

Design and Implementation of an Automated Anomaly Detection System for Crime

Jeffrey T. Bordogna, Donald E. Brown, James H. Conklin

Abstract—Anomaly detection in the law enforcement domain is very important because it gives police agencies a method for discerning the difference between normal fluctuations and fundamental changes in a specific crime distribution. Currently, crime analysts must monitor the data manually to detect these anomalies because an automated tool does not exist. This paper describes the analysis, design, and implementation of a system, entitled Sentinel, which directly addresses the anomaly detection problem.

The Sentinel system is comprised of a robust anomaly detection back-end program built around a Zero Modified Poisson model and a user-friendly web interface. The back-end program constantly monitors user specified crime levels over time and automatically alerts users via the web interface when these crime levels change significantly.

This tool has been completed and integrated into the Web-based Criminal Analysis Toolkit (WebCAT), a criminal analysis program developed by the Systems Engineering Department at the University of Virginia. Although more long term comprehensive testing remains, initial results show that the tool provides a more accurate, time efficient method of detecting anomalies in criminal data sets than currently exists.

I. INTRODUCTION

According to a 2004 Virginia Crime Analysis Network (VCAN) survey, 40% of crime analysts do not use software programs to analyze criminal activity [1]. Yet it has been well documented that this type of analysis, especially using intelligent data mining and data fusion techniques, is necessary for crime analysts to keep pace with their information processing requirements [2]. In response, applications such as the WebCAT system were created to provide the necessary tools for police agencies that were unable to develop or purchase their own analysis programs [3]. Although analysis capabilities have increased as a result, these tools still critically depend on the particular analyst using them. An analyst must have both the time to run the multitude of searches required for each criminal incident as well as the intuition in order to accurately analyze the statistical reports [4]. Given the limited resources of many police agencies, finding the time and the right people to do these jobs is rarely the norm [5].

This project's goal was to improve the analysis capabilities of police agencies by offering a tool that increases the detection rate of anomalies in criminal data sets while eliminating

unnecessary time overhead associated with this process. In order to effectively develop this tool, requirements were set by eliciting the needs of crime analysts and researching prior work in both the anomaly detection and crime analysis fields. Alternative solutions to fulfill these requirements were generated and subsequently evaluated to create a prototype design for the system and then this system was tested in an iterative user feedback loop. This document outlines the design process by breaking it down into the following pieces:

- 1) Anomaly Detection Background
- 2) Sentinel Detection Program
- 3) Sentinel User Interface
- 4) System Testing

After discussing the details of the project in these sections, the conclusion and future works section serve to give a general overview of how successful the project was and work that remains to be done on it in the future.

II. ANOMALY DETECTION BACKGROUND

The general idea of anomaly detection - finding events that are fundamentally different from others in a data set - has been around for as long as we as a society have been analyzing data. However, a systematic approach to detecting anomalies, meaning anything outside of an analyst manually sifting through a data set to find unusual points, is a fairly recent phenomenon and largely has coincided with the advent of the computer age.

A. History

The birth of this subject is often attributed to Jim Anderson who used the idea to improve computer security auditing and surveillance capabilities in the early 1980's [6]. Specifically, he attempted to define a baseline behavior for how a computer system was being used and then compare new usage of that system to the baseline level in order to check whether or not the new usage is anomalous [7]. He labeled this process Intrusion Detection because it involved detecting intrusions of a computer system by an unauthorized source.

Anomaly detection, a method of intrusion detection, compiles information about data points in a data set and then uses this information to create a model depicting the key characteristics of a normal point for that set. It then compares all new data points to this model to determine if they contain the necessary characteristics to be considered normal [8]. This approach is extremely proficient at detecting anomalous data points when they occur, however, it is often too liberal in its designation of such points, resulting in a significant number of false positive alerts. A false positive alert occurs

This work was supported by the Department of Criminal Justice Services
J. Bordogna is with the University of Virginia, Charlottesville, VA 22904
USA jtb4x@virginia.edu

D. Brown is with the University of Virginia, Charlottesville, VA 22904
USA deb@virginia.edu

J. Conklin is with the University of Virginia, Charlottesville, VA 22904
USA conklin@virginia.edu

when the system declares that a data point is anomalous when in fact it is not. Preliminary discussions with local police agencies yielded the following: Not responding to a legitimate problem is five times as costly to police, as responding to a false problem [9]. Since in the criminal domain, the cost of not detecting an anomaly generally exceeds the cost of numerous false alarms, using a model that is more prone to generating false positives than false negatives is acceptable.

B. Use in Crime Applications

Considerable previous work in anomaly detection has focused on either manufacturing systems or computer security. Few anomaly detection approaches have been developed that can adequately manage complex human behavior systems with high variance and large unexplained error.

The most popular method for detecting anomalies in criminal data sets is for the most part, still a manual process. A crime analyst must read through computer generated monthly reports and attempt to distinguish significant differences by comparing values from month to month. This is an expensive process, especially in terms of time and accuracy, because even the most experienced analyst will be unable to uncover many of the patterns of crime over time [10].

Applications such as WebCAT and University of Arizona's COPLink have been developed in order to provide police agencies more flexible and powerful analysis options than these simple reports [11], [1]. These programs allow users to create queries against a database using powerful filtering capabilities and then visualize the data using easily understood charts and graphs. This addition of charts and graphs to crime analysts' toolkits dramatically increases their ability to determine trends in data, while the ability to quickly aggregate similar criminal events significantly decreases the analysis time.

Despite the improvement in anomaly detection with these new tools, a more systematic approach is necessary. For example, with WebCAT, in order to monitor a crime level for a particular set of parameters over time (such as number of burglaries after 10pm), you must run a new query each time you want the crime level information updated. Given that there are 59 different crime types in the NIBRS (National Incident Based Reporting System) crime data standard and an infinite number of crime scenarios that can occur given these crime types (including time of day, weapon used, etc.), there is little chance of adequately analyzing all these situations if they have to be manually updated and subsequently analyzed each week or month [12].

Although few crime analysis applications address these issues, a system called ATAC (Automated Tactile Analysis of Crime) developed by Bair Software has a somewhat similar component called Trend Detection [13]. However, instead of predicting an aggregate crime level for anomaly detection purposes, Bair's trend detection tool simply predicts the next day that a particular incident will occur. It is hard to imagine that a tool predicting the time of a single incident could

be accurate enough in its predictions to be useful to crime analysts given the extremely high variance in crime counts.

C. Zero Modified Poisson Model

As discussed briefly before, attempting to model criminal behavior is a difficult problem. Specifically, fundamental crime distribution changes over time and small amounts of data for a particular crime level can both create modeling concerns. For instance, in a small town where there may be only 2 murders per year, attempting to predict when those two murders will occur is a very difficult task. In cases like this, the distribution of crimes cannot be thought of as continuous but rather discrete, ruling out the use of least squares statistics [23]. Osgood proposes to use a Poisson distribution because Poisson-based regression models are built on assumptions about error distributions that are consistent with the nature of event counts [14].

Hansen takes this analysis of crime incidents further by developing a control structure for examining trends and making predictions that is based on a Zero Modified Poisson (ZMP) model coupled with a Epanechnikov kernel density estimation [15]. The Zero Modified Poisson model takes into account the potentially large number of time periods where there may be no crime incidents for a particular set of filtering conditions. The Epanechnikov kernel density estimation part of the algorithm not only smoothes out the trend curve to give a more continuous feel to the discrete data set, but also adjusts estimations based on seasonal patterns of crime throughout the year [15]. An example of this seasonality pattern can be seen in Figure 1 which shows all incidents in Virginia during 2004.

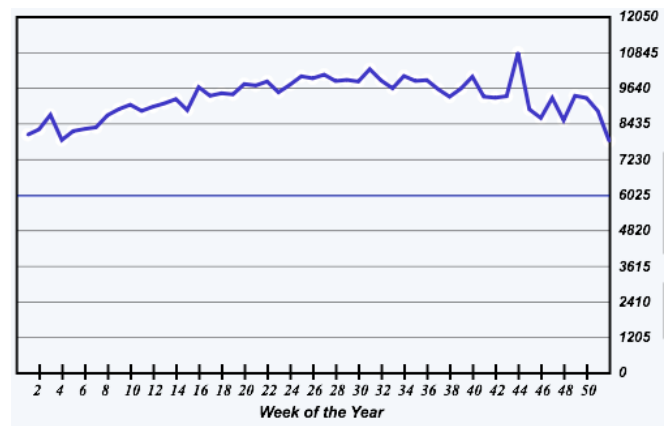


Fig. 1. Seasonality of Crime - All Incidents in Virginia 2004 [16]

The anomaly detection process runs on a weekly basis and anomalies are detected based on three basic control chart rules:

- 1) The current week is 3 standard deviations from its predicted value
- 2) 2 of last 3 weeks are 2 standard deviations from their predicted values
- 3) 4 of last 5 weeks are 1 standard deviation from their predicted values

If a new data point fails any of these tests (in combination with its previous data points for tests 2 and 3), it is cast as an anomaly. Tests comparing the algorithm against current methods of anomaly detection, specifically standard control charts which compare all new points to the distribution's mean, showed that the ZMP algorithm outperformed the others in most metrics [15]. Due to these successful results, the ZMP algorithm was used as the theoretical base for this design project.

III. SENTINEL DETECTION PROGRAM

Although Hansen designed an anomaly detection program to test the ZMP algorithm, it is not extensible enough to adequately handle flexible inputs and multiple concurrent users. Therefore, the requirements of this program were developed in part to solve these problems. It needed to be able to monitor multiple years worth of data, handle a combination of user-specified filtering parameters, and consistently output the correct alert information to the users who requested the anomaly detection service. In addition, it would need to be built on a stable development platform that could adequately handle a large number of users and a complex algorithm.

A. Alternatives

The four alternative development platforms chosen for creating the sentinel detection program were:

- 1) Stored Procedure: A central database contains a series of scheduled queries that mimic the more complicated processes of the Sentinel algorithm.
- 2) Statistical Package: The algorithm is replicated by code in a powerful statistical program such as R and is then accessed by the website via a command line web service.
- 3) Web-based Script: The algorithm is created in an online script that is triggered by another web page when it needs to be executed.
- 4) Scheduled Server Application: A customized desktop program is developed that can be scheduled to run at standard intervals or constantly monitor the database for new data to analyze.

B. Analysis

The most important metrics used when comparing these alternatives were cost of development in time and materials, the speed of processing, and the ability to handle the complexity of the algorithm. The stored procedure option in theory dramatically decreases the speed of processing since data would never have to leave the database to be analyzed. However, in practice, this option performs poorly with respect to processing speed since SQL is not optimized to handle mathematical operations and in fact is unable to emulate the powerful statistical algorithms used in Hansen's algorithm.

A statistical software package is powerful enough to adequately run the necessary algorithms but can not be programmed to deal with the complexity of the calendar

based criminal data. As a simple example, a year can contain either 365 or 366 days and in order to compare data from these years, the values must first be standardized. Other similar problems include the irregularities in week date ranges between years in the Gregorian calendar system and crime incident dates that are listed with both a start and end date because the exact time of the crime could not be determined.

The final two options, the web-based script and the desktop program, both allow a high degree of flexibility when programming the algorithm and both require a similar amount of development time and cost. However, the processing speed difference between the two is very significant. A web script running a less complex version of the algorithm for testing purposes took close to a minute to run for one crime level with a medium amount of input data. That same algorithm and data set was completed in less than 20 seconds by the desktop program.

C. Results

Ultimately, the custom built desktop program was chosen as the development environment for the anomaly detection algorithm because of its flexibility and speed. The program was developed in C# to easily integrate with existing WebCAT code and was compiled as a command line application so that it could be accessed by other desktop programs or by the WebCAT website via web services. The program is controlled by a scheduler that tells it to search the database for any evidence of new criminal data or user search parameters each week. If an analyst knows that there is new data available and does not want to wait for the scheduler, he or she can override this setting and have the algorithm run on command from the Sentinel web page.

To deal with the calendar based data problem, weekly units are defined by date ranges instead of Gregorian weeks so that data can accurately be compared from year to year. This also allows for an easier method of standardizing weekly incident counts to control for leap years. A crime with both a start and end date is broken down into a uniform distribution and its "pieces" are equally assigned to each day within that date range.

When an anomaly is discovered by the system, alert information is sent to the central database along with key summary statistics so that the Sentinel web interface can display it to the user in a clean, easy to understand format.

IV. SENTINEL USER INTERFACE

The two main components of the Sentinel system for the user are sentinels and alerts. A sentinel is defined as the unique set of filtering conditions that an analyst specifies to monitor a particular crime level. The sentinel time frame automatically gets updated each week so that a user can see changes in crime patterns over time. If the current week incident count is detected as an anomaly by the Sentinel algorithm, the sentinel creates an alert to make the analyst aware of the potential problem.

The Sentinel user interface can be broken down into three main parts:

- 1) Anomaly Detection Management System: System that allows users to manage a list of their sentinels and alerts.
- 2) Sentinel Page: Unique page for each created sentinel that displays key reporting and analysis tools.
- 3) Alert Page: Unique page for each found alert that lets user further analyze the detected anomaly information.

A. Alternatives

One major system design choice that affects several key components of the Sentinel user interface is whether to allow users to create/edit their own sentinels. Generally, making criminal analysis more automated decreases analysis time per crime scenario and thus, opens up the possibility of analyzing scenarios other than the high profile crimes that agencies currently spend most of their time on [16]. When discussing this concept with crime analysts, it became clear that users may not realize this potential and would instead continue to concentrate on the same high profile crime scenarios as before. As a way to force openness on the part of the users, a design architecture was considered that would not allow the user to specify his or her desired crime scenarios. Instead, all crime scenarios, or at least as many as the system could handle, would be monitored over time. This way, a user would be forced to consider new crime scenarios, with the hopes that crime analysis could become more complete.

If users are allowed to create and manage their own list of sentinels, a new major component is added to the user interface: a sentinel creation form. This form consists of form fields that a user can fill out in order to specify the crime level that the sentinel will be monitoring. Alternatives for designing this form consist of including all possible data filters, only including a few key filters, or creating a combination of the two by offering both forms and allowing the user to choose which to use. Furthermore, a decision needed to be made whether to allow users to specify the value of the Sensitivity Modifier m , that is, the ratio between how likely the system is to report false positives vs. false negatives. The alternatives for this choice are to allow users to specify this value, set a default value for this field and allow users to edit it, or set its value and do not allow users to alter it.

Although the evaluation of alternatives for a sentinel management system heavily depends on the initial design choice of whether to allow users to create their own sentinels, the choices remain the same: create a viewable master list of sentinels and/or alerts that is automatically ranked by relevance, importance, and date, create this same list but allow it to be sorted and altered by users, or use a folder/file system to let the user classify the sentinels/alerts their own way.

A final set of design choices deals with the layout and content of the sentinel and alert home pages. However, unlike other aspects of the user interface, these two pages consist of many small design choices rather than a few large ones. Thus,

instead of choosing between alternatives, users are choosing which features and/or information modules they would like to see on each page.

B. Analysis

Two general metrics used in these design choices were analysis time and Likert scale based usability scores. The major design choice for the user interface was whether to automatically choose the finite set of sentinels to be monitored. While user feedback suggested that this would result in a greater variety of crime scenarios being monitored because users would be forced to analyze crime levels that they otherwise would not, they believed that the level of noise created from the significant increase in the number of sentinels and the subsequent increase in the number of alerts, would distract the user from accomplishing his or her original goal.

The design choice for the setup of the “create sentinel” form involves a direct tradeoff between analysis speed and usability. If a short form with only a few key filtering parameters is used, analysts are able to more quickly create and analyze sentinels. However, if a user needs to filter the data set by more parameters than are available in this short form, they cannot do so. Similarly, giving a user the ability to select the sensitivity modifier m in this form slows down the sentinel creation process but allows the flexibility for users to alter the threshold for anomaly detection rules.

The Sentinel anomaly detection management system was the only requirement for which user testing revealed a dominant alternative. Although the time to find and manage sentinels with a sentinel list was about the same as with a folder/file system, users overwhelmingly chose the folder/file system as the alternative of choice due to the familiarity with this type of resource management system.

While there were no large, definitive choices to make for the sentinel and alert home pages, valuable user feedback suggested that any analysis page in the system follow three basic rules:

- 1) The page should have some form of graphical representation of the key information.
- 2) Where applicable, relative crime information should be displayed to the user so that he or she can compare crime levels to each other or over time.
- 3) Although more analysis options are generally better, at a certain point, too many options discourage users from using the analysis page.

C. Results

As suggested by user testing, the Sentinel system was created with the ability for users to create their own sentinels and subsequently manage them in a folder/file system. To compromise between the two approaches to the “create sentinel” form, both a short and advanced form were created to allow the user to choose between time to fill out the form and number of available filtering parameters. Similarly, a default sensitivity modifier (as determined by tradeoff

preferences of tested users) is set for each sentinel but this value can be changed by the user if he or she likes.

The design of both the sentinel and alert home page was heavily influenced by the advice of users. Both pages feature key analysis tools and information while not overwhelming the user. The sentinel home page consists of four main modules:

- 1) Sentinel Meta-Data: List of key characteristics of the particular sentinel including name, description, chosen filters, and time frame.
- 2) Alert List: List of all alerts that have been detected by the system in the past for this particular sentinel.
- 3) Data Summary: Display of the incident count for the current week, last week, one month ago, and one year ago.
- 4) Weekly Crime Graph: Line graph, as shown in Figure 2, depicting the predicted value vs. actual number of incidents for the most recent time period.

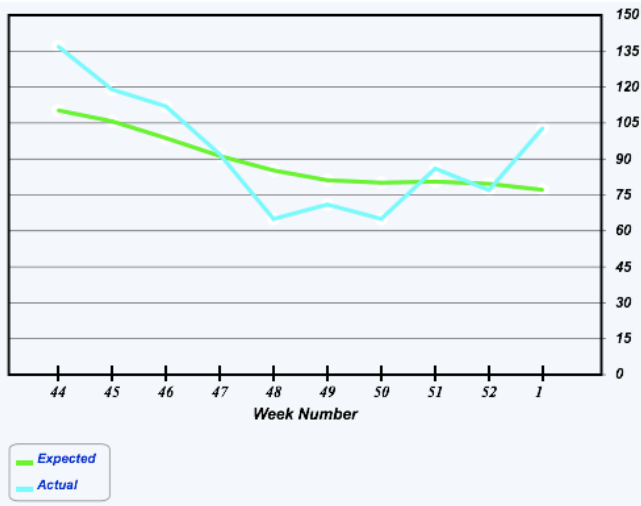


Fig. 2. Graph of Predicted vs. Actual Crime Incident Counts - Charlottesville 10/29/05 - 1/7/06 [16]

The alert page is very similar in that it displays important alert meta-data as well as a line graph comparing the predicted value vs. actual number of incidents for the most recent set of weeks leading up to the alert. This graph differs only slightly from the sentinel graph in that its timeframe does not get updated over time (so that this graph and specific timeline can be saved for referencing later) and weeks that are flagged as anomalies are in a different color to serve as a visual indicator of which particular weeks were involved in the anomaly alert. The data summary section differs from the sentinel page in that it shows which of the three control tests were not passed by this week's incident count. Finally, the alert list on the sentinel page is replaced by a brief overview of the sentinel that this alert belongs to.

V. SYSTEM TESTING

As with its design, testing of the system can be broken up into two parts: the anomaly detection system and the

user interface. While it was beyond the scope of this design project to extensively test the accuracy of the theoretical model, the program running the algorithm needed to be tested for flexibility and effectiveness. A test data set of all possible date filtering options and date specification peculiarities was formulated and entered into the program as inputs. The system outputted the correct number of incidents per week, proving that the system is flexible enough to handle the necessary data parameters. To test the effectiveness of the system, anomalies were manually inserted into a normal data set that was subsequently monitored by the algorithm. Each of these anomalies was found successfully by the program.

Two types of user testing were employed to test the effectiveness of the user interface. One test simply asked users to perform a series of simple tasks that initiated them with the various components of the Sentinel system. They were then asked to fill out a survey saying whether they liked or disliked these components. Some of the responses to these questions were on the Likert scale and some were free response. Overall, feedback from these users, a mixed group of crime analysts and test users, was very positive. Aside from a few minor edits, the only common critique of the interface was that some of the key features of various pages could be more salient. The ease of use of the interface as a whole and the extensive analysis tools were two areas that users really liked.

The other user interface test was used to validate the fact that this tool accomplishes its goal of improving the detection rate of anomalies in data sets and reducing the time overhead associated with the process. Users were given several tasks associated with detecting anomalies for a crime scenario and then asked to perform these tasks using both the current query and control chart approach in WebCAT and the new anomaly detection approach in Sentinel.

For the WebCAT method, two situations were considered because some police agencies analyze their data weekly and some monthly. In both cases, a user was asked to create and analyze a query. The time it took to do this task was then multiplied by either 52 or 12 depending on the situation. This gave the amount of analysis time required for 1 specific crime level per year. The equivalent time in Sentinel is equal to the time it takes to set up the sentinel plus the analysis time required to view alerts when they arise. The average number of weeks delayed in receiving alerts for each anomaly was also included in addition to the time metric.

According to Hansen's research, five of the fifty-two weeks in 2005 were legitimate anomalies for robberies in Charlottesville [15]. The ZMP algorithm found all 5 anomalies though produced 5 additional false alarm alerts. Thus, analysts using the Sentinel system would be alerted 10 times and would need to analyze each of those situations further. A summary of the test results for the three scenarios can be seen in Table I

The results from testing are fairly clear. Using the Sentinel system dramatically reduced analysis time as compared to both WebCAT situations, especially the weekly WebCAT scenario. Although the time difference between the Sentinel

TABLE I

DETECTING ANOMALIES IN WebCAT VS. SENTINEL DURING 1 YEAR

Scenario	Form Creation (min)	Analysis (min)	Avg. Detection Delay (weeks)
WebCAT (weekly)	52	156	0
WebCAT (monthly)	12	36	2
Sentinel	1	30	0

and monthly WebCAT scenario was somewhat small, a WebCAT user on average detected an anomaly two weeks late. Furthermore, the anomalies in this test were fairly easy to detect, quite unlike most situations a user will encounter. Thus, this test gives a minimum difference level between the two systems since in reality, the user will not be able to always identify anomalies that are not as obvious.

VI. CONCLUSIONS AND FUTURE WORKS

A. Conclusions

This paper has described the design and implementation of the Sentinel system, an automated anomaly detection tool that has been integrated into the WebCAT system. The goal of the tool was to make crime analysis more effective by increasing the detection rate of anomalies in data sets and reducing the unnecessary time overhead associated with this process. Although more extensive testing remains, initial user testing has shown that the Sentinel system can in fact accomplish these two objectives. In addition, the testing of the back-end anomaly detection program has shown it to be a flexible, yet robust environment for the ZMP anomaly detection algorithm. Although this tool cannot simply replace existing criminal analysis techniques, it can be viewed as an important and progressive development for this field.

B. Future Works

Although the ZMP algorithm works well in predicting weekly criminal incident counts, room for improvement exists. Future work on this project should entail further analysis of the fundamental nature of crime and more extensive testing of other potential models for use in this system.

In addition, several users suggested that this tool could be very helpful as an administrative monitoring tool for how well a particular police agency is controlling certain crime levels over time. Therefore, this idea of combining user specified goals with the current anomaly detection thresholds should be explored.

VII. ACKNOWLEDGMENTS

The authors gratefully acknowledge Butch Johnstone of the Virginia Department of Criminal Justice Services, Mike Lawton of DaPro Systems, and Detective Janice Coles of the University of Virginia Police Department for their invaluable aid in designing this system.

REFERENCES

- [1] C. Almazlinos, D. Bowman, R. Eagan, T. Kulinski, D. Nguyen, D. Brown, J. Conklin, and P. Hansen, "Webcat: The development, performance analysis, and deployment of a web-based crime analysis toolkit," *Systems and Information Engineering Design Symposium*, 2006.
- [2] D. Brown, C. Pittard, and A. Spillane, "Asset: a simulation test bed for evaluating data association algorithms." *Computers and Operations Research*, vol. 19, no. 6, pp. 479–493, 1992.
- [3] C. Hawkins, J. Pittman, F. Prats, W. Wheeler, D. Brown, J. Dalton, and B. Johnstone, "Webcat: the design and implementation of the web-based crime analysis toolkit," *Systems and Information Engineering Design Symposium*, pp. 233–239, 2003.
- [4] T. Bynum, *Using Analysis for Problem Solving: A Guidebook for Law Enforcement*. US Department of Justice, Office of Community Oriented Policing Services, 2001.
- [5] T. Rich, "The use of computerized mapping in crime control and prevention programs," *Reading, D.C: National Institute of Justice*, 1995.
- [6] A. K. Ghosh and A. Schwartzbard, "A study in using neural networks for anomaly and misuse detection," *Proceedings of the 8th USENIX Security Symposium*, pp. 23–26, 1999.
- [7] J. Anderson, "Computer security threat monitoring and surveillance," Anderson Co, Tech. Rep., 1980.
- [8] R. Kemmerer and G. Vigna, "Intrusion detection: a brief history and overview," *Computer*, vol. 35, pp. 27–30, 2002.
- [9] B. Simmons, "Personal interview of capt. purcell mccue," 1996.
- [10] R. Boba, *Crime analysis and crime mapping*. Sage Publications, 2005.
- [11] Coplink. [Online]. Available: www.coplink.com
- [12] (2007) National archive of criminal justice data. [Online]. Available: <http://www.icpsr.umich.edu/NACJD/index.html>
- [13] Automated tactile analysis of crime. [Online]. Available: <http://www.bairsoftware.com/>
- [14] D. Osgood, "Poisson-based regression analysis of aggregate crime rates," *Journal of Quantitative Criminology*, vol. 16, no. 1, pp. 21–43, 2000.
- [15] P. Hansen, "Automated monitoring of crime patterns using non-homogeneous zero modified poisson models," Master's thesis, University of Virginia, 2006.
- [16] Webcat - web based criminal analysis toolkit. [Online]. Available: webcat.sys.virginia.edu