DNN-based anomaly prediction for the uncertainty in visual SLAM

Vasileios Bosdelekidis*, Tor A. Johansen*, Nadezda Sokolova†*
* Department of Engineering Cybernetics, Norwegian University of Science and Technology, NO-7491 Trondheim, Norway
Correspondence: vasileios.bosdelekidis@ntnu.no
† SINTEF, 7034, Trondheim, Norway

Abstract—**The method described in this paper proposes a supervised Deep Neural Network (DNN) approach for the prediction of anomalies in camera-based navigation. The method is inspired by the unsolved issues of Integrity Monitors (IMs) when some of the sensor measurement covariances are unknown or inconsistent. Especially, the focus is on predicting when the estimation error distribution would require fatter tails to include outliers. The developed method takes into account single-frame image features as well as transient changes in the error. In the best of our knowledge, this is the first work that predicts anomalies in the error covariance of SLAM estimates and associates them with low-level image features. Finally, the prediction method can be used with other sensors as well, allowing the future development of navigation algorithm- and sensor-agnostic safety monitoring frameworks.**

Index Terms—**Deep Neural Network, covariance prediction, Sensor Fusion, Visual SLAM, Navigation**

I. Introduction

Autonomous vehicle navigation in safety critical operations has gained huge interest in the research community. However, the development of frameworks to monitor such systems and identify hazardous situations has been limited to specific navigation algorithms or sensors [1]. Solution Separation (SS) has been promoted several times as a promising framework for sensor-agnostic safety monitoring (e.g. [2], [3]), as far as the error covariances due to each sensor can be estimated. However, this is very often not the case. In addition, traditional Protection Level (PL) computation formulas developed for GNSS/Inertial Measurement Unit (IMU) systems might not be reliable. Estimating global positions from visual measurements entails many more sources of error in comparison to estimating from GNSS/IMU measurements, for example raw image noise or feature extraction and association errors [4]. Therefore, bounding the true position error due to each sensor might require special computations. The extension of the SS framework to navigation systems that are based on auxiliary sensors (e.g. camera, LiDAR) is also challenging due to the additional computational complexity, since in a SS framework a computationally heavy navigation algorithm has to be executed multiple times at each step.

Another requirement to guarantee reliable safety monitoring is the identification of the rare cases that cause a navigation system to fail, in order to conduct extensive testing of the framework under those conditions. The requirements of real universality of IM and identification of environment conditions that cause sensor faults or degradation, without the need of efforteous manual selection of relevant features, inspired the work presented in this paper. The focus is on camera-based navigation, although the method can be extended to any sensor.

Simultaneous Navigation and Mapping (SLAM) is a very popular framework when it comes to visual navigation. Consequently, there is a variety of SLAM methods, and the development of a safety monitor that can be integrated with a majority of these methods may be impossible. Methods that allow the uncertainty estimation in SLAM have been proposed in the past [5], [6], [7], however this would require modifications in the internals of the SLAM algorithm, or, to obtain some internal matrices and do the computations externally.

Methods based on covariance prediction attempted to relate features of the sensor input to covariance matrices and managed to overcome, in a large degree, the above limitations. An example is the method from Hu and Kantor [8] who predicted the variation of covariance of a Gaussian error distribution. Later contributions that use offline training, and the features are directly derived from the input measurements, are promising to predict the exact error covariance, based solely on the input measurements, without the requirement for manual identification of relevant features (e.g. [9] and [10]).

A DNN-based anomaly detection approach gives a simpler solution to the problem, as the model can be trained to learn when the distribution of estimated position errors should have fatter tails than the normal distribution to reliably compute error bounds. Anomaly detection in a DNN setting attempts to learn a feature representation of the raw inputs in the dataset, in a way that anomalous instances are distinguishable from normal instances. Other methods attempt to learn directly an anomaly score mapping function $\tau(.) : X \rightarrow \mathbb{R}$ [11]. The most popular approach for learning-based anomaly detection is to use autoencoders, where, in an unsupervised manner, the network is trained to reconstruct normal data from their low dimensional representations.The reconstruction error for anomalous data will be very high. The biggest disadvantage of using this approach in our problem is that our knowledge of which data is normal, in terms of not causing unbounded errors from a navigation algorithm, is very limited.

Wen and Keyes [12] proposed an anomaly detection method based on Convolutional Neural Networks (CNNs), utilizing transfer learning from a larger dataset, as the occurrence of anomalies is very rare. However, pre-trained models are usually available for specific type of data, whereas the detection of anomalies that can affect negatively a safety monitor is a problem that lacks similar data. Conventional neural network methods tend also to neglect past information, which makes them inappropriate for learning long-term dependencies among sensor measurements that cause anomalies. Long Short Term Memory Networks (LSTMs) are designed to model short-term as well as long-term data dependencies by controlling the addition and forgetting mechanisms of new and old information [13]. LSTMs have found large utilization in recent studies on anomaly detection. The largest focus was to detect abnormalities in sensor measurement time-series extending the framework of autoencoders, as in [14], or in the time-series of a specific IM test statistic (e.g. [15]). Literature on directly associating navigation faults with anomalies in single-sensor readings is still lacking, whereas a few works have used neural network architectures for prediction of failures, based on sets of measurements and probable actions undertaken by the system (e.g. [16]). Wyk et al. [17] tackled the problem of identifying anomalous sensor readings during automated vehicle navigation, via combined CNN and Kalman Filter (KF)-based anomaly detection. However, a central assumption is redundancy in sensor measurements and that a KF is applicable with the sensor input at hand. An interesting conclusion of their experiments is the superiority of CNNs in comparison to a combination of Recurrent Neural Networks (RNNs) with LSTMs, when there are normal values between consecutive anomalous values. However, they did not evaluate the combination of CNNs with LSTM, as is used in our work.

In comparison to previous research, our approach achieves clearer quantification of anomalies associated with input images during SLAM navigation, based on the error of position estimates to a reference trajectory. The problem of labeling measurement anomalies or outliers in a time-series without misclassifying inliers is an open research topic. The most relevant approaches that attempt to deal with the problem, utilize hypothesis testing. An example is the work of Tong and Barfoot [18] where their statistical testing approach also deals with the problem of misclassified inliers in a sequence of error samples that fail the test. Nonetheless, in this paper, the start and end of anomalies is identified with a simpler statistical method.

The main contributions and potential benefits of the work presented herein can be summarized as follows:

1) Early anomaly prediction in the position errors of camera-based navigation, taking into account low-level image features and presence of dynamic objects.
2) The algorithm can classify one or more subsequent outliers and image features that are likely the real origin of SLAM failures.
3) The method can benefit existing sensor- and navigation algorithm agnostic IM systems by alarming for unbounded covariance. Although the DNN was developed for visual input, the same logic can be applied for any type of sensor.
4) The developed network expects raw sensor inputs and classifies the output of a SLAM algorithm. Therefore, the robustness of various SLAM algorithms can be evaluated objectively with the same network and under the challenging conditions present in a standard dataset.

Section II describes the offline methodology to label the training dataset (II-A), the utilized DNN model (II-B) and the bias initialization procedure to cope with the class imbalance problem (II-C). In section III we describe the datasets used for evaluation of the model's performance (III-A), the result of the statistical anomaly labeling for the creation of training data (III-B) and the performance of the model on the training, validation and test sets (III-C). Section IV concludes the paper.

II. DNN MODELING FOR ANOMALY PREDICTION IN VISUAL-BASED NAVIGATION

In the following, a raw measurement sample refers to M sequential images. We are given a set of K pairs of measurement samples and associated true errors of positions estimated by a SLAM algorithm, at each time step i. The errors are assumed to follow a normal distribution $\mathcal{N}(0, \boldsymbol{R}_i)$ in the nominal case, with $\boldsymbol{R}_i$ the covariance matrix at step i. Then, we optimize for the parameters of a DNN to predict that specific features in input images will cause an error to fall outside the distribution $\mathcal{N}$. For example, a sample $\{\boldsymbol{\xi}_i...\boldsymbol{\xi}_{i+M}\}$ starting at image i contains the images $\boldsymbol{\xi}_k \in \mathbb{R}^m, k = i, i+1, ..., i+M$ stacked in a vector, with m being the number of pixels in the image (or a down-scaled version of it). Then we obtain a set of low dimensional features, which are vectorized such that for each image $f(\boldsymbol{\xi}_k) \in \mathbb{R}^r$, $r \ll m$. Let $\boldsymbol{W}$ be a weight matrix and $\boldsymbol{b}$ a bias which can shift the neural network activation function to the left or right and is also learned by the neural network. Then the hypothesis for a sample i will be:

$$h_i(\boldsymbol{\xi}_i, ..., \boldsymbol{\xi}_{i+M}) = g(\boldsymbol{W} \cdot \Lambda(f(\boldsymbol{\xi}_i), ..., f(\boldsymbol{\xi}_{i+M})) + \boldsymbol{b}), \quad (1)$$

where we abstracted the specifics of the hidden layers that are applied between the input and output layer. Λ is a function that combines features of all images in the set and finds the relationship between each other. g was selected to be the Sigmoid function, as the desired range of values for the binary classification problem is $[0, 1]$. Section II-B will give an idea of the layers that compute function Λ. It is attempted to estimate the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ by optimizing the cost function:

$$J(\boldsymbol{W}, \boldsymbol{b}) = \frac{1}{K} \sum_{i=1}^{K} Cost(h_i, y_i), \quad (2)$$

with y_i denoting the true label corresponding to image i. The labeling of normal or anomalous cases had to be done as a preprocessing step, and the procedure is described in section II-A. Section II-B describes the DNN architecture to learn relevant features, while section II-C explains the method to initialize realistically the bias of the output due to the imbalance of the two classes.

A. Isolation of anomalies in visual navigation

The targeted problem is to identify feature levels in images that can cause a sudden increase in the estimated absolute position error of a SLAM algorithm of interest. A Ground Truth (GT) is therefore required for training the model. In that GT the input images are associated with the label "anomaly - 1" or "normal - 0", depending if they cause an anomaly in the error distribution of the SLAM estimates. The SLAM algorithm should be executed with the optimal calibration parameters for this purpose. However, sensor faults and degradation may happen very rarely, and, many times, there is only a small effect by challenging conditions of very short duration. Therefore, it is important to predict how the SLAM algorithm would behave if some specific conditions are persistent. The requirement to predict image features that cause suddenly or progressively an anomaly creates a trade-off for the anomaly isolation algorithm.

The algorithm that is adopted in this work to detect anomalies in the error samples is a modification of the peak detection algorithm of Brakel [19]. The detection depends on the z-score of a position error sample, that is, the number of standard deviations that the error sample is above the error mean [20]. Here, a moving average, that is tolerant to outliers, is used as reference. The outliers affect the moving average in a small degree, although a slow adaptation is allowed, assuming that only measurements at the beginning of the peak cause the fault. In case of multidimensional positioning the maximum among all axes errors is selected, and the anomaly labeling is done based on that error.

Let N be the number of error samples which coincides with the number of steps in the estimated camera trajectory. Let $e_1, e_2, ..., e_N$ denote the sequence of error samples and $e_1^*, e_2^*, ..., e_N^*$ a sequence with weighted error samples, so that, for each $j \in [1, N]$

$$e_j^* = \begin{cases} e_j, & \text{if } j \textit{ an outlier} \\ \alpha e_j + (1-\alpha)e_{j-1}^*, & \text{otherwise} \end{cases}, \tag{3}$$

with α being a parameter that sets the influence of an outlier to the mean. A reasonable value for α for non-stationary signals is in the range of $[0.01, 0.1]$ and is expected to be set as zero for a stationary signal.

Then at each step i and for a configurable horizon of size L samples the mean will be:

$$m_i = avg(e_{i-L+1}^*, ..., e_i^*) \tag{4}$$

and the standard deviation:

$$\sigma_i = std(e_{i-L+1}^*, ..., e_i^*) \tag{5}$$

A sample is flagged as an outlier when its z-score is above a threshold which can be selected based on the assumption that the error follows a normal distribution and the number of anomalous samples that we expect. Z-score tables, like the one in [21], can be used to select a threshold based on the guessed probability of anomaly.

It is seen that the outliers are included slowly in the computation of the moving mean. In that way, the algorithm can classify subsequent outliers until the mean has been adapted to include them. The algorithm will stop detecting after a sequence of outliers has adapted the mean value. Although this algorithm can prevent some false positives, it is important to force an earlier finish of labeling anomalies.

In this paper we adapt a simple method for stopping the labeling of outliers based on the dynamics of the error. Specifically when L errors have been added in a sequence, with the first element corresponding to the latest error, an anomaly is still valid if the error appears to be still increasing. We evaluate if the maximum error is the current one or at least one of the latest ones, to tolerate for noise:

$$\underset{j \in i, \dots, i-L+1}{\arg\max} \; e_j^* < T_p, \tag{6}$$

where $T_p \in \mathbb{Z}^+$ a positive integer used as threshold to tolerate that any of the latest T_p errors is the maximum. In this way, the method labels as anomalies only the first samples that lead to a peak in an error plot, while the method without the stopping rule labels all images associated with an error peak.

B. Deep Convolutional Neural Network

To learn the relationship of low-level image features and transient changes in the environment, the DNN architecture of Figure 1 was implemented. All input images are down-sampled using the OpenCV library [22] to low-dimensional images of width W and height H, where the downsampling method is bilinear interpolation.

A Time Distributed Layer (TDL) [23], shown as the large rectangle in Figure 1, was selected to compare a set of M sequential images and learn, in this way, the effect of transient changes (e.g. dynamic objects) to the predicted output. In the TDL the same layers are applied to each image to extract relevant features, but one set of optimal parameters is produced for all images in the sample. The extraction of low level features from the images is achieved by combining two convolutional layers with non-linear activation function and two subsequent max-pooling layers. The TDL keeps a 1-1 relation of input image and corresponding output. The LSTM layer is introduced to learn the temporal dependence among observations, e.g. image frames in chronological order [23]. A flattening or pooling operation after the TDL is introduced since the requirement is to have only one dimension per output from the Time Distributed wrapping to insert them to the LSTM layer.

Finally, a fully connected layer of 256 units and a drop-out layer are added in the output.

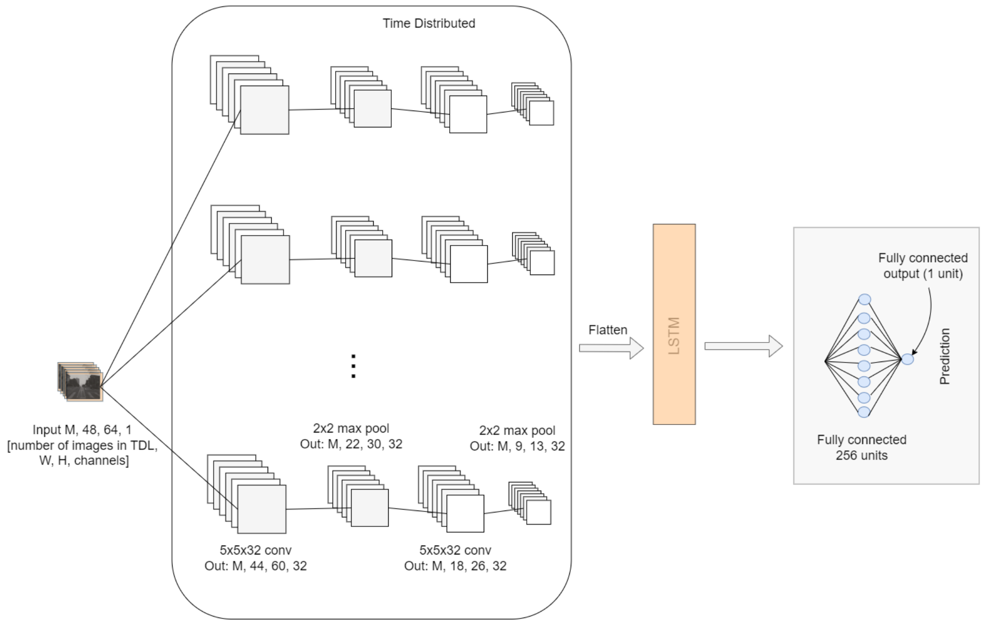


Fig. 1: DNN diagram. The indicated output dimensions might be inaccurate and they depend on the padding and stride parameters used. Modified architecture from Liu et al. [9] to use TDL. In that paper they used leaky rectify activation as non linear activation function.

C. Output bias initialization

The anomalies are significantly fewer than the normal cases, and this can be an apparent issue if a strict anomaly labeling method is selected. Using a zero-bias can make it difficult to obtain good convergence initially. In contrast, we are aware of the class imbalance and, therefore, an initial bias b_0 can be obtained based on the probability of a positive class [24]:

$$\begin{aligned} p_0 &= pos/(pos+neg) = 1/(1+e^{-b_0}) \\ &\Rightarrow b_0 = -ln(\frac{1}{p_0}-1) \\ &\Rightarrow b_0 = ln\frac{pos}{neg}, \end{aligned} \tag{7}$$

where pos and neg are respectively the number of positive (anomalous) and negative (normal) examples in the training set.

III. Experiments

A. Training data and appropriability for the problem

The input to the CNN is sequential images and position error pairs $\{\{\boldsymbol{\xi}_i ... \boldsymbol{\xi}_{i+M}\}, e_i | i \in [1, K]\}$. The number of frames in each sample was pre-selected to be $M = 7$. The Visual SLAM algorithm ORBSLAM2 [25] is executed for the computation of the estimated camera positions. Then the errors to a reference trajectory can be computed. The training can be done offline, for any SLAM algorithm, by using the same set of data every time. The current dataset for training consists of three trajectories from two different sources. In all cases a car is driving in an urban environment, with illumination challenges, repetitive patterns and dynamic motion of pedestrians and other cars. All the data are open-source and the sources are the following:

- UrbanLoco [26]: The dataset targets the problem of navigation in dense urban environment. It is distributed with GT positions from a SPAN-CPT module that integrates a GNSS and an Inertial Navigation System (INS). Data used are from a car driving in a city in California.
- Complex Urban Dataset / KAIST [27]: Data from stereo camera, GNSS and IMU. Another dataset in dense urban environment that can be used for navigation. It has data from multiple cities and countries. The specific trajectory that was used is from South Korea. The GT was created by us with a basic integration of GNSS + INS in Error State Kalman Filter (ESKF) [28], therefore it has limited accuracy.

The images' aspect ratio varies between the datasets. Figure 2 shows some example images from both datasets. A frame rate of 10 Hz is used for both datasets, where downsampling is employed if necessary. In this paper we evaluate the monocular SLAM case. If a dataset contains data exclusively from a front facing stereo camera, we will use images only from the left camera. Both datasets provide their own extrinsic and intrinsic camera parameters.

Figures 3, 4, 5 depict the GNSS, GT and estimated camera positions from ORBSLAM2 (labeled as "cam" in the figures) relative to the initial position, in North-East-Down (NED) coordinates, for the three trajectories. Computation of the optimal transform to align the camera poses with

Fig. 2: Example images from the experimentation datasets. Two trajectories are evaluated from UrbanLoco and one from KAIST.

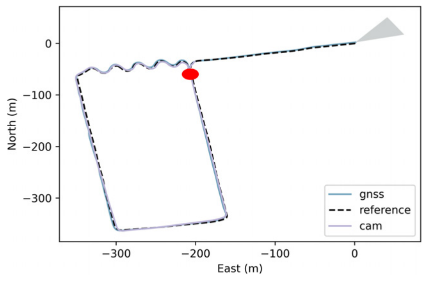


Fig. 3: The first trajectory from UrbanLoco. GNSS, GT and estimated camera positions from ORBSLAM2.

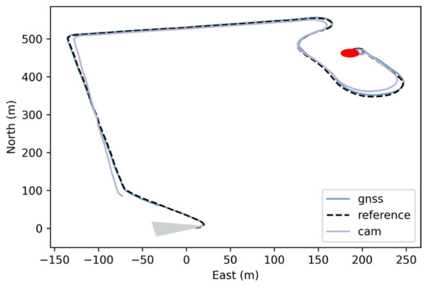


Fig. 5: The second trajectory from UrbanLoco. GNSS, GT and estimated camera positions from ORBSLAM2.

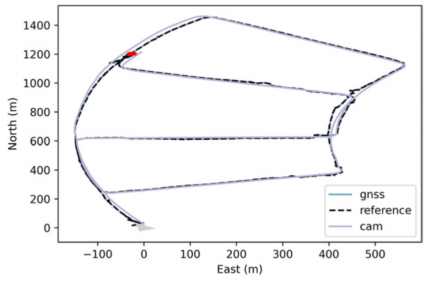
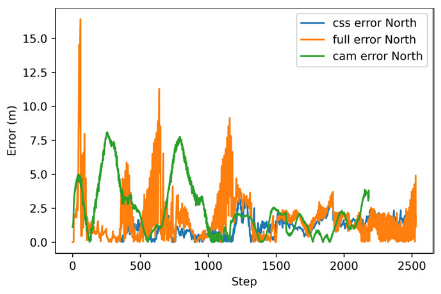


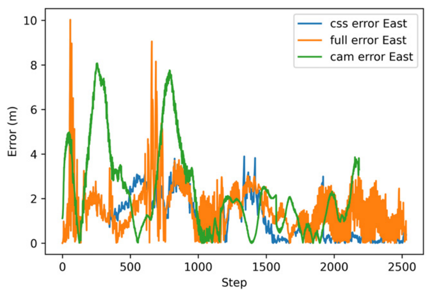
Fig. 4: The trajectory from KAIST. GNSS, GT and estimated camera positions from ORBSLAM2.

the reference was achieved with the Umeyama method [29]. The gray arrow in the figures shows the starting position, and the red circle the ending position. Figure 6 compares the true absolute errors for the first Urbanloco trajectory of the position estimates from three navigation solutions; using the camera alone in ORBSLAM2, integrating an IMU and GNSS, or integrating an IMU, a GNSS and the camera position estimates from ORBSLAM2. One can see that the inclusion of camera may deteriorate the navigation performance some times, leading to large position errors. However, many times a camera can complement the IMU and GNSS, showing comparable or better performance than the IMU and GNSS integration. A camera can be very assistive in cases of GNSS or IMU unavailability or faults. Therefore, the integration of auxiliary sensors with conventional sensors (e.g. IMU and GNSS) may lead to superior performance, although the detection of anomalous measurements independently from each sensor is an essential prerequisite.

During the experimentation it is expected that the majority of the true position error samples is concentrated in a small region. As the estimated camera position might be prone to errors due to intrinsic calibration or alignment parameter inaccuracies, the distribution is not necessarily concentrated close to zero. Figures 7a and 8a show that distribution for the first Urbanloco and KAIST trajectories respectively. Since we deal with a 2-dimensional problem, the illustrated error is the



(a)



(b)

Fig. 6: Position errors over time obtained for three navigation solutions. ”css error” is the error from the integration of GNSS and IMU in ESKF. In the ”full error” the ORBSLAM2 camera position estimates are integrated with the IMU and GNSS. ”cam error” refers to the error of position estimates from ORBSLAM2. The errors in (a) north and (b) east axis are shown.

maximum observed among the North and East axes at each timestep.

B. Anomaly labeling result

The CNN is fed with GT labels that correspond to each image and show if it causes an anomaly or not. The decision to label an image as anomalous or not depends on our intuition and the labeling algorithm described in section II-A. Figures 7b, 7c, 8b and 8c illustrate for the two trajectories the samples that are labeled as anomalous or normal, with or without the ending rule, together with the error plot and the moving mean. The z-score threshold was selected as 4.5 and influence parameter as $\alpha = 0.01$. The anomaly ending rule is used in all further experiments and the threshold used was $T_p = 4$. Importantly, different parameters can lead to more or fewer labeled anomalies, where, based on the given problem, we might select to be more or less conservative.

C. Training and evaluation of the network

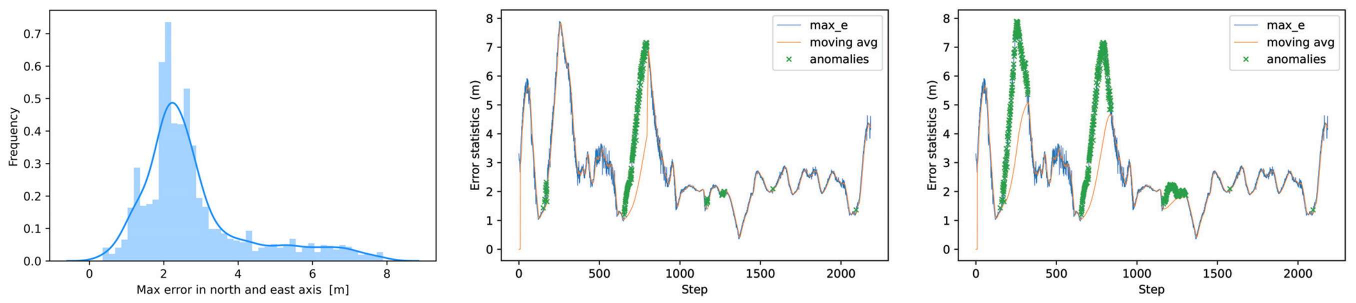
Each sample can contain subsequent images from the same trajectory. A set of 7400 samples of image sequences from the UrbanLoco and KAIST trajectories (1900 and 1000 samples from the two UrbanLoco trajectories and 4500 samples from the KAIST trajectory) was split randomly in the training (66.6%) and test sets ($33,3\%$), with a fixed seed so that the DNN never sees the test set during training. The samples that appear as anomalies when labeling without the ending rule and as normal when applying the ending rule (see figures 7 and 8) are not considered for training or validation in this paper. 30% of the training samples were selected for validation, using the cross-validation method. The size of a batch was selected as 50 samples, and main criterion is that enough positive examples are included, although large batch size might cause memory exhaustion. The network is trained for 2000 epochs, binary cross-entropy is used as the loss function and the optimization is accomplished with the Adam optimizer with learning rate 10^{-5}. Figure 9 evaluates the evolution of the loss, recall and precision of the DNN during training and validation. The two latter metrics indicate the presence of false alarms and of missed anomalies, respectively. The DNN generalizes quite well to the validation set, although the significant fluctuations indicate some sensitivity to noise. Despite the continuous improvement of the precision and recall, the validation loss curve seems to flatten after a while. This indicates that although the classifier makes correct predictions, the margin between the calculated class probabilities becomes smaller. Steps that can improve the method further are denoising, regularization and increased dataset size. Finally, considering the context of the problem, someone might want to improve further from the recall of 0.9 which was achieved until now for the test set. Figure 10 plots the Area Under Precision-Recall Curve (AUPRC). It shows the performance that can be achieved for different values of the classification threshold. In this problem, the presence of False Negatives is usually far more costly than the presence of False Positives, although one would also like to avoid many false alarms that cause an IM to stop the autonomous operation.

For a classification threshold of 0.5, Table I shows the confusion matrices of the predictions of the model on the test set. In addition, the performance of the CNN with TDL is compared with the performance of a CNN that does not take into account temporal dependence among frames, i.e. with the TDL removed. It is visible that the CNN performs better in learning anomalies when the TDL is present.

Finally, Table II compares the performance of the model on the test set for different z-score and T_p threshold values. The performance is similar and remains good in all cases, at least for the recall metric. A general trend is that stricter anomaly labeling results to worse performance and, if there are very few anomaly samples in the training set, additional procedures might be needed, such as transfer learning, data augmentation or class weighting.

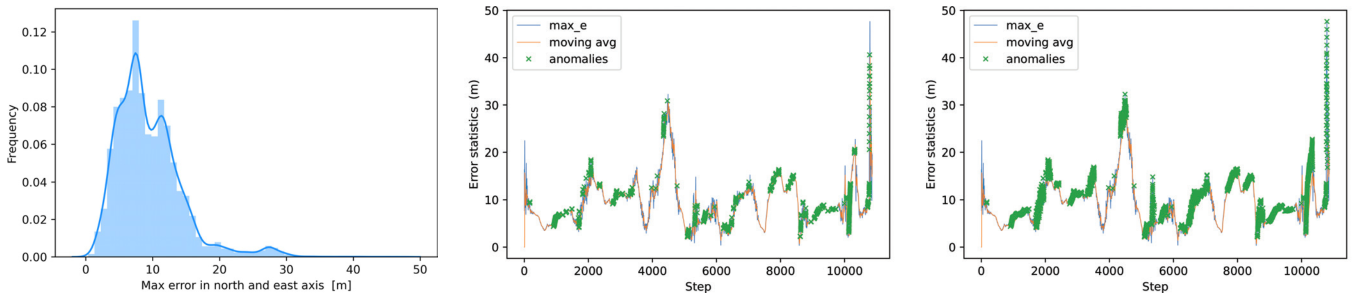
IV. Conclusion

This paper investigated a deep CNN for associating anomalous increases of the estimation error from a visual SLAM algorithm to low level image features. The CNN is trained in a supervised manner with several image trajectories captured by ground vehicles in dense urban environments, whereas the



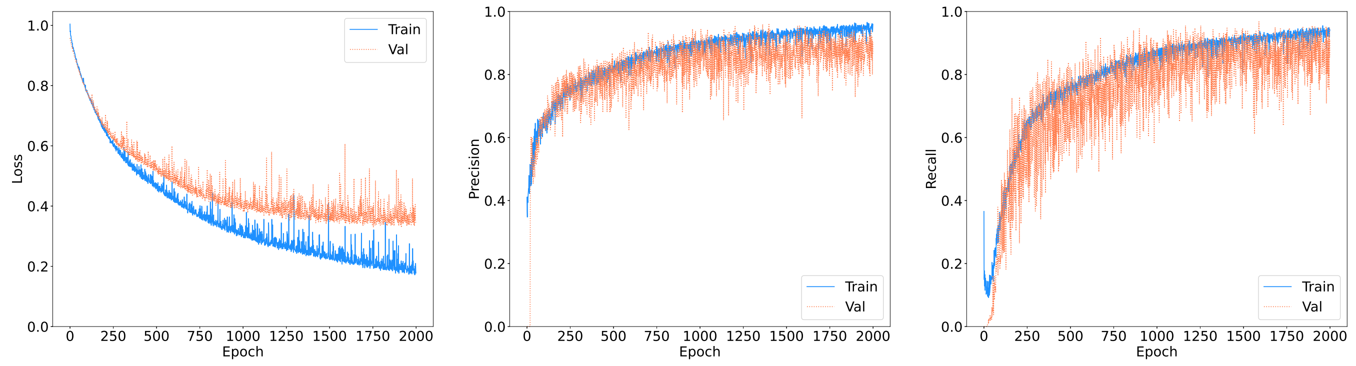
(a) (b) (c)

Fig. 7: For the first UrbanLoco trajectory (a) error distribution of the estimated camera position, (b) marked anomalies with the anomaly ending rule, and (c) marked anomalies without the anomaly ending rule.



(a) (b) (c)

Fig. 8: For the KAIST trajectory (a) error distribution of the estimated camera position, (b) marked anomalies with the anomaly ending rule, and (c) marked anomalies without the anomaly ending rule.



(a) (b) (c)

Fig. 9: Evolution of the (a) loss, (b) precision and (c) recall of the DNN with TDL during training, where, with cross-validation, 1/3 of the training samples are selected for validation.

class labels are specified based on the statistical properties of the position error. A TDL is included to learn temporal dependences among sequential image frames. This allows the investigation of the effect that dynamic objects have on the accuracy of visual-based position estimates, as well as the early prediction of anomalies. The results demonstrated a good performance of the CNN on test data and a tangible improvement in learning anomalies in comparison to a CNN without the TDL. The presented approach is a novel and significant contribution in the domain of monitoring the safety of autonomous vehicle navigation and can be extended to evaluate the robustness of any navigation algorithm or sensor in challenging environmental conditions.

V. Acknowledgements

The work is funded by the Research Council of Norway and Centre for Autonomous Marine Operations and Systems for the corresponding PhD project at the Norwegian University of Science and Technology (RCN grant number: 305051)

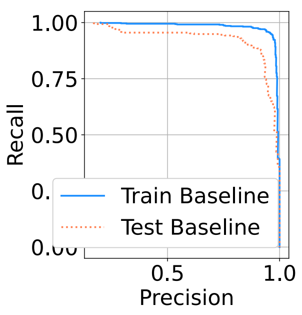


Fig. 10: AUPRC performance of the network for the train and test sets, where the precision-recall points are obtained using various classification thresholds.

TABLE I: Confusion matrix for the predictions of the test samples with and without TDL in the network, where the counts for the latter architecture are shown in parentheses.

		Prediction outcome		
		p	**n**	**total**
actual value	**p′**	283 ()	31 ()	314
	n′	42 ()	1559 ()	1601
	total	325 ()	1590 ()	

References

[1] T. Hassan, A. El-Mowafy, and K. Wang, “A review of system integration and current integrity monitoring methods for positioning in intelligent transport systems.” *IET Intell Transp Syst.*, 2020.

[2] A. Appleget, R. C. Leishman, and J. Gipson, “Evaluation of sensor-agnostic all-source residual monitoring for navigation,” *Proceedings of the 2021 International Technical Meeting of The Institute of Navigation*, 2021.

[3] Q. Meng and L.-T. Hsu, “Integrity monitoring for all source navigation enhanced by kalman filter based solution separation,” *IEEE Sensors Journal*, 2020.

[4] C. Zhu, M. Meurer, and C. Günther, “Integrity of visual navigation—developments, challenges, and prospects,” *NAVIGATION: Journal of the Institute of Navigation*, 2022.

[5] A. I. Mourikis and S. I. Roumeliotis, “Analysis of positioning uncertainty in simultaneous localization and mapping (slam),” *International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[6] H. Carrillo, I. Reid, and J. A. Castellanos, “On the comparison of uncertainty criteria for active slam,” *International Conference on Robotics and Automation*, 2012.

[7] M. Tkocz and K. Janschek, “Towards consistent state and covariance initialization for monocular slam filters,” *Journal of Intelligent Robotic Systems*, 2015.

[8] H. Hu and G. Kantor, “Parametric covariance prediction for heteroscedastic noise,” *International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[9] K. Liu, K. Ok, W. Vega-Brown, and N. Roy, “Deep inference for covariance estimation: Learning gaussian noise models for state estimation,” *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[10] F. Wu, H. Luo, H. Jia, F. Zhao, Y. Xiao, and X. Gao, “Predicting the noise covariance with a multitask learning model for kalman filter-based gnss/ins integrated navigation,” *IEEE Transactions on Instrumentation and Measurement*, 2021.

[11] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, “Deep learning for anomaly detection: A review,” *ACM Computing Surveys*, 2021.

[12] T. Wen and R. Keyes, “Time series anomaly detection using convolutional neural networks and transfer learning,” *ArXiv*, 2019.

[13] B. Lindemann, B. Maschler, N. Sahlab, and M. Weyrich, “A survey on anomaly detection for technical systems using lstmnetworks,” *Computers in Industry*, 2021.

[14] D. Park, Y. Hoshi, and C. C. Kemp, “A multimodal anomaly detector for robot-assisted feeding using an lstm-based variational autoencoder,” *IEEE Robotics and Automation Letters*, 2017.

[15] D. Kim and J. Cho, “Improvement of anomalous behavior detection of gnss signal based on tdnn for augmentation systems,” *Sensors*, 2018.

[16] G. Kahn, P. Abbeel, and S. Levine, “Land: Learning to navigate from disengagements,” *IEEE Robotics and Automation Letters*, 2021.

[17] F. v. Wyk, Y. Wang, A. Khojandi, and N. Masoud, “Real-time sensor anomaly detection and identification in automated vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[18] C. H. Tong and T. D. Barfoot, “Batch heterogeneous outlier rejection for feature-poor slam,” *IEEE International Conference on Robotics and Automation*, 2011.

[19] J.-P. G. v. Brakel. Robust peak detection algorithm using z-scores. Version: 2020-11-08. [Online]. Available: https://stackoverflow.com/questions/22583391/peak-signal-detection-in-realtime-timeseries-data/22640362#22640362

[20] B. Murphy and R. Barr. The z-score. Accessed: 2022-06-17. [Online]. Available: https://mat117.wisconsin.edu/4-the-z-score/

[21] A. Alvarado. Z-table. Accessed: 2022-06-17. [Online]. Available: https://castle.eiu.edu/ aalvarado2/z_table.pdf

[22] G. Bradski, “The opencv library,” *Journal of Software Tools*, 2000.

[23] A. Ravi and F. Karray, “Exploring convolutional recurrent architectures for anomaly detection in videos: a comparative study,” *Discover Artificial Intelligence*, 2021.

[24] A. Karpathy. A recipe for training neural networks. Version: 2019-04-25. [Online]. Available: https://karpathy.github.io/2019/04/25/recipe/

[25] R. Mur-Artal and J. D. Tardós, “Orb-slam2: an open-source slam system for monocular, stereo and rgb-d cameras,” *IEEE Transactions on Robotics*, 2017.

[26] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu, “Urbanloco: A full sensor suite dataset for mapping and localization in urban scenes,” *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[27] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, “Complex urban dataset with multi-level sensors from highly diverse urban environments,” *International Journal of Robotics Research*, 2019.

[28] S. I. Roumeliotis, G. S. Sukhatme, and G. A. Bekey, “Circumventing dynamic modeling: evaluation of the error-state kalman filter applied to mobile robot localization,” *IEEE International Conference on Robotics and Automation*, 1999.

[29] S. Umeyama, “Least-squares estimation of transformation parameters between two point patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1991.

TABLE II: Performance on the test set for different values of the z-score threshold T_z and anomaly ending tolerance threshold T_p. The percentange $PERC$ of labeled anomalies to normal samples is also shown.

T_z	T_p	$PERC$	Precision	Recall
4.5	3	15	0.79	0.87
4.5	4	20	0.87	0.9
4.5	5	23	0.89	0.87
5	4			
3	5	57	0.92	0.9