# REAL-TIME PEDESTRIAN DETECTION USING APACHE STORM IN A DISTRIBUTED ENVIRONMENT

Du-Hyun Hwang, Yoon-Ki Kim and Chang-Sung Jeong

Department of Electrical Engineering,
Korea University, Seoul, Republic of Korea

doohh88@korea.ac.kr
vardin@korea.ac.kr
csjeong@korea.ac.kr

## *ABSTRACT*

*In general, a distributed processing is not suitable for dealing with image data stream due to the network load problem caused by communications of frames. For this reason, image data stream processing has operated in just one node commonly. However, we need to process image data stream in a distributed environment in a big data era due to increase in quantity and quality of multimedia data. In this paper, we shall present a real-time pedestrian detection methodology in a distributed environment which processes image data stream in real-time on Apache Storm framework. It achieves sharp speed up by distributing frames onto several nodes called bolts, each of which processes different regions of image frames. Moreover, it can reduce the overhead caused by synchronization by computation bolts which returns only the processing results to the merging bolts.*

## *KEYWORDS*

*Distributed stream processing, Apache storm, Image Processing, Pedestrian Detection.*

## 1. INTRODUCTION

Recently, the data in IT industry has been dramatically increasing. Besides, the volume of the data is also increasing continuously. Especially, size of digital pictures and resolution of video is bigger than before. In addition to this, the sharp data increase for services in an era of Internet of Things(IoT) makes it difficult to process image data stream in real-time in just one node. Therefore it is essential to process large-scaled stream image data in a distributed environment.

In this paper, we propose a pedestrian detection methodology which is an efficient model for dealing with image data stream in the distributed environment. We use Apache Storm [1] for the implementation which is a distributed stream processing framework in real-time. Apache Storm runs topologies on a Storm cluster which consists of spouts and bolts. The spout is a streamer task which makes sequence of tuples that is a data model of Storm and bolts are tasks for processing jobs. A target of our distributed image stream processing is a pedestrian detection. The pedestrian detection is an important technique which can help to prevent many accidents in an automobile field and creates profit by analysing the number of customers on shops. Therefore we consider for

David C. Wyld et al. (Eds) : NETCOM, NCS, WiMoNe, CSEIT, SPM - 2015
pp. 211–218, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.51618

pedestrian detection and propose an efficient way for detecting pedestrians on the distribute environment.

Our methodology is has several advantages as follows: Firstly, it speeds up the processing by being operated in parallel on the each node by distributing frames onto several computation. Secondly, it can reduce the overall computation load by dividing the frame into several region to detect on the each frame. Finally, the overhead caused by synchronization can be reduced by returning only the processing result.

The outline of our paper is as follows: In section 2, we describe related works about Apache Storm framework and pedestrian detection algorithm which we selected. In section 3, we present a model for processing pedestrian detection efficiently on the distributed environment and explain a topology for a workflow running on the Storm cluster. In section 4, we explain our results of experiment.

## 2. RELATED WORKS

### 2.1. DISTRIBUTED STREAM PROCESSING

Recently, big data systems like Hadoop [2] have been used on various fields. Hadoop is populist platform on the big data systems. However, applications become more various, and users have needed to get process results more quickly. Therefore many stream process platforms appear to provide services which can process the big data in real-time.

Big data needs to process data in distributed and parallel environment because it is not structured like data saved in database. For this solution, Hadoop appear which can process big data using MapReduce [3] that is parallel processing framework. However, according to increase of data like RFID, twit and CCTV, batch systems like Hadoop have reached limit to process these large-scaled data. For resolving this problem, various techniques have been suggested to process it.

### 2.2. APACHE STORM

Apache Storm is a distributed stream data processing system, which has been used in Twitter for various critical computations. Apache Hadoop is an essential framework for distributed processing large-scaled data and already has been used in Hadoop ecosystem in many ways these days. However, it does not cover real-time stream processing. For this reason, Apache Storm appeared and has been a solution for real-time processing. It is possible to use on the Hadoop or alone with Zookeeper [4] that is nodes manager.

Apache Storm has various features. Firstly, it guarantees fault-tolerant and high availability. If errors happen on the process works, Storm reassign it immediately. Therefore, the works can be operated continuously. Secondly, its latency of process is short, because it does not save data and processes in real-time. Third, it is scalable. It can add additional nodes on the Storm cluster easily. Finally, it guarantees reliable. Every data can be processed without any loss.

Storm architecture is similar with Hadoop. It consists of one master node which is called Nimbus and one or a couple of worker nodes which are called Supervisor. In addition, Storm relies on Zookeeper for managing nodes in Storm cluster. Zookeeper gives information of supervisor's state to Nimbus. Then, Nimbus assign works to supervisors and each supervisor processes assigned tasks.

Storm's data model is unbounded sequence of tuples which is consist of a field and a value. The stream of tuples flows through topologies, which are directed graphs. The topology's vertices represent computations and the edges represent the data flow. The vertices divided into two type, Spout and Bolt. Spout is a supplier of stream, which reads tuples form sources and provides it to topologies. And bolt is processing unit. Every works of Storm is on the Bolt, and it emits the results of processing to other bolts.

### 2.3. PEDESTRIAN DETECTION

Pedestrian detection techniques are for finding people on the load. Mostly research in this filed has focused in finding people standing than sitting or lying, although it could be useful for saving a life in a disastrous situation. Anyway, this technique is useful in many ways like counting people shopping in shops and warning to people about dangers ahead. The applications of pedestrian detection technique are striking.

Detectors in OpenCV [5], open vision library, are representative among published detectors.

- The histogram of Oriented gradients (HOG) [6] (INRIA) – HOG detector of Dalal2005 [6], which is learned through INRIA Person Database. It is difficult to detect object of a template which is smaller size than 64(w)x128(h).
- HOG(Daimler) – HOG detect which is learned through Daimler Pedestrian Dataset. It could detect objects of a small template.
- Hogcascades – Detector which is applied cascade technique in HOG feature.
- Haarcascades – Detector of Viola2001 [7], whose speed to detect objects is fast that others.

Haarcascades was used as our detector for finding pedestrians, since our purpose of this paper is just detecting in distributed environment rather than focusing in accuracy of detection.

## 3. DISTRIBUTED PEDESTRIAN DETECTION MODEL

To detect pedestrians in the distributed environment, we propose a methodology and a workflow for the detections.

### 3.1. METHODOLOGY OF DETECTION

The frame rate of video is higher than before. The 30fps's CCTV translates 30 frames during 1 secondly. In this situation, the location of objects have changed little between a few continuous frames since the frame rate is too fast for pedestrians to move to other locations. Figure 1 shows that there are little changes of pedestrian's location between continuous frames of the video. We suggest the efficient methodology for detecting pedestrians in the distributed environment in real-time using the characteristic explained above.

(a) (b)

(c) (d)

Figure 1. The continuous frames of fps 15 video (a) Frame 1, (b) Frame 2, (c) Frame 3 and (d) Frame 4

### 3.1.1. DISTRIBUTING FRAMES PHASE

In distributing frames phase, the frames of the stream source are distributed onto several nodes for processing to detection in parallel. If the fps of image data stream is very high, sampling frames is needed for reducing workload. Since the locations of the pedestrians has little changed between continuous frames, sampling can reduce execute time.

### 3.1.2. DETECTING PEDESTRIAN PHASE

In detecting pedestrian phase, the pedestrian detection is operated in each node in parallel. As noted above, the detection does not need to be operated in every part of the frame since the locations of pedestrians have not changed between a few frames in case of image data stream of high fps. Therefore, each node operates to detect pedestrians on the different regions of the different frames to reduce the overall computation and processing time.

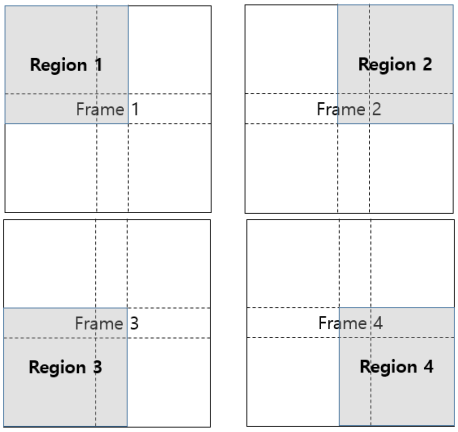



Figure 2. Detect regions of each frames

### 3.1.3. PROCESSING RESULTS TRANSMISSION PHASE

Each node which is operated for detection does not transmit the frames as the results, but only the information about the locations of pedestrians. For this method, a node for merging this results receives the original frames from the source separately. Transmitting only the information about the location of pedestrians can reduce network load caused by communications, since it can reduce the size of total traffic. Moreover, the total execute time can be reduced by removing a function of synchronization, since the original frames are transmitted separately. If the frames are transmitted as the results from the each node which process to detect pedestrians, the synchronization is needed for arranging the frames in order.

### 3.2. WORKFLOW

In this section, we explain the topology which is run on the Storm cluster. For reducing the network load and the works, features are delivered instead of frames. If the frames as the result of the detection are delivered, synchronization between the frames will be needed and happen network load for delivering every frame again.

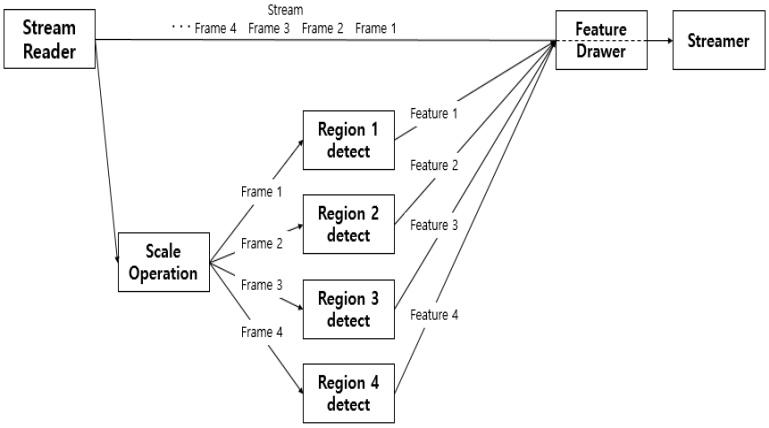



Figure 3. The topology of the detection

The workflows are as follows.

- **Spout:** The spout generates stream of sequential tuples from video sources. Video frames are serialized for delivering as tuples and emitted to Bolt for scale operation and feature drawer. Received tuples are deserialized to frames for processing image.
- **Scale Bolt**: It controls the volume level of the frame properly to reduce the load of each operations. However, if frames' volume level become too small, it comes difficult to operate the detection.
- **Detect Bolt:** Its operation is to detect pedestrians in parallel. Each bolt receives the frames in sequence and detects the pedestrians in particular area. For example, detect bolt 1 receives the frame 1 and finds pedestrians in some part of the frame. Detect bolt 2 receives the frame 2 and find pedestrians as well but in another part of the frame 2. If pedestrians are detected, then the bolt emits the features to a next bolts, which are information of the detection result.
- **Feature Drawer:** feature drawer bolt receives features from the detect bolts, which are information about detected pedestrians. At the same time, it receives frames directly from the spout simultaneously. Then, it draws the features on the received frame.
- **Streamer:** Streamer bolt emits the tuple stream of the result as mjpeg format to web service to watch the result of processing.

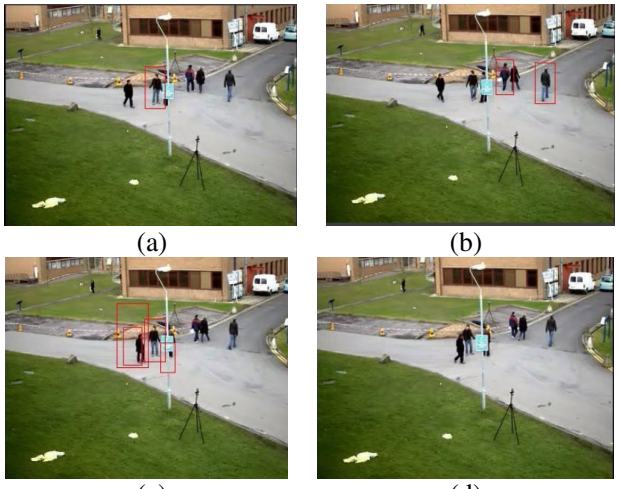
## 4. IMPLEMENTATION

### 4.1. EXPERIMENT ENVIRONMENT

Our system described above is implemented on a laptop with an Intel® Celeron® CPU B800 @ 1.50GHz processor running Window8. Program is developed by Java using Eclipse Luna using Storm framework. The library is used the OpenCV. Distributed environment is composed by Oracle VM VirtualBox. Three virtual machine are generated and run Linux Ubuntu 14.04 64bit. Apache-Storm 0.9.5 and Zookeeper 3.4.6 is installed for running Storm.

## 4.2. DETECTION RESULT

Figure 4 shows the result of pedestrian detection. Since the location of pedestrians is little changed between frames, it is possible to detect every pedestrians separately.



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

Figure 4. Detection pedestrians on each part of sampled frames. (a) Upper left, (b) Upper right, (c) Bottom left, (d) Bottom right

## 4.3. PERFORMANCE EVALUATION

In this section, we explain about the evaluation of our experiment.

Figure 5 shows overall information of Topology which operates pedestrian detection and the data flows. Table 1 and Table 2 show detail information of spout and bolts. Executors are threads in a worker process. Pedestrian bolt which detects pedestrians in a frame has 4 executors for reducing execute latencies.

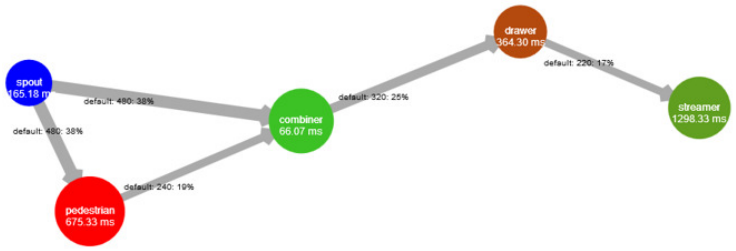



Figure 5. Topology stats visualized

| Spout Id | Executors | Tasks | Node | Emitted | Transferred | Complete latency (ms) | Acked | Failed |
|---|---|---|---|---|---|---|---|---|
| spout | 1 | 1 | N3 | 380 | 760 | 2166.4 | 400 | 100 |

Table 1. Spouts stats

| Bolt Id | Executors | Tasks | Node | Emitted | Transferred | Capacity (last 10m) | Executed latency | Executed | Process latency (ms) | Acked | Failed |
|---|---|---|---|---|---|---|---|---|---|---|---|
| combiner | 1 | 1 | N3 | 380 | 380 | 0.231 | 28.333 | 960 | 43.469 | 980 | 0 |
| drawer | 1 | 1 | N1 | 340 | 340 | 1 | 329 | 380 | 256.211 | 380 | 0 |
| pedestrian | 4 | 4 | N1, N2, N3 | 520 | 520 | 0.9 | 592.957 | 460 | 531.083 | 480 | 0 |
| streamer | 1 | 1 | N1 | 500 | 0 | 0.247 | 80.684 | 380 | 1334.842 | 380 | 0 |

Table 2. Bolts stats

## 5. CONCLUSIONS AND FUTURE WORKS

In this paper, we have presented the pedestrian detection methodology in the distributed environment using Apache Storm framework. Actually, the image processing in the distributed environment is not appropriate due to network load problem caused by communications between nodes for translating frames. However, we cannot but use distributed processing since the only one node cannot cope with processing high resolution and high fps image data stream in real-time. To conclude, we have suggested the methodology for detecting pedestrian in real-time in the distributed environment to reduce works as distributing the frames and dividing the region for detection into several parts. Besides, it reduces the overhead cause by synchronization by returning the only processing result. In the future, stream image processing in the distributed environment will be an essential technique and need research to reduce network load problem between nodes.

## ACKNOWLEDGEMENTS

This research was supported by Korea university and MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2015-H8501-15-1004) supervised by the IITP(Institute for Information & communications Technology Promotion)

## REFERENCES

[1] Toshniwal, Ankit, et al. "Storm@ twitter." Proceedings of the 2014 ACM SIGMOD international conference on Management of data. ACM, 2014.

[2] Shvachko, Konstantin, et al. "The hadoop distributed file system." Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on. IEEE, 2010

[3] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[4] Hunt, Patrick, et al. "ZooKeeper: Wait-free Coordination for Internet-scale Systems." USENIX Annual Technical Conference. Vol. 8. 2010.

[5] Bradski, Gary. "OpenCV." Dr. Dobb's Journal of Software Tools (2000).

[6] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on. Vol. 1. IEEE, 2005.

[7] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on. Vol. 1. IEEE, 2001.

[8] Kothiya, Shraddha V., and Kinjal B. Mistree. "A review on real time object tracking in video sequences." Electrical, Electronics, Signals, Communication and Optimization (EESCO), 2015 International Conference on. IEEE, 2015.

[9] Im, Dong-Hyuck, Cheol-Hye Cho, and IlGu Jung. "Detecting a large number of objects in real-time using apache storm." Information and Communication Technology Convergence (ICTC), 2014 International Conference on. IEEE, 2014.

[10] Benenson, R., Mathias, M., Timofte, R., & Van Gool, L. (2012, June). Pedestrian detection at 100 frames per second. In Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on (pp. 2903-2910). IEEE.

## AUTHORS

**Du-Hyun Hwang** is currently working towards a master's degree at Department of Electrical Engineering, Korea University. His current research interests are distributed parallel computing, computer vision and GPU processing.

**Yoon-Ki Kim** is currently working toward the PhD degree in Electronic and Computer Engineering at the Korea University. His research interests include real-time distributed and parallel data processing, IoT, Sensor processing and computer vision.

**Chang-Sung Jeong** is a professor at the department of EE/CE at Korea University. He received his MS.(1985) and Ph.D.(1987) from North western University, and B.S.(1981) from Seoul National University. Before joining Korea University, he was a professor at POSTECH during 1982-1992. He also worked as an associate researcher at UCSC during 1998-1999.