Temporal Signature Modeling and Analysis

by

Jiangqin Sun

B.S. Soochow University, 2007
M.S. Harbin Institute of Technology, 2009

A dissertation submitted in partial fulfillment of the
requirements for the degree of Doctor of Philosophy
in Imaging Science

Chester F. Carlson Center for Imaging Science
College of Science
Rochester Institute of Technology

December 6, 2014

Signature of the Author _____

Accepted by _____
Coordinator, Ph.D. Degree Program Date

CHESTER F. CARLSON CENTER FOR IMAGING SCIENCE
COLLEGE OF SCIENCE
ROCHESTER INSTITUTE OF TECHNOLOGY
ROCHESTER, NEW YORK

CERTIFICATE OF APPROVAL

Ph.D. DEGREE DISSERTATION

The Ph.D. Degree Dissertation of Jiangqin Sun
has been examined and approved by the
dissertation committee as satisfactory for the
dissertation required for the
Ph.D. degree in Imaging Science

Dr. David Messinger, Dissertation Advisor

Dr. Tony Harkin, Committee Member

Dr. Joel Kastner, Committee Member

Dr. Carl Salvaggio, Committee Member

Date

ABSTRACT

A vast amount of digital satellite and aerial images are collected over time, which calls for techniques to extract useful high-level information, such as recognizable events. One part of this thesis proposes a framework for streaming analysis of the time series, which can recognize events without supervision and memorize them by building the temporal contexts. The memorized historical data is then used to predict the future and detect anomalies. A new incremental clustering method is proposed to recognize the event without training. A memorization method of double localization, including relative and absolute localization, is proposed to model the temporal context. Finally, the predictive model is built based on the method of memorization. The "Edinburgh Pedestrian Dataset", which offers about 1000 observed trajectories of pedestrians detected in camera images each working day for several months, is used as an example to illustrate the framework.

Although there is a large amount of image data captured, most of them are not available to the public. The other part of this thesis developed a method of generating spatial-spectral-temporal synthetic images by enhancing the capacity of a current tool called DIRISG (Digital Imaging and Remote Sensing Image Generation). Currently, DIRSIG can only model limited temporal signatures. In order to observe general temporal changes in a process within the scene, a process model, which links the observable signatures of interest temporally, should be developed and incorporated into DIRSIG. The sub process models could be categorized into two types. One is that the process model drives the property of each facet of the object changing over time, and the other one is to drive the geometry location of the object in the scene changing as a function of time. Two example process models are used to show how process models can be incorporated into DIRSIG.

Acknowledgements

The Ph.D. study at RIT was a great experience for me. During the last few years, I was offered a lot of opportunities to learn and grow. I feel grateful to all the people I meet at RIT, who altogether makes RIT a warm place to stay.

First, I would like to thank my advisor, Dr. David Messinger, for his constant support, encouragement, and patience. He has been a wonderful advisor in balancing guidance and giving freedom to students, which allows us to work effectively and creatively. This work would not be possible without his mentoring.

I also would like to thank my committee members: Dr. Tony Harkin, Dr. Carl Salvaggio and Dr. Joel Kastner for taking the time to provide their support and give me valuable feedback on my work.

I would like to thank Dr. Michael Gartley for his kindness to teach me DIRSIG. Many thanks to Jason Faulring for helping me setting up the parking lot experiment. I thank Rolando Raqueño, Andrew Scott, and Nina Raqueño for their discussion, feedback, and support in the parking lot modeling in DIRSIG.

I would like to all the CIS admin staff, who makes things so well organized. In particular I'd like to thank Cindy Schultz, Sue Chan, Marilyn Lockwood, and Joyce French.

It would be very difficult for me to complete the Ph.D. study without the support from friends, roommates, and fellow graduate students. Special thanks to Dr. Weihua Sun, Lin Chen, Dr. Yujie Qiu, Wei Yao and Fan Jiang for their support and companion through the toughest first year study in U.S..

Most importantly, I would like to thank my family for being always supportive to whatever decision I made. I would like to thank my Fiancé Philipp for his caring, encouragement, and making me a better person.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

More than 150 Earth observation satellites are currently in orbit carrying sensors to monitor the earth and provide us a large number of valuable images[101]. Many companies and government agencies are still working on constructing next generation satellites. For example, the ESA is developing five new missions called Sentinels to complement the capacities of the existing satellites in the next several years[2]. Besides, there are many airborne sensors available to take images of certain sites when required. During the 2012 summer, RIT (Rochester Institute of Technology) performed a large scale experiment in Avon area of Rochester which is named as SHARE2012. Several types of airborne sensors (including WASP, ALS-60, ProSpecTIR VS, MircroHSI and PI Sensor[86]) flew over the site with targets set up in a designed way. The ground truth was measured and recorded along with the weather information. The collected data are shared around the world. The large amount of temporal digital satellite and aerial images calls for the corresponding development in data processing techniques to combine and fuse the temporal data from different sources in order to understand the data in a high level, such as extracting hidden events or activities. One part of the thesis will focus on building a generic framework to capture the temporal events from the data.

Although there is a large amount of image data captured and stored over time, most of them are not available to the public. The experiment event like SHARE2012

at RIT does not happen often, because such large scale experiments involve intense amount of efforts and time. Besides, a lot of the data do not offer the "ground truth", which is required to test and validate the algorithm. The other part of the thesis will focus on generating temporal synthetic images by enhancing the capacity of a current tool called DIRISG (Digital Image and Remote Sensing Image Generation).

In this thesis, we will illustrate temporal signature modeling in DIRSIG first to provide a method to generate the spatial-spectral-temporal synthetic remote sensing images. This could further aid the research in temporal signature analysis by offering the test data and their "ground truth".

1.1 Temporal Signature Modeling in DIRSIG

The DIRS lab (Digital Imaging and Remote Sensing Laboratory) at Rochester Institute of Technology has spent the past 20+ years developing the DIRSIG tool. Over the past 20+ years, DIRSIG has been involved with a great development starting from simplistic thermal image rendering of a 2D scene. Nowadays, the DIRSIG is a complex synthetic image generation model of 3D scene, which also is designed to produce broad-band, multi-spectral and hyper-spectral imagery through the integration of a suite of first principles based radiation propagation sub models. These sub models are responsible for tasks ranging from the BRDF (bi-directional reflectance distribution function) predictions of a surface to the dynamic scanning geometry of a line scanning imaging instrument. In addition to sub models that have been specifically created for the DIRSIG model, there are also components such as MODTRAN (MODerate resolution atmospheric TRANsmission) and FASCODE which are included in DIRSIG as workhorses for the multi- and hyper-spectral community. All modeled components are combined using a spectral representation, and the integrated radiance images can be simultaneously produced for an arbitrary number of user defined band passes. According to the surface temperature of the scene, the self-emitted radiances are calculated by a passive thermodynamic model which includes the time history of environmental and meteorological parameters[31]. Then

the surface reflected radiances are determined to compute the sensor reaching radiances through the included MODTRAN model. DIRSIG can produce multi or hyper-spectral remote sensing images between the bandpass 0.2 to 20μm with high radiometric fidelity[31, 92]. In addition, DIRSIG also produces per-pixel "truth", so many algorithm developers take this advantage of DIRSIG to validate their algorithms and make improvements.

However, except for vehicle moving, the solar and historical weather related short term temporal changing, DIRSIG does not currently include long term temporal signatures of the scene easily. To perform trade studies, algorithm training or even hypothesis testing, the user needs to manually create the scene with each individual element changing frame by frame as a function of time, which is time consuming and labor intensive. Take the scene of MCV (Midland Cogeneration Venture) Power Plant in Michigan for example: Figure 1.1 shows the WASP[73] images of the midland scene. By zooming in the center part of the scene, more details of the scene can be observed; there are various kinds of activities going on there, such as tanks with water filled in and released out, stacks releasing plumes, parking lots with cars arriving and leaving, a lake with surface temperature changing over the year and so on. In order to accurately describe the scene at a specific time, the user is required to manually set all characterizations for each scene element. For example, the user needs to figure out how much water is in the tanks and what is the temperature, which direction the plume would be blown and how strong the wind is, how many cars in the parking lot and how they are distributed, how the surface temperature of the lake looks like and so on. When time changes, the user needs to re-attribute all these properties to the scene element, which is a tedious process.

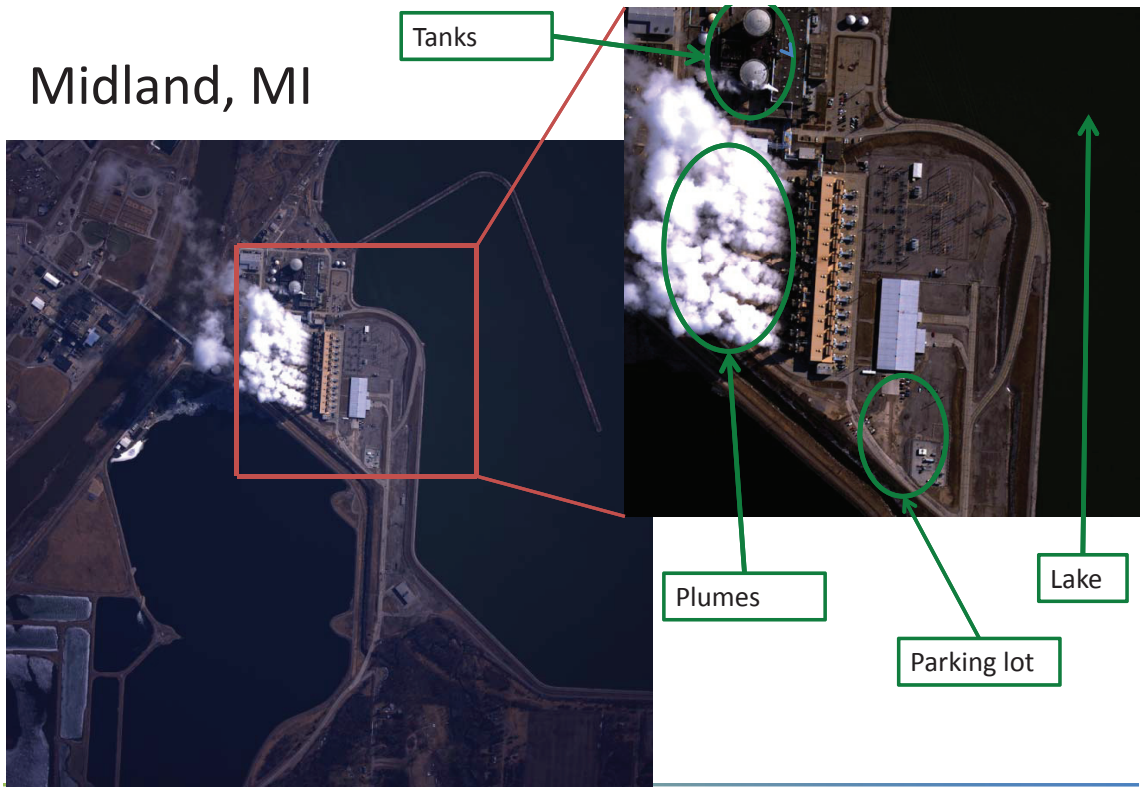


Figure 1.1: Midland Scene

GIS (Geographic Information System) is a spatial temporal system designed to store, manage, process, and visualize time varying geographical data. Over time, GIS uses snapshot model, space time composite, spatio-temporal object model, event-based/state-based model, object-oriented model, version-difference model and so on to represent the spatial temporal geographical data in the spatio-temporal database[100]. The dynamical property of GIS is shown in the database, and the representation of the relationship between the dynamic object elements in GIS is achieved through the management of so called relational database. Recently, the integration of process models and GIS is beginning to be realized and the proposed next generation GIS will use process models to govern the dynamics, adaption and evolution among the object elements in the system[102].

BIM (Building Information Modeling) is a intelligent model-based building design system, which incorporates the physical and functional characteristics into the

system. BIM which integrates the time information to the three spatial dimensions is often referred to as 4D BIM. 4D BIM uses a process model to direct the life cycle of a project by linking all the model elements in the construction schedule[3]. Each model element describes a discrete, time-driven construction activity as stated in the schedule. 4D BIM can facilitate the decision makers to learn a intuitive understanding of the process by visualizing the whole result consisting of different event phenomena from different model elements on the time axe.

In the field of computer graphics, a lot of work has been done on measuring and modeling the time-varying appearance of the natural phenomena[98, 44, 107]. Gu et al[44] develops a model called space-time appearance factorization to factor space and time-varying effects. Sun et al[98] measured the time-varying BRDFs of a wide range of phenomena with a self-developed acquisition system at a time sample space within 36 seconds and shared the database online. Those modeled natural phenomena include drying of various types of paints[107], wetting and drying of rough surfaces(cement, plaster and fabrics)[107, 44], the accumulation of dusts on surfaces[52, 107], corrosion and rusting of metals[44, 74], the weathering stone[34] and so on. These time-varying appearance of the different materials and surfaces are also interesting to remote sensing communities.

The first part of this thesis is intended to show the research of incorporating temporal signatures of the scene into DIRSIG, and these temporal signatures of the scene are driven by the process model. The enhanced DIRSIG could automatically create a scene with the property of each individual element driven by an external physical model as a function of time. The global process model could comprise many sub process models, each of which is designed to describe the temporal changing characterizations of the corresponding element in the scene. The changing characterizations of a scene element may include temperature, material property, geometry position and orientation, and so on. This research would enhance the ability of DIRSIG to simulate a complex scene with diverse activities and aid the algorithm development and test community to save significant and labor. By adopting the method of incorporating process models into DIRSIG, finally we expect to be able to create a scene which could capture not only spatial-spectral information,

but also include temporal information at that moment extracted from a "motion library", which is driven by a process model. Take Midland scene in Michigan for example, the process model could comprise an external physical two-tanks model to tell what the height and temperature in the two tanks are, a user defined plume model to control how the plume behaves and how the plume is affected by the weather, a statistically machine learned parking lot model to predict how the cars are distributed in the parking lot, the ALGE hydrodynamic model [53, 41, 42] to simulate the surface temperature of the lake and so on.

1.2 Temporal signature analysis

In this thesis, we will also consider the other way around of temporal signature modeling, which is temporal signature analysis. Still take Midland scene in Michigan for example, in this area Midland Cogeneration Venture (MCV) Power Plant is located. MCV is one of the largest gas-fired cogeneration plants in the United States, which produces 1,633 megawatts of electric power and additionally 1.5 million pounds per hour of process steam for industrial use[1]. With the large amount of electric generation, MCV produces some environment phenomena with interesting patterns varying along with the capacity of electric output of the plant. There are tanks, plumes, parking lots, and the lake as marked in the image in Figure 1.1. The observed phenomena of each scene element would vary with the different amount of the electric output. A larger amount of the electric output will correspond with a larger plume size, more fuel-gas stacks turned on, higher temperature of the lake surface, more water transferred through the tanks and probably more cars in the parking lot. All these observations can be considered as features which contribute to determine the event type going on in this area.

In the real world, almost everything is changing over time at different time scales. In many cases, we have real-time measurements of the scenario in different aspects to describe the current situation of the scenario. Significant research has been done in the field of time signature analysis in different applications[75][24][64][14]. In

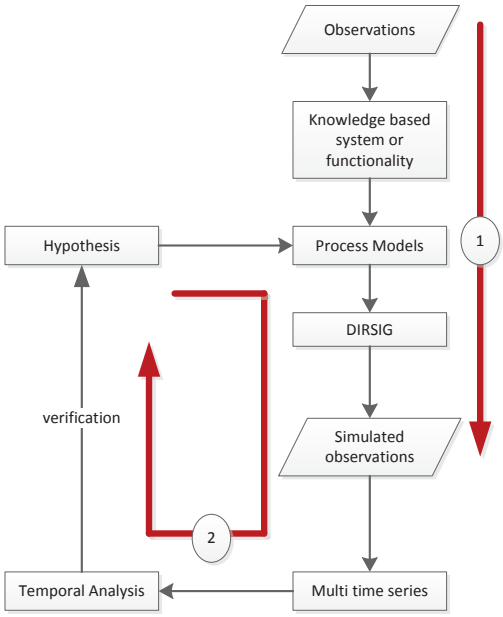
computer system security, system call sequences are analyzed to detect intrusions based on the temporal behavior of applications[55][49]. In medicine, longitudinal clinical records of patients are analyzed to discover the temporal pattern of the disease or medical knowledge[79][19]. In sociology, there is interesting research done to extract events, such as sporting events and earthquakes from Social media according to the temporal activities of users[72][116][84]. In remote sensing, temporal satellite images are analyzed to find model the dynamics of vegetation, land cover, ecological processes and so on[28][30]. Additionally, temporal signature analysis is also the main research topics of gesture recognition, speech recognition, on-line signature recognition and so on. However, to our knowledge, there is not a general framework built to illustrate the chain from recognition, modeling of the historical data (the term 'memorization' is used in this thesis to describe the process of modeling the historical data), to prediction of streaming data. In this part of the thesis, a framework will be built to learn events and their temporal contexts from the continuously collected data. The learned knowledge from the collected time series can then be used to make predictions and detect anomalies.

The "Edinburgh Informatics Forum Pedestrian Data set", which offers about 1000 observed trajectories of pedestrians detected in camera images each working day for several months, will be used to illustrate the framework. The proposed framework will implemented from recognizing the situation (event type) in the scene during prescribed time interval over time by twice applications of a new incremental clustering method, to memorizing the historical events by a temporal map as absolute localization and the Markov chain model as relative localization of an event, and to final prediction and anomaly detection with the predictive model built based on the method of memorization.

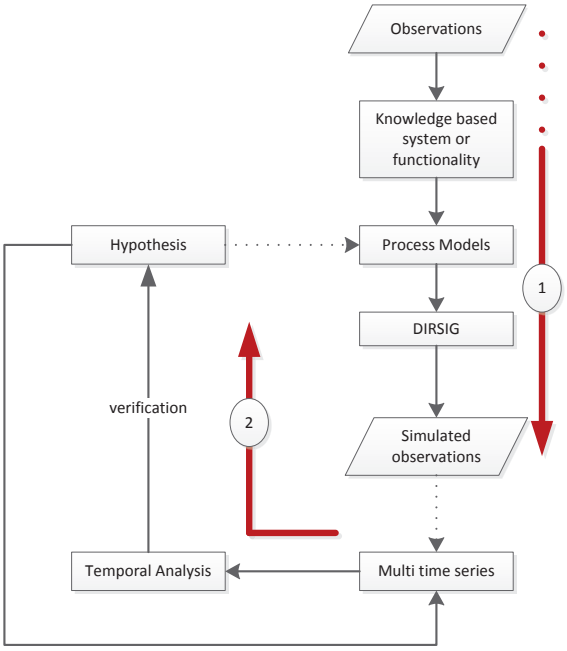
1.3 Summary

In summary, in the thesis, two parts of research are to be performed. The first part is to incorporate the process models into DIRSIG to generate simulated remote

sensing images in spatio-spectral-temporal spaces. The second part of the thesis will analyze the time series in order to recognize and predict events. Figure 1.2a shows the big background of the thesis, while Figure 1.2b shows the two problems will be solved in this background.



(a) Big background of the thesis



(b) Problem structure of this thesis

Figure 1.2: Problem structure

Chapter 2

DIRSIG Simulation

In this chapter, a detailed description of DIRSIG will be firstly stated to show some basic understanding and knowledge of how DIRSIG could be possibly extended in functionalities. Then, the timing mechanism of DIRSIG is investigated in order to show in which way the DIRSIG can be enhanced in the temporal dimension. Lastly, possible process models which can be incorporated into DIRSIG are listed.

2.1 The understanding of DIRSIG

DIRSIG is a integration of sub-models[85]. The sub-models include a scene sub-model, ray tracer sub-model, thermal sub-model, radiometry sub-model and sensor sub-model. There are several databases in DIRSIG to support the simulation. The databases are for material, weather, and atmospheric propagation. In order to fully understand DIRSIG, the simulation will be explained in two perspectives. First, the first principle working mechanism between the sub-models of DIRSIG is stated to show how rays travel through the scene to the sensor to form an image. Then we look at DIRSIG based on the files which DIRSIG needs and generates to accomplish a scene simulation. By doing this, we could understand what information DIRSIG needs to drive the simulation and how the capacity of DIRSIG could be enhanced in terms of incorporating temporal signatures into the scene simulation.

2.1.1 First principles based working mechanism of DIRSIG

DIRSIG is a first principles based synthetic image generation model by integration of a suite of the first principle based sub-models. Figure 2.1 shows the interaction between sub-models and data bases.

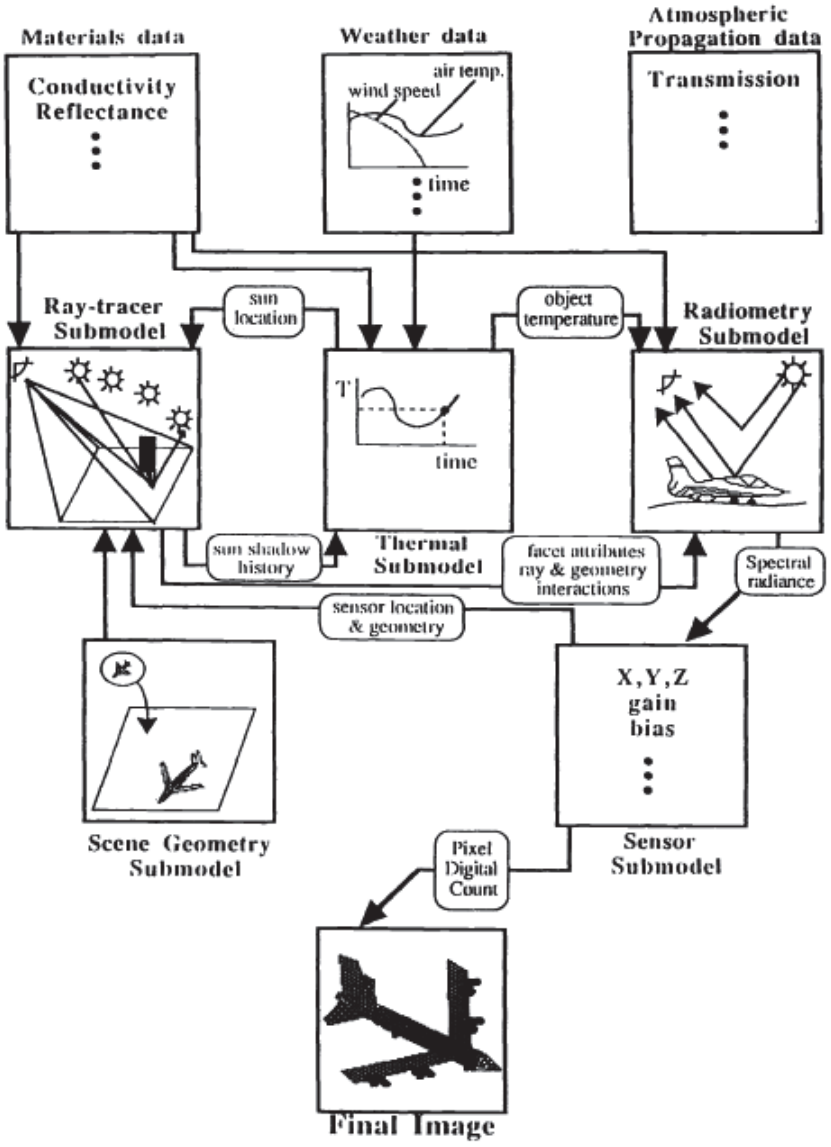


Figure 2.1: Interactions between sub-models in DIRSIG[85]

To simulate an image, DIRSIG starts by looking at the scene through the sensor sub-model. The ray tracer sub-model is adopted to send out a ray into the scene through each pixel of the image from the sensor sub-model. The scene is constructed through the scene sub-model, which represents the scene with facets. Each facet uses a set of information describing its own properties which include the coordinates of the facet vertices, zenith and azimuth angle, normal vector, the material, temperature calculation method and facet thickness.

When the ray hits on a facet of the scene, it reads the properties of the facet. If the temperature calculation method is set to invoke the DIRSIG internal thermal model to work, the facet properties are feed into the thermal sub-model; otherwise, a temperature can be set by the user externally. Aside of the facet properties, the thermal model also needs the weather data, the current solar load and solar history of the pixel. The weather data could come from forecast data or from a measured data record. In order to get the information of the current solar load and solar history of the pixel, rays are sent out from the intersection point on the facet in the direction of the sun starting at the current time back to the previous 24 hours in a certain time interval. Then the status of whether the facet is blocked by other objects is determined. With all the information about the facet, the weather and solar load and history, the facet temperature could be calculated through the thermal sub-model of DIRSIG.

With the knowledge of the facet temperature, facet orientation, facet material property, solar position, background and atmospheric data, the radiometry sub-model could calculate the spectral sensor reaching radiance according to the "big equation"[91]. As shown in figure 2.2, the radiometry sub-model considers (A) the reflected solar radiance on the target, (B) the reflected solar scattered down-welled radiance on the target, (C) the solar scattered upwelled radiance, (D) the self emitted spectral radiance from the target, (E) the reflected down-welled spectral radiance on the target due to the self emission of the atmosphere integrated over the skydome, (F) the upwelled spectral radiance due to the self emission of the atmosphere, (G) the reflected solar radiance of the background on the target, and (H) the reflected radiance on the target due to the background self emission. The parameters of the

radiometry sub-model such as spectral transmission, emission and scattering are dependent on the atmospheric conditions. MODTRAN4 or MODTRAN5 is used to characterize the atmospheric propagation for 0.2 to 100 μm spectral range in a spectral resolution of 0.2 cm^{-1}. Fascode is an alternative atmospheric propagation model when a higher spectral resolution of the image is required. The output from the atmospheric model facilitates the radiometry sub-model to calculate the sensor reaching radiance.

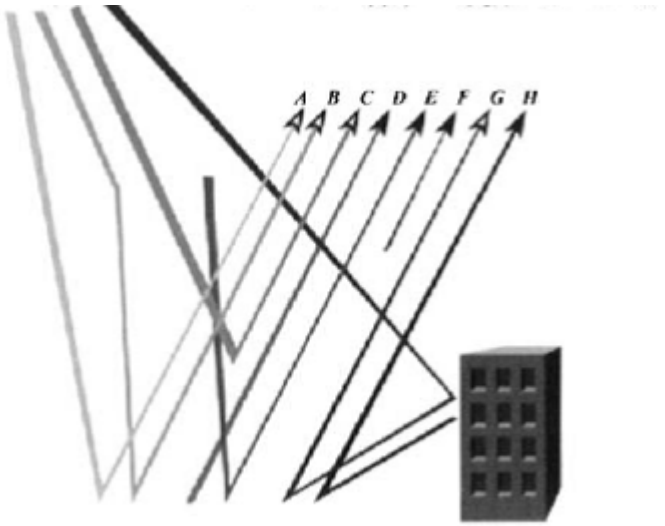


Figure 2.2: Sensor reaching radiance[91]

The sensor reaching radiance map is then passed to the sensor sub-model. The sensor sub-model performs postprocessing on the radiance map to make the image look "real" by introducing geometry distortions, motion blurs, noises, sampling effects and so on.

DIRSIG4 also includes polarization modeling and LIDAR modeling. When the polarization modeling is enabled, MODTRAN-P which is a polarized version of MODTRAN4 will be used to aid the radiometry sub-model to produce polarized radiance in terms of spectral stokes vectors and a polarization angle. This requires knowledge of the polarization properties of the the material. The capacity of 3D

LIDAR imaging in DIRSIG is achieved by using a different ray tracing method from the existing one. The technique is called photon mapping which is a two-pass method[54]. First the photon map structure is built by tracing photons through the model. Then the result is rendered by using the information in the photon map via the photon density based radiance estimation. The model would predict the returned fluxes from the scene as a function of time with respect to the shooting of the source laser.

2.1.2 Input files of DIRSIG

DIRSIG release 4.2.0 features Extensible Markup Language (XML) input formats[33]. Most programming languages support reading and writing of XML file, which allows new features and capabilities to be easily added to DIRSIG. The input files are separated into five XML input files, which are collected into a simulation manifest (.sim) file, as shown in Figure 2.3. The five XML input files are .scene file, .atm file, .platform file, .ppd file and .tasks file. The input files supply the information to the sub-models of DIRSIG to run the simulation. Each XML input file contains one type of information of the whole simulation. Then each sub-model reads the information among the five XML input files according to what it needs.

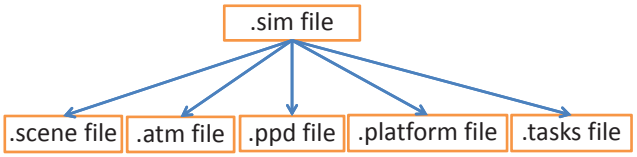


Figure 2.3: DIRSIG XML input files

The scene sub-model needs the 3D geometry of the scene, the property of each facet in the scene, and geolocation of the scene, which can be obtained from .scene file. The thermal sub-model needs the facet property, environment weather, atmospheric conditions, the time of the day, the geolocation of the object, and so on which can be obtained from .scene file, .atm file and .tasks file. The radiometry sub-model needs the facet property, atmospheric conditions, the time of the day, the

sensor position, environment weather which can be obtained from .scene file, .atm file, .ppd file and .tasks file. The ray tracer sub-model connects the other sub-models of DIRSIG and it therefore need to read inputs from all five XML files.

2.1.2.1 The .scene file

The XML .scene file is used to carry the information of the scene, as shown in Figure 2.4 including the geodetic location of the scene, the geometry structures (.gdb file) and material property of the objects in the scene (.mat file), the distribution of the objects in the scene (.odb file), the property maps of the scene (map list) and the landmark data. Besides, currently the parameters which drive the plume model are also contained in the .scene file.

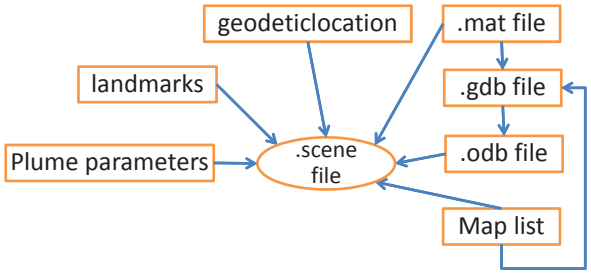


Figure 2.4: The .scene file

DIRSIG uses a .gdb (Geometric Database) file to describe the geometry structure of the object and the corresponding properties of each facet in the object. The detailed format is described in the Table 2.1 DIRSIG User's Manual. DIRSIG .gdb file can only be read by DIRSIG and DIRSIG related tools such as bulldozer and Blender. The .gdb file is a core part of the simulation model which contains rich information about the objects in the scene. The .gdb file forms a hierarchical structure of objects, parts and facets, which is created to aid the first principles based image simulation. The .gdb file could be obtained by importing a wavefrom OBJ file through tool bulldozer in "Object Mode". Through bulldozer, the attributes of the objects can be assigned manually, which is then saved as a .gdb file. Recently, the python interface code is developed in Blender to read and work with .gdb file.

However, on the other hand, the .gdb file can be viewed as txt file, which can be written through scripts in the designed format as shown in Table 2.1. For the purpose of generating the scene automatically, the .gdb file is considered as a txt file, the parameters of which can be driven by external physical model through scripts.

The objects are distributed in the scene through the .odb (Object Database) file, which is listed in the .scene file. An example below shows the basic format. The .odb file contains a series of OBJECT entries, each entry include the .gdb file with the detailed location and file name. Each OBJECT element also contains a UNITS element which describe the physical units of the geometry file. The INSTANCES element in each OBJECT element indicates the information of the location, the scale factors and the rotation factors of the object described in the .gdb file relative to the scene center. Like the .gdb file, the .odb file can also be obtained through the tool bulldozer or Blender by importing the .gdb file in "SCENE MODE" and save the refined .gdb file in .odb format. It can also be considered as a txt file and edited through external scripts.

```
OBJECT {
        GDB_FILENAME = .\gdb_odb\ground.gdb
        UNITS = METERS
        INSTANCES {
            INFO = 200 0 0 2.5 2.5 1 0 0 0
        }
}
```

Besides, the file of the material property (.mat file) is also listed in the .scene file. As mentioned before, in the .gdb file, each facet is assigned with a material property which is achieved by using a material ID. This material ID is then used to find the corresponding entry in the .mat file. Each material in the .mat file has an entry ID. Along with the entry material ID, the detailed optical and thermal properties of the material are described, such as specific heat, mass density, thermal

Table 2.1: Example GDB file

OBJECT	KEYWORD: OBJECT
TRUCK_OBJ	Name of object
1-0-0	Object ID
PART	KEYWORD: PART
TRUCK_BACK1_ATTS	Name of part
1-1-0	Part ID
FACE	KEYWORD: FACE
TRUCK_BACK1_1	Name of facet
1-1-1	Facet ID
painted_steel_side	Name assigned to this facet
27	ID assigned to this facet
truck_back1	Facet name [no longer used]
-1.0	Facet temperature [C] (-1.0 means computed by THERM)
0.01	Facet thickness [cm]
0.0	Facet self-generated power [no longer used]
0.0	Facet exposed area [no longer used]
null	[unused field]
null	[unused field]
null	[unused field]
4	Number of vertices (can be 3 or 4)
0.000000 152.400000 30.480000	Vertex #1 Coordinates
0.000000 0.000000 30.480000	Vertex #2 Coordinates
53.340000 0.000000 30.48000	Vertex #3 Coordinates
53.340000 152.400000 30.48000	Vertex #4 Coordinates
0.000000 0.000000 -1.000000	Normal vector
180.000000	Zenith (slope) angle [degrees]
0.000000	Azimuth angle [degrees]
0.000000	[unused]
FACE	Another facet
TRUCK_BACK_12	
1-1-12	
PART	Another part
TIRE_1	
1-2-1	
OBJECT	Another object
ROAD_1	
2-1-1	
END	KEYWORD: END (Last line in file)

conductivity, solar absorption, thermal emissivity, exposed area, thickness, spectral emissivity file (.ems), specularity and so on.

DIRSIG also offers the property mapping functionality. If the user has a property map available to characterize the whole scene, the map could be included in the .scene file. The map could be used to describe the texture, temperature, material, radiance, reflectance or varying surface normals. In each map element, there are variables such as the file name of the map, the insert point of the map in the scene, the material ID with which the map will be associated, the GSD and so on. The association between the map and the geometry is achieved through matching the material ID stated in the map element and the material ID assigned in the .gdb file.

2.1.2.2 The .atm file

As the name tells, the .atm file includes all the information about the atmosphere.

In order to supply at least 48-hours of weather data to make the simulation more reliable, a weather history file (.wth) is contained in the .atm file. In the .wth file, the weather information such as air temperature, pressure, relative humidity, dew point, wind speed, direct insolation, diffuse insolation, sky exposure, cloud type, precipitation type, precipitation rate and precipitation temperature is listed in each row at each previous (back to 48 hours) relative time to the simulation time stamp. The .wth file could be obtained through the tool make_weather by inputing the file name, location (latitude, longitude), time (month, day, year), time offset from GMT, peak insolation, average transmission, diffuse insolation, air temperature at sunrise, peak air temperature, local time of peak air temperature, air pressure, dew point temperature, wind speed, sky exposure factor, cloud type, precipitation information (type, rate, temperature).

There are four standard atmosphere model settings available in DIRSIG, through which is also assigned in the .atm file. They are simple, uniform, classic, and threshold atmosphere model respectively. For the simple atmosphere model, only the apparent sky temperature is required to be known to make a very simple scene simulation without considering the complex atmosphere conditions. For the uniform

atmosphere model, the spectrally constant hemispherical irradiance and sky fraction is required to be known. For the classic atmosphere model, modtran is called to predict atmospheric properties by inputing the tape5 file. DIRSIG then extracts the result from the output tape7.scn file which is then saved as an .adb (Atmospheric Database) file through tool make_adb. The .adb file, which is included in the .atm file, is a look up table containing three sections: source paths section, sensor paths section, down-welled path section to describe the irradiance of the sources and the transmission, scattered radiance and emitted radiance within the atmosphere. For the threshold atmosphere model, as the classic atmosphere model, modtran is also called to predict the atmospheric properties with high fidelity. However, the .adb file is not precalculated and listed in the .atm file. Instead, the threshold atmosphere model uses a series sampling parameters to set modtran to render the atmosphere properties at certain spatial and temporal points.

Therefore, for different purposes of simulation, a different atmosphere model is used with different input files into the model. The figure 2.5 shows the configuration of the .atm file.

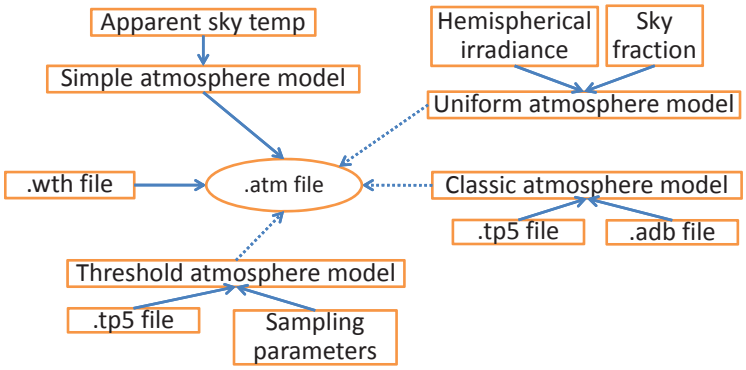


Figure 2.5: The .atm file

2.1.2.3 The .platform file

In the .platform file, more than one instrument can be mounted. The instrument can be a generic passive sensor, a mono-static LIDAR, a bi-static LIDAR source, a bi-

static LIDAR receiver or a data recorder. For a passive sensor, the user is required to tell the .platform file the focal length and the focal plane settings which include the clock, the geometry of the sensor, the response function and the truth map requested. For a LIDAR instrument, the information about the clock, the transmitter, the receiver and the output format should be included. Relevant information should be known if any other type of instruments is used. The instrument mount can be configured in the .platform file to have different types of scanning. There are six scanning methods available in DIRSIG. They are Static/Fixed scan, line scan, whiskbroom scan, lemniscate scan, tabulated scan and scripted scan. For each scanning method, the .platform file asks for quantitative descriptions about the mount, such as the rotation angle and the jitter effects.

2.1.2.4 The .ppd file

The platform positioning data(.ppd) file contains the information of the position and orientation of the platform as a function of time. In each "entry" element of the .ppd file, the scene location and the rotation angles of XYZ axis in radians are needed at each relative time stamp (with respect to the main simulation time). Besides, the jitter of the location and orientation variables can be set in the .ppd file to associate the real world uncertainty as a function of time. By including the time dependent platform positioning data, DIRSIG could take snaps from different perspectives at different time according to the requirement.

2.1.2.5 The .tasks file

The .tasks file offers the information of the absolute simulation time to DIRSIG. If multiple tasks are requested at discrete time periods, there is a task element for each period with an assigned start time and an assigned stop time. During each task period, DIRSIG takes snaps in the same frequency as the clock set in the sensor instrument through the .platform file.

2.2 Timing in DIRSIG

As the statement about DIRSIG above shows, DIRSIG is an image simulation tool reaching out in the spectral, spatial and temporal spaces. The model is inherently consistent in the spectral and spatial spaces between different functionalities because of the first principle based calculation. For the temporal aspect of modeling, DIRSIG already has built a general structure of time line which goes through different facilities and make connections between them as shown in Figure 2.6. At each time of the simulation, there are many "timing seeds" carrying the information of time and spreading it through out the simulation. Each time dependent input file of DIRSIG should get the "timing seed" to locate its coordinates in the temporal space.

The five XML input files of DIRSIG could be all time dependent which is because of the time dependent sub models inherited in DIRSIG. DIRSIG incorporates a time dependent thermal model named THERM[31] to calculate the facet temperature. The THERM model is invoked when the facet temperature is set as -1. Then the THERM model computes the facet temperature based on the material thermodynamic properties and environmental weather conditions from the .wth file. The calculated temperature is feed back to the facet of the scene. At different time of the day, the surface temperature of the facet calculated from the THERM varies, which results in a different simulated scene image. Therefore, the .scene file is informed of the time through the "timing seeds" in terms of the temperature only affected by the factors characterized in THERM model. Besides, DIRSIG uses Modtran to characterize the time dependent profile of the atmosphere with the .tp5 file. Therefore, the time seeded .atm file could also be obtained. Due to the time dependent sub models inherited in DIRSIG, the sensor reaching radiance is changing over time. And at different times, the requirement of sensor settings may vary and the perspective of the sensor may also change. In addition, as mentioned above, the uncertainty of the platform position data could change as a function of time. Therefore, the time seeded .ppd file and .platform file are also required. Since the .tasks file is directly connected with the simulation time, it gets the timing seeds as well. With the five

time seeded XML files, DIRSIG runs the simulation and renders an image with the "timing seeds".

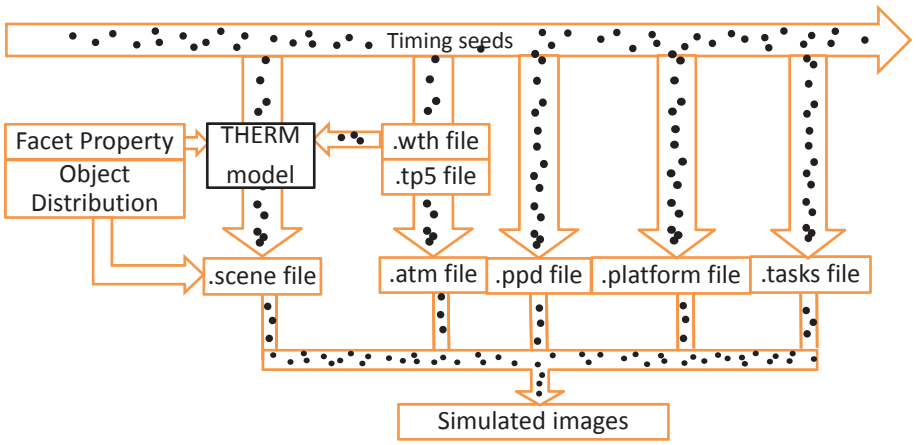


Figure 2.6: Primary timing in DIRSIG

However, as noticed from Figure 2.6, there are two blocks of information that drive the THERM model and the .scene file which have not got the "timing seeds". The two blocks of information describes the facet property and the geometric distribution of the object in the scene, and they are actually time dependent and need the time stamp to obtain the corresponding information. There are two possible results if the user performs the simulation through current design of DIRSIG. One is that DIRSIG will render an unmatched simulated image of the scene if the property of the two blocks is changed. The other is that the user needs to manually offer the correct property of the facet and the geometric distribution of the object at each desired simulation time.

To avoid the two negative results as mentioned above to occur, the process model is proposed to drive the facet property and the object distribution in the scene changing over time by bringing the "timing seeds" to them as shown in Figure 2.7.

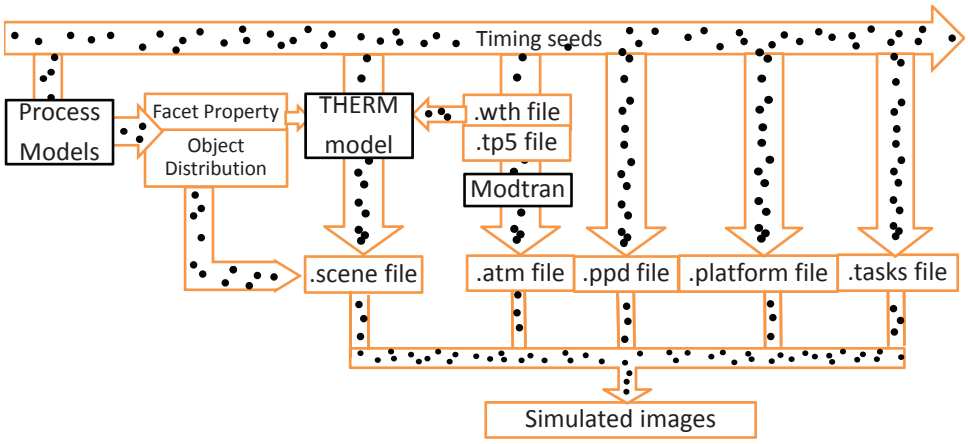


Figure 2.7: Developed timing in DIRSIG

2.3 Process models

The process model is used to describe what desired processes should be performed over time. At each moment, the process model produces a corresponding state of the object. All the states through the time are combined as a process. Any change of an object could be described through a process model which may be deterministic, stochastic or rule based[46] according to the characteristics of the process. The process model can include a set of sub process models, each of which predicts how the corresponding scene element changes over time. There are rich varieties of scene objects with various possible changes over time, which requires corresponding sub process models to characterize all the process.

The surface temperature of the water body of a lake may change due to the weather and human activities. ALGE[41] is a 3D hydrodynamic model which solves momentum, mass and energy conservation equations to predict the surface temperature of a water body. By assigning a time stamp and related conditions to ALGE, it could generate corresponding surface temperature map of the lake.

The traffic in the city can vary as a function of time as well. The open source tool SUMO (Simulation of Urban MObility[60]) is a traffic process model developed by German Aerospace Centre and Centre for Applied Informatics Cologne. A vehicle is

tracked individually in SUMO with its identifier, departure time, and route through the road network. The velocity and position of the vehicle is calculated using a so-called car-following model according to the state of the vehicle in front of it to avoid a collision. The type of the vehicle can also be set to have a corresponding driving characteristics on the road. Any number of vehicles can be defined in SUMO to make the simulation of large scale scenarios possible. It can be incorporated into DIRSIG through the temporal linkage of the "timing seeds".

MuSES[32] developed by ThermoAnalytics is commonly considered as a standalone thermal signature prediction tool for vehicles, which could used to describe the temporal changing of the temperature of a vehicle. MuSES also offers a plume radiance module, sea surface module, battery module and so on to provide the temporal thermal signatures[4].

The distribution of cars in a parking lot can vary during the day and over the week. Also a special event will result in a different situation in the parking lot. A parking lot model PARKVIEW developed during this project[99] can be used to describe the temporal signature of the car distribution in the parking lot.

Gartley, et al [43] developed a microDIRSIG model to predict contaminated surface properties. And the time-varying appearance of the natural phenomena[98, 44, 107] modeled by the computer graphics community can be transplanted into DIRSIG to aid the scene simulation for the remote sensing community.

Besides, the vegetation on the ground can be influenced by the weather or human activities, which will result in a different texture and material map of the ground. A fire will change the appearance of the forest and the extent of damage will be different at different observation time. The distribution of the crowd is different from the start of an event to the end of an event. All the changes can be represented through process models.

Chapter 3

Temporal Signature Modeling in DIRSIG

Temporal signature modeling in DIRSIG will be achieved by incorporating process models into DIRSIG. Then the process model will drive the property of the scene element changing as a function of time. As explained in Chapter 2, the DIRSIG input are represented in five XML files. The motivation of this research is to use a process model to automatically drive the input files changing as a function of time, so that each scene element could know its own properties at an assigned time and the rendered DIRSIG spatial-spectral images could also contain temporal signatures.

3.1 Work flow

Before building the work flow of incorporating the process model into DIRSIG, the sub process models used to describe the activities of each scene element are assumed to be ready for usage. These sub process models could be user defined or external ready-made functionality, which could suitably be fitted in to describe the properties of the scene element changing with time. Figure 3.1 is the work flow chart showing how the process model would work with DIRSIG to generate spatial-spectral-temporal images.

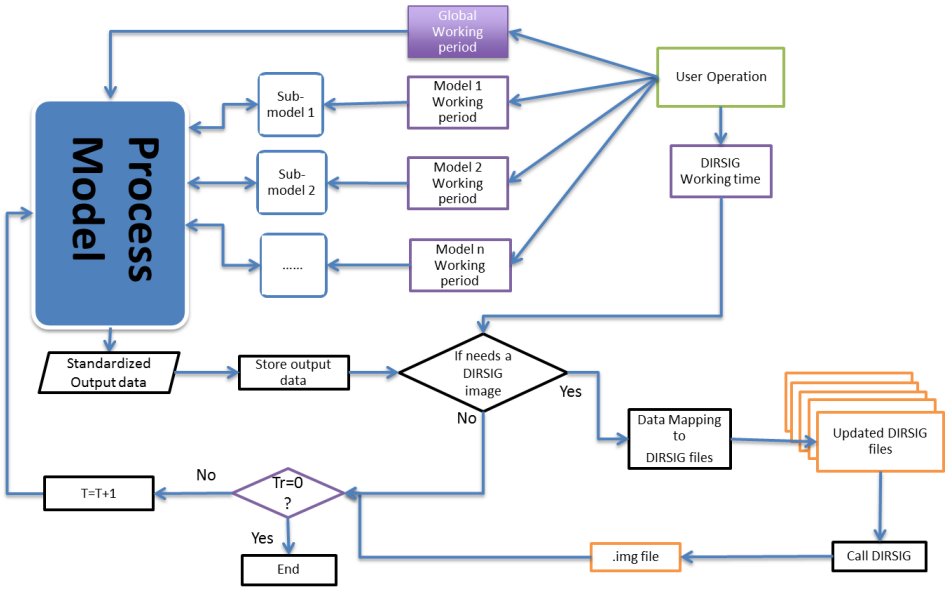


Figure 3.1: Work flow chart of the process model

First, start from user, where the user sets the initial working time period for the global process model and each sub process model by considering what is interesting and significant. For example, at night there may be very few cars in the parking lot; therefore the parking lot process model may not be asked to work during night to reduce the computational load. At the same time, users would also decide when they want DIRSIG to take a snap shot of the scene to observe the activities going on.

Second, the sub process models of the overall process model work together to generate a set of standardized output data. During this step, the output data are required to be written in a standard format, so that the meaning of each set of data could be understood by the computer. This data standardization could be achieved by writing the output data in a model defined format or plotting a diagram to indicate linked pairs of the variables between process models and DIRSIG.

Third, the standardized output data are stored in a database. The user then can review all the output data sets from external process models and have a general idea what is happening during the defined time period. The stored data could be used to diagnose or validate the algorithm for the developer, which acts like the DIRSIG

ground truth.

Fourth, by looking at the DIRSIG working time table, a check is made to determine whether a DIRSIG image is needed at that moment. If yes, the standardized output data are mapped to the corresponding variables of the DIRSIG input files, which is a key step to achieve the incorporation of the process model and DIRSIG. The method of data mapping will be explained in detail below. After mapping the data into DIRSIG input files, at the same time DIRSIG input files could be updated; then DIRSIG can be called to generate a physics-based simulation of the scene. If DIRSIG images are not needed at that time, go to the fifth step.

Fifth, check whether all the tasks required by the user have been finished. This is determined by whether the process model has finished generating output data for the whole predefined working time period. If yes, the whole model is complete. If not, the process model keeps working for the next time step until the whole time period is filled out.

As described in section 2.3, there are various kinds of process models. How would all these process models be incorporated into DIRSIG in a general format? To solve this question, the inputs of DIRSIG used to describe the property of an object in the scene are reviewed briefly again here. In DIRSIG, the object uses the .gdb file containing sub facets to describe its geometry shape and physical properties in a desired resolution. The object is distributed through the .odb file into the scene. Therefore, all attributes of the object including facet properties of the object and the geolocation of the object could be described by the .gdb and the .odb file. Due to this specific input design of DIRSIG, the sub process models could be categorized into two types. One is that the process model drives the property of each facet of the object changing over time, and the other one is that the process model drives the geometry location of the object in the scene changing as a function of time. The first type of process model would involve many-to-one data mapping to get incorporated into DIRSIG. An example process model of this type will be shown in section 3.2. The example process model is a two-tanks thermal model, which will drive the surface temperature of the two tanks changing as a function of time during the DIRSIG simulation. While the second type of process models could be

incorporated into DIRSIG through one-to-one data mapping. An example process model of this type will be shown in section 3.3. The example process model is a parking lot model named PARKVIEW, which drives the distribution of cars in the parking lot changing over time during the DIRSIG simulation.

3.2 Process models driving per facet property changing

In this section, we will show how the process model which drives the facet property changing is incorporated into DIRSIG.

DIRSIG uses OBJ file format to describe the solid geometric surfaces. Wavefront OBJ file is a geometry definition file format which carries the 3D geometry information of an object, such as the position of each vertex, facet index combinations, normal vectors of each facet, texture vertices and so on. The size of each facet of the OBJ file is mainly determined by the smoothness of the surface and the resolution of the geometry. A DIRSIG gdb file is written facet by facet associating the geometry with facet properties which includes temperature and material properties. Each facet has a unique property value for the temperature and the material. Suppose a process model should generate a high resolution characterization map to describe the change occurring at each time. Then the high resolution characterization map needs to be mapped onto the low resolution DIRSIG geometry. This means that each facet could possibly have more than one corresponding property values. However, one variable in DIRSIG input files could only be assigned with one value. In order to include the entire information from the characterization map, the facet is split or up sampled into sufficient number of facets, as shown in Figure 3.2.

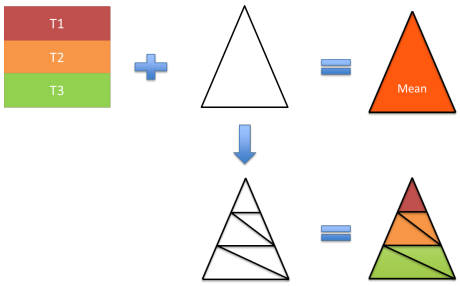


Figure 3.2: Many-to-one Mapping

The UV mapping technique is adopted to relate the high resolution characterization map to the low resolution geometry and render the DIRSIG gdb file which carries all the information from the characterization map. The high resolution characterization map is driven by the external process model. In this section, an example two-tanks thermal process model will be created to generate the high resolution temperature maps of the two tanks at different time.

3.2.1 UV mapping

Given characterization maps, the OBJ file will be then transformed into a .gdb file to carry thermal and material properties aside from geometry information for each facet as follows. First, the geometry information, including coordinates of vertices and facet index combinations, are read from OBJ files and stored in matrices. The next step is to up sample the geometry. A straight-forward way to up sample the geometry is to add the same number of vertices on each edge of all the facets evenly. However using this method to increase the geometry resolution would introduce many redundant vertices with unnecessary information carried with the gdb file. Take a tank with hot water for example. We would like to consider more about how much hot water is in the tank and what is the temperature by observing the temperature profile of the outside of the tank. Ideally the temperature is uniform horizontally and only differs vertically, so a temperature map with $n \times 1$ dimensions could be enough to include the temperature and height information rather than the one with $n \times n$ dimensions, where n is determined by the defined resolution. Therefore, adopting the above method of up sampling geometry evenly in different

direction is not economical.

Here a method of up sampling the geometry in two different resolutions for two directions is proposed. Before explaining details of this method, the method of how the characterization map is covered upon the geometry structure should be mentioned first.

- Dropmap: if a top-view characterization map is accessible, it would be covered upon the object in a dropping motion. For those facets which are vertical to the horizon, the characterization is set as default or defined as "continuous" with the same value as the nearest facet which is not vertical.

- Wrapmap: if a round-view characterization map is accessible, it would be wrapped around the object in "wrapping" motion. For those facets which are horizontal, the characterization is set as default or defined as "continuous" with the same value as the nearest facet which is not horizontal.

Figure 3.3: An example image to be mapped onto geometry ("cameraman.tif" with size 256×256)

The coordinates of the vertices of the object read from the OBJ file are written in a matrix as $[\mathbf{X}, \mathbf{Y}, \mathbf{Z}]$, and $\mathbf{X} = [x_1, x_2, \ldots, x_i, \ldots, x_m]^{'}$, $\mathbf{Y} = [y_1, y_2, \ldots, y_i, \ldots, y_m]^{'}$,

$\mathbf{Z} = [z_1, z_2, \ldots, z_i, \ldots, z_m]^{'}$, where m is the number of vertices. The minimum and maximum values of x_i, y_i, z_i are represented as x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max} respectively. Assume a map with sampled size $pn \times pm$ would be mapped to a geometry object in two different mapping methods, dropmapping and wrapmapping respectively. Here, we use "cameraman.tif" as shown in Figure 3.3, which could then be sampled into different sizes to demonstrate the result of data mapping.

For the dropmapping method, to up sample a facet with vertex v1(x_1, y_1, z_1), v2(x_2, y_2, z_2), v3(x_3, y_3, z_3) as shown in figure 3.4, the up sampled geometry resolution in x and y axis is calculated as

$$\triangle x = \frac{x_{max} - x_{min}}{pm} \tag{3.1}$$

$$\triangle y = \frac{y_{max} - y_{min}}{pn}. \tag{3.2}$$

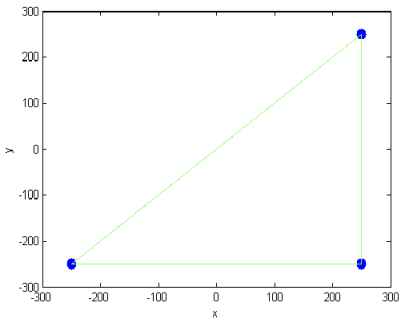


Figure 3.4: Initial facet with 3 vertices

With the calculated geometry resolution in x, y direction, the up sampled points on the facet in xy plane are found as shown in Figure 3.5.

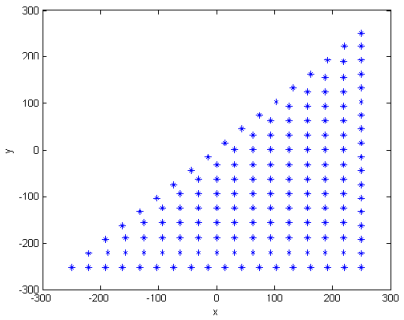


Figure 3.5: Up sampled points on a facet in xy plane for drop-mapping method

A plane with three known facet vertices can be represented by

$$n_x(x - x_1) + n_y(y - y_1) + n_z(z - z_1) = 0, \tag{3.3}$$

where (n_x, n_y, n_z) is a normal vector of the facet, and can be calculated by cross product of vector $(v2 - v1)$ and vector $(v3 - v1)$.

Therefore, the z value of all other points on the facet can be found by

$$z = \frac{-(n_x(x - x_1) + n_y(y - y_1))}{n_z} + z_1. \tag{3.4}$$

So far, we have finished up sampling the geometry structure of a facet when dropmapping method is set and all coordinates of the refined vertices are found. The UV mapping technique is used to achieve the linkage between the refined vertices of the object and the pixel value of the map.

In order to adopt UV mapping technique, first, the left down corner of the map is registered to the object with coordinates in xy plane as [x_{min},y_{min}]; the right up corner of this map is registered to the object with coordinates in xy plane as [x_{max},y_{max}]. Then, x,y coordinates of all vertices are scaled into [0,1], resulting in uv values calculated as

$$u_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{3.5}$$

$$v_i = \frac{y_i - y_{min}}{y_{max} - y_{min}}. \tag{3.6}$$

With the uv values, all the pixel values of the map could be linked to the corresponding facet of the object, as shown in Figure 3.6. Figure 3.7 shows a higher resolution map mapped to both facets of the same plane.

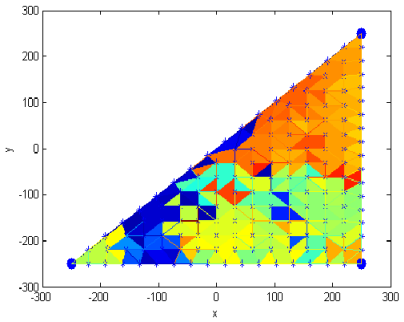


Figure 3.6: Up sampled facets with UV mapped information to half the plane

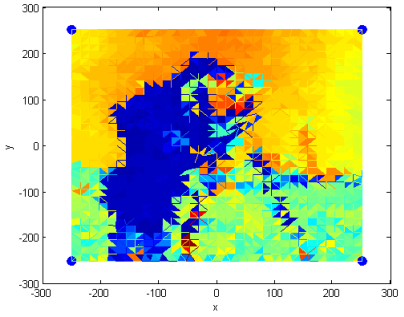


Figure 3.7: Up sampled facets with UV mapped information in higher resolution to the full plane

For wrapmapping method, we use a cylinder OBJ file as an input as an example to show steps of this method. The geometry structure is shown in Figure 3.8.

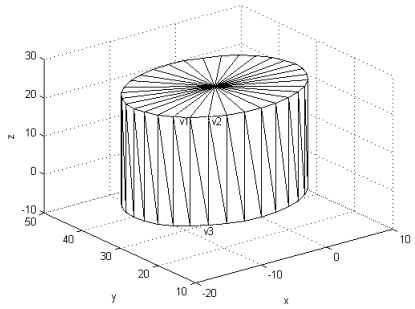


Figure 3.8: Cylinder.obj

To up sample a facet with vertex v1(x_1, y_1, z_1), v2(x_2, y_2, z_2), v3(x_3, y_3, z_3) with wrapmapping method, we first transfer the coordinate system to a new Cartesian system so that its corresponding cylindrical coordinate θ could range from 0 at point [x_{min},y_{min},z_{min}] to 2π at point [x_{min},y_{min},z_{max}] and keep z the same as in the old coordinate system. In the new cylindrical coordinates, we have v1(θ_1, ρ_1, z_1), v2(θ_2, ρ_2, z_2), v3(θ_3, ρ_3, z_3) for the above facet. Then the facet is up sampled with steps $\triangle\theta$ and $\triangle z$ defined as

$$\triangle\theta = \frac{2 \cdot \pi}{pm} \tag{3.7}$$

$$\triangle z = \frac{z_{max} - z_{min}}{pn} \tag{3.8}$$

in θ and z space.

After up sampling the facet in θ and z space, a list of θ_i and z_i on the facet can be obtained, which is a key to connect the geometry and the map. First of all, the left down corner of this map is registered to the position with coordinates [x_{min},y_{min},z_{min}]; the right up corner of this map is registered to the position with coordinates [x_{min},y_{min},z_{max}]. Then using θ_i and z_i, uv values are calculated as

$$u_i = \frac{\theta_i}{2\pi} \tag{3.9}$$

$$v_i = \frac{z_i - z_{min}}{z_{max} - z_{min}} \tag{3.10}$$

and used to link the pixel value of the map with the up sampled facets of the object. When the relationship between the pixel values of the map and the facet of geometry is found, the cylindrical coordinate system is back transformed into the original Cartesian system.

In the original Cartesian system, Figure 3.9 shows the up sampled points on the initial facet, and Figure 3.10 shows the up sampled facets with UV mapped pixel values from a map, while Figure 3.11 shows the whole cylinder OBJ mapped with a 16× 256 "cameraman.tif" image.

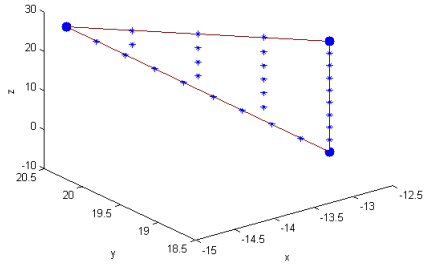


Figure 3.9: Up sampled points on a facet for wrapmapping method

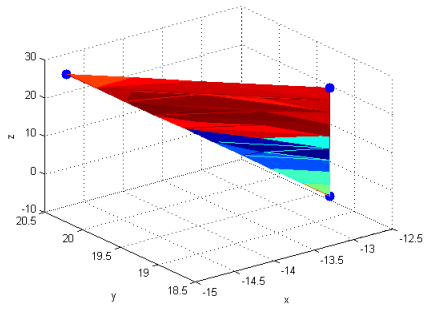


Figure 3.10: Up sampled facets with UV mapped information

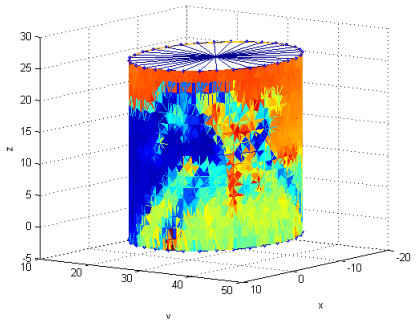


Figure 3.11: Wrapmapped cylinder.obj with 16×128 "cameraman.tiff"

3.2.2 Two-tanks thermal model

This thesis uses a two-tanks model as a notional example to show how DIRSIG incorporates the external physical model to predict the distribution of thermal or other characterizations of the object facets described in DIRSIG gdb file. This two-tanks model as shown in Figure 3.12 includes three valves (represented as v1,v2,v3 respectively) and two tanks (tank A, tank B). When v1 is open, tank A and tank B are connected and hot water could be released through the pipe into tank B. When v2 is open, the hot water could come into tank A; while when v3 is open, the cooled water in tank B would be released into outside.

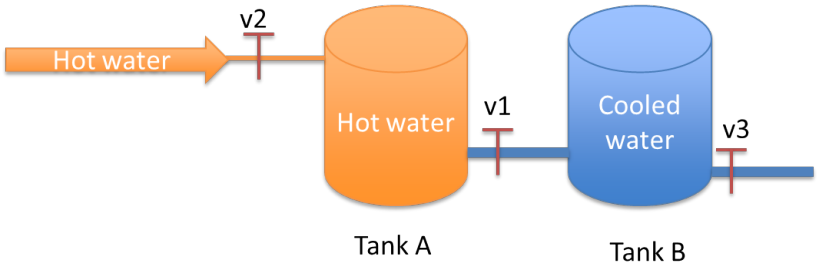


Figure 3.12: Two tanks model

A physical process model could be defined and developed to describe how the temperature and height of water in the two tanks changes over time with the valve open/close information as follows, based on the law of conservation of energy.

The heat in tank A at time t is

$$Q_{tankA,t} = V_{tankA,t} \cdot \rho_w \cdot C \cdot T_{tankA,t}, \tag{3.11}$$

where $V_{tankA,t}$ is water volume in tank A at time t; C is specific heat capacity; $\rho_w$is water density; $T_{tanksA,t}$ is the temperature of water in tank A at time t.

The heat in tank B is

$$Q_{tankB,t} = V_{tankB,t} \cdot \rho_w \cdot C \cdot T_{tankB,t}, \tag{3.12}$$

where $V_{tankB,t}$ is the water volume in tank B at time t; $T_{tanksB,t}$ is the temperature of water in tank B at time t.

When the valve is open at moment t, according to the law of conservation of energy, heat in tank A at time t is equal to

$$Q_{tankA,t} = Q_{tankA,t-1} - Q_{tankA,t-1} \times P_{lost} - Q_{toB} + Q_{in}, \tag{3.13}$$

where P_{lost} is the percentage of energy lost to the environment; Q_{toB} is the heat transferred to tank B; $Q_{in}$is the heat coming from outside through pipe 2.

The volume in tank A will also change as

$$V_{tankA,t} = V_{tankA,t-1} - V_{toB} + V_{in}, \tag{3.14}$$

where $V_{toB}$is the water volume transferred from tank A to tank B; V_{in} is the water volume coming from outside through pipe 2.

While the heat in tank B at time t is equal to

$$Q_{tankB,t} = Q_{tankB,t-1} - Q_{tankB,t-1} \times P_{lost} - Q_{out} + Q_{toB}, \tag{3.15}$$

And the volume in tank B will also change as

$$V_{tankB,t} = V_{tankB,t-1} + V_{toB} - V_{out}, \tag{3.16}$$

where V_{out} is the water volume released to outside through pipe 3.

The heat transferred from tank A to tank B at time t is

$$Q_{toB} = V_{toB} \cdot \rho_w \cdot C \cdot T_{tankA,t}, \tag{3.17}$$

where V_{toB} is the volume transferred from tank A to tank B and it can be calculated as

$$V_{toB} = A_{pipe1} \times v_{pipe1,t}, \tag{3.18}$$

where A_{pipe1} is the intersection area of pipe 1; $v_{pipe1,t}$ is the water velocity at time t in pipe 1.

Heat coming from pipe 2 at time t is

$$Q_{in} = V_{in} \cdot \rho_w \cdot C \cdot T_{in}, \tag{3.19}$$

where V_{in} can be calculated as

$$V_{in} = A_{pipe2} \times v_{pipe2,t}. \tag{3.20}$$

where A_{pipe2} is the intersection area of pipe 2; $v_{pipe2,t}$ is the water velocity at time t in pipe 2; and T_{in} is the temperature of water coming from outside to tank A through pipe2, which is assumed as constant.

Heat released from tank B to outside through pipe 3 is

$$Q_{out} = V_{out} \cdot \rho_w \cdot C \cdot T_{tankA,t}, \tag{3.21}$$

where V_{out} is the water volume released from tank B to outside which can be calculated as

$$V_{out} = A_{pipe3} \times v_{pipe3,t}. \tag{3.22}$$

The water velocity in pipe1 is calculated as

$$v_{pipe1,t} = \sqrt{2 \cdot g \cdot (h_{tankA} - h_{tankB})} \times V1_t. \tag{3.23}$$

where g is gravitational acceleration;h_{tankA} is the height of water in tank A; h_{tankB} is the height of water in tank B; $V1_t$ is the valve 1 open/close information. If $V1_t$=1, then the valve1 is open; otherwise it is closed.

The water velocity in pipe 2 is calculated as

$$v_{pipe2,t} = velocity2 \times V2_t, \tag{3.24}$$

where $velocity2$ is a constant value; $V2_t$ is the valve 2 open/close information. If $V2_t$=1, then valve2 is open; otherwise it is closed.

The water velocity in pipe 3 is calculated as

$$v_{pipe3,t} = \sqrt{2 \cdot g \cdot h_{tankB}} \times V3_t, \tag{3.25}$$

where $v3$ is the valve 3 open/close information. If $V3_t$=1, then valve3 is open; otherwise it is closed.

By setting the initial status of the water inside the two tanks and some size parameters of the tanks and pipes, assume the open/close time series of v1, v2, v3 looks like as shown in Figure 3.13, the two-tanks process model will generate temperature and height information over time as shown in Figure 3.14.

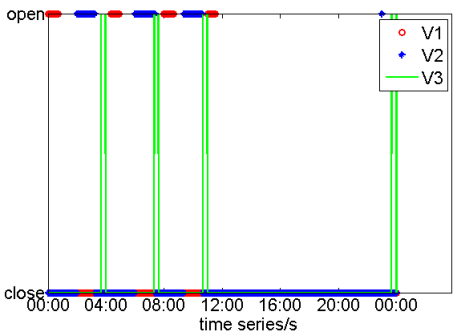
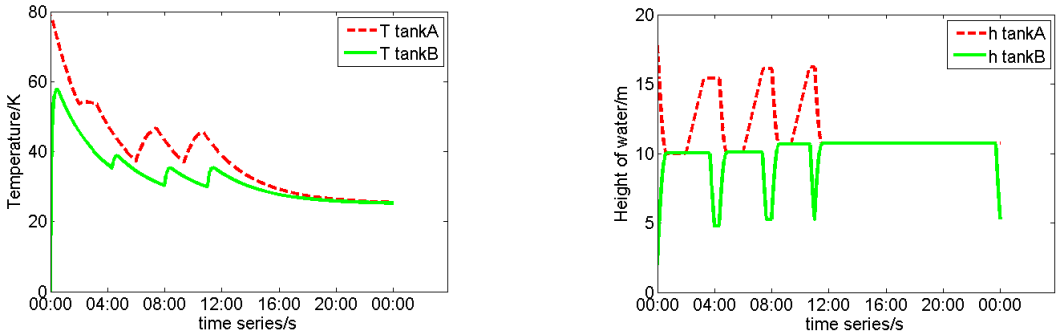


Figure 3.13: Valve open/close time line



(a) Temperature (b) Height

Figure 3.14: Temperature and height of the water changing over time

3.2.3 Two-tanks thermal model incorporated DIRSIG simulation

With the two-tanks thermal model, the temperature and height information of the two tanks at each time could be obtained and then be standardized into two images with each size as $n \times 1$, where n is determined by the resolution specified by the user. The standardized temperature images are mapped to the geometry of the two tanks respectively. Figure 3.15 is an example of the mapped result at a certain time.

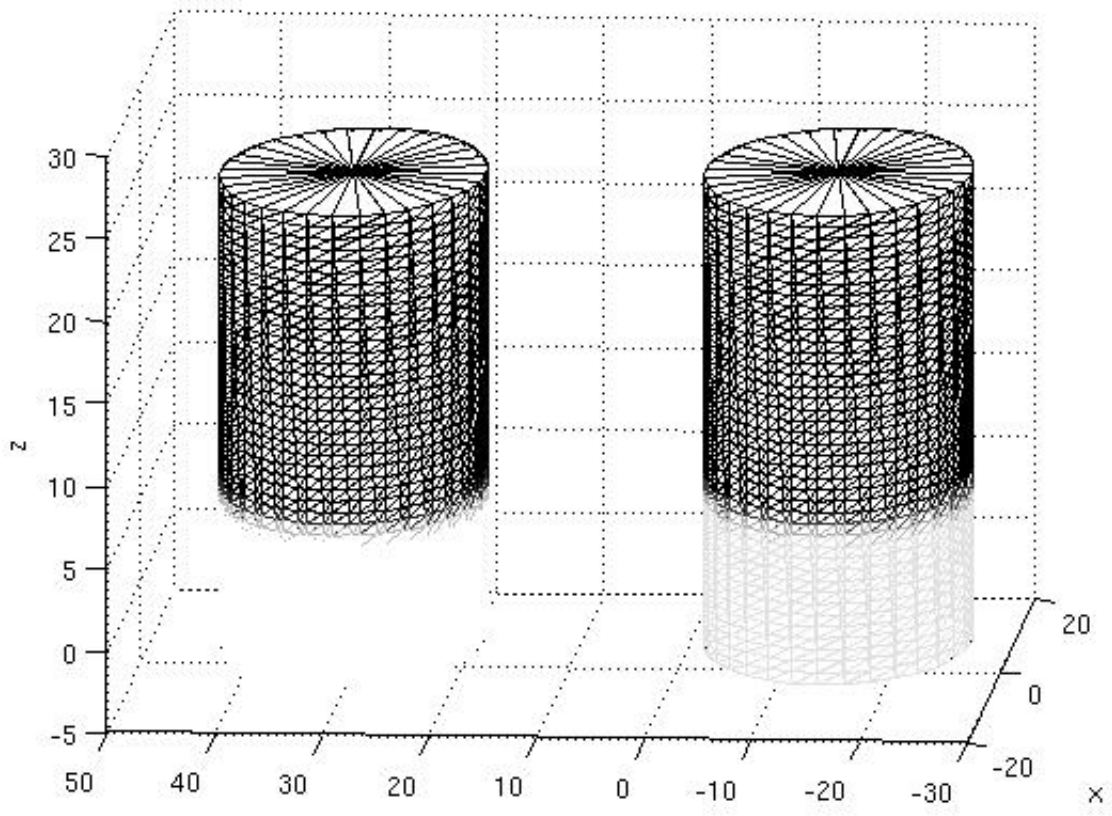
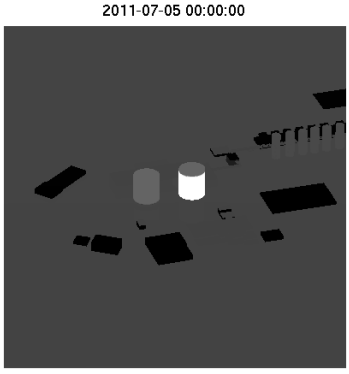


Figure 3.15: Two tanks mapped with temperature maps (Brightness is indicative of tank wall temperature.)

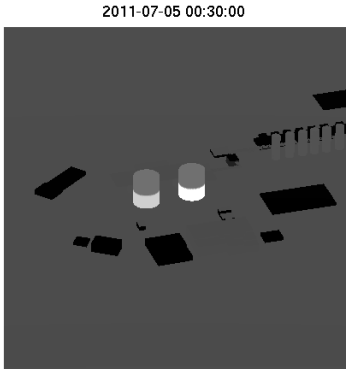
Figure 3.16 shows a DIRSIG simulation in RGB bands of part of Midland scene in Michigan in the morning, where the two tanks are found sitting in the middle of the scene. After the incorporation of the two tanks physical process model with DIRSIG, the simulation could be upgraded to also include temporal signatures of the two tanks state changing over time. Figure 3.17a shows a DIRSIG thermal simulation of part of Midland scene at midnight of a certain day. One tank is filled fully with hot water and the other one is filled with a small amount of cold water. Then the valve between the two tanks is opened, resulting in the hot water of one tank released immediately into the other tank and mixed with cold water in the other tank. After half an hour, another DIRSIG thermal simulation of the same

scene renders an image as shown in Figure 3.17b, which is the same as we expected to see.

Figure 3.16: RGB DIRSIG simulation of part of Midland scene



(a) Thermal simulation at midnight



(b) Thermal simulation at half hour later

Figure 3.17: DIRSIG simulation of part of Midland scene with two tanks process model included

Therefore, with the aid of incorporation of the two tanks process model and DIRSIG, the state of the water in the two tanks at each time of interest could be included in the finial simulated DIRSIG images.

3.3 Process models driving per object geolocation changing

In this section, we will show how the process model which drives the object geolocation changing is incorporated into DIRSIG.

A good example of the object geolocation changing over time can be the cars in the parking lot. The parking lot process model is interesting to many applications. For example, transportation planners need to develop good parking policies to overcome the problem of traffic congestion and insufficient parking spots; and security managers need to decide how many personnel would be sent out in case of emergency. Several parking models have been developed to analyze individual travel and parking behavior [113, 20, 103] . A well-known hierarchical parking model is suggested by Young and Taylor[113] to cover the whole process from parking design to policy analysis. The PAMELA [103] contains tools to predict adaptive parking choice behavior, car movements at the parking lot, and apply a Tobit regression analysis [70] to define the parking duration. PARKAGENT[20] simulates the behavior of each driver and assumes that the parking durations are uniformly distributed between the minimum and maximum parking time for each type of driver. All these models are simulated from the perspective of the driver, by building a parking choice model of each driver.

Another parking model named PARKVIEW is reported in paper [99], which is based on the statistical description of the parking lot itself to generate probability of occupancy and parking duration of each parking spot. PARKVIEW analyzes the parking lot from the perspective of a viewer, and returns a status map of the parking lot at different times of the day. An experiment is set up to show how the statistical description of the parking lot can be obtained easily and accurately from images taken of the parking lot by using the proposed method in this paper. Traditional parking models take data from field survey. 1500 questionnaires were sent to the inhabitants to understand the driver's behavior in terms of parking time, location and parking preferences for PAMELA model[103], which is costly and

difficult to repeat and then update the data. Besides, the accuracy of the data is subject to many factors related to the respondents. The outcome of PARKVIEW is a prediction of status of each parking spot at different times of the day, which could be then used for other statistical analysis. For example, by checking how many cars would come to the parking lot and how many would leave from the output of this model at a specific time, transportation managers would have an idea how many cars would be on the road nearby the parking lot so that they know what to do to respond different situations.

This built parking lot process model PARKVIEW is then incorporated into DIRSIG by editing DIRSIG input files based on the output of PARKVIEW.

3.3.1 The PARKVIEW model

In this section of the thesis, the PARKVIEW model will be described to view how the parking lot is occupied at different times based on the statistical description of a parking lot. The statistical description of a parking lot includes the distribution of parking duration, parking lot occupancy over time, and the preference of parking spots. The initial status of the parking lot is determined by the occupancy of the parking lot at the initial time and the preference score of the parking spot. The initial probability of a parking spot to be occupied is written as

$$P_{0,i} = occ_0 \times wp_i \tag{3.26}$$

where occ_0 is the initial occupancy of the parking lot and wp_i is the term to weight the probability of the parking spot i to be occupied by considering its preference score. The weighting term wp_i can be calculated as

$$wp_i = \frac{pref_i}{\frac{1}{N}\sum_{i=1}^{N} pref_i} \tag{3.27}$$

where N is the total number of the parking spots; $pref_i$ is the preference score of parking spot i .

Due to the inclusion of the weighting term, the probability of the parking spot

is unbalanced from each other, some of which may be stretched to be larger than 1. However, any probability larger than one is equivalent to one. Therefore, the part of the probability larger than one will be tailed off, which results that the mean of the total probability of all parking spots to be occupied shifts to be smaller than it is supposed to be. Thus, further adjustment of the probability of the parking spot to be occupied should be conducted to make the probability in the range of 0 to 1 and thus achieve the mean occupancy value occ_0 of the parking lot. A schematic diagram is shown in Figure 3.18 to illustrate the process. First, the maximum probability of all parking spots is found and forced to be 1. Then the adjustment is achieved by setting the mean value and the ratio of the difference between mean value and each probability to the difference between the mean value and the maximum value as constant, written as

$$\frac{m - P_{0,i}}{m - max} = \frac{m - P'_{0,i}}{m - max'}, \frac{m - P_{0,i+1}}{m - max} = \frac{m - P'_{0,i+1}}{m - max'}, \frac{m - P_{0,i+2}}{m - max} = \frac{m - P'_{0,i+2}}{m - max'} \cdots \tag{3.28}$$

where $max' = 1$, $m = occ_0$, $max = max(P_{0,i}, P_{0,i+1}, P_{0,i+2}, ...)$.

Then the new initial probability $P'_{0,i}$, $P'_{0,i+1}$, $P'_{0,i+2}, ...$ of the parking spot to be occupied are calculated according to equation 3.28.

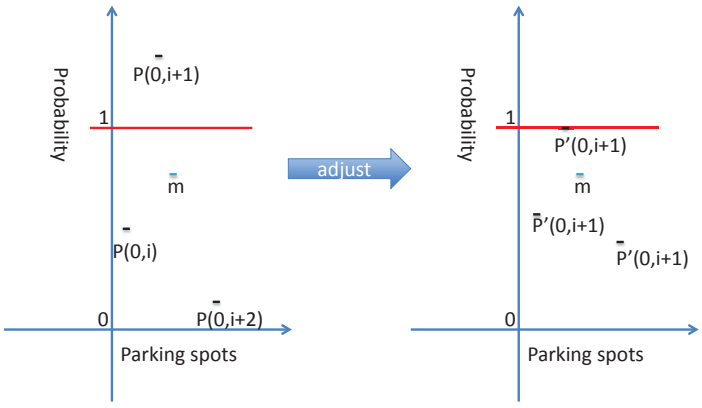


Figure 3.18: Probability Adjustment

To implement the car distribution of the parking lot with the calculated probability $P'_{0,i}$, a random number r with uniform distribution on the open interval (0,1) is generated by MATLAB. Since r is uniformly distributed on the interval (0,1), the probability of r to be smaller than x is x, where x is in the range (0,1). This property is used to determine whether a parking spot i is occupied or not, and the occupancy status of the parking spot is marked by $C_{0,i}$ as defined by

$$C_{0,i} = \begin{cases} 1 & \text{occupied} \\ 0 & \text{empty} \end{cases}. \tag{3.29}$$

So far, the initialization of the parking lot is done. For the following time t, any car parked in the parking lot could have a chance to leave with a probability determined by the cumulative distribution of parking duration. The leaving status of the car in the parking spot i is marked by $L_{t,i}$ as defined by

$$L_{t,i} = \begin{cases} 1 & \text{leaving} \\ 0 & \text{staying} \end{cases}. \tag{3.30}$$

With a certain number of cars leaving, the total number of cars in the parking lot is written as n_t as calculated by

$$n_t = \sum_{i=1}^{N} C_{(t-1),i} - \sum_{i=1}^{N} L_{t,i}. \tag{3.31}$$

According to the distribution of the parking lot occupancy at the moment t, the desired car number in the parking lot is m_t calculated as

$$m_t = N \times occ_t. \tag{3.32}$$

Therefore, at the moment t, the probability of an empty parking spot i to be occupied is defined as

$$P_{t,i} = \begin{cases} \frac{m_t - n_t}{N - n_t} \times wp_i & m_t > n_t \\ 0 & m_t <= n_t \end{cases}. \tag{3.33}$$

Again, if the probability $max(P_{t,i}) > 1$, then $\{P_{t,i}\}$ will be adjusted into $\left\{P'_{t,i}\right\}$ in the range of 0 to 1. With the probability set $\left\{P'_{t,i}\right\}$ for all parking spots at the moment t, the parking spot i having an arriving car is marked by $CM_{t,i}$ as

$$CM_{t,i} = \begin{cases} 1 & \text{with car arriving} \\ 0 & \text{no car arriving} \end{cases}. \tag{3.34}$$

However, a car could only park on an empty parking spot or a spot with a leaving car. Therefore, the flag $CF_{t,i}$ which truly marks the status of a parking spot i occupied by a new arriving car at the moment t is defined as

$$CF_{t,i} = (CM_{t,i} \cap \sim C_{t-1,i}) \cup (CM_{t,i} \cap L_{t,i}). \tag{3.35}$$

With arriving cars, the occupancy status $C_{t,i}$ of the parking spot is updated with

$$C_{t,i} = \left| C_{t-1,i} - (-1)^{C_{t-1,i}} \times CF_{t,i} \right|. \tag{3.36}$$

Possible values of $C_{t,i}$ are 0,1 and 2. 0 represents the parking spot is empty; 1 and 2 represents the parking spot is occupied with a car. 1 could be changed to 2 or 2 could be changed to 1 with a new car arriving to the parking spot.

If a car is leaving while no new cars would refill that parking spot, the occupancy status $C_{t,i}$ is further updated as

$$C_{t,i} = C_{t,i} \cap \sim L_{t,i} \cap \sim CF_{t,i}. \tag{3.37}$$

3.3.2 PARKVIEW incorporated DIRSIG simulation results

Assume that a city is going to hold an event and any city member who wants to join this event needs to get registered in a certain place from 10:01am to 11:40am

of a certain day. There are 200 spots in a parking lot and the first 100 spots near the registration center are highly preferred by the driver. The preference score is shown in Figure 3.19. The registration takes 15 minutes with 3 minutes deviation to finish, and the parking duration is assumed to have the same distribution pattern as the time taken to finish the registration, as shown in Figure 3.20a. The cumulative distribution of parking duration is shown in Figure 3.20b. The simulated occupancy of the parking lot is assumed to be a normal distribution across 10:01am to 11:40am as shown in Figure 3.21.

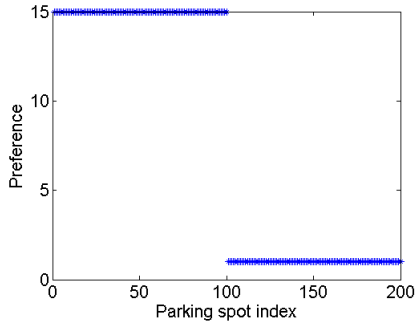
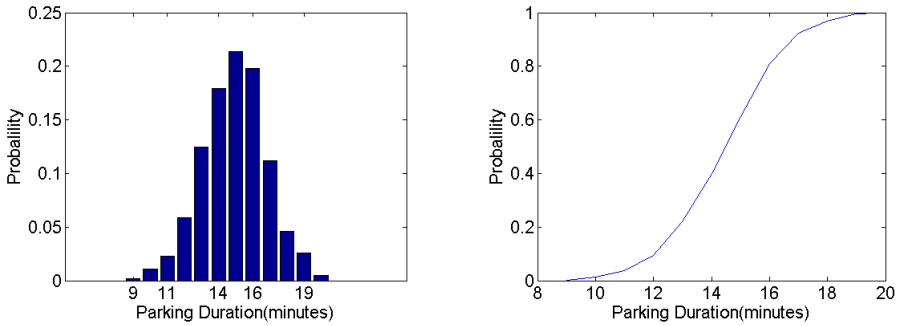


Figure 3.19: Preference of the parking spot



(a) Desired histogram of parking duration (b) Cumulative distribution of parking duration

Figure 3.20: Parking duration

Assume 100 snapshots are taken of the parking lot with even time interval from

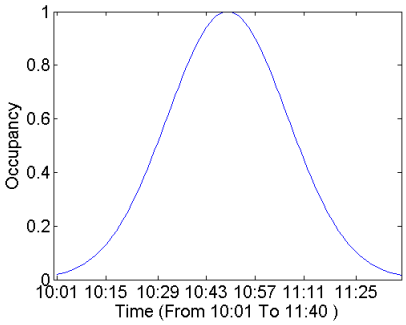


Figure 3.21: Distribution of parking lot occupancy

10:01am to 11:40am. By adopting the parking lot process model PARKVIEW built as above, the parking spot status on each frame is described by $C_{t,i}$. The parking duration of each car staying in the parking lot is extracted and the normalized histogram distribution is shown in Figure 3.22. The simulated occupancy of the parking lot over time is shown in Figure 3.23. The occupancy of each parking spot across the time interval is shown in Figure 3.24, which indicates the parking spots with higher preference score have corresponding higher occupancy over time. Overall, by looking at the figures, the simulated result of PARKVIEW has a good match to the desired one in statistical manner. To better convince the reader, several simulated snapshots of the parking spot are shown in Figure 3.25.

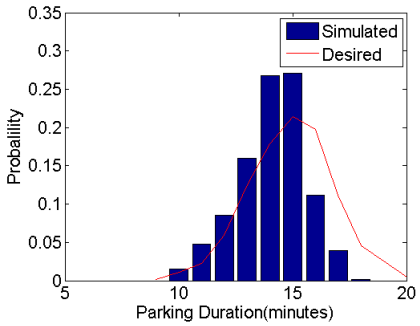


Figure 3.22: Histogram of simulated parking duration

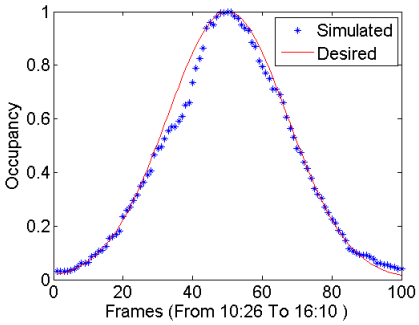


Figure 3.23: Simulated occupancy distribution of the parking lot

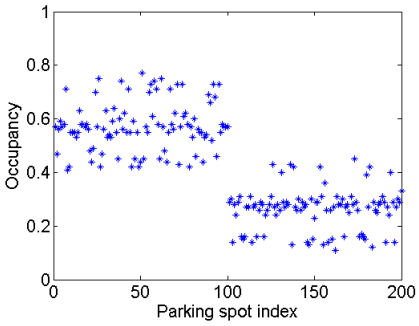
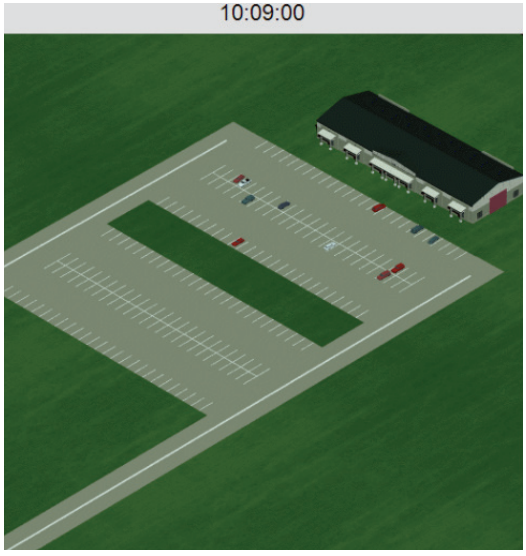
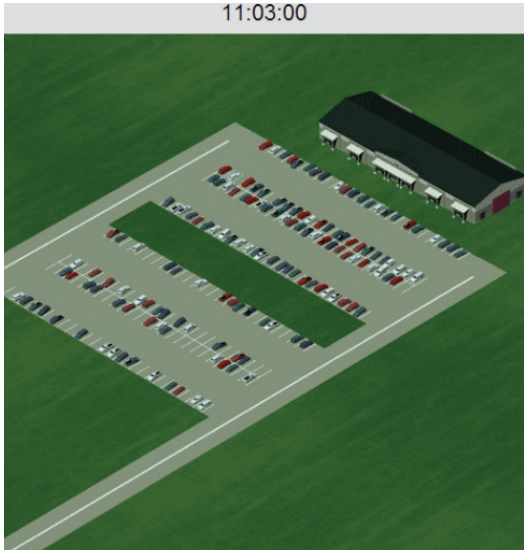


Figure 3.24: Simulated occupancy of parking spots



(a) Frame at 10:09:00



(b) Frame at 11:03:00

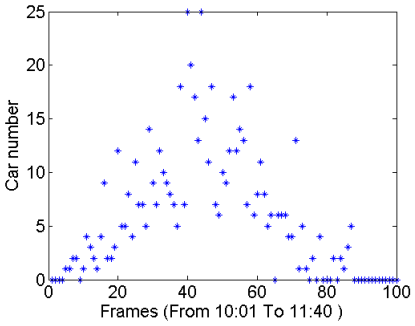
(c) Frame at 11:11:00

(d) Frame at 11:35:00

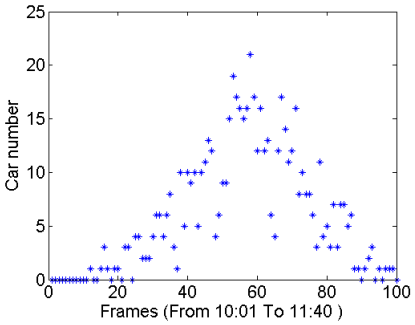
Figure 3.25: Simulated status of the parking lot

The output $C_{t,i}$ of PARKVIEW can also be used to extract other transportation information, such as how many cars are arriving or leaving the parking lot at certain times as shown in Figure 3.26. The skewness of the number distribution of cars

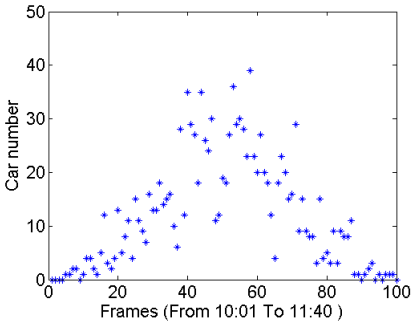
arriving and leaving are calculated as 1.0279 and 0.8343 respectively, which indicates that there are more cars arriving to the parking lot at an earlier time and more cars leaving the parking lot at a later time. In addition, the total number of moving cars at a certain time as shown in Figure 3.26 may also be interesting to transportation managers.



(a) Number of arriving cars



(b) Number of leaving cars



(c) Number of total moving cars

Figure 3.26: Number of arriving cars and leaving cars

3.3.3 Experiment: statistical description extraction of parking lots

The statistical description of a parking lot in terms of parking lot occupancy, parking duration and parking spot preference could be obtained from general estimation as done in section 3.3.2, or from field survey. However, in order to accurately and efficiently describe the status of the parking lot, a method of extracting a statistical description of the parking lot from real images is proposed in the paper[99].

An experimental set up was placed on the rooftop of the Chester F. Carlson Center for Imaging Science at RIT to record the distribution of cars on several parking lots of RIT campus by taking photos of the parking lot with an interval of 5 minutes from 10:26am to 4:10pm on Wednesday, September 21, 2011. 70 frames were collected in this experiment. A sample image is shown in Figure 3.27. The parking lots marked on Figure 3.27 are analyzed.

Figure 3.27: Original image of the parking lot

3.3.3.1 Experiment data processing

Due to the perspective of the camera, the objects appear smaller further from the building. In order to make the size of the object true to reality in the image, the original image is transformed to have a nadir view perspective of the parking lot. This transformation is achieved by using MATLAB image registration toolbox. Four corner points of a parking lot are selected as control points and four corner points of a rectangle as base points. The nadir view image of the parking lots is shown in

Figure 3.28. In Figure 3.28, the parking spots are identified manually by clicking the image where there is a parking spot, which takes the author 3 to 5 minutes to finish pointing out the total 250 parking spots. A red square is drawn around the clicked point by considering the view angle of the camera in horizontal direction at the corresponding point. There are several approaches to recover the parking lot structure automatically. Wang and Hanson [108] suggest to extract the structure of a parking lot by treating the vehicles as 3D microstructures, which requires multiple images form different angles. However, most of the time the requirement can not be satisfied which makes this method difficult to be generalized. Seo and Urmson [94] present a self-supervised learning algorithm that estimates the structure of the parking lot with double layers, but the low-level layer is based on the assumption that the parking lot contains a number of well-illuminated empty parking spots, which is not always true. Therefore, if the parking lot is almost fully occupied as Figure 3.27, the fact that there are not enough "seeds" extracted from the low-level layer to be fed into the high-level layer makes this method impractical in our case. To automatically extract the parking lot structure is still an open and challenging problem.

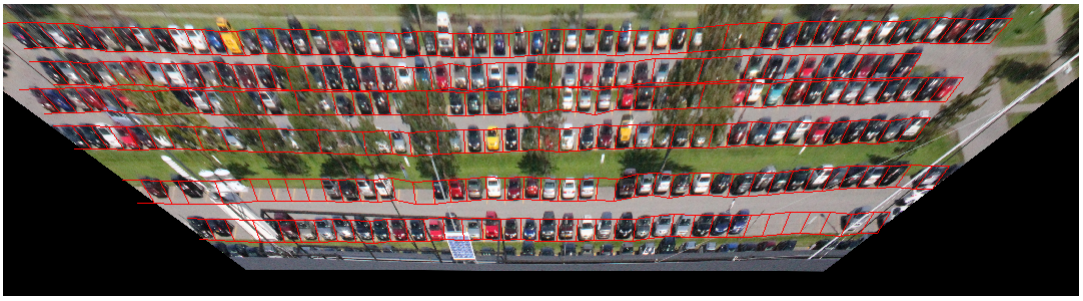


Figure 3.28: Perspective transformed image of the parking lot

With the information of the parking spot location, the status detection of the parking spot becomes easier. Three steps are done to retrieve the status of the parking spot at each frame, as shown in Figure 3.29.

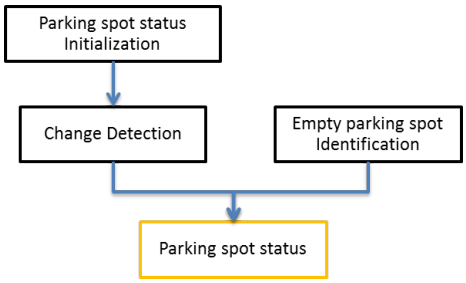


Figure 3.29: Diagram of extracting the parking spot status

The first step is initializing the status of the parking spot. Grass, asphalt and tree are recognized as background, while any non-background pixel inside the parking spot is contributed to a car identification. The detailed diagram is shown in Figure 3.30. First, background training samples (grass, tree, asphalt) are fed into Gaussian Maximum Likelihood (GML) Classifier. The corresponding class maps are generated by the GML classifier. Considering vehicles are much more diverse in color than the background, morphological image opening is performed on each class map in order to remove possible pixels of a vehicle from the background class map. After image opening operation, the class maps are fused into one background map and then the background map is inverted to render a foreground map with all possible vehicle pixels highlighted as shown in Figure 3.31. If the ratio of the highlighted pixel number to the total pixel number inside the red square of the parking spot is larger than 0.2, the parking spot is declared to be occupied and the status of the parking spot is marked as 1.

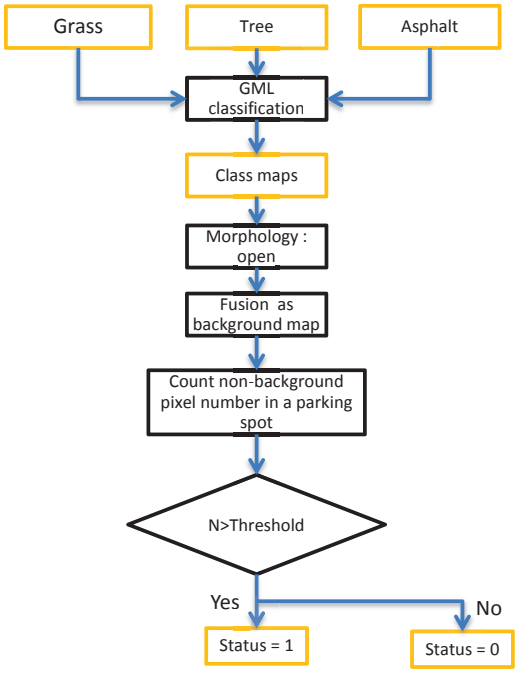


Figure 3.30: Diagram of parking spot status initialization

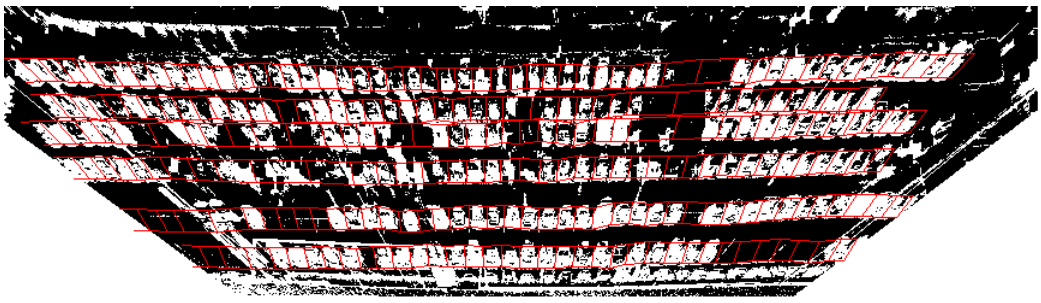


Figure 3.31: Vehicle class map

After initialization of the parking spot status, the following frame is compared to the frame before it. The status change of the parking spot is detected by using the diagram as shown in Figure 3.32. The diagram starts from two successive frames from the image set, I1 and I2. Edge and intensity features of the vehicle are used as two clues of a status change of the parking spot. If the change of either feature is

larger than a certain threshold, a status change of the corresponding parking spot is identified and the change flag of that parking spot is marked as 1.

To detect the change of the edge feature in the parking spot, MATLAB edge detection toolbox is used to generate a edge map of the parking lot. The total edge length el_i inside the red square of the parking spot is calculated. The difference ed_i of the edge feature is characterized by

$$ed_i = \left| \frac{el_i - el_i'}{min(el_i, el_i')} \right| \tag{3.38}$$

where el_i' is the total edge length inside the red square of parking spot i of the former frame. The threshold of the edge feature difference ed_i to declare a change, is set as 1.

Due to the weather changing over time, the illumination over the entire image or part of the image may change suddenly with clouds covering overheads. The local difference should be considered to recognize a change in the intensity field. Two successive frames are subtracted from each other and the rendered difference map is converted into a binary map in order to identify large differences. A window with 6×3 (units: parking spot width) is adopted to analyze the local difference from the binary difference map. If the ratio of the number of pixels marked as 1 of the binary difference map to the total number of pixels inside the window is larger than 0.3, the mean and standard deviation difference are used to determine whether there is a change in the parking spot. The mean difference is calculated with the local mean background difference deducted as

$$\triangle m = \left| \frac{\sum_{j \in s}(I_j - I_j')}{N_s} - \frac{\sum_{j \in W, j \notin M}(I_j - I_j')}{N_W - N_M} \right| / min\left(\frac{\sum_{j \in s} I_j}{N_s}, \frac{\sum_{j \in s} I_j'}{N_s} \right) \tag{3.39}$$

where s is the pixel index set inside the red square of the parking spot; W is the pixel index set inside the local window; M is the pixel index set with the pixel value of the binary difference map as 0 inside the local window; N_s is the pixel number inside the red square of the parking spot; N_W is the pixel number inside the local window; N_M is the number of pixels with pixel value of the binary difference map

as 0 inside the local window.

The standard deviation difference is calculated according to

$$\triangle\sigma = \left| \frac{\sigma - \sigma'}{min(\sigma, \sigma')} \right| \tag{3.40}$$

where σ and σ' are the standard deviation of the pixel value inside the red square of the parking spot of the two successive frames from the image set.

If $\triangle m > 1$ or $\triangle\sigma$>2, the status of the parking spot is determined to have a change, and the change flag of the parking spot is marked as 1.

However, if the ratio of the number of pixels marked as 1 to the total number of pixels inside the window of the binary difference map is smaller than 0.3, morphological image opening of the binary local difference map is found to do a better job in detecting the change in intensity field. Note that the binary local difference map is obtained by converting the difference map of the local area inside the window of two successive frames into binary. Then the morphological opening operation is performed on the binary local difference map with the structure element size proportional to the ratio of the number of pixels marked as 1 to the total number of pixels inside the window of the binary difference map. If 40% of the pixel value inside the red square of the parking spot is 1 of the morphological opened binary local difference map, then a status change is detected and the change flag is set as 1 for the parking spot.

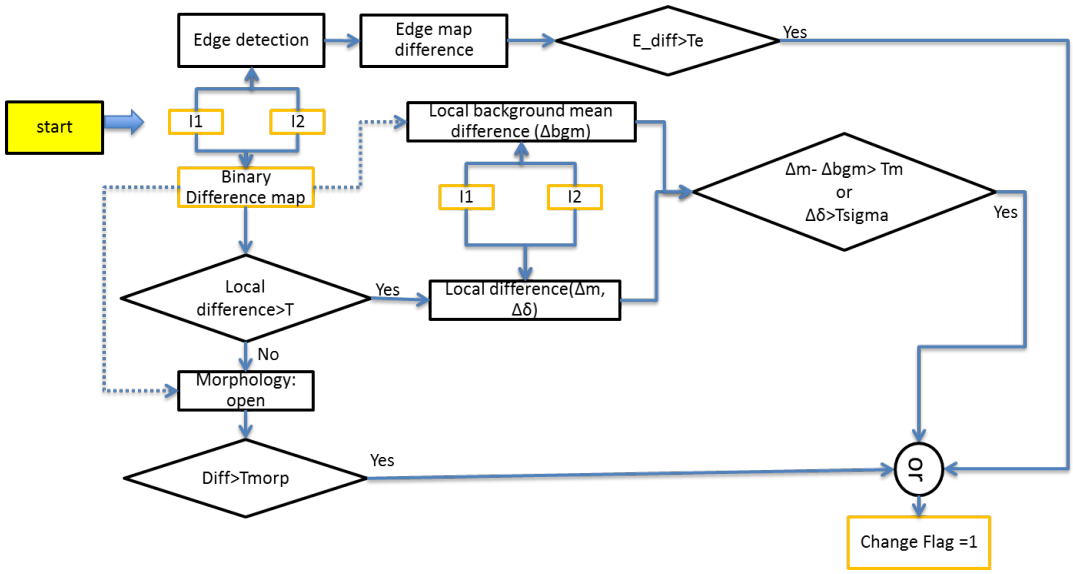
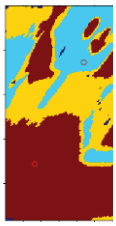


Figure 3.32: Diagram of change detection of the parking spot status

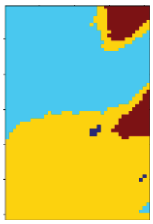
So far, we can successfully detect a status change of the parking spot. However, the status change could result from different scenarios, for example: a vehicle fills in an empty parking spot; a vehicle leaves with an empty parking spot left; and one vehicle replaces another vehicle. Therefore, in order to know how the parking spot is changed, the empty parking spot should be also identified. The empty parking spot tends to be smooth and uniform in color and shape. Based on this knowledge of the empty parking spot, K-means classification is used to classify the image. Two parking spot images with different parking status are shown in Figure 3.33 and Figure 3.34 respectively. Two boxes are dawn around the top center and the bottom center of the parking spot as shown in Figure 3.33 and 3.34. The class indices inside the two boxes are inspected. The empty parking spot has uniform class set inside at least one of the two boxes. The spectral angle between the pixel from the uniform class of the box and a background pixel is calculated to determine whether the uniform class has the same spectrum as the background. If the spectral angle is smaller than 0.02, the parking spot has a good chance to be empty. However, we still cannot be sure it is an empty spot because sometimes the vehicle is small or the vehicle is not parked properly in the parking spot. Either box may miss the feature

from the vehicle. Therefore, the class index around the center of the parking spot is also checked to see whether the center of the class map inside the parking spot also contains pixels with the same spectrum as the background. If so, then the parking spot is claimed as empty. The status of the parking spot is set as 0.

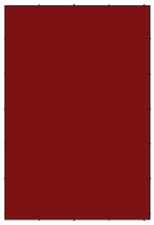
(a) Original parking spot



(b) Classification result



(c) Clusters in box 1



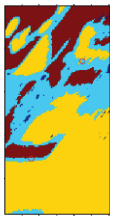
(d) Clusters in box 2

Figure 3.33: Empty parking spot

Scenarios	Parking spot status of one frame	Parking spot status of the following frame
An empty spot is filled with a vehicle	0	1
A vehicle leaves	1	0
A vehicle is replaced by another one	1	2
A vehicle is replaced by another one	2	1
A vehicle leaves	2	0

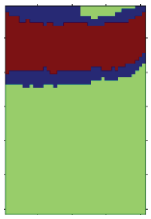
Table 3.1: Scenarios of parking spot status change

(a) Original parking spot



(b) Classification result

(c) Clusters in box 1



(d) Clusters in box 2

Figure 3.34: Occupied parking spot

3.3.3.2 Experiment Result

By going through the three image processing steps, the parking status could then be extracted. Possible status of the parking spot is marked according to Table 3.1. The result is checked frame by frame. There are errors in parking spots having obstacles such as trees, railings, poles and so on. Those "bad" parking spots are then removed from the result to better represent the statistical information of the parking lot. Among "good" parking spots, the result is analyzed as in Table 3.2.

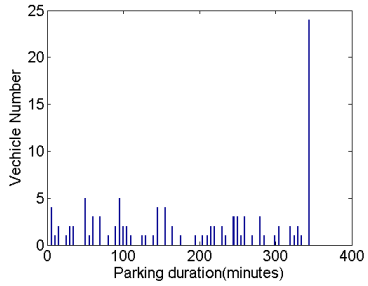
With the extracted parking spot status from the real images, the statistical

	missing	false alarm	hit
status initialization	0	2	216
status change	7	4	337
empty parking spot identification	6	2	103

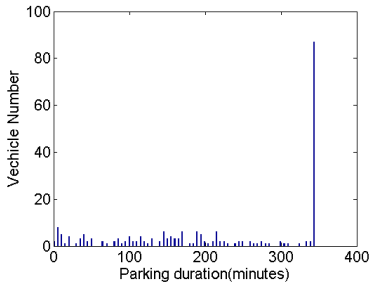
Table 3.2: Result Accuracy analysis of the parking status extraction

description of the extracted data is obtained for each type of parking lots. The histogram of parking duration, the distribution of parking lot occupancy, and the parking spot occupancy for each type of parking lots is shown in Figures 3.35, 3.36, and 3.37, respectively. By looking at the histogram of parking duration, we could find that most of the vehicles stay in the parking lot for the whole time of the experiment. There are peaks at some other parking durations, which indicates certain events occur on campus, for example the student takes classes for certain periods. The parking lot occupancy distribution indicates that the unreserved parking lot is fully occupied for most of the time during the day. Besides, the high occupancy of unreserved parking spots also indicates that people prefer free parking spots and are not willing to take the risk of getting a fine by parking illegally on the reserved parking spot. The parking spot occupancy could then be converted into preference score. The preference score is actually used as a probability weighting term as mentioned in section 2. If one parking spot is definitely preferred over the other one, the ratio of preference scores between those two parking spots should be infinity by setting the preference score of the less preferred parking spot as 0 and the preference score of the more preferred parking spot as any non-zero value, say A. The preference score of any other degree of preference is generated by interpolating between 0 and A.

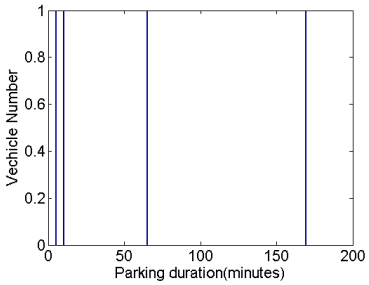
So the statistical description of the parking lot is successfully extracted from the real images of parking lots on the RIT campus, which can then be fed into PARKVIEW to regenerate the parking status for each parking spot over time.



(a) Reserved parking lot

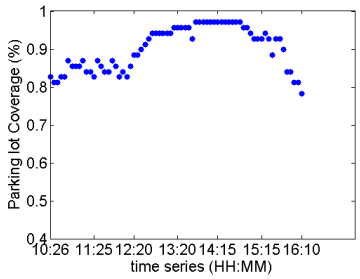


(b) Unreserved parking lot

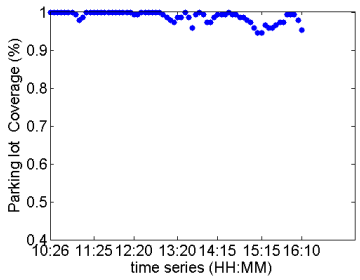


(c) 20 minutes parking lot

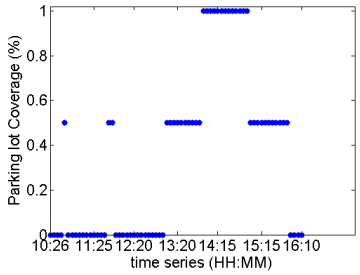
Figure 3.35: Histogram of parking duration



(a) Reserved parking lot



(b) Unreserved parking lot



(c) 20 minutes parking lot

Figure 3.36: Parking lot occupancy over time

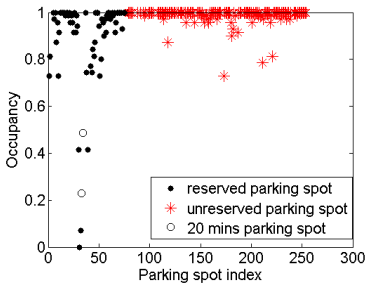


Figure 3.37: Parking spot occupancy

3.3.4 Summary

In this section of the thesis, a parking lot process model PARKVIEW is built based on the statistical description of the parking lot which includes parking lot occupancy, parking duration and parking spot preference. The output of PARKVIEW is the parking spot status map of the parking lot at each time during the process. This simulation result of the parking lot from PARKVIEW can be interesting to transportation managers or security managers. We use the simulation result of PARKVIEW to feed into DIRSIG by updating DIRSIG input file, so that the scene simulation capacity of DIRSIG can be enhanced by incorporating this parking lot process model. In order to show an accurate and efficient way to extract the statistical description of the parking lot, an experiment is set up during certain time of a week day on the RIT campus to take photos of several parking lots. The experiment data are processed by using three image processing steps to fully retrieve the parking spot status for each frame of the image set. The statistical description is extracted with the information of the parking spot status over time.

Chapter 4

Temporal Signature Analysis

With the trend of the increasing temporal image data available, as well as other forms of temporal data, methods of understanding those temporal information in a high level are required. In chapter 3, the illustrated enhanced DIRSIG can aid the user to generate the images with temporal signatures included, which will offer a great convenience to the analysis of the temporal signature in the scene. In this chapter, firstly, we will describe the problem we have encountered and how we are inspired to come up with the method. Then, a framework of the temporal signature steaming analysis ranging from recognizing, memorizing to predicting the event is proposed.

4.1 Problem statement

As we said, we want to understand a large amount of temporal images or information derived from temporal images in a high level, to identify activities and make decisions from those images. We assume that the features from each image have already been extracted and each feature has been quantized as a time series. Then the problem will be that of how to extract high level information from multivariate time series. Several methods of time series analysis have been looked at.

4.1.1 Kalman filter

Kalman filtering[111] is well known in the field of object tracking and navigation. At first, we considered to use the Extended Kalman Filter (EKF), which can model the nonlinear process, to track the time series by modeling the time series as a sinusoid with changing frequencies and amplitudes as shown in Figure 4.1. The green line in Figure 4.1 is the ideal signal, while the blue "*" points are the measurements of the signal.

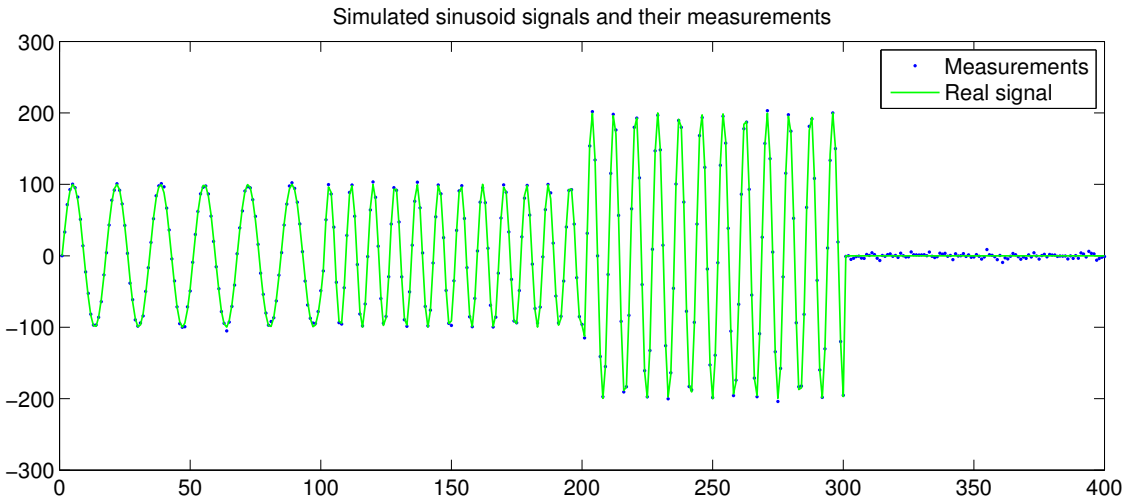
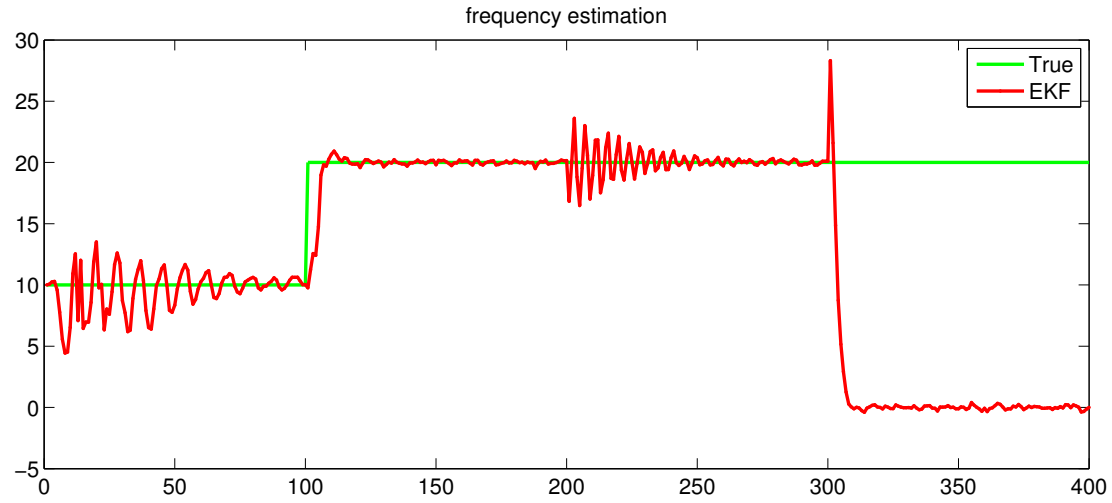
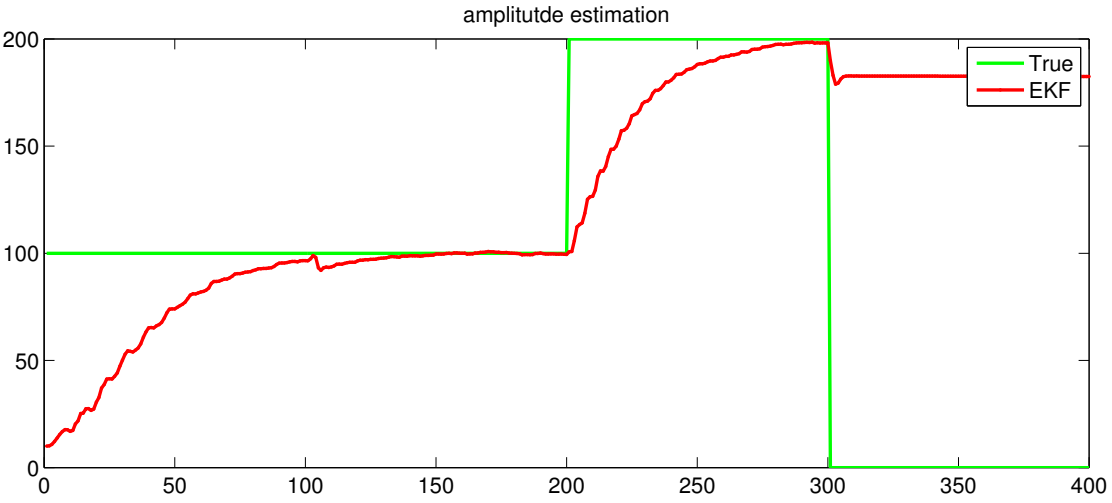


Figure 4.1: Simulated different modes of sinusoid signals and their measurements

At each step, the EKF first estimates the state vector (here including frequency and amplitude) and error covariance to achieve a priori prediction. Then with the measurements of the time series at that step, the EKF updates the state vector and covariance matrix to obtain a better posteriori prediction in a way that minimizes the posteriori error covariance. When the state of the time series changes, the EKF adjusts the state vector as shown in Figure 4.2a and Figure 4.2b to capture the change to achieve the local stable mode.



(a) Frequency estimation



(b) Amplitude estimation

Figure 4.2: The estimation result of state vector using Extended Kalman filter

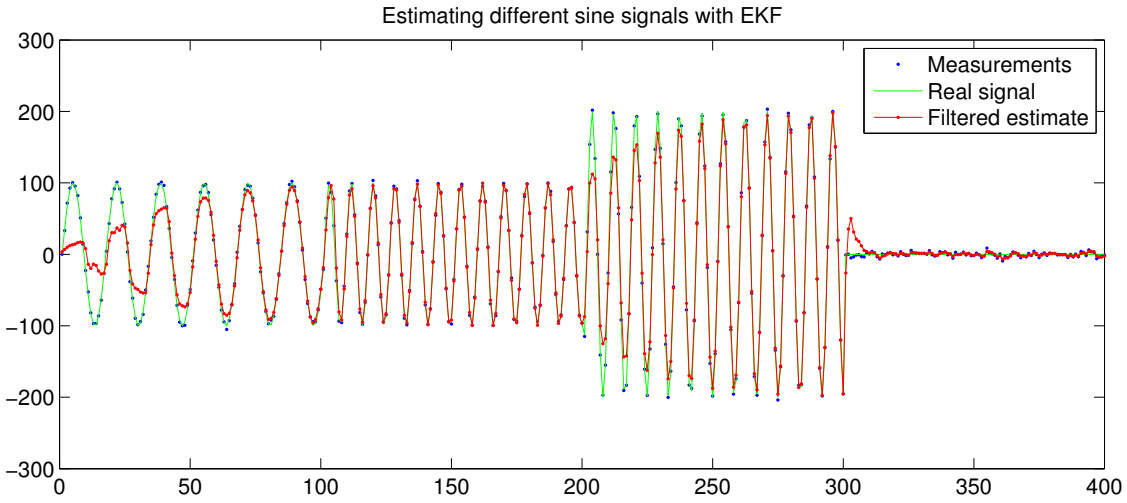


Figure 4.3: Sine signal estimation

However, in this way, the Kalman filter only remembers the local mode. When it comes to estimate the global state of the mode, we want to change the model of the time series into a way that the model is seen by the Kalman filter as stable. Then we considered to use a Fourier sinusoid series to fit the time series and use the Kalman filter to estimate the parameters of the time series. However, the Fourier sinusoid series does not converge at the discontinuous points[26], which is known as "Gibb's phenomenon". These discontinuities of the time series are also interesting to us as they can be the abnormal activities or the joint point of operational mode change in an ongoing process. What's more, the Kalman filter works well when the initial estimation of the state vector has the same negative or positive sign as the actual model. However, when the initial estimation is not with the same (+/-) sign as the actual model, Kalman filter does not work, which is also validated by [114]. Therefore, the method of fitting the time series with a Fourier sinusoid series and then use a Kalman filter to estimate the parameters of the Fourier sinusoid series was not pursued further.

4.1.2 ARIMA

Auto Regressive Integrated Moving Average [24] (ARIMA) is a method to model the non stationary time series which performs certain differences of the time series to achieve a stationary process. ARIMA is widely regarded as a powerful modeling technique[75] and has many applications in different fields, especially in the field of medicine[97][16]. The biggest challenge by using ARIMA in our situation is to find the right order of difference operation d to achieve a stationary process when the process is totally unknown and dynamically changing over time.

4.1.3 Human cognitive system

A normal human being can always easily recognize an event when they know enough knowledge about the event. For example, people know that when there is a big sale event in the mall, the mall will be very crowded. Therefore, when they see the mall is crowded, it is of high probability that the mall has a big sale event. Further more, by seeing the mall is crowded again and again on the Black Friday of Thanksgiving days, people can assure that on next year's Black Friday the mall will still be crowded. Figure 4.4 shows basically how human beings work with unknown events with the aid of knowledge and recall the event under certain triggers. These abilities of human beings are also what we want the computer to have. Therefore, in the thesis, we will use the way of human cognitive process to analyze the time series in order to recognize the events and find the temporal pattern to make predictions.

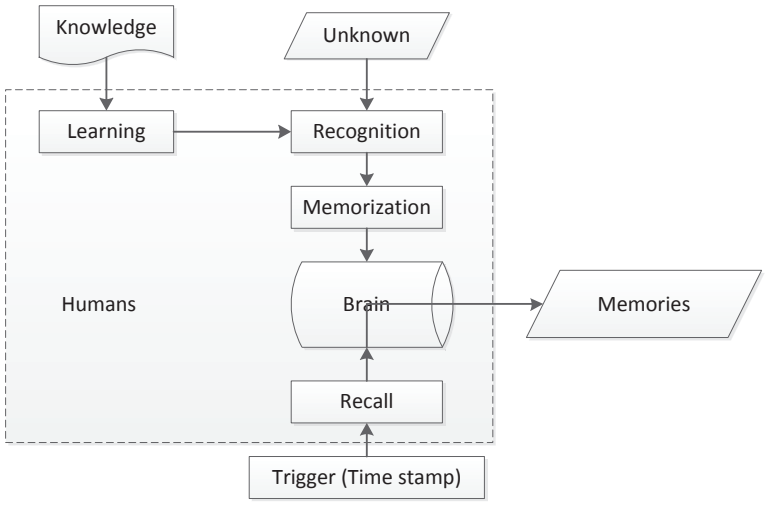


Figure 4.4: Human cognitive process

4.2 Proposed methods

The diagram in Figure 4.5 shows the framework of the proposed method of temporal signature analysis in streaming way. As we humans do, the proposed system is designed to learn knowledge of different activities and events continuously over time.

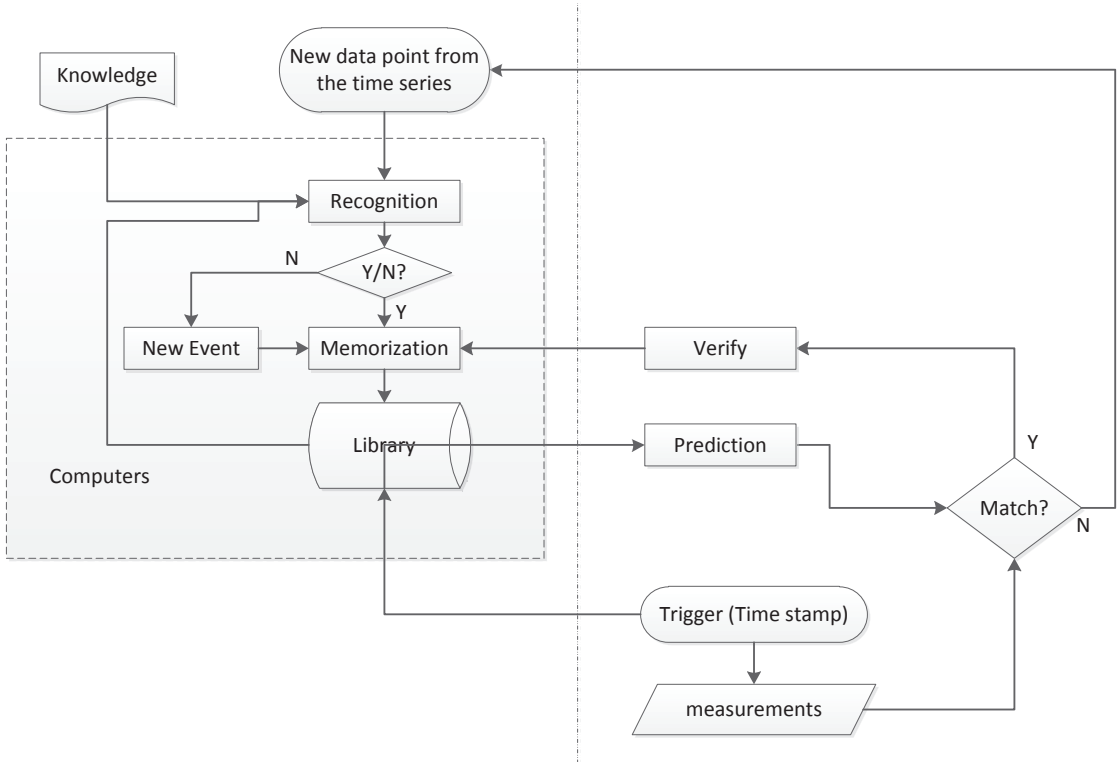


Figure 4.5: Proposed method of temporal analysis

When there are new data points from the time series fed into the model, the algorithm either uses the learned knowledge in library or the external existing knowledge system to recognize the new data point into an event. If the new data point cannot be recognized, a new event tag is created and assigned to the new data point.

After that, the information associated with the event is memorized, including the event type, event features, timestamps and so on. During the step of memorization, the impression or the memory of the event among all the events in the temporal space is built, which tells the temporal context of the event.

Then, the memorized event is stored into the library. The library has the record of the event information (including event type, the features of the event, start time, duration and so on), and the memory of the event.

After the library has stored a certain amount of historical information, we can predict what will happen at a certain time by recalling and analyzing the historical

event in the library. For large scale analysis, there may be more than one events that occur at the same time. However, in this thesis, we only consider that there are only one event that could occur at one time.

By checking the similarity of the predicted result and the new measurements, the new measurements go to two different directions of the algorithm. One is to go to the memorization block to make a contribution to boost the impression of a certain existing event occurring pattern, so that we are more confident to predict that event in the future. The other one is that the new measurements are considered as unknown, when they do not match all the predicted event in the library. Then the new measurements go to the start of the algorithm through the rest of the steps as mentioned above.

There are three key steps involved in the framework, which are recognition, memorization and prediction. The three key steps forms an abbreviate version of the whole framework as shown in Figure 4.6. In chapter 5 and chapter 6, the new method is proposed for each of the three steps. The Edinburgh pedestrian dataset is used as an example to assist the explanation of the whole framework and illustrate the result of those new methods. The information of the Edinburgh pedestrian data set is given in section 4.3.

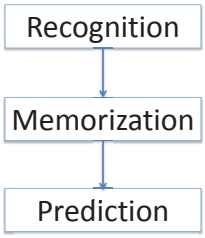


Figure 4.6: Three key steps of the proposed method

In the rest part of this chapter, some basic knowledge involved in time series analysis is described.

4.2.1 Preprocess the time series

The time series is sometimes pre-processed to make the data more reliable and approachable. The pre-processing of the time series normally includes: missing data estimation, outlier removal and noise reduction.

4.2.1.1 Missing data estimation

If only a couple of sparse data values are missing. The simplest way to estimate the missing data is to linearly interpolate the adjacent data. For example, we have an evenly sampled time series, and at a certain point the value is not collected. The time series is represented as $y_1, y_2, \cdots, y_{i-1}, ?, y_{i+1,} \cdots, y_n$. The missing data y_i at the ith step can be estimated by linear interpolation method as following equation

$$y_i = \frac{y_{i-1} + y_{i+1}}{2} \tag{4.1}$$

For general non evenly spaced data points, for example the time series above is sampled at temporal location $x_1, x_2, \cdots, x_{i-1}, x_i, x_{i+1}, \cdots, x_n$, the missing data y_i at temporal location x_i can be calculated as

$$y_i = y_{i-1} + \frac{y_{i+1} - y_{i-1}}{x_{i+1} - x_{i-1}}(x_i - x_{i-1}) \tag{4.2}$$

Other interpolation methods such as the cubic spline use a nonlinear function for each of intervals can be also adopted to achieve smoother and better interpolation results.

If there are a large amount of continuous data missing, it is very hard to estimate the missing data only basing on the neighboring data set. One way to estimate the the missing data is to relate the data set with other data sets. When another data set, which has certain relationship with the data set having missing data, is available, the missing data can be calculated based on the relationship between the two data sets. One other possible way is to estimate the pattern of the data set and fill in the missing data based on the pattern of the data set itself.

4.2.1.2 Outlier removal and noise reduction

Outliers and noises removal are popular problems in many fields. In the field of time series, there are many existing methods as reviewed in [59]. How to remove the outlier and reduce the noise is related to a specific situation or applications. The overall property of the data set should be understood before performing outlier removal or noise reduction, so that the signal is kept while the noise and outliers are effectively removed.

For outlier removal, the most common method is to check the mean and standard deviation of the data set. If a data point deviates from the mean more than 3 (which depends on the specific situation) times standard deviation of the data set, then that data point is regarded as an outlier. Anomalies are very similar to outliers in definitions. They both look significantly differently from the expected signal. In this thesis, we consider anomalies are the unexpected signal, from which useful information can be explored. When we perform outlier removal on the data set, steps should be taken to check the data point is a outlier or an anomaly.

Moving-average and binning are common methods of smoothing the time series and reducing the noise. For moving-average, the noise is reduced by using a fixed size window to move along the time series and take average of all the values inside the window as the finial smoothed signal. For binning, the time series is segmented into small bins, and the mean or median value inside the bin is taken as the finial smoothed signal. The window size or the bin size of the noise reduction method should be related to the sampling frequency of a time series. When the data set is very sparsely sampled, smaller size of the window or the bin should be used in order to avoid removing or blurring the signal.

4.2.2 Knowledge acquisition

There are several possible ways to get the knowledge.

First, the knowledge about the event can be the time frame of the event. Then by observing and recording the time series of the features during that time frame, the event can be learned. The learned feature pattern of an event can be used to map

out all the other events with the similar pattern occurred in the future to achieve the event recognition and detection.

Second, the knowledge about the event can be obtained by existing understood data sets. Those existing understood data sets can be used for training to get the mathematic descriptions. This will be illustrated with RIT twitter datasets.

Third, the knowledge about the event can be the rules defined by the user, or the intuitive description of some features. For example, people running are different from people walking in the speed and acceleration of moving, and it is common sense that the speed and acceleration of running is bigger than walking. By making the rule (here is the threshold) of the observed features (the speed and acceleration of moving), we can distinguish people who are running and walking.

Lastly, the knowledge about the event can be learned incrementally from the measurements. In the following chapters, this way of knowledge acquisition is adopted.

4.2.3 Event recognition

To recognize an event, the measurements of the event should be found to differentiate one event from the other. These measurements are called features of the event. The feature generation are data or event dependent. Different data sets may have different features to support the event recognition. Normally, more than necessary numbers of features are generated, which is followed by feature selection. Principle Component Analysis (PCA) is one of the feature selection methods to find the best features. There are tons of ways to recognize the event. The selection of event recognition method is also data dependent. To solve different problems, one event recognition method may be better than the other. The problem of event recognition can be solved in three different spaces: raw time series space, selected feature space, and frequency space.

In the raw time series space, one of the most straightforward ways is to measure the similarity of the time series by using Euclidean distance. More delicate method is Dynamic Time Warping (DTW), which can handle the misaligned two time series very well in evaluating the similarity of them.

In the selected feature space, when the clusters of the data are known, classification methods can be used to classify the events by comparing the test measurements with the known cluster features. Otherwise, when the clusters of the data are not known, the unsupervised classification method (often called as clustering method) should be applied. One typical clustering method is k-means. When the features are not sufficient to identify each event, anomaly detection methods can be used to find the anomaly events. RX and TAD are two example methods of anomaly detection. Besides, Spectral Angle Mapper (SAM) is also a very simple method to compare the similarity of two feature vectors from the geometric perspective[62].

The frequency domain of the time series can be analyzed to find the cyclic behavior, and certain filters can be designed to filter out the anomaly events.

Note that some of the event recognition methods are designed to be used to overview the static existing data sets, in order to understand the data set and fetch out interesting events by one time. For example, Fourier filter is designed to find the cyclic behavior of the whole data sets. RX and TAD also need the whole existing data sets to additionally detect the anomaly, because they need to know what the majority of data look like to identify the anomaly. These methods are good candidates for understanding the existing data sets to aid the definition of anomalies or events, which can be used for training. In the proposed framework, there are continuous data fed in. These new fed data should be incrementally compared to the feature templates in the library by using suitable similarity methods, such as Euclidean distance, SAM and so on.

In chapter 5, a new incremental clustering method called GMD is proposed to continuously recognize the measurements and tag them into event indices. In the following part of this section, some representative methods of event recognition from three different spaces (raw time series space, feature space, and frequency space) are briefly described.

4.2.3.1 Euclidean distance

The Euclidean distance between two time series $A = \{x_1, x_2, \cdots, x_n\}$ and $B = \{y_1, y_2, \cdots, y_n\}$ can be calculated as

$$d = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2}. \tag{4.3}$$

Obviously, the larger distance between two time series, the less similar they are. Euclidean distance is an easy and straightforward way to calculate the similarity of two series. However, it is very sensitive to the noise and outliers.

4.2.3.2 k-means classification

k-means is an unsupervised classification method to partition a series of observations in k clusters, where k is assumed to be known and set by the users. Each observation belongs to the nearest mean value of the k clusters. The process is optimized by iterative procedures. Firstly, an arbitrary cluster vector is assigned. Secondly, each observation is assigned to the nearest cluster. Thirdly, a new cluster mean vector is calculated based on all the observations in each cluster. Fourthly, the second and third steps are repeated until the change between each iterative is small enough.

The mathematical description of k-means classification method is: assuming there are n observations $x_1, x_2, \cdots, x_n$ and k clusters $c = \{c_1, c_2, \cdots, c_k\}$, the goal is to minimize the sums of squared distance SS between each observation and its assigned cluster center. The sum of squared distance SS can be represented as

$$SS = \underset{c}{argmin} \sum_{i=1}^{k} \sum_{x_j \in c_i} \|x_j - u_i\|^2, \tag{4.4}$$

where u_i is the mean value of the cluster c_i.

4.2.3.3 RX anomaly detection

RX anomaly detection[87] is widely used in anomaly detection in remote sensing with hyper-spectral imagery. RX detector assumes that the selected observations have a Gaussian distribution and the anomalous targets are far away from the selected data center. The mathematical description of RX detector is

$$R(\mathbf{X}) = (\mathbf{X} - \mathbf{m})^T \mathbf{S^{-1}} (\mathbf{X} - \mathbf{m}) \quad \begin{matrix} \geqslant \eta & target \\ < \eta & background \end{matrix} , \tag{4.5}$$

where $\mathbf{X}$ is the variable of the test observation, $\mathbf{m}$ is the mean of the selected observations, $\mathbf{S}$ is the covariance matrix of the selected data set, η is the threshold to determine whether the test observation is from a target or the background.

4.2.3.4 TAD anomaly detection

The topological anomaly detection algorithm[17] (TAD) is a novel anomaly detection algorithm, used to model hyper-spectral data by incorporating a topological method. In TAD, the background is modeled as a set of connected components of a graph without any assumptions on the geometry, linear or nonlinear, or statistical distribution of the data, which is one of the advantages over RX detector[18].

There are several steps taken to implement TAD[18]. The first step is to normalize the data, so that the highest 10% of the observations in the data have Euclidean L2 norm equal to 2 and the lowest 1% have Euclidean L2 norm equal to 1. The first step is optional. The second step is to model the background. A random subsample from all the observations is selected to model the background. A graph is constructed by adding an edge between the closest 10% (a variable) of pairs of points. Those components containing greater than 2% of the samples, are considered as background. The third step is to rank all the observations in the data set to find the anomalous targets, which is equal to the sum of the distances to the 3^{rd}, 4^{th}, and 5^{th} nearest neighbors in the background observations. The threshold of the TAD ranking can be decided by the percentage of the background observations in the subsamples.

4.2.3.5 Fourier filter

In order to analyze the frequency domain of the time series, Fourier transformation is normally taken to obtain the frequency components of the time series data. The continuous-time Fourier transformation of a function $x(t)$ is represented as

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-i2\pi ft} dt, \tag{4.6}$$

while the inverse continuous-time Fourier transformation is represented as

$$x(t) = \int_{-\infty}^{+\infty} X(f) e^{i2\pi ft} df. \tag{4.7}$$

In some situations, we do not have the continuous function description of the data, and only the function of the data is observed in certain time space. In this case, the discrete-time Fourier transformation (DTFT) should be used as in the following equation

$$X(k) = \sum_{j=1}^{N} x\left(j\right) e^{-i2\pi \frac{(j-1)(k-1)}{N}}, \tag{4.8}$$

while the equation of the inverse discrete-time Fourier transformation is

$$x(j) = \frac{1}{N} \sum_{k=1}^{N} X\left(k\right) e^{i2\pi \frac{(j-1)(k-1)}{N}}. \tag{4.9}$$

Certain filters can be designed to filter out some frequency components of the data in the frequency domain and remain the useful or interesting frequency components. The anomaly or noise information of the data are often shown up as very low values in the frequency domain. According to this property, the filter is designed to be piece-wised windows with value 1 and 0, where the frequency components having high values has 1, otherwise 0, as described in the equation

$$W(k) = \begin{cases} 1 & X(k) > \eta \\ 0 & X(k) < \eta \end{cases} . \tag{4.10}$$

The filtered frequency domain $\tilde{X(k)} = X(k) \cdot W(k)$, and the corresponding data in the time domain $\tilde{x(j)} = FT^{-1}\left(\tilde{X(k)}\right)$, where FT^{-1} is the operator of inverse discrete-time Fourier transformation.

4.2.4 Event memorization

An intuitive way of event memorization is that each type of event is memorized in an individual "memory zone", which can be achieved by creating a table for each event type in the event memory database, as shown in Table 4.1. The table will continuously grow when the event occurs again and again in the future.

Event Type 1			
Occurring times	Feature value index	Time stamp	Duration
1	i11,i12,i13...	t11,t12,t13...	T1
2	i21,i22,i23...	t21,t22,t23...	T2
⋮	⋮	⋮	⋮

Table 4.1: An example of event memory table

In this thesis, we propose a memorization method called double localization including absolute localization and relative localization to build the temporal context of events. The memorization with absolute localization is similar to the method described. However, instead of creating a table for each event type to record the temporal locations of the event, the proposed method uses the temporal map to absolutely localize events in the temporal space. By setting up the coordinates for the temporal map, the prior temporal pattern can be already embedded into the memory. The memorization with relative localization can be achieved by learning the relative temporal order between the events. The relative temporal order between events can be modeled by methods including: context free grammar [40, 78, 90],

Hidden Markov chain[35, 50, 100, 95], Bayesian network and so on. In this thesis, the first order Markov chain is used as an example to show how events are relatively localized in the memorization step.

In addition of memorizing the temporal part of the event, the static descriptions of the event are also memorized, such as the features of the event, in order to recognize similar events in the future.

A detailed illustration of event memorization can be found in Chapter 6.

4.2.5 Event prediction

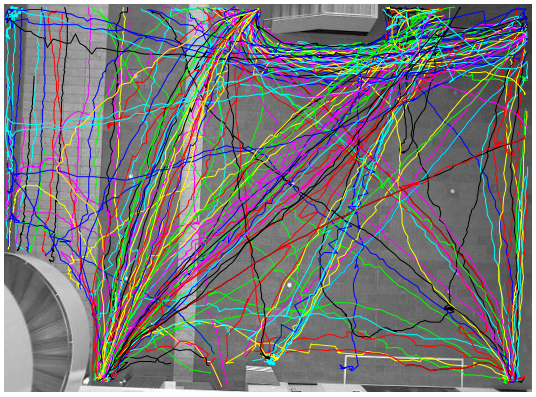
In order to the predict the future, the historical information should be well understood and the temporal pattern of the historical information should be properly modeled. In order to model the historical information, there are different ways including regression related methods (such as ARIMA as we tried), Markov models, Bayesian networks, and so on. In this thesis, the historical information is memorized by double localization. Therefore, our predictive model is based on how the historical data are memorized. That is to build the predictive model by incorporating the absolute temporal context with the relative temporal context. Detailed illustration can be found in Chapter 6.

4.3 Data sets

One dataset used in this thesis is from university of Edinburgh[71]. A camera is fixed overhead about 23m above the floor, watching the Informatics Forum, the building of the school of Informatics at the university of Edinburgh. People can be seen walking through the Informatics Forum. An example scene view of the Forum is shown in Figure 4.7a. The main entry/exit points (marked) are at the bottom left (front door), top left (cafe), top center (stairs), top right (elevator and night exit), bottom right (labs). Figure 4.7c and Figure 4.7d shows some other perspectives of the Forum. A series of image processing has been done by university of Edinburgh to extract the trajectories of the pedestrian. As claimed, there are 92,000+ observed

trajectories. A view of a small samples of trajectories are shown in Figure 4.7b.

(a) A view of the Forum scene



(b) A view of a small samples of trajectories in the Forum dataset

(c) Informatics Forum-1

(d) Informatics Forum-2

Figure 4.7: Example scene views of the Edinburgh Forum

This other dataset is from twitter. The twitter data on RIT campus are collected to understand the activities going on campus. Some preprocessing on the data is done to understand what kind of activities the twitter data can tell. A detailed description can be found in Appendix A.

In this thesis, only the Edinburgh dataset is used to illustrate the framework for streaming analysis of time series data.

Chapter 5

Recognition by incremental clustering of data streams

In recent days, data are collected almost everywhere over time. Those temporal collected data points are formed as an ordered sequence which is often called a data stream[45]. It is interesting that useful information could be inferred from the data streams. The first step to understand the data stream is to check which cluster the new data point could be assigned to. There are two situations, which also implies two different ways of solving the problem. One of the situations is that the data source has been understood, which means that data has been trained and labeled with meaningful tags. In this case, the new measurement data point will be compared with known descriptions of the tags and thus assigned with a certain tag. The techniques involved in this situation are also known as supervised classification. On the other hand, sometimes, the data source is not understood. We need to cluster the data stream with available measurements and update the clusters with every new data point, the technique of which is often regarded as incremental clustering[27].

Regarding to the first situation when the data source has been understood, it is relatively easier to solve than the second situation. This is to classify the new point to an existing cluster. The solution to this case is relatively clear already. Therefore, in this part of thesis, the second situation, when the data source is not trained or

learned before, is mainly considered.

There are several challenges of incremental clustering of data streams. Firstly, we do not have the overview of the whole data set. Instead, the data points are coming continuously over time. The clustering methods performed on the static data set using partitioning technique (for example: k-means[61], k-medoids[37], and the fuzzy c-means[47]) cannot be adopted here. Secondly, the number of clusters in the data is not clear, which requires an unsupervised learning from the available data set and updates with the new data. Thirdly, the memory for storing the data points and cluster information is limited, while there is an unlimited data stream. This requires us to find the most essential information and only keep the useful information in the memory. There are more challenges and issues related to data stream clustering, which are illustrated elsewhere[56].

5.1 Review of related work

In order to search for a good method of incremental clustering of data streams, the literature in terms of static clustering, clustering of time series, incremental clustering has been reviewed.

When we talk about clustering, it usually means static clustering. Here "static" means that the data set will not change over time and all the data are available to be processed in one batch. Any point of the data set can be accessed without considering the temporal order. Static clustering methods can be divided into five major categories[48]: partitioning methods, hierarchical methods, density-based methods, grid-based methods, and model-based methods. A detailed survey of static clustering can be found in[21].

The specialties of clustering the time series data are the input data format and similarity measurement methods. The input data can be the raw data of the time series in the time domain or the frequency domain, features extracted from the time series, and model parameters derived by modeling the time series with the family of Auto Regression (AR), or Auto Regression Moving Average (ARMA) related

methods[110]. The similarity between two time series can evaluated by (1) Euclidean distance, root mean square distance, and Minkowski distance; (2) Pearson's correlation coefficient and related distances; (3) Short time series distance; (4) Dynamic time warping distance; (5) Probability-based distance function for data with errors; (6) Kullback–Liebler distance; and so on. Detailed descriptions of the similarity measurement can be found in [110].

Our goal is to incrementally cluster the data stream. [89] presents an incremental system for clustering streaming time series in the variable domain. The basic idea is to find groups of variables that behave similarly through time. An Online Divisive-Agglomerative Clustering system is presented for incremental clustering of streaming time series by constructing a hierarchical tree-shaped structure of clusters using a top-down strategy. However, we are not interested in clustering the variables of the time series over time, since we assume here that the variables are the features of the data source and the features should be very distinct from each other so that they are highly representative of the object we want to describe. Therefore, we are more interested in incrementally clustering the data stream in the example domain.

COBWEB[38] is a conceptual clustering system, which incrementally organize the observations in a hierarchical tree with each node representing the concept of the object. The concept description is also given to summarize the category-conditional probability of the properties at the node. STREAM[83] uses a buffer to hold the data stream. When the buffer is full, leveraged k-medians is adopted to cluster the data in the buffer into k clusters. Only the weighted centroids of the clusters are kept for the next around of data buffering. The weighted centroids will be clustered into final k clusters when all the data have been checked.

BIRCH[115] incrementally builds a CF (Clustering Feature) tree as new data objects are inserted as the first phase. Each leaf node of the tree is a cluster, which is summarized by a CF vector, including the number N of points in the cluster, the linear sum of the N data points and the square sum of the N data points. BIRCH goes through 4 different phases. Some phases are optional. The next essential phase is to cluster all the leaf entries of the tree by a global or semi-global algorithm to capture the major distribution pattern in the data. CluStream[5] is a complete

system composed of two components, online and offline ones. For online micro-clustering, CluStream use k-means to generate q clusters as initialization. With each new data point, it incrementally updates the initial clusters by absorbing or creating a new cluster by checking whether the new data point falls in the maximum boundary of the nearest micro-cluster. It uses an temporal extended version of CF vector[115] to summarize the micro-clusters. The current set of micro-clusters together with their micro-cluster identification list is stored in a pyramidal time frame, and indexed by their time of storage. This information is then used by the offline component with a variety of user-defined parameters to get final higher level k clusters.

D-Stream is a density based method[29] which aims to capture the cluster with arbitrary shapes and without the need of initializing the cluster number k as some methods require[83, 115, 5] . This algorithm also adopts two components: online and offline components. The online component maps each incoming data point into a grid and the offline component computes the density of the grid and clusters the grids based on the density after a preset time step "gap". DenStream is another density-based approach[25]. However, density-based stream clustering methods are highly dependent on the order of the incoming data points, since they use static density thresholds to define a dense grid and assign a decay factor to each point over time. Additionally, they have no memory of historical clusters; therefore, they need to learn the every cluster as the first time.

A graph based incremental clustering method, RepStream, is presented in [69] by using representative points to cluster incoming points and retain useful cluster information in the knowledge repository. Each cluster is defined by two types of representative points: exemplar points and predictor points.

In this thesis, a new incremental clustering method (GMD) is presented. The name of our clustering method GMD is short for Gaussian-Merging-cluster Description, which describes three key steps involved. The method assumes the core of each cluster is Gaussian distributed. It incrementally updates the description of the Gaussian distributed cluster with each new point added to that cluster. Then by checking the separateness of the clusters, clusters are hierarchically merged if they

are not separable. Each node of the hierarchical tree is a cluster, which is summarized by a cluster description. This method has advantages in several ways. First, the user does not need to set the cluster number. Second, it can capture arbitrary shapes of clusters. Third, each cluster is summarized by a cluster description, which can represent any shape of the cluster. Fourth, this algorithm has the flexibility of tracking all the temporal occurrences of the cluster, which could be used later for temporal rule inspection. Fifth, it can detect outliers quite accurately. Sixth, it can handle high-dimensional data. Finally, the algorithm is insensitive to the order of the incoming data points.

Detailed illustration of this method will be shown in section 5.2.

5.2 Method

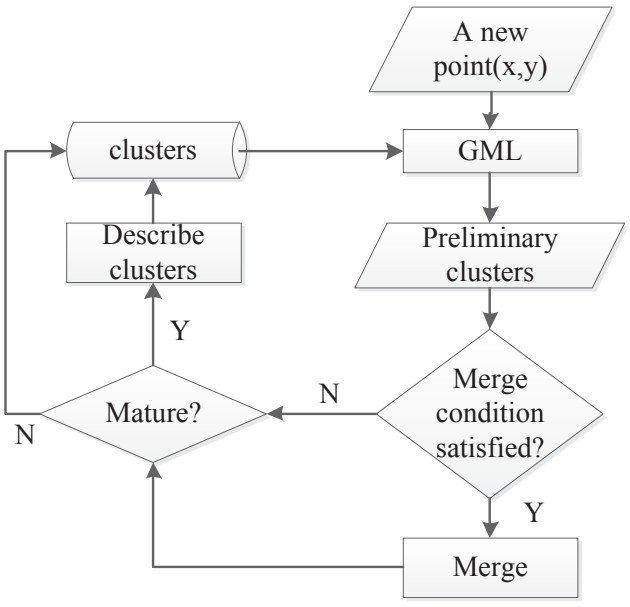


Figure 5.1: Workflow of incremental clustering

Figure 5.1 shows the workflow of our incremental clustering method. When we want to cluster a data stream with no data collected yet, the problem is how to initialize

the cluster distribution. One approach is to use a distance threshold to merge two points into one cluster. That is, when the distance between two points is smaller than the distance threshold, they belong to the same cluster.

Here we propose to assume that the core of each cluster is Gaussian distributed, which offers some advantages of saving computing load over the former approach. When the first data point arrives, a Gaussian cluster is automatically generated with the new coming data as the center and a pre-defined covariance matrix S_0. When the second point arrives, it is either assigned to the first cluster or a new Gaussian cluster is created with itself as the center and a pre-defined covariance matrix S_0, according to the class discrimination value D_i (where i is the cluster index) calculated by Gaussian Maximum Likelihood (GML). The likelihood that the new point in a member of cluster i is D_i and the maximum likelihood over all classes is D_{max}. If the maximum probability D_{max} is larger than a threshold T_D, the new point is assigned to the cluster with the maximum probability.

The immediate following step is a rule-based merging. It is to merge the clusters by counting the distinct cluster number in the neighborhood of the new point. If there is more than one cluster in the neighborhood, they will be merged into one if the merging rule is satisfied. The user can define different merging rules according to specific problems. Like this, the clusters will grow with each new coming point until a certain time when the number of points belonging to that cluster is larger than a threshold TN_{mature} and we call the cluster is "mature".

The mature cluster is then summarized by a cluster description. The approach used to generate the cluster description is most similar to the one used in [6], which is to find minimal number of rectangles for two dimensional data (or hyper-rectangles for high dimensional data) to describe the shape of the cluster. However, our approach is different in several ways. A detailed explanation will be shown in section 5.2.3. To do this, each dimension of the mature cluster is first divided into N_{parts} parts. The number of points inside each cell is counted. If the number is higher than the threshold TN_{dense}, the cell is marked as dense. After that, the minimal number of rectangles or hyper-rectangles are found to describe the dense cells. All the points in the non-dense cells are kept as representative points of the cluster together with

the rectangles or hyper-rectangles.

5.2.1 Gaussian Maximum Likelihood

Every time a new point $X_i = (x_{i,1}, x_{i,2}, ...x_{i,d})$ (where d is the dimension of the point, i is the point index) comes and it is not absorbed into any existing cluster, a new cluster C_{n+1} is created, where n is the number of existing clusters. We assume cluster C_{n+1} is Gaussian distributed, and described by a normal distribution $\aleph(m_{n+1}, S_0)$ which is equal to $\aleph(X_i, S_0)$, where m_{n+1} is the mean of cluster C_{n+1}. S_0 is the predefined covariance matrix which should be set by the user, which can be represented as

$$S_0 = \begin{bmatrix} \sigma_{011} & \sigma_{012} & \cdots & \sigma_{01d} \\ \sigma_{021} & \sigma_{022} & \cdots & \sigma_{02d} \\ \vdots & \vdots & \vdots & \vdots \\ \sigma_{0d1} & \sigma_{0d2} & \cdots & \sigma_{0dd} \end{bmatrix}. \tag{5.1}$$

The user could simply assume that the data in different dimensions are not correlated if they have no information of the correlation between different features of the point; the covariance matrix can then be simplified as

$$S_0 = \begin{bmatrix} \sigma_{011} & 0 & \cdots & 0 \\ 0 & \sigma_{022} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \sigma_{0dd} \end{bmatrix}, \tag{5.2}$$

where σ_{0dd} is the initial variance of the data in dimension d. Therefore, what the user has to set for the covariance matrix are the variances of the data in each dimension and correlation between dimensions can be learned from data later on. Figure 5.2a shows a newly generated one-dimensional Gaussian distribution.

In this way, the points can be aggregated in certain extent according to the theory of GML. According to Bayesian probability theory, the posteriori probability of a point X_I assigned to cluster j is

$$P(C_j|X_i) = \frac{P(X_i|C_j)P(C_j)}{P(X_i)}, \tag{5.3}$$

where $P(X_i)$ is the probability of data point X_i occurs, $P(C_j)$ is the prior probability of cluster C_j, $P(X_i|C_j)$ is the probability of X_i occurs under cluster C_j. Since we assume the cluster is Gaussian distributed and C_j is described by $\aleph(m_j, S_j)$. $P(X_i|C_j)$ can be written as

$$P(X_i|C_j) = \frac{1}{(2\pi)^{\frac{d}{2}}\,|S_j|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(X_i-m_j)^T S_J^{-1}(X_i-m_j)\right]}. \tag{5.4}$$

Since $P(X_i)$ is the same for all clusters, a class discrimination metric is defined as[91]

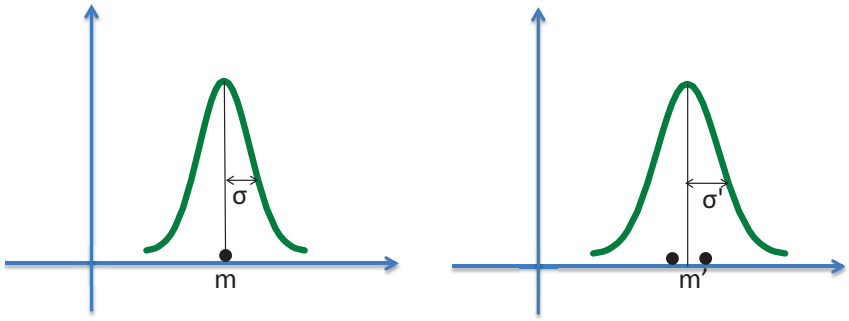
$$D(C_j|X_i) = p(X_i|C_j)P(C_j) = \frac{P(C_j)}{(2\pi)^{\frac{d}{2}}\,|S_j|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(X_i-m_j)^T S_J^{-1}(X_i-m_j)\right]}. \tag{5.5}$$

The prior probability of cluster C_j, $P(C_j)$, can be assumed to be equal among all clusters when there is no information about it. Therefore, in this case, the class discrimination matrix can be further simplified into

$$D(C_j|X_i) = \frac{1}{(2\pi)^{\frac{d}{2}}\,|S_j|^{\frac{1}{2}}} e^{\left[-\frac{1}{2}(X_i-m_j)^T S_J^{-1}(X_i-m_j)\right]}. \tag{5.6}$$

One way to guess the prior probability of a cluster of the data stream is to accumulatively count the data points inside the cluster and divide it by the total number of data points. However, the approach does not account for the temporal occurrent pattern of the cluster, which indicates that some clusters may have higher probability to occur during a specific time period while they may have very low probability during other time periods. In the next chapter, we will investigate the temporal occurrent pattern of clusters, according to which the prior probability of a cluster will be calculated.

According to equation 5.6, the class discrimination value of the new data point as-



(a) A new-born Gaussian cluster (b) Updated Gaussian cluster

Figure 5.2: Gaussian clustering

signed to each cluster can be calculated and is represented as $Discrim = (D_1, D_2, \cdots, D_i, ..., D_n)$. The maximum value D_{max} of $Discrim$ is found as D_k. If D_{max} is larger than the threshold T_D, then the new point is assigned to cluster C_k. Suppose the old cluster has data set $Y = (Y_1, Y_2, ..., Y_d) = (X_1, X_2, ..., X_{n_{points}})^T$, where n_{points} is the number inside the cluster C_k. The mean m_k and covariance matrix S_k of data set Y will be updated into $m_k^{'}$ and $S_k^{'}$, which can be represented as

$$m_k^{'} = \frac{1}{n_{points}+1} \cdot (m_k \cdot n_{points} + X_i), \tag{5.7}$$

$$S_k^{'} = \begin{bmatrix} cov'(Y_1,Y_1) & cov'(Y_1,Y_2) & \cdots & cov'(Y_1,Y_d) \\ cov'(Y_2,Y_1) & cov'(Y_2,Y_2) & \cdots & cov'(Y_2,Y_d) \\ \vdots & \vdots & \vdots & \vdots \\ cov'(Y_d,Y_1) & cov'(Y_d,Y_2) & \cdots & cov'(Y_d,Y_d) \end{bmatrix}, \tag{5.8}$$

and

$$\begin{aligned} cov'(Y_q, Y_p) &= E[(Y_q - \mu_q)(Y_p - \mu_p)] \\ &= \frac{n_{points}}{n_{points}+1} cov(Y_q, Y_p) + \frac{1}{n_{points}+1}(X_{iq} - m_{kq}^{'})(X_{ip} - m_{kp}^{'}), \end{aligned} \tag{5.9}$$

where p, q are any two dimension index, and μ_p, μ_q are the mean value of the data in dimension p and q.

5.2.2 Rule-based merging

With Gaussian assumption of the initial cluster shape, the data points are aggregated into clusters. However, due to the unexpected order of the data points coming to the system, the points belonging to the same cluster may look far away from each other in the beginning of clustering, which results in creating more clusters than needed. Over time, more and more data points are collected and fill in the gaps between two clusters and the clusters may even overlap as shown in Figure 5.3. This requires the merging step to further aggregate the clusters when more data points are acquired.

Figure 5.3: Overlapped clusters

Therefore, with each incoming data point X_i, following the GML step, a rule based merging is used. It is done by first checking the neighborhood of point X_i to see how many clusters there are in the neighborhood of that point. The size of the neighborhood is defined by a circle (or hyper-sphere for high dimension data) which is drawn with the data point X_i as the center and threshold $T_{neighborhood}$ as radius, as shown in Figure 5.4a. The cluster list inside the neighborhood is found by the function

$$\{C_j | j \in \Omega_{X_i}\} = ClusterIndList(X_i, T_{neighborhood}), \tag{5.10}$$

where Ω_{X_i} is the cluster index set in the neighborhood of data point X_i.

If there are multiple cluster types in the neighborhood, we need to consider whether to merge the clusters detected inside the neighborhood according to the merging rules. There are several rules which could restrict the clusters to merge. The first one is to test whether the merged cluster will be too big, because in some applications we know that the cluster could not be bigger than certain threshold $TR_{bigCluster}$. If the merged cluster is too big, then the two clusters will not be merged. However, if the cluster size is not an issue, which can be any size, $TR_{bigCluster}$ can be set as infinity so that this restriction can be safely ignored. The second rule is whether the two clusters are tight enough or whether they are highly inseparable. This is determined by counting the number of the points of the cluster in the neighborhood, which can be represented as

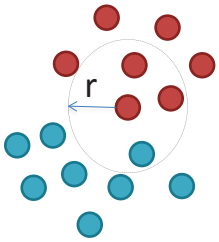
$$\{N_j | j \in \Omega_{X_i}\} = NumOfPoints(\{C_j | j \in \Omega_{X_i}\}, X_i, T_{neighborhood}). \tag{5.11}$$

If N_j is larger than the threshold TN_{fuse}, cluster C_j and cluster C_x (C_x: the cluster which X_i is assigned to) will be merged. By setting higher TN_{fuse}, the single-linkage problem of common hierarchical clustering methods can be resolved. For example, if $TN_{fuse} = 1$, clusters in both cases in Figure 5.4a and Figure 5.4b will be merged into clusters as shown in Figure 5.5a and Figure 5.5b respectively; while if $TN_{fuse} = 2$, only clusters in Figure 5.4b will be merged.

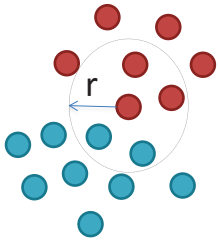
In order to speed up the algorithm, a simple method can be used to count the number of points in the cluster in the neighborhood. Since only the clusters near the point X_i have a high probability of being in the neighborhood of X_i, it is better to search for the K nearest clusters rather than to check all the data points. Therefore, firstly the K nearest clusters are found by comparing the center of the clusters as

$$\{C_j | j \in \Psi_{X_i}\} = KNearestClusters(\{C_j | j \in \Omega\}, X_i, K), \tag{5.12}$$

where Ψ_{X_i} is the cluster index set of K nearest clusters of X_i, Ω is the index set of all clusters, K is the number of nearest clusters. Following that, only the data

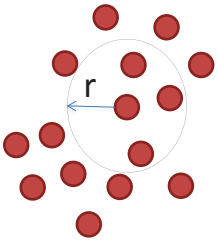


(a) Neighborhood of a data point with $T_{neighborhood}$ = r and N_j=1

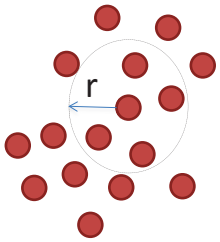


(b) Neighborhood of a data point with $T_{neighborhood}$ = r and N_j=2

Figure 5.4: Neighborhood of the new incoming point



(a) Merged Cluster with neighborhood $T_{neighborhood}$ = r and $TN_{fuse} = 1$



(b) Merged Cluster with neighborhood $T_{neighborhood}$ = r and $TN_{fuse} = 2$

Figure 5.5: Merged clusters

points belonging to $\{C_j | j \in \Psi_{X_i}\}$ will be checked to see whether they are inside the neighborhood of X_i.

When two clusters are merged, the mean and covariance of the combined cluster are calculated according equation 5.7 and 5.8 by iterating each point in the cluster to be merged.

5.2.3 Cluster descriptions

When too many data points are assigned to the cluster, it is too expensive to represent the cluster with all of the data points belonging to that cluster. Although

the cluster is assumed to be Gaussian distributed and the mean and covariance of the cluster are updated with each new data point added or a group of points added from the merging process, the cluster may not be Gaussian distributed in reality. Therefore, we can not rely on the Gaussian description of the cluster, which only offers us a description of the spatial size and location of the cluster. A better cluster description method should be used to represent the arbitrary shape of the cluster.

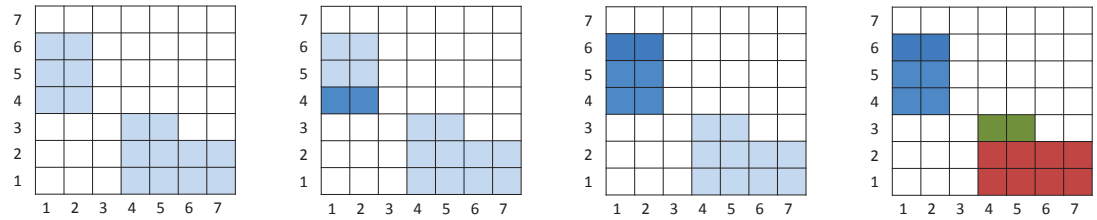
When the number of points belonging to that cluster is larger than a threshold TN_{mature}, we regard the cluster as mature and the mature flag of the cluster is marked as 1, which will trigger the cluster description algorithm.

As mentioned earlier, the approach used in this thesis to generate the cluster description is similar to the one used in CLIQUE[6]. The goal is to find the minimal number of rectangles for two dimensional data (or hyper-rectangles for high dimensional data) to describe the shape of the cluster. In CLIQUE, two steps are adopted to generate the cluster description. The first step is to greedily cover the cluster with a number of maximum rectangles. The second step is to discard the redundant rectangles to generate a minimal cover.

Our approach is different in two ways. First, the method of identifying the minimal number of rectangles is different. Second, aside from the rectangles, the residual points which are not represented by the rectangles are also kept as a supplemental description of the cluster. A detailed illustration is shown below.

Suppose the cluster has range r_i in dimension i, which is the length from the minimum value to the maximum value of the cluster in dimension i, the range of the cluster in all dimensions are represented as the vector $Range[r_1, r_2, ..., r_i, ..., r_d]$. By setting a fixed cell size $size_{cell}[s_1, s_2, ...s_i, ..., s_d]$ for all dimensions, each dimension of the mature cluster can be divided into n_i parts, where $n_i = \frac{r_i}{s_i}$. The number of points inside each cell is counted. If the number inside the cell is higher than the threshold TN_{dense}, the cell is marked as dense.

After that, the minimal number of rectangles or hyper-rectangles should be found to describe the dense cells. We start with a cell with the minimum cell index in the first dimension. Along each dimension, the maximum connected segments are found and taken as the base for the next dimension. Take a group of dense cells as shown



(a) Original cells with dense cells marked as light blue

(b) A line segment found as base (marked as dark blue)

(c) The rectangle found (marked as dark blue)

(d) Final rectangle representation of dense cells

Figure 5.6: Finding the minimal number of rectangles

in Figure 5.6a for an example, along the first dimension, a line segment is found as shown in Figure 5.6b. The line segment is taken as the base to scan over the next dimension, from which a plane segment will be found as shown in Figure 5.6c. The plane segment will be used to search for a cubic segment (here only two dimensional cells are used as example in the figures; therefore the search for the cubic segment is not needed), and so on until all the dimensions are scanned. A maximum hyper-rectangles is then generated to represent the portion of the dense cells in the cluster. The next round of the scan will be performed until all the dense cells in the cluster are checked and represented by hyper-rectangles. The final rectangle representation of dense cells is shown in Figure 5.6d.

All the points in the non-dense cells are kept as representative points of the cluster together with the rectangles or hyper-rectangles to describe the cluster as shown in Figure 5.7. Before the cluster is mature, all the points assigned to that cluster will be kept as the residual points, and the cluster is described by those residual points.

5.2.4 Cluster data management and parameters

For the description of the method above, we could see there are several different types of information about the cluster, including mean, covariance, mature flag,

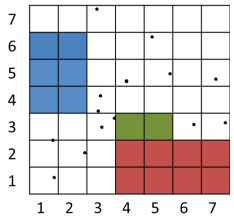


Figure 5.7: Final description of clusters

number of points, rectangles, residual points and so on. To better maintain the information of the cluster, each cluster is regarded as a structure which carries all the different types of information. Apart from the information types listed above, we may also want to keep track of the occurring time of the cluster, defined as when a point comes to the system and it is assigned to the cluster, as a time list of the cluster.

It seems that we have used a lot of parameters, such as thresholds, to control the algorithm, which is generally undesirable. However, empirically we have observed that the result of algorithm is not sensitive to most of them. To sum up, there are 7 parameters: 1) prior variance of the data in each dimension S_0; 2) threshold T_D of discrimination value for GML; 3) threshold $T_{neighborhood}$ of the radius of the neighborhood; 4) maximum size of the cluster in each dimension $TR_{bigCluster}$; 5) threshold TN_{fuse} of tightness; 6) the number of points to define a mature cluster TN_{mature}; 7) The sub cell size:$size_{cell}[s_1, s_2...., s_d]$; and 8) threshold of number of points in a cell TN_{dense}, to define it is a dense cell.

We observed that a useful principle of generating good results with the proposed method is to have small prior variances of the data in each dimension S_0, high threshold T_D of discrimination value for GML, large number of points to define a mature cluster TN_{mature}, small sub cell size $size_{cell}[s_1, s_2...., s_d]$, high threshold of number of points in a cell TN_{dense}, and small threshold $T_{neighborhood}$ of the radius of the neighborhood. By doing this, the speed of the algorithm may be comprised because it will slow the merging process and extend the time of a cluster to be mature. We can also understand this as that we let the time to form a cluster and it is of course that more data over time will give more confidence to the process of

clustering.

In some cases, the size of the cluster is not limited; therefore we can set the maximum size of the cluster in each dimension $TR_{bigCluster}$ as infinity. However, in some cases, we may have some knowledge that the cluster should not be larger than certain threshold $TR_{bigCluster}$. For example in the Edinburgh data set we will use later, we know a general size of the entrance of the building or the room. Therefore, when we cluster the start point or end point of the trajectories, we could set the maximum size of the cluster in each dimension as $TR_{bigCluster}$ to avoid the chance of incorrect merging.

5.2.5 Short-comes and possible solutions

As someone may already notice that our method of incremental clustering does not have the step of cluster splitting, which means that when two clusters are merged, there is "no way back". This may cause some problems in some situations.

However, we have several reasons for not including splitting in the algorithm. First, including splitting in the algorithm will of course increase the computational load. Second, it will be very complicated to keep track of the occurrence time list of the cluster. If we really want to do that, all the historical points should be stored so that the correspondent occurrent time list of the split cluster could be recognized. This will require extra memory for storing all the historical points. Third, we have a possible solution of avoiding the problem due to no splitting in the algorithm.

As we talked above about the parameters, there are several parameters which could be tuned to make the algorithm robust when it is time to merge two clusters or assign a data point to an existing cluster rather than create a new cluster. Those parameters are: 1) prior variance of the data in each dimension S_0; 2) threshold T_D of discrimination value for GML; 3) threshold $T_{neighborhood}$ of the radius of the neighborhood; 4) maximum size of the cluster in each dimension $TR_{bigCluster}$. Smaller prior variance of the data in each dimension and higher threshold T_D of discrimination value make the new incoming data point more likely to be classified as a new cluster rather than assigned to an existing cluster. Only if the new data point is

so obviously close to an existing cluster center, it is then assigned to the existing cluster. Smaller $T_{neighborhood}$ will also decrease the chance of merging two clusters. In addition, threshold $TR_{bigCluster}$ will stop two clusters merging when they are too large. In summary, there are parameters which can be tuned to merge two clusters quite safely.

The results on simulated data and real data shows that our algorithm without splitting are generally robust.

5.3 Result on simulated data

5.3.1 Simulated Gaussian distributed data

Two Gaussian distributed datasets are randomly generated and are incrementally fed into the system one point at a time. The incremental clustering result is shown in Figure 5.8. As we could see, the algorithm automatically aggregates the clusters carefully. With more data coming, the system can finally determine the true cluster structure. The rectangles plotted on the figure are the cluster descriptors as we illustrated above to represent the cluster when the cluster is mature.

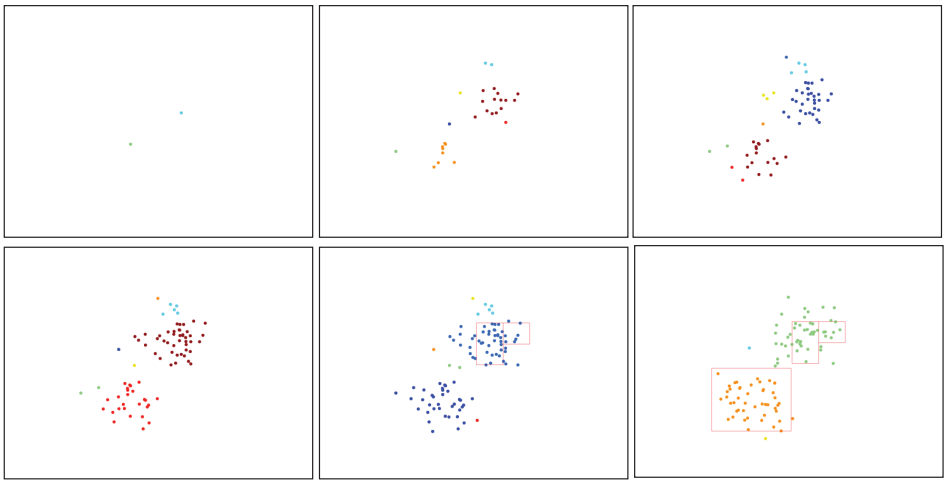
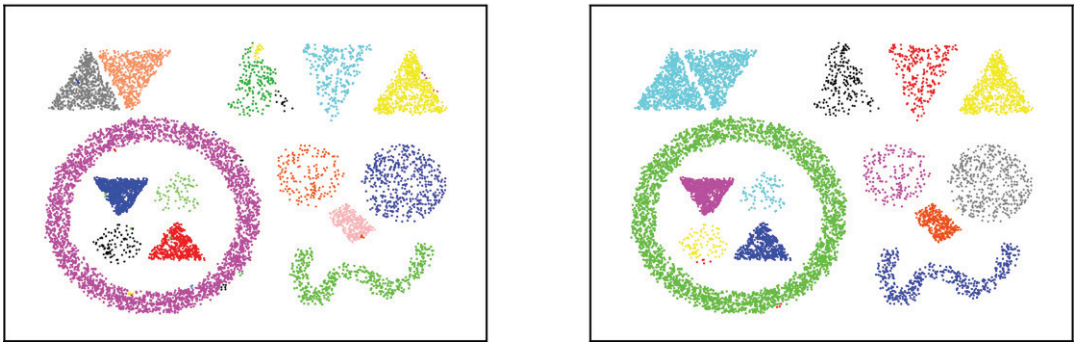


Figure 5.8: Result of incremental clustering on a data set with two synthetic Gaussian clusters

5.3.2 Simulated data set with arbitrary shapes

This simulated data set with arbitrary shapes is from [69] where a graph based incremental clustering method, RepStream, is presented by using representative points to cluster new incoming points and retain useful cluster information in the knowledge repository. The results of RepStream with several different parameters, including neighborhood connectivity value k and density scaler value α, are shown in Figure 5.9.



(a) k=4 (b) k=5

Figure 5.9: Clustering result of RepStream on points with arbitrary shapes with different neighborhood connectivity value k and density scaler value $\alpha = 4.0$

The result using our method is shown in Figure 5.10.

By comparison as shown in Figure 5.10, we see that our GMD method performs comparably to, if not better than, RepStream in clustering. The result of our method GMD shown in Figure 5.10 is similar to Figure 5.9 (b). Both methods mistakenly merge the two triangle clusters in the upper left corner. Our method has one more falsely clustered point in the third triangle in the up part of the picture. However, RepStream has more falsely clustered points in the left corner of the picture, for example, points from the ring and points from the sphere inside the ring.

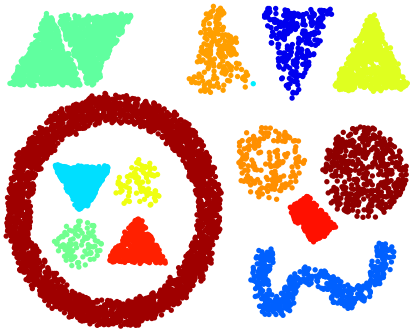
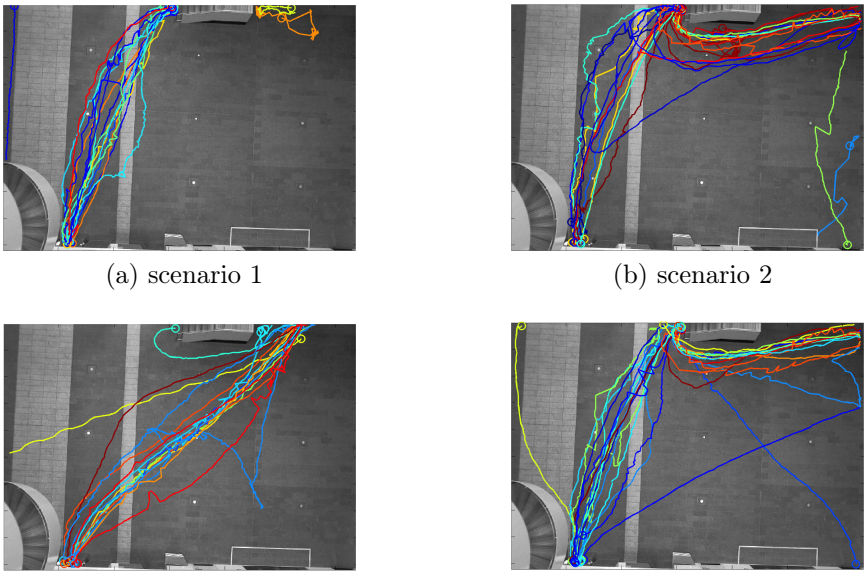


Figure 5.10: Result of incremental clustering on a data set with arbitrary shapes by the proposed GMD method

5.4 Result on Edinburgh pedestrian dataset

At different times, there may be different scenarios present in the scene. For example, there are more people walking in certain directions than others. Suppose we have four scenarios of the scene as shown in Figure 5.11. Our goal is to distinguish these scenarios automatically.

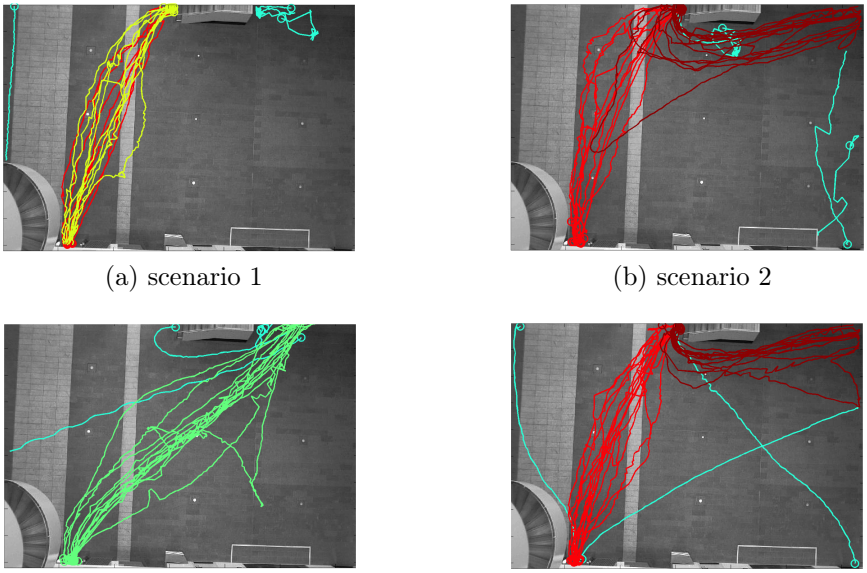


(a) scenario 1 (b) scenario 2

(c) scenario 3 (d) scenario 4

Figure 5.11: Four different scenarios of the scene

Intuitively, we want to count the number of trajectories performing different activities, as shown in Figure 5.12. We assume people walking from different start points to different end points are performing different activities. These number of trajectories with different activities are then used to distinguish different scenarios of the scene by our incremental clustering method, GMD. Now the problem is how to cluster trajectories into different activities, which will also be solved by using the GMD method we proposed above.



(a) scenario 1

(b) scenario 2

(c) scenario 3

(d) scenario 4

Figure 5.12: Four different scenarios of the scene with trajectories in different direction labelled

Therefore, in this thesis, we use two steps of clustering to finally distinguish the scenarios during different time as shown in Figure 5.13. The first step is to cluster the trajectories into different activities. The number of occurrences of each activity is additionally used to cluster the scenarios of the scene. In this thesis, different scenarios of the scene are considered as different events occurred to the scene, and only one event type is used to describe the scene during one period.

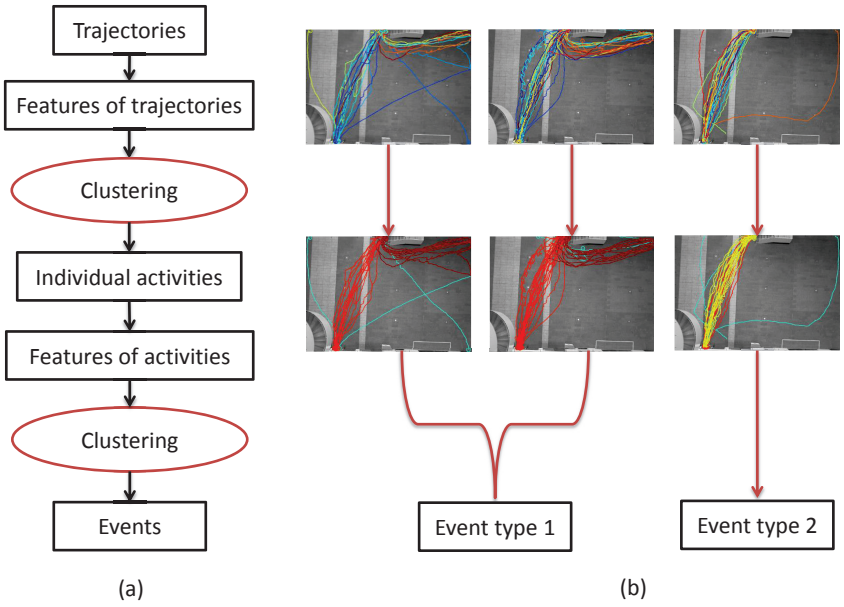


Figure 5.13: The workflow of two step clustering

5.4.1 Incremental clustering of trajectories

5.4.1.1 Review on clustering of trajectories

The author who generates this Edinburgh pedestrian dataset builds a model of normal behavior based on the collected trajectories by using Gaussian mixture models. The trajectories are approximated by cubic spline curves and corresponding probability scores are generated by the model to access the likelihood of them to be normal behaviors[71].

The family of Dirichlet process related methods can be found in the application of trajectory clustering. Wang et al.(2011)[109] proposed a nonparametric Bayesian model called Dual Hierarchical Dirichlet Processes(Dual-HDP), where trajectories are treated as documents and observations of an object on a trajectory are treated as words in documents, in order to cluster the trajectories, detect abnormal trajectories and model semantic regions. A non-parametric Bayesian model is used by Kooij et al.(2012)[58] to jointly discover the dynamics of low-level actions and high-level behaviors of the tracked people. Kooij et al. use Markov chains of actions to

capture high-level temporal dynamics and use switching linear dynamics systems to represent low-level motion dynamics. Kuettel el al.(2010) [63] present a method that builds on Hierarchical Dirichlet Processes and learn dependencies between the motion patterns to find the local temporal rules. In order to find global temporal rules, Kuettel el al. use DDP-HMM (Dependent Dirichlet Processes-Hidden Markov Model) to jointly learn co-occurring activities and their time dependencies.

Density-based methods are found to be used to cluster trajectories [9, 12, 88]. Akasapu(2011)[9] uses a relative density-based clustering algorithm RDBKNN described in Liu(2003)[68] to cluster the trajectories. OPTICS[13] which is an extension of DBSCAN[36] with an infinite number of distance parameters ε_i is used in Andrienko(2007)[12] and Rinzivillo(2008)[88] to cluster the trips in the city in order to create a visual analytics tool. Fu(2005)[39] uses predefined template trajectories to compare the similarity of the vehicle trajectory in order to cluster the trajectory and detect anomalies. Atev(2010)[15] combines ideas from two spectral clustering methods and use the trajectory-similarity measure based on the modified Hausdorff distance. Gaussian process regression flow is used by Kim(2011)[57] to model the velocity vector mean and variance, which is then adopted to classify the incoming trajectory by comparing with the template trajectories. Stauffer(2000)[96] uses on-line Vector Quantization to generate a codebook of prototype representations of the trajectory and then accumulate joint co-occurrences of representations, which will be adopted later to create a hierarchical binary-tree classification of the representations.

However, to our best knowledge, there are no reported algorithms for incremental clustering of trajectories to date.

5.4.1.2 Feature selection for trajectory clustering

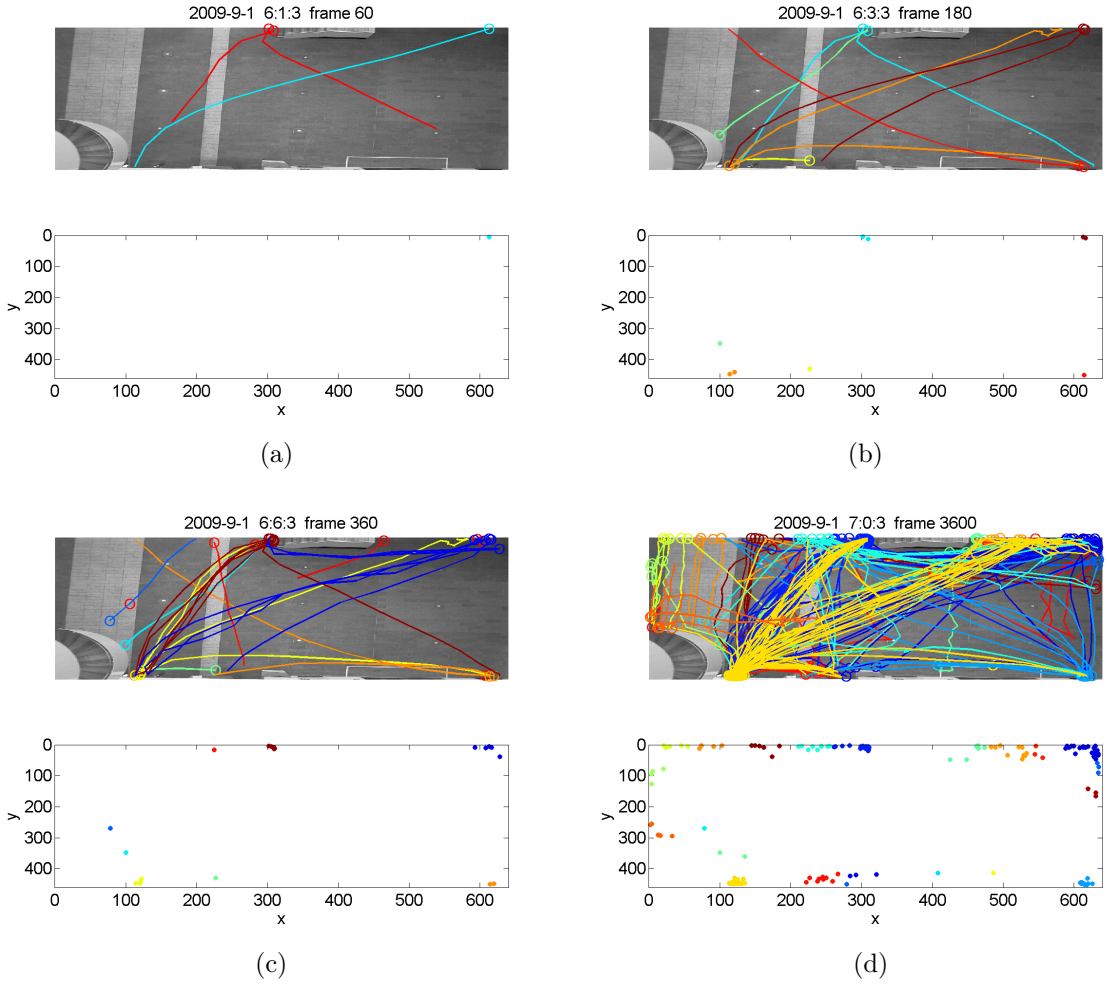
For the Edinburgh pedestrian data set, a certain trajectory may be categorized into different activities based on different interests. Example features which describe the characteristics of a trajectory can be the speed, direction, wandering time on certain spots over the all trajectory, and the start and end point. With the speed feature,

we can detect people who walk too fast, too slow, or change walking speed all the time. With the direction feature, we can detect people who change destinations or who are new to this place. With the wandering time feature on certain spots, we can detect people who are waiting or talking with others. Some activities may need a combination of features, which depends what the interesting events are for the specific applications.

In this part of thesis, in order to visualize the effectiveness of the clustering method, we use the start point and the end point of the trajectory as features. By using the start point as the feature, the feature vector of trajectory i can be represented as: $[x_{i0}, y_{i0}]$, where 0 represents the first point of the trajectory. By using the start point and the end point as the feature, the feature vector of trajectory i can be represented as: $[x_{i0}, y_{i0}, x_{ie}, y_{ie}]$, where e represents the last point of the trajectory. This means that each trajectory in the data set will be represented by this four dimensional feature vector for analysis.

5.4.1.3 Results of the trajectory clustering

By only using the start point of the trajectory as the feature, the result of incrementally clustering of trajectories at different time stamps is shown in Figure 5.14. The upper picture in each sub-figure is the result in the trajectory domain; while the lower picture is the result in the feature domain, from which we can also infer the semantic area of the scene. The result in the trajectory domain is the clusters of trajectories which share the same start area. To better visualize the clusters in the trajectory domain, the clusters with top 9 number of trajectories sharing the same start area within one hour of data at frame 3601 are shown individually in different figures in Figure 5.15.



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

Figure 5.14: Incremental clustering of trajectories with start points as features

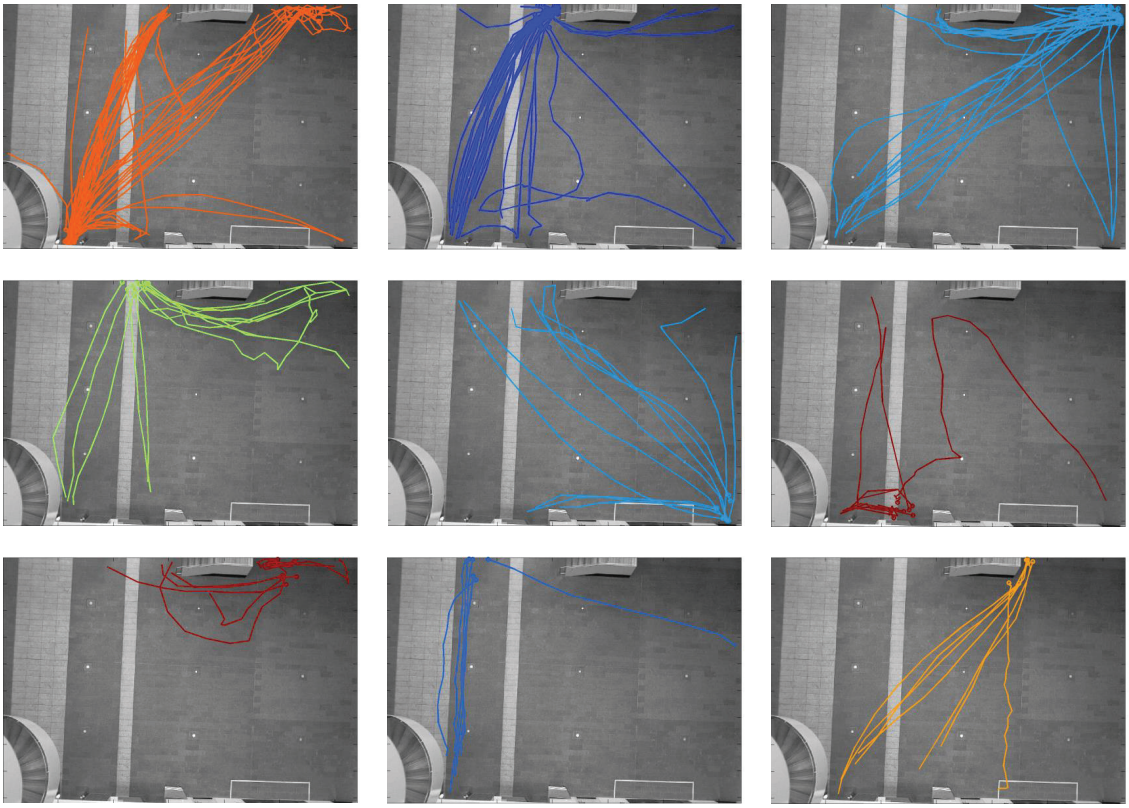


Figure 5.15: The clusters with top 9 number of trajectories sharing the same start area within one hour of data

By using both the start point and the end point of the trajectory, the trajectories which share the same start area and end area are incrementally clustered into one cluster over time as shown in Figure 5.16. The lower left sub-figure in each Figure 5.16 is the start point feature space, and the lower right sub-figure is the end point feature space.

To better visualize the clusters in the trajectory domain, the clusters with top 9 number of trajectories sharing the same start and end areas within one hour of data are shown individually in different figures in Figure 5.17.

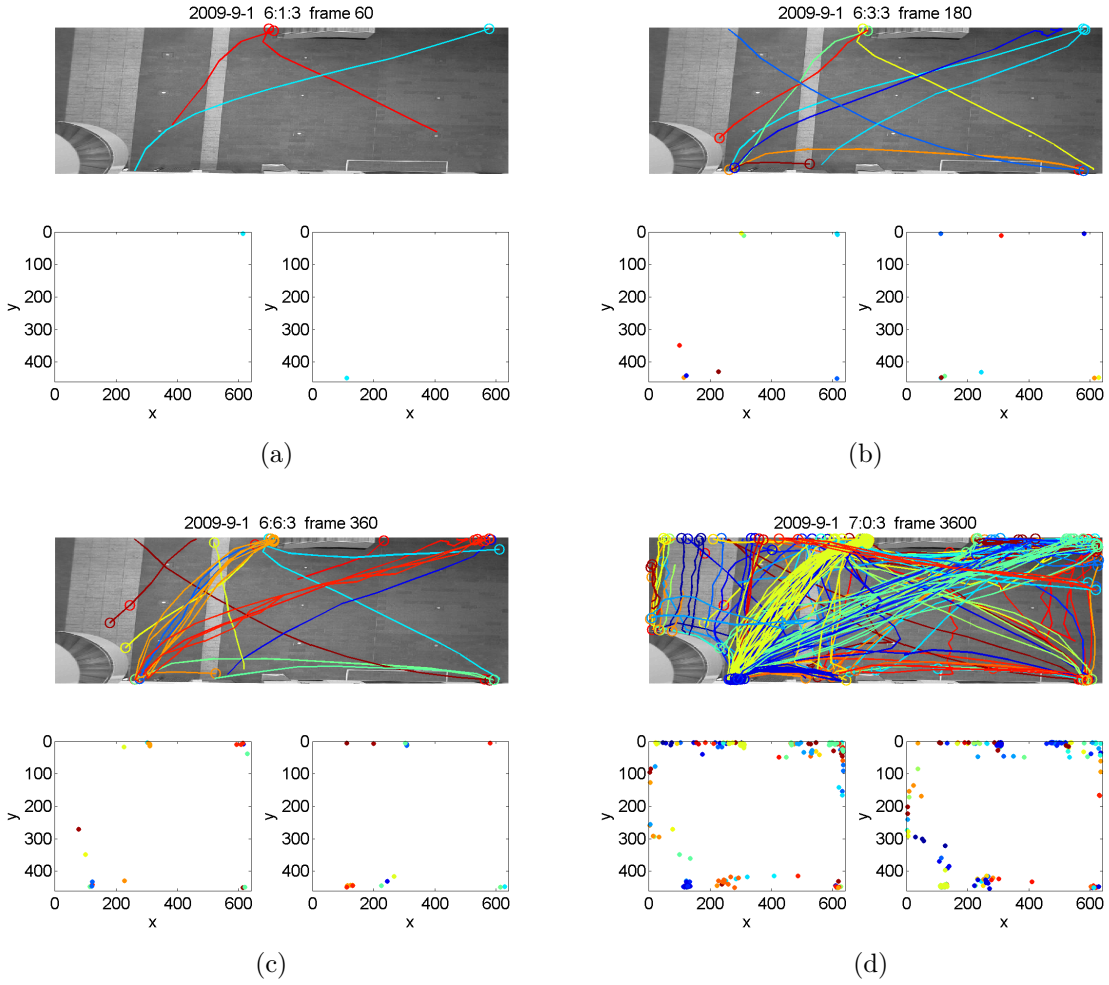


Figure 5.16: Incremental clustering of trajectories with start and end points as features

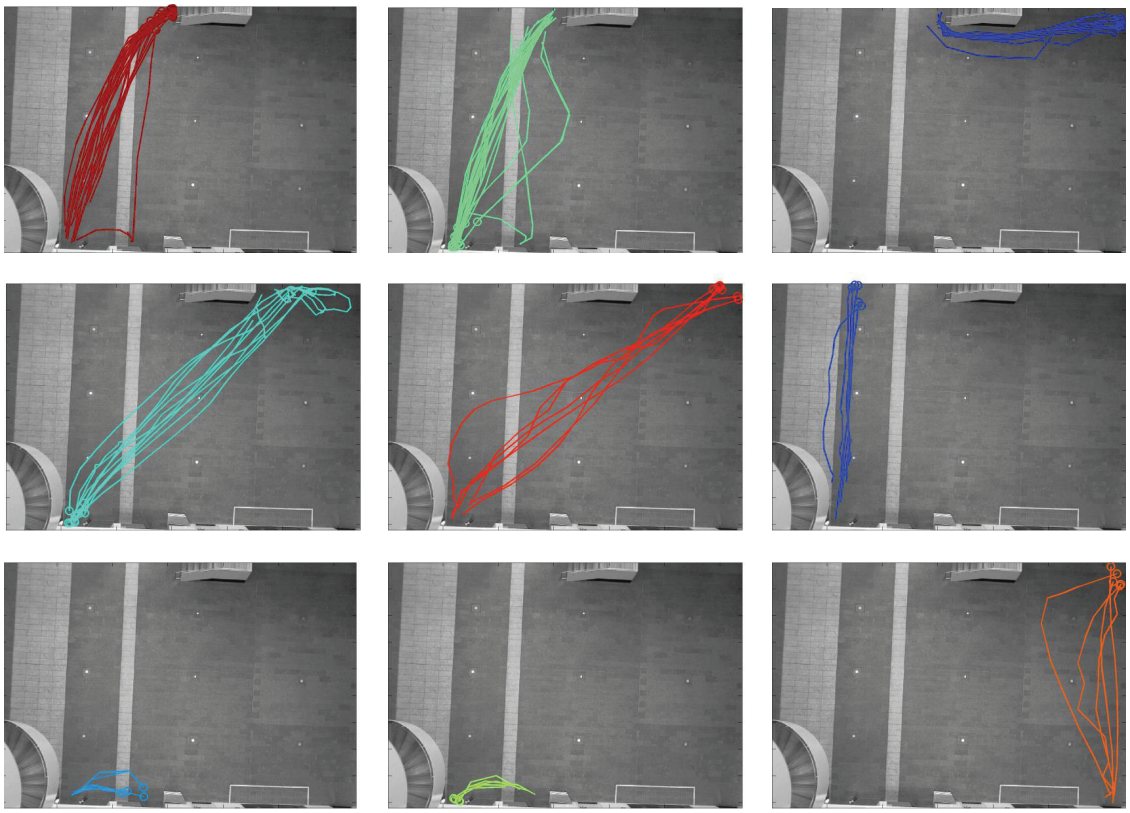


Figure 5.17: The clusters with top 9 number of trajectories sharing the same start and end areas within one hour of data at frame 3601

5.4.2 Byproduct of clustering trajectories: extraction of semantic areas

By clustering the trajectories with either start points or end points as features, or with both start points and end points as features, we can cluster these feature points into clusters $\{C_1, C_2, ..., C_i, ..., C_N\}$, where N is the number of clusters found so far, and $C_i = \aleph(X_i, S_i)$. If both start points and end points are used as features, each cluster of the features is then a pair of areas where people have the possibility of entering and leaving the Edinburgh forum. These areas are considered as semantic areas which tell meaningful structure of the forum. By using the clusters found from the previous step, we then identify these semantic areas. Suppose the feature vector

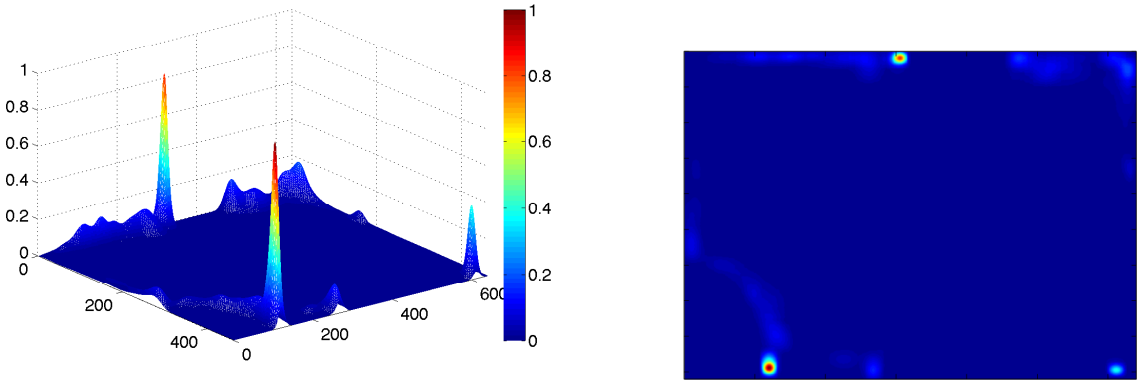
X_i is represented as $[x_{i0}, y_{i0}, x_{ie}, y_{ie}]$, and the covariance matrix is represented as

$$S_i = \begin{bmatrix} \sigma_{i11} & \sigma_{i12} & \sigma_{i13} & \sigma_{i14} \\ \sigma_{i21} & \sigma_{i22} & \sigma_{i23} & \sigma_{i24} \\ \sigma_{i31} & \sigma_{i32} & \sigma_{i33} & \sigma_{i34} \\ \sigma_{i41} & \sigma_{i42} & \sigma_{i43} & \sigma_{i44} \end{bmatrix}. \tag{5.13}$$

The start point cluster and end point cluster could then be separated from the four feature cluster. The mean and covariance of the start point cluster can be represented as:$X_{is} = [x_{i0}, y_{i0}]$ and $S_{is} = \begin{bmatrix} \sigma_{i11} & \sigma_{i12} \\ \sigma_{i21} & \sigma_{i22} \end{bmatrix}$ respectively; while the mean and covariance of the end point cluster can be represented as $X_{ie} = [x_{ie}, y_{ie}]$ and $S_{ie} = \begin{bmatrix} \sigma_{i33} & \sigma_{i34} \\ \sigma_{i43} & \sigma_{i44} \end{bmatrix}$ respectively. In order to find the distribution map $Map_{semantic}$ of the semantic areas, all the clusters of start points and end points are summed and weighted by the number n_i of the feature points in each cluster as

$$Map_{semantic} = \sum_{i=1}^{N} n_i \times \aleph(X_{is}, S_{is}) + \sum_{i=1}^{N} n_i \times \aleph(X_{ie}, S_{ie}). \tag{5.14}$$

The distribution map of semantic areas is shown in Figure 5.18. The peaks of the distribution are found and labeled as semantic areas in the order of significance as shown in Figure 5.19. The semantic areas not only tell the meaningful physical structure of the scene, but also could help to further cluster the events, which will be explained in the next section.



(a) (b)

Figure 5.18: The distribution of semantic areas

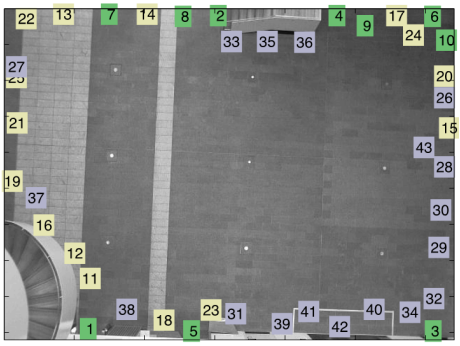


Figure 5.19: The scene with semantic areas labeled

5.4.3 Incremental clustering of events

As we mentioned before, the trajectories are first clustered into different activity types. Suppose activity types are represented as $[A_1, A_2, ..., A_i, ..., A_n]$, where n is the number of all activity types. All these activity types form a space Ω_A with each dimension describing the property of an activity type. One property of an activity type in this thesis is the number of occurrences of the activity. Suppose during time period $[t, t + \triangle t]$, the number of occurrences of each activity in space Ω_A is $[m_1, m_2, ..., m_i, ..., m_n]$ respectively. Theoretically, the property values of all the activity types should be used to distinguish one point from the other in space

Ω_A. However, in real situation, most of the activities do not show up very often. Figure 5.20 shows the number of occurrences of all the detected activities over approximately 100 days of data from the Edinburgh data set and Figure 5.21 shows the cumulative summation of the occurrence number of all the activities. There are in total 728 distinct activity types found, but we could see that only the first few activities have significant numbers of occurrences and there are more than 76,160 trajectories fed into the system. Those activities with low occurrences are useful in detecting anomaly events. In order to better visualize and understand the event clusters as the first step, fewer number of events is better. Therefore, in this thesis, we will only consider the top 5 activities as an example to show how incremental clustering method GMD is used to cluster events. These events, because of their common occurrences, could also be used to describe the "normal" activity in the scene.

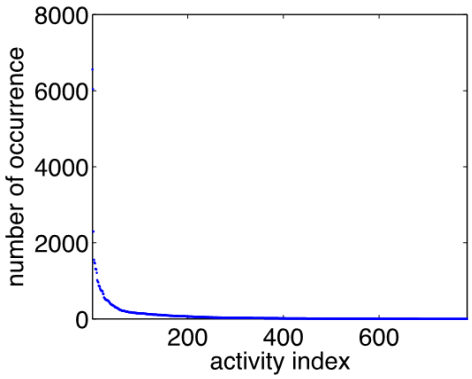


Figure 5.20: The number of occurrences of all activities

One more issue we need to mention is that in the beginning, the activity cluster indices are not stable yet due to the inherent property of the GMD incremental clustering method. The cluster index can change due to merging and new cluster index will be generated because a new cluster is created. Therefore we will not necessarily have a stable space Ω_A for clustering the event. Every merging between

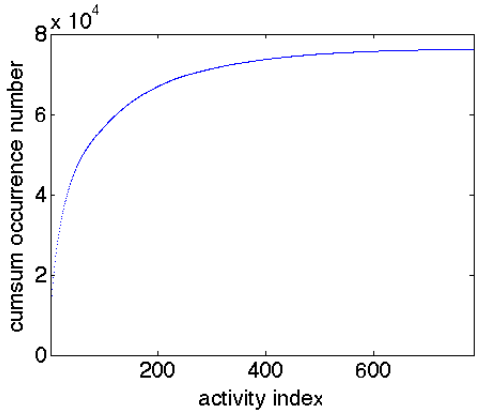


Figure 5.21: The cumulated summation of occurrence number of all activities

two activities will result in a projection in space Ω_A, and then all the event clusters in the changed space Ω_A should be redefined.

This issue could be solved with the following workflow structure (although not implemented here) as shown in Figure 5.22. Instead of redefining all the event clusters whenever there is a merging between two activities, an offline mode in addition to the online (streaming) mode (as shown in Figure 5.13) is supplied to improve the efficiency of the algorithm. In the figure, $\triangle T_1$, $\triangle T_2$ and $\triangle T_3$ are time intervals to trigger the next step. $\triangle T_1$ is the time interval to collect the features of the trajectories, which is equal to 10 seconds in this thesis. $\triangle T_2$ is the time interval to define an event, which is equal to 10 minutes in this thesis. $\triangle T_3$ is the time interval to trigger the offline mode. The suggestion value for $\triangle T_3$ could be days. Since $\triangle T_3$ is significant larger than $\triangle T_1$ and $\triangle T_2$, the procedure in the left box in Figure 5.22 is considered as offline.

The purposes of the offline procedure are: 1, offering the relatively new fixed space Ω_A for the online event clustering; 2, redefining all the historical events in the relatively new fixed space Ω_A. We say the space Ω_A is relatively new because the time interval $\triangle T_3$ to trigger the update of the space is longer than the time the space can change. As noted before, every merging of two trajectory clusters results in a change in space Ω_A.

To achieve these two purposes, the semantic areas should be found with the latest information of clusters of trajectories. Each activity A_i along with its time stamp list Tl_i is then mapped to a pair of semantic areas $AreaPair_j = (AreaPair_{j1}, AreaPair_{j2})$, where $AreaPair_{j1}$ is the start area of a type of activities and $AreaPair_{j2}$ is the end area of a type of activities. The set of area pairs is represented as $\{AreaPair_j, j = 1, 2, 3, ..., n'\}$, where n' is the number of all activity types. The set of area pairs $\{AreaPair_j, j = 1, 2, 3, ..., n'\}$ forms a new space Ω'_A for the next step of online clustering of events. At the same time, all the historical events defined in the old space Ω_A should be redefined in the new updated space Ω'_A. This is done by clustering the new property vector $[m_1, m_2, ..., m_i, ..., m_{n'}]$ in Ω'_A for each event time interval.

The offline mode is triggered every time interval $\triangle T_3$ until the space Ω_A is stable.

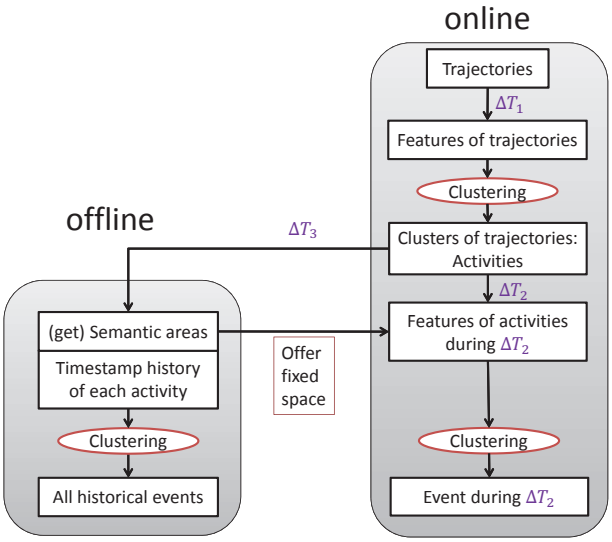


Figure 5.22: Clustering structure with online and offline modes

In this thesis, as we said, in order to better explain the result of clustering, we only consider the top 5 activities. The set of area pairs used to form the space Ω_A is $\{(2, 1), (1, 2), (1, 9), (4, 1), (2, 3)\}$, as shown in Figure 5.23.

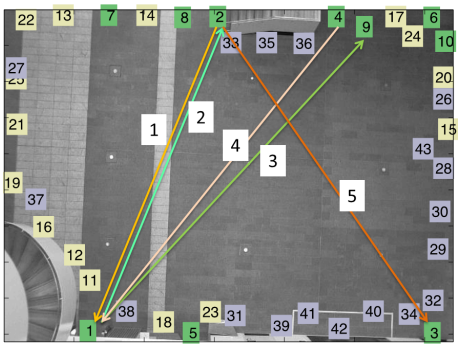
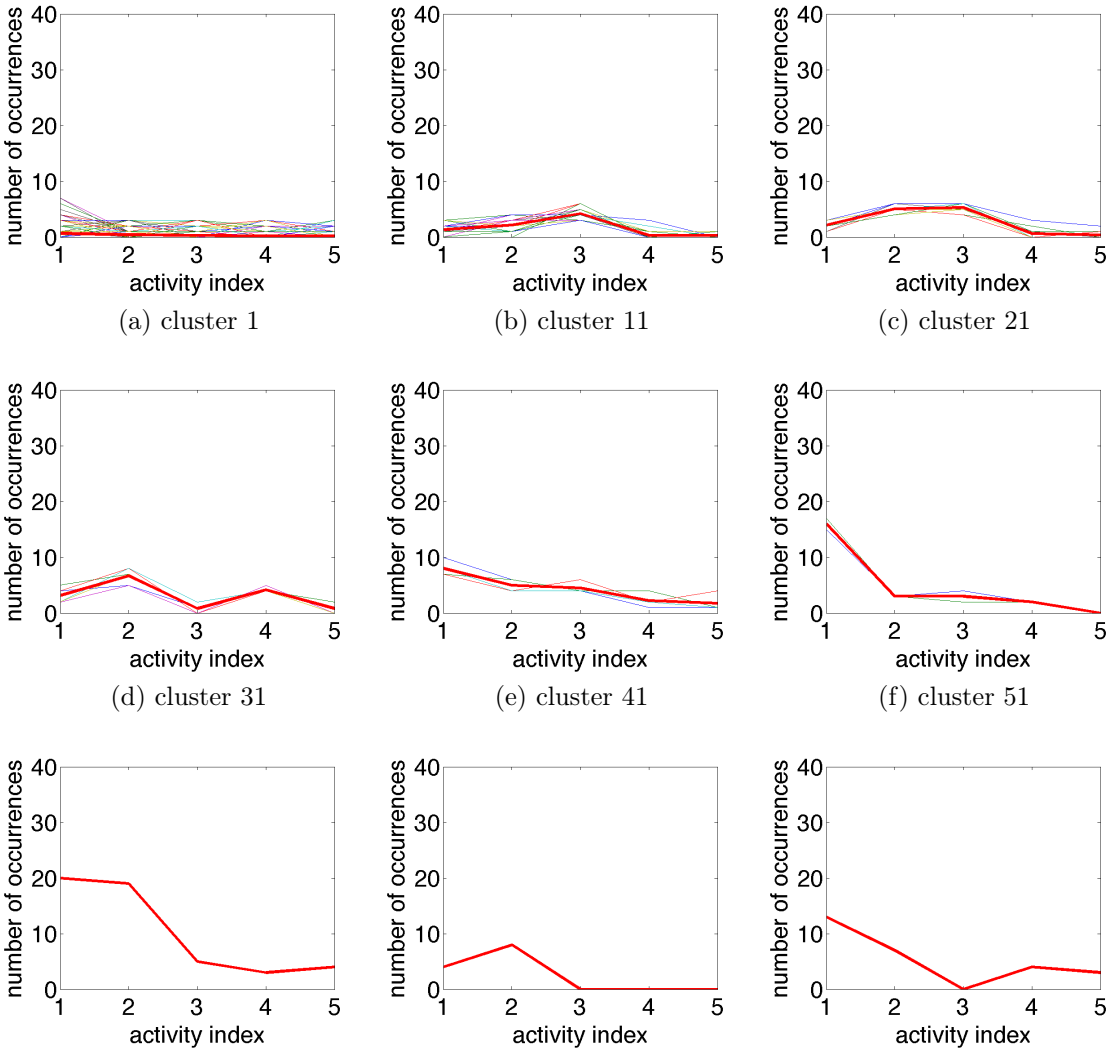


Figure 5.23: 5 activities considered in this thesis

5.4.3.1 Feature selection for event clustering and results

Each activity can have several properties, such as the number of occurrences during $\triangle t$, normality, importance score, and so on. In this thesis, we will use the number n_i of the occurrences of an activity with $AreaPair_i$ as one dimension of features to characterize an event during $\triangle t$. As we said, 5 activities are used to describe the event of the scene. Therefore, the feature vector has 5 dimensions and can be represented as $feature = [n_1, n_2, n_3, ..., n_5]$. After going through all the data, 86 types of events are found. The event clusters are sorted in a decreasing order of the number of occurrences of the event, and event clusters 1, 11, 21, 31, 41, 51, 61, 71, 81, 91 are shown in Figure 5.24. The red line in each sub figure is the mean of the cluster. We could see that different event clusters have different feature values across all the 5 activities.

However, in some cases, we do not care the exact number of occurrences of activities. We may only want to know whether there is a high number of occurrences of activities. Therefore, the raw number should be categorized into levels, such as "very low", "low", "normal", "high", "very high" and so on, by setting certain thresholds to each level. In this thesis, the categorized levels ("low", "high") will be used as features to cluster the event. After going through all the data, 16 types of events are found. The result of event clustering with categorized levels as features is shown in Figure 5.25.

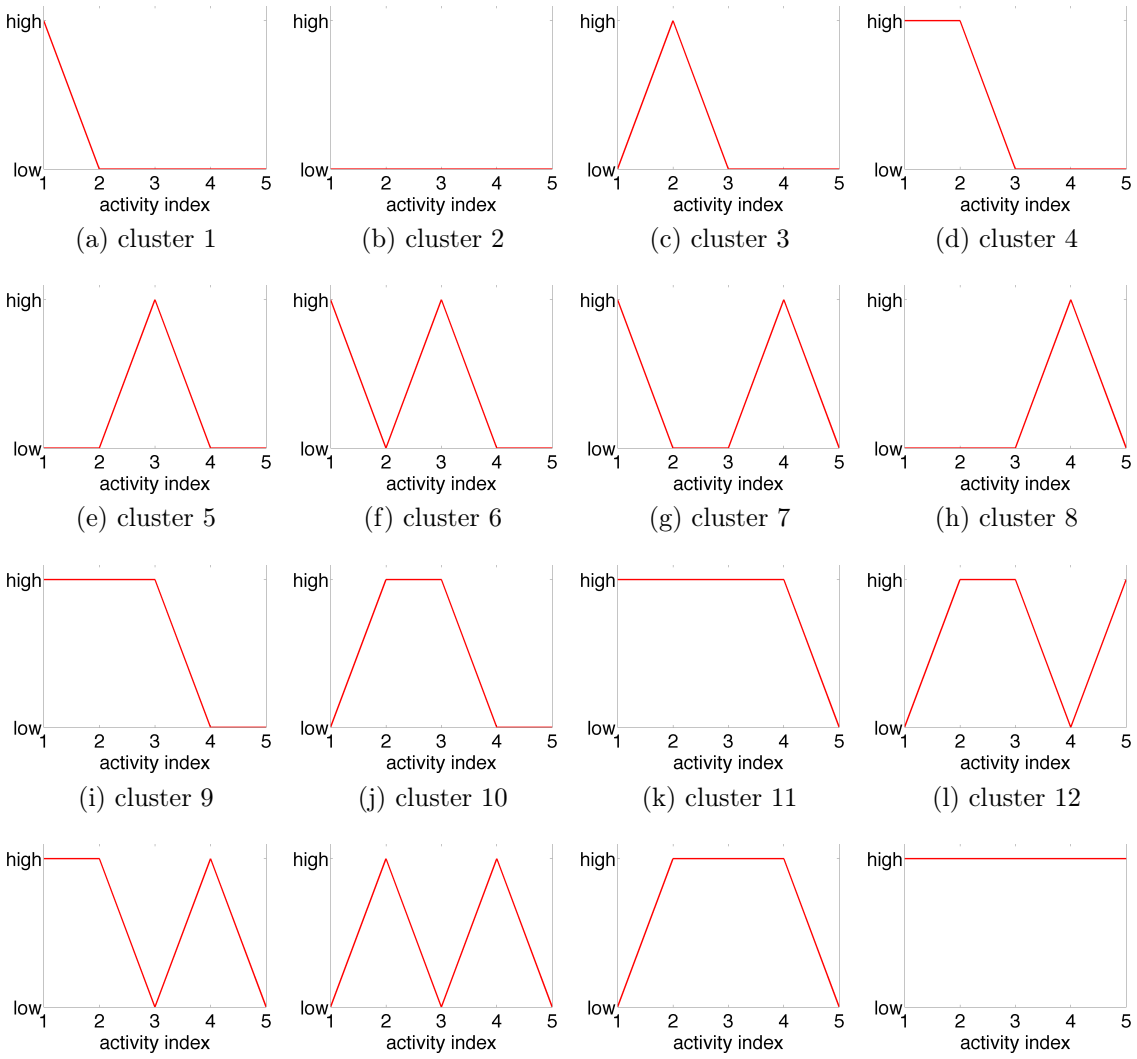


(a) cluster 1 (b) cluster 11 (c) cluster 21

(d) cluster 31 (e) cluster 41 (f) cluster 51

(g) cluster 61 (h) cluster 71 (i) cluster 81

Figure 5.24: Event clusters with raw numbers as features



(a) cluster 1 (b) cluster 2 (c) cluster 3 (d) cluster 4

(e) cluster 5 (f) cluster 6 (g) cluster 7 (h) cluster 8

(i) cluster 9 (j) cluster 10 (k) cluster 11 (l) cluster 12

(m) cluster 13 (n) cluster 14 (o) cluster 15 (p) cluster 16

Figure 5.25: Event clusters with categorized features

Chapter 6

Memorization and prediction

The previous chapter talked about recognizing an event by clustering methods. By event clustering, each measure over time is recognized as an event. That is in the temporal space, each point is associated with a certain event. In order to understand what happened in the past or predict what will happen in the future, those points should be organized and the pattern of those points should be analyzed. In this thesis, we call the process of organizing the information of events in the temporal space as memorization. By proper memorization, we could fetch the information from memory quickly and make predictions by analyzing the pattern of the memories.

In this chapter, we will talk about the method of memorization and how to use the memory to predict the future.

6.1 Memorization by double localization

6.1.1 What to memorize and how?

In order to memorize something, we may want to know: what it is, when it occurs, where it is, how it is used, how it is related to others, and so on. One group of the information is the static description of an object, which will not change over time, such as what it is. The other group of the information is the temporal aspect of

that object, such as when it occurs. The information of where it is, how it is used and how it is related to others may change over time or not, depending on a specific situation. In this thesis we only record what it is and when it occurs as an example to show the process of memorization.

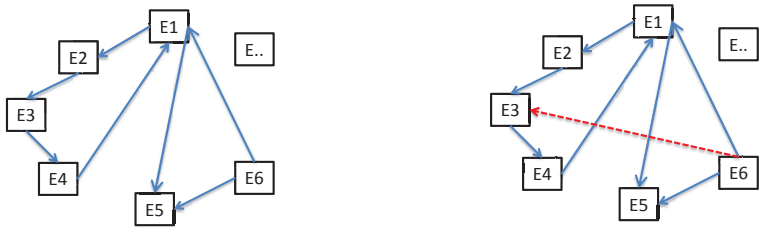
The next question is how to memorize the events. It is straightforward to memorize what the event is by recording the feature descriptions of the event as shown in Figure 5.24 and 5.25. For the information of when the event occurs, we could also simply record the timestamps every time it occurs. However, when an event occurs intensively over time, there will be too many timestamps to be efficiently understood. In order to prepare for the next step of the process, prediction, the temporal information of an event should be memorized effectively. We propose to memorize the events by double localization. A memory built with double localization tells the temporal contexts of an event, including absolute temporal context and relative temporal context.

Absolute temporal contexts seek to localize an event absolutely with its timestamps. For example, event A is a normal event and occurs very often over time. However, event A only occurs in the morning. If event A occurs in the afternoon, it is an abnormal event. We say event A did not occur in the expected temporal context. The idea of temporal maps is used to hold the temporal locations of an event, which will be explained in details later.

Relative temporal contexts are the temporal contexts between events, which tells the relative relationship between events. One simple method of modeling the temporal context between events is known as a Markov chain, telling the probability that one event occurs after another event. For example, there are several types of events, indexed as 1,2,3,...., as shown in Figure 6.1. After event 1, event 2 and event 5 could occur. After event 2, event 3 could occur and so on. Figure 6.1(a) shows the normal temporal context between events. However, when event 3 occurs after event 6 as shown in Figure 6.1(b), which is not expected under the normal temporal context as shown in Figure 6.1(a), this situation is considered as abnormal. The probability (or transition) matrix can be learned from the data and used to describe the temporal context between events. In this thesis, the Markov chain model is used

as an example to show how to localize an event with the temporal context between events. More advanced methods will be reviewed in the following section. These methods can capture deep relative relationship between events, not only relative relationship between two events.

With the two types of temporal contexts, we could say that an event is double localized (absolutely localized and relatively localized). An event is absolutely localized by its temporal map and relatively localized by the probability matrix from Markov chain model.



(a) Normal context(blue arrows indicate normal transitions)

(b) Abnormal context (red arrows indicate abnormal transitions)

Figure 6.1: Temporal context models between events

6.1.2 Related work

6.1.2.1 Absolute localization: Exploring temporal association rules

The notion of association rule is proposed to represent the co-occurrence of items in shopping transaction database[7]. In order to locate the association rules in the temporal space, the notion of temporal association rule is proposed to incorporate the time in the discovery of association rules[10]. Calendar-based methods are found to be used to describe the temporal pattern of the association rules[65][105].

The part of absolutely localizing an event in the temporal context of our proposed method is inspired by the work of exploring calendar-based temporal association rules. In order to find calendar-based temporal association, a calendar schema is defined and then all large item sets for all star calendar patterns on the given calendar

schema are found[65]. An example used in [65] for explaination is that: periodic cycle "every seven day" or "every week" can be expressed by a calendar pattern $\langle *, i \rangle$, where $1 \leqq i \leqq 7$ and $*$ can be translated as "every", under the calendar schema $R = (week, day)$ depending on which day the cycle starts. A calendar pattern with at least one wild-card symbol $*$ is called a star calendar pattern.

The large item sets at a time interval can be considered as a large number of concurrent events. However, in this thesis, only one event is used to describe one time interval. In order to transform the problem from exploring temporal association rules in shopping transactions to discovering temporal pattern of an event, we assume that each large item set can be represented by one description. Therefore, now what we need to discover is the possible star calendar patterns associated with the description. However, there are no good visualization methods used to aid to find the pattern. In this thesis, the idea of temporal maps are used as complements.

6.1.2.2 Relative localization: Modeling dynamics of trajectories or behaviors

Context-Free Grammars (CFGs) are important concepts in linguistics to describe the structure of sentences and words in natural language. CFGs are used to model the dynamics of movements of vehicles on the free way[40] and recognize human behaviors[78][90]. To conquer the limitation that CFGs require manually setting the grammars, Xu(2012)[112] adopts Liang's nonparametric model of HDP-SCFG[66], and presents an unsupervised framework to analyze the video events.

A location predictor "WhereNext" is proposed in [77] to predict with a certain level of accuracy the next location of a moving object, by building T-pattern Tree which is learned from the trajectory patterns. The T-pattern Tree works like a decision tree to predict the next location of the trajectory. However, the T-pattern Tree may have the replicate nodes, which cause redundancy of the information and also is difficult for visualization.

Besides, Hidden Markov Model (HMM) and Conditional Random Fields (CRFs) related methods are often used to model the temporal structure of videos and

texts[35, 50, 100, 95]. The learned temporal structure can be used as templates to recognize interesting dynamics (behaviors, activities, trajectories, and so on) from the sequences.

These methods offer good ways of modeling the temporal structure of certain length of sequences, which tells the relative relationship or relative order between the elements among the sequences. In order to capture the overall temporal context, additional processing is required.

6.1.2.3 Similar terminology with different meanings

By searching related work in this field, terminology "context" shows in several areas but carries different meanings. One example is in [76], the term "temporal context" is mentioned, while it means the temporal related events detected in the natural language, which are additionally converted into first order logic Suggested Upper Merged Ontology[81], such as "earlier", "during", "overlapsTemporally" and so on.

The term "context modeling" is also often used in the work of context aware applications [22, 67] where the contexts are different entities of the environment existing and influencing to the application at the same time. In [22, 67], the definition of situation is also given, which is the status of all the entities in the temporal space. The situation defined in [22, 67] is similar to the event defined in this thesis, while the relationships between situations is similar to what we want to explore in this thesis. In [22, 67], the relationships between situations are represented by Allen's temporal logic[11].

6.1.3 Temporal map

6.1.3.1 Temporal measures as sequences

When we talk about temporal measurements of an object, it is natural to have a time sequence in mind. At each time t_i, there is a measurement x_i of the object and the time t_i, where t_i linearly goes from the past to the future in one dimensional space R. The time sequence can be represented as (x_i, t_i), $i = 1, 2, 3, ..., N$, where

N is the number of measurements. In order to find interesting event patterns and their periods, one possible method is to segment the time series to locate the event periods and additionally all the sub sequences during the event periods are clustered. After that, the temporal pattern of an event is determined.

Take the Edinburgh data for an example, from the previous work of incrementally clustering, we could recognize the trajectories and identify which activity the trajectory belongs to. Therefore, we could count the number of people performing each activity during a time interval. The time interval 10 minutes is used throughout this thesis as an example. For now, suppose we are interested in a specific activity, activity 1, which is people walking from semantic area 2 to semantic area 1 (cluster 1). The period of the event is found by segmenting the time series of the number of people performing activity 1, as shown in Figure 6.2. We consider an event starts when there is an increasing number of people participating in the scene and ends when there is a decreasing number of people. Therefore, the second-derivative is applied to the time series to find the start point of a new event. So far, when we have the segmentation result of the time series as shown in Figure 6.2, we want to check the result and see whether we could find some patterns of the events. During the three day as shown in the Figure 6.2, we could find in the late morning from 10:00 to 12:00 there is an increasing number of people showing in the scene on all the three days. However, it is very difficult to visualize such patterns. In addition, we have years of data. It will be even more difficult to get some general ideas of the temporal data by just visualization.

In the following section, temporal maps are introduced to offer better visualization of the long term time series.

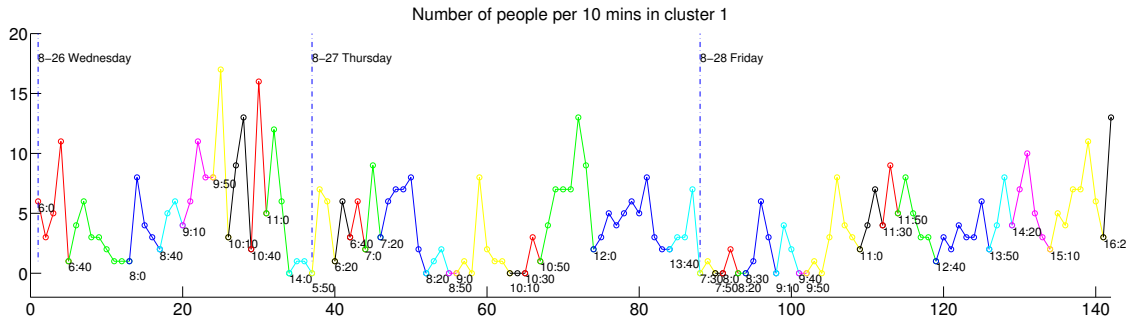


Figure 6.2: Segmentation of temporal measures in the form of sequences

6.1.3.2 Temporal measures as maps

The temporal map is a way to hold and visualize the temporal measures. In order to explain the idea of the temporal map, it is important to realize the serial time t can be represented in a multiple dimensional space. Time can be also modeled[23][8]. One way of collapsing one dimensional serial time t into several dimensions is to use a calendar (including year, month, day, weekday) and time string (including hour, minute, second). A time point t can be localized by time tags: year, month, day, hour, minute and second with the accuracy in the order of seconds. In order to better represent certain periods, additional time tags are created, such as season, quarter, decade, century and so on. These time tags are created to assist our life, and in return, people perform activities according to the time system with these time tags. Therefore, when we want to explore hidden rules or discover temporal patterns from human activity related data, it is useful to take advantage of the existing time tags.

Earlier work of visualizing time series on maps is found in [104, 82]. Van(1999)[104] first clusters the daily time series with a bottom-up clustering algorithm, and different cluster indices are shown in a calendar with different colors as shown in Figure 6.3. Nocke(2003)[82] indicates that the techniques of information visualization can be used to support the specification of a model, analysis of a model and evaluation of a model. The visualization system openDX is used as a platform and enhanced by additional functionalities. Figure 6.4, called rectangular view by [8], is used to show the effectiveness of visual exploration in model specification.

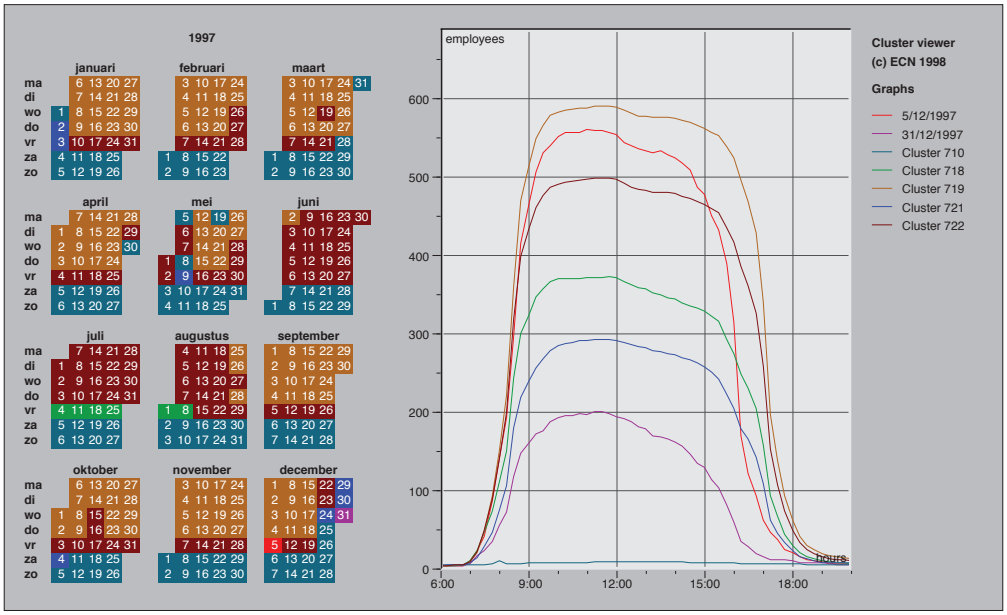


Figure 6.3: A calendar view of clusters of daily time series data on the number of employees present at ECN[104]

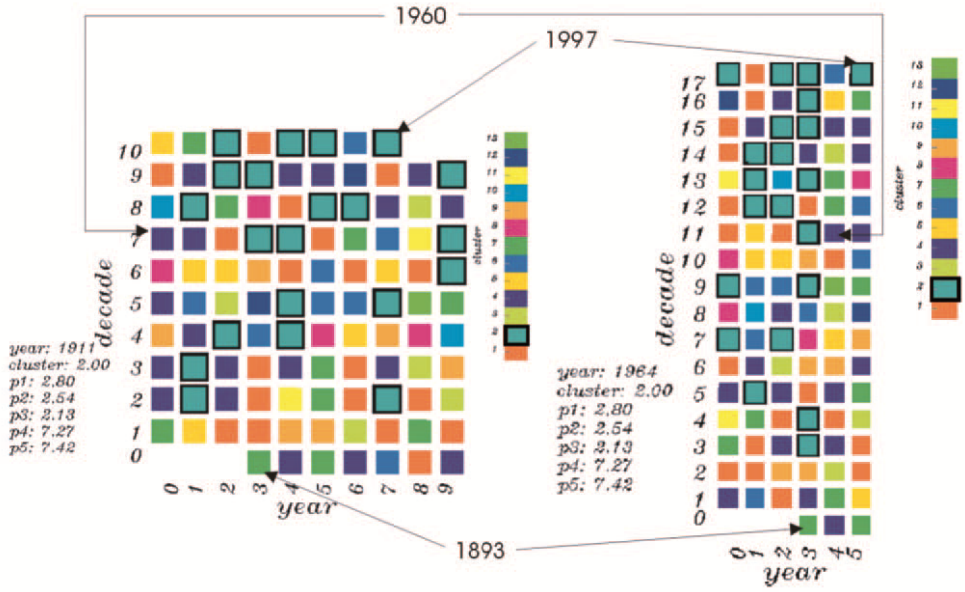


Figure 6.4: Rectangular view[8] of a temporal clustering of meteorological time-oriented data from the Potsdam observation station. Changing the periodicity (denoted as decade) from 10 years (left) to six years (right) makes the temporal pattern of cluster 2 obvious[82].

Temporal maps can be generated by using two time tags X, Y as coordinate labels, where $X, Y \in \{Year, Month, Day, Hour, Minute, Second, and\ so\ on\}$, and instances of label X should be subdivisions of each instance of label Y in the temporal domain, which is denoted mathematically as $X \sqsubseteq Y$. For example, we could form a temporal map with time tags X: Day and Y: Year, because Day $1 - 365$ is a subdivision of any Year. However, we could not form a temporal map with time tags X: Weekday(1-7) and Y: Day, because a weekday is not subsection of any day in the temporal domain and they are equal in length. Time tags X, Y used as coordinate labels of the temporal map can also be self-designed in order to visualize specific problems as shown in Figure 6.4 .

The reasons for using the temporal map in this thesis are not only because it offers very good visualization of the large scale of time series data, but also because we want to effectively memorize the temporal data. The effective memorization of the temporal data is important because it could assist the next step of the learning

workflow, prediction, by building temporal context as we mentioned already. The temporal map can be used to model the absolute temporal context by holding the temporal data in the selected coordinates. The two coordinate labels of the temporal map can be selected based on the interest of the user and the general intuition about the hidden temporal pattern. The general intuition about the hidden temporal pattern comes from the basic knowledge about the problem. For example, people go to work normally during the day across the year; people have meals at noon every day; the tree becomes green in the spring every year; and so on. This general intuition about the hidden temporal pattern can be embedded into the model before we build more delicate models to find subtle patterns.

Take the Edinburgh data for an example. We have knowledge that people on campus are normally active during the day. Therefore, we are interested to see how the scene changes during the day and over the days. The coordinate label X represents every 10 minutes from 5:30AM to 5:00PM during the day and the coordinate label Y represents the day. A temporal map of the occurrence number of activity 1 is shown in Figure 6.5. By comparing with Figure 6.2, there is much more information which can be visualized and we could have better overviews of the whole temporal pattern. We could see there are more people showing in the scene in the morning than the afternoon. Especially around 8:00AM and around 11:20AM there are high occurrence numbers of activity 1.

(Note: the data during some periods are not available. The large smooth dark blue areas on the temporal map indicates with high probability that there are no data collected during those time because it is less likely that there are no people showing in the scene for so long time. Detailed information about the Edinburgh dataset can be found from website

http://homepages.inf.ed.ac.uk/rbf/FORUMTRACKING/).

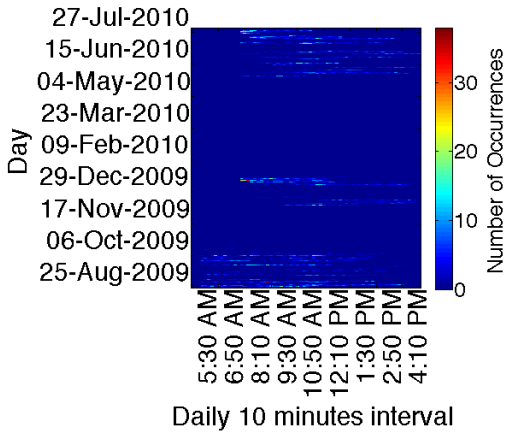


Figure 6.5: The temporal map of the occurrence number of activity 1 (from semantic area 2 to semantic area 1)

In the previous chapter, we have successfully recognized the event for each time interval $\triangle t = 10$ minutes of the scene with the Edinburgh dataset. By using categorized levels ("low", "high") as features, there are 16 types of events found through all the data, as shown in Figure 5.25. The event characterizes the situation of the scene during the time interval 10 minutes. In order to memorize the temporal occurrences of the event, the temporal map can be used to absolutely localize the event. The same coordinates are used as in Figure 6.5, but the pixel value on the temporal map as shown in Figure 6.6 indicates the event index. To better visualize the temporal distribution of an event, event 4 is shown individually in a temporal map as shown in Figure 6.7.

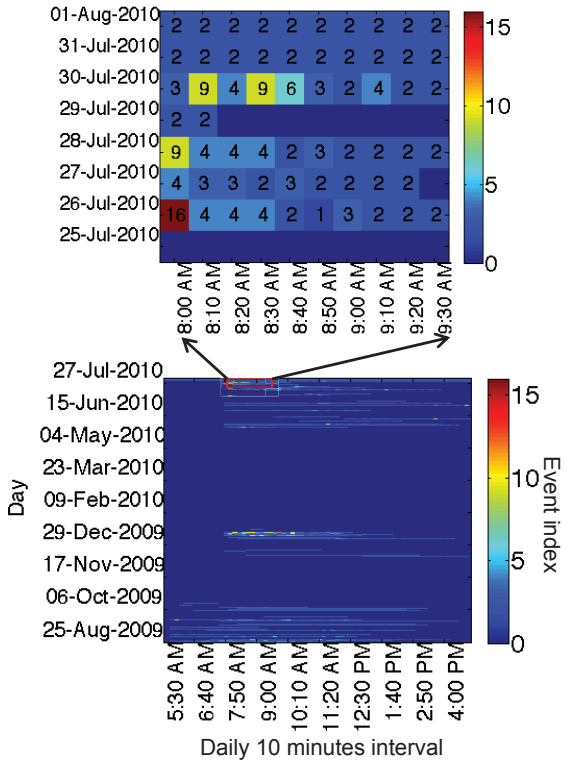


Figure 6.6: The temporal map of all 16 events with part zoomed-in

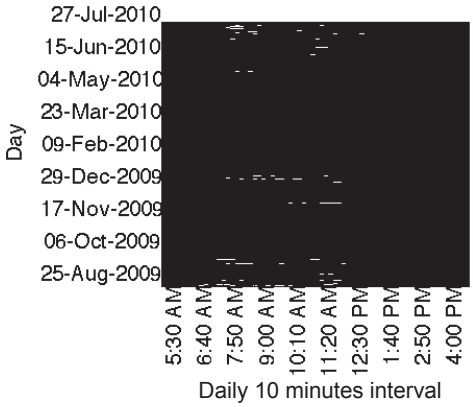


Figure 6.7: The temporal map of event 4

In order to incorporate more prior knowledge of hidden rules, the temporal map can be extended to the temporal cube or temporal hyper-dimensional space with

coordinates $(L_1, L_2, ..., L_n)$ in dimensions n, with $L_i \sqsubseteq L_{i+1}$. Figure 6.3 is an example of the temporal cube by showing the third dimension with multiple maps. The extension of temporal maps to more than 2 dimensions will not be pursued in this thesis and it is considered as possible future work.

6.1.4 Probability (transition) matrix

A first-order Markov chain (often abbreviated as Markov chain) is a stochastic process with the Markov property. The Markov property is that the next state depends only on the current state and not on the sequence of events that preceded it. That is: suppose there is a sequence of random variables $X_1, X_2, ..., X_i, ..., X_m$, and $Pr(X_{n+1} = x|X_1 = x_1,\ X_2 = x_2,\ ...,\ X_n = x_n) = Pr(X_{n+1} = x|X_n = x_n)$, where $Pr(X_{n+1} = a|X_n = b)$ indicates the probability of the state transferring from b to a. All possible values of X_i are presented as $\{s_1, s_2, s_3, ..., s_i, ..., s_k\}$, which forms a state space set S, where s_i is a possible state in the chain and k is a finite number of all possible states. A probability (or transition) matrix P can be used to represent the transition probability between different states

$$P = \begin{bmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,j} & \cdots & p_{1,k} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j} & \cdots & p_{2,k} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,j} & \cdots & p_{i,k} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ p_{k,1} & p_{k,2} & \cdots & p_{k,j} & \cdots & p_{k,k} \end{bmatrix}, \tag{6.1}$$

where $p_{i,j}$ is the probability of state s_i transferred to state s_j. Additionally, since the probability of transitioning from state s_i to some state must be 1, so $\sum_{j=1}^{k} p_{i,j} = 1$ for transitioning from s_i to s_j in one step.

In this thesis, the (first-order) Markov chain model is used to model the relative locations between events, so that an event can be localized relatively in the temporal space by referring to the previous event, which offers another way to memorize the event. The probability matrix can be learned from historical streaming data by

using maximum likelihood. An occurrence matrix O is created in order to calculate the probability matrix, as

$$O = \begin{bmatrix} o_{1,1} & o_{1,2} & \dots & o_{1,j} & \dots & o_{1,k} \\ o_{2,1} & o_{2,2} & \dots & o_{2,j} & \dots & o_{2,k} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ o_{i,1} & o_{i,2} & \cdots & o_{i,j} & \cdots & o_{i,k} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ o_{k,1} & o_{k,2} & \cdots & o_{k,j} & \cdots & o_{k,k} \end{bmatrix}, \tag{6.2}$$

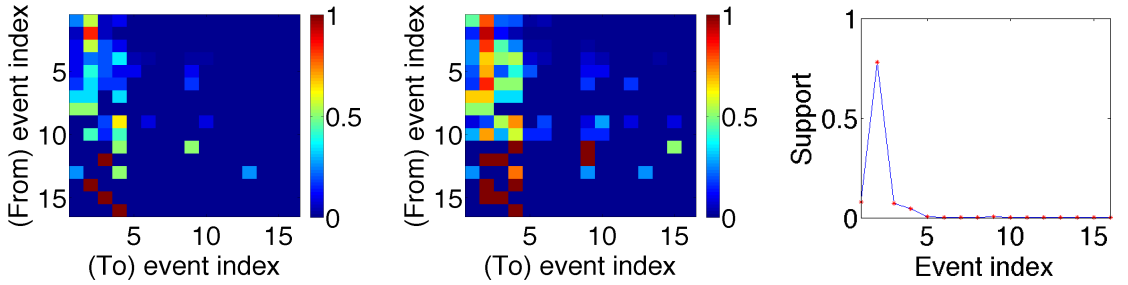
where $o_{i,j}$ indicates the occurrence number of state i transferred to state j. Each element of the probability matrix can be calculated by

$$p_{i,j} = \frac{o_{i,j}}{\sum_{j=1}^{k} o_{i,j}}. \tag{6.3}$$

With each current state s_i, if the next observed state is s_j , the element $o_{i,j}$ of the occurrence matrix O is updated by adding 1 and the corresponding row of the probability matrix is also updated by using equation 6.3.

By going through all the Edinburgh data, the learned probability matrix with one step (i.e., the probability of going from i to j in 1 time step) is shown in Figure 6.8a and the learned probability matrix with three steps is shown in Figure 6.8b. To better understand the meaning of the probability matrix, the support of each event is given in Figure 6.8c, where support is defined by the percentage of occurrence number of an event among all events.

Note: the probability matrix in k steps tells the probability of state transitioning from s_i to s_j in k steps.



(a) Probability matrix in one step (b) Probability matrix in 3 steps (c) Support (the percentage of occurrences)

Figure 6.8: Probability Matrix and support of events learned from Edinburgh dataset

6.2 Prediction

After recognizing and memorizing the event, we can now use the historical memorized data to predict the likely future events.

6.2.1 predictive model

The predictive model is based on how the events are memorized. The events are memorized by the method called double localization (absolute localization by temporal maps and relative localization by probability matrix).

It is well known that the probability matrix from the Markov chain model can be used for prediction, as shown in Figure 6.9. At time t, the corresponding event type is E_i. With the information of current state E_i, the possible states at the next time $t+1$ can be obtained from the probability matrix, supposing they are E_j,E_k with probability p_j and p_k respectively. However, the Markov chain model is considered as time-homogeneous. This means that the prediction result with Markov chain model is independent of time t, which is not true in some cases. Take the temporal distribution of event 4 for an example, as shown Figure 6.7. It is obvious that event 4 occurs much more often in the morning than in the afternoon. If only the Markov chain probability matrix is used in prediction, the result will be biased.

In order to take consideration of the time t in the predictive model, the absolute localization part (temporal maps) of the memorization is used. The temporal map is used to calculate the probability of each event occurring at each timestamp on the coordinate X of the map by maximum likelihood

$$p_{s_i,t} = \frac{N_{s_i,t}}{\sum_{j=1}^{k} N_{s_j,t}}, \tag{6.4}$$

where $N_{s_i,t}$ indicates the number of state (or event) s_i that occurred at timestamp t from historical data on the temporal map, and k is the number of all states (events) observed at timestamp t. In other words, the temporal map can tell what kind of events could possibly occur at a specific time interval from historical data. In the example of the Edinburgh data, the temporal map can tell what kind of events could possibly occur every 10 minutes during the day from historical data. On the other hand, the probability matrix of the Markov chain model tells the relative locations between events, which could be used to predict the event from another perspective.

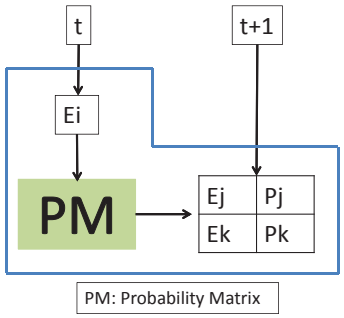


Figure 6.9: Prediction with probability matrix

Therefore, the predication model is built by incorporating the probability matrix and the temporal map as shown in Figure 6.10. The probability obtained from the temporal map is used to weight the probability obtained from the probability matrix. Suppose the probability list obtained from the Markov chain model with current event E_i for each event is $\{p_1, p_2, ..., p_i, ...p_k\}$, and the probability list obtained from the temporal map with the timestamp $t+1$ is $\{p'_1, p'_2, ..., p'_i, ...p'_k\}$. The

final prediction result at timestamp $t+1$ can be represented as the following equation

$$p''_{i,t+1} = \frac{p_i \times p'_i}{\sum_{i=1}^{k} p_i \times p'_i}, \tag{6.5}$$

where k is the number of all possible events.

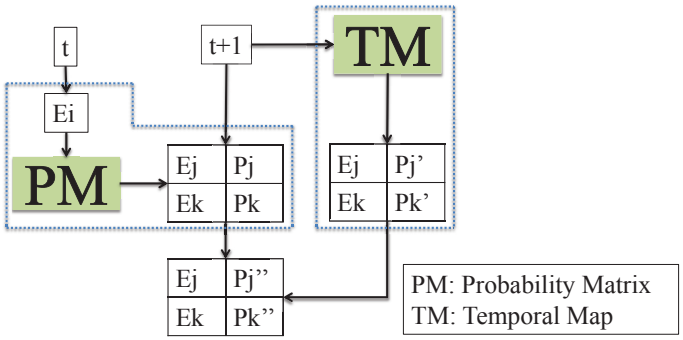


Figure 6.10: The predictive model with the probability matrix and the temporal map incorporated

6.2.2 Streaming prediction based on Edinburgh data

Until now, the three key steps of the whole framework of streaming analysis of the time series data has been explained, from recognition, memorization to prediction. The system can continuously recognize an event by the GMD incremental clustering method, and memorize the recognition result at each timestamp by double localization. The memory stored in the library can then be used to predict the future event likelihood and detect anomalies. The Edinburgh pedestrian data is used to illustrate the whole process as shown in Figure 6.11. At each time interval t , the trajectory is recognized into the activity, and then the activity observed in the scene defines (are recognized into) the event type. The event type associated with the time interval t is memorized into the library by double localization, which builds a temporal context. At the next time interval $t + 1$, the memory in the library is used for prediction and detecting anomalies in different levels of information from trajectories, activities to events.

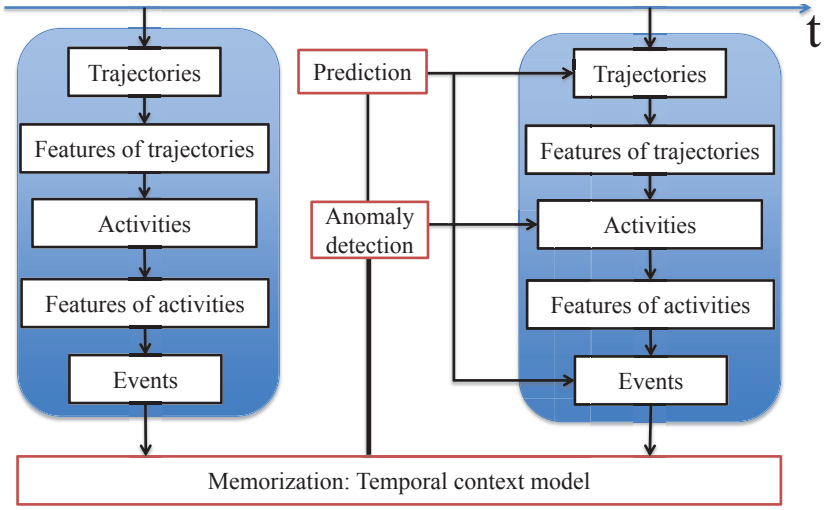


Figure 6.11: Stream prediction based on Edinburgh data

The event level of information is used as an example to show the prediction result. Two successive frames of the whole process (on 2009-9-23 8:10AM and 8:20AM respectively) are shown in Figure 6.12 and Figure 6.13 as an example. The left sub figure of Figure 6.12 and Figure 6.13 is the temporal map, the upper right sub figure is the probability matrix and the lower right sub figure is the final result. On the lower right sub figure, there are three pieces of information displayed. The second is "Current" which shows the current measurement of the event index. The third one is "Prediction" which shows the prediction result based on current event measure and timestamp. The first one is "Last prediction" which shows the prediction result from the previous timestamp. In another word, "Last prediction" shows the prediction information of what could happen at the current timestamp, while we already have the current measurement of the event. Therefore, the current measurement is compared with the previous prediction to check whether the current event is as predicted or it is an anomaly. In the result, if the current event is as predicted, it is shown in green; otherwise it is shown in red as shown in Figure 6.12 and Figure 6.13.

In order to visualize a longer term of the prediction result instead of frame by frame, another visualization of the prediction result is shown in Figure 6.14-6.16. Figure 6.14 shows a short period of the prediction result so that the structure of the

visualization can be clearly seen. At each timestamp, there are several predictions shown in gray, and the end of the prediction line tells the prediction result including the event type and the probability of the event which could occur. The probability of the event is represented by the various size of the gray spot in the figure. The size of the gray spot is proportional to $p^{0.1}$, where p is the probability, 0.1 is chosen so that the really small probability of an event can also be visualized instead of overwhelmed by the thick line. The actual measurement of the event is shown in green and red lines, where the green line indicates a normal event and the red line indicates abnormal. Figure 6.15 shows a longer period of the prediction result when most of the time the event occurs as predicted, which indicates the period is generally "normal". Figure 6.16 shows a longer period of the prediction result when there are a lot of unexpected events occurred, which indicates something new occurred in the scene.

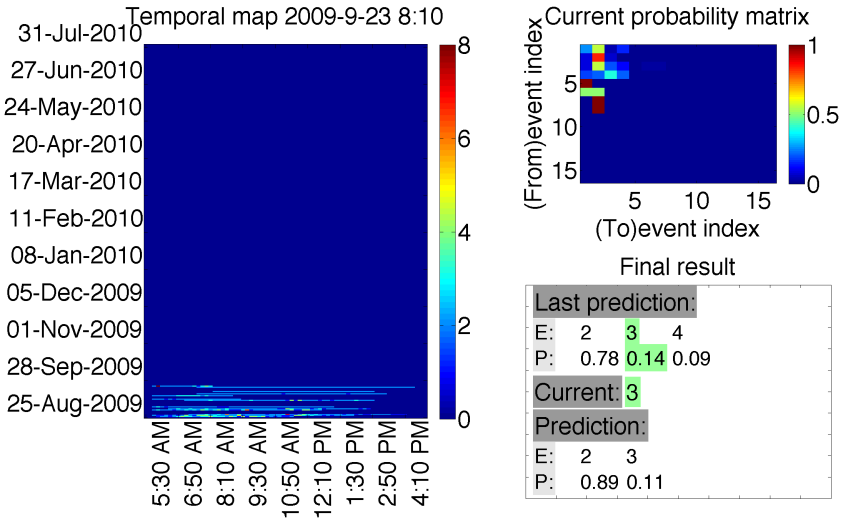


Figure 6.12: The steaming analysis at 2009-9-23 8:10AM

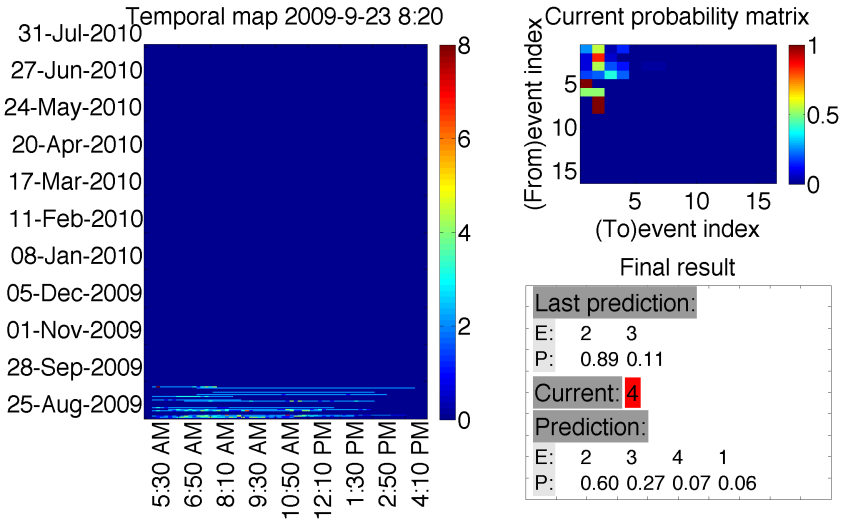


Figure 6.13: The steaming analysis at 2009-9-23 8:20AM

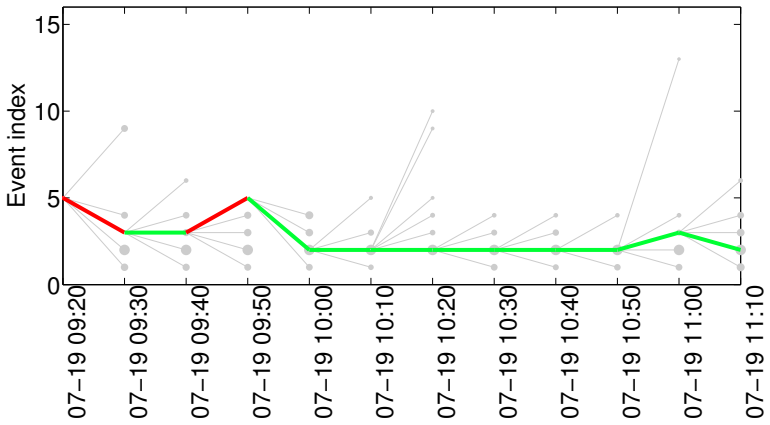


Figure 6.14: The prediction result during a short period

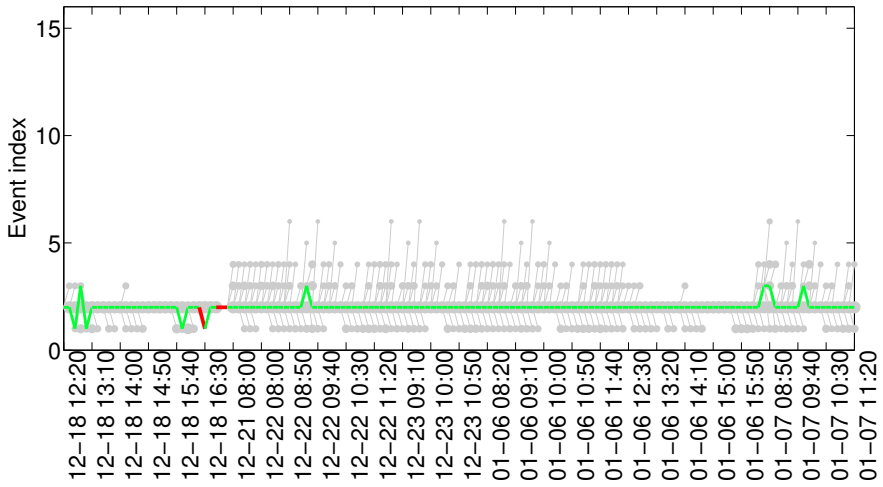


Figure 6.15: The prediction result during a long period when there are few anomalies

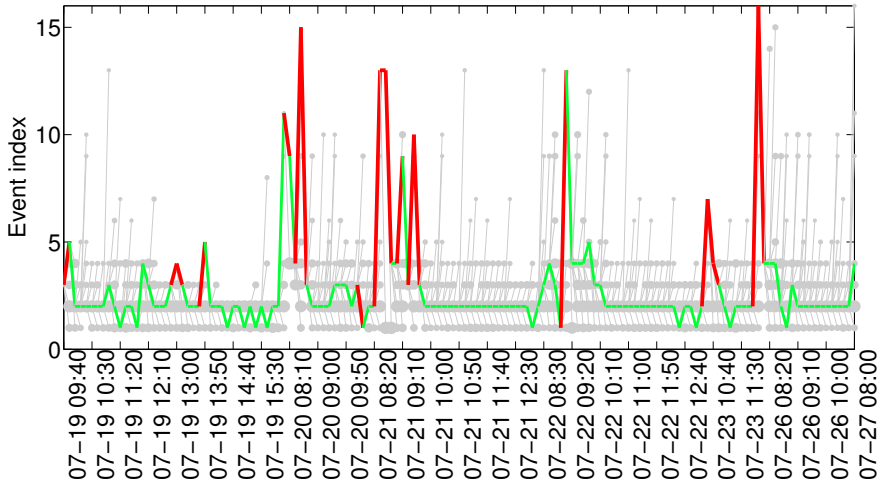


Figure 6.16: The prediction result during a long period when there are more anomalies

Chapter 7

Conclusion

This thesis contains two main parts. The first part is temporal signature modeling (i.e. process models) in DIRSIG, described in Chapter 2 and 3. The second part is temporal signature analysis, described in Chapter 4, 5, and 6.

In the first part of the thesis, we developed a framework of incorporating temporal signatures in DIRSIG to enhance the capacity of the current version of DIRSIG, so that we could generate spatial-spectral-temporal synthetic images. The process models are used to link temporal signatures of the scene object.

Due to this specific input design of DIRSIG, the sub process models could be categorized into two types. One is that the process model drives the property of each facet of the object changing over time, and the other one is that the process model drives the geometry location of the object in the scene changing as a function of time. Two example process models are used to show how the two types of process models will be incorporated into DIRSIG. One example process model is a notional two-tanks hydrodynamic & thermodynamic model, controlled by the state of valves in the scenario. The other example process model is a parking lot model called PARKVIEW which is based on the statistical description of a parking lot to generate the status map of the parking lot over time. An experiment is also done to show how the statistical description of the parking lot can be obtained through an image-based method.

The user can generate spatial-spectral-temporal synthetic images with DIRSIG easier than before by using our proposed framework of incorporating temporal signatures in DIRSIG.

In the second part of the thesis, a framework is developed for streaming analysis of time series data: from recognition, to memorization, to prediction. This framework can learn events and their temporal contexts from a streaming time series without supervision. Those learned events and temporal contexts can then be used to predict the future and detect anomalous activities. An incremental clustering method is proposed to recognize events, while the memorization method of double localization is proposed to learn the temporal contexts, including the absolute temporal context and relative temporal context. A predictive model is built based on double localization from the memorization step.

By using the proposed framework of temporal signature analysis, take the Edinburgh dataset for an example, we could not only get current high level information of the scene, such as activities performed in the scene and events describing the situation of the scene, but also we could reason whether current activities or events are normal by referring to the temporal context learned from historical data. In the end, 92,000+ observed trajectories and their temporal patterns over one year can be summarized by a temporal map of 16 events and a probability (transition) matrix between them. Our proposed framework offers an effective method of extracting useful and manageable information from a huge amount of raw data of trajectories.

We expect our proposed framework of temporal signature analysis has wide applications in different fields, such as the internet usage, policy planning, security, dynamics understanding and so on.

Chapter 8

Future work

Since the real data is not always ideally obeying certain temporal rules, the temporal characteristics of the real data are often contaminated by noises to some extent. One concern with the predictive model in the framework of streaming analysis of time series data is that with enough long data, each timestamp may show up with all the possible events, which results in all activities eventually being deemed "normal". However, this is not right from the interest of the user, because if a predictive system predicts everything in the future is possible, actually the prediction system tells us nothing. To resolve this concern, three possible aspects worthy to be pursued in the future are described here.

The first aspect of the future work is to model "forgetting" and selective memorization in the system. 'Forgetting' something which is too old can make the prediction result less "noisy" based on the memory. Selective memorization is to remember something which is especially interesting even though it is already very old. The score of interestingness of something can come from the preference of the user. We could define something which rarely occurs as interesting, or something occurring often as interesting. By modeling "forgetting" and selective memorization, the system is more "focused" on something.

The second aspect of the future work is to improve the accuracy of absolute localization of an event. By incorporating more prior knowledge of hidden rules,

as mentioned earlier, the temporal map could be extended into hyper-dimensional temporal space. By doing this, more specific absolute patterns could be explored, which will limit the range of the prediction result.

The third aspect of the future work is to embed a better method for relative localization. In this thesis, the simple first-order Markov chain is used to capture the relative temporal positions among events. However, there are potentially more complicated temporal hidden structures among the events in the temporal space. Much work has been done in this field to capture the temporal structure of events, such as variable-order Markov models, Conditional Random Fields (CRFs), Context Free Grammars (CFGs), and so on. These methods have deeper memory of the relative positions among events than a first-order Markov chain. By adopting those methods, better and more "focused" prediction results should be expected.

Additionally, human memory has been studied for very long time [93][51]. Temporal context models are built to explain the process of human memory. An interesting property of human memory is that every free recall of an item can reconsolidate the memory of that target item and items occurred close in time to the target item. This property can be used in artificial intelligent systems to simulate the memory of interesting items. The interesting items are the ones which are frequently queried by the user. Every query will trigger an "recall" in the system. Therefore, by using the property above, the artificial intelligent systems have better memories of interesting items. However, human memory can be easily disrupted by representation of similar items, weakened or even erased due to some reasons explained in [106][80]. For the future work, the temporal context models of a "perfect" human memory should be studied and embedded in the artificial intelligent systems.

Bibliography

[1] Midland cogeneration venture(http://www.eqt.se/en/portfolio/companies/mcv/).

[2] Sentinels(http://www.esa.int/esaeo.html).

[3] 4d bim or simulation-based modeling, April 2011.

[4] Modules (http://www.thermoanalytics.com/cae-software/modules), 2012.

[5] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment, 2003.

[6] Rakesh Agrawal, Johannes Gehrke, Dimitrios Gunopulos, and Prabhakar Raghavan. *Automatic subspace clustering of high dimensional data for data mining applications*, volume 27. ACM, 1998.

[7] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Int. Conf. Manage. Data*, volume 22, pages 207–216. ACM, May 1993.

[8] Wolfgang Aigner, Silvia Miksch, Wolfgang Muller, Heidrun Schumann, and Christian Tominski. Visual methods for analyzing time-oriented data. *Visualization and Computer Graphics, IEEE Transactions on*, 14(1):47–60, 2008.

[9] Ajaya K Akasapu, P Srinivasa Rao, LK Sharma, and SK Satpathy. Density based k-nearest neighbors clustering algorithm for trajectory data. *International Journal of Advanced Science and Technology*, 31(1), 2011.

[10] Juan M Ale and Gustavo H Rossi. An approach to discovering temporal association rules. In *Proceedings of the 2000 ACM symposium on Applied computing-Volume 1*, pages 294–300, New York, New York, USA, 2000. ACM, ACM Press.

[11] James F Allen. Maintaining knowledge about temporal intervals. *Comm. ACM*, 26(11):832–843, Nov. 1983.

[12] Gennady Andrienko, Natalia Andrienko, and Stefan Wrobel. Visual analytics tools for analysis of movement data. *ACM SIGKDD Explorations Newsletter*, 9(2):38–46, Dec. 2007.

[13] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM, 1999.

[14] Cláudia M Antunes and Arlindo L Oliveira. Temporal data mining: An overview. In *Proc. Workshop Temporal Data Mining*, pages 1–13, 2001.

[15] Stefan Atev, Grant Miller, and Nikolaos P Papanikolopoulos. Clustering of vehicle trajectories. *Intelligent Transportation Systems, IEEE Transactions on*, 11(3):647–657, 2010.

[16] A.H. Baqui, R.B. Sack, R.E. Black, K. Haider, A. Hossain, A.R.M.A. Alim, M. Yunus, HR Chowdhury, and AK Siddique. Enteropathogens associated with acute and persistent diarrhea in bangladeshi children$<$ 5 years of age. *Journal of Infectious Diseases*, 166(4):792–796, 1992.

[17] Bill Basener, Emmett J Ientilucci, and David W Messinger. Anomaly detection using topology. In *Defense and Security Symposium*, pages 65650J–65650J. International Society for Optics and Photonics, 2007.

[18] William F Basener and David W Messinger. Enhanced detection and visualization of anomalies in spectral imagery. In *SPIE Defense, Security, and Sensing*, pages 73341Q–73341Q. International Society for Optics and Photonics, 2009.

[19] Iyad Batal, Dmitriy Fradkin, James Harrison, Fabian Moerchen, and Milos Hauskrecht. Mining recent temporal patterns for event detection in multivariate time series data. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 280–288, New York, New York, USA, 2012. ACM, ACM Press.

[20] I. Benenson, K. Martens, and S. Birfir. Parkagent: An agent-based model of parking in the city. *Computers, Environment and Urban Systems*, 32(6):431–439, 2008.

[21] Pavel Berkhin. A survey of clustering data mining techniques. In *Grouping multidimensional data*, pages 25–71. Springer, 2006.

[22] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive and Mobile Computing*, 6(2):161–180, Apr. 2010.

[23] Claudio Bettini, Sushil Jajodia, and Sean Wang. *Time granularities in databases, data mining, and temporal reasoning.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2000.

[24] G.E.P. Box, G.M. Jenkins, and G.C. Reinsel. *Time series analysis: forecasting and control*, volume 734. Wiley, 2011.

[25] Feng Cao, Martin Ester, Weining Qian, and Aoying Zhou. Density-based clustering over an evolving data stream with noise. In *SDM*, volume 6, pages 326–337. SIAM, 2006.

[26] L. Carleson. On convergence and growth of partial sums of fourier series. *Acta Mathematica*, 116(1):135–157, 1966.

[27] Moses Charikar, Chandra Chekuri, Tomás Feder, and Rajeev Motwani. Incremental clustering and dynamic information retrieval. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 626–635. ACM, 1997.

[28] LC Chen. Detection of shoreline changes for tideland areas using multi-temporal satellite images. *International Journal of Remote Sensing*, 19(17):3383–3397, Nov. 1998.

[29] Yixin Chen and Li Tu. Density-based clustering for real-time stream data. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, New York, New York, USA, 2007. ACM, ACM Press.

[30] KS Cheng, C Wei, and SC Chang. Locating landslides using multi-temporal satellite images. *Adv. Space Res.*, 33(3):296–301, 2004.

[31] DCS Corporation. Airsim thermal signature prediction and analysis tool model assumptions and analytical foundations. Technical report, 1991.

[32] A.R. Curran and T.G. Gonda. Applications of the muses infrared signature code. Technical report, DTIC Document, 2005.

[33] DIRS. Dirsig4 xml inputs(http://www.dirsig.org/dwiki/index.php). Technical report, Rochester Institute of Technology, 2008.

[34] J. Dorsey, A. Edelman, H.W. Jensen, J. Legakis, and H.K. Pedersen. Modeling and rendering of weathered stone. In *ACM SIGGRAPH 2006 Courses*, page 4. ACM, 2006.

[35] Thi V Duong, Hung Hai Bui, Dinh Q Phung, and Svetha Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 838–845. IEEE, 2005.

[36] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[37] Vladimir Estivill-Castro and Alan T Murray. *Spatial clustering for data mining with genetic algorithms*. Queensland University of Technology Australia, 1997.

[38] Douglas H Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine learning*, 2(2):139–172, 1987.

[39] Zhouyu Fu, Weiming Hu, and Tieniu Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–602. IEEE, 2005.

[40] Juan-M García-Huerta, Hugo Jiménez-Hernández, Ana-M Herrera-Navarro, Teresa Hernández-Díaz, and Ivan Terol-Villalobos. Modelling dynamics with context-free grammars. In *IS&T/SPIE Electronic Imaging*, pages 902611–902611. International Society for Optics and Photonics, 2014.

[41] AJ Garrett and AL Boni. Alge: A 3-d thernial plume prediction code for lakes, rivers and estuaries (u). 1997.

[42] A.J. Garrett, J.M. Irvine, A.D. King, T.K. Evers, D.A. Levine, C. Ford, and J.L. Smyre. Application of multispectral imagery to assessment of a hydrodynamic simulation of an effluent stream entering the clinch river. *Photogrammetric engineering and remote sensing*, 66(3):329–335, 2000.

[43] MG Gartley, JR Schott, and SD Brown. Micro-scale modeling of contaminant effects on surface optical properties. In *Proceedings of SPIE, the International Society for Optical Engineering*, pages 70860H–1. Society of Photo-Optical Instrumentation Engineers, 2008.

[44] J. Gu, C.I. Tu, R. Ramamoorthi, P. Belhumeur, W. Matusik, and S. Nayar. Time-varying surface appearance: acquisition, modeling and rendering. In

ACM Transactions on Graphics (TOG), volume 25, pages 762–771. ACM, 2006.

[45] Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *Knowledge and Data Engineering, IEEE Transactions on*, 15(3):515–528, 2003.

[46] Mitasova H. and Mitas L. Process modeling and simulations (http://www.ncgia.ucsb.edu/giscc/units/u130/u130.html), 12 1998.

[47] Lawrence O Hall, Ibrahim Burak Ozyurt, and James C Bezdek. Clustering with a genetically optimized approach. *Evolutionary Computation, IEEE Transactions on*, 3(2):103–112, 1999.

[48] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques*. Morgan kaufmann, 2006.

[49] Steven A Hofmeyr, Stephanie Forrest, and Anil Somayaji. Intrusion detection using sequences of system calls. *Journal of computer security*, 6(3):151–180, 1998.

[50] Somboon Hongeng and Ramakant Nevatia. Large-scale event detection using semi-hidden markov models. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1455–1462. IEEE, IEEE, 2003.

[51] Marc W Howard and Michael J Kahana. A distributed representation of temporal context. *Journal of Mathematical Psychology*, 46(3):269–299, 2002.

[52] S. Hsu and T. Wong. Simulating dust accumulation. *Computer Graphics and Applications, IEEE*, 15(1):18–22, 1995.

[53] E.J. Ientilucci and S.D. Brown. Advances in wide area hyperspectral image simulation. In *Proc. of SPIE Vol*, volume 5075, page 111, 2003.

[54] H.W. Jensen. *Realistic image synthesis using photon mapping*. AK Peters, Ltd., 2001.

[55] Anita Jones and Song Li. Temporal signatures for intrusion detection. In *Computer Security Applications Conference, 2001. ACSAC 2001. Proceedings 17th Annual*, pages 252–261. IEEE, 2001.

[56] Madjid Khalilian and Norwati Mustapha. Data stream clustering: Challenges and issues. *arXiv preprint arXiv:1006.5261*, 2010.

[57] Kihwan Kim, Dongryeol Lee, and Irfan Essa. Gaussian process regression flow for analysis of motion trajectories. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 1164–1171. IEEE, 2011.

[58] Julian FP Kooij, Gwenn Englebienne, and Dariu M Gavrila. A non-parametric hierarchical model to discover behavior dynamics from tracks. In *Computer Vision–ECCV 2012*, pages 270–283. Springer, 2012.

[59] Eric J. Kostelich and Thomas Schreiber. Noise reduction in chaotic time-series data: A survey of common methods. *Phys. Rev. E*, 48:1752–1763, Sep 1993.

[60] D. Krajzewicz, G. Hertkorn, C. Rössel, and P. Wagner. Sumo (simulation of urban mobility). In *Proc. of the 4th Middle East Symposium on Simulation and Modelling*, pages 183–187, 2002.

[61] K Krishna and M Narasimha Murty. Genetic k-means algorithm. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 29(3):433–439, 1999.

[62] FA Kruse, AB Lefkoff, JW Boardman, KB Heidebrecht, AT Shapiro, PJ Barloon, and AFH Goetz. The spectral image processing system (sips)-interactive visualization and analysis of imaging spectrometer data. *Remote sensing of environment*, 44(2):145–163, 1993.

[63] Daniel Kuettel, Michael D Breitenstein, Luc Van Gool, and Vittorio Ferrari. What's going on? discovering spatio-temporal dependencies in dynamic scenes. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 1951–1958. IEEE, IEEE, Jun. 2010.

[64] Srivatsan Laxman and P Shanti Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, Apr 2006.

[65] Yingjiu Li, Peng Ning, X Sean Wang, and Sushil Jajodia. Discovering calendar-based temporal association rules. *Data Knowl. Eng.*, 44(2):193–218, 2003.

[66] Percy Liang, Michael I Jordan, and Dan Klein. Probabilistic grammars and hierarchical dirichlet processes. *The handbook of applied Bayesian analysis*, 2009.

[67] Hsien-Chou Liao and Chien-Chih Tu. A rdf and owl-based temporal context reasoning model for smart home. *Information Technology Journal*, 6(8):1130–1138, Aug. 2007.

[68] Qing-Bao Liu, Su Deng, Chang-Hui Lu, Bo Wang, and Yong-Feng Zhou. Relative density based k-nearest neighbors clustering algorithm. In *Machine Learning and Cybernetics, 2003 International Conference on*, volume 1, pages 133–17. IEEE, 2003.

[69] Sebastian Lühr and Mihai Lazarescu. Incremental clustering of dynamic data streams using connectivity based representative points. *Data & Knowledge Engineering*, 68(1):1–27, Jan 2009.

[70] G.S. Maddala. *Limited-dependent and qualitative variables in econometrics*, volume 3. Cambridge Univ Pr, 1983.

[71] Barbara Majecka. Statistical models of pedestrian behaviour in the forum. *Master's thesis, School of Informatics, University of Edinburgh*, 2009.

[72] Adam Marcus, Michael S Bernstein, Osama Badar, David R Karger, Samuel Madden, and Robert C Miller. Twitinfo: aggregating and visualizing microblogs for event exploration. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 227–236. ACM, 2011.

[73] D McKeown, J Cocburn, J Faulring, K Kremens, D Morse, H Rhody, and M Richardson. Wildfire airborne sensor program (wasp): A new wildland fire detection and mapping system. In *Proceedings of the Tenth Forest Service Remote Sensing Applications Conference*, pages 5–9, 2004.

[74] S. Merillou, J.M. Dischler, and D. Ghazanfarpour. Corrosion: simulating and rendering. In *Graphics Interface*, pages 167–174, 2001.

[75] T. Mitsa. *Temporal data mining*, volume 12. Chapman & Hall/CRC, 2009.

[76] Dan Moldovan, Christine Clark, and Sanda Harabagiu. Temporal context representation and reasoning. In *International Joint Conference on Artificial Intelligence*, volume 19, page 1099. Citeseer, 2005.

[77] Anna Monreale, Fabio Pinelli, Roberto Trasarti, and Fosca Giannotti. Wherenext: a location predictor on trajectory pattern mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 637–646. ACM Press, 2009.

[78] Darnell Moore and Irfan Essa. Recognizing multitasked activities from video using stochastic context-free grammar. In *AAAI/IAAI*, pages 770–776, 2002.

[79] Robert Moskovitch and Yuval Shahar. Medical temporal-knowledge discovery via temporal abstraction. In *AMIA Annual Symposium Proceedings*, volume 2009, page 452. American Medical Informatics Association, 2009.

[80] Karim Nader, Glenn E Schafe, and Joseph E Le Doux. Fear memories require protein synthesis in the amygdala for reconsolidation after retrieval. *Nature*, 406(6797):722–726, 2000.

[81] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the international conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9. ACM, 2001.

[82] Thomas Nocke, Heidrun Schumann, Uwe Bohm, and Michael Flechsig. Information visualization supporting modelling and evaluation tasks for climate models. In *Simulation Conference, 2003. Proceedings of the 2003 Winter*, volume 1, pages 763–771. IEEE, 2003.

[83] Liadan O'callaghan, Adam Meyerson, Rajeev Motwani, Nina Mishra, and Sudipto Guha. Streaming-data algorithms for high-quality clustering. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pages 0685–0685. IEEE Computer Society, 2002.

[84] Daniela Pohl, Abdelhamid Bouchachia, and Hermann Hellwagner. Automatic sub-event detection in emergency management using social media. In *Proceedings of the 21st international conference companion on World Wide Web*, pages 683–686. ACM Press, 2012.

[85] R. R. Schott, Raqueno and C. Salvaggio. Incorporation of a time-dependent thermodynamic model and a radiation propagation model into infrared three-dimensional synthetic image generation. *Optical Engineering*, 31(7/1505), 1992.

[86] Nina Raqueno. Rit's share2012 collection data summary(http://twiki.cis.rit.edu/twiki/bin/view/main/share2012collectionresults).

[87] Irving S Reed and Xiaoli Yu. Adaptive multiple-band cfar detection of an optical pattern with unknown spectral distribution. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 38(10):1760–1770, 1990.

[88] Salvatore Rinzivillo, Dino Pedreschi, Mirco Nanni, Fosca Giannotti, Natalia Andrienko, and Gennady Andrienko. Visually driven analysis of movement data by progressive clustering. *Information Visualization*, 7(3-4):225–239, 2008.

[89] Pedro Pereira Rodrigues, João Gama, and Joao Pedro Pedroso. Hierarchical clustering of time-series data streams. *Knowledge and Data Engineering, IEEE Transactions on*, 20(5):615–627, 2008.

[90] Andrea Rosani, Nicola Conci, and Francesco GB De Natale. Human behavior recognition using a context-free grammar. *Journal of Electronic Imaging*, 23(3):033016, Jun. 2014.

[91] J.R. Schott. *Remote sensing: the image chain approach*. Oxford University Press, USA, 2007.

[92] J.R. Schott, S.D. Brown, R.V. Raqueno, HN Gross, and G. Robinson. An advanced synthetic image generation model and its application to multi/hyperspectral algorithm development. *Canadian Journal of Remote Sensing*, 25(2):99–111, 1999.

[93] Per B Sederberg, Samuel J Gershman, Sean M Polyn, and Kenneth A Norman. Human memory reconsolidation can be explained using the temporal context model. *Psychonomic bulletin & review*, 18(3):455–468, 2011.

[94] Y.W. Seo and C. Urmson. Utilizing prior information to enhance self-supervised aerial image analysis for extracting parking lot structures. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 339–344. IEEE, 2009.

[95] Cristian Sminchisescu, Atul Kanaujia, and Dimitris Metaxas. Conditional models for contextual human motion recognition. *Computer Vision and Image Understanding*, 104(2-3):210–220, Nov 2006.

[96] Chris Stauffer and W. Eric L. Grimson. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757, 2000.

[97] D.F. Stroup, S.B. Thacker, and J.L. Herndon. Application of multiple time series analysis to the estimation of pneumonia and influenza mortality by age 1962–1983. *Statistics in Medicine*, 7(10):1045–1059, 2006.

[98] B. Sun, K. Sunkavalli, R. Ramamoorthi, P.N. Belhumeur, and S.K. Nayar. Time-varying brdfs. *Visualization and Computer Graphics, IEEE Transactions on*, 13(3):595–609, 2007.

[99] J. Sun and D. Messinger. Parking lot process model incorporated into dirsig scene simulation. In *Proceedings of SPIE*, volume 8390, page 83900I, 2012.

[100] X. Tang, Y. Liu, J. Zhang, and W. Kainz. Advances in spatio-temporal analysis: An introduction. *Advances in Spatio-Temporal Analysis: in: ISPRS Book Series*, 5:1, 2007.

[101] A.J. Tatem, S.J. Goetz, and S.I. Hay. Fifty years of earth observation satellites: Views from above have lead to countless advances on the ground in both scientific knowledge and daily life. *American scientist*, 96(5):390, 2008.

[102] Paul M. Torrens. Process models and next generation geographic information technology, summer 2009.

[103] P. van der Waerden, H. Timmermans, and A. Borgers. Pamela: Parking analysis model for predicting effects in local areas. *Transportation Research Record: Journal of the Transportation Research Board*, 1781(-1):10–18, 2002.

[104] Jarke J Van Wijk and Edward R Van Selow. Cluster and calendar based visualization of time series data. In *Information Visualization, 1999.(Info Vis' 99) Proceedings. 1999 IEEE Symposium on*, pages 4–9. IEEE, IEEE Comput. Soc, 1999.

[105] Keshri Verma and Om Prakash Vyas. Efficient calendar based temporal association rule. *ACM SIGMOD Record*, 34(3):63–70, 2005.

[106] Matthew P Walker and Robert Stickgold. Overnight alchemy: sleep-dependent memory evolution. *Nature Reviews Neuroscience*, 11(3):218–218, 2010.

[107] J. Wang, X. Tong, S. Lin, M. Pan, C. Wang, H. Bao, B. Guo, and H.Y. Shum. Appearance manifolds for modeling time-variant appearance of materials. In *ACM Transactions on Graphics (TOG)*, volume 25, pages 754–761. ACM, 2006.

[108] X. Wang and A.R. Hanson. Parking lot analysis and visualization from aerial images. In *Applications of Computer Vision, 1998. WACV'98. Proceedings., Fourth IEEE Workshop on*, pages 36–41. IEEE, 1998.

[109] Xiaogang Wang, Keng Teck Ma, Gee-Wah Ng, and W Eric L Grimson. Trajectory analysis and semantic region modeling using nonparametric hierarchical bayesian models. *International journal of computer vision*, 95(3):287–312, 2011.

[110] T Warren Liao. Clustering of time series data-a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[111] G. Welch and G. Bishop. An introduction to the kalman filter, 1995.

[112] Weiguang Xu, Jianjiang Lu, Yafei Zhang, and Jiabao Wang. *An Unsupervised Framework of Video Event Analysis*, volume 114 of *Advances in Intelligent and Soft Computing*, chapter chapter 42, pages 329–337. Springer Berlin Heidelberg, 2012.

[113] W. Young and M. Taylor. A parking model hierarchy. *Transportation*, 18(1):37–58, 1991.

[114] P. Zarchan and H. Musoff. *Fundamentals of Kalman filtering: a practical approach*, volume 208. Aiaa, 2005.

[115] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.

[116] Siqi Zhao, Lin Zhong, Jehan Wickramasuriya, Venu Vasudevan, Robert LiKamWa, and Ahmad Rahmati. Sportsense: Real-time detection of nfl game events from twitter. *arXiv preprint arXiv:1205.3212*, 2012.

Appendix A

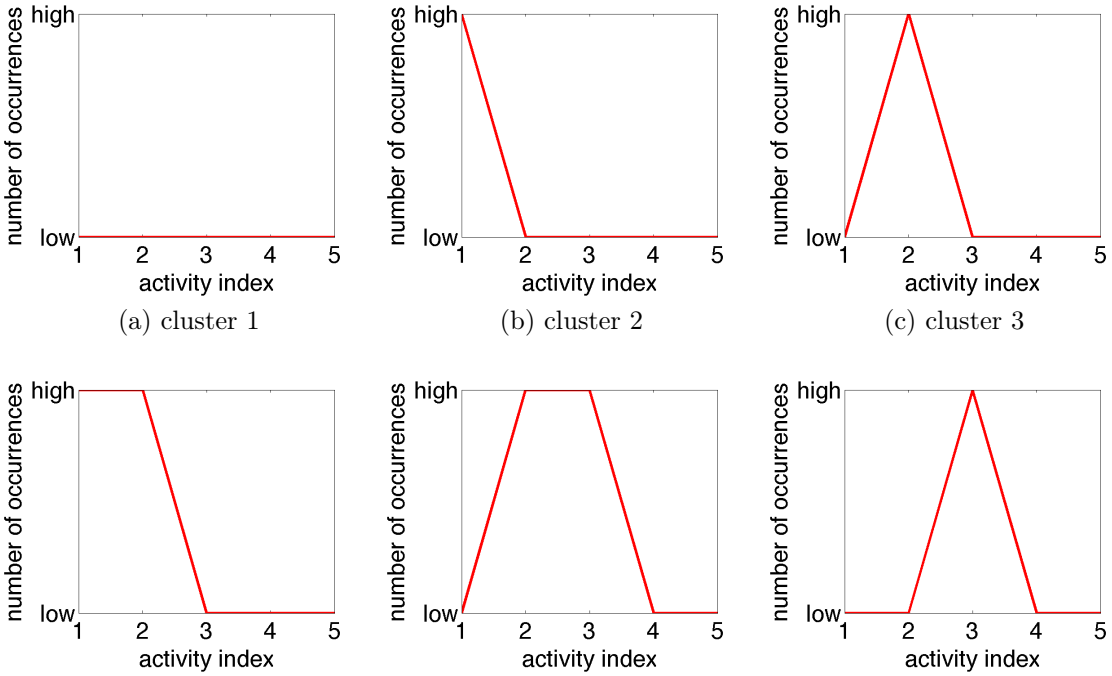
Results of streaming analysis of randomized data

In order to see how the results of real data are different from the randomized data, the real data are randomized in two levels: the activity level and the event level. The results of streaming analysis of each randomized dataset are shown as below.

A.1 Results of temporally randomized activities

The occurrence moments (time points) of all the trajectories are randomized in the original time space from August 25, 2009 to August 2, 2010 during the time period when the camera collections are available.

Still the top 5 activities are used to identify the events occurred on the scene. The quantized number (0 and 1) of occurrences of each activity is used as the features, where 1 represents there is a high number of occurrences of the activity and 0 represents there is a low number of occurrences of the activity. The threshold to declare there is a high number of occurrences is still 5. By going through the whole randomized activities with the incremental clustering method GMD, 6 events are found as shown in Figure A.1.

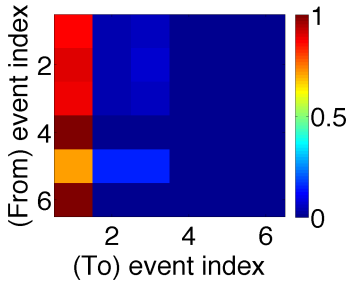


(a) cluster 1 (b) cluster 2 (c) cluster 3

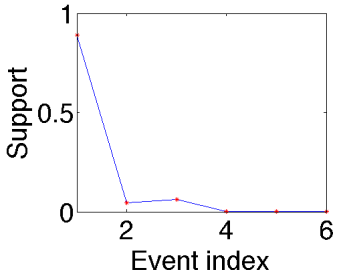
(d) cluster 4 (e) cluster 5 (f) cluster 6

Figure A.1: Event clusters with categorized features by using the activity temporally randomized data

By going through all the activity temporally randomized data, the learned probability matrix with one step is shown in Figure A.2a. The support of each event is given in Figure A.2b.



(a) Probability matrix in one step

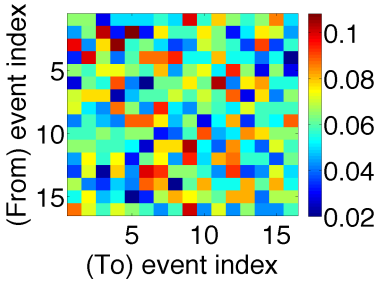


(b) Support (the percentage of occurrences)

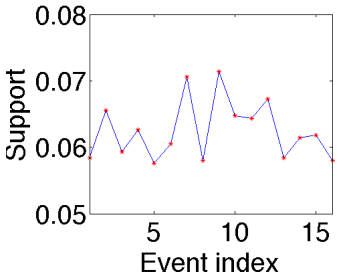
Figure A.2: Probability Matrix and support of events learned from the activity temporally randomized data

A.2 Results of temporally randomized events

The original 16 events detected from the real Edinburgh data are randomized in the temporal space with random numbers of occurrences of each event, which means all the events have the same probability to occur at each time. By going through all the event temporally randomized data, the learned probability matrix with one step is shown in Figure A.3a. The support of each event is given in Figure A.3b.



(a) Probability matrix in one step

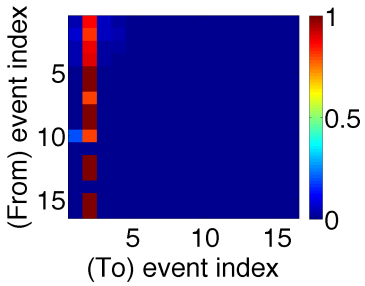


(b) Support (the percentage of occurrences)

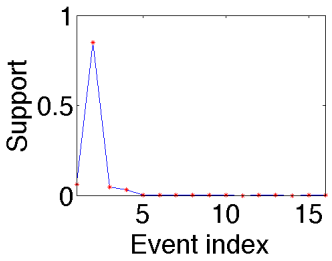
Figure A.3: Probability Matrix and support of events learned from the event temporally randomized data with random numbers of occurrences of each event

The original 16 events detected from the real Edinburgh data are randomized in

the temporal space with correspondent support of occurrences of each event as the real data, which means the possible of each event to occur is proportional to the support of events from the real data. By going through all the event temporally randomized data, the learned probability matrix with one step is shown in Figure A.4a. The support of each event is given in Figure A.4b.



(a) Probability matrix in one step

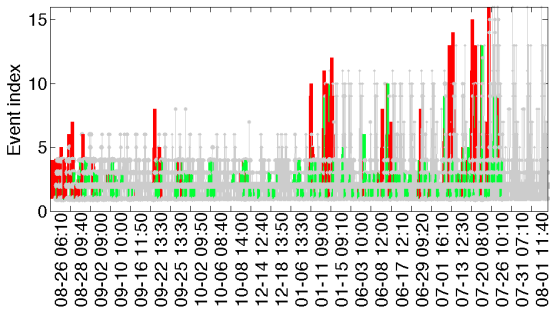


(b) Support (the percentage of occurrences)

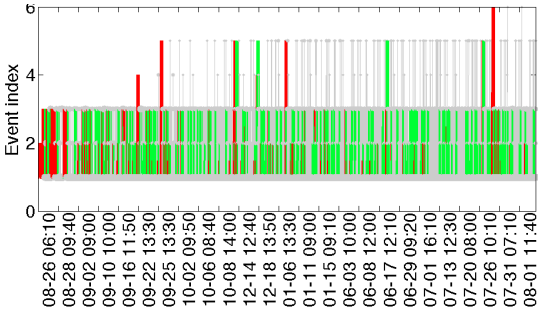
Figure A.4: Probability Matrix and support of events learned from the event temporally randomized data with real support

A.3 Comparison of the prediction results between real data and randomized data

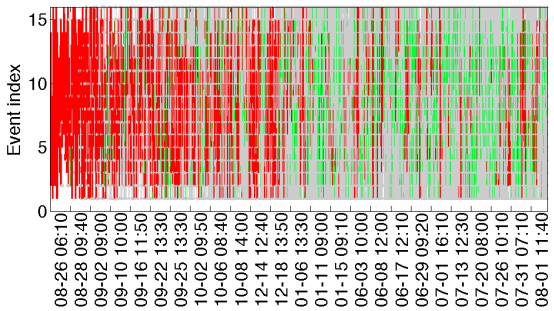
The prediction results of different datasets during all the time period we have are shown in Figure A.5. And the hourly anomaly number of each dataset is shown in Figure A.6.



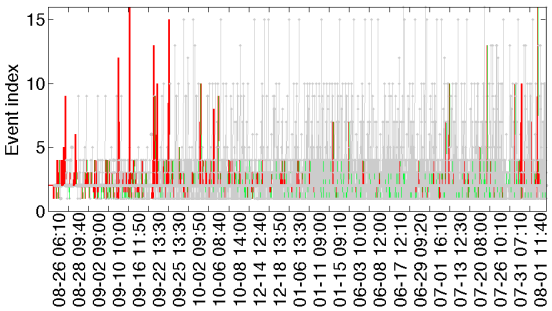
(a) With the real Edinburg data



(b) With the activity temporally randomized data

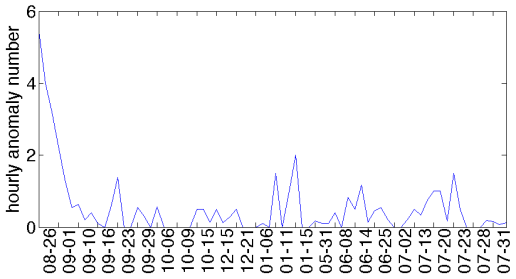


(c) With event temporally randomized data generated with random occurrence numbers of events

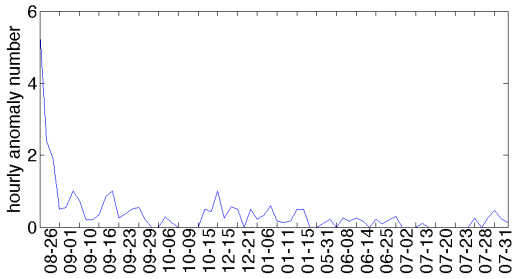


(d) With event temporally randomized data generated based on the real support of events

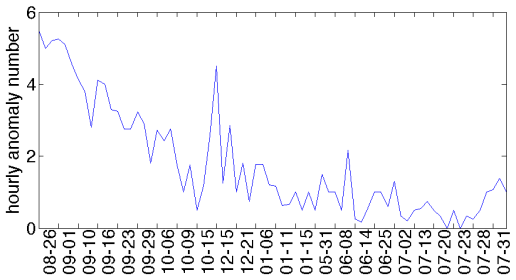
Figure A.5: Prediction results



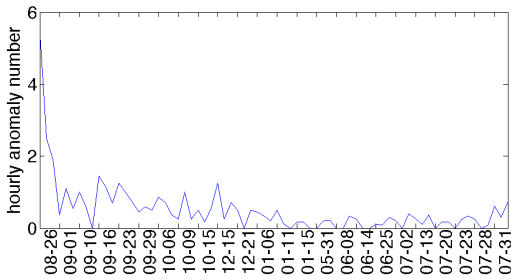
(a) With the real Edinburg data



(b) With the activity temporally randomized data



(c) With the event temporally randomized data generated with random occurrence numbers of each event



(d) With event temporally randomized data generated based on the real support of events

Figure A.6: Hourly anomaly numbers of 4 datasets

A.4 Discussions

By looking at the transition probability matrix of the randomized datasets, as shown in Figure A.2a, Figure A.3a and Figure A.4a, we could see the probability of one event transiting to the other event only randomly depends on the support of the event. However, for the real data, there are certain possible hidden rules, which means certain events have higher probabilities of occurring after some events aside from the dependency of the support of the events, as shown in Figure 6.8a.

By looking at the hourly anomaly number of different datasets over time as shown in Figure A.6, since the real data is not totally randomized and there are possible rules which are learned by the system, we still could see some higher peaks

over time along the curvature of hourly anomaly numbers as compared with the randomized dataset. However, the hourly anomaly number of both real and randomized datasets decreases over time. This is because the system incrementally learned all the possibilities, and after certain a length of time, all the possibilities are learned, which determines everything seems possible. That is to say we will lose prediction powers. On one hand, this is the limitation of our current learning model. On the other hand, this is because the real data is not ideal with pure and clean temporal patterns from the perspective of our learning model. We assume there are certain temporal patterns hidden in the real data, however mixed with noises. The future work will be to improve the current learning model by considering several aspects as we illustrated in the chapter of future work. By incorporating those three aspects, we could expect that the predictive model outputs fewer varieties of possibilities and therefore the predictive power can be increased.

Appendix B

RIT twitter data

Due to the fast development of social networks, more and more people use smart phones to socialize with each other. Twitter offers a very fast and easy way for users to update their status at any place and time, which can be regarded as an important feature to detect activities. In this thesis, the twitter data on RIT campus are collected to understand the activities going on campus. In order to give you some general ideas of RIT academic schedules, a brief introduction is given as follows.

Before September 2013, RIT has been using 10-week quarter system for full-time undergraduate and graduate programs. There are four quarters for each academic year, with three among which are primary for students to take courses which starts from Labor Day in early September and ending in late May. During the summer academic quarter, most of the students are not on campus. Most of the people staying on campus during the summer are graduate students doing their thesis projects, and research faculties or staff. There are one week breaks between academic quarters and a two-week break for Christmas. During the academic quarter, courses on campus are offered from 8:00 am to 10:00 pm on each weekday. During each year, there are certain large scale special events going on campus, such as first year student orientation, the commencement ceremony after the final exam in the spring quarter, two career fairs in the spring quarter and the fall quarter respectively, Imagine RIT festival for 1 day starting from the first Friday of May and night events

(like social free dinner, hockey game, student activities and so on). When a certain large scale event is held on campus, there will be many people gathering on campus and certain associated features like the cars in the parking lot and electric usage will also increase. And on some days the school is closed such as on Memorial day and Independence day, which can also be observable from the several recordable features, such as the car number in the parking lot, internet traffic, and electricity usage, social network usage (such as twitter).

From September 2013, RIT starts to use semester academic system. The event or activity features may follow a new pattern.

B.1 Data collection

Twitter offers the search API for developers to obtain the tweet information from the current time point back to a week or 1500 tweets nearby a specific geolocation. Figure B.1 shows the geospatial distribution of all the tweets nearby RIT campus within 1 mile from December 21, 2012 to December 28, 2012. The place that has the highest density of tweets can be considered as the place where there are people and activities around. Figure B.2 shows the temporal distribution of the tweets sent out at each featured place. We could see that on December 23 to 25 there are no tweets sent out around student life center, which is consistent as we expected because the student life center is closed on December 24 and 25, and on December 22 it only opens for very short time. Besides the students on campus probably all leave for their Christmas holiday on December 23. Therefore, tweets information can be used as an important feature to detect human activities.

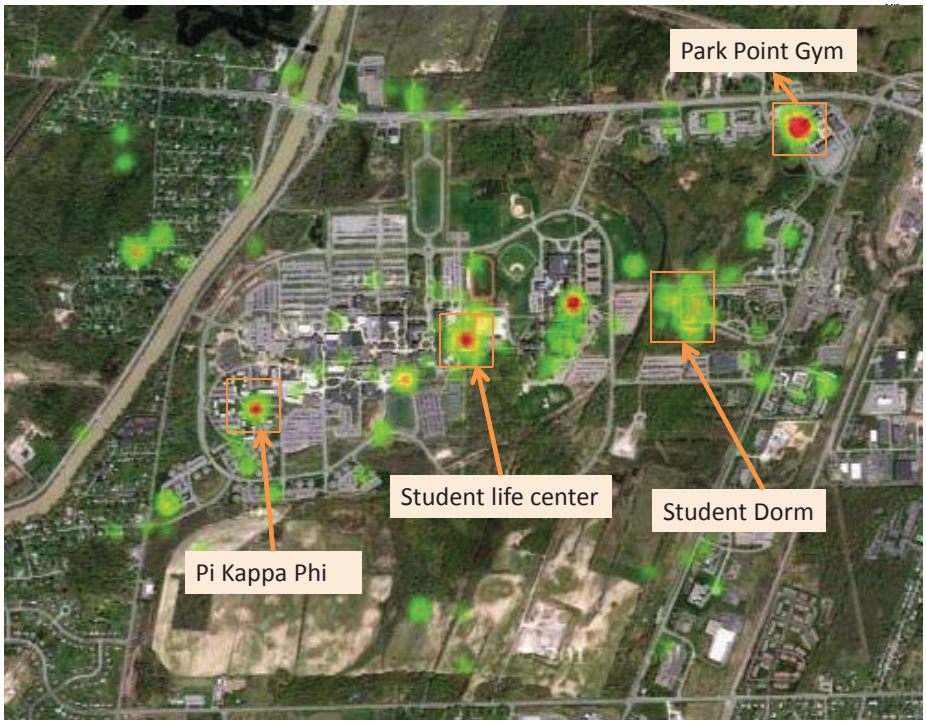


Figure B.1: Tweets spatial distribution as heat map on RIT campus from December 21, 2012 to December 28, 2012 (note: the higher density of tweets shows more red with the support from Google map API and twitter search API)

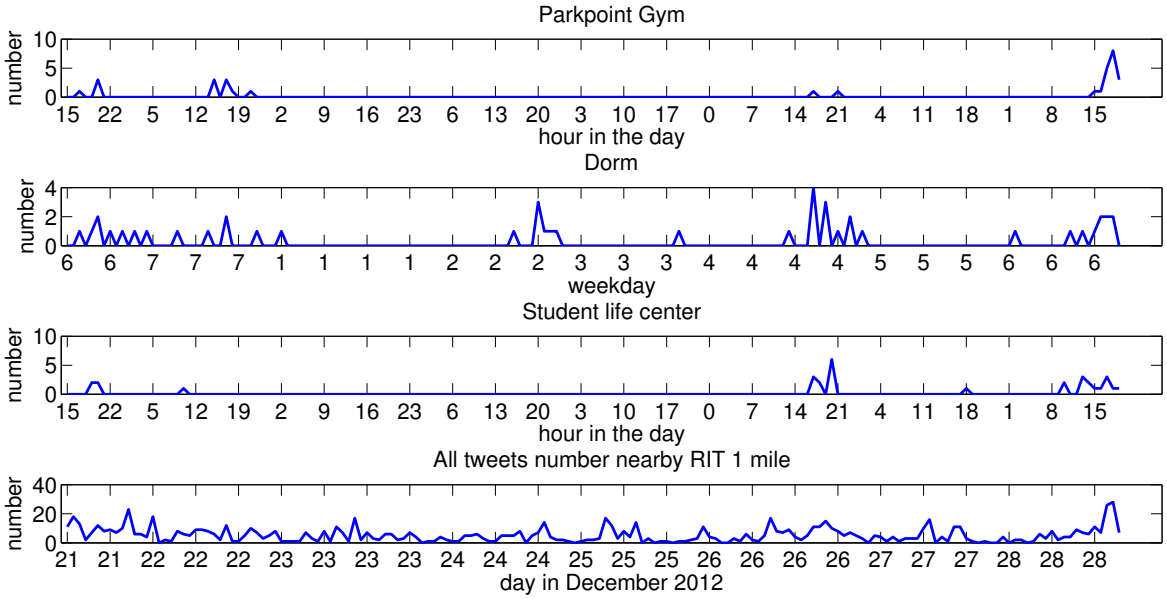


Figure B.2: Tweets temporal distribution on RIT campus from December 21, 2012 to December 28, 2012

The twitter data have been collected since December 21, 2012 until September 16, 2014. The data are saved both in MySQL and in csv files with twitter username, tweet text content, tweet geolocation(longitude and latitude), and tweet time.

B.2 Preliminary data processing

The twitter data from December 21, 2012 to May 28, 2013 which are extracted from the twitter API are used as an example to illustrate the processing process. The tweet number sent by users per hour are counted and plotted as shown in Figure

B.3. During each day, there are 24 measurement points.

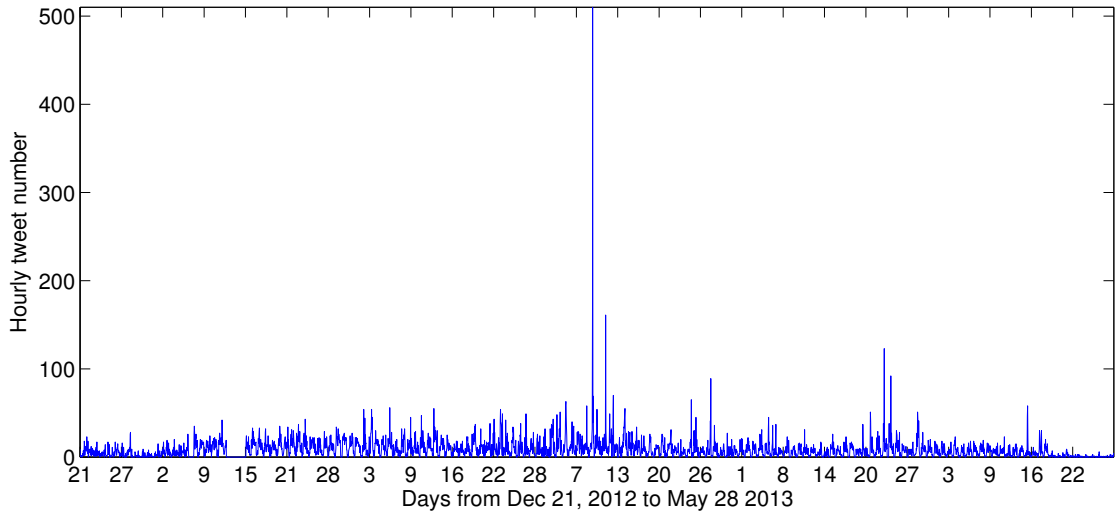


Figure B.3: Hourly tweet number

B.2.1 Preprocessing the data

Due to the mistakes made during the data collection, the twitter information on January 13 and 14, 2013 is missing and the information on part of the day January 6, 7, 12 and 15, 2013 is also missing. Since currently we do not have the information of other features which can aid the process of estimating the missing data of the twitter information, the days on which the twitter information is missing are assumed to be normal and the missing part of the data is assumed to be the same as on the nearest day when the information is available. The data series with the missing data estimated and filled in are shown in Figure B.4.

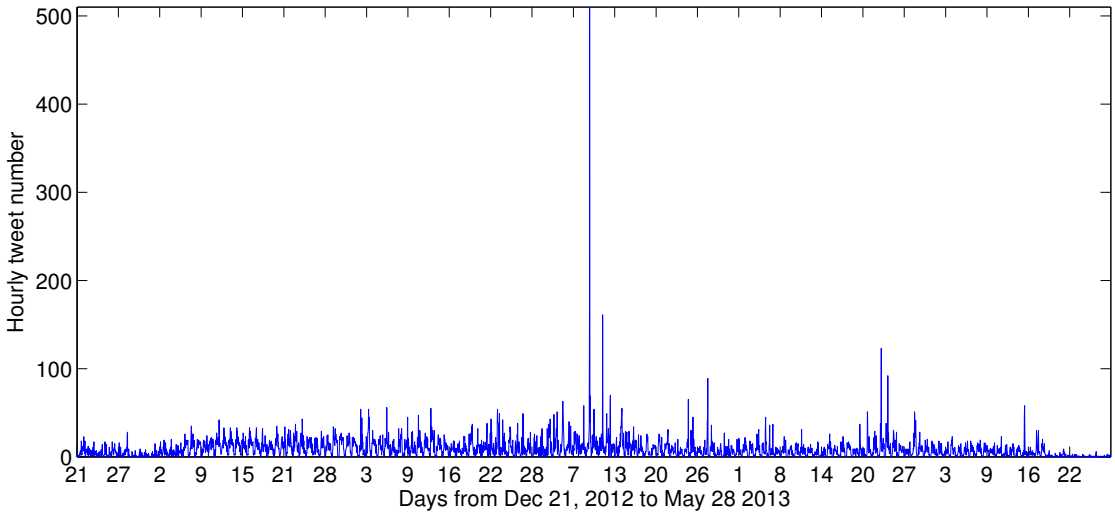


Figure B.4: Hourly tweet number with missing data estimated

Besides, the preprocessing step of the data to remove the outliers omitted here, as we do not have clear criteria to determine whether a measure point is an outlier or an anomaly. The text content of the tweets will be analyzed to give a general idea of the data points, which may be helpful to determine whether the outlier is because of a special event. The text content extraction of the tweets will be implemented after the step of event recognition. In addition, the collected twitter data are believed to be noise free.

B.2.2 Result of event recognition

We assume that the event is daily based and each event is associated with a special pattern of time series; therefore the day with significant different distribution of twitter numbers per hour is considered as anomaly. The methods illustrated in section 4.2.3 are used to understand the dataset as the first step, in order to check whether our assumption about the events is correct. The anomaly percentage of the whole data set is set as 15%, which is then used to determine the threshold of each method.

K-means, RX detector and TAD detector are three methods of event recognition based on the features generated from the time series.

In the case of data collected from twitter API, 8 features of the time series during a day are selected. The 4 features among 8 are obtained from the time series itself during each day, which are the mean, variance, skewness, and maximum of the time series during each day. The rest 4 features are obtained from the histogram of the time series during each day. The histogram of the time series during each day is first calculated. The rest 4 features are the summation of the histogram value in the highest 15% bins, the skewness of the histogram, the mean and the variance value of the big histogram (bigger than the mean of the histogram).

PCA (Principle Component Analysis) is performed on the data of the 8 features. The percentage of variance along each principle component is shown in Figure B.5. As we see, principle component 2 to 8 does not carry significant variance of the data, which is less than 1% variance of the data. Therefore, the first 1 principle component is selected as features which are used for the further processing. We should be aware that different numbers of samples can render different distribution of percentage of variance along each principle component. Figure B.6 shows the percentage of variance carried in the first significant principle component with different number of samples fed in.

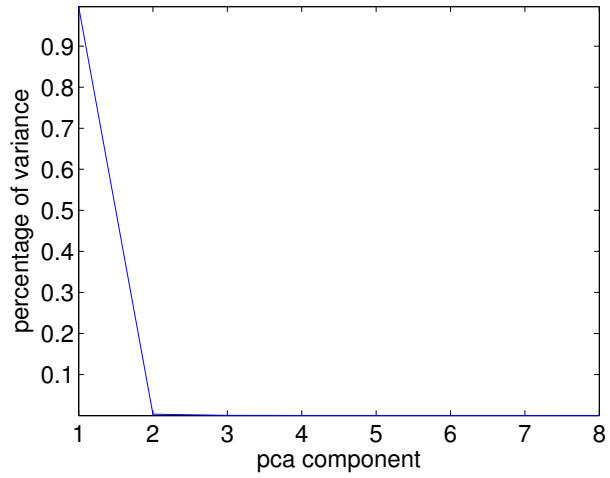


Figure B.5: Percentage of variance along each principle component

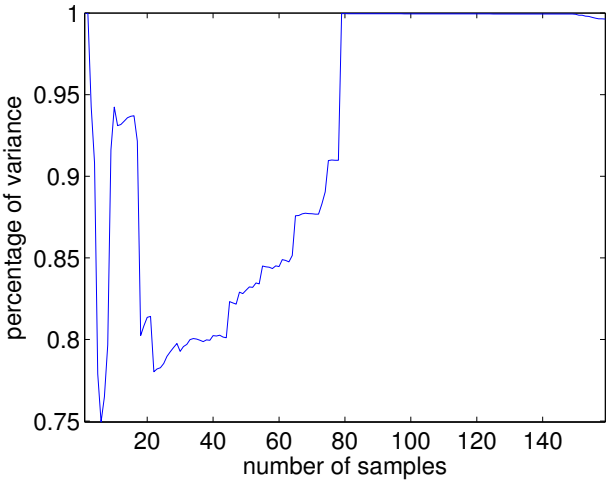
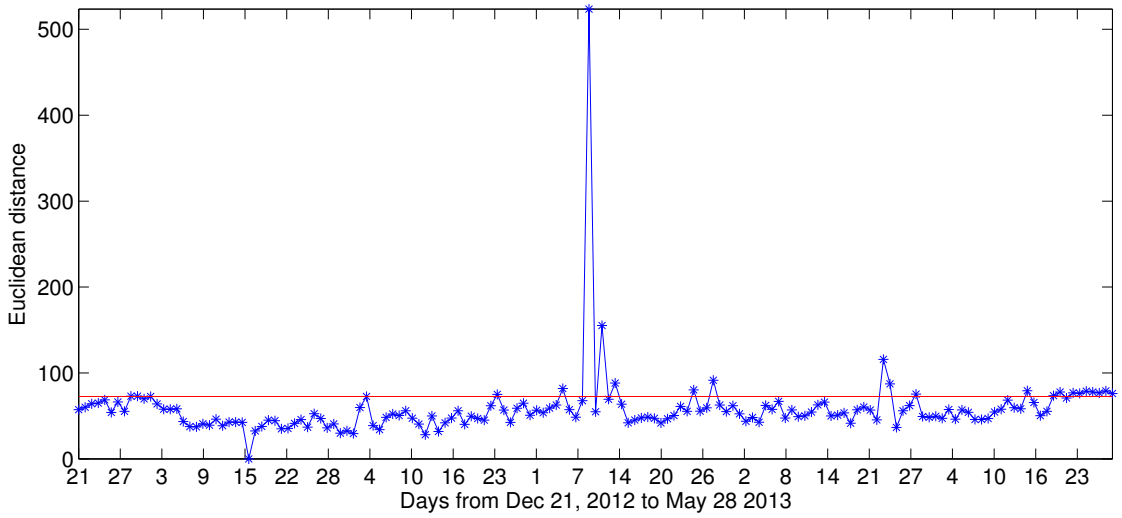


Figure B.6: Percentage of variance carried in the first PC VS. the number of samples

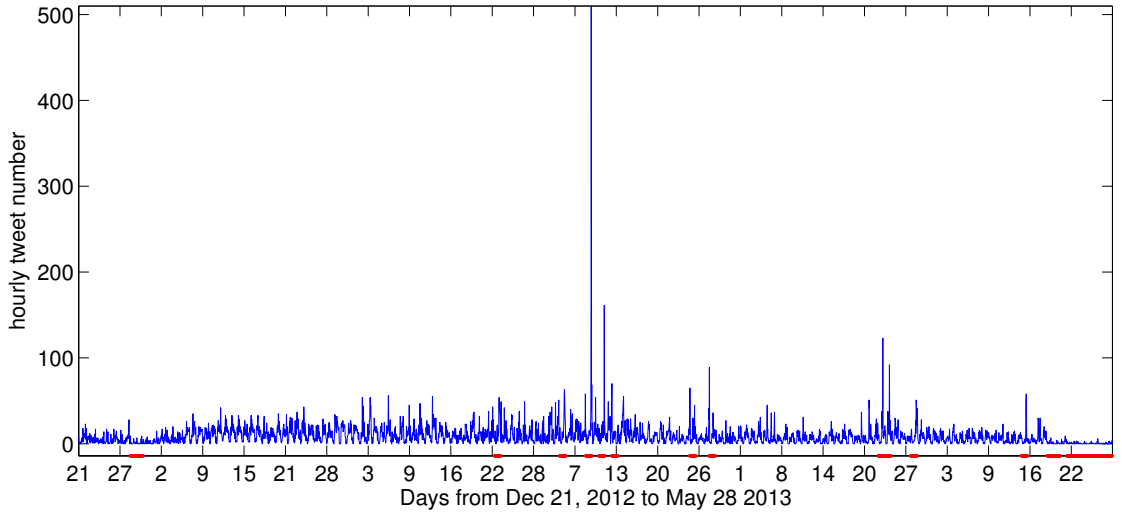
B.2.2.1 Result of Euclidean distance

As we mentioned, the Euclidean distance between two time series is one method to determine the similarity of two time series. The measured time series of one normal day is assumed to be known and it is selected as January 16, 2013. The similarity of the time series between all the other days and the assumed normal day is quantized

by the Euclidean distance and the result is shown in the lower picture of Figure B.7. The bigger value of the Euclidean distance implies less similarity between the tested day and the normal day. By assuming that there are 15% of the data which are anomalous, the threshold of the Euclidean distance is calculated. The tested days with highest 15% Euclidean distance are considered as anomalous as shown in the upper picture of Figure B.7 and marked with the red color.



(a) Euclidean distance and the threshold to determine anomalous days (the red line is the threshold)



(b) Anomalous days (Marked as the red color)

Figure B.7: Result of Euclidean distance similarity method

B.2.2.2 Result of k-means classification

The unsupervised classification method k-means is used to classify the data set in the selected 1 PC space and k is preliminarily assumed to be 6. The classified result in the PC space is shown in Figure B.8. The corresponding result in the time series space is shown in Figure B.9. In this case, the clusters with few elements (less then 20% of the whole data set) are considered as anomalous, which results in the data set marked with blue and black are normal. The anomaly percentage in this case is 13%.

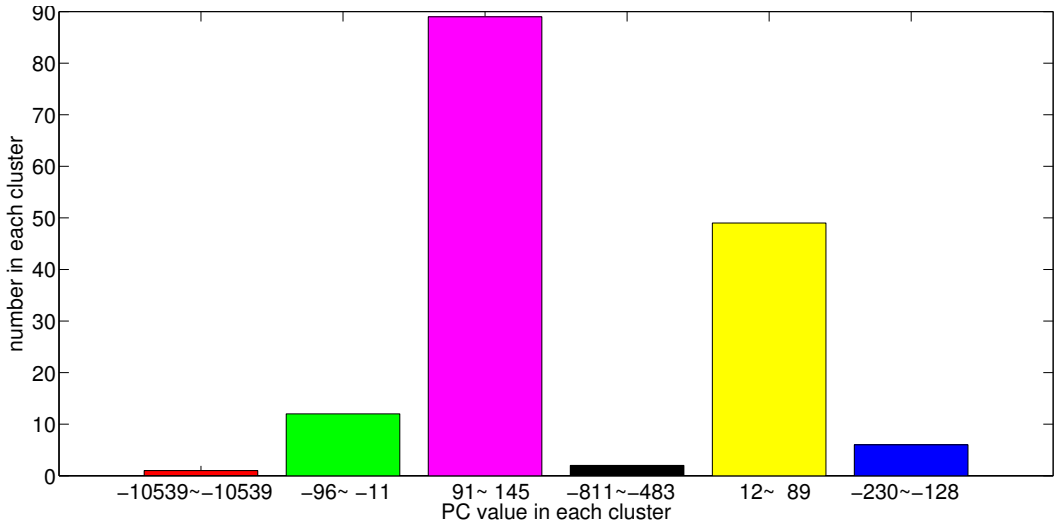


Figure B.8: Classified PC space using k-means classifier with bar plot

B.2.2.3 Result of RX detector

RX detector is used to detect anomalies in the space composed by the selected 1 significant PC. The threshold is calculated by assuming that the anomaly percentage is 15%. The result in the PC space is shown in Figure B.10. The corresponding result in time series space is shown in Figure B.11. Note that the anomalies are marked by red color.

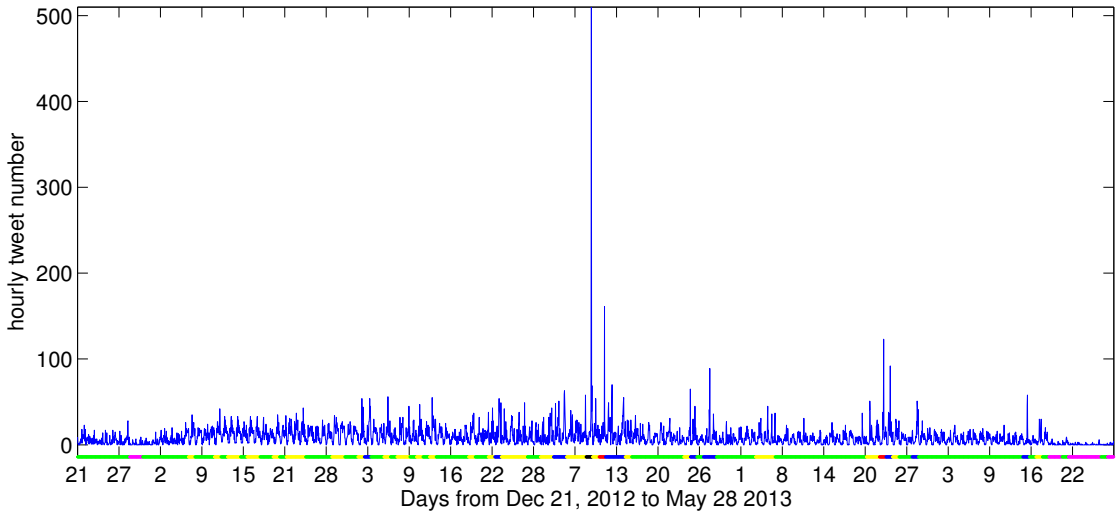


Figure B.9: Classified time series using k-means classifier

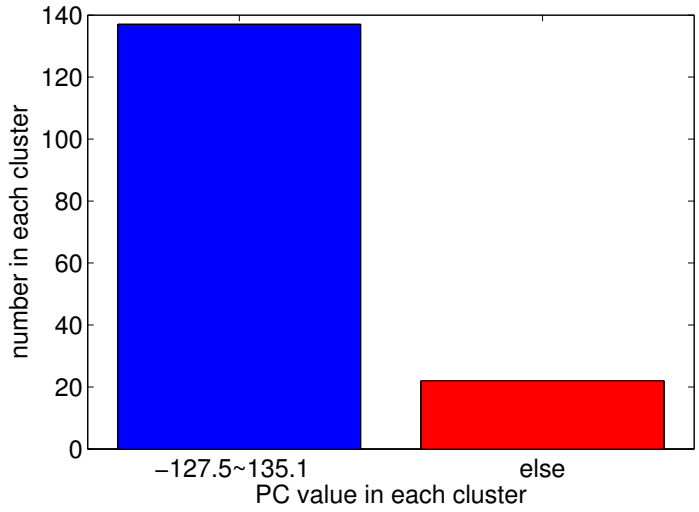


Figure B.10: Result of RX detector in PC space with bar plot

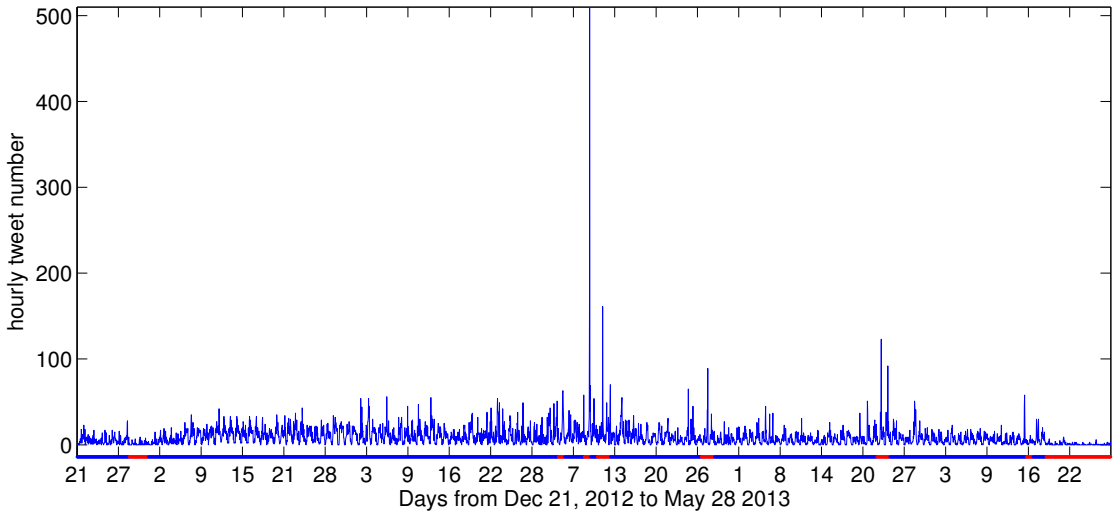


Figure B.11: Result of RX detector in time series space

B.2.2.4 Result of TAD anomaly detection

The TAD method is used to detect the anomaly day from the whole data set in the selected PC space. The TAD score is shown in the lower figure of B.13. The threshold is calculated by selecting the top 15% TAD scores, which is then used to separate the anomaly day from the normal day. The anomaly days are marked with red color in upper figure of B.13. In the PC space, TAD method separates the anomaly day from the normal day as shown in Figure B.12.

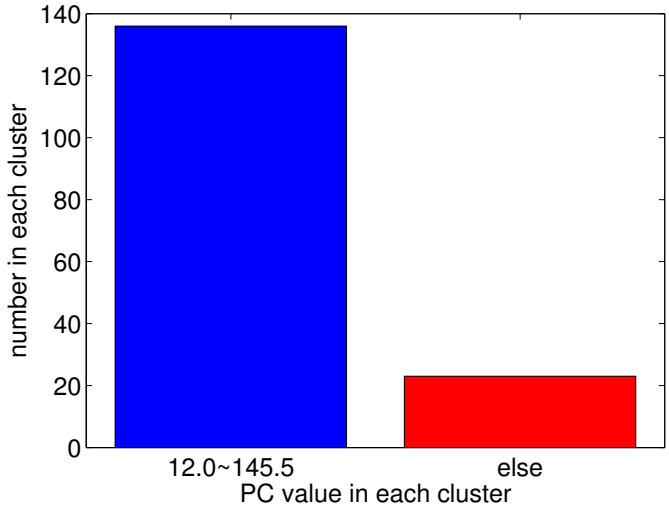
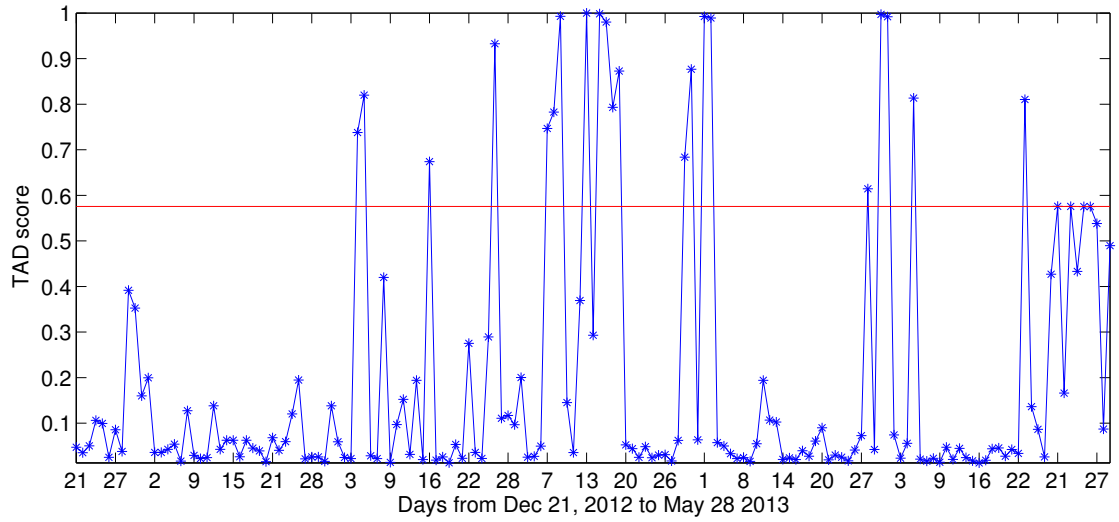
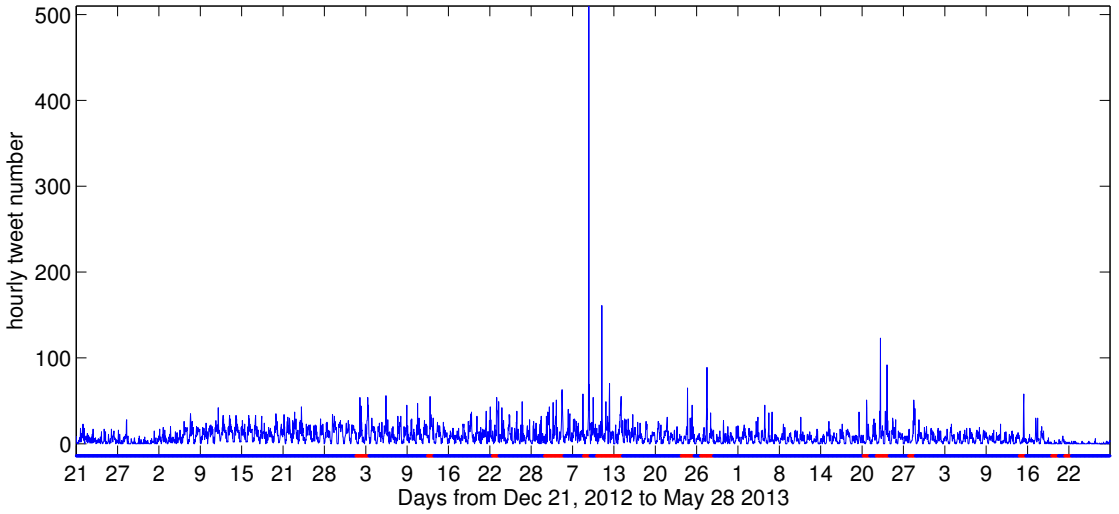


Figure B.12: Result of TAD detector in PC space with bar plot



(a) TAD scores and the threshold marked as a red line



(b) Anomalous days (marked as read color)

Figure B.13: Result of TAD in time series space and TAD score

B.2.2.5 Result of Fourier filter

First the original time series as shown in Figure B.4 is transformed in frequency domain by performing Fourier transformation. In order to obtain the cyclic behavior of the time series and to smooth the time series, a filter is designed to keep the significant frequency components. The significant frequency components are found by using a peak finder tool. The inverse Fourier transformation is performed on the filtered frequency domain to obtain the smoothed time series as shown in Figure B.14. The difference between the original and filtered time series is shown in Figure B.15, which is also considered as the anomaly or noises. The threshold is then calculated based on the assumption the anomaly percentage of the whole data set is 15%. The result of Fourier filter method by using the calculated threshold to determine the anomaly is shown in Figure B.16. The anomaly days are marked as red.

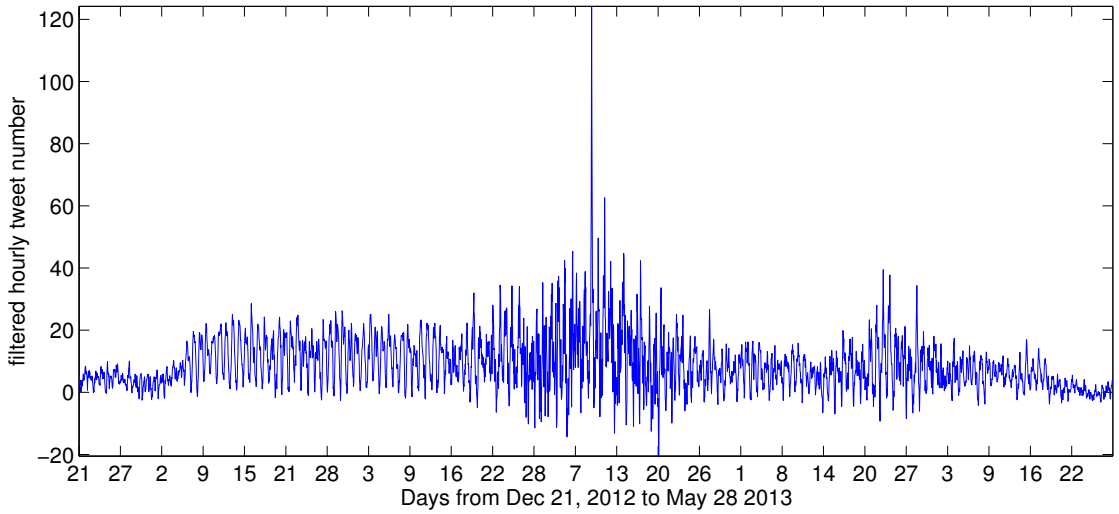


Figure B.14: Fourier filtered time series

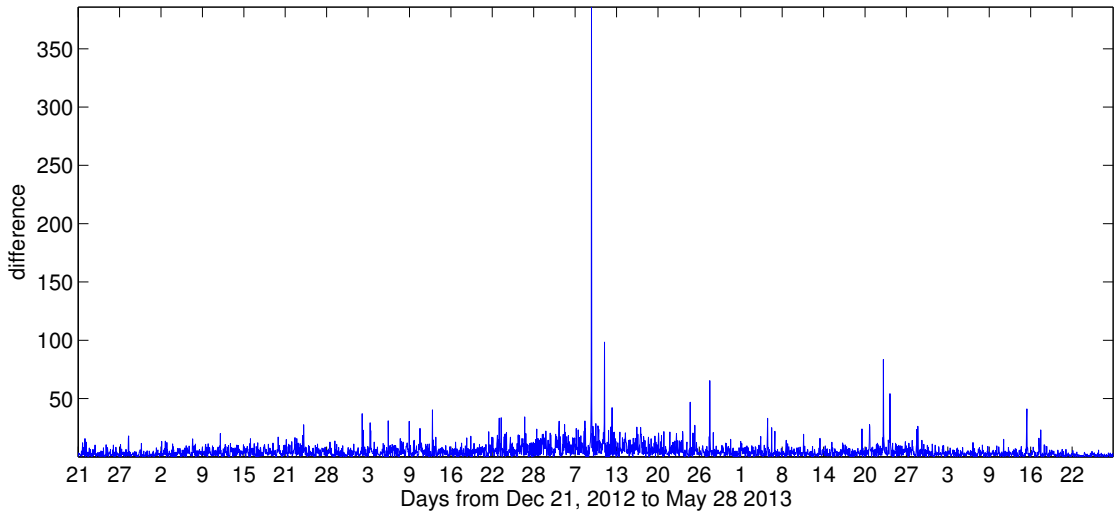


Figure B.15: Difference between the original and filtered time series

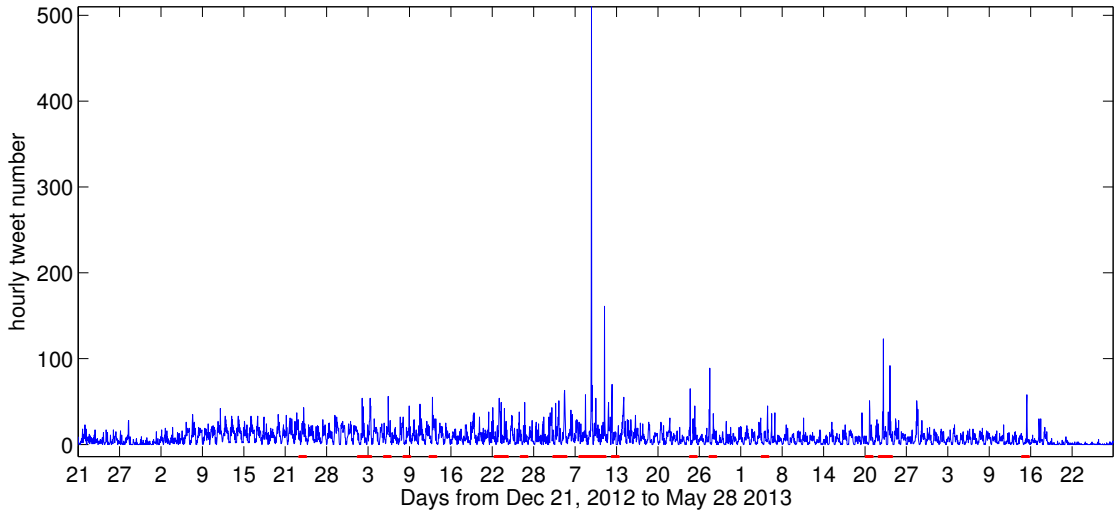


Figure B.16: Result of anomaly detection by using Fourier filter

B.2.2.6 Result comparison

As we see from above results, the five methods of event recognition can detect the event in different levels. Table B.1 shows how the results of different event recognition methods differ from each other. As we do not know the standard criteria to determine the anomaly, it is not so easy to quantize the effectiveness of the different event detection methods. However, from the point of determine the threshold of different methods, TAD detector can separate the anomalous days and normal days more obviously, by comparing to other methods, such as Euclidean distance and Fourier filter, which can also been seen from Figure B.17 to Figure B.20. These figures shows the histogram of the values of differences which is used to separate the background and the target.

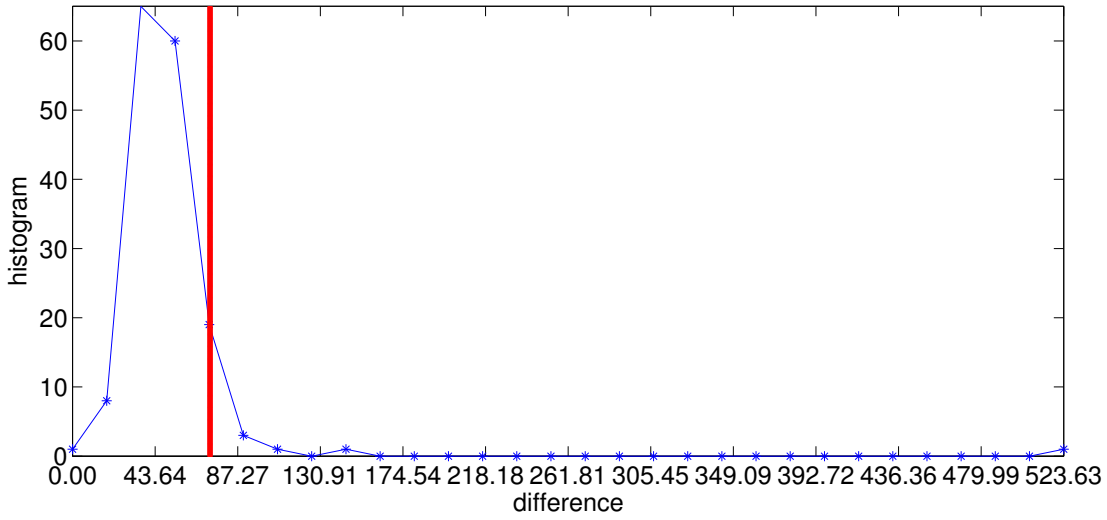


Figure B.17: The histogram of Euclidean distance and the threshold chosen (marked as a red line)

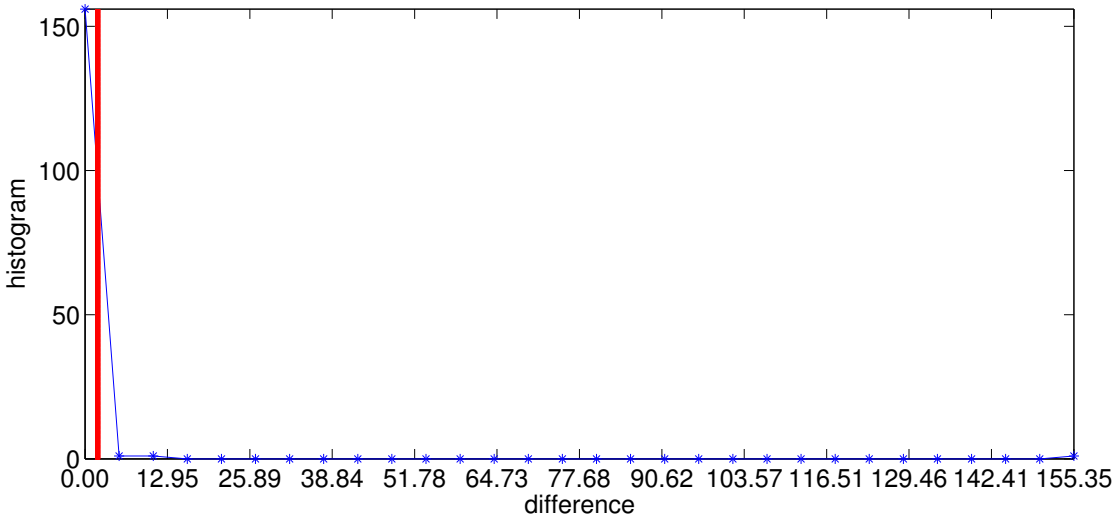


Figure B.18: The histogram of RX score and the threshold chosen (marked as a red line)

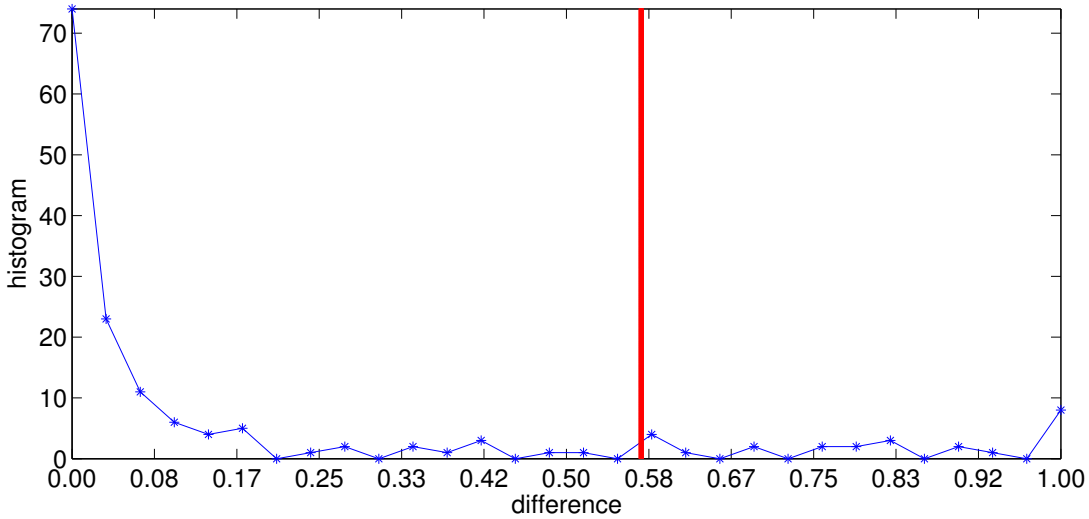


Figure B.19: The histogram of TAD score and the threshold chosen (marked as a red line)

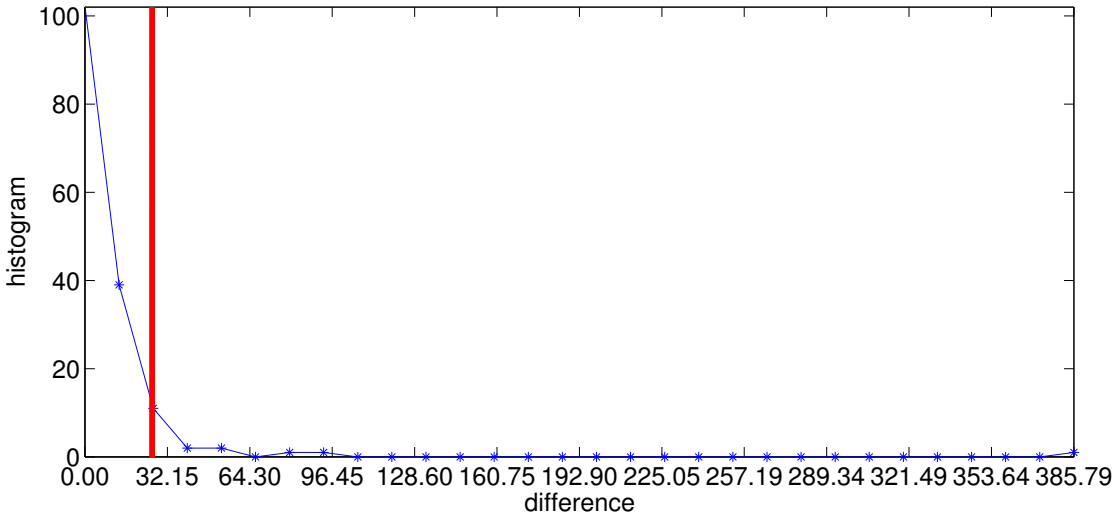


Figure B.20: The histogram of differences between FFT filtered series and raw time series; the threshold is marked as a red line

B.2.3 Event content understanding

In order to check the type of the detected anomalous event, we need to understand what is happening on the anomalous days. Twitter API offers the twitter text content, which is considered as one method to access the event content. Besides, by checking the twitter text content, we can also determine whether the detected events are just noises, which should be ignored. The most frequent words are extracted from the twitter text content, which is considered as the preliminary understanding of the event content.

B.2.3.1 Method

Several steps are done to get the most frequent words in the tweet texts.

1. The tweet texts are extracted in the interesting time range.

2. The text string is then separated into words by the delimiter of space between words.

3. A blacklist is created to store the words which are not interesting, such as the

day(yyyy-mm-dd)	Euclidean distance	K-means	RX	TAD	Fourier filter
2012-12-29	x		x		
2012-12-30	x		x		
2012-12-31			x		
2013-01-24					x
2013-01-29		x			
2013-02-02		x		x	x
2013-02-03		x		x	x
2013-02-06					x
2013-02-09					x
2013-02-13		x		x	x
2013-02-23	x	x		x	x
2013-02-24					x
2013-02-27					x
2013-03-03		x		x	
2013-03-04		x		x	x
2013-03-05	x	x	x	x	x
2013-03-08					x
2013-03-09	x	x	x	x	x
2013-03-10					x
2013-03-11	x	x	x	x	x
2013-03-12		x	x	x	
2013-03-13	x	x		x	x
2013-03-14		x		x	
2013-03-24		x		x	
2013-03-25	x	x		x	x
2013-03-27		x	x	x	
2013-03-28		x	x	x	x
2013-03-29	x				
2013-04-05					x
2013-04-21		x		x	x
2013-04-23	x	x	x	x	x
2013-04-24	x	x	x	x	x
2013-04-28	x	x		x	
2013-05-15	x	x		x	x
2013-05-16			x		
2013-05-19	x		x		
2013-05-20	x		x	x	
2013-05-22	x		x	x	
2013-05-23	x		x		
2013-05-24	x		x		
2013-05-25	x		x		
2013-05-26	x		x		
2013-05-27	x		x		
2013-05-28	x	x	x		

Table B.1: Event recognition result comparison

Conjunction words, Preposition words, and so on. However, currently the blacklist is not rich enough to cover all the non-interesting words and it is also sometimes hard to decide which words are non-interesting.

4. The words in step 2 are then filtered by deleting the words which are shown on the blacklist.

5. The times of each word tweeted by the twitter users are counted.

B.2.3.2 Result

By using the method described above, the favorite top 30 tweet words of selected anomalous days are listed in Tables B.2 and B.3. The frequency of each word shown up on the tweet text is listed besides the tweet word.

B.2.3.3 Discussion and future work

As we see from the tableB.2 and B.3, the frequent tweet words on some anomaly days can tell something about what possible events occurred. For example, on February 3, 2013, superbowl is hold and the keyword of "superbowl" is on the top 3 sent by the user. On February 13, 2013, one day before valentine's day, the user may be more active on the social network which results in the day is regarded as anomaly. On May 15, 2013, it looks like #rit and #tuft have a sport game together.

However, it is still very difficult to understand how the tweet number is related to the activity. On one hand, some normal days may have a high tweet number. For example on March 9, 2013, the tweet number is very large, but by looking at the top tweet words, it is hard to guess what is going on on that day. The high number of tweets may be because of some tweets from a popular user is retweeted over and over again, as "@trawwquotes:" is quoted 727 times. On the other hand, some days, with events like Imagine RIT and Commencement ceremonies when we expect to have very high tweet numbers, are not categorized as anomaly days.

Therefore, by using only twitter information to detect all the days having interesting events, the result may be biased or not accurate. However, we find the twitter data can well separate the days of RIT defined holidays and of non-holidays. Here

2/3/2013	2/13/2013	3/3/2013
49ers, (22)	@chr_seungmigpb (22)	http://t.co/lseu8dqdph (117)
wind. (21)	que (19)	mac (116)
superbowl (16)	les (16)	h-gang (116)
ravens (15)	nun (14)	#hitsingle (115)
tonight. (12)	@nejooe: (14)	amo (20)
love (12)	lool (13)	pas (16)
good (12)	pas? (13)	les (13)
@gerrythornberg: (11)	rit. (11)	cst (12)
super (11)	cest (11)	que (10)
commercials (10)	@kissminzy2ne1: (10)	che (9)
@rit (10)	cst (10)	@sofiane78vp (8)
@vancedeatherage: (9)	say (9)	@gwaadacaaps (8)
commercial (9)	@ariiportilla (9)	vous (7)
prefer (9)	& (8)	qui (7)
game (9)	jvais (8)	sur (6)
time (8)	time! (8)	pour (6)
@xalyciavanheese: (8)	mdrr (8)	des (6)
especially (7)	kiss_kyungsoo: (7)	love (5)
beyonc?'s (7)	college (7)	jai (5)
lewis (7)	des (7)	bien (5)
@secsirita: (7)	@elizakeeys (7)	tes (5)
@caseygonta: (7)	pour (7)	#souffrant (5)
flows (7)	toi (7)	comme (4)
ray (7)	jai (6)	moi (4)
naturally (7)	@eazyscotty78 (6)	debt, (4)
team (7)	sorry (6)	@monsterxcept (4)
well (7)	@iamsicajung (6)	non (4)
hair (7)	trop (6)	allez (4)
bowl (7)	@lollipop_yoona (6)	mdrr (4)
& (6)	new (6)	@marokinio78 (4)

Table B.2: tweet content on an anomaly day

3/4/2013	3/9/2013	5/15/2013
http://t.co/lseu8dqdph (60)	@trawwquotes: (727)	#rit (48)
mac (60)	shit, (110)	#tufts (31)
h-gang (60)	side. (101)	quarter (13)
#hitsingle (60)	didn't (85)	offense (12)
pas (17)	everybody (78)	time (11)
@rich_billsmafia (11)	love (75)	goal (11)
vous (10)	wrong (69)	another. (11)
que (10)	give (69)	first (10)
mdrr (9)	dear"" (63)	ball (10)
brony (8)	text (63)	minutes?! (9)
& (8)	worth (61)	left (9)
today, (8)	fuck (58)	game (8)
@nejooe: (8)	cheater, (54)	score (7)
@deesse_goddess (8)	people (53)	love (7)
love (7)	head? (51)	watkins (7)
@xgivemeciasten (7)	broke? (51)	rochester (7)
@keymangbujoli (6)	business. (51)	jumbos (7)
@mamzelleab (6)	mf's (51)	lead. (7)
what's (6)	pregnant? (51)	great (6)
cst (6)	again." (51)	remaining (6)
wrong, (6)	worried (51)	goalie (6)
tes (6)	money. (51)	crease (5)
class (6)	care, (47)	high (5)
rit (6)	expect (45)	violation (5)
good (6)	retweet (45)	final (5)
time (6)	everyday, (44)	play. (5)
say (6)	& (43)	stadium (5)
savez (6)	mom. (39)	half (5)
fan. (5)	girlfriend!! (38)	stick (4)
telling (5)	boyfriend, (37)	keep (4)

Table B.3: tweet content on an anomaly day

RIT defined holidays include national holidays and RIT quarter break or semester break. Figure B.21 shows the hourly tweet number from December 2013 to March 2014 with RIT defined holidays marked. And we also see some differences between weekdays and weekends. Normally, during the holidays and weekends, the tweet number is significant lower than the other days.

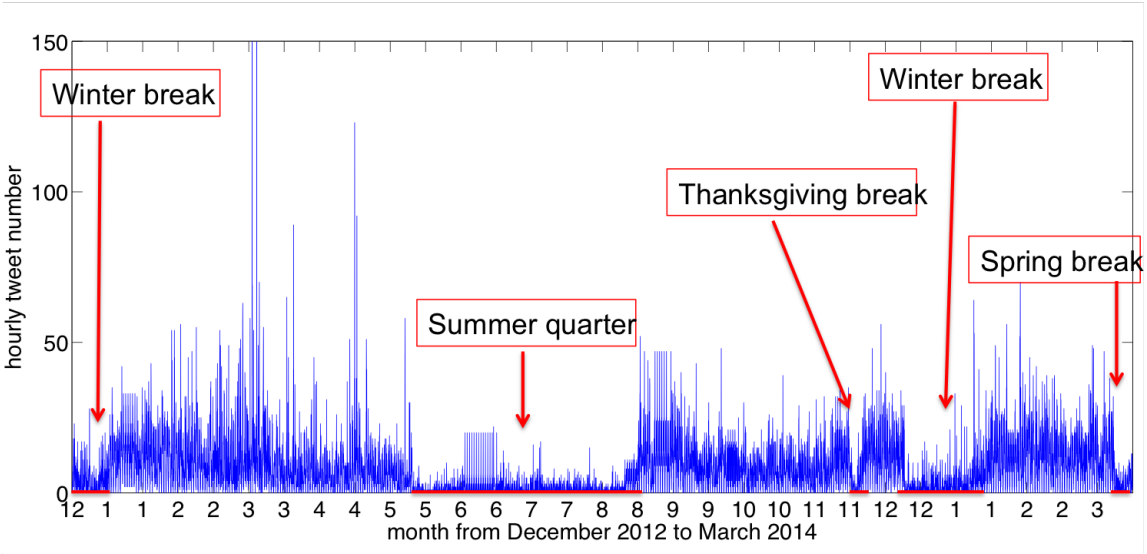


Figure B.21: Hourly tweet number from December 2012 to March 2014 with RIT defined holidays marked

Therefore, with the twitter data, the next step would be to recognize the daily events, such as holidays and nonholiday, rather than try to separate the specific daily events, such as the day with the career fair, Imagine RIT and so on. With this prior knowledge of twitter data, we could continuously recognize each day as a holiday or a nonholiday, and then memorize the event with the method of double localization we illustrated in section 6.1. During the memorization step, a temporal map can be set up as the absolute localization. In order to embed effective prior temporal patterns into the temporal map, "weekday" and "month" can be used as the coordinates of the

temporal map. Further more, the temporal map can be extended into temporal cube by adding another coordinate such as "year", so that we could learn how the twitter user activity on RIT campus changes over the year. In order to relative localize an event, the Markov chains can be used learn the transition probability from one event to another one. Based on the temporal map and transition probability matrix from the Markov chains learned from the historical twitter data, we could then make predications and detect anomalies. Detailed implementation of the whole frame work with twitter data is not pursued in this thesis as we did with Edinburgh dataset.