Efficient and Robust Orientation Estimation of Strawberries for Fruit Picking Applications

Nikolaus Wagner, Raymond Kirk, Marc Hanheide and Grzegorz Cielniak

Abstract— **Recent developments in agriculture have highlighted the potential of as well as the need for the use of robotics. Various processes in this field can benefit from the proper use of state of the art technology [1], in terms of efficiency as well as quality. One of these areas is the harvesting of ripe fruit.**

In order to be able to automate this process, a robotic harvester needs to be aware of the full poses of the crop/fruit to be collected in order to perform proper path- and collision-planning. The current state of the art mainly considers problems of detection and segmentation of fruit with localisation limited to the 3D position only. The reliable and real-time estimation of the respective orientations remains a mostly unaddressed problem.

In this paper, we present a compact and efficient network architecture for estimating the orientation of soft fruit such as strawberries from colour and, optionally, depth images. The proposed system can be automatically trained in a realistic simulation environment. We evaluate the system's performance on simulated datasets and validate its operation on publicly available images of strawberries to demonstrate its practical use. Depending on the amount of training data used, coverage of state space, as well as the availability of RGB-D or RGB data only, mean errors of as low as 11° could be achieved.

I. Introduction

In recent years, global developments have highlighted the benefits brought to modern agriculture by state-of-the-art (SOTA) technology. Rising cost and decreasing availability of human workers, most recently emphasised by the shortage of seasonal workers caused by the COVID-19 pandemic, introduce the need for a higher degree of automation in industrial agriculture [2]. Due to that, the appeal of increased use of robotics for agriculture becomes apparent [3].

One possible application for automation is the harvesting of ripe fruit. This task implicitly poses several technical challenges, ranging from the detection of the fruit to be harvested over the path planning for the gripper up to the actual step of harvesting and storing. In this paper, we focus our attention on the first part, the recognition and detection of the ready to harvest fruit, specifically of ripe strawberries.

Strawberries, like many other varieties of fruit but unlike most man-made objects, have several characteristics which can pose a difficulty during harvesting. Primarily, they are relatively soft and bruise easily, which makes grasping them a challenge. But also the perception aspect poses difficulties. While the 2D-detection in RGB images is relatively well-addressed in prior work [4][5], estimation of the 6D-pose still remains a problem. Having access to the full pose, however, is essential for an ideal execution of the harvesting process. More specifically, optimal path planning not only requires full 3D-positions, which can for example be obtained by projecting 2D-detections into 3D-space utilising an additional depth sensor [6], but also the rotational information needs to be considered to avoid damaging the produce. Therefore it becomes imperative to accurately estimate the orientation of strawberries for the application of automated harvesting. The challenges introduced by this task and possible solutions will be discussed in more detail in the following sections.

In order to overcome the aforementioned difficulties, we propose a set of novel contributions including:

- a compact and efficient network architecture capable of estimating the rotational configuration of rotationally symmetric objects, in our case individual strawberries, from colour and additional depth image cues;
- a system capable of auto-generating RGB & depth (RGB-D) data with corresponding segmentation maps in simulation, which we use for the training process, covering large variations in lighting, appearance and variety of fruit;
- validation of the proposed framework on publicly available datasets of real strawberries.

II. Related Work

The idea of estimating the rotation of an object is of course not new; several conventional methods, using local or global descriptors, have been tried and tested in the past [7][8]. More recently, learning-based approaches have been evaluated as well. Park et al. [9] propose a learning based combination of local and global descriptors, capable of estimating the full 6D poses of objects from cluttered images, using 2.5D image data and CAD models of the objects of interest. Xiang et al. [10] propose *PoseCNN*, which estimates the full 6D pose of objects as well, using RGB images with the addition of depth maps for the purpose of registration, using ICP, after the initial regression step. *RotationNet* by Kanezaki et al. [11] takes a different approach, using only RGB images taken from multiple viewpoints of models which the network then learns to estimate the poses of.

Pose estimation applications for agriculture have been researched thoroughly as well; Guo et al. [12] present a point cloud-based algorithm using 3D-reconstruction and an ICP approach for estimating poses of fruit which works well, but is computationally expensive and comparatively slow. Lin et al. [13] detect and estimate the poses of guavas using a low-cost RGB-D sensor with reasonable accuracy, but their approach is relatively slow as well; this is due to the plant-specific characteristics requiring the authors to also reconstruct branches in order to avoid collision. Strawberry foliage on the other hand can be pushed aside by a gripper and therefore we do not face this issue. Eizentals

& Oka [14] investigate pose estimation of bell peppers, but their algorithms are based on LIDAR-data, which limits their applicability because the necessary, typically expensive, sensors have to be available.

In our approach, we choose a novel configuration of low-cost input data, rotation annotation style and an estimation tool more compact and specialised than existing solutions, hence more portable and faster. We address the issue of rotational symmetry and circumvent it by using a custom annotation style for the rotational configuration. Furthermore, we implement a way to generate arbitrary amounts of RGB-D training data with segmentation maps from simulation and also investigate the possible performance on real data.

III. METHODS

This section describes the individual components of our system including the types of sensors providing the input data, orientation representation, the proposed orientation estimation network architecture and a simulation tool for automated training and data annotation.

A. Sensors & Data Formats

Most object detection frameworks use RGB or RGB-D cameras as an input [15][16]. So, in order to minimise the need for further hardware, an orientation estimation tool should ideally also utilise these sensors. Furthermore, many industrial camera systems already provide off-the-shelf (OTS)-solutions for RGB-D sensors [17][18][19], which increases their attractiveness for the application in question. Therefore, we also choose an RGB-D camera as the input for our orientation estimator. This also affords the option to run it on RGB data alone, albeit with decreased accuracy.

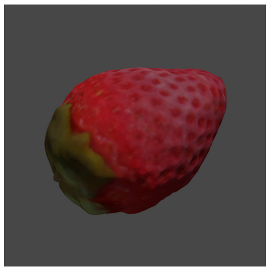
B. Rotation Parameters

For the format of the orientation output once again application-specific constraints have to be taken into consideration, the most important one being that the objects of interest, i. e. strawberries, are roughly rotationally symmetric around a defined main axis. Thus, an estimation of popular orientation representations like roll, pitch and yaw (RPY)-angles [20] would fail, since vastly different combinations of angles can lead to very similar views.

This would confuse an estimation tool, therefore we resort to estimating the parameters of a unit vector pointing from the stem to the tip, i. e. along the main rotational axis, of the strawberry. By doing so, we avoid the aforementioned problems and obtain unique representations for every possible view. This concept is illustrated in Fig. 1.

C. Orientation Estimator

After defining the input and output data formats, we finally focus on the actual tool estimating rotation parameters from RGB-D images. Since we consider object detection and image segmentation a solved problem, as mentioned in Section I, we decide to only feed images of individual berries into our estimation tool in order to reduce the complexity of the required architecture. We therefore only need a relatively


(a) Rendering obtained from the rotational configurations: RPY: 300° / 10° / 30° equivalent to XYZ: 0.35 / 0.82 / 0.49

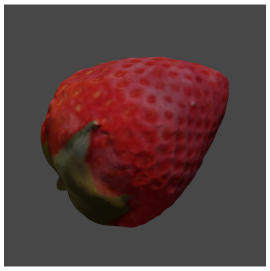

(b) Rendering obtained from the rotational configurations: RPY: 50° / 40° / 170° equivalent to XYZ: 0.27 / 0.79 / 0.49

Fig. 1: Renderings of two different rotational configurations from the same viewpoint of the same object.

simple image feature extractor with a three-dimensional regression output so we can estimate an orientation for each individual berry. Many solutions exist for the image feature extraction task, each with individual benefits and peculiarities [21]. We choose a VGG16-style network [22], as it is a well-established feature extractor with all the capabilities needed for our application. We add a three-dimensional linear regression output instead of the typical one-hot-encoded classification output of standard VGG16-networks [22], thus adding the capability of estimating the three parameters of a unit vector as discussed in Section III-B. A schematic of the resulting rotation estimator is illustrated in Fig. 2.

D. Loss Function

To train the network, we additionally require a loss function to evaluate errors made by the network when predicting an orientation. Xiang et al. introduce *ShapeMatch-Loss* [10], which focuses on matching the 3D shape of an object. This is a good approach for man-made objects which are always of the same shape, very much unlike plants. Also, comparing 3D shapes is typically computationally expensive, making it a poor choice for our use case.

Park et al. [9] represent the poses of objects of interest as quaternions and calculate the loss as a simple Euclidean distance of two quaternions. This is a lightweight approach capable of dealing with natural, non-man-made objects, but based on quaternions, unlike our rotation representation.

We therefore create our own custom loss function which combines robustness with computational efficiency. For two vectors, one being the ground truth and one being the estimation of the rotational configuration for a single object, a straightforward choice for a loss function would be the angle between them. However, we also want to be able to use the loss value for training. For this, values between 0 and 1 are preferable over raw angles, so training algorithms can be kept agnostic of the type of problem. Due to other problems which have to be considered when designing a loss function [23], we resort to using a modified version of the cosine similarity for quantifying the quality of the

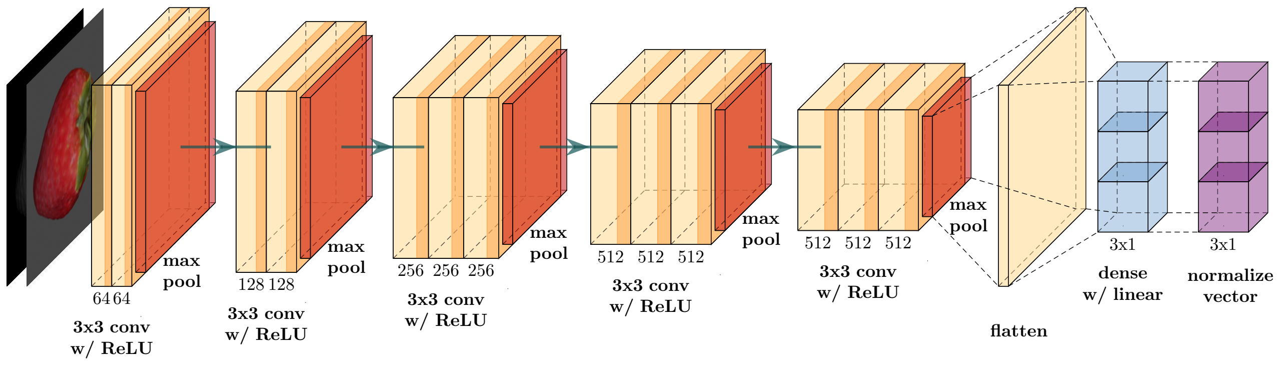


Fig. 2: The modified VGG16 network used for rotation estimation.

estimations. Cosine similarity itself is defined as the cosine of the angle between two vectors:

$$cos.\ similarity = cos(\phi) = \frac{u \cdot v}{|u| \cdot |v|}$$

where u and v are the two vectors in question and ϕ is the angle between them. With slight modifications, this function instead gives us a loss value between 0 and 1 for each set of directional vectors, with 1 being two vectors pointing in completely opposite directions and 0 being two vectors with identical directions. The loss function for this becomes:

$$I = \frac{1 - cos(\phi)}{2}$$

with ϕ as above and I being the loss value. A graphic representation of this loss function is illustrated in Fig. 3.

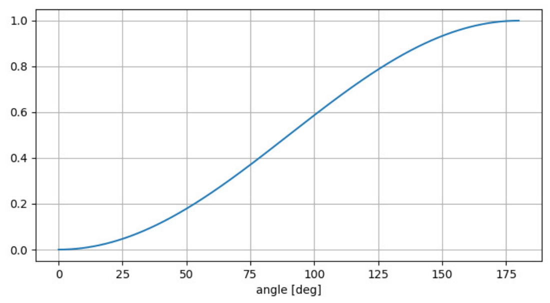


Fig. 3: The custom loss function based on the cosine similarity.

Theoretically, we could also use simple linear scaling to obtain error values between 0 and 1. However, our loss function creates a nonlinear relationship between the angular error and the error value used for training, which is preferable since our loss function does not ”punish” smaller errors as strictly, leading to easier convergence and less overfitting.

E. Realistic Simulation Environment

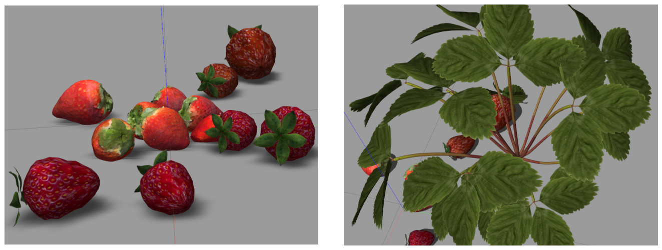
Nowadays, a large number of datasets is publicly available for many applications, but most of them are only annotated in terms of location, segmentation and type of objects of interest. For our application, however, further annotation is required. In order to train our network to estimate the rotation parameters discussed in Section III-B, we need an annotated strawberry-dataset with rotational parameters.

Annotating datasets manually is a tedious task, especially concerning rotation parameters. While the manual annotation of, for example, segmentation masks can be done sufficiently well manually, annotating rotational information is another matter. The time needed for the same number of images is vastly higher and the quality of annotations significantly lower, since without a proper perspective view a lot of guessing is required for an untrained person.

Due to all of these reasons, we resort to creating new, artificial datasets, obtained from simulation-based renderings, which contain all the annotations we need for training and testing our rotation estimation tool. Since we are running a robotic simulation, most of our previous work is based on the well-established Gazebo-simulator [24]. In order to minimise additional overhead, we also choose this simulator as a basis for the creation of our artificial training datasets.

Naturally, we need to provide simulation models of the objects we want to generate datasets of. After obtaining point clouds of strawberries with the help of a multi-view stereo imaging system as in [25] and refining them using Blender software [26], we end up with models usable in common simulation environments, including Gazebo. The renderings in Fig. 1, for example, were obtained using the strawberry models thus generated. We are, however, not limited to individual berries; to create more diverse datasets, we can also include cluttered scenes, foliage and the like in the simulation environment so that this may also be represented in the dataset. Possible examples are shown in Fig. 4.

To evaluate the performance of the model on real data, we manually annotated some strawberry images in terms of orientation as well. For this, we made use of an existing database featuring high-resolution images of strawberries [27] and added the necessary annotations manually. This dataset contains 1611 different berries of 15 varieties in total. However, it does have the downside of not providing depth information, which means that we can only evaluate



(a) A cluttered scene. (b) A scene with foliage.

Fig. 4: Examples of non-individual berries which can be simulated and automatically annotated by the proposed system.

the network configuration trained on RGB data alone.

The mentioned database also has the additional downside that all images are taken from 22 discrete viewpoints, identical for all berries. This means that, while large, the dataset is not very diverse in terms of rotational configurations. Furthermore, all images in this dataset have been taken in controlled conditions, i. e. with identical backgrounds and very similar lighting conditions. This again highlights the benefits of the automated annotations tools, as pointed out previously. Due to the sparsity of suitable real datasets, we still choose this dataset as an evaluation basis, as it offers a good middle ground and can be annotated in terms of rotation from the prior knowledge of the viewpoint from which each image was taken, as mentioned above.

IV. Evaluation

This section describes the evaluation methodology used to assess the performance of our system including data collection, evaluation metrics and experimental results.

A. Data Collection

Using Gazebo, it is possible to simulate colour- and depth-cameras with little effort; ready to use models, featuring both sensor types, already exist. The Intel RealSense cameras [18], for example, provide Gazebo plugins for easy simulation of this system. We choose to use these cameras with their simulation plugins, since they are robust, well-established and a popular choice in robotics. Gazebo is also capable of simulating sensor noise, but for the sake of simplicity we have chosen not to add any to this proof-of-concept example.

We are now still missing one crucial feature for creating artificial datasets: using Gazebo, it is not directly possible to create segmentation maps for rendered images created from the simulated cameras. This causes problems because it does not allow us to fit the bounding boxes for the annotations correctly to the objects of interest. In order to overcome this issue we have created a Gazebo-plugin [28], which, in simple terms, performs a form of raytracing for each pixel in the image in order to create a full segmentation map with unique IDs for each object of interest. We thus can obtain RGB-D images with corresponding segmentation maps as well as the state information (containing position and orientation) for each object of interest, as they are being published by Gazebo. We use this information to automatically annotate the image stream from the simulated cameras with the information obtained by the segmentation plugin and the Gazebo state publisher. We then export the resulting data as a ready-to-use dataset in a COCO-like format [29]. This allows us to create and annotate as much data as necessary or desired with all the features we require. Some examples of annotated berries in such a dataset are shown in Fig. 5.

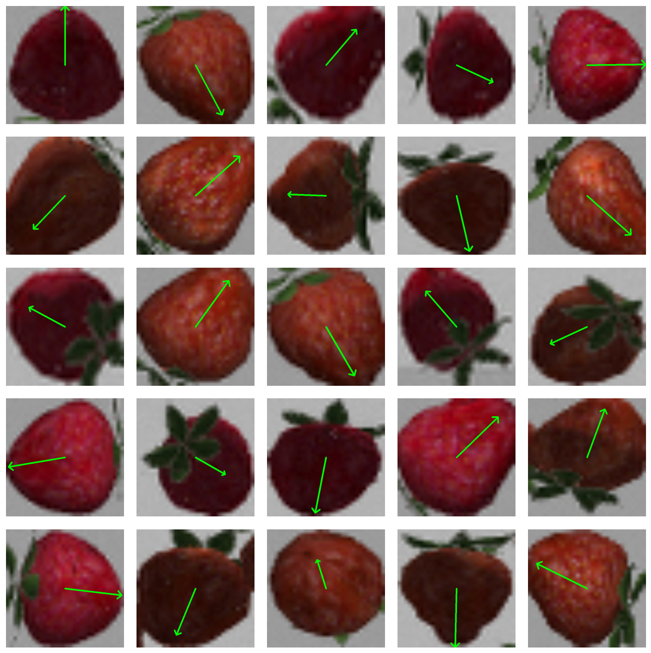


Fig. 5: Example images of training data generated by the automated annotation tool. The orientation annotation is depicted as an arrow projected onto the 2D colour images.

The tool, however, is obviously not limited to such images as shown here, since it can automatically annotate anything that can be simulated in Gazebo. This means that lighting conditions and other environmental factors can be varied arbitrarily for the purpose of data augmentation.

B. Evaluation Metrics

We train and test the proposed rotation estimation network on a set of simulated data generated using the tools and methods introduced in Sections III-E and IV-A. While the renderings for the evaluation are being obtained from the same simulation environment with the same strawberry models being used, all other conditions are being chosen to be as different as possible from the training data: different lighting conditions and different backgrounds were used to create the images, and most importantly, some rotational configurations, which had not been present in the training set, are being chosen. These conditions are selected in order to verify as well as possible that the network does not overfit on the simulated images due to their low variance in comparison to real data. For this example, no further image augmentation was used in order to analyse the performance achievable from the unaltered, simulation-generated datasets alone.

The quality of an estimation made by the network is evaluated in terms of the angle between ground truth and the network's estimate of the orientation of a berry. Furthermore, we manually annotate a real dataset, i. e. a number of images of real strawberries, in terms of rotation, to test the network's domain transfer capabilities and performance on real data. Finally, we measure the inference time of individual samples for all data. The given performance is measured when run on an NVIDIA GeForce GTX 1660 Ti [30].

C. *Performance on Simulated Data*

The distribution of angular errors resulting from the test sets is illustrated in Fig. 6. For these results, the network was trained on 3050 samples using a 9:1 validation split. The test dataset consisted of 97 different samples. The training and test processes were conducted on the RGB-D data as well as the RGB data only. The results of both variants are shown.

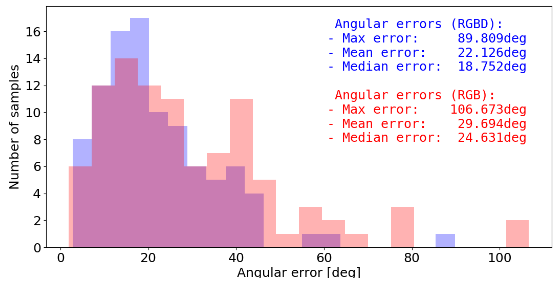


Fig. 6: Absolute performance of the rotation estimator on the simulated data in terms of the angular errors made when making a prediction. Performances of estimators trained only on RGB data vs. those trained on RGB-D data are shown.

The error distribution, with a mean error of roughly 22.1° and a median error of around 18.8° when trained on RGB-D data, illustrates the feasibility of the implemented approach. If trained only on RGB images without the additional depth component, the results become comparatively worse with a mean error of 29.7° and a median error of 24.6°. We therefore can conclude that using RGB-D data does indeed provide a benefit over using RGB data only. The inference time for individual samples was measured to be around 12ms.

Furthermore, we also observe that the network consistently performs slightly worse on some samples than on others (note the presence of the second peak around 40° errors in Fig. 6 for both evaluated estimators), which calls for further investigation. We therefore resort to analysing the training data used and relating it to the performance of the estimator on the test data by creating a 2D-state-space of azimuth and elevation of the directional vector estimated by the network. This allows us to observe the coverage of the state space by the training data used and to relate the performance of the network to the coverage. This is illustrated in Fig. 7.

In this graph, coverage of the state space achieved by the training data provided is illustrated by a coloured background; in areas where training data did not cover the state space sufficiently, the background is left white. Smaller dots illustrate test samples in the state space. The performance of the network for each individual sample is illustrated by lightness of the dots; darker dots correspond to better, lighter

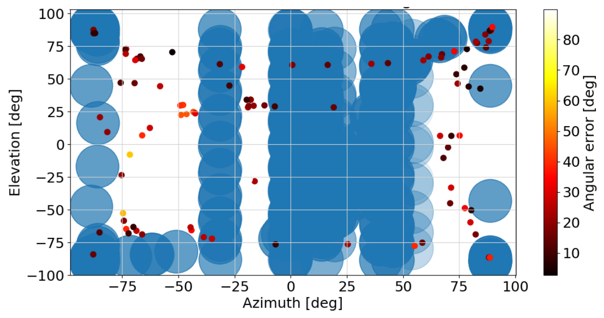


Fig. 7: Performance on test data in relation to available training data. Areas with available training data are coloured blue; for some parts of the state space no training data was provided intentionally in order to evaluate the influence. Performance is encoded in the smaller dots, with lighter colours corresponding to a higher degree of error.

dots to poorer performance. Another conclusion can thus be drawn from this illustration: areas with a state space more densely populated by training data also yield a better performance on the test data, which matches the expected behaviour. Therefore, we conclude that the network did indeed learn to deduct the rotational configuration from an individual rendering successfully.

Fig. 8 shows examples of good and bad estimates, allowing for another conclusion: the network appears to perform better on side views of fruit than on angled, bottom or top views. We can thus assume a better visibility of the berry tip orientation correlates with easier deduction by shape alone. Improved usage of available depth data, e.g. by usage of an alternative feature extractor for depth maps, could enhance performance on other views. Especially viewpoints with the berry tip pointing towards the camera may benefit from this, however further research is needed to confirm.

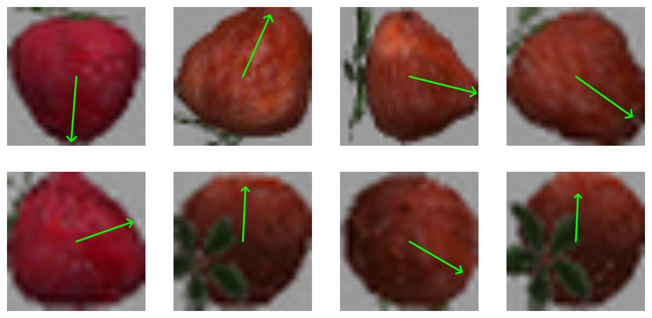


Fig. 8: Orientation vectors estimated by the network from simulated data, projected onto RGB images. Four good (top) and bad (bottom) estimations each are given.

In order to achieve better performance for other viewpoints, further effort needs to be put into providing more or better training data. For a more consistent performance, the aim should also be to cover all of the state space illustrated in Fig. 7 with sufficiently diverse training data, so that the network may learn to handle every possible viewpoint.

D. *Performance on Real Data*

To evaluate performance on the real dataset described in Section III-E, we train the network on 5000 real samples using a 9:1 validation split and test it on 200 different, real samples. The results for this are given in Fig. 9 in the same way as previously in Fig. 6.

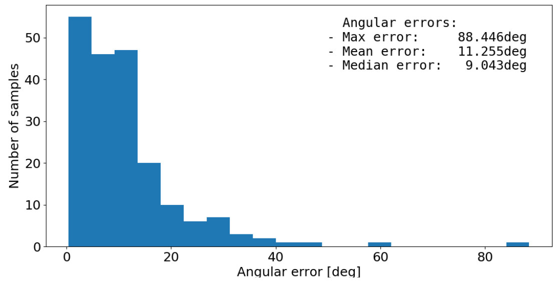


Fig. 9: Absolute performance of the rotation estimator on real data in terms of the angular errors made when making a prediction. Only RGB data has been used.

We observe significantly better performance on this dataset than on the simulated one, with a mean error of 11.3° and a median error of 9°. To discover the reason for this, we relate the test samples to the training samples like in Fig. 7. An illustration for the real data, in line with the illustrations given for simulated data, is provided in Fig. 10.

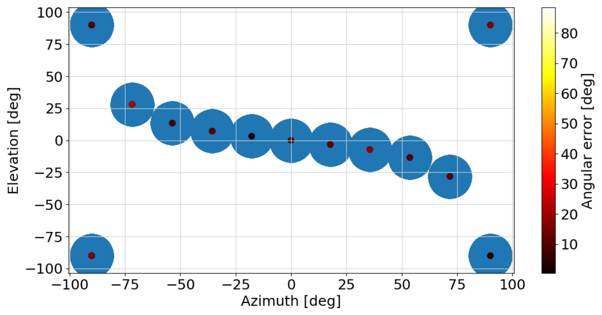


Fig. 10: Performance on test data in relation to available training data. Areas with available training data are coloured blue. Performance is encoded in the smaller dots, with lighter colours corresponding to a higher degree of error.

In these training data, the state space is very sparsely populated due to reasons mentioned above. This means that the network learns to estimate poses present therein very well, but would likely fail on rotational configurations absent therefrom. Furthermore, there are no cluttered scenes or foliage either. We can circumvent these problems by producing arbitrarily large, diverse datasets in simulation. To illustrate the real dataset and the network's performance on it, some good and bad estimates are shown in Fig. 11, in line with the simulated examples in Fig. 8.

This figure also highlights that a dataset is only as good as the annotations provided. In this case, bounding boxes are chosen very narrowly which leads to rather aggressive cropping of the images, possibly not giving enough context for the network. This is another issue which can be circumvented by generating artificial datasets; The size of the bounding boxes can be defined to include as much context as desired.

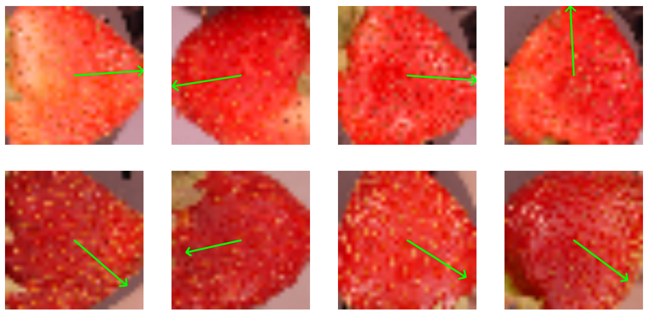


Fig. 11: Orientation vectors estimated by the network from real data, projected onto RGB images. Four good (top) and bad (bottom) estimations each are given.

However, due to the very specific lighting conditions and varieties of berries present in the real data, which differ largely from the models currently available to us, training the network on simulated and performing estimates on real data leads to poor results. To achieve proper domain transfer, more models are needed.

Finally, inference time was measured again for real samples and was found to be around 7ms. The lower inference time for this dataset is likely due to more samples being evaluated and common overhead being more evenly distributed.

V. Conclusions

In this paper, we have presented a compact and portable feature extraction network trainable with the help of a custom loss function in order to be able to estimate the orientation of individual objects, specifically strawberries. We have also presented a novel approach to generating diverse training data from robotic simulation environments. Plugins for the Gazebo simulation tool have been introduced which allow the automated generation of COCO-style-datasets with colour and depth information, segmentation maps as well as bounding box information. The auto-generated training data has been used to train and evaluate the proposed network.

It has been shown that the estimation tool introduced was capable of deducing rotational configurations of simulated berries after being trained on a different set of simulated data. The estimator was then tested on some manually annotated, real data and the capabilities of learning to estimate the poses of real data have been shown.

The proposed network is a compact and efficient tool suitable for real-time applications due to its low inference time of around 10-15ms per object.

Future work could include the investigation of possible benefits of using further and more training data, be it from simulation or real, to enable the network to achieve proper domain transfer from simulation to real data as well. Finally, we call for the inclusion of more information, such as rotational configurations of objects, into newly generated datasets, so that more training data for this type of problem may be more easily available.

References

[1] I. Robotics and A. Society. Agricultural robotics & automation. [Online]. Available: https://www.ieee-ras.org/agricultural-robotics-automation

[2] European Trade Union Confederation (ETUC), "National measures targeting seasonal workers to address labour shortages (particularly in the agricultural sector)," Briefing Note, ETUC, May 2020. [Online]. Available: https://www.etuc.org/sites/default/files/publication/file/2020-05/Covid-19 Briefing Seasonal Workers Final_updated 29 May 2020.pdf

[3] T. Duckett, S. Pearson, S. Blackmore, and B. Grieve, *Agricultural Robotics: The Future of Robotic Agriculture*, ser. UK-RAS White Papers. UK-RAS Network, 6 2018.

[4] R. Kirk, G. Cielniak, and M. Mangan, "L*a*b*fruits: A rapid and robust outdoor fruit detection system combining bio-inspired features with one-stage deep learning networks," *Sensors*, vol. 20, no. 1, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/275

[5] X. Shen, "A survey of object classification and detection based on 2d/3d data," *ArXiv*, vol. abs/1905.12683, 2019.

[6] N. Burrus. Kinect calibration. [Online]. Available: http://nicolas.burrus.name/index.php/Research/KinectCalibration

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

[8] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE transactions on pattern analysis and machine intelligence*, vol. 27, pp. 1615–30, 11 2005.

[9] K. Park, J. Prankl, and M. Vincze, "Mutual hypothesis verification for 6d pose estimation of natural objects," in *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 2192–2199.

[10] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox, "Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes," 2018.

[11] A. Kanezaki, Y. Matsushita, and Y. Nishida, "RotationNet: Joint Object Categorization and Pose Estimation Using Multiviews from Unsupervised Viewpoints," 2018.

[12] N. Guo, B. Zhang, J. Zhou, K. Zhan, and S. Lai, "Pose estimation and adaptable grasp configuration with point cloud registration and geometry understanding for fruit grasp planning," *Computers and Electronics in Agriculture*, vol. 179, p. 105818, 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169920314046

[13] G. Lin, Y.-C. Tang, X. Zou, X. jun tao, and J. Li, "Guava detection and pose estimation using a low-cost rgb-d sensor in the field," *Sensors*, vol. 19, p. 428, 01 2019.

[14] P. Eizentals and K. Oka, "3d pose estimation of green pepper fruit for automated harvesting," *Computers and Electronics in Agriculture*, vol. 128, pp. 127 – 140, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169916307050

[22] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014.

[15] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu, and M. Pietikäinen, "Deep learning for generic object detection: A survey," 09 2018.

[16] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, pp. 1–21, 01 2019.

[17] Z. Zhang, "Microsoft kinect sensor and its effect," *IEEE MultiMedia*, vol. 19, no. 2, p. 4–10, Apr. 2012. [Online]. Available: https://doi.org/10.1109/MMUL.2012.24

[18] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel realsense stereoscopic depth cameras," 2017.

[19] Occipital, Inc. Structure sensor - 3d scanning, augmented reality and more for mobile devices. [Online]. Available: https://structure.io/structure-sensor

[20] R. Grassmann and J. Burgner-Kahrs, "On the merits of joint space and orientation representations in learning the forward kinematics in se(3)," in *Robotics: Science and Systems*, 2019.

[21] M. M. Rahman, M. Islam, R. Sassi, and M. Aktaruzzaman, "Convolutional neural networks performance comparison for handwritten bengali numerals recognition," *SN Applied Sciences*, vol. 1, 12 2019.

[23] J. A. Rivera, D. Pardo, and E. Alberdi, "Design of loss functions for solving inverse problems using deep learning," in *Computational Science – ICCS 2020*, V. V. Krzhizhanovskaya, G. Závodszky, M. H. Lees, J. J. Dongarra, P. M. A. Sloot, S. Brissos, and J. Teixeira, Eds. Cham: Springer International Publishing, 2020, pp. 158–171.

[24] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, pp. 2149–2154.

[25] J. He, R. Harrison, and B. Li, "A novel 3d imaging system for strawberry phenotyping," *Plant Methods*, vol. 13, 12 2017.

[26] Blender Online Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[27] A. Durand-Petiteville, D. Sadowski, and S. Vougioukas, "A strawberry database: Geometric properties, images and 3d scans," Datset, ETUC, 2018. [Online]. Available: https://datadryad.org/stash/dataset/doi:10.25338/B8V308

[28] N. Wagner, "Melanoneiro/gazebo_segmentation: Preliminary release of Gazebo Segmentation Plugin," Mar. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4593865

[29] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context," *ArXiv*, vol. abs/1405.0312, 2014.

[30] NVIDIA Corporation. GeForce GTX 16 Series Graphics Cards. [Online]. Available: https://www.nvidia.com/en-gb/geforce/graphics-cards/gtx-1660-ti/