Visual Detection of Vacant Seats Under Multi-View Camera System

Seunghan Jo Taeksoo Kim Seongmin Lee Jihoon Hong
Seoul National University
{pasa3232, taeksu98, s.lee.0713, athlon77}@snu.ac.kr
https://github.com/pasa3232/Vacant_seats_detector

1. Abstract

We propose Vacant Seat Detector, a project that finds empty seats at popular cafes. This project creates a layout of the room in bird's eye view and produces a real-time visual depiction of table capacity and table occupancy. Through a pipeline that consists of object detection via YOLOv7, conversion of image features to and from world coordinates, and precise feature detection with k-means clustering, we produce accurate graphical output of the café. By placing a video of this information at the entrance of a café or on an online platform, businesses can provide customers an efficient method of finding appropriate cafes for their circumstances. This project also generalizes beyond cafes to any in-demand space that uses ordinary, single-color tables and involves typical illumination conditions.

2. Introduction

Many of us have had experiences where we entered our favorite café just past noon and frantically searched for an empty seat, only to be disappointed. Then we probably decided to leave and walked around the block, only to repeat the process of entering another café, searching, and heading back to the streets. In areas with high demand for cafes, we often waste an annoying amount of time for unsatisfactory results, and we also face an inconvenience when we must coordinate a meeting with others. This demonstrates the need for a system that informs customers of empty seats ahead of time.

In this paper, we propose an approach for creating a visual representation of available seats for any given space. We created a pipeline for identifying the location, capacity, and occupancy of tables in real-time. Our system takes as input video files of multiple perspectives of the room. It then involves heavy use of object detection and conversion of image features to the world coordinate system. The system then outputs a visual depiction of the table and chair layout and dark colors indicating occupancy. This information can be posted outside cafes and on online platforms, thus allowing users to find an available café with almost no time cost.

Our novelty is in proposing a proof-of-concept that works for cafes adhering to typical illumination conditions and consisting of single-color tables. The project does not rely on many heuristics, and the parts that do only require trivial modifications. It is also robust to occlusion in images because it uses the world coordinates for only the target features, effectively isolating them from neighboring objects in the image. We also demonstrate that our proposed model generalizes beyond the setting of cafes to other spaces that adhere to our conditions.

3. Related Work

Scene Reconstruction. Reconstructing 3D scenes using images or videos is not an easy task which requires feature extraction, correspondence search, camera calibration estimation, reconstruction and non-linear optimization. One of the most promising method of 3D reconstruction is Structure-from-Motion (SfM) which reconstructs 3D scene from 2D images or videos while obtaining the the camera poses simultaneously. Starting from [1, 9], a wide variety of SfM strategies have been introduced, such as incremental SfM [14] or hierarchical SfM [2]. Recently, [13] managed to build a general-purpose pipeline of SfM which is robust, accurate, complete and scalable.

Object Detection. Human or object detection is a task of localizing all subjects that are human or objects for an input image or video sequence. Before the emergence of deep learning, many detection algorithms made effort to create the consistent descriptor of target's appearance and shape, despite the geometric and photometric transformations the target may face in different images. One of the most successful algorithms among them was the histogram of oriented gradients descriptor, widely known as the 'HOG' descriptor [16], which endeavors to describe the local object

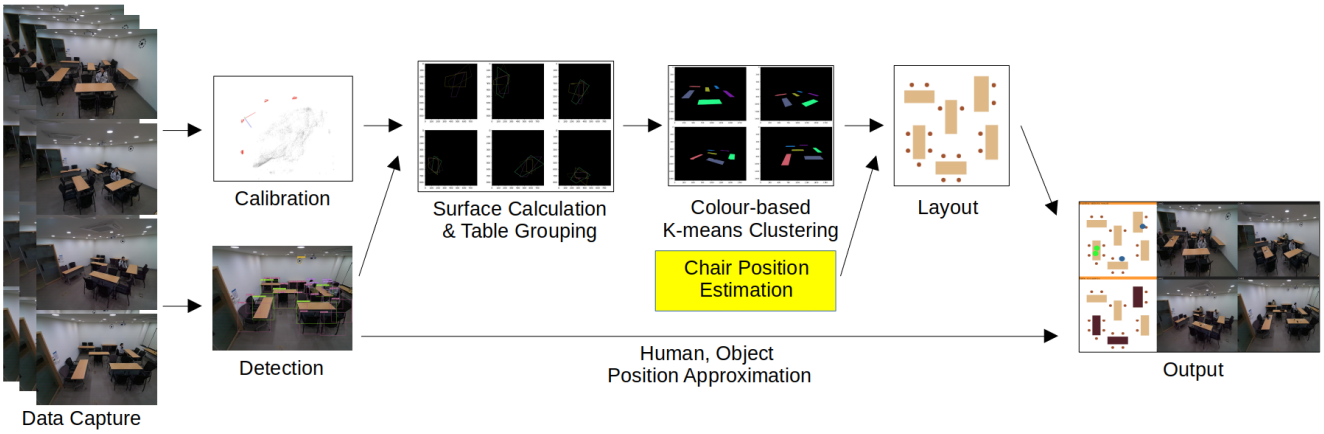

Figure 1. Method overview. From multi-view images, our pipeline generates the layout of tables and chairs and displays occupancy of each table in real time.

appearance and shape by the distribution of intensity gradients and edge directions. [5,8,10] further improve the original HOG detection algorithm in matter of time, efficiency or accuracy. As deep learning algorithms arose, the task of human or object detection was completely dominated by them. Detection with deep learning divides into two; two-stage detection and one-stage detection. Two-stage detectors [3,4,12] use two separate models to first determine the regions of interests, and then classify each region. On the other hand, one-stage detectors [7, 15] use a single convolutional neural network to predict the bounding boxes and their class probabilities. Among them, YOLO, [11], a one-stage detector trained with COCO dataset [6], is well known for its extremely fast inference and high accuracy.

Line Detection (Hough Line Transform). Objects that belong to a particular category can be found through existing detection methods such as YOLO or HOG. However, these methods display rectangular bounding boxes around objects and not precise contours, indicating the need for an algorithm that expands upon them and finds precise features. One contender is Hough line transform, a method that uses image gradients to find line segments in a grayscale image. When its parameters are tuned appropriately, the algorithm returns an image with straight lines at sharp edges. This is useful for images that involve many rectangular shapes, such as tables. However, the downside to this algorithm is that in many cases, it finds too many lines and warrants another algorithm that distinguishes between lines that are useful and lines that are not. As a result, Hough line transform can be a very complicated heuristic algorithm that underperforms for images with many features. A more effective alternative that is also generalizable to round shapes is color-based k-means clustering.

Image Segmentation via Clustering. Color-based K-means clustering is an algorithm that uses color to find semantically meaningful regions in an image. The idea is that neighboring pixels that are all a particular color belong to the same feature. The algorithm uses starting points as initial cluster centers and then maps every pixel to one of these starting points, using color and distance as metrics. It then updates the centers with the average of the clusters. The algorithm continues until it converges, and clusters no longer change. K-means clustering is efficient, but it depends heavily on the choice of initial points and number of clusters. It works best when a smart method of finding initial points exist for situations where the number of clusters is already known.

4. Method

We propose a generalized pipeline that produces a layout of the space with tables and chairs, monitors the occupancy of tables by detecting them in accordance with humans, and finally informs the arrivants the availabilty and choices they have.

4.1. Data Preparation

Since acquiring real CCTV footage from cafes was impossible, we prepared and used our own dataset. The idea was to simulate an actual café at SNU Institute of Computer Technology Room 302 with four Azure Kinect RGBD cameras. We placed these cameras on top of tripods and desks, to have them at the highest possible location. This was to ensure that the perspective of cameras is like that of CCTVs at cafes. The perspective of such cameras is closer to bird's eye view, which means that table surfaces are more likely to appear to be on the same plane. This property is helpful for finding correspondence points on table surfaces. The data collection environment is shown in Figure 2 along with the view from each camera.

We prepared three kinds of datasets for calibration, layout, and customers. The calibration dataset was used to

(a) (b)

Figure 2. Data collection in Building 139 Room 302. (a) Full view of Cam0 to Cam3 from left to right. (b) Collected data example from each camera view.

compute camera extrinsic parameters, and the layout dataset was used to identify the table and chair layout. The customer dataset was made from scenarios involving humans, to identify real-time table occupancy.

First, we took videos of the four of us moving in front of the cameras while holding calibration posters. This was to produce image frames for camera calibration. Then, we took another set of videos of the empty room, to capture the environment with just the tables. This was to ensure that all tables could be identified for layout generation. Finally, we captured various possible café scenarios involving humans and objects for the customer dataset. For instance, we simulated customers sitting at different tables, customers grouped together at one table, and customers who are not present at tables but have their backpacks placed on nearby chairs.

4.2. Layout generation

Calibration and Triangulation. To generate the layout of the cafe, it is necessary that we identified the unique tables and chairs in each image. We tackled this problem in two steps: determining the table configuration in world coordinates, and then the configuration of chairs around each table. However, one big difficulty was that a homography transformation produces reasonable results only on objects that belong to a single plane, or at least on planes with similar normal directions. Thus, a simple homography is not sufficient to represent the transformation between images taken from different cameras under our system, which further complicates the process of matching identical tables across different images.

We overcame this issue by taking advantage of the fact that the objects of our main interest belong to the same plane in world coordinates. To elaborate, we aimed to produce a bird's eye view layout of the cafe, so the most critical information contained in each image was the table surfaces. In other words, we only had to find correspondences between table surfaces across the images, and generate the table layout by projecting the plane onto a 2-dimensional space.

Therefore, we first computed the plane equation of table

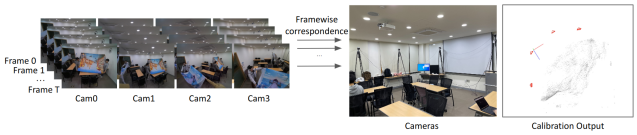

Figure 3. Camera calibration result using modified COLMAP.

surfaces from the prepared dataset. We began with the calibration of the cameras to acquire their external parameters using a structure-from-motion software called COLMAP. Videos of the four of us moving around the room holding posters were taken, and correspondence information between each set of four images taken at an identical point in time was added to the COLMAP database. Then, utilizing all the correspondence information of all frames, the software optimized the extrinsics of each camera, which we appropriately transformed to a unified world coordinate centered at camera 0. Figure 3 shows the process and the result of calibration. Afterwards, on a single set of 4 images taken at the same moment, we manually picked 6 correspondence points on the table surfaces. We performed triangulation on those points to acquire 6 3D coordinates, with which we computed the table surface plane equation.

Table Layout Generation. With the plane equation, we proceeded to the first step: determining the table layout. To identify where approximately each table is positioned in each image, we ran object detection with YOLO v7 model and acquired the bounding boxes of tables. Then, every table bounding box in the 4 images was projected onto the table surface plane. The intuition was that bounding boxes in different images corresponding to the same table, when projected, will overlap significantly where the surface of the table is positioned. Therefore, the projected bounding boxes were clustered into groups of 4 based on the area of intersection over union, where each cluster contained bounding boxes that correspond to the same, unique table. We shall denote the set of bounding boxes of table t_i as $S_i = \{B_{i,0}, B_{i,1}, B_{i,2}, B_{i,3}\}$ where $B_{i,j}$ is the bounding box of t_i in the image taken from camera j. The groups of table bounding boxes are shown in Figure 4.

The overlapping region, however, does not fully represent all table surface points in the plane. This is because the images were taken from different perspectives and some were occluded by chairs. We conducted one last additional process, colour-based K-means clustering on table bounding boxes, to identify all pixels in each image corresponding to the same table surface. One point was randomly selected from the region in the plane where the projected S_i overlapped, and it was used as one of the starting points in K-means clustering of the pixels in each $B_{i,j}$. The cluster $C_{i,j}$ that contained the starting point also contained all the pixels of $t_i's$ surface in the image taken from camera j, which were then projected onto the surface plane. The points on

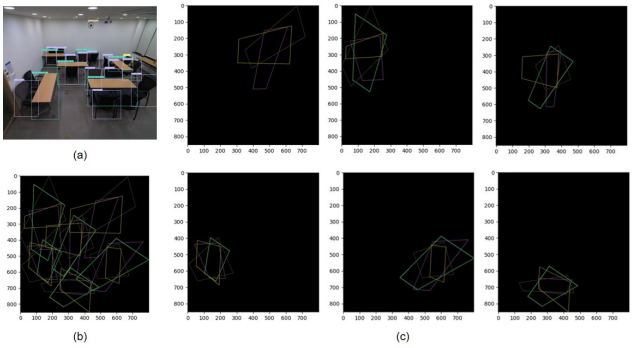

Figure 4. Table bounding boxes clustered. (a) Output of YOLO object detection on a frame from cam3. (b) Table bounding boxes in all 4 images projected onto the surface plane. (c) Grouped bounding boxes corresponding to a unique table.

the surface projected from $C_{i,j}$ $(j = 0, 1, 2, 3)$ were aggregated to identify the rectangular region of $t_i's$ surface. The result of colour-based K-means clustering is shown in Figure 5.

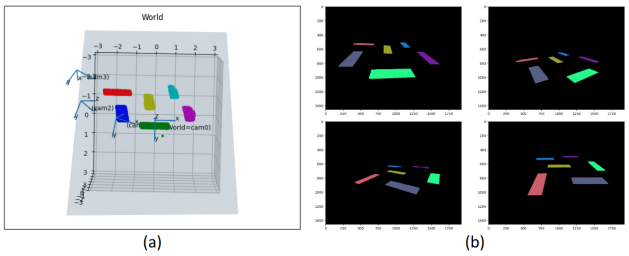

Figure 5. Aggregated points visualized. (a) On 3D table surface (b) Reprojected to each camera view. Reprojected points are well aligned with tables in each camera view shown in Figure 6

Chair Layout Generation. Having identified the table layout, we moved on to the next step of determining the chair configuration around each table. Here, we first selected 6 points around each table on the surface plane where a chair is likely to be positioned if there was a seat. The points are marked in green in Figure 6. Then, we adopted a heuristic algorithm to determine if there actually is a chair at each of those points. It proceeds by assigning each chair bounding box to one of the points, and identifies the points where a chair has been assigned in at least one of the 4 images as an available seat.

The result of this algorithm is directly used to add chair configurations in the table layout previously generated. Most chair positions were correctly identified, but a few of them were mislocated. This was due to the imperfect object detection by YOLO which failed to detect all chairs in each image, and the overly heuristic approach of assigning a chair to a fixed point. However, we could not come up with an algorithmic solution to this problem, so we fixed

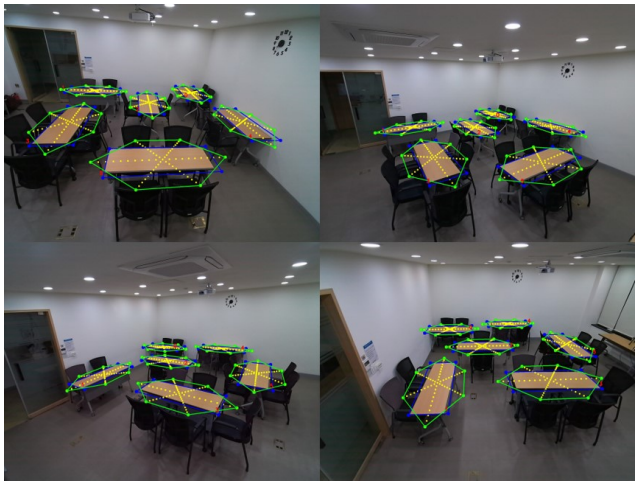

Figure 6. Potential chair locations around tables.

Algorithm 1 Chair Identification

$m \leftarrow$ *# cameras*
$n \leftarrow$ *# unique tables*
$l \leftarrow$ *# points around each table*

function ASSIGN(m, n, l)
  $assign[0..m-1][0..n-1][0..l-1] \leftarrow 0$
  for $i \leftarrow 0$ *to* $m-1$ **do**
    $S \leftarrow$ *set of chair bounding boxes in* img_i
    for $j \leftarrow 0$ to $|S|-1$ **do**
      $B \leftarrow j^{th}$ *chair bounding box in* S
      $t \leftarrow$ *closest table to* B *not fully assigned*
      $p \leftarrow$ *closest unassigned point around table* t *to* B
      $assign[i][t][p] \leftarrow 1$
  $avail[0..n-1][0..l-1] \leftarrow 0$
  for $t \leftarrow 0$ *to* $n-1$ **do**
    for $p \leftarrow 0$ to $l-1$ **do**
      if $\exists\ i$ *s.t.* $assign[i][t][p] == 1$ **then**
        $avail[t][p] \leftarrow 1$
  return $avail$

a few points by hand and used the corrected version. Both versions are illustrated in Figure 7.

4.3. Occupancy Determination

Object position approximation. To determine the occupancy of tables, we needed to figure out which table each object in the image belonged to. One naive approach would be to simply assign each object the table closest to its bounding box, but there were two problems. The objects were occluded by each other so some objects were not captured by YOLO in some cameras, and the widespread camera centers made it difficult to accurately identify the distance between an object and the table, further complicating the task. Thus, we took a two-step approach: first, we approximated the 3D position of each object onto the ta-

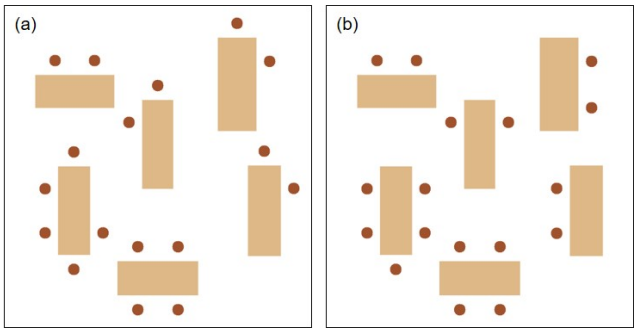

Figure 7. Table and chair layout of the room. (a) Generated version. (b) Manually corrected version.

ble surface plane, and then assigned the closest table on the plane.

The first step proceeded identically to the table layout generation process. Bounding boxes of the same object label were all projected to the table surface plane, and boxes corresponding to the same object were grouped using a quick greedy method that maximizes the overlap within each group. For example, bounding boxes labeled as 'person' in each of the 4 images taken at the same point of time were projected to the table surface plane. If there were 3 people at the moment, there would be 12 bounding boxes projected in total, which were then grouped to groups of 4. Each group of bounding boxes then represented a unique person. This step assumed that all objects relevant to the occupancy of tables had an intersection with the table surface plane. This is true for most cases, since common indicators of table occupancy are humans, objects on the tables, or clothes and bags placed on a chair.

Having found a group of bounding boxes for each object, the center of the region where all bounding boxes in the group overlap was interpreted as its 3D position approximated onto the table surface plane. The second step was then straightforward. The table whose center was the closest to the approximated position of the object was found, and the object was considered to 'belong' to whoever was using that table.

Table Occupancy After assigning each object to a unique table, the 'frame occupancy', i.e. the occupancy of each table at a given frame, was easily acquired. If a table had an object assigned, it was marked as occupied, while those without an object assigned were marked as vacant. Frame occupancy, however, is an incorrect measure of the actual occupancy of the tables. This is because people may leave their belongings and beverage on the table and move around to visit the counter or the restroom, in which case some tables would be erroneously marked as occupied. Therefore, we took into consideration a certain number(n) of consecutive previous frame occupancies to determine the actual occupancy of each table. Each table was considered as occupied if it was marked occupied for more than half of n previous frames, and vacant if otherwise.

Algorithm 2 Object Bounding Box Grouping Algorithm

$m \leftarrow$ *# cameras*
$cls \leftarrow$ *# object class*
$p \leftarrow$ *# table surface plane*

function GROUPS(m, cls, p)
  for $i \leftarrow 0$ *to* $m-1$ **do**
    $classes[i] \leftarrow img_i$ *yolo outputs with class* cls
    $bboxes[i] \leftarrow$ *polygon projected* $classes[i]$ *on* p
  $num\_groups \leftarrow$ *minimum value of* $len(bboxes[i])$
  for $i \leftarrow 0$ *to* num_groups **do**
    $group[i] \leftarrow$ *empty set*
  for $i \leftarrow 0$ *to* $m-1$ **do**
    for $p \in$ *all permutations of* $\{1..len(bboxes[i])\}$ **do**
      $S \leftarrow 0$
      for $j \leftarrow 0$ *to* $num\_groups - 1$ **do**
        $S \leftarrow S +$ *overlap*$(bboxes[i][p[j]], group[j])$
    $r \leftarrow$ *argmax of* p *that has maximum* S
    for $j \leftarrow 0$ *to* $num\_groups - 1$ **do**
      $group[j] \leftarrow group[j] \cup \{bboxes[i][r[j]]\}$
  return $group$

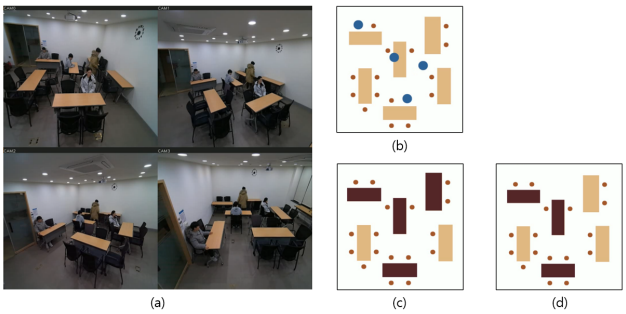

Figure 8. Table occupancy prediction. (a) Input frames. (b) Approximated person position. (c) Frame occupancy (d) Actual occupancy where n = 20.

5. Results

Figure 9 and 10 show some qualitative results of our algorithm applied to different sequences. Our algorithm covers not only the cases for human presence, but also the cases when objects exist in the captured data.

Our algorithm successfully produced the outputs in real-time when GPUs were used. Generating the layout took approximately 13 seconds, but this overhead could be perfectly hidden during actual service since it can be conducted offline before the café opens and starts serving customers. An important assumption behind this is that the layout remains unchanged throughout the day. Displaying information of table capacity and occupancy took roughly 200ms

Figure 9. Our algorithm applied on a sequence including humans. Humans are represented as blue circles. Occupied tables are colored in dark brown.

Figure 10. Our algorithm applied on a sequence including humans and objects. Humans are represented as blue circles and objects are represented as green circles. Occupied tables are colored in dark brown.

for each set of images at a given time with a Intel i9-12900K cpu machine with a single NVIDIA RTX3090.

6. Conclusion

In this project, we created a layout of the room in bird's eye view and produced a real-time visual representation of table occupancy. To do this, we first identified the three-dimensional location of tables, using a combination of triangulation, intersection-over-union (IOU) criterion, and color-based K-means clustering. Then we determined table capacity, by finding the number of chairs surrounding each table. We employed a heuristic algorithm based on six potential chair locations for each table. Finally, we determined table occupancy, by first determining each table's framewise occupancy, and then using this information over multiple consecutive frames to determine the table's actual occupancy. Trial-and-error was used to find the appropriate number of frames for determining the true table occupancy.

Our work can be used in a real setting with a few assumptions and modifications. First, we assume that all images in each video frame are taken by the same set of cameras. This means that they are all associated with the same intrinsic camera parameters. If this assumption were to not hold, then we would expect a small amount of error in the triangulation process. This is not due to a problem with the implementation, but rather due to the error in the camera parameters produced by COLMAP. We also assume that it is possible to distinguish objects in images based on their color. This is critical because without this assumption, YOLO may not find correct bounding boxes around objects, if at all. Issues presented by such challenging illumination conditions could partially be mitigated using a high-contrast filter, such as ImageEnhance in the PIL library. Our next assumption is that there are no color variations within the surface of a table, and that this property is reflected in the images. This prevents our work from being applicable to tables with prints. However, our assumption is reasonable because most cafes use tables with single, defining colors. Also, our project has notable strengths in that it does not require tables to be of a particular color, nor does it require them to have the same color in different images. This is because the k-means clustering algorithm for finding table edges requires color consistency within a single table, but not between tables or images.

In addition, our algorithm for determining table capacity works under the assumptions that no table supports more than six people, that the short side of the table has room for at most one chair, and that the long side of the table can support at most two chairs. These characteristics about chairs apply to most tables in a café, so using them as the basis for our heuristic algorithm is reasonable. Also, changing the specific parameters for chair arrangements would require minimal manual effort but still lead to good results. However, our algorithm would not work for outliers such as non-rectangular tables or long tables.

Finally, to determine table occupancy for a given frame, we used 20 previous frames. As our cameras took images at a rate of five frames per second, these frames spanned a total of 4 seconds. This parameter on frame number produced stellar results, implying that in most cases, just 4 seconds of information is enough to distinguish between people moving near tables and people occupying them. If our project were to involve cameras with different frames per second, then we could easily change our parameter so that they represent the same amount of information in time.

All in all, our project delivered a working real-time application that can help people identify the location and vacancy of tables at most restaurants and cafes. We hope that our project can encourage further research on improving the robustness for empty seat detection.

References

[1] Paul Beardsley, Phil Torr, and Andrew Zisserman. 3d model acquisition from extended image sequences. In *European conference on computer vision*, pages 683–695. Springer, 1996. 1

[2] Riccardo Gherardi, Michela Farenzena, and Andrea Fusiello. Improving the efficiency of hierarchical structure-and-motion. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 1594–1600. IEEE, 2010. 1

[3] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7036–7045, 2019. 2

[4] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015. 2

[5] M Kachouane, S Sahki, M Lakrouf, and N Ouadah. Hog based fast human detection. In *2012 24th International Conference on Microelectronics (ICM)*, number 3, pages 1–4. IEEE, 2012. 2

[6] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 2

[7] Youtian Lin, Pengming Feng, Jian Guan, Wenwu Wang, and Jonathon Chambers. Ienet: Interacting embranchment one stage anchor free detector for orientation aerial object detection. *arXiv preprint arXiv:1912.00969*, 2019. 2

[8] Kosuke Mizuno, Yosuke Terachi, Kenta Takagi, Shintaro Izumi, Hiroshi Kawaguchi, and Masahiko Yoshimoto. Architectural study of hog feature extraction processor for real-time object detection. In *2012 IEEE Workshop on Signal Processing Systems*, number 4, pages 197–202. IEEE, 2012. 2

[9] Roger Mohr, Long Quan, and Françoise Veillon. Relative 3d reconstruction using multiple uncalibrated images. *The International Journal of Robotics Research*, 14(6):619–632, 1995. 1

[10] Yanwei Pang, Yuan Yuan, Xuelong Li, and Jing Pan. Efficient hog human detection. *Signal processing*, 91(4):773–781, 2011. 2

[11] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016. 2

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2

[13] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4104–4113, 2016. 1

[14] Noah Snavely, Steven M Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3d. In *ACM siggraph 2006 papers*, pages 835–846. 2006. 1

[15] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019. 2

[16] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. An hog-lbp human detector with partial occlusion handling. In *2009 IEEE 12th international conference on computer vision*, number 1, pages 32–39. IEEE, 2009. 1