A Virtual Reality-Based Digital Twin of workspaces with Social Distance Measurement Feature

Abhishek Mukhopadhyay[*], G S Rajshekar Reddy[*], KamalPreet Singh Saluja[*], Subhankar Ghosh[*], Naveen Talwar[*], Anasol Peña-Rios[+], Gokul Gopal[+] and Pradipta Biswas[*]

[*]BTIRC /Indian Institute of Science, India
[+]BT

17th December 2021

A Virtual Reality-Based Digital Twin of workspaces with Social Distance Measurement Feature

The Covid-19 pandemic resulted in a catastrophic loss to global economies, and social distancing was consistently found to be an effective means to curb the virus's spread. However, it is only as effective when every individual partakes in it with equal alacrity. Past literature outlined scenarios where computer vision was used to detect people and to enforce social distancing automatically.

We have created a Digital Twin (DT) of an existing office and laboratory space for remote monitoring of room occupancy and automatically detecting violation of social distancing. To evaluate the proposed solution, we have implemented a Convolutional Neural Network (CNN) model for detecting people, both in a limited-sized dataset of real humans, and a synthetic dataset of humanoid figures.

Our proposed computer vision models are validated for both real and synthetic data in terms of accurately detecting persons, posture and intermediate distances among people. We presented two applications of the work in estimating maximum occupancy of an arbitrary shaped room maintaining social distancing and developing synthetic dataset for non-conventional traffic participants in the context of autonomous vehicles. Finally, we used intermittent layer and heatmap based data visualization techniques to explain the failure modes of a CNN.

1. Introduction

The CoVID-19 pandemic is considered to be one of the worst disasters for humanity after the Second World War. Social distancing, wearing masks and frequent sanitization of ones' hands in common spaces became the norm. Social distancing has been proved to be effective in hindering virus transmission by increasing the physical distance between people or reducing the congestion in socially dense community atmospheres such as schools, colleges, and workplaces [15, 3, 19, 16]. Existing approaches of enforcement of social distancing involve manual inspection of common spaces using personnel. Most of such methods require several personnel on the ground doing laborious job of monitoring people around them. Automating the monitoring of human beings and enforcing social distancing saves humongous effort from such personnel. Efforts are already underway to use computer vision techniques to perform tasks like using traditional and thermal cameras to measure body temperature and to identify excessive pedestrian flow at public places. Enforcing social distancing involves detecting the humans in a space and measuring the distances between them. Existing approaches using object detection datasets [41, 11] and models [58, 85] enable us to detect humans but monitoring the distance between two individuals is a challenging problem. Existing datasets in object detection and allied fields focus on the entity identification or relation between two entities like in the case visual captioning [60] but relating two entities using a physical metric is a novel challenge.

In this context, we propose a Digital Twin of a workspace through an interactive and immersive Virtual Reality Experience. Users can move around the space virtually and remotely, as they would in the real-world. The benefits of using a DT as a visualization medium are multifold. Firstly, it provides an interactive and intuitive virtual experience that can also be used in VR. Users can navigate around the virtual environment as they would in the real world. Secondly, a virtual environment protects the privacy of the occupants through abstract humanoid figures than a direct video feed. In the virtual world, a virtual camera is simulated at the same position from where the real-world feed was recorded. We then map the two-dimensional centroid coordinates onto the virtual camera's feed. Moreover, through a Ray cast operation, the two-dimensional coordinates are mapped to the three-dimensional coordinates of the virtual world, and hence people's movements are simulated in real time.

The digital twin is equipped with weather monitoring and room occupancy measurement feature. We utilized existing object detection model and used transfer learning technique to detect persons in workspaces. We recorded images under 32 combinations including different positions and postures. We also undertook posture estimation to improve accuracy of the digital twin and help in explaining violation of social distancing. For example, whether more than one person is sitting together or sitting and standing or only standing- all indicate different activity trend for violating social distance. Policies can be made accordingly like prohibiting people to eat together in office spaces. We determined the postures of humans using the recorded dataset using interior-hip angle obtained from existing state-of-the-art pose estimation model and this method classifies whether a person is sitting or standing with 84% accuracy. After mapping real world position and posture to Digital Twin, we converted the bounding box predictions from person detection module into real world dimensions, followed by an investigation into the effectiveness of our proposed approach using both virtual and real-world data. We found a correlation of 0.82 between actual and measured distance with $R^2 = 0.67$.

Further, we investigated finding the optimal number of people that can be accommodated in office space based on the standard social distancing guidelines. Finding the maximum number of people that can fit in a defined space while maintaining social distancing is challenging for numerous reasons. This problem is modelled as a classical circle packing problem [77], when each individual is depicted as the center of a circle, where the circles are to be arranged in a given space such that their centers stay within the bounds, and simultaneously avoid other objects (such as furniture) intrinsic to that space. These categories of problems cannot be solved in polynomial time, thereby falling under the NP-Hard category. Our approach is based on a heuristic algorithm which aims to provide close approximate solutions. We also accounted for the fact that with the obstacles in a workplace such as pieces of furniture and building structures, the polygon to pack may resemble a convex highly irregular structure. We adopted basic concepts of circle packing algorithm [83] and modified it for irregular shapes as discussed in further detail in Section 6.1. Additionally, we used data visualization techniques to explain the working of a complex machine learning system like a Convolutional Neural Network (CNN) to help us to debugging the performance of the system.

The key contributions of this work are as follows:

- We proposed a validated technique for synthetic data generation using photo realistic Digital Twin.
- We validated accuracy of person detections between real and synthetic image and in both cases, it was found to be more than 91%.
- We designed an algorithm for estimating the maximum room occupancy under social distancing norms via circle packing techniques and compared the results to that of proved optimal packing structures.

We briefly discuss about Related Work in section 2, followed by the methodologies of the proposed approach in section 3. We discuss about developing a VR based simulator in section 4. We discuss in detail about validating the proposed models of person detection, pose estimation and social distance measurements in section 5 followed by applications and visualization techniques in sections 6 and 7, respectively. The penultimate section highlights the utility and value addition of the system, followed by concluding remarks.

2. Related work

2.1 Digital Twins

The first digital twin (DT) implementation dates back to NASA's Apollo program [17], where they were used in live missions to replicate the problem scenarios faced by the crew in space. NASA [21] formalized the DT definition in 2012 as an integrated Multiphysics, multiscale, probabilistic simulation of an as-built vehicle or system that uses the best available physical models, sensor updates, fleet history, among other available data to mirror the life of its corresponding flying twin. Tao et al. [63] highlighted the state-of-the-art in industrial DTs, according to the which, DTs have been implemented in three key application areas (i) product design, (ii) production, and (iii) prognostics and health management (PHM), the majority of which was focused on PHM. Khajavi et al. [35] explored a DTs' use in a smart building scenario by replicating a part of its front facade. The facade was visualized by assigning different yellow shades to the respective lux values received from the sensor.

Several commercial solutions have also emerged due to the diverse possibilities and benefits. One example is the Azure Digital Twins (ADT) [36], a cloud-based service that aimed to democratize DT deployment by providing a software as a service solution. Steelcase, a company known for designing workspaces, developed a space-sensing sensor network using ADT [45]. By implementing a suite of wireless infrared sensors, they generated analytics on how their spaces were being utilized, which in turn was used to enhance reliability and efficiency. ICONICS [44] also utilized ADT to create a virtual representation of a physical space to improve energy efficiency, optimize space usage, and lower costs. Nikolakis et al. [51] utilized digital manikins for simulating human activities in a DT of a factory floor. It was concluded that the twin could help maintain better ergonomics among the shop floor workers.

2.2 Digital Twins in Covid-19

Through real-time sensor data and accurate simulations, digital twins could play a vital role in containing the coronavirus' spread of the coronavirus. For example, Milne et al. [96] modeled a city in Australia to understand the effectiveness of social distancing, and they reported that it contributed a substantial factor in flattening the epidemic curve. A consortium among Aalto University, Finnish Meteorological Institute, VTT Technical Research Centre of Finland, and the University of Helsinki [97] studied the transmission of the virus by modeling possible scenarios in indoor spaces. They examined various situations like when a person coughs in aisles in grocery stores. A blog post from Buro Happold [98] concluded that the traditional workplace model is not effective in managing social distancing. Unity Technologies [99] built an open-source simulator concept for visualizing the spread of Covid-19 in a fictitious three-dimensional grocery store environment. Industry big players, Google and Amazon, have also added their efforts to make social distancing hassle-free in indoor and outdoor spaces. Google released a web application called SODAR [100] that uses WebXR technology to help workers maintain the necessary distance. It worked by drawing a 2-meter circle around the user as he or she walked and alerted the user if another person were to enter this circle. Amazon [101] has also developed a mirror-like tool that helps employees observe physical distancing in an office workspace. It applied Augmented Reality and Machine Learning techniques to provide visual feedback to the employees. It portrayed a person inside a red circle if they were to enter within 6-feet of any other person.

2.3 Person Detection

Although earlier work [13] explored enforcement of social distancing through DTs or 3D simulations, there is not much-reported work on rigorous validation of the system. Pedestrian or person detection is one of the key research areas in the computer vision domain. It has applications in autonomous vehicles, video surveillance, robotics, among others. In the early stage of pedestrian/person detection research, people used Haar wavelet features [54, 46, 73] or component-based pedestrian detection [46, 79, 75]. With the increase of computational power of the systems, people started to use gradient-based representation [20, 86, 75] and Deformable Part based Model (DPM) and its variants [14, 20, 59]. Dalal and Triggs [10] used a normalized histogram of orientation features to make feature descriptors. They used linear support vector machines (SVM) to train the descriptors for detecting pedestrians. Zhu and others [86] used HOG features combined with a cascade-of-rejectors approach to make person detection faster and more accurate. Felzenszwalb et al [14] followed the "divide and conquer" detection philosophy, where the training could be considered as the learning of a proper way of decomposing an object, and the inference could be considered as an ensemble of detections on different object parts. Later they developed a grammar model for person detection [20]. They constructed the grammar model to describe of the number of visible people. Sadeghi and Forsyth [59] speeded up the performance of DPM by introducing various mechanisms. To speed up feature extraction, they used HOG features and interpolated templates.

Hosang et al [23] first used Convolutional Neural Networks (CNN) for pedestrian detection. They compared different sized ConvNets with architectural differences and parameters. Although Fast and Faster RCNN methods performed well for general object detection, their performance lacked in detecting smaller pedestrian due to the low resolution of the feature map. Zhang et al [85] addressed this issue by feature fusion using a boosted forest technique. Cao et al [5] introduced a unified Multi-layer Channel Features (MCF) which integrated handcrafted features (HOG + LUV) in each layer of the CNN. Then they used multi-stage cascade AdaBoost to learn from features extracted in the layers. Hu et al [25] used feature maps extracted by CNN as input features for assembling boosted decision models to detect pedestrians. Tian et al [64] optimized pedestrian detection task with semantic segmentation to improve hard negative detection. To overcome the problem of occlusion, illumination and lighting variance, Xu et al [81] proposed a cross-modality learning framework with input images from RGB camera and thermal camera. Wang et al [76] also addressed the occlusion problem by proposing bounding box loss function named repulsion loss function. Apart from computer vision, recent efforts also investigated estimating room occupancy by listening Wifi probe emitted from mobile phones and use it to measure slip and fall risk [89].

2.4 Human Pose Estimation

Human pose estimation (HPE) had been extensively studied in computer vision. 2D HPE methods are categorized into single-person and multi-person settings. The former setting has two popular approaches: (I) Regression methods that directly map from input images to body joints; (II) Body Part Detection methods that has two steps: the first is to generate heat maps of key points for body part localization and the second step involves assembling the detected points into the whole-body skeleton. There are two deep learning-based approaches for multi-person setting as well. (I) Top-down methods [55, 27, 80, 62, 39, 47, 74, 26, 4, 84] that detect all people first and then utilize single-person HPE methods to construct key points for each; (II) Bottom-up methods [6, 56, 30, 29, 49, 18, 65, 37, 50, 32, 8] first detect the body key points without knowing the number of people, then group the key points into individual poses. Several of these systems reportedly failed to detect persons due to occlusion or truncation. Iqbal and Gall [31] built a convolutional pose machine (CPM)-based pose estimator to estimate the joint candidates and used Integer linear programming (ILP) to detect poses even in the presence of severe occlusion. Numerous works used top-down approaches, but bottom-up processes are faster than the top-down approaches since they do not detect posture for each person. Cao et al [6] built a detector named OpenPose that uses a CPM to predict joints via heatmaps and Part Affinity Fields (PAFs) to associate the key points to each person. Although OpenPose achieved high-performance in high-resolution images, they reported a poor performance with low-resolution images and occlusions. To address this problem, Kreiss et al [37] proposed BifPaf that uses Part Intensity Field (PIF) to predict the locations of body parts and a Part Association Field (PAF) to represent the joints association. These methods outperform previous OpenPose methods on low-resolution and occluded images.

2.5 Digital Twins as Synthetic Data

Synthetic data has exhibited propitious outcomes in the past [66, 22] and the turnaround time for generating labelled data for a new product variant is also drastically reduced [61]. A further benefit of using synthetic data is the ability to rapidly iterate without undergoing a time-consuming data acquisition process. However, disparities with real world scenario prevent researchers from depending on synthetic data entirely. Tremblay [67] aimed to bridge this gap between real and synthetic data by employing a technique known as Domain Randomization. Recently, Unity Technologies introduced a tool called the Perception package [1], which made it simpler to generate synthetic datasets. They tested an object detection model trained with data produced by the perception package on the Faster R-CNN [82] and found that it performed better with objects that had complex orientations, configurations, and lighting conditions than the model trained with real world data. Microsoft AirSim [43] is a tool/simulator used for the sole purpose of testing drones and autonomous vehicles in a VE. However, it was designed as more of a testing and evaluation platform than to generate synthetic data. UnrealROX [42] is another tool built over the Unreal engine to generate photorealistic synthetic datasets but targeted more towards robotic vision researchers. Unity Technologies [33] also implemented a similar approach for vision-based robotics in the Unity engine. The ParallelEye dataset [40] carried out similar efforts of generating synthetic datasets for training traffic vision models in the Unity engine. Exploiting the virtual nature of these datasets, they simulated various environmental conditions and camera parameters with the goal of generating diverse data. Although it was found that a Variable AutoEncoder (VAE) can detect camera rotation and emotion of Frey face, but neither VAE [2, 7, 52] nor Generative Adversarial Network (GAN) [12, 28, 34] can add multiple objects and persons in an image by keeping a few features constant and varying others. In summary, synthetic data has proven to be a successful alternative for collecting data to train neural networks.

2.6 Visualization of CNN

Although CNN based object recognition has achieved impressive performance, working with CNNs poses the challenge of working with a black box. The features learned in different layers of the CNN are difficult to understand unless we can visualize how they work. Explainable AI (XAI) looks to overcome these concerns, providing transparent models (white box) that allow humans to understand how an AI decision has been made; therefore, they do not rely on data only, but can be improved by human observations [102]. A brief literature survey on the application of CNN visualization techniques can be found in [103].

2.7 Summary

In summary, past literature has primarily focused on using DTs in industrial scenarios [63]. While there is literature on using twins for workspaces, only Nikolakis et al. [51] focus on mapping a person's position and posture using expensive depth cameras. Fuse et al [91] proposed a robot navigation model to determine robot’s trajectory and to help robots maintain distance in robot-human space. Ratsamee et al [92] also proposed a robot navigation model in robot-human space based on human motion and their facial orientation. Synthetic data has also proven to be a successful alternative to generating annotated datasets and particularly essential during a pandemic. Moreover, we infer that the existing state-of-the-art object detection models fail to detect humans with the same degree of accuracy as they do on general object detection. Numerous approaches were proposed to overcome this limitation. Similarly, even though various datasets and approaches exist for human pose estimation, techniques to estimate poses under large-occlusion in a multi-person setting, fall short of their single-person counterparts. Besides, the existing approaches rely on estimating key points to determine the pose rather than a direct posture mapping like standing, sitting, walking and so on. In this work, we address all these limitations using approaches detailed in the subsequent sections.

3. Our Proposed Approach

There has not been a unified approach when it comes to modeling Digital Twins in the past literature, and according to Tao et al. [63], a generic framework is critically needed. They also outlined five dimensions that are to be addressed while modeling a twin: a physical part, a virtual part, data, a connection between these, and a service modeling. We designed a DT of a laboratory space using the Unity 3D [68] game engine and its modelling tool, Probuilder [104]. The twin served as a three-dimensional illustration of the physical space whose dimensions were accurately mapped to the twin. Furthermore, the furniture and other objects in the physical space were also replicated in the virtual world. To improve the VE's photorealism, baked global illumination was used, which entails computing the lighting behavior and characteristics beforehand and storing them as texture files; this technique also reduces the computational load present in real-time global illumination. Additionally, Physically Based Materials or PBR [38] were used as they physically simulate real-life materials' properties such that they accurately reflect the flow of light and thereby achieve photorealism. We deployed the twin on a Virtual Reality (VR) setup, specifically, the HTC Vive Pro Eye [24] since VR allows for immersive and interactive virtual walkthroughs.

The physical and the virtual world are connected through sockets. Specifically, we map the weather properties of the space, such as Temperature and Humidity measured via the DHT-11 sensor. Furthermore, through a Ray cast operation, the two-dimensional coordinates are mapped to the three-dimensional coordinates of the virtual world, and hence people's movements are simulated. To ensure the twin was as photorealistic as feasible for data generation, we employed Unity's Ray tracing [72] tools, instead of the traditional Rasterized renderer. Ray tracing is a rendering technique that involves tracing individual rays of light as it bounces off virtual objects in the scene. Specifically, we used Unity's path tracing algorithm [71] with a sample count of 4096, i.e., the algorithm traces 4096 rays of light and requires 4096 frames to generate a single image. Hence, if the simulation runs at 30 frames per second, it will take around 2.3 minutes to generate one image. To automate the process and increase the dataset's diversity, we utilized Unity's perception package. We were able to generate high-fidelity ray traced synthetic datasets of humanoids in a sitting or standing pose through the perception package and its in-built randomizer. By exploiting this randomizer, the humanoids pose, i.e., the position and orientation, were changed according to a random seed with each iteration. By randomizing their pose on a fixed z-axis, we were also able to ensure that the humanoids did not clash with one another.

3.1 Planned Physical Setup

In the planned deployment, each meeting room will have a set of weather monitoring sensors and cameras (Figure 1). Data from sensors and cameras will be collected and processed on a local computer. Processing will involve noise cancellation from sensor readings through low pass filtering and calculating the number of people inside each meeting room using a Convolutional Neural Network (CNN). The processed data will be sent to a central computer equipped with a high-end Graphics Processing Unit (GPU) using network sockets. The VR-based Digital Twin will be deployed on this machine and will be updated with a real-time sensor feed. A demonstration video of the implementation can be found at https://youtu.be/XGYvDnwbyhM while a web version can be found at http://cambum.net/BT/BTWebGL/

Figure 1 below shows a schematic diagram of the planned deployment of the Digital Twin implementation, gathering real-time data from a camera and IoT sensors like temperature, humidity. A similar setup was earlier deployed for smart manufacturing capabilities [105].

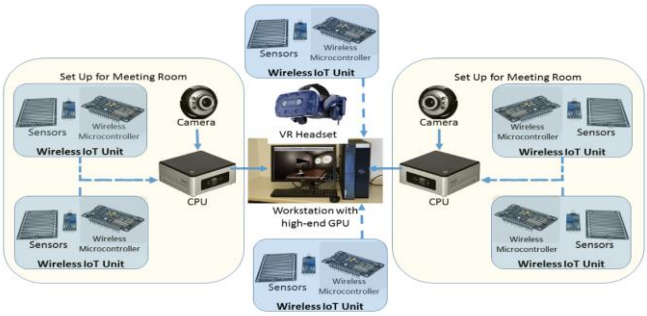


Figure 1: Planned setup of the VR-based Digital Twin.

3.2 Person Detection

We developed the Digital Twin for measuring occupancy of laboratory and enforcing social distancing. For detecting presence of humans in the laboratory workspace, we used ImageAI python library for custom object detection training using YOLOv3 architecture. We chose YOLOv3 as our person detection model based on comparison studies that suggested that YOLOv3.

- Is better than Faster RCNN, Mask RCNN, SSD, and RetinaNet in terms of accuracy and latency [48, 57, 90].
- Performed better than other models on artwork or synthetic images when the model was fine-tuned with artwork [58].

We used a transfer learning technique to fine-tune the model with a person dataset downloaded from Open Images

Dataset [53]. This dataset contained both real images as well as artwork images. We used 2022 images in total with the label 'person', showing single or multiple individuals. We separated the total dataset into an 80:20 ratio for training and validation. We prepared the dataset by converting annotation files into xml format. The existing annotation files were in Darknet format, which is the actual backend used for training YOLO. We trained the model using Keras with Tensorflow backend. Model was trained for 200 epochs with batch size of 4 using NVIDIA GeForce RTX 2070 GPU.

3.3 Pose Estimation

We developed the DT for a laboratory space where sitting and standing are the two common postures. We undertook a study to classify and to reciprocate these two postures in the DT to make it more realistic. We used a PyTorch KeypointRCNN model with a ResNet50 backbone to detect key points of the human body trained with ImageNet and COCO 2017 dataset after comparing with other pose estimation models like AlphaPose, OpenPose. The model takes input of a list of tensors, shapes, and range between [0-1]. During inference, and returns a list of dictionaries. The fields of dictionaries are as follows:

- Predicted boxes are in [x1, y1, x2, y2] format, with $0 \leq x1 < x2 \leq w$ and $0 \leq y1 < y2 \leq h$.
- Predicted labels and score of those labels for each image.
- Location of predicted key points.

The model may predict multiple bounding boxes and sets of key points for a prediction. We used the threshold value of 0.9 to filter out confident estimates and to eliminate multiple predictions.

3.4 Mapping Position and Posture to Digital Twin

We mapped each person's position and posture (sitting or standing) to the twin via virtual humanoids. This way we are not only able to visualize and generate analytics of how spaces are being used, but we are also respecting the privacy of everyone as we are not showing face or other identifiable personal features. We mapped virtual camera's position and orientation to the real-world camera. The PyTorch KeypointRCNN processes the real-world camera's video feed, through which the centroid values, i.e., the pixel position of the individuals on the screen space, was extracted. This data was packed into a JSON format and is relayed to the DT in real-time via UDP sockets. Since the real-world camera's view and the virtual camera's view are identical, the pixel positions (centroids) from the real-world camera can be mapped with ease to the virtual camera's screen space. We define a three-dimensional vector by substituting the centroid for the x and y values and the virtual cameras near clip plane as the z value. We then subtract from this vector, the virtual camera's position vector and normalize the result. The result is a direction vector that establishes the direction to travel to the centroid point from the virtual camera, or the direction in which to place the humanoid.

The main Unity feature used is the NavMesh [69] AI surface and agent. Given a layout with multiple obstacles, NavMesh automatically generates accessible and non-accessible surfaces for an agent to walk or occupy. Once this NavMesh surface is generated, the NavMesh agents (the virtual humanoids in this case) automatically avoid obstacles while moving from one point to another (also avoid colliding with one another). A* algorithm is used to compute the path between source and destination. Here, the respective destinations are continuously updated from the output of Python script which provides real-time positions of persons detected in the physical world. The virtual humanoids also have an animator controller (known as third person character controller) and a set of sitting, standing, and walking animations which are triggered correspondingly based on the posture and movements of persons detected and provided by the Python script.

We aimed to add animations (life-like movements) to the humanoids which would be spawned in the virtual world based on the detection and mapping done in the real world. Using digital humanoid models, design engineers can position and manipulate operators of varying anthropometry within the simulated work environment. Digital humanoid models are composed of more than 90 different links/joints and 140 degrees of similar to that used in many biomechanical models of the human body [93]. The humanoid model's armature (base skeleton rig) is rigged automatically with the motion capture data [94] to fit real life poses which follows the ergonomics of any work environment and hence mimic real-life movements. (Figure 2). These skeletal animations were imported into the unity project and mapped onto the existing humanoids and were called/executed based on the movements in the real-world detection (Figure 2). The various animations used in this use case included idle, walking, sitting, stand to sit and sit to stand animations to mimic the poses which may occur in a typical working environment. The animations were exported in the form of fbx format from [2] and were imported into unity game engine where all the animations were imported into an animator which would control the flow of the animations to be executed based on the actions being performed by humans in the real world. The Unity animator system provides an animation controller [95] which allows one to arrange and maintain a set of animation clips and associated animation transitions for a character or object. It allows for changing multiple animations and switching between them when certain game conditions occur. In our case humanoid would switch from a walk animation clip to a sitting animation clip whenever a person would sit on a chair in the real-world.

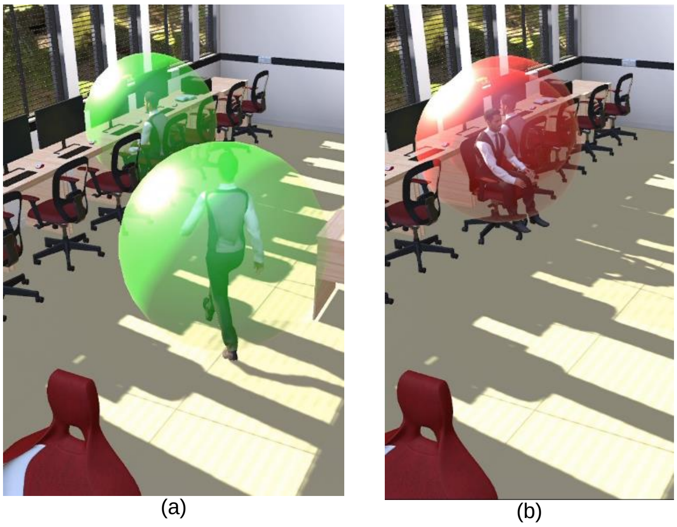


Figure 2: Movement of humanoids in virtual environment. (a) Humanoid is walking towards a chair. (b) Sitting posture of humanoid.

3.5 Social Distance Measurement

Bertoni [87] worked on silhouettes of people in outdoor environment by using 3D distances. In our case, we measured distance between persons in indoor environment. Initially, we fixed a camera at a particular height of the room in real world. Then we used this height and fixed the camera in virtual world in such a way so that field of view of the camera will be same in both virtual and real world. We used trained model to generate a set of bounding boxes and a unique ID for each humanoid. To measure distances between persons, we calculated the distance between persons from bounding box references generated by YOLO following Punn's work [57] on measuring distance among pedestrians from video recorded by road surveillance camera. We calculated bounding boxes and corresponding centroids for each bounding box in a frame recorded through the VE. We computed pairwise Euclidian distance between centroids (Equation 1) and $p \times p$ matrix, where p denoted number of persons detected at any instance.

$$d(m,n) = \sqrt{\sum_{i=1}^{n}(m_i - n_i)^2} \qquad (1)$$

Where n is two-dimensional space and m, n are two centroids in 2D space. It may be noted that the present version of our algorithm does not take posture of a person as input, but we tested the performance of the algorithm for both standing and sitting posture.

3.6 Explanation through Visualization

We mapped each person's position and posture (sitting or standing) To understand how the CNN performs, we investigated two different types of CNN visualization techniques:

(I) Visualizing intermediate layers of a CNN model following Zeiler and Fergus [106]. This visualization technique was useful to understand how successive convnet layers transformed their input. It also gave us the idea of what type of features were extracted by different filters of different layers of CNN model from input images.

(II) Grad-CAM [107] based visualization aims to understand which part of the image had a maximum association in predicting person classes. To obtain the class discriminative localization map corresponding to a specific class, we calculated the gradient with for feature maps of the last convolutional layer. These gradients were globally average pooled to obtain weights corresponding to the class, followed by a weighted combination of activation maps where finally, we applied a ReLU function. Thus, we obtained a coarse heatmap of the same size as the feature map in the last convolutional layer of the CNN model. In the final step, we resized the coarse heatmap to the input image size and overlapped on the input image. Thus, the Grad-CAM based heatmap helped us visualize which part of the image had a maximum association with the class of interest.

We applied these two techniques on synthetically generated and real images to understand if there was any difference in extracting features for predicting persons in the images. In the following sections, we described our approach for developing the VR-based Digital Twin and using it to train and explain the functioning of the CNN in detail.

4. VR Simulator Development

4.1 Modelling

The construction of an accurate virtual twin requires precise information about the object's geometrical dimensions and physical properties. Moreover, there is more than one way of implementing it. Building Information Modelling (BIM) [108] is a growing technology in the AEC industry that advances planning and designing infrastructure by portraying the building's properties in 3D. BIM is used in several past works [109, 110], and commercial services such as Tridify [111], PiXYZ [112], and Unity Reflect [113] to expedite the process of importing BIM files into game engines like Unity. Another technique, highlighted by a Siemens patent [114], is the use of depth scanners for generating a point cloud illustration of a room and then matching the point cloud data with the corresponding objects. However, due to the immediate lockdown and social distancing measures enforced in the wake of the Covid-19 pandemic, the above techniques were not feasible and could not be duly arranged. Hence, we manually modeled a part of the office workspace with the aid of an architectural drawing for our approach. We started with a meeting room that could accommodate a total of 12 people and then continued to the encompassing areas. We used Probuilder [104] and ProGrids [115] for modeling and rapid prototyping. The 3D models for the workspace furniture were procured from the online market TurboSquid [116] and Sketchfab [117], and were placed in the environment accordingly.

4.2 Realistic Rendering

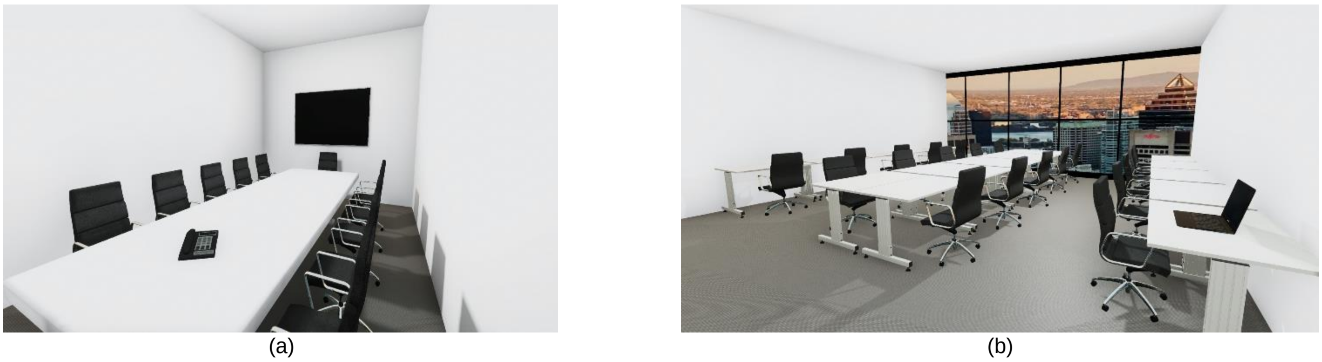
Through multiple photographs taken with standard digital cameras, we were able to ascertain the different materials that made up the meeting room, and we aimed to replicate these materials in the twin through Physically Based Rendering (PBR). PBR materials [38] enable physically simulating real-life materials' properties such that they accurately reflect the flow of light and thereby achieve photorealism. A PBR material entails several parameters such as the albedo, metallic and smoothness properties, normal maps, height maps, diffuse map, occlusion maps, among others. The respective texture maps used in our twin for the walls and the floor mat were obtained from freepbr.com.

Global Illumination (GI) is one of the most significant factors that decide how realistically a twin can resemble a real object. GI facilitates realistic light rendering by bouncing light off from surfaces, i.e., it accounts for indirect light in the scene. We employed Baked GI for our environment, which entails computing the lighting and generating lightmap textures beforehand and is therefore computationally inexpensive during runtime. Its counterpart, Realtime GI, involves calculating the light during runtime and places a substantial load on the GPU. Furthermore, Reflection Probes are placed in the environment to simulate reflections and strengthen the photorealism. Finally, Unity's Post-Processing tool is used to implement Anti-aliasing, Ambient Occlusion, Color-grading, and Auto-Exposure. The final intended result is shown in Figure 3.

For smoother processing, we optimized the twin by deleting several unnecessary polygons, such as the height adjuster in the chairs and the trays underneath the desks. Low-poly humanoid models were placed in the environment to be recognized by the person detection model. Their behavior was driven by Unity's NavMeshAgents [69]. Agents avoid one other and other obstacles in a scene through spatial reasoning obtained from a baked NavMesh. We also enabled Ray traced rendering in the virtual environment by employing Unity's path tracing algorithm. In this context, Physically Based Rendering is a category of virtual materials which mimic the real-world materials' physical properties. We have compared the performance between Rasterized Rendering and Raytracing.

4.3 Interactive Dashboards

We configured interactive dashboards inside a VR based workspace simulator. They displayed real-time sensor data like temperature and humidity, the latest statistics about the Coronavirus pandemic at the place of deployment. Sensors were interfaced with the VR machine through their respective wireless module(s). These wireless modules used a peer-to-peer connection to communicate with a VR machine using UDP protocol at a frequency of 1 Hz. Data obtained from the temperature and humidity sensors are shown as a separate circular bar (Figure 4(c)). Instantaneous values were converted to time-series values when a user dwells using his/her eye gaze using an HTC Vive Pro Eye headset or when selecting the dial via a hand-controller, providing a detailed view (Figure 4(b)). The color of the circular bars changes if the value exceeds a pre-defined threshold (Figure 4(c)). Any abrupt change in the sensor readings was also reflected instantly via both visual and haptic feedback. Haptic feedback is generated through the hand controller. The live sensor data's values could be used further for making decisions regarding air conditioning or maintaining a room temperature of the office workspace.


(a)

(b)

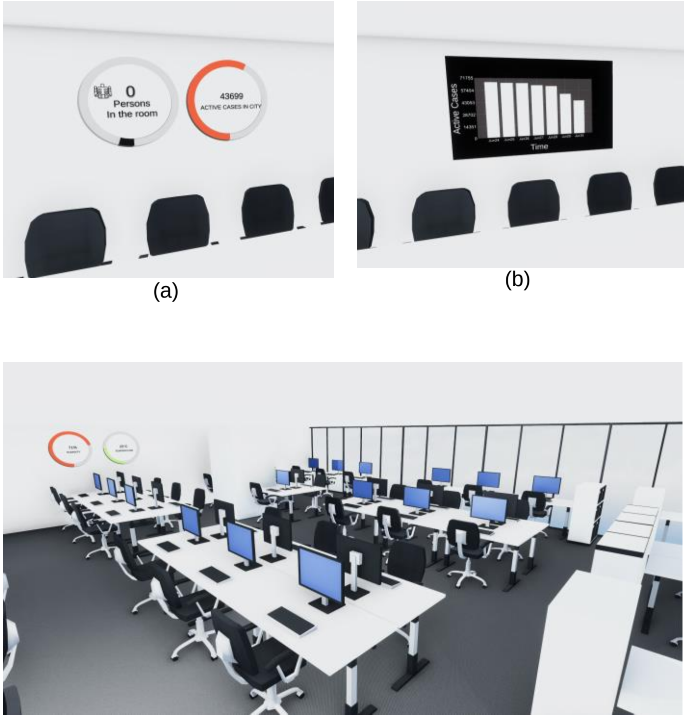
Figure 3: The Digital Twin rendered with baked global illumination and post-processing using Unity.

In addition, the dashboard displays real-time statistics from the Coronavirus pandemic obtained from the Covid19-India API [https://api.covid19india.org/]. The dashboard shows the number of active cases for the region wherein the actual workspace is present. The data was shown as a circular bar (Figure 3(a)) depicting the number of active cases to date. When a user dwells using his/her eye gaze, detailed statistics were shown for the latest phase [54] as a bar graph (Figure 4(b)).

4.4 Connecting CNN to the VR Environment

The physical implementation idea involves processing live video in a separate computer and sending the number of people detected in the live video feed into the VR setup. However, at the present stage of development, and given the previously mentioned constraints due to Covid-19, we connected the CNN model for detecting humanoids within the VR environment via a Real Time Streaming Protocol (RTSP) connection, streaming the Game View of the Unity camera to the CNN where the person detection process (as described in section 5.1) happens. Once the person detection results are obtained, we filter our predictions using their corresponding confidence scores. We select the persons with more than 0.6 confidence score, and if such person is found, we stream the results back to Unity through a UDP connection. Currently, there is no in-built option in Unity to stream its camera view; therefore, a custom solution has been compiled by us using the FFmpeg module and an RTSP Server. These functions have been implemented to stream the Unity view through the RTSP connection. Since the CNN processing speed would be different from Unity's streaming speed, we considered the RTSP buffer's latest sample to pass on to CNN. We tested person detection model on the videos recorded in both real and virtual worlds. The model processes each frame and localize persons/humanoids if detected in the frame (as shown in Figure 5). The localization is done by annotating bounding box around person. Figure 5 shows each person is annotated with one bounding box is labelled with number in the figure.

Once Unity receives object results, we add or delete humanoids inside the virtual environment. Digital humanoid models are composed of more than 90 different links/joints and 140 degrees of like that used in many biomechanical models of the human body [118]. We used Mixamo's [119] motion capture data to automatically rig the humanoid's armature (base skeleton rig) so that it reflects realistic poses of a human.



(a)

(b)

(c)

Figure 4: VR Model of the Office Space.

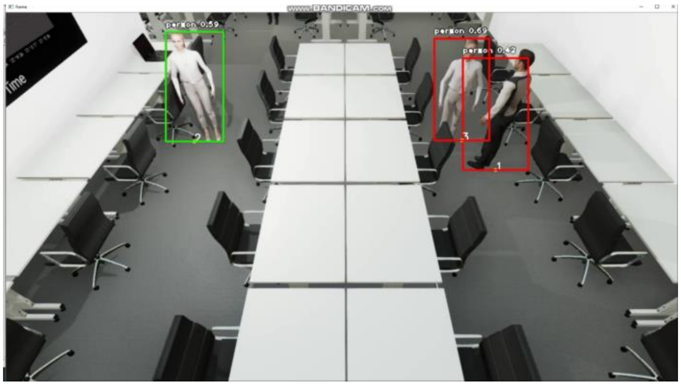


Figure 5: Humanoids are detected by person detection model and annotated with bounding boxes. Here bounding boxes are turned into red if social distancing is violated otherwise coloured in green.

4.5 Comparison with Similar Approaches

The ParallelEye dataset [40] took similar effort as ours on using VR based synthetic dataset for autonomous vehicles. UnrealROX [42] is another tool built over the Un-real engine to generate photorealistic synthetic datasets but targeted more towards robotic vision researchers. The tool focused on simulating a broad range of indoor robotic activities, in terms of both object interactions and pose. We extended the idea for a different use case and compared the system in terms of accuracy with real dataset. A different way of generating synthetic dataset uses Variable Auto Encoders (VAE) [120; 121; 122] and Generative Adversial Networks (GANs) [123]. We also compared our approach with the same. We ran the real images through a GAN implementation. A GAN consists of a Generator, which tries to fool another network known as

Discriminator that learns to distinguish between real and fake images. We used one version of the GAN, called SinGAN [124], an unconditional generative model that can be trained on a single image [66]. The model learns the internal distribution of the image's patches [125; 126; 127; 128] using multi-scale adversarial training and can generate similar images of different scales. This model is like the GAN model, except the training samples are patches of the input image rather than a set of images, and the network consists of a pyramid [129; 130; 131] of GANs of different scales. As the authors of the SinGAN paper claimed, it might produce unrealistic distorted results on coarser scales. Still, we were able to generate realistic fake images on finer scales that are indistinguishable from the real image. At finer scales, the Generator learns smaller patch distribution than at a coarser scale, giving better results in smaller scales and preserving the image's global structure (Figure 6).

There is no existing VAE-based algorithm that takes a single image and can synthesize fake images as many numbers as we want. If we have enough dataset, VAE can capture the distribution and generate more data from the same distribution. Conventional GAN has problems of non-convergence [132] and mode collapse [133], and researchers have improved it over time. Although, SinGAN model can synthesize more indistinguishable fake images similar to the original image as shown in figure 6, it offers less customization compared to VR based digital twin. As shown in earlier sections, in a digital twin, we can easily change number of persons, dress colours, number of persons, posture of persons in an image dataset keeping background and ambient light constant. Although it was found that a VAE can detect camera rotation and emotion of Frey face [134], but neither VAE nor GAN can add multiple objects and persons in an image by keeping a few features constant and varying others.

5. Validation

Validation of the entire system and in particular the social distancing module was challenging due to ongoing Covid 19 restrictions. For example, it would be risky and unethical to request volunteers to stand or sit in close proximity for generating training or testing data in the middle of the pandemic. However, we would not also be able to validate the system properly if we only have data with limited participants standing and sitting far apart from themselves. Hence, we validated all models using both real and synthetic data. We collected real data after following social distancing norm and following appropriate ethical approval. We also generated synthetic data using the VR Digital Twin. For validation through VR generated synthetic data, we followed earlier examples of Parallel Eye Dataset [40] and Leban's [33] robot workspace but used the previously proposed digital twin of laboratory space. In the following paragraphs, we described methods to generate both real and synthetic data.

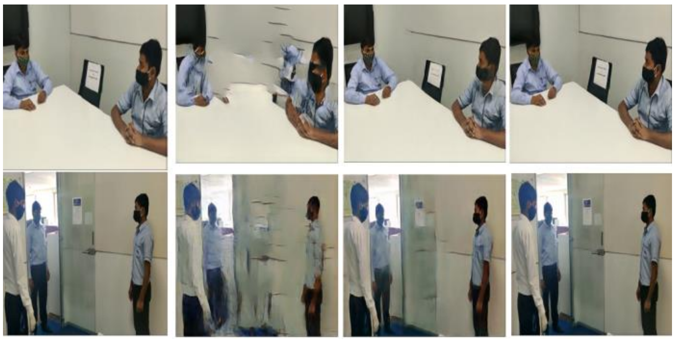


Figure 6: Left column: (a) Original input images (1st column); (b) Random samples from a single image at n=6 (2nd column), n=11 (3rd column), and n=25 (4th column).

Real Data: We deployed a Logitech HD camera in a fixed position of the office and laboratory space to get similar view of the room as seen in the synthetic dataset preparation setup. To validate person detection model, we recorded short videos in the physical world (Figure 7(d)) and the virtual environment (Figure 7(c)). In physical world video, we recorded multiple situations such as occluding persons and varying lighting conditions at the same physical space used for VR modelling (Figure 7(c)). In the virtual environment video, we recorded keeping the ambient light and room setup as constant parameters and the following as independent parameters: (I) changing number of humanoids in frames (one to four humanoid), (II) posture of humanoids (seating and standing), (IV) occluding humanoids (yes or no). We also captured a live video with the same setup for a duration of 5 minutes for pose estimation. We randomly selected then 300 images 640 × 480 pixels resolution from this video to test the model for pose classification.

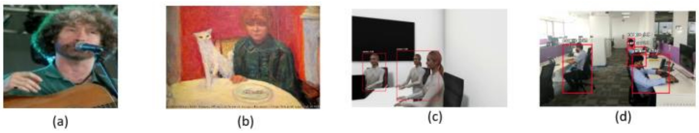


Figure 7: Left column: (a) Original input images (1st column); (b) Random samples from a single image at n=6 (2nd column), n=11 (3rd column), and n=25 (4th column).

Synthetic Data: To ensure the digital twin was as photorealistic as feasible for data generation, we employed Unity's Ray tracing [72] tools, instead of the traditional Rasterized renderer. Ray tracing is a rendering technique that involves tracing individual rays of light as it bounces off virtual objects in the scene. Specifically, we used Unity's path tracing algorithm [71] with a sample count of 4096, i.e., the algorithm traces 4096 rays of light and requires 4096 frames to generate a single image. Hence, if the simulation runs at 30 frames per second, it will take around 2.3 minutes to generate one image. To automate the process and increase the dataset's diversity, we utilized Unity's perception package. We were able to generate high-fidelity ray traced synthetic datasets of humanoids in a sitting or standing pose through the perception package and its in-built randomizer. By exploiting this randomizer, the humanoids pose, i.e., the position and orientation, were changed according to a random seed with each iteration. By randomizing their pose on a fixed z-axis, we were also able to ensure that the humanoids did not clash with one another. We placed humanoid figures in close proximity in synthetic data.

5.1 Person Detection

We tested our trained model on both real and synthetic sequences of images. We calculated the accuracy of our model using the formula: Accuracy = (TP+TN)/(TP+FP+FN+TN), where TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, and False Negative, respectively.

Office space: We tested a total of 9000 images divided into three classes – real, synthetic images generated without ray tracing and synthetic images generated through ray tracing. We considered synthetic images without ray tracing as ray tracing is computationally intensive and in practical implementation, we may need to deploy the digital twin without ray tracing based on availability of GPUs. We found an overall accuracy of 96.044% (std error 0.07) for real images, 96.981% (std error 0.126) for synthetic images without using raytracing and 94.25% (std error 0.09) with synthetic images using ray tracing (Figure 8(b)). We analyzed accuracies to determine if the performance of CNN is significantly different between real and synthetic images. We listed accuracies for all conditions (different number of persons, posture, and occlusion) separately and found that except when one person is occluded, the interquartile range for all conditions to be zero and median, first and third quartile is 100% for both real and synthetic images

Laboratory space: We tested total of 9600 images divided into two classes – real and synthetic to compare performance of the model. We found an overall accuracy of 91% (std error 0.11) for real images and 94% (std error 0.04) for synthetic images (Figure 8(b)). We also observed overall latency of 14.25 frames/second and 14.17 frames/second for real images and synthetic images, respectively. We measured accuracy for all 16 combinations (Postures (2) × Occlusion (2) × Number of Persons (4)) and found lowest accuracy when one person was occluded in real image. In all other conditions, we found accuracy was higher than 80%. We undertook non-parametric statistical hypothesis testing to compare accuracy and latency between real and synthetic images. Using Wilcoxon Signed-Rank test, we did not find any significant difference neither in accuracy nor in latency at p<0.01.

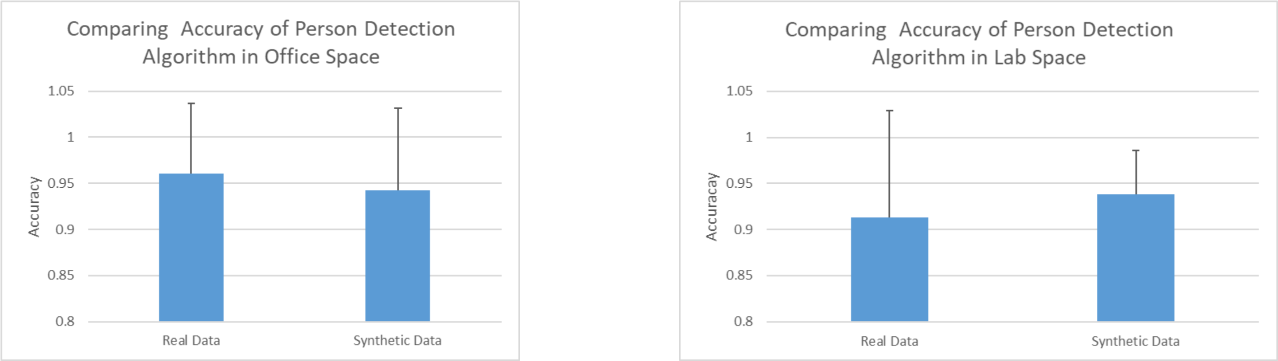
- We observed few cases (for e.g., one person sitting with occlusion) where accuracy on real images were comparatively lower than synthetic images. It might be due to uncontrolled lighting illumination, similar color contrast between dresses of persons and background and so on.
- The difference in accuracy among two conditions were 3%, which means in practical cases we will not miss a person.

5.2 Pose Estimation

We tested the model performance on both synthetic and real data as explained in section 3 and real data. We obtained 17 key points of the human body as output from the model. We calculated distance vectors from left hip to left shoulder and to left knee. By using the dot product formula between these two vectors, we obtained left hip's interior angle. We set the threshold for hip's interior angle to be 160 degree and we concluded if the angle was more than the threshold value for a posture, we labelled as 'standing', else labelled as 'sitting'. We found accuracy of 83.82% and 84.73% for real data and synthetic data respectively.

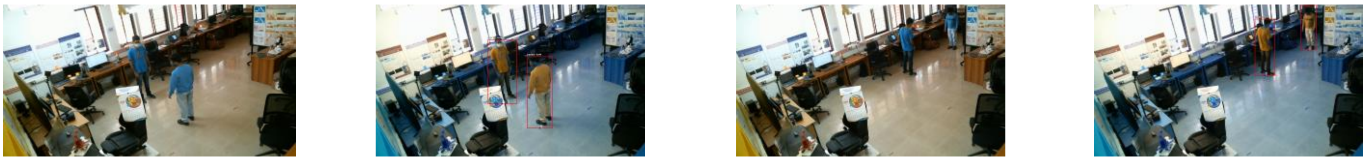
5.3 Social Distancing

We fixed a camera at a particular height of the room in real world. We then collected distances between persons in different position of the room and that is also with different distances. To fix the distances between persons, we used distance of 2ft, 4ft, 6ft, and 8ft and put markers on the floors. We then asked our lab colleagues to stand on those markers and captured images (Figure 9). We measured correlation of actual Euclidian distance among people in real workspace with the virtual distance measured from the VE, and the correlation coefficient was found to be $r = 0.82, p < 0.01, R^2 = 0.67$. Figure 10 shows the scatter plot explaining correlation between real world and pixel wise distances. In this context, we measured correlation of this measurement with the distance measured through the virtual environment in office space and the correlation coefficient was r=0.99, p<0.01.



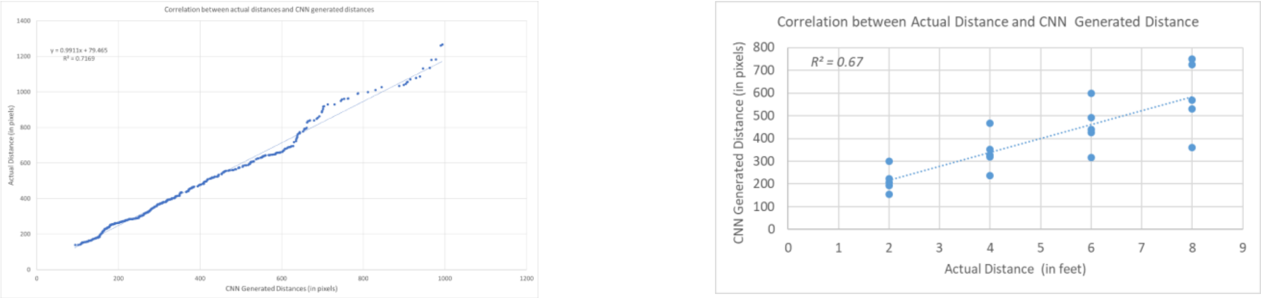
(a) (b)

Figure 8: Comparing Accuracy of Person Detection on Real and Synthetic Data in both office and laboratory space.



(a) (b) (c) (d)

Figure 9: Sample images for measuring social distances between two persons in different position of the room. (a) and (b) are showing persons are standing in 4ft and (c) and (d) are showing persons standing in distance of 8ft.



(a) (b)

Figure 10: Scatter Plot between Distances measured from Real World environment and using person detection model. (a) Office space; (b) Laboratory space.

5.4 Discussion

We compared performance of a representative CNN model among real and synthetically generated images using a VR digital twin of office space and laboratory space.

Office space: We noted that the interquartile range of accuracies across different combinations of number of persons, occlusion and posture is zero with first, second and third quartile is at 100%. The highest number of images with less than 100% accuracy occurred when one or more persons are occluded, and it was similar for both real and synthetic images. We observed few cases (for e.g., two persons sitting with occlusion) where accuracy on real images were comparatively lower than synthetic images. It might be due to uncontrolled lighting illumination, similar color contrast between dresses of persons and background and so on. The difference in accuracies among three conditions were less than 2%, which will not have much effect for practical use cases. The calculated distances from synthetic image correlated with 0.99 coefficient with real distances.

We found three conditions (two persons sitting with occlusion, three persons standing without occlusion, three persons sitting with and without occlusion) where accuracy on real images were comparatively less than on synthetic images without ray tracing. We observed that, although model was able to detect persons in those conditions in real world, false positive rate was higher. It might happen due to uncontrolled ambient lighting conditions in real world, indistinguishable similarity between dress colors and background in the images.

Laboratory Space: We did not observe reduction in accuracy of the model with increasing the number of persons. The lowest accuracy (83.34%) was found with the combination of three persons in sitting postures and occluded. We also observed that with four persons in all combinations, accuracy was above 94%. A video demonstration of the complete system can be found at https://youtu.be/4D2iGaDrP2c.

6. Applications

In this section, we described two possible applications of the system beyond remote monitoring of laboratory space. The first case study extended the social distancing measure to estimate maximum occupancy of a room maintaining social distance. The second case study presents a validation of the concept for developing digital twin of an Indian road.

6.1 Maximum Occupancy of a Room

In the previous section, we reported that we were able to measure social distancing by computing the inter-person distances in the virtual world and our analysis showed a high correlation with the ground truth obtained from the real data. We extend the work to estimate the maximum number of occupants in a closed space maintaining social distancing. To determine the maximum occupancy, we approached the problem through a circle packing technique [77], which can be defined as the arrangement of circles inside a given boundary such that no two circles overlap, and some (or all) of them are mutually tangential. In our case, individuals were represented as the center of a circle of radius 3ft. If two individuals were to maintain a distance of more than 6ft from each other, then the distance between their circle centers would be greater than or equal to twice their radius.

It should be noted that while the circle packing technique requires the circles to be within the bounds of the space, in our case only their centers are required to do so. For instance, people could be stationed at the corner of a room or leaning against a wall in a meeting room, in which case their circle bounds will overlap with the room boundary. However, their centers would be within the bounds of the room. We termed the region where the people can be stationed as the feasible region. The feasible region excludes the obstacles in a space, forming an irregular concave polygon. Hence, defining the feasible region is crucial for the proposed algorithm. To define the feasible region in the twin, we employed Unity's NavMesh feature. A Navigation mesh in Unity is an abstract data structure used to aid Non-Playable Characters (NPCs) or Artificial Intelligence agents in pathfinding through complicated spaces [69, 78]. Constructing a Navigation mesh involves first defining the surfaces where NPCs would be able to walk. We did this by assigning the furniture, pillars, and other obstacles in the room as NavMesh obstacles. Unity generates a custom polygon that curves around the obstacles defined earlier after mesh building, resulting in the feasible region, as depicted in Figure 11(a). The distance maintained by the NavMesh from the obstacles can be controlled by adjusting the Humanoid settings of the NavMesh, specifically by increasing or reducing the humanoid's radius. A larger radius leads to a more spaced distance from the obstacles and vice versa.

Packing Process: We adopted and modified the layout strategy from Yuan et al. [83], since we found it to be adequate considering our specific use-case of packing circles in irregular shapes. Yuan's algorithm implemented a left bottom placement strategy, namely placing circles in the left-most and bottom-most position first and growing from there.

Our algorithm was integrated into the Unity game engine and exploited its physics simulation capabilities. We utilized a ray casting approach to counter the problem of assessing whether a circle's center is within the feasible region. In computer graphics, ray casting involves shooting an imaginary straight ray from a point to ascertain which objects it collides with along its path. The circle's center was designated as the point of origin for our ray, the direction being downwards, and its parameters were set such that it travelled one meter. We then checked to see if the ray collided with the feasible region; if it did, the circle was deemed valid. To ensure the circles did not overlap, we assigned a collision detection protocol. The protocol states that if a circle collides with any other already packed circle upon instantiation, it is not deemed valid. We employed an imaginary sphere collider around every circle. If another circle were to overlap, then their sphere colliders would throw a trigger event, stating that a collision has occurred. Thereby every circle undergoes a two-step check before they are deemed valid to be packed. Furthermore, due to the polygon's irregular nature, we employed 128 discrete points for our layout circle [83].

Jump-Off Circles: Due to the irregular structure of the feasible region, the algorithm may face a dead-end, i.e., it reaches a point where it has no way of moving forward, while a large portion of the feasible region remains unpacked. We implemented 'jump-off circles', which are randomized circles where the algorithm can jump to and resume packing when it faces a dead-end. While packing, the algorithm is designed to choose the first out of sixteen circles (in the counterclockwise direction) that meets the previous two criteria. After finding this circle, it also examines the remaining circles as to whether they satisfy the two criteria. If they do, the algorithm designates those circles as 'jump-off points' and stores them in a list data structure. The central circle from which they originated is designated as the 'jump-off circle'. The 'jump-off circle' and the list of 'jump-off points' are stored as key-value pairs in an ordered dictionary. If later, these 'jump-off points' are found to be overlapping with other packed circles, they are deemed invalid and removed from the dictionary.

Consequently, when the algorithm reaches a dead-end, it jumps to a jump-off circle, picks the earliest jump-off point in the clockwise direction, and resumes packing. If no jump-off circles exist, the algorithm assumes no space is left to pack and ceases the packing process. Figure 11(b) shows the result of the packing process after implementing jump-off points, and the pseudocode for the algorithm is outlined below.

ALGORITHM 1: Estimating Maximum Space Occupancy

```
Input: left-most position on the feasible region to start the packing process from
Output: packing structure
function PACKINGPROCESS (position)
    create a packed circle with centre position
    create a layout circle with centre position
    for every discrete point on the layout circle with position i, do
        create a circle with centre i
        if circle is inside the feasible region then
            if circle does not overlap other packed circles then
                position = circle's centre
                flag = 0
                if circle has jump-off points then
                    store jump-off points in a dictionary, and designate circle as jump-off circle
        else
            flag = 1
    if flag = 0 then
        return PACKINGPROCESS (position)
    else if flag = 1 and jump-off circles exist then
        position = jump-off point
    else if flag = 1 and no jump-off circles exist then
        return
end
```

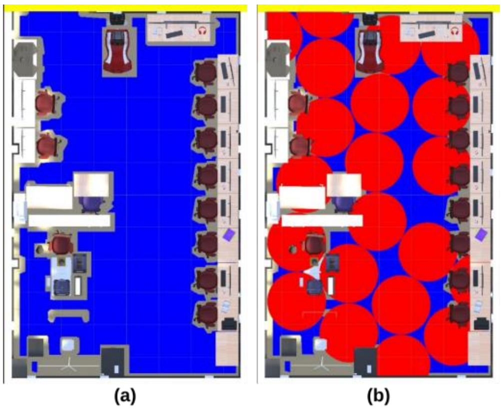


Figure 11: (a) The feasible region (in blue) extracted from Unity's NavMesh, and which conveniently ignores the obstacles in the room. (b) The result of the packing process on the feasible region. The algorithm was able to pack 21 circles.

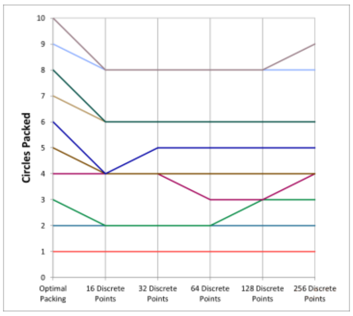


Figure 12: A parallel coordinate plot depicting the comparison of our algorithm's packing w.r.t. different discrete points.

Validation: Since our algorithm allows circles to cross the packing polygon's boundary, the traditional packing efficiency calculation of dividing the total area of the polygon by the total area of the packed circles could not be used. Hence, we compute the pixel count in a top view image of the feasible region before packing (marked in blue in figure 11) and compared it with the pixel count of packed circles after packing. In Figure 11(b), where the circle radius was 3 ft, the blue pixels covered, or the packing efficiency was found to be 89.2%. Due to our unique circumstance, we do not have a ground truth metric to compare this packing efficiency. However, there exists optimal packing structures for packing n circles in a unit square, where n ranges from one to twenty [77]. We compare this optimal packing to our algorithms' performance of packing n circles ranging from one to ten and reported the absolute differences in Table 1. The absolute differences between our algorithm and the optimal packing were found to be at most two circles. We further analysed our algorithms' ability to pack circles by increasing the number of discrete points and reported results in Figure 12. It was found that upon increasing the discrete points, our algorithm was able to more closely resemble the optimal packing (256 discrete points was able to pack 9 circles for the optimal packing of 10), at the expense of more computation power. It should be noted that this algorithm serves only to estimate the maximum occupancy limit of a space, similar to the occupancy limits of an elevator, and does not serve as a definite tool to support public health decisions.

Table 1: Comparison of our algorithm's packing to the proved optimal packing of n circles in a unit square

Diameter of Circle	*Optimal Packing*	*Our Algorithm Packed (16 Discrete Points)*	*Absolute Difference*
1.000	1	1	0
0.586	2	2	0
0.509	3	2	1
0.500	4	4	0
0.414	5	4	1
0.375	6	4	2
0.349	7	6	1
0.341	8	6	2
0.333	9	8	1
0.296	10	8	2

6.2 Digital Twin of an Indian Road

The concept of using digital twin to generate customized synthetic data can also be explored for developing (semi)-autonomous vehicles. Presently, there are limited data available for developing computer vision models for detecting non-conventional traffic participants. It is also difficult to generate real dataset for certain types of road objects like children or animals. We developed a digital twin of an Indian road. Our approach was similar to the Parallel Eye Dataset [40] but was richer in terms of developing VR models of non-conventional traffic participants like animals, three wheelers and so on. We compared performance of YOLOv3 for detecting unusual traffic participants for both real and synthetic datasets. We modelled an Indian road in Unity, comprising of diverse types of vehicles and stray animals. We employed the terrain package to create an artificial environment, and we placed several three-dimensional models of animals and Indian vehicles arbitrarily to reflect a real-world scenario. The synthetic dataset was generated from the perspective of a person in a driver's seat of a car. We compared performance of YOLOv3 on both real (obtained from an Indian road dataset [88]) and synthetic images. We found that the pattern in average detection rate for different classes (for instance, the detection rate for car is the highest for both datasets) are similar across real and synthetic data (Figure 13).

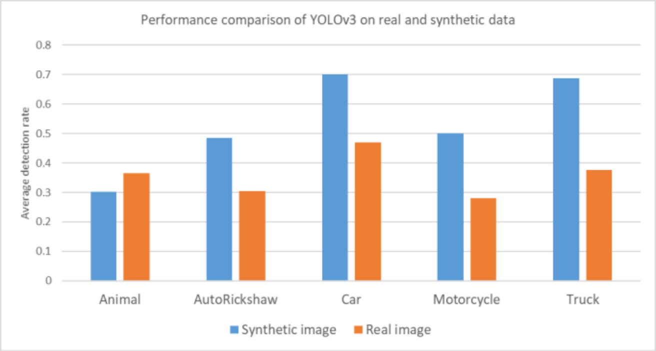


Figure 13: Performance comparison of YOLOv3 in both virtual and real environment in the context of an Indian road scenario.

7. Explanation through visualization

In the previous section, we reported that YOLO had its lowest accuracy where one or more persons were partially occluded. To understand this result, we used an intermediate layer visualization technique and the Grad-CAM technique to explain the person detection model's performance. Grad-CAM calculates each pixel value of the feature maps in the last convolution layer on the predicted class [107]. It does not need any information related to bounding box regression, which is typically used to localize an object in the image. As the YOLO model does not allow reading data from intermittent layers, we used a VGG16 classification model pretrained with the ILSVRC ImageNet dataset. We prepared our dataset combining five different classes (i.e., airplane, bicycle, car, motorbike, and face) of images downloaded from Kaggle, Google Image, the Caltech face image dataset, and the Georgia Tech Face database. We trained the model with a total of 3513 images parted in training and validation datasets (80:20) for 100 epochs. To understand how the CNN model can classify the input image, we need to understand how our model sees the input image by looking at the output of its intermediate layers. We visualized activations in the (n/4)-th convolutional layer, (n/2)-th convolutional layer, (3n/4)-th convolutional layer, and the n-th convolutional layer of the trained model. To visualize the heatmap generated by the Grad-CAM method, we used a pretrained VGG16 model. Although this pretrained model does not include any person class, it has different classes related to dresses (e.g., 't-shirt', 'jeans'), which are relevant for the localization of the individual in the image. We generated a heatmap corresponding to the 't-shirt' and 'jeans' classes to identify people in the images. We visualized the performance of the CNN on person prediction in both real and synthetic generated images. We generated the output of CNN from the layers mentioned above for both types of images to understand whether CNN handles synthetic generated and real images differently or in the same manner. We found that the first few convolution layers of the model extracted basic features (edges, contours) of the object and retained maximum information from the input image (Figure 14(b), 15(b)). As we found deeper in the model, activations became less visually interpretable (Figure 14(c) – 14(e), 15(c) – 15(e)). The model started to extract abstract features (e.g., patch-based features like the texture of body parts of a humanoid in Figure 14 or a person in Figure 15). In the deeper level of the network resolution, the feature map starts decreasing, but spatial information increases. If we observe all four-feature map outputs (Figure 14(c) – 14(e) and 15(c) – 15(e)), it is evident that in each transformation model it eliminated the background or any irrelevant information and refined useful information related to class of objects.

We also visualized heatmaps of class activation to understand which part of the object was responsible for letting the model classify correctly. In this context, the class activation map tells us which part of an image corresponds to a class of objects. In Figure 16, we showed a heatmap on real-world images synthetically generated images (Figure 16(a) – (c)). We found that different body areas were strongly activated, where brown color corresponded to the highest gradient score, and cyan color corresponded to the lowest gradient score. Heatmap based visualization helped us identifying which part of the image contributes to the case of false positive or false negative results.

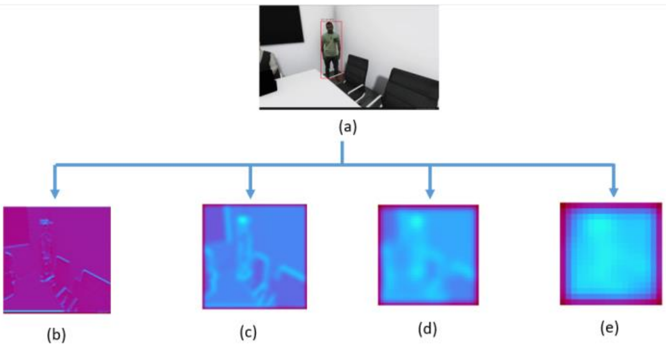


Figure 14 (a) Input image (humanoids in a synthetic generated image). A red box indicates that YOLO detected a person; (b) 28th channel of the activation of 3rd convolution layer; (c) 28th channel of the activation of 7th convolution layer; (d) 28th channel of the activation of the 10th convolution layer; (e) 510th channel of the activation of the 13th convolution layer. (Please note that this figure is best viewed in electronic form).

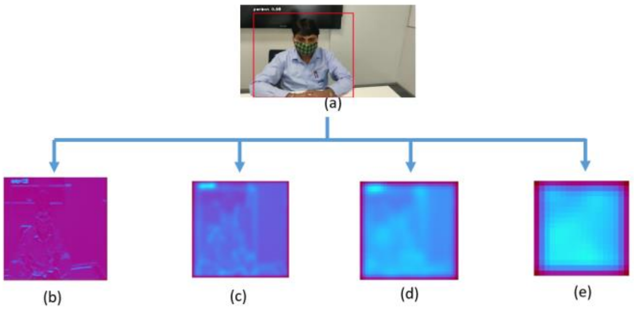


Figure 15 (a) Input test image (person in real-world image); (b) 28th channel of the activation of 3rd convolution layer; (c) 28th channel of the activation of 7th convolution layer; (d) 28th channel of the activation of the 10th convolution layer; (e) 510th channel of the activation of the 13th convolution layer. (Please note that this figure is best viewed in electronic form).

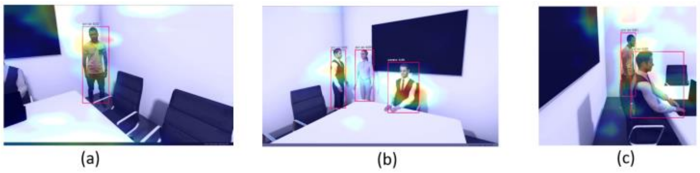


Figure 16 Grad-CAM based heatmap for three different situations where YOLOv3's performance was different in terms of accuracy (shown in red bounding boxes). (a) YOLOv3 failed to detect a partially occluded person; (b) YOLO detected all individuals; (c) YOLOv3 detected all individuals with different postures and dressing different colors.

To understand how different independent variables contributed to CNN's performance, we tested on synthetic images with different parameters. We started with an occluded person's image where the YOLOv3 failed to detect individuals, and the accuracy dropped to 50% (Figure 17(a)). We found that occlusion made it difficult for the model to get enough information from the partially occluded person to localize it, although it could generate a heatmap for whole body areas of a standing person as it was fully visible (Figure 17(a)). We did a second check with a different image in which YOLOv3 was able to detect all the people (Figure 17(b)). When we looked closely at the heatmap areas, we found strong associativity between the bounding box region and the heatmap areas. Although the female humanoid in this image was weakly classified, the heatmap covered the maximum upper body parts visible in the image for all three individuals. As previously mentioned, the brown color corresponded to the highest gradient score, and the cyan color corresponded to the lowest gradient score. We tested the heatmap with a third image where the humanoid figures were wearing different colors (green and white) and were positioned in different postures (i.e., one person was sitting, and a second person was standing). The Grad-CAM heatmap gave a strong visual cue about the location of these two different individuals (Figure 16(c)). These results confirm our idea of using this visualization technique to analyze failure modes of a CNN model and take corrective steps, such as increasing the camera field of view and location to record a full-frontal view of the humanoids, in this case, to increase the accuracy of the model.

8. General discussion

This section summarizes our work and identifies its novelty and utilities.

8.1 Summary

This work proposes a new way of validating the accuracy of CNNs through synthetic customized video generated in an immersive environment. A case study demonstrates this implementation's possible application towards the development of an automated social distance measurement system in a physical office and laboratory space. We have presented training and testing accuracy of detecting individuals using CNN based person detection model and used data visualization techniques to explain the working of the model for both real-world and synthetically generated video.

8.2 Accuracy of Person Detection

We noted that we achieved 100% median accuracy for person detection and 0.99 correlation on physical distance measurement. CNN is a rapidly evolving field with new models are frequently appearing in literature and the accuracy of person detection even with occlusion can further be increased using customized CNN models. However, it may be noted that this paper is not focused on developing CNN for person detection, rather proposing a new way of validating CNN model using VR based synthetic dataset. If a different CNN model is used other than YOLOv3, we can also train it with synthetic dataset and can achieve similar accuracy in real life deployment. The present social distance measurement algorithm works within the visual field of a webcam inside a room, but future version will implement

3D distance measurement like Bertoni's [40] Monocular 3D Localization algorithm.

8.3 Utility

The proposed VR prototype will be deployed as a VR based digital twin of an office space implementing real-time person detection and environmental variable monitoring capabilities through interactive dashboards. In addition, the VR interface would show real time Covid-19 statistics at the place of deployment and measure the number of people in the space, and their relative position and posture. This can be very valuable to monitor social distancing measures in office spaces. A second benefit could be that an observer can undertake a detailed remote virtual walk through the office space, which would not be possible with a standard multi-screen video from security cameras.

The concept of validating a CNN through synthetic video can have utilities beyond these particular use cases of measuring social distancing at an office space or laboratory space. For example, for both unmanned ground and aerial vehicles, synthetic videos can be used to validate machine vision systems where real-time video generation is difficult, for example, inside a hazardous place such as a nuclear power station or a high-security zone such as inside of a military facility.

8.4 Value Addition

- During the past few months, a plethora of computer vision projects on calculating social distancing was produced. However, most of these systems were not as rigorously validated as traditional machine vision systems for autonomous vehicles or face recognition due to a lack of appropriate data. Bertoni's [40] algorithm was validated for outdoor environment but not for indoor office workspace. Our paper proposes a new way of validating a machine learning-based person detection system using synthetically generated video in an immersive environment.
- Digital twins are traditionally used to optimize or simulate the process life cycle or maintenance of assets. Our work proposes a new use of digital twins for enhancing workplace safety by measuring social distance.
- We also showed one application of the visualization technique in synthetic images to understand why CNN-based object detection models worked or failed to detect individuals from the images. We compared the performance of YOLO using different independent parameters to understand how it works in different situations. The heatmap based visualization helped us to get a visual explanation about the working of the CNN model. This approach is novel in that a similar approach can be used for other CNN models for different applications, and it is a step further towards the collective goal of explainable AI (XAI).

9. Conclusion

This work presented a VR based digital twin implementation of a physical office and laboratory space towards the goal of using it as an automatic social distancing measurement system. The VR environment was enhanced with an interactive dashboard showing information collected from physical sensors and the latest statistics on Covid-19. We presented a person-detection method and a pose-estimation model to determine the number of people in a room, their respective poses and mapped them in a virtual environment for measuring social distance. Our proposed pipeline along with the Digital Twin of the shared space visualizes both environmental and human behavior aspects preserving privacy of individuals and improves latency of such monitoring systems as it does not require to stream live video. We intend to extend our human pose dataset with wide range of poses with multi-person occluded setting. We also presented a circle-packing algorithm which can be implemented using Unity to obtain the optimal number of people that can be accommodated in any given space maintaining social distance and proposed to use VR generated synthetic data for training and testing computer vision models. We also used two different data visualization techniques to explain how a complex CNN works, therefore looking towards the advancement of explainable AI, and we used it to improve the performance of the CNN. Our hope is that the proposed solution will help measure occupancy accurately and contribute to enhancing the safety of workspaces by enforcing social distancing measures.

References

[1] Unity Perception Package. Unity Technologies. Retrieved May 27, 2020 from https://github.com/Unity-Technologies/com.unity.perception

[2] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. arXiv preprint arXiv:1509.00519, 2015

[3] Considerations relating to social distancing measures in response to COVID-19 - second update. C. Adlhoch. Retrieved March 10, 2021 from https://www.ecdc.europa.eu/en/publications-data/considerations-relating-social-distancing-measures-response-covid-19-second

[4] Yuanhao Cai, Zhicheng Wang, Zhengxiong Luo, Binyi Yin, Angang Du, Haoqian Wang, Xiangyu Zhang, Xinyu Zhou, Erjin Zhou, and Jian Sun. 2020. Learning delicate local representations for multi-person pose estimation. In European Conference on Computer Vision, pages 455–472. Springer, 2020

[5] Jiale Cao, Yanwei Pang, and Xuelong Li. 2017. Learning multilayer channel features for pedestrian detection. IEEE transactions on image processing, 26(7):3210–3220, 2017

[6] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 2017. Realtime multi-person 2d pose estimation using part affinity fields. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7291–7299, 2017

[7] Ricky TQ Chen, Xuechen Li, Roger Grosse, and David Duvenaud. Isolating sources of disentanglement in variational autoencoders. arXiv preprint arXiv:1802.04942, 2018

[8] Bowen Cheng, Bin Xiao, Jingdong Wang, Honghui Shi, Thomas S Huang, and Lei Zhang. Higherhrnet: Scale-aware representation learning for bottom-up human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pages 5386–5395. 2020

[9] Components 101. NodeMCU ESP8266. Retrieved May 10, 2021 from https://components101.com/development-boards/nodemcu-esp8266-pinout-features-and-datasheet

[10] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). volume 1. pages 886–893. Ieee

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition. pages 248–255. Ieee

[12] Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep generative image models using a Laplacian pyramid of adversarial networks. arXiv preprint arXiv:1506.05751

[13] Tingxiang Fan, Zhiming Chen, Xuan Zhao, Jing Liang, Cong Shen, Dinesh Manocha, Jia Pan, and Wei Zhang. 2020. Autonomous social distancing in urban environments using a quadruped robot. arXiv preprint arXiv:2008.08889

[14] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. 2009. Object detection with discriminatively trained part-based models. IEEE transactions on pattern analysis and machine intelligence. 32(9):1627–1645

[15] Neil M Ferguson, Derek AT Cummings, Simon Cauchemez, Christophe Fraser, Steven Riley, Aronrag Meeyai, Sopon Iamsirithaworn, and Donald S Burke. 2005. Strategies for containing an emerging influenza pandemic in southeast asia. Nature, 437(7056):209–214

[16] Neil M Ferguson, Derek AT Cummings, Christophe Fraser, James C Cajka, Philip C Cooley, and Donald S Burke. 2006. Strategies for mitigating an influenza pandemic. Nature, 442(7101):448–452

[17] Stepehen Farguson. Apollo 13: The First Digital Twin. Retrieved May 10, 2021 from https://blogs.sw.siemens.com/simcenter/apollo-13-the-first-digital-twin/

[18] Mihai Fieraru, Anna Khoreva, Leonid Pishchulin, and Bernt Schiele. 2018. Learning to refine human pose estimation. In Proceedings of the IEEE conference on computer vision and pattern recognition workshops. pages 205–214

[19] Christophe Fraser, Steven Riley, Roy M Anderson, and Neil M Ferguson. 2004. Factors that make an infectious disease outbreak controllable. Proceedings of the National Academy of Sciences. 101(16):6146–6151

[20] Ross Girshick, Pedro Felzenszwalb, and David McAllester. 2011. Object detection with grammar models. Advances in Neural Information Processing Systems. 24:442–450

[21] Edward Glaessgen and David Stargel. 2012. The digital twin paradigm for future nasa and us air force vehicles. In 53rd AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference 20th AIAA/ASME/AHS adaptive structures conference 14th AIAA. page 1818

[22] Stefan Hinterstoisser, Olivier Pauly, Hauke Heibel, Marek Martina, and Martin Bokeloh. 2019. An annotation saved is an annotation earned: Using fully synthetic training for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops. pages 0–0

[23] Jan Hosang, Mohamed Omran, Rodrigo Benenson, and Bernt Schiele. 2015. Taking a deeper look at pedestrians. In Proceedings of the IEEE conference on computer vision and pattern recognition. pages 4073–4082

[24] HTC vive pro eye overview. Retrieved May 10, 2021 from https://www.vive.com/sea/product/vive-pro-eye/overview/

[25] Qichang Hu, Peng Wang, Chunhua Shen, Anton van den Hengel, and Fatih Porikli. 2017. Pushing the limits of deep cnns for pedestrian detection. IEEE Transactions on Circuits and Systems for Video Technology. 28(6):1358–1368

[26] Junjie Huang, Zheng Zhu, Feng Guo, and Guan Huang. 2020. The devil is in the details: Delving into unbiased data processing for human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pages 5700–5709

[27] Shaoli Huang, Mingming Gong, and Dacheng Tao. 2017. A coarsefine network for keypoint localization. In Proceedings of the IEEE International Conference on Computer Vision. pages 3028–3037

[28] Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. 2017. Stacked generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition. pages 5077–5086

[29] Eldar Insafutdinov, Mykhaylo Andriluka, Leonid Pishchulin, Siyu Tang, Evgeny Levinkov, Bjoern Andres, and Bernt Schiele. 2017. Arttrack: Articulated multi-person tracking in the wild. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6457–6465

[30] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. 2016. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In European Conference on Computer Vision. pages 34–50. Springer

[31] Umar Iqbal and Juergen Gall. 2016. Multi-person pose estimation with local joint-to-person associations. In European Conference on Computer Vision. pages 627–642. Springer

[32] Sheng Jin, Wentao Liu, Enze Xie, Wenhai Wang, Chen Qian, Wanli Ouyang, and Ping Luo. 2020. Differentiable hierarchical graph grouping for multi-person pose estimation. In European Conference on Computer Vision. pages 718–734. Springer

[33] Jonathan Leban, Amanda Trang, Sarah Wolf, Jacob Platin, Anthony Navarro, Sujoy Ganguly and Sarah Gibson. Teaching robots to see with unity. Retrieved May 10, 2021 from https:/ / blogs.unity3d.com/2021/03/02/teaching-robots-to-see-with-unity/

[34] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2017. Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196

[35] Siavash H Khajavi, Naser Hossein Motlagh, Alireza Jaribion, Liss C Werner, and Jan Holmström. 2019. Digital twin: vision, benefits, boundaries, and creation for buildings. IEEE Access. 7:147406–147419

[36] Ines Khelifi. Azure digital twins: Powering the next generation of iot connected solutions. Retrieved May 10, 2021 from https://azure.microsoft.com/en-in/blog/azure-digital-twins-powering-the-next-generation-of-iot-connected-solutions/

[37] Sven Kreiss, Lorenzo Bertoni, and Alexandre Alahi. 2019. Pifpaf: Composite fields for human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pages 11977–11986

[38] Helen Leblanc. What are PBR Materials. Retrieved May 10, 2021 from https://info.e-onsoftware.com/learning_vue/what-are-pbr-materials

[39] Wenbo Li, Zhicheng Wang, Binyi Yin, Qixiang Peng, Yuming Du, Tianzi Xiao, Gang Yu, Hongtao Lu, Yichen Wei, and Jian Sun. 2019. Rethinking on multi-stage networks for human pose estimation. arXiv preprint arXiv:1901.00148

[40] Xuan Li, KunfengWang, Yonglin Tian, Lan Yan, Fang Deng, and Fei-Yue Wang. 2018. The paralleleye dataset: A large collection of virtual images for traffic vision research. IEEE Transactions on Intelligent Transportation Systems 20(6):2072–2084

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

[42] Pablo Martinez-Gonzalez, Sergiu Oprea, Alberto Garcia-Garcia, Alvaro Jover-Alvarez, Sergio Orts-Escolano, and Jose Garcia-Rodriguez. 2019. Unrealrox: an extremely photorealistic virtual reality environment for robotics simulations and synthetic data generation. Virtual Reality. pages 1–18

[43] Microsoft. Welcome to airsim. Retrieved May 10, 2021 from https://microsoft.github.io/AirSim/

[44] Microsoft Corporation. Iconics makes smart buildings even smarter with azure digital twins Retrieved May 10, 2021 from https://customers.microsoft.com/en-us/story/iconics-partner-professional-services-azure-iot

[45] Microsoft Corporation. Steelcase demonstrates the smart and connected workplace with new iot-powered solutions. Retrieved May 10, 2021 from https://customers.microsoft.com/en-US/story/steelcase-manufacturing-azureiot-en

[46] Anuj Mohan, Constantine Papageorgiou, and Tomaso Poggio. 2001. Example-based object detection in images by components. IEEE transactions on pattern analysis and machine intelligence. 23(4):349–361

[47] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. 2019. Posefix: Model-agnostic general human pose refinement network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pages 7773–7781

[48] Abhishek Mukhopadhyay, Imon Mukherjee, and Pradipta Biswas. 2019. Comparing cnns for non-conventional traffic participants. In Proceedings of the 11th International Conference on Automotive User Interfaces and Interactive Vehicular Applications: Adjunct Proceedings pages 171–175

[49] Alejandro Newell, Zhiao Huang, and Jia Deng. 2016. Associative embedding: End-to-end learning for joint detection and grouping. arXiv preprint arXiv:1611.05424

[50] Xuecheng Nie, Jiashi Feng, Jianfeng Zhang, and Shuicheng Yan. 2019. Single-stage multi-person pose machines. In Proceedings of the IEEE/CVF International Conference on Computer Vision. pages 6951–6960

[51] Nikolaos Nikolakis, Kosmas Alexopoulos, Evangelos Xanthakis, and George Chryssolouris. 2019. The digital twin implementation for linking the virtual representation of humanbased production tasks to their physical counterpart in the factory-floor. International Journal of Computer Integrated Manufacturing. 32(1):1–12

[52] John Paisley, David Blei, and Michael Jordan. 2012. Variational bayesian inference with stochastic search. arXiv preprint arXiv:1206.6430

[53] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. 2016. We don't need no bounding-boxes: Training object class detectors using only human verification. In Proceedings of the IEEE conference on computer vision and pattern recognition. pages 854–863

[54] Constantine Papageorgiou and Tomaso Poggio. 2000. A trainable system for object detection. International journal of computer vision. 38(1):15–33

[55] George Papandreou, Tyler Zhu, Nori Kanazawa, Alexander Toshev, Jonathan Tompson, Chris Bregler, and Kevin Murphy. 2017. Towards accurate multi-person pose estimation in the wild. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pages 4903–4911

[56] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V Gehler, and Bernt Schiele. 2016. Deepcut: Joint subset

partition and labeling for multi person pose estimation. In Proceedings of the IEEE conference on computer vision and pattern recognition. pages 4929–4937

[57] Narinder Singh Punn, Sanjay Kumar Sonbhadra, and Sonali Agarwal. 2020. Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsort techniques. arXiv preprint arXiv:2005.01385

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

[59] Mohammad Amin Sadeghi and David Forsyth. 2014. 30hz object detection with dpm v5. In European Conference on Computer Vision. pages 65–79. Springer

[60] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. 2020. Textcaps: a dataset for image captioning with reading comprehension. In European Conference on Computer Vision. pages 742–758. Springer

[61] Srinivas Annambhotla, Cesar Romero and Alex Thaman. Synthetic data: Simulating myriad possibilities to train robust machine learning models. Retrieved May 10, 2021 from https://blogs.unity3d.com/2020/05/01/synthetic-data-simulating-myriad-possibilities-to-train-robust-machine-learning-models/

[62] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep high-resolution representation learning for human pose estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pages 5693– 5703

[63] Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. 2018. Digital twin in industry: State-of-the-art. IEEE Transactions on Industrial Informatics. 15(4):2405–2415

[64] Yonglong Tian, Ping Luo, Xiaogang Wang, and Xiaoou Tang. 2015. Pedestrian detection aided by deep learning semantic tasks. In Proceedings of the IEEE conference on computer vision and pattern recognition. pages 5079–5087

[65] Zhi Tian, Hao Chen, and Chunhua Shen. 2019. Directpose: Direct end-to-end multi-person pose estimation. arXiv preprint arXiv:1911.07451

[66] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. 2017. Domain randomization for transferring deep neural networks from simulation to the real world. In 2017 IEEE/RSJ international conference on intelligent robots and systems (IROS). pages 23–30. IEEE

[67] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. 2018. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pages 969–977

[68] Unity Technologies. For all the creators. Retrieved May 10, 2021 from https://unity.com/

[69] Unity Technologies. Navmesh. Retrieved May 10, 2021 from https://docs.unity3d.com/ScriptReference/AI.NavMesh.html

[70] Unity Technologies. Probuilder. Retrieved May 10, 2021 from https://unity3d.com/unity/features/worldbuilding/probuilder

[71] Unity Technologies. Unity path tracing. Retrieved May 10, 2021 from https://docs.unity3d.com/Packages/com.unity.render-pipelines.high-definition@7.1/manual/Ray-Tracing-Path-Tracing.html

[72] Unity Technologies. Unity real-time ray tracing. Retrieved May 10, 2021 from https://unity.com/ray-tracing

[73] Paul Viola, Michael J Jones, and Daniel Snow. 2005. Detecting pedestrians using patterns of motion and appearance. International Journal of Computer Vision. 63(2):153–161

[74] Jian Wang, Xiang Long, Yuan Gao, Errui Ding, and Shilei Wen. 2020. Graph-pcnn: Two stage human pose estimation with graph pose refinement. In European Conference on Computer Vision. pages 492–508. Springer

[75] Xiaoyu Wang, Tony X Han, and Shuicheng Yan. 2009. An hoglbp human detector with partial occlusion handling. In 2009 IEEE 12th international conference on computer vision. pages 32–39. IEEE

[76] Xinlong Wang, Tete Xiao, Yuning Jiang, Shuai Shao, Jian Sun, and Chunhua Shen. 2018. Repulsion loss: Detecting pedestrians in a crowd. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pages 7774–7783

[77] Eric W Weisstein. Circle packing. Retrieved May 10, 2021 from https://mathworld.wolfram.com/

[78] Wikipedia. Navigation mesh. Retrieved May 10, 2021 from https://en.wikipedia.org/wiki/Navigation_mesh

[79] Bo Wu and Ramakant Nevatia. 2005. Detection of multiple, partially occluded humans in a single image by bayesian combination of edgelet part detectors. In Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1. pages 90–97. IEEE

[80] Bin Xiao, Haiping Wu, and Yichen Wei. 2018. Simple baselines for human pose estimation and tracking. In Proceedings of the European conference on computer vision (ECCV). pages 466–481

[81] Dan Xu, Wanli Ouyang, Elisa Ricci, Xiaogang Wang, and Nicu Sebe. 2017. Learning cross-modal deep representations for robust pedestrian detection. In Proceedings of the IEEE conference on computer vision and pattern recognition. pages 5363–5371

[82] You-Cyuan Jhang, Adam Palmar, Bowen Li, Saurav Dhakad, Sanjay Kumar Vishwakarma, Jonathan Hogins, Adam Crespi, Chris Kerr, Sharmila Chockalingam, Cesar Romero, Alex Thaman and Sujoy Ganguly. Training a performant object detection ml model on synthetic data using unity perception tools. Retrieved May 10, 2021 from https://blogs.unity3d.com/2020/09/17/training-a-performant-object-detection-ml-model-on-synthetic-data-using-unity-computer-vision-tools/

[83] Z Yuan, Y Zhang, MV Dragoi, and XT Bai. 2018. Packing circle items in an arbitrary marble slab. In IOP Conference Series: Materials Science and Engineering. volume 399. page 012059. IOP Publishing

[84] Feng Zhang, Xiatian Zhu, Hanbin Dai, Mao Ye, and Ce Zhu. 2020. Distribution-aware coordinate representation for human pose estimation. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pages 7093–710.

[85] Liliang Zhang, Liang Lin, Xiaodan Liang, and Kaiming He. 2016. Is faster r-cnn doing well for pedestrian detection? In European conference on computer vision. pages 443–457. Springer

[86] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. 2006. Fast human detection using a cascade of histograms of oriented gradients. In 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’06). volume 2, pages 1491–1498. IEEE

[87] Lorenzo Bertoni, Sven Kreiss and Alexandre Alahi. 2021. Perceiving Humans: From Monocular 3D Localization to Social Distancing. IEEE Transactions on Intelligent Transportation Systems. DOI: 10.1109/TITS.2021.3069376

[88] Girish Varma, Anbumani Subramanian, Anoop Namboodiri, Manmohan Chandraker, and C. V. Jawahar 2019. IDD: A dataset for exploring problems of autonomous navigation in unconstrained environments. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV) (pp. 1743-1751). IEEE.

[89] Jinson K V. Google will help insurers measure slip and fall risks in buildings. Retrieved July 19, 2021 from https://techiestechguide.com/google-will-help-insurers-measure-slip-and-fall-risks-in-buildings/

[90] Abhishek Mukhopadhyay, Pradipta Biswas, Ayush Agarwal, and Imon Mukherjee. 2019. Performance Comparison of Different CNN models for Indian Road Dataset. In *Proceedings of the 2019 3rd International Conference on Graphics and Signal Processing* (*ICGSP '19*). Association for Computing Machinery, New York, NY, USA, 29–33. DOI:https://doi.org/10.1145/3338472.3338480

[91] Yotaro Fuse, and Masataka Tokumaru. "Navigation Model for a Robot as a Human Group Member to Adapt to Changing Conditions of Personal Space." *Journal of Advanced Computational Intelligence and Intelligent Informatics* 24, no. 5 (2020): 621-629.

[92] Photchara Ratsamee, Yasushi Mae, Kenichi Ohara, Masaru Kojima, and Tatsuo Arai. "Social navigation model based on human intention analysis using face orientation." In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1682-1687. IEEE, 2013.

[93] Woldstad, J. C. (2000). Digital human models for ergonomics.

[94] Mixamo. Adobe. Retrieved August 27, 2021 from https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack

[95] Animation Controller. Unity Manual. Retrieved August 31, 2021 from https://docs.unity3d.com/Manual/class-AnimatorController.html

[96] Milne, G. J., & Xie, S. (2020). The effectiveness of social distancing in mitigating COVID-19 spread: a modelling analysis. MedRxiv.

[97] 8. Vuorinen, V., Aarnio, M., Alava, M., Alopaeus, V., Atanasova, N., Auvinen, M., ... & Hayward, N. (2020). Modelling aerosol transport and virus exposure with numerical simulations in relation to SARS-CoV-2 transmission by inhalation indoors. Safety Science, 104866.

[98] 9. Sharma, S 2020, Social distancing in the workplace: the new norm, Buro Happold, Accessed 7 August 2020, <https://www.burohappold.com/articles/social-distancing-in-the-workplace/#>.

[99] 10. Fort, J., Adam Crespi, Chris Elion, Rambod Kermanizadeh, Priyesh Wani, Danny Lange, “Simulation + Coronavirus”, Unity Technologies White Paper 2020.

[100] 11. Google Sodar n.d., Google Inc., Accessed 7 August 2020, <https://sodar.withgoogle.com>.

[101] 12. Brad Porter, Amazon introduces 'Distance Assistant', The Amazon Blog, Accessed 09 August 2020, <https://blog.aboutamazon.com/operations/amazon-introduces-distance-assistant> , June 23,2020

[102] Hagras, H. (2018). Toward human-understandable, explainable AI. Computer, 51(9), 28-36.

[103] Mukhopadhyay, A., Mukherjee, I., & Biswas, P. (2020, September). Decoding cnn based object classifier using visualization. In 12th International Conference on Automotive User Interfaces and Interactive Vehicular Applications (pp. 50-53).

[104] ProBuilder n.d., Unity Technologies, Accessed 6 August 2020, https://unity3d.com/unity/features/worldbuilding/probuilder.

[105] Murthy LRD, Arjun S, Saluja KPS and Biswas P (2019), Smart Sensor Dashboard, 7th International Conference on PLMSS (Product Life Cycle

Modelling, Simulation and Synthesis) 2019, https://arxiv.org/abs/2005.05025

[106] Zeiler, M. D., & Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[107] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In Proceedings of the IEEE international conference on computer vision (pp. 618-626).

[108] Volk, R., Stengel, J. and Schultmann, F., 2014. Building Information Modeling (BIM) for existing buildings—Literature review and future needs. Automation in construction, 38, pp.109-127.

[109] Kupriyanovsky, V., Klimov, A., Voropaev, Y., Pokusaev, O., Dobrynin, A., Ponkin, I. and Lysogorsky, A., 2020. Digital twins based on the development of BIM technologies, related ontologies, 5G, IoT, and mixed reality for use in infrastructure projects and IFRABIM. International Journal of Open Information Technologies, 8(3), pp.55-74.

[110] Lu Q., Xie X., Heaton J., Parlikad A.K., Schooling J. (2020) From BIM Towards Digital Twin: Strategy and Future Development for Smart Asset Management. In: Borangiu T., Trentesaux D., Leitão P., Giret Boggino A., Botti V. (eds) Service Oriented, Holonic and Multi-agent Manufacturing Systems for Industry of the Future. SOHOMA 2019. Studies in Computational Intelligence, vol 853. Springer, Cham. https://doi.org/10.1007/978-3-030-27477-1_30

[111] Tridify 2020, Tridify Ltd., Accessed 6 August 2020, <https://www.tridify.com>.

[112] PiXYZ Plugin n.d., PiXYZ Software, Accessed 6 August 2020, <https://www.pixyz-software.com/plugin/>.

[113] Unity Reflect n.d., Unity Technologies, Accessed 6 August 2020, <https://unity.com/products/unity-reflect>.

[114] Johnson R., "Method for creating a digital twin of a room," Eur. Patent Application 16186640.5, Mar.7, 2018.

[115] ProGrids n.d., Unity Technologies, Accessed 6 August 2020, https://docs.unity3d.com/Packages/com.unity.progrids@3.0/manual/index.html

[116] TurboSquid 2020, Accessed 6 August 2020, https://www.turbosquid.com/

[117] Sketchfab 2020, Accessed 6 August 2020, https://sketchfab.com/

[118] Woldstad, J. C. (2000). Digital human models for ergonomics.

[119] Mixamo. Adobe. Retrieved August 27, 2021 from https://www.mixamo.com/#/?page=1&type=Motion%2CMotionPack

[120] Paisley, J., Blei, D., & Jordan, M. (2012). "Variational Bayesian inference with stochastic search". In: International Conference on Machine Learning (pp. 1367–1374).

[121] Burda, Y., Grosse, R., & Salakhutdinov, R. (2015). Importance weighted autoencoders. arXiv preprint arXiv:1509.00519.

[122] Chen, R. T., Li, X., Grosse, R., & Duvenaud, D (2018). "Isolating sources of disentanglement in VAEs". In: Proceedings of the 32nd International Conference on Neural Information Processing Systems. Curran Associates Inc. (pp. 2615–2625).

[123] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y. (2014). Generative adversarial nets. Advances in neural information processing systems, Volume 27, (pp. 2672-2680).

[124] Shaham, T.R., Dekel, T. and Michaeli, T., 2019. Singan: Learning a generative model from a single natural image. In Proceedings of the IEEE International Conference on Computer Vision (pp. 4570-4580).

[125] Xian W, Sangkloy P, Agrawal V, Raj A, Lu J, Fang C, Yu F, Hays J. (2018). Texturegan: Controlling deep image synthesis with texture patches. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 8456-8465).

[126] Zhang, K., Zuo, W., Chen, Y., Meng, D., & Zhang, L. (2017). Beyond a gaussian denoiser: Residual learning of deep cnn for image denoising. IEEE Transactions on Image Processing, 26(7), 3142-3155.

[127] Zontak, M., & Irani, M. (2011). Internal statistics of a single natural image. In CVPR 2011 (pp. 977-984). IEEE.

[128] Zontak, M., Mosseri, I., & Irani, M. (2013). Separating signal from noise using patch recurrence across scales. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1195-1202).

[129] Denton , E.L., Chintala, S. and Fergus, R. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In Advances in neural information processing systems (pp. 1486–1494).

[130] Huang, X., Li, Y., Poursaeed, O., Hopcroft, J. and Belongie, S. (2017). Stacked generative adversarial networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 5077-5086).

[131] Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint arXiv:1710.10196.

[132] Barnett, S.A. (2018). Convergence Problems with Generative Adversarial Networks (GANs). Mathematical Institute, University of Oxford.

[133] Park, S. W., Huh, J. H., & Kim, J. C. (2020). BEGAN v3: Avoiding Mode Collapse in GANs Using Variational Inference. Electronics, 9(4), 688.

[134] Kingma DP and Welling M (2014), Auto Encoding variational bayes, Proceedings of International conference on Learning Representations (ICLR 14).

Offices worldwide

The services described in this publication are subject to availability and may be modified from time to time. Services and equipment are provided subject to British Telecommunications plc's respective standard conditions of contract. Nothing in this publication forms any part of any contract.

Registered office: 81 Newgate Street, London EC1A 7AJ.
Registered in England No: 1800000.