Gaze - Intelligent Video Cloud Service

Contents

[Gaze - Intelligent Video Cloud Service 1](#_Toc519524717)

[Overview 1](#_Toc519524718)

[Design 1](#_Toc519524719)

[Architecture 2](#_Toc519524720)

[Developer role 2](#_Toc519524721)

[Computational Graph 4](#_Toc519524722)

[Type of Nodes 4](#_Toc519524723)

[Execution engine 5](#_Toc519524724)

[Built-in component 5](#_Toc519524725)

[Video sources 5](#_Toc519524726)

[Intermediate nodes 7](#_Toc519524727)

[Sinks 9](#_Toc519524728)

# Overview

The Gaze Live Video Analytics Service Platform integrates the computing power of private clouds with data from IoT devices such as cameras, providing scalable video analytics capacity on the Edge.

# Design

Cameras are ubiquitous in today's society, collecting large amounts of data all the time. However, these data are like "dark matter", and the knowledge contained is unknown. The huge value is not fully explored.

With this in mind, we propose Gaze to integrate the computing power of private clouds with data from IoT devices such as cameras. Provides scalable, scalable video analysis capabilities at the Edge.

## Architecture



Distributed execution engine:

1) Organize cloud computing and IoT device resources based on Kubernetes;

2) Processing video streams and images based on open source libraries such as GStreamer and OpenCV;

3) Provide calculation graph execution services for the upper layer.

## Developer role

Developers who do video analysis based on the Gaze platform can be divided into:

* Component Developers (dev 1): They have a computer vision background and are responsible for developing video cognition and analysis components. Possible components include: vehicle count, pedestrian count, license plate recognition, wanted identification, specific events (such as fighting) recognition, and more.
* Application developers (dev 2): They are close to the application scenario, understand the requirements, are responsible for developing the calculation diagram, and assembling the above components into a complete application to solve specific problems.



As shown in the system data flow diagram above, the basic flow of Gaze's work is:

1. The video cognition and analysis component created by the component developer is entered and stored through Gaze's web interface;
2. The application developer creates a calculation graph, enters it through Gaze's web interface, and stores it;
3. The execution engine obtains the calculation graph, generates a container image, creates a Pod, and schedules Kubernetes for scheduling through Helm charts;
4. The execution engine obtains the IP address/port of the allocated container;
5. The execution engine schedules the video source to send a video stream to the specified IP address/port;
6. (public/private) containers in the cloud do real-time processing of video;
7. Once processing is complete, the container sends the video stream to the aggregation point. According to the component definition, the aggregation point can be a monitor screen, a big data store, an Event Hub, a Kafka queue, and the like.

## Computational Graph

The application computational graph, also called the data flow graph, is a directed acyclic graph. Each step in the video analysis is to compute a node in the graph, and the edges between the nodes describe the dependencies between the computations.

Here is the code example and visualization of the simplest calculation graph:

|  |
| --- |
| x = VideoTestSource()  x = EdgeDetection()(x)  x = AutoVideoSink()(x) |

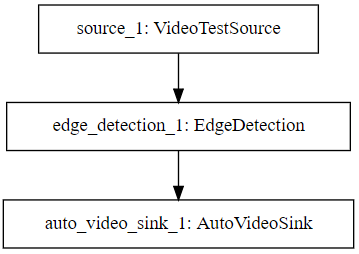


Figure 1. The visualization of the simplest calculation graph

This code generates a video stream based on the built-in VideoTestSource class object, performs basic processing with the EdgeDetection class object, and finally sends the AutoVideoSink class object to the window.

### Type of Nodes

Computational graph nodes can be divided into three categories:

1. Video sources: This node has no input, only output. Such as "test video source", "camera video source", "file video source", "network stream video source", etc.;
2. Analysis nodes: both input and output. The input is one or more video streams, and the output is a video stream or an event stream.

Such as "vehicle counting", "pedestrian counting ", "license plate recognition", "wanted identification ", "specific event (such as fighting) identification";

1. Sinks: only input, no output. Such as "Hadoop Storage Sink", "Event Hub Event Sink", "Kafka Event Sink", "Window Display Sink ", "Network Stream Sink ", etc.

## Execution engine

The execution engine generates a Dockfile based on the calculation graph, compiles it into a container image, and deploys it to the cloud through Kubernetes.

## Built-in component

### Video sources

#### Test video source

This component is used to generate test video data in a variety of formats. By default, this component will generate data indefinitely. The following image is a static image taken from the "Test Video Source":

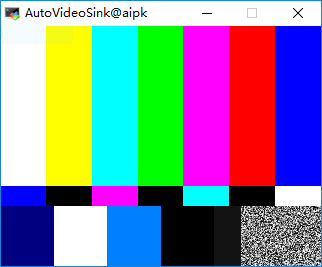


Figure 3. Test video source

#### Camera video source

This component is used to capture video from a webcam device.

#### File video source

Read data from a video file in the local file system.



Figure 4 File video source

#### Network streaming video source

The network source that reads the RTP stream UDP packets from the network. The following image is a static image taken from the "network streaming video source":

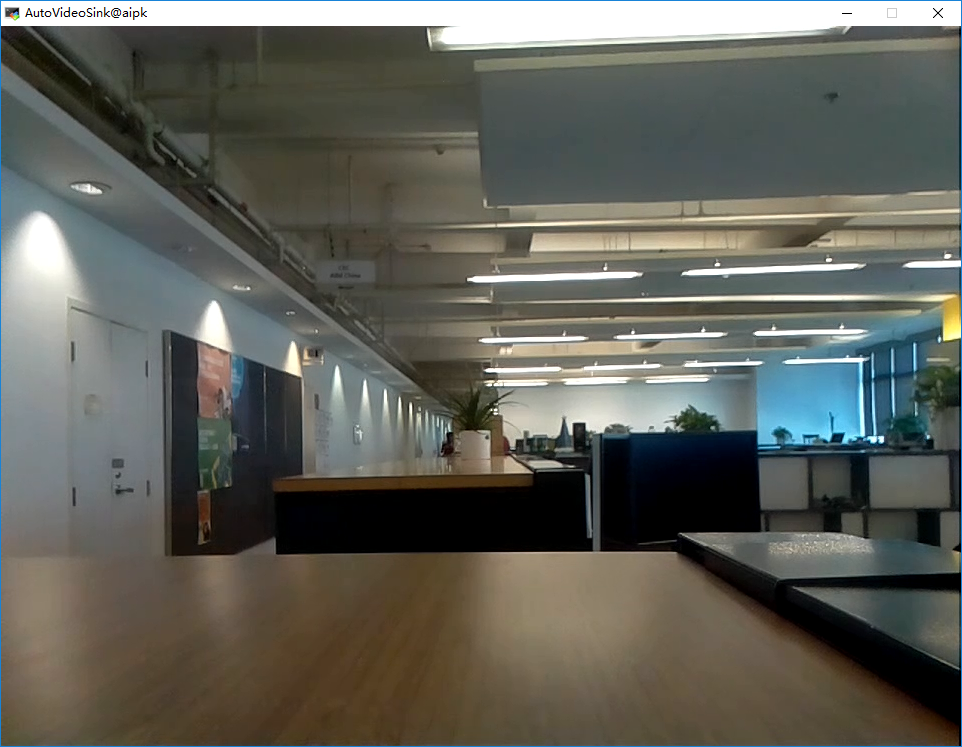


Figure 4 Network streaming video source

### Intermediate nodes

The intermediate node is responsible for the processing of the video. Here are some examples of intermediate nodes:

#### Edge detection

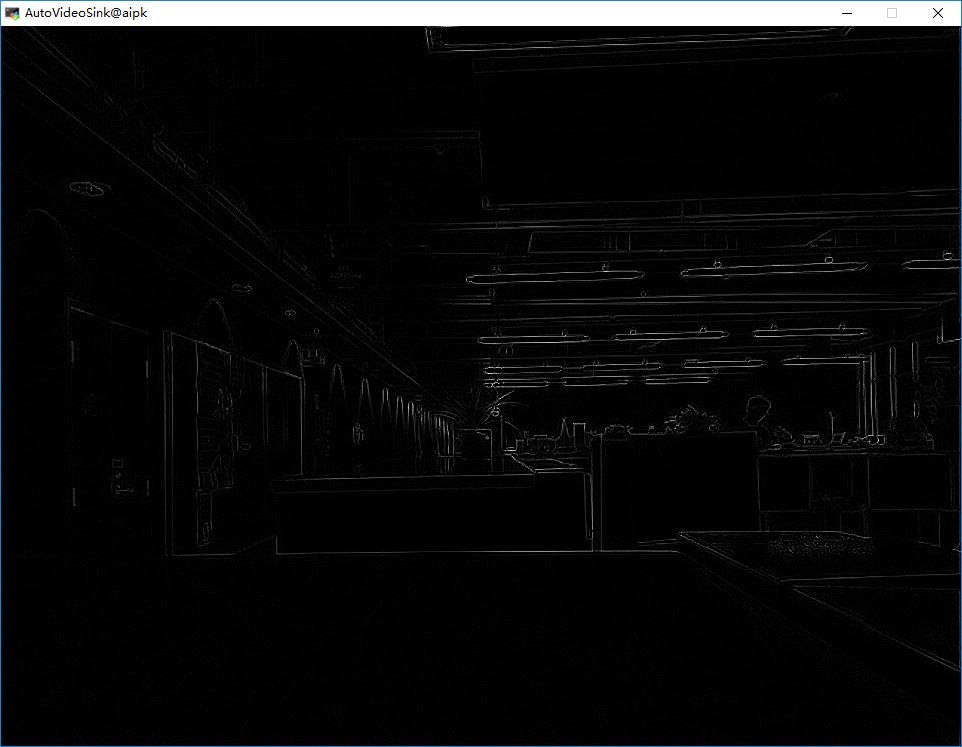


Figure 5 Edge detection based on webcam

#### Pedestrian counting

Count the number of people appearing in the video stream over a certain period of time. The core count is the target detection and target tracking technique.

#### Specific event recognition

Refers to the technique of identifying pedestrian behavior by analyzing video. It is divided into individual identification and group identification. It is mainly based on the image sequence of human motion, combined with high-precision skeleton estimation algorithm to extract the human skeleton motion sequence, thus achieving behavior recognition.

### Sinks

#### Hadoop Storage Sink

Write incoming data to files in the distributed file system HDFS.

#### Auto video sink

The incoming data is displayed in the window as a live video.

#### Web interface

The administrator can browse how many applications are executing in the web interface.

#### Component in-take

Component developers use the component entry module to upload components.

#### Application graph in-take

The application developer uploads the application calculation graph using the calculation graph entry module.