INDOOR 3D INTERACTIVE ASSET DETECTION USING A SMARTPHONE

Revekka Kostoeva[1] , Rishi Upadhyay[1], Yersultan Sapar[1], and Avideh Zakhor[1]

[1]Department of Electrical Engineering and Computer Sciences, University of California, Berkeley
{rkostoeva, rishi.upadhyay, yersultan, avz}@berkeley.edu

Keywords: Asset Management, Asset Detection, Asset Recognition, Asset Localization, Augmented Reality, Building Information Modelling, Mobile Mapping, Indoor Mapping

ABSTRACT

Building floor plans with locations of safety, security and energy assets such as IoT sensors, thermostats, fire sprinklers, EXIT signs, fire alarms, smoke detectors, routers etc. are vital for climate control, emergency security, safety, and maintenance of building infrastructure. Existing approaches to building survey are manual, and usually involve an operator with a clipboard and pen, or a tablet enumerating and localizing assets in each room. In this paper, we propose an interactive method for a human operator to use an app on a smart phone to (a) create the 2D layout of a room, (b) detect assets of interest, and (c) localize them within the layout. We use deep learning methods to train a neural network to recognize assets of interest, and use human in the loop interactive methods to correct erroneous recognitions by the networks. These corrections are then used to improve the accuracy of the system over time as the inspector moves from one room to another in a given building or from one building to the next; this progressive training and testing mechanism makes our system useful in building inspection scenarios where a given class of assets in a building are same instantiation of that object category, thus reducing the problem to instance, rather than category recognition. Experiments show our proposed method to achieve accuracy rate of 76% for testing 102 objects across 10 classes.

1. INTRODUCTION

Building floor plans with locations of safety, security and energy assets such as Internet of Things (IoT) sensors, thermostats, fire sprinklers, EXIT signs, fire alarms, smoke detectors, routers etc. are vital for asset management, climate control, emergency security, safety, and maintenance of building infrastructure (Minoli et al., 2017). Existing approaches to building survey are manual, and usually involve an operator with a clipboard and a pen, or a tablet, enumerating and localizing assets in each room. As such, the process is tedious, time consuming, and error prone. Also, it does not result in any contextual data i.e. the proximity and relationship between the sensors, and the proximity and relationship between the assets and the room (Asplund et al., 2018), (Wagner et al., 2003), (LaFlamme et al., 2006).

In this paper, we propose an interactive method for a human operator to use an app on a smartphone to (a) create the 2D layout of a room, (b) detect assets of interest, and (c) localize them within the layout. The output is the layout of each room in a building with the location of each asset marked in 2D and 3D in the layout, and an associated picture for each asset. This "as built" recovery of the assets in a building can be used in Building Information Modeling (BIM) of buildings by architects, owners, construction firms and facility managers (Boyes et al., 2017). Furthermore, it can be integrated with Facilities Management (FM) software tools such as TRIRIGA from IBM or Archibus.

The driving force behind our approach is the ubiquity of smartphones, and advances in machine learning and augmented reality (AR) on mobile platforms such as smartphones. Today's mobile devices are equipped with powerful processors and a myriad of sensors such as accelerometers, gyroscopes, and high-resolution cameras (Zhang et al., 2018). Thus, they are well suited for AR tracking systems and applications as well as running object detection models (Carmigniani, 2011), (Craig, 2013). Furthermore, tracking is now scalable to large environments (Zhang, 2001).

AR allows the placement of virtual objects in the real world (Ruan et al., 2012) by assigning anchors tied to a location in the real world (Azuma, 1997), (Azuma et al., 2001). By integrating object detection with AR, it is possible to develop a marker-less based system to free the user from the need to use QR codes or other types of markers to extract a 3D position of assets (Yan, 2014).

The outline of this paper is as follows. In Section 2, we provide an overview of our system. Section 3 describes the user interface and operation of the app, and Section 4 is on the deep learning methods used in our system. Section 5 includes experimental results, and Section 6 is conclusions and future work.

2. SYSTEM OVERVIEW

Our goal is to develop a smartphone based app which can be used for fast, and semi-automated asset detection and localization during building survey by inspectors and auditors. In this paper, we are concerned with 10 classes of assets related to safety, security and comfort of users, but our method can be extended to a much larger class of assets. In particular, our system in this paper is designed to recognize the following class objects: router, fire sprinkler, fire alarm, fire alarm handle, EXIT sign, cardkey reader, light switch, emergency lights, fire extinguisher, and outlet.

To automatically detect assets, we leverage existing advances in deep learning. Broadly speaking, there are two general methods for applying machine learning methods to object detection: instance recognition and category recognition. Instance recognition refers to situations where one is interested in finding all instances of a particular brand and model number of a given asset such as "Honeywell 1231304 programmable 7-day thermostat" (Zhang et al., 2006). This is in contrast with category recognition whereby one is interested in finding all objects in the same category regardless of brand or model number. An example of category recognition in the building context would be to find all thermostats inside a building regardless of their brand or

model number (Wang et al., 2006). Both problems require training neural networks with examples of the object to be detected. As expected, category recognition is more involved and requires a significantly larger number of heterogeneous training examples than instance recognition.

For our particular application of surveying a building and detecting and localizing assets of interests, a number of considerations need to be taken into account in choosing between instance and category recognition. To begin with, the number of publicly available training examples for our desired class of objects is small, making category recognition less attractive. For example, the picture of a fire extinguisher found in the public image databases Google shown in Figure 1b is quite different from a picture of an installed fire extinguisher in an actual building taken under realistic lighting conditions, as shown in Figure 1a. As such, it is not possible to rely solely on publicly available image databases for training the models for our application. Furthermore, the assets found inside a given building are usually limited to few brands/models and as such are quite amenable to instance recognition. Specifically, one can envision using instances of particular assets inside a building to construct an instance recognition engine, which is enhanced as the inspector progressively adds new examples of the same instance as he or she visits more rooms inside a given building or more buildings with similar devices in a campus. This bootstrapping strategy improves the recognition accuracy of the instance recognition model over time, and can also be used as a starting point of a category recognition model to be used in this or other applications.

(a) *(b)*

Figure 1. An example of fire extinguisher (a) captured by the operator in an actual room; (b) from Google Image Database.

3. USER EXPERIENCE

We start with the user experience in creating the layout for each room, and then describe the way assets in a given room are tagged, located and documented.

3.1 Layout Generation

Upon launching the app, the user first creates the layout of the room and then localizes the assets within the room. There are multiple approaches to creating the layout of a room. One way is to point the viewfinder on the smartphone to each vertical wall to detect it, compute the equation of the plane for that wall, find the intersection of those planes to find the layout, and then project them into the x-z horizontal plane to create a 2D layout. To extend the layout to 3D, we can detect the plane associated with the ceiling and find its intersection with the remaining vertical planes. We found this method to be error prone and inaccurate. Rather, we opt to use a simpler approach by instructing the user to click on the corners of the room, in a clockwise or counter-clockwise manner to generate the 2D layout. To extrude to 3D the user can also place an anchor on the ground and another one right above it on the ceiling to extract the approximate room height. In both approaches we take advantage of the positioning and tracking capabilities of modern smartphones via their AR capabilities.

3.2 Asset Documentation

The initial recognition model for a given building can either be generated from previously inspected buildings, or it can be made in situ from the assets of the building under consideration itself. In this paper, we take the latter approach in our experiments. As mentioned earlier, this initial recognition model is refined as the operator progresses from room to room inside a given building.

Once the layout for a given room is created, the work flow of our system for detecting and localizing assets in that room is as follows: the user finds the asset through the phone viewfinder and taps on the smartphone screen. The application takes a screenshot of the viewfinder, which is run through the neural network on the smartphone, detecting and classifying assets by assigning a probability to each of the N classes of objects it has been trained on. The most likely class, together with its associated probability or confidence is then displayed on the screen. The human operator will then either confirm or dis-validate the auto-recognition results. In the latter case, a dropdown menu with all categories of assets is displayed so that the user can choose the correct class, localize the asset, and draw its associated bounding box. The final output is a 2D layout of a room superimposed with the location of the detected assets.

Figure 2 shows an example of the above interactive process. The user points the viewfinder on the app to the object of interest and taps on the screen to capture a picture to be input to the recognition engine, as shown in Figures 2a and 2b for an outlet and a light switch respectively. The recognition engine then outputs the category with the highest confidence or probability and puts an AR anchor on that object based on the 3D position of that object in space. As seen, for the objects in Figures 2a and 2b the confidence is 0.92 and 0.87 respectively, and the detected category by the system is correct in both cases. Figure 2c shows an example of an outlet which has erroneously been detected by the recognition system to be a light switch. As seen, the confidence reported by the system is 9%. In this case, the user clicks on the "UNDO" button at the bottom of the screen, in which case a list of alternate objects in a dropdown menu shown in Figure 2d is presented to the user, giving a chance for the user to correct the error by choosing the correct category of light switch. Once the erroneous category is chosen, the user is given a chance to place the anchor in the right location on the image as shown in Figure 2e, and draw the corresponding bounding box for the object as shown in Figure 2f.

Note that the location of the anchor is not only registered in the 2D image as a feedback to the user, but also in a fixed 3D coordinate system for all objects, which is also registered with the floor plan the user generates for the room at the outset. The final output is a 2D or 3D layout of a room superimposed with the location of the detected assets, where the correctly detected and classified assets are displayed in green and misclassified assets are displayed in red, as seen in Figure 3.

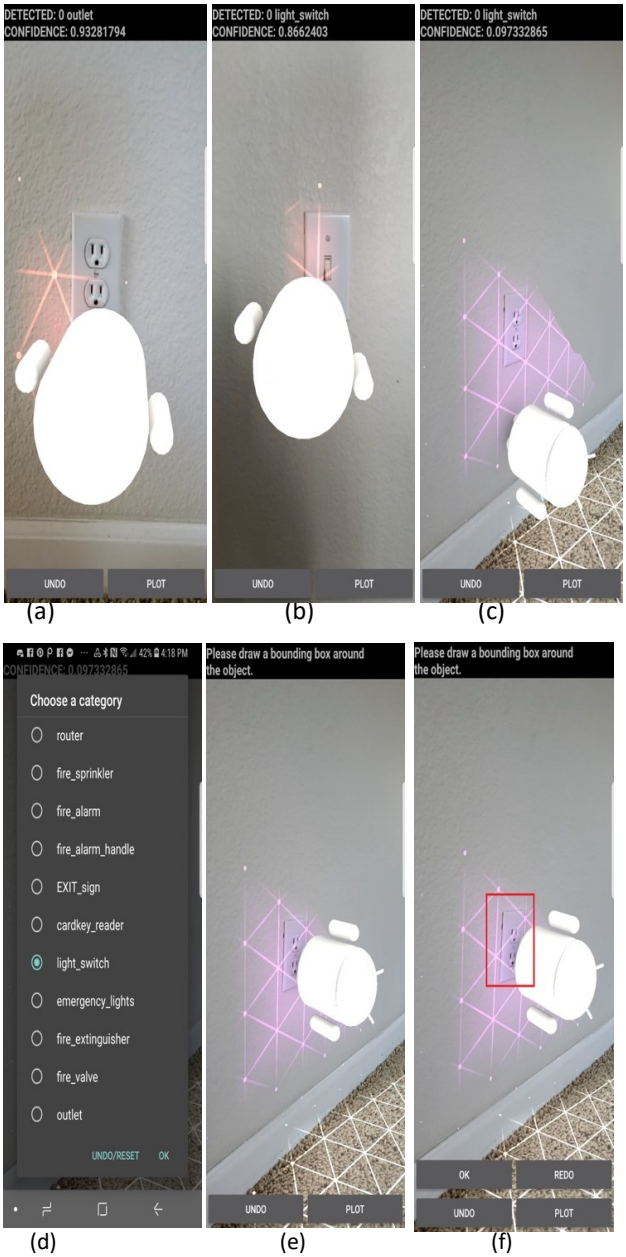

Figure 2: Screenshots of our system showing (a) correct detection of an outlet with confidence level of 93%; (b) correct detection of a light switch with confidence level of 87%; (c) incorrect detection of an outlet as lighting switch with confidence level of 9.7%; (d) dropdown menu showing the incorrect detection, which the user can correct; (e) user placing anchor in the correct place; (f) user drawing a bounding box around the object to enable the system to use this image for future training.

Regardless of whether the system detects objects correctly or erroneously, all positive and negative examples generated during the human interaction process are saved and used for future training of the learning algorithm. It is this bootstrapping activity that makes our system more valuable over time as more operators use it. In this way, it is somewhat similar to search engines whose performance improves over time as more users
use them. Our ultimate goal is to create an algorithmic pipeline that requires little new training, creates few false alarms, has low miss rate, and high precision and accuracy.

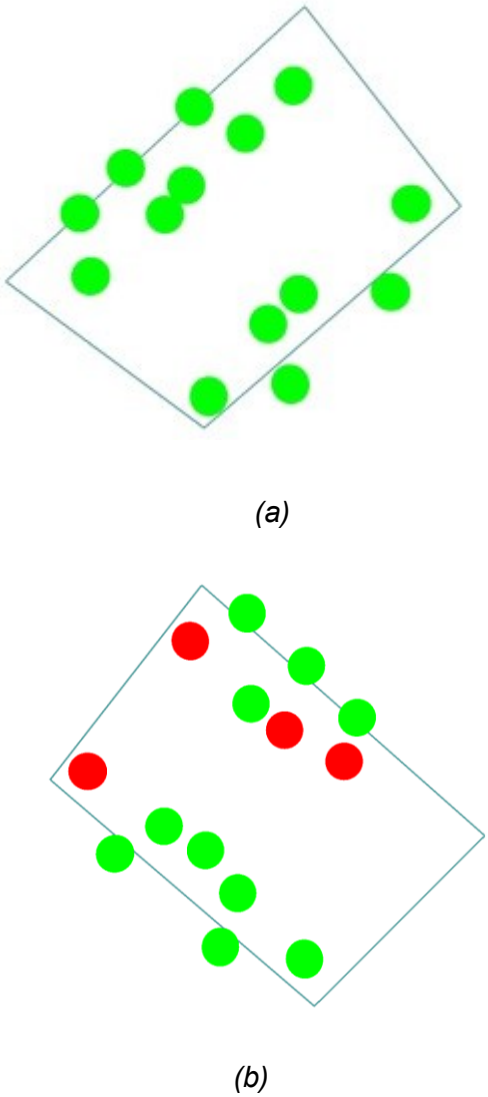

Figure 3: 2D floor plan of a room with (a) all correctly classified assets; (b) correctly detected assets (green) and misclassified/incorrectly detected (red) assets.

4. DEEP LEARNING PIPELINE

Our approach for detecting assets consists of a training and testing phase. We chose the Single Shot Detector (SSD) model which was pre-trained on the MSCOCO dataset (Lin et al., 2014) as a starting checkpoint for our model. SSD requires only a single pass through the neural network during inference, thus making it inherently faster and in turn more suitable for our use case of integrating object detection with anchor placement (Liu et al., 2016). In our system, we retrain the last layer of the neural network using the training examples obtained in situ. We use few-shot learning by training on multiple views of a few instances of an object. The input images are down sampled to 600 × 600 pixels. The model takes in images and ground truth bounding boxes for each object. Our SSD model has 6 neural network layers, the last layer of which is retrained with our training data for 30,000 steps. We fine-tune the model using RMSProp with an initial learning rate of 4, decay factor of 0.9, momentum of 0.9, batch size of 15 and 8000 decay steps. The

model returns the class label with the highest predicted score for the detected asset and the confidence level for that prediction.

4.1 Data Augmentation

To make the model more robust (Perez et al., 2017) to various input object sizes, orientations, view angles, and room lighting conditions (Taqi et al., 2018), each training image is randomly sampled by one of the data augmentation options native to the Tensorflow Object Detection API. These consist of: (a) use the original image; (b) flip the original image horizontally; (c) flip the original image vertically; (d) crop the original image such that at least 0.1, 0.3, 0.5, 0.7, 0.9, or 1.0 fraction of the input bounding box remains and that the minimum overlap with the new cropped image under which the bounding box is kept is 0.1, 0.3, 0.5, 0.7, 0.9, or 1.0; (e) scale the original image with a scale ratio ≥ 0.5 and ≤ 2.0; (f) adjust the brightness of the original image by a factor in the range [0, 0.2]; (g) adjust the contrast of the original image by a contrast factor equal to original contrast times a value in range [0.8, 1.25]; (h) rotate the original image by 90 degrees.

4.2 TensorFlowLite (TFLite)

To reduce the size and complexity of our model to operate on a smartphone, we froze our 86 MB 30,000-step TensorFlow model and converted our 23 MB frozen inference graph into a 22 MB TFLite model, which is an offline model optimized for smartphone devices requiring low latency and a small binary size (Ushakov et al., 2018).

5. RESULTS

Table 1 shows experimental results of our proposed system for ten object categories carried out in five rounds of experiments. The object classes currently in our system are router, fire sprinkler, fire alarm, fire alarm handle, EXIT sign, cardkey reader, light switch, emergency lights, fire extinguisher, and outlet. Table 2 shows the summary of various models used in the five rounds of our experiments to be described below.

In the first round of our experiments, we used 218 training examples over all ten classes collected from Cory Hall, with fire sprinkler having the most examples at 60, and fire extinguisher with fewest examples at 2. A total of 12 medium sized rooms with 106 objects were tested in this round, resulting in overall accuracy of 69%. As expected, classes with a larger number of training examples such as sprinklers and routers achieved higher accuracies of 86% and 100%, respectively, than classes with fewer examples such as fire extinguisher achieving 0% accuracy. The training examples and data augmentation do not account for all types of appearances of a fire extinguisher, e.g. whether it is hung by itself or placed behind a glass with writing on it. In addition, the test size for fire extinguisher class is quite small, making the zero percent accuracy not statistically significant.

For round two of our experiments, the training set consisted of the original training set in round 1, plus the incorrectly classified images in the first round which were interactively corrected by the user, resulting in a total of 252 training examples. In this round, we tested the system in 4 rooms visited in round 1 with 49 assets, and three new rooms with 24 assets, resulting in a total of 73 assets. As expected there is overall improvement in accuracy from 69% in round 1 to 85% in round 2.

In round 3 of our experiments, we used the training examples for round 2, in addition to the incorrectly classified images in round 2, which were interactively corrected by the operator, resulting in a total of 265 examples. In this round we visited 7 new rooms in a new building Soda Hall, not previously seen in either of the two previous rounds. Overall accuracy is at 76%, which is an improvement over round 1, but slight decrease compared to round 2. This can be explained considering that the training was carried out in Cory Hall, and testing in Soda Hall. Figure 4 shows an example of two routers in Cory and Soda Halls, indicating a significant difference in appearance. For classes with few examples for training, generally speaking the number of tests is also few since those assets are less common in buildings, resulting in statistically insignificant accuracy measurements. Nevertheless, as shown in Figure 5, there is a general correlation between the number of training examples and the accuracy. For example, the sprinkler class with 69 training examples resulted in 91% accuracy while the fire alarm handle with 8 examples resulted in 14% accuracy.

In round 4 we used the same model as round 3, and tested the model on another entirely new building Evans Hall. As evidenced in Table 1, the accuracy for Evans is 55%, which is lower than Soda Hall. This can be partially explained by the fact that Soda and Cory Halls are part of the electrical engineering and computer science department, and as such might have similar looking infrastructure devices, whereas Evans Hall houses many different academic departments. Figure 6 shows two examples of emergency lights in Cory and Evans Halls indicating a significant difference in appearance between them. This discrepancy resulted in the accuracy for the emergency lights in Evans Hall to be zero percent.

(a) *(b)*

Figure 4: Router from round 3: (a) Cory Hall; (b) Soda Hall.

(a) *(b)*

Figure 6: Emergency lights from Round 4: (a) Evans Hall, (b) Cory Hall.

In round 5, we returned to Cory Hall and tested the model on yet new rooms, using training examples from round 3, all from Cory

Hall, as well as 15 misclassified images from round 3, all collected from Soda Hall. The accuracy of the model dropped slightly from 83% to 76% from the last time the model was tested in Cory Hall. It can be argued that this model can generalize better between the assets in two buildings, at the cost of slightly lower performance on one building. An example of fire sprinklers for round 5 and round 2 are shown in Figure 7 indicating the more steep angles of data collection and augmentation in round 5 as compared to round 2; this partially explains the lower accuracy of sprinklers in round 5 as compared to round 2.

(a) *(b)*

Figure 7. Fire sprinklers from Cory Hall (a) from training set; (b) undetected and misclassified fire sprinkler from round 5.

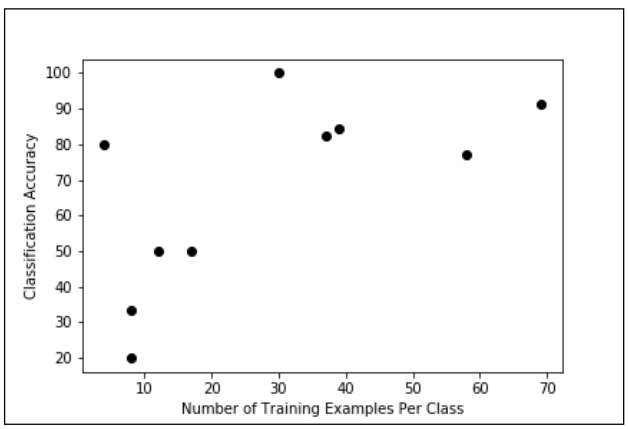

Figure 5. Round 3 experiment classification accuracy vs. number of training examples per class.

6. CONCLUSIONS AND FUTURE WORK

We have proposed an interactive way of documenting and localizing assets inside a room using a smartphone equipped with native AR capability, and capable of running machine learning models in real time. The AR capabilities of the smartphone remove the need for measuring and localizing assets in a room by hand by offering a more accurate and less involved way to extract an asset's position. Future work involves: (a) developing online incremental learning algorithms which learn from user feedback real time without having to resort to batch training; (b) extending the layout creation to more than one room, (c) improving the accuracy of the system via additional augmentation of training examples, and (d) adding support for more classes such as the fire valve.

Round 1 – All Rooms (Cory Hall), Model 0											
Name	A	B	C	D	E	F	G	H	I	J	Total
Correct	30	12	4	9	4	7	2	1	0	4	73
Wrong	5	1	16	6	0	0	2	1	2	0	33
Total	35	13	20	15	4	7	4	2	2	4	106
Accuracy %	86	92	20	60	100	100	50	50	0	100	69
# training ex.	60	35	8	7	57	26	8	7	2	8	218
Round 2 – All Rooms (Cory Hall), Model 1											
Correct	26	14	8	7	5	11	5	3	1	2	82
Wrong	4	0	5	1	3	0	2	0	0	0	15
Total	30	14	13	8	8	11	7	3	1	2	97
Accuracy %	86	100	62	88	63	100	71	100	100	100	85
# training ex.	64	36	24	14	57	26	11	8	4	8	252
Round 3 – New Building (Soda Hall), Model 2											
Correct	32	14	16	5	10	9	11	1	4	1	103
Wrong	3	3	3	5	3	0	10	2	0	3	32
Total	35	17	19	10	13	9	21	3	4	4	135
Accuracy %	91	82	84	50	77	100	52	33	100	25	76
# training ex.	69	36	28	17	57	27	11	8	4	8	265
Round 4 – New Building (Evans Hall), Model 2											
Correct	0	8	8	4	3	7	0	0	1	1	32
Wrong	0	0	2	2	0	2	0	14	0	6	26
Total	0	8	10	6	3	9	0	14	1	7	58
Accuracy %	N/A	100	80	67	100	78	N/A	0	100	14	55
# training ex.	69	36	28	17	57	27	11	8	4	8	265
Round 5 – Rooms (Cory Hall), Model 3											
Correct	21	10	13	5	5	5	9	3	4	3	78
Wrong	10	1	2	1	5	0	2	1	2	0	24
Total	31	11	15	6	10	5	11	4	6	3	102
Accuracy %	68	91	87	83	50	100	82	75	67	100	76
# training ex.	69	37	30	17	58	30	12	8	4	8	307

Table 1: A = Fire sprinkler, B = Fire alarm, C = outlet, D = light switch, E = router, F = EXIT sign, G = cardkey reader, H = emergency lights, I = fire extinguisher, J = fire alarm handle.

Model #	Origin of Training Examples	Total Number of Training Examples Across 10 Categories
0	Cory Hall	218
1	Model 0 training images + misclassified images from Round 1 (Cory Hall)	252
2	Model 1 training images + misclassified images from Round 2 (Cory Hall)	265
3	Model 2 training images + misclassified images from Round 3 (Soda Hall)	307

Table 2: Detailed description of the training set used for each model.

REFERENCES

Asplund, A. and Hanna, G., 2018. Using Mobile Augmented Reality and Reasoning Systems in Industrial Maintenance.

Azuma, R., Baillot, Y., Behringer, R., Feiner, S., Julier, S. and MacIntyre, B., 2001. *Recent advances in augmented reality*. NAVAL RESEARCH LAB WASHINGTON DC.

Azuma, R.T., 1997. A survey of augmented reality. *Presence: Teleoperators & Virtual Environments*, *6*(4), pp.355-385.

Boyes, G.A., Ellul, C. and Irwin, D., 2017, October. Exploring BIM for Operational Integrated Asset Management-A Preliminary Study Utilising Real-world Infrastructure Data. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* (Vol. 4, No. 4W5, pp. 49-56).

Carmigniani, J., Furht, B., Anisetti, M., Ceravolo, P., Damiani, E. and Ivkovic, M., 2011. Augmented reality technologies, systems and applications. *Multimedia tools and applications*, *51*(1), pp.341-377.

Craig, A.B., 2013. *Understanding augmented reality: Concepts and applications*. Newnes.

LaFlamme, C., Kingston, T. and McCuaig, R., 2006. Automated mobile mapping for asset managers.

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. Microsoft coco: Common objects in context. In *European conference on computer vision* (pp. 740-755). Springer, Cham.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y. and Berg, A.C., 2016, October. Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

Minoli, D., Sohraby, K. and Occhiogrosso, B., 2017. IoT considerations, requirements, and architectures for smart buildings—Energy optimization and next-generation building management systems. *IEEE Internet of Things Journal*, *4*(1), pp.269-283.

Perez, L. and Wang, J., 2017. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*.

Ruan, K. and Jeong, H., 2012, May. An augmented reality system using Qr code as marker in android smartphone. In *Engineering and Technology (S-CET), 2012 Spring Congress on* (pp. 1-3). IEEE.

Taqi, A.M., Awad, A., Al-Azzo, F. and Milanova, M., 2018, April. The impact of multi-optimizers and data augmentation on TensorFlow convolutional neural network performance. In *2018 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)* (pp. 140-145). IEEE.

Ushakov, Y.A., Polezhaev, P.N., Shukhman, A.E., Ushakova, M.V. and Nadezhda, M.V., 2018, November. Split Neural Networks for Mobile Devices. In *2018 26th Telecommunications Forum (TELFOR)* (pp. 420-425). IEEE.

Wagner, D. and Schmalstieg, D., 2003. *First steps towards handheld augmented reality* (p. 127). IEEE.

Wang, C., Ding, C., Meraz, R.F. and Holbrook, S.R., 2006. PSoL: a positive sample only learning algorithm for finding non-coding RNA genes. *Bioinformatics*, *22*(21), pp.2590-2596.

Yan, Y., 2014. Registration issues in augmented reality. *University of Birmingham*, *2015*.

Zhang, H., Berg, A.C., Maire, M. and Malik, J., 2006. SVM-KNN: Discriminative nearest neighbor classification for visual category recognition. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on* (Vol. 2, pp. 2126-2136). IEEE.

Zhang, X., Genc, Y. and Navab, N., 2001. Taking AR into large scale industrial environments: Navigation and information access with mobile computers. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on* (pp. 179-180). IEEE.

Zhang, W., Han, B. and Hui, P., 2018, October. Jaguar: Low Latency Mobile Augmented Reality with Flexible Tracking. In *2018 ACM Multimedia Conference on Multimedia Conference* (pp. 355-363). ACM.