Vision-based Docking of a Mobile Robot

Andreas Kriegler, Wilfried Wöber
UAS Technikum Vienna
{mr18m016,woeber}@technikum-wien.at

Abstract. *For mobile robots to be considered autonomous they must reach target locations in required pose, a procedure referred to as docking. Popular current solutions use LiDARs combined with sizeable docking stations but these systems struggle by incorrectly detecting dynamic obstacles. This paper instead proposes a vision-based framework for docking a mobile robot. Faster R-CNN is used for detecting arbitrary visual markers. The pose of the robot is estimated using the solvePnP algorithm relating 2D-3D point pairs. Following exhaustive experiments, it is shown that solvePnP gives systematically inaccurate pose estimates in the x-axis pointing to the side. Pose estimates are off by ten to fifty centimeters and could therefore not be used for docking the robot. Insights are provided to circumvent similar problems in future applications.*

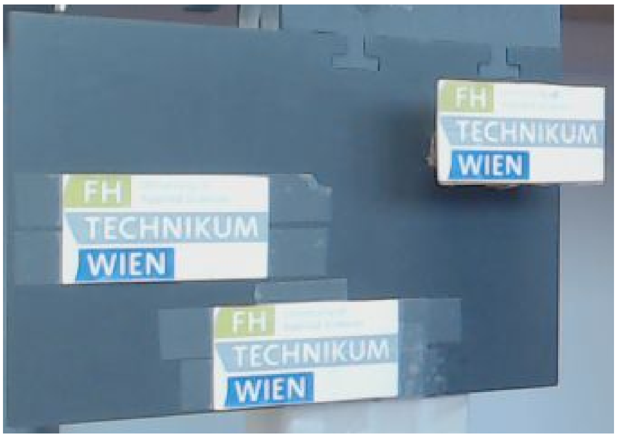

Figure 1. The visual target used for docking. The target location is on the ground infront. The origin for the PnP solvers is in the upper left corner. The logos are roughly 9x3 centimeters in size. The upper right logo was raised during experiments to remove coplanarity.

1. INTRODUCTION

Docking can be understood as the localization and navigation of a robot towards a target location [1]. In contrast to path-planning across larger distances, docking does not require obstacle avoidance methods but instead seeks highly accurate pose estimates [28]. As long as the pose of the robot and the target location are known in a reference coordinate system path planning algorithms can easily generate control commands. In the xy ground-plane, the pose $\vec{x}$ consists of three degrees of freedom, x, y, and θ as the rotation about its own axis z, and is described using the state at time t

$$\vec{x}_t = \begin{pmatrix} x & \dot{x} & y & \dot{y} & \theta & \dot{\theta} \end{pmatrix}_t^T \tag{1}$$

where $\dot{x}$, $\dot{y}$ and $\dot{\theta}$ describe the speed of the robot in x and y and its rotation respectively. As Thrun *et al.* [32] write outlining the motion model and measurement model, taking multiple control steps $\vec{u}_t$ with only an initial measurement or observation $\vec{z}_t$ leads to large uncertainties about its pose, they propose a measurement step after every control to restore confidence in the belief $bel(\vec{x})$. These measurements can be non-vision methods such as evaluating detections from LiDAR-scans [22] or can come from a camera setup providing visual feedback [6]. Yurtsever *et al.* [34] show in their survey on automated driving systems (ADS) that computer vision (CV) based approaches to navigation have become increasingly popular. Artificial landmark detection as described by Luo *et al.* [19] and gradient based optical flow [20] rival modern non-vision solutions. Classical non-vision systems typically employ LiDAR technology, indoor GPS or wireless fingerprinting [17]. While LiDARs are still widely used commercially (such as MiRs and Robotinos) recent advances in deep learning and their application in the ADS domain are of more scientific interest. Deep Convolutional Neural Networks (CNNs) have proven successful at tackling a variety of perception problems, including object detection [26] and pose estimation [30]. Open source implementations for different learning tasks are plentiful and can be used to provide perception for a robotics system. Due to the strong capabilities of CNNs as general feature extractors, it is possible to learn multiple visual targets which can be different depending on the environment or application. This relaxes the constraint of using specifically designed visual markers that classical CV methods pose. The learning task of the object detector in this work is comparatively simple (only one class of logos exist and they are easily distinguishable from the rest of the target, see Fig.1).

In previous work the LiDAR of the mobile robot, a robotino, was used to create a map of the environment and localization was implemented with the amcl package. While this pipeline in combination with obstacle avoidance methods has been useful for path-planning across the room, only employing the AMCL the robot arrives at the target position with great inaccuracy (10cm to 20cm). Therefore, for this project an entirely vision-based solution for docking was developed which is bound to take over the task of generating pose-estimates from the AMCL once the robot comes close to the docking target.

The aim of this work therefore is to approach and dock onto desired targets in a semi industrial environment with sufficiently high accuracy. To contribute to the transition of state-of-the-art CNNs from public datasets to real world problems an appropriate combination of old and new algorithms is presented in this work. A CNN based object detectors is used for image processing and object detection, followed by a camera pose estimation algorithm using point correspondences from the detections. The presented method could be easily adapted to learn new target positions outfitted with a visual marker with minimal setup requirements.

2. STATE OF THE ART

The problem of estimating the pose of a calibrated camera, assuming a known 3D scene, is known as the PnP-problem [29]. The idea is to use a feature detector such as SIFT [16] or SURF [2] to extract features from multiple sequential images. Since an image of a known 3D point gives two nonlinear constraints on camera pose and calibration, using three points (or more precisely three image-object point pairs) would give all 6 pose parameters. As [33] point out, such minimal cases lead to polynomial systems with multiple solutions, hence one additional point is used. This leads to four necessary points for estimating the pose (and one intrinsic parameter) and six points for estimation of 3D pose and five additional calibration parameters. The problem is formulated diffently for the planar two-dimensional or the general, aforementioned three dimensional case. Direct Linear Transformation (DLT, [9]) allows the estimation of the homography matrix $\boldsymbol{H}$ for the planar problem, requiring at least four 2D-3D point correspondences. For the general case, DLT estimates the projection matrix $\boldsymbol{P}$ and requires at least six such correspondences. In either case, $\boldsymbol{H}$ or $\boldsymbol{P}$ can be expressed with a set $\boldsymbol{A}\vec{x} = 0$ of multiple pairs of independent equations. Since individual pixels are generally noisy, no exact solution can be obtained using DLT, only an approximate solution by obtaining the SVD of $\boldsymbol{A}$. It should be noted, that for the noisy and overconstrained case, only the eigenvector of $\boldsymbol{A}^T\boldsymbol{A}$, corresponding to the smallest eigenvalue, should be computed. A continuation to DLT is the family of PnP algorithms. Efficient PnP or EPnP [14] uses the notion that each of the n 3D-2D point pairs are expressed as weighted sum of four virtual control points, and solves the pose problem from these control points. Perspective-Three-Point or P3P is a method applicable if only three correspondences are obtained, and in turn returns four real, possible solutions, the newest implementation being Lambda Twist P3P [25]. A fourth point pair can be used to remove this four-solution ambiguity.

Kartoun *et al.* [12] were able to achieve docking times averaging 85 seconds but attributed the success of their method to the unique hardware on the robot and a generously large docking station. Burschka *et al.* [3] take the aforementioned approach to the outdoors, using a Kanade-Lucas tracker [18] to track points in image sequences, followed by RANSAC and DLT. They achieve good results for rotation, but struggle with estimating translation. In the work of Mehralian *et al.* [21] an Extended Kalman Filter (EKF, [11]) is combined with PnP algorithms to create EKFPnP. They achieve better robustness against noisy features, although no details are given regarding the feature tracker.

In the field of deep learning, pose estimation is a well researched problem [23], camera pose estimation is less so [13] and no architectures or datasets exists specifically designed for docking a mobile robot. The dataset would need to include the complete pose of the robot for every captured image to allow end-to-end training. Instead, Shalnov *et al.* [30] were able to create a deep model using a CNN for camera pose estimation via object detections of human heads. In the work of Pavlakos *et al.* [24] a geometric approach to object pose estimation using semantic keypoints is taken but their published dataset only uses outdoor objects and is thus not applicable to docking. Lastly, as part of Zhou *et al.* [35]'s Centernet, they are proposing to regress from centerpoints to other object properties including pose but their framework is unnecessarily complex for the task at hand.

While the methods are numerous, no single framework exists that combines deep learning for object detections with a PnP-solver, all for the application of mobile robot docking. This work shows the hesitation of using CNNs for robot docking is unwarranted, as long as the learning task is managable in complexity.

3. METHODS AND IMPLEMENTATION

The robotino mobile robot used in this project was equipped with a Logitech C920 USB webcam. A remote desktop with an NVIDIA GTX 1080 GPU runs ROS to control the robot and process the images.

To showcase the flexibility of the pipeline regarding the visual target, no QR-tags or ARUCO markers [8, 27] were used. Three small paper printouts of a logo were instead fixed on a board roughly 20 by 15 centimeters in size and this board was used for training the detector. Video data was collected while arbitrarily moving the robot around close to the target. From the roughly 4500 recorded images 100 were selected to form the training set. The chosen images show the target from different viewing angles, distances, lighting conditions while a few images do not show the target at all to control for false positives. The bounding box coordinates of the three logos in all 100 images were manually annotated. Creating annotations took around three hours to complete. Resizing the images to 512x512 RGB-images allows the usage of Che *et al.* [4]'s toolbox with many different object detectors implemented.

Accuracy of the detector is important, since wrong detections would lead to wrong pose estimates and erroneous controls, while inference speed is important to enable a smooth docking, although inference times below 70 milliseconds are unnecessary, due to the bottleneck imposed by transporting 960x720 images from the camera to the remote desktop via Wi-Fi using the ROS image_transport package for compressed transfer. Looking at various speed-accuracy tradeoff comparisons between object detectors, Faster R-CNN [26] with pretrained ResNet [10] backbones seems to be a sweet spot, ResNet50 was chosen for this implementation. Faster R-CNN belongs to the class of detectors using a separate region proposal network to generate bounding box proposals. For the optimizer the default stochastic gradient descent with momentum of 0.9 was used and learning rate was kept default at 0.01 with a linear step learning rate scheduler and warmup. Other parameters and image augementation steps were kept default to Che *et al.* [4]'s configuration of Faster R-CNN for PascalVOC [5], including a 50 percent chance of a random horizontal flip. From the infered bounding boxes, image coordinates of the upper-left and lower-right corners of all three logos are saved for the PnP-solver. The Faster R-CNN network was trained for fifty epochs which amounted to 37 minutes training time on a GTX 1080 graphics card. GPU memory usage was 2GB showing a weaker graphics unit would suffice. Both bounding box and classification loss plateued after training for ten epochs.

The required pose estimate at timestep t for path planning can be described with the transformation matrix $\mathbf{T}_{base,t}^{target} \in \mathbb{R}^{4x4}$ from the base link of the robot to the target position near the station

$$\mathbf{T}_{base,t}^{target} = \begin{bmatrix} \mathbf{R} & \vec{t} \\ \vec{0} & 1 \end{bmatrix}_t \tag{2}$$

with $\mathbf{R} \in \mathbb{R}^{3x3}$ and $\vec{t} \in \mathbb{R}^{3x1}$ being the rotation matrix and translation vector to be estimated at sampling time t respectively. Physically measuring the transformation from the base link of the robot to the camera sensor as well as relating the logos at $\mathbf{K}_{logo}$ to the target location allows an estimated camera pose from a reference coordinate system on the logo-board $\mathbf{T}_{logo}^{camera}$ to be linearly tranformed into $\mathbf{T}_{base}^{target}$. Getting the transformation $\mathbf{T}_{logo}^{camera}$ with a calibrated camera and assuming the pinhole camera model means solving correspondences of points in 2D image space and those same points in the 3D real world. After calibrating the camera using the ROS *camera_calibration* package, the measured points in the object frame and saved image coordinates are combined in the Open-CV *solvePnP* algorithm using the intrinsic camera parameters. Available variations of the algorithm are *iterative*, which is the default method based on Levenberg-Marquardt optimization [15] to find a pose which minimizes reprojection error (sum of squared distances), *P3P* based on [7] which requires only four of the six point pairs and *EPnP* mentioned earlier. All three variations were tried and tested. The estimated rotation and translation vectors, after using *Rodrigues* to transform the rotation vector into the rotation matrix **R**, form $\mathbf{T}_{logo}^{camera}$ and therefore finally $\mathbf{T}_{base,t}^{target}$. As Siegwart *et al.* [31, p. 81ff] write, desired velocity can then easily be generated using estimated parameters k_ρ and k_α for a linear controller.

The entire pipeline can be quickly summarized as follows:

1. Create the visual target with arbitrary logos. Physically measure the logo corners and their position in relation to T_{logo}. Relate T_{logo} to T_{target} and on the robot T_{camera} to T_{base}.

2. To avoid bias in data collection, implement a random-walk in logo vicinity but constrain θ to enable the camera to face the logo most of the time. Annotate bounding-box coordinates of the logos for select images.

3. Train the Faster-RCNN object detector with this dataset. For docking, load the model and obtain bounding-boxes using ROS image-callbacks.

4. Use the inferred coordinates together with the measurements and intrinsic parameters of the camera in SolvePnP to obtain $T_{base,t}^{target}$ at every timestep t.

5. Use a simple linear controller to generate ROS motion control commands to guide the robot towards the docking target.

4. RESULTS AND DISCUSSION

During inference, processing a single image within the ROS pipeline takes the detector approximately 35ms. On average the detection would result in five bounding box proposals, sorting by confidence and extracting the top three boxes gives six image coordinates close to the ground thruth typically within one to four pixels. Evaluating the mIoU gives 96.3% for thirteen test images. Object detection results are therefore both accurate and confident. The PnP-solver, the second major component of the framework, proved to be more troublesome producing inaccurate results. All three implementations of the solvePnP algorithm express the translation vector $\vec{t}_{logo}^{camera}$ using the right-hand coordinate system $\mathbf{K}_{logo}$. Preliminary results quickly showed that all methods are accurate in estimating y and z translation, but struggle with the x coordinate. To get a better understanding of the pose estimates, in particularly the estimated translation vector, an extensive field study was conducted. The robot was steered towards six points and the ground truth translation and rotation were noted. These poses are described by $\mathbf{K}_{idx}$ in Fig 2 where $idx \in \{dock, amcl, left\_close, left\_far, right\_close, right\_far\}$. At each point fifteen images were captured, supplied to the Faster-RCNN model and the obtained image coordinates from bounding boxes, specifically six points per image, given to the PnP-solvers. After first results were analyzed, showing again large errors in x, changes were made in hopes of achieving more accurate pose results. In particular, the following major changes were made:

1. The upper right logo was raised from the plastic board to remove the coplanarity of all six points. By removing the coplanarity more information is available for estimating the camera pose [9].

2. Since solvePnP, unlike regular DLT, does not estimate intrinsics, they are a possible cause of error. The camera was recalibrated and the new parameters used. The focal lengths and distortion coefficients differed slightly.

3. The autofocus of the camera was turned off. Captured images were still sharp and logos clearly visible nonetheless.

Afterwards, the same study was undertaken, capturing sequences of fifteen images at six locations, and using the detector followed by solvePnP to obtain camera pose estimates again. The translation vectors were than saved and subsequently plotted to give a visual representation of the results. Figure 2 shows the results of this experiment in a 3D plot. The most notable thing here is the *iterative* algorithm flipped the signs for all three axis in almost all estimates. Its results are therefore point-symmetrical about the origin, a known issue when using solvePnP. Also of note is that the large error in the x direction still persists. This error occurs throughout all experiments and is not intuitive; the estimates in x are strangely placed. All points lie on the negative (right) half plane (with exception of some *iterative* estimates), but the estimates for the locations with ground-truth in the

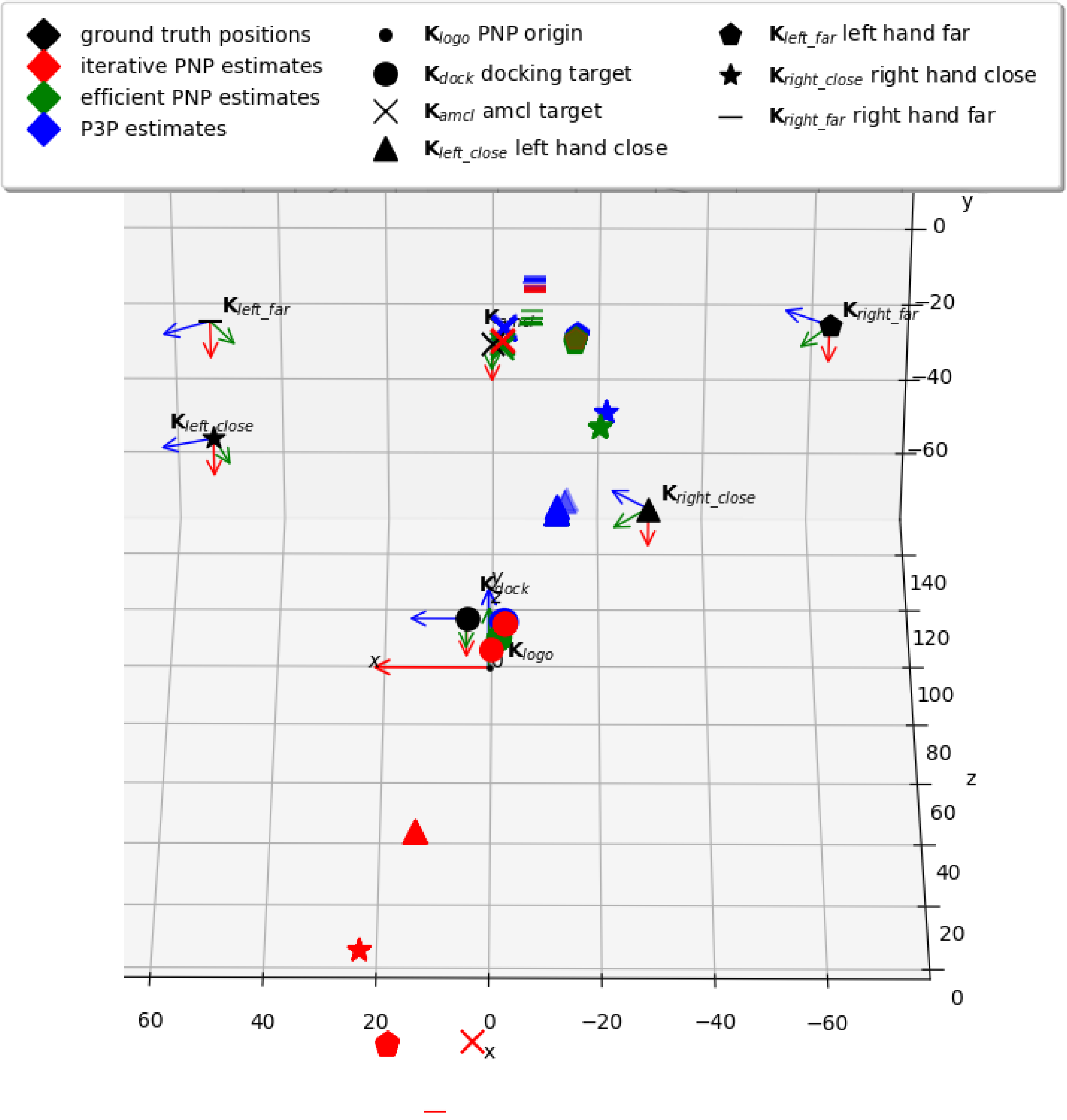

Figure 2. Pose estimates using the OpenCV solvePnP algorithm, for images captured at described six locations. Different colours are for different methods with black being ground truth locations. Different symbols mark the six different locations. Units are in centimeters. The 6 coordinate systems give the pose of the robot. The accuracy in estimating y and errors in x are similar as with the previous experiment. The point symmetry about the origin for the iterative algorithm is visible, having incorrectly flipped all three axis signs. Variance stays largely the same even with larger z.

left half are not simply mirrored across the z-axis. The distance gets consistently underestimated yet it seems with larger absolute value of x in ground-truth the absolute estimates in x also seem to increase. The estimates for the location $\mathbf{K}_{left\_far}$ break this pattern, being very close to the estimates for $\mathbf{K}_{amcl}$, the location where the vision based navigation is supposed to take over after using amcl localization. It can also be observed that variance only slightly increases about the estimates in z with increasing z distance. Clusters are very compact, an improvement compared to the first experiment. This can be attributed to using more precise intrinsic camera parameters. It is also visible that all algorithms are accurate for estimating the small offset in y.

Unfortunately, the reason for this seemingly systematic error in x could not be determined as of yet but considering the flipped signs for almost all estimates made by the *iterative* method, numeric instability is likely to contribute to the fragile nature of the solvePnP class.

5. SUMMARY AND OUTLOOK

In this work a novel framework for docking a mobile robot using only vision-based sensors and algorithms was developed. A CNN based object detector yielded bounding boxes of logos with high accuracy and confidence. Measurements of the logos were taken and related in a coordinate system. The family of solvePnP algorithms implemented in OpenCV was used to estimate the camera pose using the detector results and intrinsic parameters. All methods consistently estimated wrong distances in one of the directions, namely the x-axis. Following preliminary experiments, changes were made, in particular the coplanarity of the object points was removed and recalibration of the camera undertaken, and the same experiments run again. Unfortunately the errors persisted, although improvements regarding the scatterness of the pose estimates could be made. Consequently, no control commands were generated and docking of the robot could not take place in this instance. For future reference, it is important to note the fragility of the solvePnP algorithms. The source of the errors is unclear and while additional point pairs could improve results regarding compactness, it seems unlikely they could alleviate the large errors in predicting the x coordinates.

References

[1] F. Alijani. Autonomous vision-based docking of a mobile robot with four omnidirectional wheels, 01 2017. Master's thesis.

[2] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*. Springer, 2006.

[3] D. Burschka and E. Mair. Direct pose estimation with a monocular camera. In *International Workshop on Robot Vision*. Springer, 2008.

[4] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[5] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2), 2010.

[6] M. Fichtner and A. Grobmann. A probabilistic visual sensor model for mobile robot localisation in structured environments. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 2, pages 1890–1895. IEEE, 2004.

[7] X.-S. Gao, X.-R. Hou, J. Tang, and H.-F. Cheng. Complete solution classification for the perspective-three-point problem. *IEEE transactions on pattern analysis and machine intelligence*, 25(8), 2003.

[8] S. Garrido-Jurado, R. Munoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer. Generation of fiducial marker dictionaries using mixed integer linear programming. *Pattern Recognition*, 51, 2016.

[9] R. Hartley and A. Zisserman. Multiple view geometry in computer vision second edition. *Cambridge University Press*, 2000.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2016.

[11] R. E. Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Fluids Engineering*, 82(1), 03 1960.

[12] U. Kartoun, H. Stern, Y. Edan, C. Feied, J. Handler, M. Smith, and M. Gillam. Vision-based autonomous robot self-docking and recharging. In *2006 World Automation Congress*. IEEE, 2006.

[13] A. Kendall and R. Cipolla. Geometric loss functions for camera pose regression with deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2017.

[14] V. Lepetit, F. Moreno-Noguer, and P. Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2), 2009.

[15] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of applied mathematics*, 2(2), 1944.

[16] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2), 2004.

[17] J. Y. Lu and X. Li. Robot indoor location modeling and simulation based on kalman filtering. *EURASIP Journal on Wireless Communications and Networking*, 2019(1), 2019.

[18] B. D. Lucas, T. Kanade, et al. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th international joint conference on Artificial intelligence*, volume 2. IJCAI, 1981.

[19] R. C. Luo, C. T. Liao, K. L. Su, and K. C. Lin. Automatic docking and recharging system for autonomous security robot. In *2005 IEEE/RSJ Inter-*

national Conference on Intelligent Robots and Systems. IEEE, Aug 2005.

[20] C. McCarthy, N. Barnes, and R. Mahony. A robust docking strategy for a mobile robot using flow field divergence. *IEEE Transactions on Robotics*, 24(4), Aug 2008.

[21] M. A. Mehralian and M. Soryani. Ekfpnp: Extended kalman filter for camera pose estimation in a sequence of images. *arXiv preprint arXiv:1906.10324*, 2019.

[22] J. Moras, V. Cherfaoui, and P. Bonnifait. A lidar perception scheme for intelligent vehicle navigation. In *2010 11th International Conference on Control Automation Robotics & Vision*, pages 1809–1814. IEEE, 2010.

[23] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. *arXiv preprint arXiv:1502.06807*, 2015.

[24] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis. 6-dof object pose from semantic keypoints. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017.

[25] M. Persson and K. Nordberg. Lambda twist: an accurate fast robust perspective three point (p3p) solver. In *Proceedings of the European Conference on Computer Vision*. ECCV, 2018.

[26] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[27] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and vision Computing*, 76, 2018.

[28] J. Röwekämper, C. Sprunk, G. D. Tipaldi, C. Stachniss, P. Pfaff, and W. Burgard. On the position accuracy of mobile robot localization based on particle filters combined with scan matching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3158–3164. IEEE, 2012.

[29] G. Schweighofer and A. Pinz. Globally optimal o (n) solution to the pnp problem for general camera models. In *Proceedings of the 2008 British Machine Vision Conference*. BMVC, 2008.

[30] E. Shalnov and A. Konushin. Convolutional neural network for camera pose estimation from object detections. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 2017.

[31] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza. *Autonomous mobile robots*. MIT press, 2011.

[32] S. Thrun, W. Burgard, and D. Fox. *Probabilistic robotics*. MIT Press, 2002.

[33] B. Triggs. Camera pose and calibration from 4 or 5 known 3d points. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volume 1, pages 278–284. IEEE, 1999.

[34] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *arXiv preprint arXiv:1906.05113*, 2019.

[35] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. *CoRR*, abs/1904.07850, 2019.