Identification of Irregular Motion in Automotive Navigation Systems Using Novelty Detection

Martin Pöllot*, Dominic Springer[x], Ralph Schleifer[x], Dieter Niederkorn[x] and André Kaup*
*Chair of Multimedia Communications and Signal Processing
Friedrich-Alexander University Erlangen-Nürnberg (FAU), Cauerstraße 7, 91058 Erlangen, Germany
[x]Audi AG Ingolstadt, Auto-Union-Straße 1, 85045 Ingolstadt, Germany

***Abstract*—Automated display testing for visual unpleasant and erroneous navigation sequences is an important step to preserve a high quality standard for premium vehicle manufacturers. This paper presents a novel error detection algorithm for navigation sequences based on novelty detection on motion parameters obtained from real world navigation sequences. Motion parameters are accumulated through key point matching using BRISK and subsequent homography calculation. With these parameters one is able to describe the motion between two successive frames. Combinations of translational and rotational components allow the novelty detection algorithm to predict outliers. These outliers either show positioning errors or abnormal motion behaviour which are both unacceptable for high quality. Experimental results demonstrate that this algorithm works significantly better than state of the art, where one has to know errors before analysing the data set in order to determine thresholds for particular errors. The gains in precision and recall are 49.67% and 6.06% respectively, the accuracy is 1.37% higher compared to optimized threshold results.**

I. Introduction

Nowadays automobiles are equipped with a high number of components for information and entertainment purposes. With every new production cycle a plethora of new features are introduced not only for the driver, but also for fellow passengers. These components are generally controlled by a central processing unit. With an increase in complexity, the time and efforts for testing novel systems in order to find new errors during development, increases as well. Since errors have an impact on the quality of experience on the user side, errors should be eliminated during development.

In the scope of this paper, we consider errors from the onboard infotainment system with special view on navigation sequences. These sequences are recordings of a map indicating position of car and giving directions. There are several erroneous movements which can occur single or as combination of multiple errors. These errors contain drifts, where the position of the car changes horizontally. Jumps are also considered as errors. They occur when the car covers too much ground between two frames. This results in a jump from the former to the current position. Other errors cover for example backwards motion or an angle that is too high. Pairs of frames containing erroneous movement are shown in Fig. 1.

The scope of this paper is to develop an algorithm for automated testing that allows error detection regarding erroneous movements in navigation sequences with high reliability. These defective movements emerge from the fusion of multiple noisy sensor inputs, most commonly from a gyroscope, odometer and GPS-signals. Generally these errors occur infrequently when the driver is leaving a proposed route. If no route guidance is activated, errors can occur when the defective positioning is not unique anymore, for example when passing a motorway service area. Errors appear on rare occasions when there are two or more possible positions for the car resulting from the data fusion. Although GPS data as well as data from the sensor fusion is available, only the displayed scene is used. The reason for that is that the noisy sensor data of each input contains a lot of errors itself and are combined before they are being processed further. Another reason is, that the passengers only notice the errors displayed. The algorithm is therefore capable of indicating an improvement of the sensor fusion and the displayed navigation scenery. The indication is done by utilizing a machine learning (ML) method to detect irregularities displayed by comparing extracted motion parameters with a learned model of normality.

Fig. 1. Multiple examples for displayed errors from pairwise consecutive navigation frames. From top to bottom: the first row shows a horizontal drift to the right with a rotation of almost 40 degree. In the second row, one can see a jump resulting from a frame freeze. Motion is dominated by a vertical part, horizontal and rotational movement are negligible. The last row shows a rotation of 90 degree and a vertical jump. This is due to the fact that first the frame is rotated, then the translation is applied. Horizontal displacement is small.

The reason for using computers in the first place is, that

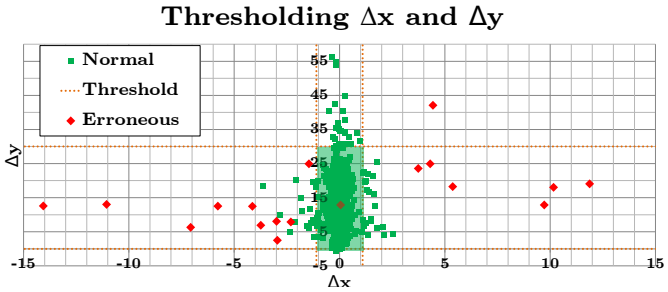


Fig. 2. Example scatter plot of the thresholding method. Every point shows a pair of motion parameters Δx and Δy calculated between two consecutive frames. The chosen thresholds are set as follows: $\Delta x \in [-1.1, 1.1]$ and $\Delta y \in [0, 30]$. These thresholds are obtained from all training sequences and are optimized to find as many errors as possible. The light green area in the center of the plot shows motion parameters that will be categorized as error free. Points lying in the remaining area will be marked as erroneous. The ground truth for all frames is shown with green dots for normal motion, red dots indicate erroneous motion. As one can see, one error in the center of the green shape will not be recognised, aswell as several normal pairs will falsely be marked as erroneous.

the amount of data to be tested exceeds human capacity. Therefore, one aspires an automated test setup that is able to process the amount of test data in a reasonable amount of time. With rising popularity and many innovations in the field of machine learning during the past few years, these methods are able to fulfil the task, if given carefully chosen parameters. The method then can decide if an observation fits into a prior learned context of regularity. The procedure is necessary to find well-known errors, but also unknown and unspecified anomalies within the displayed footage. These errors can then be analysed and prevented during later stages of development. For a ML algorithm to work properly, it is necessary to supply the algorithm with distinct features. In this paper, these features are extracted from the homography matrix which is calculated using binary robust invariant scalable key points (BRISK [1]). This is done to achieve comparable performance regarding state of the art. To make the testing algorithm as robust as possible for automation, different resolutions, icon-sizes and displayed features have to be handled. In the end, this algorithm should be easily implementable in automated testing environments.

II. State of the art

For state of the art error detection systems, motion parameters for a test scene are collected and then analysed. According to a set of thresholds that are applied to the tested data [2], errors are seperated from correct movements. With this setup, erronoeous movements are found with ease. Regarding unobtrusive errors, however, a clear affiliation is hard to find and might in the worst case need further attention and manual sorting. Another downside is that an error that is not known in prior and not captured by the chosen thresholds goes unchecked. This is clearly not wanted for test automation. The number of false alarms should be kept as low as possible and false negatives should not occur, especially when there are a wide variety of different sequences to be tested in a limited amount of time. Another drawback is the effort it takes to adapt the system to changing conditions, such as

Fig. 3. Left: Center of a typically displayed navigation scene for the driver in a car. In this example, route guidance is active. Static areas can be found at the top and bottom of the screen. In these static areas, information regarding destination or warnings concering the car are showed amongst many other, sometimes situationally dependant features. Right: A binary mask is shown with black overlay masking static and transparent areas, where white areas are taken into account.

changes in resolution or framerate. The general approach will be presented in the following three subsections.

A. Keypoint Detection, Description and Matching

An effective detection and description of key points $\boldsymbol{k}$ from an image $\boldsymbol{I}$ is a well-studied problem in many computer vision applications. Considering the description of these key points, the most common methods are Lowe's scale-invariant feature transform (SIFT, [3]) and Bay's speeded up robust features (SURF, [4]). Both SIFT and SURF descriptors have been improved in the past years. Calonder et al. showed with the algorithm binary robust independent elementary features (BRIEF [5]), that binary implementations can further reduce the computation time. BRIEF, however, is not invariant to scale and rotation changes unless coupled with a detector providing it, which led to the oriented fast and rotated BRIEF (ORB) descriptor by Rublee et al. in [6].
In this paper we use the binary robust invariant scalable key points (BRISK) [1] algorithm. It is similar to ORB, very fast and the accuracy in navigation sequences is sufficient. It is also easy to implement using Python [7] with OpenCV [8]. Automotive displays show many situational information such as speed, radio, destination, time and countless other features that are relevant for the driver or passengers. Areas in the image containing this information are irrelevant for the key point detection. Therefore, a static mask that covers the distracting elements, but not the map, is applied (see Fig. II). Motion in navigation sequences is dominated by rotational movement and zoom with only small translatory differences between successive frames [9]. After detecting a certain amount of key points $\boldsymbol{k}_1$ in one frame $\boldsymbol{I}_1$, the next frame $\boldsymbol{I}_2$ is being processed in order to obtain its key points $\boldsymbol{k}_2$. For the matching key points $\tilde{\boldsymbol{k}}_1$, the Hamming distance of all BRISK key points $\boldsymbol{k}_2$ corresponding to every analysed key point in $\boldsymbol{k}_1$ is calculated. Using a brute force matcher, every key point is guaranteed to be assigned to the closest key point of the set of key points $\boldsymbol{k}_2$. After this procedure, matched key points that do not belong to a homogeneous set are removed using random sample consenus (RANSAC, [10]). Fig. 5 shows matched keypoints in two consecutive frames using BRISK.

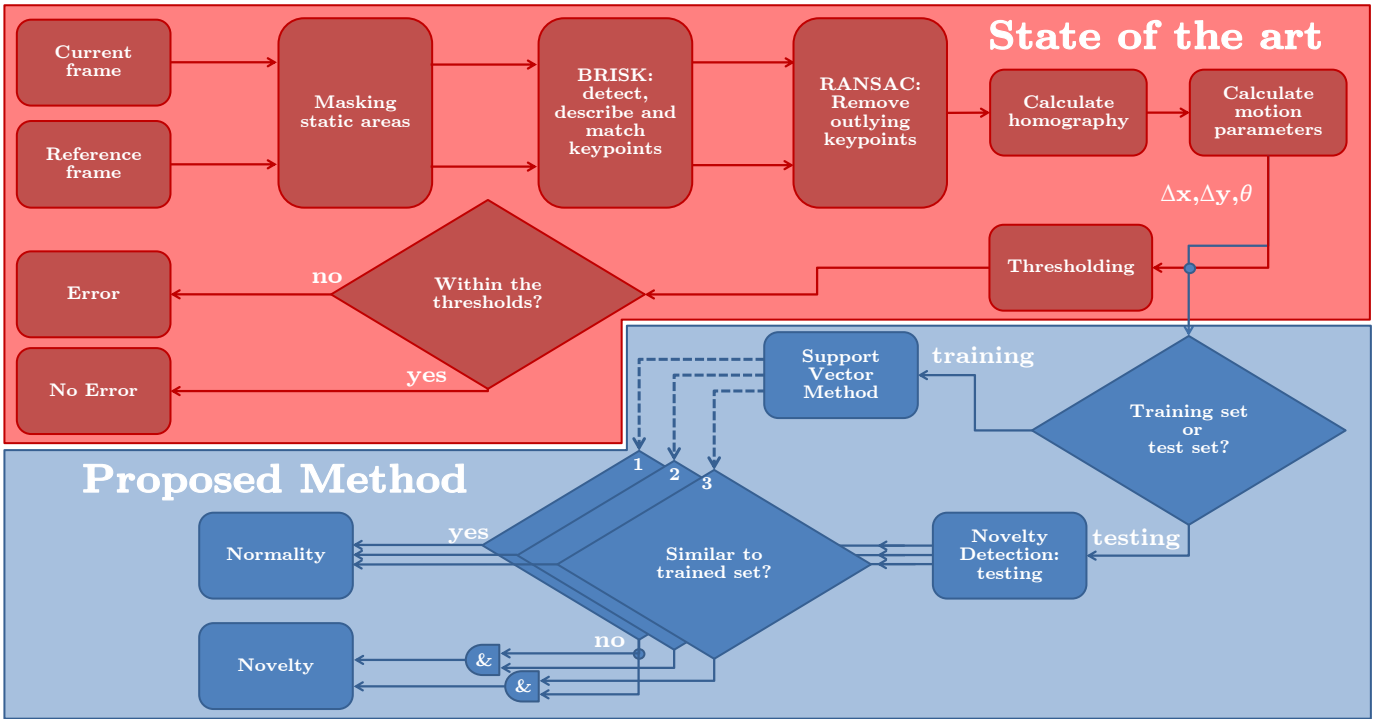


Fig. 4. Proposed irregular motion detection algorithm (blue) compared to the state of the art thresholding method (red). Two consecutive frames are first masked, then their key points are detected and matched, followed by the homography calculation. The resulting motion parameters are then used in the corresponding methods. Note that the output of the thresholding procedure is either an error or no error, whereas the output of our novelty detection is either a normal motion or a before unseen motion novelty.

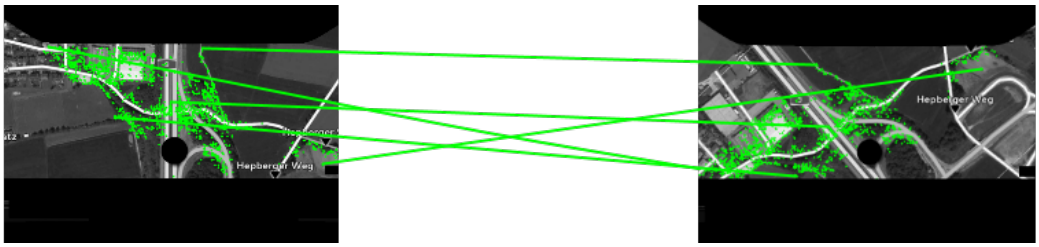


Fig. 5. Example of the set of matching key points (green) between two consecutive frames I_n and I_{n+1} with a resolution of 1440x540 pixels each. Some matching key points have been connected exemplary. Masking has been done to increase the robustness of the algorithm so it only checks map parts of the image. RGB has been converted to grayscale. Calculated motion parameters for this pair of frames are as follows: rotation $\theta = 41.08°$, $\Delta x = 17.23$ pixel, $\Delta y = -1.47$ pixel

B. Motion Parameter Calculation

In order to calculate the motion between two consecutive frames $\boldsymbol{I}_1$ and $\boldsymbol{I}_2$, the set of matching key points $\tilde{\boldsymbol{k}}$ is used to calculate the corresponding homography matrix $\boldsymbol{H}$. This set has to contain at least four different key points for the estimation to work. In general, there are several hundreds of key points matching for two consecutive frames (see Fig. 5). The homography matrix describes the relation of a planar surface in two images [11]. Since we are looking top down onto a navigation map, assuming a pinhole camera model, the movement of the camera position can be extracted [12]. This is equivalent to the motion of the automobile since the navigation sequences are restricted to a 2D top-view, which simplifies the homography matrix $\boldsymbol{H}$ by removing its skew factors. Using the homography matrix $\boldsymbol{H}$, the coordinates x, y and z of a point $\boldsymbol{p}_1(x, y, z)$ on the planar surface in frame $\boldsymbol{I}_1$ can be warped to match the most similar point $\boldsymbol{p}_2$ on that surface in the following frame $\boldsymbol{I}_2$. Note that the coordinate z in $\boldsymbol{p}$ that represents the height of the camera is set to 1 since we restricted the view to 2D top-view only and restricted the zoom to a fixed value. The scaling factor s is therefore equal to 1 since the height z does not change. Every pixel with coordinates $(x, y, 1)^T$ in the reference frame can be warped to the new position $(x', y', 1)^T$ in the current frame by using $\boldsymbol{p}_2 = \boldsymbol{H} \cdot \boldsymbol{p}_1$. With the homography matrix in the form of (1), the translational motion t_x and t_y and rotation θ of the image plane can be extracted; however, these do not correspond to the movement of the car. In order to get the movement of the car the coordinates of the centre of the car $(x_c, y_c, 1)^T$ have to be determined. By simply warping the car centre coordinates,

the new position of the automobile are obtained:

$$\begin{pmatrix} x_c' \\ y_c' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos(\theta) & \sin(\theta) & t_x \\ -\sin(\theta) & \cos(\theta) & t_y \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x_c \\ y_c \\ 1 \end{pmatrix} \quad (1)$$

Now one simply subtracts the original center coordinates to obtain the distance the car travelled. Since the origin of the frame is in the top left corner and a normal forward motion in the navigation scene would yield a negative Δy, we invert the sign of Δy. This is just for later convenience. Note that the first two elements in the third line in the homography matrix are not exactly 0 due to noise and small estimation errors. This results in a value for z that is not exactly 1 so that we have to normalize the resulting vector for z after calculation.

C. Thresholding

In order to check for various errors, a set of thresholds is defined based on observations that have to be done before testing. During these observations, it has to be defined what value is acceptable in order to set the thresholds accordingly. The thresholds are chosen as done in [12] with respect to finding as many errors as possible. For example, one such threshold for Δy is the maximum allowed number of pixels in forward motion. If the motion is higher than the threshold, the map in the navigation scene was jumping which would be signalled as an error. Therefore, Θ is set to the highest forward motion Δy allowed. An error is also detected if Δy is negative, since the car is not allowed to move backwards. From these two observations, one can extract thresholds for $\Delta y \leq 0$ and $\Delta y > \Theta$. One can go on and define any number of thresholds for other variables, depending on the errors one wants to detect, as seen in [12]. The results of this method can be seen in Fig. I. The major drawback, however, is on the one hand to define enough thresholds to achieve a precise error detection. On the other hand, the method should not need too many thresholds in order to stay adaptive. This contradictory trade-off situation is very costly regarding adaption time.

III. Irregular Motion Detection Using Novelty Detection

In this novel approach, we abandon manually setting a high number of thresholds for known errors as described above. Fig. II gives an overview of the proposed pipeline (blue) and the state of the art (red). With that we overcome the disadvantage of missing errors that are unknown in prior and increase the number of found errors whilst decreasing the number of false alarms. To achieve that, we use a machine learning technique named ND. It is widely used in fault detection and aims to find abnormalities that are very rare or where there is no data that describes the errors behaviour [13]. Given the fact that one can never train all possible errors, the procedure aims at learning what an error is not: normality. Therefore, ND can be used as a one-class classification tool, where one class has to be distinguished from all other possibilities [14]. Usually, the normal class is very well sampled while the other classes are almost or completely absent. This might have different reasons, for example when the task is to monitor a system where it is easy to gather data when it is running normal and abnormal events occur at rare intervals. We consider our motion data as training data $s_1, \ldots, s_n \in \boldsymbol{S}$ where n is the number of frames of a sequence $\boldsymbol{S}$. For ND, normal patterns of $\boldsymbol{S}$ are available for training, abnormal ones are scarce. From these observations, a model of normality $M(\phi)$, where ϕ represents the free parameters of the model, is derived and then used to calculate novelty scores $\xi(s)$ for previously unseen test data s. The larger the novelty score $\xi(s)$, the more the unseen test data s differs from the learned normality corresponding to the model $M(\phi)$. Obviously, a decision boundary ρ has to be defined, such that $\xi(s) \leq \rho$ labels data as normal and $\xi(s) > \rho$ as abnormal.

In this approach, we choose to use a support vector (SV) method proposed by [15]. It defines the novelty boundary in high dimensional feature space. Therefore, support vectors are mapped using a kernel K that requires beforehand fixing of the ratio ν between positive training data allowed to fall outside the boundary of normality and all (positive) training data. This parameter strongly influences the performance of the algorithm. For that we set ν to 0.0001 and only use positive data during training phase. This way we assure, that any observed test data that differs slightly from model $M(\phi)$ will be counted as abnormality. Since false negatives must not occur, it is better to make the algorithm more sensible, even for small derivations from the trained model $M(\phi)$.

The kernel (2) we use is the Gaussian radial basis function kernel (RBF kernel), which contains the squared Euclidean distance $\left\|s - s'\right\|^2$ between the two feature vectors and acts as the main comparison element in the algorithm. γ acts as a weighting factor, has to be greater than 0 and is set to 0.2 in this approach, since this factor resulted in best test results.

$$K(s, s') = \exp\left(-\gamma \frac{\left\|s - s'\right\|^2}{2\sigma^2} \right) \quad (2)$$

$$f(s) = \operatorname{sign}\left(\sum_i \phi_i K(s_i, s) - \rho \right) \quad (3)$$

We use this kernel in the decision function $f(s)$ that will output $+1$ for a normal observation or -1 for abnormal or novel observations[1] (3), according to [15]. The parameter ϕ can be seen as a multiplier containing our free parameters with $\phi \geq 0, \sum_i \phi_i = 1$ and ρ is our decision boundary. An implementation of this algorithm is included in Python's machine learning package scikit-learn [16] and was used with parameters mentioned earlier.

The decision function, however, is not scale invariant. In order to overcome this problem, one can either scale all training data or proceed with several combinations of parameters. Scaling is not an option, because we want to keep the motion parameters as they are. Scaling would require further analysis in order to put all parameters into one interval. We decided to use

[1]We use the convention as in [15] that $\operatorname{sign}(x)$ equals 1 for $x \geq 0$ and -1 otherwise.

TABLE I
TRAINING SEQUENCES FROM BOTH DISPLAYS.

Name	Resolution	Map Type	Frames
trainingSeq1a	1440x540	Imagery	3,012
trainingSeq1b	1024x480	Imagery	13,458
trainingSeq2a	1440x540	Standard	4,618
trainingSeq2b	1024x480	Standard	1,529
trainingSeq3a	1440x540	Standard	8,298
trainingSeq3b	1024x480	Standard	2,485
trainingSeq4a	1440x540	Standard	5,870
trainingSeq4b	1024x480	Standard	1,898
trainingSeq5a	1440x540	Standard	3,510
trainingSeq5b	1024x480	Standard	1,245
trainingSeq6a	1440x540	Standard	3,746
trainingSeq6b	1024x480	Standard	1,340

TABLE II
TEST SEQUENCES FROM BOTH DISPLAYS.

Name	Resolution	Map Type	Frames	Errors
testDrift1	1440x540	Standard	2,557	9
testDrift2	1440x540	Standard	2,560	6
testDrift3	1440x540	Standard	12,045	36
testDrift4	1440x540	Imagery	3,645	20
testDrift5	1440x540	Imagery	5,001	13
testDrift6a	1440x540	Imagery	1,366	33
testDrift6b	1024x480	Imagery	7,740	22
testDrift7a	1440x540	Standard	3,746	0
testDrift7b	1024x480	Standard	1,340	0

a combination of training parameter sets and combine their outcome. The training parameters obtained from section II-B are the angle θ describing the rotation between two frames, Δx and Δy containing the translational movement of the car's position. Although the SVM is calculating support vectors in a high dimensional space, it did not adapt well to too many different input arguments at once. To achieve an accurate estimation for irregular motions, we therefore train three different sets of motion parameters and later combine their output to one prediction. Therefore, the first set of parameters contains only the angle θ and the horizontal movement Δx. We choose these parameters since they are the strongest indicators for occurring erroneous movements. The second set uses the parameters $|\theta|$, $|\Delta x|$ and ${}^{\Delta x}/_{\Delta y}$. With this combination we can check the influence of a high derivation of either θ or Δx. By taking the absolute value we temporally lose the orientation of the movement. The fraction of ${}^{\Delta x}/_{\Delta y}$ grants a comparison of the impact of sideways movement. The third set only contains Δx and Δy and acts as an assurance for the first set without the impact of the angle θ. After predicting all three sets separately, set 1 and 2 as well as set 1 and 3 are compared and later combined, so that only outliers in both combinatory sets are selected as erroneous frames.

IV. EXPERIMENTS

This section provides the performance of the proposed algorithm compared to the standard thresholding approach. We apply the method to several navigation sequences with synthetical and real-world data to show the improvement of the proposed algorithm. For quality measuring we use precision and recall, as well as the accuracy for an overall performance as definded by [17].

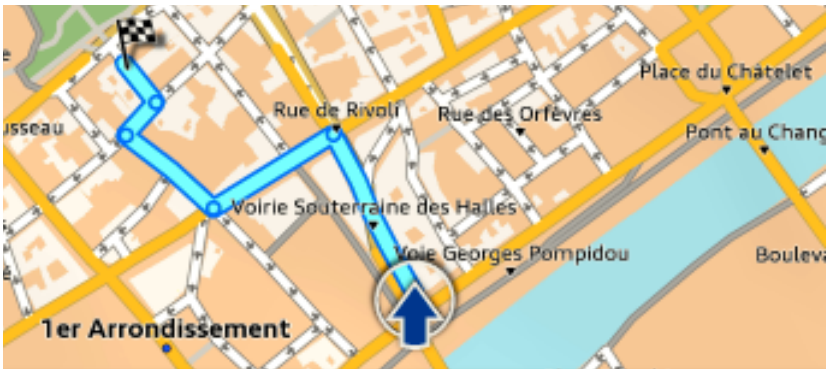
A. Acquisition of Navigation sequences

The sequences are obtained by screen grabbing the video signal via DVI connection directly from the display of a test cube and are then saved as png or bmp files (Fig. 6). These test cubes are equipped with many tools for analysis which are connected to a computer for further interaction. The test cubes contain regular displays, control and input devices set up just like in a real car with all necessary connections. In order to record a navigation sequence where the automobile would have to actually drive in order to produce different GPS positions and different sensor data, a work-around has to be used. On the one hand, the navigation system offers a mode for presentation where one can manually add a starting position. After entering a destination, the navigation will start and simulate the cars movement on set route with fixed speeds. We will call sequences that are obtained in this mode simulated. On the other hand, a real car can be equipped with a logging device that records all relevant information such as position, speed, heading or the number of active satellites. The recorded file called a trackfile can later at the test cube be used to recreate the route that was driven during recording. We will call these sequences real-world sequences. The advantage of the real-world sequences is that we can repeat tests as many times as necessary. Another advantage is that we only use the position from the car and do not need any navigation route to be active. We also have more precision in corners; since the presentation mode ignores corners and goes through them with speed unchanged. Furthermore, we can purposely set the navigation destination to a place where the car was not going to during recording. This way, we can force the positioning to be faulty and show irregular movements, when we set the navigation destination to a place that conflicts with the recorded route. This happens, because the positioning of the car is not only dependant on the sensor data but also on an active navigation route guidance.

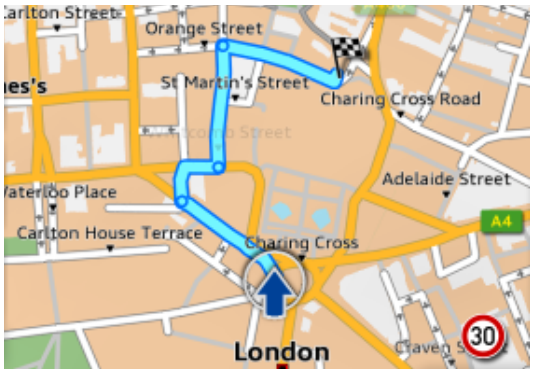
All sequences recorded conform with the following requirements: the map has to be shown in a top-down 2D-view, locked to the position of the car with the car always heading upwards. All icons showing Points Of Interest (POI) have to be disabled in order to achieve more robust results for the homography estimation. For even more accurate results, the overlay of satellite images should be activated. To get consistent results, the height of the camera or the zoom level is restricted to 100 meter above the car. The framerate, which has a great impact on the received speed of the car, is 15 frames per second in average. These high framerates are unusual for infotainment testing; but can be achieved by special grabbing setups. Currently the time between two frames is not taken into account since it is very difficult to obtain a constant framerate.

B. Training Data Set

The training data set consists of twelve sequences. Half of these sequences have a resolution of 1440x540 pixels. These sequences were recorded from display 1 which is in front of the driver. The other sequences were recorded from display 2

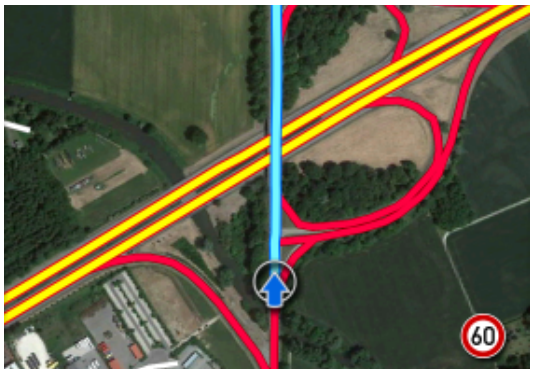


(a) Display 1 with standard map design



(b) Display 2 with standard map design

(c) Display 1 with satellite imagery design



(d) Display 2 with satellite imagery design

Fig. 6. Cutouts from frames acquired in navigation sequences with active route guidance and different map designs. (a) and (b) show the standard map desgin, whereas (c) and (d) show satellite imagery. (a) and (c) are obtained from display 1 with a total resolution of 1440x540 pixels, (b) and (d) are recorded from display 2 with a total resolution of 1024x480 pixels

which is centered in the middle of the console with a resolution of 1024x480 pixels. In total, $51,009$ frames comprised in all training sequences cover only synthetic navigation routes since real-world data might be erroneous. The navigation routes were set up in multiple cities with many corners. All images have an 8-bit depth and are compressed losslessly. One third of the frames use the map design with activated satellite imagery, the rest uses a simplified standard map. Please refer to Table I for more detail. After the training data acquisition, its motion parameters were calculated and the thresholds $\Delta x \in [-1.1, 1.1]$, $\Delta y \in [0, 30]$ and $|\theta| < 25°$ have been set for the thresholding method. The paramters have also been passed to train the ND.

C. Test Data Set

The test data set consists of nine sequences, recorded with the same settings as the training sequences. However, they make use of the trackfiles mentioned earlier. Therefore, they contain frames with erroneous movements. These erroneous movements are, as mentioned before, mainly drifts or jumps. Drifts are movements in the horizontal direction and are not allowed. These errors occur mostly, when route guidance conflicts with the GPS signal and sensor fusion. In these cases, the car arrow will follow the proposed route until the displacement surpasses a threshold. The car is then replaced to the correct position. This often happens when using motorway access or exit roads. Sometimes, freezes occur right after the replacements due to drifts. With an active route guidance, an updated route has to be calculated. This results in a high workload for the processing unit and can result in a freeze of the display where the motion is temporally stopped. After recalculation, the position of the car is updated. While one travelles a certain distance during this process, the distance covered during that time results in a jump. Every sequence has been checked visually and with motion parameters for erroneous frames, which are listed in Table II. Note that some of the sequences use satellite imagery, others only the standard map design.

D. Results

This section provides the evaluation of the proposed algorithm compared to the standard thresholding approach. For the thresholding approach, we used the intervals $\Delta x \in [-1.1, 1.1]$, $\Delta y \in [0, 30]$ and $|\theta| < 25$ that give the best results for all training data. Other thresholds have not been used. Examples for found errors or anomalies in the case of ND are shown in Fig. 8. For measuring the overall performance, precision (4) and recall (5) have been calculated and are shown in Fig. 7. Accuracy (6), which is used as a statistical measure for how well a binary classification test correctly identifies or excludes

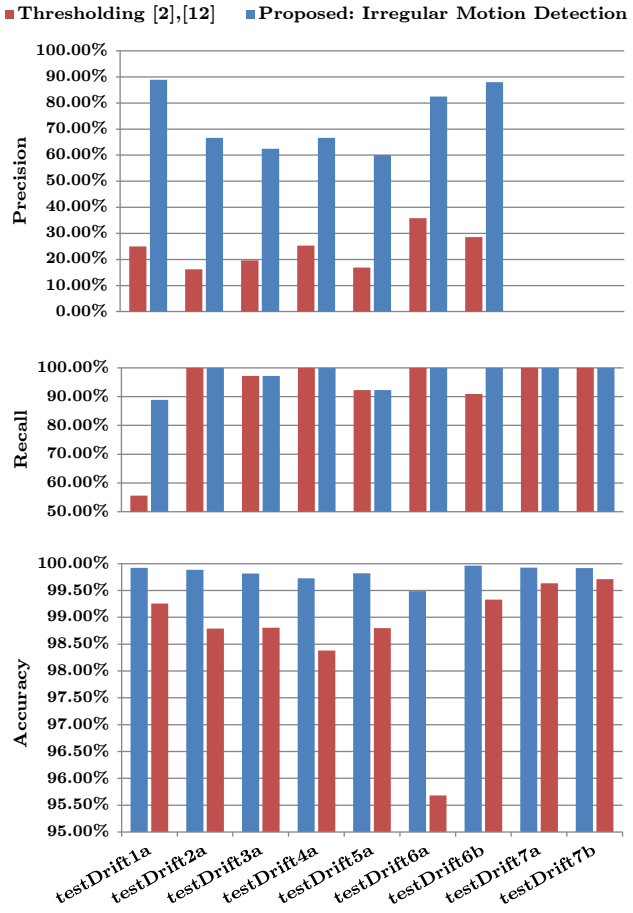


Fig. 7. These three graphs show from top to bottom precision, recall and accuracy of tested sequences. In all graphs one can see the improvement of the proposed method.

a condition overall, has also been calculated according to (6) and is likewise to be seen in Fig. 7.

$$\text{Precision} = \frac{\text{tp}}{\text{tp} + \text{fp}} \tag{4}$$

$$\text{Recall} = \frac{\text{tp}}{\text{tp} + \text{fn}} \tag{5}$$

$$\text{Accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}} \tag{6}$$

Hereby, tp means "true positive" and covers all decisions, where erroneous motion parameters have been found and marked correctly as such. fp stands for "false positive" and counts the number of times, that the algorithm marked a frame's motion parameters incorrectly as erroneous. This is often times called a *false alarm* or Type 1 error. The more important Type 2 error –a *miss*– would be the number of "false negatives" fn, where the analysed frame actually contains erroneous movements but treated them as normal. This number should be as small as possible, since errors should not go unchecked. "True negatives" tn represent frames without erroneous movements that were classified as error free.
Fig. 7 shows the precision and recall of both methods. The

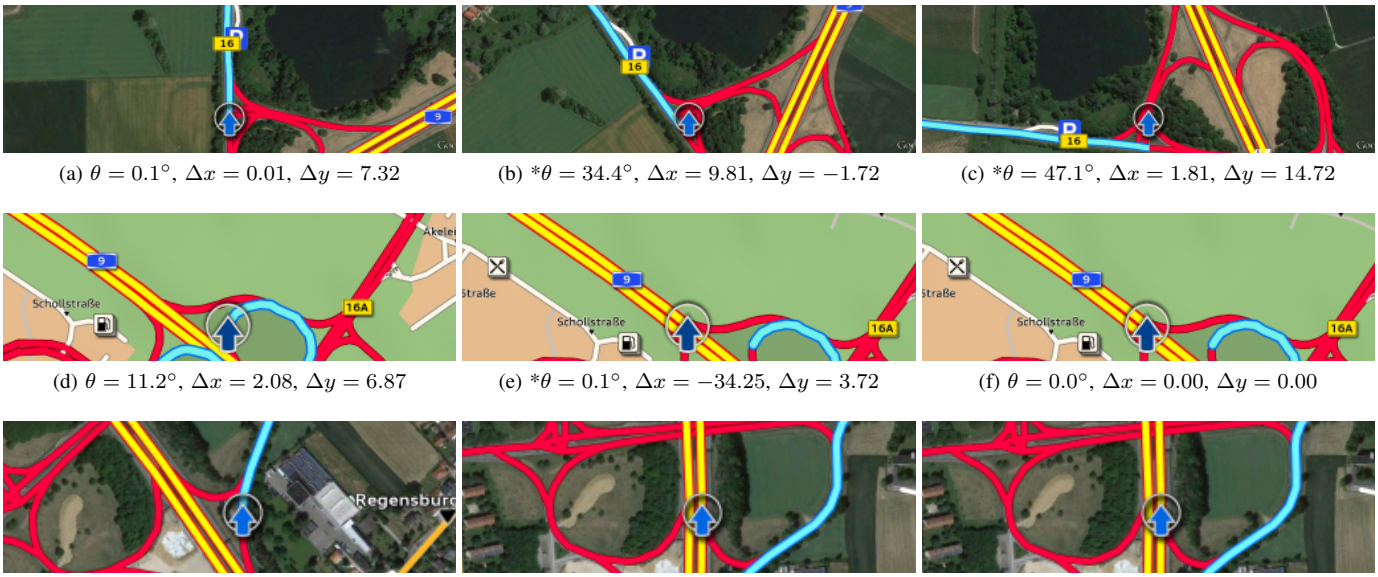
TABLE III
COMPARISON OF BOTH ERROR DETECTION METHODS.

Name	Proposed Method tp	fp	fn	Thresholding [12], [2] tp	fp	fn
testDrift1a	8	1	1	5	15	4
testDrift2a	6	3	0	6	31	0
testDrift3a	35	21	1	35	143	1
testDrift4a	20	10	0	20	59	0
testDrift5a	12	8	1	12	59	1
testDrift6a	33	7	0	33	59	0
testDrift6b	22	3	0	20	50	2
testDrift7a	0	2	0	0	5	0
testDrift7b	0	1	0	0	7	0

proposed method has a clear advantage over the simple threshold method. On average, the precision was raised by 49.67% except for the last two sequences that purposely had 0 errors. The number of false positives was lowered significantly, which results in less time for evaluation of predicted error frames. Some sequences purposely used the standard map design which is not as robust as the satellite imagery overlay which resulted in more false positives. Another reason is, that some sequences have POIs activated (Fig. 8a to 8f). However, the proposed method behaves superior, since it has same or a better detection rate and still fewer false positives as can be seen in Table III. This is also represented in the recall in Fig. 7. The proposed method improves the already high rate of 90.86% of the threshold method even further, on average to 96.92%. Out of 139 total errors in all sequences, the proposed method missed only three errors where the thresholding approach missed eight. This means that less errors that should not be contained in high quality navigation sequences went unchecked. Regarding the accuracy in Fig. 7, the improvements of the proposed method are visible. On average, the accuracy was raised by 1.37%. The dent in the thresholding method for sequence 6a results from many corners in that sequence that were handled poorly by the thresholding method. Combined with a low amount of images in that sequence the accuracy dropped significantly. Regarding a fairly huge amount of sequences that have to be tested, this increase of accuracy decreases significantly the number of frames that have to be checked by human observation. As mentioned earlier, the preparation time to set up automated tests with the proposed method decreases the consumed time even further.

V. CONCLUSION

In this paper, an irregular motion detection algorithm for navigation systems has been proposed. Using novelty detection, a clear advantage over state of the art thresholding methods could be accomplished. The requirements for recording appropriate sequences for a robust detection were evaluated and furthermore tested on standard map design, as well as on real-world satellite imagery. It was shown that our method achieves an average gain in precision of 49.67% which saves time during the evaluation after the detection of the frames that has to be done by human observation. Additionally, the recall was raised to 96.92%, which contains a gain of 6.06%



(a) $\theta = 0.1°$, $\Delta x = 0.01$, $\Delta y = 7.32$ (b) $*\theta = 34.4°$, $\Delta x = 9.81$, $\Delta y = -1.72$ (c) $*\theta = 47.1°$, $\Delta x = 1.81$, $\Delta y = 14.72$

(d) $\theta = 11.2°$, $\Delta x = 2.08$, $\Delta y = 6.87$ (e) $*\theta = 0.1°$, $\Delta x = -34.25$, $\Delta y = 3.72$ (f) $\theta = 0.0°$, $\Delta x = 0.00$, $\Delta y = 0.00$

(g) $\theta = 4.2°$, $\Delta x = 0.67$, $\Delta y = 15.57$ (h) $*\theta = -41.1°$, $\Delta x = -49.11$, $\Delta y = 13.37$ (i) $\theta = 0.0°$, $\Delta x = 0.00$, $\Delta y = 0.00$

Fig. 8. Cropped frames detected by the irregular motion detection. Note that the motion is calculated between two frames so there is always a pair of frames that behaves irregularly to the rest of frames. The frame indicating the error is the subsequent frame. Leading * indicates erroneous motion. (a) to (c) show frams 175 to 177 with two consecutive detected erroneous frames in *testDrift6a*, whereas (d) to (f) show only one erroneous frame detected in sequence *testDrift3a*, frames 3975 to 3977. (g) to (i) show frames 2984 to 2986 which contain one detected erroneous frame in *testDrift6b*. The motion parameters with respect to the subsequent frame are contained for each frame shown. Δx and Δy are in pixels. (e) to (f), as well as (h) to (i) are no errors regarding the motion, which is 0. The algorithm will, however, detect the following jumps in the succeeding frames due to the repositioning.

compared to the thresholding method. As for the accuracy, a gain of 1.37% was achieved, thus the accuracy resulting in 99.80%. Although only shown for two different display sizes, this method is expected to be working on any resolution with a moving map in the background and the requirements proposed. We also showed that the tedious acquisition of thresholds is not necessary anymore. One has to record error-free training data though, but this has to be done only once. Further work could contain the acquisition of a better set of training data. Additionally, the positioning symbol could be determined automatically for every display. An algorithm that recognises the positioning symbol and its centre would further reduce the required preparation steps. Therefore, future work will also include the application of an adaptive mask algorithm.

References

[1] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *Proceedings of the 13th IEEE International Conference on Computer Vision*, Barcelona, Spain, November 2011, pp. 2548–2555.

[2] G. Yammine, "Freeze detection in 2D navigation video sequences overlaid with real satellite images," in *Proceedings of the IEEE International Workshop on Multimedia Signal Processing (MMSP)*, Banff, Canada, September 2012, pp. 43–48.

[3] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, Kerkyra, Greece, September 1999, pp. 1150–1157.

[4] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *Proceedings of the Ninth European Conference on Computer Vision*, Graz, Austria, May 2006.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, *BRIEF: Binary Robust Independent Elementary Features*. Berlin/Heidelberg, Germany: Springer Berlin/Heidelberg, September 2010, vol. 4, pp. 778–792.

[6] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proceedings of the 13th International Conference on Computer Vision*, Barcelona, Spain, November 2011, pp. 2564–2571.

[7] G. Rossum, "Python reference manual," Centre for Mathematics and Computer Science (CWI), Amsterdam, The Netherlands, Tech. Rep., May 1995.

[8] *The OpenCV Reference Manual*, 2nd ed., Itseez, April 2014.

[9] D. Springer, F. Simmet, D. Niederkorn, and A. Kaup, "Robust rotational motion estimation for efficient HEVC compression of 2D and 3D navigation video sequences," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing ICASSP*, Vancouver, BC, Canada, May 2013, pp. 1379–1383.

[10] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, June 1981.

[11] D. Kriegman and S. Belongie, *Homography Estimation*, October 2007.

[12] G. Yammine, "Freeze detection in 2D navigation video sequences by matching of extracted line segments," in *Proceedings of the IEEE International Conference on Vehicular Electronics and Safety*, Istanbul, Turkey, July 2012, pp. 55–60.

[13] M. Markou and S. Singh, "Novelty detection: A review," *Signal Processing*, vol. 83, no. 12, pp. 2481–2497, December 2003.

[14] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "Review: A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, June 2014.

[15] B. Schölkopf, R. C. Williamson, A. J. Smola, J. Shawe-Taylor, J. C. Platt *et al.*, "Support vector method for novelty detection," in *Proceedings of the Twelfth Conference on Neural Information Processing Systems*, vol. 12. Denver, Colorado, USA: Citeseer, December 1999, pp. 582–588.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, November 2011.

[17] C. J. V. Rijsbergen, *Information Retrieval*, 2nd ed. Newton, MA, USA: Butterworth-Heinemann, March 1979.