



# Video Recognition Is Not Rocket Science

**Building an Application  
to Analyze Video  
Streams and Detect  
Specific Human Actions**

# Contents

**Automating Camera Surveillance  
of Healthcare Facilities 3**

**Automatically Detecting Human  
Interactions in a Video Stream 4**

**Building a Video Recognition System  
Using Serverless Architecture 5**

**Uploading Data and Integrating  
Machine Learning 7**

**Categorizing and Labeling Events  
Captured on Video 8**

**Reducing Video Monitoring Staff Using  
Cloud-Native Serverless Technology 9**

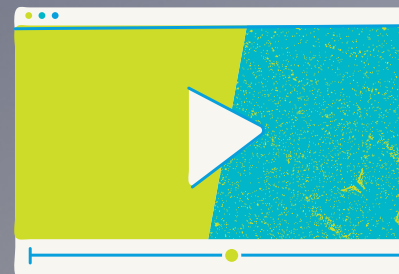
# Automating Camera Surveillance of Healthcare Facilities

---

Facing a shortage of qualified workers, healthcare facilities across the US are looking for work-arounds that will help them function effectively with a smaller staff.

Our prospective customer is no exception. They needed to figure out how they could help their small team to more efficiently monitor medical facility video surveillance recordings. At the time, the facility employed about seven people who constantly watched the recordings to ensure staff were present where needed and to spot anything unusual. These workers struggled to keep up with the recordings, so our customer came to Lineate with the idea to automate this routine.

Two of our solution architects tackled the problem by building an AI proof-of-concept using cloud-native serverless technologies to demonstrate a system the customer could use in the future. We created an AWS serverless implementation that processes surveillance video, detects abnormal events, and sends email notifications to human workers. The system also classifies objects and certain human actions in the video stream. We launched this proof of concept very quickly thanks to native cloud services and serverless development.



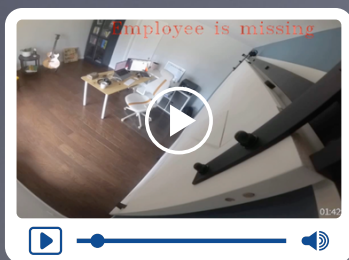
# Automatically Detecting Human Interactions in a Video Stream

The possible applications for video recognition are endless. Hospitals and other healthcare facilities typically use video recognition to detect anything out of the ordinary, such as a patient not getting proper attention or a nurse being away from their station for too long. Using automation to identify problems like these is a huge value for many hospitals. For our very time-bound proof of concept for our customer, we picked a simple but common problem for many healthcare facilities: making sure nurses are at their stations.

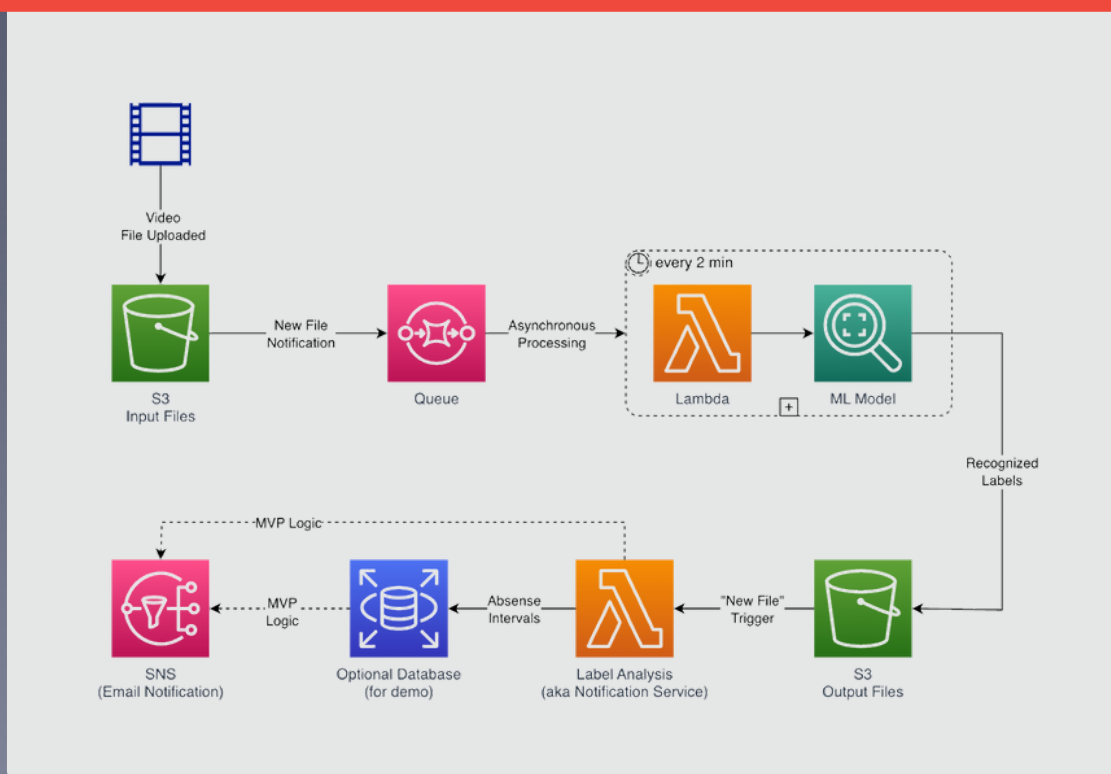
The system created for our proof of concept takes a video stream of a nurse's station and automatically detects humans entering and leaving the video. The video below shows this system in action. In the video, our solution architect Eugene sits down at his desk and then occasionally leaves the room and later comes back.

[Watch Video](#)

To access the proof of concept and see how we implemented the stream recognition using cloud services, please [fill out this form](#) and try it out yourself.



# Building a Video Recognition System Using Serverless Architecture

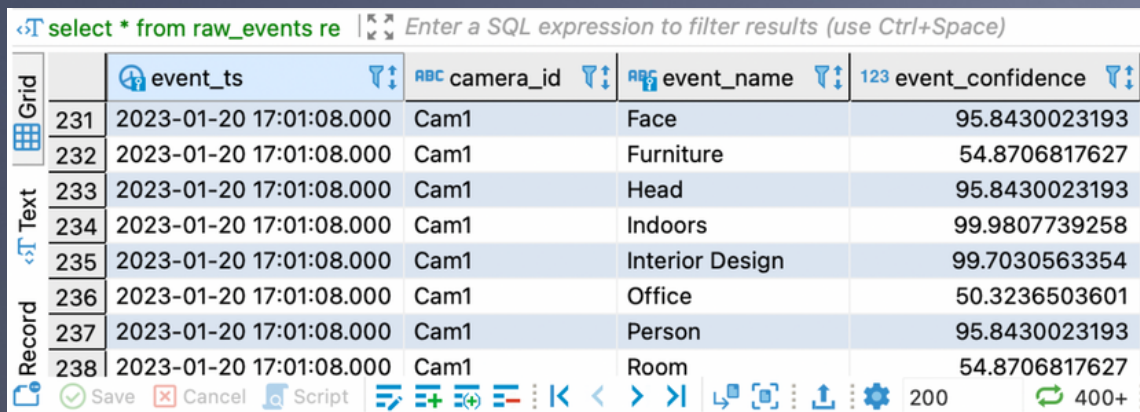


From the beginning we knew that AWS has all the tools required for implementing our use case, including a very powerful video labeling service, the AWS Rekognition API. All we needed to do was pipe our videos into the ML service, then monitor its results and translate the detected labels into the types of events our customer wanted to detect with their video monitors.

To transfer our videos into AWS Rekognition, we created an S3 bucket for video uploads and wrote a lambda function that subscribed to bucket changes and triggered the AWS pre-trained model. This model uploaded the resulting JSON files with recognized labels to another configured S3 bucket.

# REC

The second (non-ML) part of the system is another lambda function that monitors for new label-files, parses them, and uploads them into a PostgreSQL database table formatted as a series of labels and times the events occurred.



The screenshot shows a SQL query interface with the following table data:

Grid	event_ts	camera_id	event_name	event_confidence
231	2023-01-20 17:01:08.000	Cam1	Face	95.8430023193
232	2023-01-20 17:01:08.000	Cam1	Furniture	54.8706817627
233	2023-01-20 17:01:08.000	Cam1	Head	95.8430023193
234	2023-01-20 17:01:08.000	Cam1	Indoors	99.9807739258
235	2023-01-20 17:01:08.000	Cam1	Interior Design	99.7030563354
236	2023-01-20 17:01:08.000	Cam1	Office	50.3236503601
237	2023-01-20 17:01:08.000	Cam1	Person	95.8430023193
238	2023-01-20 17:01:08.000	Cam1	Room	54.8706817627

Finally, we built an algorithm to convert raw events into real-life events. We also created an additional lambda service to pull these events into an SNS email notifications system.



# Uploading Data and Integrating Machine Learning

---

For our video-recognition proof of concept we built a serverless asynchronous pipeline that consists of AWS S3, for receiving data input and saving ML recognition results, and AWS SQS, for asynchronous communication between components. We were able to build the proof of concept in such a short period of time thanks to the AWS Rekognition API, which has many pre-trained ML models for recognizing objects and creating labels in videos and images. To quickly build the infrastructure and connect everything together we used the AWS Serverless Application Model framework.

For simplicity, because we were working with a sample data set we used file-based input. However, when working with real camera streams it is more cost-effective to use AWS Kinesis Video Streams.

The pipeline in our proof of concept system starts when we upload the sample data set to S3. The data then runs through the AWS SQS notification message service to a lambda function, which reads new files and sets the Rekognition API for label detection. The Rekognition API works in asynchronous mode, and when the results are ready it triggers another lambda function through AWS SQS message in order to store recognized labels on S3 and then send the labels in a pre-processed way to a PostgreSQL database like the one illustrated earlier. Finally, the last set of lambda functions in the pipeline works with the recognized labels to analyze events and send email notifications to human workers if needed.

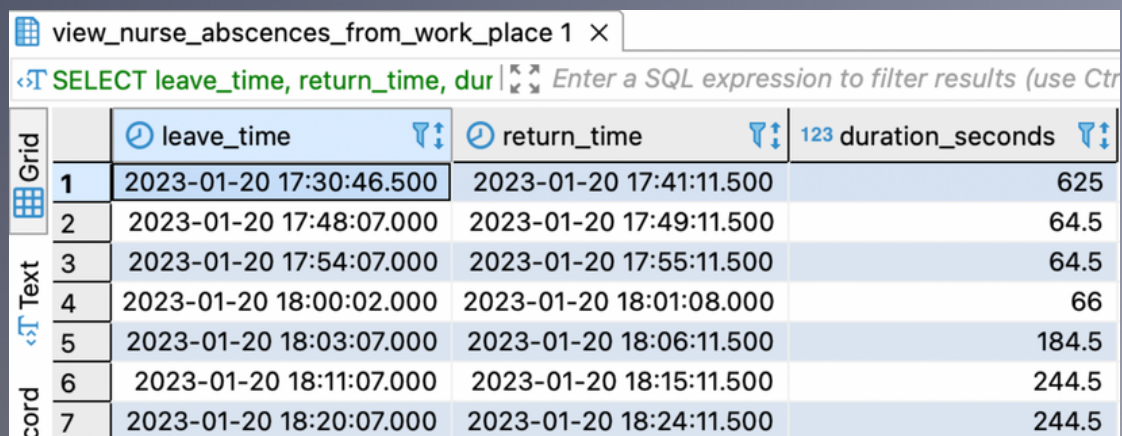
# Categorizing and Labeling Events Captured on Video

To keep the prototype simple we filtered out all events except the ones labeled “Person” and treated them as “A nurse was present at workplace.” To detect a final event in which a nurse is absent for a substantial period of time we needed to identify large time gaps.

Following is the algorithm:

- Generate gap events to cover portions of time when labeled video data is not present
- Combine all “Nurse Present” events
- Smooth out the data by using a window average for the event confidence column
- Group consequent positive events (“Nurse Missing”) and calculate the duration of the absence periods

This is how the final result looked in the database.



The screenshot shows a database query result for a table named 'view\_nurse\_absences\_from\_work\_place 1'. The query is 'SELECT leave\_time, return\_time, dur'. The result is a table with 7 rows and 3 columns: 'leave\_time', 'return\_time', and 'duration\_seconds'. The 'duration\_seconds' column has a value of 123. The rows show the following data:

	leave_time	return_time	duration_seconds
1	2023-01-20 17:30:46.500	2023-01-20 17:41:11.500	625
2	2023-01-20 17:48:07.000	2023-01-20 17:49:11.500	64.5
3	2023-01-20 17:54:07.000	2023-01-20 17:55:11.500	64.5
4	2023-01-20 18:00:02.000	2023-01-20 18:01:08.000	66
5	2023-01-20 18:03:07.000	2023-01-20 18:06:11.500	184.5
6	2023-01-20 18:11:07.000	2023-01-20 18:15:11.500	244.5
7	2023-01-20 18:20:07.000	2023-01-20 18:24:11.500	244.5



# Reducing Video Monitoring Staff Using Cloud-Native Serverless Technology

---

Our prototype proves that the idea to build an application to analyze video streams and detect human action is viable. In fact, the AWS Rekognition API can solve the simple problem of automating video surveillance . It not only has pretrained computer vision models with thousands of labels but also provides tools that allow you to train custom label models on your own data set. These custom models increase accuracy and help build solutions for more complex use cases.

If your use case can be covered by existing labels and features of ML services, you don't need a data science department to design and train ML models from scratch. Existing video recognition services are powerful enough to cover common use cases. And, if you want to automate a video monitoring system for a healthcare facility, like our client did, it's important to know that the majority of AWS services are HIPAA compliant and that Google Cloud can also provide similar solutions.



# Thank you.

Can we help you with your  
ambitious goals?

Talk to us today at  
[lineate.com/contact-us](https://lineate.com/contact-us)

