Artificial Intelligence Review (2023) 56:9221–9265
https://doi.org/10.1007/s10462-022-10358-3

Review and analysis of synthetic dataset generation methods and techniques for application in computer vision

Goran Paulin[1] · Marina Ivasic-Kos[2,3]

Published online: 30 January 2023

Abstract
Synthetic datasets, for which we propose the term synthsets, are not a novelty but have become a necessity. Although they have been used in computer vision since 1989, helping to solve the problem of collecting a sufficient amount of annotated data for supervised machine learning, intensive development of methods and techniques for their generation belongs to the last decade. Nowadays, the question shifts from whether you should use synthetic datasets to how you should optimally create them. Motivated by the idea of discovering best practices for building synthetic datasets to represent dynamic environments (such as traffic, crowds, and sports), this study provides an overview of existing synthsets in the computer vision domain. We have analyzed the methods and techniques of synthetic datasets generation: from the first low-res generators to the latest generative adversarial training methods, and from the simple techniques for improving realism by adding global noise to those meant for solving domain and distribution gaps. The analysis extracts nine unique but potentially intertwined methods and reveals the synthsets generation diagram, consisting of 17 individual processes that synthset creators should follow and choose from, depending on the specific requirements of their task.

Keywords Computer vision · Synthetic dataset · Synthset · Generation methods

1 Introduction

Datasets have been one of the drivers of advances in computer vision, natural language processing, and other areas of artificial intelligence that rely on deep learning (Savva et al. 2019). Although there are many publicly available datasets (Fisher 2021), it is often necessary to build a new dataset due to the domain's specificity or to improve the existing

✉ Marina Ivasic-Kos
marinai@uniri.hr

Goran Paulin
gp@kreativni.hr

[1] Kreativni odjel d.o.o., Rijeka, Croatia

[2] Faculty of Informatics and Digital Technologies, University of Rijeka, Rijeka, Croatia

[3] Centre for Artificial Intelligence, University of Rijeka, Rijeka, Croatia

one while bearing in mind that the dataset's quality is not measured solely by its size but by various factors such as the diversity, integrity, and distribution of data (Nowruzi et al. 2019).

Datasets are created by collecting existing data, obtained by direct measurement, or non-existent, such that it needs to be generated beforehand. A set of images intended for supervised learning must be annotated to make ground truth (GT) for model training and evaluation. Annotation can be done at the image level by assigning a class label (e.g. "player" or "ball") or, at the object level (Fig. 1), by marking objects with a bounding box or segments belonging to an individual object. Annotating real images is slow, subjective to human error and, in some areas, such as medicine, an expert process that requires the extreme attention of annotators. All of the above makes creating an annotated dataset time consuming and, therefore, expensive.

A potential solution to this problem is a synthetic dataset (for which we propose the term *synthset*, used hereinafter). Synthsets are not a novelty, they have been used in computer vision since 1989 (Pomerleau 1989), but significant development of methods and techniques for their generation belongs to the last decade.

Synthetic data is defined in (Parker 2003) as data not obtained by direct measurement. The history of its application in machine learning is directly related to computer vision. It dates back to 1989, when synthetic data usage in autonomous driving (Pomerleau 1989) and optical flow analysis (Little and Verri 1989) were first mentioned. Synthetic data solves the problem of collecting a sufficient amount of real data for training the supervised machine learning models and offers the possibility of automatic annotation, which remain the two primary reasons for its application. Synthetic data can be an unlimited source of balanced and varied data. It can simulate rare events or situations we have not yet encountered, including dangerous ones (Aranjuelo et al. 2021). Also, it allows overcoming restrictions on the use of real data conditioned by respect for privacy or other regulations (Rubin 1993). Since in the process of creating synthetic data the generator is intrinsically aware of individual elements of the newly created image, down to the pixel level, it is possible, in addition to RGB image, to directly output different maps such as depth (Pomerleau 1989), optical flow (Little and Verri 1989) and segmentation (Taylor et al. 2007), and additionally, automatically label individual elements in different ways: from 2D (Taylor et al. 2007) and 3D bounding boxes (Mitash et al. 2017) to complete scene metadata (Wrenninge and Unger 2018).

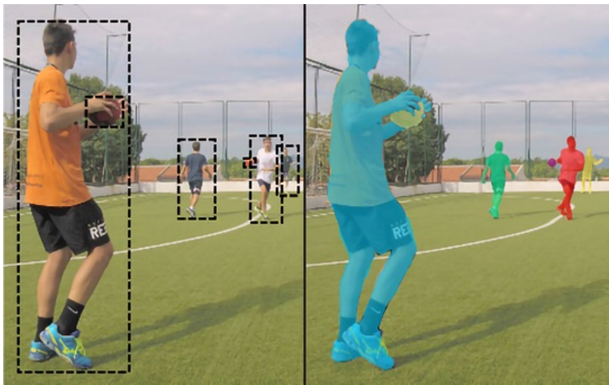

Fig. 1 Marking objects with a bounding box (left) and object segmentation marking pixels belonging to an individual object (right) (Burić et al. 2019)

Although performed outside the computer vision domain, a 2016 study on the effectiveness of synthetic data (Patki et al. 2016) showed that the results achieved using real data in 70% of cases could be reproduced using synthetic data alone. The authors recruited 34 data scientists to measure if synthsets affect the quality of the features generated in terms of predictive accuracy. Scientists were given four different dataset (original, control one, and three synthesized with different noise levels) to extract features from them. Predictive accuracy was computed by training a classifier and evaluating tenfold cross-validation accuracy on the training dataset. By performing the two-sample *t*-test between 3 pairs of evaluated accuracy numbers grouped into four subsets, they found no significant difference between real and synth data for 7 out of 15 comparisons. After examining confidence intervals for the remaining eight tests, they found that for half, the mean of accuracies for features written over synth data was higher compared to those written over the real data. These results showed the tremendous potential of synthetic data but yet need to be achieved in the computer vision domain.

Economically, the cost of creating a single synthetic image is much lower than the cost of creating a real equivalent, as well as the production time (Lange 2020). Once the environment for generating synthetic images and associated annotations has been prepared, the size of the synthset can be significantly increased at a negligible cost per additional image.

According to Tripathi et al. (2019), the generated synthetic data must be *efficient*, *task-aware*, and *realistic*. Efficiency results from a simultaneous reduction in the amount of data to save on the resources needed for training while increasing the samples' diversity (Nowruzi et al. 2019). Synthetic data must be aware of the task and create examples that help improve the target model performance (Prakash et al. 2019; Dvornik et al. 2021). In doing so, they must be visually realistic to minimize the domain gap between real and computer-generated images (which can vary between non-realistic and photorealistic) and thus improve generalization (Rozantsev et al. 2015). If the domain gap within the synthset itself is not minimized, domain adaptation can be performed by mixing synthetic and real datasets in specific percentages (Pishchulin et al. 2011), creating mixed (hybrid) datasets.

Within the field of computer vision, synthetic data, as pairs of images and associated annotations, are applied for different computer vision tasks such as object detection (Cazzato et al. 2020), pose estimation (Munea et al. 2020), scene understanding (Tosi et al. 2020), action recognition (Host et al. 2022), object tracking (Zhang et al. 2021) and object classification (Cai et al. 2021). Synthetic data is predominantly used in the domains such as autonomous driving (Janai et al. 2020) and autonomous flying (Shah et al. 2018), surveillance (Santhosh et al. 2020), and virtual reality (Garbin et al. 2020). Outside the domain of computer vision, the application of synthetic data is important for neural programming and bioinformatics (Nikolenko 2021).

In Fig. 2, examples of different synthsets are shown. They were chosen to demonstrate some of the different methods that can be used for creating synthesized data and different computer vision tasks for which these images were created.

In (Nikolenko 2021), a survey of synthetic data for deep learning is given, mainly focusing on the computer vision domain and discussing the usage of synthetic datasets for basic computer vision problems, synthetic-to-real domain adaptation problem, and privacy-related applications of synthetic data. While his review mentions some of the existing generation methods, it does not go in-depth about techniques that lead to their utilization. This leaves a gap in the literature, motivating us to ask *which synthetic dataset generation methods and techniques exist and what are the best practices for building efficient, task-aware, and realistic synthsets to represent dynamic environments* such as traffic, crowds, and sports. Therefore, this study aims to close that gap by providing an overview of existing

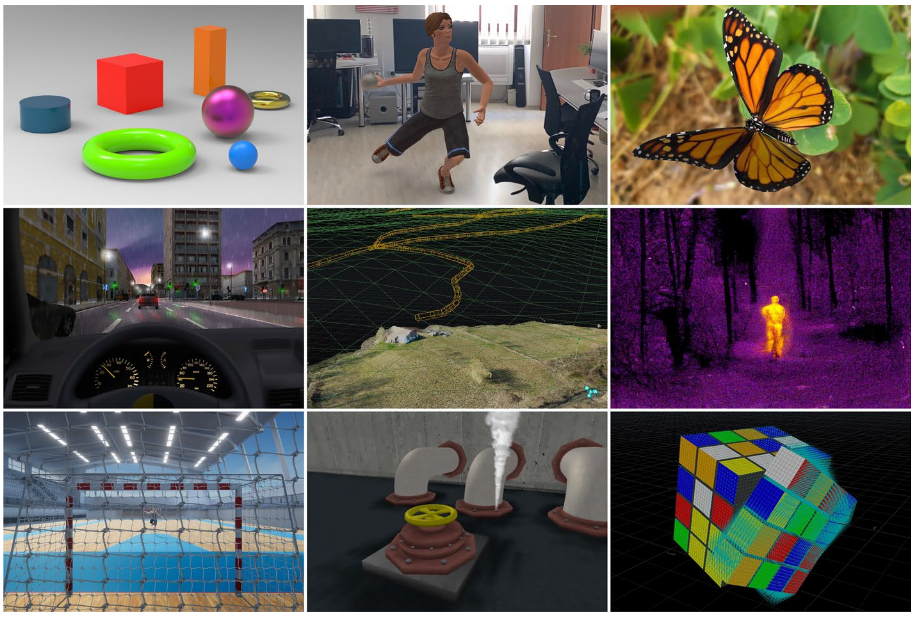

Fig. 2 Examples of selected synthsets generated by different methods for different tasks. For each image, a caption is given in the form: synthset generation method/task. Top row: generator/scene understanding (left), compositing/pose estimation (middle), digital content creation + compositing/segmentation (right). Middle row: commercial computer game/autonomous driving (left), digital content creation/autonomous flying (middle), generative adversarial training/surveillance (right). Bottom row: game engine/action recognition (left), simulation/robot navigation (middle), programming/optical flow (right)

synthsets in the computer vision domain and analyzing the methods and techniques of their generation.

Although generation methods intended for use in dynamic environments should support the ability to create continuous video sequences using animation and simulation techniques, we also included synthsets focused on static environments to extract all the existing generation methods and resulting techniques that are potentially applicable in generation methods suitable for dynamic environments.

The main contributions of this article are:

- Review of the development of methods and techniques for generating synthsets.
- Systematization of synthset generation methods.
- Systematization of synthset generation processes.

The rest of the article is organized as follows: Sect. 2 provides the overview of the field, chronologically presenting methods and techniques for synthset generation, in order of introduction, and discussing the development and their mutual influence. In Sect. 3, systematizations of synthset generation methods and processes are proposed, as well as the synthsets generation diagram. Section 3 also includes the analysis of generation methods usage in computer vision tasks and domains. Section 4 gives the conclusion and suggestions for implementing synthset generation in dynamic environments.

2 Review of synthsets generation methods and techniques

2.1 Research method

Given that our goal was to answer which synthetic dataset generation methods and techniques exist and what are the best practices for building efficient, task-aware, and realistic synthsets to represent dynamic environments, we started research using curated lists of datasets intended for machine learning in the computer vision domain:

- Synthetic for computer vision (https://github.com/unrealcv/synthetic-computer-vision).
- Machine learning datasets (https://www.datasetlist.com).
- CVonline: image databases (http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm).

These lists helped us in the early phase of the research to gain an insight into the ratio of available synthetic and real datasets, their application domains, the content of the individual synthsets, and the papers in which they are described. Unfortunately, they also pointed out the unavailability of a large part of (especially older) datasets that were eventually removed from the Internet.

We analyzed synthsets generation descriptions in these papers, along with the occurrences of keywords that we could use to search the articles in the scientific databases via the Web of Science (Wos) and Scopus platforms, giving preference to IEEE and ACM peer-reviewed articles (including conference papers and preprints). The combination of keywords “synthetic data”, “synthetic dataset”, “synthetic datasets” (in OR mode), and “computer vision” (in AND mode) proved to be the optimal search terms. We limited the search to the period from 1979 to September 2020, when the initial literature search was concluded.

Such a search resulted in a large volume of articles (~2000) published from 1989 to September 2020 and, when repeated in 2022, in ~2800 articles published to the present day (Fig. 3).

Further filtering eliminated most articles that did not contain descriptions of the synthsets generation. Then, by analyzing the available descriptions, we arrived at the final number of 165 works (Fig. 4) that we selected because of their original contributions in using certain methods or synthset generation techniques (treated as “first occurrences”).

These selected papers have introduced a total of nine different synthset generation methods and 345 techniques (Fig. 5). We organized them into 17 processes (Fig. 6 in Subsect. 3.1) that synthset creators should follow and choose from, depending on the specific requirements of their task.

Although we kept periodically searching for new methods and techniques after initial research, there is a noticeable gap after the year 2019 (Figs. 4 and 5) due to using variations and recombining already discovered methods and techniques, implying stagnation of new methods and techniques during the most recent years.

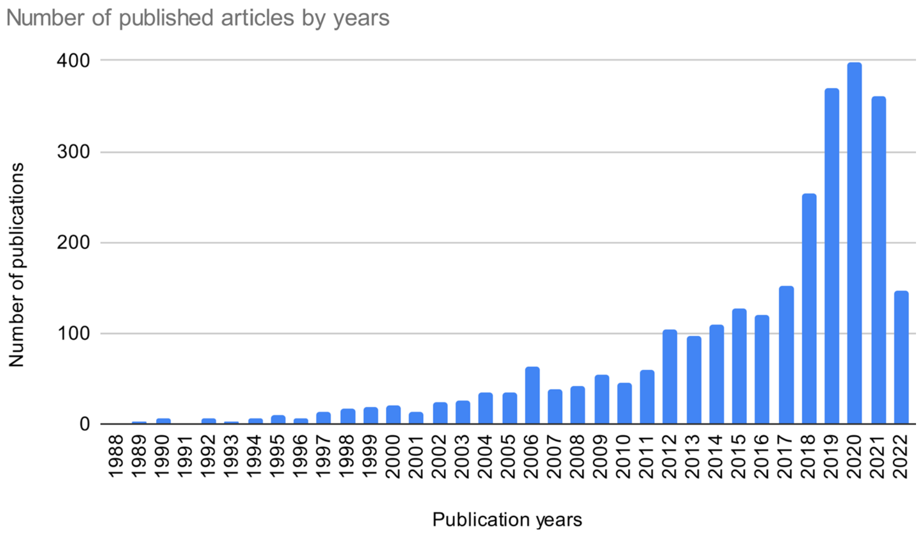

Fig. 3 Histogram of search results (N = 2,787) from Web of Science, using the combination of keywords "synthetic data", "synthetic dataset", "synthetic datasets" (in OR mode), and "computer vision" (in AND mode) as search terms

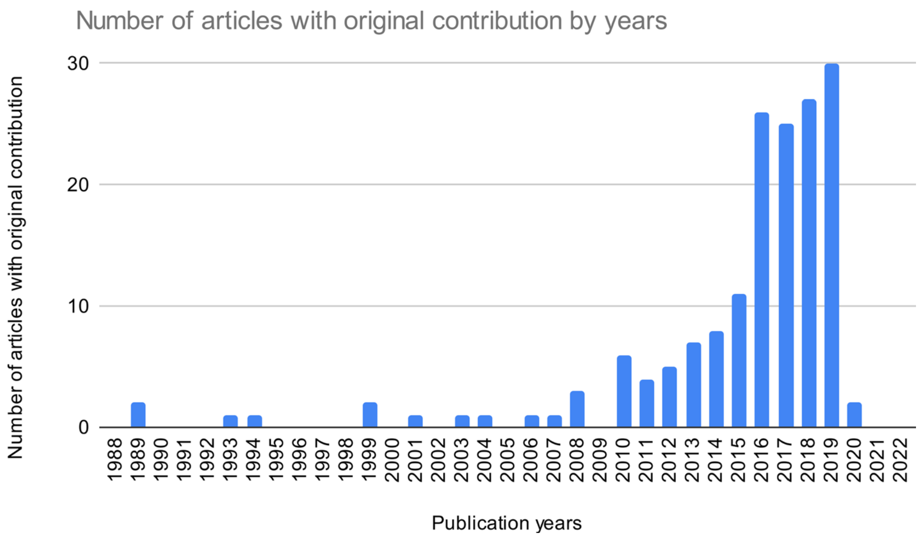

Fig. 4 Histogram of filtered results (N = 165) containing descriptions of the synthsets generation and original contributions in using certain methods or synthset generation techniques (treated as "first occurrences")

The selection of 165 analyzed works includes 85 generated synthsets, 12 simulators, and six publicly available and named generators. Their complete lists are available in the Supplementary material (Online Resource 1) and contain data including the year of origin, reference work, application domain, evaluation method, generation/creation method, the list of used tools and renderers, number of different classes they contain,

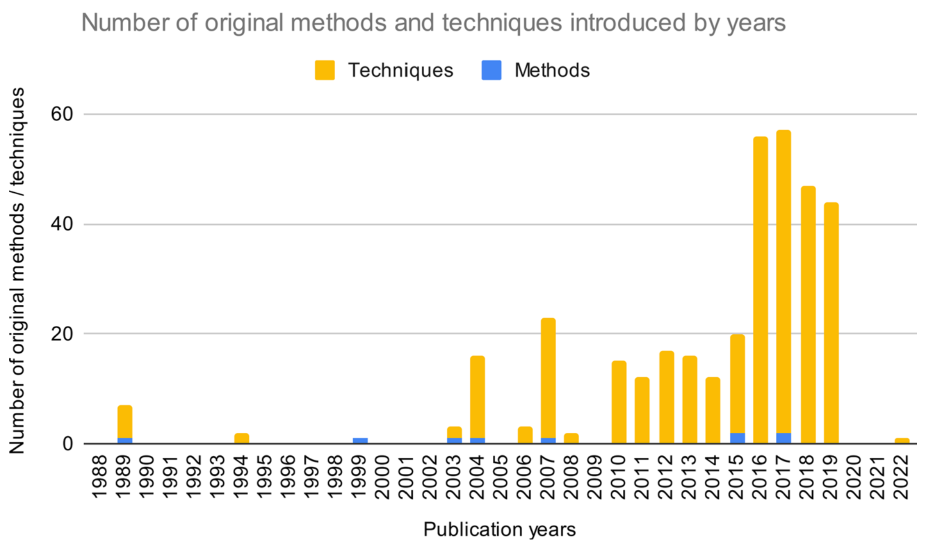

Fig. 5 Histogram of original methods (N = 9) and techniques (N = 345) found in published papers (N = 165), by year of introduction

number of video sequences, number of (annotated) images, image dimensions and the number of frames per second (related to the video sequences).

Papers containing methods and techniques suitable for generating synthsets representing dynamic environments are described in Subsect. 2.2. They are presented and analyzed chronologically (according to the year of publication) to help the reader to better understand the mutual influence and evolution of proposed synthsets generation methods and related techniques.

Data collected from all papers were used to compile a systematization of the synthset generation process. Systematization in condensed form is presented in Table 1 (subsect. 3.2) and is available in the extended form in the supplementary material (Online Resource 1).

2.2 Chronological overview

The first known application of synthetic data for machine learning was reported in the domain of autonomous driving (Pomerleau 1989). It was motivated by the large amount of data required to train the network, which was difficult to collect, especially in different driving conditions and with the possibility of changing the camera orientation. Therefore Pomerleau et al. created a virtual road *generator*. To train the neural network, they used two sets of roads simulated in *different light conditions* and with a realistic degree of *noise*: one set in 30 × 32 px resolution, representing the blue channel of the *RGB* camera, and the other in 8 × 32 px, representing the *depth map* (D) obtained with a laser range finder. After 40 epochs of training, the neural network achieved an accuracy of about 90% on the new simulated road images proving that it can operate the vehicle effectively "under certain

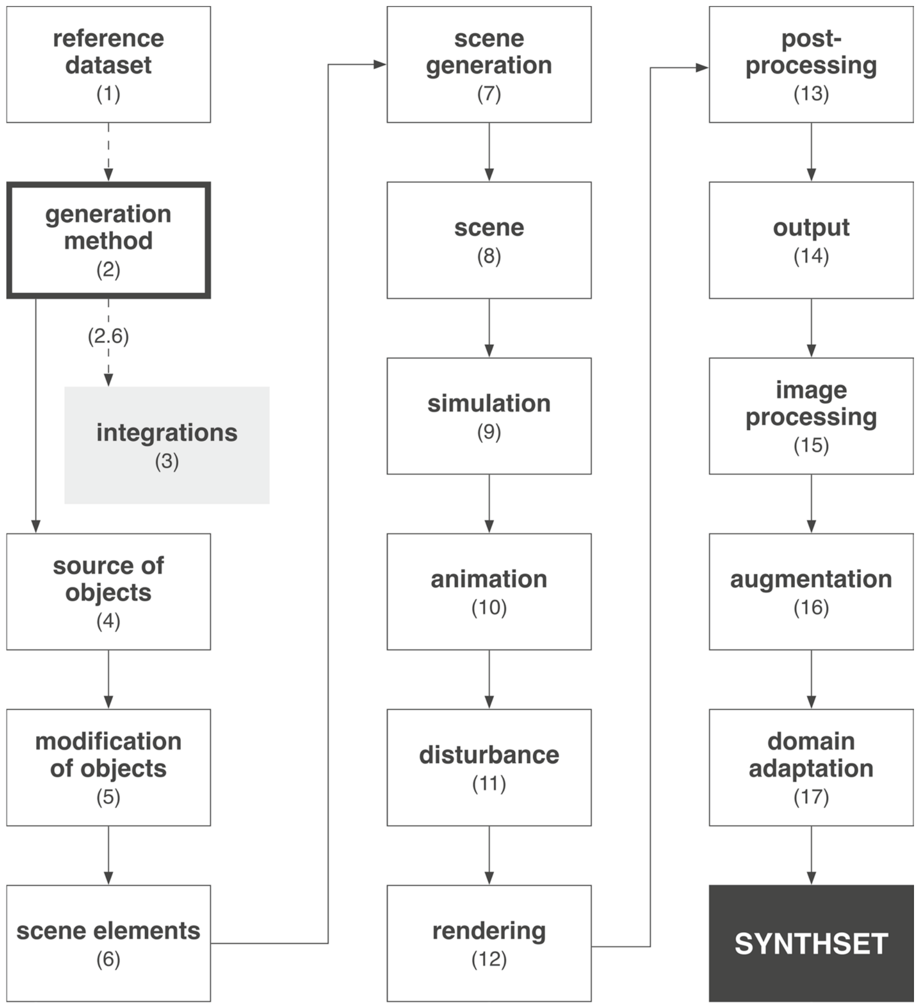

Fig. 6 The synthsets generation diagram based on the analyzed papers (N = 165)

field conditions". When using low-resolution synthesized images, it is difficult to distinguish between real and synthesized roads, which explains the model's high accuracy.

In the same year, the first use of synthetic data in optical flow analysis was mentioned in (Little and Verri 1989). Although authors did not report details of the synthesized images, they pointed out that they allowed them to compare optical streams generated by different algorithms with *true projected velocity fields* in synthetic images, which can be treated as GT.

Inspired by Little and Verri (1989), Barron et al. (1994) generated four simple and one complex synthesized video sequence to evaluate optical flow analysis techniques' performance. The main advantage of synthetic inputs is the possibility of controlling the 2D motion field and scene properties, as well as the possibility of methodical testing during which it is possible to quantify performance. They also introduced post-processing of synthesized data by *pre-smoothing*, using a Gaussian kernel with a standard deviation of 1.5

Table 1 Condensed systematization of the synthset generation process

1. Reference dataset
1.1. Algorithmic estimation of shading parameters
1.2. Distribution of features
2. Generation method
2.1. Generator
2.2. Programming
2.3. Simulator
2.4. Digital content creation
2.5. Commercial computer game
2.6. Game engine
2.7. Compositing
2.8. Physics engine
2.9. Generative adversarial training
3. Integrations
3.1. Unreal Engine 4
3.1.1. Machine learning
3.2. Unity
3.2.1. Reinforcement learning
4. Source of objects
4.1. 2D Object
4.1.1. Previously rendered images
4.1.2. Photographs
4.2. 3D Object
4.2.1. Internet database
4.2.2. Procedurally generated
4.2.3. Manually generated
4.2.4. OSM map conversion
4.2.5. L-System
4.2.6. LiDAR + RGB camera
4.2.7. Photogrammetry
4.2.8. SAR
5. Modification of objects
5.1. 3D Object
5.1.1. Parametric
5.1.2. Manual
5.1.3. Retopologizing
5.1.4. Scaling on different axes
5.1.5. Pre-processing of LiDAR scene
6. Scene elements
6.1. 2D Object
6.2. 3D Object
6.2.1. Structure
6.2.2. Possibility of deformation
6.2.3. Properties
6.3. Light
6.3.1. Light source
6.3.2. Properties
6.4. Camera
6.4.1. Type
6.4.2. Properties
6.4.3. Direction
6.4.4. Viewpoint
6.5. Background
6.5.1. Photo
6.5.2. Uniform color
7. Scene generation
7.1. 2D
7.1.1. Blending
7.2. 3D
7.2.1. Manual
7.2.2. Procedural
7.2.3. Physics simulation
7.2.4. Generative adversarial training
7.2.5. Augmented reality
8. Scene
8.1. 2D Scene
8.2. 3D Scene
9. Simulation
9.1. Physics
9.1.1. Rigid bodies
9.1.2. Physical soil
9.1.3. Deformable objects
9.1.4. Fluid dynamics
9.1.5. Thermodynamics
9.1.6. Obstacles
9.1.7. Magnetism
9.1.8. Soft bodies
9.1.9. Cloth
9.1.10. Slicing
9.1.11. Fracture
9.2. Atmospheric conditions
9.2.1. Wind
9.2.2. Rain
9.2.3. Snow
9.2.4. Clouds
9.3. Crowds

Table 1 (continued)

9.4. Seasonal changes	13.6. Fog
9.5. Perturbations	13.7. Motion blur
9.6. Object deformation	13.8. Focus
10. Animation	13.9. Sun glare
10.1. Motion capture (mo-cap)	13.10. Gamma curve
10.1.1. Source	13.11. Vignette
10.1.2. Type	13.12. Chromatic aberration
10.2. Manual	13.13. Automatic exposure
10.3. Automatic transitions	13.14. Ambient occlusion
10.4. Motion path	13.15. Codec errors
10.5. Procedural	13.16. Lens contamination
10.6. Ragdoll	**14. Output**
11. Disturbance	14.1. Direct image output
11.1. Occlusion	14.1.1. RGB
11.1.1. Position	14.1.2. Depth map
11.1.2. Texture	14.1.3. Optic flow map
11.2. Missing frames	14.1.4. Silhouettes
12. Rendering	14.1.5. Segmentation map
12.1. Type of shading	14.1.6. Object masks
12.1.1. Realistic	14.1.7. Surface normals map
12.1.2. Non-realistic	14.1.8. Motion boundaries map
12.2. Renderer	14.1.9. Specular regions map
12.2.1. Renderers	14.1.10. Transparent regions map
12.2.2. Settings	14.2. Automatic annotation
12.2.3. Fragment shader	14.2.1. 3D Location of object centroid
12.2.4. Rendering path	14.2.2. Projection of object centroid onto image
12.3. Hardware	14.2.3. 2D Bounding box around entire object
12.3.1. CPU	14.2.4. 2D Bounding box around visible part of object
12.3.2. GPU	14.2.5. Camera position
12.3.3. Distributed	14.2.6. Camera orientation
12.4. Output settings	14.2.7. Horizontal field of view
12.4.1. Stereoscopy	14.2.8. Image dimensions
12.4.2. Temporality	14.2.9. Grasp location
12.4.3. File format	14.2.10. Distance from the starting point
12.4.4. Data compression	14.2.11. Path
12.5. Method of delivery to the model	14.2.12. Number of pedestrians
13. Post-processing	14.2.13. Crowd behavior
13.1. Noise	14.2.14. Event log
13.2. Smoothing	14.2.15. Visibility of the object in the fog
13.3. Image distortion	14.2.16. 3D Bounding box around object
13.4. Video ghosting	14.2.17. GPS location
13.5. Antialiasing	

Table 1 (continued)

14.2.18. Compass	15.4. Warping
14.2.19. Speed	**16. Augmentation**
14.2.20. Acceleration	16.1. Occlusion
14.2.21. Detailed data on collisions	16.2. Cropping
14.2.22. Illumination (position, orientation, type, intensity)	16.3. Contrast of the depth map
14.2.23. 2D Projection of 3D facial points	16.4. Brightness of the depth map
14.2.24. 3D Positions of key points	16.5. Color replacement
14.2.25. UV Coordinates of key points	16.6. Shearing
14.2.26. Visibility indicator of each key point	16.7. 2D Rotation
14.2.27. Visual odometry (egomotion)	16.8. 3D Rotation
14.2.28. Scene metadata	16.9. Truncation
14.2.29. Human metadata (sex, age, race, height, and weight)	16.10. Distractor objects
14.2.30. Actions	16.11. Brightness
14.2.31. Intensions	16.12. Contrast
14.2.32. LiDAR point cloud	16.13. Mirroring
14.2.33. Object rotation	16.14. Homography
14.2.34. Oculus Touch data	16.15. Sharpening
14.2.35. LeapMotion data	16.16. Embossing
14.2.36. Data glove data	16.17. Color channel inversion
14.2.37. 3D Joint locations	**17. Domain adaptation**
15. Image processing	17.1. Active learning
15.1. Down-sampling	17.2. Transfer learning
15.2. Cropping	17.3. Style transfer
15.3. Removing the background by segmentation before training	17.4. Adversarial training

px in space and time. This technique reduces temporal aliasing and the quantization effect. They noted that synthetic test data, compared to real ones, give better results with twice less smoothing.

Hamarneh and Gustavsson (2004), using *MATLAB*, developed the first generator of spatially and temporally deformable shapes for medical image segmentation. It synthesizes sequences of 16 *grayscale images* at a resolution of 160 × 182 px, used to detect and segment similar shapes in 2D sequences. They introduced *local noise*, overlapping and touching *occlusions*, and *missing frames* into the synthesized sequences to harmonize them with real examples.

Koenig and Howard (2004) created Gazebo, one of the first *simulators* designed for robotic navigation that allows training with synthetic images generated in *real-time* in the virtual 3D world. Although created primarily as an exterior simulator, its basic weakness is the inability to simulate different *physical soil* models. Also, it does not support *deformable objects* and *fluid dynamics* and *thermodynamics*.

Desurmont et al. (2006) built the first sports synthetic video set designed to track football players. It consists of 13 sequences lasting 400 s each, at a resolution of 1400 × 1050 px. The GT is available for player positions and the positions and parameters of seven static and three *pan-tilt-zoom (PTZ) cameras* that simulate the standard conditions of a television broadcast of a football match. Despite the faithful camera positions, the main weakness of the synthetic set was the lack of any realistic noise and image distortion.

Saxena et al. (2007) generated synthetic images for supervised learning, manually annotating the correct robotic grip's location on five classes of 3D models. They used a *POV-Ray renderer* to synthesize the images and concluded that the grasping algorithm's accuracy improves with the increase in graphical realism. During the generation of 2500 images, they randomized different properties of objects: color, size, and text (on book covers), effectively introducing *domain randomization* (*DR*) method that Tobin et al. (2017) popularized in 2017. They pointed to the time-consuming preparation of 3D objects, concluding that the problem can be solved using 3D objects available on the Internet, with minor modifications.

Taylor et al. (2007) also noted the problem of the resources required to create a virtual world and decided to use the existing world of the *commercial game* Half-Life 2, which allows modifications by the player. Scripted controls support the execution of repeatable scenarios with adjustable parameters such as the positions of multiple synchronized PTZ cameras, lighting (*time of day*, artificial light sources) and *sequences of various events* (actions) in the scene. They built ObjectVideo Virtual Video (OVVV), a system for generating surveillance synthsets, rendering four simultaneous sequences. They increased the realism of the synthetic images by adding *pixel-level noise*, *video ghosting* (by multiplying the image with a delayed copy of the previous signal), and *radial distortion* that simulates the camera lens's imperfection. Unlike real-world imaging, in which *antialiasing* is automatically present due to imperfect optics, synthetic images' aliasing effect is reduced by the Super-Sampling Antialiasing Algorithm (SSAA) algorithm, rendering images at twice the resolution and smoothing blocks of 2×2 pixels while downscaling. When comparing real and synthetic data to evaluate tracking algorithms, the authors achieved similar results (measuring total error).

Ragheb et al. (2008) generated the Virtual Human Action Silhouette (ViHASi) synthset for evaluating Silhouette-Based Human Action Recognition (SBHAR) methods. Using *MotionBuilder*, intended for the film industry, they combined nine 3D models of virtual characters and 20 action classes and rendered them at 640×480 px using up to 40 fixed cameras. In subsequent treatment, in addition to noise, *partial occlusions* are added (pairs of horizontal and vertical lines 0–20 px wide). Testing their own SBHAR method, they found that adding up to 15% of noise reduces recognition ability to 81.73% and adding up to 50% of noise to only 25%. When applying occlusion, the lack of horizontal pixels significantly impacts recognition degradation (recognition drops to 20%) compared to the lack of vertical pixels (recognition drops to 69.70%). The main disadvantage of the ViHASi synthset is the lack of *natural transitions* between individual actions.

The synthset generation method of Taylor et al. (2007) was used by Marín et al. (2010) by *adjusting the camera* movement and rotation to capture only virtual passers-by (and not the road in front of the vehicle). They successfully detected pedestrians, proving that it is possible to learn models on a virtual set for successful detection on real scene shots.

Baker et al. (2011) created a hybrid (a combination of real and synthetic images) Middlebury dataset for evaluation of optical flow analysis methods. 3Delight *Renderman-compatible renderer* allowed them to use *linear color space* and *ambient occlusion* to approximate *global illumination*. GT is created by projecting 3D motion, represented by optical flow vectors, onto a 2D image. The problem with this approach is that each optical flow vector can represent the motion of more than one object, so from the value written in a single pixel, it is not possible to determine which object the motion belongs to. The authors suggested that when creating future datasets for this purpose, *different materials*, lighting changes, *atmospheric effects* and *transparency* should be considered.

Kaneva et al. (2011) created two photorealistic synthsets, Virtual City and Statue of Liberty, to explore the robustness of feature descriptors in changing lighting conditions and scene views. They used a 3ds Max *Mental Ray renderer* with a *Daylight system* to illuminate the scene. It varies 5 times of the day, avoiding night, and the cameras *vary the lens width* from 50 to 200 mm. Comparing their Statue of Liberty set with the real one (photos of the Statue of Liberty), they found distinctly similar performance and ranking of descriptors. The authors used their synthetic dataset to evaluate the performance of five feature descriptors: Scale Invariant Feature Transform (SIFT), Gradient Location and Orientation Histogram (GLOH), DAISY (a dense computed SIFT–like descriptor), Histogram of Oriented Gradients (HOG), and Structural SIMilarity (SSIM). They computed key points at different image locations using spatial local maxima of a Difference of Gaussian (DoG) filter and measured the correct match rate and false-positive rate between key points in pair of images shifted in time (with 2–8 h difference between them), forming a Receiver Operating Characteristic (ROC) curve of descriptor performance. Significant performance degradation is observed when the lighting changes, caused by the movement of shadows and reflections during Sun's journey through the day.

Butler et al. (2012), as well as their predecessors (Baker et al. 2011), warned of the problem of transparency in the creation of synthsets in the domain of optical flow. They used "Sintel", Blender's open-source animated film, to generate their MPI-Sintel synthset. They modified Blender's internal motion blur to give each pixel's exact *motion vector* (to be used as GT). Unlike hybrid set by Baker et al. (2011), MPI-Sintel contains longer sequences, larger motions, *specular reflections*, *motion blur*, *defocus blur*, and *atmospheric effects*. Authors advised caution when using this synthset for training and evaluating algorithms that depend on *physics laws* because animation does not necessarily follow them.

Satkin et al. (2012) suggested using existing 3D model databases to solve the problem of understanding a scene from monocular images. They started from the assumption that human environments are not random but consist of different sizes, shapes, orientations, and positions of objects in certain interrelationships, which can be learned if there is a sufficient amount of data. That is why they created a synthset of 500 images in the categories "bedroom" and "living room". They used *Google SketchUp*, which automatically calibrates the camera by marking a vanishing point and enables filling the scene with ready-made Google 3D Warehouse objects. Since they did not need photorealism, they used *OpenGL* to render objects and, in addition to RGB images, generated the corresponding *object mask* and *surface normals*. They used *voxelization* to automatically group related objects and estimate free space on the stage.

Pepik et al. (2012) considered that the *2D bounding box* is not an optimal representation for detecting objects containing moving parts. So they created a synthset by generating unrealistic, *gradient renderings* of 3D CAD models (cars and bicycles) from which they directly learnt HOG features. Their model automatically learns volumetric parts and can determine the viewpoints and location of individual parts of the object. They compared real, synthetic and mixed datasets for model training and concluded that independently used synthset behaves worst, but in combination with real data, gives the best results in all tested cases.

Haltakov et al. (2013) used the open-source driving simulator VDrift, to create different traffic scenarios, similar to real ones, and to generate a synthset for multi-class image segmentation in the domain of autonomous driving. They used uniform coloring of textures of objects belonging to different classes to render *segmentation annotations*. When rendering such images, they excluded lighting, shadows, reflections, antialiasing and *mipmapping* to get the correct color value for each pixel. Their model achieved good results (89.0% accuracy) using only textured renders. The use of depth or optical flow features gave relatively poor results (80.6 and 75%, respectively). The combination of textures and depth or optical flow features gave the best results (91.2 and 90.1%, respectively), and the combination of all features (texture, depth and optical flow) gave a slightly worse result (91.0%), suggesting that the information encoded in the depth and optical flow features is relatively similar.

Handa et al. (2014) created a benchmark for RGB-D visual odometry, 3D scene reconstruction, and simultaneous localization and mapping (SLAM) in the interior. When creating a synthset consisting of images of the trajectory through two different scenes (living room and office) and targeting photorealism, they paid special attention to light phenomena present in real images: *specular reflections*, *shadows*, and *color bleeding*. The office scene was created entirely *procedurally* but, due to the tool used (POV-Ray), the disadvantage of this is the impossibility of using a polygonal model to evaluate surface reconstruction. In addition to RGB, they also *added noise to the depth map* but omitted motion blur.

Vázquez et al. (2014) continued the work of Marín et al. (2010), noting that, despite the potentially high visual similarity between the synthetic and real sets, even the change in *camera properties* leads to *dataset shift problem*. They developed a V-AYLA integrated framework based on *active learning* to eliminate it. V-AYLA uses 10 times less data from the real set than from the synthset to adapt the domain, achieving as good detection results as when using a manually annotated real set for training and testing.

Courty et al. (2014) built Agoraset, a synthset of realistically rendered virtual people viewed from 64 cameras, which, treated as *particles*, move in realistic dynamic paths through eight different scenes. The set is intended for tracking and segmentation in crowd analysis. Since they use between 200 and 2000 pedestrian avatars, the novelty they introduced is storing each pedestrian's identification tag in a *16-bit grayscale segmentation mask*. They emphasized the need for synthset realism so that what is learned can be transferred to real situations. The disadvantages of Agoraset are the limitation of variations in the geometry through which pedestrians move, the small number of textures used and the time-stretched motion capture (mo-cap) of the walking animation to adjust the speed of each pedestrian, which leads to unrealistic dynamics of movement.

Sun and Saenko (2014) challenged previous conclusions about photorealism's necessity in the domain of object detection. By building two synthsets in parallel, the first with realistic renderings of 3D objects on a *randomly selected 2D background* from the ImageNet dataset, and the second with grayscale renderings on a white background, equally successful detection results were obtained using a fast adaptation approach based on decorrelated features. Their method is successful because it rejects "background statistics", retaining only the shape and texture characteristic for the entire category, but not for an individual object.

Rozantsev et al. (2015) introduced *algorithmic estimation of shading parameters* based on a smaller sample of real images. By combining these parameters, using a simple 3D model of the object they are detecting, laid on a 2D background, they synthesize the desired number of training images. Their key assumption is that the *synthesized images must be as similar as possible* to the real ones, but not necessarily in terms of realism but *in terms of the features* they contain, on which the detection method used relies. They confirmed that the synthset significantly improves the model's performance compared to training exclusively with realistic images but noted a point where too many synthetic images begin to act negatively. According to their experience, this point depends on the detection method, so for AdaBoost they recommend using 100, for DPM 50, and for CNN 15–20 synthetic images for each real one.

Hattori et al. (2015), in the domain of video surveillance, built a massive (2.5 M images) synthset that aims to adjust the surveillance system for a new location using a familiar geometry layout, virtually reconstructed, and 36 different virtual pedestrian models with three possible walk configurations. They paid special attention to the *perspective distortion* of the camera because when using cameras with a wide field of view and a small focal length, it is necessary to learn to recognize people's distorted images.

Veeravasarapu et al. (2015) analyzed the impact of various factors (geometry, appearance, lighting, physics, environment, camera, and rendering parameters) during the construction of virtual scenes on individual features. They concluded that the influence of synthset's photorealism on model performance (for testing on real data) depends primarily on the closeness of distributions between synthetic and real data. To perform the analysis, they built their own synthset in Blender, and since they introduced *rain*, which can significantly affect the appearance of two sequential frames due to the effect of blurring individual parts of the image, they noted the implementation of the rain is not physically correct and does not affect the change of the visual properties of the surfaces in contact with it. Khan et al. (2019) later continued experimenting with rainy conditions and found that adding 3% of rain images improves the segmentation (of rain images) by about 10%. However, training the network with more than 15% of rain images negatively impacts performance, which can be explained by the lower representation of rain images in the test sets. They also found that clouds and puddles reduce network performance more than rain does with its occlusions. Clouds, due to their shape, are often recognized as some other class (e.g., car), and puddles create reflections along the Y-axis and have an extremely negative impact on the segmentation of all classes whose images reflect.

Su et al. (2015) built a large synthset (over 2.4 M images in 12 classes) that they combined with the real set (12 K images) to estimate the viewpoint on 2D images using CNN. Synthset was created by laying rendered ShapeNet 3D models on 2D backgrounds from the SUN397 database, using *alpha blending* to prevent classifiers from learning unrealistic patterns at the junctions of rendered 3D objects and the background.

Peng et al. (2015) noted that most freely available 3D objects often lack *realistic textures*, *appropriate poses*, and backgrounds. To test the effect of these factors on CNN detectors' quality, they built their own synthset. They proved that if CNN learns on synthset tuned for the target task, it will show a high degree of invariance with respect to these factors, but if it was previously trained for classification on, i.e. ImageNet dataset, it learns better when the mentioned factors are explicitly stimulated.

Richardson et al. (2016) used a *morphable 3D facial model* to generate a relatively photorealistic synthset that teaches ResNet-based CNN to reconstruct a textured 3D facial model from a single photograph. Relative photorealism was achieved using the *Phong shading model*, which neglects the skin's physical properties, such as *subsurface scattering* (SSS). Still, the reconstruction's success proved that photorealism is not crucial for the CNN model's quality.

Mueller et al. (2016) built a hybrid UAV123 set as a benchmark for tracking unmanned aerial vehicles (UAVs) in low flight. It consists of 123 sequences and 112,578 images. Of these, eight sequences (in different virtual environments) were synthesized using *Unreal Engine 4 (UE4)* without any post-processing effects. Simulation within UE4, which is used to generate synthetic images, can also be used for the online evaluation of various tracking algorithms.

Movshovitz-Attias et al. (2016) analyzed the impact of photorealism on synthset quality. For this purpose, they built two synthsets of rendered cars: RenderCar (819,000 images of rendered 3D models subsequently laid on randomly selected backgrounds) for training, and RenderScene (1800 images of renderings of 3D models in the associated 3D scene) to validate the CNN detection model. Varying the complexity of *materials* and lighting, they concluded that complex materials and lighting, the unity of the 3D scene (compared to the combination of 3D models and 2D backgrounds) as well as the *quality settings of the renderer*, contribute to the quality of the synthset because they take into account elements of 3D model interaction with the scene such as shadows and reflections. A practical disadvantage of this approach is the significant rendering resources it requires, limiting the size of such synthsets and making them unusable for training. But adding even a small amount of photorealistic synthset to the real one gives better results than using a combination of different real training sets because it can introduce features not present in the real set and thus encourage the model to generalize better. The problem of the *quantitative threshold* indicated by Johnson-Roberson et al. (2017) is proposed to be solved by a larger number of 3D models, but they do not consider the possibility of their procedural creation. They concluded there is no significant difference no matter what form of occlusion is used (from monochrome squares to patches of photographic textures of different shapes) but that the effect of occlusion depends solely on the class of the object. The authors also emphasized the importance of aligning the synthset for training with the *statistics of the angles* at which the objects were recorded in the real set. They introduced a *variable camera shutter speed* and a *vignette* in the synthset production process. Images are initially recorded in *PNG format* but are subsequently compressed in *JPG* because, although compression is not visually noticeable, it has been observed to affect classifier performance. Their CNN uses a weighted SoftMax (wSM) loss function that does not allow the model to randomly "guess" a class but requires that similar views of the 3D object have similar probabilities, thus achieving a more stable prediction.

Wood et al. (2016) generated 1 M synthetic eye images using Unity and a *procedurally animated* morphable eye model for gaze estimation purposes. The eye model was created by *3D scanning* of the eye area in high resolution (5 M points) and reduced to 229 vertices by subsequent *retopologization*. Since ray-tracing was not available in Unity at the time of their work, and the realistic display of the eyeball depends on the use of *refraction* that cannot be achieved by standard rasterization, the authors simulated physically correct refraction using a *fragment shader* (a GPU program that processes each pixel during rasterization). Also, the scene is illuminated with a *high dynamic range (HDR) image lighting*.

Veeravasarapu et al. (2016) continued exploring the impact of photorealism. Their synthset is built using the *stochastic generation* of 3D scenes. During rendering in Blender, utilizing the Monte Carlo method, they varied the *number of samples per pixel* and concluded that training different CNN architectures is invariant to the number of samples after 40 samples per pixel even when the resulting image contains visible noise. Analyzing the network's performance on different parts of the image, they noticed that the most problematic are the boundaries of 3D objects where, compared to real images, the expected effects are missing (*color bleeding*, *penumbra*). Therefore, they suggested modelling the appropriate effects of sensors and lenses, and with this aim, they introduced a *chromatic aberration* effect in post-processing.

Shafaei et al. (2016) built a synthset in the domain of autonomous driving, using, due to legal issues, an unnamed computer game. Driven by the *limitations* imposed by commercial games, they proposed cooperation with game developers to make better use of games' resources in computer vision in the future through a more accessible interface.

Richter et al. (2016) solved a problem pointed out by Shafaei et al. (2016) using a *detouring technique*. They inject a *wrapper* between the operating system and the game, which allows them to record, modify, and reproduce rendering commands. This way, they mark different resources in the rendering process (geometry, textures, shading programs) and track them from frame to frame. Using *rule mining*, which suggests linking the tags thus obtained to the content of the rendered images, and a human annotator who validates them, they are able to annotate at an average rate of 7 s per image. The authors stated that some of the key advantages of using computer games to extract synthsets are the *natural arrangement of objects on scenes*, *realistic textures*, *realistic movement of vehicles and characters*, and the presence of *small objects* that enrich images with details, contributing to the overall realism.

Chen et al. (2016) found that *clothing textures' diversity* is a key ingredient for effective pose estimation, with sufficient data to train the neural network. Using the *morphable model (SCAPE)*, they generated 10,556 different human 3D models. They then collected a large number of images of sportswear on a simple background that aids their segmentation. They segmented 1000 different garments for the upper and lower body from such images, which they used as textures in combination with a previously prepared set of textures for heads, hands, shoes and skin, the colors of which they also randomize. They applied poses from the CMU MoCap dataset to 3D models and rendered them on one of 795 realistic 2D backgrounds. Thus, they generated a total of 5,099,405 images for training and included a selection of 1574 in the Human3D + synthset.

Ros et al. (2016) built SYNTHIA synthset intended for segmentation in the domain of autonomous driving and with it introduced a *simulation of the seasons* that drastically changes the appearance of the images, increasing the realism of exterior scenes. To train a CNN model, they combined realistic and synthetic images using Balanced Gradient Contribution (BGC), which builds batches from both domains in a predefined ratio and thus uses synthetic images for sophisticated regularization. Although the images are rendered at 960 × 720 px, they are used for training at a resolution of 180 × 120 px to save memory and speed up training. The negative consequence of this is the loss of recognizability of smaller objects such as traffic signs and roadside poles. Given the mode of creation and automatic annotation, the main disadvantage of this synthset, which is also pointed out by Tian et al. (2018), is the impossibility of using it for other tasks in computer vision (such as tracking and object detection).

Gaidon et al. (2016) corrected a flaw observed in the work of Ros et al. (2016) by adding GT for object detection and tracking to segmentation while building a Virtual KITTI synthset. They concluded that previous synthsets in the domain of autonomous driving are not detailed enough (from the *layout of objects* to the fact that they do not contain licence plates), so, during the construction of 3D scenes, they adapted the content to the reference video set (KITTI), to minimize the gap between synthetic and real sets. They see the advantage of synthsets in the possibility of analyzing the impact of a single factor (Latin: ceteris paribus) and what-if analysis, especially for rare events. In GT, they introduced the *visibility of an individual object in fog*.

To simultaneously predict the volumetric occupancy of a 3D scene and an object category from a single depth image, Song et al. (2017), using *Planner5D*, built SUNCG, a synthset consisting of 130,269 images based on 45,622 manually designed complex interiors, with realistic furniture layouts. Like Satkin et al. (2012), they used *voxelization*, but they voxelized each object used to compose the scene *separately and only once* to speed up the process. Since their SSCNet model uses only depth information, without color, objects such as windows represent a problem, as well as objects with similar geometry but different function.

Zhang et al. (2017) used modified SUNCG (Song et al. 2017) 3D scenes to build their MLT synthset. They explored the impact of realism on interior scenes by rendering them using a combination of an Open GL renderer and a *physically-based Mitsuba renderer* with a Path Space Metropolis Light Transport (MLT) integrator, with different types of interior and exterior lighting. They concluded that increased realism significantly affects the quality of prediction.

Veeravasarapu et al. (2017) introduced *generative adversarial training* in the building process of 3D scenes intended for photorealistic rendering of synthsets. The 3D scene is treated as a parametric generative model that varies in the direction of real images, using a generative adversarial network (GAN) to change the layout of the objects on the stage and the lighting. The dynamics (movement of vehicles and pedestrians) are neglected, and the intrinsic attributes of 3D objects (shape and texture) are not varied.

Dosovitskiy et al. (2017) used UE4 to develop CARLA, a simulator in the domain of autonomous driving. The authors noted that *generalization to new weather conditions* is easier to achieve than *generalization to new cities*. This can be explained by the fact that the CNN model was trained using images of only one city (the other was reserved for testing) but under different weather conditions (which affected the change in lighting and noise in the images and contributed to greater variability). The same problem is present in other synthsets that prefer 3D models of buildings of *specific architectural styles* (Ros et al. 2016; Kong et al. 2020).

Tobin et al. (2017) were the first to explore *domain randomization* systematically. They concluded that with enough variability in the simulator, the real world could look like just another variation to the model. Using a non-photorealistic renderer built into the *MuJoCo Physics Engine*, they generated hundreds of thousands of images of *randomized simple geometric objects* that vary in shape, scene position, unrealistic textures, lighting, and camera position. The created synthset were used to train the VGG-16 object detection network. They indicated a significant effect of texture when using smaller datasets: the performance of 10,000 images using only 1000 different textures corresponds to the performance of a set of only 1000 images using all available textures. In that sense, when randomizing, the *randomization of textures* is more important than the position of objects.

Varol et al. (2017) built SURREAL, a massive (6.5 M image) synthset intended for pose estimation. To ensure realism, synthetic bodies were created using the *human SMPL 3D model*, whose parameters were adjusted by the MoSh method using 3D data of *mo-cap markers*. They added the clothes texture to such 3D bodies, put them in poses and rendered them over 2D background images. Despite a large number of synthesized images, the authors concluded that, due to the high variability of the synthset, good results could be achieved with as little as 55 K images. They also found that increasing clothing variation positively affects model performance, which speaks in favor of domain randomization (Tobin et al. 2017). For mo-cap, the authors recommended matching the distribution of the target set.

Larumbe et al. (2017) created a parametric head pose simulator and generated the UPNA Synthetic Head Pose Database synthset in it. They introduced a *2D projection of 3D facial points* into the automatic annotation. The authors applied Principal component analysis (PCA), creating an orthonormal basis of 199 principal components of texture and shape, which allowed them to vary the 3D head model's face. PCA, due to its nature (perturbations along the natural variations in the dataset), together with other feature transform methods (Nanni et al. 2021), is more often used for augmenting real datasets (Abayomi-Alli et al. 2020), but it was found that it can cause deterioration of results when augmentation magnitude used to vary colors is significantly increased (Bargoti and Underwood 2017). This should be kept in mind when using it for varying the texture's color.

Zimmermann and Brox (2017) applied 39 actions from the *mo-cap animation library* in the *Mixamo* animation tool to 20 different virtual characters producing a Rendered Hand-pose Dataset, focused on synthetic hand models, meant for training CNNs using RGB data exclusively. When selecting 2D images of cities and landscapes, which they randomly used for the background, they ensured that background images did not contain people because their presence negatively affected the model.

Richter et al. (2017) criticized predecessor techniques for collecting data from commercial games (without the ability to access code): primarily the inability to generate annotations at the speed at which the game is performed (in real-time) and the inability to segment at the instance level. They developed an approach that integrates *dynamic software updating* and *bytecode rewriting*, with which they introduced *visual odometry (ego-movement)* into annotations. They added *snow* to the atmospheric conditions and recorded its negative impact on the optical flow (due to the imposed movement in the foreground) and detection (because it reduces the contrast).

Dwibedi et al. (2017) criticized Georgakis et al. (2017), stating that the step of semantic segmentation does not generalize well in new scenes. Therefore, they proposed a new, simpler approach to ensure only *patch-level realism* for detector training. They cut objects from 2D images (using Big Berkeley Instance Recognition Dataset), employing CNN for segmentation, and pasted them on a random background (from UW Scenes dataset), without considering the size, lighting or composition of the scene. The key to their method is how the pasted objects are blended with the background. They trained the model using three *different blending modes*: no blending, Gaussian Blurring and Poisson Blending. In this way, the model becomes invariant to blending, which increases performance (AP) by 8%.

Tsirikoglou et al. (2017) used *detailed 3D geometry*, *physically based materials*, Monte Carlo based rendering, and faithful simulation of camera optics and sensors to produce one of the most realistic synthsets for autonomous driving applications. Their method combines the procedural generation of a *unique world for each rendered frame* with distributed cloud-based rendering, using an infrastructure designed for the film industry. They optimized the computationally demanding generation of unique worlds by generating only the geometry visible to the camera directly or in reflections and shadows. They emphasized that photorealism is difficult to achieve using insufficiently detailed geometry or physically inaccurate transport of light and identified five key goals for achieving realism: overall scene composition, geometry, lighting, material properties and optical effects. In the experiments, they trained DFCN with only 25,000 synthetic images (in the predecessor range) for semantic segmentation and achieved 36.93%, compared to GTA V (Richter et al. 2016) (31.12%) and SYNTHIA (Ros et al. 2016) (20.7%), concluding that it was worth focusing on maximizing variation and realism.

Unlike McCormac et al. (2017), who randomly sampled scenes from ceneNet and objects from ShapeNet, Jiang et al. (2018) learnt the grammar of the scene from SUNCG (Song et al. 2017) and ShapeNet and described it in *Spatial And-Or Graph (SAOG)*. By sampling SAOG, different scene configurations were created. The authors used *Mantra PBR renderer* and encountered the problem of selecting rendering parameters since lower quality settings (fewer samples per pixel) did not allow them to synthesize images useful enough to surpass state-of-the-art models.

Müller et al. (2018) considered ready-made games to be impractical for use as a synthset's generators due to extremely limited customization options. They preferred modern, fully adaptable game development tools that are characterized not only by photorealism but also by *realistic physics simulation*, significantly reducing the gap between simulated and real worlds. Using UE4, they created their own simulator (Sim4CV) intended for autonomous driving and flying. The simulator contains a large selection of PBR (Physically-Based Rendering) textured high-poly 3D objects used as building blocks and generates frames at a resolution of 320 × 180 px to reduce latency to a minimum, which is essential for end-to-end training.

Tremblay et al. (2018a) continued work of Tobin et al. (2017) research on the potential of domain randomization (DR) as a simpler, cheaper alternative to generating extremely photorealistic synthsets. They built their DR synthset by *randomly placing* an arbitrary number of 3D objects of interest (cars, in their case) on a random 2D background and introduced a new component, *flying distractors*, in the form of various simple geometric bodies. All objects on the scene (both those of interest and distractors) are textured using *random textures*, which is a key novelty.

After the unrealistic DR synthset (Tremblay et al. 2018a), Tremblay et al. (2018b) created the realistic Falling Things (FAT) synthset, using the method of *physically simulating dropping 3D objects onto a scene* presented in the work of McCormac et al. (2017). FAT is intended for the detection of objects in the household, and the authors used the measured statistics to quantitatively confirm the optimal distribution of variability achieved thanks to the applied method. Tremblay et al. (2018c) used FAT for pose estimation for semantic robotic grasping of household objects and showed that a combination of photorealistic images and domain randomization could achieve sufficient dataset variability to use such a trained model in a real environment without additional tuning. Measuring the effect of dataset size, they found that a synthset based solely on domain randomization achieved the best result when using 300 K images (66.64 AUC), an exclusively photorealistic set with 600 K images (62.94 AUC), and when a combination is used in such a way that *no set is represented by less than 40%*, the best result (77.00 AUC) is achieved already with 120 K images.

Texture, highlights, and shading are some of the visual signs that allow people to understand the material from which an object in the image is built. Deschaintre et al. (2018) looked for a way to extract four image maps (diffuse albedo, specular albedo, specular roughness, and normals) from the image, using a neural network, and then, using the *Cook-Torrance BRDF shading model*, recreate object's material during rendering. For this purpose, they built Synthetic SVBRDFs And Renderings synthset, varying 800 *spatially-varying bi-directional reflectance distribution functions (SVBRDF)* in Allegorithmic Substance *procedural materials* created by graphic artists from the film and video industry. Although it achieves good results, this method's disadvantage is the use of input images exclusively from the frontal view, which does not show the behavior of reflections at grazing angles, so the *Fresnel effect's reconstruction* cannot be learned.

Wrenninge and Unger (2018) followed Tsirikoglou et al. (2017) and generated an equally photorealistic synthset, Synscapes. They introduced *scene metadata* into annotations, which describe all the scenes' properties for each generated image, and used the *OpenEXR* format to store the images. They found that motion blur (as a consequence of the observer's speed) and time of day (Sun's height) are the parameters that most affect the network's predictive performance. Motion blur is particularly interesting because it increases with the speed of the vehicle on which the camera is located and blurs the features in the image that remain recognizable to humans but seem significantly different to

CNN compared to the previously learned. Sun approaching the horizon reduces the contrast in the image, but the image is not necessarily darker due to the *auto-exposure* used. The strong shadows disappear, without which it becomes more difficult to distinguish the learned features. The authors emphasized that there is no indication that neural networks can undo domain shifting on their own, and therefore realism should be built into synthsets when generating them.

Mayer et al. (2018) analyzed different ways of constructing synthsets in the domains of optical flow and stereo disparity: *procedural scene randomization* and *manual geometry modelling*. A combination of *manual scene compositing* and manual geometry modelling was excluded as an option because it results in less data generated using the same effort. However, they did not consider the possibility of both *procedural modelling and scene compositing* that can reconcile both approaches. They found that *training using multiple datasets* is most effective when *conducted in stages*, using datasets separately and subsequentially. Authors also found that in some domains, such as optical flow, realism is not necessary—forcing realism with complex scene lighting will not necessarily help even when test data is realistically illuminated. They also emphasized the importance of the moment the data is presented to the neural network: the early training phase prefers simpler data and the latter more complex.

The use of *Structured Domain Randomization (SDR)* instead of DR is supported by the results of Dvornik et al. (2018). The authors conducted object detection experiments by placing objects cut from 2D images at different positions on 2D backgrounds. They concluded that detection accuracy significantly decreases when objects are located at *unrealistic positions*, meaning that the *visual context becomes a crucial information* source whenever visual information is corrupted, ambiguous, or incomplete. The same authors provided an in-depth discussion (2021) about the importance of context when blending target objects into random backgrounds to generate more realistic relationships in synthetic images.

Prakash et al. (2019) presented *Structured Domain Randomization (SDR)*, which considers the structure and context of the scene. Unlike DR (Tremblay et al. 2018a), which places objects and distractors randomly according to a uniform probability distribution, SDR does so according to the probability distribution arising from a specific problem. In this way, SDR allows the neural network to consider the area around the object, as a context, during detection. In the author's formulation, SDR includes three components: global parameters, one or more context-representing curves, and objects arranged along those curves. The SDR approach balances between extreme photorealism and non-realism, characteristic for DR, producing quite realistic images but primarily containing great diversity. Unlike DR, where performance is saturated around 50 K images (Tobin et al. 2017), SDR achieves saturation with 10 K images. That is why SDR can be used to initialize the network when there is not enough annotated real data. Tremblay et al. (2018a) showed that lighting is the most important parameter for DR. However, when using SDR these are *context*, *saturation,* and *contrast*, with saturation indicating the importance of matching textures between the two domains.

Li et al. (2019) started from the assumption that the complexity and diversity of the real world cannot be realistically replicated in a virtual environment. They advanced the idea of laying 3D vehicle models on background photos from Abu Alhaija et al. (2018) using real road videos instead of photos, combined with *LiDAR data* to produce accurate depth maps. This approach allowed them not only to fit 3D objects into any plane but also to generate a *realistic flow of vehicle traffic and the dynamics of pedestrian movement*. As part of the pre-processing, they removed moving objects, closed the holes, replicated the illumination, and improved the textures. The resulting images produced by the *PBRT renderer* are photorealistic and form their AADS synthset. The problems of this approach, compared to the use of fully virtual worlds, are the *limited field of view* of LiDAR due to which, although there is a 3D scene environment, the point of view on synthesized images can not deviate significantly from the original, and the inability to vary lighting and weather conditions.

In the domain of visual reasoning applied to video, Girdhar and Ramanan (2019) built a CATER synthset designed to test the ability to recognize compositions of object movements that require long-term reasoning. The authors concluded that for higher-level tasks, such as action recognition, current neural network architectures are usable with a sufficiently large training dataset, but that mid-level tasks, such as object tracking, present a major challenge in the case of *long-term occlusions*.

Wang et al. (2019c) built the IRS synthset to evaluate stereo disparities in interior scenes. They paid special attention to shaping light effects (*change of brightness*, *reflection and transmission of light*, *lens reflection*), considering them important for applying the learned model in a real environment. In doing so, they used a *deferred rendering path* optimized for 3D scenes with a large number of lights that require a high level of lighting fidelity.

Wang et al. (2019b) used GTA V's realism to create a GCC synthset designed for crowd counting. Their predecessors (Richter et al. 2016, 2017; Johnson-Roberson et al. 2017) did this without interfering with the game, but since GTA V does not contain the necessary *crowd* scenes, the authors were forced to create their own scenes using an add-on based on the Script Hook V library. The additional problem is that GTA V does not support more than 256 people on the scene, which is why they separated the target areas into multiple scenes, rendered them separately and then merged the images. This way, they brought to the scene a total of 7,625,843 differently animated and mutually varied virtual characters. The number of characters in individual images varies between 0 and 3995, and an average of 501 are visible. To adapt the domain, they introduced the *Structural Similarity Index (SSIM)* in the traditional Cycle GAN, which allowed them to retain *local texture* and *structural similarity*. According to the authors' examples, the primary role of domain adaptation was *color correction* (reduction of saturation and change of image tone), which could be more easily achieved using the appropriate *3D look-up table (LUT)*, mapping one color space to another.

Chociej et al. (2019) created the ORRB (OpenAI Remote Rendering Backend), a simulator designed to visually train robots. ORRB is focused on domain randomization and *optimized for use in the cloud*: 88 rendering servers, each using eight V100 GPUs, producing 3438 frames per second, allowing for hyper-production of massive synthsets. The authors introduced *ambient occlusion* as a post-processing effect, in contrast to Baker et al. (2011), where it was initially used in rendering. They also pointed to the importance of the *seed* they used to make randomization and rendering deterministic to the extent that the game development tool (Unity) allows it. *Determinism* is extremely important in creating synthsets because it allows reproducibility and thus controls changes of individual parameters that affect the outcome of randomization.

Nowruzi et al. (2019) investigated object detectors' performance under conditions of a limited amount of real data. They used existing synthsets (Wrenninge and Unger 2018; Richter et al. 2017; Dosovitskiy et al. 2017) from the domain of autonomous driving, asking *how much real data*, compared to synthetic, *is needed*. Using only real sets, they concluded that removing as much as 90% of the set has a smaller negative effect on detector reliability than removing the next 5%. This may explain the adequacy of using the minimum real set (relative to synthset) to improve model performance and recommended *optimal ratio of 15–20 synthetic images* for each real one (Rozantsev et al. 2015).

Hinterstoisser et al. (2019) offered a solution in the same domain when real data is not available at all for specific training. They showed that neural network-based object detectors (Faster-RCNN, R-FCN, MaskRCNN) could be efficiently trained using exclusively synthetic data by *freezing the layers responsible for extracting features* of generic models initially trained on real data. This method is successful because the initial feature extractors (InceptionResnet and Resnet) are already trained enough that, when applied to synthetic data, they act as "projectors" and result in features that are close to the real domain.

Wang et al. (2019a) researched the detection of products in vending machines. For domain adaptation, as well as Atapour-Abarghouei and Breckon (2018), they used *style transfer* to make the rendered objects in their synthset more realistic. In doing so, they applied style transfer exclusively to individual objects of interest, omitting the environment because Cycle GAN changes it too much. The shape of individual products in vending machines can differ significantly due to the deformation of the packaging, which is why they simulated it by *deformation of the geometry*, randomizing the surface points' positions. The authors used PVANET, SSD and YOLOv3 for detection. They achieved the best results with PVANET (95.54%), while SSDs (90.02%) and especially YOLOv3 (62.33%) had problems with larger occlusions and objects' shape distortion that results from the use of a wide-angle camera.

Kar et al. (2019) created a Meta-Sim framework designed to generate synthetic urban environments for autonomous driving. They used SDR (Prakash et al. 2019) but learned distribution from real data, relevant for solving a specific task. Their generative model uses *Maximum Mean Discrepancy (MMD)* metrics to compare distributions, which allows them to *optimize scene parameters at each individual object's level*. This way, they retain the structure generated by probabilistic grammar but transform the distribution of attributes. They did this because they believed that the problem of domain adaptation consists of two components: the appearance (visual style of objects), which was dealt with by predecessors, but also content (layout and types of different objects on stage), which is why they introduced the term *distribution gap*.

Kong et al. (2020) built Synthinel-1, a synthset designed to segment buildings in aerial images. They used nine *different architectural styles* to achieve domain randomization, thus solving the problem observed in the work of Dosovitskiy et al. (2017). A new problem they faced is the size of the required virtual world in each frame because 10–20 km^2 of urban environment densely populated with 3D objects already sets significant memory requirements that the use of *levels of detail (LODs)*, criticized in (Fonder and Droogenbroeck 2019), cannot solve.

While Kong et al. (2020) preserved *spatial context* by keeping placed objects in spatial relationships similar to real-world environments, Hoeser and Kuenzer (2022) emphasized the temporal component, arguing that the *spatio-temporal context* of target objects and their environment is a major characteristic of Earth observation data, especially in areas with a strong tidal influence. They used *Synthetic-aperture radar (SAR)* data (digital elevation maps, DEM) as one source of 3D objects in their SyntEO synthset generation approach, applied to predicting offshore wind farms in Sentinel-1 images. SyntEO includes expert knowledge in the data generation process in a highly structured manner, which essentially follows the work of Prakash et al. (2019) on Structured Domain Randomization (SDR).

2.3 Discussion

Today, autonomous driving is one of the top topics in computer vision research and is also responsible for the trend of using synthsets in computer vision. It is particularly interesting that this (Pomerleau 1989) happened more than three decades ago during the pioneering use of neural networks and the realization that their training requires large amounts of data. With 90% accuracy, the first results were more than promising at the time—however, it was not noticed that this success should be attributed primarily to the similarity with real data resulting from the use of low-resolution sensors. We could say that then, back in 1989, the real and synthesized worlds looked almost identical to the computer.

The ability to generate custom data instead of using existing data has become increasingly important in all areas of computer vision. It encouraged the development of various synthsets generation methods and numerous techniques that accompanied the development of computer graphics. This early period was also marked by the boom of the computer game industry. Unlike the computer to which the synthesized and real roads looked almost identical, the average teenager, though deeply immersed in the virtual world of his favorite computer game, was fully aware that this pixelated world at average 320×200 pixels had little to do with reality. There was a big gap, one that researchers would identify as a "reality gap" many years later (Tremblay et al. 2018a), but an inexhaustible imagination neatly patched it up. Unfortunately, imagination is not something that computers can boast of.

Desires for more realistic worlds of computer games encouraged the computer industry to accelerate the development of graphics hardware, and 3D graphics slowly became an integral part of a large number of computers in the late nineties. It was also the first decade of synthset application—a decade that required researchers to generate them programmatically or program the generators that would create them. Few were ready for it. The initial enthusiasm for neural networks was suppressed, favoring numerous methods that we call "traditional" today, and the number of synthsets created in that period is almost negligible.

The beginning of the new millennium brought 3D graphics accelerators to players' homes, and the beginning of the mass development of 3D games also encouraged the development of various tools for the production of 3D content. Inspired by the possibility of 3D display of interactive worlds in real-time, the first simulators (Koenig and Howard 2004) were programmed as ideal polygons for robotic navigation, and, as well as to Pomerleau's car, such worlds seemed realistic enough to robotic sensors.

Although they did not report using the 3D tools already available at the time, Desurmont et al. (2006) decided to program the first sports synthset, and by naming it and sharing it with others, they started a positive practice. Unfortunately, this practice is still not generally accepted, so many synthsets have remained unnamed, unspecified and inaccessible to other researchers to this day.

At the end of this decade, photorealism became the holy grail of computer graphics. Researchers had only just begun to use renderers and very quickly encountered the problem of a lack of 3D content that they were forced to create themselves (Saxena et al. 2007). As they were surrounded by 3D games with plenty of 3D content, an attempt was made to solve the problem by using them (Taylor et al. 2007). However, commercial computer games are not designed for this purpose, so researchers ran into two problems: the technically limited ability to modify existing games and the legal problem of using the resulting content.

It turned out to be easier to solve technical than legal problems, which shifted the researchers' focus on the desire to control the content from which synthsets are created completely. Digital content creation tools, meanwhile, have matured to usability and satisfied that desire (Ragheb et al. 2008). Thanks to them and photorealism, which they approached at the expense of the speed of generating content by slow CPU rendering, entering the third decade of synthsets use, the synthesized and real worlds successfully collided for the third time.

3D graphics, in addition to computer games, began to dominate the film industry, and for the film, only photorealistic was good enough. Photorealism encouraged even faster development of 3D tools and the rise of hardware infrastructure that allowed to overcome the shortcomings of slow CPU rendering by using a large number of networked computers—the farms. Having access to such infrastructure, some researchers (Kaneva et al. 2011; Handa et al. 2014) pushed the boundaries of photorealism in synthsets, proving that the success of trained models depends on the visual similarity between the synthsets and the real world. At the same time, others (Pepik et al. 2012; Sun and Saenko 2014) have tried to prove that photorealism is not imperative, but most such attempts have ended with the conclusion that hybrid sets need to be used—those in which insufficient similarity between the synthesized and real worlds is "corrected" by using real images and features extracted from them.

However, being a result of rendering, photorealism alone is not enough to make the synthesized world similar enough to the real. For that, it is necessary to have enough different realistic variants of 3D objects and to realistically arrange them in the 3D scene, correctly defining the context.

In the middle of the third decade of using synthsets, we came close to solving the latter problem (Handa et al. 2014). The use of procedural techniques, which implement the rules of mutual organization of objects in the real world, made it possible to replace the often "impossible" randomly generated 3D scenes with "possible" and all the more realistic. Nevertheless, the problem of the availability of many different realistic variants of 3D objects for compositing scenes was not solved.

Meanwhile, CPU rendering has been replaced by several orders of magnitude faster GPU rendering, allowing massive synthsets to emerge (Wood et al. 2016) and raising the question of how much synthesized data is actually enough to train neural networks, which started to dominate.

Part of the answer to this question is found in the work of Tobin et al. (2017), who concluded that with sufficient variability in the simulator, the real world could look like another variation of the model. The other part is offered by Tripathi et al. (2019), who concluded that the generated synthetic data must be efficient, task-aware and realistic—with efficiency resulting from a simultaneous reduction in the amount of data, in order to save on the resources needed for training while increasing the diversity of samples.

That brings us back to the previous problem: the available amount of different realistic variants of 3D objects. Although the number of realistic and free or commercially available 3D objects produced has grown unquestionably over the years, it does not exhaust all the possible variations that we should pass on to the neural network to maximize its generalizing power.

Therefore, we agree with other authors (Tsirikoglou et al. 2017; Mayer et al. 2018; Nikolenko 2021) that the solution lies in the procedural approach. However, unlike them, we do not advocate using proceduralism solely as an intelligent randomizer of scene elements but also for automated production of unlimited amounts of 3D object variants *at the geometry level.*

By adopting this idea, one of the most critical tasks in generating synthsets would become the ability to procedurally describe what we otherwise consider intuitive while manually modelling 3D objects.

A banal example of this could be a simple chair, like the ones used in (Dosovitskiy et al. 2017). Today, using datasets such as COCO (Lin et al. 2014) and PASCAL VOC (Everingham et al. 2015), we could find a multitude of annotated photos of chairs. We could also find a multitude of 3D chair models and even vary them by changing the textures and context of the scene. Using such assets, we could build massive and diverse synthsets that are likely to achieve state-of-the-art results in some future experiments. However, can we exhaust the sample space of all possible chairs in this way? We think not. To exhaust it, we would first have to explain to ourselves, and then to the neural network, what a chair is. If we looked for the definition of a chair, we would come across something like "a separate seat for one person, typically with a backrest and four legs". Would that be a sufficient criterion for manual annotation? Again, we think not because a chair is not necessarily just a seat with 3 or 4 legs and a potential backrest or armrest. Procedural generation of 3D geometry allows us to shape and transfer such definitions from our understanding to computer instructions on how to generate the broadest range of possible chair combinations. In doing so, the computer would have to reach for artistic creation. Generative adversarial networks (Veeravasarapu et al. 2017; Wang et al. 2019b) try to do this their way. However, they can only combine what they have already "seen" without necessarily producing good enough results (Tian et al. 2018) that could be used to reliably train a neural network that seeks to maximize generalization while relying on sufficiently realistic examples.

In conclusion, by using the procedural creation of 3D objects at the geometry level, we could potentially satisfy the necessary diversity of synthsets and, instead of generating an infinite number of possible combinations of images, perhaps even find a new way to train neural networks directly by transmitting them information what an object actually is.

3 Analysis of synthsets generation

The largest number of synthsets (20% of the reviewed articles) was created in the domain of autonomous driving, so it should be considered that a significant part of the methods and techniques of their generation is optimized for this application.

3.1 The synthsets generation diagram

Based on the analyzed papers, we propose the diagram of synthsets generation processes (Fig. 6).

The synthsets generation diagram consists of *17 processes* where 16 (1–2 and further 4–17) make a linear sequence. The exception is the process under number 3 (integrations), which represents an alternative way of using the synthesized data by integrating generation methods with one of the machine learning platforms (Lerer et al. 2016; Qiu and Yuille 2016; Zhu et al. 2017; Müller et al. 2018). As such, it is not the subject of interest of this article.

The numbering used to denote individual processes derives from the synthsets generation diagram, built during this research, according to the chronologically first appearance of a particular item in the papers covered in Sect. 2 and the order (1–17) in which individual items participate in the process of generating a synthset. Numbering is used hereinafter in the text, within parentheses, to reference systematization hierarchy items.

3.2 The synthsets generation processes

To build a synthset generation pipeline, authors should make a series of decisions by choosing a path through 17 processes selecting appropriate synthset generation methods and techniques (Table 1) depending on the specific requirements of their task. The course of these paths (for compositing, game engine, and digital content creation methods) is shown in Figs. 7, 8 and 9.

Table 1 presents condensed systematization of the methods and techniques used within the synthset generation process. The systematization is built according to the chronological first appearance of a particular method or technique in analyzed works and in the order (1–17) in which they participate in the synthset generation process. Extended systematization is available in the Supplementary material (Online Resource 1).

3.2.1 Reference dataset

If a reference dataset is available during the construction of the synthset, its optional analysis (1) makes it possible to perform algorithmic estimation of the shading (12) parameters (Rozantsev et al. 2015) and determine the distribution of features that is desirable to achieve with the synthset (Veeravasarapu et al. 2015).

3.2.2 Generation method

The first mandatory process is the selection of the synthset generation method (2). In the analyzed works, nine *different methods* (2.1–2.9) were identified, shown in Fig. 10.

Some methods are optionally contained in more complex methods, which is indicated by a dashed line in Fig. 10. Game engine method (2.6) allows direct integration with machine learning platforms (3), and the highlighted methods (2.4, 2.6 and 2.7) are shown in detail in Figs. 7, 8 and 9. The “2D” and “3D” labels refer to the type of scene (2D or 3D) that is created.

The *method of using the generator* (2.1) (Pomerleau 1989) relies on an existing generator that can be pre-programmed or built using a tool for digital content creation or a physics or game engine. The generator allows the user to change the synthesis parameters, and the output does not necessarily generate in real-time. In the *programming method* (2.2) (Little and Verri 1989), the synthset is created algorithmically, by direct programming of the output. The method of *using the simulator* (2.3) (Koenig and Howard 2004) relies on the programming of the respective or the use of a game engine or a modified commercial computer game to conduct a specific simulation. Unlike generators, simulators, such as Gazebo, Sim4CV or CARLA, most often generate output in real-time. The *digital content creation method* (2.4) (Saxena et al. 2007) relies on tools, such as 3ds Max, Blender or After Effects, that often contain a physics engine and enable automation by programming. The possibility of using the *commercial computer game method* (2.5) (Taylor et al. 2007) depends on the license which regulates whether the game may be used to generate a synthset. If it can, the biggest problem is the way to access the content within the game, possibly adapt it to own needs and output it in the appropriate form (final image with the corresponding annotation). The technical basis of every computer game is the *game engine* (2.6) (Rivera-Rubio et al. 2015), such as Unity or Unreal Engine, which is, when used as a production tool, also a distinct method that can produce a synthset without first making a game. The *compositing method* (2.7) (Su et al. 2015) treats the scene as a set of 2D layers laid on top of each other using a minimum of two layers (one each for the image's background and foreground). An integral part of most game engines is the *physics engine* (2.8) (McCormac et al. 2017), which can also exist as separate software specializing in a particular physics simulation type. Physics engine can directly produce a synthset, making its application a distinct method. Chronologically, the latest method is *generative adversarial training* (2.9) (Veeravasarapu et al. 2017) which uses neural networks (GANs) during the synthset generation process but is limited by the inability to create appropriate annotations automatically and is more suitable for use as an auxiliary tool for domain adaptation (17.4).

Processes 4–17, their branching and convergence of individual items can be observed in Figs. 7, 8 and 9.

3.2.3 Source of objects

In the case of creating *2D scenes* (8.1), the most common *sources* of 2D objects intended for blending (7.1.1) (Su et al. 2015) are previously rendered images (4.1.1) (Mayer et al. 2016), which may contain masks for automatic extraction of the object of interest, and photographs (4.1.2) (Georgakis et al. 2017), which require manual extraction. When it comes to *3D scenes* (8.2), 3D objects can be found on the Internet (4.2.1) (Wu et al. 2014), procedurally (4.2.2) (Hamarneh and Gustavsson 2004) or manually (4.2.3) (Peris et al. 2012) generated, built by converting OSM maps to 3D geometry (4.2.4) (Tian et al. 2018), generated as L-System (4.2.5) (Ubbens et al. 2018) or by using photogrammetry (4.2.7) (Chen et al. 2020), used as a point cloud, obtained by LiDAR (4.2.6) (Li et al. 2019), which can also be converted to 3D geometry, and used as a digital elevation model (DEM), obtained by Synthetic-aperture radar (SAR) (4.2.8) (Hoeser and Kuenzer 2022).

3.2.4 Modification of objects

3D objects are often pre-processed when introduced to the 3D scene. Their *modification* can be parametric (5.1.1) (Queiroz et al. 2010), manual (5.1.2), and reduced to retopologizing (5.1.3) or scaling on different axes (5.1.4) (Carlucci et al. 2017). If the 3D objects are obtained with LiDAR, it is possible to pre-process the entire LiDAR scene (5.1.5) (Li et al. 2019).

3.2.5 Scene elements, scene generation and scene

While typical *elements* (6) *of a 2D scene* are 2D objects (6.1) and a background (6.5) (Pishchulin et al. 2011), a background (as a 2D object) can also be part of a *3D scene*, with 3D objects (6.2), lights (6.3) (Pomerleau 1989) and at least one camera (6.4) (Desurmont et al. 2006).

The *3D scene can be generated* manually (7.2.1) (Mayer et al. 2018), procedurally (7.2.2) (Johnson et al. 2017), by physics simulation (7.2.3) (McCormac et al. 2017), generative adversarial training (7.2.4) (Veeravasarapu et al. 2017) and using augmented reality (7.2.5) (Sharma et al. 2018).

The product of scene generation using selected scene elements is 2D (8.1) or 3D *scene* (8.2).

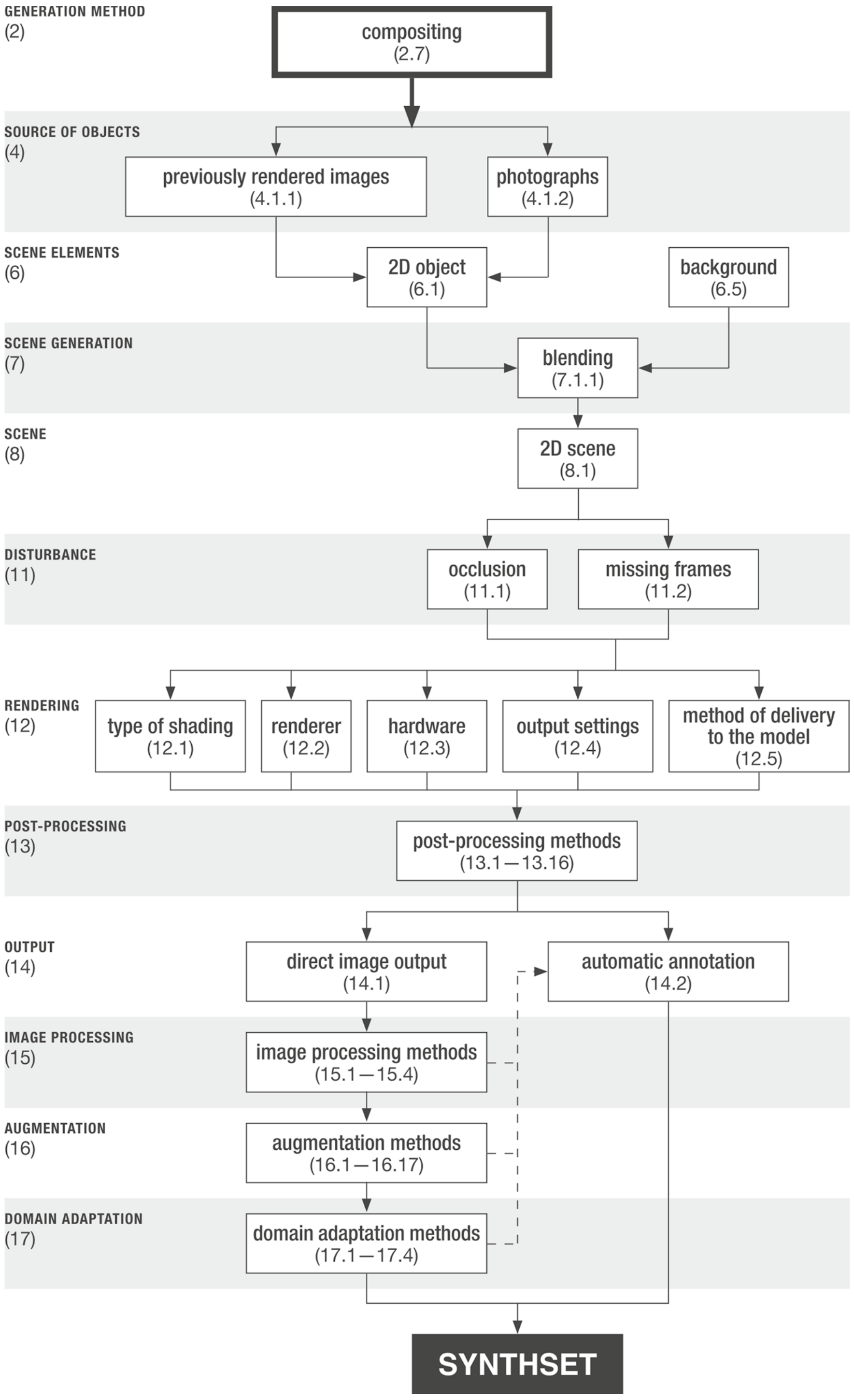

Fig. 7 Diagram of the synthset generation process for the compositing method

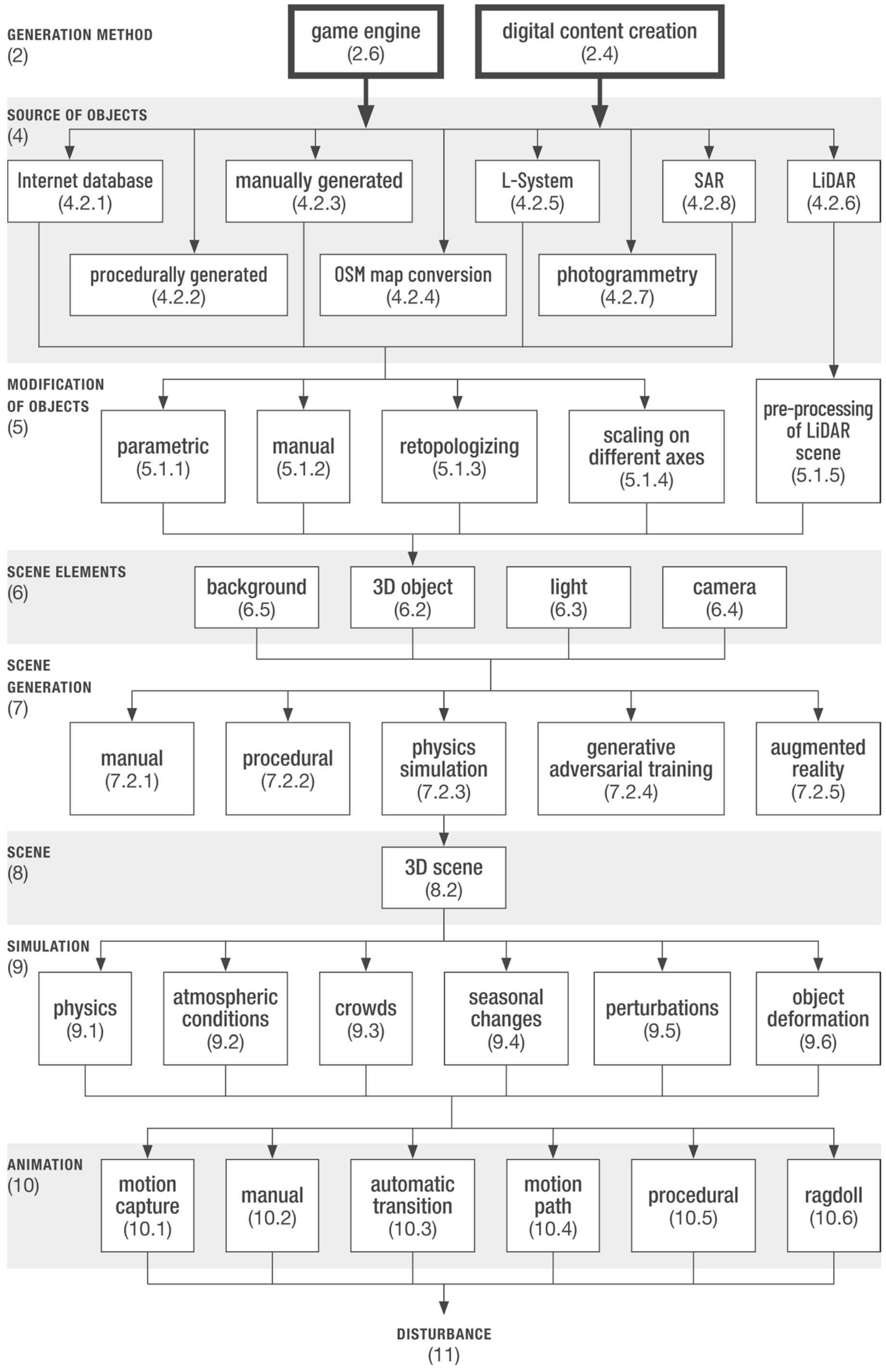

Fig. 8 Part 1 (of 2) of the diagram of the synthset generation process for the game engine method and the digital content creation method (showing processes 2–10)

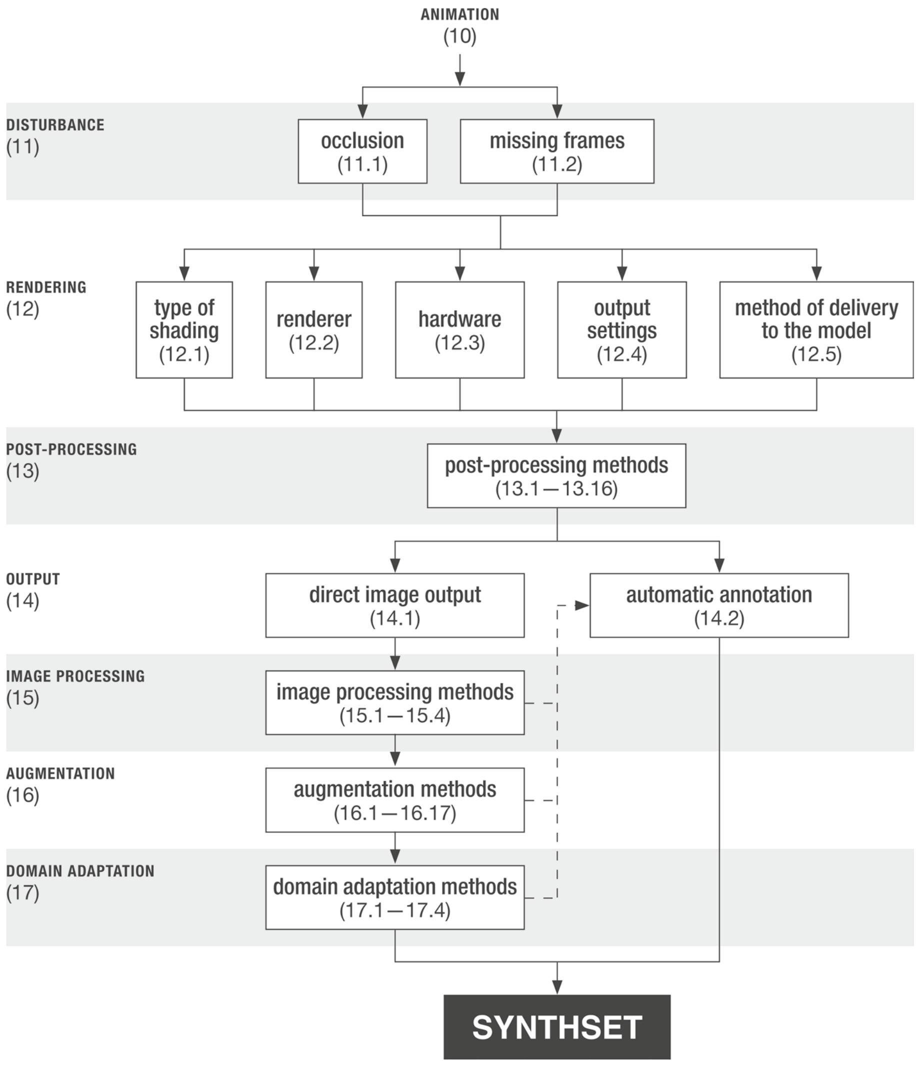

Fig. 9 Part 2 (of 2) of the diagram of the synthset generation process for the game engine method and the digital content creation method (showing processes 11–17)

3.2.6 Simulation and animation

Simulation and animation both add temporal change to the scene elements. The main difference between them is that simulation uses a predefined set of rules (i.e., physics laws) to calculate the resulting change, while animation allows creators (animators) to affect this change in both a realistic and unrealistic manner directly. Although it is technically possible to perform *simulation* (9) and *animation* (10) on 2D scenes, they are performed exclusively on 3D scenes to generate synthsets. Physics (9.1) (Lin et al. 2016), atmospheric conditions (9.2) (Vacavant et al. 2013), crowds (9.3) (Courty et al. 2014), seasonal changes (9.4) (Ros et al. 2016), perturbations (9.5) (Solovev et al. 2018) and object deformations (9.6) (Wang et al. 2019a) are

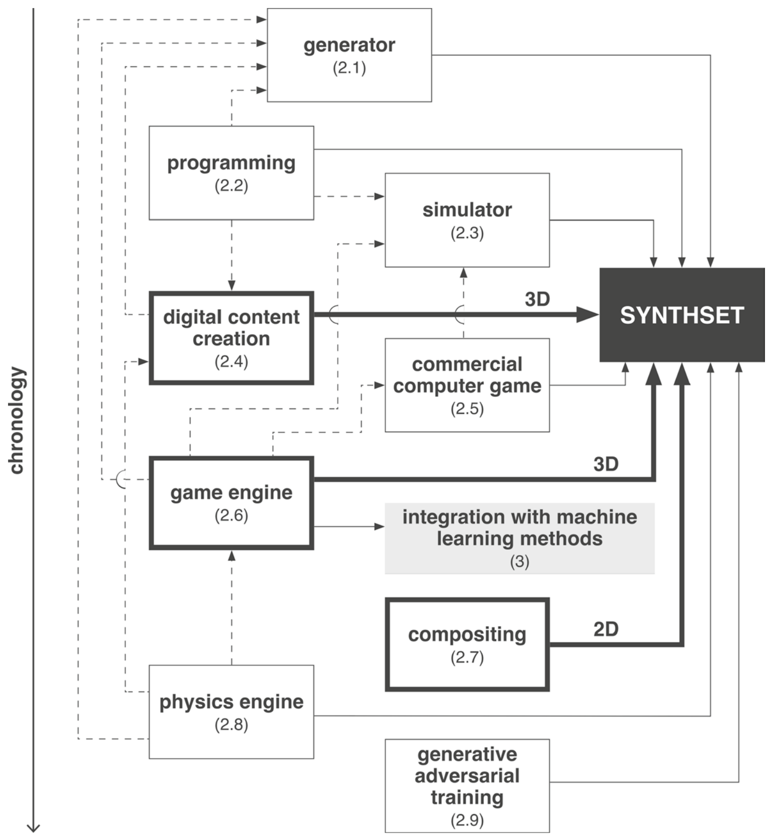
chronology
generator (2.1)
programming (2.2)
simulator (2.3)
digital content creation (2.4)
commercial computer game (2.5)
game engine (2.6)
integration with machine learning methods (3)
compositing (2.7)
physics engine (2.8)
generative adversarial training (2.9)
3D
3D
2D
SYNTHSET

Fig. 10 Synthset generation methods and their mutual influence (best viewed in digital format)

simulated. The animation includes motion-capture (10.1) (Pishchulin et al. 2011), manual animation (10.2) (Grauman et al. 2003), (Ragheb et al. 2008), automatic transitions between animations (10.3) (Ragheb et al. 2008), predefined motion paths (10.4) (Henry et al. 2013), procedural animation (10.5) (Wood et al. 2016), and ragdoll animation (10.6) (De Souza et al. 2017).

3.2.7 Disturbance

The first convergence of 2D and 3D pathways during synthset generation is present in the process of *adding disturbance* (11). The disturbance is most often introduced as some form of occlusion (11.1) and as missing frames (11.2) in video sequences (Hamarneh and Gustavsson 2004).

3.2.8 Rendering

The *rendering* process (12) applies exclusively to 3D scenes. It defines the type of shading (12.1) (Tsirikoglou et al. 2017) and selects the renderer (12.2), the rendering hardware (12.3) (Papon and Schoeler 2015), the output settings (12.4) (Peris et al. 2012) and, limited by previous selections, the method of delivering the output to the model being trained (12.5) (Mnih et al. 2013).

3.2.9 Post-processing

The second convergence of 2D and 3D paths occurs in the *post-processing* process (13) when noise (13.1) (Pomerleau 1989) is added to the generated image; performs smoothing (13.2) (Barron et al. 1994); introduce image distortion (13.3) and, for video sequences, video ghosting (13.4) (Taylor et al. 2007); performs antialiasing (13.5) (Taylor et al. 2007); add fog (13.6) (Tarel et al. 2010), motion blur (13.7) and focus (13.8) (Butler et al. 2012), and sun glare (13.9) (Mayer et al. 2016); performs game curve manipulation (13.10) (Mayer et al. 2016); add vignette (13.11) (Movshovitz-Attias et al. 2016), chromatic aberration (13.12) (Veeravasarapu et al. 2016), automatic exposure (13.13) (Wrenninge and Unger 2018) and ambient occlusion (13.14) (Chociej et al. 2019); and simulate codec errors (13.15) and lens contamination (13.16) (Temel et al. 2019).

3.2.10 Output

In the process of *output* (14), direct image output (14.1) (Pomerleau 1989) and automatic annotation (14.2) are created, which can be influenced by the result of each of the upcoming three processes (15–17).

3.2.11 Image processing

The *image processing* process (15) uses the methods of down-sampling (15.1) (Mnih et al. 2013), cropping (15.2) (Mnih et al. 2013), removing the background by segmentation before training (15.3) (Moiseev et al. 2013) and warping (15.4) (Zioulis et al. 2019).

3.2.12 Augmentation

When generating synthsets, reducing the gap between the computer-generated image and the image captured by a real camera is necessary. This reduction is made during post-processing (3.2.9), using a set of techniques that are also applied when augmenting real datasets, but here they are used to increase the image's quality in terms of realism. Unlike post-processing, the augmentation process during synthset generation increases the synthset's quantity, thus saving the resources needed to generate unique images. However, considering the need for the synthset to be efficient (Tripathi et al. 2019), augmentation gives way to the optimization of the preceding processes in order to simultaneously save on the resources needed for training while increasing the samples' diversity. If the basic synthset needs to be expanded by *augmentation* (16) (Hamarneh and Gustavsson 2004), one of the following methods is used: occlusion (16.1), cropping (16.2), changing the contrast of the depth map (16.3), changing the brightness of the depth map (16.4), replacing the white background color in the depth map with a random color away from the center of mass of the object (16.5), shearing (16.6), 2D rotation (16.7), 3D rotation (16.8), truncation (16.9), distractor objects (16.10), brightness change (16.11), contrast change (16.12), mirroring (16.13), homography (16.14), sharpening (16.15), embossing (16.16) and color channel inversion (16.17).

3.2.13 Domain adaptation

If the *adaptation of the domain* (17) will not be carried out by mixing the synthset with the real dataset, it can be done as part of the synthset generation process by using active learning (17.1) (Vazquez et al. 2014), transfer learning (17.2) (Papon and Schoeler 2015), style transfer (17.3) (Atapour-Abarghouei and Breckon 2018) and adversarial training (17.4) (Atapour-Abarghouei and Breckon 2018).

3.3 Usage of generation methods in computer vision tasks and domains

The analyzed works include 85 produced synthsets, 12 simulators, and six generators, all publicly available and named, which we were able to explore in more detail.

Observing the representation of generation methods in computer vision tasks and domains (Table 2), we can conclude that digital content creation is the most frequently used method (29.13%). It is the favored method for optical flow and the only method used in scene reconstruction tasks. This method's popularity can be explained by the flexibility that digital content creation tools provide for scene generation, unencumbered by the real-time performance imperative inherent in the game engine method. Also, this method enables the achievement of the highest degree of photorealism.

The second most used method is the game engine (19.42%). Along with the commercial computer game method, it is also one of the preferred methods in autonomous driving and the preferred method for pose estimation. Its key advantage is the possibility of real-time performance but at the expense of photorealism. The method of using the simulator (16.5%) is preferred in robot navigation, as well as in autonomous flying, where it benefits from physics simulation.

The low representation (0.97% each) of compositing method, physics engine method and a combination of digital content creation, game engine and simulator methods can be explained by the exclusive application of the compositing method for the optical flow task, incorporating a physics engine into the game engine and, in the case of using a combination of methods, by preferring simpler synthset production processes by most authors. In optical flow, the use of the digital content creation method prevailed over time.

The column named "not specified" refers to 9.71% of papers in which the authors did not specify the method of generating their own synthsets.

Focusing on produced synthsets (excluding simulators and generators), we can follow the introduction and usage of specific methods during the documented period (2006–2021). It is evident from the Gantt chart (Fig. 11) that once introduced methods mostly survive, being used to this day.

In the observed set, the exceptions are the compositing and the physics engine methods used in shorter periods. The prevalence of digital content creation and game engine methods (Fig. 12) can explain the reduced interest in their use. Both digital content creation and game engine methods offer integrated physics engines, ease of use, and a large number of additional options (which include photorealistic rendering and integrated tools for automatic annotation), attracting an increasing number of researchers.

3.4 Discussion

Over the past three decades, synthsets have become the solution to many problems related to preparing a large amount of quality data for deep learning. With a potentially unlimited amount of data generated quickly and economically, they are characterized by the possibility of automated annotation. The automated annotation also eliminates the potential human error inherent in manual annotation, which can jeopardize the training process.

In this article, the past and current synthset generation practices are examined, the systematization of the synthset generation process is proposed, and nine different synthset generation methods are identified.

Of the nine identified generation methods, three compete for building synthsets to represent dynamic environments (such as traffic, crowds, and sports): *digital content creation*, *game engine,* and *compositing*. While the selection of the first two methods follows the previously established trend of representation of generation methods in existing synthsets and meets the requirements of photorealism (Zhang et al. 2017), compositing is chosen as the third option because it can be realized in parallel with one of the previous two methods, using the resources created in the generation process.

Although programming (from scratch) is still widely used in many domains, we do not recommend it as a generation method for dynamic environments (due to their complexity and photorealistic requirements) but only for the automation of processes within the selected methods. The same is true for the method of using a generator that is not optimal to build from scratch, given that the infrastructure required for generation is already developed within various tools for creating digital content and game engines. The use of commercial computer games is not a desirable option because of technical (Müller et al. 2018) and legal (Shafaei et al. 2016) problems. The method of using a standalone physics engine is excluded because physics simulation is not necessary (which also excludes the method of using a simulator), and if we want to use it, the physics engine is available within the digital content creation tools and game engines. We currently suggest avoiding the method of generative adversarial training for producing entire synthetic images due to its complexity and problematic annotations (Tian et al. 2018). However, GAN has enormous potential to provide a large variety of scene elements that could be blended into images using compositing methods; therefore, we can recommend future research in this direction.

The main advantage of using the digital content creation method, in addition to achieving photorealism, is the unlimited ability to manage all aspects of the 3D scene, which is a prerequisite for effective domain randomization (Tobin et al. 2017). The generative character of the digital content creation tools enables the procedural generation of a unique scene for each synthesized frame (Tsirikoglou et al. 2017), but also of each individual element of the scene, including geometry, which, for technical reasons, can hardly be achieved using only the game engines. Additional advantages of this method are the ability to distribute rendering to multiple computers and render arbitrary frames nonlinearly. However, in the case of limited rendering resources, at the expense of maximum photorealism, it is possible to apply a compromise and make appropriate adjustments to the geometry and other elements of the scene for application within the game engine and thus realize real-time rendering, with sufficient photorealism.

The compositing method depends on the availability of elements for the construction of the foreground and background. Foreground elements can be generated automatically during the use of the digital content creation method or the use of the game engine, and

Table 2 Representation of generation methods in computer vision tasks and domains (sorted by summarized percentages in descending order)

Generation methods/tasks and domains	Digital content creation (DCC)	Game engine	Simulator	Not specified	Programming	Commercial computer game	Generator	Compositing	DCC + game engine + simulator	Physics engine	∑
Autonomous driving	2.91	5.83	3.88	1.94	0.97	5.83	–	–	0.97	–	22.33
Pose estimation	2.91	3.88	–	1.94	0.97	–	–	–	–	–	9.71
Robot navigation	–	1.94	5.83	0.97	0.97	–	–	–	–	–	9.71
Object detection	2.91	0.97	0.97	–	0.97	–	0.97	–	–	–	6.80
Surveillance	0.97	0.97	1.94	–	–	1.94	0.97	–	–	–	6.80
Optical flow	3.88	–	–	0.97	–	–	–	0.97	–	–	5.83
Segmentation	1.94	–	–	–	1.94	–	1.94	–	–	–	5.83
Autonomous flying	–	0.97	2.91	0.97	–	–	–	–	–	–	4.85
Scene understanding	0.97	–	–	0.97	0.97	–	0.97	–	–	0.97	4.85
Object classification	1.94	0.97	–	0.97	–	–	–	–	–	–	3.88
Scene reconstruction	3.88	–	–	–	–	–	–	–	–	–	3.88
Stereo disparity	0.97	1.94	–	–	–	–	0.97	–	–	–	3.88
Action recognition	1.94	–	0.97	–	–	–	–	–	–	–	2.91
Image features evaluation	1.94	–	–	–	–	–	–	–	–	–	1.94
Object reconstruction	0.97	–	–	–	0.97	–	–	–	–	–	1.94
Object tracking	–	0.97	–	–	0.97	–	–	–	–	–	1.94
Gaze estimation	–	0.97	–	–	–	–	–	–	–	–	0.97
Light fields analysis	0.97	–	–	–	–	–	–	–	–	–	0.97

Table 2 (continued)

Generation methods/tasks and domains	Digital content creation (DCC)	Game engine	Simulator	Not specified	Programming	Commercial computer game	Generator	Compositing	DCC + game engine + simulator	Physics engine	∑
Texture generation	–	–	–	0.97	–	–	–	–	–	–	0.97
∑	29.13	19.42	16.50	9.71	8.74	7.77	5.83	0.97	0.97	0.97	100.00%

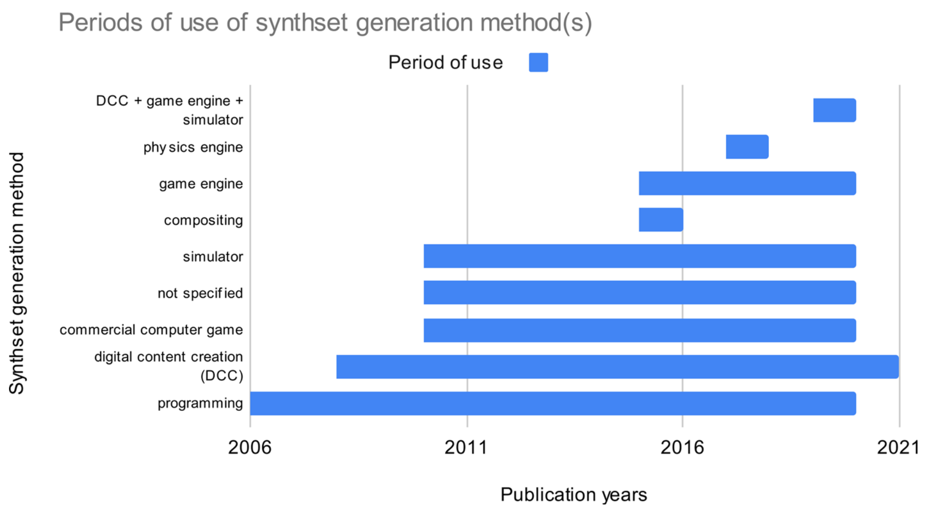

Fig. 11 Gantt chart representing introductions and periods of use of synthset generation method(s) between 2006 and 2020

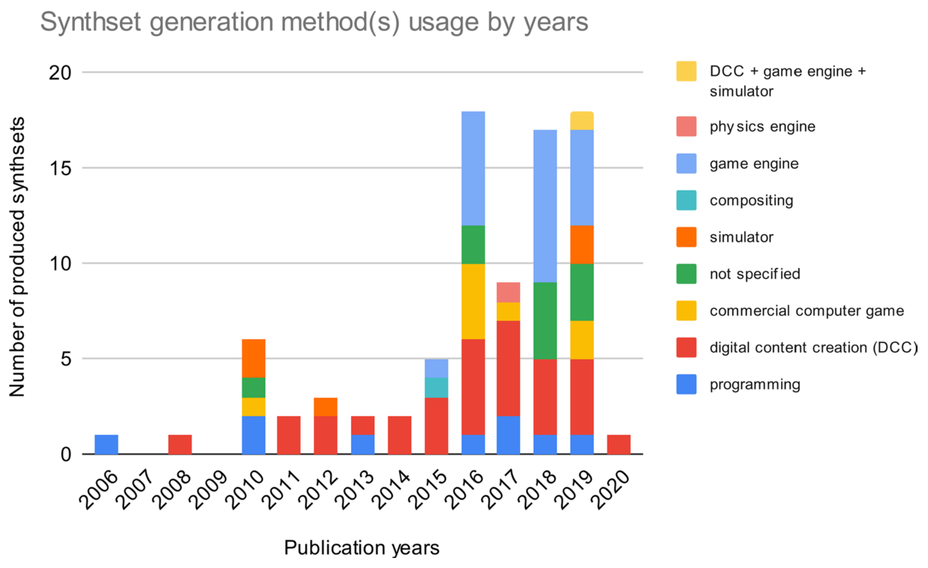

Fig. 12 Histogram representing synthset generation methods usage by years between 2006 and 2020

background elements can be created by photographing (Zimmermann and Brox 2017) during the creation of a reference (real) dataset.

There is no previous comparison of efficiency for the selected primary generation methods. Thus, for future work, to determine which method and by what criteria is the best in practice, we recommend building and evaluating two primary or all three proposed synthsets in parallel, using the *procedural generation* of not only individual scenes but all elements of the scene.

4 Conclusion

This paper focused on analyzing the literature related to the use of synthsets in the computer vision domain, intending to discover and systematize the process of their generation and the methods and techniques involved.

We first presented synthetic data and its advantages over real datasets. Since our motivation was to find best practices for building synthsets to represent dynamic environments, we singled out papers containing methods and techniques suitable for such tasks and described them, emphasizing the original techniques and their effects, but also potential problems of their use.

We then discussed prospects of using the procedural generation of 3D objects variants at the geometry level, believing that it can solve some of the fundamental problems identified. The analysis has presented the 17 individual processes of synthsets generation. We identified nine different generation methods and further analyzed compositing, game engine and digital content creation methods, which we consider the most suitable for building synthsets appropriate to represent dynamic environments. The advantage of using the digital content creation method is its ability to achieve photorealism while managing all aspects of the 3D scene, including 3D objects on a geometry level, as a prerequisite for effective domain randomization. The game engine method can solve the problem of limited rendering resources with sufficient photorealism, and the compositing method can be realized in parallel with the previous two methods, using the resources created in the process. We organized these methods into possible generation paths that synthset creators should follow and choose from, depending on the specific requirements of their future synthsets.

In addition, we have proposed the systematization of all the elements involved in the process of generating a synthset. We also presented the statistics of the use of different methods in specific application domains, commenting on the reasons for the observed distribution to help the authors of future synthsets choose the generation method as one of the key steps in the synthset generation process. Since there is no previous comparison of the effectiveness of the three selected methods for building synthsets to represent dynamic environments, we proposed parallel construction of these synthsets using the respective methods and procedural generation of all scene elements, including the geometry of 3D objects.

For future work, we propose building a collection of synthsets for the same task to enable a fair comparison of all nine generation methods, stressing their pros and cons, and summarizing the evaluation criteria in terms of qualitative and quantitative aspects. We also suggest that forthcoming synthset authors document their synthset evaluation methods along with the synthset generation process.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s10462-022-10358-3.

Funding This study was funded by Hrvatska Zaklada za Znanost, IP-2016-06-8345, Marina Ivasic-Kos.

References

Abayomi-Alli OO, Damaševičius R, Wieczorek Michałand Woźniak M (2020) Data augmentation using principal component resampling for image recognition by deep learning. In: Rutkowski L, Scherer Rafałand KM, Pedrycz W et al (eds) Artificial intelligence and soft computing. Springer International Publishing, Cham, pp 39–48

Abu Alhaija H, Mustikovela SK, Mescheder L et al (2018) Augmented reality meets computer vision: efficient data generation for urban driving scenes. Int J Comput Vis 126:961–972. https://doi.org/10.1007/s11263-018-1070-x

Aranjuelo N, García S, Loyo E et al (2021) Key strategies for synthetic data generation for training intelligent systems based on people detection from omnidirectional cameras. Comput Electr Eng 92:107105. https://doi.org/10.1016/j.compeleceng.2021.107105

Atapour-Abarghouei A, Breckon TP (2018) Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer. Proc IEEE Comput Soc Conf Comput vis Pattern Recognit. https://doi.org/10.1109/CVPR.2018.00296

Baker S, Scharstein D, Lewis JP et al (2011) A database and evaluation methodology for optical flow. Int J Comput Vis 92:1–31. https://doi.org/10.1007/s11263-010-0390-2

Bargoti S, Underwood J (2017) Deep fruit detection in orchards. In: 2017 IEEE International Conference on Robotics and Automation (ICRA). pp 3626–3633. https://doi.org/10.1109/ICRA.2017.7989417

Barron JL, Fleet DJ, Beauchemin SS (1994) Systems and experiment performance of optical flow techniques. Int J Comput Vis 12:43–77. https://doi.org/10.1007/BF01420984

Burić M, Ivašić-Kos M, Paulin G (2019) Object detection using synthesized data. In: ICT innovations 2019 web proceedings. pp 110–124

Butler DJ, Wulff J, Stanley GB, Black MJ (2012) A naturalistic open source movie for optical flow evaluation. In: Fitzgibbon A, Lazebnik S, Perona P et al (eds) Computer vision—ECCV 2012. Springer, Berlin, Heidelberg, pp 611–625

Cai W, Liu D, Ning X et al (2021) Voxel-based three-view hybrid parallel network for 3D object classification. Displays 69:102076. https://doi.org/10.1016/j.displa.2021.102076

Carlucci FM, Russo P, Caputo B (2017) A deep representation for depth images from synthetic data. Proc—IEEE Int Conf Robot Autom. https://doi.org/10.1109/ICRA.2017.7989162

Cazzato D, Cimarelli C, Sanchez-Lopez JL et al (2020) A survey of computer vision methods for 2D object detection from unmanned aerial vehicles. J Imaging. https://doi.org/10.3390/jimaging6080078

Chen W, Wang H, Li Y, et al (2016) Synthesizing training images for boosting human 3D pose estimation. Proc—2016 4th Int Conf 3D Vision, 3DV 2016 479–488. https://doi.org/10.1109/3DV.2016.58

Chen M, Feng A, McCullough K, et al (2020) Generating synthetic photogrammetric data for training deep learning based 3D point cloud segmentation models. https://arxiv.org/abs/2008.09647

Chociej M, Welinder P, Weng L (2019) ORRB—OpenAI remote rendering backend. http://arxiv.org/abs/1906.11633

Courty N, Allain P, Creusot C, Corpetti T (2014) Using the agoraset dataset: assessing for the quality of crowd video analysis methods. Pattern Recognit Lett 44:161–170. https://doi.org/10.1016/j.patrec.2014.01.004

Deschaintre V, Aittala M, Durand F et al (2018) Single-image SVBRDF capture with a rendering-aware deep network. ACM Trans Graph. https://doi.org/10.1145/3197517.3201378

Desurmont X, Hayet JB, Delaigle JF, et al (2006) Trictrac video dataset: Public hdtv synthetic soccer video sequences with ground truth. Work Comput Vis Based Anal Sport Environ 92–100

Dosovitskiy A, Ros G, Codevilla F, et al (2017) CARLA: an open urban driving simulator. http://arxiv.org/abs/1711.03938

Dvornik N, Mairal J, Schmid C (2021) On the importance of visual context for data augmentation in scene understanding. IEEE Trans Pattern Anal Mach Intell 43:2014–2028. https://doi.org/10.1109/TPAMI.2019.2961896

Dvornik N, Mairal J, Schmid C (2018) Modeling visual context is key to augmenting object detection datasets. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 11216 LNCS: 375–391. https://doi.org/10.1007/978-3-030-01258-8_23

Dwibedi D, Misra I, Hebert M (2017) Cut, paste and learn: surprisingly easy synthesis for instance detection. Proc IEEE Int Conf Comput Vis 2017-Octob:1310–1319. https://doi.org/10.1109/ICCV.2017.146

Everingham M, Eslami SMA et al (2015) The pascal visual object classes challenge: a retrospective. Int J Comput vis 111:98–136

Fisher R (2021) CVonline: Image databases. http://homepages.inf.ed.ac.uk/rbf/CVonline/Imagedbase.htm. Accessed 14 Mar 2021

Fonder M, Van Droogenbroeck M (2019) Mid-Air: a multi-modal dataset for extremely low altitude drone flights. In: 2019 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW). IEEE, pp 553–562

Gaidon A, Wang Q, Cabon Y, Vig E (2016) VirtualWorlds as proxy for multi-object tracking analysis. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-Decem:4340–4349. https://doi.org/10.1109/CVPR.2016.470

Garbin SJ, Komogortsev O, Cavin R et al (2020) Dataset for eye tracking on a virtual reality platform. ACM symposium on eye tracking research and applications. ACM, New York, pp 1–10

Georgakis G, Mousavian A, Berg AC, Košecká J (2017) Synthesizing training data for object detection in indoor scenes. Robot Sci Syst. https://doi.org/10.15607/rss.2017.xiii.043

Girdhar R, Ramanan D (2019) CATER: a diagnostic dataset for compositional actions and temporal reasoning. http://arxiv.org/abs/1910.04744

Grauman K, Shakhnarovich G, Darrell T (2003) Inferring 3D structure with a statistical image-based shape model. Proc IEEE Int Conf Comput vis 1:641–648. https://doi.org/10.1109/iccv.2003.1238408

Haltakov V, Unger C, Ilic S (2013) Framework for generation of synthetic ground truth data for driver assistance applications BT—pattern recognition. In: Weickert J, Hein M, Schiele B (eds) Springer. Springer, Heidelberg, pp 323–332

Hamarneh G, Gustavsson T (2004) Deformable spatio-temporal shape models: extending active shape models to 2D+time. Image Vis Comput 22:461–470. https://doi.org/10.1016/j.imavis.2003.11.009

Handa A, Whelan T, McDonald J, Davison AJ (2014) A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM. Proc—IEEE Int Conf Robot Autom. https://doi.org/10.1109/ICRA.2014.6907054

Hattori H, Boddeti VN, Kitani K, Kanade T (2015) Learning scene-specific pedestrian detectors without real data. In: 2015 IEEE conference on computer vision and pattern recognition (CVPR). pp 3819–3827

Henry KM, Pase L, Ramos-Lopez CF et al (2013) PhagoSight: an open-source MATLAB® package for the analysis of fluorescent neutrophil and macrophage migration in a zebrafish model. PLoS ONE. https://doi.org/10.1371/journal.pone.0072636

Hinterstoisser S, Lepetit V, Wohlhart P, Konolige K (2019) On pre-trained image features and synthetic images for deep learning. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 11129 LNCS:682–697. https://doi.org/10.1007/978-3-030-11009-3_42

Hoeser T, Kuenzer C (2022) SyntEO: synthetic dataset generation for earth observation and deep learning—demonstrated for offshore wind farm detection. ISPRS J Photogramm Remote Sens 189:163–184. https://doi.org/10.1016/j.isprsjprs.2022.04.029

Host K, Ivasic-Kos M, Pobar M (2022) Action recognition in handball scenes. In: Arai K (ed) Intelligent computing. Springer International Publishing, Cham, pp 645–656

Janai J, Güney F, Behl A, Geiger A (2020) Computer vision for autonomous vehicles: problems, datasets and state-of-the-art. Found Trends Comput Graph Vis 12:1–308

Jiang C, Qi S, Zhu Y et al (2018) Configurable 3D scene synthesis and 2D image rendering with per-pixel ground truth using stochastic grammars. Int J Comput Vis 126:920–941. https://doi.org/10.1007/s11263-018-1103-5

Johnson J, Fei-Fei L, Hariharan B, et al (2017) CLEVR: a diagnostic dataset for compositional language and elementary visual reasoning. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua:1988–1997. https://doi.org/10.1109/CVPR.2017.215

Johnson-Roberson M, Barto C, Mehta R et al (2017) Driving in the matrix: can virtual worlds replace human-generated annotations for real world tasks? Proc—IEEE Int Conf Robot Autom. https://doi.org/10.1109/ICRA.2017.7989092

Kaneva B, Torralba A, Freeman WT (2011) Evaluation of image features using a photorealistic virtual world. Proc IEEE Int Conf Comput Vis. https://doi.org/10.1109/ICCV.2011.6126508

Kar A, Prakash A, Liu MY, et al (2019) Meta-sim: learning to generate synthetic datasets. Proc IEEE Int Conf Comput Vis 2019-Octob:4550–4559. https://doi.org/10.1109/ICCV.2019.00465

Khan S, Phan B, Salay R, Czarnecki K (2019) CVPR workshops—ProcSy: procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In: proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR) workshops. pp 88–96

Koenig N (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. IEEE/RSJ Int Conf Intell Robot Syst 3:2149–2154. https://doi.org/10.1109/iros.2004.1389727

Kong F, Huang B, Bradbury K, Malof JM (2020) The synthinel-1 dataset: a collection of high resolution synthetic overhead imagery for building segmentation. Proc—2020 IEEE Winter Conf Appl Comput Vis WACV 2020:1803–1812. https://doi.org/10.1109/WACV45572.2020.9093339

Lange D (2020) Synthetic data: a scalable way to train perception systems. https://developer.nvidia.com/gtc/2020/video/s22700-vid. Accessed 31 May 2020

Larumbe A, Ariz M, Bengoechea JJ et al (2017) Improved strategies for HPE employing learning-by-synthesis approaches. In: 2017 IEEE International Conference on Computer Vision Workshops (ICCVW). pp 1545–1554

Lerer A, Gross S, Fergus R (2016) Learning physical intuition of block towers by example. 33rd Int Conf Mach Learn ICML 2016 1:648–656

Li W, Pan CW, Zhang R et al (2019) AADS: Augmented autonomous driving simulation using data-driven algorithms. Sci Robot. https://doi.org/10.1126/scirobotics.aaw0863

Lin T-Y, Maire M, Belongie S et al (2014) Microsoft COCO: common objects in context. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T (eds) Computer vision—ECCV 2014. Springer International Publishing, Cham, pp 740–755

Lin J, Guo X, Shao J et al (2016) A virtual reality platform for dynamic human-scene interaction. SIGGRAPH ASIA 2016 virtual reality meets physical reality: modelling and simulating virtual humans and environments. Association for Computing Machinery, New York

Little JJ, Verri A (1989) Analysis of differential and matching methods for optical flow. In: [1989] Proceedings. Workshop on Visual Motion. IEEE Comput. Soc. Press, pp. 173–180

Marín J, Vázquez D, Gerónimo D, López AM (2010) Learning appearance in virtual scenarios for pedestrian detection. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. https://doi.org/10.1109/CVPR.2010.5540218

Mayer N, Ilg E, Fischer P et al (2018) What makes good synthetic training data for learning disparity and optical flow estimation? Int J Comput Vis 126:942–960. https://doi.org/10.1007/s11263-018-1082-6

Mayer N, Ilg E, Hausser P, et al (2016) A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-Decem:4040–4048. https://doi.org/10.1109/CVPR.2016.438

McCormac J, Handa A, Leutenegger S, Davison AJ (2017) SceneNet RGB-D: Can 5M synthetic images beat generic imagenet pre-training on indoor segmentation? Proc IEEE Int Conf Comput Vis 2017-Octob:2697–2706. https://doi.org/10.1109/ICCV.2017.292

Mitash C, Bekris KE, Boularias A (2017) A self-supervised learning system for object detection using physics simulation and multi-view pose estimation. IEEE Int Conf Intell Robot Syst 2017-Septe:545–551. https://doi.org/10.1109/IROS.2017.8202206

Mnih V, Kavukcuoglu K, Silver D, et al (2013) Playing Atari with deep reinforcement learning. http://arxiv.org/abs/1312.5602

Moiseev B, Konev A, Chigorin A, Konushin A (2013) Evaluation of traffic sign recognition methods trained on synthetically generated data. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 8192 LNCS:576–583. https://doi.org/10.1007/978-3-319-02895-8_52

Movshovitz-Attias Y, Kanade T, Sheikh Y (2016) How useful is photo-realistic rendering for visual learning? Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 9915 LNCS:202–217. https://doi.org/10.1007/978-3-319-49409-8_18

Mueller M, Smith N, Ghanem B (2016) a benchmark and simulator for UAV tracking BT—computer vision—ECCV 2016. In: Leibe B, Matas J, Sebe N, Welling M (eds). Springer International Publishing, Cham, pp 445–461

Müller M, Casser V, Lahoud J et al (2018) Sim4CV: a photo-realistic simulator for computer vision applications. Int J Comput Vis 126:902–919. https://doi.org/10.1007/s11263-018-1073-7

Munea TL, Jembre YZ, Weldegebriel HT et al (2020) The progress of human pose estimation: a survey and taxonomy of models applied in 2D human pose estimation. IEEE Access 8:133330–133348. https://doi.org/10.1109/ACCESS.2020.3010248

Nanni L, Paci M, Brahnam S, Lumini A (2021) Comparison of different image data augmentation approaches. J Imaging. https://doi.org/10.3390/jimaging7120254

Nikolenko SI (2021) Synthetic data for deep learning. Springer International Publishing, Cham

Nowruzi FE, Kapoor P, Kolhatkar D, et al (2019) How much real data do we actually need: Analyzing object detection performance using synthetic and real data. https://doi.org/10.48550/arXiv.1907.07061

Papon J, Schoeler M (2015) Semantic pose using deep networks trained on synthetic RGB-D. Proc IEEE Int Conf Comput Vis 2015 Inter:774–782. https://doi.org/10.1109/ICCV.2015.95

Parker SP (2003) McGraw-Hill dictionary of scientific and technical terms, 6th edn. McGraw-Hill Education, New York

Patki N, Wedge R, Veeramachaneni K (2016) The synthetic data vault. In: 2016 IEEE international conference on data science and advanced analytics (DSAA). pp 399–410

Peng X, Sun B, Ali K, Saenko K (2015) Learning deep object detectors from 3D models. Proc IEEE Int Conf Comput Vis 2015 Inter:1278–1286. https://doi.org/10.1109/ICCV.2015.151

Pepik B, Stark M, Gehler P, Schiele B (2012) Teaching 3D geometry to deformable part models. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. https://doi.org/10.1109/CVPR.2012.6248075

Peris M, Martull S, Maki A, et al (2012) Towards a simulation driven stereo vision system. Proc—Int Conf Pattern Recognit 1038–1042

Pishchulin L, Jain A, Wojek C et al (2011) Learning people detection models from few training samples. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. https://doi.org/10.1109/CVPR.2011.5995574

Pomerleau DA (1989) Alvinn: an autonomous land vehicle in a neural network. Adv Neural Inf Process Syst 1:305–313

Prakash A, Boochoon S, Brophy M, et al (2019) Structured domain randomization: bridging the reality gap by context-aware synthetic data. Proc - IEEE Int Conf Robot Autom 2019-May:7249–7255. https://doi.org/10.1109/ICRA.2019.8794443

Qiu W, Yuille A (2016) UnrealCV: Connecting computer vision to unreal engine. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 9915 LNCS:909–916. https://doi.org/10.1007/978-3-319-49409-8_75

Queiroz R, Cohen M, Moreira JL et al (2010) Generating facial ground truth with synthetic faces. Proc—23rd SIBGRAPI conf graph patterns images. SIBGRAPI 2010:25–31. https://doi.org/10.1109/SIBGRAPI.2010.12

Ragheb H, Velastin S, Remagnino P, Ellis T (2008) ViHASi: Virtual human action silhouette data for the performance evaluation of silhouette-based action recognition methods. 2008 2nd ACM/IEEE Int Conf Distrib Smart Cameras, ICDSC 2008. https://doi.org/10.1109/ICDSC.2008.4635730

Richardson E, Sela M, Kimmel R (2016) 3D face reconstruction by learning from synthetic data. Proc - 2016 4th Int Conf 3D Vision, 3DV 2016 460–467. https://doi.org/10.1109/3DV.2016.56

Richter SR, Vineet V, Roth S, Koltun V (2016) Playing for data: Ground truth from computer games. Lect Notes Comput Sci (including Subser Lect Notes Artif Intell Lect Notes Bioinformatics) 9906 LNCS:102–118. https://doi.org/10.1007/978-3-319-46475-6_7

Richter SR, Hayder Z, Koltun V (2017) Playing for benchmarks. https://doi.org/10.48550/arXiv.1709.07322

Rivera-Rubio J, Alexiou I, Bharath AA (2015) Appearance-based indoor localization: a comparison of patch descriptor performance. Pattern Recognit Lett 66:109–117. https://doi.org/10.1016/j.patrec.2015.03.003

Ros G, Sellart L, Materzynska J, et al (2016) The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2016-Decem:3234–3243. https://doi.org/10.1109/CVPR.2016.352

Rozantsev A, Lepetit V, Fua P (2015) On rendering synthetic images for training an object detector. Comput Vis Image Underst 137:24–37. https://doi.org/10.1016/j.cviu.2014.12.006

Rubin DB (1993) Discussion of statistical disclosure limitation. J off Stat 9:461–468

Santhosh KK, Dogra DP, Roy PP (2020) Anomaly detection in road traffic using visual surveillance: a survey. ACM Comput Surv. https://doi.org/10.1145/3417989

Satkin S, Lin J, Hebert M (2012) Data-driven scene understanding from 3D models. BMVC 2012 - Electron Proc Br Mach Vis Conf 2012 1–11. https://doi.org/10.5244/C.26.128

Savva M, Kadian A, Maksymets O, et al (2019) Habitat: A platform for embodied AI research. Proc IEEE Int Conf Comput Vis 2019-Octob:9338–9346. https://doi.org/10.1109/ICCV.2019.00943

Saxena A, Driemeyer J, Kearns J, Ng AY (2007) Robotic grasping of novel objects. Adv Neural Inf Process Syst. https://doi.org/10.7551/mitpress/7503.003.0156

Shafaei A, Little JJ, Schmidt M (2016) Play and learn: using video games to train computer vision models. Br Mach Vis Conf 2016, BMVC 2016 2016-Septe:26.1–26.13. https://doi.org/10.5244/C.30.26

Shah S, Dey D, Lovett C, Kapoor A (2018) AirSim: high-fidelity visual and physical simulation for autonomous vehicles. 621–635. https://doi.org/10.1007/978-3-319-67361-5_40

Sharma S, Beierle C, D'Amico S (2018) Pose estimation for non-cooperative spacecraft rendezvous using convolutional neural networks. In: 2018 IEEE Aerospace Conference. pp 1–12

Solovev P, Aliev V, Ostyakov P, et al (2018) Learning state representations in complex systems with multimodal data. http://arxiv.org/abs/1811.11067

Song S, Yu F, Zeng A, et al (2017) Semantic scene completion from a single depth image. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua:190–198. https://doi.org/10.1109/CVPR.2017.28

De Souza CR, Gaidon A, Cabon Y, López AM (2017) Procedural generation of videos to train deep action recognition networks. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua: 2594–2604. https://doi.org/10.1109/CVPR.2017.278

Su H, Qi CR, Li Y, Guibas LJ (2015) Render for CNN: viewpoint estimation in images using CNNs trained with rendered 3D model views. Proc IEEE Int Conf Comput Vis 2015 Inter:2686–2694. https://doi.org/10.1109/ICCV.2015.308

Sun B, Saenko K (2014) From virtual to reality: fast adaptation of virtual object detectors to real domains. In: proceedings of the British machine vision conference 2014. British Machine Vision Association, pp 82.1–82.12

Tarel JP, Hautière N, Cord A et al (2010) Improved visibility of road scene images under heterogeneous fog. IEEE Intell Veh Symp Proc. https://doi.org/10.1109/IVS.2010.5548128

Taylor GR, Chosak AJ, Brewer PC (2007) OVVV: using virtual worlds to design and evaluate surveillance systems. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit. https://doi.org/10.1109/CVPR.2007.383518

Temel D, Chen M-H, AlRegib G (2019) Traffic sign detection under challenging conditions: a deeper look into performance variations and spectral characteristics. IEEE Trans Intell Transp Syst. https://doi.org/10.1109/tits.2019.2931429

Tian Y, Li X, Wang K, Wang FY (2018) Training and testing object detectors with virtual images. IEEE/CAA J Autom Sin 5:539–546. https://doi.org/10.1109/JAS.2017.7510841

Tobin J, Fong R, Ray A, et al (2017) Domain randomization for transferring deep neural networks from simulation to the real world. IEEE Int Conf Intell Robot Syst 2017-Septe:23–30. https://doi.org/10.1109/IROS.2017.8202133

Tosi F, Aleotti F, Ramirez PZ, et al (2020) Distilled semantics for comprehensive scene understanding from videos. In: 2020 IEEE/CVF conference on computer vision and pattern recognition (CVPR). IEEE, pp 4653–4664

Tremblay J, Prakash A, Acuna D, et al (2018a) Training deep networks with synthetic data: Bridging the reality gap by domain randomization. IEEE Comput Soc Conf Comput Vis Pattern Recognit Work 2018a-June:1082–1090

Tremblay J, To T, Birchfield S (2018b) Falling things: a synthetic dataset for 3D object detection and pose estimation. IEEE Comput Soc Conf Comput Vis Pattern Recognit Work 2018b-June:2119–2122

Tremblay J, To T, Sundaralingam B, et al (2018c) Deep object pose estimation for semantic robotic grasping of household objects. http://arxiv.org/abs/1809.10790

Tripathi S, Chandra S, Agrawal A, et al (2019) Learning to generate synthetic data via compositing. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2019-June:461–470. https://doi.org/10.1109/CVPR.2019.00055

Tsirikoglou A, Kronander J, Wrenninge M, Unger J (2017) Procedural modeling and physically based rendering for synthetic data generation in automotive applications. https://doi.org/10.48550/arXiv.1710.06270

Ubbens J, Cieslak M, Prusinkiewicz P, Stavness I (2018) The use of plant models in deep learning: an application to leaf counting in rosette plants. Plant Methods 14:1–10. https://doi.org/10.1186/s13007-018-0273-z

Vacavant A, Chateau T, Wilhelm A, Lequièvre L (2013) A benchmark dataset for outdoor foreground/background extraction. In: Park J-I, Kim J (eds) Computer vision—ACCV 2012 workshops. Springer, Berlin, Heidelberg, pp 291–300

Varol G, Romero J, Martin X, et al (2017) Learning from synthetic humans. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua:4627–4635. https://doi.org/10.1109/CVPR.2017.492

Vazquez D, Lopez AM, Marin J et al (2014) Virtual and real world adaptation for pedestrian detection. IEEE Trans Pattern Anal Mach Intell 36:797–809. https://doi.org/10.1109/TPAMI.2013.163

Veeravasarapu VSR, Hota RN, Rothkopf C, Visvanathan R (2015) Model validation for vision systems via graphics simulation. http://arxiv.org/abs/1512.01401

Veeravasarapu VSR, Rothkopf C, Ramesh V (2016) Model-driven simulations for deep convolutional neural networks. http://arxiv.org/abs/1605.09582

Veeravasarapu VSR, Rothkopf C, Visvanathan R (2017) Adversarially tuned scene generation. In: 2017 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 6441–6449. https://doi.org/10.1109/CVPR.2017.682

Wang K, Shi F, Wang W, et al (2019a) Synthetic data generation and adaption for object detection in smart vending machines. https://doi.org/10.48550/arXiv.1904.12294

Wang Q, Gao J, Lin W, Yuan Y (2019b) Learning from synthetic data for crowd counting in the wild. Proc IEEE Comput Soc Conf Comput Vis Pattern Recognit 2019b-June:8190–8199.

Wang Q, Zheng S, Yan Q, et al (2019c) IRS: A large synthetic indoor robotics stereo dataset for disparity and surface normal estimation. https://arxiv.org/abs/1912.09678

Wood E, Baltrušaitis T, Morency L-P, et al (2016) Learning an appearance-based gaze estimator from one million synthesised images. In: proceedings of the ninth biennial ACM symposium on eye tracking research & applications. Association for Computing Machinery, New York, NY, USA, pp 131–138

Wrenninge M, Unger J (2018) Synscapes: a photorealistic synthetic dataset for street scene parsing. http://arxiv.org/abs/1810.08705

Wu Z, Song S, Khosla A, et al (2014) 3D ShapeNets: a deep representation for volumetric shapes. http://arxiv.org/abs/1406.5670

Zhang Y, Wang C, Wang X et al (2021) FairMOT: on the fairness of detection and re-identification in multiple object tracking. Int J Comput Vis 129:3069–3087. https://doi.org/10.1007/s11263-021-01513-4

Zhang Y, Song S, Yumer E, et al (2017) Physically-based rendering for indoor scene understanding using convolutional neural networks. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua:5057–5065. https://doi.org/10.1109/CVPR.2017.537

Zhu Y, Mottaghi R, Kolve E, et al (2017) Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE international conference on robotics and automation (ICRA). pp 3357–3364

Zimmermann C, Brox T (2017) Learning to estimate 3D hand pose from single RGB images. Proc IEEE Int Conf Comput Vis 2017-Octob:4913–4921. https://doi.org/10.1109/ICCV.2017.525

Zioulis N, Karakottas A, Zarpalas D, et al (2019) Spherical view synthesis for self-supervised 360° depth estimation. Proc - 2019 Int Conf 3D Vision, 3DV 2019 690–699. https://doi.org/10.1109/3DV.2019.00081