HVE-WP-2020-3

Video-based Accident Reconstruction from Vehicle Camera System

Gregorij Kurillo, Evan Hemingway, Louis Cheng
Applied BioMechanics
Alameda, CA

2020 HVE Forum
Austin, TX
February 24 – 28, 2020

Video-based Accident Reconstruction from Vehicle Camera System

Gregorij Kurillo, Evan Hemingway, Louis Cheng
Applied BioMechanics, Alameda, CA

Abstract

Vehicle surveillance camera systems are becoming prevalent in public transportation as well as private vehicles. In this paper, we present Video-based Accident Reconstruction System (VARS), a software tool developed for the motion analysis of traffic accidents as captured by vehicle and land surveillance cameras. Working with the point cloud data of an accident site, this 3D interactive tool provides frame-to-frame motion analysis of the vehicle and the location of objects in the environment (e.g., other vehicles, pedestrians). Based on an annotation of keyframes in the video and the point cloud, this software uses photogrammetry and computer vision techniques to extract the path and velocity of the vehicle(s) and surrounding objects of interest. The tool can render drive-through videos from different vantage points such as from the vehicle camera, driver's viewpoint, and other stationary or moving camera views. Furthermore, the calculated vehicle motion data can be exported for a vehicle dynamics analysis in programs such as HVE. The latter half of this paper provides step by step illustrations of the path reconstruction process for input to and use in HVE.

Introduction

Vehicle surveillance camera systems are now ubiquitous in public transportation and are becoming prevalent in private vehicles. The video information recorded by such systems may be accompanied by GPS data and other telemetry obtained from the vehicle (e.g., speed, braking events). In the case of traffic accidents, the video information is extremely valuable in determining what happened in the accident and understanding the possible roles played by surrounding factors. To gain further insight into what happened, it is often desirable to visualize and analyze the accident not only from the viewpoint of the attached camera systems on the vehicle, but also from other relevant stationary or moving locations. One of the major challenges in the video analysis is in mapping the video information into a 3D simulation. This process can be done manually by analyzing the video frame-by-frame and matching the camera view from a simulated vehicle path to the video camera's path, although this process may prove to be tedious and time consuming. While commercial photogrammetry software is available for video analysis using footage from stationary surveillance video cameras, there are currently few, if any, available products that can facilitate a vehicle path reconstruction directly from video captured by a moving camera.

In this paper, we present Video-based Accident Reconstruction System (VARS), a software tool developed for the motion analysis of traffic accidents captured by vehicle and land surveillance cameras. The development of VARS is ongoing, but the functional features implemented to date have been used successfully in the analyses of real world accidents. This paper presents the technical details of the software and demonstrates its functionality and utility for HVE by reconstructing a test drive with a GPS-based data recorder and a real-world accident.

Video-based Accident Reconstruction System (VARS)

VARS implements photogrammetry and computer vision techniques to extract the paths of vehicles and surrounding objects of interest using surveillance video and point cloud data of the accident site as inputs. The software features an interactive user interface (UI) that provides the following functionality:

- frame-by-frame video analysis (currently for a single view video input)

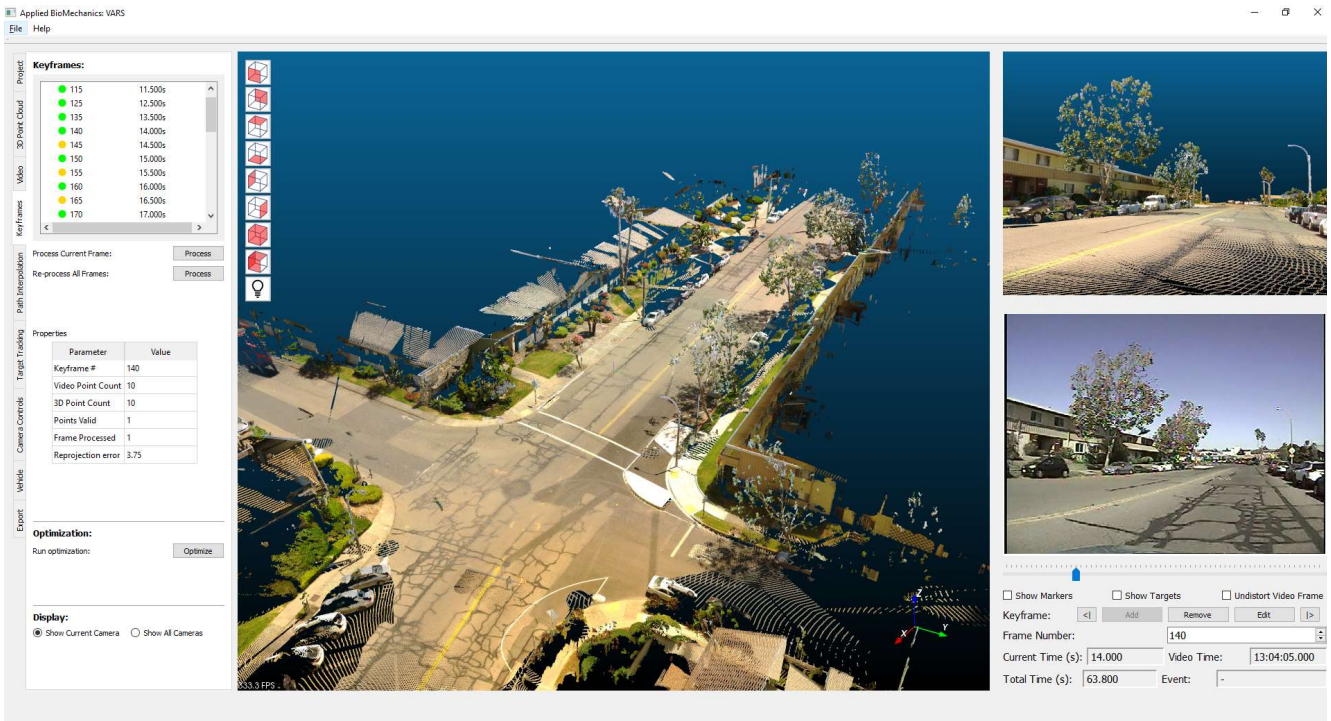

Figure 1. *The user interface of VARS featuring a 3D visualization of an accident site's point cloud and an interactive video player.*

- 3D visualization of point cloud data and/or mesh model of the environment
- registration of camera pose at selected keyframes
- reconstruction of vehicle path for drive-through simulation
- rendering of drive-through videos from different vantage points such as driver's point of view, top view and vehicle chase view
- estimation of object pose (e.g., parked vehicles, traffic cones, moving pedestrians) from video data
- export of vehicle path and object coordinates for use in vehicle dynamics simulation software such as HVE.

Figure 1 shows the main user interface of VARS. The central window displays the point cloud of the environment and provides an interactive view of the scene with a toolbar for quick selection of fixed views. As will be soon described, the central window is used to select 3D keypoints for frame registration. The two side graphics windows feature the current rendered camera viewpoint (top right) and the current video frame as captured by the vehicle camera (bottom right). Video and keyframe controls are provided under the camera window along with information on the current frame number and time stamp. The panel on the left side of the UI, divided into several sections, provides granulated controls for the data analysis, including: (1) project information, (2) point cloud controls for setting point size and lighting, (3) keyframes for managing path segmentation, (4) path interpolation with export to video capabilities, (5) target (object) tracking, (6) manual camera controls, (7) vehicle mesh settings, and (8) controls for exporting path data.

The software user interface (UI) is built within a Qt Framework and using the C++ programming language. Vision processing is supported by the OpenCV (Open Source Computer Vision) Library [1], while the point cloud analysis and visualization are implemented using PCL (Point Cloud Library) [2].

In the remainder of this paper we will present the processing pipeline for data analysis using the video and point cloud. We will compare the calculated vehicle path with a GPS-based data recorder, and analyze the vehicle dynamics associated with the calculated vehicle motion using the HVE program.

Cameras and Input Video Data

Vehicle video surveillance systems consist of one or more cameras that are mounted on the vehicle frame. Different camera models may be used on the same vehicle, depending on requirements such as size, mounting location (interior vs. exterior), field of view, high dynamic range, and night vision. The resolution of the camera is one of the factors that determines the level of detail visible in the image. Cameras come in various resolutions with the most common resolutions being VGA (640x480 px), 720p HD (1280x720 px), and full HD (1920x1080 px). Another important factor is the type of the lens mounted on the camera. The lenses are typically characterized either by their focal length or angular field of view. For a given sensor size, the shorter the focal length, the wider the angular field of view. Since the sensor size varies across camera models, the focal length is typically expressed as a 35 mm equivalent focal length for ease of comparison. Cameras with wider angular field of view typically exhibit large geometric distortions of the captured image, which are most apparent on the edges of the video frame.

In vehicle video surveillance systems, the output of a camera is typically recorded by a central recording device which timestamps the acquired video frames and stores them into persistent memory such as a solid state drive. Oftentimes these devices also store other metadata such as vehicle events (braking, signals, door opening, accelerometer data, and GPS data). In order to store video data from multiple views across a substantial time period (e.g., 30 days), the video image may be reduced in: (1) resolution from its original capture, (2) quality due to applied video compression, and (3) frame rate.

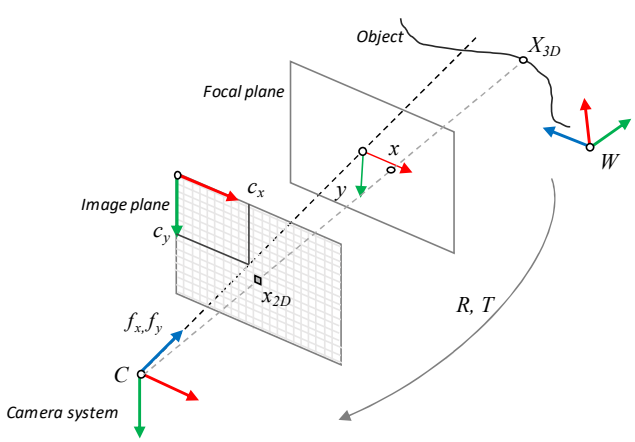

Figure 2. *Pinhole camera model geometry.*

When using video data from vehicle video surveillance systems for photogrammetry and computer vision-based analyses, all of the above factors should be taken into consideration.

The Camera Model

The key step to extracting the camera path and other information from a video sequence is to determine an accurate calibration of the camera and to register it to the world space. The digital camera is a very complex opto-electrical device, but it is often sufficient to describe its image creation properties by a simplified geometric model. We use a standard pinhole camera model (**Figure 2**) with radial and tangential distortion correction [3]:

$$x_{2D} = \underbrace{\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}}_{P} X_{3D}. \qquad (1)$$

The model in Eq. 1 is a linear mapping of the 3D scene point X_{3D} that can be seen by the camera to the projected image of the point, represented by its pixel coordinates x_{2D}. The linear mapping is represented by the 3×4 projection matrix, P. The dimensions of the vectors X_{3D} and x_{2D} are 4×1 and 3×1, respectively, as they are expressed using homogeneous coordinates. The parameters f_x and f_y are the camera focal lengths (in pixel units) in the horizontal and vertical directions of the image while c_x and c_y are the image plane coordinates of the optical center of the lens (typically around the center of the image). These four scalar parameters are considered *intrinsic* camera parameters since they are properties specific to a particular camera. Even for cameras of the same model, the intrinsic parameters somewhat differ and require some refinement to find the correct values. The mapping also depends on the *extrinsic* camera parameters R and T, which are the 3×3 orientation matrix and the 3×1 position vector of the camera expressed in the world coordinates, respectively. In this application, the coordinate system associated with the point cloud is chosen as the world coordinate system.

Camera distortion is captured by a nonlinear mapping from the projected point locations predicted by the linear mapping of Eq. 1 to their actual recorded location on the image. A radial distortion is applied to the result of Eq. 1 using two parameters k_1 and k_2 as follows:

$$x_{corr} = x(1 + k_1 r^2 + k_2 r^4),$$
$$y_{corr} = y(1 + k_1 r^2 + k_2 r^4).$$

Here, $x = x_{2D}(1)$, while $y = x_{2D}(2)$ and x_{corr} and y_{corr} are their corrected values. The distance r is given by $r = \sqrt{x^2 + y^2}$. Following radial distortion, a tangential distortion may be applied to account for any misalignment between the image plane and the lens principal plane. The deformation applied to the image is described with tangential distortion parameters p_1 and p_2 as follows:

$$x_{corr} = x + (2 + p_1 xy + p_2(r^2 + 2x^2)),$$
$$y_{corr} = y + (2 + p_1(r^2 + 2y^2) + 2p_2 xy).$$

The tangential distortion is typically very small or zero for high quality vision cameras. To estimate all of the intrinsic parameters, cameras can be calibrated using a known target, such as a black and white planar chessboard target of known dimensions [4]. Based on a set of captured points, a system of equations incorporating the camera model from Eq. 1 can be formed and unknown parameters can be derived. OpenCV computer vision library includes several calibration methods that can be used for calibration with the target.

If it is not possible to collect custom target data, the camera's intrinsic parameters can be estimated from the camera specifications (focal length, sensor size, image resolution). Using selected scene features from one or two video frames and their corresponding elements in the point cloud of the accident scene, a non-linear optimization can be applied to simultaneously refine and adjust both the extrinsic and intrinsic parameters. This process may require an iterative approach where the initial refined solution can be used as the input to further optimization.

Camera Pose Estimation

Given the initial intrinsic camera parameters, 3D point cloud of the scene and a video frame, there is sufficient information to estimate the camera pose (i.e., the extrinsic camera parameters given by the matrices R and T) by the 3D-2D mapping between the points represented by the matrix P in Eq. 1. Although machine learning (ML) techniques could serve to match 3D and video imagery, this could be a difficult task in accident reconstruction applications, as the time between the video data and point cloud acquisition may be months or years apart. There may be changes in the lighting conditions, objects may no longer be present (e.g., parked cars, road work), the environment may have significant changes (e.g., grown or trimmed vegetation, re-building of structures), and there may be occlusions or missing data in the point cloud. Therefore, manual point correspondence selection remains the best option for analysis. In order to limit the amount of manual interaction, we limit the point matching only to selected keyframes and then interpolate the results.

In the first step, the user ideally selects between 8 and 12 keypoints in the point cloud window and the video frame (**Figure 3**). The keypoints for matching can include various static landmarks that are visible in both data sets such as road markings, traffic signals, light posts and features on neighboring structures. The selection of keypoints in the video should be distributed across the video frame while avoiding selection points situated close to the distorted edges in the keyframe. Furthermore, the selected keypoints should include different distance values that are not coplanar or collinear.

Once the selection is complete, the pose estimation algorithm determines the camera pose for a given set of object points, their corresponding image projections, and the camera intrinsic parameters (**Figure 3**). The algorithm finds a pose that minimizes the reprojection error of 3D points to their 2D counterparts while rejecting large outliers via the RANSAC procedure [5].

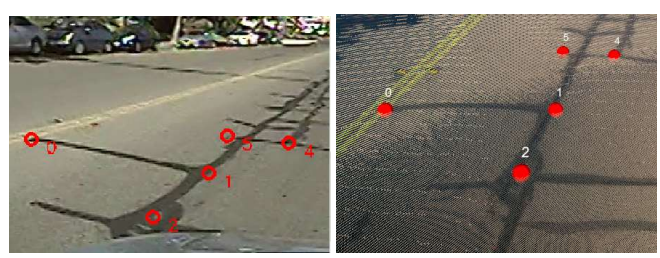

Figure 3. *Selected keypoints in the video frame (left) and corresponding keypoints in the point cloud (right).*

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

Figure 4. *An (a) original (distorted) input video frame, (b) undistorted video frame, (c) rendered 3D view, and (d) overlay of the undistorted video frame and rendered 3D view.*

The reprojection error is defined as the Euclidean distance between the observed keypoints in the video frame and the image of the keypoints from the point cloud, as calculated using Eq. 1.

The determination of camera pose is repeated for each keyframe in the video sequence. The keyframes should ideally be selected at constant time intervals. The choice of time interval between two keyframes depends on the vehicle speed and the frame rate of the video. For example, time intervals between 0.5 s and 1.0 s work well for a video with frame rate of 10 FPS. Additional keyframes can be introduced to capture important events, such as the start of a turn, the time of impact and appearance of an object of interest. **Figure 4** compares a selected keyframe as captured by the camera with the corresponding reconstructed view using the point cloud. For accurate comparison, the video frame is undistorted and overlaid with the rendered 3D view from the estimated camera pose.

Once the sequence is segmented into keyframes and the camera pose is determined for each frame, the software can further optimize both the intrinsic and extrinsic parameters or the extrinsic parameters only. We use the Levenberg-Marquardt algorithm [6] to solve the non-linear least squares problem for reprojection error minimization. The optimization can include additional constraints to create a smooth transition between keyframes. In **Figure 5**, constraining the camera height as a linear function of distance from the initial point reduces the jitter due to the errors in 2D-3D point selection process.

Path Interpolation

After the camera pose is estimated for each keyframe, the poses in the remaining frames can be interpolated. The interpolation of the camera 3D position is performed using Kochanek-Bartels Cubic Splines (also TCB splines) [7]. TCB splines take a sequence of position keyframes and perform a cubic polynomial interpolation between each pair of key frames by choosing incoming and outgoing tangent vectors in a specific way. The shape of the interpolated trajectory around the nodes is controlled via three parameters: (1) *tension* which controls the curvature around a control point, (2) *continuity* which controls the amount of discontinuity between incoming and outgoing tangents at a control point, and (3) *bias* which controls the direction of the path at a control point. The three parameters can be adjusted through the VARS UI. The algorithm was modified to allow for the interpolation of non-uniformly sampled data points.

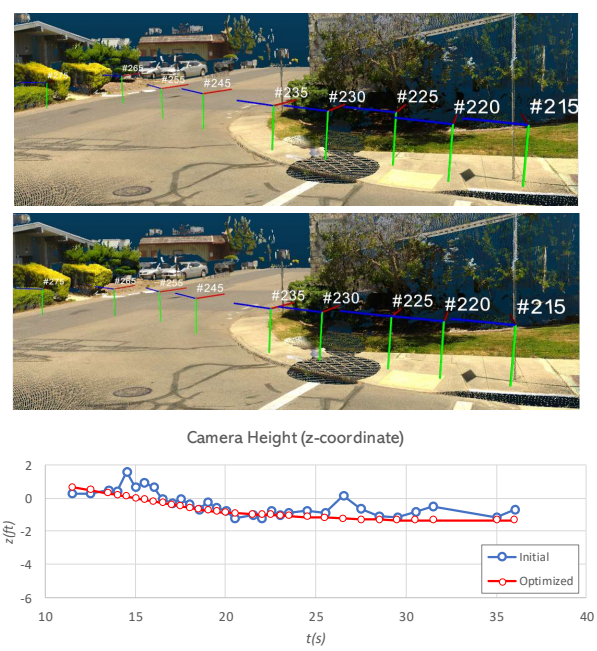

Figure 5. *Camera poses in selected keyframes before (top) and after (bottom) the global optimization. The chart shows the value of the vertical coordinate of the camera pose.*

Figure 6. *Original video and corresponding rendered camera view for several example frames.*

The orientation of the camera is represented using quaternions. Accordingly, we employ the SLERP algorithm for interpolating camera orientations [8], which performs a spherical linear interpolation. SLERP ensures a smooth transition of camera orientation angles between different keyframes.

To determine the actual vehicle path (i.e., of the vehicle center of mass - CG), the transformation between the camera and the vehicle CG is required. The software can then export the path (including position and orientation) of the coordinate system attached to the vehicle CG.

Drive-through Rendering

The interpolated path can now be used to render the drive-through. To validate the path reconstruction, VARS software renders the point cloud/mesh scene with the same camera parameters as the original input video. In our rendering pipeline, we first render the undistorted view of the scene from the virtual camera and then apply the additional distortion mapping to its view (based on the camera distortion coefficients) to emulate the distortion effects of the original camera lens. This produces an output that very closely matches the input video. **Figure 6** shows the output video rendering for a selected set of camera frames of the drive-through. Note that the camera pose for non-keyframes (frames #120, #187, and #257) was interpolated as described previously.

The rendering from the original camera point of view is used to visually verify the fidelity of the reconstructed path. If needed, one can return and correct specific keyframes, optimize the path, and re-interpolate the output. Once the path is validated, the software can generate other viewpoints such as the driver's point of view, top view and vehicle chase view, as shown in **Figure 7**. The drive-through may be simulated with or without a vehicle mesh that is imported as an OBJ Wavefront file.

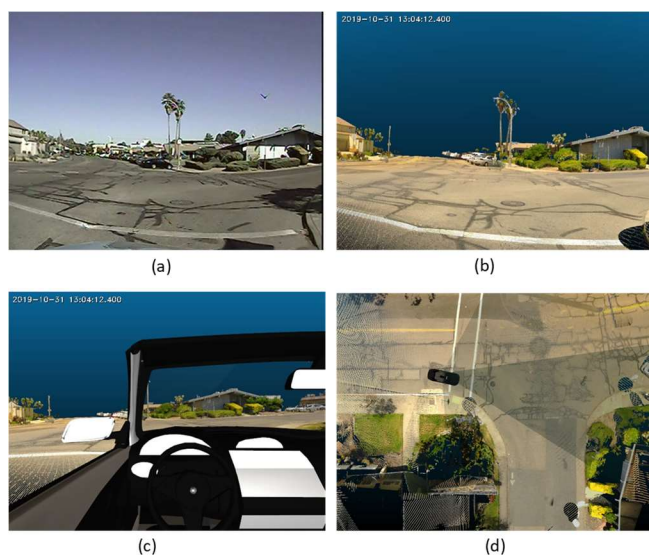

Figure 7. *(a) Original video frame, (b) rendered frame from camera viewpoint, (c) rendered frame from driver's viewpoint (FOV 90˚), and (d) rendered frame from top-view with driver cone of vision displayed.*

Object Tracking

Given the path of the vehicle camera, VARS software can also be used to estimate the position of certain objects captured by the camera. Because the inverse transformation from 2D (video) space to 3D (point cloud) space is indeterminate and can only be performed up to a scale factor, the software maps image plane points to scene points on the ground surface (e.g., traffic cone, parked car tires, pedestrian, obstacle in the road). Accordingly, the intersection of the ray with the ground plane identifies the position of the object.

The accuracy of this estimation depends on several factors, including the fidelity of camera intrinsic parameters, accuracy of the camera pose estimation, the resolutions of the video and point cloud in the region of interest, and the accuracy of point selection. Objects far away from the camera with a smaller pixel footprint result in higher variance of the corresponding 3D ray estimation and leading to a higher chance for error in the position estimation of the object. The variance can be reduced for static targets by marking them in several consecutive frames and determining the average position of the target. This technique is especially useful for parked vehicles.

Figure 8 shows an example of target estimation on a manhole selected in one of the video frames. Given camera intrinsic parameters and the current camera pose, the selected 2D point transforms into a ray in the 3D space (shown in green). The software automatically detects the intersection of the ray with the point cloud and assigns the 3D target point coordinates in the scene. The 3D coordinates can then be exported for further processing. For example, a parked car could be added at the right position in the model of the roadway.

Figure 8. *Example showing target estimation from the video frame. The center of the manhole is selected in the video image. The intersection between the corresponding ray (green line) and ground plane defines the location in 3D space which, in this example, aligns with the actual location of the manhole (red marker).*

Case Study 1

To demonstrate the accuracy of the VARS software in the reconstruction of a vehicle path, we conducted a test drive for comparison with GPS collected data. We also input the reconstructed path to HVE to demonstrate the use of VARS as a pre-processor to an HVE analysis.

We conducted the test drive on Willow Street in Alameda, CA. In the test, a sedan traveled down a 2-lane street, with gentle left and right steering maneuvers introduced along the travel path. The point cloud data was obtained by a laser scanner. A total of 10 scans were collected alongside the street block to cover a distance of about 550 feet (**Figure 9**).

Figure 9. *Output path as recorded by VBOX (left) and VARS (right). The green bars in the left image show the start and end of the segment analyzed with VARS.*

For the video data collection, an Apollo camera (Model RR-CIR236) was mounted on a tripod and placed on the passenger seat of a convertible sedan. The total height of the camera was approximately 6 feet from the ground. The camera's analogue output was recorded using ArcSoft ShowBiz software with a resolution of 720x480 px and frame rate of 29.97 FPS. For the analysis, the video was resampled to 640x480 px and 10 FPS, which is one of the typical output formats for bus videos using the Apollo camera system.

A Video VBOX Lite GPS data logger (RaceLogic Ltd) was used as ground truth. The VBOX video camera was mounted on the windshield to help synchronize the two recordings and the data collection log rate was set at 10 Hz.

The vehicle, driven from south to north, made half a U-turn at the crossroad before coming to a full stop. We present the results for the first segment of the drive while the vehicle was already moving near the boundary of the point cloud scan. **Figure 9** shows the output path as recorded by the VBOX and the path recovered from the video of the drive. The segment of the matching path is highlighted in the figure.

Figure 9 was developed following the analysis steps outlined in the previous discussions, as illustrated by the results in **Figures 3 to 8**.

Figure 10 shows the driven distance from the starting point of the video analysis to the end of the drive. The VARS calculated driven distance was in agreement with the GPS-based measurement by VBOX.

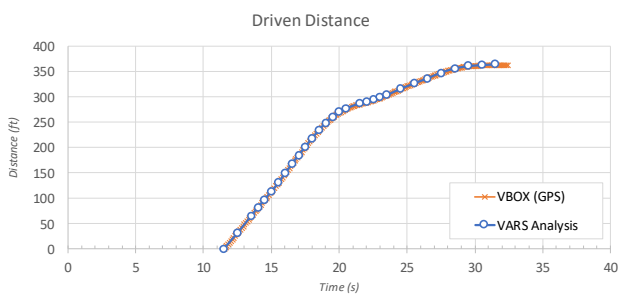

Figure 10. *Comparison of the driven distance as estimated from an analysis using VARS and as measured by VBOX from GPS data.*

The VARS estimated vehicle speed also matched the GPS-based VBOX measurement. **Figure 11** shows a comparison of the speed profiles. The VARS speed was calculated based on the calculated 3D vehicle displacements over known intervals of time. The maximum recorded vehicle speed was 24.9 mph and 24.4 mph for the VARS analysis and the VBOX measurement, respectively.

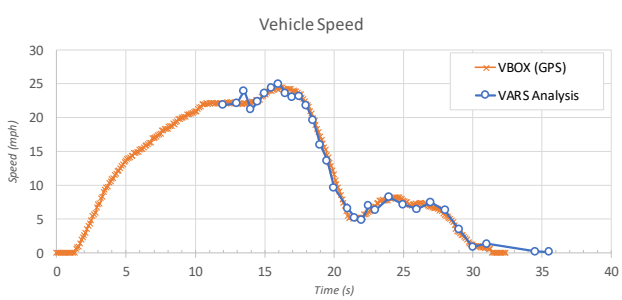

Figure 11. *Comparison of speed profile calculated from VARS analysis and measured by VBOX.*

To demonstrate the use of VARS as a pre-processor to an HVE analysis, we input the calculated VARS vehicle motion (**Figures 9 to 11**) to HVE's Simon solver for a vehicle dynamics analysis. By characterizing the vehicle path with keyframes from VARS, the Driver Model Path Follower option using Simon calculates the necessary driver input to best fit the specified travel path. **Figure 12** illustrates the 8 keyframes from the VARS analysis serving as input to HVE (top figure). The bottom of **Figure 12** shows that the resulting HVE vehicle path compared well with the VARS motion (Path Follower).

Refinement to the HVE driver input improved the matching of the path (Driver Controls) (**Figures 12 to 14**). For example, increasing the steering input increased the yaw angle, which improved the path matching. Further, increased braking toward the end of the drive also improved matching of the vehicle speed.

The need for these refinements could have been the result of the limited resolution of the prescribed vehicle path. In the current version of HVE, the path description is limited to eight keyframes. If additional keyframes could be included as input, the Path Follower prediction could possibly have been improved.

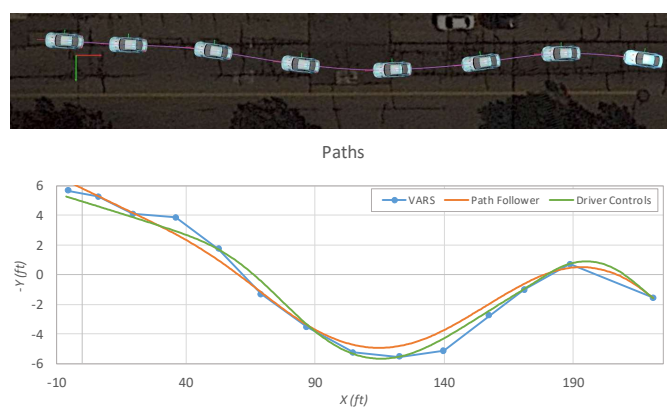

Figure 12*. Keyframes from the VARS analysis to serve as input to HVE (top graph). The bottom graph compares HVE calculated vehicle paths with the VARS path. The "Path Follower" path is the HVE output using VARS keyframes as input; the "Driver Controls" path is HVE output using adjusted driver input. The vertical scale of the bottom graph is amplified to highlight the differences in the paths.*

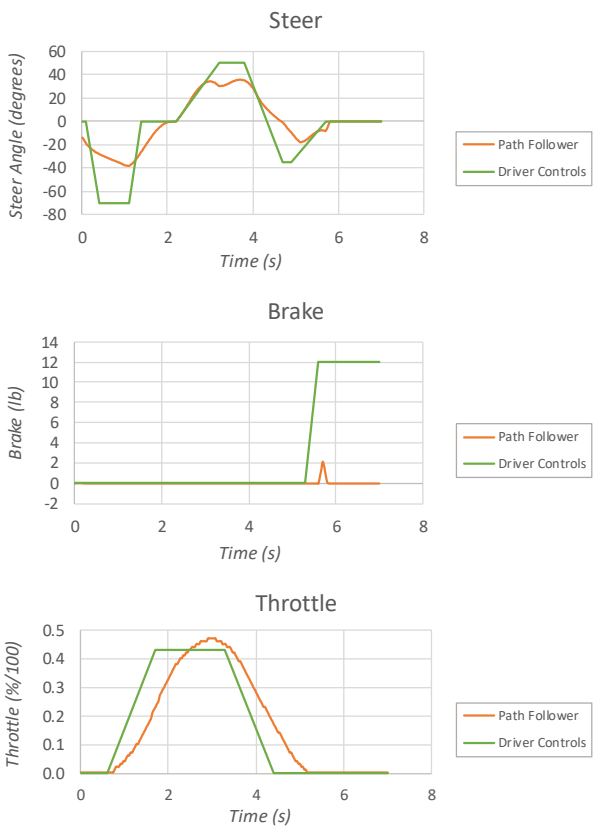

***Figure 13**. Driver input time history as calculated by HVE using the VARS keyframes (Path Follower), and adjusted to improve the matching of vehicle motion in VARS (Driver Controls).*

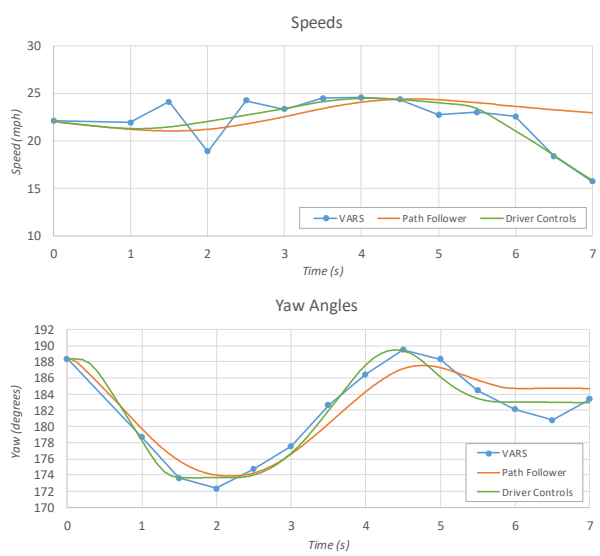

***Figure 14**. Comparison of VARS vehicle motion with that calculated by HVE using path follower on VARS keyframes (Path Follower), and adjusted to improve the matching of VARS vehicle motion (Driver Controls).*

Case Study 2

Case study 2 is a demonstration of the VARS software in the reconstruction of a real world accident. The subject accident involves a sideswipe collision of two vehicles at an intersection in San Francisco. One of the vehicles lost control as a result of the impact, and subsequently collided into a guardrail at the nearest street corner. That vehicle was equipped with an onboard system, which captured the collision sequence on video. The system also captured the speed time history.

In this case study, we analyze the collision sequence using the VARS software, and compare with results of an accident reconstruction performed several years ago. In that reconstruction analysis, the pre-impact motion was semi-quantitatively performed by frame-to-frame analysis. HVE's EDSMAC4 solver was used to simulate the collision sequence.

Figure 15 compares selected output frames from the onboard video, and the corresponding reconstructed views by VARS and EDSMAC4. In our experience, the matching can be achieved to allow for overlay of the reconstructed video with the accident video.

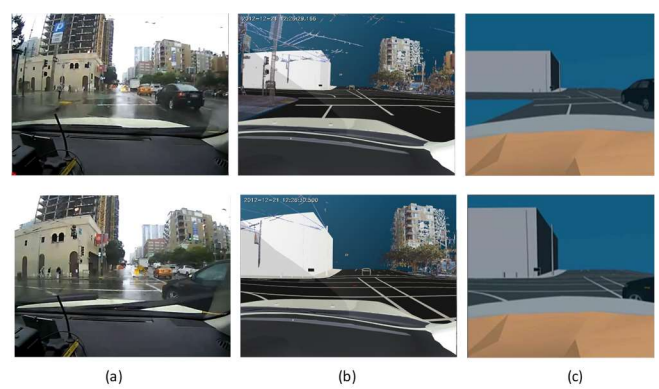

***Figure 15**. Selected output frames from (a) onboard video, and the corresponding reconstructed views by (b) VARS, and (c) HVE EDSMAC4 simulation.*

Figure 16 compares the speed time history of the VARS and the EDSMAC4 analysis with the onboard surveillance system of the subject vehicle. Good matching of the speed profile was achieved up to the impact with the guardrail. In the EDSMAC4 analysis, a rigid barrier was used to model the guardrail, which resulted in a velocity change at impact over a shorter duration as compared to the onboard speed data.

Reducing the impact stiffness of the barrier would have resulted in better matching with the speed data post-impact.

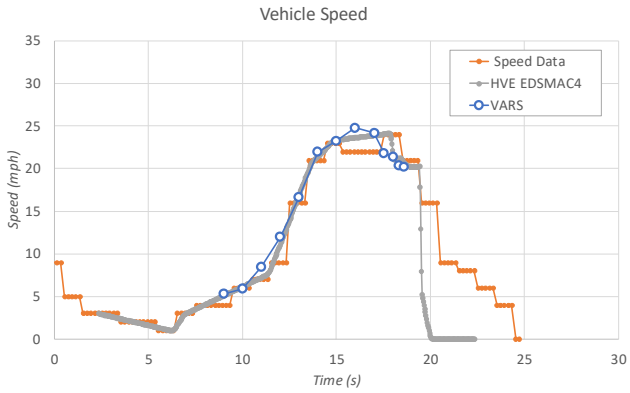

Figure 16. *Comparison of speed profile calculated from VARS analysis, HVE analysis, and the vehicle surveillance system.*

The predicted vehicle paths by VARS (green line) and EDSMAC4 (white vehicle depicted in 1 second intervals) are compared in **Figure 17.** The semi-quantitative estimate of the pre-impact position follows the calculated VARS path well. In future HVE analysis, use of VARS to establish pre-impact vehicle motions would further improve the overall accuracy of the simulation.

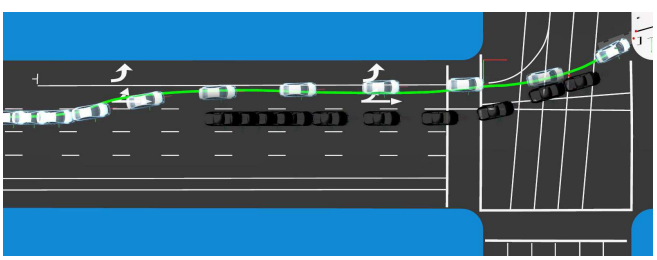

Figure 17. *Comparison of vehicle travel path as calculated by VARS (green line) and HVE EDSMAC analysis (vehicle position depicted at 1 second intervals).*

Conclusions

This study presents Video-based Accident Reconstruction System (VARS), a software tool developed for the motion analysis of traffic accidents captured by vehicle and land surveillance cameras. Using a photogrammetric approach for a single video input and a point cloud of the environment, the VARS software tool has demonstrated its capability to produce an accurate vehicle travel path and its corresponding speed profile. The VARS software is capable of reproducing onboard video views, while matching vehicle motion data as measured by a GPS-based data logger.

The VARS analysis specializes in recreating the kinematics (geometry) of vehicle movement. Coupled with vehicle dynamics software such as HVE, the VARS/HVE combined analysis is capable of recreating vehicle dynamics in reconstruction of real world accidents. We have demonstrated that this can be accomplished through use of the VARS output as input to vehicle dynamics simulation tools.

In general, the extracted speed accuracy depends on the alignment of the camera poses in the keyframe, density of the keyframe segmentation, video frame rate, and vehicle movement between the keyframes. Speed data could also be extracted from the interpolated frames to provide higher sampling density and averaged across multiple frames to further smooth the noise.

The development of VARS is ongoing. The functional features implemented to date have been used successfully in the analyses of real world accidents. The current implementation supports a single video input. Future development will combine video data from multiple cameras to enhance the ability to accurately recreate vehicle movement. Additional future development includes improvements in areas such as camera calibration and user interaction with point cloud data (e.g., view navigation, keypoint selection).

References

1. Bradski, G., *The OpenCV Library*, Dr. Dobb's Journal of Software Tools (2000). URL: https://opencv.org
2. Point Cloud Library (PCL), URL: http://www.pointclouds.org
3. Ma, Y., Soatto, S., Kosecka, J., Sastry, S.S., *An Invitation to 3-D Vision: From Images to Geometric Models*, Springer-Verlag, New York, NY, 2004
4. Tsai, R.Y., *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses*, IEEE Journal of Robotics and Automation, RA3(4): 323–344, 1987

5. Fischler, M.A., Bolles, R.C., *Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography*, Communications of the ACM, 24(6):381–395, 1981
6. Moré, J.J., *The Levenberg-Marquardt algorithm: Implementation and theory," Numerical Analysis*, pp. 105-116. Springer, Berlin, Heidelberg, 1978
7. Eberly, D., *Kochanek-Bartels Cubic Splines (TCB Splines)*, 2008, URL: https://www.geometrictools.com/Documentation/KBSplines.pdf
8. Shoemake, K., *Animating Rotation with Quaternion Curves*, Proceedings of SIGGRAPH, 19(3):245-254, 1985
9. Engineering Dynamics Corporation, HVE Software, URL: http://www.edccorp.com/products/hve.html