IEEE Global Conference on Signal and Information Processing 2014

A System for Large-Scale Analysis of Distributed Cameras

Ahmed S. Kaseb, Everett Berry, Youngsol Koh,
Anup Mohan, Wenyi Chen, He Li, Yung-Hsiang Lu, Edward J. Delp
School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN, USA
{akaseb, epberry, koh0, mohan11, chen345, yunglu, li522}@purdue.edu, ace@ecn.purdue.edu

***Abstract*—Thousands of cameras are connected to the Internet providing streaming data (videos or periodic images). The images contain information that can be used to determine the scene contents such as traffic, weather, and the environment. Analyzing the data from these cameras presents many challenges, such as (i) retrieving data from geographically distributed and heterogeneous cameras, (ii) providing a software environment for users to simultaneously analyze large amounts of data from the cameras, (iii) allocating and managing computation and storage resources. This paper presents a system designed to address these challenges. The system enables users to execute image analysis and computer vision techniques on a large scale with only slight changes to the existing methods. It currently includes more than 65,000 cameras deployed worldwide. Users can select cameras for the types of analysis they can do. The system allocates Amazon EC2 and Windows Azure cloud instances for executing the analysis. Our experiments demonstrate that this system can be used for a variety of image analysis techniques (e.g. motion analysis and human detection) using 2.7 million images from 1274 cameras for three hours using 15 cloud instances to analyze 141 GB of images (at 107 Mbps).**

***Index Terms*—Network Camera, Image Analysis, Big Data, Cloud Computing**

I. Introduction

Thousands of cameras are connected to the Internet providing image/video real-time streams. The streams contain valuable information related to traffic, weather, environment, landmarks, etc. From the same stream, meteorologists may study the formation of storms, city planners may review traffic management, and emergency responders may consider evacuation routes. Our team has discovered more than 65,000 public cameras deployed by various entities, such as government agencies (e.g. National Park Service and departments of transportation), universities, companies, or individuals. This tremendous amount of valuable information is lost if not analyzed. Hence, there is a need to analyze this data for a better understanding of the world around us.

Meanwhile, hundreds of image/video analysis methods are designed by researchers for a wide range of applications. Researchers usually evaluate their methods using datasets. For example, Bay et al. [1] used a dataset of 216 images to evaluate the Speeded Up Robust Features (SURF) method for keypoints detection. Dalal and Triggs [2] used two datasets of over 2500 images to evaluate their human detection method. *What would researchers do if they wish to evaluate analysis methods using 2,000,000 images? How could researchers use their methods to analyze the data from 65,000 cameras?*

This large-scale analysis of the data from cameras is beneficial for two main reasons: (i) It shows how competing analysis methods compare in a wide variety of real-life situations. (ii) Researchers have demonstrated that the visual information from many cameras can reveal useful information. AMOS (Archive of Many Outdoor Scenes) [3] periodically downloads images from thousands of webcams. The images are used to analyze environments and usage of public space. However, analyzing the data from thousands of cameras is a challenge. Hence, there is a need to build a computer system to tackle this challenge.

This paper presents CAM^2 (*Continuous Analysis of Many CAMeras*) as a system that addresses the following problems: (i) Given image analysis methods that analyze a single frame, how can these methods be executed on a very large scale without adding much burden to the user? A user may design analysis methods that can detect weather conditions, analyze traffic, monitor crowd, etc. CAM^2 takes responsibility for executing the analysis methods on the data from thousands of cameras. (ii) How can these analysis methods be executed on cameras selected by the user, and at the desired frame rates for the desired durations? CAM^2 retrieves data from the selected cameras and executes the analysis methods based on the specified parameters. (iii) How can data be retrieved from many cameras that are geographically distributed and heterogeneous, i.e. different brands, resolutions, etc.? CAM^2 simplifies migrating existing analysis methods by providing a simple Application Programming Interface (API) that hides the heterogeneity. CAM^2 allocates cloud instances to meet the computation and storage requirements of the analysis.

Our experiments show that CAM^2 is capable of analyzing 2.7 million images from 1274 cameras over three hours using 15 cloud instances, which means 141 GB of images (at 107 Mbps). The average resolution of the cameras is 0.44 Mega Pixels (MP). The cameras are deployed in North America, West Europe, and East Asia. For the experiments, we use two analysis methods: motion detection using background subtraction [4] and human detection [2].

This paper has the following contributions: (i) To our knowledge, this is the first system that enables users to execute methods analyzing the data from thousands of cameras simultaneously for extended periods of time. (ii) The system provides an API that makes it easy to migrate existing analysis

methods with only slight changes. Thus, this system can be used for a wide range of applications. (iii) CAM^2 provides access to more than 65,000 cameras (more are being added) that we discovered worldwide. (iv) This system allows users to specify parameters for selecting cameras, the frame rates, and the durations for executing the analysis methods. (v) CAM^2 allocates cloud resources to meet the computation and storage requirements of different analysis methods.

II. Background

Some web sites show periodic snapshots from cameras, for example, http://www.webcams.travel/ and http://www.wunderground.com/webcams/. Some government agencies (particularly departments of transportation) deploy cameras watching traffic. The AMOS project [3] has retrieved more than 400 million images from 18,000 cameras since 2006. Many studies have demonstrated that useful information can be extracted from the vast amount of visual data. Lalonde et al. [5] used time-lapse images for improving physics-based illumination models. Thakur et al. [6] used webcams to study vehicular traffic and the mobility models. Chen et al. [7] detected weather from images. Jacobs et al. [8] developed analysis methods to separate the AMOS dataset into subsets based on the scene in the images, such as the weather or the terrain. Graham et al. [9] used 1100 geolocated cameras for a plant phenology monitoring system. The above studies focus on specific tasks. Yu et al. [10] developed a platform for general real-time analysis of streaming video for cameras using local computation nodes. They built a system which analyzes several cameras for retail data mining. Previous studies focus on collecting and storing camera images, developing applications for analyzing large sets of images, or processing videos from small numbers of cameras. None of the existing studies integrates these functionalities into one system with an API. Neither does any existing work allow users to select cameras for different analyses at desired frame rates. In contrast, CAM^2 has an API that allows users to select cameras, specify frame rates, and execute a wide range of analysis methods on a large scale.

Two analysis methods are used for our experiments: (i) the background subtraction method proposed by KaewTraKulPong and Bowden [4], which uses an adaptive mixture model, and (ii) the human detection method proposed by Dalal and Triggs [2], which uses Histograms of Oriented Gradients (HOG). It is able to detect humans with different sizes, pose variations, and backgrounds. For each of the two analysis methods, we use the corresponding OpenCV [11] implementation. This paper does not present new analysis methods; instead, the paper presents a system that can execute analysis methods on a large scale.

III. CAM^2 System

This section demonstrates what could be achieved using CAM^2. It also shows how to use the API of the system to migrate existing analysis methods. Finally, it briefly discusses how the system is designed to meet the analysis requirements.

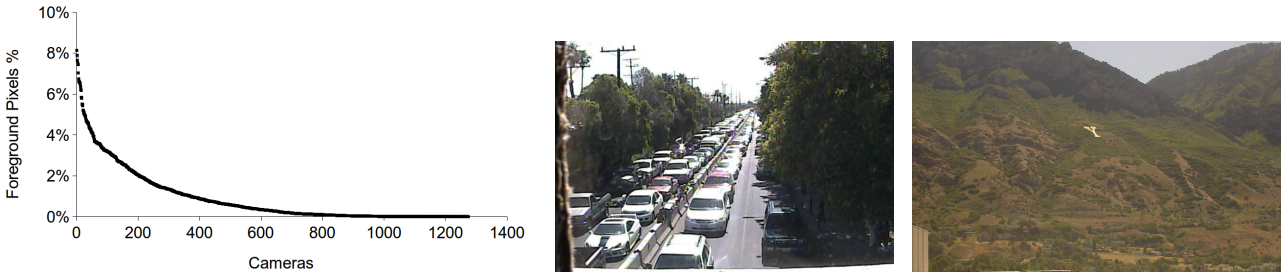
A. Large-Scale Analysis

The primary goal of CAM^2 is to enable users to execute a variety of methods to analyze the data from thousands of cameras. The sole responsibility of users is to submit their analysis methods (in Python) that analyze a single frame. Users can select cameras based on different criteria, such as country, state, city, and timezone. The system is responsible for executing the same submitted analysis method for all the selected cameras, and providing the aggregated results as well. In order to meet the analysis resource requirements, the system allocates and manages cloud instances.

In order to demonstrate the capability of CAM^2 to perform large-scale analysis, we conducted an experiment that analyzed 2.7 million images from 1274 cameras over three hours using one frame every five seconds from each camera. During the experiment, the system used 15 Amazon cloud instances to analyze 141 GB of images, which means a data rate of 107 Mbps ($141 \times 8 \times 1024/(3 \times 60 \times 60)$). The average resolution of the cameras is 0.44 MP (approximately 768×576). The cameras are deployed across North America, West Europe, and East Asia. The experiment calculates the average amount of motion in the images from each camera. First, the foreground of each frame is detected using OpenCV's implementation of the method proposed by KaewTraKulPong and Bowden [4]. Then, the percentage of the foreground pixels with respect to the entire image is calculated. The average percentage of foreground pixels over a period of time is an indication of the average amount of motion in the images.

Figure 1 shows the results of the experiment. Figure 1 (a) shows the distribution of the average amount of motion for all the cameras. The horizontal axis represents the cameras, while the vertical axis represents the average percentage of foreground pixels. This experiment indicates that 84% cameras had less than 1% foreground pixels and 2% cameras did not have any detected motion at all over the period of three hours. Since the motion indicates the existence of new information from a camera, this experiment demonstrates the ability of CAM^2 to analyze the data from thousands of cameras simultaneously, and identify which cameras deserve further investigation due to high degrees of motion. Figure 1 (b) shows a snapshot from a high-motion camera that is monitoring traffic, while figure Figure 1 (c) shows a snapshot from a low-motion camera that is looking at a landscape. It should be noted that panning cameras usually have the most amount of motion due to the frequent and sudden scene changes.

CAM^2 allocates cloud resources in order to meet the analysis requirements. The system is able to distribute the loads based on the different capabilities of each cloud instance. In this experiment, CAM^2 allocated 15 instances in order to meet the frame rate requirements. The achieved frame rate was 99% of the desired frame rate. Several factors could negatively affect the achieved frame rate: (i) Too few cloud instances are allocated and these instances are overloaded. The achieved frame rates drop noticeably when the CPU or memory utilization of the instances is over 80%. (ii) The



(a) (b) (c)

Fig. 1: Results of estimating the average motion in 2.7 million images from 1274 cameras over three hours using one frame every five seconds. (a) Distribution of the average percentage of foreground pixels for all the cameras. (b) A camera with high motion in Mexico. (c) A camera with low motion in USA.

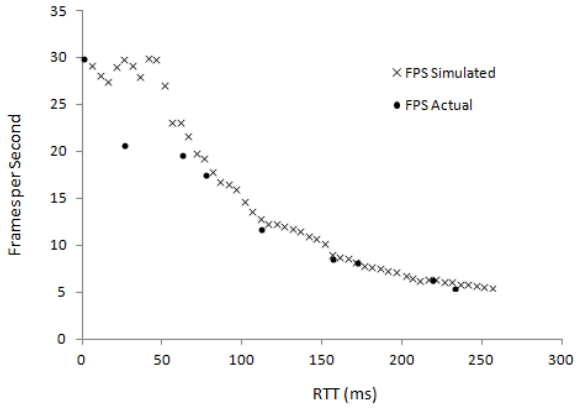


Fig. 2: The relation between the frame rate of a Motion JPEG stream from a camera and the Round Trip Time (RTT). The network emulator *netem*[12] was used for the emulated data.

cameras are geographically far from the cloud instances. For example, a cloud instance in USA analyzes the data from a camera in Europe.

The frame rates achieved from a camera are dependent on the Round Trip Time (RTT) from the cloud instance. Figure 2 shows the relationship between RTT and the achieved frame rates. The horizontal axis represents the RTT and the vertical axis represents the achieved frame rates. To observe the effects of RTT, the network emulator *netem*[12] was used. The experiment consisted of a network camera providing a Motion JPEG video stream and a cloud instance located close to the camera. The cloud instance can analyze frames from the network camera at a rate of 30 frames per second (fps). Then, *netem* added delays (0 to 255ms) between the network camera and the cloud instance. To verify the emulation results, the same network camera was analyzed from different cloud instances located at different parts of the world. The emulation results and the actual measured frame rates match closely. The figure shows both the measured and emulated data. This experiment shows the need of allocating a cloud instance geographically close to the network camera if a high frame rate is desired.

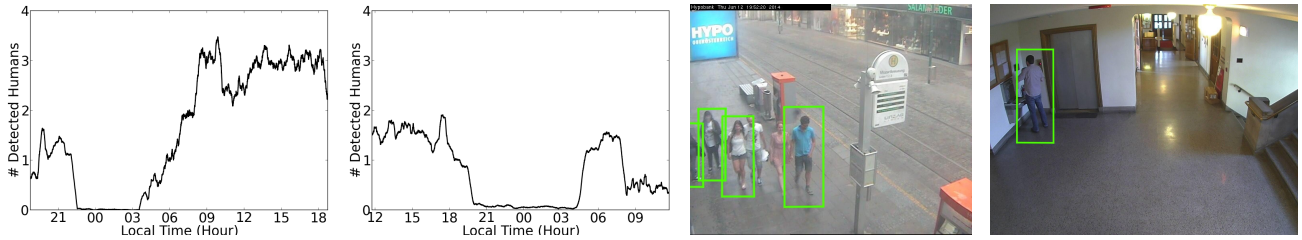
In addition, CAM^2 enables users to analyze the data from cameras for extended periods of time. Before submitting an analysis method, the user specifies the duration of analysis (e.g. 24 hours, or 7 days), and the desired frame rate. In order to demonstrate CAM^2's capability performing analysis for extended periods of time, we conducted an experiment analyzing two cameras for 24 hours using one frame every five seconds. The experiment detects and counts the number of people in the images using OpenCV's implementation of the method proposed by Dalal and Triggs [2]. The selected cameras were located in Linz, Austria and Chicago, USA. The resolutions are 1280×960 and 704×576 respectively.

Figure 3 shows the results of the experiment. Figure 3 (a) shows the average number of detected humans in the images from the camera in Austria over 24 hours, while Figure 3 (b) is from the camera in USA. The horizontal axis represents the local time, while the vertical axis represents the average number of detected humans. The average is post-processed using a moving average filter with a 30-minute window. It is clear that few people are detected during the night, while more people are detected during the day of each camera respectively. More people appeared in the first outdoor camera (a bus stop) than the second indoor camera (a corridor). This experiment demonstrates the ability of CAM^2 to analyze the data from cameras for extended periods of time. This can be beneficial for a wide range of image analysis and computer vision applications, such as weather detection, traffic monitoring, etc.

B. Application Programming Interface (API)

CAM^2 provides an API for users to develop analysis methods that can be easily executed on a large scale. The API is simple to use, and it requires few changes to existing methods. As a result, users can focus on designing the analysis methods, while CAM^2 handles everything else needed to execute the methods on a large scale. The API currently supports Python using OpenCV.

The general structure of video processing methods can be divided into three main stages. The first stage allocates resources and performs required initialization. The second stage reads and processes each frame, and saves the results. The third stage computes the final results, such as calculating the statistics, generating the summary, releasing resources, etc. Figure 4 (left) shows a snippet from a background subtraction method that follows the same structure. Lines 1-2 correspond



(a) (b) (c) (d)

Fig. 3: The average number of people in the images from two cameras over 24 hours using one frame every five seconds: (a) Linz, Austria. (b) Chicago, USA. The numbers are determined using a moving average filter with a 30-minute window. (c) Sample output for the first camera (4 out of 6 people detected). (d) Sample output for the second camera (one person detected).

```
1  cap = cv2.VideoCapture('video.avi')
2  fgbg = cv2.createBackgroundSubtractorMOG()
3
4  while(1):
5      ret, frame = cap.read()
6      fgmask = fgbg.apply(frame)
7      cv2.imwrite('foreground.jpg', fgmask)
```

```
1  def initialize(self)
2      self.fgbg = cv2.createBackgroundSubtractorMOG()
3
4  def on_new_frame(self):
5      frame = self.get_last_frame()
6      fgmask = self.fgbg.apply(frame)
7      self.save('foreground.jpg', fgmask)
```

Fig. 4: An example of migrating existing image analysis method to CAM^2: (left) A snippet from OpenCV for background subtraction. (right) A snippet from the corresponding CAM^2 method.

to the initialization stage. Lines 5-7 correspond to the processing stage: reading the input frame, subtracting the background, and saving the foreground mask respectively.

CAM^2 adopts the event-driven programming model: analysis methods are invoked when new frames arrive. The API supports different events for the stages of initialization, processing, and finalization, and it allows saving results in different formats, including text, image, and video. To migrate an existing analysis method to CAM^2, a user needs to map the three stages to the corresponding events in CAM^2 as shown in Figure 4 (right). The second line corresponds to the initialization stage. The fourth line changes from `while` to the event of a new frame. Lines 5-7 correspond to processing the new frame. As can be seen in this figure, migrating an existing analysis method to CAM^2 requires only a few changes.

C. CAM^2 Architecture

This section describes the architecture of CAM^2, as shown in Figure 5. The system consists of two main components: (i) the website through which users can select cameras and submit analysis methods and (ii) the cloud-based distributed system which handles executing the methods for analyzing the data from the selected cameras. Analysis methods are unaware of the underlying computing facilities, which reduces the burden on the user. Moreover, cloud instances retrieve data from the cameras directly without passing through any central server and this greatly enhances scalability and reliability.

We discovered more than 65,000 public cameras on the Internet. These cameras are heterogeneous, i.e., they have different brands, resolutions, and frame rates. As a result, retrieving data from these cameras requires different methods. One contribution of CAM^2 is its capability of handling and hiding this heterogeneity so that the same methods can analyze

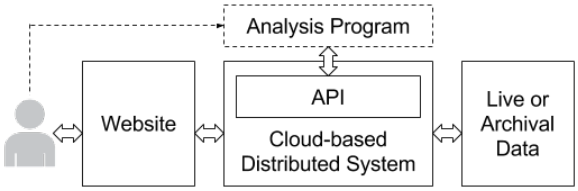


Fig. 5: Architecture of CAM^2. Thick arrows represent communication. The dash block is the responsibility of users, while other blocks are the responsibility of CAM^2.

the data from many cameras. CAM^2 maintains metadata about the cameras, including approximate locations, brands, resolutions, frame rates, etc. CAM^2 will soon support non real-time data from archives.

IV. Conclusion

This paper describes the CAM^2 system, which is capable of analyzing the data from thousands of distributed cameras. CAM^2 can retrieve data from the heterogeneous cameras and execute analysis methods on cloud instances. The event-driven API simplifies migrating existing analysis methods to CAM^2. The experiments demonstrate that this system can analyze 2.7 million images from 1274 cameras over three hours, which means 141 GB of images (at 107 Mbps).

V. Acknowledgements

This project is partly supported by NSF CNS-0958487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF. We would like to thank Amazon and Microsoft for providing the cloud instances, and the organizations that provide the visual data. The list is available at https://cam2.ecn.purdue.edu/acknowledgements.

REFERENCES

[1] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proceedings of the European Conference on Computer Vision*, vol. 3951 of *Lecture Notes in Computer Science*, (Graz, Austria), pp. 404–417, May 2006.

[2] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, (San Diego, CA, USA), pp. 886–893, June 2005.

[3] N. Jacobs, N. Roman, and R. Pless, "Consistent temporal variations in many outdoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, (Minneapolis, MN, USA), pp. 1–6, June 2007.

[4] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems* (P. Remagnino, G. Jones, N. Paragios, and C. Regazzoni, eds.), pp. 135–144, Springer US, 2002.

[5] J.-F. Lalonde, A. A. Efros, and S. G. Narasimhan, "Webcam clip art: Appearance and illuminant transfer from time-lapse sequences," *ACM Transactions on Graphics*, vol. 28, pp. 131:1–131:10, Dec 2009.

[6] G. S. Thakur, P. Hui, and A. Helmy, "A framework for realistic vehicular network modeling using planet-scale public webcams," in *Proceedings of the ACM International Workshop on Hot Topics in Planet-scale Measurement*, (Low Wood Bay, Lake District, UK), pp. 3–8, June 2012.

[7] Z. Chen, F. Yang, A. Lindner, G. Barrenetxea, and M. Vetterli, "Howis the weather: Automatic inference from images," in *Proceedings of the IEEE International Conference on Image Processing*, (Orlando, FL), pp. 1853–1856, Sept 2012.

[8] N. Jacobs, R. Souvenir, and R. Pless, "Passive vision: The global webcam imaging network," in *Proceedings of the IEEE Applied Imagery Pattern Recognition Workshop*, (Washington, DC, USA), pp. 1–8, Oct 2009.

[9] E. A. Graham, E. C. Riordan, E. M. Yuen, D. Estrin, and P. W. Rundel, "Public internet-connected cameras used as a cross-continental ground-based plant phenology monitoring system," *Global Change Biology*, vol. 16, pp. 3014–3023, Nov 2010.

[10] T. Yu, B. Zhou, Q. Li, R. Liu, W. Wang, and C. Chang, "The service architecture of real-time video analytic system," in *Proceedings of the IEEE International Conference on Service-Oriented Computing and Applications*, (Taipei, Taiwan), pp. 1–8, Jan 2009.

[11] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, vol. 25, pp. 120, 122–125, Nov 2000.

[12] "Netem." http://www.linuxfoundation.org/collaborate/workgroups/networking/netem. Accessed June 15, 2014.