

Geometric Change Detection in Digital Twins

Tiril Sundby¹, Julia Maria Graham¹, Adil Rasheed^{1,*}, Mandar Tabib² and Omer San³

¹ Department of Engineering Cybernetics, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology, 7034 Trondheim, Norway; tirils@live.no (T.S.); julia.graham@live.no (J.M.G.)

² Mathematics and Cybernetics, SINTEF Digital, 7031 Trondheim, Norway; mandar.tabib@sintef.no

³ Mechanical and Aerospace Engineering, Oklahoma State University, Oklahoma, OK 74078, USA; osan@okstate.edu

* Correspondence: adil.rasheed@ntnu.no; Tel.: +47-90291771

† Current address: O. S. Bragstads plass 2, 7035 Trondheim, Norway.

Abstract: Digital twins are meant to bridge the gap between real-world physical systems and virtual representations. Both stand-alone and descriptive digital twins incorporate 3D geometric models, which are the physical representations of objects in the digital replica. Digital twin applications are required to rapidly update internal parameters with the evolution of their physical counterpart. Due to an essential need for having high-quality geometric models for accurate physical representations, the storage and bandwidth requirements for storing 3D model information can quickly exceed the available storage and bandwidth capacity. In this work, we demonstrate a novel approach to geometric change detection in a digital twin context. We address the issue through a combined solution of dynamic mode decomposition (DMD) for motion detection, YOLOv5 for object detection, and 3D machine learning for pose estimation. DMD is applied for background subtraction, enabling detection of moving foreground objects in real-time. The video frames containing detected motion are extracted and used as input to the change detection network. The object detection algorithm YOLOv5 is applied to extract the bounding boxes of detected objects in the video frames. Furthermore, we estimate the rotational pose of each object in a 3D pose estimation network. A series of convolutional neural networks (CNNs) conducts feature extraction from images and 3D model shapes. Then, the network outputs the camera orientation's estimated Euler angles concerning the object in the input image. By only storing data associated with a detected change in pose, we minimize necessary storage and bandwidth requirements while still recreating the 3D scene on demand. Our assessment of the new geometric detection framework shows that the proposed methodology could represent a viable tool in emerging digital twin applications.

Keywords: digital twin; dynamic mode decomposition; motion detection; pose estimation; object detection; object classification; machine learning



Citation: Sundby, T.; Graham, J.M.; Rasheed, A.; Tabib, M.; San, O. Geometric Change Detection in Digital Twins. *Digital* **2021**, *1*, 111–129. <https://doi.org/10.3390/digital1020009>

Academic Editor: George A. Papakostas

Received: 13 March 2021

Accepted: 11 April 2021

Published: 15 April 2021

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Copyright: © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

With the recent age of digitalization, the world is quickly embracing new technologies like digital twins, which are seen as an enabler for many autonomous systems. A digital twin [1] is defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making. The capability of digital twins can be ranked on a scale from 0 to 5 (0—standalone, 1—descriptive, 2—diagnostic, 3—predictive, 4—prescriptive, 5—autonomy) as shown in Figure 1. A standalone or descriptive digital twin consists of computer-aided design (CAD) models that are the physical representations of the objects building it [2]. CAD models based on the accuracy requirements in a digital twin can be enormous in size. For a digital twin to achieve a capability level of 5 (i.e., full autonomy), it will require the ability to understand the 3D environment in real-time and to keep track of the changes.

The requirements of frequently updating the digital twin with the evolution of the physical assets lead to a situation wherein the storage and bandwidth requirements for archiving the CAD models as a function of time will quickly exceed the available storage [3] and communication bandwidth capacity [4,5]. Fortunately, major advancements have been made in the fields of motion detection, object detection, classification, and pose estimation, which can address these challenges. A brief description of the methodologies in the context of our workflow is given below.

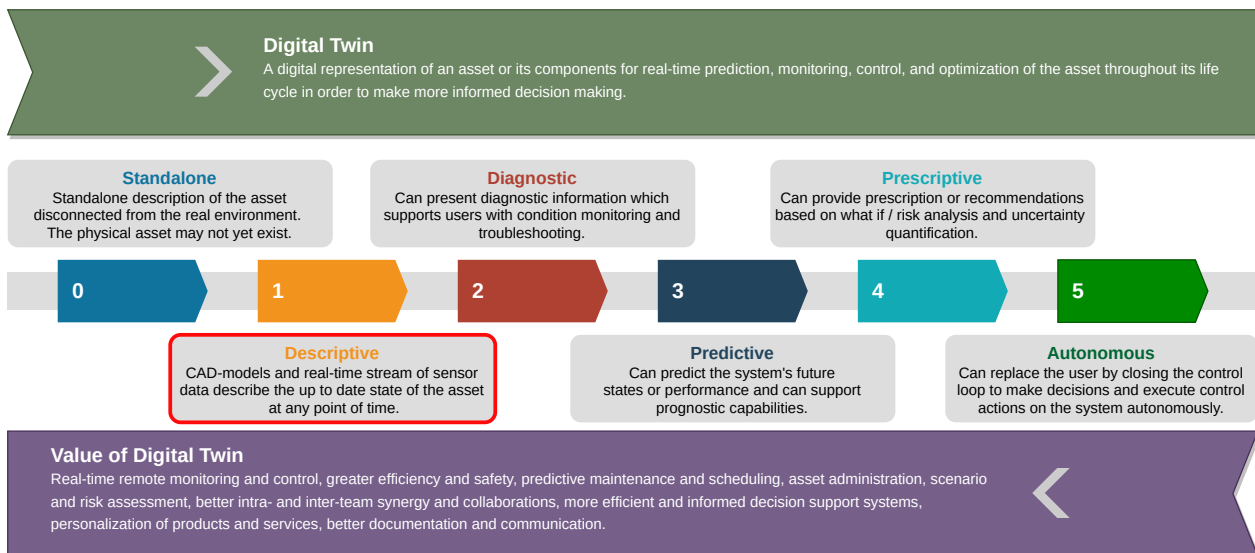


Figure 1. Digital twin capabilities, on a scale from 0 to 5.

Motion detection: Traditional motion detection approaches can be categorized into forms such as background subtraction, frame differencing, temporal differencing, and optical flow [6]. Amongst these, the background subtraction method is the most reliable method that involves initializing a background model first, and then a comparison between each pixel of the current frame with the assumed background model color map providing information on whether the pixel belongs to the background or the foreground. If the difference between colors is less than a chosen threshold then the pixel is considered to belong to the background, otherwise it belongs to the foreground. Typical background detection methods comprise eigen-backgrounds, a mixture of Gaussians, the concurrence of image variations, running Gaussian average, kernel density estimation (KDE), sequential KD approximation, and temporal median filter (overviews of many of these approaches are provided by Bouwmans [7] and Sobral and Vacavant [8]). Many of these methods face challenges with camera jitter, illumination changes, shadows, and dynamic backgrounds, and there is no single method currently available that is capable of handling all the challenges in real-time without suffering performance failures. From the perspective of digital twin set-up, real-time efficiency of motion detection with adequate accuracy matters a lot. In this regard, the authors in [9–11] proposed the use of dynamic mode decomposition (DMD) and its variants (compressing dynamic mode decomposition (CDMD) and multi-resolution DMD (MRDMD)) as novel ways of background subtraction for a motion detection approach. They demonstrated that the DMD worked robustly in real-time using just personal-laptop-class computing power with adequate accuracy. Hence, for our proposed methodology of geometric change detection, we employed DMD as the preferred algorithm in our workflow for motion detection.

Object detection and classification: Conventional object detection involves separate stages comprising (a) object localization to find where the object lies in a frame, which involves steps like informative region selection and feature extraction, and (b) a final object classification stage. Most of the traditional approaches have been computationally inefficient as they use a multi-scale sliding window in region selection and manual non-

automated methods for feature extraction to find the object location in the image. With the advent of deep learning, the object detection process has become more accurate and more efficient as the entire input image is fed to convolutional neural networks (CNNs) to generate relevant feature maps automatically. The feature maps are then used to identify regions and object classification [12,13]. Although possible, this used to be a computationally expensive process. However, with the development of you only look once (YOLO) models [14], which enable prediction of bounding boxes and class probabilities directly from entire images in just one evaluation, such tasks are now possible in real-time. For this reason, the YOLO model was chosen as our preferred approach for object detection and classification in our workflow.

Pose estimation: Traditional methods to estimate the pose of a given 3D shape in an image involve establishing the correspondences between an object image and an object model, and for known object-shapes, these methods can generally be categorized into feature-matching (involving key-point detection) methods and template-matching methods. The feature-matching approach refers to a method of extracting local features or key-points from the image, matching them to the given 3D object model, and then using a perspective-n-point (PnP) algorithm to obtain the 6D (3D translation and 3D rotations) pose based on estimated 2D-to-3D correspondences. These methods perform well on textured objects but usually struggle with poorly-textured objects and low-resolution images. To deal with these types of object, template-matching methods try to match the observed object to a stored template. However, these methods struggle in cluttered environments subjected to partial occlusion or object truncation. More information on such methods can be found in reviews [15,16]. In recent times, CNN-based methods have shown robustness to the challenges faced by the traditional methods, and they have been developed to enable pose estimation from a single image. These methods either treat the pose-estimation problem as classification-based or regression-based, or as a combination of the two, and work by either (a) detecting the pose of an object from an image by 3D localization and rotation estimation [17,18], or (b) detecting key-point locations in images [19–21] (which helped in pose estimation through PnP). However, most deep pose estimation methods are trained for specific object instances or categories (category-specific models). In the context of a digital twin, we should select a methodology that is robust to the environment and can generalize to newer unseen object categories and their orientations. In this direction, a recent category-free deep learning model Xiao et al. [22] estimates the pose of any object based on inputs of its image and its 3D shape (an instance-specific approach), and this model performs the pose-estimation well even if the object is dissimilar to the objects from its training phase. Hence, we employed this pose estimation in our proposed workflow.

Overall, the novelty of this work lies in combining the motion detection (using DMD), object detection and classification (using YOLO), and pose estimation (using deep learning) algorithms to develop a workflow for geometric change detection in the context of descriptive digital twins. This will ensure that instead of tracking and storing the full 3D CAD model evolution over time, we only need to archive bare minimum information (like the changes in pose) whenever a motion is detected. This information can be used to apply transformations on the original state of the CAD models to recreate the state of these models at any point in time. To demonstrate the applicability of the workflow we developed a dedicated experimental setup to test it.

Section 2 gives a detailed description of the proposed workflow, experimental setup, algorithms, software, hardware framework, and the data generation process. The results and discussions are then presented in Section 3. Finally, the main conclusions and potential extension of the work are presented in Section 4.

2. Materials and Methods

The proposed workflow for geometric change detection is presented in Figure 2. We consider a cubical room consisting of 3D objects, each capable of moving with six degrees of freedom. 3D CAD models of these objects are saved at time $t = 0$. An RGB camera mounted

on a Raspberry Pi 4 [23] is pointed towards the collection of objects. When the scene is stationary, the camera does not record anything. However, as soon as the objects start to move, the motion is detected using DMD. The whole sequence of the motion (between t_1 and t_2) is then recorded. When things become stationary again, the whole sequence is deleted after saving the last video frame containing the final change. The last frame is then analyzed using YOLO to detect objects and extract the corresponding bounding boxes, which are then used as inputs to the pose estimation algorithms to estimate the effects of rotation ($\Delta\phi, \Delta\theta, \Delta\psi$). It should be pointed out that our current set-up is not suited for estimating the translation ($\Delta x, \Delta y, \Delta z$) as it would require some more modifications to the setup and algorithms. These are proposed as future work in Section 4. In the following section, we give a brief overview of the implementation of the experimental setup, dataset generated using it, methods of motion detection, object detection, and pose estimation employed in this work, as well as the software and hardware frameworks utilized.

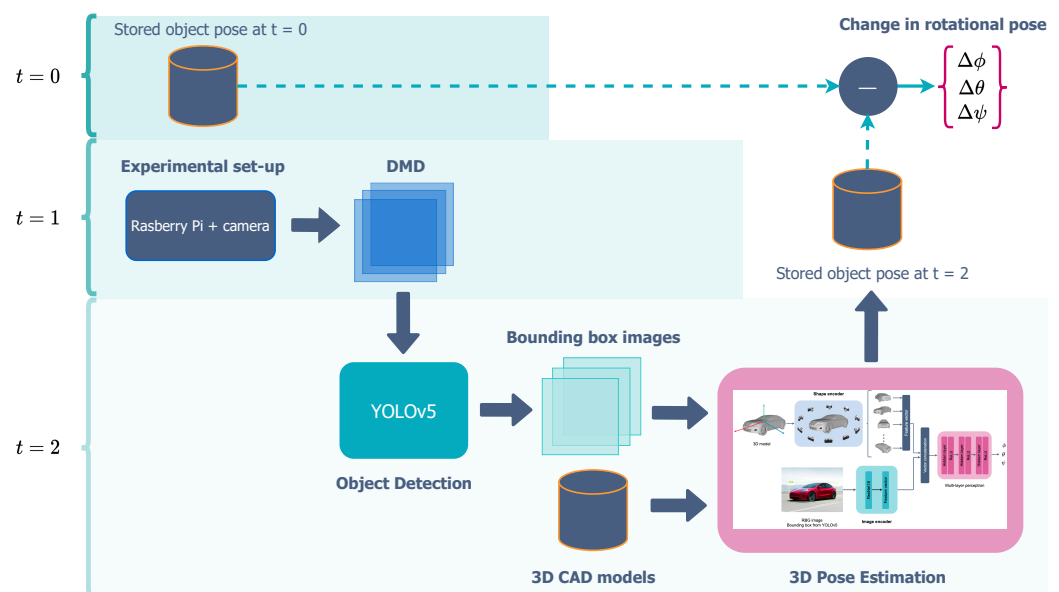


Figure 2. Workflow for geometric change detection.

2.1. Experimental Setup

The experimental setup (shown in Figure 3) developed for this work consists of a square platform with laser-cut slots along which the 3D models can move and rotate. A vertical rod at the center supports two more steel arms capable of rotating along the axis defined by the vertical rod. One of these arms supports a light source while the other supports the Raspberry Pi camera setup. The Raspberry Pi 4 is the fourth generation single-board computer developed by the Raspberry Pi Foundation in the United Kingdom. It is widely used within a range of fields. The Raspberry Pi is designed to run Linux, and it features a processing unit powerful enough to run all the algorithms utilized in this work. The hardware specifications of the Raspberry Pi 4 are presented in Table 1. The Raspberry Pi 4 was connected to a Raspberry Pi HQ camera, version 1.0, and a 6 mm compatible lens during experiments. The specification of the camera is given in Table 2.

Table 1. Specification of Raspberry Pi 4 Model B.

Processor	Broadcom BCM2711, quad-core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5 GHz
Memory	8 GB LPDDR4
GPU	Broadcom VideoCore VI @ 500 MHz
OS	Raspbian Pi OS



Figure 3. The Raspberry Pi 4B and Raspberry Pi camera mounted on the camera tripod.

Table 2. Specification of the 6 mm IR CCTV lens.

Sensor	Sony IMX477
Sensor Resolution	4056 × 3040 pixels
Sensor Image Area	6.287 mm × 4.712 mm
Pixel Size	1.55 μ × 1.55 μ
Focal Length	6 mm
Resolution	3 MegaPixel

Both the arms can be rotated independently and moved up and down to illuminate and record the scene on the platform from different angles. All the scripts for capturing and processing the scene, and trained algorithms, are installed on the Raspberry Pi setup for operational use. Furthermore, the Raspberry Pi is connected to a power source and an ethernet cable to ensure a stable internet connection. An external monitor is also connected for displaying the results.

2.2. Datasets

For the training of all the algorithms used in this work, two datasets (the ObjectNet3D [24] and the Pascal3D [25]) were used. These datasets provide 3D pose and shape annotations for various detection and classification tasks. The ObjectNet3D database consists of 100 object categories, 90,127 images containing 201,888 objects and 44,147 3D shapes. Objects in the images are aligned with the 3D shapes. The alignment provides both accurate 3D pose annotation and the closest 3D shape annotation for each 2D object, making the database ideal for 3D pose and shape recognition of 3D objects from 2D images. Similarly, the Pascal3D database consists of 12 image categories with more than 3000 instances per category.

The 3D CAD models (shown in Figure 4) of the objects used in this work were downloaded from the GrabCAD [26] and Free3D [27] websites in the Standard Tessellation Language (STL) format and then converted to object (OBJ) format. The STL format stores the 3D model information as a collection of unstructured triangulated surfaces each described by the unit normal and vertices (ordered by the right-hand rule) of the triangles using a three-dimensional Cartesian coordinate system. The STL files do not store any information

regarding the color or texture of the surfaces. The OBJ files, which are very similar to STL files, also include this information. In the proposed workflow the STL files were used for 3D printing and the associated OBJ files were used as input to the pose estimation network.

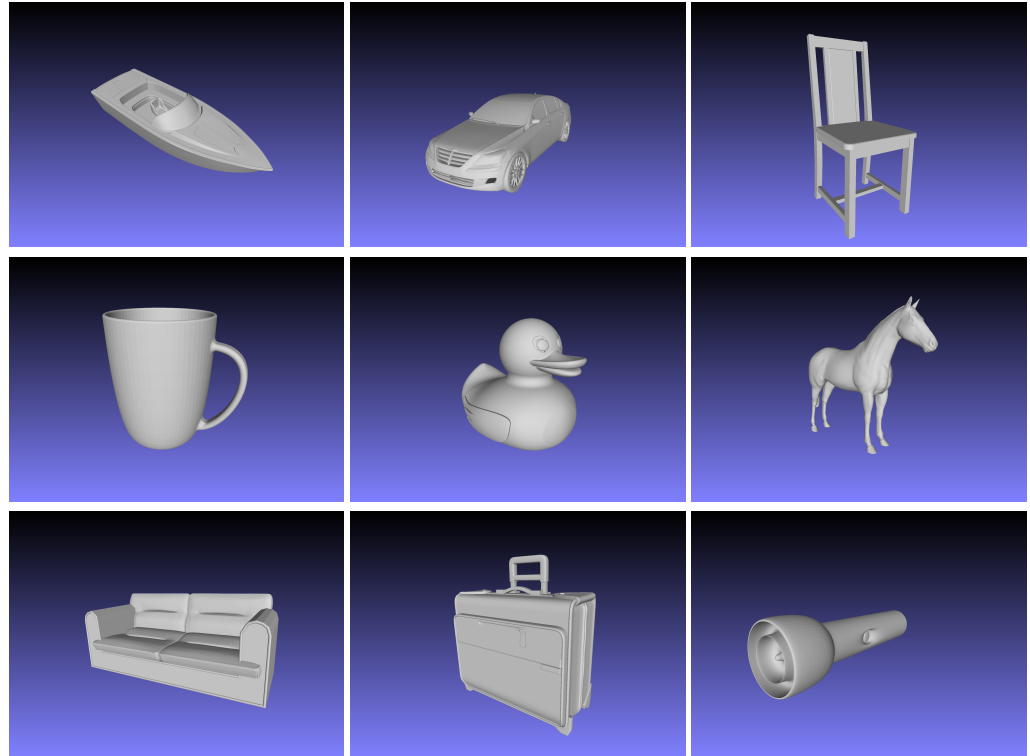


Figure 4. 3D CAD models used for 3D printing and as inputs to the 3D pose estimation network, displayed in MeshLab.

For the testing of the trained algorithms, data were collected in the form of images and videos using the experimental set-up described in Section 2.1. In total, 450 images of the nine object categories presented in Figure 4 were collected. The images consist of single objects seen from different angles, multiple objects seen from different angles, and objects occluded by the other objects. In addition, the images were taken both in natural light and with additional lighting from four different angles using a torch. The image processing is further described in Section 2.4.

2.3. Motion Detection

DMD is an equation-free method that is capable of retrieving intrinsic behavior in data, even when the underlying dynamics are nonlinear [28]. It is a purely data-driven technique, which is proving to be more and more important in the arising and existing age of big data. DMD decomposes time series data into spatiotemporal coherent structures by approximating the dynamics to a linear system that describes how it evolves in time. The linear operator, sometimes also referred to as the Koopman operator [10], which describes the data from one time-step to the next, is defined as follows:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t \quad (1)$$

Consider a set of sampled snapshots from the time series data. Each snapshot is vectored and structured as a column vector with dimensions $n \times 1$ in the following two matrices:

$$\mathbf{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_{m-1}\}, \quad \mathbf{X}' = \{\mathbf{x}_2, \dots, \mathbf{x}_m\} \quad (2)$$

where \mathbf{X}' is the \mathbf{X} -matrix shifted one time-step ahead, each of dimension $n \times (m - 1)$.

Relating each snapshot to the next, Equation (1) can be rewritten more compactly as

$$\mathbf{X}' = \mathbf{A}\mathbf{X} \quad (3)$$

The objective of DMD is to find an estimate of the linear operator \mathbf{A} and obtain its leading eigenvectors and eigenvalues. This will result, respectively, in the modes and frequencies that describe the dynamics.

Computing the leading DMD modes of the linear operator proceeds according to the Algorithm 1:

Algorithm 1 Standard DMD

Steps

- (I) Structure data vectors into matrices \mathbf{X} and \mathbf{X}' as described in Equation (2)
 (II) Compute the SVD of \mathbf{X}

$$\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^* \quad (4)$$

where \mathbf{U} and \mathbf{V} are square unitary matrices of sizes $n \times n$ and $m \times m$ respectively, and $\mathbf{U}\mathbf{U}^* = \mathbf{V}\mathbf{V}^* = \mathbf{I}$.

- (III) To reduce the order of the system, the following matrix is defined

$$\tilde{\mathbf{A}} = \mathbf{U}^*\mathbf{A}\mathbf{U} = \mathbf{U}^*\mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1} \quad (5)$$

- (IV) where $\tilde{\mathbf{A}}$ is projected onto the r leading modes of \mathbf{U} as a result of a truncated SVD. Compute the eigendecomposition of $\tilde{\mathbf{A}}$:

$$\tilde{\mathbf{A}}\mathbf{W} = \mathbf{W}\mathbf{\Lambda} \quad (6)$$

- (V) Which ultimately leads to the DMD *modes* $\mathbf{\Psi}$, where each column of \mathbf{W} represents a single mode of the solution.

$$\mathbf{\Psi} = \mathbf{X}'\mathbf{V}\mathbf{\Sigma}^{-1}\mathbf{W} \quad (7)$$

The initial amplitude of the system is obtained by solving $\mathbf{\Psi}\mathbf{b}_0 = \mathbf{x}_0$. The predicted state is then expressed as a linear combination of the identified modes.

$$\hat{\mathbf{x}} = \mathbf{\Psi}\mathbf{\Lambda}^k\mathbf{b}_0 \quad (8)$$

Note that each DMD mode ψ_j is a vector that contains the spatial information of the decomposition, while each corresponding eigenvalue λ_j^k along the diagonal of $\mathbf{\Lambda}^k$ describes the time evolution. The part of the video frame that changes very slowly in time must therefore have a stationary eigenvalue, i.e., $|\lambda| \approx 1$. This fact can be used to separate the slowly varying background section in a video from the moving foreground elements, hence enabling object localization.

2.4. Object Detection and Classification

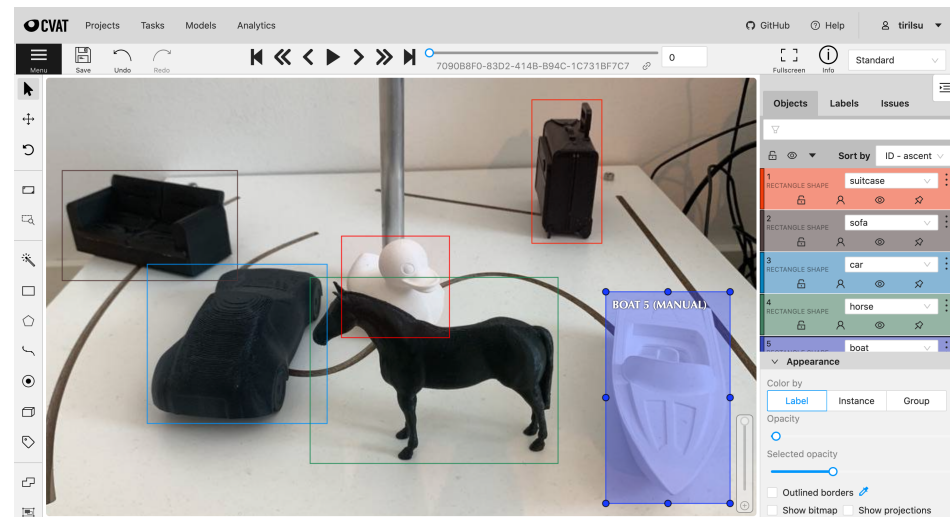
For detecting objects in images, we chose YOLOv5, which is the latest version in the family of YOLO algorithms at the time of writing. YOLOv5 is the first YOLO implementation written in the PyTorch framework, and it is therefore considered to be more lightweight than previous versions while at the same time offering great computational speed. There are no considerable architectural changes in YOLOv5 compared to the previous versions YOLOv3 and YOLOv4. In the current work, YOLOv5 was retrained to recognize the 3D CAD models from our experimental set-up as specified in Section 2.2. We used the Computer Vision Annotation Tool (CVAT) for labeling images. The CVAT is an open-source, web-based image annotation tool produced by Intel. To annotate the images,

we drew bounding boxes around the objects that we wanted our detector to localize, and assigned the correct object categories. The most important hyperparameters for the YOLO algorithm are presented in Table 3.

Table 3. Choice of training parameters for YOLOv5x.

Parameter	Value
Image size	640
Batch size	16
Epochs	300
Device	GPU
Learning rate	10^{-3}

Furthermore, the fully annotated dataset was opened in Roboflow, where techniques of preprocessing and augmentation were applied. Data augmentation involved rotation, cropping, gray-scaling, changing exposure, and adding noise. The final image dataset was split into training, validation, and test sets. The respective ratios corresponded to 70:20:10. The procedures from CVAT and Roboflow are illustrated in Figure 5.



(a)



(b)

Figure 5. Procedure of annotating dataset used for supervised learning in image annotation tools Computer Vision Annotation Tool (CVAT) and Roboflow. (a) Screenshot of the image annotation process in CVAT. (b) Screenshot of the fully annotated dataset in Roboflow.

2.5. Pose Estimation

For pose estimation, we implemented the method presented by Xiao et al. [22]. The essence of the method is to combine image and shape information in the form of 3D models to estimate the pose of a depicted object. The information that relates a depicted object to its 3D shape stems from the feature extraction part of the method. Two separate feature extractions are performed in parallel. One for the images themselves by using a CNN (ResNet-18), and one for the corresponding 3D shapes. The 3D shape features are extracted either by feeding the 3D models as point clouds to the point set embedding network PointNet [29], or by representing the 3D shape through a set of rendered image views that circumvents the object at different orientations, and further passes these images into a CNN.

The outputs of the two feature extraction branches are then concatenated into a global feature vector before initiating the pose estimation part. This part consists of a fully connected multi-layer perceptron (MLP) with three hidden layers of size 800-400-200, where each layer is followed by batch normalization and a ReLU activation. The network outputs the three Euler angles of the camera (azimuth, elevation, and in-plane rotation) concerning the shape reference frame. Each angle is estimated as belonging to a certain angular bin $l \in \{0, L_\theta - 1\}$, for a uniform split of L_θ bins. This is done through a softmax activation in the output layer, which yields the bin probabilities. Along with each predicted angle belonging to an angular bin, an angle offset, $\delta_{\theta,l} \in [-1, 1]$, relative to the center of an angle bin is estimated to obtain an exact predicted angle.

For this report, the 3D model features were extracted by using the method of rendering different views of the 3D shape as mentioned above and using them as input to a CNN. This was done with the help of the Blender module for Python [30]. A total of 216 images were rendered per input object. An illustration of some rendered images that were collected with Blender is depicted in Figure 6.

For the image features, YOLOv5 was first applied to create bounding boxes to localize each object, before being fed to the ResNet-18. The full workflow is illustrated in Figure 7.

The final MLP network was trained on the ObjectNet3D dataset, as this provided better system performance than the network trained on Pascal3D. Furthermore, the network was trained using the Adam optimizer for 300 epochs. The parameters used for training this part of the network are presented in Table 4.

By comparing the real change in rotation of our camera with the estimated rotation angles, we obtained the pose estimation error. The angle output from a 3D pose estimation may be defined using a rotation matrix, eventually describing an object's change in pose.



Figure 6. A selection of rendered images using Blender.

Table 4. Choice of training parameters for the 3D pose estimation network.

Parameter	Value
Batch size	16
Epochs	300
Device	GPU
Learning rate	$10^{-4}/10^{-5}$

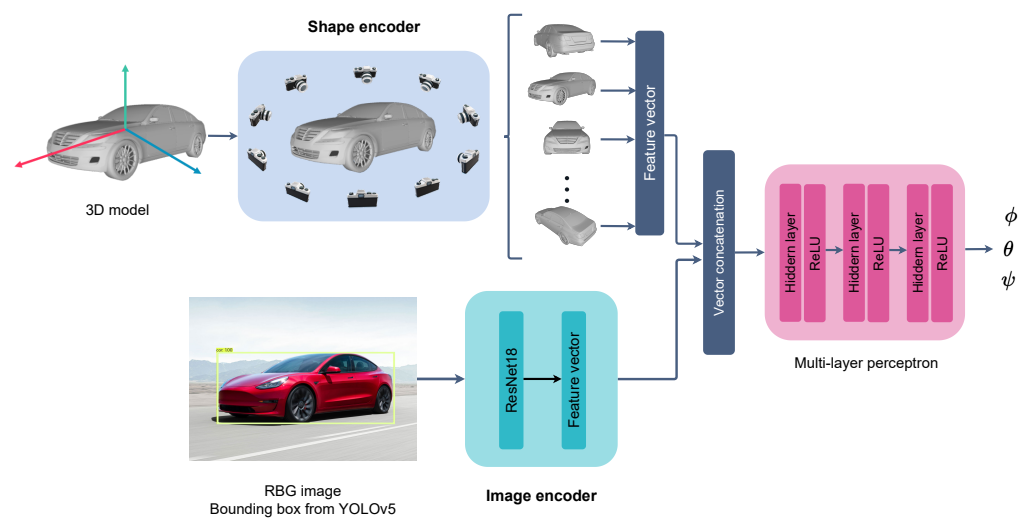


Figure 7. Architecture of the 3D pose estimation network adapted from [22].

2.6. Software and Hardware Frameworks

Both the object detection and 3D pose estimation architectures used in this work were implemented in Python 3.6 using the open-source machine learning library PyTorch [31]. The pose estimation network uses Blender 2.77, an open-source library implemented as a Python module to render multiviews of the 3D CAD models. Furthermore, the MeshLab software was used to visualize 3D objects, and the Python library Matplotlib was used for creating most of the plots and diagrams in the article. The DMD algorithm was also implemented in Python 3.6, enabling real-time processing through the OpenCV (Open Source Computer Vision Library) [32], which is an open-source software library for computer vision and machine learning.

The pose estimation architecture in this project was trained and tested on the NTNU Idun computing cluster [33]. The cluster has more than 70 nodes and 90 GPGPUs. Each node contains two Intel Xeon cores and a minimum of 128 GB of main memory. All the nodes are connected to an Infiniband network. Half of the nodes are equipped with two or more Nvidia Tesla P100 or V100 GPUs. Idun's storage consists of two storage arrays and a Lustre parallel distributed file system. The object detection network applied in this work was trained and tested using Google Colab. Google Colab is an open-source computing platform developed by Google Research. The platform is a supervised Jupyter notebook service where all processing is performed in the cloud with free access to GPU computing resources. Google Colab provides easy implementations and compatibility between different machine learning frameworks. Once all the algorithms were trained they were installed on the Raspberry Pi for operational use.

3. Results and Discussion

This section presents the results obtained from the different steps of our proposed geometric change detection workflow. Video frames where DMD detected the end of motion were used as input to the object detection network. Here, a pre-trained YOLOv5 network detected objects in the images and cropped the images according to their bounding box estimations. The cropped images were then used as inputs to the pose estimation network, together with the respective CAD models of the objects.

We now present the results of motion detection using DMD followed by the results obtained from the object detection using YOLO. Lastly, we present the results from the pose estimation using a single RGB image. Throughout this section, we discuss the applicability of the work to digital twins.

3.1. Motion Detection with DMD Results

The first step in our workflow, as mentioned earlier, involves real-time motion detection and identification of moving objects in the videos recorded under different conditions. For this article six videos were recorded using the experimental setup in an operational mode. Three of these videos were recorded under normal conditions without any disturbances. These are, therefore, referred to as the baseline videos. The other three videos were recorded under the influence of three distinct kinds of disturbances that are considered common challenges in the context of a digital twin. The three challenges whose impact on the motion detection was evaluated are as follows.

- **Dynamic background:** This includes objects moving in the background without being a part of the digital twin.
- **Lighting conditions:** This includes poor lighting condition that fail to illuminate the solid objects resulting in objects in the scene appearing like blobs. The problem is augmented for solid objects of very dark colors. Besides, the camera-lens system takes a few seconds to calibrate the lighting conditions at the beginning of the video sequences. This sudden change in the lighting conditions (exposure, contrast, and brightness) can be mistaken for motion.
- **Monochrome frames:** This includes a situation where the moving object has a similar color and pixel intensity as the background.

The video frames were converted to gray-scale and re-scaled by a factor of 0.25 to reduce the memory and computation requirements. Furthermore, the video frames were flattened and stacked as the column vectors of the matrix X on which the DMD algorithm (as described in Section 2.3) was applied to compute the DMD modes. The computed DMD mode with the lowest oscillation frequency, i.e., smallest changes in time, corresponded to the static background. The static background and the moving foreground (corresponding to moving objects) were separated from each other following the steps earlier explained. The results of background subtraction for three baseline videos from our dataset are presented in Figure 8. The top row shows an example frame from the original video. In the second row, the predicted background frame is based on the computed DMD modes with the lowest oscillation frequency. The third row displays the predicted foreground frame based on the computed DMD modes with high oscillation frequencies. Finally, the fourth row displays the filtered results after extracting the predicted background frame from the original video frame. Looking at the results in Figure 8, the DMD algorithm appears to successfully classify DMD modes based on their associated high or low frequencies. Judging by the results in the fourth row, it can be said that for the baseline videos the DMD could correctly identify the pixels corresponding to motion.

Next, the DMD algorithm was applied to the videos recorded under subtle background movements (a screen displaying moving trees). The results are presented in Figure 9. It can be seen that the disturbances in the background of the scene can also be detected as foreground motion. Fortunately, this situation should not arise in the context of a digital twin as everything in the scene will be considered a part of it and all such motions will be relevant.

Another challenge for DMD-based motion detection is the method's ability to detect moving foreground objects with the same pixel intensity as the background, i.e., monochrome frames. The results corresponding to this situation are presented in Figure 10. Here, we see that it has been difficult to identify the foreground objects (duck and boat) when the pixel intensity of foreground objects is similar to the background. Furthermore, Figure 11 presents the results corresponding to a sudden change in the lighting conditions. The changes in lighting conditions resulted in detections quite similar to those observed in the baseline results. However, such sudden changes in the lighting conditions are a disturbance that can be detected by additional light sensors to detect false motions.

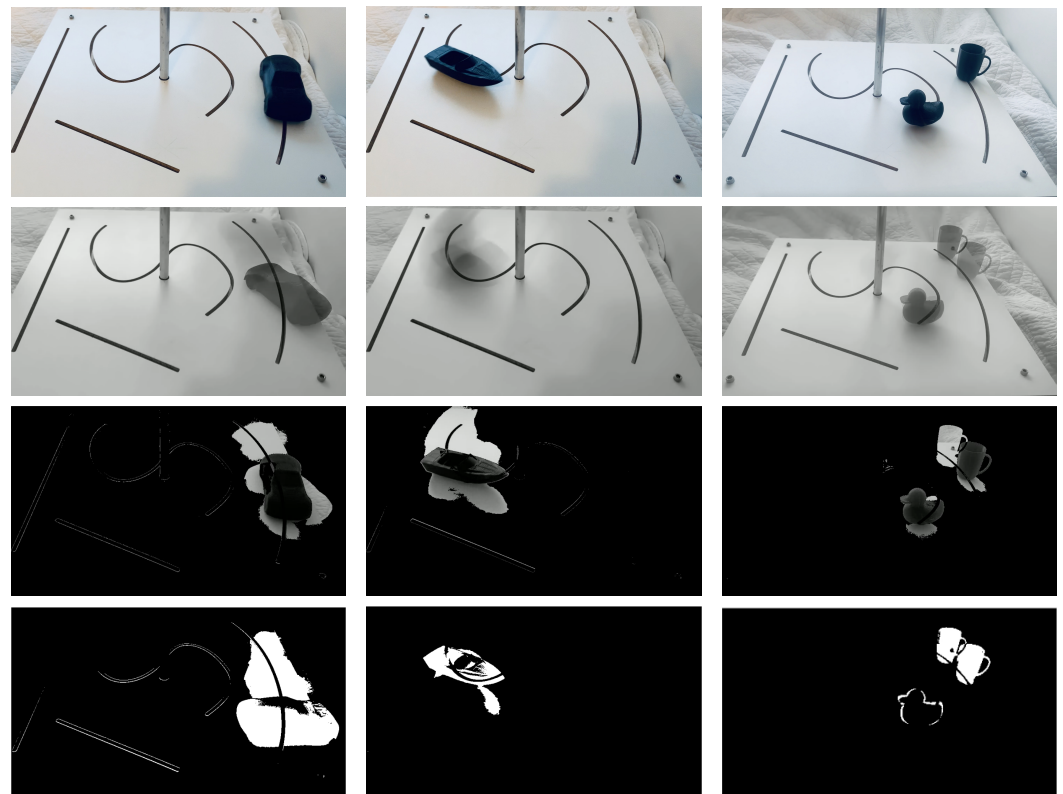


Figure 8. Visual evaluation results for frames from three baseline videos. The top row shows the original baseline video frames. The second and third rows show the predicted static background and the predicted foreground frame, respectively. The fourth row shows the filtered foreground after subtracting the background from the original frame.

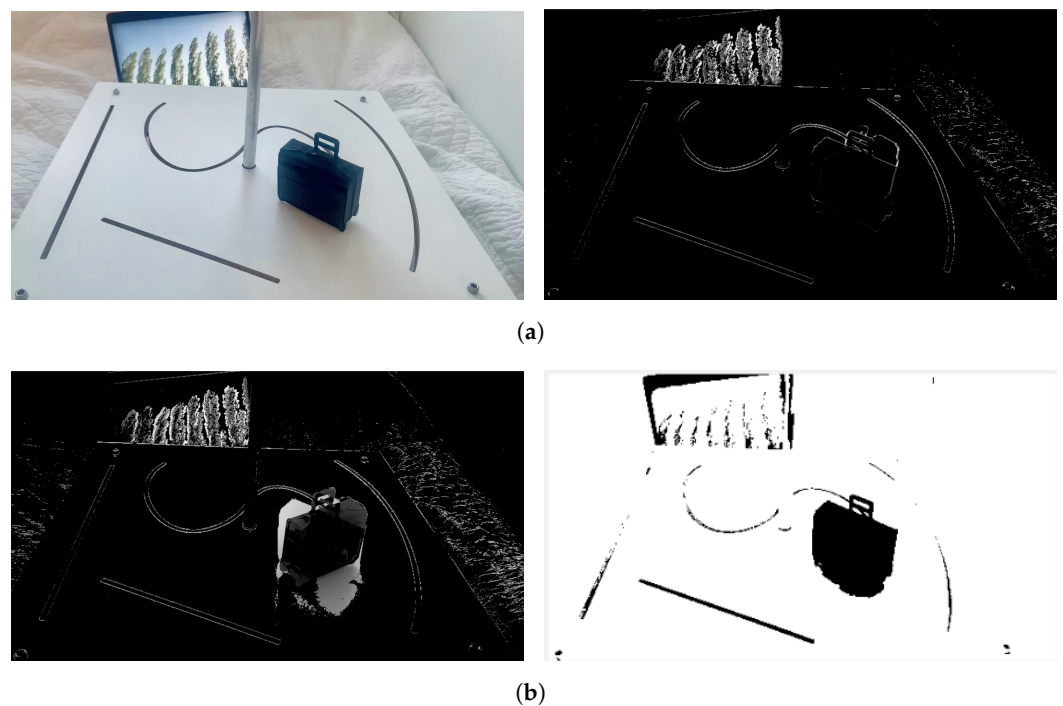


Figure 9. Dynamic mode decomposition (DMD) results for a scene subjected to noise from a dynamic background. (a) Original video frame to the left and the DMD foreground prediction of the dynamic background to the right. (b) DMD foreground prediction of the dynamic background and the moving 3D object to the left, and the filtered foreground to the right.



Figure 10. DMD results on foreground object with same pixel intensity as background object. The original frame is displayed to the left and the filtered foreground is displayed to the right.

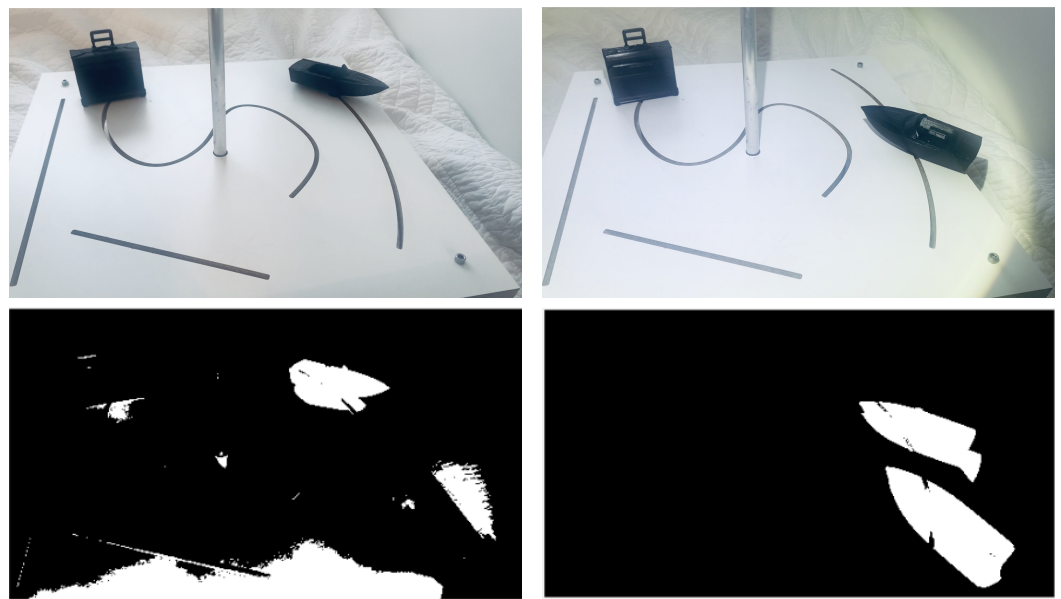


Figure 11. DMD results for a scene subjected to sudden changes in lighting conditions. The first row displays original frames, and the second row displays filtered foreground frames.

The results presented so far display some of the strengths and limitations of the DMD algorithm for detecting motion in the realistic data-set obtained from our experimental setup. In the context of digital twins, it has been able to detect motion well for baseline video frames, and the known challenging scenarios can be mitigated to an extent by proper planning in the operational environment like using a light sensor or making the full scene a part of the DT.

Next, we used object detection on the last frame of the interval over which motion was detected by the DMD algorithm.

3.2. Object Detection Using YOLOv5 Results

The object detection algorithm YOLOv5 was implemented in PyTorch and retrained for 300 epochs on the dataset of 463 images (and their augmented versions) collected from our experimental set-up. The model outputs images containing labeled bounding boxes for all the detected objects in an image. The performance of the retrained network was measured by comparing against the ground truth, i.e., the manually labeled images. A precision–recall curve and training loss curves were used as diagnostic tools for deciding when the model was sufficiently trained.

Figure 12 displays the plotted precision–recall curve for our object detection network on the chosen object categories. The overall mean network performance on all object classes resulted in a high mAP of 0.936. This is a good result marked by the precision–recall curve plotted as a thick blue line. Most object categories barring the boat category obtained high mAP results, resulting in a high average score.

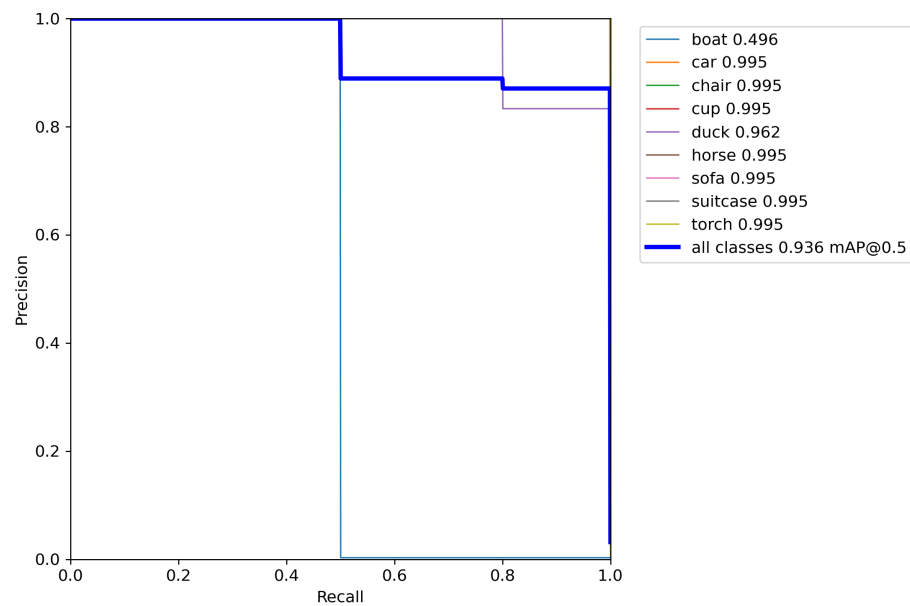


Figure 12. Precision–recall curve for our nine object categories from YOLOv5.

Figure 13 illustrates an example of the detected objects and their bounding boxes. Furthermore, we know, based on several previously published sources, that the YOLO network can be trained to output close to perfect predictions given proper training data and the right configurations. We would like to stress that in the context of digital twins we might already know most of the objects that can feature in it and hence we can always retrain the object detection algorithm with the new objects to achieve even higher performance. In the current work, the purpose of object detection and classification is to extract the bounding boxes including the images to be fed to the 3D pose estimation module, and YOLO seems to achieve this objective satisfactorily.

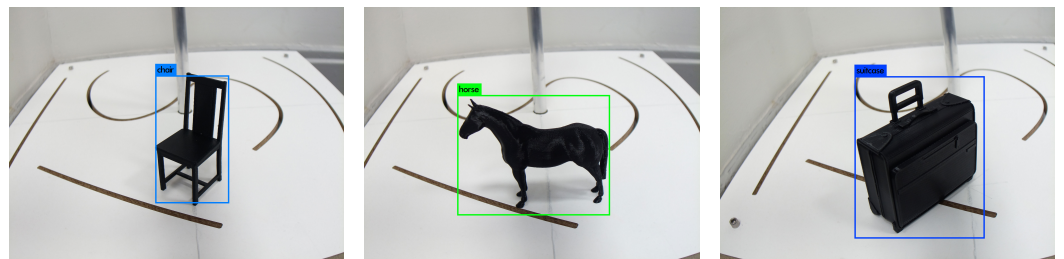


Figure 13. Examples of objects detected by YOLOv5.

3.3. 3D Pose Estimation Results

Following the video frame extraction from our motion detection algorithm, we evaluated the performance of the implemented pose estimation algorithm. The algorithm we used builds upon a network that does not require training on specific categories used for testing [22]. The intention is for the estimation method to work on novel object categories. The method requires a 3D CAD model in OBJ format as input, used for rendering images of each object from different angles, as explained in Section 2.5. It also requires the input of a cropped-out bounding box containing the object used for pose estimation. This was provided by the trained YOLOv5 network. The results are presented as the percentage error, which is defined as

$$\text{Percentage error} = \frac{|\Delta\phi_{real} - \Delta\phi_{pred}|}{\Delta\phi_{real}} \times 100 \quad (9)$$

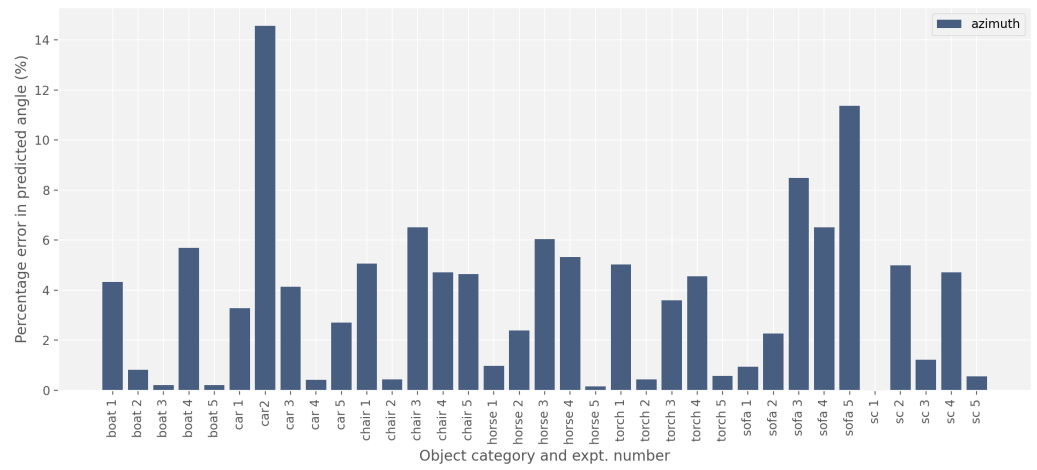
Out of the total eleven 3D printed objects presented in Figure 14, pose estimation for only seven categories are shown in Figure 15. Pose estimations for the two boat models are combined in the results for the boat object category as little difference was observed for the two colors. The remaining three objects (cup, and black and white ducks) were discarded from the testing phase because they proved to be unrecognizable to the pose estimation algorithm. The cup model was discarded due to rotational symmetry, which is a common problem in pose estimation applications [34–36]. The pose estimation algorithm did not predict any meaningful results for the cup model (except for the elevation angle). Furthermore, we discarded the duck model due to a lack of texture. The features of both the white and the black models appeared unrecognizable to the pose estimation algorithm. In Figure 15, we present the result of several rotations along the three axes in 3D. As can be seen, the percentage error in the estimation of any angle is less than 20%. For the in-plane rotation, the estimation is particularly good with an error of less than 10%.



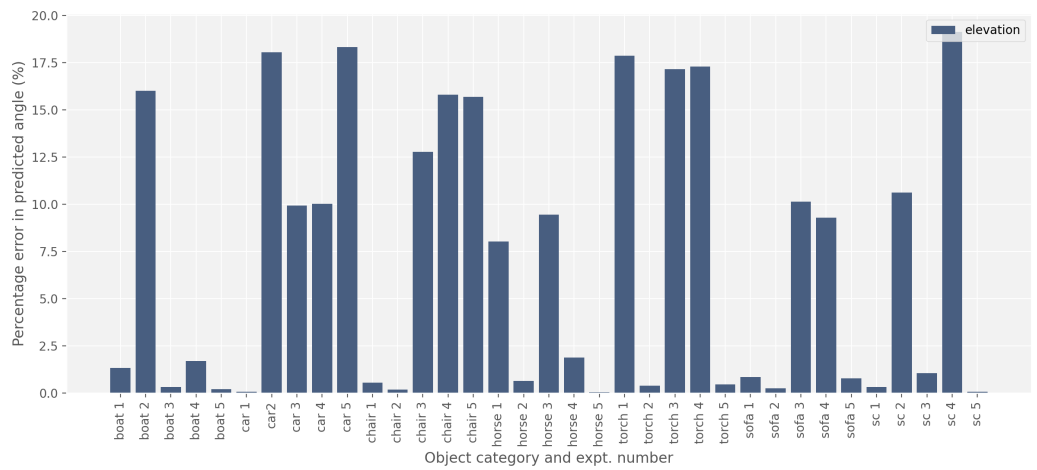
Figure 14. 3D-printed CAD models used in experiments.

Our most prominent observations from the preliminary results presented in Figure 15 are as follows.

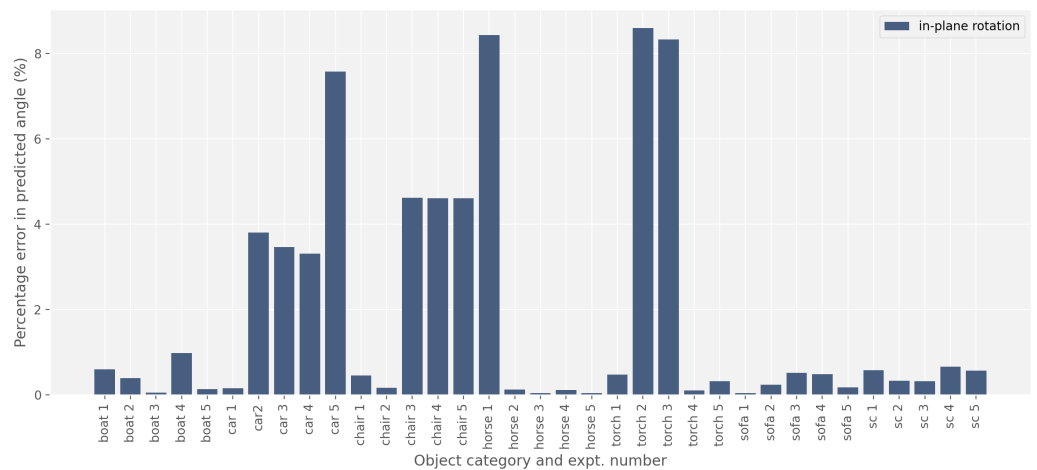
- **Symmetrical objects:** Detecting the pose of partly or fully symmetrical objects is a key challenge in pose estimation. Specific measures have to be applied in order to address this challenge [34,36]. Our 3D cup model is not entirely symmetrical due to the handle; however, from most in-plane rotation, the object appeared to be entirely symmetrical about the rotational axis. In a digital twin context, this will be the most difficult challenge to resolve.
- **Shape similarities from different angles:** The shapes of some objects appear similar even when seen from completely different angles. This typically poses a challenge to the pose estimation network, especially when it appears in combination with poor lighting conditions. For instance, the front and back of the 3D car model may appear very similar in images, making it almost impossible for the pose estimation algorithm to correctly distinguish between the different orientations. This to a certain extent explains the relatively larger error associated with the car and torch. For objects having little symmetry, the estimation was consistently more accurate.
- **Objects with dark light-absorbing surface:** Detecting the pose of such objects is a challenge for the 3D pose estimation network. For the pose estimation method to work, it should be able to extract relevant features. Customizing the lighting conditions to enhance such features can be one way of ensuring that the algorithm works for such objects in a digital twin. This explains the relatively higher error in estimating the elevation and in-plane rotation angles of torch and chair.



(a) Azimuth angle



(b) Elevation angle



(c) In-plane rotation angle

Figure 15. Percentage errors of predicted angles for elevation and in-plane rotation, based on best-fit azimuth predictions. Azimuth errors were calculated according to Equation (9).

4. Conclusions and Future Work

In the current work, we have presented preliminary results from a workflow consisting of motion detection using DMD, object detection and classification using YOLO, and pose estimation using a 3D machine learning algorithm to detect rotational changes of the 3D objects in the context of a digital twin. We demonstrated the potential of the workflow using a custom-made experimental setup. The main conclusions from the work can be enumerated as follows:

1. Although the DMD algorithm struggled to distinguish a static background from the moving foregrounds clearly under challenging conditions (low light and low contrast conditions), it turned out to be a robust technique for detecting motion. This is one of the requirements for our proposed approach to geometric change detection.
2. YOLOv5, when retrained on the images of our models, had an overall mAP of 0.936. Furthermore, the algorithm proved to be very accurate in extracting the bounding box around the detected objects, a prerequisite for the pose-estimator to work in the next step.
3. Changes in all the three angles (azimuth, elevation, and in-plane rotation) were predicted reasonably well with the error percentage well below 20% for all the objects.

We admit that there are several shortcomings associated with the proposed workflow. In the current work, we focused only on the rotational aspect of the object's orientation changes while ignoring the translational element completely. For the geometric change detection to be relevant for a digital twin, we will need information about the full 6-DOF. We can solve the problem by employing a depth camera in the setup and then using simple trigonometry to extract the translational information. Furthermore, we can retrain all the trainable algorithms on the images of objects taken under a much wider variety of lighting, texture, and color conditions to make the entire workflow more robust. We can argue that most of the objects in any digital twin will be a priori known to tailor the whole workflow for at least those objects.

Author Contributions: Conceptualization, A.R.; data curation, T.S.; formal analysis, T.S., J.M.G., A.R., M.T. and O.S.; funding acquisition, T.S. and A.R.; investigation, T.S., J.M.G., A.R., M.T. and O.S.; methodology, T.S., J.M.G. and A.R.; project administration, A.R.; software, T.S. and J.M.G.; supervision, A.R.; validation, T.S.; visualization, T.S.; writing—original draft, T.S., J.M.G., A.R., M.T. and O.S.; writing—review and editing, J.M.G., A.R., M.T. and O.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Research Council of Norway through the EXAIGON project, project number 304843.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: All the 3D CAD models were available from <https://grabcad.com/> (accessed on 13 April 2021) and <https://free3d.com/> (accessed on 13 April 2021). ObjectNet3D [24] dataset can be downloaded from <https://cvgl.stanford.edu/projects/objectnet3d/> (accessed on 13 April 2021), PASCAL3D dataset [25] was downloaded from <https://cvgl.stanford.edu/projects/pascal3d.html> (accessed on 13 April 2021).

Acknowledgments: We would also like to thank Glenn Angell for helping in building the experimental setup. Training of the machine learning algorithms was conducted on the NTNU Idun computing cluster [33] and Google Colab.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

References

1. Rasheed, A.; San, O.; Kvamsdal, T. Digital Twin: Values, Challenges and Enablers From a Modeling Perspective. *IEEE Access* **2020**, *8*, 21980–22012.
2. Zheng, Y.; Yang, S.; Cheng, H. An application framework of digital twin and its case study. *J. Ambient. Intell. Humaniz. Comput.* **2019**, *10*, 1141–1153. [[CrossRef](#)]
3. Philip Chen, C.L.; Zhang, C.Y. Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Inf. Sci.* **2014**, *275*, 314–347. [[CrossRef](#)]
4. Minerva, R.; Lee, G.M.; Crespi, N. Digital Twin in the IoT Context: A Survey on Technical Features, Scenarios, and Architectural Models. *Proc. IEEE* **2020**, *108*, 1785–1824.
5. West, T.D.; Blackburn, M. Is Digital Thread/Digital Twin Affordable? A Systemic Assessment of the Cost of DoD's Latest Manhattan Project. *Procedia Comput. Sci.* **2017**, *114*, 47–56. [[CrossRef](#)]
6. Kulchandani, J.S.; Dangarwala, K.J. Moving object detection: Review of recent research trends. In Proceedings of the International Conference on Pervasive Computing (ICPC), Pune, India, 8–10 January 2015; pp. 1–5. [[CrossRef](#)]
7. Bouwmans, T. Traditional and recent approaches in background modeling for foreground detection: An overview. *Comput. Sci. Rev.* **2014**, *11–12*, 31–66. [[CrossRef](#)]
8. Sobral, A.; Vacavant, A. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Comput. Vis. Image Underst.* **2014**, *122*, 4–21. [[CrossRef](#)]
9. Kutz, J.N.; Fu, X.; Brunton, S.L.; Erichson, N.B. Multi-resolution dynamic mode decomposition for foreground/background separation and object tracking. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 11–18 December 2015; pp. 921–929. [[CrossRef](#)]
10. Kutz, J.N.; Erichson, N.B.; Askham, T.; Pendergrass, S.; Brunton, S.L. Dynamic Mode Decomposition for Background Modeling. In Proceedings of the 16th IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 1862–1870. [[CrossRef](#)]
11. Erichson, N.B.; Brunton, S.L.; Kutz, J.N. Compressed dynamic mode decomposition for background modeling. *J. Real Time Image Process.* **2019**, *16*, 1479–1492. [[CrossRef](#)]
12. Zhao, Z.Q.; Zheng, P.; Xu, S.T.; Wu, X. Object Detection With Deep Learning: A Review. *IEEE Trans. Neural Networks Learn. Syst.* **2019**, *30*, 3212–3232. [[CrossRef](#)]
13. Borji, A.; Cheng, M.M.; Jiang, H.; Li, J. Salient Object Detection: A Benchmark. *IEEE Trans. Image Process.* **2015**, *24*, 5706–5722. [[CrossRef](#)]
14. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [[CrossRef](#)]
15. Sahin, C.; Garcia-Hernando, G.; Sock, J.; Kim, T.K. A Review on Object Pose Recovery: From 3D Bounding Box Detectors to Full 6D Pose Estimators. *arXiv* **2020**, arXiv:2001.10609.
16. He, Z.; Feng, W.; Zhao, X.; Lv, Y. 6D Pose Estimation of Objects: Recent Technologies and Challenges. *Appl. Sci.* **2020**, *11*, 228. [[CrossRef](#)]
17. Kendall, A.; Grimes, M.; Cipolla, R. PoseNet: A Convolutional Network for Real-Time 6-DOF Camera Relocalization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 2938–2946. [[CrossRef](#)]
18. Xiang, Y.; Schmidt, T.; Narayanan, V.; Fox, D. PoseCNN: A Convolutional Neural Network for 6D Object Pose Estimation in Cluttered Scenes. *arXiv* **2018**, arXiv:1711.00199.
19. Oberweger, M.; Rad, M.; Lepetit, V. Making Deep Heatmaps Robust to Partial Occlusions for 3D Object Pose Estimation. *arXiv* **2018**, arXiv:1804.03959.
20. Pavlakos, G.; Zhou, X.; Chan, A.; Derpanis, K.G.; Daniilidis, K. 6-DoF Object Pose from Semantic Keypoints. *arXiv* **2017**, arXiv:1703.04670.
21. Peng, S.; Liu, Y.; Huang, Q.; Bao, H.; Zhou, X. PVNet: Pixel-wise Voting Network for 6DoF Pose Estimation. *arXiv* **2018**, arXiv:1812.11788.
22. Xiao, Y.; Qiu, X.; Langlois, P.A.; Aubry, M.; Marlet, R. Pose from Shape: Deep Pose Estimation for Arbitrary 3D Objects. *arXiv* **2019**, arXiv:1906.05105.
23. Gay, W. *Raspberry Pi Hardware Reference*; Apress: New York, NY, USA, 2014; [[CrossRef](#)]
24. Xiang, Y.; Kim, W.; Chen, W.; Ji, J.; Choy, C.; Su, H.; Mottaghi, R.; Guibas, L.; Savarese, S. ObjectNet3D: A Large Scale Database for 3D Object Recognition. In *Computer Vision—ECCV 2016, Proceedings of the ECCV 2016, Amsterdam, The Netherlands, 11–14 October 2016*; Springer: Cham, Switzerland, 2016; pp. 160–176. [[CrossRef](#)]
25. Xiang, Y.; Mottaghi, R.; Savarese, S. Beyond PASCAL: A benchmark for 3D object detection in the wild. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision, Steamboat Springs, CO, USA, 24–26 March 2014; pp. 75–82. [[CrossRef](#)]
26. GRABCAD. Available online: <https://grabcad.com/> (accessed on 4 March 2021).
27. FREE3D. Available online: <https://free3d.com/> (accessed on 4 March 2021).

28. Tu, J.H.; Rowley, C.W.; Lichtenburg, D.M.; Brunton, S.L.; Kutz, J.N. On dynamic mode decomposition: Theory and applications. *arXiv* **2013**, arXiv:1312.0041.
29. Qi, C.R.; Su, H.; Mo, K.; Guibas, L.J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 652–660.
30. Blender—A 3D Modelling and Rendering Package. Available online: <https://www.blender.org/> (accessed on 13 April 2021).
31. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. *arXiv* **2019**, arXiv:1912.01703.
32. Bradski, G. The openCV library. *Dr. Dobbs's J. Softw. Tools* **2000**, *25*, 120–125.
33. Sjalander, M.; Jahre, M.; Tufte, G.; Reissmann, N. EPIC: An Energy-Efficient, High-Performance GPGPU Computing Research Infrastructure. *arXiv* **2020**, arXiv:1912.05848.
34. Corona, E.; Kundu, K.; Fidler, S. Pose Estimation for Objects with Rotational Symmetry. *arXiv* **2018**, arXiv:1810.05780.
35. Hodan, T.; Barath, D.; Matas, J. EPOS: Estimating 6D Pose of Objects with Symmetries. *arXiv* **2020**, arXiv:2004.00605.
36. Ammirato, P.; Tremblay, J.; Liu, M.; Berg, A.C.; Fox, D. SymGAN: Orientation Estimation without Annotation for Symmetric Objects. In Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV), Snowmass, CO, USA, 1–5 March 2020; pp. 1657–1666. [[CrossRef](#)]