Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 51-29 (2018) 159–164

Convolutional Autoencoder aided loop closure detection for monocular SLAM[★]

Marco Leonardi[*] Annette Stahl[*]

[*] *Department of Engineering Cybernetics, Centre for Autonomous Marine Operations and Systems, NTNU AMOS, Trondheim, Norway. e-mails: marco.leonardi@ntnu.no, annette.stahl@ntnu.no*

Abstract: A correct loop closure detection is an important component of a robust SLAM (simultaneous localization and mapping) system. Loop closing refers to the process of correctly asserting that a mobile robot has returned to a previous visited location. Failing to detect a loop closure does in general not pose a threat to the positioning and mapping system of a robot, but a wrong loop closure can lead to drift of the robot and can therefore jeopardize the robot's mission. In this paper a robust, highly parallelizable standalone algorithm for globally detecting loop closures is proposed. The algorithm is purposely built with the goal of avoiding false positives, while maintaining reasonable true positives performances. Tests conducted on the KITTI and the Scott Reef 25 dataset show that when bag-of-words approaches perform poorly, our presented approach is able to avoid wrong loop closures.

Keywords: loop closure detection, monocular SLAM, loop closures, relocation system, error bounding, underwater navigation, underwater SLAM

1. INTRODUCTION

Loop closure detection is an important task linking together the localization and the mapping in the SLAM process. Loop closure detection makes it possible to reduce drift, perform relocalization when tracking is lost and allows to reconstruct the true topology of the scene.

Loop closure detection for visual SLAM is currently mainly based on bag-of-words models Engel et al. (2014) Mur-Artal et al. (2015) Cummins and Newman (2011), these have been proven to work very well in practice, but most of the time these algorithms are employed in indoor or urban environments. Early work in loop closure is linked to EKF-SLAM and the use of a *validation gate*[1] for data association Bailey and Durrant-Whyte (2006), which make use of the state estimation and can easily result in a wrong data association.

In this paper a method for loop closure that does rely only on images is presented, in this way a global identification of loop closures occurs, decoupling the drift in position estimation from the performance of the loop closure detection system. This method is independent from other components of a general SLAM system, allowing it to be further modified and plugged easily into a motion estimation and mapping algorithm to generate a SLAM algorithm. As this is an image-to-image method for loop closure detection that does not use any information about the estimated position at the time images are received, this method does suffer from the problem of *perceptual aliasing*, i.e. for equal sensed images captured at different positions within the environment (repetitive patterns in the environment), the algorithm will relocalize to the coordinates where the first image instance has been captured, thus indicating a wrong loop closure.

2. RELATED WORK

Early attempts in the field of visual SLAM implemented loop closing, but not in a global way. The keyframe based SLAM method called PTAM Klein and Murray (2007) does provide loop closing based on the correlation of thumbnails of the keyframes, but this does not allow to perform large loop closures. Another popular keyframe based method for SLAM called ORB-SLAM Mur-Artal et al. (2015) is able to perform global loop closure detection, but the process is quite complex and involves many variables relative to the current state of the SLAM algorithm. An equally notable large scale monocular SLAM approach that provides global loop closure detection is LSD-SLAM Engel et al. (2014), which similarly to ORB-SLAM uses a bag-of-words approach for finding loop closure candidates.

One of the most notable works in appearance based localization is FAB-MAP as presented in Cummins and Newman (2008) Cummins and Newman (2011). The authors propose a probabilistic approach for solving the problem of recognizing places based on visual words and learn a generative model of place appearance.

Current state-of-the-art monocular SLAM approaches consist of depth prediction based methods exploiting con-

★ This work was supported by the Norwegian Research Council through the Centre for Autonomous Marine Operations and Systems at NTNU

[1] After the prediction step in EKF-SLAM a prediction of the measurement is performed, this yields to a value-range in sensor space where to expect an observation. The area is called validation gate and is used to narrow the search and exclude other potential matches

2405-8963 © 2018, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2018.09.486

volutional neural networks Tateno et al. (2017) Ummenhofer et al. (2017). Such approaches, thanks to the learned priors about the shape of the objects, are able to provide a dense 3D reconstruction, with very low noise. Notably they are the first kind of monocular SLAM[2] that work in true scale[3], but still they don't provide any loop closure detection mechanics.

Williams et al. (2009) compares several types of loop closure detection for monocular SLAM: map-to-map, where correspondences are sought between features in two submaps, image-to-map where correspondences are sought between the latest frame from the camera and the features in the map and image-to-image, where correspondences are sought between the latest image from the camera and the previously seen images. They conclude that image-to-map systems perform best because such methods use as much information as possible, especially when combining image-to-image information and image-to-map information, but they have the problem of scalability. In a more recent survey Lowry et al. (2016) concluded that it is still a long way towards an universal place recognition system, but deep learning based techniques are the most promising at the current time.

3. APPROACH

The goal of this work is to build a standalone, global and highly reliable loop closure detection algorithm for monocular SLAM applications. The term standalone indicates the independence of the algorithm from other software and hardware systems in the robot. With the term global we indicate the ability to recover a loop closure independently from time and current position estimation. The term reliable means that the loop closure detection has to be robust and correct. For achieving a standalone algorithm only the images captured by the camera (and eventually the camera parameters) and the current single or multiple estimation of the position are exploited.
To achieve globality the information which can be obtained from the camera must, at least partially, retained within a database. Then the information or features coming from a new image has to be matched with the information stored previously in a database in order to detect a loop closure. To achieve reliability we employed a multi-step process where only images looking similar to each other were processed exploiting a robust point-wise matcher, which is robust regarding intensity changes.

A practical implementation of an algorithm would start by selecting the input images that will be further processed, since not every image contains useful information that can be exploited for re-localization (for example images where only a white wall is present). In addition, the image information database benefits from maintaining only relevant information, increasing its robustness and reducing the time needed for search operations. The processing then proceeds with an information extraction procedure followed by a matching procedure for identifying loop closure. After the matching attempt the obtained image information is also stored in the database, together with the single or multiple position estimation at the time when the image has been captured.

Considering related work, we can observe that feature based image matching methods have proven to be robust to scale, rotation, shift and illumination changes, but it's not computationally efficient to search for a similar image in a large database. Visual word based methods, thanks to the inverted index, provide a way to use features to search for similar images in a large database in a much more efficient way. However, one of the problems of visual words methods is that they require the generation of a vocabulary, which depends on the images supplied in the training phase. Another problem is that a match based only on features does not guarantee that the matched images belong to the same scene, given that matching features can be also present on partial and common repetitive patterns in the images. The main contribution of this paper is the use of a direct method that selects the images to undergo a further analysis with direct features matching, in order to avoid wrong image associations.

The direct method used is based on lossy image compression, which is achieved by reading the encoding layer of a convolutional autoencoder (CAE). CAEs are a kind of convolutional neural networks (CNN or ConvNet), which are deep, feed-forward artificial neural networks that have been used for feature extraction Masci et al. (2011) Geng et al. (2015) and image denoising Gondara (2016) Stowell and Turner (2015). The usage of a CAE still requires a training phase on a selected database, but as it will be shown, even a simple CAE is able to generalize very well, and so this method is able to provide loop closure candidates over very different types of images.

4. IMPLEMENTATION

In this section we describe the details of the implementation of the algorithm that was introduced above (sec. 3). In our implementation the images have been resized to 256×256 pixels. This choice was based on empirical tests with a convolutional neural network, that required images of fixed size as input. For higher resolutions the network did not converge properly within the training phase[4].

Here the proposed mechanism starts with the analysis of the output of a Canny edge detector applied to the resized image and the amount of non-zero pixels is calculated. The subsequent quality test consist in checking if the ratio between non-zero pixels and the total pixels of the image is larger than a threshold (4% in our implementation, the threshold has been chosen in accordance to the results of empirical tests, as it turned out that this threshold helps to avoid the further processing of images without enough information content). Images that have passed the quality test are then forwarded to two different processing stages: In the first SIFT (Lowe (2004)) is used to determine keypoints long with a descriptor, the second one is the convolutional autoencoder, which is used for generating a compressed representation of the images . The output of the most inner layer of the convolutional autoencoder

[2] Except inertial sensors aided visual SLAM
[3] The scale of the world cannot be observed and drifts over time in monocular SLAM, being one of the major error sources Engel et al. (2014)

[4] Machine learning problems are known to be sensible to the input space dimension Keogh and Mueen (2017)

(which in our implementation has a dimension of 4096) is then stored in an array together with the current best estimate of the position. Using a CUDA accelerated search with cosine distance metric the first 10 nearest compressed images are selected and their respective keypoint descriptor is retrieved from the database. Searching efficiently for similar data points in such high dimensions is an open problem in computer science Aggarwal (2001)[5]. In high dimensional spaces points essentially become uniformly distant from each other Aggarwal et al. (2001) when the Euclidean distance metric is used, and so different comparison metrics have to be employed. Generally these metrics do not have the strong and intuitive meaning that the Euclidean distance has. Exhaustive search operations are very inefficient on CPUs, as exhaustive search on CPU is an $O(n)$ operation, a CPU implementation is unlikely to work with real time performance. Given the simplicity and independence between each of the single operations of exhaustive searching, a parallel implementation on a modern GPU (that can run much more threads concurrently compared with a single CPU) keeps the observed search operation time close to constant. The cosine distance metric is one of the few metrics that can be applied in such a high dimensional space, and in this particular application it has proven to be a reliable metric. The keypoint descriptor of the current image is then compared to the keypoint descriptors of the 10 retrieved candidate images and finding 5 close descriptors indicate that the scene part has been seen before.

The strength of the algorithm is that the keypoint descriptor comparison is performed only between descriptors coming from images that are considered as similar by the autoencoder, thereby allowing an independent second verification of the scene similarity. In the current implementation the compressed image representation (obtained by the convolutional autoencoder) is finally inserted in the database.

In the following we describe the convolutional autoencoder (CAE) structure and training: The CAE that generates a compressed representation of the images, takes as input a RGB image with 256 × 256 pixels using normalized values in the interval $[0, 1]$. The encoding step starts with a convolutional layer based on 32 filters of size 3x3x3, and relu[6] as activation function, then max-pooling is executed with a pool size of 2x2. The same set of operations is repeated 3 times, each time with the output of the previous set of operations, with the only difference that the last convolution layer has only 16 filters, with the goal of further reducing the dimensionality of the encoding layer. To complete the autoencoder a decoding layer has to be implemented. A first thought is to perform an inverse operation of the pooling. Pooling is a sampling process that involves loss of information, so inverting it can involve zero-filling or interpolation, preventing such information from being completely recovered. While there is a strong consensus in the deep learning community about the superior performance of the max-pooling Scherer et al. (2010), there seems to be not such consensus about how to perform unpooling, so the strategy employed in this work is just a simple resizing with nearest neighbor interpolation. After the resizing layer a convolutional layer based on 32 filters of size 3x3x3 is in line, and relu as activation function, then again another equal resizing layer. The final structure of the network consists of a convolutional layer as previously presented and another convolutional layer where the numbers of filters is equal to the original image dimensionality. In this convolutional layer the activation function is a sigmoid function, that also represents the output of the network for the training phase. A zero-padding strategy is employed for every operation that involves sliding window operations, like convolution and pooling. At the start of the training procedure all the weights are initialized by sampling from a zero mean Gaussian distribution with standard deviation of 0.05. The training procedure involves minimizing the error between the target and the actual output of the network through a sigmoid cross entropy given logits[7] function, which allows the network to perform multi-class classification (see Google Tensorflow documentation (2018)), to act as an autoencoder. The optimization is done through a first-order gradient descent method, based on adaptive estimates of lower-order moments called ADAM Kingma and Ba (2014), which has been found to be one of the most successful optimization strategies by the deep learning community Goodfellow et al. (2016). Even if ADAM is a gradient descent method that is able to adjust the learning rate, it still needs a reference parameter for it, which has been set to 0.001. The dataset consists of 300 images of indoor and outdoor environments captured from a hand-held camera. The dataset was split in 70% training and 30% validation examples/images. The optimization was performed on mini-batches of 32 images at the time for 5000 epochs. At every epoch the loss on the validation set is calculated, finally the network weights which performs best on the validation set represent the final outcome of the training procedure.

5. EXPERIMENTS

Performance analysis in SLAM is a complex issue, especially when it comes to a stability analysis of the SLAM algorithm. Therefore, they are often produced by simulations or extended field tests. In order to benchmark the SLAM algorithm a selection of the KITTI dataset Geiger et al. (2013) is used, which contains large sequences of images coming from front looking stereo cameras mounted on a car. We tested our loop closure detection algorithm on the KITTI dataset (sequences 00 and 02) using only the left images. As we plan to use this loop closure detection method for underwater SLAM, we tested it also on an underwater SLAM dataset Scott Reef 25 from the Australian Centre for Field Robotics mar (2009). Given that the dataset is composed of stereo images, we choose again to use the left images.

For comparison we tested the FAB-MAP Cummins and Newman (2008) on the same datasets. There exists an improved version of it called FAB-MAP 2.0 Cummins and Newman (2011), but the implementation is not freely

[5] All current indexing techniques (based on space partitioning) degrade to linear search for sufficiently high dimensions Datar et al. (2004) Weber et al. (1998) Gionis et al. (1999)

[6] Rectified linear unit: $f(x) = x^+ = max(0, x)$

[7] Logits are functions that maps probabilities $x \in [0, 1]$ to $\mathbb{R}$ $y \in (-\infty, \infty)$

available to the public. FAB-MAP has been run with default parameters, both our algorithm and FAB-MAP have been run by accepting as loop closure candidates only frames that have at least 9 frames inbetween. As FAB-MAP provides a probabilistic value for each couple of frames to represent a loop closure, it's needed to choose a threshold for asserting which of the images represent a loop closure, this threshold has been set to 99%. We used for the tests an Intel Core i7-5820K with 32GB of RAM and a Nvidia TITAN GPU with 6GB of GDDR5.

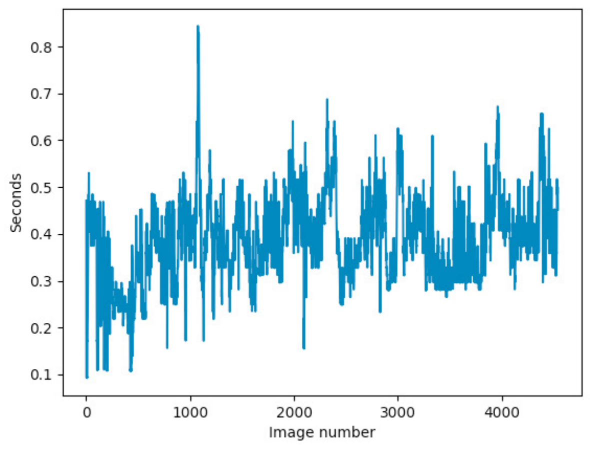

Fig. 1. Required time in seconds for processing a new image plotted against the images of KITTI dataset sequence 00.

Table 1 shows the results of the experiments. We observe a superior performance in terms of correctness of the identified loop closures on all the datasets, while FAB-MAP performs better in terms of computation time. The low performance of FAB-MAP on the underwater dataset is probably due to the fact that the used vocabulary is not adequate for the environment. Tests with different values for the main parameter of FAB-MAP (the true positive rate $p(z = 1|e = 1)$, that represent the probability of observing a feature given the existence of that feature) show little to no effect on the performance. Figure 1 shows a plot of the computation time for the images present in the sequence 00 of the KITTI dataset. The computation time of our solution can be approximated as a constant. Currently the algorithm may detect wrong loop closures candidates, like shown in figure 2 based on the observation and detection of single similar images in the video stream/sequence. This *perceptual aliasing* can be reduced or avoided if we also consider temporary information in the algorithm performing an additional consistency check for identified loop closure candidates.

Regarding the memory consumption of the algorithm, it is notable that each stored image occupies 16384 bytes (4094 32bit floats), assuming a frame rate of 30 frames per seconds and that every image pass the test of information content, the algorithm will allocate around 1.65 GB/hour of GPU memory. Newer GPUs support 16 bit float natively Ho and Wong (2017), which will immediately halve the memory consumption without any measurable performance impact on the algorithm.

Dataset	**Images size**	**No. of images**	**Autoencoder**	**FAB-MAP**	**FAB-MAP**	**FAB-MAP**
			described params	$P = \mathbf{0.35}$	$P = \mathbf{0.39}$	$P = \mathbf{0.45}$
			prec., time, loops	prec., time, loops	prec., time, loops	prec., time, loops
KITTI, sequence 00	1241x376	4541	99.7%, 1782s, 848	-	99.4%, 1226s, 344	-
KITTI, sequence 02	1241x376	4661	100%, 2189s, 302	-	94.5%, 1222s, 73	-
Scott Reef 25	1360x1024	9831	100%, 8825s, 15	21.1%, 7446s, 119	23.4%, 7403s, 111	22.7%, 7873s, 119

Table 1. The table shows the results of the loop closure detection experiments. P is $p(z = 1|e = 1)$ in FAB-MAP, that corresponds to the P_OBSERVE_GIVEN_EXISTS parameter in the FAB-MAP configuration file.

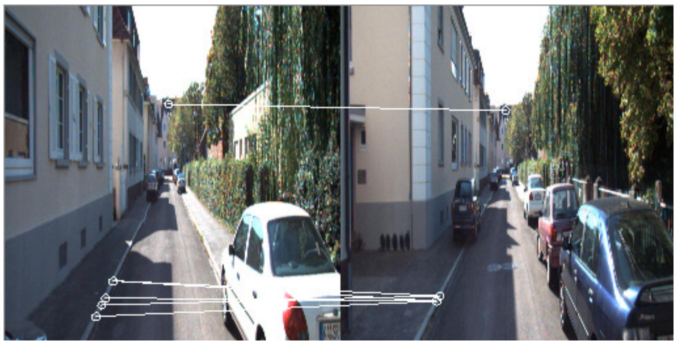

Fig. 2. Sub-optimal loop closure detection due to *perceptual aliasing*.

6. IMPROVEMENTS AND TAILORING

In this first implementation the memory on GPU is allocated and deallocated everytime a set of similar images has to be retrieved. This does take most of the computation time required for this operation. A more efficient solution would be to just add to the GPU memory the new images compressed by the autoencoder.

The convolutional autoencoder has great potential for improvements and further research must be conducted for finding an optimal network architecture for encoding images. Especially from a training point of view, for instance a multi step training would be beneficial, because even ADAM does still suffer from the vanishing gradient problem. Further parallelism can be achieved by splitting the image in multiple patches in order to run in parallel multiple instances of a single autoencoder that, due to the lower input dimensionality, would be easier to train.

Feeding inertial measurements to a Kalman Filter can provide a state covariance with a guaranteed physical meaning of state uncertainty (as the measurement error is bounded within a certain time period), so restricting the search of potential matches using the state covariance can improve robustness and speed up the search process.

As previously stated in the current implementation, at the end of the algorithm the compressed representation of the current image is added to the database. A more efficient solution would be to store a measure of the position state/keyframe uncertainty (a measure could be the magnitude of the state covariance matrix if an EKF is used for position estimation). And, after a loop closure, eventually replace the matched image in the dataset if the state uncertainty that comes with the new image is lower than the matched image currently in the dataset.
It has to be noted that together with position estimation also attitude could be stored, but current research in nonlinear observer theory has produced globally exponential stable estimators for attitude estimation Grip et al. (2015) and as IMUs today are very cheap, very small, and present in almost every electronic device, attitude estimation using computer vision is, for most of the applications, no longer necessary.

7. CONCLUSION

In this paper a global, standalone loop closure detector for monocular SLAM has been presented. Tests conducted on relevant datasets known to the SLAM community show very good performances in correct loop closure detection, with comparable computation time given parallelization on GPU.

Using compressed image representations for selecting which images attempt to match with direct features does indeed guard the loop closure detection algorithm from matching images given features that lie on repetitive patterns, and also provide a way to match directly feature descriptors instead of using visual words.

It has to be noted that our approach does not allow to detect loop closures with a large variance of the view points, but is exact where other loop closure detection systems are more likely to fail.

This work follows the direction of analyzing the use of deep learning for SLAM purposes, given that the SLAM community has found deep learning techniques to be currently irreplaceable for complex SLAM operations.

ACKNOWLEDGEMENTS

We would like to thank the Autonomous Vision Group part of the University of Tübingen for providing the KITTI dataset and the Australian Centre for Field Robotics marine robotics group for providing the Scott Reef 25 dataset. We would like to thank Trym Vegard Haavardsholm for his help in reviewing this work.

REFERENCES

(2009). Marine robotics datasets, australian centre for field robotics. http://marine.acfr.usyd.edu.au/datasets/. Accessed: 2018-01-15.

Aggarwal, C.C. (2001). On the effects of dimensionality reduction on high dimensional similarity search. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, 256–266. ACM.

Aggarwal, C.C., Hinneburg, A., and Keim, D.A. (2001). On the surprising behavior of distance metrics in high dimensional space. In *International conference on database theory*, 420–434. Springer.

Bailey, T. and Durrant-Whyte, H. (2006). Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3), 108–117.

Cummins, M. and Newman, P. (2008). Fab-map: Probabilistic localization and mapping in the space of appearance. *The International Journal of Robotics Research*, 27(6), 647–665.

Cummins, M. and Newman, P. (2011). Appearance-only slam at large scale with fab-map 2.0. *The International Journal of Robotics Research*, 30(9), 1100–1123.

Datar, M., Immorlica, N., Indyk, P., and Mirrokni, V.S. (2004). Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, 253–262. ACM.

Engel, J., Schöps, T., and Cremers, D. (2014). Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, 834–849. Springer.

Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR).*

Geng, J., Fan, J., Wang, H., Ma, X., Li, B., and Chen, F. (2015). High-resolution sar image classification via deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 12(11), 2351–2355.

Gionis, A., Indyk, P., Motwani, R., et al. (1999). Similarity search in high dimensions via hashing. In *Vldb*, volume 99, 518–529.

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *Data Mining Workshops (ICDMW), 2016 IEEE 16th International Conference on*, 241–246.

Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). *Deep learning*, volume 1. MIT press Cambridge.

Google Tensorflow documentation, T.d. (2018). Sigmoid cross entropy given logits. https://goo.gl/xeXVSY. Accessed: 2018-02-19.

Grip, H.F., Fossen, T.I., Johansen, T.A., and Saberi, A. (2015). Globally exponentially stable attitude and gyro bias estimation with application to gnss/ins integration. *Automatica*, 51, 158–166.

Ho, N.M. and Wong, W.F. (2017). Exploiting half precision arithmetic in nvidia gpus. In *High Performance Extreme Computing Conference (HPEC), 2017 IEEE*, 1–7.

Keogh, E. and Mueen, A. (2017). Curse of dimensionality. In *Encyclopedia of Machine Learning and Data Mining*, 314–315. Springer.

Kingma, D.P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Klein, G. and Murray, D. (2007). Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*, 225–234.

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2), 91–110.

Lowry, S., Sünderhauf, N., Newman, P., Leonard, J.J., Cox, D., Corke, P., and Milford, M.J. (2016). Visual place recognition: A survey. *IEEE Transactions on Robotics*, 32(1), 1–19.

Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J. (2011). Stacked convolutional auto-encoders for hierarchical feature extraction. In *International Conference on Artificial Neural Networks*, 52–59. Springer.

Mur-Artal, R., Montiel, J.M.M., and Tardos, J.D. (2015). Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5), 1147–1163.

Scherer, D., Müller, A., and Behnke, S. (2010). Evaluation of pooling operations in convolutional architectures for object recognition. In *International conference on artificial neural networks*, 92–101. Springer.

Stowell, D. and Turner, R.E. (2015). Denoising without access to clean data using a partitioned autoencoder. *CoRR*, abs/1509.05982.

Tateno, K., Tombari, F., Laina, I., and Navab, N. (2017). CNN-SLAM: real-time dense monocular SLAM with learned depth prediction. *CoRR*, abs/1704.03489.

Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., and Brox, T. (2017). Demon: Depth and motion network for learning monocular stereo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 5.

Weber, R., Schek, H.J., and Blott, S. (1998). A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, volume 98, 194–205.

Williams, B., Cummins, M., Neira, J., Newman, P., Reid, I., and Tardós, J. (2009). A comparison of loop closing techniques in monocular slam. *Robotics and Autonomous Systems*, 57(12), 1188–1197.