Gaze-based Object Detection in the Wild

Daniel Weber*,†, Wolfgang Fuhl*,†, Andreas Zell†, and Enkelejda Kasneci‡
†University of Tübingen, Germany
‡Technical University of Munich, Germany

Abstract—In human-robot collaboration, one challenging task is to teach a robot new yet unknown objects enabling it to interact with them. Thereby, gaze can contain valuable information. We investigate if it is possible to detect objects (object or no object) merely from gaze data and determine their bounding box parameters. For this purpose, we explore different sizes of temporal windows, which serve as a basis for the computation of heatmaps, i.e., the spatial distribution of the gaze data. Additionally, we analyze different grid sizes of these heatmaps, and demonstrate the functionality in a proof of concept using different machine learning techniques. Our method is characterized by its speed and resource efficiency compared to conventional object detectors. In order to generate the required data, we conducted a study with five subjects who could move freely and thus, turn towards arbitrary objects. This way, we chose a scenario for our data collection that is as realistic as possible. Since the subjects move while facing objects, the heatmaps also contain gaze data trajectories, complicating the detection and parameter regression. We make our data set publicly available to the research community for download.

Index Terms—object detection; eye tracking; gaze; dataset; heatmap; machine learning; human-robot collaboration

I. INTRODUCTION

Recent research has shown that eye tracking has becoming increasingly relevant for a variety of applications. These include even dynamic real-world scenarios, such as driving [1], medicine [2], and sports [3]. Especially the combination with computer vision problems [4], has in turn great potential for the employment of eye tracking in other fields, such as robotics [5]. In the field of robotics, the focus is often on the interaction with the environment, for example, detecting and grasping objects [6]. In such settings, however, the interaction entities are often unknown due to the enormous amount of potentially existing objects. For this purpose, a semantic understanding of scenes must be present. In conveying this understanding, humans can play an important role and provide assistance to the robot. One modality that has proven to be particularly suitable and helpful for such human-robot collaboration (HRC) settings is the human gaze [7]. Gaze allows objects to be intuitively selected by the human and communicated (e.g., gaze pointing) to the interaction partner (e.g., robot). An additional advantage of the gaze modality is that it is far more unambiguous than gestures and, unlike speech, can also be used effortlessly in the case of unknown objects whose class name may not be known at all.

In this work, we address the problem of unknown object detection in real-world scenarios based on gaze. This is an essential challenge for HRC, as an example. After all, if the robot could detect an unknown object by the fact that the human is looking at it, this paves the way for further interaction possibilities. We refer to object detection in a similar manner to face detection. In face detection, the task is to estimate whether there is a face or not. In our task, the challenge is to find out whether the current gaze pattern belongs to a perceived object or not. While there is work investigating unknown object detection on static imagery, there is little research addressing unknown object detection on videos and settings in the wild. Along this line, [8] used fixations to infer the saliency of objects. A gaze map was used by [9], who combined it with candidate regions to segment objects. In the work by [10], gaze points were grouped into clusters to determine whether a cluster belonged to an object of interest and whether it was looked at intentionally or unintentionally. However, all these related works used multiple gaze points on one image, which is only possible if the stimulus (image of the observed scene) is static or if, for instance, eye tracking data from multiple people is used, as in [4]. Contrary to all aforementioned related works, we present a method capable of using gaze data from a single person in dynamic scenes, i.e., with non-static stimuli, to detect unknown objects.

Our way to meet this challenge is by considering and analyzing gaze data across multiple frames and constructing a heatmap from it. In contrast, [5] significantly reduced the amount of candidate bounding boxes of unknown objects on a static image using only one gaze point. In another recent work in a HRC scenario, [7] achieved segmentation of unknown objects and calculated corresponding bounding boxes in 3D space in real time. Although only one gaze point was required here, the scene image including depth information was needed. Some other approaches dispense with the gaze altogether, but focus rather on single-class images [11], or use additional information, e.g., from a depth sensor [12]. While robots typically have many sensors, they often have limited computing power. Additionally, there is often only one object of interest at a time, obviating the need to detect all objects at once. By completely omitting image data and employing gaze data instead, we can accomplish the task of detecting unknown objects of interest and still saving large amounts of required computer resources.

In this work, we build on existing work and pave the way for successful human-robot interaction through the following main contributions:

*Both authors contributed equally to this research.
• We present a method for detecting unknown objects in a scene without stimulus, based solely on gaze information.
• We only use heatmaps instead of scene images, enabling thus for a significantly faster approach than image-based object detection, while at the same time requiring considerably less computational resources.
• We make our unique data set, which contains both gaze data and bounding boxes of the observed objects, publicly available to the research community for download at https://cloud.cs.uni-tuebingen.de/index.php/s/QPzJC48xDGjNZK.

II. METHOD

In this work, we follow two goals. First, we classify which gaze points or ranges of gaze points belong to an object, and we assign temporal windows to the gaze points, which belong to an annotated bounding box. This creates a classification problem in which the gaze points windows with an associated bounding box are assigned to class one and gaze points windows without a bounding box are assigned to class zero.

The second goal is to regress the bounding box parameters on the gaze points. These parameters are the width and height, as well as the x and y position. For this task, we also assigned the gaze points to temporal windows. For the regression, we used only temporal windows with associated bounding box, since all others have no parameters for the regression.

We decided to use a spatial distribution as a feature since this worked best in our initial evaluations. This spatial distribution is a heatmap as previously proposed by [13] to classify gaze position data. To create such a heatmap, the gaze position data of a temporal window are used, and the individual gaze positions are assigned to cells in the heatmap (grid). Each time window results in one heatmap. After the assignment, the heatmap is divided by the sum over all values to obtain a distribution. As an extension to the approach in [13], we extended the 2D heatmap to 3D. This was possible because the software used for gaze determination generates 3D gaze points [14] based on a k-nearest neighbor regression.

In the case of the 3D heatmap, a cell is assigned to each gaze point based on its spatial position with the difference to the 2D heatmap that the depth or distance of the gaze points is additionally considered along the z-axis. The assignment procedure is illustrated in Fig. 1.

A formal description of the generation of the heatmap in 3D is given in Equation 1.

\[
\text{heat} \left( \frac{p_x}{R_x} \cdot G_x, \frac{p_y}{R_y} \cdot G_y, \frac{p_z}{R_z} \cdot G_z \right) \rightarrow 1. \quad (1)
\]

The gaze positions in x, y, and z coordinates in an Euclidean coordinate system are denoted by \(p_x\), \(p_y\), and \(p_z\), respectively. The constants \(R_x\), \(R_y\), and \(R_z\) represent the maximum resolution of the stimulus in x and y direction and the maximum depth supported by the software Pistol [14]. By dividing the gaze points by the maximum resolution, these ranges are normalized between 0 and 1. Subsequently, these values are multiplied by the number of grid cells (\(G_x\), \(G_y\), and \(G_z\)) and rounded to the nearest integers, denoted by “[...].” These new values correspond to the index in the heatmap and the selected cell is incremented by one, denoted by “+=”. In the case of a 2D heatmap, the cell for depth (z coordinate) is fixed at one.

Equation 2 describes the normalization of the heatmap in 3D and 2D since for the 2D case there would be only one depth.

\[
\text{heat}(x, y, z) = \frac{\text{heat}(x, y)}{\sum_{i=1}^{G_x} \sum_{j=1}^{G_y} \sum_{k=1}^{G_z} \text{heat}(i, j, k)}. \quad (2)
\]

The variables \(x\), \(y\), and \(z\) are the indexes to the heatmap corresponding to the x-axis, y-axis, and z-axis. Finally, the one-dimensional vector resulting from the flattening of the heatmap can be used as an input feature for various machine learning techniques.

III. STUDY DESIGN & DATA ACQUISITION

In this section, we describe the dataset we used. In order to evaluate our approach, a dataset was required which contains not only eye tracking information but also, in addition to the gaze points, the bounding boxes of the objects that the participants were looking at. Since, to the best of our knowledge, no such dataset exists or is publicly available, we collected a novel data set. At the beginning, a calibration was performed with each participant, following the procedure described in [14]. Subsequently, the subjects were allowed to move freely around the site. In this course, they should look at arbitrary objects they encountered. There was no specification as to how long they were supposed to look at the objects. To evaluate gaze accuracy, the participants were asked to look at the calibration marker again at the end of each recording. All recordings were conducted with the Pupil Invisible eye tracker, a head-mounted eye tracker developed by Pupil Labs, whose scene camera provides RGB images with a resolution of 1088 × 1080. Each participant captured three recordings (each recording was about five minutes long, including calibration and evaluation), resulting in 14 valid videos in total. This led
to a total length of about one hour of recording, consisting of 102,620 frames of which 27,946 contained objects.

Finally, we labeled the obtained data with DarkLabel [15].

Fig. 2 shows individual example moments from the recordings. Due to the errors related to the gaze estimation, the gaze points are, especially for small objects, not always on the labeled object, even though the participant was actually looking at it. In fact, even for a human, it is not always easy to determine the target object, and sometimes only possible considering the context and the observation of an image sequence. This demonstrates quite clearly the difficulties and challenges associated with this task. Our final, publicly available dataset only contains the gaze information and bounding boxes, yet no stimuli-related information.

IV. EVALUATION

In this section, we evaluate the classification of the gaze points with respect to the affiliation to an object, and we try to extract the position and the size of the object from those. To this end, we applied a variety of different, well-established machine learning methods and list here a selection comprising the best of them. In the classification experiments, we always specify the mean accuracy of a 5-fold cross validation. For the regression experiments, the mean error as a percentage of the image resolution from a 5-fold cross validation is given. We evaluated different heatmap grid sizes as well as different time window sizes. We conducted our evaluations on a computer system with Windows 10, an AMD Ryzen 9 3950X 16-core processor with 3.50 GHz, and 64 GB DDR4 Ram. All machine learning methods were implemented on the Matlab version 2021b and for reproducibility we restrict ourselves to Matlab’s default parameters.

The assignment of classes (object or no object) to time windows was done based on the presence of an annotated object in the time window. This means that if there was an annotated object in the time window, the class was set to one, and zero otherwise. In the regression, only time windows with an existing annotated object were used. Here, the parameters of the annotated object closest to the central timestamp of the time window were chosen. This was assigned because, in most cases, our subjects moved while looking at an object. Thus, there are usually different positions and sizes of bounding boxes in a time window.

Fig. 3 and Table I show a summary of the results of our classification experiment. Comparing the results of the three methods (KNN, bagged trees, and Gaussian SVM) for the 2D heatmap feature, the approach based on bagged trees achieves the best results. Looking at the progression over the grid and time window size, we can see that the KNN and the bagged trees perform best with a high number of grid cells and large time windows. In contrast, the Gaussian SVM performs best at a small number of grid cells but still large time windows. Moving on to the 3D heatmaps, the accuracy of the KNN method improves by 4 percent to 92 percent, which is also significantly better than the bagged trees.

The best results of our regression experiment are shown in Table II. Looking at the individual methods (Gaussian process regression, bagged trees, and Gaussian SVM), we see that all methods perform similarly well. As expected, based on the spatial heatmap feature, the position estimation is the most accurate. In contrast, the regression of the bounding box size, using only gaze data an no stimuli, is even more difficult than the position estimation and therefore less accurate. Comparing the results for the 2D and the 3D heatmap feature, the
TABLE II: Best regression error results as the average absolute error of a 5-fold cross validation in percentage. The columns X and Y denote the position of the bounding box, W is the width, and H is the height of the bounding box.

| Feature       | Method     | Error |
|---------------|------------|-------|
| 2D heatmap    | Gaussian Process | 6.1 6.8 12.2 15.1 |
|               | Bagged Trees | 6.4 6.9 12.0 14.3 |
|               | Gaussian SVM | 6.4 6.9 13.4 15.5 |
| 3D heatmap    | Gaussian Process | 5.8 6.0 9.9 11.6 |
|               | Bagged Trees | 6.4 6.7 10.5 12.3 |
|               | Gaussian SVM | 6.2 6.2 11.0 12.9 |

Fig. 4: Qualitative evaluation of the bounding box parameter regression. The results are from the Gaussian Process Regression with a time window size of 100, a grid cell number of 15 and the 3D heatmap feature.

Position results remain about the same, with some overall improvement. In terms of bounding box size, the best results improve significantly for all of the three methods. All in all, the Gaussian process method combined with the 3D heatmap feature performs best.

Fig. 4 shows a qualitative extract of the Gaussian process regression in comparison to the ground truth. Naturally, the position is more accurate than the bounding box size, since humans tend not to observe the entire object when looking at it. Overall, however, both can be determined quite well.

Hereafter, we will investigate the runtime and memory requirements. It should be borne in mind that classical object detectors pursue a slightly different goal than we do. Whereas in their case all objects are to be detected, we are primarily interested in the existence of an object of interest, that is, the one that the human is looking at. Since classical object detectors only use scene images and do not obtain information about human gaze behavior, they cannot know whether a human is looking at an object, nor which object. Thus, it would be a matter of chance whether the statement is correct.

With the regression task, the detection of all objects would be possible. Here, however, we encounter a different real-world problem, outside of laboratory conditions, which also makes our method so appealing. Since we are in a wild world, the objects of interest are extremely diverse and their number tremendous. The vast majority of objects in our dataset, such as doorknobs, light switches, and fire extinguishers, are simply not part of any publicly available data sets, such as Microsoft COCO [16] or ImageNet [17], that are typically used for training. Since the methods differ too much in this respect, we need a benchmark that covers more the commonalities. Therefore, in the remainder of this section, we will establish a baseline comparison in terms of speed and computing resources. As a baseline, we use state-of-the-art object detectors. These include Faster R-CNN [18], FCOS [19], and RetinaNet [20], each with a ResNet-50-FPN backbone [21], SSDlite320 [22] and Faster R-CNN both with a MobileNetV3 Large backbone [23], as well as SSD300 [22] with a VGG16 backbone [24]. These are supplemented by various YOLOv5 [25] variants. In order to test the speed, we measured the runtime of all methods on the CPU for 1000 individual predictions, i.e. 1000 different inputs with a batch size of one. The resource consumption was determined by measuring the amount of memory required for a single input. For our method with the heatmap input features, we used a time window size of 250 ms. For the classic object detectors, the 1088 × 1080 × 3 RGB images were used as input. The summary of the results are shown in Table III.

The fastest are the Gaussian SVM and the KNN with the 2D heatmap feature. The Bagged Trees are slower, but the runtime increases proportionally less as the number of grid cells increases. Consequently, the runtime for the 3D heatmap feature is in the range of one minute for the 1000 predictions.

TABLE III: Comparison of the required resources for the different input features. The time column indicates the execution time for 1000 different inputs at a batch size of one in seconds. The memory column specifies the required memory of a single input in kilobytes. For the 2D and 3D heatmap features, the results shown are from a time window size of 250 ms and a grid cell number of 30.

| Feature       | Method     | Time [s] | Memory [KB] |
|---------------|------------|----------|-------------|
| 2D heatmap    | KNN        | 10.8     | 424         |
|               | Bagged Trees | 57.8     | 1134        |
|               | Gaussian SVM | 8.6      | 406         |
| 3D heatmap    | KNN        | 276.9    | 3045        |
|               | Bagged Trees | 64.7     | 1467        |
|               | Gaussian SVM | 610.6    | 3650        |

RGB Image

| Method     | F. R-CNN [18] (RN50) | F. R-CNN [18] (MN) | FCOS [19] | RetinaNet [20] | SSD300 [22] | SSDlite320 [22] | YOLOv5s [25] | YOLOv5l [25] | YOLOv5x [25] | SSD300 [22] | VGG16 backbone [24] | SSDlite320 [22] | R-CNN [18] (MN) | SSDlite320 [22] | YOLOv5s [25] |
|------------|----------------------|-------------------|-----------|----------------|-------------|-----------------|-------------|-------------|-------------|-------------|------------------|----------------|-----------------|----------------|-------------|
| Time [s]   | 8 705.3              | 1 205.6           | 4723.2    | 5 164.5        | 900.8       | 163.7           | 200.6       | 1 127.6     | 2 174.5     | 3 677.9     | 7 052 3 743 456 | 276.9          | 545 390 580   | 529 3784       | 312 104      |
| Memory [KB]| 1 745 456            | 545 400           | 995 416   | 1 390 580      | 293 788     | 270 168         | 312 104     | 421 904     | 622 536     | 940 508     | 1 745 456 276.9 545 400 995 416 1 390 580 293 788 270 168 312 104 421 904 622 536 940 508 |
while the runtime for KNN and Gaussian SVM increases considerably from a few seconds to several minutes. Nonetheless, it is immediately apparent that the runtime is in general significantly lower compared to the object detectors using the RGB images as input features. While only the smaller models like YOLOv5n and SSDlite remain under three minutes, the other models are much slower. In particular, the computation time required by the popular Faster R-CNN (RN50) exceeds that of the Bagged Trees by a factor of over 100.

A similar picture emerges with respect to the RAM allocated for one single prediction. The memory requirements of the bagged trees are larger for small inputs, but do not increase as much in proportion to the number of grid cells as for the KNN and the Gaussian SVM. Overall, the heatmap features require only a few 100 KB to a few MB. This is substantially less than the most frugal neural network YOLOv5n, which needs around 270 MB. Faster R-CNN with the ResNet-50 backbone requires the most memory with over 1.7 GB. Again, the factor is more than 100 times larger than for the Gaussian SVM with the maximum number of 50 grid cells. Compared to the Bagged Trees, it even exceeds 860 times.

In summary, our method is several orders of magnitude faster than conventional object detectors while requiring only a fraction of their resources.

V. CONCLUSION

In this work, we addressed object detection in the wild by means of gaze data. Our results show that it is possible to detect objects and determine their bounding box based solely on gaze information. Additionally, we have used a variety of machine learning methods to show that they work for solving such challenges. Besides, the functionality of several machine learning methods proves that our heatmap feature, which we have extended to 3D, can be used efficiently for this problem. In comparison to classical object detectors that use image input features, we have shown that object detection by means of our heatmap features is significantly faster while only requiring a fraction of the computational resources. This is of major relevance due to the fact that robots usually have only limited computing capacity at their disposal and cannot be equipped with powerful graphics units as they consume a lot of power.

However, a significant amount of work remains for the future as we plan to extend our proof of concept to a real robot by making the gaze of the human collaborator accessible to it. Our approach can serve as a foundation for future applications in the field of human-machine interaction and HRC, where robots can learn new objects from humans through instant knowledge sharing. Hence, we hope our methods and dataset can help to advance researchers in this challenging context.

REFERENCES

[1] C. Braunagel, D. Geisler, W. Stolzmann, W. Rosenstiel, and E. Kasneci, “On the necessity of adaptive eye movement classification in conditionally automated driving scenarios,” in Proceedings of the Ninth Biennial ACM Symposium on Eye Tracking Research & Applications, 2016, pp. 19–26.

[2] K. Harezlak and P. Kasprzak, “Application of eye tracking in medicine: A survey, research issues and challenges,” Computerized Medical Imaging and Graphics, vol. 65, pp. 176–190, 2018.

[3] B. W. Hosp, F. Schulz, O. Hörer, and E. Kasneci, “Soccer goalkeeper expertise identification based on eye movements,” PloS one, vol. 16, no. 5, p. e0251070, 2021.

[4] K. Shanmuga Vadivel, T. Ngo, M. Eckstein, and B. Manjunath, “Eye tracking assisted extraction of attentionally important objects from videos,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3241–3250.

[5] D. Weber, T. Santini, A. Zell, and E. Kasneci, “Distilling location proposals of unknown objects through gaze information for human-robot interaction,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 11 086–11 093.

[6] J. Mohler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, “Learning ambidextrous robot grasping policies,” Science Robotics, vol. 4, no. 26, 2019.

[7] D. Weber, E. Kasneci, and A. Zell, “Exploiting augmented reality for extrinsic robot calibration and eye-based human-robot collaboration,” in Proceedings of the 2022 ACM/IEEE International Conference on Human-Robot Interaction, 2022.

[8] F. Xiao, L. Peng, L. Fu, and X. Gao, “Salient object detection based on eye tracking data,” Signal Processing, vol. 144, pp. 392–397, 2018.

[9] R. Shi, N. K. Ngan, and H. Li, “Gaze-based object segmentation,” IEEE Signal Processing Letters, vol. 24, no. 10, pp. 1493–1497, 2017.

[10] X. Luo, J. Shen, H. Zeng, A. Song, B. Xu, H. Li, P. Wen, and C. Hu, “Interested object detection based on gaze using low-cost remote eye tracker,” in 2019 9th International IEEE/EMRS Conference on Neural Engineering (NER). IEEE, 2019, pp. 1101–1104.

[11] Y. Pang, X. Zhao, L. Zhang, and H. Lu, “Multi-scale interactive network for salient object detection,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9413–9422.

[12] J. Bao, Y. Jia, Y. Cheng, and N. Xi, “Saliency-guided detection of unknown objects in rgb-d indoor scenes,” Sensors, vol. 15, no. 9, pp. 21 054–21 074, 2015.

[13] W. Fuhl, N. Sanamrud, and E. Kasneci, “The gaze and mouse signal as additional source for user fingerprints in browser applications,” arXiv preprint arXiv:2101.03793, 2021.

[14] W. Fuhl, D. Weber, and E. Kasneci, “Pistol: Pupil invisible supportive tool to extract pupil, iris, eye opening, eye movements, pupil and iris gaze vector, and 2d as well as 3d gaze.” arXiv preprint arXiv:2201.06799, 01 2022.

[15] DarkLabel, https://github.com/darkgmr/DarkLabel, 2021, accessed: 2021-12-07.

[16] 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 European conference on computer vision. Springer, 2014, pp. 740–755.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2009, pp. 248–255.

[18] S. Ren, K. He, R. Girshick, and J. Sun, “Faster R-CNN: Towards real-time object detection with region proposal networks,” Advances in Neural Information Processing Systems, vol. 28, pp. 91–99, 2015.

[19] Z. Tian, C. Shen, H. Chen, and T. He, “FCOS: Fully convolutional one-stage object detection,” in The IEEE International Conference on Computer Vision (ICCV), 2019.

[20] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in Proceedings of the IEEE international conference on computer vision, 2017, pp. 2980–2988.

[21] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[22] W. Liu, D. Anguelov, D. Erhan, Ç. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, “Ssd: Single shot multibox detector,” in European conference on computer vision. Springer, 2016, pp. 21–37.

[23] A. Howard, M. Sandler, G. Chu, L.-C. Chen, B. Chen, M. Tan, W. Wang, Y. Zhu, R. Pang, V. Vasudevan et al., “Searching for mobilenetv3,” in Proceedings of the IEEE/CVF international conference on computer vision, 2019, pp. 1314–1324.

[24] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv preprint arXiv:1409.1556, 2014.

[25] G. Jocher et al., “ultralight/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference,” Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6229336