Edge-based Content-aware Crowdsourcing Approach for Image Sensing in Disaster Environment

Ziming Zhao
School of Computer,
National University of Defense Technology
Changsha, China
zhaoziming@nudt.edu.cn

Fang Liu
School of Computer,
National University of Defense Technology
Changsha, China
liufang@nudt.edu.cn

Zhiping Cai
School of Computer,
National University of Defense Technology
Changsha, China
zpcai@nudt.edu.cn

Nong Xiao
School of Computer,
National University of Defense Technology
Changsha, China
nongxiao@nudt.edu.cn

ABSTRACT

Photos obtained via crowdsourcing can be used in image sensing for disaster management. Due to the weak communication environment after a disaster, it is difficult to transfer the huge amount of crowdsourced photos. To address this problem, we propose COCO, a content-aware crowdsourcing system that leverages edge computing to support real-time image sensing in disaster environment. COCO filters the crowdsourced images at the data source and only uploads the images that contain relevant objects which the application is interested in. We use a machine-learning based computer vision detector to understand the content of images. Considering the resource constraints of mobile devices, we implement the computer vision detector at the edge server which located in the close proximity to data source. As the unstable network bandwidth is normal in disaster environment, we propose an adaptive mechanism to further improve the sensing performance. We have implemented the COCO prototype which is evaluated via a real-world dataset. The experimental results demonstrate the effectiveness of COCO.

CCS CONCEPTS

• **Information systems → Mobile information processing systems**;

KEYWORDS

Edge Computing, Crowdsourcing, Image Sensing

ACM Reference Format:
Ziming Zhao, Fang Liu, Zhiping Cai, and Nong Xiao. 2017. Edge-based Content-aware Crowdsourcing Approach for Image Sensing in Disaster Environment. In *MobiQuitous 2017: the 14th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, November 7–10, 2017, Melbourne, VIC, Australia.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3144457.3144499

MobiQuitous 2017, November 7–10, 2017, Melbourne, VIC, Australia

ACM ISBN 978-1-4503-5368-7/17/11…$15.00
https://doi.org/10.1145/3144457.3144499

1 INTRODUCTION

During times of disaster, people often take photos to document and make sense of these events. The photos are full of sensing information, such as the surroundings and individuals, road conditions, that present the real situations and provide vivid description of in-situ objects[7]. So, crowdsourcing for image sensing is important and helpful to support disaster recovery and emergency management[1]. For example, crowdsourcing has been applied in the Haiti earthquake to collect images of disaster scenes, and the images are stitched into a zoomable panorama as a new map in the disaster areas[4].

Although the image-based information is beneficial for disaster management, the image sensing via crowdsourcing fails to efficiently support the disaster management due to two main limitation. 1) Strict Bandwidth Constraint. The network bandwidth becomes very limited in capacity, which is caused by the destruction of communication infrastructure. Though there are some schemes[2, 5] to remedy the network communication, the bandwidth bottleneck still remains. 2) Slower Uploading Speed. Under the limited network bandwidth, users will spend more time on uploading images. The slower uploading speed makes the real-time image analytics difficult to realize. Moreover, the longer transmission time will cost higher energy which is the most precious resources for mobile devices in disaster environmen [9].

A large number of crowd-sourced images provide no useful information for the applications of image sensing. For example, People Locator[14], a face recognition based Lost People Find application, collected 15,000 crowd-sourced images of survivors during Haiti earthquake. The image uploading of this application is no constraints, so many images of survivors contain zero face, or contain partially occluded, turned-away, damaged faces. Human is hard to identify the people in these images, let alone the face recognition algorithm. Image crowdsourcing mainly aims to provide sensing information for a certain application. Collecting the images contain no useful information consumes the limited resources and is not helpful to the application. Hence, the applications of image sensing

require a system to provide crowdsourced images which contain relevant objects that the application is interested in. For example, if there is a system which guarantees the crowdsourced images uploaded to the People Locator must contain one or more faces, many images will not be transmitted through the weak network. However, to achieve the goals of the content-aware system, there are several challenges to address.

- The computing resources of mobile deviced is limited. We should use machine learning based computer vision detectors, such as object detection, to understanding the content of each image rather than relying on humans. Most of computer vision detectors are based on a deep neural network which is known for pushing traditional computing systems to their limits and requires the GPU acceleration to get the best performance. Smartphones are often resource constrained in terms of processor and memory capacity. Processing computation-intensive tasks need to be assisted by external computing resources.
- The network bandwidth is not stable in disaster environment. With the begin and end of the network congestion, the bandwidth varies a lot. In this environment, improving the performance efficiency of COCO needs a careful design. The computer vision detectors may become the bottleneck of our processing pipeline when the network is idle. The network might also become the bottleneck during the network congestion.
- The computer vision detectors are not entirely accurate. Some images may meet the content constraints, but not be uploaded to the cloud. This is caused by the fault judgment of the computer vision detectors and leads to some degree information loss. Though the content constraints can improve the uploading speed, the side effect of information loss should not be ignored.

To address these challenges, we propose a content-aware system, called COCO, to collect images in disaster environment. COCO can constrain the content of crowdsourced images, and images which not satisfy the constraints will not be uploaded to the application in the cloud. Moreover, by monitoring the changes of bandwidth, COCO adaptively adjust its behaviors to obtain the fastest uploading speed. COCO also limited the upper bound of information loss when optimizing the uploading speed. To achieve these design goals, we have the following contribution.

- To solve the resource constraint in mobile devices, COCO proposes an edge-based model to constrain the content of images uploaded. This model uses the Mobile Edge Computing technologies to provide powerful computing resources at the proximity of the data source.
- To optimize the uploading speed under different bandwidth, we find the optimal processing speed of the computer vision detectors under a fixed bandwidth and propose an adaptive mechanism to make the process speed optimal with bandwidth variation.
- We have implemented the COCO prototype and evaluated its performance by using a real-world image set. The experiment demonstrates that COCO can constrain the content of images uploaded to the cloud and control the information loss within the limit. Moreover, the adaptive mechanism speeds up the image uploading by 21% ∼ 35%, compared with the system without the adaptive mechanism.

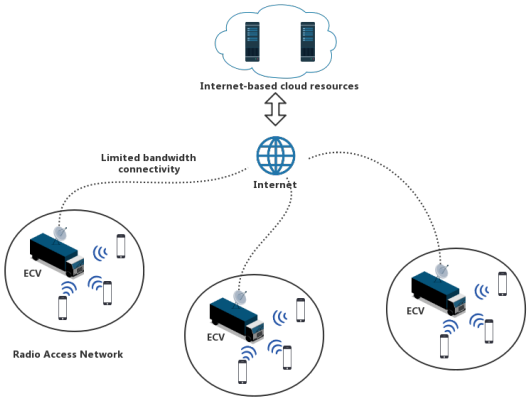

Figure 1: Emergency Communication System in Disaster

The rest of this paper is organized as follows. Section 2 and 3 respectively describes the background and architecture of COCO. The design details of COCO are described in Section 4. We evaluate the performance in Section 5. Section 6 discusses the related work and Section 7 concludes this paper.

2 BACKGROUND

Ground infrastructures are often damaged and rendered useless during natural disasters, and it could be many days or weeks before these can be repaired. In this environment, emergency communication is widely used to keep the communication running[8]. The emergency communication system in disaster environment is shown in Fig 1. Emergency communication devices, such as emergency communication vehicle (ECV), can provide the radio access network(RAN) for user access, and the backend network (BN) between the ECV and cloud usually uses satellite communication or microwave communication which is not affected by any localized condition. Supported by the state-of-the-art radio interface technologies, the bandwidth of RAN can easily achieve high bandwidth[13]. However, the BN should transmit data over a long distance, so the bandwidth of BN is hard to improve[10]. The BN is the bottleneck of the emergency communication.

Mobile Edge Computing[6] is a new technology and has got fast development during this year. Mobile Edge Computing provides an IT service environment and cloud-computing capabilities at the edge of the mobile network, within the RAN and in close proximity to mobile subscribers. Supported by the mobile edge computing technology, the ECV can be changed to include computation modules such as localized processing and content storage. So, with the help of Mobile Edge Computing, mobile devices in disaster environment can use the powerful computing resources in the ECV to process the computation-intensive tasks.

3 EDGE-BASED DATA COLLECTION ARCHITECTURE

We propose a novel system architecture to collect the crowd-sourced images that contain specific contents from the smartphones. The architecture is based on the mobile edge computing technologies, so the computer vision detectors can be implemented in the ECV and the images that not contain the specific contents will not be delivered to the cloud through the backend network.

The architecture is shown in Figure 2. The smartphones compress the images and upload the images through the ECV. In the ECV, every network connection between smartphones and ECV has a queue to buffer the uploaded images. The Device Comm receives the images and sends them to the queue. When the queue is full, the Device Comm will stop receiving images until the queue has free space again. The Detector is the computer vision detectors to classify the images based on whether the images contain the specific contents. The images that contain the specific contents will send to the Comm Module for delivering to the cloud through backend network. COCO sets a buffer between the Detector and Communication Module in order to overlap the communication with computation. Many users may simultaneously upload images through the ECV, so a polling mechanism is used to allocate processing time in each image queue and every queue will be allocated the same time slice. The emergency communication devices provide public network access, so the Comm Module shares the bandwidth of backend network with other applications. The saved bandwidth of COCO can be provided for other applications, and the available bandwidth is also influenced by other applications in a period of peak demand.

The controller, listener, and detector set make up the adaptive mechanism in COCO. This mechanism aims to make the system get the best image uploading speed under the changes of network bandwidth. We will introduce the mechanism in detail in the next section.

4 DESIGN DETAIL

The network bandwidth between the ECV and cloud is unstable in the disaster environment, which hardly influences the performance of COCO. For further improving the performance, we optimize the system from details and propose an adaptive mechanism to adjust our system to changes of bandwidth.

4.1 Detector Set

As described in Section 3, the available bandwidth of our system is influenced by other applications that communicate with the outside through the ECV. In a period of peak demand, the backend network will be congested and the available bandwidth drops significantly. Similarly, in the idle time, the available network bandwidth will also increase.

Under different bandwidth, the bottleneck of the processing pipeline may different. When the network congestion occurred and bandwidth significantly dropped, the bottleneck of the processing pipeline is the network bandwidth. However, when the network was idle and the bandwidth increased, detection speed of detector may become the bottleneck. If the bottleneck is the network bandwidth, reducing the size of data transmitted can improve the uploading speed. If the detection speed is the bottleneck, speeding up the detector can improve the uploading speed.

Increasing the threshold of the detector can reduce the size of data transmitted, but sacrifice the recall. Lower recall means more information loss, so it is not a good method. Instead, increasing the accuracy of the detector also can reduce the size of data transmitted without reducing the recall, because more accurate detector achieves higher precision when getting the same recall. So, COCO needs methods to increase the detection accuracy and speed.

COCO uses a speed-accuracy trade off as the method to increase the detection accuracy and speed because COCO does not simultaneously need both high detection accuracy and high detection speed. When the network congestion occurs, the system needs the high detection accuracy rather than the high detection speed, appropriate sacrificing the detection speed will not influence the uploading speed. When the network is idle and the bandwidth increases, the system needs high detection speed rather than high detection accuracy, sacrificing the accuracy increases the size of data transmitted but the network bandwidth is not the bottleneck in this time.

There can be some tradeoff between detection speed and accuracy. For example, using more complicated neural network model (improve the number of layers and number of neurons in each layer) can improve the accuracy of the detector, but slow down the detection process. Moreover, some detector has the parameter to make the tradeoff between speed and accuracy, such as the scale factor of image pyramid and scan step size.

Using the speed-accuracy tradeoff, we can get several detectors which have different detection speed and accuracy. These detectors are composed of the detector set in COCO. These detectors can get the same recall with carefully adjusting the threshold of each detector. So, we can set an appropriate recall rate according to the acceptable information loss. Using different detectors do not increase the information loss.

4.2 Optimal Detector

Based on the detector set, COCO can switch detectors with the changes of network bandwidth. So, we need to select the optimal detector under the current bandwidth to get the fastest images uploading speed. An appreciate detection speed can improve the image uploading speed, but higher detection speed is not always the better. We formulate the image uploading speed as follows. Assume that the image queues totally contain N images to upload, the radio of images contain faces is β, and the average image size is S. To simplify the formulation, we assume that the buffer between detector and communication module is unlimited. The image detection and communication have high parallel efficiency with high overlapping ratio between two parts. So, the overall uploading time T can be estimated as the major of the processing time of each part, and be expressed as

$$T = max(\frac{N * S}{V}, \frac{N * \beta * \alpha * S}{B}) \quad (1)$$

where V is the detection speed and B is the available network bandwidth. The number of images delivered to the cloud is $N * \beta * \alpha$ rather than $N * \beta$ because the detector is not entirely accurate. we can estimate the value of α by the ratio of recall to precision. We

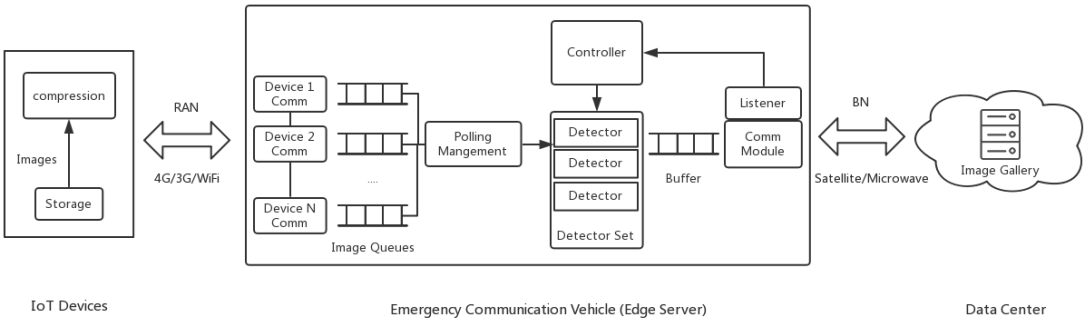

Figure 2: System Architecture

can formulate the image uploading speed as

$$UploadingSpeed = \begin{cases} V & V < \frac{B}{\beta * \alpha} \\ \frac{B}{\beta * \alpha} & V \geq \frac{B}{\beta * \alpha} \end{cases} \quad (2)$$

The speed higher than $\frac{B}{\beta * \alpha}$ can not improve the uploading speed but send more images to the buffer. The uploading speed reaches a maximum when the detection speed is $\frac{B}{\beta * \alpha}$. Though the value of β and α changes significantly based on image set, it can be regarded as constant in the long run. So, the best detection speed is determined by the available network bandwidth. Based on the best detection speed, we can select the optimal detector.

4.3 Adaptive mechanism

With the begin and end of the network congestion, the network bandwidth varies a lot and the optimal detector also changes. In this section, we propose an adaptive mechanism to automatically select the optimal detector with changes of the bandwidth. The adaptive mechanism is composed of two parts, namely listener and controller. The Listener detects the changes of network bandwidth which are caused by the network congestion, and the controller judges which is the optimal detector.

4.3.1 Listener. The network bandwidth can not remain constant, and the bandwidth fluctuation is normal. Fortunately, not all changes need to signal the controller. The controller is only interested in the great changes which are caused by network congestion rather than the normal bandwidth fluctuation.

The algorithm of bandwidth change detection is shown in Algorithm 1. The algorithm distinguishes the changes of bandwidth caused by network congestion through the variance of the last k bandwidth records. If the variance is larger than a threshold T, the Listener judges the bandwidth variation caused by the network congestion, and signals the controller to make the speed-accuracy tradeoff.

4.3.2 Controller. Through the Listener, COCO can determine when to make the speed-accuracy tradeoff. The role of the controller is to judge which is the optimal detector. From the formula 2, the detector is optimal when the speed of images entered the buffer (SIEB, $V * \beta * \alpha$) is equal to the network bandwidth (B). So, we choose the detector which makes the SIEB equal to the network bandwidth.

The SIEB of a single image varies greatly, and the network transmission speed is also the same. So, COCO compares the average speed of N images instead of a single image. Moreover, COCO relaxes the condition of equality. The two speeds are considered equal when the speed difference is smaller than a threshold T.

Algorithm 1 Network Congestion Detection

Parameters: $currentQueueLen, T, B, k$
procedure $BandwidthDetection(bwArray, len)$
// $bwArray$: The array that records the bandwidth.
// len: The current length of $bwArray$.

1: $bwArray[len] = getCurrentBandwidth()$
2: $len = len + 1$
3: $variance = calculateVariance(bwArray, k)$ //Calculate the variance of last n records.
4: **if** $variance > T$ **then**
5: //Send signal to Controller to start the speed-accuracy trade off

end procedure

5 EVALUATION

5.1 Experiment Setup

To evaluate the performance of COCO, we implement COCO prototype which collects the images that contain one or more faces from the smartphones. The prototype is composed of three parts, namely the client application, edge server and cloud storage service. The client application is programmed in Java and installed into the Android-based smartphones for evaluation. The smartphone is equipped with Snapdragon 835 8-core CPU at 2.45GHz, a 128GB ROM and a 6GB RAM. The edge server is programmed in C++ and uses the OpenCV library to process images. We use the Multi-task Cascaded Convolutional Network (MTCNN)[17], a real-time face detector that uses a deep neural network, as the detector in our system. The edge server is implemented in the Ubuntu 16.04 operating system running on an 8-core CPU each at 3.40GHz and a NVIDIA

Algorithm 2 Selecting the Optimal Detector

Parameters: $N, T, interval, scaleFactor$
procedure $SelectDetector(currentDetector)$

1: $averageNTS = getNetTransSpeed(N)$ //record the average network transmission speed of each image in the next N images
2: $averageSIEB = getSIEB(N)$ //record the average SIEB of the next N images
3: $speedDiff = averageNTS - averageSIEB$
4: **if** $-T < speedDiff < T$ **then**
5: $break$ //Two speeds are equal, $currentDetector$ is optimal
6: **else if** $speedDiff > T$ **then**
7: $SelectDetector(fasterDetector)$ //test the faster, but less accurate detector
8: **else**
9: $SelectDetector(slowerDetector)$//test the more accurate, but slower detector

end procedure

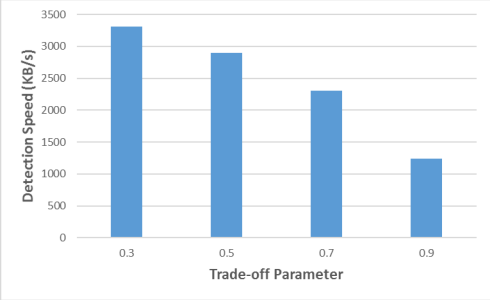

Figure 3: Detection Speed [KB/s] for different trade-off parameters.

GTX 1080 GPU. We use the object storage service of Alibaba Cloud as the cloud storage service.

The smartphones are connected via a high-speed 5GHz 802.11n network to the edge server, and the network between the edge server and cloud is a WAN network. We use the Wonder Shaper[16], a tool to limit the bandwidth of network adapters, to simulate the low bandwidth. As for the experimental data set, we collected 500 images from the Flickr, and all the images were taken in the real disaster environment, such as China 512 earthquake and Nepal earthquake. The average frame size is 1MB.

5.2 Efficiency of the Speed-Accuracy Trade Off

The MTCNN has a parameter to do the tradeoff between detection speed and accuracy. We used this trade-off parameter to do the speed-accuracy tradeoff in our system.

We manually classified the experiment image set and find that only 229 images contain completed faces. 271 images contain zero face or the hardly damaged faces. Using the classification results as ground truth, we set the trade-off parameter of the detector and used the detector to classify the images. We change the trade-off parameter of the detector from 0.3 to 0.9 with the interval of 0.2. The experiment results are as follows.

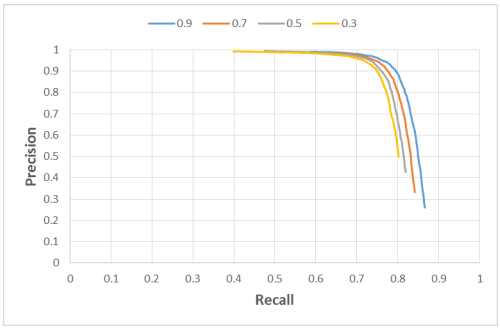

Figure 4: Detection performance of detector with different trade-off parameters.

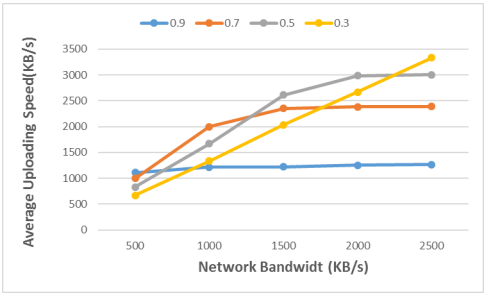

Figure 5: Average uploading speed under different conditions.

The detection speed is measured by the size of images processed per second. As shown in Figure 3, the detection speed drops from 3300 KB/s to 1200 KB/s with the scale factor growing. Figure 4 shows the relationship between recall and precision under each trade-off parameter. Getting the same recall, the detector with larger parameter has higher precision. So, the detector with larger trade-off parameter is more accurate.

The experiment results show that the trade-off parameter can make an effective tradeoff between detection speed and accuracy. Table 1 is shown the precision of the detector with different parameter when the recall is adjusted to 0.8. We regarded the detectors with different parameter as different detectors and used the 4 detectors in Table 1 to make up the detector set in the next experiment.

Table 1: The precision of detectors when the recall is adjusted to 0.8

Trade-off Parameter	Precision
0.3	53.3%
0.5	66.7%
0.7	80.3%
0.9	88.1%

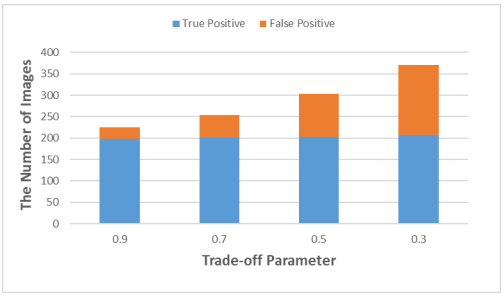

Figure 6: The constituent of collected images in the cloud.

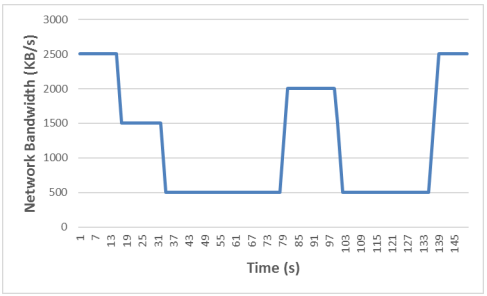

Figure 7: The network bandwidth changes over time.

5.3 Effectiveness of Optimal Detector

We used the experiment in this section to prove that the bottleneck of our system changes with the network bandwidth, and the speed-accuracy tradeoff can improve the uploading speed.

In this experiment, 5 smartphones simultaneously uploaded images through the edge server, and each smartphone uploaded 100 different images. We did 5 sets of experiments under 5 different bandwidth (from 500KB/s to 2500KB/s). In each set of experiments, we tested the performance of the system which uses the detector in detector set, and we did an experiment with every detector in detector set. Our detector set has 4 different detectors as described in 5.2, so we totally did 20 experiments under different conditions. Figure 5 shows the average uploading speed of the 5 smartphones under different conditions. The uploading speed of each smartphone is the ratio of the size of 100 images to total uploading time. Figure 6 shows the constituent of collected images in each experiment. There are only 4 results because the network bandwidth does not influence the results.

From the results, we can find that the optimal detector is different under different bandwidth. When the bandwidth is low, the detector that has the lower detection speed is optimal. This is because the detector that has the lower detection speed is more accurate that makes the low bandwidth network transmits fewer data to the cloud. As shown in Figure 6, the true positive image numbers of each detector are roughly the same because each detector has the same recall, but the lower speed detector significantly decreases the false positive image numbers. So, when the bandwidth is low, the network bandwidth is the bottleneck of our system and improving the accuracy of the detector can speed up the uploading.

For the system that uses one fixed detector, the uploading speed will not increase with the rising bandwidth after the bandwidth exceeded a certain value because the detection speed has become the bottleneck of the processing pipeline. So, we should change a faster detector in order to increase the uploading speed. Though the faster detector uploads more false positive images to the cloud, but the higher bandwidth counteract this side effect.

5.4 Performance of COCO

In this section, we did experiment to test the performance of our system under the changes of network bandwidth. To prove the effectiveness of the adaptive mechanism, we also tested the performance of the systems that use the fixed detector. In this experiment, the systems have the same processing time (150 seconds) and are

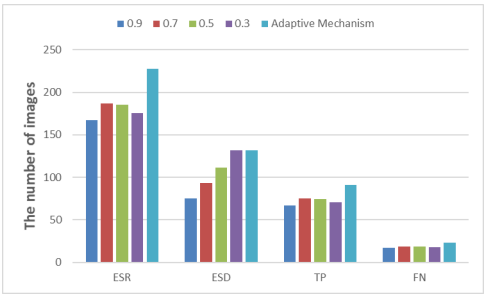

Figure 8: The performance of each system.

under the same network environment. We emulated the changes of network bandwidth during network congestion. That the network bandwidth changes over time is shown in Figure 7. During the processing time, 5 smartphones used the system continuously upload images to the cloud. We compared the uploading results of the systems.

We compared the performance of each system by the number of images processed during the 150 seconds .The experiment results are shown in Figure 8. Our detector set includes 4 detectors, so there are 4 different systems that use the fixed detector. The first set of results is the number of images the edge server received from the smartphones (ESR for short), which determines the image uploading speed. Within the same amount of time, the system with adaptive mechanism uploaded 21% ~ 35% more images than systems that use the fixed detector.

The second set of results is the number of images edge server delivered to the cloud (ESD for short). Though the systems are under the same network condition, there are still some gaps in the ESD of different systems. This is because the detection speed is the bottleneck of some systems.

The third set of results is the number of true positive images the cloud received (TP for short). The detector can not have 100% precision, so there are some false positive images in the ESD. The TP is determined by the ESD and precision of the detector. The detector with 0.9 trade-off parameter has the highest precision but the ESD of the system uses the detector is low. The system which uses the fixed detector with 0.3 trade-off parameter has a high value of ESD, but the precision of the detector is low. The system with adaptive mechanism can have the advantage of both, so it got the highest TP.

The fourth set of results is the number of false negative images (FN for short). The false negative images are the images that meet the content requirement but not delivered to the cloud because of the fault judgment of detector. The FN is determined by the recall of detector and the ESR. In our system, the recall rates of all the detectors in detector set are set to the same value (0.8 in this experiment) according to the upper bound of information loss. Hence, the adaptive mechanism does not influence the recall. The system with adaptive mechanism has the highest ESR, so the FN is also the highest. The FN can be further decreased by setting higher recall but that may sacrifice the image uploading speed.

6 RELATED WORK

Eyewitness photo is increasingly playing a more significant role in disaster response and recovery efforts. Caltech[3], a community sense and response systems, gather and share images from Internet-enabled devices for real-time awareness of dangerous earthquakes. BEES[11] is also an image sharing system to offer real-time situation awareness in the disaster environment, which uses the content-based redundancy elimination to improve the bandwidth and energy efficiency. These crowdsourcing systems collect images for the general image sharing and do not have a specific requirement for the content of images. However, COCO collected crowdsourced images for a specific application which has a clear requirement to the content of images, such as Lost People Find.

Closest in spirit to COCO is a computer vision system called SAPPHIRE[15] allows client devices to continually analyze streams of video and distil out frames that contain objects of interest. Moreover, Vigil[18] is also a wireless video surveillance system that only uploads significant associated video frames to the cloud. However, these systems are not designed for disaster environment, they can not get their best performance under the unstable network environment. COCO uses the mobile edge computing technologies assist the smartphones to process the computer vision algorithm and the adaptive mechanism ensure the performance efficiency under the unstable network environment.

Mobile edge computing is new and poised for rapid growth along with the growth in mobile devices and cloud technology[6]. This technology pushes the frontier of cloud service to the edge of the network. The edge server located in close proximity to associated mobile devices can overcome many particular problems caused by the disaster[12], such as the weak network environment.

7 CONCLUSION

In this paper, we proposed a content-aware system for continuous collection of crowd-sourced images from smartphones in disaster environment. COCO can constrain the content of crowdsourced images, and only upload the images that meet the content requirement of the application to the cloud. COCO uses the computer vision detector to understand the content of images, and implements the detector in the edge computing server which located on the emergency communication vehicle. We also propose an adaptive mechanism to improve the performance efficiency of our system under the unstable network bandwidth. Using the adaptive mechanism, the image uploading speed is improved by around 21% $\sim$ 35%.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (61433019, U1435217) and The National Key Research and Development Program of China (2016YFB1000302). Fang Liu is the corresponding author.

REFERENCES

[1] National Research Council et al. 2007. *Improving disaster management: the role of IT in mitigation, preparedness, response, and recovery.* National Academies Press.
[2] Kevin Fall. 2003. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications.* ACM, 27–34.
[3] Matthew Faulkner, Robert Clayton, Thomas Heaton, K. Mani Chandy, Monica Kohler, Julian Bunn, Annie Liu, Annie Liu, Michael Olson, and Ming Hei Cheng. 2014. Community sense and response systems: your phone as quake detector. *Communications of the Acm* 57, 7 (2014), 66–75.
[4] K. A. Frenkel. 2010. Panning for science. *Science* 330, 6005 (2010), 748–749.
[5] Stephen M George, Wei Zhou, Harshavardhan Chenji, Myounggyu Won, Yong Oh Lee, Andria Pazarloglou, Radu Stoleru, and Prabir Barooah. 2010. DistressNet: a wireless ad hoc and sensor network architecture for situation management in disaster response. *IEEE Communications Magazine* 48, 3 (2010).
[6] Yun Chao Hu, Milan Patel, Dario Sabella, Nurit Sprecher, and Valerie Young. 2015. Mobile edge computingâĂŤA key technology towards 5G. *ETSI White Paper* 11, 11 (2015), 1–16.
[7] Sophia B Liu, Leysia Palen, Jeannette Sutton, Amanda Hughes, and Sarah Vieweg. 2008. In search of the bigger picture: The emergent role of on-line photo sharing in times of disaster. In *Proceedings of the information systems for crisis response and management conference (ISCRAM).* 969–980.
[8] B. S. Manoj and Alexandra Hubenko Baker. 2007. Communication challenges in emergency response. *Communications of the Acm* 50, 3 (2007), 51–53.
[9] Antti P. Miettinen and Jukka K. Nurminen. 2010. Energy efficiency of mobile clients in cloud computing. In *Usenix Conference on Hot Topics in Cloud Computing.* 4–4.
[10] Kohta Ohshima, Hiroshi Hara, Yoichi Hagiwara, and Matsuaki Terada. 2012. Field investigation of the radio transmission performance and distance in a environmental wireless sensor network. In *Information Networking (ICOIN), 2012 International Conference on.* IEEE, 132–137.
[11] Xue Liu Dan Feng Wen Xia Shunde Cao Jie Wu Yuanyuan Sun Yuncheng Guo Pengfei Zuo, Yu Hua. 2017. BEES: Bandwidth- and Energy- Efficient Image Sharing for Real-time Situation Awareness. In *in Proceedings of the 37th International Conference on Distributed Computing Systems (ICDCS).*
[12] Mahadev Satyanarayanan, Grace Lewis, Edwin Morris, Soumya Simanta, Jeff Boleng, and Kiryong Ha. 2013. The Role of Cloudlets in Hostile Environments. *IEEE Pervasive Computing* 12, 4 (2013), 40–49.
[13] Zhenhong Shao, Yongxiang Liu, Yi Wu, and Lianfeng Shen. 2011. A rapid and reliable disaster emergency mobile communication system via aerial ad hoc bs networks. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on.* IEEE, 1–4.
[14] George Thoma, Sameer Antani, Michael Gill, Glenn Pearson, and Leif Neve. 2012. People locator: A system for family reunification. *IT Professional* 14, 3 (2012), 13–21.
[15] Swagath Venkataramani, Victor Bahl, Xian-Sheng Hua, Jie Liu, Jin Li, Matthai Phillipose, Bodhi Priyantha, and Mohammed Shoaib. 2015. SAPPHIRE: An always-on context-aware computer vision system for portable devices. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015.* IEEE, 1491–1496.
[16] WonderShaper. [n. d.]. http://lartc.org/wondershaper/. ([n. d.]).
[17] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. 2016. Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters* 23, 10 (Oct 2016), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342
[18] Tan Zhang, Aakanksha Chowdhery, Paramvir Victor Bahl, Kyle Jamieson, and Suman Banerjee. 2015. The design and implementation of a wireless video surveillance system. In *Proceedings of the 21st Annual International Conference on Mobile Computing and Networking.* ACM, 426–438.