Preprint of the paper which appears in:
European Conference on Mobile Robots (ECMR) 2021

Industrial Manometer Detection and Reading for Autonomous Inspection Robots

Jonas Günther Martin Oehler Stefan Kohlbrecher Oskar von Stryk

***Abstract*— Autonomous mobile robots for industrial inspection can reduce cost for digitalization of existing plants by performing autonomous routine inspections. A frequent task is reading of analog gauges to monitor the health of the facility. Automating this process involves capturing image data with a camera sensor and processing the data to read the value. Detection algorithms deployed on a mobile robot have to deal with increased uncertainty regarding localization and environmental influences. This imposes increased requirements regarding robustness to viewing angle, lighting and scale variation on detection and reading. Current approaches based on conventional computer vision require high quality images or prior knowledge. We address these limitations by leveraging the advances of neural networks in the task of object detection and instance segmentation in a two-stage pipeline. Our method robustly detects and reads manometers without prior knowledge of object location or exact object type. In our evaluation we show that our approach can detect and read manometers from a distance of up to $3\,\mathrm{m}$ and a viewing angle of up to $60°$ in different lighting conditions with needle angle estimation errors of $\pm 2.2°$. We publish the validation split of our training dataset for manometer and needle detection at https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2881.**

I. Introduction

Autonomous mobile robots play an essential role in future inspection tasks, especially in potentially hazardous environments such as oil and gas platforms. The use of robots not only helps to keep humans out of harms way. Moreover, they can collect consistent and high quality inspection data using numerous sensors, such as cameras, thermal imaging and gas sensors. Since many existing industrial plants are not digitized it is a necessity for those robots to be intelligent enough to also read analog gauges, with manometers being the most common. This task can be performed via teleoperation by streaming robot camera data, but this requires a continuous high bandwidth wireless connection to the robot and poses high burden and stress on remote operators for safe navigation in narrow environments. A much bigger benefit is unlocked if the robots can autonomously navigate and process sensor data, alleviating humans from permanently steering and monitoring them and allowing robots to carry out inspection tasks fully autonomously as shown in Fig. 1.

Fig. 1: Gauges in existing facilities are often not equipped with digital read-out capabilities. The ability to detect and read such dial indicators enables mobile robots to autonomously perform inspection tasks.

Autonomous mobile robots that are deployed on existing industrial plants operate in environments subject to a range of environmental conditions, resulting in challenging requirements on robustness of detections. Perfectly positioning the robot for inspection is not always possible, making it necessary to be robust against variations in viewing angle and scale. As the location of manometers might not be previously known, false detections have to be avoided.

Existing approaches rely on prior knowledge about the manometer in form of a reference image [1] or the manometer location. Other methods [2] make use of a circle detection with Hough transform which lacks robustness to different lighting conditions or mirror artifacts and therefore are error prone. These restrictions limit application where such prior knowledge may not be given or a high robustness is needed.

This work presents a new solution for dynamic detection and reading of manometers ready to use for real world applications. We only assume common properties found in industrial manometers. These are a circular shape, minimum and maximum values at $45°$ and $315°$ with the rotational axis of the needle at the center and a needle with a thinner tip than end.

We divide the problem into two steps: Detection of the manometer and read-out of the value. The detection step is solved by a Mask *Region-based Convolutional Neural Network* (R-CNN) implementation, which also generates object masks. These masks are then used in a post-processing

Jonas Günther, Martin Oehler and Oskar von Stryk are with the Simulation, Systems Optimization and Robotics Group, Computer Science Department, Technische Universitat Darmstadt, Germany. jonas_guenther@outlook.de
{oehler, stryk}@sim.tu-darmstadt.de
Stefan Kohlbrecher is with Energy Robotics GmbH, Germany. kohlbrecher@energy-robotics.com

Research presented in this paper has been supported in parts by the German Federal Ministry of Education and Research (BMBF) within the subproject "Autonomous Assistance Functions for Ground Robots" of the collaborative A-DRZ project (grant no. 13N14861) and by the German Federal Ministry of Economics & Technology's "EXIST Forschungstransfer" (grant no. 03EFLHE061). This work has been co-funded by the LOEWE initiative (Hesse, Germany) within the emergenCITY center.
978-1-6654-1213-1/21/$31.00 ©2021 IEEE

step to deskew the image and determine the needle angle.

II. Related Work

Our method builds on previous work performed in the fields of object detection and instance segmentation which we briefly discuss in this chapter.

A. Reading of Gauges

Relevant research concerning both, detection and reading of pointer gauges, has mainly been done for robotics challenges such as the *Autonomous Robot for Gas and Oil Sites* (ARGOS) Challenge. Hutter et al. [3] have addressed the detection of a manometer using a sliding window approach and *Histogram of oriented gradients* (HOG) features which get classified by a *Support-vector Machine* (SVM). They use a reference image of the manometer and match *Scale-invariant feature transform* (SIFT) features to estimate a homography for warping the image. To read the value a circle and line detector is used to detect the center point and needle line of the manometer. Merriaux et al. [1] follow a similar approach. Having a reference image of the gauge they use SIFT descriptor matching to detect it in the image taken by the robot. Next, a homography is estimated to deskew the image and line detection is applied to find the needle and read the manometer. More recently Hilario et al. [2] used a circle detector to detect manometers and a line detector to detect the needle and read the value. Dumberger et al. [4] use a neural network based on *You Only Look Once* (YOLO) [5] to detect manometers. Further, they detect the needle and dial of the gauge with circle and line detectors and then convert the image to polar coordinates to finally read the value. They also apply optical character recognition to find the minimum and maximum values of the gauge. Their pipeline is missing the step of deskewing though as it is work in progress. An end-to-end deep learning approach is proposed by Lin et al. [6] which uses a *Convolutional Neural Network* (CNN) to detect a pointer gauge and classify its value. Further research that concentrated on automatic reading of a pointer gauge was conducted by Chi et al. [7] and Ye et al. [8]. [7] only consider upfront, high quality images and use a region growing method to first obtain the dial of the gauge. Subsequently, the image is transformed to polar coordinates with the dial center as its origin and the scale is extracted. Using line detection to find the needle the gauge can then be read. [8] take a different approach by subtracting two images of a gauge with different pointer position to determine its origin. The needle is detected after preprocessing with a line detector and then the value is read.

B. Object Detection and Instance Segmentation

The problem of localizing and categorizing objects is very important to gain knowledge from images. The task is usually to find and classify a bounding box (x, y, w, h) that encapsulates the object contained in an image. Instance segmentation is an even finer task as it aims to predict the pixel mask of an object. Much research has been done to solve these tasks and benchmarks such as the *Common Objects in Context* (COCO) [9] challenge or PASCAL *Visual Object Classes* (VOC) [10] have been created. These benchmarks are nowadays dominated by *Deep Neural Network* (DNN) solutions which can mostly be divided into one- and two-stage approaches.

One stage approaches see object detection as a pixel-wise classification task or a regression problem. Liu et al. [11] use a fully convolutional approach with their *Single Shot Detector* (SSD) network and add several feature layers after a backbone network which predict offsets to predefined template boxes. Redmon et al. [12] propose the YOLO network whose basic idea is to divide the input image into a $s \times s$ sized grid and then each cell predicts the object centered in it as well as b bounding boxes along with their confidence scores. They implement this with a fully convolutional DNN achieving a high inference speed. The network has been further refined in the following years up to the current version 3 which introduced a better backbone classifier and multi-scale predictions [5]. *You Only Look at Coefficients* (YOLACT) was introduced by Bolya et. al [13] in 2019 with the goal to be able to do instance segmentation in real-time. They propose a fully convolutional architecture that does two tasks in parallel: Generation of mask prototypes and prediction of mask coefficients, which subsequently produce the final masks by linear combination.

Two stage approaches usually first search *Regions of Interest* (ROIs) which then get classified. The most prominent detectors in this category are the family of R-CNN. First introduced by Girshick et al. [14] in 2014 the R-CNN uses selective search [15] to obtain ROIs and then extracts feature vectors using a CNN which finally get classified. In following years further improvements have been made with Fast R-CNN [16] by generating feature maps from the whole image through a CNN and using feature pooling to generate a single feature vector. Faster R-CNN added the *Region Proposal Network* (RPN) to the architecture which is a small CNN that is used in a sliding window over the feature map of the input image to generate ROIs. This approach allows for an end-to-end training of the entire network and achieved state of the art on PASCAL VOC and COCO [17]. For instance segmentation, Mask R-CNN replaces the ROI pooling layer with a new ROI align operation to enable pixel-to-pixel alignment of ROI features and the input image. An additional mask head then predicts the mask. Further improvements to Mask R-CNN have been made by introducing a separate scoring of the masks [18].

C. 6D Pose Estimation

The goal of 6D pose estimation is to determine the rigid transform from the object coordinate system to the camera coordinate system, consisting of the object orientation in 3D and its 3D translation. If the 6D pose of a manometer is known, it is possible to transform an image of it so that it is seen upfront. Peng et al. [19] propose a network called PVNet that regresses unit vectors pointing to possible keypoints for each pixel in an image. Keypoints are derived with a *Random Sample Consensus* (RANSAC) voting and are

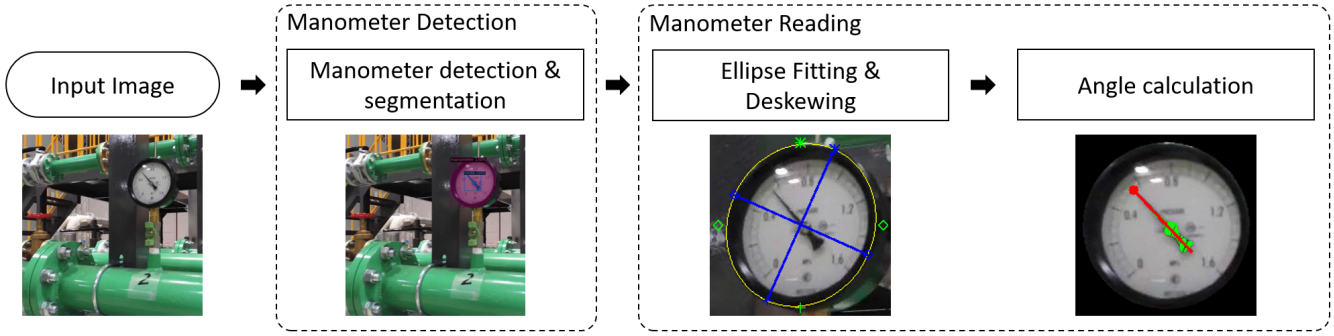

Fig. 2: The proposed pipeline for manometer detection and reading.

subsequently used for solving a perspective-n-point problem to obtain the 6D pose.

Another approach is brought forward by Xiang et al. [20] with PoseCNN. It is a multi-task CNN that predicts the semantic labels, estimates the 3D translation by detecting the object center and regresses the 3D orientation to a quaternion representation. To estimate the translation, known camera intrinsics are assumed.

Sundermeyer et al. [21] make use of autoencoders to learn implicit representations of object orientations from sample images of an object, handling symmetries well. They propose a pipeline that uses an object detector for localization of the object and subsequent estimation of the 6D pose by the autoencoder.

III. Method

This chapter presents the design of the developed approach. A two stage pipeline is proposed for improved transparency and modularity. The first stage is realized with a Mask R-CNN network for instance segmentation. The mask is then used in the second stage to deskew and read the manometer.

A. Two Stage Detection and Reading Pipeline

The two main parts of our pipeline are the detection and segmentation of manometers and the subsequent reading of the manometer value as visualized in Fig. 2. The pipeline is constructed as follows: Images from the robot's camera are sent to an instance segmentation DNN. The network creates pixel-wise masks for each detected manometer and its needle.

Viewed from an angle, the round shape of a manometer is projected to an ellipse. We use this relationship to fit an ellipse to the detected manometer mask and deskew the image by warping it to a circle.

In the final step, the angle of the needle and manometer value is determined by fitting a line to the needle mask.

The advantages of this design are twofold. Firstly, one can replace the instance segmentation network and method depending on the desired speed/accuracy trade-off or when new methods show large improvements. Secondly, trust in the results is increased as intermediate steps can be inspected. This adds transparency compared to end-to-end solutions. The following sections explain each step in more detail.

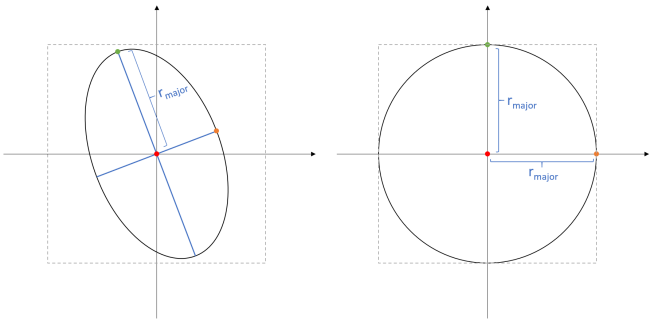

Fig. 3: Schematic view of finding point correspondences. The marked border is a quadratic bounding box around the ellipse.

B. Manometer Detection

For the detection of the manometers and their needle, Mask R-CNN as detailed in [22] is used since it promises very accurate predictions while maintaining a reasonable inference speed. With the usage of ResNet-101 [23] in a *Feature Pyramid Network* (FPN) [24] backbone architecture an even higher robustness to scale variation can be achieved. In contrast to approaches used in the ARGOS challenge [3] [1] a manometer detection with a DNN does not need prior knowledge about the manometer or its position and also promises a higher robustness to scale and view angle variation than classic feature descriptors. If multiple manometers are detected in this step the subsequent reading step is performed for each of them.

C. Manometer Reading

The assumption that manometers have a circular geometric shape when seen upfront is used to deskew the image. Viewed from an angle, the circular shape projects to an ellipse. Using *least squares* regression an ellipse is fitted onto the instance mask of the manometer. Given the ellipse the goal is to stretch the minor axis of the ellipse to the length of the major axis so that it becomes a circle. The necessary scaling is an affine operation and can be expressed by warping the image with an affine transformation matrix $T \in \mathbb{R}^{2\times3}$.

To determine the transformation matrix, three point correspondences are needed as it has 6 Degrees of Freedom. One set of points is formed by the ellipse center and the end points of the major and minor axis. The correspondences are the center of the quadratic bounding box and the closest points intersecting its sides in horizontal and vertical direction in r_{major} distance as visualized in Fig. 3. Using these point correspondences also results in a rotation of the image if the ellipse is rotated. We rotate it back by first multiplying the transformation matrix with a rotation matrix of the inverse ellipse angle.

Next, the image and the needle mask are warped with T resulting in a deskewed view. Then a line is fitted onto the points of the needle mask with *least squares* regression. The line end points are obtained by using the maximum and minimum coordinates of the mask points depending on the needle orientation. To determine the tip of the needle we make the assumption that the part of the needle opposite to the tip is thicker than the tip. Therefore, we count the mask pixels in the neighborhood of the two end points in a window of size $s \times s$ where s is a configurable parameter. The end point with less needle mask pixels in its neighborhood is assumed as tip of the needle.

Finally, the needle angle is calculated as follows:

$$angle_{needle} = \text{atan2}(y_{end} - y_{tip}, x_{end} - x_{tip}) \quad (1)$$

We assume the angle of the minimum and maximum value $(angle_{min}, angle_{max})$ to be $45°$ and $315°$ respectively to determine the manometer value in percent:

$$value_{pct} = \frac{(angle_{needle} - angle_{min})}{angle_{max} - angle_{min}} \quad (2)$$

Optionally, if the minimum and maximum values $(value_{min}, value_{max})$ are known the numeric value of a linear scale can be calculated:

$$\begin{aligned} value = {} & value_{min} \\ & + value_{pct} * (value_{max} - value_{min}) \end{aligned} \quad (3)$$

IV. Implementation

This chapter briefly discusses our implementation. First, a dataset is created which subsequently is used for training of the Mask R-CNN implementation.

A. Creating a Dataset

For training a supervised machine learning algorithm the most important asset is a sufficiently large amount of high quality ground-truth data. There are already multiple large scale datasets for instance segmentation available like COCO [9], Open Images [25] or Pascal VOC [10]. After examination of various datasets, none have been found with a class for manometers or gauges. Therefore, a new dataset has been created with images of manometers. Manometers have been photographed in different positions and backgrounds with different cameras to ensure a high variance in images.

After first training experiments, the dataset has been extended with background-only images containing round objects to reduce false positives. To produce ground-truth information, the images have been annotated by hand.

The final dataset consists of 175 labeled images and 63 background-only images. Since this still is only a small amount of data compared to the needs of deep learning algorithms, image augmentation is used. Each image is augmented with gaussian noise, motion blur, fog, compression and clouds as implemented in the Python library *imgaug* [26], extending the dataset to a total of 1050 labeled images and 378 background only images.

B. Training

We use the Mask R-CNN implementation *Detectron2* [27] by *Facebook AI Research* (FAIR) which has built-in support for learning from background only images. Due to the dataset still being comparatively small the network is initialized with pre-trained weights on the COCO dataset provided by FAIR. The first 2 stages of the ResNet FPN backbone are frozen so that they keep their learned feature extraction abilities. All other layers are then fine tuned on the manometer dataset. We split our dataset into a training and validation set at a ratio of approximately 80/20 and additionally use random flipping and rotation of the input images. Training is performed with *Stochastic Gradient Decent* for $270\,000$ iterations. The learning rate is set to 2.5×10^{-4} and lowered after the $210\,000$th and $250\,000$th iterations to 2.5×10^{-5} and 2.5×10^{-6} respectively. Additionally, momentum of 0.9 is used. Training is performed on a Nvidia RTX 2080 Ti which has capacity for a mini batch size of four images.

C. Reading

The reading step has been implemented in Python 3 for the *Robot Operating System* (ROS) framework. We filter manometer detections that have no needle detection contained in their bounding box as an additional false positive safeguard and implement an option called *zoom* onwards. If this option is enabled, manometers without an associated needle are cropped out of the image and reevaluated by Mask R-CNN. Otherwise they are ignored. This improves detection accuracy of manometer images taken from a large distance as the cropped image gets upscaled before being passed to Mask R-CNN. For reading we assume the manometer to be upright in the image. The robots *Inertial Measurement Unit* (IMU) or *Simultaneous Localization and Mapping* (SLAM) can be used for correction of the image orientation.

V. Evaluation

Our processing pipeline is composed of two separate steps, detection and reading. We evaluate each step individually. Due to a missing benchmark for manometer detection and reading we describe the setup of our evaluation and the analyzed parameters. Lastly the results are shown and discussed.

A. Manometer Detection

To evaluate the detection step of our pipeline we use the validation split of our dataset. This verifies that our approach works with manometers in different environments. We use

(a) The manometer is mounted on a neutral background.

(b) Angles are marked on the floor by hand.

(c) The camera is positioned on the measuring points using a lead.

Fig. 4: The setup used for evaluation.

the COCO evaluation metrics and report *average precision* (AP)@[IoU=0.5:0.95] (averaged over IoU thresholds in 0.05 steps) averaged over all classes and per class for the detection and segmentation task respectively - see Table I.

The relatively low AP on the bounding box prediction is due to the training with random rotation which increases the size of bounding boxes. Our validation split is available at https://tudatalib.ulb.tu-darmstadt.de/handle/tudatalib/2881 for future comparison with other research.

B. Manometer Reading

To our best knowledge there is no existing benchmark on manometer detection and reading as this is a very specific task. Therefore we create our own evaluation benchmark. The main parameters which make a reliable reading challenging for mobile robots have been identified by us as lighting, view angle and distance to the gauge. In order to test the capabilities of the presented approach systematically against these variables and to determine their influence on the measurement result the following controlled test environment as shown in Fig. 4 has been created. A manometer model which has not been part of our training data is installed on a neutral background (see Fig. 4a) at a height of $94\,\mathrm{cm}$ (measured at its mounting screw) and the camera is mounted on a tripod at $1\,\mathrm{m}$ height and aligned with one foot of the tripod. Then the angles are marked on the floor in a resolution of $10°$ as shown in Fig. 4b and lines are drawn for each angle up to a distance of $3\,\mathrm{m}$. Due to the radial symmetry of the manometer it is sufficient to evaluate $90°$ on one side to show the influence of the perspective on the measurement result. Measuring points are marked on the lines in $0.5\,\mathrm{m}$ steps for each angle resulting in 60 measuring points total (6 points per angle). Images are taken with the camera by positioning it on a measuring point with the help of a lead as shown in Fig. 4c and placing the camera-aligned tripod foot on the line as well. The parameter of lighting is difficult to standardize.

TABLE I: Results on the validation split. AP is averaged over IoU thresholds [0.5:0.95] in 0.05 steps.

	AP	$AP_{manometer}$	AP_{needle}
Segmentation	43.306	63.351	23.261
Detection	35.940	53.024	18.856

Therefore only 2 different lighting scenarios have been used for the evaluation. One set of images has been taken during daylight and another set during night with artificial lighting. All images are taken with a Panasonic Lumix DMC-GX80 with a $12\,\mathrm{mm}$ wide-angle lens and automatic settings at a resolution of 4592×3448 pixels.

Due to the manual marking of angles and lines an angle error of $\pm 0.5°$ has been measured at $2\,\mathrm{m}$ distance. The total measurement error resulting from manual marking, positioning of the camera and alignment of the camera with the tripod foot is expected to be $< 1.5°$ and $< 5\,\mathrm{cm}$. The ground-truth manometer value is determined as $13.6\,\mathrm{bar}$. All images from the measuring points are passed into our ROS implementation. We use the *zoom* option since the images have a high resolution which results in them being resized to about $1/4$ of their original size when fed into the network. In images from a larger distance the manometer is already small compared to the image size. Resizing and interpolation has the effect that the needle and its features are underrepresented in the image that gets analyzed by the network so that the needle can't be detected anymore which is required for successful reading.

a) Results: We calculate the deviations from the ground-truth manometer value as shown in Table II and Table III for the different lighting conditions respectively. It can be seen that manometer values can be read with a very low deviation of maximum $\pm 0.13\,\mathrm{bar}$ which is equivalent to $\pm 2.2°$ needle angle from a view angle of up to $70°$ and a distance up to $3\,\mathrm{m}$.

TABLE II: Deviations in bar from the ground-truth manometer value for the artificial-lighting evaluation dataset.

Deviation in bar							
		Distance in m					
		0.5	1	1.5	2	2.5	3
Angle in °	0	-0.06	-0.01	0.00	0.02	-0.02	0.02
	10	-0.04	0.01	-0.00	-0.02	0.04	0.03
	20	-0.09	-0.01	-0.01	-0.02	0.04	0.10
	30	-0.03	0.06	0.07	0.05	0.09	0.09
	40	0.00	0.05	0.07	0.08	0.05	0.05
	50	0.07	0.10	-10.71	0.07	0.05	0.12
	60	0.02	0.00	0.03	-0.01	0.02	-0.03
	70	-	-	-	-	-	-
	80	-	-	-	-	-	-
	90	-	-	-	-	-	-

TABLE III: Deviations in bar from the true manometer value of the daylight evaluation dataset.

Deviation in bar							
		Distance in m					
		0.5	1	1.5	2	2.5	3
Angle in °	0	0.04	0.04	0.07	0.04	0.10	0.03
	10	0.05	0.06	0.09	0.03	0.07	0.08
	20	0.04	0.13	0.09	0.07	0.08	0.09
	30	0.04	0.05	0.08	0.04	0.11	0.02
	40	0.04	0.10	0.05	0.08	0.03	0.03
	50	0.04	0.03	0.06	0.06	0.04	0.04
	60	0.07	0.05	0.06	0.06	-0.01	0.05
	70	0.11	-	-	0.11	0.06	-
	80	-	-	-	-	-	-
	90	-	-	-	-	-	-

The reason for the large deviation at $50°$ in the artificial lighting scenario is a very inaccurate needle mask, where the tip of the needle is thicker than its end. We expect that this kind of error can be minimized with a larger training dataset. Overall we find that detection and reading worked well up to an angle of $60°$ with up to $3\,\mathrm{m}$ in both lighting scenarios.

b) Program runtime: The time needed for computation of the different steps of the implemented pipeline is measured with the *timeit* Python module. It is observed that more than $98\,\%$ of the computation time needed is consumed by running inference with the Mask R-CNN network as is shown by Table IV. On an embedded platform like the Nvidia Jetson Xavier NX this takes about $1.13\,\mathrm{s}$ so that the detection could run at $0.8\,\mathrm{Hz}$ on a robot. Note that these values are without any optimization and results may vary through different CUDA versions. Also note that for these measurements the *zoom* option was disabled. Using it will add an additional runtime of *detection & segmentation* per detected manometer without needle.

Compute Platform	Program Component	
	detection & segmentation	*reading*
CPU only (Intel i7-7700K)	$3.52\,\mathrm{s}$	$< 0.1\,\mathrm{s}$
Nvidia Jetson Xavier NX (CPU + GPU)	$1.13\,\mathrm{s}$	$< 0.1\,\mathrm{s}$
AMD Ryzen 3960X + Nvidia RTX 2080 Ti	$0.14\,\mathrm{s}$	$< 0.1\,\mathrm{s}$
Intel i7-7700K + Nvidia GTX 1060	$0.2\,\mathrm{s}$	$< 0.1\,\mathrm{s}$

TABLE IV: Runtime of the different program components on different platforms. GPUs use CUDA 10.2.

VI. Conclusion

We have presented a new approach for manometer detection and reading which meets the requirements of mobile robots. Our approach leverages the advances in instance segmentation by DNNs and has shown to be very robust to viewpoint changes in terms of distance and angle as well as to different lighting conditions with a remarkably low reading error. Our benchmark implementation is ready to run on an embedded system with a frequency close to $1\,\mathrm{Hz}$ making it ready to be used on mobile robots.

References

[1] P. Merriaux, R. Rossi, R. Boutteau, V. Vauchey, L. Qin, P. Chanuc, F. Rigaud, F. Roger, B. Decoux, and X. Savatier, "The vikings autonomous inspection robot: Competing in the argos challenge," *IEEE Robotics & Automation Magazine*, vol. 26, no. 1, pp. 21–34, 2018.

[2] J. Hilario, C. Penaloza, D. Hernandez-Carmona, J. Balbuena, D. Quiroz, J. Ramirez, and F. Cuellar, "Development of a mobile robot for inspection of analog gauges in industrial plants using computer vision," in *Int. Conf. CIS and Conf. RAM*, 2019, pp. 209–214.

[3] M. Hutter, R. Diethelm, S. Bachmann, P. Fankhauser, C. Gehring, V. Tsounis, A. Lauber, F. Guenther, M. Bjelonic, L. Isler *et al.*, "Towards a generic solution for inspection of industrial sites," in *Field and service robotics*. Springer, 2018, pp. 575–589.

[4] S. Dumberger, R. Edlinger, and R. Froschauer, "Autonomous real-time gauge reading in an industrial environment," in *Int. Conf. ETFA*, vol. 1. IEEE, 2020, pp. 1281–1284.

[5] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[6] Y. Lin, Q. Zhong, and H. Sun, "A pointer type instrument intelligent reading system design based on convolutional neural networks," *Frontiers in Physics*, vol. 8, p. 580, 2020.

[7] J. Chi, L. Liu, J. Liu, Z. Jiang, and G. Zhang, "Machine vision based automatic detection method of indicating values of a pointer gauge," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[8] X. Ye, D. Xie, and S. Tao, "Automatic value identification of pointer-type pressure gauge based on machine vision." *JCP*, vol. 8, no. 5, pp. 1309–1314, 2013.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*. Springer, 2014, pp. 740–755.

[10] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *IJCV*, vol. 111, no. 1, pp. 98–136, 2015.

[11] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *ECCV*. Springer, 2016, pp. 21–37.

[12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. CVPR*, 2016, pp. 779–788.

[13] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "Yolact: Real-time instance segmentation," in *Proc. ICCV*, 2019, pp. 9157–9166.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. CVPR*, 2014, pp. 580–587.

[15] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *IJCV*, vol. 104, no. 2, pp. 154–171, 2013.

[16] R. Girshick, "Fast r-cnn," in *Proc. ICCV*, 2015, pp. 1440–1448.

[17] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *TPAMI*, vol. 39, no. 6, pp. 1137–1149, 2016.

[18] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring r-cnn," in *Proc. CVPR*, 2019, pp. 6409–6418.

[19] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proc. CVPR*, 2019, pp. 4561–4570.

[20] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," *arXiv preprint arXiv:1711.00199*, 2017.

[21] M. Sundermeyer, Z.-C. Marton, M. Durner, and R. Triebel, "Augmented autoencoders: Implicit 3d orientation learning for 6d object detection," *IJCV*, vol. 128, no. 3, pp. 714–729, 2020.

[22] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proc. ICCV*, 2017, pp. 2961–2969.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. CVPR*, 2016, pp. 770–778.

[24] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. CVPR*, 2017, pp. 2117–2125.

[25] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov *et al.*, "The open images dataset v4," *IJCV*, pp. 1–26, 2020.

[26] A. B. Jung, "imgaug," https://github.com/aleju/imgaug, 2020, online; accessed 10-Dec-2020.

[27] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick, "Detectron2," https://github.com/facebookresearch/detectron2, 2019.