Chapter 9
Vehicle Tracking for Bridge Load Dynamics Using Vision Techniques

Ryan Brown and Al Wicks

Abstract Structural health monitoring for bridges is an important field that is growing in necessity in the United States with the aging of the interstate and highway system. Most health monitoring systems rely on detecting the motion of the bridge through strain gauges, accelerometers and GPS units. These sensors are very good at measuring the output motion of the bridge, but do not take into account the input signal from the vehicles. Adding the ability to directly measure the location of the input forces on the bridge would improve the ability to model the bridge dynamics. In this paper we propose a system that can identify a vehicle on a bridge and track its location through multiple video frames. Previous work in vehicle tracking has focused on traffic pattern research but has not been adequately translated into a sensing application for structural dynamics. The algorithm was tested and the results show that vehicles are able to be tracked along a bridge with acceptable error in the location output. This method allows a researcher to provide a dynamic input load to his model, rather than estimating or using some load distribution. Combining this with the structural sensing on the bridge will allow for more accurate modeling of the bridge dynamics.

Keywords Mechatronics • Computer vision • Dynamic structural loads • Vehicle tracking • Background segmentation

9.1 Introduction

The field of structural health monitoring is rapidly growing in both publications and necessity. A cursory keyword search of 'Structural Health Monitoring' shows an exponential growth in the number of publications containing the phrase since 1985 [1]. The current state of the art in bridge health monitoring focuses on measuring the vibrations and displacements in the bridge structure typically using accelerometers, GPS units, temperature sensors and strain gauges [2–4]. These sensor are well studied and can be arranged in a multitude of ways to gather precise and useful data. However, none of these sensors measure solely the input forces on the bridge. All of their measurements are of bridge vibration due to the input forces from vehicles. The ability to know the location of the input force on the bridge, especially in real world conditions with multiple vehicles at multiple speeds, is not feasible with the listed sensing packages.

Our system allows for the measurement of the location of multiple vehicles moving at multiple speeds and directions on the bridge. With this information, the measured vibrations and displacements of the bridge can be linked with a known input, which can lead to better modeling and more accurate monitoring of the bridge.

9.2 Vehicle Tracking Algorithm

9.2.1 Types of Vehicle Detection Schemes

Tracking a vehicle on a roadway has been studied and accomplished using a variety of techniques. The techniques fall into two main categories: Shape detection/Template matching or Background-foreground segmentation. Shape Detection or Template Matching is the process of attempting to search for car-like shapes in an image and track those shapes. Template matching is typically used in scenarios where the type of vehicle is important information [5, 6]. The basic process is to generate a library of vehicle templates and then convolve them with the image. The higher the response of the convolution at

R. Brown (✉) • A. Wicks
Department of Mechanical Engineering, Virginia Tech, 445 Goodwin Hall, 635 Prices Fork Road, Blacksburg, VA 24061, USA
e-mail: brownrc@vt.edu

A. Wicks, C. Niezrecki (eds.), *Structural Health Monitoring, Damage Detection & Mechatronics, Volume 7*,
Conference Proceedings of the Society for Experimental Mechanics Series, DOI 10.1007/978-3-319-29956-3_9

a particular point, the more likely that particular vehicle exists at that point. The main drawback of a template matching scheme is the necessary processing time to convolve several different template across each video frame.

Background-Foreground Segmentation is the process of detecting the statistical properties of pixels in an image over time to create a model of what the background of that image should be [7, 8]. This allows the detection of deviations from that model and can be used to segment the image into a foreground-background set. This method is faster than the template matching method, but it does not detect the type of vehicle, nor if the motion is actually a vehicle. It merely detection deviations from an iteratively updating background model. However, in the context of roadways, most background deviations will be vehicles so this limitation is not significant. Another limitation is that the viewed scene needs to remain mostly static across multiple frames as the background model is iteratively built and updated. As the target use for this algorithm is a stationary camera over a roadway, this limitation is also not an issue. Based on the features of these two methods, we chose to use a background-foreground segmentation to identify the vehicles in the roadway

9.2.2 *Overview of Vehicle Location Algorithm*

The proposed vehicle location algorithm has two main components. First, the bridge image is marked by the researcher with a region of interest (ROI) and the width and length of that region is provided. This information is used to create a projective transformation between the bridge image and the defined ROI. This transformation is used to project the locations of the detected vehicles to coordinates inside the rectangular ROI. This is discussed in more detail in Sect. 9.2.3.

Once the transformation has been found, the frames are processed one by one with the background subtraction method to identify the vehicles in the image. The vehicle contours are found using a binary image contour method and then their centroids are calculated to find the location of the vehicle in the original image. Once these locations are found, the transformation from the previous step is used to project the points into the new bridge coordinates. These coordinates can be viewed and/or saved for later analysis. Figure 9.1 shows the flow of the algorithm. The algorithm was written using C++ and the OpenCV library [9].

9.2.3 *Bridge Projective Transformation*

To appropriately transform the perspective bridge images to a rectangular image, a projective transformation matrix must be created. This is a 3×3 matrix that transforms a set of points in one plane to another plane. First, eight points defining two arbitrary quadrangles are chosen. These points are used to find the transformation from one quadrangle to another. The points must be converted to homogenous coordinates, where

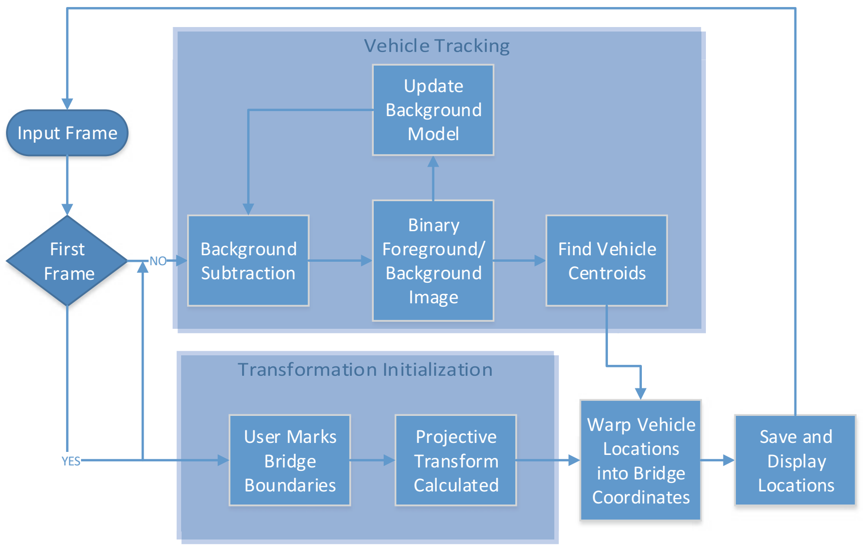
Vehicle Tracking
Update Background Model
Input Frame
First Frame
NO
Background Subtraction
Binary Foreground/ Background Image
Find Vehicle Centroids
Transformation Initialization
YES
User Marks Bridge Boundaries
Projective Transform Calculated
Warp Vehicle Locations into Bridge Coordinates
Save and Display Locations

Fig. 9.1 Vehicle tracking algorithm flowchart

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Once the location points have been converted, a set of linear equations can be solved that will result in a 3×3 transformation matrix. This matrix can be used to transform one homogenous coordinate to its corresponding point in the other quadrangle.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} wx' \\ wy' \\ w \end{pmatrix} \rightarrow \begin{pmatrix} x' \\ y' \end{pmatrix}$$

In the above equation, A is the transformation matrix, (x, y) are the original coordinates to be transformed and (x', y') are the resulting transformed coordinates. This transformation is generated from the bridge image once and the same transformation is applied to each vehicle location to warp it onto the bridge coordinates.

Some limitations that this transformation method imposes on the system are:

1. The bridge must not curve inside the ROI
2. The bridge must not significantly change height inside the ROI

The method will not work on bridges that fail either of these two conditions. Methods of removing these limitations are discussed below in Sect. 9.5.

9.2.4 *Background Subtraction and Vehicle Tracking*

The particular background subtraction method chosen is a Mixture of Gaussians method. This method assumes that each pixel's intensity can be modeled by Gaussian mixture model. This model is updated with each frame to account for changes in the background scene. Some assumptions made by this model are that there is more background visible than foreground, and that the modes of the background intensity values do not have very high variance [10]. The original version of this background segmentation method was developed by Stauffer and Grimson [11], and has been expanded and refined since. The particular method used in this experiment was developed by KaewTraKulPong and Bowen and includes a refinement in the accuracy and the ability to detect shadows from foregrounds objects and omit them from the detected foreground [12].

The method assumes that the probability of a certain pixel having a certain value at some time is a mixture of N Gaussian distributions. These distributions are controlled by previous values of that pixel. If the predicted pixel value differs by some threshold from the detected pixel value, then the pixel is marked as foreground. The value is then incorporated into the background model. This process is repeated for each pixel and the models updated with each frame.

9.3 Experiments

To evaluate the performance of the algorithm, simulated road data was created in software using computer generated roadway and vehicle images. This data allowed a comparison of the algorithm generated locations against ground truth locations. This was not possible using real data, as there was no access to the necessary information using a real bridge or roadway. The images from road cameras were already in perspective, so there was no ability to generate ground truth vehicle locations on the bridge from that data. In Sect. 9.5, methods of testing this algorithm with real data are discussed. Figure 9.2 shows a transposed sample frame from the simulated data. For all the simulated data, consider the X coordinate to be the width of the bridge from left to right in the image and the Y coordinate to be the length of the bridge from the top of the image to the bottom, shown in the arrows on Fig. 9.2.

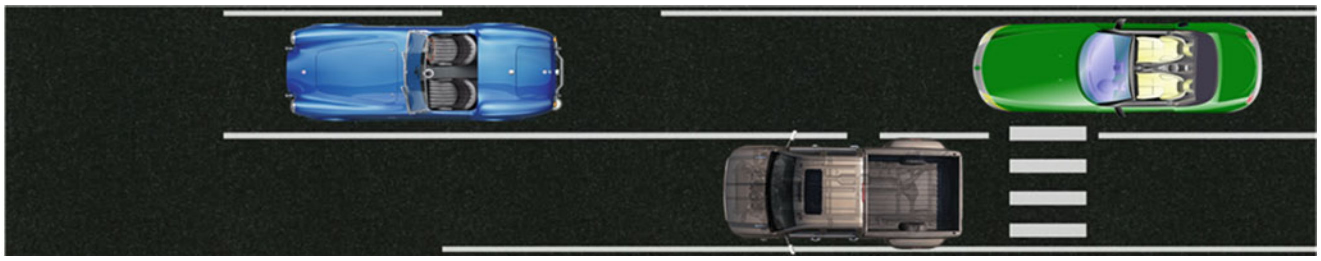

Fig. 9.2 Simulated road data

9.3.1 Simulated Data Creation

The simulated data was created using the OpenCV C++ library. The vehicles were moved along the simulated roadway at different speeds and the image was warped to different shapes to simulate different camera positions. Some of the different camera angles are shown in Fig. 9.3. These different simulated camera angles were used to verify the performance of the algorithm and determine if there was growth in error relative to any of the particular camera orientation variables. Five thousand frames were analyzed for each of the different camera angles were created with multiple vehicles in most frames moving at different speeds.

9.3.2 Experiments on Simulated Data

Each simulated data set of 5000 frames was used in tests with different ROIs on the bridge. The ROIs were generated at different starting and ending locations along the bridge. Markings at even increments were used to identify the length of each ROI. The projective transformations were calculated and the X-Y location errors were stored for the runs. As part of each experiment, to determine the length along the bridge that the vehicles have moved, the width of the lower part of the ROI was calculated in pixels. The width of the actual bridge is known and can be used to create a pixel per foot value for each ROI width. This value is used to scale the calculated X-Y locations onto the ground truth X-Y locations.

9.4 Results of Experiments

The most important result of the experiments is the overall error in the X-Y location. The width of each bridge is set at 28 ft. The error in the X direction was negligible, less than ±0.5 ft. across 28 ft. The error in the Y direction is more pronounced. This is discussed further below.

9.4.1 Accuracy of Location

Focusing on the error in the Y direction, histograms for all runs on the three bridges shown above in Fig. 9.3 are shown below in Fig. 9.4.

The histograms are shown with lines at the standard deviation and twice the standard deviation of the data distribution. Most of the mass of the histogram is within the first standard deviation, and most of the mass of the histogram is within ±5 ft of error. Based on the length of a typical mid-size sedan, the Toyota Camry, which is 15.8 ft., the majority of the error in the Y dimension is confined to within the bounding box of a typical vehicle. There is mass in the histograms outside the ±8 ft. that would keep the error within the vehicle bounding box, but the probability of that occurring is much smaller than the alternative.

The standard deviation in the Bridge 3 dataset is altered based on some mass appearing in the −20 to −10 error range. This mass is caused by errors in the background subtraction method and vehicle contours appearing in areas that they are not,

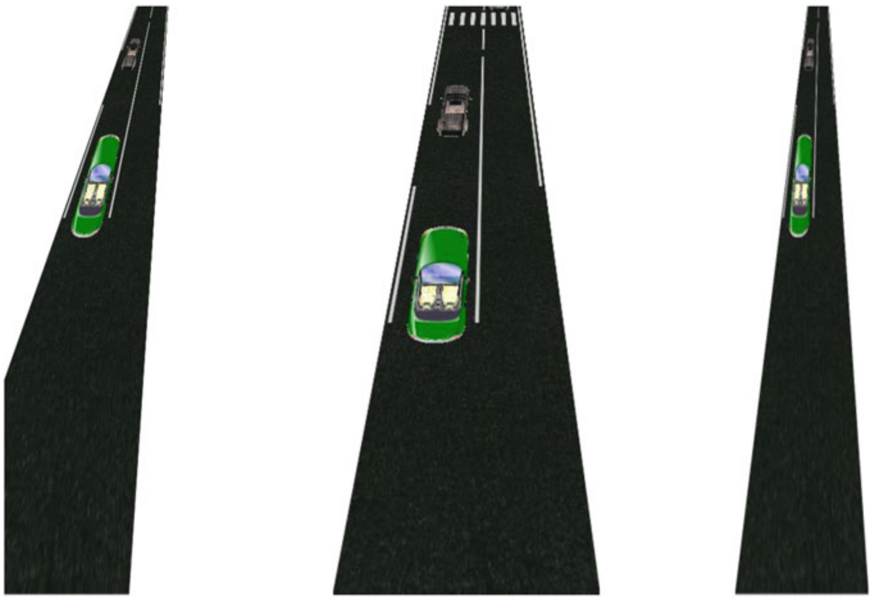

Fig. 9.3 Examples of warped simulated data (bridge 1, 2 3 from *left* to *right*)

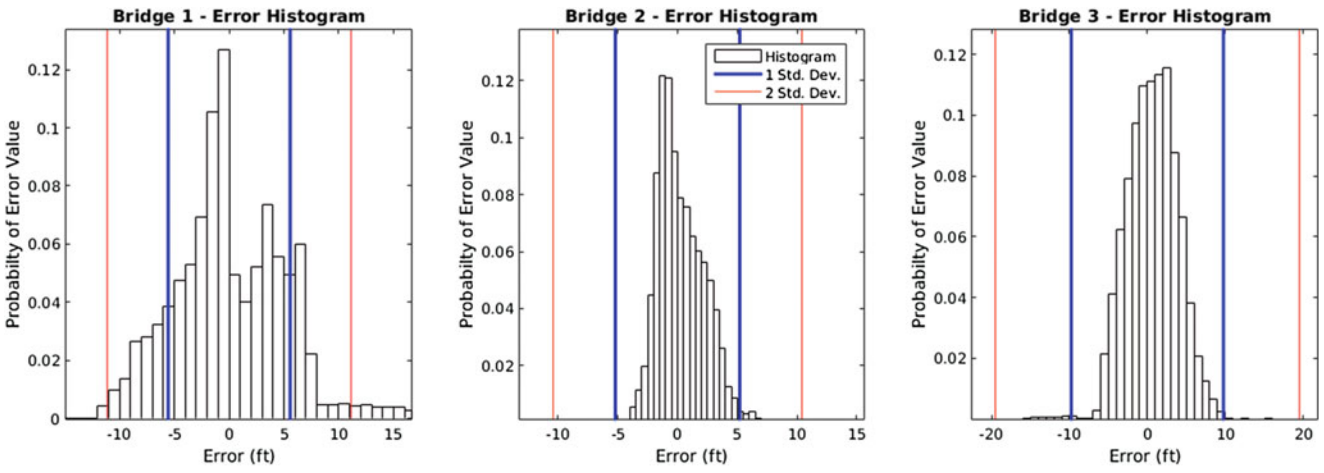

Fig. 9.4 Histograms of Y error normalized for probability

or contours being split into two separate blobs. Either way, the probability of that error is much smaller than the probability of the typical distribution of error that all exists within the ±10 ft. range. To eliminate these errors is a feature for future work and is discussed below in Sect. 9.5.

9.4.2 *Types of Error and Potential Causes*

Figure 9.4 shows a good overview of the errors that exist in the system, but looking closer at individual data runs gives more insight into the type of error that occurs. Figure 9.5 shows a set of data runs where single vehicle tracks are easily seen and separated. The error is not randomly distributed around the desired value, but instead has a relationship with the distance the vehicle has traveled down the simulated bridge. As the vehicle travels down the bridge, the error first decreases and then increases. This distribution of error has to do with the length of the bridge, the location of the ROI on the bridge and the accuracy of the user input. Figure 9.6 shows the same information as Fig. 9.5, but for multiple ROI on the same bridge. Different ROI cause different amounts of error. The slope of each vehicle “error line” scales with the maximum and minimum error in that ROI.

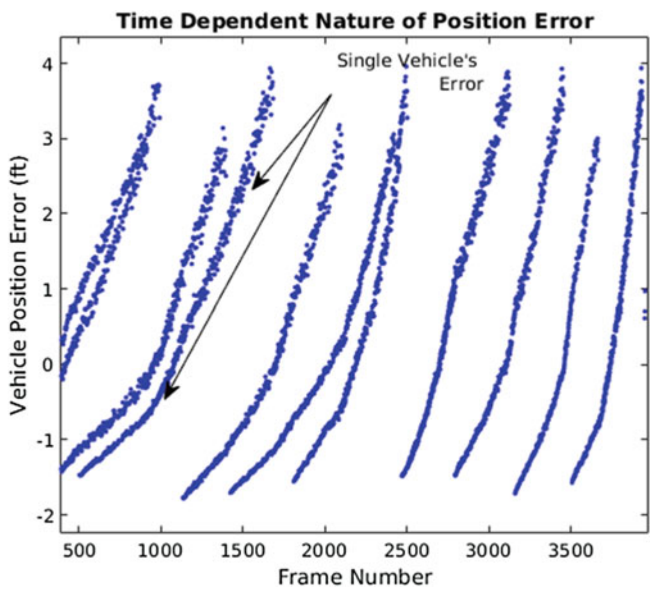

Fig. 9.5 Single vehicle data runs and associated error

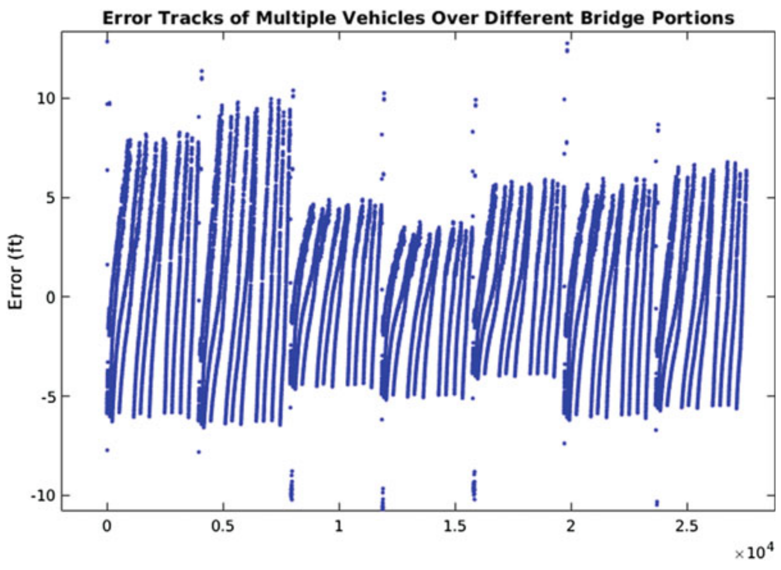

Fig. 9.6 Multiple data runs with different ROI

To determine the cause of the error, the traits of the ROIs were compared against the error to see if there was any correlation between them. The length of the ROI did not correlate to the error in the vehicle location, but the pixels per foot value calculated from the lower width of the ROI had a higher degree of correlation. These two values are plotted in Fig. 9.7. The decreasing trend in pixels per foot as error increases is noticeable, but not extreme. However, this finding makes sense based on the operation of the algorithm. The pixels per foot value is used to translate the pixel location of the X-Y coordinate to a location in feet. If this value is off, the location has a higher degree of error. With a smaller pixel per foot value, a single pixel deviation from the actual value can cause more error, as it is a higher percentage of the overall value. This leads to more error in the ROIs that are further from the base of the image.

9.5 Conclusions and Future Work

The information from Sect. 9.4.2 leads us to say that the best usage case for this method is a system where the ROI is near the bottom of the image, and the camera is aligned closer to the centerline of the roadway, but the errors present in the cases where this is not true are still not so great as to prevent the system from being useful for data collection. The largest limitation

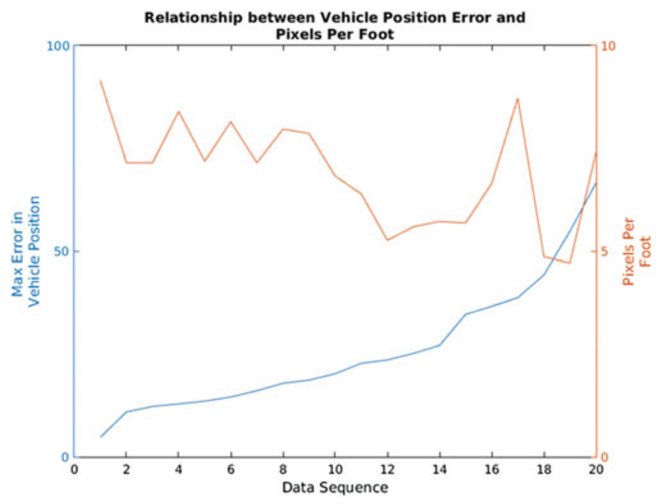

Fig. 9.7 Pixels per foot and vehicle location error

in the current state of the system is the inability to work with non-rectangular bridges. The system as it is currently cannot handle more than minimal curvature in the roadway. Also, it assumes that the bridge height does not significantly change over the ROI. These assumptions prevent the algorithm from analyzing a significant portion of highway bridges.

The curvature limitation can be overcome by calculating the transformation based on the camera location instead of an ROI on the bridge. If the camera location is known relative to a known point on the bridge, a projective transformation can be calculated that will perform the same transformation as the ROI method discussed above. This is not an unrealistic expectation to know the camera position, and in a real data scenario this would be the transformation generation method of choice.

The height limitation is harder to overcome. The camera has a much more difficult time distinguishing the height of an object from its distance from the camera. This could be accomplished using some relationships between the angles of the two edges of the bridge, but the relationships would need to be researched before they could be implemented.

This system can be tested using real road data if there was a way to identify the ground truth location of the vehicles on the road. Attaching GPS monitoring devices to vehicles driving over an instrumented bridge would allow for real world testing of the algorithm. This would be a "plug-and-play" style testing, as the same algorithm as used on the simulated data could be used without any changes on the real world data.

Eliminating the errors in the background segmentation is not a high priority item, as they contribute very small percentages of the overall error in the system. However, a region growing step would prevent the vehicle contours from separating and creating false vehicle blobs in the image. The effects of this on the accuracy of the detected locations would need to be studied.

In the cases where it is not limited by assumptions, the system performs well and is able to resolve the locations of the simulated vehicles to within ±5 in most cases and ±10 ft at worst. This error is within the bounding box of most vehicles that would be detected on the highway, apart from very small vehicles like Smart Cars or motorcycles. As these are the minority on the road, for most situations, this error is acceptable.

References

1. Google Books N-gram Viewer [Online]. http://books.google.com/ngrams. Accessed 01 Jan 2015
2. Hu, X., Wang, B., Ji, H.: A wireless sensor network-based structural health monitoring system for highway bridges. Comp. Aid. Civil Infrastruct. **28**(3), 193–209 (2013). Retrieved from http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8667.2012.00781.x/full
3. Hackmann, G., Guo, W., Yan, G., Sun, Z., Lu, C., Dyke, S.: Cyber-physical codesign of distributed structural health monitoring with wireless sensor networks. IEEE Trans. Parallel. Distrib. Syst. **25**(1), 63–72 (2014). http://doi.org/10.1109/TPDS.2013.30
4. Magalhães, F., Cunha, A., Caetano, E.: Vibration based structural health monitoring of an arch bridge: from automated OMA to damage detection. Mech. Syst. Signal Process. **28**, 212–228 (2012). http://doi.org/10.1016/j.ymssp.2011.06.011
5. Li, T., Li, D.M., Dou, Y.M.: A novel vehicle type recognition based on template matching. Adv. Mater. Res. **945–949**, 1856–1860 (2014). http://doi.org/10.4028/www.scientific.net/AMR.945-949.1856

6. Thiang, A., Lim, R.: Type of vehicle recognition using template matching method of the International Conference on Electrical. (2001). Retrieved from http://faculty.petra.ac.id/thiang/download/paper/Pengenalan_Mobil_P017.pdf
7. Yao, L., Ling, M.: An improved mixture-of-Gaussians background model with frame difference and blob tracking in video stream. Sci. World J. **2014**, 1–9 (2014). http://doi.org/10.1155/2014/424050
8. Oh, J.-S.: Improved MOG algorithm based on adaptive threshold for effective background classification. TECHART **1**(4), 54 (2014). http://doi.org/10.15323/techart.2014.11.1.4.54
9. Bradski, G.: OpenCV. Dr. Dobb's journal of software tools (2000)
10. Power, P., Schoonees, J.: Understanding background mixture models for foreground segmentation. Proceedings of Image and Vision, pp. 267–271 (2002). Retrieved from http://kauri.auck.irl.cri.nz/~johanns/publications/MOG-fg-seg-ivcnz02-preprint.pdf
11. Stauffer, C., Grimson, W.: Adaptive background mixture models for real-time tracking. Proceedings of the 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149), pp. 246–252 (1999). IEEE Comput. Soc. http://doi.org/10.1109/CVPR.1999.784637
12. KaewTraKulPong, P., Bowden, R.: An improved adaptive background mixture model for real-time tracking with shadow detection. In: Video-Based Surveillance Systems, pp. 135–144. Springer US, Boston, MA (2002). http://doi.org/10.1007/978-1-4615-0913-4_11