International Journal of Innovative Computing, Information and Control
Volume **11**, Number **5**, October **2015**

ICIC International ©2015 ISSN 1349-4198
pp. 1667–1680

AN APPLICATION OF IMAGE PROCESSING IN VEHICLE CRASH MOTION DETECTION FROM HIGH FRAME RATE VIDEO

Sondre Sanden Tørdal[1], Andreas Klausen[1], Hamid Reza Karimi[1]
Kjell G. Robbersmyr[1], Mladen Ječmenica[2] and Ole Melteig[2]

[1]Department of Engineering
Faculty of Engineering and Science
University of Agder
Jon Lilletunsvei 9, N-4879 Grimstad, Norway
hamid.r.karimi@uia.no

[2]Department of Technology
Institute of Process Technology
Telemark University College
Kjølnes Ring 56, N-3918 Porsgrunn, Norway

Received December 2014; revised May 2015

Abstract. *During real-life crash tests it is common to capture high frame rate (HFR) videos to observe the vehicles motion. This motion can also be measured by means of accelerometers and gyroscopes installed in the center of gravity of the vehicle. Both rotational and translational data are captured from the HFR videos by using two well-known image processing algorithms; the Hough transform and template matching. The motion of a bus in an oblique crash and a Ford Fiesta in a pole crash is extracted from HFR videos and compared to real measurements obtained from accelerometers and gyroscopes.*
Keywords: Hough transform, Template matching, Vehicle crash, Oblique crash, High frame rate video, Image processing

1. **Introduction.** In the context of transportation safety, the motion detection of a vehicle during a crash test is of high interest, since such information can help us have a better understanding of the vehicle crash. The crash itself occurs in a very short time-interval and is referred to as a crash pulse [1]. To measure the motion of the vehicle, equipment such as accelerometers and gyroscopes can be used to collect data. However, the vehicles motion can also be detected by examining high frame rate videos (HFR) captured during the crash. This video can be processed using image processing techniques to detect lines or track features of the vehicle during the crash. Lines are detected using a Hough transform (HT) algorithm while templates are identified by using a template matching (TM) algorithm.

The motivation for extracting the motion of the vehicle during the crash is to establish reliable simulation models of the vehicle crash from the motion data. In literature there are mainly two main categories of modeling and simulation of a vehicle crash: lumped parameter model (LPM) and finite element method (FEM). The LPM technique may consist of up to several masses connected together by means of springs and dampers. To obtain all the spring and damper parameters, real life crash data is used to optimize the model. Jonsén et al. [2] used an LPM model and identified the model parameters for the spring and damper using measured crash data. The LPM model parameters can also be identified by using optimization algorithms. Kim et al. [3] used an optimization algorithm to determine the LPM parameters. Pawlus et al. [4] presented an analytical method to determine the LPM parameters of a simple mass-spring-damper model (also known as the Kelvin Model [5]) which uses features of a crash pulse. Munyazikwiye et al. [6] presented a state-space mathematical model to investigate the frontal crash of a Ford Fiesta. Karimi et al. [7] presented a novel wavelet-based approach to reproduce the acceleration pulse of a vehicle involved in a crash. The second category uses FEM models to simulate the vehicle crash. Zaouk et al. [8] introduced an FEM model of a vehicle to simulate several crashes and compared the simulation results with real life crash data.

Real life crash measurements are necessary to validate the simulated results which are obtained from the LPM or FEM approaches. Such validation can be done by installing the measuring devices such as accelerometers and gyroscopes inside the vehicle. This equipment is expensive and time consuming to install and calibrate. Another issue is introduced from the high kinematic forces affecting the equipment during the crash. The equipment has to withstand these kinematic forces and still present reliable results. Therefore, it is desirable to investigate other techniques to measure the motion of the vehicle during the crash, such as image processing of HFR videos captured during the crash test. A popular method for object tracking is to use the Hough transform (HT) to detect edges or lines in a time sequence of images, and correlate those lines with an object. This information can be used to obtain the angle of the object during the crash. Fernandes and Oliveira [9] state that the HT is a popular tool for line detection due to its robustness to measurement noise and missing data. The HT has also been used for real-time line detection, which states the efficiency of the HT to process a huge amount image frames from the HFR videos. Ji et al. [10] also state that line detection with the HT is used as a frequent tool in high level object detection. Furthermore, Greenspan et al. [11] used the HT to detect objects in a time sequence of images.

Other researchers have detected the speed of the vehicle using image processing, mainly with template matching techniques. In [12] Jhumat presented a method to determine the speed of a vehicle by capturing video frames of the vehicle. By identifying the centroid of the car in the image using threshold and morphological operations on the mask of the car, the speed was calculated using the difference between position in the images. Ibrahim et al. [13] used object tracking to determine the vehicle speed by counting the number of frames a vehicle uses to pass by the scene. Doğan et al. [14] proposed a procedure to track the speed of a vehicle by tracking several points (up to 52 points) on a vehicle. The mean velocity of these points was calculated to be the vehicle speed. Jurie and Dhome [15] conclude that the TM can be used efficiently in real time on low cost hardware. The TM algorithm is also robust to occlusions and illumination changes. Besides, Prabhakar et al. [16] succeed to track moving objects from a surveillance camera using the TM algorithm.

This paper is an extension based on previous research work [17], where the HT was used to detect the roll and yaw angles of a vehicle during an oblique crash. As inspired by [14] where the speed of a vehicle was extracted from a video surveillance camera by tracking several features, the proposed feature tracking algorithm in this paper can also track the pitch angle of the vehicle from the angle between two features. This algorithm is used on two HFR videos of vehicle crashes to extract the vehicle speed and pitch angle after the crash. The performance is verified by comparing the results to the measured motion obtained through gyroscope and accelerometer data. The motivation for developing the presented algorithms is to extract more information from a vehicle crash based on a high frame rate video.

2. **Hough Transform.**

The Hough transform [18] is a statistical algorithm which is suitable for extracting features in images such as straight lines. This algorithm is a voting process which compares all possible lines in an image with $n \times m$ pixels. The algorithm returns the indicated lines in the image, based on two predefined thresholds set by the user. The first threshold sets a lower boundary for the minimum length of the indicated Hough lines. The second threshold describes the upper boundary for the gap length between two detected lines. If this gap length is smaller than the boundary, it will merge the two lines together. The algorithm will define the indicated lines in the image as follows:

$$\rho = x \cos\theta + y \sin\theta \tag{1}$$

where (x, y) are the measured coordinates in the image, θ $(-\pi/2 \le \theta \le \pi/2)$ is the angle between the normal axis and the x-axis, and ρ is the normal distance from the origin. Figure 1 illustrates the geometric relationship between the indicated line segment and the reference coordinate system where O indicated the origin.

Points $P_1(x, y)$ and $P_2(x, y)$ are also shown in Figure 1. These points are used to specify the two endpoints of the indicated finite Hough line. This information is useful when calculating the angle between the indicated line segment L and the x-axis. However, MATLAB Image Processing Toolbox provides some predefined functions for the Hough transform which can be used to extract the angle of yaw and roll motions of the vehicle as a function of time.

Example: Edge Line Detection of a Square. The Hough transform can easily be explained by using a simple example where four lines should be detected at the edges of a square which is rotated 40° clockwise. The rotated square is shown in Figure 2(A), both with and without the red indicated lines.

The result of the Hough transform can be illustrated as a histogram containing the four indicated Hough peaks as shown in Figure 2(B).

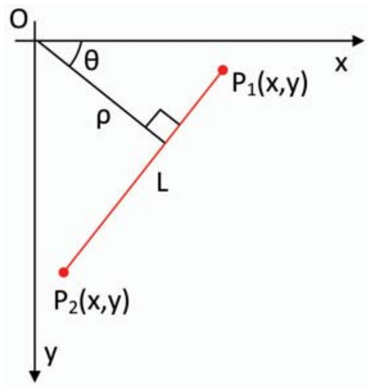
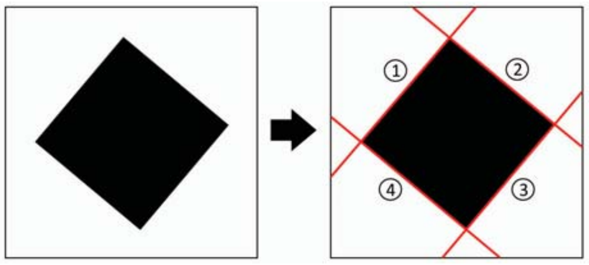
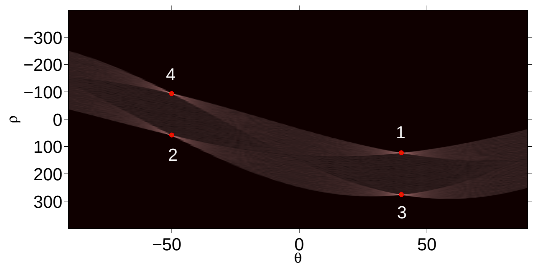


FIGURE 1. Geometric relationship between the indicated line and origin



(A) Hough line detection of rotated square using the Hough transform



(B) Histogram of the Hough transform with indicated Hough peaks

FIGURE 2. HT example

These four Hough peaks are represented with numbered red dots in the histogram, where the numbers represent the four lines indicated at the edges of the rotated square shown in Figure 2(A). Each of the four peaks is located at given values for θ and ρ which corresponds to the coordinate system shown in Figure 1.

3. Bus Oblique Crash.

3.1. **System description.** During the oblique bus collision, HFR videos were captured from three different positions: from the front, the back and the top. The camera from the back contains lower quality video than the camera from the front. Therefore, the videos captured from the top and front are used to detect the global angles of the bus during the oblique impact. A simple illustration of the test setup is shown in Figure 3, where the bus, the two cameras and the local and global coordinate systems are illustrated.

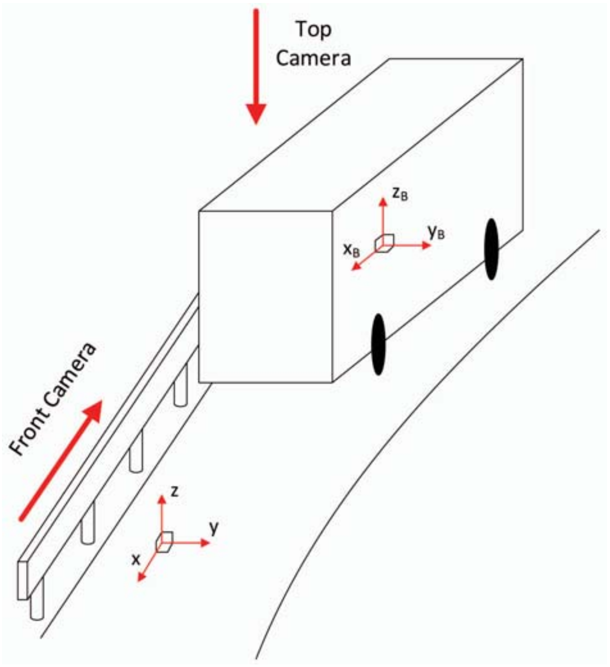


FIGURE 3. Bus impact test set up with HFR camera directions

Both videos are captured with a frame rate of 250 frames per second (FPS), which should ensure that the angles measured from the videos will be of high resolution.

3.2. **Detection implementation.** The HFR video captured of the bus during the oblique impact is processed with MATLAB's Image Processing Toolbox and the Hough Transform function. Before the angle detection by the Hough transformation can start, the video has to be converted to a preferable video format. The MPEG-4 format is chosen since this format is supported by MATLAB's video import function *VideoReader* and because MPEG-4 is less space demanding than uncompressed formats.

It is preferable to convert the video to grayscale video before the movie processing. This is done by using the *rgb2gray* function in MATLAB. The result of the converting and import process in MATLAB is shown in Figure 4, where also the area of interest is indicated with a red rectangle.

By isolating a small region of interest, as shown in Figure 4 the amount of image information is reduced significantly. This isolation is necessary to remove other straight lines which may have the same angle as the bus during the video. The resulting frame which is cropped to isolate the field of interest is shown in Figure 5.

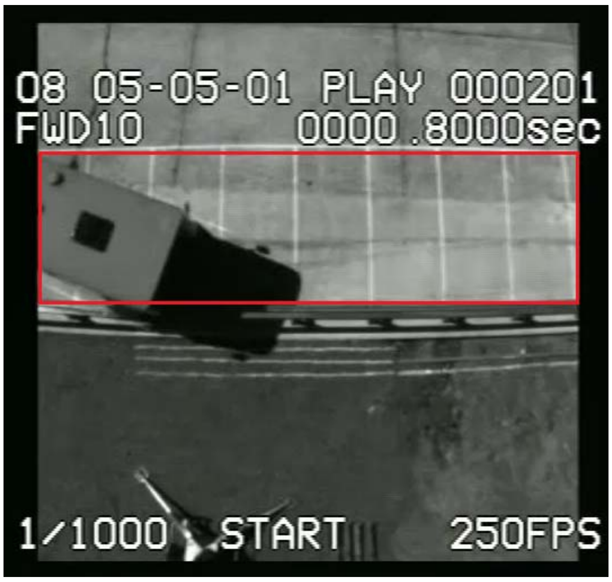


Figure 4. First frame of original video

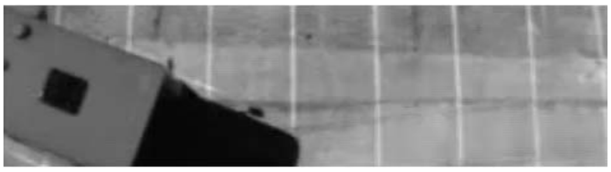


Figure 5. First isolated frame of interest

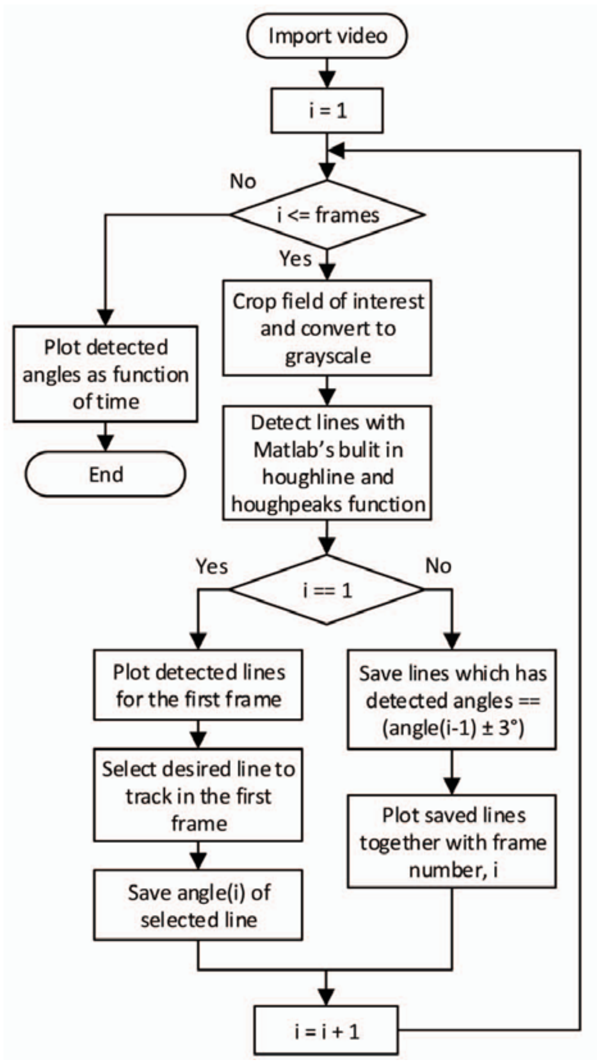
After each single frame has been cropped and converted to grayscale, the frames are stored in MATLAB for Hough line detection. It should be mentioned that a Canny edge detector [19] has been used to improve the robustness of the Hough line detection. The result of this edge detection is not used when plotting the video frames together with the detected Hough lines.

3.3. **The proposed algorithm.** When the HFR video is prepared as described in Section 3.2, the angles are extracted from each frame by implementing the HT. A MATLAB script is used to detect the angles of the bus in every frame using an HT function. A flowchart describing the MATLAB algorithm is shown in Figure 6(A).

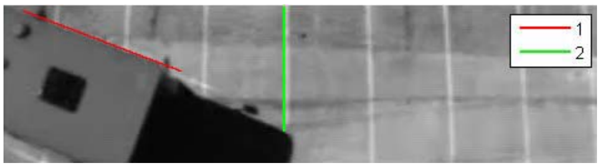
After the first frame is processed by the proposed HT algorithm, the first frame together with a set of indicated lines is plotted together in a figure, and the algorithm pauses. This enables the user of the algorithm to select the line which aligns at the roof edge of the bus. The MATLAB figure showing the selection of lines detected in the first frame is shown in Figure 6(B).

As shown in Figure 6(B), it is reasonable to track the red line (line 1) through the HFR video. This angle should represent the yaw angle of the bus as a function of time.

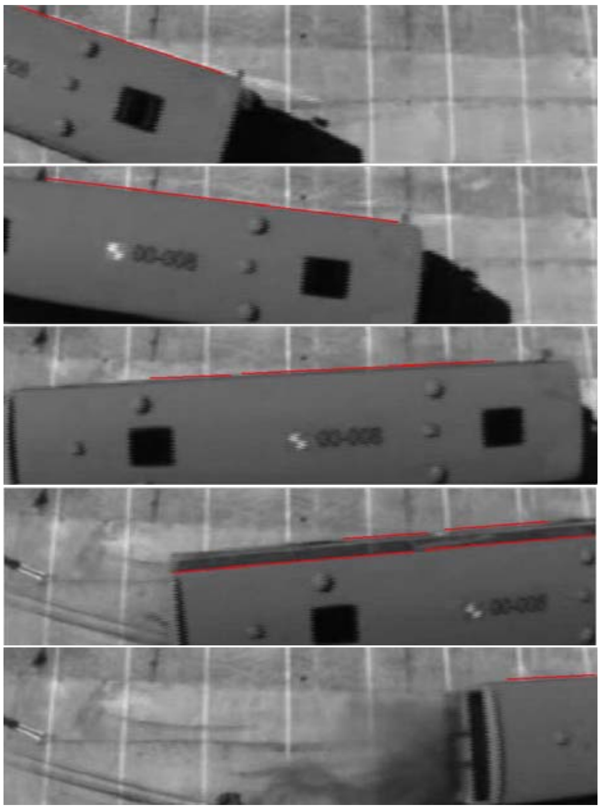
When the user has selected the desired edge to track in the first frame, the algorithm stores the detected angle temporary. This temporary angle is used for filtering out lines in the next frame which is not within the range of the temporary angle plus a threshold set to $\pm 3^\circ$. The temporary angle is then updated by the average angle of the lines within the threshold. This process is repeated for all the frames and the result of this algorithm is shown in Figure 6(C) where some selected frames are shown during the processing.


Import video
i = 1
No
i <= frames
Yes
Crop field of interest and convert to grayscale
Plot detected angles as function of time
End
Detect lines with Matlab's bulit in houghline and houghpeaks function
Yes
No
i == 1
Plot detected lines for the first frame
Save lines which has detected angles == (angle(i-1) ± 3°)
Select desired line to track in the first frame
Plot saved lines together with frame number, i
Save angle(i) of selected line
i = i + 1

(A) Flowchart for MATLAB program



(B) The first video frame used to choose desired edge to track ($i = 1$)



(C) Selected frames with detected lines through the HFR video ($i \neq 1$)

FIGURE 6. HT algorithm with sample figures at certain frames

During the processing, the angle detected from each frame is saved and is in the end plotted as a function of time. These angles are also numerically differentiated to obtain information for angular velocity and acceleration.

4. **Ford Fiesta into Rigid Steel Pole.**

4.1. **System description.**

An illustration of the geometrical dimensions of the Ford Fiesta is shown in Figure 7. The geometrical dimensions are needed in order to find the corresponding length of one pixel in the video frames.

All the geometric dimensions were measured before the real life crash where initiated. The measurements are gathered from the calibration test report of Robbersmyr [20]. The geometrical dimensions of the Fiesta are presented in Table 1.

During the real life crash test the accelerations in all three dimensions were measured by two ENDEVECO 3-D accelerometers mounted close to the vehicles center of gravity, and an IMAR gyro instrument measured the pitch rate. The center of gravity is illustrated on Figure 7. The center of gravity is measured by using load cells on each single wheel. By measuring the load on each wheel in two different tilted configurations the center of

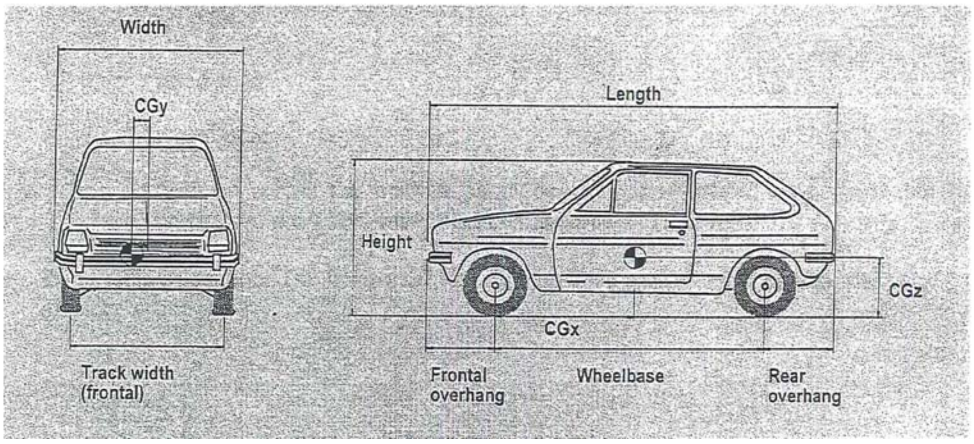


Figure 7. An overview of the dimensions of the Ford Fiesta [20]

Table 1. Geometrical parameters of Ford Fiesta [20]

Name	Value	Unit
Weight	873	kg
Width	1.580	m
Length	3.640	m
Height	1.390	m
Track Width	1.380	m
Wheel Base	2.280	m
Frontal Overhang	0.660	m
Rear Overhang	0.700	m

Table 2. Center of gravity properties of Ford Fiesta [20]

Name	Value	Unit
Longitudinal location CG_x	0.870	m
Lateral location CG_y	0.030	m
Height CG_z	0.520	m

gravity in all directions can be determined. The result of this measurements is presented in Table 2.

The Ford Fiesta was accelerated up to a speed of 35.4 km/h which was measured 5 m before the impact with the rigid steel obstruction [20]. The set-up used to accelerate the Fiesta into the desired speed is shown in Figure 8. The truck pulls a rope through a set of pillars to accelerate the Fiesta towards the rigid steel construction. The release mechanism is 5 m before the steel pole.

4.2. **Detection implementation.** In order to reduce the amount of information to process in the template matching algorithm, a region of interest is defined which is cropped from the original HFR video. The region of interest is shown in Figure 9.

The region of interest is then cropped from the original HFR video and stored in MATLAB for further processing. The number of templates which is desirable to track is chosen to be two and therefore both of the templates have to be specified by the user. The first template is illustrated in Figure 10.

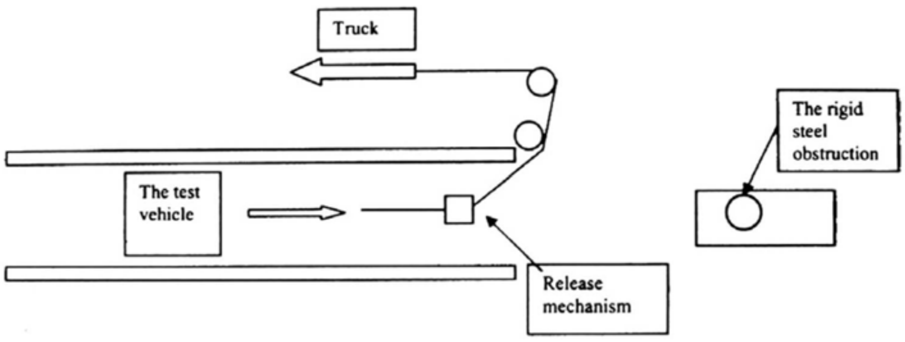


FIGURE 8. An overview of the test set-up used for the Ford Fiesta [20]

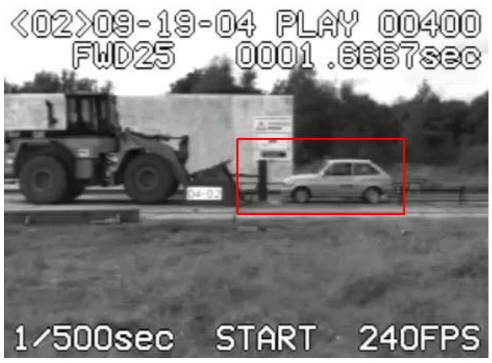


FIGURE 9. First frame used to describe the region of interest in the HFR video

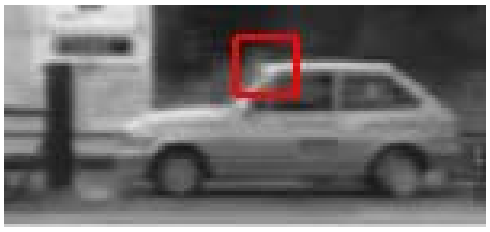


FIGURE 10. Selection of the first template in the first cropped HFR video frame

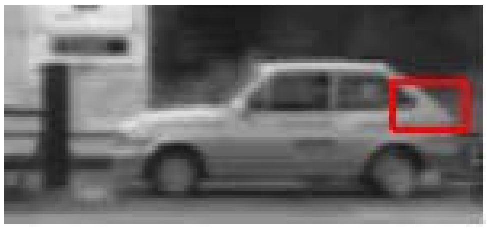


FIGURE 11. Selection of the second template in the first cropped HFR video frame

The second template is also chosen by the user and is illustrated by Figure 11.

These two templates are identified in all the frames in the cropped HFR video. These locations are saved in MATLAB and are used to identify both the pitch angle and the translation of the Ford Fiesta during impact. Since the length of the Fiesta is given in Table 1, it is possible to calculate how many millimeters one pixel in the HFR video represents. The length of one pixel in the video is calculated by the following formula:

$$L_{pix} = \frac{L}{|x_1 - x_2 + 1|} \tag{2}$$

where L_{pix} is the length of one pixel, x_1 is the first known location of the object, x_2 is the second known location of the object, and L is the known length in meter between points x_1 and x_2. In this detection process of the crash of the Fiesta the length of the whole car is used. It is easier to achieve an accurate result of the pixel length by using the largest known length of the object, so the length of the car is used to determine the pixel length L_{pix}.

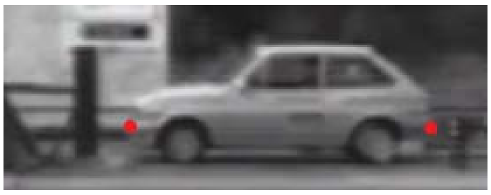


FIGURE 12. Illustration of the two points x_1 and x_2

As the length of each pixel and the two templates are described, the remaining work is carried out using a MATLAB algorithm which processes the remaining frames in the HFR video. The results are plotted as function of time, since the time for the first and last frames are known from the HFR video.

4.3. **MATLAB algorithm.** As described in Section 4.1 all the frames in the HFR video are processed using MATLAB's Image Processing toolbox. The MATLAB algorithm is presented in simplified flowchart which is shown in Figure 13.

The MATLAB algorithm uses a simple *for* loop to process every video frame. However, for the first frame the user has to specify the region of interest, the length of the object, and the two templates which are detected in the remaining frames. MATLAB's vision toolbox offers a Template Matching function which locates the position (x, y) of the template in a given image. The locations of both the detected templates in each single frame are stored for later calculation. The length of each pixel is used to calculate the translation of the object. The angle between the two points is calculated using an arctangent function.

5. **Experimental Results.**

5.1. **Oblique bus into railing crash.** By using the HT algorithm described in Section 2, the yaw and roll angles of the bus are extracted. To ensure that the obtained angles from the HFR video are correct they are compared to other measurements of the angles. From the real life oblique bus impact test, Robbersmyr and Bakken [21] logged the yaw rate in deg/s of the bus by using a gyroscopic device placed in the center of gravity inside the bus. The data from the gyroscope are compared with the measured angle from the HT algorithm. By numerical integration of the angular velocity, the angle of the bus is obtained.

In Figure 14, the measured angle from the HFR video is compared with the numerically integrated angle obtained from the gyroscopic data. The blue curve shows the yaw angle from the HFR video and the red curve shows the integrated gyroscope data. The resulting curves are quite similar, but some deviations are observed at the initial angles. To obtain the angular velocity from the HFR video, a polynomial was fitted to the discrete angular data. This polynomial was differentiated with respect to time to describe the angular velocity measured from the video.

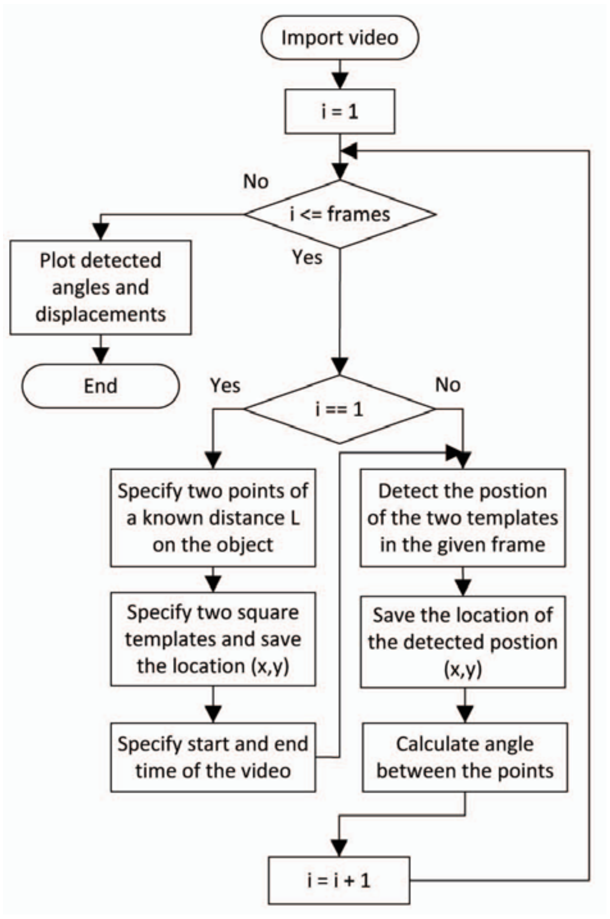

Import video
i = 1
No
i <= frames
Yes
Plot detected angles and displacements
End
Yes
i == 1
No
Specify two points of a known distance L on the object
Specify two square templates and save the location (x,y)
Specify start and end time of the video
Detect the postion of the two templates in the given frame
Save the location of the detected postion (x,y)
Calculate angle between the points
i = i + 1

FIGURE 13. TM algorithm flowchart

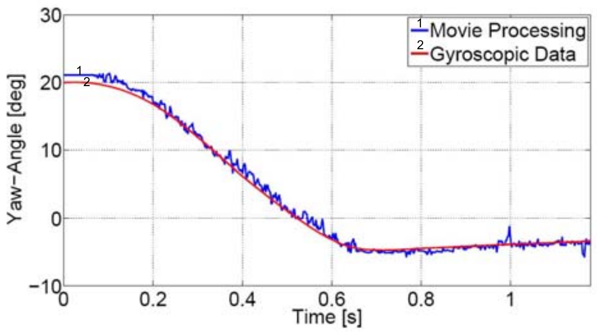


FIGURE 14. Comparison of yaw rotational angle vs. gyroscopic data

In Figure 15, the yaw rates from the HFR video and gyroscope are compared. As shown, there are some deviations between the curves. The cause of inaccuracy may come from the simple method used to differentiate the yaw angle measured from the video. Applying a more accurate differentiation algorithm together with interpolation may lead to more accurate results.

The video was also used to detect the global roll motion of the bus. This angle was extracted from the video captured in front of the bus during the impact. The camera location is illustrated in Figure 3 and is indicated as the *Front Camera* on the illustration. The result of using the MATLAB algorithm described in Section 3.3 to extract the global roll motion of the bus is illustrated in Figure 16.

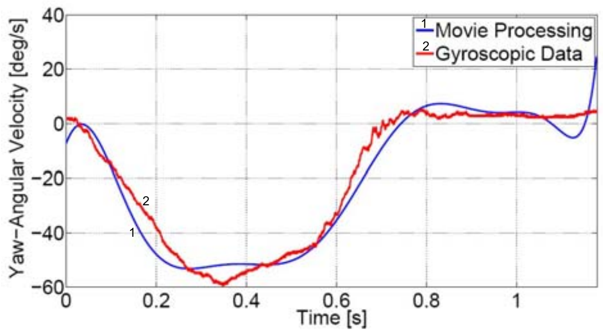


FIGURE 15. Comparison of yaw rotational velocity vs. gyroscopic data

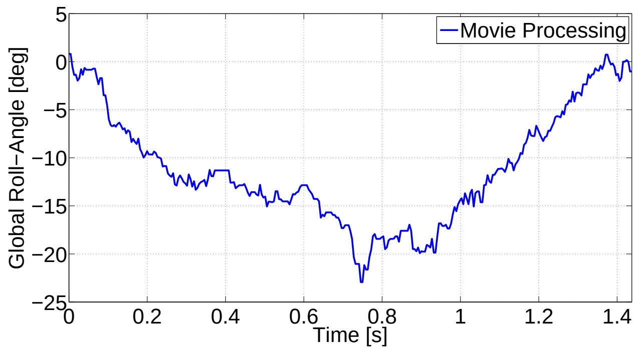


FIGURE 16. Global roll angle from HFR video

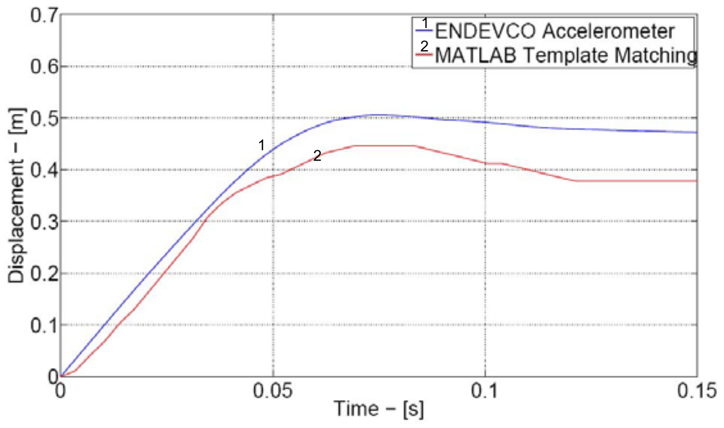


FIGURE 17. Displacement comparison of Ford Fiesta during impact

Unfortunately, the actual roll rate of the bus was not measured during the crash, and thus the angle extracted from the HFR video cannot be verified. However, during the MATLAB algorithm the detected edges of the bus are easily monitored by plotting the indicated edge in each movie frame. During the visual feedback, no lines were incorrectly identified.

5.2. **Ford Fiesta into pole crash.** The MATLAB algorithm described in Section 4.3 is used to detect the displacement and yaw angle of the Fiesta described in Section 4.1. The result is compared with the data measured with the ENDEVECO accelerometer and the results are shown in Figure 17. It should be noted that all the results from the MATLAB algorithm are smoothed with MATLAB's own *smooth* function.

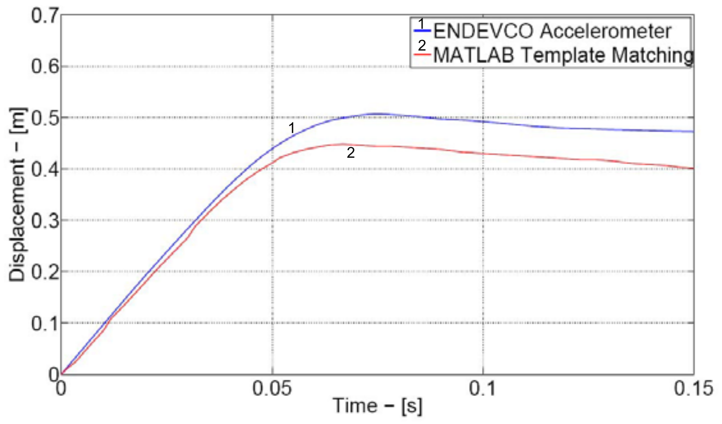


Figure 18. Displacement comparison of Ford Fiesta during impact

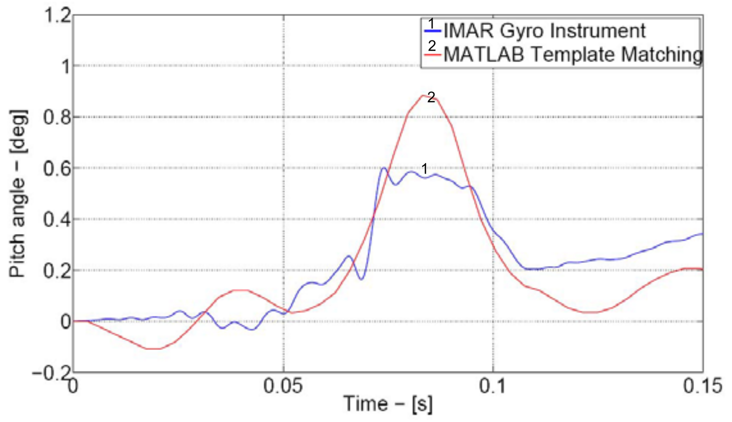


Figure 19. Pitch angle comparison of Ford Fiesta during impact

In the figure, the blue curve (curve 1) shows the actual displacement of the car during the crash, and the red curve (curve 2) shows the measured displacement from the HFR video. As seen on the figure, the displacements measured with the TM algorithm deviates from the displacement measured with the accelerometer. This may be caused by the low pixel resolution of the HFR video captured of the Ford Fiesta during impact. Which again will give a low resolution of the measurements.

In order to achieve a more accurate measurement, the TM algorithm was applied to an HFR video captured closer to the Fiesta during impact. The result is shown in Figure 18.

Figure 18 shows that the displacement measured by the TM algorithm deviates less from the displacement measured by the accelerometer. The shapes of the two curves are quite similar, but the deviation between the curves may be caused by inaccurate measurements from the accelerometer. The pitch angle is also calculated from TM algorithm and this angle is compared with the angle derived from the gyroscopic device. The result is shown in Figure 19.

In the figure, the blue curve (curve 1) shows the measured pitch angle from the gyroscope and the red curve (curve 2) shows the pitch angle measured from the HFR video. The result presented in Figure 19 shows large deviations between the two curves. This deviation may be caused due to the low resolution of the captured HFR video. The detected yaw angle of the bus crash, which was detected using the Hough transform was much more precise compared to the TM algorithm used on the Fiesta.

6. Conclusions.

In this paper the Hough transform (HT) and a template matching (TM) algorithm are used to detect the motion of a bus and a Ford Fiesta during a crash, respectively. During the oblique bus crash, both yaw and roll angles have been detected from a high frame rate (HFR) video using the Hough transform (HT). The x-directional displacement and the pitch angle of a Ford Fiesta are determined from a video using a TM algorithm. The result of the yaw angle detection using the HT algorithm shows promising results compared to the yaw angle measured with a gyroscopic device placed inside the bus during the crash. The x-directional displacement detected by the TM algorithm matches the shape of the displacement measured with an accelerometer placed inside the Fiesta, but the pitch angle detected using the TM algorithm has large deviations from the measured pitch angle.

A concluding remark of this paper is that a combination of the Hough transform and the template matching algorithm should be used to detect rotations and translations respectively. Also high resolution high frame rate videos should be used in order to achieve satisfactory results. Our next work will focus on development of more advanced image processing algorithms to improve the aforementioned results.

Acknowledgment. This work is supported by the grant of the collaborative research project between the University of Agder and Telemark University College.

REFERENCES

[1] J. R. Weaver, A simple occupant dynamics model, *Journal of Biomechanics*, vol.1, pp.185-191, 1968.
[2] P. Jonsén, E. Isaksson, K. G. Sundin and M. Oldenburg, Identification of lumped parameter automotive crash models for bumper system development, *International Journal of Crashworthiness*, vol.14, no.6, pp.533-541, 2009.
[3] C. H. Kim, A. R. Mijar and J. S. Arora, Development of simplified models for design and optimization of automotive structures for crashworthiness, *Structural and Multidisciplinary Optimization*, vol.22, no.4, pp.307-321, 2001.
[4] W. Pawlus, K. G. Robbersmyr and H. R. Karimi, Investigation of vehicle crash modeling techniques: Theory and application, *International Journal of Advanced Manufacturing Technology*, vol.70, pp.965-993, 2014.
[5] M. Huang, *Vehicle Crash Mechanics*, CRC Press, Boca Raton, USA, 2002.
[6] B. B. Munyazikwiye, K. G. Robbersmyr and H. R. Karimi, A state-space approach to mathematical modeling and parameters identification of vehicle frontal crash, *Systems Science & Control Engineering: An Open Access Journal*, pp.351-361, 2014.
[7] H. R. Karimi, W. Pawlus and K. G. Robbersmyr, Signal reconstruction, modeling and simulation of a vehicle full-scale crash test based on morlet wavelets, *Neurocomputing*, vol.93, pp.88-99, 2012.
[8] A. K. Zaouk, N. E. Bedewi, C. Kan and D. Marzougui, *Validation of a Non-Linear Finite Element Vehicle Model Using Multiple Imoact Data*, 1996.
[9] L. A. F. Fernandes and M. M. Oliveira, Real-time line detection through an improved hough transform voting scheme, *Pattern Recognition*, vol.41, pp.299-314, 2008.
[10] J. Ji, G. Chen and L. Sun, A novel hough transform method for line detection by enhancing accumulator array, *Pattern Recognition Letters*, vol.32, no.11, pp.1503-1510, 2011.
[11] M. Greenspan, L. Sahng and P. Jasiobedzki, Efficient tracking with the bounded hough transform, *Computer Vision and Pattern Recognition*, vol.1, pp.520-527, 2004.
[12] S. Jhumat, Vehicle speed estimation in accident prone areas using image processing, *International Journal on Advanced Research in Computer and Communication Engineering*, vol.3, no.5, pp.6420-6423, 2014.
[13] O. Ibrahim, H. ElGendy and A. M. ElShafee, Speed detection camera system using image processing techniques on video streams, *International Journal of Computer and Electrical Engineering*, vol.3, no.6, pp.771-778, 2011.
[14] S. Doğan, M. S. Temiz and S. Külür, Real time speed estimation of moving vehicles from side view images from an uncalibrated video camera, *Sensors*, vol.10, no.5, pp.4805-4824, 2010.
[15] F. Jurie and M. Dhome, Real time robust template matching, *British Machine Vision Conference*, 2002.

[16] N. Prabhakar, V. Vaithiyanathan, A. P. Sharma, A. Singh and P. Singhal, Object tracking using frame differencing and template matching, *Journal of Applied Sciences, Engineering and Technology*, pp.5497-5501, 2012.

[17] S. S. Tørdal, A. Klausen, H. K. Karimi, K. G. Robbersmyr, M. Jecmenica and O. Melteig, On detection of yaw and roll angle information for vehicle oblique crash using hough transform, *The 11th World Congress on Intelligent Control and Automation (WCICA)*, pp.5951-5955, 2014.

[18] P. V. C. Hough, *Method and Means for Recognizing Complex Patterns*, 1960.

[19] L. Ding and A. Goshtasby, On the canny edge detector, *Pattern Recognition*, vol.34, pp.721-725, 2001.

[20] K. G. Robbersmyr, Calibration test of a standard ford fiesta 1.1l, model year 1987, according to NS-EN 12767, (reference no. 04-02), *Project Report 43/2004*, 2004.

[21] K. G. Robbersmyr and O. K. Bakken, Impact test of safety barrier, test tb51 (reference no. 00-008), *Technical Report 008*, Agder Research, 2001.