



Undergraduate Project Report 2015/16

[Video Synopsis and Indexing System Based on Spark]

Name:	[Li Zijian]
Programme:	[Internet of Things]
Class:	[2012215118]
QM Student No.	[120726859]
BUPT Student No.	[2012213313]
Project No.	[IC_3313]

Date [2016/5/14]

Table of Contents

Abstract.....	3
Chapter 1: Introduction	5
1.1 Introduction of the project	5
1.1.1 Motivation.....	5
1.1.2 Functionalities realised	5
1.1.3 Technical context	6
1.2 Structure of report	7
Chapter 2: Background	8
2.1 Video synopsis technology	8
2.1.1 JavaCV and OpenCV	8
2.1.2 Motion Detection Algorithm.....	8
2.1.3 Video Writer and MPEG-4.....	10
2.2 Apache Spark.....	10
Chapter 3: Design and Implementation.....	14
3.1 Design	14
System overview	15
3.1.1 Design of video synopsis algorithm.....	15
3.1.2 Design of Spark cluster	18
3.1.3 Design of Web pages.....	19
3.2 Implementation	20
3.2.1 Algorithm	20
3.2.2 Spark cluster.....	23
3.2.3 Web pages	28
Chapter 4: Results and Discussion	32
Chapter 5: Conclusion and Further Work	38
5.1 Conclusion.....	38
5.2 Further work	39
References	40
Acknowledgement	41
Appendix.....	42
Risk Assessment.....	49
Environmental Impact Assessment	50

Abstract

Surveillance videos are big data, and people are usually interested in the moving targets of them. Since surveillance cameras often record with fixed scenes as backgrounds, the generated videos contain long period of clips without moving targets. Finding moving targets in such video clips will be manpower and time consuming. This project focuses on implementing a system using video synopsis algorithm to recombine the moving targets in a long surveillance video into a short video clip containing all moving targets.

The project proposes a system that can be divided into two parts: the frontend web pages and the backstage processing algorithm based on Apache Spark. In the web pages, users can upload as well as download video clip. This project designs and implements a synopsis algorithm in the backstage which uses background subtraction to grab all the frames containing moving targets in a video and then concatenate all the frames into a new shorter clip. The algorithm connects to Spark cluster. After users upload a video to be condensed on web page, the algorithm will process it and return success information to users after generating the new video. Then users can download the new video from the download page.

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

摘要 (Chinese translation of the Abstract)

监控视频是一类典型的大数据，人们只对监控视频中的移动目标感兴趣。但是，由于监控摄像头往往拍摄的是固定场景下的视频，所以监控视频中的大量片段不包括移动目标。从监控视频中寻找感兴趣的移动目标需要消耗大量的人力物力。所以本项目致力于实现一个系统，该系统可以将一段冗长视频中的移动目标提取出来，并且浓缩成一段简短的，只含有动态目标的片段。

系统可以分为前端页面和后台处理程序两部分，用户可以从前端页面上传下载视频，后台基于 spark 的浓缩算法来对视频进行浓缩。用户上传视频后，后台算法利用背景差分提取含有动态目标的帧，然后将提取出来的帧浓缩成新的，简短的视频。处理完成后会向用户返回信息，然后用户可以从下载页面获取下载链接。

Chapter 1: Introduction

1.1 Introduction of the project

1.1.1 Motivation

This final year project is implemented to condense a long video clip into a shorter one which only contains frames with moving targets. Surveillance cameras are widely used, especially in monitoring traffic condition. Users are usually interested in moving targets when checking traffic conditions or dealing with an accident. Yet when it comes to traffic condition at night or in low-load period of transportation, the video clips contain few moving targets in sporadic period. It will cost large amount of time and manpower to find moving targets in such video clips. Thus this project is to design and implement a web based system that enable users to upload a surveillance video to be processed, and using video surveillance algorithm connecting with Spark cluster to condense the video clip and generate a shorter one for user to download. Apache Spark is a lightning fast distributed computing framework which can process large files efficiently.

1.1.2 Functionalities realised

The system is basically divided into two major parts: the web pages interacting with users, as well as the program mapping users' requests and responding to the requests. Figure one shows the flow chart of the whole project, which contains the main functionalities of the system:

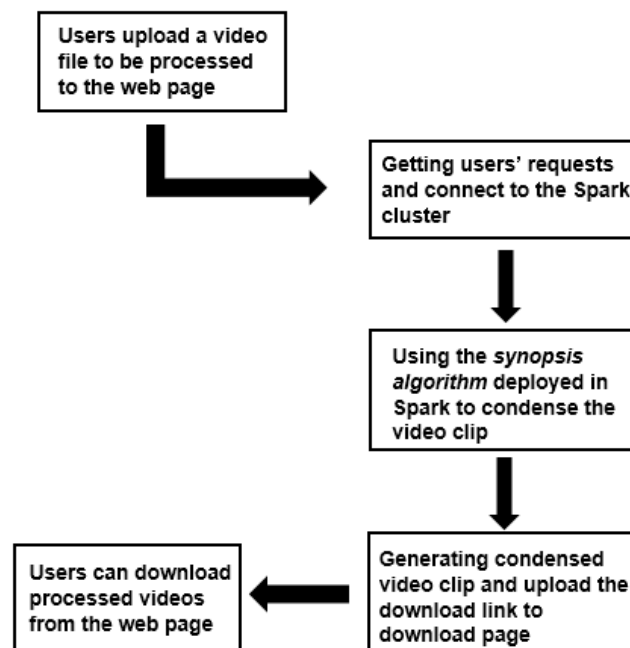


Figure 1 Flow chart of the system

As shown in the flow chart, the functionalities to be realised include: web pages from which users can upload and download a video file; a Spark cluster for the system to connect; the backstage program which can respond to users requests, connect to Spark cluster, and condense the video clips using video synopsis algorithm.

Clearly, the key part of the system is to implement the functionalities of the backstage programs. The programs use JavaCV libraries to process the video files, generate intermediate results of the processing video (the frames contains moving targets as images) and generate the final condensed video clip for users to download. Additionally, the project uses the bootstrap framework to establish the webpage and basic Spring MVC framework to map and respond to users' request.

As for the Spark cluster, the system uses the Standalone mode to establish it. The whole cluster contains three virtual machines, one master, aka cluster manager, as well as two workers.

All the technical details will be thoroughly illustrated in chapter 2 and chapter 3.

1.1.3 Technical context

The programming language of this project is JAVA and the local video synopsis system is established on windows 7 system, while the Spark cluster consists of three Ubuntu virtual machines.

Hardware context:

Local:

Operating System (OS): Windows 7 Ultimate

OS type: 64-bit operating system

RAM: 4 GB

Spark Cluster:

Operating System: Ubuntu 15.10

OS type: 64-bit operating system

RAM: 1 GB

Programming software:

IntelliJ Idea 15 Ultimate

1.2 Structure of report

The first part of the final report is the abstract, which concisely illustrates the purpose and the objective of the project;

Chapter 1 is the introduction that introduces the architecture of the system and the structure of this report. The introduction concludes the motivation and the function of the system, as well as technique environment where the system is implemented;

Chapter 2 is background of the whole report. This chapter will detailedly state relevant background information about video processing, synopsis algorithm and Spark cluster;

Chapter 3 is the key part of the report, design and implementation. This chapter illustrates the details about designing the video synopsis system, and the techniques to implement the algorithm, the web page and the Spark cluster;

Chapter 4 is results and discussion in which I will show the results generated by the system and discuss them;

Chapter 5 is conclusion and further work that states and concludes the project and the next step of the project.

Chapter 2: Background

2.1 Video synopsis technology

The whole process of the video synopsis technology can be divided into two parts: Firstly use motion detection algorithm to detect and extract the moving targets, and then recombine all the moving targets into a new video clip. Generally, JavaCV and OpenCV are two main library subtracting and recombining video files. Since the programming language of this project is Java, JavaCV is the library used to process video files.

2.1.1 JavaCV and OpenCV

OpenCV standing for open source computer vision, it is a library of programming functions aimed at computer vision. OpenCV is a cross-platform and free library for both academic and commercial use under the open-source BSD license. ^[1] OpenCV has realised lots of algorithms that can be used to resolve problems in motion tracking, facial recognition, human-computer interaction, mobile robotics and augmented reality. The basic skills to accomplish these function are to acquire, analyse, process and understand images as well as high-dimensional data to produce computable numerical or symbolic data. However, OpenCV is developed for C and C++ which are not capable to be used for Spark. Fortunately, OpenCV has some wrappers in other programming languages.

JavaCV is one of the wrappers that is used in Java computer vision. It is a Java interface of OpenCV, thus when you use JavaCV, most of the methods are from OpenCV. But JavaCV provides more than just one-to-one wrapper from OpenCV. In fact, it bundles the whole number of image processing libraries, including FFmpeg, Open Kinect and others. ^[2]

To put it in a nutshell, it does not matter using library from OpenCV and JavaCV, they usually generate same result. It depends more on whether it is Java or C++ generating better performance in the project.

2.1.2 Motion Detection Algorithm

Motion detection is the first and essential process to extract information of moving targets, contributing to tracking objects, recognition and so on. The widely and frequently used methods of motion detection include frame difference method and background subtraction method.

1) Frame Difference method

Frame difference method is a simple and easy method. It compares every pixels in the current frame

with the ones in the previous frame to extract moving targets to the difference image. This method use the current frame to subtract the previous one, and the unchanged part is eliminated and the different part is remained. This change can be caused by movement or noise, thus there is the necessity of a binary process to distinguish the noise, which is assumed as Gaussian white noise in the binary process. ^[3] Setting $I_j(x, y)$ as the current frame, $I_{j-1}(x, y)$ as the previous frame, and TH as the predefined threshold of the noise, then the difference image:

$$D_j(x, y) = I_j(x, y) - I_{j-1}(x, y) \quad (1)$$

Using binary process, the formula (2) is to decide whether one pixel is noise or foreground (pixels of moving objects) ^[4], set $F(x, y)$ as the foreground:

$$F(x, y) = \begin{cases} 1 & D(x, y) \geq TH \\ 0 & D(x, y) < TH \end{cases} \quad (2)$$

This method compares every binary processed pixels of the difference image with the noise threshold, if it is larger than the threshold, it will be set as foreground.

Although this method is simple and fast, it is very sensitive to the threshold and may consider one moving object as some different objects. Thus, this project use another method to detect moving objects, background subtraction.

2) Background Subtraction method

Background subtraction algorithm is a process to obtain foreground moving targets in a particular scene ^[5], so it is very suitable to be used in processing surveillance videos. The algorithm needs to set a background model in a frame, using current frame to subtract the background frame to extract the moving objects. The background model will be updated during the period processing the video file, which means after loading each frame in the model, the method will compare it with the former background image and generate an improved background. So this method can reduce the impact of environment change. For example, the leaves swinging with the wind will be considered as background rather than moving targets.

The algorithm uses Mixture of Gaussians (MOG) background subtraction model for motion detection due to its capability to extract background with the existing of moving object and it has good performance under complex environment. Mixture of Gaussians model is a probabilistic model which is generated from K (normally 3 to 5) Gaussian distribution, aka Normal distribution. It can be seen as improved from Single-Gaussian model. The formula of Gaussian distribution is:

$$P(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \quad (3)$$

In which x means a pixel, and $p(x)$ means its probability to appear, μ is the mean of all the pixels and σ^2 is the variance.

When using Single-Gaussian background model, all the image will be binary processed, and the pixels of the background should lie near the centre of the distribution.

If a pixel is near the centre of the background model, it will be seen as background point. Otherwise it is the foreground point (moving object).

The Mixture of Gaussian model consists finite number of Single-Gaussian distribution model, which means every time the model gets a new frame of a video clip, it will compare pixels of this frame to the background model to see whether a pixel is near one of the centres in each Gaussian Distribution. Then it decides whether the pixel is background point or not and update the Mixture of Gaussian model to get a new background.

2.1.3 Video Writer and MPEG-4

Recombining all the grabbed images containing moving targets to a new video clip is a simple process in the whole procedure. As mentioned in 2.1.1, JavaCV has library to create a video writer which can concatenate all the images as frames into a video clip.

MPEG-4 is a method to define the compression of audio and visual digital data. ^[6] I choose MPEG-4 method to compress the video clip since it absorbs many of the features of MPEG-1 and MPEG-2 and other related standards, and the generated video clip is not too large.

2.2 Apache Spark

Apache Spark is a fast and general engine for large-scale data processing. ^[7] Apache Spark provides API in many programming language like Scala, Java and Python. It is a fast distributed computing framework based on in-memory computation, which is flexible and efficient. Spark retains fault tolerance and scalability of Map Reduce, and even faster.

Spark generally can be two things, the Spark core and the Spark project such as Spark Streaming and Spark SQL based on the core. Meanwhile, Spark can run on multiple platform, Hadoop, Standalone, Mesos or even on cloud like Amazon EC2.

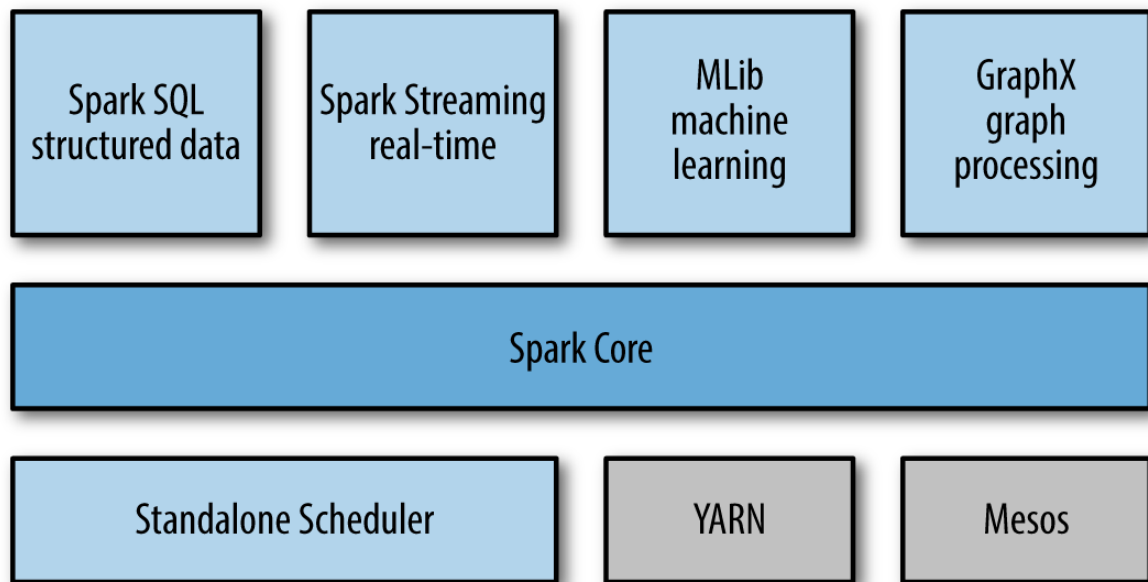


Figure 2 Spark ecosystem ^[8]

As shown in figure 2, Spark core includes the fundamental functionalities including task scheduling and memory management and the technologies of Spark this project used are all from Spark core.

A Spark application can run on both local and cluster.

When Spark runs in local mode, it is deployed on a single JVM (Java Virtual Machine). Using Spark's built-in Standalone scheduler in local mode can make the driver, executors and master in the same JVM. The local mode uses a number of threads, which is allocated in master URL (explain later), to do parallel computation.

Master URL in local mode can use the *local*, *local[n]* or *local[*]*: *local* means using only one thread, *local[n]* means using n threads, and *local[*]* means using as many threads as the processor can.^[9]

Spark running in local mode performs well in debugging, testing and development. It is also convenient since no advanced set up launching the application is needed. Yet local mode is used generally for testing, and running on a single machine cannot reveal Spark's distributed computation capability, this project use Spark running on cluster mode.

A Spark application uses master/worker architecture to work on cluster. The cluster can be deployed on Hadoop Yarn mode, Mesos mode and Standalone mode. Each Spark application has a driver program, talking to the coordinator, master, to manage workers. The driver program has a SparkContext (called JavaSparkContext in Java) instance, which is the start of spark execution. SparkContext is a vital client of Spark execution and can ask cluster manager (aka master in

Standalone mode) to allocate resources such as CPU and memory to worker node. In each worker, there is a process called executor that runs computation and store data. ^[10] After cluster manager allocates resources, the driver program sends executors the code to run the tasks.

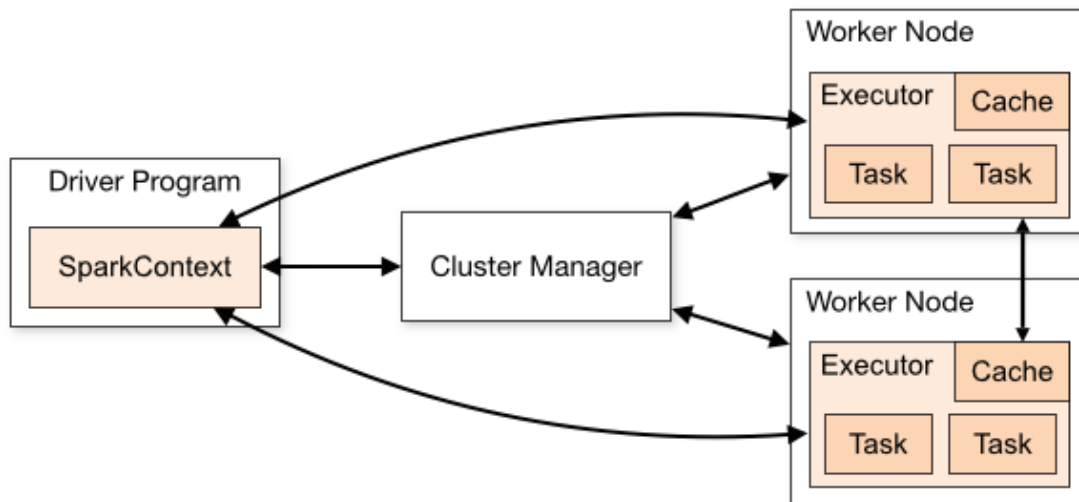


Figure 3 Spark working Architecture ^[10]

Spark has three types of cluster manager: Hadoop YARN, Mesos, and Spark built-in Standalone, thus it can be deployed on three modes.

Hadoop YARN (Yet Another Resource Negotiator): This mode uses hadoop YARN as cluster manger, and uses YARN components for computation. Its major components include:

Resource Manager: One per cluster. It runs as the master daemon and manages Application Master and Node Manager.

Node Manager: One or more in one cluster. It is used to offer resources and sends heartbeats message to Resource Manager.

Application Master: One per job (task). It monitors tasks, and negotiates with Resource Manager for resource container to run jobs, as well as coordinates all the tasks.

YARN mode can use hadoop to store and process large amount of data, and runs the ResourceManager on the master. When the application starts, it negotiates with the ResourceManager and the ResourceManager will make a container request, after which the ApplicationMaster starts to run the tasks. ^[11]

Apache Mesos: Mesos is a general manager, which is good at scalable partitioning between multiple

Spark instances.

Both Hadoop Yarn mode and Apache Mesos mode require third party to run the Spark application and they both need high configuration of hardware, which is hard for my laptop to realize. Hence, I choose to use the Spark Standalone mode to deploy the cluster.

Spark Standalone cluster manager is built in Spark so it is simple and convenient to set it up. In Standalone mode, the cluster manager is also the master of the whole cluster, and you can connect with it by the *spark: //master URL* after the cluster has been started.^[9] When you set the Spark cluster in Standalone mode, the Cluster Manager will be selected as you set after the cluster is started. The driver program can connect to the master with its URL, and the master will send tasks to the workers to run.

Spark Standalone cluster has two modes to be deployed: the client mode and the cluster mode. In former mode, the driver launched in the same process as the clients which submits the application; in the latter one, driver is launched from one of the worker nodes. I used the client mode in my project.

Plus, Standalone mode can restart the application when application exit with non-zero exit code.^[12]

The details about how to set the Spark cluster in Standalone mode will be illustrated in Chapter 3.

Chapter 3: Design and Implementation

This chapter will detailedly illustrate the procedure designing and implementing the whole system. First chapter 3 will propose the whole architecture of the system and then explain them in parts.

3.1 Design

System overview

Familiarizing the specification is the first and essential step before designing the video synopsis and indexing system.

First of all, this project is based on web, hence some concise and user-friendly web pages are needed. Meanwhile, there should be a backstage program which can respond to users request from web pages and it definitely should be able to process video files and condense them into short clips which can index back to the content of original video files. The video synopsis algorithm is no doubt the core of the backstage program and the whole system

As mentioned in the first Chapter, the backstage program needs to connect to the Spark cluster, leading to the necessity of designing and implementing a Spark cluster on the laptop for the whole system as well. Figure 4 shows the frame diagram designed for the project:

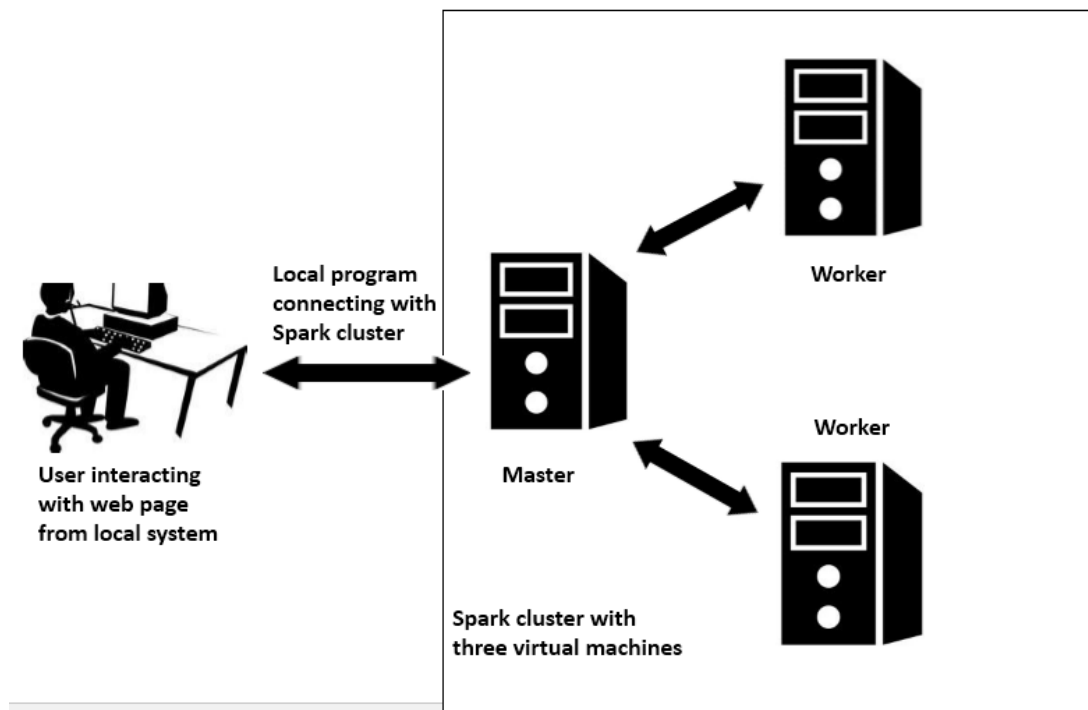


Figure 4 Frame diagram of the project

Users can interact with the web pages, and the backstage program will respond the request from the web page and connect to the cluster for processing the files users upload. The following part of chapter 3.1 will dividedly illustrates each part of the system.

3.1.1 Design of video synopsis algorithm

Figure 5 is the flow chart of the video synopsis algorithm, and the flow chart will be detailedly illustrated under the figure.

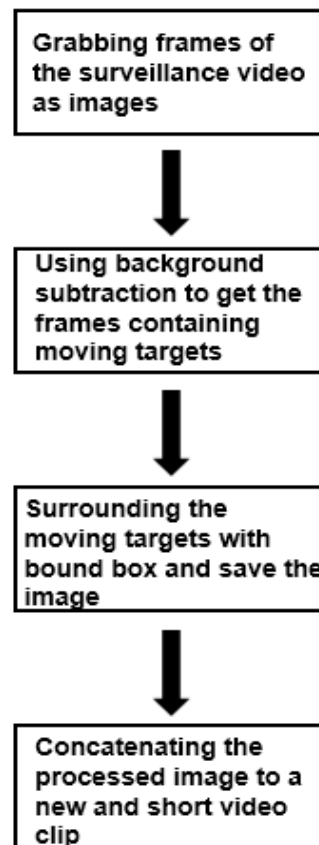


Figure 5 Flow chart of synopsis algorithm

The video files this project dealing with are all surveillance videos, and each of the surveillance cameras normally records videos from a fixed scene. All the objects that are not parts of the fixed scene can be considered as moving objects in the video clips. Since users' interests focus on moving targets in the surveillance videos, the algorithm obviously needs to extract objects which the fixed scene does not consist of. Hence, the fixed scene recorded by the surveillance camera can be regarded as the background of the video clip, which users have no interest in, and all the moving objects not belonging to the fixed scene can be regarded as foreground.

The question is how to extract moving targets and the background scene from a video clip. As known, all the video files are constituted by large amounts of frames. Each of the frames is just a still picture. Chapter 2 has already illustrated that the motion detection methods processing video files are in fact dealing with each of the frames in a video file. Hence the algorithm has to divide the video file into pieces of frames at first, sending the frames to motion detection method for processing.

After dividing the video file into frames, the algorithm should process the frame in sequence to extract the moving objects. As said in Chapter 2, the algorithm uses background subtraction to separate the foreground moving targets and the background scene. It uses Mixture of Gaussian model to do the background subtraction due to its good performance dealing with environment change. Even a fixed scene will not be the same all the time. For example, the sunlight condition may change, so it can be lighter or darker; the leaves may swing with the wind. Mixture of Gaussian model perform well dealing with such changes. After set the background model, the background will be improved and updated every time loading a new frame of the video. When there is a background model, the fastest and most convenient way to get moving objects is to use the new frame of the video to subtract the background frame. This step works just like the frame differentiate method in chapter 2. There is also a threshold to decide whether the change from the background is cause by noise or movement.

After subtract the background from a frame and make sure there is a moving object, the algorithm generates foreground masks of the moving objects. A foreground mask is just a black and white version of the original foreground. Each pixel has value 0 or 1. Figure 5 is a comparison between input image and the foreground mask.

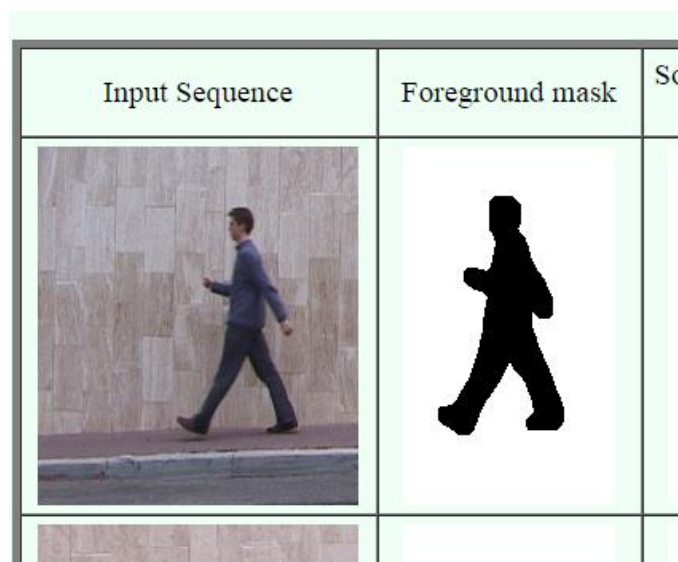


Figure 6 Foreground mask ^[13]

Then the algorithm needs to find the contour of the foreground mask for further process. Finding contour of an object is to find the coordinate of its pixels on the original frame. It is not a complex process since JavaCV has method to find the contour.

After find the contour of the foreground masks, the coordinates of them is got and it is time to think about how to do the next step. Clearly just keeping the foreground mask is not workable since it is just a blur shape of objects. Since the algorithm has already got the coordinate of the foreground mask in the original unprocessed frame, it has got the coordinate of the original moving objects in the frame. And in the first step the algorithm has already had the frame from the video. Hence, the moving objects can be surrounded by a rectangular bound box to be emphasized in the original image using their coordinates. Since some moving object are too far away from the surveillance camera and are too small in the video, and users are not interested in such small object. The algorithm only keeps the grabbed images containing moving objects that are large enough. Whether an object is large enough is decided by the area of the rectangle surrounding them.

All in all, the part of algorithm to get moving targets is to divide the video clip into frames, mark the moving targets on the frame, and save the frames containing moving objects users interested in as the intermediate results of the whole video synopsis algorithm. The following is the flow path of it:

Table 1 Algorithm to get moving targets

Input	Original video <i>videoName</i> Minimum size of object contour <i>minSize</i>
Output	Processed frames of original video <i>grabbedImage</i>
Begin:	
1:	Using Mixture of Gaussian model to set background model
2:	Grabbing a frame <i>grabbedImage</i> from <i>videoname</i>
3:	While <i>grabbedImage</i> != null do
4:	Using frame differentiate method between current frame and the background model
5:	if moving targets exist then do
6	Find the contour of the moving targets

```
7:          Calculate the area of the contour contourSize
8:          if contourSize ≥ minSize then do
9:              Surround the moving target in the original frame grabbedImage and save it
10:         end if
11:     end if
12     grabbedImage = next frame in videoName
13: end
```

After saving all the frames with moving targets users interested in, the next step is to recombine all the saved images into a new video clip. This step can be accomplished conveniently since JavaCV has method to concatenate images into a video. The process to convert images into a video clip basically is to adding images together one by one, each images being one frame of the video file. One thing should be paid attention to is that the all the frames must have the same height and width so that they can be concatenated into a video. Hence the algorithm needs to set the height and width of the video to make it suitable for adding images having been saved. The background image is saved as well in the step that generates frames containing satisfied moving targets. And the size of background image is the same as all the frames having moving object. Thus, the algorithm uses the background image to set the height and width of the video, and then read all the saved images from the last step, concatenate them into a new video file. To make the video clip go faster, the algorithm increases the fps (frame per second). At last, the algorithm will save the new video clips to given path.

This method also solves the problem of indexing back to the original video. Since all the surveillance videos show the date and the time when every frames are recorded, thus the time of each frames in the condensed video clip can be seen as well. When users saw an accident or something else they need full details, they can go back to this part in the original video according to the time.

At the same time, some fault tolerance mechanism is of vital importance, for example when the video files does not exists or the video name is null, the program should be able to return the problem.

3.1.2 Design of Spark cluster

As said in chapter 2, the system uses standalone mode to deploy the Spark cluster since it does not need third party to establish the cluster, while Hadoop YARN and Mesos mode have high requirement to hardware.

Building a Spark cluster needs master node and worker nodes. Since this project is done in one laptop, it uses three Ubuntu virtual machines to build the system, one master and two workers. Meanwhile it uses the driver program in local Windows system to connect the cluster and send tasks.

To set the master and workers architecture in Spark, and make them capable to process video files, it is necessary to install Scala and Spark on the virtual machine. Additionally it is needed to install OpenCV on the cluster to make them capable to process videos. Since Spark needs hadoop environment to run, both local windows and Ubuntu virtual machine have to install hadoop environment.

To make the master and workers connect and communication conveniently, it is necessary to modify the IP addresses on the three virtual machines. The IP addresses of the three virtual machines are entered in the */etc/host* file of the virtual machines for them to communication. What's more, the host names of the three virtual machines are changed for convenience. Setting up a cluster needs to modify the *conf/slaves* file in the Spark root directory. Entering the name of the slave host, it will map to the IP address in the */etc/host* file.

Of course, it is necessary to modify the original video synopsis algorithm into the driver program to make it able to run on Spark. As said in chapter 2, I need to create a *sparkConf* instance to set the configuration of the Spark configurations, including the name of the Spark application, the master URL for the driver program to connect, and the one of vital importance, the instance *sparkContext* to start the application. Additionally, since Spark is on cluster mode, the jar file should be deployed in the master machine earlier than start the application. The jar file is for the master to send tasks to all the other worker nodes.

After all the configuration Spark cluster have been set, I just need to start the application, and the driver program will connect to the Spark cluster, then call the video synopsis algorithm to run.

Since process video files on the cluster is a long period, the algorithm uses the logger to output the progress of the whole running system in order to monitor the process locally.

3.1.3 Design of Web pages

Since the core part of this project is the algorithm to process video, the web page just needs to be concise and easy for users to use.

As mentioned before, the functionality of the web pages is for users to upload a video file and download the processed file. Therefore, I decided to design two web pages, one is for users to upload

a video file they want to be processed, and the other one is for users to download the video file.

The upload page should let users to choose video files from local to upload; and the download page should show the name of the processed videos, the date and time when processing the video, which for users to know information about specific video, and it should also provide the download link that users can use to download condensed video clip.

To make the web page more appealing, Twitter's Bootstrap front-end framework is used to develop the basic layout of the web page.

To connect the web page with the backstage program, I use simple Spring MVC (model-view-controller) model making the program responding to users' request. The model is the classes written before to realize video synopsis and some classes for fundamental process like accessing the video path or generating the download link. The controller is to map and respond all the request from the web page and the view is obviously the web pages.

3.2 Implementation

3.2.1 Algorithm

IntelliJ Idea 15 is the software to develop the system. As mentioned in 3.1.1, the first part of the algorithm is to do the background subtraction and extract the moving targets. When doing background subtraction, the first step is to set the Mixture of Gaussian background model.

```
BackgroundSubtractorMOG2 mog = new BackgroundSubtractorMOG2(30, 40, false);
```

The code above is how to set the model. 30 is the length of history frames, 40 is threshold on the squared Mahalanobis distance to decide whether it is well described by the background model ^[15], and false means does not enable shadow detection. The length of history frames means how many frames the model need to compare to set the background.

After creating the instance of background model, the next step is to divide the video files into frames

```
FrameGrabber grabber = new OpenCVFrameGrabber(file);
```

```
try {  
    grabbedImage = grabber.grab();  
} catch (Exception e) {  
    closeGrabber(grabber);  
}
```

```
e.printStackTrace();  
break;  
}
```

The *FrameGrabber* is a Class provided by JavaCV, and the method *FrameGrabber.grab()* can grab the next frame in the video file every time the program calls it.

The next step is to process the frame and generate the foreground mask, and then find the contour of the mask.

```
mog.apply(new Mat(grabbedImage), new Mat(foreground), -1); //Computes a foreground mask,  
output binary image  
opencv_imgproc.cvDilate(foreground, foreground, null, 5);  
opencv_imgproc.cvErode(foreground, foreground, null, 6);  
opencv_imgproc.cvSmooth(foreground, foreground, opencv_imgproc.CV_MEDIAN, 3, 3, 2, 2);  
opencv_imgproc.cvFindContours(foreground, storage, contour,  
    Loader.sizeof(CvContour.class), opencv_imgproc.CV_RETR_LIST,  
    opencv_imgproc.CV_CHAIN_APPROX_SIMPLE);
```

As in the comment, the first line is to compare between the background model and the *grabbedImage* to compute a foreground mask into *foreground*. Next step is to dilate, erode and smooth the mask to make its contour clearer.

The last step to extract the moving targets is to surround the moving objects with bound box and save the processed image as intermediate results:

```
if (boundBox.width() * boundBox.height() > 900) {  
    _log.info("x : " + boundBox.x() + " ~ " + (boundBox.x() + boundBox.width() + 1) + "y : " +  
        boundBox.y() + " ~ " + (boundBox.y() + boundBox.height() + 1));  
  
    opencv_core.cvRectangle(grabbedImage,  
        opencv_core.cvPoint(boundBox.x(), boundBox.y()),  
        opencv_core.cvPoint(boundBox.x() + boundBox.width(), boundBox.y() +  
boundBox.height()),  
        opencv_core.CV_RGB(255, 0, 0), 1, 8, 0);
```

```
CvFont font = new CvFont();
opencv_core.cvPutText(grabbedImage, "0",
    opencv_core.cvPoint(boundingBox.x(), boundingBox.y()), font,
opencv_core.CvScalar.RED);
_log.debug("save picture into ..." + imagePath.concat("/").concat(counter + ".jpg"));
opencv_highgui.cvSaveImage(imagePath.concat("/").concat(counter + ".jpg"),
grabbedImage);
    counter++;
}
```

900 is the minimum area of the moving targets in the video since some moving object is too far from the surveillance camera, which users have no interest in.

After saving the intermediate images, the next step is to concatenate all the saved image into a new video clip.

```
try {
    _log.info("Start read background jpg...");
    bufferedImage = ImageIO.read(new File(backgroundPath + "background.jpg"));
    _log.info("End read background jpg...");
} catch (IOException e) {
    e.printStackTrace();
    return null;
}
int width = bufferedImage.getWidth();
int height = bufferedImage.getHeight();
CvVideoWriter vw = opencv_highgui.cvCreateVideoWriter(resultPath,
    opencv_highgui.CV_FOURCC((byte) 'D', (byte) 'I', (byte) 'V', (byte) 'X'), fps,
opencv_core.cvSize(width, height));
```

As mentioned in 3.1, CvVideoWriter is the class to write video, and the algorithm uses the height and width of background image as the height and width of the new video clip. 'D', 'I', 'V', 'X' stands for MPEG-4 coding form.

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

Then I need to add the saved images one by one to write the condensed video clip.

```
for (int index = 1; index < listOfFiles.length; index++) {  
    BufferedImage screen = getImage(index);  
    BufferedImage bgrScreen = convertToType(screen, BufferedImage.TYPE_3BYTE_BGR);  
  
    img = IplImage.createFrom(bgrScreen);  
    opencv_highgui.cvWriteFrame(vw, img);  
}
```

opencv_highgui.cvWriteFrame() is the method to write a image into a video file.

3.2.2 Spark cluster

As mentioned in 3.1.2, the Spark cluster is built by three Ubuntu virtual machines, one master, two workers. First step is to modify their IP address and the host name using comment line in Linux (sudo nano). Take the master as an example:

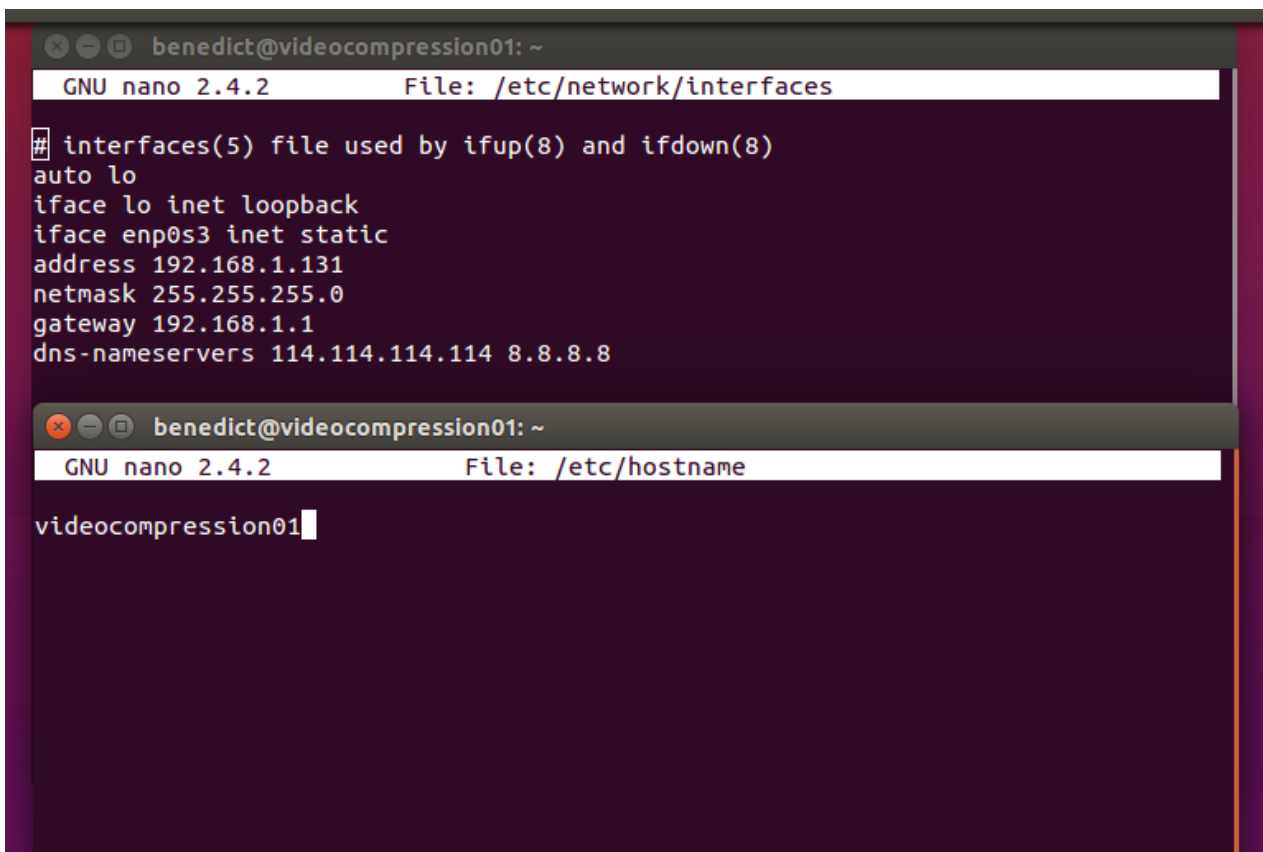
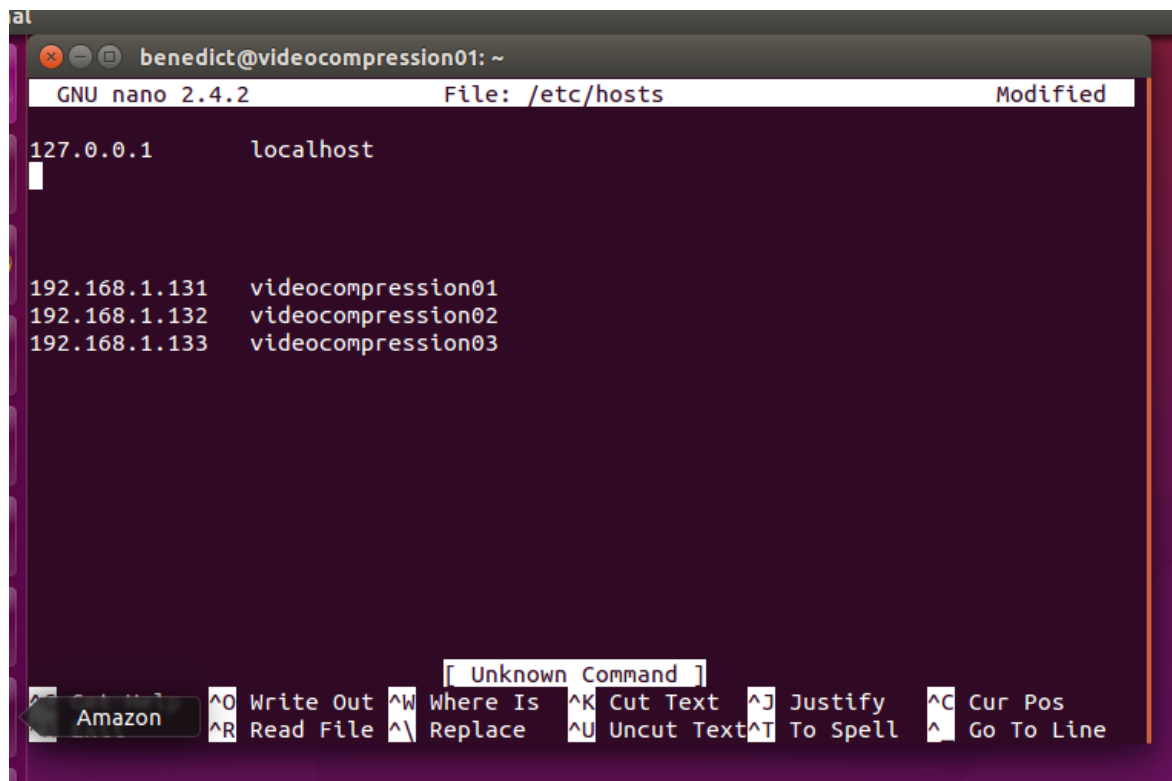


Figure 7 Modified IP address and host name

After modifying the host name and IP address in the three virtual machines, the next step is to register

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

all the virtual machines' host names and IP addresses into their /etc/hosts files for the latter connection in the cluster:



The screenshot shows a terminal window titled 'benedict@videocompression01: ~'. The window displays the contents of the /etc/hosts file using the GNU nano 2.4.2 editor. The file contains the following entries:

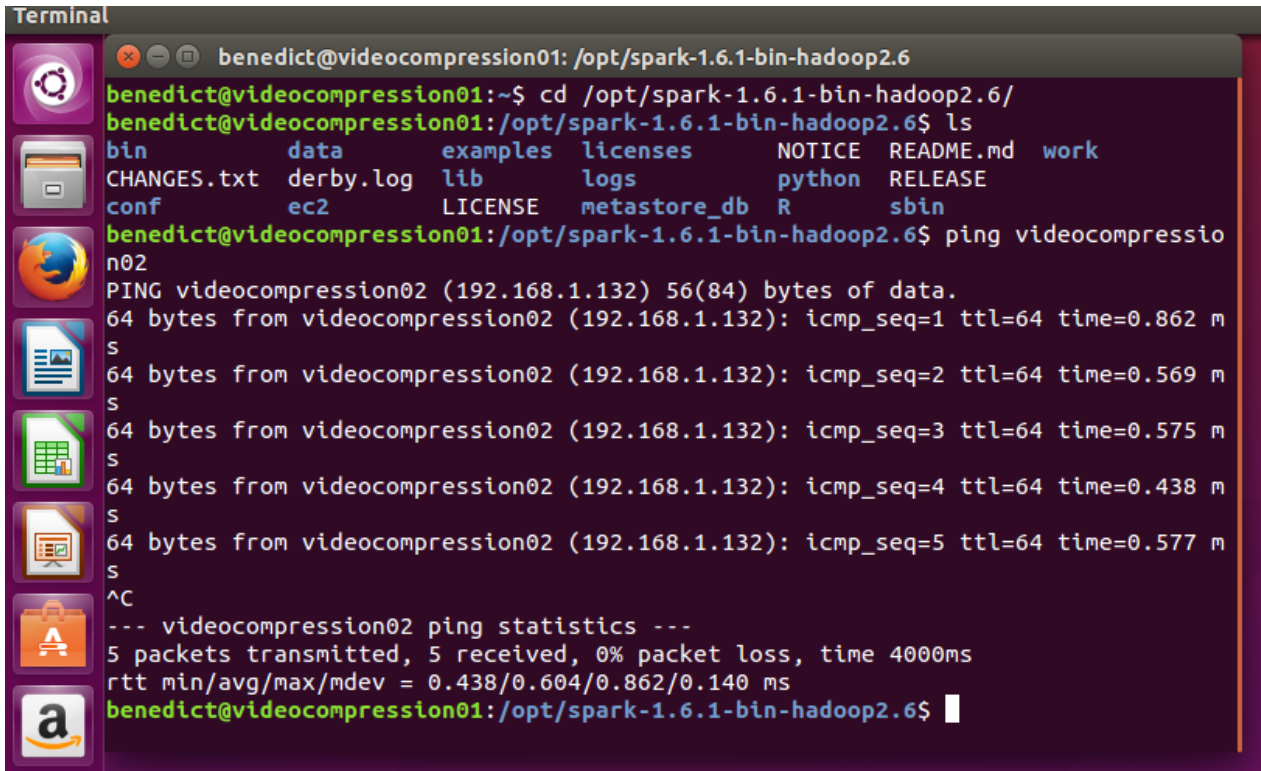
```
127.0.0.1    localhost

192.168.1.131 videocompression01
192.168.1.132 videocompression02
192.168.1.133 videocompression03
```

The bottom of the terminal shows the nano editor's command palette with various shortcuts like ^O Write Out, ^R Read File, ^W Where Is, ^K Cut Text, ^J Justify, ^C Cur Pos, ^X Exit, ^U Uncut Text, ^T To Spell, and ^_ Go To Line. An 'Unknown Command' message is also visible.

Figure 8 Mapping between IP address and host name

Then I ping one of the virtual machines from the master to test:

A terminal window titled "Terminal" with a dark background and light text. The window shows a user named "benedict" at a machine named "videocompression01". The user is in the directory "/opt/spark-1.6.1-bin-hadoop2.6". They run "cd /opt/spark-1.6.1-bin-hadoop2.6/" and then "ls", which lists various files and directories including "bin", "data", "examples", "licenses", "NOTICE", "README.md", "work", "CHANGES.txt", "derby.log", "lib", "logs", "python", "RELEASE", "conf", "ec2", "LICENSE", "metastore_db", "R", and "sbin". Then, they run "ping videocompression02", which shows five successful ping requests to the IP 192.168.1.132 with varying response times. Finally, they run "ping statistics", which shows that 5 packets were transmitted and received with 0% packet loss and an average round-trip time of 0.604 ms.

```
Terminal
benedict@videocompression01: /opt/spark-1.6.1-bin-hadoop2.6
benedict@videocompression01:~$ cd /opt/spark-1.6.1-bin-hadoop2.6/
benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$ ls
bin      data      examples  licenses  NOTICE  README.md  work
CHANGES.txt  derby.log  lib       logs      python   RELEASE
conf      ec2       LICENSE   metastore_db  R        sbin
benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$ ping videocompression02
PING videocompression02 (192.168.1.132) 56(84) bytes of data.
64 bytes from videocompression02 (192.168.1.132): icmp_seq=1 ttl=64 time=0.862 ms
64 bytes from videocompression02 (192.168.1.132): icmp_seq=2 ttl=64 time=0.569 ms
64 bytes from videocompression02 (192.168.1.132): icmp_seq=3 ttl=64 time=0.575 ms
64 bytes from videocompression02 (192.168.1.132): icmp_seq=4 ttl=64 time=0.438 ms
64 bytes from videocompression02 (192.168.1.132): icmp_seq=5 ttl=64 time=0.577 ms
^C
--- videocompression02 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4000ms
rtt min/avg/max/mdev = 0.438/0.604/0.862/0.140 ms
benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$
```

Figure 9 Ping videocompression02

One virtual machine successfully communicates with another.

After successfully set the communication between the virtual machines, the next step is to establish the Spark cluster. I need to switch to the root directory of Spark, and modify the *conf/slaves* file. Entering the host name of the worker nodes in the *slaves* file, the master will find the IP address in the */etc/hosts* file modified before.



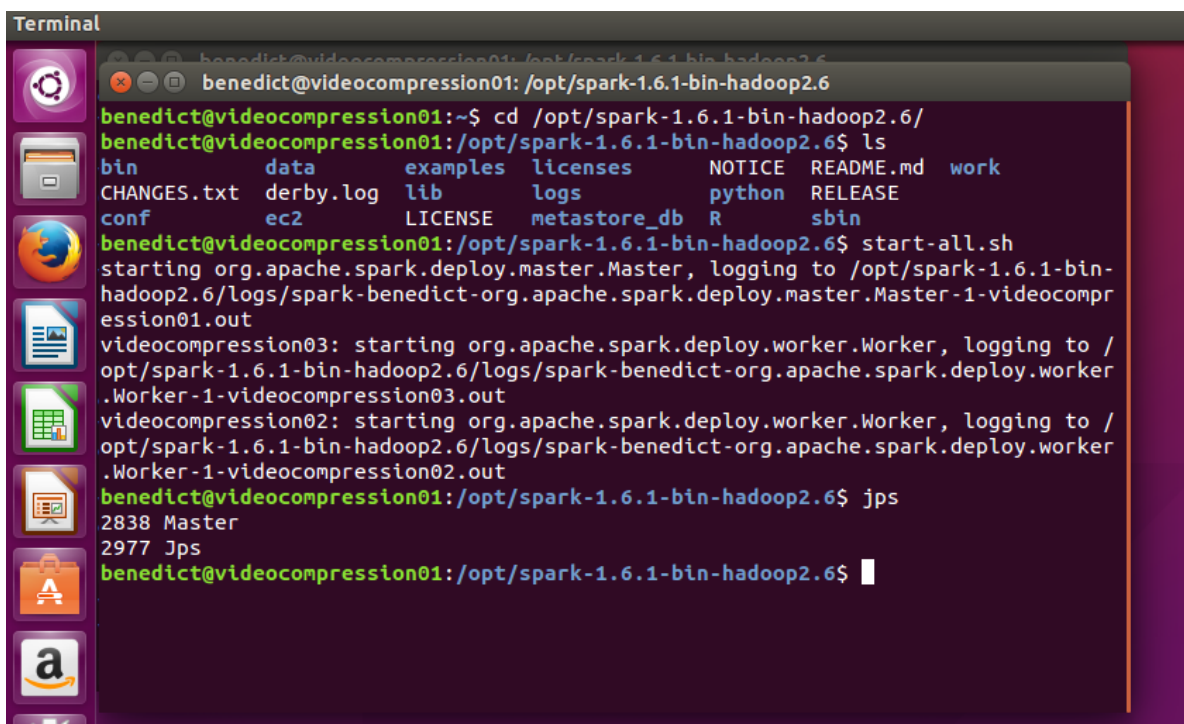
```
terminal
benedict@videocompression01: /opt/spark-1.6.1-bin-hadoop2.6
GNU nano 2.4.2 File: conf/slaves

# Unless required by applicable law or agreed to in writing, software
# distributed under the License is distributed on an "AS IS" BASIS,
# WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
# See the License for the specific language governing permissions and
# limitations under the License.
#

# A Spark Worker will be started on each of the machines listed below.
videocompression02
videocompression03
#localhost
```

Figure 10 set the worker nodes

Then it is time to start the cluster. Using strat-all.sh to start the cluster and the comment line jps to check.



```
Terminal
benedict@videocompression01: /opt/spark-1.6.1-bin-hadoop2.6

benedict@videocompression01:~$ cd /opt/spark-1.6.1-bin-hadoop2.6/
benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$ ls
bin      data      examples  licenses  NOTICE  README.md  work
CHANGES.txt  derby.log  lib       logs      python   RELEASE
conf      ec2       LICENSE   metastore_db  R        sbin

benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$ start-all.sh
starting org.apache.spark.deploy.master.Master, logging to /opt/spark-1.6.1-bin-hadoop2.6/logs/spark-benedict-org.apache.spark.deploy.master.Master-1-videocompression01.out
videocompression03: starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark-1.6.1-bin-hadoop2.6/logs/spark-benedict-org.apache.spark.deploy.worker.Worker-1-videocompression03.out
videocompression02: starting org.apache.spark.deploy.worker.Worker, logging to /opt/spark-1.6.1-bin-hadoop2.6/logs/spark-benedict-org.apache.spark.deploy.worker.Worker-1-videocompression02.out
benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$ jps
2838 Master
2977 Jps
benedict@videocompression01:/opt/spark-1.6.1-bin-hadoop2.6$
```

Figure 11 start the cluster

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

After starting the cluster, I can check whether it is available for the local windows to connect to the Spark cluster, since the driver program needs to connect to the cluster from local windows. We can connect to the cluster using the 8080 port of the master node to check the status of the cluster from local web browsers.

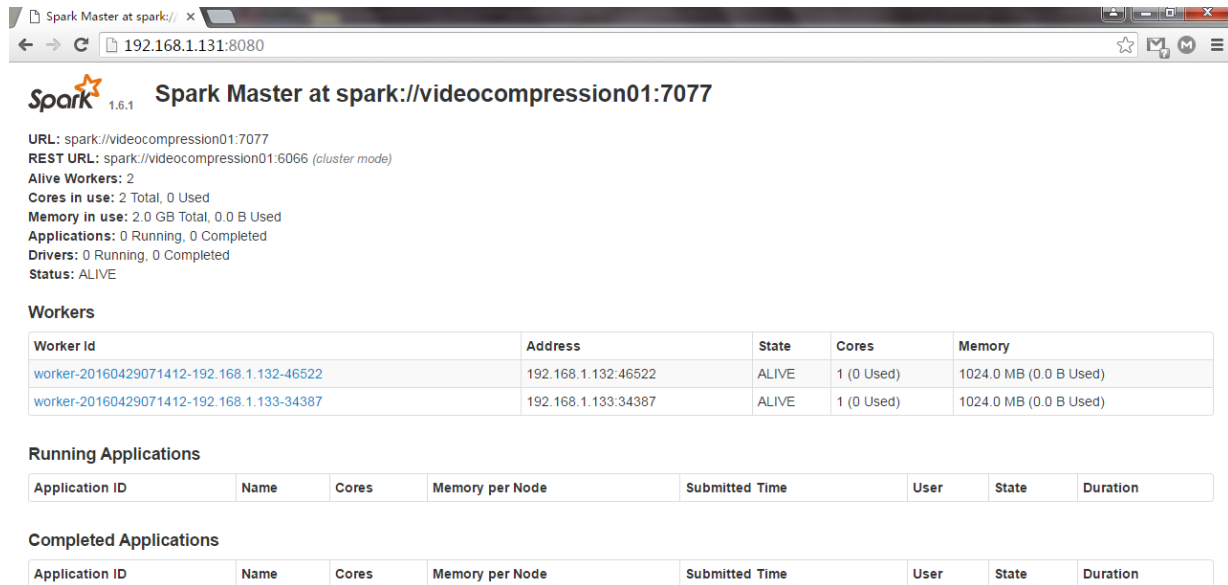


Figure 12 connect to the cluster from local

From figure 12, it can be seen that the local system successfully connect to the cluster, and see the two worker nodes in the cluster.

Up to now, the Spark cluster is successfully established and able to be connected from local.

Then I need to create the instance of Spark in the program. As mentioned before, the program need to have *Sparkconf* to set all the configuration of the application, and *JavaSparkContext* to start the application. Meanwhile, I need to deploy the jar package of the whole program earlier than start the application, since the driver program need to find the jar package, and let the master node send it to the worker nodes.

```
SparkConf sparkConf = new SparkConf();
sparkConf.setAppName("VideoSynopsis");
sparkConf.setMaster("spark://192.168.1.131:7077");
String[] jars = new String[1];
jars[0] = "/home/benedict/Downloads/synopsis.jar";
```

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

```
sparkConf.setJars(jars);
```

```
JavaSparkContext ctx = new JavaSparkContext(sparkConf);//Instantiating a spark context to start  
the spark application
```

As seen from the code, the name of the application is “VideoSynopsis” and the master URL is the IP address I have set in the master node of the cluster.

3.2.3 Web pages

As mentioned in 3.1.3, the web pages use the framework from twitter, and insert the .css files in to .html files. Hence, the web pages are implemented as:

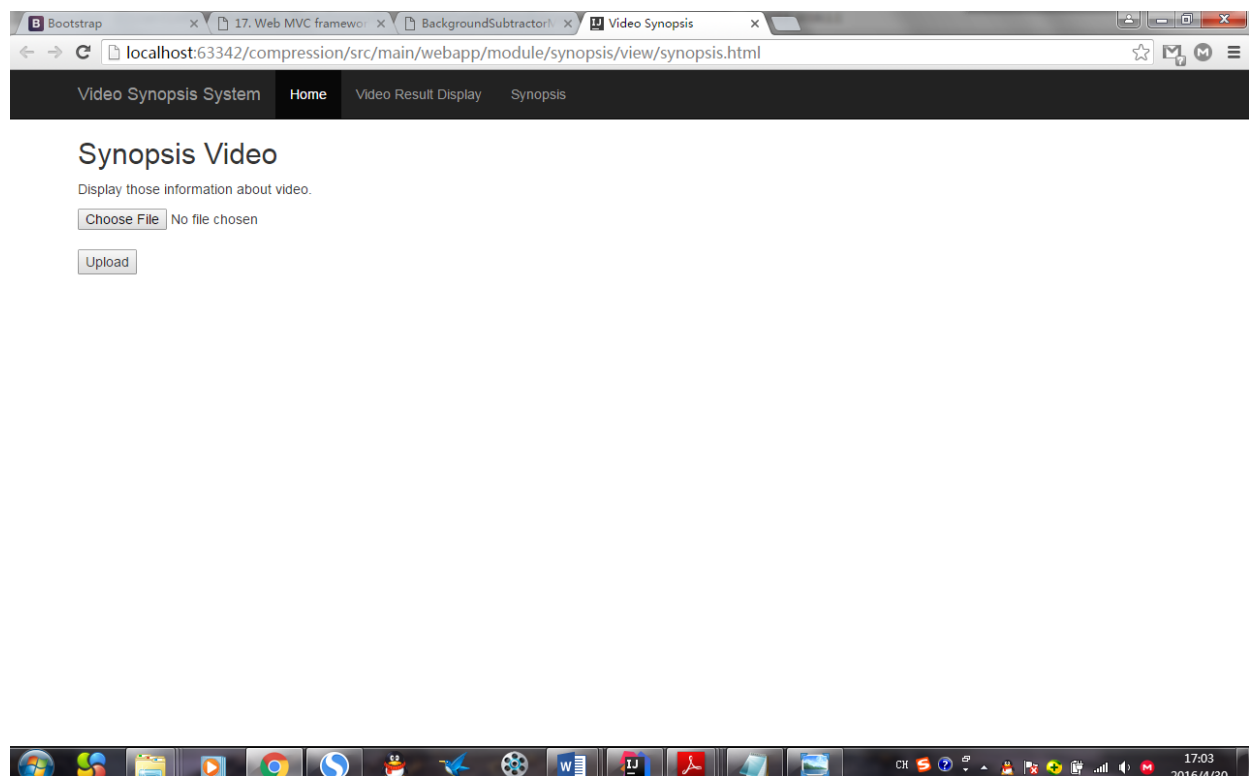


Figure 13 Upload page

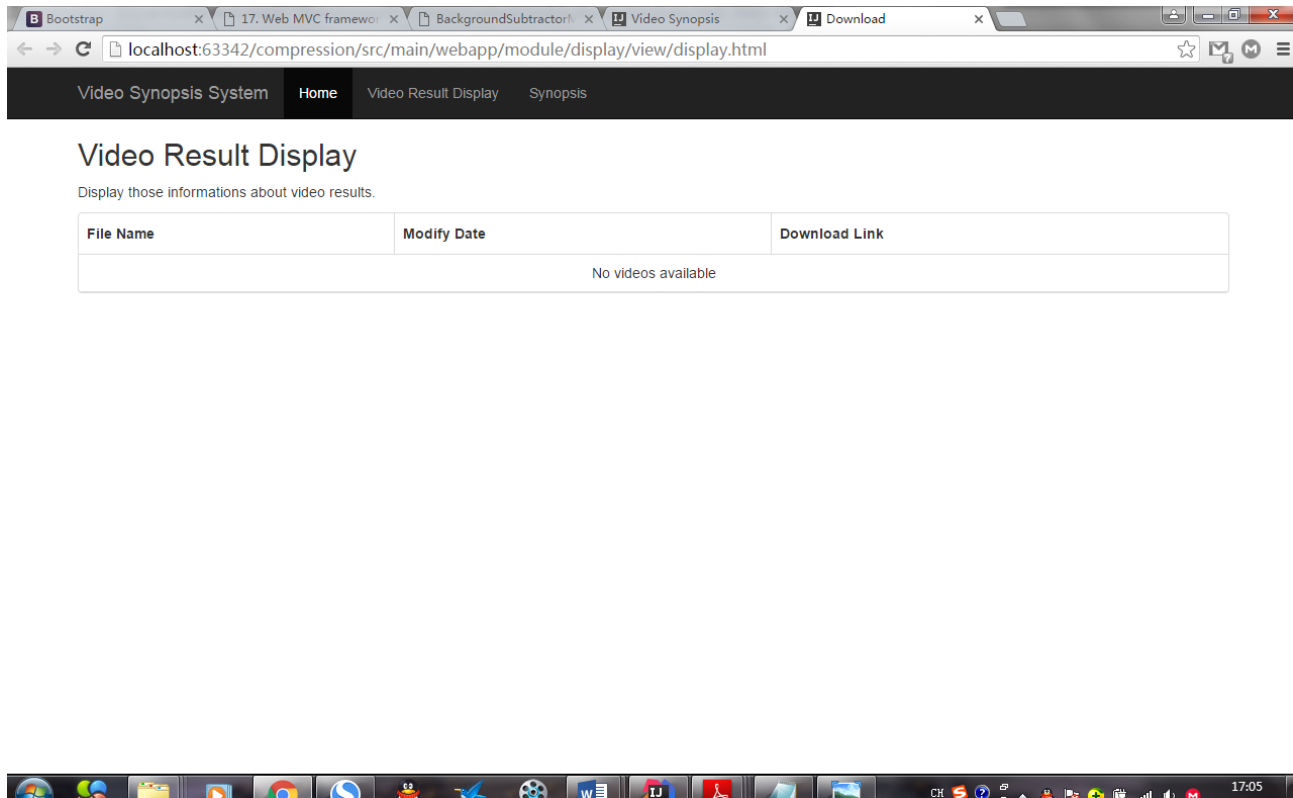


Figure 14 Download page

As seen in the upload page, users can choose a video file and upload it for processing; while the download pages shows the condensed video files' names, modified dates and their download links.

After implement the web page, the system needs a controller that can respond to users' request from the web pages. Additionally, it is needed to add some other classes to be used for the controller. Using Spring MVC framework is convenient to accomplish that. The following codes are some part from the controller.

```
@RequestMapping(value = "/upload", method = RequestMethod.POST)
@ResponseBody
public String upload(MultipartHttpServletRequest request, HttpServletResponse response) {

    Iterator<String> itr = request.getFileNames();
    MultipartFile mpf = request.getFile(itr.next());
    _log.debug(mpf.getOriginalFilename() + " successfully uploaded !");

    uploadedFile.type = mpf.getContentType();
    videoService.videoSynopsis(mpf);
}
```

```

        return "[{\"result\":\"success\", \"uploadTime\":\"\" + sdf.format(new Date()) +
        \"\"}]";
    }

```

As seen in the code, this method map to the POST request generated when users press the upload button in the web page. The method get the name of video file from the *request* object, and use the object *videoService* to call the method to process the video. The object *videoService* is an instance of a class I wrote to call the video synopsis algorithm to process the video.

The following code of the controller is to process the information on the download page:

```

@RequestMapping(value = "/files", method = RequestMethod.GET, produces = "text/plain")
@ResponseBody
public List<VideoFilesDao> displayFiles(HttpServletRequest request, Model model) {

    return videoService.displayResultFile();
}

```

Since the download page just needs to show the information of the processed video files, so I use a method called *displayResultFile()* to display the information. Still, this method is from the object *videoService*. The following is the code of this method:

```

public List<VideoFilesDao> displayResultFile() {
    String videoPath = propUtils.getProperty("result.video.path");
    if (StrUtils.isEmpty(videoPath))
        return null;
    File resultPath = new File(videoPath);
    if (!resultPath.exists() || !resultPath.isDirectory()) {
        return null;
    }
    List<VideoFilesDao> videoFilesDaoList = new LinkedList<>();
    VideoFilesDao videoFilesDao;
    for (File video : resultPath.listFiles()) {

```

```
        videoFilesDao = new VideoFilesDao();
        String name = video.getName();
        videoFilesDao.setFileName(name);
        videoFilesDao.setModifyDate(sdf.format(new Date(video.lastModified())));
        videoFilesDao.setDownloadLink("http://localhost:8080/video/get/".concat(name));
        videoFilesDaoList.add(videoFilesDao);
    }
    return videoFilesDaoList;
}
```

As shown in the code above, it returns all the information of the processed video in an object called *videoFilesDaoList*, which is an array list of *VideoFilesDao*. *VideoFilesDao* is a data access object which contains some method to get or set the basic information of the video files such as file name and download link.

Chapter 4: Results and Discussion

After successfully implementing the whole system, I run the web system to check the result.

The first step is no doubt starting the Spark cluster on Ubuntu virtual machines and deploying the application into web server. The system uses tomcat 8 as the web server. Running the program, the console shows:

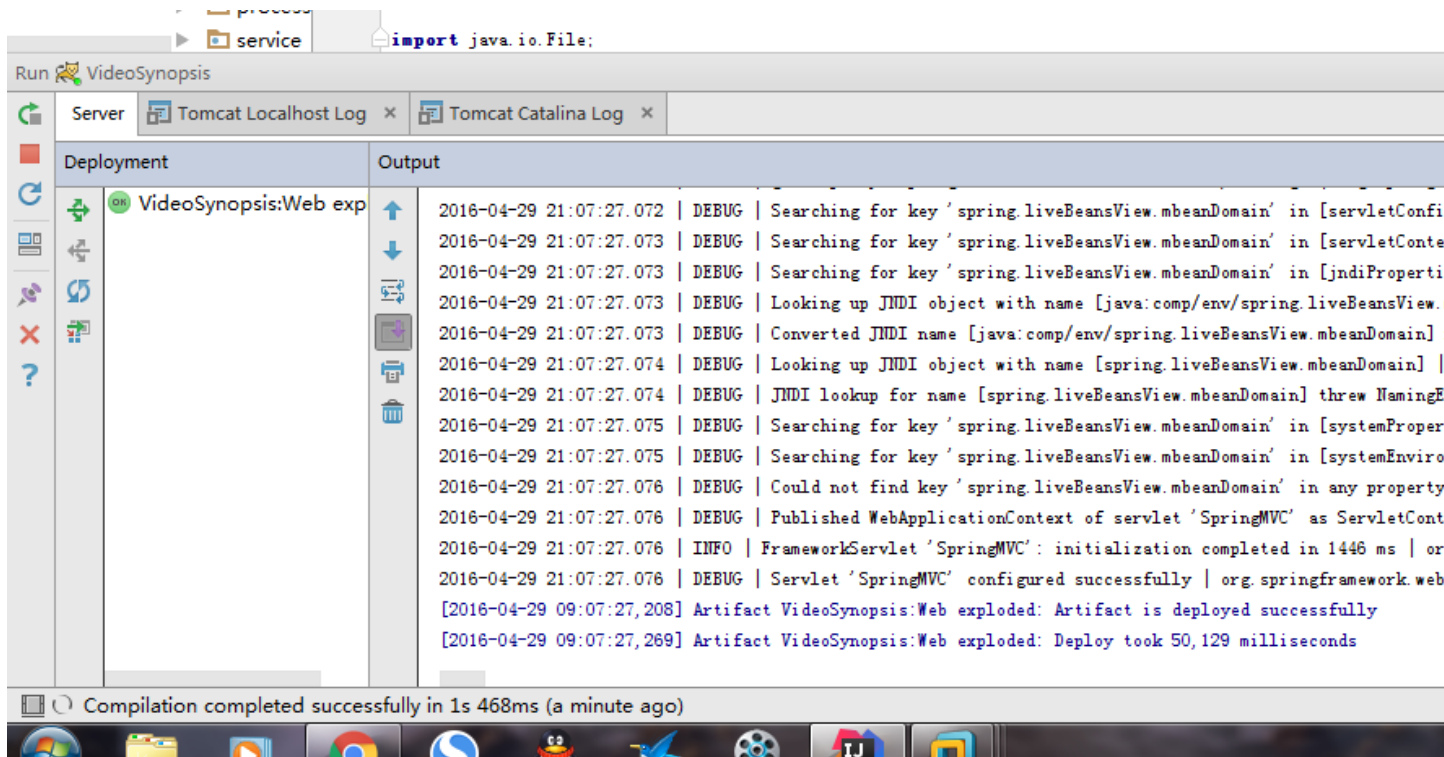


Figure 15 successfully deploy the web application

As you can see, the application is successfully deployed. Checking the status of Spark cluster is the same as shown in figure 12 in 3.2.2, checking the 8080 port of the master.

After deploying the application, the home page of the application will automatically pop out from the web browser. From the home page, you can choose to upload a video or checking the results from the navigation bar on the top.

The home page pops out from chrome browser as I set:

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

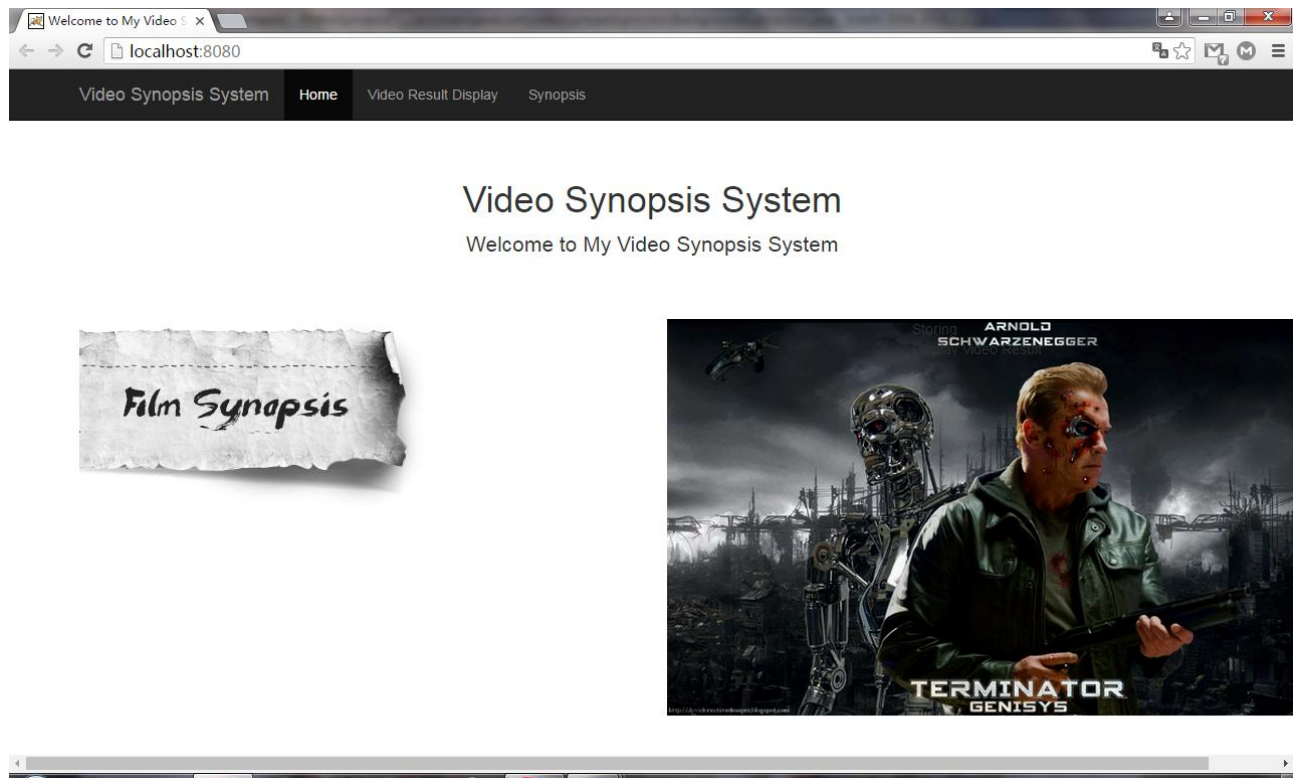


Figure 16 home page

Then click the synopsis to upload a video file. All works well,

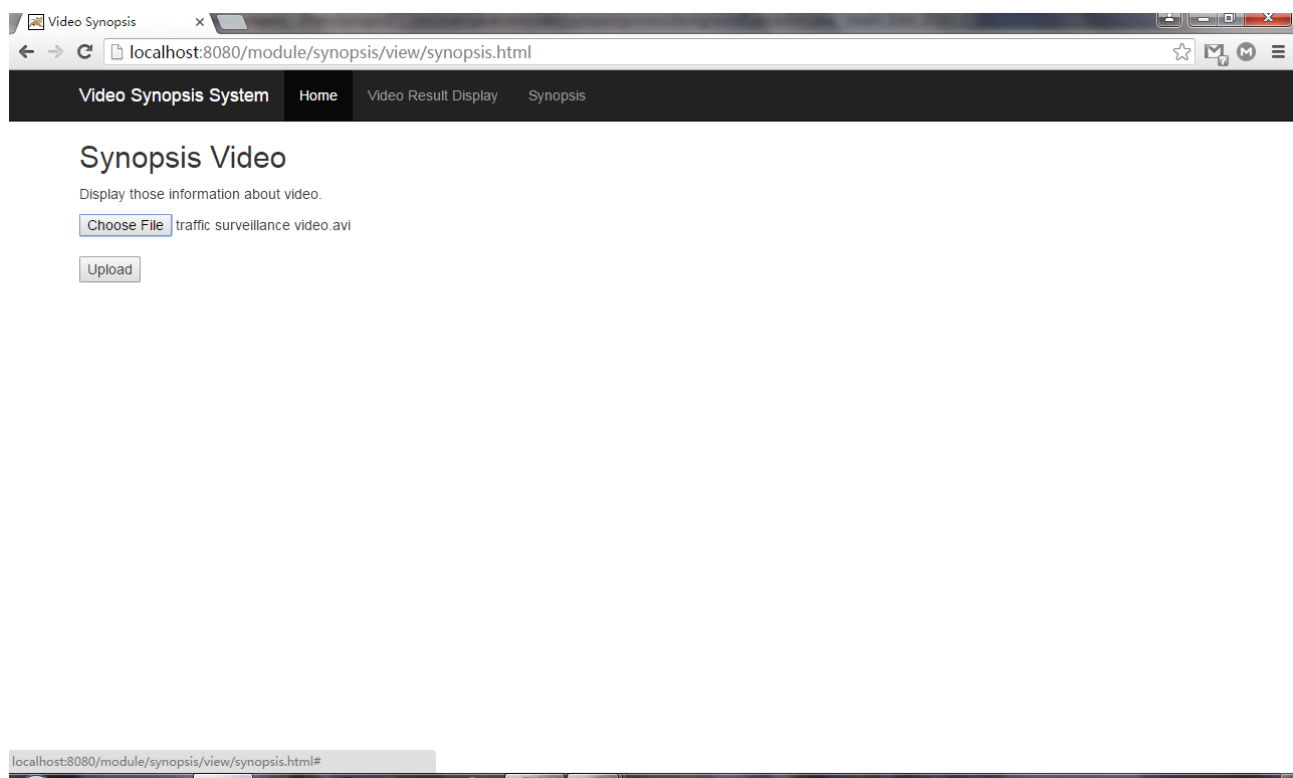


Figure 17 upload traffic surveillance video.avi

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

After uploading the video file, click upload button to upload the file to be processed. Then there will be a slightly long duration for the system to connect to the Spark cluster and process the video file.

After the video file is processed, the web page will return a success message to show that the video has already been processed. You can also log on the Spark master's 8080 port to check the status of the task.

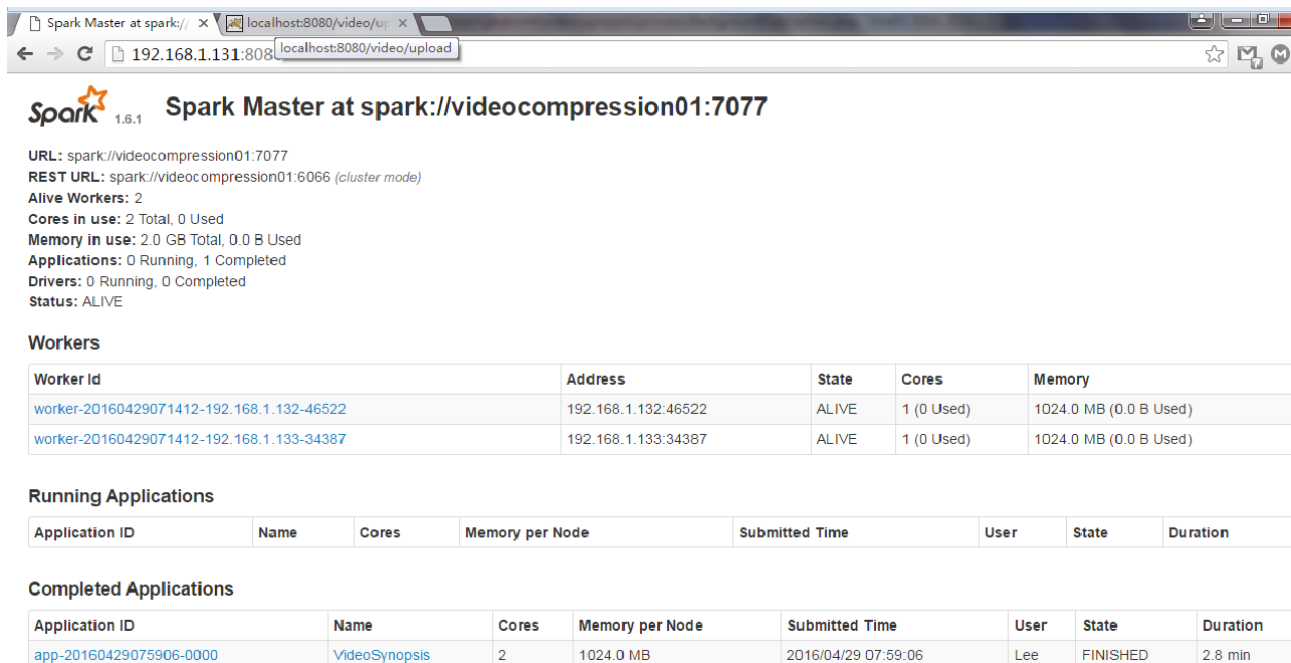
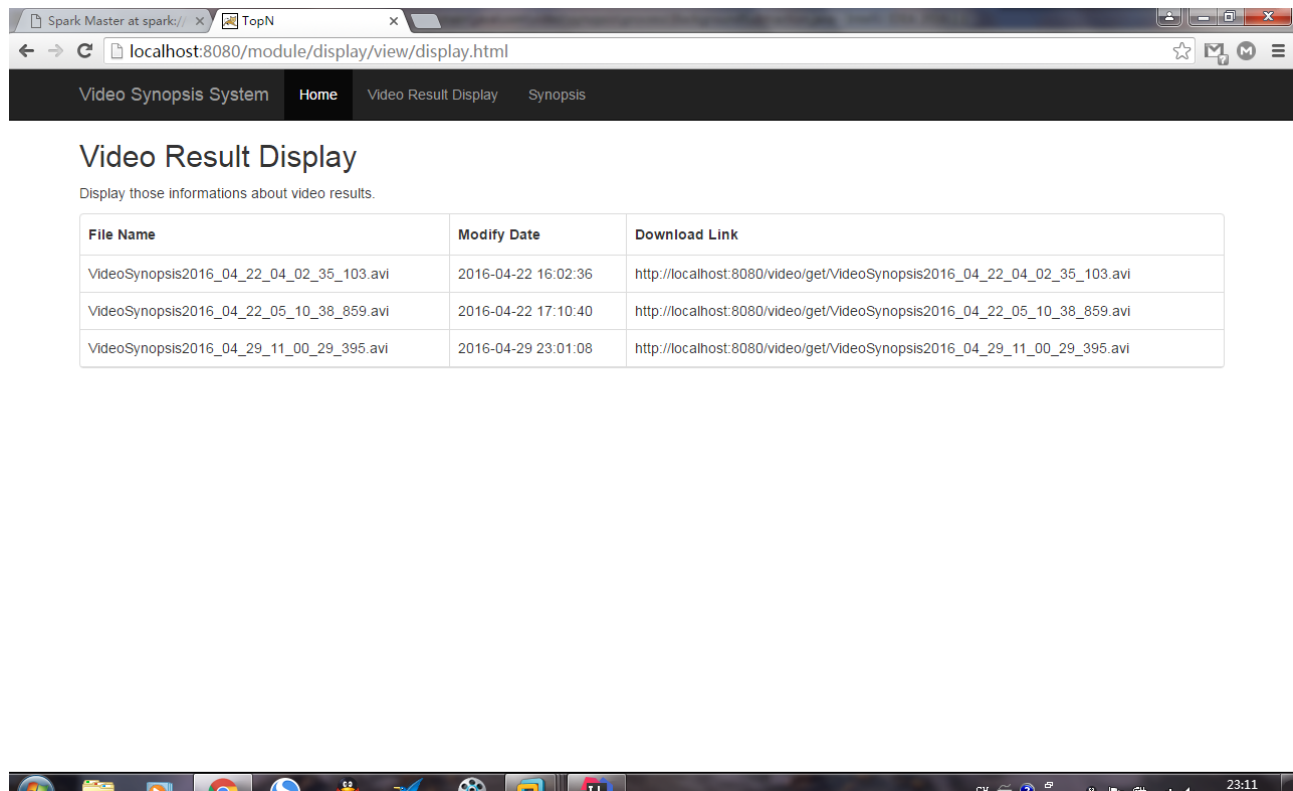


Figure 18 Finished task seen from port 8080

As shown under the “Completed Application” we can find that the application “VideoSynopsis” has been finished. And meanwhile you can see the two worker nodes alive in the cluster, which is the reason why the number of cores under the completed application is 2. And the memory per node is exactly the number I have set in the virtual machine.

Up to now, the completed procedures run well, then I log back to the web page to check whether the information of the processed video is shown in the download page.

[IC_3313] [Video Synopsis and Indexing System Based on Spark]



The screenshot shows a web browser window with the URL `localhost:8080/module/display/view/display.html`. The page title is "Video Result Display" and it includes a navigation bar with "Video Synopsis System", "Home", "Video Result Display", and "Synopsis". Below the title, it says "Display those informations about video results." and displays a table with three rows of video information.

File Name	Modify Date	Download Link
VideoSynopsis2016_04_22_04_02_35_103.avi	2016-04-22 16:02:36	http://localhost:8080/video/get/VideoSynopsis2016_04_22_04_02_35_103.avi
VideoSynopsis2016_04_22_05_10_38_859.avi	2016-04-22 17:10:40	http://localhost:8080/video/get/VideoSynopsis2016_04_22_05_10_38_859.avi
VideoSynopsis2016_04_29_11_00_29_395.avi	2016-04-29 23:01:08	http://localhost:8080/video/get/VideoSynopsis2016_04_29_11_00_29_395.avi

Figure 19 Information of processed videos

As seen in figure 19, all the information is correct in the download page. Then I can download the video using its download link.

Before checking the final video clip, as mentioned before, the synopsis algorithm will keep the grabbed images as intermediate results, the following figures show frames of the moving objects in three seconds:



Figure 20 Intermediate image



Figure 21 Intermediate image



Figure 22 Intermediate image

Figure 20 to figure 22 have two moving objects surrounded with red bound box, and the objects are in three continuous seconds. As seen from the figures, the intermediate frames clearly reflect the movement locus of the objects. If you check the image clearly, you can see two or three extremely small moving objects that are not surrounded by bound box. That is because they are too far from the surveillance camera, so that their area is too small to meet the condition set in the algorithm.

Finally, let's check the condensed video after the series of processes.



Figure 23 Comparison between the condense video clip and the original video

As shown in figure 21, the left side is the condensed video and the right side is the original video. The processed video can be played well in any normal video player, and I play it in Thunder player. As you can see, all the moving targets under the surveillance camera are marked by red bound boxes. The duration of the original video is about twenty two minutes, while the condensed video is just about three minute. Therefore, the video synopsis procedure works because of the extraction of moving targets and the increase of fps (frames per second).

Chapter 5: Conclusion and Further Work

5.1 Conclusion

This project focuses on implement a video synopsis and indexing system based on Spark. Its purpose is to condense a long surveillance video file which has low density of moving objects in a long period of time into a new video clip which displays all the moving targets in the original video in a much shorter duration. To achieve this purpose, I have done these in the project:

1. I have wrote a video synopsis algorithm which could be divided into two major parts: using Mixture of Gaussian background model to extract frames containing moving targets as intermediate results; using video writer in JavaCV library to recombine the intermediate frames sequentially into the condense video clip.
2. I have deploy a Standalone Spark cluster, one master and two workers. The local program can connect to the Spark cluster to use the cluster's ability of computation for the video synopsis procedure.
3. Plus, I have implement a web page interacting with users. Users can upload the video file they want to be condensed, and the web page will send request to the backstage controller which can call the method mentioned above to process the video file. After condensing the video file, users can download the processed file from the download page.

To be honest, I have met some difficulties while doing the project. For instance, the first one is how to extract the moving target from the video clip. I have tried to use some simple motion detection method that can roughly marker the moving targets in the video file. The problem is the poor accuracy of finding moving objects. Sometimes the leaves and even the bushes will be marked as moving targets. Meanwhile, this method is just find the moving targets rather than extract them. So I discussed with my supervisor and some Master students, deciding to use Gaussian background subtraction to extract the moving targets. What's more, there are also problems when I recombine the frames into video file. The default coding format generate very large video files, the condensed video generate by the former coding format is about 4 GB. Luckily, I change to use the MPEG-4 coding format and it generate video file in proper size, about 120 MB.

5.2 Further work

I have to admit that there are still some aspect in my project need to be improved. The first thing is that the Spark cluster Standalone mode uses the master to make scheduling decision, which has the possibility for single point of failure. The potential improvements are relocating the cluster into the other two modes (Hadoop Yarn and Mesos), or setting up multiple master cluster. ^[13]

Another improvement can be made is to improve the quality of the generate video and try to reduce its size. In spite that MPEG-4 coding format can generate video files in proper size, but the generated condensed video file is almost the same size as the original longer video file. The possible solution is to modify the part of algorithm generating the new video file for a further increment of the density of moving objects or modify the coding format.

References

- [1] Wikipedia, *OpenCV*. Retrieved 2016, Web site: <https://en.wikipedia.org/wiki/OpenCV>
- [2] Ffriend, (2014, May). *OpenCV (JavaCV) vs OpenCV (C/C++ interfaces)*, Retrieved April, 2016, from Stack Overflow. Web site: <http://stackoverflow.com/questions/21207755/opencv-javacv-vs-opencv-c-c-interfaces>
- [3] ZHAN Chaohui, DUAN Xiaohui*, XU Shuoyu, SONG Zheng, LUO Min. *An Improved Moving Object Detection Algorithm Based on Frame Difference and Edge Detection*, IEEE Computer Society, pp. 519-523
- [4] HAN Jiankang, *Moving Area Detection and Tracking Based Video Condensation*, Beijing University of Posts and Telecommunications, 2012 TP391.41
- [5] Kamna Kohli, *Motion Detection Algorithm*. The International Journal of Computer Science & Applications (TIJCSA), Volume 1, No. 12, February 2013 ISSN – 2278-1080
- [6] Wikipedia, *MPEG-4*. Retrieved April 2016. Web site: <https://en.wikipedia.org/wiki/MPEG-4>
- [7] Apache Spark, web site: <http://spark.apache.org/>
- [8] Figure from Learning Spark, *Introduction to Data Analysis with Spark*. Retrieved April, 2016. Web site: <https://www.safaribooksonline.com/library/view/learning-spark/9781449359034/ch01.html>
- [9] Jacek Laskowski (2015), *Mastering Apache Spark*. Published with Gitbook
- [10] Apache Spark, *Cluster Overview*. Retrieved April, 2016. Web site: <http://spark.apache.org/docs/latest/cluster-overview.html>
- [11] Apache Spark, *Running Spark on YARN*, Retrieved April, 2016. Web site: <http://spark.apache.org/docs/latest/running-on-yarn.html>
- [12] Apache Spark, *Spark Standalone Mode*. Retrieved April 2016. Web site: <http://spark.apache.org/docs/latest/spark-standalone.html#connecting-an-application-to-the-cluster>
- [13] Matlab works, *How to extract foreground mask of human from video?* Retrieved April, 2016. Web site: <http://cn.mathworks.com/matlabcentral/answers/264980-how-to-extract-foreground-mask-of-human-from-video?>

Acknowledgement

My deepest and sincerest gratefulness firstly goes to my supervisor, Dr. Zhang, for his patience, kind and passion. Dr. Zhang has not only show his great ability in academic, giving professional advice to my problems, but also show his enthusiastic and sincerity in helping all his students.

Next, I want to say thank you to all the BUPT and QM lecturers. It is them who taught me how to learn and how to accommodate lectures in university.

Then I want to thank the Master students of Dr. Zhang, who have also give me great help in familiarizing and designing this project.

At I want to thank my roommates and my family, who have always been very supportive to me. Even when I adjusted the system into the midnight, they were still supportive and cheering me up.

Appendix

Specification

北京邮电大学 本科毕业设计（论文）任务书 Project Specification Form					
学院 School	International School	专业 Programme	Internet of Things	班级 Class	2012215118
学生姓名 Name	LI Zijian	学号 BUPT student no	2012213313	学号 QM student no	120726859
设计（论文）编号 Project No.	IC_3313				
设计（论文）题目 Project Title	Video synopsis and indexing system based on Spark				
论文题目（中文）	基于Spark的视频浓缩与索引系统				
题目分类 Scope	Implementation	Computer Software	Software		
主要任务及目标 Main tasks and target:					By
Task 1: Study the related video processing technologies and Spark framework					15 January 2016
Task 2: Program software to implement video synopsis algorithm					30 March 2016
Task 3: Program software to implement distributed video synopsis function based on Spark					30 April 2016
Task 4: Program a Web based video synopsis and indexing system that can interact with users					15 May 2016
Measurable outcomes					
1) Software to implement video synopsis					
2) Software to implement distributed video synopsis function based on Spark					
3) A Web based video synopsis system that can interact with users, and a technique report					
主要内容 Project description:					
<p>Surveillance video is a typical type of big data, while users are usually interested in the moving targets of the surveillance video in practice. Finding the interesting targets in large amounts of surveillance videos needs a mass of manpower and resources. One of the project aims is to design and implement a video synopsis and indexing system. The system can recombine all moving targets in a very short video clip and index the original surveillance videos based on the extracted moving objects. On the other hand, the video processing algorithm is so complex that the system needs to take a long time for processing massive videos. However, Spark is a popular distributed computing framework, which can process large amount of videos efficiently. This project is to design and implement a Spark-based video synopsis and indexing system.</p>					
Project outline					
<p>The application can be divided to two basic parts: the foreground web page interacting with users and the background program for condensing, reserving and uploading videos.</p> <p>The foreground of the system is supposed to be a web page which can interact with users. Users are able to upload videos to the web server as well as manage the basic parameters of the videos.</p> <p>When uploading the videos, users are able to choose videos of specific cameras, and videos from one camera should be in a continue interval. Meanwhile, users are allowed to change the files where the videos from. In addition, users can choose the storage path of the condensed videos. What's more, users should set the ratio of original and processed videos, which is the standard for the background program to condense the original videos. After uploading, the videos will be uploaded to the web server.</p> <p>Of course users should be able to download the condensed videos. Hence the web page should display basic information of condensed videos. Such as providing the users with the time duration of the processed video.</p> <p>The background should be able to condense surveillance videos for specific users, reserve them and upload them on the web. Spark is used to process the videos on the web server.</p> <p>After users uploading videos to the web server, it will load the videos to Spark cluster and the distributed computing system will deal with the videos as the users expect.</p> <p>The key mission for Spark cluster is to condense a long surveillance video into short video clips. The program is supposed to detect the moving target in the video and recombine them in a much shorter clip. The algorithm used to process the videos should be able to deal with original videos according to the requirements receiving from the web page which set by users. After the videos condensed, Spark cluster will write it to the exact path which users set for users to download.</p>					

Fill in the sub-tasks and select the cells to show the extent of each task													
	Nov	Dec	Jan	Feb	Mar	Apr	May						
Task 1: Study the related video processing technologies and Spark framework													
Search information concern about Spark and													
Study basic skills about web-based													
Study video synopsis process													
Study Spark computing system													
Task 2: Program software to implement video synopsis algorithm													
Design and implement the web page interacting													
Implement the web server													
Implement the framework of the application													
Study and implement the synopsis algorithm													
Task 3: Program software to implement distributed video synopsis function based on													
Search and implement the software of Spark													
Implement the software of distributed video													
Task 4: Program a Web-based video synopsis and indexing system that can interact with													
Connect the web page with the distributed													
Test the results													
Accomplish the final report													

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

Early-term progress report

北京邮电大学
BBC6521 Project 毕业设计 2015/16

Early-term Progress Report
初期进度报告

学院 School	International School	专业 Programme	Internet of Things	班级 Class	2012215118
学生姓名 Student Name	Li Zijian	学号 BUPT Student No.	2012213313	学号 QM Student No.	120726859
设计（论文）编 号 Project No.	BZHANGHAITAO4	电子邮件 Email	2012213313@bupt.edu.cn		
设计（论文）题 目 Project Title	Video Synopsis and Indexing System Based on Spark				
<p>已完成工作： Finished Work:</p> <p>In the first phase of the final project, my focus is on the learning the whole structure of the project, searching concerning documents and reference, as well as designing and implementing the framework of front web page.</p> <p>Firstly I did some research of basic Spark knowledge, learning principles of MapReduce and Spark streaming. Then I learned JavaCV and some of its algorithm processing videos. After that, I install 64-bit Ubuntu operating system on my laptop and set up Spark cluster.</p> <p>After the fundamental configuration of back end, I started to design and implement the front end web page. Up to now, I have implemented the basic framework of some web pages in the application. I have already written the framework of the header JSP file, which is the fundamental component of each web page in the whole application, so as the footer JSP file. Meanwhile, I implement the framework of the JSP page which can show surveillance videos that are under processing of already processed for user to download, as well as the JSP pages that show surveillance cams.</p>					
是否符合进度？ On schedule as per GANTT chart?			[YES/NO] YES		

下一步:

Next steps:

In the next phase of the project, I will focus on further implementation of the front end web pages, concatenating each web pages and connecting the web pages with the back end cluster in laboratory. Meanwhile, I will start to implement the video synopsis algorithm and learn how to implement it on distributed Spark cluster.

[IC_3313] [Video Synopsis and Indexing System Based on Spark]

Mid-term progress report

北京邮电大学 本科毕业设计（论文）中期进展情况检查表 Mid Term Check Form					
学院 School	International School	专业 Programme	Internet of Things (物联网)	班级 Class	2012215118
学生姓名 Name	LI Zijian	学号 BUPT student no	2012213313	学号QM student no.	120726859
设计（论文）编号 Project No.	IC_3313				
设计（论文）题目 Project Title	Video synopsis and indexing system based on Spark				
题目分类 Scope	Implementation	Computer Software	Software		

主要内容：（毕业设计（论文）进展情况，字数一般不少于1000字）

Main body: The progress of the research on the project. Total number of words is no less than 1000.

目标任务 Targets set at initiation	At mid-term oral, I expect to complete the web page where users can upload and download videos.
是否完成目标 Targets met? Yes/No	Yes

目前已完成任务 Finished Work	<p>The key part of my project is to set up Apache Spark and embed the video synopsis algorithm in it. The whole background system should load surveillance video files in HDFS and process them offline. Meanwhile, there should be a web page which can interact with users.</p> <p>At this moment, I have done some work parallelly in setting up Spark, designing and programming the video synopsis algorithm and framework of web page.</p> <p>I have discussed with my supervisor and decided to focus on the background video processing system and I will complete the algorithm on Spark before connecting it with the web page. Up to now, I have learned the basic knowledge of Apache Spark and familiarize the Resilient Distributed Datasets (RDD), the basic abstraction in Spark, which represents an immutable, partitioned collection of elements that can be operated on in parallel. Meanwhile, I have already set up Spark on Ubuntu system and done some basic tests to make sure it is successfully installed.</p> <p>As for the video synopsis algorithm, I need to firstly program it in Java by virtue of some libraries from JavaCV and OpenCV. Thus, I have learned about video processing in Java language, especially in how to achieve video synopsis. After learning from my supervisor and some files on web page, I realized the algorithm should be separate in two parts: Firstly I need to do the background subtraction of the long video files, detect the moving target and extract frames of the video files as thousands of images which will be saved in local files; and then I need to concatenate the images into a condensed video clip. Up to know I have done some basic operations in background subtraction using Gaussian of Mixture Models (GMM). In my ongoing algorithm, I use some classes in JavaCV and openCV. I use the class FrameGrabber to grab every frames in the video file, and use the class BackgroundSubtractorMOG2 to segment background and foreground of the grabbed frame. The algorithm use BackgroundSubtractorMOG2.apply (Mat, Mat, Double) to compute the foreground mask from the grabbed image and output a binary image, then dilate, erode and smooth the binary image. And then find the contours from the binary image for object detection. Then it can use the contours to find the region of interest, that is to say, finding the moving target, and then I use a red bound box to show the moving target satisfying the condition. Then I save the processed image to a local file for further process.</p> <p>Besides the background of the system, I also program some basic jsp page for the front web page. I have already program the framework of the jsp file for web page where users can see surveillance videos that are under processing or processed.</p>
--------------------------	---

<p>尚需完成的任务 Work to do</p>	<p>The first work I need to do of the rest of my project is to implement and improve the video synopsis algorithm. Up to now I can roughly do the background subtraction, grab the video frame, detecting the moving target in the surveillance video and save the processed image to local file. In the next phase, I need to improve the background subtraction program, extract the background of surveillance video and save it to local file, too. Meanwhile, I need to extract the region of interest (moving target) from the video file, separate them from the background. Then I need to concatenate all the image of the moving target on the background to get the required synopsis clip.</p> <p>After accomplished the algorithm, I need to embed it into Spark, I have to accomplish a program of Spark which can read video files from HDFS, condense the video clips and save it to the HDFS. Then I need to connect the system to the web page.</p>	
	<p>能否按期完成设计(论文) Can finish the project on time or not: Yes/No</p>	<p>Yes</p>
<p>存在问题和解决办法 Problems and</p>	<p>存在 问题 Problems</p>	<ol style="list-style-type: none"> 1. Spark do not have method to load a whole video file. 2. The whole algorithm need to adapt with Spark. 3. It is my first time writing codes about video processing, and I barely know about JavaCV and OpenCV. 4. Lots of reference about video processing is using C++, yet I need use Java.
<p>存在问题和解决办法 Problems and Solutions</p>	<p>拟采取 的办法 Solutions</p>	<ol style="list-style-type: none"> 1. Modify the interface of the class which use to load files. 2. Trying to use Spark's API to adapt it with the java algorithm. 3. Learn from books, internet and my supervisor, and try to compare the methods in C++ with them in Java.

<p>最终论文结构 Structure of the final report</p>	<p>Abstract Short review of the whole report.</p> <p>Introduction Introducing the project and report. Briefly illustrate the goal of the project and what I have done. Meanwhile contenting the structure if the report.</p> <p>Background Giving the relevant background information about the project. Illustrating techniques used in the project.</p> <p>Design and Implementation Details of the project, illustrating the technique details of the whole system, and separately discussing each part of the system</p> <p>Result and Discussion Showing the results acquired from the system and analyzing them.</p> <p>Conclusion and Further Work Summarizing the projects and its outcomes.</p>
	<p>Concluding some improvement that can be done in the project.</p> <p>Reference</p> <p>Acknowledgement Giving acknowledgement to people who helped me during the project.</p> <p>Appendix</p>
<p>日期 Date</p>	<p>05/03/2016</p>

Risk Assessment

Description of Risk	Description of impact	Likelihood rating	Impact rating	Preventative action
Cannot connect to Spark cluster	The whole application will stop running	3 (moderate)	2 (serious)	Shun down the firewall of the local system
Laptop crash down	Code lost, application crashed, cluster crashed	1 (rare)	4 (Major)	Backup the code and the cluster
RAM of the laptop is insufficient	The application runs incredible slowly, and may crash down	4 (likely)	2 (serious)	Shutdown some unrelated software before running the application

Environmental Impact Assessment

This project is totally based on software development, hence it will not generate pollution or waste resources and has no environmental impact.