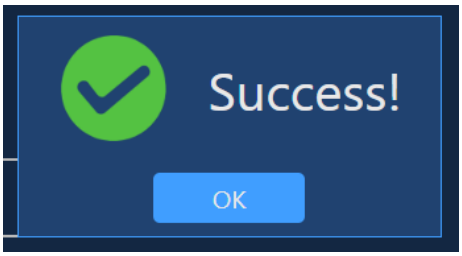

Figure 2.12. Aupera web application page – success notification

H. If the pop-up message reports an error, try launching the task with default parameters or check the settings.

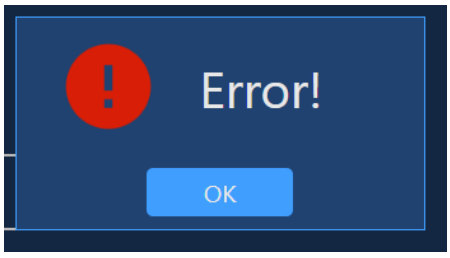

Figure 2.13. Aupera web application page – error notification

I. If the task was launched, the task will appear in the task table on the left side of the screen

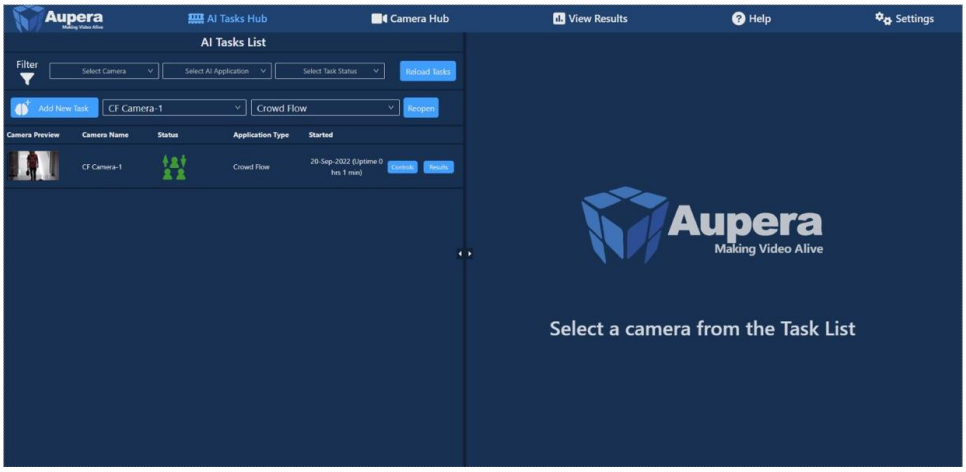

Figure 2.14. Aupera web application page – crowd flow task control

J. Select a task in task list or click ***control*** button on the row of the task to show the Crowd Flow Control on the right side of the screen.

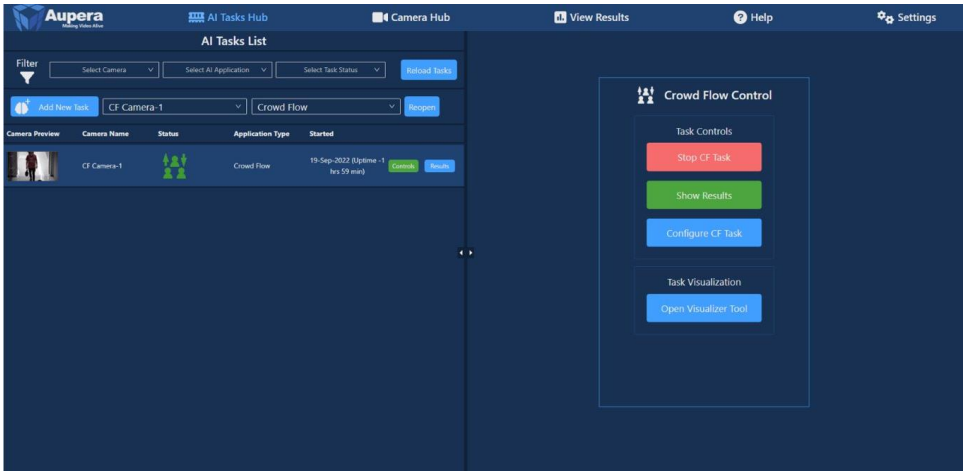

Figure 2.15. Aupera web application page – crowd flow task control

K. To view results, click the ***Show Results*** button or the ***result*** button on the specific task row on the table. You will be redirected to the Results page and a corresponding camera will be selected automatically. If you have more than one task launched, you can switch between them with the ***Display Results For*** drop-down box. More than one Crowd Flow task can be launched per camera, so the dropdown box will display both Camera name and Task ID to help distinguish task between each other.

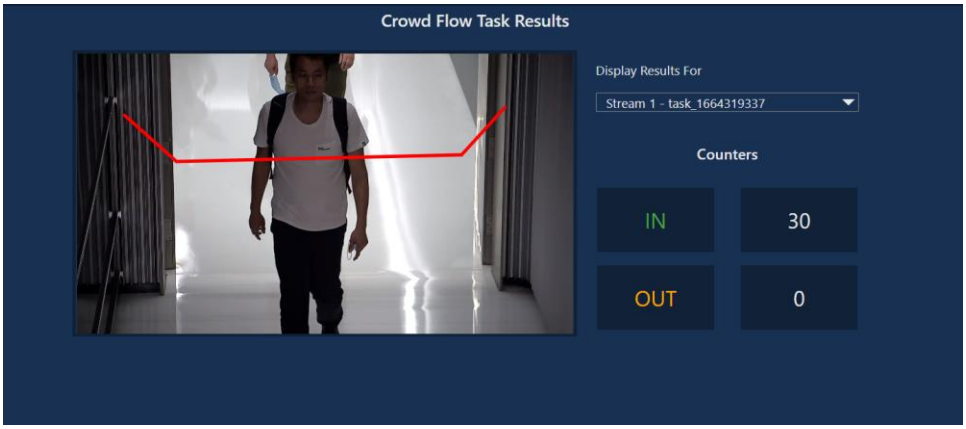

Figure 2.16. Aupera web application page – crowd flow task result

L. To visualize the AI pipeline of the task, click the ***Open Visualizer Tool*** Button at the bottom of the controls.

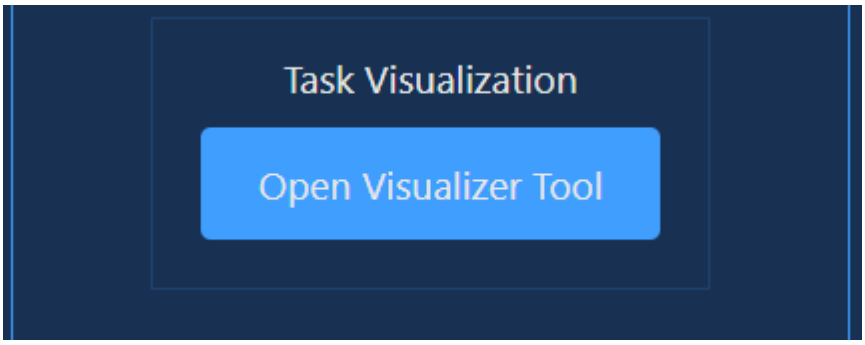

Figure 2.17. Aupera web application page – crowd flow task visualizer tool

The Visualizer will open in a new tab. Then, the AI pipeline graph representing current Crowd Flowd task will appear.

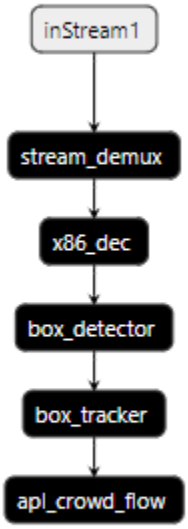

Figure 2.18. Aupera web application page – crowd flow task AI pipeline graph

M. To update Crowd Flow task set up, click ***Configure CF Task*** button, then the Crowd Flow Task Setup window will appear. Any parameter of the running task can be changed. See Figure 2.10 to see more details in Crowd Flow Task set up. After completing the changes to Crowd Flow Task setup, click ***Update*** button to save changes. Updating task does not reset result counters.

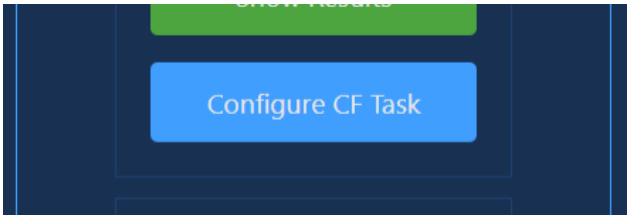

Figure 2.19. Aupera web application page – crowd flow Configure Crowd Flow Task Button

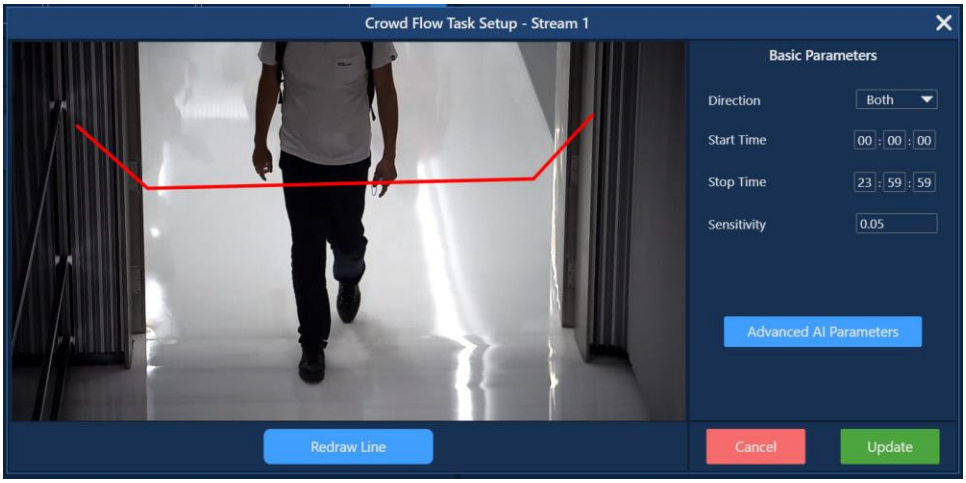

Figure 2.20. Aupera web application page – Configure Crowd Flow Task popup

N. To stop task, click ***Stop CF Task*** button, then Stop CF Task confirmation will notify that the current task will be stopped. To proceed deleting the task, click ***OK*** button. To undo stop task, click ***Cancel*** button.

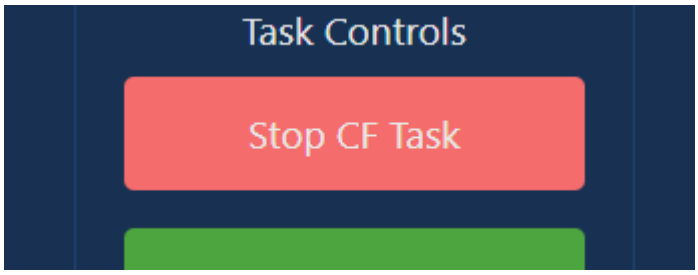

Figure 2.21. Aupera web application page – crowd flow Stop Crowd Flow Task Button

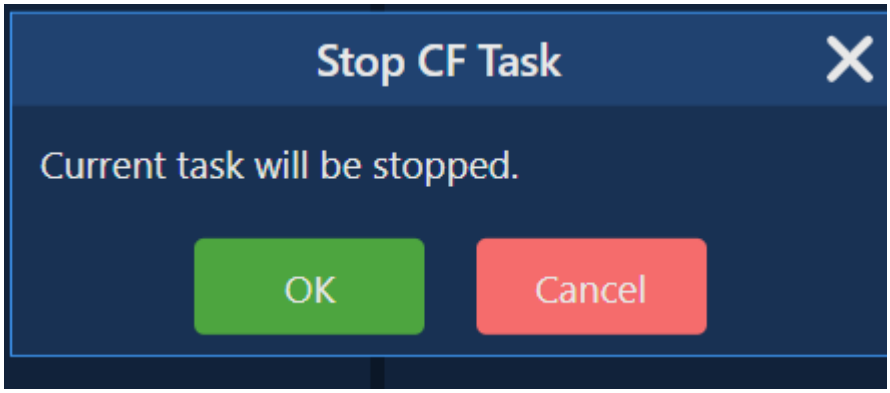

Figure 2.22. Aupera web application page – crowd flow Stop Crowd Flow Task confirmation popup

2.4 Running Custom Pipelines

A. Click ***Add New Task*** button, select a camera from the drop down, and in ***Select Application Type*** drop down, select Custom Pipeline

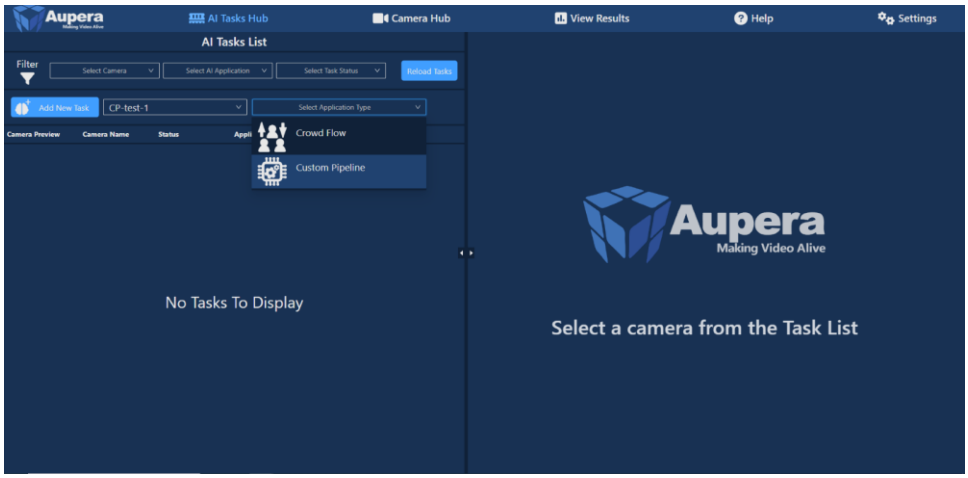

Figure 2.23 Aupera web application page – AI Tasks Hub Page, Camera Selected

B. After selecting Custom Pipeline, AI Editor will appear. In the Editor you can type a PBTEXT configuration or ***Import From File*** it from a file by clicking the corresponding button.

NOTE: To run custom pipelines, please use the examples provided in section 2.4.3 (also accessible here). If you're running Custom Pipeline through Aupera Web Client, please only use the files that include "using_rtsp" in their name.

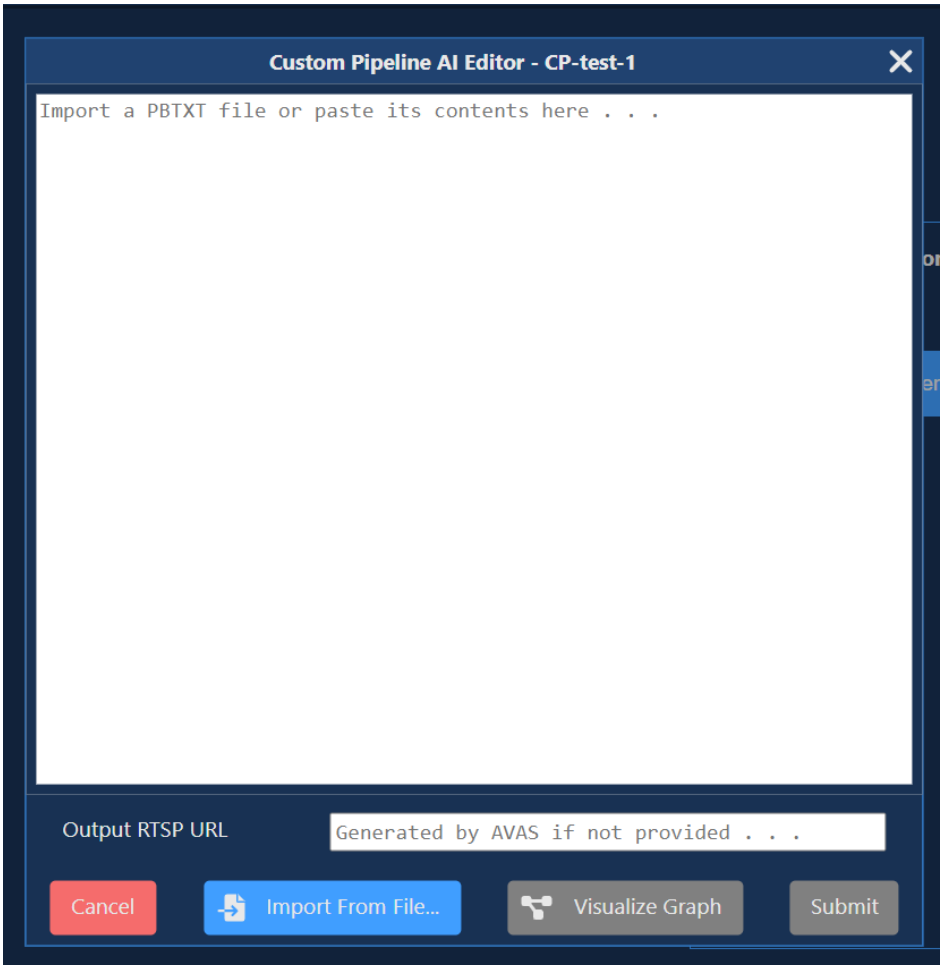

Figure 2.24. Aupera web application page – custom pipeline task AI editor

C. After PBTEXT has been typed or imported, it can be visualized. Clicking on the "*Visualize*" button will open a new tab in which the Graph will be displayed.

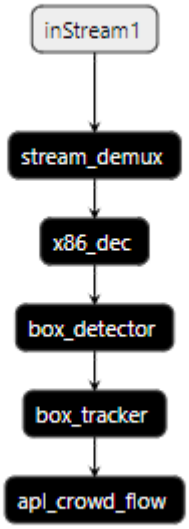

Figure 2.25. Aupera web application page – custom pipeline task AI pipeline graph

D. Before starting the task, please enter *Output RTSP URL* (this value needs to follow a specific format of **rtsp://<vmaccel_instance_ip_address>:8554/<user_specifcied _name>**). Click Submit to start the CP task.

NOTE: **<user_specifcied_name>** can be any arbitrary name that the user chooses.

E. Whether the task was started successfully or not, a corresponding message will be displayed as the pop-up.

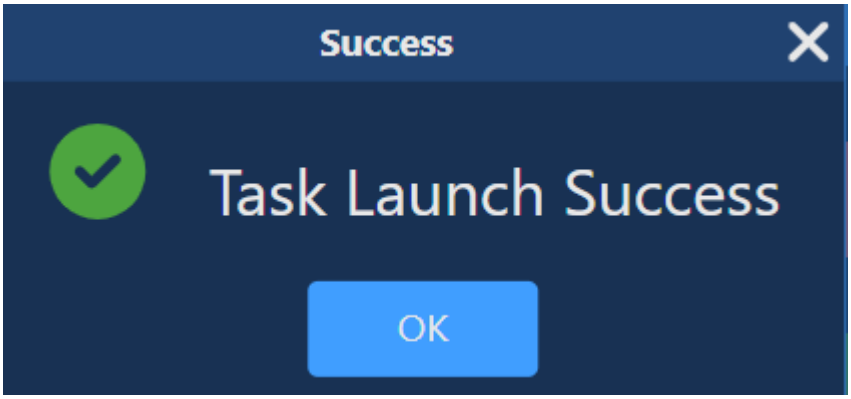

Figure 2.26. Aupera web application page – task launch success notification

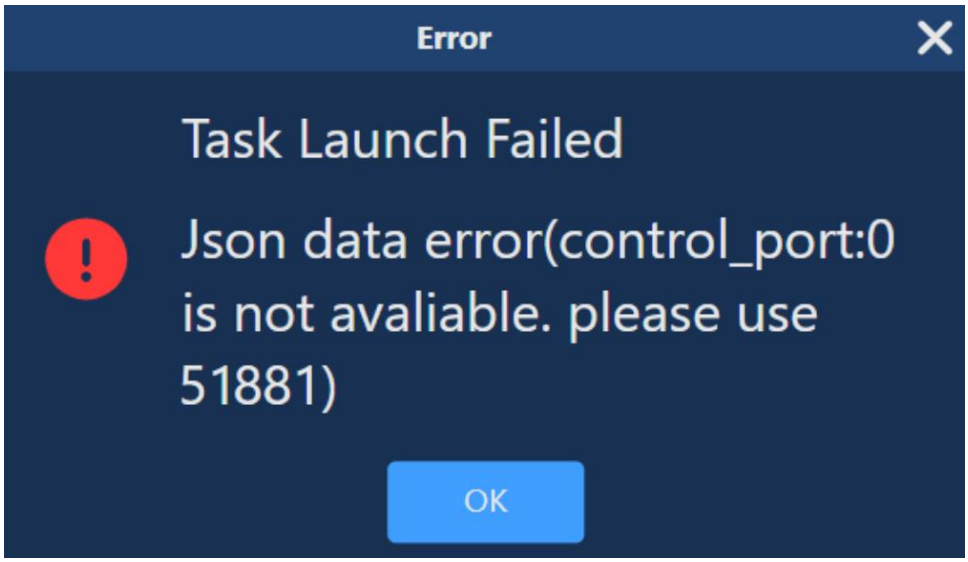

Figure 2.27. Aupera web application page – task launch failed notification

F. If the task was started, but then crashed after some time, message about that will be displayed

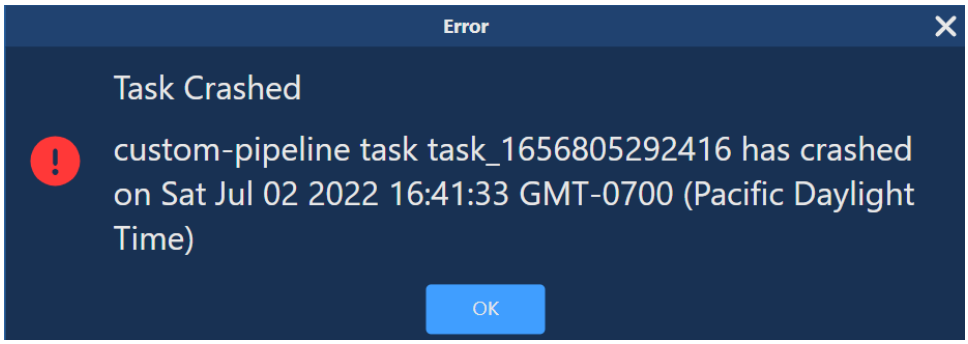

Figure 2.28. Aupera web application page – task crashed notification

G. When the task is launched, it will appear in the task list. When the task row in task list or the ***control*** button on the row is clicked, the right side of the screen will show the CP control component.

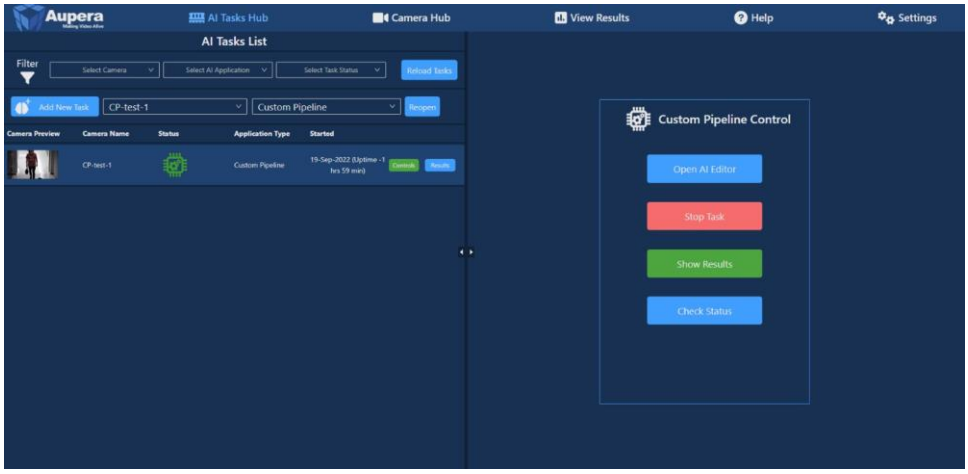

Figure 2.29. Aupera web application page – custom pipeline task control

H. If AI Editor is opened for a running task, current PBTEXT configuration will be displayed in the editor field. However, task update is not supported now, so please stop and start the task again in case any changes in PBTEXT are required.

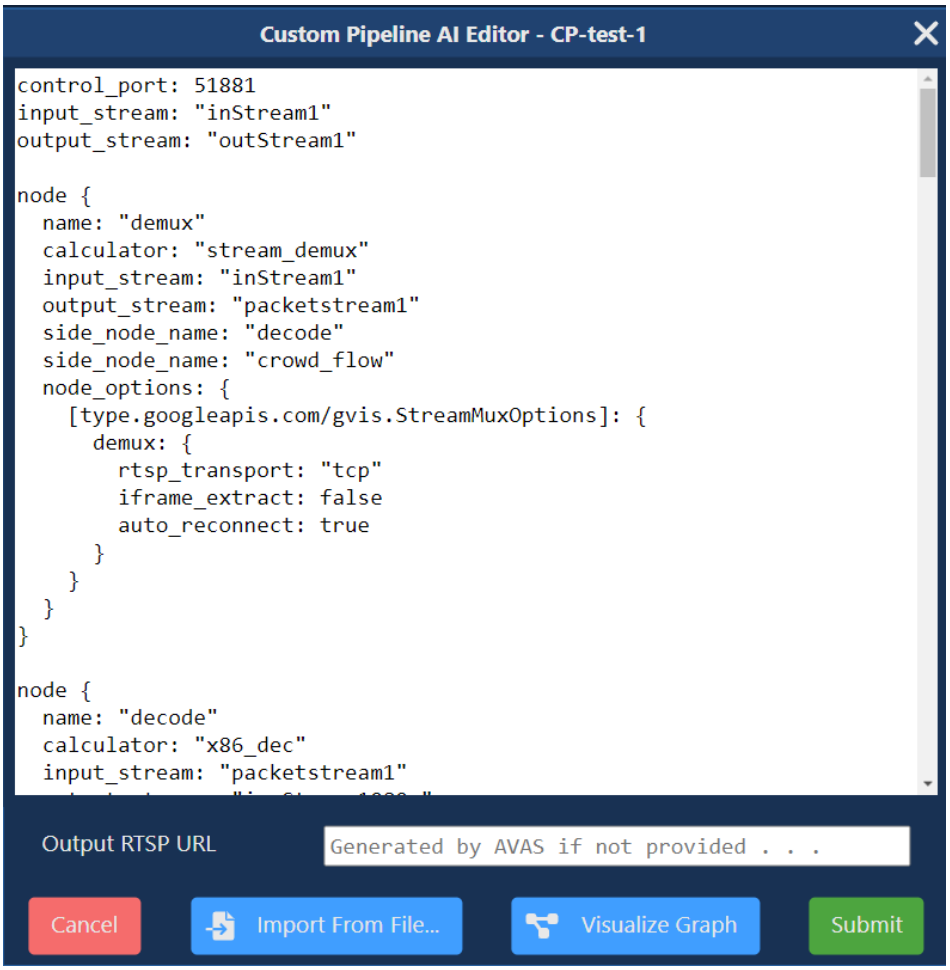

Figure 2.30. Aupera web application page – custom pipeline task pbtxt example

I. To see the CP task results (output video), either click on ***Show Results*** in the CP control component, click ***result*** button on the specific task in task table, or navigate to CP Results using the Header, then choose desired camera in the ***Display Results For*** list.

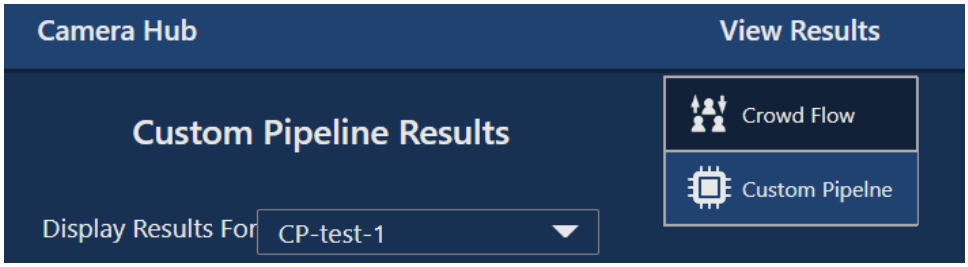

Figure 2.31. Aupera web application page – custom pipeline task result

NOTE: After the Custom Pipeline Task started, it may take up to a few seconds before an output video stream starts. “Video failed to load” error may occur when checking custom pipeline result right after the task is started. Wait a few seconds and click the refresh button to try displaying the output video again.

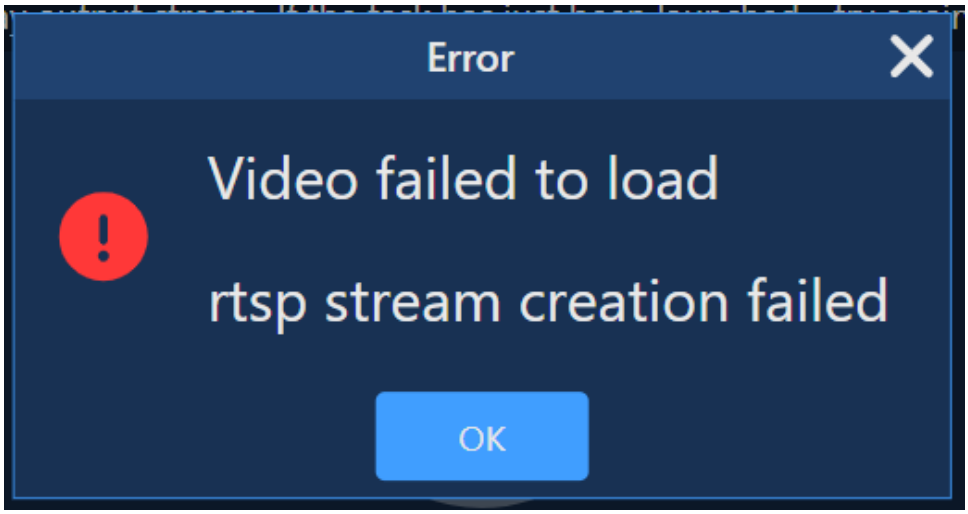

Figure 2.32. Aupera web application page – custom pipeline task result, “Video failed to load” error

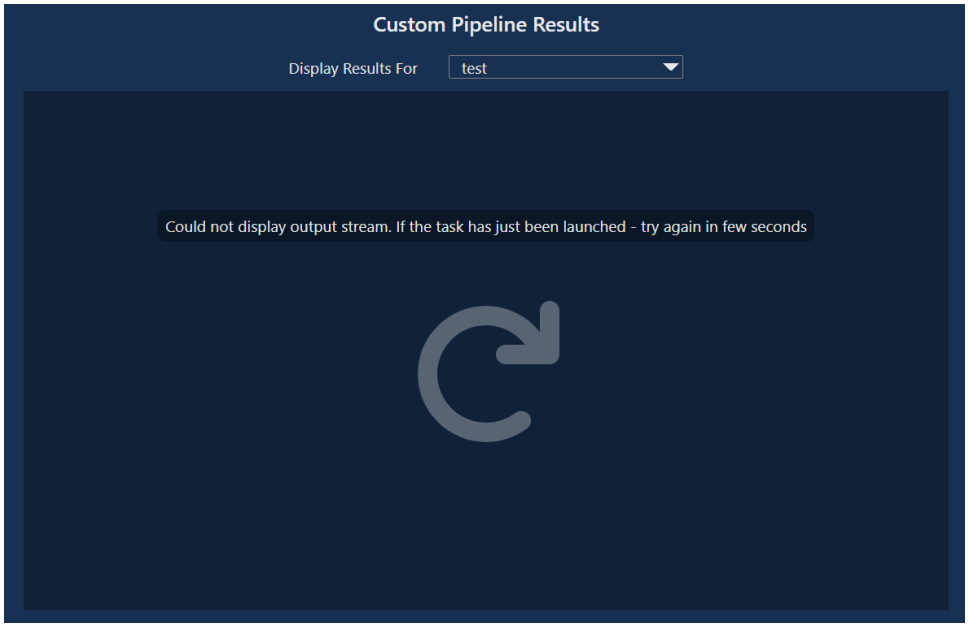

Figure 2.33. Aupera web application page – custom pipeline task result page (output video not yet available)

J. Current state of the task can be checked with the Check Status button, the status will also be shown on the table in the status column.

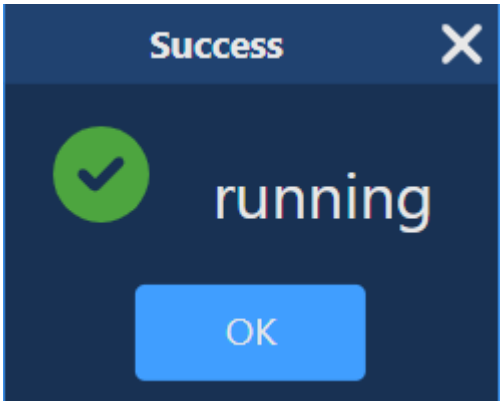

Figure 2.34. Aupera web application page – task success running notification

K. To stop the task, click Stop Task in the CP control component.

2.5 Using Camera Hub to Add New Camera, play videos, or change snapshots

A. Click ***Camera Hub***, then click ***Add New Camera*** button, then popup will appear shown in Figure 2.4.

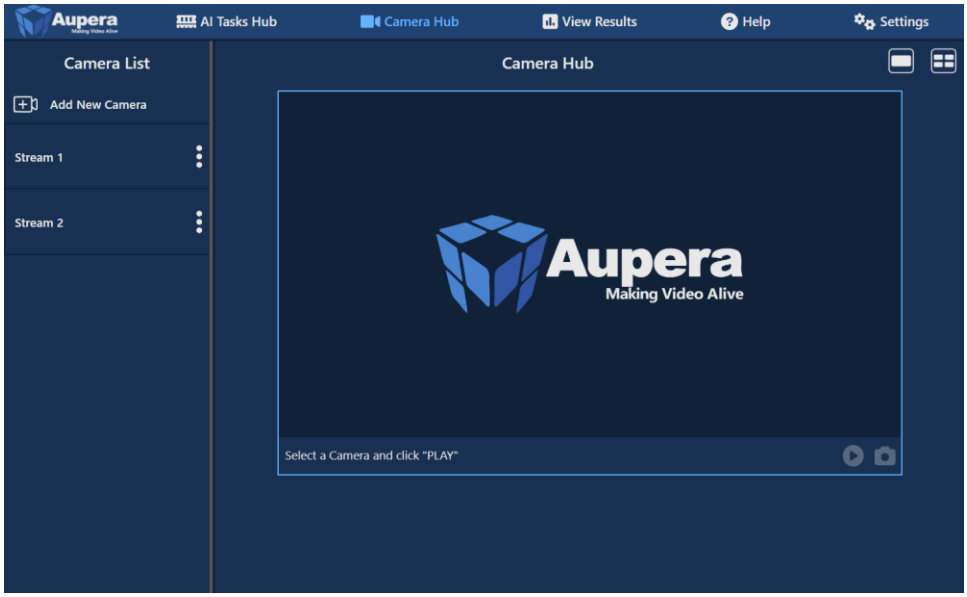

Figure 2.35 Aupera web application page – Camera Hub Page

B. To play the video, select a Camera from Camera List. Video playback will start immediately. If another camera is clicked, video playback will switch to this camera stream.

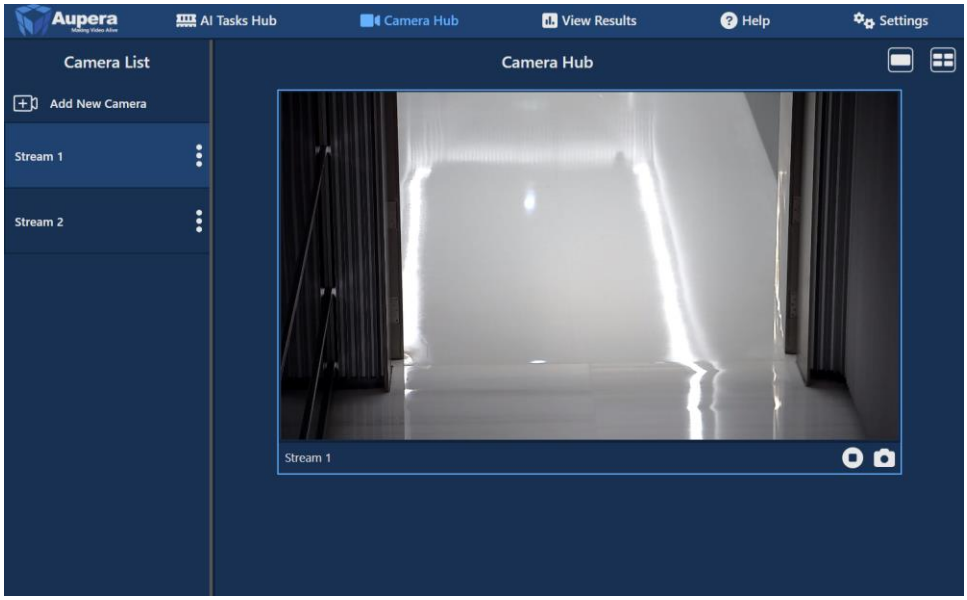

Figure 2.36 Aupera web application page – Camera Hub Page, camera selected

C. The video can also be played and stopped by clicking the stop and play button located in the bottom right corner of the video play.

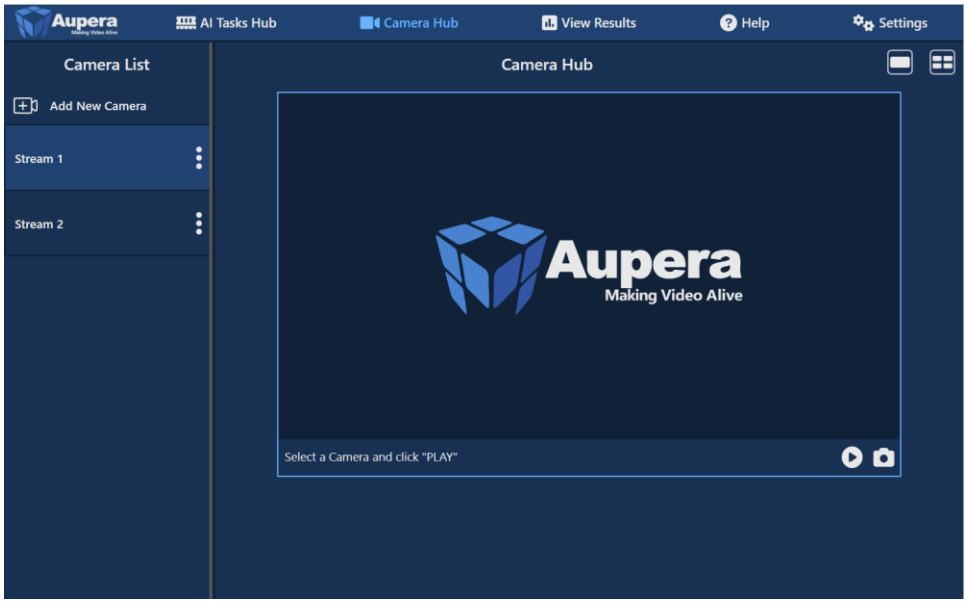

Figure 2.37 Aupera web application page – Camera Hub Page, stopped button clicked

NOTE: Video player should start playing shortly after the play button is clicked. If the video player does not start playing after clicking the play button for 10 – 15 seconds, click the stop button and play button again to restart the video play.

D. To change the snapshot, click the Camera Button beside the play button while the video is playing

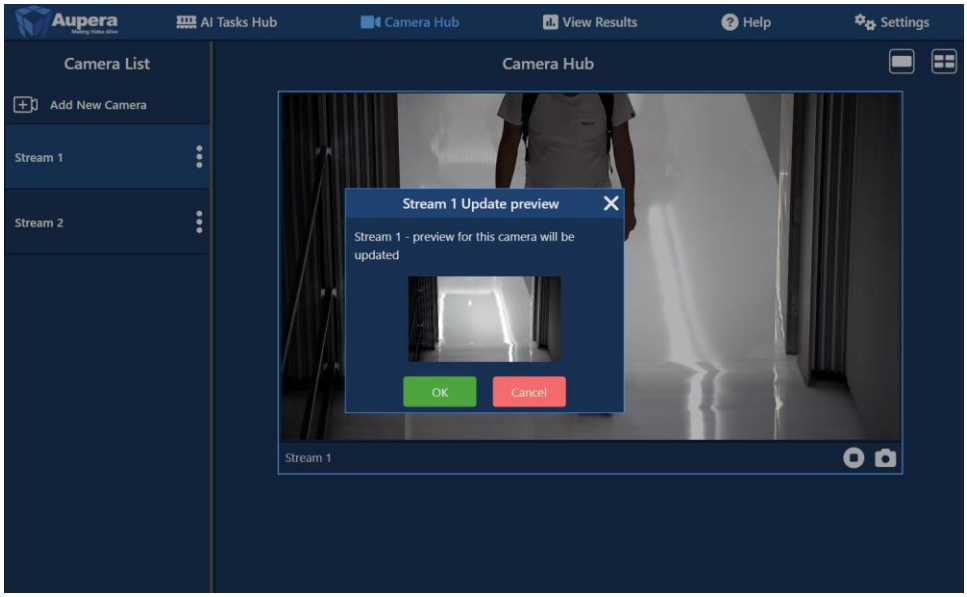

Figure 2.38 Aupera web application page – Camera Hub Page, snapshot taken

Click ***OK*** button to update the snapshot or ***Cancel*** button to cancel updating the snapshot.

E. To view four cameras in the same screen, click the four rectangles icon located on the top right corner of the screen. To play video on a specific Video Player, choose a camera frame by clicking directly on the camera frame and select a Camera to play from Camera List, or use the Play button at the bottom right corner of each camera frame to play or stop video play.

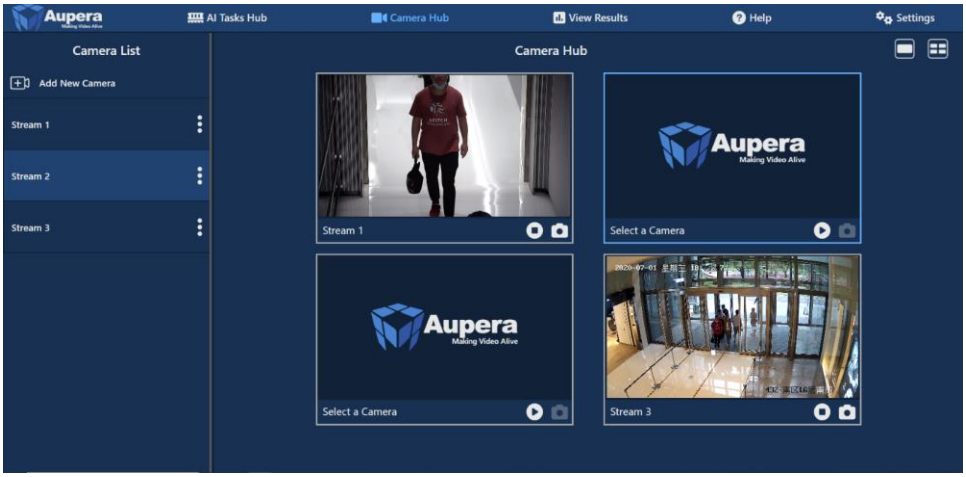

Figure 2.39 Aupera web application page – Camera Hub Page, 4 camera view

2.6 Useful tools on Web App

A. ***Filter*** located at the top of the task list can filter items in task list. To use ***Filter***, simply type in the input field or select an item by clicking the input field or the arrow button to filter the task list item by Camera, AI Application, or Task Status.

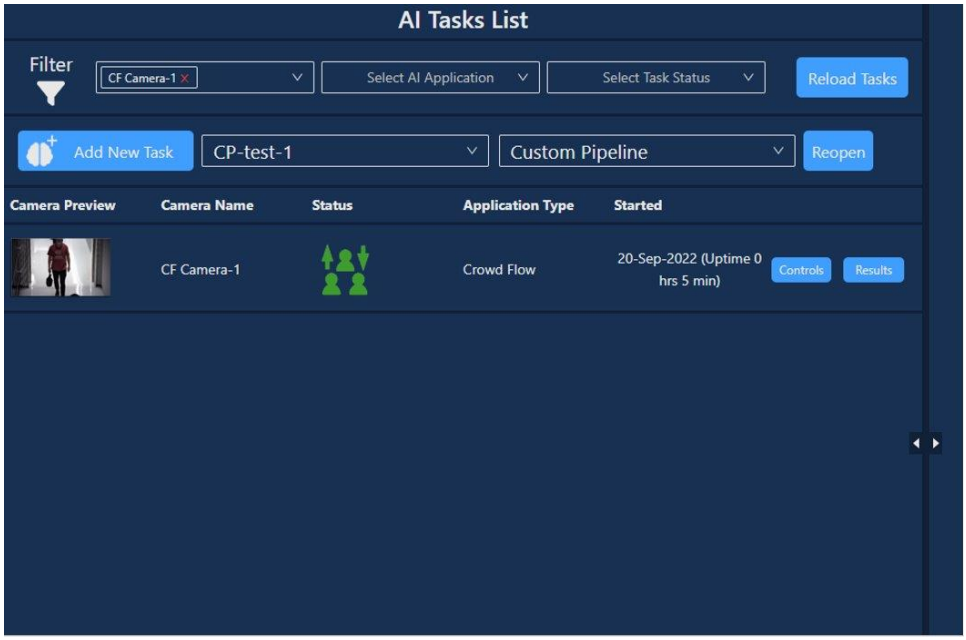

Figure 2.40 Aupera web application page – AI Tasks Hub, filter by Camera

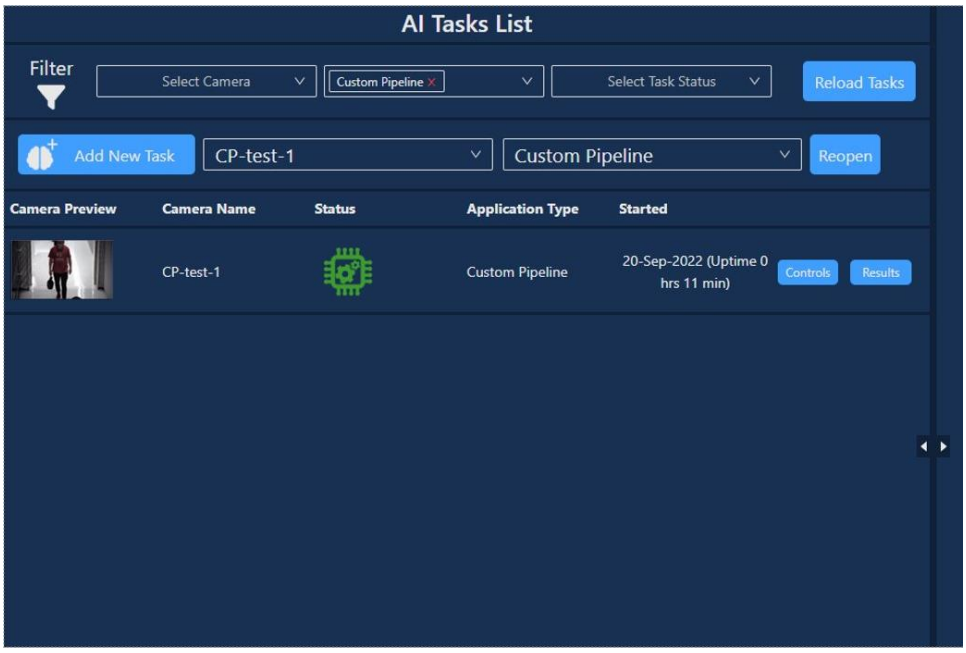

Figure 2.41 Aupera web application page – AI Tasks Hub, filter by AI Application

B. Arrow buttons in the middle of the AI Tasks Hub page help adjust the ratio of the screen for a more convenient and customized view. Right arrow can be clicked to extend the table view. Left arrow can be clicked to extend what exists

on the right side of the screen. When one view is extended, a bar will appear to help collapse the screen.

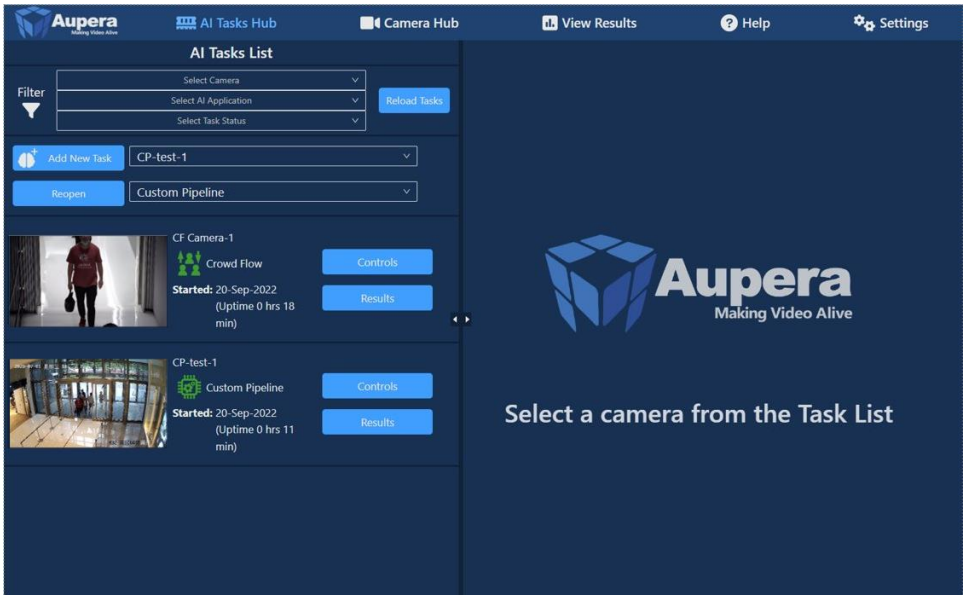

Figure 2.42 Aupera web application page – AI Tasks Hub view on smaller screen

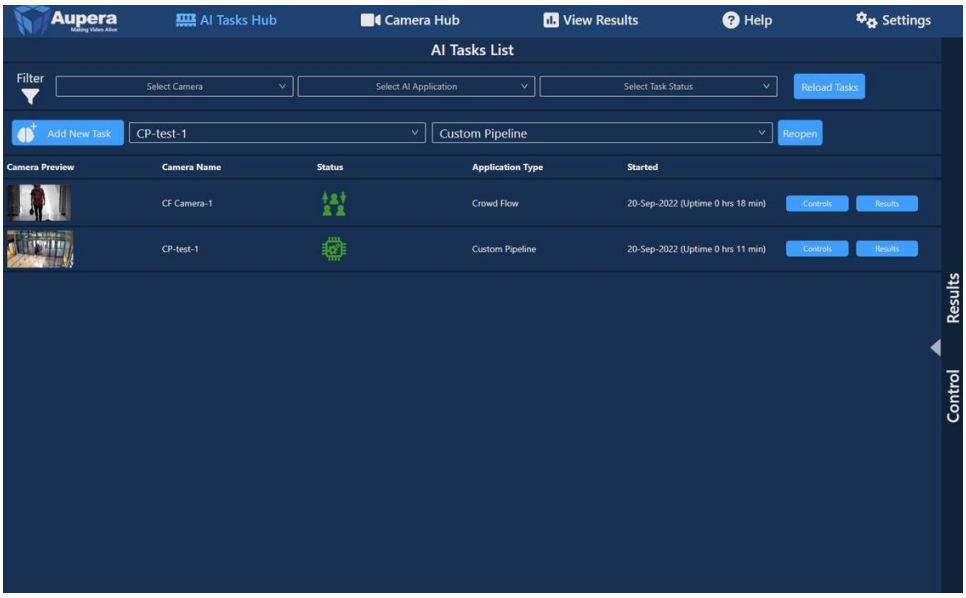

Figure 2.43 Aupera web application page – AI Tasks Hub, table(left) view extended

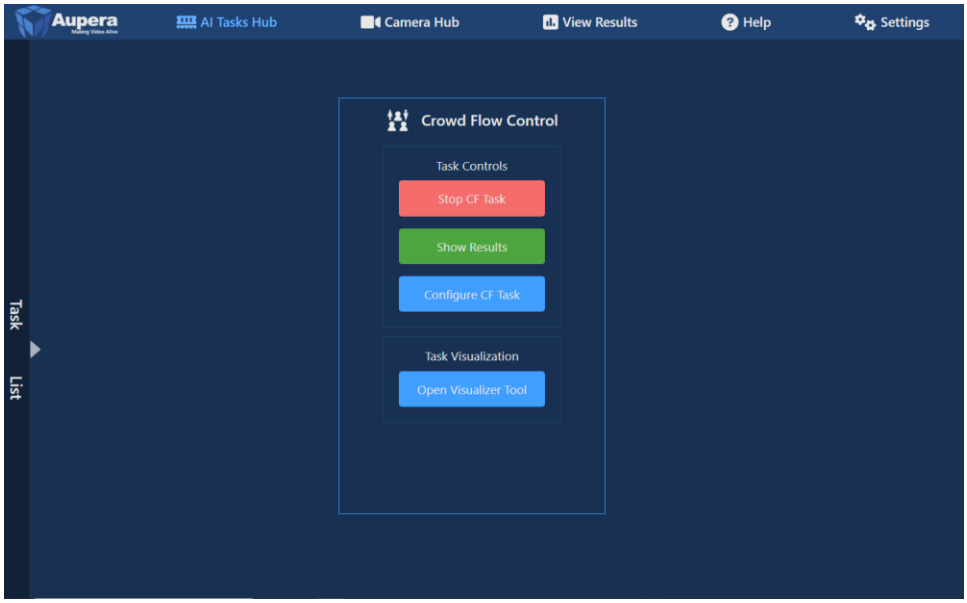

Figure 2.44 Aupera web application page – AI Tasks Hub, right view extended

2.7 Launching Your Own RTSP Streams

To test any pipeline, you need either an RTSP stream, your test video, or an RTSP address of your IP camera. If you are using the RTSP streams provided (in NOTE 3 in Section 1.1), you can skip this section and move on to the next section.

This section focuses on helping the user to broadcast their test videos via a RTSP server. Before continuing, please make sure FFMPEG is downloaded on your local machine and accessible via command line. Visit FFMPEG download page for further instructions.

For your video file residing on your local machine to be pushed into a RTSP server, you can follow the instructions below:

A. You can publish a stream using

```
ffmpeg -re -stream_loop -1 -i file.ts -c copy -f rtsp
rtsp://server_ip:8554/mystream
```

Where *file.ts* is your video file residing on your local machine and *server_ip* is the IP address of your VMAccel instance.

NOTE: The key *mystream (*in *rtsp://server_ip:8554/mystream* above) should be a unique string for each stream.

B. You can then watch the stream using VLC (or any other video player) through **media/open network** stream option, by hitting *ctrl+n* or by using

```
vlc rtsp://server_ip:8554/mystream
```

You can download VLC from here.

3 RUNNING AUPERA VMSS2.0 PIPELINES ON SERVER (VIA COMMAND LINE)

In this section, we will describe how to run the Aupera VMSS2.0 pipelines through the VMSS2.0 Server via the command line. To use AVAS, the VMSS2.0 Server, users need to launch a VMAccel terminal and go into the docker for VMSS2.0 Server. Once there, users can run any pipelines directly from the command line. In what follows, we will introduce the procedures to access AVAS docker, run VMSS2.0 pipelines, and offer some pipeline examples in detail.

3.1 Accessing VMSS2.0 Server Docker

A. In the left-hand sidebar select **"*Instances*".** Then, click on the name of your instance.

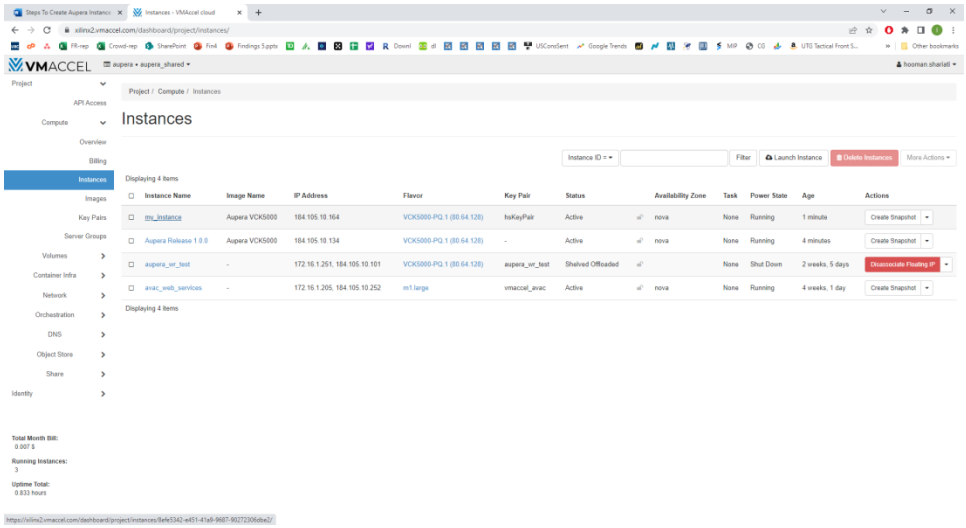

Figure 3.1. VMAccel instances page – instance selection

B. Select the ***Console*** tab as Figure 3.2 shows.

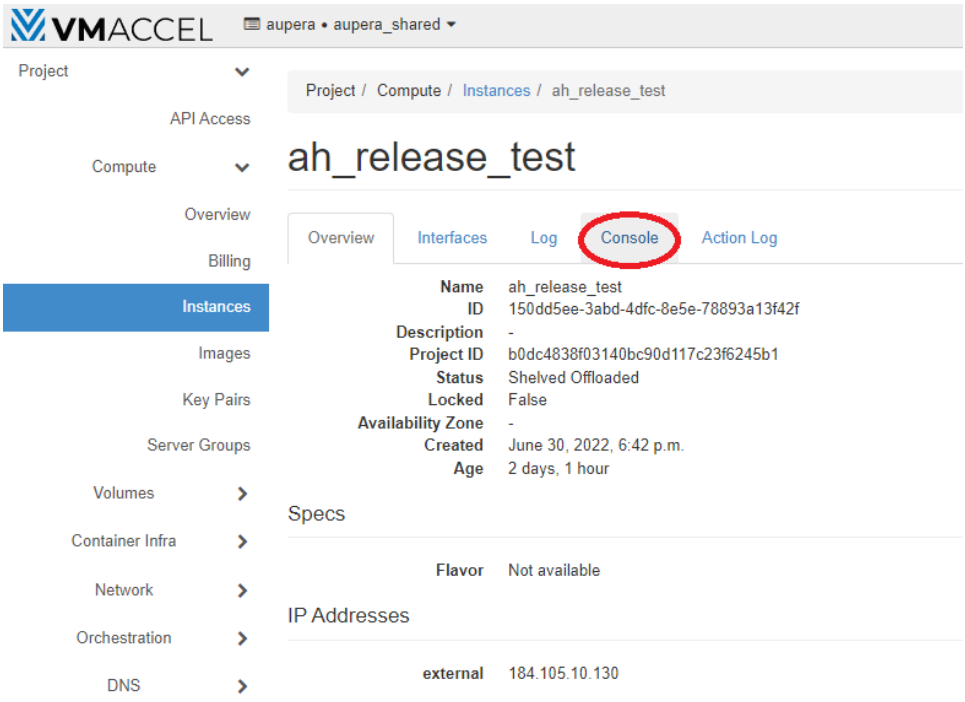

Figure 3.2. VMAccel instances page – instance console

C. Once the VNC window opens click on ***Connect*** as Figure 3.3 shows.

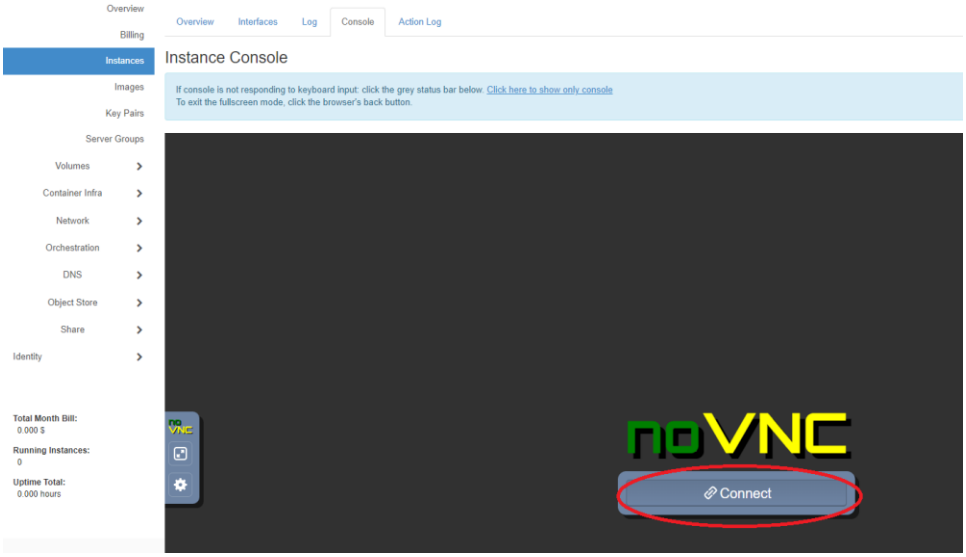

Figure 3.3. VMAccel console page – instance console connection

D. Inside the VMAccel VNC window, click on the Terminal icon to open a command line shell terminal.

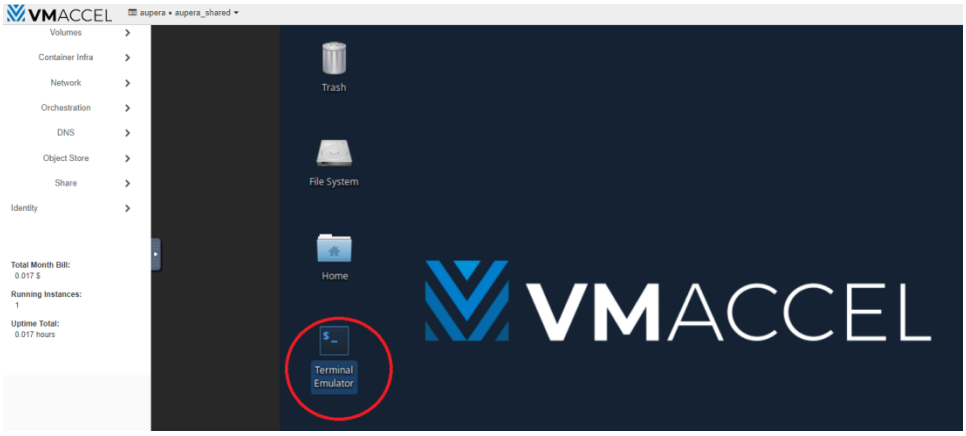

Figure 3.4. VMAccel console page – terminal emulator highlighted

E. From the terminal, you can enter the VMSS2.0 server's docker by running the command:

```
docker container exec -it aupera_server bash
```

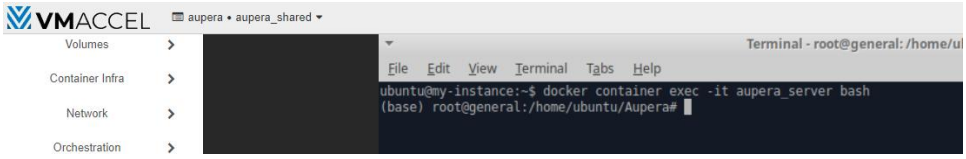

Figure 3.5. VMAccel console page – accessing VMSS2.0 server's docker

F. Once inside the docker, you will need to setup the environment (xbutil and vitis) by running the command below from any directory.

```
source set_env.sh
```

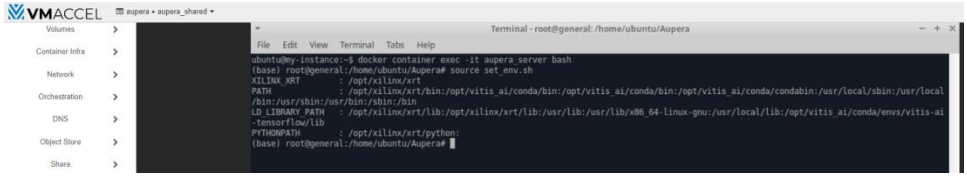

Figure 3.6. VMAccel console page – setting up software environment

At this point, the docker is ready to be used and you can proceed to the next section.

3.2 Running VMSS2.0 Pipelines

Generally, to run a VMSS2.0 pipeline, you can run the commands below from any directory inside of the VMSS 2.0 server docker. There are 3 pbtxt files that are required to pass to *avaser*:

1. **Input:** comes after *-i* parameter and contains the same number of RTSP streams as the input_streams contained in your pipeline.pbtxt.

2. **Output:** comes after *-o* parameters and contains the same number of rtsp streams (or file passes) as the output_streams contained in your pipeline.pbtxt.
3. **Config:** comes after *-c* parameter and contains your pipeline definition (the list of nodes and connections).

Below as an example of what the command should look like:

```
avaser -i input.pbtxt -o output.pbtxt -c pipeline.pbtxt
```

Below is an example of the content of an input.pbtxt file:

```
input_urls: "rtsp://10.10.190.114:554/key1"
input_urls: "rtsp://10.10.190.114:554/key2"
input_urls: "rtsp://10.10.190.114:554/key3"
```

Below is an example of the content of an output.pbtxt file:

```
output_urls: "rtsp://10.10.190.114:554/key4"
output_urls: "rtsp://10.10.190.114:554/key5"
output_urls: "/tmp/output_video_file.mp4"
```

NOTE 1: The output.pbtxt file could be empty if there are no output streams.

NOTE 2: The output.pbtxt file could contain file paths instead, in which case, the encoded video will be saved to disk instead of being sent to the RTSP streaming server.

3.3 Pipeline Examples

We have provided several examples of full pipelines here. These are also included in the aupera_server docker in the **/opt/aupera/avas/examples** folder. You can navigate to this location using the following command:

```
cd /opt/aupera/avas/examples
```

In most of the example folders there are two sets of pipelines pbtxt files: one called **using_rtsp_...pbtxt** and another called **using_video.pbtxt**.

If you'd like to try the example pipelines on the sample videos, then all you need to do is to go inside the sub-folder of a specific example (box_detector, box_detector_classifier_cascade, or apl_crowd_flow) and run the following command:

```
 avaser -i input.pbtxt -o output.pbtxt -c using_video.pbtxt
```

If you'd like to try the example pipelines on the RTSP streams that are automatically started by your VMAccel instance, then all you need to do is go inside the sub-folder of a specific example and run:

```
avaser -i input.pbtxt -o output.pbtxt -c using_rtsp.pbtxt
```

The results of our examples will be broad case in the IP address specified in the output.pbtxt file. In most cases, this is set to

```
output_urls: "rtsp://localhost:8554/out1"
```

which means that you can see the results by typing above RTSP URL into VLC; replacing “localhost” with the IP address of your VMAccel instance.

NOTE 1: If you’d like to run the pipelines on videos other that what we have provided, you will need to modify the “path” parameter in the video_stream node. As shown in Figure 3.7 below:

```
node {
  name: "video_stream_node"
  calculator: "video_stream"
  input_stream: "inStream1"
  output_stream: "imgStream1080p"
  side_node_name: "crowd_flow"
  node_options: {
    [type.googleapis.com/gvis.VideoStreamOptions]: {
      path: "/opt/aupera/avas/examples/videos/C235-2021-04-21-13-13-04_first90S.mp4"
    }
  }
}
```

Figure 3.7. Aupera video stream output path in pbtxt file

NOTE 2: If you’d like to try our example pipelines on RTSP streams other than the ones lunched by your VMAccel instance, then you will need to edit the input.pbtxt files to set the input_rtsp parameter to the URL of your RTSP streams.

For example, if the input.pbtxt of the example you are using contains:

```
input_urls: "rtsp://10.10.100.100:8554/stream1"
```

And the IP address of your VMAccel instance is 99.99.999.999:554/mystream, then you should edit your input.pbtxt to contain the following:

```
input_urls: "rtsp://99.99.999.999:554/mystream"
```

NOTE 3: To see the results of example pipeline on RTSP streams in other locations you will need to edit the output.pbtxt file in each folder to point either to the IP address of your RTSP sever; or to a valid file path. For example, if you’re inside the box_detector example folder, and the IP address of the machine running your RTSP server is 10.10.100.100, then you will need to adjust the output.pbtxt to contain:

```
output_urls: "rtsp://10.10.100.100:8554/someKey"
```

In above case, you can watch the pipeline results at the RTSP stream provided above. Alternatively, your output.pbtxt could include line similar to the one as follows:

```
output_urls: "/tmp/video_output.mp4"
```

In above case, the pipeline's results will be saved to disk in a file accessible via the path specified above.

NOTE 4: Usually, you can kill the VMSS2.0 tasks (launched via the command line) by pressing CTL + C on your keyboard. For the throughput measurements tasks, you might need to press CTL + C a few times and / or use CTL + | combination to kill all the threads.

The pipeline examples that are included with the correct release are as follows:

A. box_detector/using_rtsp_0output.pbtxt

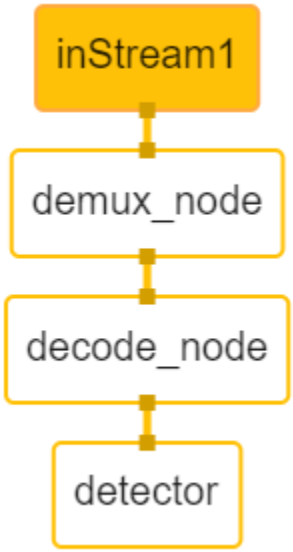

Figure 3.8. Aupera pipeline example with demux, decode, and detector nodes

The pipeline in Figure 3.8 takes one input stream, runs a box_detector network on the decoded frames, visualizes the detections on the frames, and saves the frames to disk (there is no output video).

B. box_detector/using_rtsp_1output.pbtxt

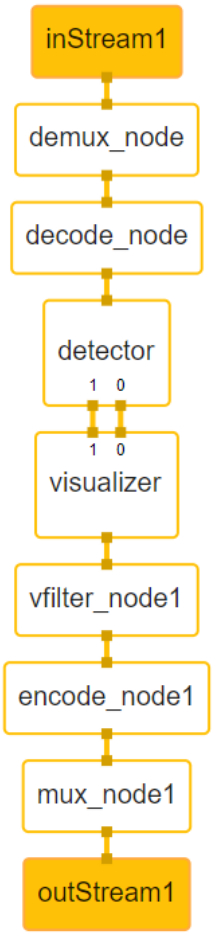

Figure 3.9. Aupera pipeline example with multiple nodes 1

The pipeline in Figure 3.9 takes one input stream and one output stream. It runs a box_detector network on the decoded frames and sends the detected bounding boxes and the frames to the box_visualizer node. The box_visualizer node, will visualize the detected bounding boxes on the frames and send them to video filter, video encoder, and mux nodes. The results are returned in an output rtsp stream or video file.

C. apl_crowd_flow/using_rtsp.pbtxt :

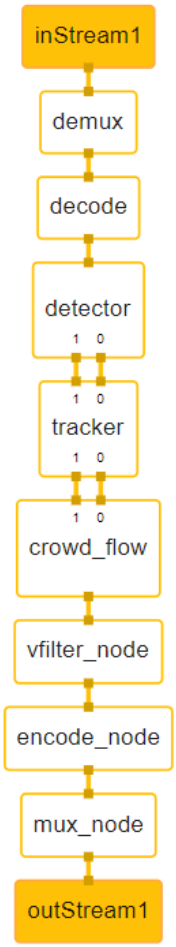

Figure 3.10. Aupera pipeline example with multiple nodes 2

The pipeline in Figure 3.10 takes one input stream and one output stream. It runs a box_detector (at some interval), which passes the detected bounding boxes and the frames to a box_Tracker. The box_tracker tracks the objects (even on frames where the detector has not been run) and sends the bounding boxes and the frames to our crowd_flow application node. This node applies the crowd_flow logic; visualizes the results; and passes the frames to the video filter, encode, and mux nodes. The results can be seen in the output rtsp stream or a video file.

D. box_detector_classifier_cascade/using_rtsp.pbtxt

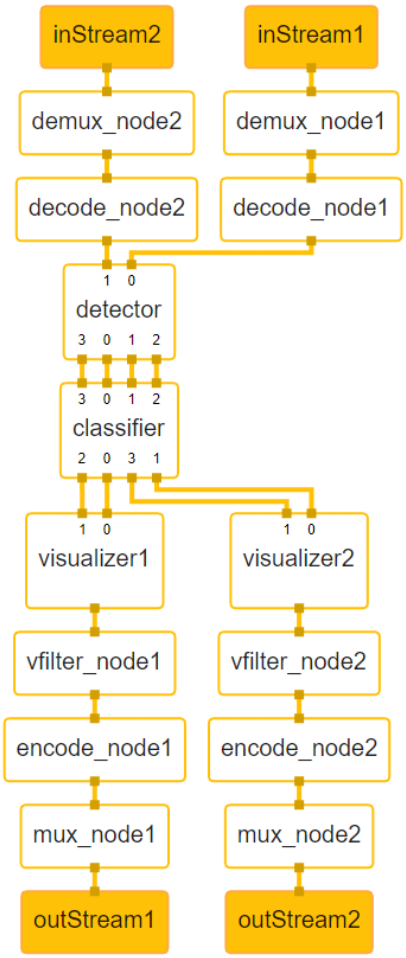

Figure 3.11. Aupera pipeline example with two input and output streams

The pipeline in Figure 3.11 takes two synchronized input streams and produces two output streams. It runs a box detector (at some interval). Then it passes the frames in-tandem with the detections to the classifier node. The classifier node then classifies the objects detected by the detector node. It passes the classifications, which sends the results of each stream to its corresponding box_visualizer node. The box_visualizer will overlay the detections and classifications on the frames and send the results to video filter, stream encode, and stream mux nodes to be displayed over RTSP stream.

E. box_detector/using_rtsp.pbtxt

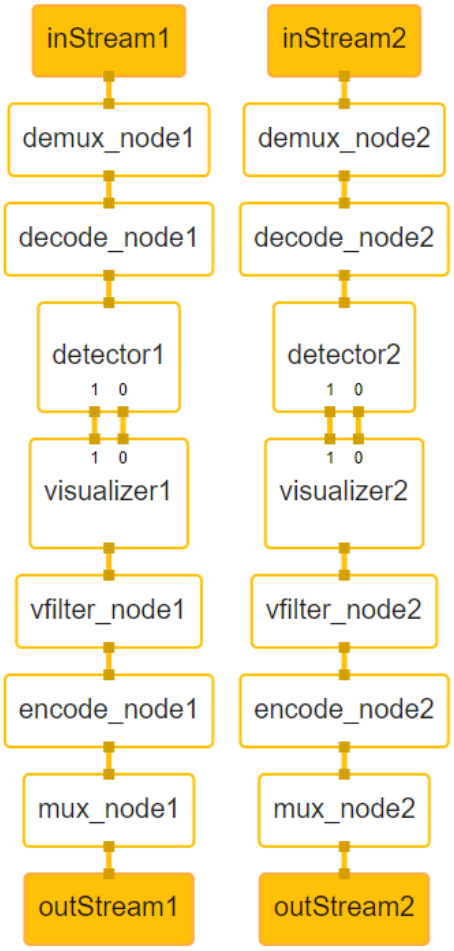

Figure 3.12. Aupera pipeline example with two input and output streams

The pipeline in Figure 3.12 is very similar to example B (figure 3.9); except it runs two detection tasks in parallel. This pipeline takes two input streams and produces two output streams. On stream 1, it runs a box detector with crowd models (head detection); while, on stream 2, it runs a box detector with retail models. The detection results are then passed to corresponding box_visualizer nodes. The box_visualizer nodes will overlay the detections on the frames and send the results to video filter, stream encode, and stream mux nodes to be displayed over RTSP stream.

F. Throughput measurement using retail application

Similar to VMSS1.0, we use the retail application as an example pipeline for throughput measurement. This application consists of running a TinyYoloV3 object detector along with 3 resnet50 classification networks. All the networks are trained on the objects in the retail scenario. We also use the same retail.mp4 video as before (provided in the example_videos folder). We maintain (and slightly exceed) the performance of

VMSS1.0 by supporting 37 streams (with I-frame-extraction) without any trackers and with 56 streams when using a tracker. We have, however, dramatically improved the accuracy of the pipeline when a tracker is used compared to VMSS1.0.

If you look inside the **example_pipelines/throughput_benchmarking/37streams**, you will find 3 configurations:

- **config_noTracker_noVideoOut.pbtxt** is the official test configuration. To run this test, you must ensure that the output.pbtxt file is empty.

- **config_withTracker_noVideoOut.pbtxt** performs the same test except the tracker is used. Here, we use a cluster size of 5 (i.e. classify each track 5 times) to achieve higher accuracy.

- **config_withTracker_withVideoOut.pbtxt** can be used to watch the visualized results on the output stream. To run this config, you will need to use the provided output.pbtxt with the correct output stream paths. Please note that since video visualization is a computationally expensive operation, it is only allowed while running the tracker. Also please note that when I-frame-extraction is true, the output video stream can only be saved at 10fps (since the video has an I-frame every 3rd frame). You can set I-frame_extraction to false, in order to run and save the video at 30fps but that would require reducing the total number of streams (since we are making the test 3 times harder by passing 3 times the number of frames to the pipeline).

Finally, inside **example_pipelines/throughput_benchmarking/56streams** we have provided a single config (**config_withTracker_noVideoOut.pbtxt**), which can be used to confirm that the framework can run the retail pipeline with 56 streams without any frame drops. You can increase the **classifications_cluster_size** parameter to achieve higher classification accuracy if needed. Although, in this example, even with a value of 1, the classification accuracy is not far lower than running without a tracker. Essentially, this parameter specifies how many times we run the classification on each track. For example, when a value of 5 is specified, for each track, we run the classifier 5 times, and use the most frequent (i.e., the mode) classification as the final result.