Large Scale Track Analysis for Wide Area Motion Imagery Surveillance

C.J. van Leeuwen, J.R. van Huis, and J. Baan

TNO Technical Sciences, Oude Waalsdorperweg 63, 2597 AK The Hague, Netherlands

ABSTRACT

Wide Area Motion Imagery (WAMI) enables image based surveillance of areas that can cover multiple square kilometers. Interpreting and analyzing information from such sources, becomes increasingly time consuming as more data is added from newly developed methods for information extraction. Captured from a moving Unmanned Aerial Vehicle (UAV), the high-resolution images allow detection and tracking of moving vehicles, but this is a highly challenging task. By using a chain of computer vision detectors and machine learning techniques, we are capable of producing high quality track information of more than 40 thousand vehicles per five minutes. When faced with such a vast number of vehicular tracks, it is useful for analysts to be able to quickly query information based on region of interest, color, maneuvers or other high-level types of information, to gain insight and find relevant activities in the flood of information.

In this paper we propose a set of tools, combined in a graphical user interface, which allows data analysts to survey vehicles in a large observed area. In order to retrieve (parts of) images from the high-resolution data, we developed a multi-scale tile-based video file format that allows to quickly obtain only a part, or a sub-sampling of the original high resolution image. By storing tiles of a still image according to a predefined order, we can quickly retrieve a particular region of the image at any relevant scale, by skipping to the correct frames and reconstructing the image. Location based queries allow a user to select tracks around a particular region of interest such as landmark, building or street. By using an integrated search engine, users can quickly select tracks that are in the vicinity of locations of interest. Another time-reducing method when searching for a particular vehicle, is to filter on color or color intensity. Automatic maneuver detection adds information to the tracks that can be used to find vehicles based on their behavior.

Keywords: Wide Area Motion Imagery, Big Data, Motion in Motion, Multi-scale Images, Data Analysis

1. INTRODUCTION

In a society with an ever-expanding rich stream of information, the analysis and interpretation of large amounts of data becomes increasingly time consuming. As new methods for gathering data are developed, more sources of information are added to the already existing streams of data, and existing data interpretation methods are augmented with more input. Analysis of large amounts of data must be done efficiently to avoid overloading the system with irrelevant information, yet reliably enough to obtain the useful facts that the human analysts want to retrieve. Since the introduction of computer data analysis, intelligent software aids human analysts in conquering the flood of data that is provided by sensors, in order to filter out the noise, and provide new insights that were impossible or extremely tedious to obtain manually.

Video is one type of data streams that is capable of providing a tremendous amount of information on practically any subject, but is also capable of providing a lot of noise that tends to occlude the useful data. Especially in the case of Wide Area Motion Imagery (WAMI), in which a single frame covers multiple square kilometers, extracting relevant events is a challenging task. Moreover, typically such data is gathered using Unmanned Aerial Vehicles (UAVs), which can gather data for extended periods of time, all of which has to be processed in order to be analyzed afterwards.

Further author information: (Send correspondence to C.J. van Leeuwen.)
C.J. van Leeuwen.: E-mail: coen.vanleeuwen@tno.nl

In this paper we propose a set of tools, which allows data analysts to search for events in a large image database obtained from an airborne platform, and in particular for surveillance of vehicles in the complete observed area. The tools that will be described in this paper are methods for quickly retrieving images for multiple scales, automated methods for extracting vehicle tracks from the video, track analysis algorithms that add high-level information to the vehicle trajectories and finally different types of queries to search for events. Combined into a Graphical User Interface (GUI), these components provide a powerful tool for inspecting object tracks from large WAMI datasets. The goal of this tool is to allow analysts to search for suspicious or anomalous events, or to find and track vehicles that were observed at a given time and location.

The setup of the paper is as follows: in Section 2 the different methods are introduced, and in Section 3 the used platform and video data will be described. Following, in Section 4 the results will be shown, and finally in Section 5 the discussion and drawn conclusions are presented.

2. METHODS

The proposed GUI can be used to search through the complete dataset, enriched with information that is either provided by the automated image processing algorithms or information from third parties. The video images are stabilized and warped to the ground plane, such that they can be inspected from a top-down perspective. By only loading the relevant area in a resolution suitable to the current zoom-level, excessive loading of data is avoided, and users can quickly inspect the video while maintaining a high quality visual overview. The automatic vehicle tracking system[1] provides information from all observed vehicles as they navigate through the area. The trajectory information can be shown to the user as an overlay on the video, which will reflect the vehicle's movement as the track progresses, or on a static map. Third party information provides information such as street names, landmarks or other known locations*. A schematic overview of the components in the tool is shown in Figure 1.

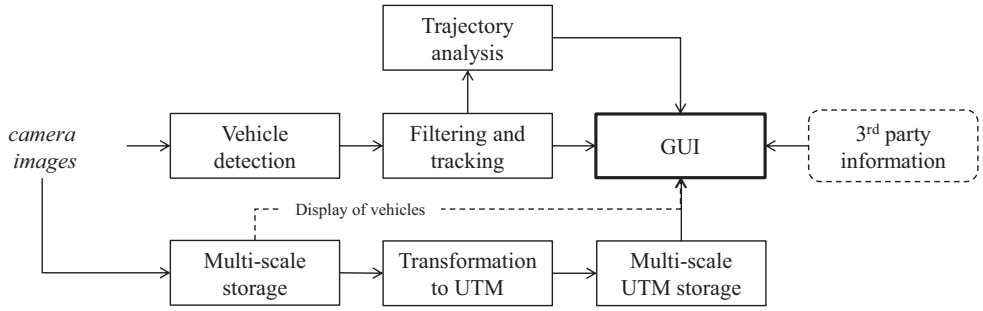


Figure 1: Flowchart depicting the different components in the tool.

2.1 Storage of imagery

To allow fluid navigation through the high definition video, we intend to store and retrieve the images in such a way that we only need to load the data that we present to the user. Normally the complete 116 Megapixel image is loaded, and is cropped and rescaled at the last moment before showing the requested part of the image on screen. There is a delay in loading excess data, either from parts that are out of the region-of-interest (ROI) or excess data that would not render due to the display resolution, and secondly there is a delay from having to crop and rescale the image when they are requested. In order to reduce the loading time we propose the following method for storing and retrieving image data inspired by the work of van Brandenburg *et al.*[2]

First to avoid having to load the parts of the image that lie outside of the current view, we store the images as a sequence of *tiles* in a video stream. Every tile has a predefined width h_t and height w_t (in the current implementation both are set to 128 pixels). When a certain part of the image is retrieved, the corresponding frame numbers can be computed, and only that part of the image is loaded by only decoding the relevant video

*In the current implementation this information is provided using Google Places (https://developers.google.com/places/), but other plugins could be used as well.

(a) Original view from camera. (b) Warped image to ground plane.

Figure 2: An example of the original image, and warped to the ground plane.

frames. To solve loading more images than is necessary for displaying at the screen resolution, instead of storing one copy of an image, we store multiple copies at different scales. By storing images at a ratio ρ at scale $\sigma = \{0, 1, ..., 12\}$ using

$$\rho(\sigma) = 2^{-\frac{\sigma}{2}}, \tag{1}$$

we make sure that for any required resolution we load at most $\sqrt{2}$ the number of pixels required. Of course by using a different scaling function we could reduce this even more, but this is a good balance between required storage and reading performance. Due to the tile-based strategy we have to load four strips of pixels outside the ROI that are at most h_t pixels wide and w_t pixels high. This means that for a requested image of $h_i \times w_i$ pixels we have to load at most

$$\sqrt{2}h_i w_i + 2w_t h_i + 2h_t w_i + 4h_t w_t \tag{2}$$

pixels. In practice this means that on average we need to read about twice the data that would be strictly required, instead of having to read over 1000 times the data.

The exact same procedure is repeated for the original image, warped to the ground plane, which results in a top-down view as can be seen in Figure 2. This image is alway rotated in such a way that north is up, and can be aligned with a map. Because it is stable over time, it is shown as the overview image in the GUI, making it easier and more intuitive to follow vehicles on this warped image. Furthermore, it allows the overlay of external information such as streets, landmarks or other known locations.

2.2 Vehicle tracking

To obtain vehicle information from the video stream, a tracking system is used to obtain accurate, long tracks of the majority of the vehicles in the image,[3] which is schematically depicted in Figure 3. The first step in obtaining these vehicle trajectories is to use a static object detector, trained using an Adaboost training algorithm on Haar features.[4] This detector works on individual images and therefore does not depend on image motion stabilization or stitching. The training set is created by manually annotating the vehicles in six frames, which due to the high resolution of the images is enough to create a reliable model. The resulting detector is then applied to all image frames in the video, and the resulting detections are stored as bounding box coordinates.

The detections obtained in the first step are filtered based on their altitude. In order to determine these altitudes, first a 3-dimensional reconstruction of the area was created using stereoscopic 3D estimation.[5] Using this 3D reconstruction, a height map for each video frame could be made. These height maps were then used to estimate the height of each detection. This filtering removes false vehicle detections, although vehicles in car parks located on top of buildings were incorrectly discarded.

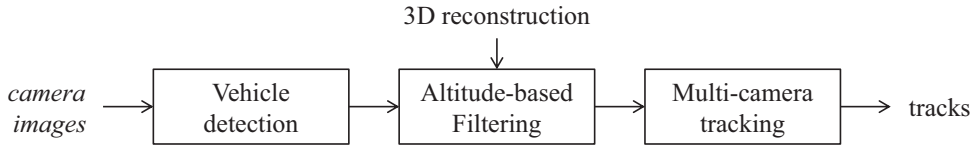


Figure 3: Flowchart depicting the object tracking process.

Finally the trajectories are constructed based on the remaining detections. Tracks are lists of associated detections, which match the expected position, based on the previous detections of that vehicle. The prediction of the tracks is performed by template matching.[6] This approach has the advantage that 1) it counters the inaccuracy of the GPS data that is embedded in the video data and 2) it can find matches when the vehicle detector missed a detection.

For a new frame, the updated location of a vehicle is predicted and converted to image coordinates. The calculated image coordinates are the center of the search area for the template matching, which is done in the Fourier domain by calculating the sum of squared differences. For speed considerations the search area is typical 50 pixels in each direction outside the template. The template matching returns the 10 best matches.

If the cost of the best match c_1 is significantly smaller than the cost of the second best match c_2 such that$c_2 > 1.2 \times c_1$; the best match becomes the new prediction. If this is not true, the center of the search area will be the new prediction. If any the bounding box of a detection overlaps more than 50% with the predicted location of the track, the detection is added to the track.

2.3 Track analysis and maneuver based queries

Vehicle route analysis is an important instrument for finding vehicles of interest. This instrument can be used to retrieve vehicles that are known to have driven a certain route or to retrieve vehicles that show abnormal behavior. Abnormal route behavior might involve vehicles that drive completely around a certain building or house block. An algorithm is constructed, which enables rotation based turn analysis by adding the turn angle to each track element. After filtering this instantaneous track angle over using a low-frequency filter, the queries can be defined to filter for vehicles that perform maneuvers such as “U-turn”, “left turn” and “right turn”.

2.4 Location based queries

In order to allow the user to limit the visible tracks, to those that are in a specific region we allow three different types of queries:

Line queries Line queries search for vehicles that passed at some time a virtual line that is drawn on the map. The intersections of all tracks are sought with the drawn line segment, and only tracks that have a valid intersection are returned for visualization. This type of query is especially useful for filtering vehicles that crossed a specific street or a bridge.

Bounding box queries By selecting a bounding box on the map, the user can select any region in which he is interested. Only vehicles that were in any frame within this rectangle (based on the UTM coordinates) are returned. Typically this is useful when the user wants to inspect vehicles within a particular area, defined by the user’s knowledge or interest.

Location based queries The location based query, filters vehicles that were within a given radius of a known location. This contrasts with the bounding box queries in the sense that the user can search for a place of interest, such as a street, landmark, monument or a building, and all vehicles that were at any time within a provided distance to that place are returned. In the current implementation we used the a public third-party location API, which also allows to search for places such as gas stations, banks, musea or schools. This is particularly useful if the user is interested in vehicles that were in an area directly related to publicly known location.

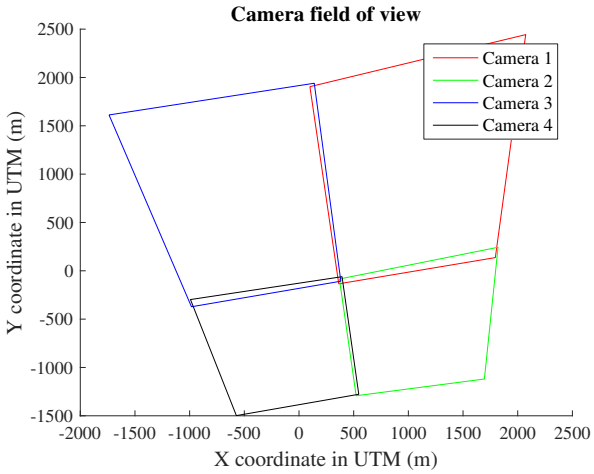


Figure 4: The field-of-views of the four cameras plotted on the ground plane in UTM coordinates.

2.5 Color based queries

The last type of query is color-based; which is applicable when a user is interested in retrieving only vehicles of a particular color or "lightness", i.e. the user wants to retrieve only light or dark vehicles. Upon choosing to filter based on color, the *average* vehicle colors are computed by using the center of the vehicle detections. Since in the middle of the detection there is often a reflection of the sun, a "donut" shaped image mask prevents taking into account an estimate of the vehicle that is over-saturated. By selecting different filters on the Hue, Saturation and Intensity channels of the vehicle colors, the user can inspect vehicles that match a description of a vehicle of interest.

3. DATA DESCRIPTION

In order to test and evaluate the methods we implemented the user interface, and loaded WAMI data obtained with the CorvusEye® 1500 system.[7] The images were recorded with four RGB cameras with a resolution of 6600×4400 pixels each, running at 2 frames per second. After stitching the images into one large frame, the total resolution is $13\,200 \times 8800$ pixels, which equals 116 Megapixels. An example of the camera setup is depicted in Figure 4 from which it is clear that the field-of-view of the cameras slightly overlap.

An example of a stitched image is depicted in Figure 5, which shows the overview of the complete stitched image (Figure 5a), as well as a close-up indicating the amount of detail captured by the cameras (Figure 5b). The ground resolution varies between 0.18 meter/pixels to 0.31 meter/pixel, which directly relates to the resolution of the warped image, which is $44\,000 \times 66\,000$ pixels, or 2.4 Gigapixels, although because of the methods described in Section 2.1, we can greatly reduce the overhead both in storage size and in loading time.

In total 3156 images are used, which show the downtown of Rochester, NY, USA, and were captured from an airplane circumnavigating the city with a timespan of approximately 6.5 minutes, covering more than $20\,\text{km}^2$.

4. EXPERIMENTS AND RESULTS

4.1 Image Storage

To evaluate the performance of the image storage process described in Section 2.1, we measure the amount of data loaded and time to load a requested image when we load random parts of the image at random scales. Notice that this provides on average poorer results than in the GUI, in which we fix the image scales to those that are known to be available. We compare the measurements with the traditional approach in which we load the complete image and zoom and crop to the requested part.

Averaged over 1000 different requested image regions and scales, we see that our methods loads 3.76 times too many pixels compared to 4276 too many pixels for the naive approach. This is also reflected by the loading

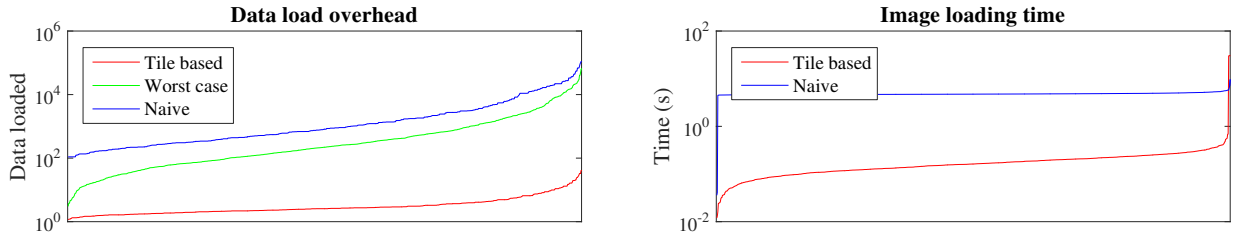
(a) Stitched frame of four combined cameras. (b) A closeup of image showing the detail.

Figure 5: An example of the overview and detail captured by the cameras.

time, which takes 0.295 s for the tile-based method versus 4.787 s for the traditional approach. In Figure 6a the mean overhead of loaded pixels is shown for the tile-based method as well as the worst-case scenario. All 1000 experiments are shown, sorted by their relative performance. In Figure 6b the loading times are shown for the 1000 requests.

4.2 Vehicle Tracking

After tracking, all tracks are linearly interpolated at the following time instances: 1) at which no detections were added to the tracks and 2) that lie between time instances containing associated detections. After filtering for incorrect tracks (due to detections of non-vehicles or detections for which no matches were found) this set contains 46 605 tracks and 4 015 764 detections, resulting in a mean track length of 86.1 frames (42 s). A histogram of the track duration is shown in Figure 7. The resulting tracks where used to evaluate the turn-based anomaly detection, as well as the other functions of the GUI.



(a) Percentage of pixels loaded for any request. (b) Time required to load sub-frames

Figure 6: Results of the image storage process showing the overhead in data loaded and in loading time.

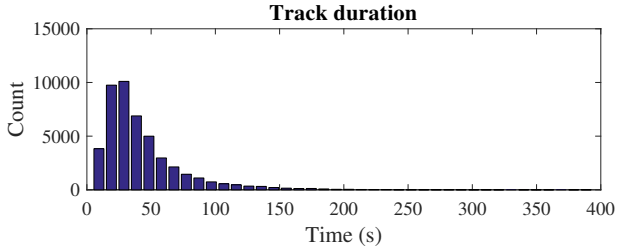


Figure 7: A histogram of the track duration shows that many vehicles can be tracked for more than 10 s.

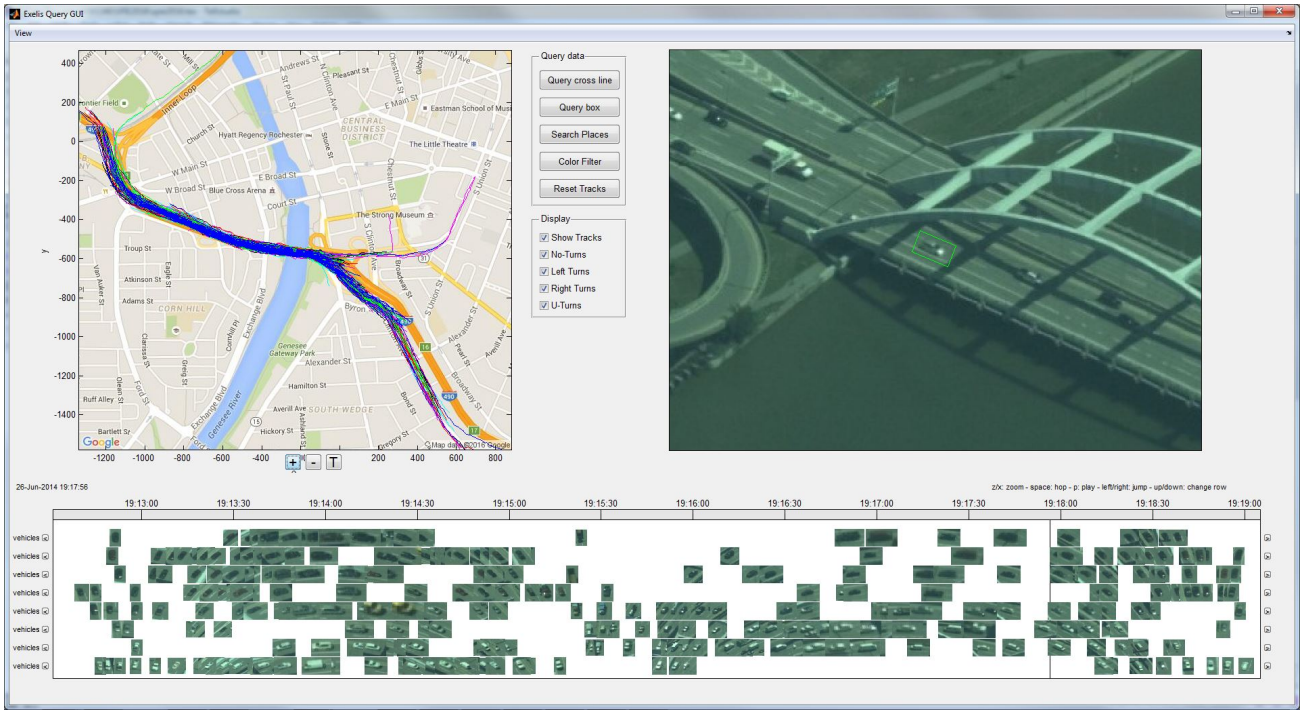


Figure 8: An example of the GUI in which the user selects tracks that crossed a bridge.

4.3 GUI

All the aforementioned functionalities come together in the form of a GUI. It allows users to inspect the WAMI images, as well as external maps, landmarks and the results of the tracking of vehicles. Using different types of queries, the GUI allows to focus on a subset of vehicles that perform certain maneuvers, have a particular color, or were ever at a specified location. In the current paper we present two examples in which most of the functionality is demonstrated.

In Figure 8 a user used the line query to inspect vehicles that crossed a bridge. On the left the vehicle tracks are shown on an externally loaded map, and on the right one instance is shown in the original camera image. In the bottom all vehicles are shown that crossed the bridge within the available dataset, sorted by lightness.

In Figure 9 a location based query is done for a particular brand gas station and additional information is provided that the user is interested in vehicles that were within 100 m of a gas station matching with the query. On the left a top-down image is shown based on the stitched WAMI image, and on the right an additional query is added indicating that the user wants to see vehicles of a specific color. Finally using the tick marks in the center, the user has specified to only show vehicles that made a turn.

5. CONCLUSION

In this article we present methods and a GUI for inspecting WAMI data, as well as large numbers of (vehicular) tracks for surveillance purposes. We presented a tile-based image storage method, which allows for fast image retrieval from an otherwise large image source. Experiments show that this leads to a reduction of data load of two orders of magnitude and on average a factor of twelve reduction in loading time.

Our presented vehicle tracking method processes 6.5 minutes of video of more than 20 km^2 at 2 fps resulting in over 46 thousands of tracks with a mean track duration of 42 seconds. It is shown that with these tracking results additional information can be extracted such as the speed and turning behavior of each track segment, providing valuable insight for an operator.

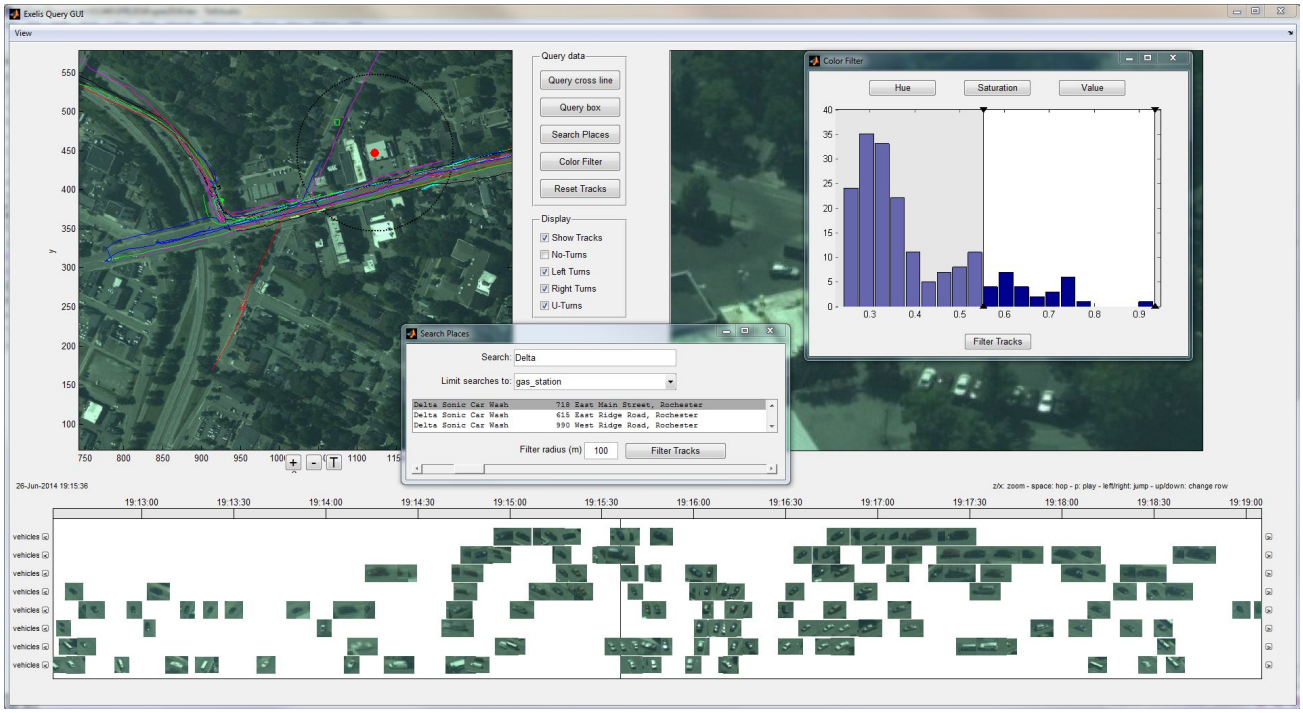


Figure 9: A screenshot of the GUI in which the user select white vehicles that turned left or right near a specific gas station.

Finally a GUI was shown, which can be utilized by a user to quickly navigate through all data, to more accurately inspect the target of interest. Tile-based image storage allows for quick navigation through the imagery, third party location information aids in guiding the operator's search, and queries on the tracks allow to filter out noise from the vehicles of interest.

ACKNOWLEDGMENTS

The authors would like to thank Harris Corporation for the availability of the CorvusEye® 1500 imagery.[7] This research is done within the Unmanned Systems program (V1340) in which TNO researches the use of different unmanned systems.

REFERENCES

[1] van Eekeren, A. W. M., Dijk, J., and Burghouts, G., "Detection and tracking of humans from an airborne platform," *Proc. SPIE* **9249** (2014).

[2] van Brandenburg, R., Niamut, O., Prins, M., and Stokking, H., "Spatial segmentation for immersive media delivery," *Proc. ICIN* **15**, 151–156 (2011).

[3] van Eekeren, A. W. M., van Huis, J. R., Eendebak, P. T., and Baan, J., "Vehicle tracking in wide area motion imagery from an airborne platform," *Proc. SPIE* **9648** (2015).

[4] Viola, P. and Jones, M., "Rapid object detection using a boosted cascade of simple features," *Proc. CVPR* **1**, I–511 (2001).

[5] Furukawa, Y. and Ponce, J., "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Mach. Intell.* **32**(8), 1362–1376 (2010).

[6] Lewis, J. P., "Fast template matching," in [*Vision interface*], **95**(120123), 15–19 (1995).

[7] CorvusEye® 1500 imagery courtesy of Harris Corporation (2014).