Using Neural Factorization of Shape and Reflectance for Ball Detection

Xavier Monté[0], Joey van der Kaaij[0], Rogier van der Weerd[0], and Arnoud Visser[0]

Intelligent Robotics Lab, Universiteit van Amsterdam, NL
www.intelligentroboticslab.nl

Abstract. Visually detecting a well-defined object like a soccer ball should be a simple problem. Under good lighting conditions this problem can be claimed to be solved, but unfortunately, the lighting conditions are not always optimal. In those circumstances, it is valuable to have a shape and reflectance model of the ball, to be able to predict how its appearance changes if the lighting changes. This is a prerequisite for playing soccer outside, with direct sunlight and clouds. The predicted appearance will be used to fine-tune an existing ball detection algorithm, based on the classic Yolo algorithm.

Keywords: reflectance estimation · view synthesis · object detection.

1 Introduction

In the RoboCup Standard Platform League (SPL) autonomous Nao robots are engaged in competitive soccer matches [3]. In this league no modifications are allowed on the hardware level, which forces the teams to concentrate on the robotic algorithms in the perceive-plan-act cycle [5]. To play soccer, the perception of the soccer ball should be done with high precision and recall [8]. Unfortunately, the computational resources on board the robot, even the Nao v6, limit the application of advanced neural network architectures to scaled-down versions optimized to achieve acceptable frame rates.

Yet, even scaled-down neural networks can perceive a soccer ball with high precision and recall [26] when properly trained. By extending the training set with challenging samples, rather than random ones, the algorithm's performance can be improved even further

To extend the training set for ball detection under difficult lighting conditions this research uses a modern breakthrough in the field of Neural Factorization of Shape and Reflectance or in short reverse image rendering. Until recently it was not possible to render realistic objects using a neural network [13]. The NeRF algorithm has made it possible to properly render any object using a neural network [13]. This means that the colour, shadow, reflection, and shape can be rendered realistic. A new feature on top of the NeRF algorithm is the NeRFactor algorithm [28]. This feature allows for rendering unknown light conditions [28].

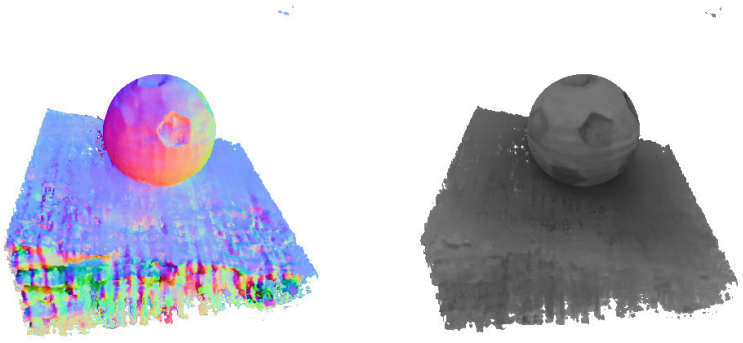
A constraint of NeRFactor is that there must be a rich data set of the object available, not a sparse data set. Two similar photos of the object must be exactly the same, the lighting, material, etc. NeRFactor has not been tested on round objects, such as the football used in this paper. In this paper, we will apply and generalize NeRFactor by training NeRFactor on a round object [12].

Once trained, such a model can come to aid when soccer-playing robots experience difficulties when they have to recognize the soccer ball under different light conditions. The angle of the light creates all kinds of shadows. Also, the colour changes under different lighting conditions. Moreover, a soccer ball is a perfectly round object; the symmetry makes the shape estimation both simpler and more sensitive to errors. Note that reliable shape estimation of round objects can also play a role in agriculture, for instance when apples, tomatoes, and melons have to be detected.

2 Theoretical background

2.1 NeRF

This study will generate training sets with the NeRFactor algorithm [28], which is an extension of NeRF algorithm [13]. NeRF describes an object as a continuous 5D function. This 5D function outputs at every 3D point (x, y, z) the radiance emitted at each direction (θ, φ) . The density at any point in space acts as a differential opacity on how much radiance is collected by a light beam through the point (x, y, z). In addition, a deep neural network is used to estimate the albedo, the RGB colour of the light reflected at any point, as a function of the viewpoint.



(a) Surface normals (b) Light visibility

Fig. 1: Two aspects of the shape and reflectance of a football

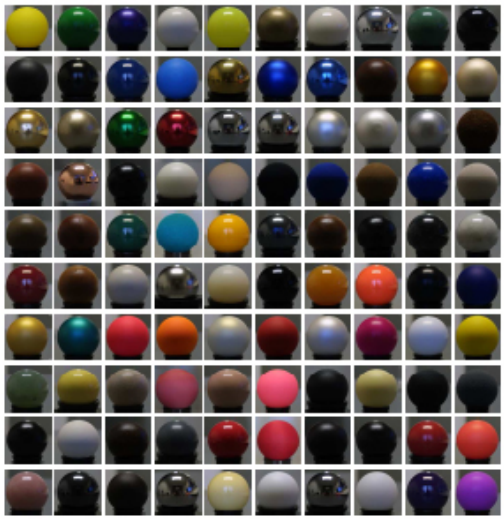


Fig. 2: The 100 BRDF balls used in the MERL dataset courtesy: MERL [15] [1]

The result is a model which estimates the shape and reflectance of the ball. The shape can be represented with the surface normals of the surface points (x, y, z). A surface normal is the vector that is perpendicular to the surface and is a unit vector. In image 1a NeRFactor's surface normals are shown on the football. The different colours are the directions of the surface normals. For rendering the images, the directions have been translated to RGB, if a surface normals go up, the colour is blue, orange is down, etc.

The reflectance of the ball can be represented with the light-visibility function. Light visibility is the visibility of any of the surface points (x, y, z) for any light source. In figure 1b the light visibility rendered by NeRFactor of the football is visible. The darker the colour, the less light is reflected from the object. The different black and white surfaces are visible on the rendering. Note that this light visibility indicates the amount of reflected light, but this value has still to be augmented with the albedo to estimate the colour of the reflected light.

The NeRF algorithm is trained with hundreds of 2D photos of the same object from different viewpoints. The Structure-from-Motion (SfM) from the COLMAP library is used to reconstruct from the 2D images a 3D object [20]. Once the reconstruction is finished, the object can be rendered from any viewpoint, generating synthetic scenes.

2.2 NeRFactor

The NeRF algorithm can generate a continuous high-resolution rendering of an object under good lighting conditions. Yet, for this study, we are interested in difficult lighting conditions. NeRFactor tried to improve NeRF by the introduc-

tion of multiple lighting conditions in contrast to the one lighting condition in NeRF [28].

NeRFactor accomplishes this by using a deep neural network that is pre-trained on the MERL dataset [15], which contains the Bidirectional Reflectance Distribution Function (BRDF) which measures the amount of electromagnetic radiation, reflected or scattered from objects measured for 100 different reflections (see Fig. 2).

This is not the only improvement of NeRFactor compared to NeRF. In addition, the algorithm has an improved geometry and reflection model by dividing the volume density estimated by NeRF into surface geometry.

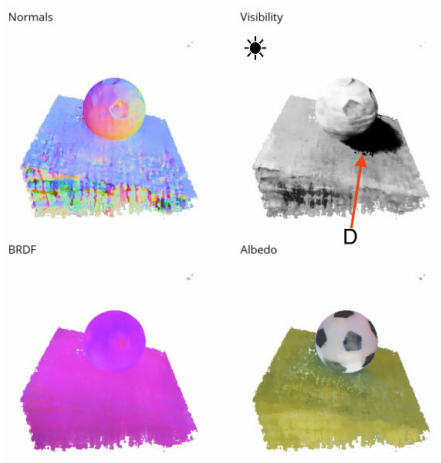


Fig. 3: NeRFactor factorization.

It also factored photos of an object into shape, reflectance, and illumination, which allows viewpoint-independent editing of lighting with shadows and material. Fig. 3 gives an example of factorization in the case of a point-based lighting condition called OLAD (One-Light-At-a-Time). In the visibility model, one can see a sharp shadow (marked with D) behind the ball opposite to the light source. With these factorized models (and several visibility models) any viewpoint and lighting condition can be rendered from the soccer ball.

3 Dataset

The correct working of both the NeRF and NeRFactor algorithms was first tested on the well-known datasets that accompany the algorithms: a hot dog, a pine cone, and a vase.

For this study, an equivalent dataset was recorded at the Intelligent Robotics Lab[2] of the RoboCup Standard Platform League[3] official soccer ball. The official ball is a soft foam ball with a black and white soccer ball print (see Fig 4a). They are 100 mm in diameter and weigh 44 gr. The recorded soccer ball was well used in several competitions [1], so some wear is visible on the ball.

(a) Nao robot with ball

(b) Actual dataset recording

To be able to reconstruct the viewpoints of the recordings, the Colmap algorithm requires that the recordings must be taken 'neatly'. This means that the recorded object should be in the centre of the frame (see Fig 4b), the light and background must be the same for recordings taken from the same angle of the object [20]. The algorithm also works better if you walk around the object in circles of different heights. The NeRF algorithm needs around 100 recordings per scene. Still, as we used a round object with a symmetrical pattern the reconstruction could not always correctly distinguish between the different angles of the object, so the scene had to be recorded several times until we had a good dataset.

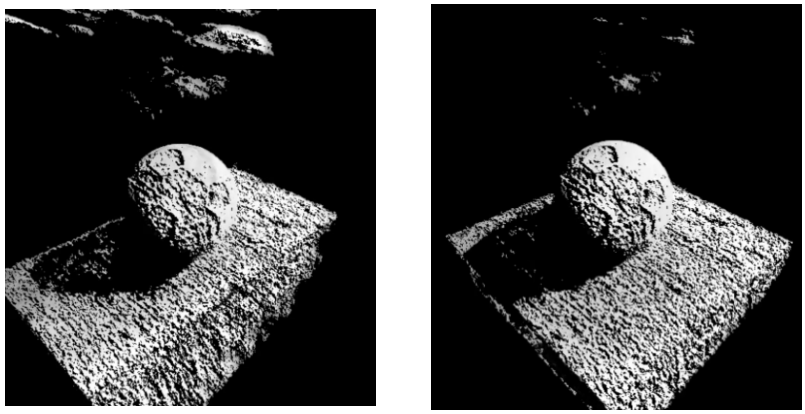
The next step is the calculation of geometry buffers [28]. Calculating geometry buffers is very resource intensive. Because the geometry buffers do not change, they can be calculated in advance. This makes it easier to execute the next steps of the algorithm. The algorithm calculates three different buffers, they are:

- The first step is the calculation of the surface points following any camera ray.
- The second step is the calculation of the surface normals.
- And the final step is the calculation of the light visibility for the whole object from each viewing angle.

The resulting light visibility map that has been made of the object shows how the light is scattered for each angle. The result can be seen in Fig. 5.

[2] https://www.intelligentroboticslab.nl/
[3] https://spl.robocup.org/

The rendering was performed with two different bounding boxes; one square and one rectangular. The render with the rectangular bounding box has more noise in the background as can be seen at the top. In addition, there should be no light visible behind the object. The geometry buffers calculated with the rectangular bounding box have more light scattering in the shadow of the ball. The calculation of the geometry buffers with the square bounding box gives a better result, so these buffers are used in the rest of the study.



(a) with a rectangular base (b) with a square base

Fig. 5: Examples of creating a geometry buffer of the light visibility for one viewing angle. A complete buffer is calculated for each viewing angle.

The last step is the factorization of the NeRFactor algorithm. The results of the NeRF algorithm and geometry are combined with the BRDF model and the different lighting conditions. For example, we can look at the results for the sunrise and night lighting condition. For these two conditions the ball is illuminated from two opposite angles and the surface normals should point exactly in the opposite direction (indicated by the rainbow color). That is indeed the case at arrow A (see Fig. 6 7), which is sky-blue for the sunrise condition and pink at night. The purple colors of the BRDF indicate the kind of reflection. Indeed, as can be seen at arrow B (see Fig. 7) the black areas are light purple (less reflection) and the white areas are dark purple (more reflection). The correctness of the albedo can be seen at arrow C (see Fig. 6), with the correct white and black pattern. In addition, one can see the warm colour of the sunset reflected on the white surface. The only downside is that the reflection model seems to have a metallic-like appearance, which is a known feature of the NeRFactor algorithm [28]. A more detailed analysis of the factorization results can be found in [14]. Yet, overall it is clear that the shape of the ball and the black hexagons are well preserved and the shadows & the colouring is realistic.

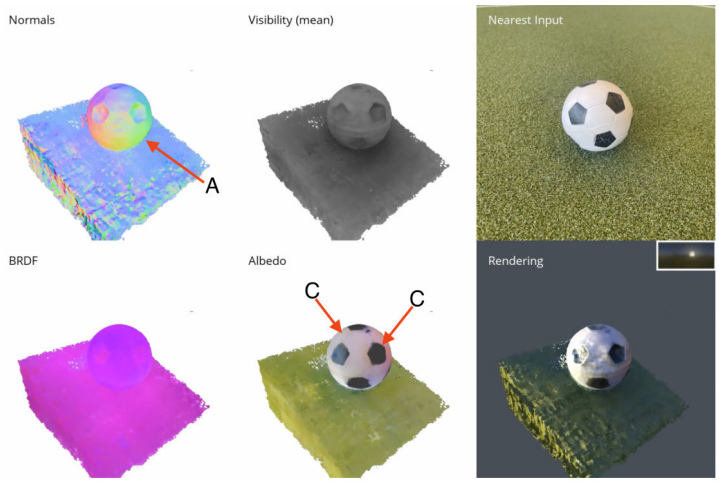


Fig. 6: Rendering of the ball at sunrise.

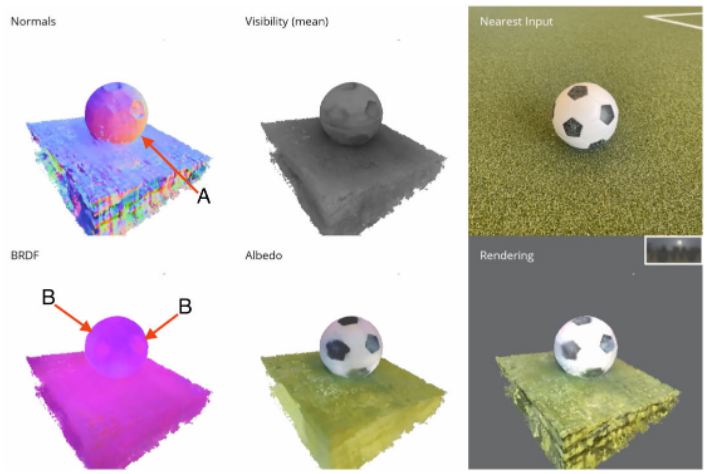


Fig. 7: Rendering of the ball at night.

The results seem to be good enough to generate a training set to fine-tune a ball detection algorithm, as the combination of YOLO and the HAAR algorithm used by the Dutch Nao Team [1].

4 Object Detection

The last 20 years have seen many breakthroughs in object detection techniques, well documented by several surveys [7,27,29]. Detection algorithms can be cate-

gorized into two-stage (R-CNNs, SPP, FPN, FCN and others) and single-stage (SSD, Yolo, Mask R-CNN, RetinaNet and others) models. Two-stage models split Region of Interest generation from Pooling operations that extract features and specify candidate bounding boxes. This achieves high localization and recognition accuracy but at the cost of inference speed. Single-stage models combine all operations in one convolutional forward pass which leads to faster inference. In recent years, much effort has been put into developing lightweight networks (examples include SqueezeNet, MobileNet) that can be deployed in resource constrained environments such as mobile and IoT devices.

For real-time detection tasks, an optimum is sought between accuracy and inference speed. The Yolo algorithm, since its first formulation in 2016 [17] has consistently pushed the boundary in this envelope. There are currently eight generations of the Yolo algorithm [17,18,19,24,9,25,23].

Yolo is a clear candidate of choice as a generic and high performing detection algorithm. It is preferred over highly optimized algorithms for specific classes given the ability to include additional classes in the future that can be relevant to the soccer framework such as goal posts, penalty markers, different robot stances, etc. As a basis for our application, variants of the Darknet based versions Yolo-v3 and Yolo-v4 are considered. The key reason for this is the maturity of this version, with well-established implementation frameworks and portability options. Modern versions based on PyTorch require their network weights to be transformed to TensorFlow Lite format before they can be used on the limited computational resources of the Nao v6 robot [16].

Ball detection

There exist pretrained models for Yolo object detection, typically trained on the COCO dataset (80 object classes) [11]. For the RoboCup one can concentrate on the relevant object classes; so we used a model which was pre-trained on only two object classes: Nao robots and soccer balls [26]. This pre-trained model was trained with datasets of RoboCup recordings labelled collaboratively [4]. The training set contained 7,500 images.

The performance of such double class detectors based on a pre-trained network can be further improved with additional data, even on small datasets [10]. The focus of this study is to improve the performance in different lighting conditions.

Synthetic dataset

The shape, reflectance, and illumination of the ball under different lighting conditions as described in Sec. 3 can be incorporated in a virtual reality environment of the Intelligent Robotics Lab created in Unity [2]. To do this the mesh has to be recovered [22]. Once the model of the ball is part of the virtual reality environment, one can view the ball from different viewpoints, including the 2D bounding box with the label. In addition, Unity Perception contains Randomizer tools that allow random object placement, position, rotation, texture, and

hue. So, not only the lighting conditions are changed, but also the background is changed (the view through the windows of the Intelligent Robotics Lab). This is comparable with the stochastic scene generation performed in the Unreal environment by Hess *et al.* [6]. An example of some of the generated images can be found in Fig. 8. For each Randomizer tool, the range can be specified. For instance, for the Intensity-range we used $[135.5, 378.5]$ Lumen and for the Temperature-range $[5250, 7750]$ Kelvin. For the Light range Ceiling we used the Intensity-range $[3050 - 7350]$ Lumen and the Temperature-range $[3600 - 6000]$ Kelvin.

Fig. 8: Synthetic images created in Unity with the mesh learned from NeRFactor

5 Training and testing Yolo-v4/3L

The Darknet[4] can be used to train Yolo models up to version 4. Based on the previous experience [26] we concentrated on the reduced model with three scales ('Tiny Yolo-v4/3L'). Key hyperparameters that were kept constant include the input size of the images (416×416), anchor box dimensions (3 per grid cell), batch size (64) and optimizer settings (SGD optimizer with momentum 0.9 and weight decay of $5e^{-4}$). A learning rate of 0.00261 was used, which was reduced by 10% after steps 10k and 15k.

The previous work reported a mean average precision (mAP@50) of 84.2% on the ball detection, based on the learning curve illustrated in Fig. 9a. This was the result of training on 7,500 images recorded during RoboCup soccer matches, enriched with live scenes recorded in the Intelligent Robotics lab. This trained network was in the next step further trained with synthetic images with

[4] https://github.com/AlexeyAB, maintained by Alexey Bochkovsky

the mesh of the ball generated based on the model generated NeRFactor (480 images). The pretrained networked started directly with a high mAP and a low loss, which was further improved in 20,000 iterations (see Fig. 9b). The resulting mean average precision improved to 92.3% on the real validation dataset.

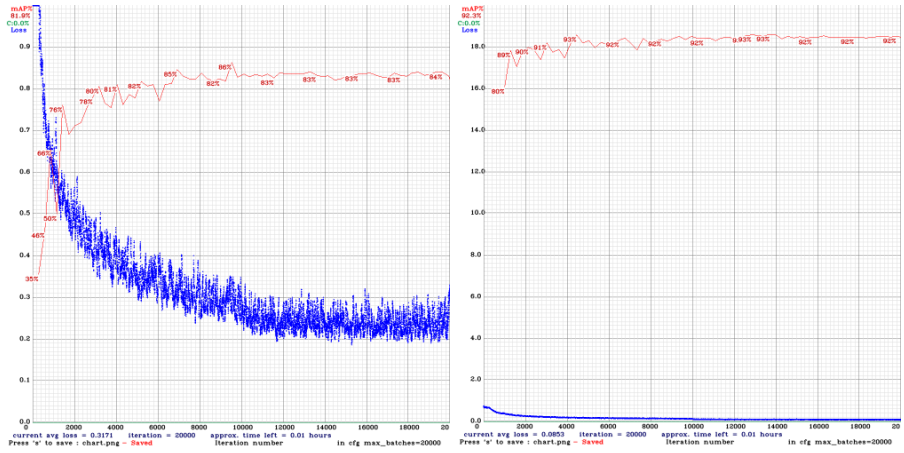


Fig. 9: Learning curves of 'Tiny Yolo-v4/3L'

Yet, the most important criterion is if the diversity of lighting conditions improved the robustness of the algorithm for real circumstances. When the model trained with the NeRF mesh is validated on the real validation set with recordings from live scenes (593 images) we get a mean average precision of 92.89%, outperforming the 84.2% earlier reported earlier (See Table 1). For convenience, also the results of Specchi *et al* [21] are given, although these results were acquired on slightly different dataset with an another resolution (320×320 instead of 416×416). The number of parameters have a strong influence on the performance, so this work with 23K parameters nicely falls between the 12K and 47K results reported by [21]. Note that the focus of Specchi *et al* [21] was to trade accuracy with speed to reach a high framerate on the Nao robot.

Model	#Parameters	mAP@50
vanderWeerd [26] - Tiny Yolo-v4/3L	23K	84.2%
Specchi *et al* [21] - Tiny Yolo-v7/YTv7_12K	12K	88.8%
our work - Tiny Yolo-v4/3L	23K	92.9%
Specchi *et al* [21] - Tiny Yolo-v7/YTv7_47K	47K	**94.0**%

Table 1: Comparison of ball-detection precision results

6 Conclusion

This experiment shows the value of being able to generate synthetic datasets which can boost the performance of object detectors such as Yolo. The focus of this research is the relative improvement; with more advanced algorithms (e.g., Yolo v7) or by generating much larger synthetic datasets we could have pushed the boundary even further. Yet, we could already see an impressive improvement on a small dataset, which seems to originate from the diversity which could be generated with the Randomizer tools. So, we trained with the data that challenged the object detector enough to further improve, instead of adding more of the same 'real' data. As long as the computational resources of the Nao robot do not facilitate more advanced algorithms the solution has to come from better training schemes.

References

1. Bolt, L., Klein Gunnewiek, F., Lekanne gezegd Deprez, H., van Iterson, L., Prinzhorn, D.: Dutch Nao Team - technical report (December 2022)
2. Borkman, S., Crespi, A., Dhakad, S., Ganguly, S., Hogins, J., Jhang, Y.C., Kamalzadeh, M., Li, B., Leal, S., Parisi, P., et al.: Unity perception: Generate synthetic data for computer vision. arXiv preprint 2107.04259 (July 2021)
3. Chown, E., Lagoudakis, M.G.: The standard platform league. In: RoboCup 2014: Robot World Cup XVIII. Springer Lecture Notes on Artificial Intelligence, vol. 8992, pp. 636–648 (May 2015)
4. Fiedler, N., Bestmann, M., Hendrich, N.: Imagetagger: An open source online platform for collaborative image labeling. In: RoboCup 2018: Robot World Cup XXII. pp. 162–169. Springer International Publishing, Cham (August 2019)
5. Hayes-Roth, B.: Architectural foundations for real-time performance in intelligent agents. Real-Time Systems **2**(1-2), 99–125 (May 1990)
6. Hess, T., Mundt, M., Weis, T., Ramesh, V.: Large-scale stochastic scene generation and semantic annotation for deep convolutional neural network training in the robocup spl. In: RoboCup 2017: Robot World Cup XXI. pp. 33–44. Springer International Publishing, Cham (2018)
7. Jiao, L., Zhang, F., Liu, F., Yang, S., Li, L., Feng, Z., Qu, R.: A survey of deep learning-based object detection. IEEE Access **7**, 128837–128868 (September 2019)
8. Kahlefendt, C.: A Comparison and Evaluation of Neural Network-based Classification Approaches for the Purpose of a Robot Detection on the Nao Robotic System. Master's thesis, Technische Universität Hamburg-Harburg (April 2017)
9. Li, C., Li, L., Geng, Y., Jiang, H., Cheng, M., Zhang, B., Ke, Z., Xu, X., Chu, X.: Yolov6 v3.0: A full-scale reloading. arXiv 2301.05586 (January 2023)
10. Li, G., Song, Z., Fu, Q.: A new method of image detection for small datasets under the framework of yolo network. In: 2018 IEEE 3rd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC). pp. 1031–1035 (October 2018)
11. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision – ECCV 2014. pp. 740–755. Springer International Publishing, Cham (September 2014)

12. Martin-Brualla, R., Radwan, N., Sajjadi, M.S.M., Barron, J.T., Dosovitskiy, A., Duckworth, D.: Nerf in the wild: Neural radiance fields for unconstrained photo collections. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7210–7219 (June 2021)
13. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Commun. ACM **65**(1), 99–106 (December 2021)
14. Monté, X.: Neural factorization of shape and reflectance of a football under an unknown illumination. Bachelor thesis, University of Amsterdam (February 2023)
15. Mungan, C.: Bidirectional reflectance distribution functions describing first-surface scattering. AFOSR Final Report for the Summer Faculty Research Program (Summer 1998)
16. Narayanaswami, S.K., Tec, M., Durugkar, I., Desai, S., Masetty, B., Narvekar, S., Stone, P.: Towards a real-time, low-resource, end-to-end object detection pipeline for robot soccer. In: RoboCup 2022: Robot World Cup XXV. Lecture Notes on Artificial Intelligence, vol. 13561, pp. 62–74 (March 2023)
17. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (January 2016)
18. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7263–7271 (July 2017)
19. Redmon, J., Farhadi, A.: Yolov3: An incremental improvement. arXiv 1804.02767 (April 2018)
20. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113 (June 2016)
21. Specchi, G., Suriani, V., Brienza, M., Laus, F., Maiorana, F., Pennisi, A., Nardi, D., Bloisi, D.D.: Structural pruning for real-time multi-object detection on nao robots. In: RoboCup 2023: Robot World Cup XXVI (July 2023)
22. Tang, J., Zhou, H., Chen, X., Hu, T., Ding, E., Wang, J., Zeng, G.: Delicate textured mesh recovery from nerf via adaptive surface refinement. arXiv preprint 2303.02091 (March 2023)
23. Terven, J., Cordova-Esparza, D.: A comprehensive review of yolo: From yolov1 to yolov8 and beyond. arXiv 2304.00501 (April 2023)
24. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Scaled-yolov4: Scaling cross stage partial network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 13029–13038 (June 2021)
25. Wang, C.Y., Bochkovskiy, A., Liao, H.Y.M.: Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv 2207.02696 (July 2022)
26. van der Weerd, R.: Real-time object detection and avoidance for autonomous nao robots performing in the standard platform league. Project report, University of Amsterdam (July 2021)
27. Zaidi, S.S.A., Ansari, M.S., Aslam, A., Kanwal, N., Asghar, M., Lee, B.: A survey of modern deep learning based object detection models. Digital Signal Processing **126**, 103514 (June 2022)
28. Zhang, X., Srinivasan, P.P., Deng, B., Debevec, P., Freeman, W.T., Barron, J.T.: Nerfactor: Neural factorization of shape and reflectance under an unknown illumination. ACM Trans. Graph. **40**(6) (December 2021)
29. Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. Proceedings of the IEEE **111**(3), 257–276 (March 2023)