Ground Truth Annotation of Traffic Video Data

Jose M. Mossi Antonio Albiol
Alberto Albiol Javier Oliver

jmmossi@dcom.upv.es

ITeam. Universitat Politécnica de Valencia, Spain

ABSTRACT

This paper presents a software application to generate ground-truth data on video files from traffic surveillance cameras used for Intelligent Transportation Systems (IT systems). The computer vision system to be evaluated counts the number of vehicles that cross a line per time unit –intensity-, the average speed and the occupancy. The main goal of the visual interface presented in this paper is to be easy to use without the requirement of any specific hardware. It is based on a standard laptop or desktop computer and a Jog shuttle wheel. The setup is efficient and comfortable because one hand of the annotating person is almost all the time on the space key of the keyboard while the other hand is on the jog shuttle wheel. The mean time required to annotate a video file ranges from 1 to 5 times its duration (per lane) depending on the content. Compared to general purpose annotation tool a time factor gain of about 7 times is achieved.

Categories and Subject Descriptors

H.5 [**Information Interfaces and Presentation**]: *Multimedia Information systems*

General Terms

Algorithms, Security, Human Factors, Verification.

Keywords

Traffic, ground truth, vehicle, video, intelligent transportation systems.

1. INTRODUCTION

Intelligent Transportation Systems are used to manage urban and road traffic. In the urban context, the timing of traffic lights has been traditionally planned according to some parameters obtained from inductive loops installed in the street asphalt. However, recent advances in computer vision allow to obtain them from the video cameras commonly installed at street poles[1]. The most common parameters that are measured are **intensity**, also referred to in the literature as density (number of vehicles to cross a line per time unit), **occupancy** (average fraction of time a vehicle is over a line or loop) and **mean speed** (mean of the speed of the vehicles crossing a line). These three magnitudes are periodically obtained and sent to the traffic control center with a predetermined cadence (usually a value between 30 seconds and 2 minutes).

To date, most computer vision traffic systems[6][23] have been developed and evaluated using independent datasets, which makes it difficult to objectively evaluate their performance and robustness as in other computer vision areas[2][11][18][8][22]. In all these areas, challenging datasets have also been catalysts for progress in the research.

A good dataset in the context of traffic analysis must be large enough to include a big number of vehicles and situations. Examples of different situations include traffic conditions –congestion, dense, fluid, etc.- weather conditions, scenario typologies such as avenues, tunnels, narrow two way streets, wide multilane boulevards, etc. This huge amount of data poses the problem of a daunting annotation task to generate ground-truth information; therefore good interfaces and frameworks are required to accomplish this goal.

Figure 1 shows a map of the city of Valencia (Spain) along with the city traffic camera network [17] and a sample view of some of them. Using these cameras, we have created our video database for evaluation of video traffic analysis algorithms (eValTraffic).

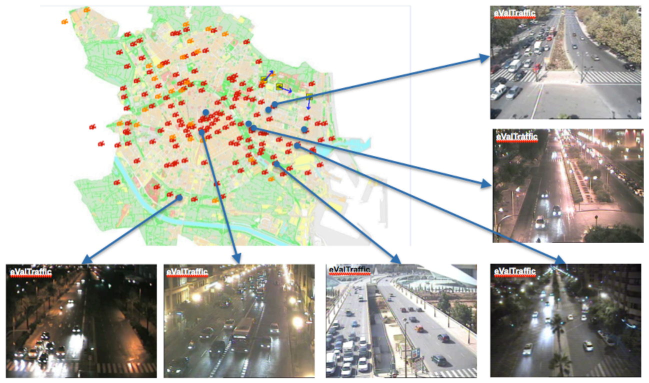


Figure 1. City map and some video from cameras

One common requirement in all the previous scenarios is to predefine a set of measurement lines where the parameters are going to be measured, like in Figure 2. The position of these lines emulates the location of inductive loops where measurements are traditionally taken.

In order to evaluate the results of computer vision traffic systems, the ground truth of the dataset needs to be established. This task is one of the most time-consuming in the evaluation process, so it is convenient to use helpful tools. In this specific scenario it is needed to determine, for each vehicle in the dataset, its pass over the line, its speed and its occupancy. Several ground truth annotation tools are available in the literature like [20] [5] [7][10]. They are powerful and general, but this paper proposes to take advantage of the specific scenario of traffic systems ground truth where, for counting a vehicle or estimating its speed, it is not necessary to segment it with a pixel level precision or a frame by frame tracking, such that the time required to annotate the videos is reduced up to 7 times.

Another aspect is that the annotation process can be speeded up with specific hardware and/or software to navigate the video in a faster way. Tools can be provided to help the human focus on the interesting frames where the action occurs and bypass the time intervals without vehicles. This paper proposes both improvements and presents a setup consisting in a standard computer or laptop, a very simple visual interface, a jog shuttle wheel, (see Figure 3), and the procedure to build, in an efficient way, the ground truth needed in computer vision based video traffic applications. This tool has been successfully used to annotate the eValTraffic dataset.

The organization of the paper is as follows. Related work is presented in section 2. Section 3 is dedicated to give an overall description of the system, while section 4 describes the details of the annotation process. A speed up software tool using motion detection is presented in section 5. In section 6, it is explained how to add a jog-shuttle wheel to be more efficient during the navigation of the video. Finally, sections 7 and 8 provide the results and conclusions.

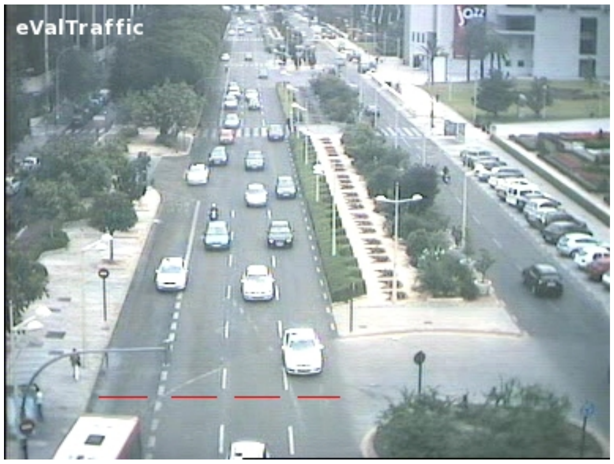


Figure 2. Setting the locations where to perform the measurements

2. RELATED WORK

The Video Performance Evaluation Resource (ViPER) [20] is a well known tool to label video and it can be taken as one of the baseline consolidated general tools for this kind of tasks. It is used in many works [9][12] to annotate or for performance evaluation and its XML output format has been adopted in many video databases such as ViSOR [19]. The ground truth generated using the ViPER sytem contains enough information to evaluate algorithms like, for example, detection, tracking, background subtraction or traffic systems at a pixel level precision, as in the works by Faro et al [9] and by Kasturi [12]. ViPER labels objects in each frame using rectangular bounding-boxes or arbitrary-shaped polygons. The user can navigate through the video with the play, backward, forward buttons or directly seek to a particular frame. Some drawbacks of the ViPER system are that every operation requires a mouse or keyboard action and the lack of efficient semiautomatic tools that speed up the process.

D'Orazio [7] and Serrano [16], among others, propose to aid the manual annotation incorporating automatic o semi-automating tools. For instance, in [7], the ground truth interface software is fed by the result of a detection or tracking algorithm and the user validates it. In case of failure, the annotation is corrected with ViPER. A different approach is presented in [12], where the user feedback obtained during the annotation process is used by the detection or tracking software to perform better in the following frames.

As video annotation of large databases requires a huge amount of work, other researchers are more focused in web cooperation interfaces. Volkmer [21] propose a web-based system called EVA, the IBM Efficient Video Annotation, which was used to annotate more than 80 hours of video in TRECVID. Several strategies for human-computer interaction and usability were tested, such as annotating exhaustively with one concept before proceeding to the next vs. annotating with several concepts simultaneously; use the mouse only vs. mouse and/or keyboard for navigation and annotation, display several images at the same time vs. one big image and a selectable number of thumbnails, etc. LabelMe [15] is another example of web-based tool for image annotation but it is more focused on still image than on video annotation.

Most papers deal with expensive annotations in terms of human employed time but have enough information to check the systems at pixel level precision or frame by frame object tracking. However in the case of traffic algorithms for measuring only the parameters indicated in the introduction, it is not necessary all that detailed information, it suffices with the instant when the vehicle reaches and departs from a measurement position to derive its speed and occupancy. The present paper exploits this fact in order to build a simpler interface to input the human annotations that results in an important reduction of the time needed to obtain the ground truth.

3. OVERALL DESCRIPTION

To annotate intensity, mean speed and occupancy of the vehicles crossing each line of measurement (see Figure 2) it is enough to annotate the following information:

- Time when a vehicle begins to cross the line.
- The speed at which each vehicle crosses it.
- Time when the vehicle finishes crossing the line to obtain the occupancy.

As it can be necessary to annotate several lanes in the same video, the first decision to be made is to choose between:

1. Navigate the video only once and annotate for each frame the information related to all the vehicles that are present at the measurement lines in that instant, or
2. Iteratively annotate the video lane by lane independently. At the end of the process, the information from all the lanes can be merged together in one file.

In the first case, an interface is needed to select the lane of interest every time that a vehicle reaches a measurement line. This is the approach used in VIPER-GT, where the mouse is used to select the objects of interest.

We used the second approach because psychophysiological studies on user's fatigue and reaction time [14] have shown that when a person is focused in just one simple task is more effective than with tasks that require to switch between actions. In this second case, the user is focused only on "*when*" the object in the video touches the line, instead of focusing on "*when*" and "*where*" as in the first approach.

In our user interface, a line is superimposed on the image and the user presses a key when a car reaches it. Then, the frame number is recorded (i.e. the time instant), next, the video is played back until the rear part of the vehicle reaches again the same line. At that moment the user presses the same key finishing the annotation for one car. This process is repeated for the following vehicles in the sequence. Notice that as only one lane is processed at a time, and the position of the line in each lane is predetermined and fixed, there is no need to point any location with the mouse, therefore no time is wasted moving the mouse or drawing a bounding box to indicate what car or cars and what position or positions are going to be annotated in each frame

The annotating interface has to provide the following features:

- Ability to navigate the video at both, high speed and high precision. A jog shuttle wheel is proposed for this task.
- Mark the time instants where things happen. A conventional keyboard is used for this purpose.
- Possibility to correct annotation errors. In the case that the operator makes a mistake and detects it, the system has to be able to let him undo previous actions.
- Provide feedback of the annotation process. As it will be explained later, the process of annotating each car requires several keystrokes. The user must be able to know easily at which state is the annotation of a vehicle at any time.

Our previous experience on video annotation has shown us that one key aspect of an efficient annotation is the ability to navigate video files at very different speeds. For instance, when there is no vehicle in the scene, fast forward navigation is usually required. The playback speed must be significantly reduced when a vehicle appears on the scene near the measurement line. Finally, precise annotation of the frame where the vehicle crosses the line requires a frame-by-frame advance. This objective can be easily achieved with a jog shuttle wheel as ones used for broadcast television edition and showed in Figures 3 and 4.

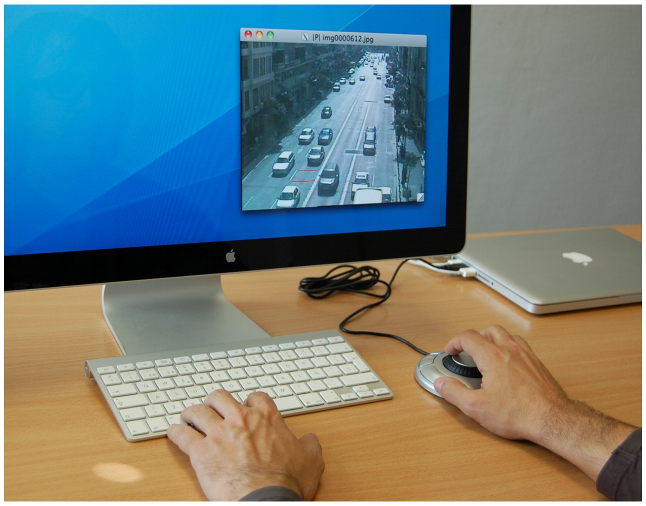


Figure 3. Setup of the ground truth annotation system

The jog shuttle wheel controls the frame-by-frame step forward o backward using the motion of the wheel. Compared with a simple keyboard interface, the jog wheel allows a very much faster and precise frame location.

The shuttle ring controls the playback speed. It has a rest position that corresponds to a pause of the video playback. The angular position of the shuttle ring sets the forward or backward playback speed, so it is very easy for the user to control the playback speed. Also, since the rest position of the shuttle ring pauses the video, it is very easy to pause the video at any instant just by releasing the shuttle.

In order to have a quick idea of which portions of the video have been annotated, the working window contains a bottom timeline.

The elements on the timeline can be seen in Figure 5.

A: Denotes the current position in the video.

B: Green vertical lines indicate the instants where vehicles have reached the auxiliary line (explained below in the paper).

C: Light orange indicates presence of a car on the main line.

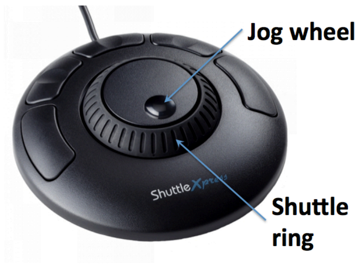


Figure 4. Jog-Shuttle Wheel used to navigate the video data

Once the annotation has finished, the ground truth data is saved in XML format because it is a well-known and human readable format. The XML format is also used in many other video applications and there exist many open source libraries to manage this kind of data [2][20]. After the last lane has been annotated, the system merges the information of all the lanes into a single file.

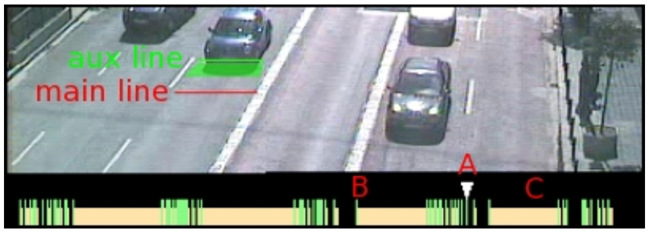


Figure 5. Timeline

4. DETAILED VEHICLE ANNOTATION PROCEDURE

4.1 Normal vehicle annotation

In order to annotate the speed for each vehicle, we propose to measure the time that a vehicle spends to cover a fixed distance. To do that, we set a second line on the screen. We call this line the "*auxiliary line"* as opposed to the measurement line mentioned in the introduction of the paper (the *"main line"*). The auxiliary line is parallel to the main line and it is positioned on the screen at a fixed distance *D* before the main line along the direction of the movement of the vehicle (it is crossed first by the vehicles). Since time is quantized by the frame rate, speed measurements will also be quantized. Finer speed granularity requires *D* to be larger. However, large values for *D* may be problematic if the vehicle's speed does not remain approximately constant while traversing this gap.

The separation D between the main and auxiliary lines can be easily obtained in real magnitude (meters) directly from the image if the scenario is properly calibrated and a homography that maps ground points to world coordinates is available.

The whole process of a vehicle annotation is illustrated in Figure 6. The first step is to navigate with the jog-shuttle until the frame in which the front edge of a vehicle reaches the auxiliary line. Then the annotation process of the car starts and involves the following actions:

- Press a key to mark the frame number corresponding to the vehicle reaching the auxiliary line. Once the key is pressed, a green rectangle touching the auxiliary line is displayed to acknowledge the action and to indicate the next action required by the user.
- Advance the video frame by frame until the front of the vehicle reaches the main line. Then, press a key to mark the frame number corresponding to the vehicle on the main line. After the keystroke, the green rectangle on the aux. line is changed by an orange one on the main line.
- Advance the video frame by frame until the tail of the vehicle leaves the main line. Then, press a key to mark the vehicle leaving the main line.

We have chosen the SPACE key because it is the largest on the keyboard. Since all annotations involve the same key, the human does not need to change the position of the finger and to waste time or concentration.

Notice each car annotation requires only 3 keystrokes. The user receives feedback about the current state of the annotation by a color code. Before the car annotation begins, the main and auxiliary lines are displayed in red (Figure 6a). After the first keystroke, that indicates that the front of the car has reached the auxiliary line, the color of the line changes from red to green (see Figure 6b). After the second keystroke, indicating that the vehicle's front has reached the main line, the auxiliary line is hidden and the main line becomes orange (see Figure 6c). Finally, after the third keystroke that triggers that the rear part of the vehicle is leaving the main line, both lines are returned to their original red color (Figure 6d).

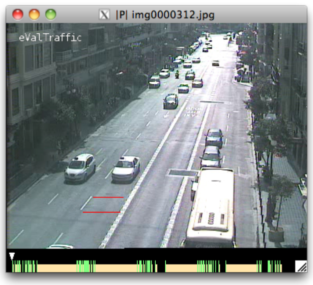
If errors are made during the annotation of one vehicle, they can be easily discarded by pressing the BACKSPACE key.

4.2 Double vehicle annotation

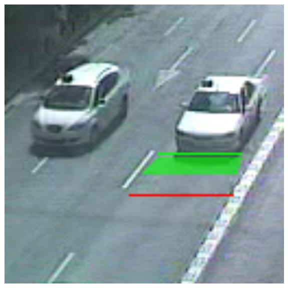
Sometimes the distance between two vehicles is very small. In this situation the second vehicle may reach the auxiliary line before the preceding one has completed to cross the main line.

So it is necessary to handle this situation where in some frames two vehicles are in the area of interest simultaneously. Two solutions are proposed. The first one is:

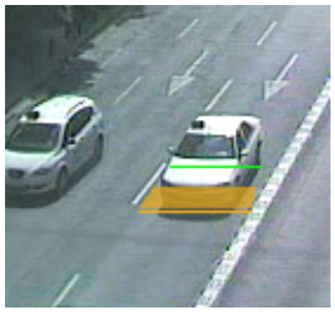
- To annotate completely the first vehicle as if the second vehicle didn't exist at all.
- Then, rewind the necessary frames until the second vehicle reaches the auxiliary line and proceed normally annotating the second one.



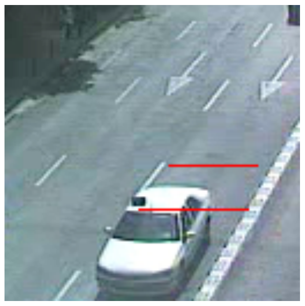
a) Vehicle approximating to the lines. Both lines shown in red.



b) User presses key for the first time to indicate that the vehicle has reached the aux line.



c) User presses key a second time to indicate that the vehicle has reached the main line.

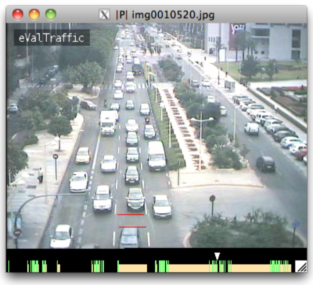


d) User presses key the third time to indicate that the vehicle has exceeded the main line and the annotation of the vehicle concludes, returning to the initial state.

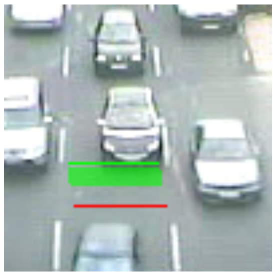
Figure 6. Sequence to annotate one vehicle.

In order to have a visual feedback that two vehicles are being simultaneously annotated, the area between the two lines is split into two halves; the left side will show the state of the first vehicle and the right side the second one. Additionally, a red line is drawn between both sides to recognize this situation more easily. Figure 7 shows an example of the process.

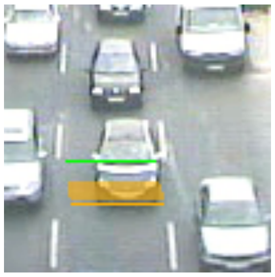
The second approach deals with double vehicle annotation without going backwards. When two vehicles are simultaneously in the area of interest, two different keys, one for each vehicle, are needed instead of the single key SPACE. We chose the numbers 1 and 2. The key 1 is used to indicate each event of the first vehicle as it was the key SPACE but each hit applies only for the first vehicle and the key 2 for each event of the second vehicle, similarly, as it was the key SPACE, but each hit applies only for the second one. When a third vehicle reaches the auxiliary line after the last event of the second vehicle, the normal procedure with the key SPACE is used again. If the second vehicle has not finished crossing, the key 1 is reused for the third one and continue working with the double vehicle annotation mode, i.e., using key 2 for the second one, and key 1 for the third one.



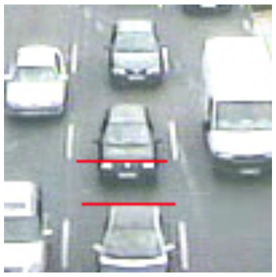
a) Vehicle approximating the lines.



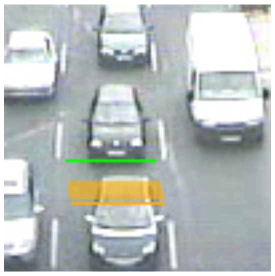
b) User presses space key for the first time to indicate that the (first) vehicle has reached the aux line.



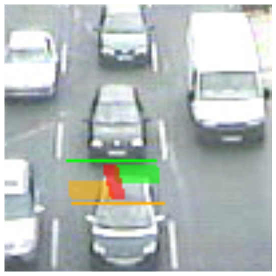
c) User presses space key a second time to indicate that the vehicle has reached the main line.



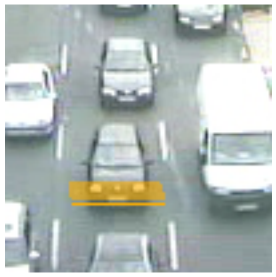
d) User presses space key the third time to indicate that the vehicle's tail has exceeded the main line and the annotation of the (first) vehicle has just finished. (frame 10568).



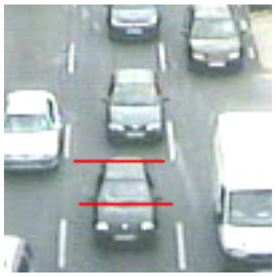
e) User rewinds from frame 10568 to 10560 to place the second vehicle at the aux line to prepare the annotation of the second car. The orange area corresponds to the first car.



f) User presses space key for the first time of the second vehicle. As two cars are marked in this frame the area is divided into two zones separated by a red line.



g) User goes forward until the (second) vehicle reaches the main line and then presses space key.



4.3 h) Finally, the space key is pressed to mark the rear part of the (second) vehicle on the main line.

Figure 7. Sequence to annotate two vehicles.

5. SPEEDING UP THE ANNOTATION PROCESS

In a traffic scenario there are two situations that happen frequently:

1) Time intervals where no vehicles pass by the lane
2) Time intervals where the vehicles are on the line and are stopped in front of the red light or in a traffic jam.

In both circumstances it is interesting to use an automatic fast forward mode that plays the video quickly and stops at a frame where motion is detected again in the zone between the lines. The first situation is caused because a new car appears in the scene and it reaches the auxiliary line, and the motion in the second situation is caused because the traffic light turns to green and the stopped vehicle restarts.

Notice that we implemented this functionality by fast-forward 16x playing until motion is detected again. This approach has the advantage compared to direct seeking that the user can see a potential failure if the automatic fast forward doesn't stop at the appropriate frame.

In order to implement this motion detection, a small rectangle around the auxiliary line is defined. This region is used as input to a background subtraction-foreground detection algorithm. We used the modules available in OpenCV [3][13] based on Mixture of Gaussians.

The motion detection has proved to be a valuable help to speed up the annotation process. The results section shows details of the time saved. Moreover, the impact of the motion detection errors is minimal:

- A false positive, that pauses the video, simply requires the user to call the motion detector again by pressing a key.
- A missed positive can be easily detected by the user since the video is displayed while fast-forwarding.

6. INTERFACING THE JOG-WHEEL

The jog shuttle wheel comes with a driver that allows the operative system of the computer to create interfaces with applications. The driver is configured to produce a computer response to every action on the jog shuttle. Typically, the computer response is to map jog shuttle actions to different keyboard keystrokes. The annotating applications must only take care of properly processing keyboard events. Table 1 shows some of the mappings that we have used as an example.

Table 1. Action association between jog wheel and application

Jog-Wheel Action	Keystroke	Application Action
Turn Jog Right	Down arrow	Forward one frame
Turn Jog Left	Up arrow	Backward one frame
Shuttle zone 0	A	None
Shuttle zone 1	B	Play Forward slow /5
Shutle zone 7	H	Play Forward fast x20
Shuttle zone 1	I	Play Backward slow /5
Shutle zone 7	O	Play Backward fast x20

7. RESULTS AND DISCUSSION

For a deeper analysis of the individual contribution of the ideas presented in the paper, the time used to annotate three video sequences of 15K frames each (10 minutes of real time video at 25 fps) is studied and compared among 4 different interfaces. Each sequence was recorded at a different location of the city. The first two were captured at daytime while the third one was taken at nighttime.

The first interface does not include any of the ideas introduced in this paper and it is similar to ViPER-GT. It is based on the use of the mouse to indicate with precision some points on the image. When a vehicle reaches the area of interest it is signaled by clicking on the edge of the front part. To compute the speed, the same point of the vehicle is clicked again in the next frame. Finally, a third click on the rear part of the vehicle is needed to compute the occupancy.

The second interface (Only Keyboard) introduces the idea of building the ground-truth using a main line and an auxiliary line. However, it does not use either the jog-shuttle wheel neither the motion detection facility to navigate through the video. One of the users' hands is used to press the SPACE key and the other is used to press keys that allow to navigate the video backward, forward, slow, fast and frame by frame.

The third method is the same as the second one plus the use of the jog-shuttle wheel without motion detection. One hand is for pressing the SPACE key and the other one controls the wheel. We name it "Wheel".

Finally, the fourth one includes all the elements described in this paper: measurement lines, jog-shuttle and motion detection. This interface is called "Motion".

Table 2 shows the times required to annotate the three video clips using each interface. Note that the annotation time depends on the number of vehicles and can be drastically reduced with the ideas introduced in this paper.

Table 2. Time used in each procedure

Time (minutes:seconds)						
Clip	number vehicles	ViPER	Mouse based	Only Keyboard	Wheel	Motion
1	69	80:18	65:30	21:59	13:58	11:20
2	72	89:23	75:50	2349	15:03	12:17
3	21	32:11	28:51	13:42	05:41	03:48
Total	162	201:52	170:21	59:57	35:00	27:42

In Table 2 we also compare our interfaces with ViPER-GT. The procedure with ViPER-GT can be summarized as follows. For each vehicle, create a bounding box in the frame when the car is located a few pixels away before the measurement line, play forward the video until the vehicle crosses the line and then create a second bounding box. With this information, the ViPER-GT system can interpolate the car coordinates after a user request. With this information, it can be derived the average velocity and occupancy for a vehicle.

From the results, we found that the largest reduction factor in the annotation time (2.84 times in average, i.e., Mouse based total time divided by Only Keyboard total time) is achieved by the idea of drawing a line on the screen and wait until a car arrives to it. This eliminates the need for precise mouse clicks. The use of the wheel is the second most important reduction factor (about 1.7). Finally, the impact of the motion detection is the smallest, and note that there are differences among the three different scenarios. This is because motion detection is more effective if there are large intervals with no cars (as in the nighttime). The overall factor achieved with motion detection is 1.26. Thus, a total reduction time factor of 6 is achieved when the whole system is compared to the Mouse interface. This factor is increased to 7 if the whole system is compared to ViPER-GT. Notice that ViPER-GT is a general annotation tool that can be used in different contexts and has not been optimized for traffic video applications.

Apart from evaluating the annotation time, we have performed a small poll to evaluate the user satisfaction with the well-known standard System Usability Scale questionnaire [4]. The test contains the next questions:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The user can rank each of the previous questions with one of five levels of agreement that range from ‘strong disagree’ to ‘strong agree’. The questionnaire was answered immediately after the annotation process. The number of users inquired was 10 each one annotated 2 different clips. A mark of 52 out of 100 was obtained for the Mouse based interface while a 78 out of 100 was achieved for the Motion version. This mark shows a high degree of satisfaction.

8. CONCLUSIONS

In this paper, we proposed a setup constituted by a laptop, a jog shuttle wheel and software to efficiently annotate the ground truth of traffic videos.

The system annotates individual vehicle events at given locations. The annotated data can be used for counting vehicles, measuring their speeds, and determining lane occupancy.

We tested the setup annotating part of our database composed by 10 scenarios of real video of the cameras of the Local Traffic Authority of the city of Valencia, Spain (about 800 traffic cameras). The tool demonstrated to be very user friendly because the user only needs to use one hand managing the jog shuttle wheel and the other hand on the key SPACE of the keyboard. It is very efficient because, operating at 25 frames per second, the mean time required to annotate a 10 minutes (15K images) video file ranges from 4 to 13 minutes per lane depending on the traffic intensity.

9. ACKNOWLEDGMENTS

The authors thank Etra I+D and Ruth López and Jaime Benlloch from Local Traffic Authority of Valencia, Spain by providing the video files. Also thanks to the VIGTA 2012 Conference Organizers, its participants and the anonymous reviewers of this journal for their valuable advice.

This work was funded by the Spanish Government project MARTA under the CENIT program and CICYT contract TEC2009-09146.

10. REFERENCES

[1] Albiol, A. et al. 2011. Detection of Parked Vehicles Using Spatiotemporal Maps. *Intelligent Transportation Systems, IEEE Transactions on*. 12, 4 (Dec. 2011), 1277 –1291.

[2] Blunsden, S.J. and Fisher, R. The BEHAVE video dataset: ground truthed video for multi-person behavior classification. *Annals of British Machine Vision Association*. 2010, 4, 1–12.

[3] Bradski, G. and Kaehler, A. 2008. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly.

[4] Brooke, J. SUS: a “quick and dirty” usability scale. *Usability Evaluation in Industry*. Taylor and Francis.

[5] Brostow, G.J. et al. 2009. Semantic object classes in video: A high-definition ground truth database. *Pattern Recognition Letters*. 30, 2 (2009), 88 – 97.

[6] Buch, N. et al. 2011. A Review of Computer Vision Techniques for the Analysis of Urban Traffic. *IEEE Transactions on Intelligent Transportation Systems*. 12, 3 (Sep. 2011), 920–939.

[7] D'Orazio, T. et al. 2009. A Semi-automatic System for Ground Truth Generation of Soccer Video Sequences. *Advanced Video and Signal Based Surveillance, 2009. AVSS '09. Sixth IEEE International Conference on* (Sep. 2009), 559 –564.

[8] Dollar, P. et al. 2012. Pedestrian Detection: An Evaluation of the State of the Art. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 34, 4 (Apr. 2012), 743 –761.

[9] Faro, A. et al. 2011. Adaptive Background Modeling Integrated With Luminosity Sensors and Occlusion Processing for Reliable Vehicle Detection. *Intelligent Transportation Systems, IEEE Transactions on*. 12, 4 (Dec. 2011), 1398 –1412.

[10] Giro-i-Nieto, X. et al. 2010. GAT: a Graphical Annotation Tool for semantic regions. *Multimedia Tools and Applications*. 46, 2-3 (2010), 155–174.

[11] i-LIDS. Image Library for Intelligent Detection Systems: *www.ilids.co.uk*.

[12] Kasturi, R. et al. 2009. Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*. 31, 2 (Feb. 2009), 319 –336.

[13] Laganière, R. 2011. *OpenCV 2 Computer Vision*. Packt Publishing.

[14] Lorist, M.M. et al. 2000. Mental fatigue and task control: Planning and preparation. *Psychophysiology*. 37, 5 (2000), 614–625.

[15] Russell, B. et al. 2008. LabelMe: A Database and Web-Based Tool for Image Annotation. *International Journal of Computer Vision*. 77, 1 (2008), 157–173.

[16] Serrano, M. et al. 2010. Interactive Video Annotation Tool. A. de Leon F. de Carvalho et al., eds. Springer Berlin / Heidelberg. 325–332.

[17] Traffic City Cameras Valencia: *camaras.valencia.es*.

[18] TREC Video Retrieval Evaluation: *http://www-nlpir.nist.gov/projects/trecvid/*.

[19] Vezzani, R. and Cucchiara, R. 2010. Video Surveillance Online Repository (ViSOR): an integrated framework. *Multimedia Tools and Applications*. 50, 2 (2010), 359–380.

[20] ViPER: The Video Performance Evaluation Resource: *http://viper-toolkit.sourceforge.net/*.

[21] Volkmer, T. et al. 2005. A web-based system for collaborative annotation of large image and video collections: an evaluation and user study. *Proceedings of the 13th annual ACM international conference on Multimedia* (New York, NY, USA, 2005), 892–901.

[22] Zhang, H.-B. et al. 2012. Adaptive photograph retrieval method. *Multimedia Tools and Applications*. (2012), 1–21.

[23] Zou, Y. et al. 2011. Traffic incident classification at intersections based on image sequences by HMM/SVM classifiers. *Multimedia Tools and Applications*. 52, 1 (2011), 133–145.