2009 Second International Conferences on Advances in Computer-Human Interactions

Ground Truth Verification Tool (GTVT) for Video Surveillance Systems

Amol Ambardekar, Mircea Nicolescu, and Sergiu Dascalu
Department of Computer Science and Engineering
University of Nevada, Reno
Reno, USA
{ambardek, mircea, dascalus}@cse.unr.edu

Abstract— **As cameras and storage devices have become cheaper, the number of video surveillance systems has also increased. Video surveillance was (and mostly is) done by human operators on a need-to-know basis. The advent of new algorithms from the computer vision community, and increased computational power offered by new CPUs have shown a strong possibility of automating this task. Different approaches have been proposed by computer scientists to solve the difficult problem of content recognition from video data. They use many different videos to prove their usefulness and accuracy. A careful comparison and evaluation needs to be done to find the most suitable method under given conditions. To compare the results given by video surveillance applications, the ground truth needs to be established. In the case of computer vision, the ground truth needs to be provided by humans, making it one of the most time-consuming tasks in the evaluation process. This paper presents a tool (GTVT) that allows the user to establish the ground truth for a given video. GTVT presents a user-friendly interface to perform the cumbersome task of ground truth establishment and verification.**

Keywords- computer vision, video surveillance, ground truth verification, human computer interaction

I. INTRODUCTION

The last decade saw a great increase in the overall processing power, amount of memory and storage capacity of computers. The 90's also saw a great decrease in digital camera prices. All these advances made many computer vision tasks possible for real-world applications. Image processing is generally considered a low-level analysis tool for computer vision techniques, although it requires substantial computation and memory resources. Cheaper digital camera prices and cheap data storage devices made video surveillance more affordable. We can see video surveillance systems installed in supermarkets, banks, airports, casinos and on traffic lights. Most of the video recorded by all these cameras is watched online by a person to find abnormalities, or just recorded for future use. Advances in computer vision techniques have showed the promise of changing the current situation. Video surveillance applications, developed using computer vision techniques, are interested in the real-time observation of humans or vehicles in some environment (indoor, outdoor, or aerial), leading to a description of the activities of the objects within the environment. A complete video surveillance system typically consists of foreground segmentation, object detection, object tracking, human or object analysis, and activity analysis. There are different approaches suggested in the literature for video surveillance [1-5].

Different video surveillance methods have presented a dilemma for an end user or developer: which is the best technique to employ in a given situation? New techniques generally try to alleviate the errors made by previous ones and present their work on a set of video sequences. The computer vision community lacks a standard database of videos that can be used for the evaluation of these approaches. Such databases are common in the machine learning and object recognition communities [6]. Even if one finds a set of videos relevant for a particular scenario, the ground truth is not readily available. The reason behind this is the daunting amount of time required to generate ground truth for a particular set of videos. The lack of standardization among different ground truth authoring techniques is another cause. The ground truth needs to be established by humans, as all these computer vision techniques are ultimately trying to imitate human vision. A typical video sequence of five minutes captured at 30 fps has a total of 9000 frames. Therefore, a tool that can help in this process is an essential requirement. The tool also needs to standardize the process of ground truth creation and help the user to finish the task in a reasonable amount of time.

A number of semi-automatic tools are available to speed up the process of ground truth generation. Video Performance Evaluation Resource (ViPER), developed by Doermann *et al.* provides a software interface that could be used to visualize video analysis results and metrics for evaluation [7, 8, 9]. The framework compares the output of the algorithm with the ground truth and measures the differences according to objective metrics. They apply this methodology to recently proposed segmentation algorithms and describe their performance. These methods were evaluated in order to assess how well they can detect moving regions in an outdoor scene in fixed-camera situations [8]. The interface was developed in Java and is publicly available for download. Jaynes *et al.* [10] developed an Open Development Environment for Evaluation of Video Surveillance Systems (ODViS). The system is different from ViPER in that it offers an application programming interface (API) and also supports the integration of new surveillance modules into the system. Once integrated, ODViS provides a

978-0-7695-3529-6/09 $25.00 © 2009 IEEE
DOI 10.1109/ACHI.2009.17

number of software functions and tools to visualize the behavior of the video surveillance system.

Even though both systems give enough functionality, they are not especially user-friendly. There is practically no information available about the usability concerns of either system. Our proposed approach is an effort towards developing a user-friendly tool that can reduce the overall efforts in establishing the ground truth for videos. GTVT concentrates on object detection and classification, rather than segmentation. The user initially creates the information file by processing a video sequence using one of the many surveillance applications. The information file contains the information (the class of the object) about the objects detected in each frame. The user starts GTVT and opens the video sequence file and the information file. The user also selects the number of types (classes) of the objects that can appear in the video. GTVT then presents the user with the detected objects (bounding boxes) in each frame and their respective classes found by the video surveillance application (given classes). Then, it allows the user to select the bounding box and choose the actual class (ground truth) of the object. If the same object is going to appear in the subsequent frames, the user can use GTVT's tracking algorithm to automatically generate the ground truth. This information is aggregated to calculate the accuracy of the video surveillance system (i.e., object recognition/categorization algorithm) on a particular video sequence. It also creates the ground truth video and the information file for future reference.

The remainder of the paper is organized as follows: Sections 2 and 3 present requirement specifications and use case modeling respectively, describing software engineering techniques used in developing GTVT. Section 4 presents the results. Section 5 gives possible directions of future developments. Finally, Section 6 concludes by discussing advantages and limitations of GTVT.

II. Requirement Specifications

Requirements specifications detail tasks that determine the needs or conditions to meet for a new product or revision of a previously developed product, taking into account the possibly conflicting requirements of the various scenarios. Requirements specification is critical to the success of a project. Throughout the tool development process we have followed a human-computer interaction paradigm so that the final product is user-friendly. The rest of this section explains user requirements, system requirements, functional and non-functional requirements.

A. System and User Requirements

This application is designed for creating ground truth video and to analyze the performance of different video surveillance algorithms. It is recommended that the user have some background of computer vision. However, we do not assume that the user has such background. System requirements presume Windows XP operating system with .Net framework 1.0 or higher installed.

B. Functional Requirements

In functional requirements, we consider what the application is supposed to do at a certain point of time or what action the application needs to take after user input. These requirements directly address the user's expectation from the system. We have divided these requirement features into three categories. Level 1 requirements include basic features that must be included in the system. Level 2 requirements include features or functionality which will make the system complete and Level 3 requirements are extra and future functionality that can be included in future versions of the application. Level 3 requirements are those requirements without which the system can work, but if they are included, the system will perform the desired tasks more efficiently. Table I depicts the functional requirements we considered during the GTVT design process.

C. Non-functional Requirements

Non-functional requirements are those not directly related to application functionality or user experience. They typically include constraints placed on the system such as performance constraints, technology constraints, and development constraints. Table II lists the non-functional requirements for our software tool.

TABLE I. Functional Reuirements

R01 [1]	GTVT should be able to open video files.
R02 [1]	GTVT should be able to open information file (text file) that contains bounding box and classification information for each frame.
R03 [1]	GTVT should be able to play, pause video file.
R04 [1]	GTVT should be able to save ground truth information file.
R05 [1]	GTVT should be able to save ground truth video.
R06 [2]	GTVT should allow user to configure classes that can appear in the video.
R07 [2]	GTVT should allow the user to select a particular bounding box. When the user selects a particular bounding box, the application should retrieve relevant classification the information from information file. It then should allow the user to choose if the classification was correct or wrong. If it was wrong, it should allow user to enter the correct classification.
R08 [2]	GTVT should include a keyframe concept where only keyframes are modified by the user and the rest of the work is done by the in-built tracking algorithm (e.g. blob tracking [11], Mean-Shift [12]).
R08 [3]	GTVT should include support for different video file formats. GTVT should also include support for different video resolutions (currently 320X240 videos supported).
R10 [3]	GTVT should include support for different user-defined information file formats.
R11 [3]	GTVT should include the concept of "project" such that information file and video can be opened or saved together. Therefore, project file format needs to be created.

TABLE II. Non-functional Requiremnts

T01	GTVT should be developed using Visual Studio .Net C# [13].
T02	GTVT needs .Net framework installed on the computer to run.
T03	GTVT should use consistent design language (e.g., same font).

III. USE CASE MODELING AND SCENARIOS

A. *Use cases*

A use case describes how the system should respond under various conditions. It analyses how the system should behave to a request from one of the users to deliver a specific goal. This is primarily done in the form of scenarios that describe sequences of interaction steps. Some of the functional requirements can be established using use case modeling. Use cases can serve as a basis for estimating, scheduling, and validating development efforts.

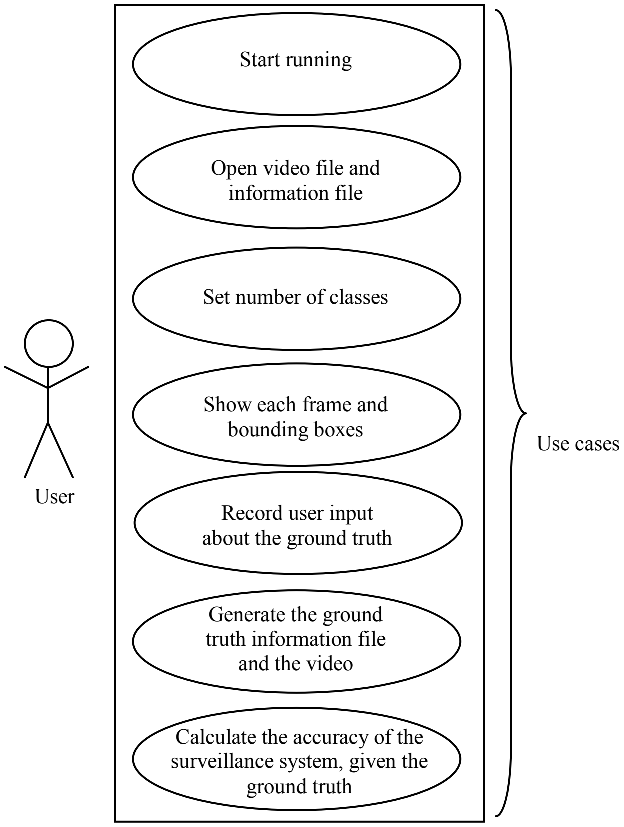

Start running
Open video file and information file
Set number of classes
Show each frame and bounding boxes
Record user input about the ground truth
Generate the ground truth information file and the video
Calculate the accuracy of the surveillance system, given the ground truth
User
Use cases

Figure 1. Use case diagram of GTVT.

Figure 1 displays how the actor interacts with the GTVT system as well as the larger scale functionality of the backend. In order to describe the functionality, detailed use case descriptions are presented as follows:

UC01 *Start running.* User opens GTVT from his/her set of applications.

UC02 *Open video file and information file*. The precondition for this use case is that the GTVT is running and the user pressed the *OPEN* button. When the user presses the *OPEN* button, user is presented with open dialog and is asked to choose the video file and the information file. The information file contains information about the bounding boxes and class information found by the video surveillance algorithm. Class information means the types of objects classified by the video surveillance algorithm, e.g., a traffic surveillance application may classify the detected objects as cars, SUVs, trucks and buses. If opening both files is successful, the information pane is updated accordingly.

UC03 *Set number of classes.* In this use case, the user sets the number of classes that are possible in the given video. For the above traffic surveillance example, it is four classes (class 0 to 3). Class -1 is implicit and denotes any object that was detected but was not classified as non-vehicles (e.g. human or bike in the above example).

UC04 *Show each frame and bounding boxes.* The precondition for this use case is that the video and information files are opened successfully and the number of classes is set. When the user plays the video or uses the scrollbar to go to a particular frame number, GTVT extracts the bounding box information from the information file for the corresponding frame. The bounding boxes are drawn in red.

UC05 *Record the user input about the ground truth.* The precondition for this use case is that GTVT presents the user with bounding boxes for each frame and the user selects one of these bounding boxes. When the user selects a bounding box (bounding box turns blue), the list box is populated with the available number of classes and presses the *START TRACKING* button.

TABLE III. RUNNING GROUND TRUTH VERIFICATION TOOL (GTVT)

Actors: Ground truth verification tool user
Pre Conditions: 1. The PC should be running with Windows XP or later. 2. .Net framework should be installed. 3. Ground truth verification tool (GTVT) should be installed.
Flow of Events: 1. User opens the GTVT application from his applications. 2. User selects the *OPEN* button, the open dialog appears. 3. User selects video file and information file. 4. User uses tools menu and selects preferences. 5. It presents a popup window that lets user enter the number of classes. 6. When user changes the progress bar value, the corresponding frame and its bounding box information from the information (info.) file is retrieved. 7. All the bounding boxes are drawn and current frame number is updated. 8. When user selects a bounding box, the user is presented with the class that was given by video surveillance system. 9. User then keeps it or changes it to a new one if it is wrong and presses *START TRACKING* button. 10. Then, user can continue to play the video until the user catches problem with the tracking. GTVT tracking algorithm is responsible for filling the ground truth values for the consecutive appearances of the given object. 11. User can go back and forth while playing the video and change the ground truth. If there is any change, changes are recorded in new information file. 12. User repeats steps 4 to 11, until all the ground truth information is entered. 13. When user presses *SAVE* button, the ground truth video file and ground truth information file are saved. 14. When user presses the *PERFORMANCE* button, the application calculates per class and overall accuracy and shows it in a new pop-up window.
Post Conditions: 1. The ground truth video and information files are saved. 2. Per class and overall accuracy of the video surveillance system is estimated.

UC06 *Generate the ground truth information file and the video.* The precondition for this use case is that the last use case is performed for all the bounding boxes of all the frames in the video. When the user presses the *SAVE* button, the user is presented with a save dialog box and is asked to save the ground truth video file and the ground truth information file.

UC07 *Calculate the accuracy of the surveillance system, given the ground truth.* The precondition for this use case is that the user has finished entering the ground truth for all the bounding boxes. When the user selects the PERFORMANCE option from the menu, GTVT presents the user with the confusion matrix and the overall percentage accuracy.

B. *Scenarios*

Table III depicts the primary scenario presented by GTVT. For the given scenario, we discuss pre conditions, flow of events and post conditions.

IV. Results

We have developed the GTVT aiming at facilitating the ground truth generation for object classification in video surveillance videos. It satisfies all the Level 1 and Level 2 requirements and satisfies the primary scenario. Level 3 requirements are extra features and do not undermine the basic operation of GTVT.

We tested GTVT using the output generated after processing a sample video using the traffic surveillance system discussed in [14]. The traffic surveillance system can classify vehicles in four classes. They are cars (class 0), SUVs (class 1), trucks (class 2), and buses (class 3). The traffic surveillance system sometimes detects objects (e.g. bike) that can not be assigned to any of the four classes. Such objects are classified as non-vehicles (class -1). To understand the difficulty of the task even for the sample video we tested, one can calculate the number of frames for which the user needs to provide the ground truth. We chose a simple traffic video that was shot at 30 fps and with the duration of only 10 seconds. Even if there is only one object that appears in each frame, the user needs to provide ground truth for all 300 frames. GTVT automates this task and the user needs to input the ground truth only when the object appears the first time and when the tracking algorithm fails. Therefore, for the sample video of 10 seconds, the user entered the ground truth only twice. This greatly reduced the amount of time required for generating the ground truth. Automating the task also reduces the possibility of manual error.

We used simple blob tracking similar to the method discussed in [11] to track objects. For the purpose of ground truth generation, it proves to be useful. It has two great advantages of being fast and reliable when the image resolution is low. We also tried the Mean-Shift tracker [12]; however, it fails more often, requiring increased user intervention. This problem is more evident in the low-resolution video sequences where object sizes are small.

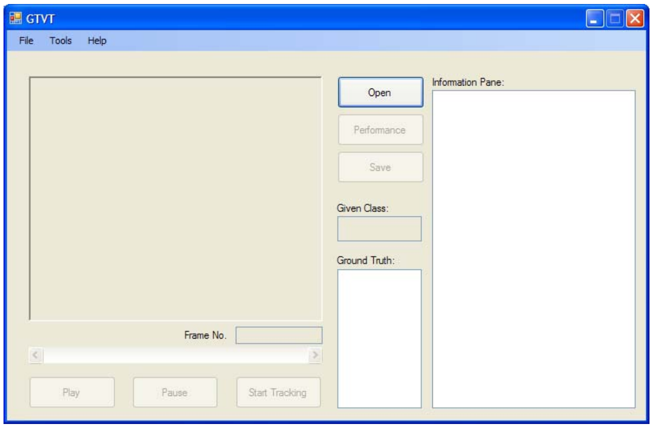
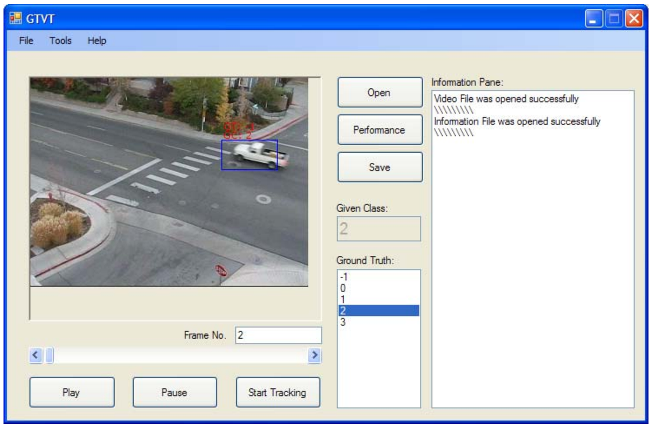


Figure 2. GTVT when running it for the first time.



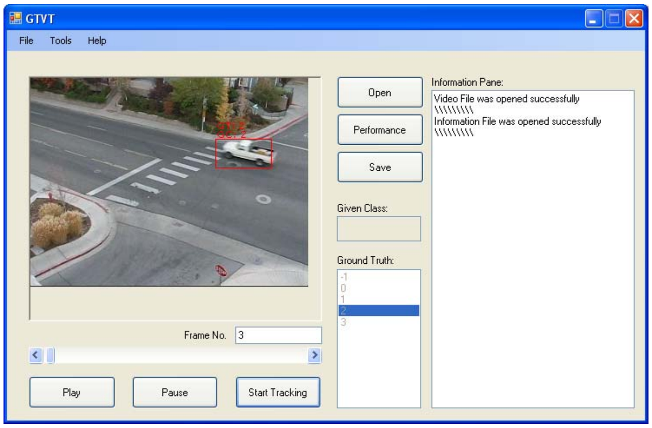
Video File was opened successfully
\\\\\\\\\\
Information File was opened successfully
\\\\\\\\\\

Given Class: 2

Ground Truth:
-1
0
1
2
3

Frame No. 2

Figure 3. After selecting a bounding box, GTVT shows that the class for the given bounding box is 2 and lets user select the ground truth. When the bounding box is selected, it turns blue.

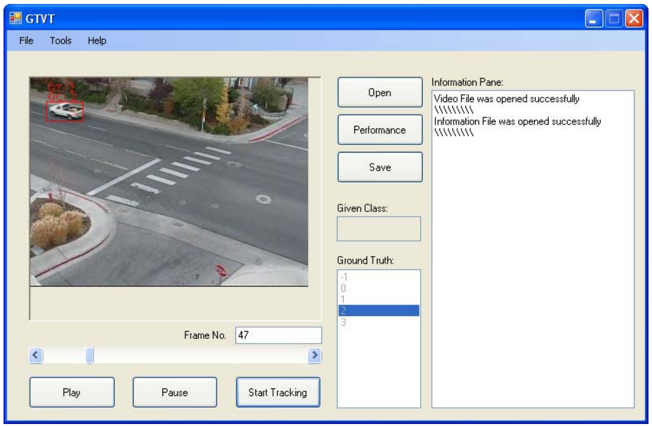


Video File was opened successfully
\\\\\\\\\\
Information File was opened successfully
\\\\\\\\\\

Ground Truth:
-1
0
1
2
3

Frame No. 3

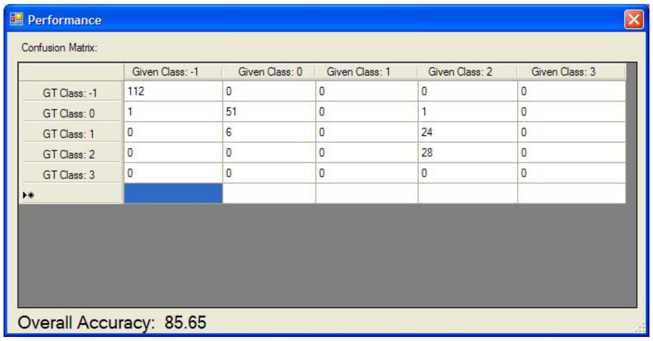
Figure 4. The successful tracking of the object in the next frame. The ground truth values are copied from the previous frame and hence no user input is required. The color of the bounding box is red in this case.



Video File was opened successfully
\\\\\\\\\\
Information File was opened successfully
\\\\\\\\\\

Frame No. 47

Figure 5. The successful tracking of the object after 47 frames.



Confusion Matrix:

	Given Class: -1	Given Class: 0	Given Class: 1	Given Class: 2	Given Class: 3
GT Class: -1	112	0	0	0	0
GT Class: 0	1	51	0	1	0
GT Class: 1	0	6	0	24	0
GT Class: 2	0	0	0	28	0
GT Class: 3	0	0	0	0	0

Overall Accuracy: 85.65

Figure 6. Performance window showing the confusion matrix and the overall accuracy of the algorithm.

To present the user interaction of the GTVT, the following screenshots are included. Figure 2 is a screenshot of the initial screen, whereas Figure 3 shows the screenshot where the user has selected a bounding box (therefore, the color of the box is blue indicating the user selection) and started tracking. It also shows the information pane stating that the video and information files were successfully opened. The list box presents the choices of the classes, the user can select (between -1 and 3). The given class (class of the object determined by the video surveillance application) is 2 in this example, and the user has also selected 2 as the ground truth in the list box. Figure 4 shows the successful tracking as GTVT selects the correct ground truth for the bounding box without the user input. The red box represents the detected object in the current frame. If the bounding box is not selected by the user, it is red. To change the ground truth, the user needs to select the bounding box which in turn enables the ground truth list box. Figure 5 shows the same object being tracked successfully. Figure 6 shows the performance window displaying the confusion matrix and the overall accuracy of the video surveillance application used.

V. Future Work

For a typical video surveillance application, the failure can be of two types: total or partial failure in detecting the objects or failure in object recognition for the detected objects. The current version of GTVT focuses on establishing the ground truth for recognition. It would be interesting to see the detection ground truth included along with classification/recognition ground truth. However, the current version of GTVT does not allow the user to select ground truth for the objects that were not detected by the video surveillance application under consideration. A Mean-Shift tracker or feature tracker will have to be included to achieve this functionality so that the user can initialize a track for the objects that were not detected by the video surveillance application.

VI. Conclusions

The operation of GTVT can be summarized as follows. The user initiates the ground truth class for a particular bounding box. The corresponding bounding box will be tracked in consecutive frames using a tracking algorithm. GTVT copies the ground truth information from the previous frame so that the user does not have to repeat the same task again. If tracking is successful and the user is satisfied with the ground truth for a particular frame, user intervention is not required. Thus, GTVT greatly reduces the overall time required for ground truth generation by making the process semi-automatic.

This tool shows how the HCI [15] paradigm can be used to create a user-friendly application that can substantially reduce the user’s efforts. In this paper, we presented a Ground Truth Verification Tool (GTVT) and demonstrated that it can greatly reduce the user’s involvement in ground truth generation. GTVT can also be used as a verification and accuracy measurement tool for several video surveillance applications. We plan to standardize the ground truth establishment technique and the way the ground truth is saved. Any prospective user of GTVT needs to make only a minor change in their video surveillance system to create the information files. Our approach suggests a good alternative to the current methods of ground truth establishment (e.g. manual or tools like ViPER). As GTVT is designed by placing the user into focus, the resulting product is more user-friendly compared to existing similar applications.

Acknowledgment

This work was supported by the Office of Naval Research award N00014-06-1-0611.

References

[1] C. Wern, A. Azarbayejani, T. Darrel, and A. Petland, “Pfinder: real-time tracking of human body,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 7, pp. 780–785, 1997.

[2] A. Bobick, J. Davis, S. Intille, F. Baird, L. Cambell, Y. Irinov, C. Pinhanez, and A. Wilson., "Kidsroom: Action Recognition in an Interactive Story Environment," *M.I.T. Perceptual Computing*, Technical Report 398, 1996.

[3] A. Azarbayjani, C. Wren, and A. Pentland, "Real-Time 3D Tracking of the Human Body," *Proceedings of IMAGE'COM*, 1996.

[4] A. J. Lipton, J. I. Clark, P. Brewe, P. L. Venetianer, and A. J. Chosak, "ObjectVideo Forensics: Activity-Based Video Indexing and Retrieval for Physical Security Applications," *IEEE Workshop on Intelligent Distributed Surveillance Systems*, pp. 56-60, 2004.

[5] D. Duque, H. Santos, P. Cortez, "Prediction of Abnormal Behaviors for Intelligent Video Surveillance Systems Computational

Intelligence and Data Mining," *IEEE Symposium on Computational Intelligence and Data Mining*, pp. 362-367, 2007.

[6] UCI Machine Learning Repository, "http://mlearn.ics.uci.edu/MLRepository.html," last accessed on 05/09/08.

[7] D. Doermann, and D. Mihalcik, "Tools and Techniques for Video Performances Evaluation," *International Conference on Pattern Recognition*, pp. 167-170, 2000.

[8] V.Y. Mariano, J. Min, J.-H. Park, R. Kasturi, D. Mihalcik, D. Doermann, and T. Drayer, "Performance Evaluation of Object Detection Algorithms," *International Conference on Pattern Recognition*, 2002, pp. 965-969.

[9] ViPER toolkit, "http://viper-toolkit.sourceforge.net/", last accessed on 03/22/08.

[10] ODViS toolkit, "http://www.metaverselab.org/software/odvis/", last accessed on 03/22/08.

[11] S. Gupte, O. Masoud, R. F. K. Martin, and N. P. Papanikolopoulos, "Detection and Classification of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 3, no. 1, pp. 37-47, 2002.

[12] D. Comaniciu and P. Meer, "Mean Shift: A robust approach toward feature space analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, 2002.

[13] Microsoft Visual C#, "http://msdn2.microsoft.com/en-us/vcsharp/default.aspx," last accessed on 04/12/08.

[14] A. Ambardekar, M. Nicolescu, and G. Bebis, "Efficient Vehicle Tracking and Classification for an Automated Traffic Surveillance System," *Proceedings of the International Conference on Signal and Image Processing*, Kailua-Kona, Hawaii, pp. 220-225, 2008.

[15] D. Benyon, P. Turner, and S. Turner, "Designing interactive systems," Addison-Wesley, 2005.