NAVIGATING IN NUMEROUS VIDEO DATA: 
USER INTERFACE DESIGN FOR AN ON-CAMERA VIDEO ANALYTICS ENGINE

Student Author

Sabriya Maryam Alam is a junior in the College of Engineering and the Purdue Honors College studying electrical and computer engineering. She is interested in the development of impactful application-based machine learning systems and intends to pursue graduate studies in a related field. In addition to her work on video analytics, Alam’s research experience includes developments for artificial intelligence–powered smart city applications and personalized health care. Outside of her research pursuits, she serves as vice president of the Purdue Muslim Student Association and is a cofounder of the Purdue chapter of the national nonprofit Girls Who Code.

Mentors

Felix Xiaozhu Lin is an assistant professor in Purdue University’s School of Electrical and Computer Engineering. He received his CS PhD from Rice University and his BS/MS from Tsinghua University. At Purdue, Lin leads the Xroads Systems Exploration Lab (XSEL) to accelerate and safeguard important computing scenarios. He and his students measure and build systems software with diverse techniques, including novel OS structures, kernel subsystem design, binary translation, and user-level runtimes. See http://felixlin.org for more information.

Mengwei Xu is a senior PhD student at Peking University, China, where he also earned his bachelor’s degree at the university’s School of Electronics Engineering and Computer Science in 2015. Xu was a visiting scholar at Purdue University’s School of Electrical and Computer Engineering from November 2018 to November 2019, working with Professor Felix Xiaozhu Lin. Xu’s research interests mainly center around mobile computing and operating systems. He has published in top-tier computer science conferences including MobiCom and MobiSys.
Abstract

Video analytics powered by artificial intelligence shows high promise in making our society smarter. Harnessing large amounts of video data, however, requires the development of processing systems demonstrating high performance and high efficiency. To this end, this work has contributed to a video analytics system powered by artificial intelligence for object detection and recognition. Rather than streaming all the video frames to the cloud, the system analyzes images on-camera and only returns those of interest to the cloud. This edge analytics research-grade software is available, but it lacks a simple web interface for general use by scientists, engineers, and other experts. To make the system versatile and user-friendly, a web interface was developed for receiving user queries (including parameters such as camera geolocations, video time span, and object class of interest) and presenting the query results. This web interface will make our video analytics system more accessible to domain experts (those in law enforcement, health care, environmental monitoring, etc.).


Keywords
camera, video, query, cloud, artificial intelligence, user interface, web, image-processing system, computer vision

INTRODUCTION

From suggested products on Amazon to movie recommendations on Netflix, machine learning and artificial intelligent systems are becoming increasingly essential components of modern products that are influencing industrial practices as well as everyday life. One particular use of machine learning techniques is computer vision, which encompasses the use of artificial intelligence to process digital videos and images. This is most commonly done in the form of object detection and recognition from sets of image data, including diverse applications such as facial recognition in photo-tagging and navigation by self-driving cars. Access to video and image data is more abundant now than ever before, and the data has great potential for becoming the basis for the development of practical solutions across many fields, including Internet of Things (IoT) applications in smart agriculture and smart cities.

Harnessing such large amounts of visual data (on the order of terabytes) requires the development of processing systems demonstrating time and cost-efficient performance. Existing video processing systems typically stream all the frames captured by a camera to a computer to be processed, regardless of their content. This process is time-consuming and costly in terms of resources and often returns unnecessary data. To this end, a video analytics system has been developed based on minimal streaming and use of machine learning algorithms for object detection and recognition. Rather than streaming all the video frames stored on a camera, the video analytics system developed efficiently analyzes images on-camera and only returns those that match a set of criteria specified by a client.

A potential application of this computer vision technology may be to process surveillance footage from various cameras, with the client being able to access particular frames of interest remotely. For example, in the case of a criminal investigation, law enforcement may be interested in video footage of a white car seen at a particular intersection. Rather than viewing all the footage from the security camera, it would be more convenient and time-efficient to be able to search the footage stored on-camera with keywords and stream only the pertinent video.

To make this possible, a key component of this system is to develop a well-organized web interface that works in tandem with the image-processing algorithm such that the user may file requests and view the relevant retrieved images. This technology will have applications in a variety of contexts, including crime investigations, smart retail, and road traffic analysis.

 PURPOSE STATEMENT

A web-based interface must be developed to make the capabilities of this image-processing system more widely accessible. This interface must be well integrated with the existing query system to process and fulfill the requests of the user.
METHOD OF DEVELOPMENT

Understanding the Video Analytics Engine

The key ideas of this project are based on optimizing the process of cloud-based video and image retrieval from a camera. Xu et al. (2019) outlines the process that this system follows.

The video analytics engine developed is used with low-cost surveillance cameras, which store video streams locally on the camera and only retrieve frames retrospectively based on user query. To do this, the system must process the images on the camera through a machine learning algorithm to determine what objects have been found in the surveillance footage prior to sending this information to the cloud. Rather than iterating over all the frames stored on the camera, the system is made more efficient by sampling frames and determining the optimal object detection model to be used based on the specific case. With the selection of an appropriate model, the frames are processed and returned in order of how probable a frame is in matching the user’s criteria (Figure 1). Since not all the frames are streamed and it is cheaper to store data on-camera, fewer resources are required for this process, and the system becomes more efficient overall. Relevant benefits include limited consumption of network bandwidth and cloud resources and ease of use in remote areas.

![Figure 1. Overview of video capture and query system.](image)

This video analytics engine uses the Tensorflow and Keras machine learning systems to perform object recognition. To process frames on a camera, the video analytics system runs a lighter neural network on-camera and then runs an object detector in the cloud to validate the initial detection. The system chooses some frames as “landmarks” and runs a higher-accuracy object detector on those frames and then uses these frames as training samples later on. The system uses multiple iterations to detect objects in these landmark frames, using the results of each iteration to find more exact information. The types of queries that the system must handle are retrieval (frames containing an object), tagging (time ranges in which the video contains an object), and counting (statistics about the object count across frames). After having received a user query, the cloud retrieves landmark frames and begins training operators for the camera that are flexible based on the specifics of the query. As the query is processed, the cloud continues to train operators with the new frames so that parallel processes occur for camera processing, frame upload, and cloud processing. The frames that are then most likely to contain the relevant information are prioritized and returned before the others are.

CREATING A USER INTERFACE FOR CLIENT

In the context of this system, online processing is the most practical solution that will allow people to benefit from video analytics. Online data processing has been used in many other contexts of big data analysis, and having such a solution for video data would prove to be similarly beneficial. Online processing allows for a query to be executed based on a person’s request and allows a user to interact with a video to explore it by, for example, trying different parameters, terminating a query, and creating new queries from scratch.

To accomplish this task, an experimental web environment was established using Django and Bootstrap. Programming languages used to set this up were Python, HTML, CSS, and JavaScript, and the workflow for the web pages created were as follows: A home page launched a page with a map of the United States, in which individual states could be selected. With the selection of a state, the page was redirected to a page with a form for the user to fill out with the relevant information pertaining to her or his request. This included what object needed to be detected, the day the frame was to be retrieved, and the window of time that was of interest. A drop-down menu allowed the user to narrow down and specify the camera from which to view the frames. Submitting this form resulted in a page that told the user how many frames had been found fulfilling his or her criteria and asked if the user would like to view them. With this confirmation, a progress bar began to show how many of the frames had been retrieved, and the window of time that was of interest. A drop-down menu allowed the user to narrow down and specify the camera from which to view the frames. The final step of this project was to integrate the cloud program and its communication with cameras with the web-based user interface that will allow people to utilize the system. The Python scripts responsible for
running the video analytics engine would be started in response to HTML requests on web pages, and then the camera frames returned by the Python code that fulfilled the user’s criteria would be returned and displayed on the web page in a gallery format.

An initial prototype was created to implement the workflow outlined above. The key step after this was to establish a mode of communication and transfer of information between the user interface and the video analytics program itself (Figure 2). The program needed the ability to be run and stopped with commands from the web interface, and the relevant outputs of the program needed to be transferred back to the web interface to be displayed. To achieve this, the user input from the form was collected and stored to be sent to the program when it was run, and a button on the web page was set with the command to run the program when clicked. After the program had run, a separate process analyzed the images that had been retrieved to determine how many frames had been found to match the user’s criteria. Based on this information, an option was provided for the user to choose whether to load these images for viewing. If the user proceeded, then a progress bar would begin showing how many of the images had been loaded and then automatically redirect to the page containing thumbnails of the generated images. These images are called from the folder containing the resultant images from the output of the program, so any changes in the results of the query are automatically reflected in this gallery. This is useful for ensuring that the retrieved images are always up to date with the most recent call of the program. Of the alternative solutions considered, this was found to be the most functional and implementable method of retrieving images for this application.

RESULTS

To make this video analytics engine a more versatile and accessible tool, we developed a web-based interface to process user-specified queries and display the relevant retrieved video frames. The existing system was developed in Python, and the most suitable web-development framework used for this application was Django. The devised web interface was integrated with the video retrieval system to allow a user to specify a geographical location, choose a camera, and fill out a query with keywords, dates of capture, and a window of time. Upon submitting the request, the user can review how many frames matched the criteria and either choose to view the resultant images or revise the query.

My aim for the conclusion of this project was to have designed and deployed a web-interface that is well suited to the current needs of the video processing system that has been created in order to fully demonstrate its functionality. A demonstration of the final system is available at can be found in Alam (2019).
As seen in the video, the overall web interface follows the workflow that was outlined at the beginning of the project.

**Home page.** A home page provides information about the system. Clicking the start button takes the user to a page where he or she may select a state from which to view footage (Figure 3).

![Figure 3. Interface home page.](image)

**Geographical source.** For the purpose of this demonstration, Indiana was the chosen state. Clicking on “Indiana” redirects to a form for the user to fill out to specify the query (Figure 4).

![Figure 4. Geographical source selection.](image)

**User query form.** Information collected includes what object the user needs detected, the date of capture, the time interval of interest, and which camera’s footage should be examined (Figure 5).

![Figure 5. User query form.](image)

**Processing of request.** Clicking the “Process Request” button sends the form to the main program for processing, and clicking “Review Results” will display search results, form values as arguments, and run the code (Figure 6). Once the code has been run, the retrieved images will have been updated, and clicking “Review Results” shows search results of how many frames have been found to meet the specified criteria.

![Figure 6. Processing of request.](image)

**Search results and image retrieval status.** If the user finds the number of frames to be useful, he or she may click “View Frames” to load the frames (Figure 7). When the progress bar reaches 100%, the page automatically redirects to the gallery page, which displays thumbnails of all the retrieved images.

![Figure 7. Search results and image retrieval status.](image)

**Thumbnail gallery.** Mousing over the images enlarges them slightly, and clicking on one results in it being maximized (Figure 8).

![Figure 8. Thumbnail gallery.](image)
Navigating in Numerous Video Data

CONCLUSION

Creating a user-friendly web interface is an important link in ensuring that a solution created is accessible and useful for a more general population. In this case, this video analytics engine was created as a viable system to aid in making image and video frame processing more reliable and efficient and ensuring that it served the purposes relevant to the needs of specific applications. A web interface that integrates with this system then allows people to use this system more broadly and brings it closer to being used as a tool for developing practical solutions.

In the current implementation of this program, the data used for testing the source code is obtained from a YouTube video stream, which is configured to search for particular objects in this stream. The specifics of the user’s query are being stored independent of this testing data. The next step of this work is to simply link these two entities together so that the source code accepts and utilizes the modified query instead. Then, upon the expansion of using this system with more cameras and with its ability to process the objects being searched, the system will be prepared for full functional use.

There are many potential future applications of this technology, including:

1. Analysis of surveillance footage, relevant to criminal investigations;
2. Smart retailing;
3. Urban efficiency and road traffic analysis;
4. Health care and imaging diagnoses;
5. Facial emotion recognition (data for responses to products, media, or experiences);
6. Energy conservation and sustainability, using security cameras to determine occupancy and then adjusting heating/cooling and lights on a variable scale; and
7. Environmental studies, use of wildlife cameras, and efficiently processing data.

Additionally, this video analytics system is focused on retrospective analysis and image retrieval based on specified criteria, but with machine learning methods processing video frames stored on a camera, there may also be potential for real-time and actionable intelligence that can be used in a variety of applications. If it is known what specific criteria a user will want prior to the collection of data, then frames can be processed as they are being captured, which would make the process even more efficient for specific applications. Future applications of this project span many disciplines and have the potential to make positive impacts on the lives and well-being of many people.

ACKNOWLEDGMENTS

I am grateful to have had the opportunity to work on this project for a summer research internship in the Xroads Systems Exploration Lab at Purdue. This research was completed under the supervision of Dr. Felix Lin, with guidance from Mengwei Xu (previous laboratory member as a visiting scholar), and with support from the National Science Foundation.

REFERENCES