Review and analysis of synthetic dataset generation methods and techniques for application in Computer Vision

Goran Paulin[1], Marina Ivašić-Kos[2]

[1] Kreativni odjel d.o.o., Rijeka, Croatia; gp@kreativni.hr; 0000-0002-6885-5393

[2] University of Rijeka, Department of Informatics, Rijeka, Croatia; marinai@inf.uniri.hr (corresponding author), 0000-0002-1940-5089

Abstract. Synthetic datasets, for which we propose the term synthsets, are not a novelty but became a necessity. Although they have been used in computer vision since 1989, solving the problem of collecting a sufficient amount of data for learning the neural network and offering the possibility of automatic annotation, significant development of methods and techniques for their generation belongs to the last decade. Nowadays, the question shifts from should you use them to how should you optimally create them. Motivated by the idea of discovering best practices for building optimal synthsets to represent dynamic environments (such as traffic, crowds and sports), this study provides an overview of existing synthsets in the computer vision domain, analyzing the methods and techniques of their 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 9 unique but potentially intertwined methods and reveals the general process of synthsets generation, consisting of 17 individual processes that you should follow and choose from, depending on the specific requirements of your future synthset.

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

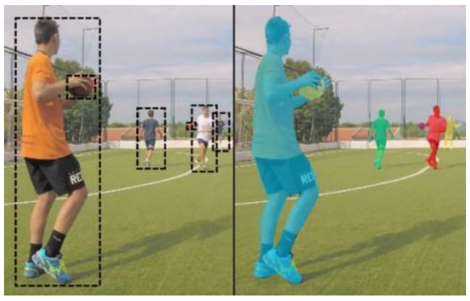

Fig. 1 Marking objects with a bounding box (left) and object segmentation marking pixels belonging to an individual object (right) (Burić et al. 2020)

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 learning the neural network and offers the possibility of automatic annotation, which remain the two primary reasons for its application. Synthetic data can be an unlimited source of data and can simulate situations we have not yet encountered. Also, it allows overcoming restrictions on the use of real data conditioned by respect for privacy or other regulations (Rubin 1993).

A 2016 study on the effectiveness of synthetic data (Patki et al. 2016) showed that in 70% of cases, using only synthetic data, the results achieved using real data could be reproduced. Also, 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. (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 learning while increasing the samples' diversity. Aware of the task, synthetic data must create examples that help improve the target neural network's performance. In doing so, they must be visually realistic to minimize the domain gap present between real and computer-generated images (which can vary between non-realistic and photorealistic) and thus improve generalization. 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 (Queiroz et al. 2010), pose estimation (Rosales et al. 2001), scene understanding (Satkin et al. 2012), action recognition (Ragheb et al. 2008), object tracking (Desurmont et al. 2006) and object classification (Carlucci et al. 2017). Synthetic data is predominantly used in the domains such as autonomous driving (Pomerleau 1989) and autonomous flying (Shah et al. 2018), surveillance (Taylor et al. 2007), and virtual reality (Lin et al. 2016). Outside the domain of computer vision, the application of synthetic data is important for neural programming and bioinformatics (Nikolenko 2019).

Motivated by the idea of *discovering best practices for building optimal synthsets to represent dynamic environments* (such as traffic, crowds and sports), the objective of this study is to provide an overview of existing synthsets in the computer vision domain and to analyze the methods and techniques of their generation.

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: Section 2 provides the overview of the field, presented chronologically, in order of introduction and development of individual methods and techniques for generating synthsets, allowing a better understanding of their mutual influence. Section 3 analyses detected methods and techniques, systematizing synthset generating methods, the general process of synthsets generation and individual processes in it. It also includes the representation of generation methods in computer vision tasks and domains. Section 4 explains the conclusion. Suggestions are given for the implementation of experimental research to confirm the effectiveness of selected methods.

2 Review of synthsets generation methods and techniques

The first known application of synthetic data for neural network 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 30x32 px resolution, representing the blue channel of the *RGB* camera, and the other in 8x32 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 field conditions". 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 (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 px in space and time. This 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 (1999), 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 160x182 px, used to detect and segment similar shapes in 2D sequences. He 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 learning 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 seconds each, at a resolution of 1,400x1,050 px. The GT is available for player positions and the positions and parameters of 7 static and 3 *pan-tilt-zoom (PTZ) cameras* that simulate the standard conditions of a television broadcast of a football match. Despite the faithful camera positions, this set's main weakness is the lack of any realistic noise and image distortion.

Taylor et al. (2007) 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 4 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 2x2 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 9 3D models of virtual characters and 20 action classes and rendered them at 640x480 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.

Saxena et al. (2007) generated synthetic images for supervised learning, manually annotating the correct robotic grip's location on 5 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 2,500 images, they randomized different properties of objects: colour, size, and text (on book covers), effectively introducing a *domain randomization (DR)* method that Tobin et al. (2017) will popularize in 2017. They pointed to the time-consuming preparation of 3D objects, noting that the problem can be solved using 3D objects available on the Internet, with minor modifications.

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 colour space* and *ambient occlusion* to approximate *global illumination*. GT is created by projecting

3D motion onto a 2D image, which is why the optical flow vector stored in a single pixel can represent the motion of more than one object, being is a weakness of this technique. The authors suggested that when creating future datasets for this purpose, *different materials*, lighting changes, *atmospheric effects* and *transparency* should be considered.

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

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. Significant degradation of performance is observed when the lighting changes, which the movement of shadows and reflections can explain due to the Sun's journey during 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, which allowed them to 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, due to feature statistics, behaves worst, but in combination with real, in all tested cases gives the best results.

Haltakov et al. (2013) used the open-source driving simulator VDrift, which allows them to create different traffic scenarios, similar to real ones, to generate a synthset for multi-class image segmentation in the domain of autonomous driving. To render *segmentation annotations*, they used uniform colouring of textures of objects belonging to different classes. When rendering such images, they excluded lighting, shadows, reflections, antialiasing and *mipmapping* to get the correct colour 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, which consists of images of the trajectory through two different scenes (living room and office), targeting photorealism, they paid special attention to light phenomena present in real images: *specular reflections*, *shadows* and *colour 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 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*. To eliminate it, they developed a V-AYLA integrated framework, which uses 10% times less data from the real set than from the synthset to *adapt the domain*, achieving just as good detection results as when using a manually annotated real set for both learning and testing.

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 learning 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.

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 8 different scenes. The set is intended for tracking and segmentation in crowd analysis. Since they use between 200 and 2,000 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.

Hattori et al. (2015), in the domain of video surveillance, built a massive (2.5M 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 3 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 impact of various factors (geometry, appearance, lighting, physics, environment, camera, rendering parameters, etc.) 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.

Su et al. (2015) built a large synthset (over 2.4M images in 12 classes) that they combined with the real set (12K 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, 8 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 on-line 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 learning, and RenderScene (1,800 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 learning 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 1M 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 (5M 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 (*colour 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 seconds 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 1,000 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 colours 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 1,574 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 960x720 px, they are used for training at a resolution of 180x120 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 the 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 colour, 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*.

Unlike McCormac et al. (2017), who randomly sampled scenes from SceneNet and objects from ShapeNet, Jiang et al. (2017) 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 encounter 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.

Tobin et al. (2017) were the first to explore *domain randomization* systematically. They concluded that with enough variability in the simulator, the real world can 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 1,000 different textures corresponds to the performance of a set of only 1,000 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 (6.5M 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 55K images. They also found that increasing clothing variation positively affects model performance, which speaks in favour of domain randomization (Tobin et al. 2017). For mo-cap, the authors recommended matching the distribution of the target set.

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 Handpose 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 made sure that background images did not contain people because their presence negatively affects 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 3 *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%.

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 320x180 px to reduce latency to a minimum, which is essential for end-to-end training.

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 5 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.

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 300K images (66.64 AUC), an exclusively photorealistic set with 600K 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 120K 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 4 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 behaviour 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 learning phase prefers simpler data and the later more complex.

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 3 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 50K images (Tobin et al. 2017), SDR achieves saturation with 10K 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.

The use of SDR instead of DR is supported by the results of Dvornik et al. (2018), who, in the domain of object detection, conducted experiments by placing objects cut from 2D images at different positions on 2D backgrounds. They concluded that detection accuracy decreases significantly when objects are located at *unrealistic positions*. This points to the fact that the *visual context becomes a crucial information* source whenever visual information is corrupted, ambiguous, or incomplete.

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 depths 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 tests 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 learning 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; Johnson-Roberson et al. 2017; Richter 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 3,995, and an average of 501 are visible. To adapt the domain, they introduce 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 *colour 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 colour 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 8 V100 GPUs, producing 3,438 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 controlled 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 from (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 9 *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.

3 Analysis

The largest number of synthsets (20% in the analyzed works) was created in the domain of autonomous driving and it should be borne in mind that a significant part of the methods and techniques of their generation is optimized for this application.

3.1 The general process of synthsets generation

From the analyzed papers, the general process of synthsets generation, presented in Fig. 2, can be extracted.

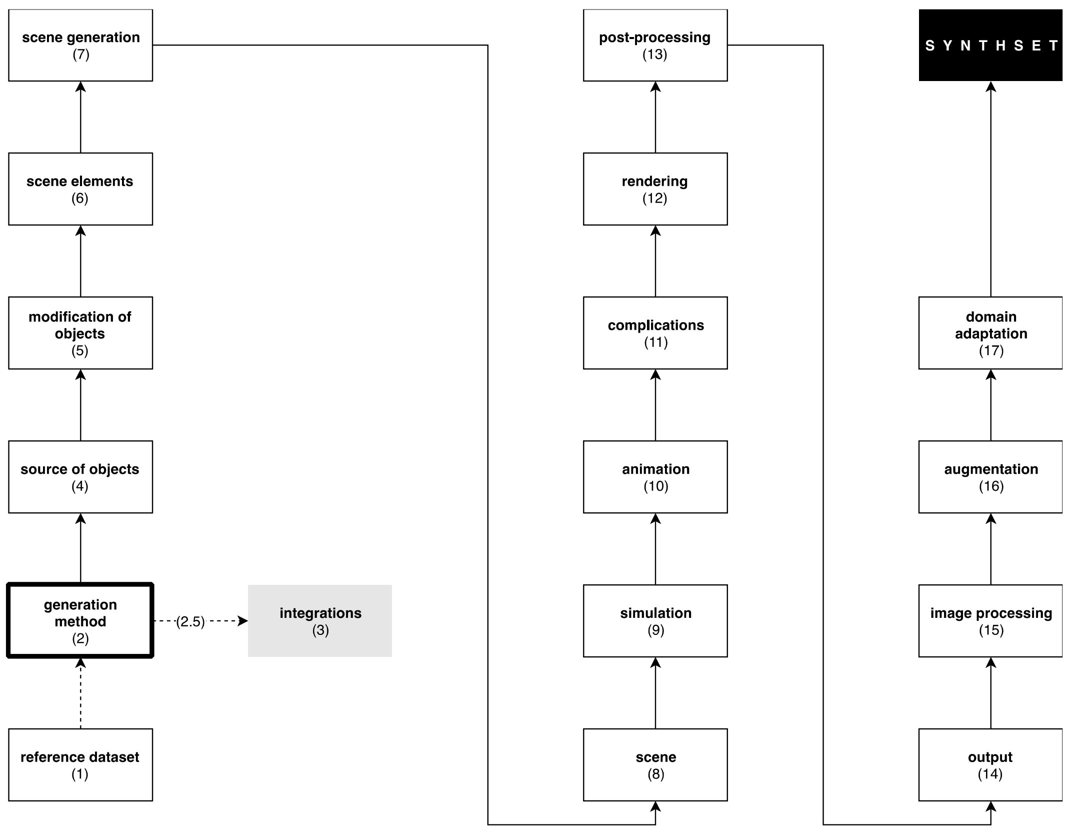

Fig. 2 The general process of synthsets generation in the analyzed papers (N=165)

The general process consists of *17 individual 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 synthesised data by integrating generation methods with one of the machine learning platforms. As such, it is not the subject of interest of this article.

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

3.2 Individual processes

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).

The first mandatory process is the selection of the synthset generation method (2). In the analyzed works, *9 different methods* (2.1 - 2.9) were identified, shown in Fig. 3.

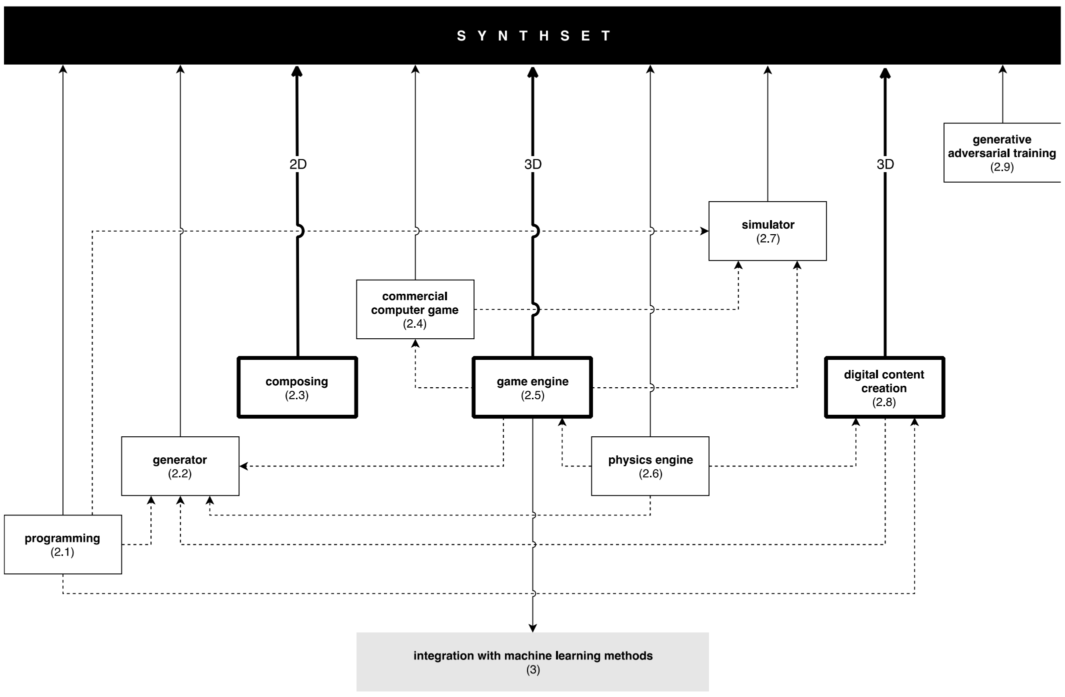
S Y N T H S E T
2D
3D
3D
generative adversarial training (2.9)
simulator (2.7)
commercial computer game (2.4)
composing (2.3)
game engine (2.5)
digital content creation (2.8)
generator (2.2)
physics engine (2.6)
programming (2.1)
integration with machine learning methods (3)

Fig. 3 Synthset generation methods and their mutual influence (best viewed in digital format)

Some methods are optionally contained in more complex methods, which is indicated by a dashed line in Fig. 3. Game engine method (2.5) allows direct integration with machine learning platforms (3), and the highlighted methods (2.3, 2.5 and 2.8) are shown in detail on Fig. 4 - The "2D" and "3D" labels refer to the type of scene (2D or 3D) that is created.

In the *programming method* (2.1) (Desurmont et al. 2006) the synthset is created algorithmically, by direct programming of the output. The *method of using the generator* (2.2) (Hamarneh 1999) 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. The *composing method* (2.3) (Su et al. 2015) treats the scene as a set of 2D layers laid on top of each other using a minimum of 2 layers (one each for the image's background and foreground). The possibility of using the *commercial computer game method* (2.4) (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* (2.5) (Rivera-Rubio et al. 2015), which is, when used as a production tool, also a distinct method that can produce a synthset without first making a game. An integral part of most game engines is the *physics engine* (2.6) (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. The method of *using the simulator* (2.7) (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 most often generate output in real-time. The *digital content creation method* (2.8) (Kaneva et al. 2011) relies on tools that often contain a physics engine and enable automation by programming. 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 Fig. 4.

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 1999) or manually (4.2.3) (Peris et al. 2012) generated, built by converting OSM maps (4.2.4) (Tian et al. 2018) to 3D geometry, generated as L-System (4.2.5) (Ubbens et al. 2018) or using photogrammetry (4.2.7) (Jung 2019) and used as a point cloud, obtained by LiDAR (4.2.6) (Li et al. 2019), which can also be converted to 3D geometry.

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 scenes (5.1.5) (Li et al. 2019).

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).

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 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).

The first convergence of 2D and 3D pathways during synthset generation is present in the process of *adding complications* (11). Complications are most often introduced as some form of occlusion (11.1) and as missing frames (11.2) in video sequences (Hamarneh 1999).

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 learning model (12.5) (Mnih et al. 2013).

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).

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 3 processes (15 - 17).

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).

If the basic synthset needs to be expanded by *augmentation* (16) (Hamarneh 1999), 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 colour in the depth map with a random colour away from the centre 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 colour channel inversion (16.17).

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, using a framework (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).

The course of these processes (for composing, game engine and digital content creation methods) is shown in Fig. 4.

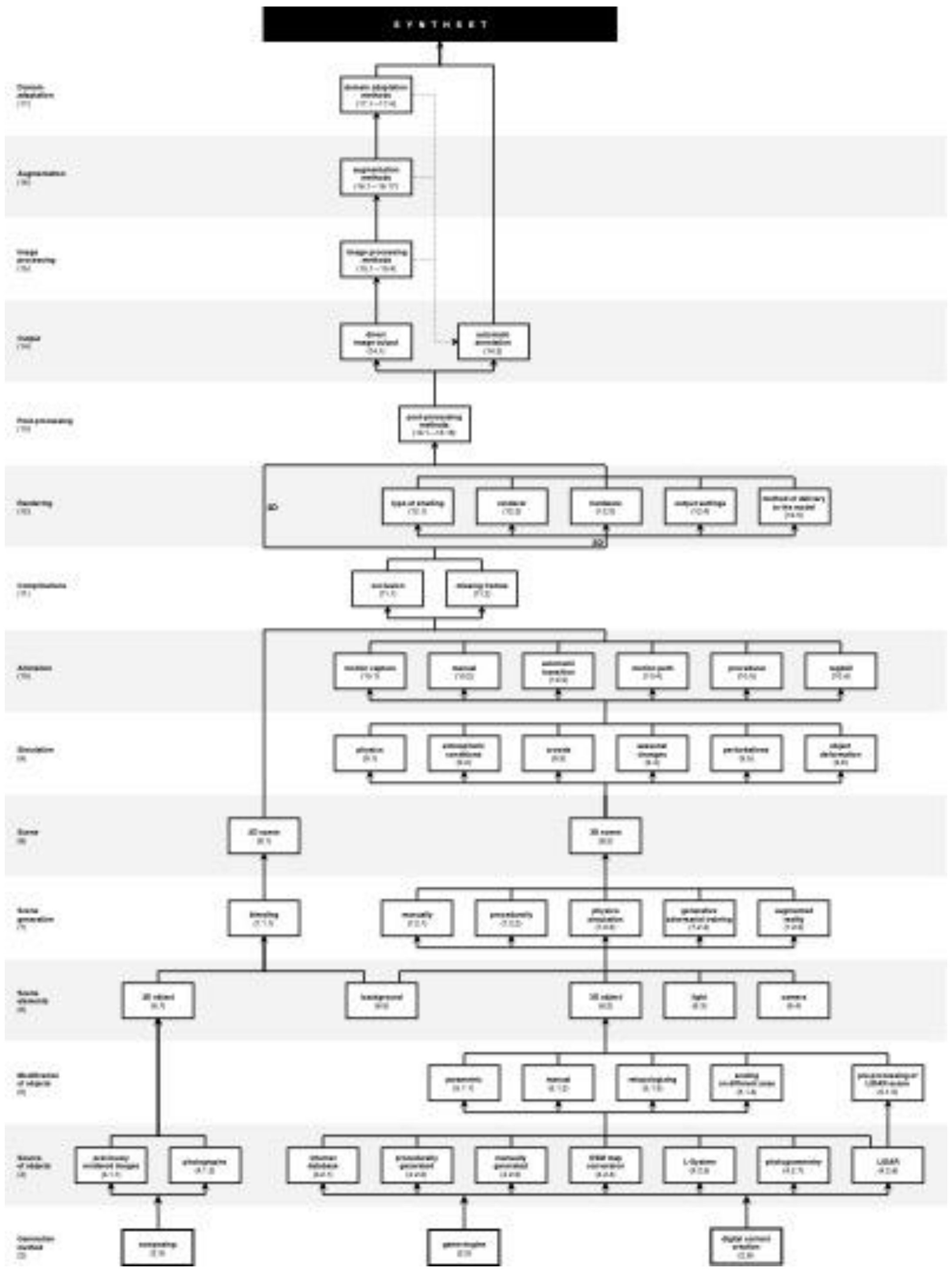

Fig 4. Synthset generation process for composing, game engine and digital content creation methods (best viewed in digital format)

3.3 Representation of generation methods in computer vision tasks and domains

The analyzed works include 85 produced synthsets, 12 simulators and 6 generators, all publicly available and named.

Observing the representation of generation methods in individual computer vision tasks and domains (Table 1), we can conclude that the most frequently used method is digital content creation (30.1%). It is the favoured method in the optical flow and scene reconstruction tasks, where it is also the only method used. This method's popularity can be explained by the flexibility that digital content creation tools provide in 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 method in terms of frequency of use is the game engine (19.4%). In addition to the commercial computer game method, the game engine method is also one of the preferred methods in autonomous driving, and it is the pose estimation's preferred method. Its key advantage, welcome in both fields, is the possibility of real-time performance, but at the expense of photorealism. The method of using the simulator (15.5%) is preferred in robot navigation, as well as in autonomous flying, where it benefits from physics simulation.

The low representation (1.18%) of composing 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 composing 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 9.7% of papers in which the authors did not specify the method of generating their own synthsets.

Table 1 Representation of generation methods in individual computer vision tasks and domains

generation methods / tasks and domains (alphabetically)	programming	generator	composing	commercial computer game	game engine	physics engine	simulator	digital content creation (DCC)	DCC + game engine + simulator	not specified	Σ
action recognition							1	2			3
autonomous driving	1			6	6		4	3	1	2	23
autonomous flying					1		3			1	5
gaze estimation					1						1
image features evaluation								2			2
light fields analysis								1			1
object classification					1			2		1	4
object detection	1	1			1		1	3			7
object reconstruction	1							1			2
object tracking	1				1						2
optical flow			1					4		1	6
pose estimation	1				4			3		2	10
robot navigation	1				2		6			1	10
scene reconstruction								4			4
scene understanding	1	1				1		1		1	5
segmentation	2	2						2			6
stereo disparity		1			2			1			4
surveillance		1		2	1		1	2			7
texture generation										1	1
Σ	**9**	**6**	**1**	**8**	**20**	**1**	**16**	**31**	**1**	**10**	103

4 Conclusion

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 learning process.

In this article, the past and current synthset generation practices are examined, the synthset generation process is systematized, and 9 different synthset generation methods are identified.

Of the 9 identified generation methods, 3 compete for building optimal synthsets to represent dynamic environments (such as traffic, crowds and sports): *digital content creation*, *game engine* and *composing*. 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), composing 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 process.

It is not recommended to use programming as a generation method, but only for the automation of individual 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 we encounter known 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 due to its complexity and problematic annotations (Tian et al. 2018).

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 composing 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 and/or the use of the game engine, and 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.

5 References

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

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 2800–2810. 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

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 (2020) Object Detection Using Synthesized Data

Butler DJ, Wulff J, Stanley GB, Black MJ (2012) A Naturalistic Open Source Movie for Optical Flow Evaluation. pp 611–625

Carlucci FM, Russo P, Caputo B (2017) A deep representation for depth images from synthetic data. Proc - IEEE Int Conf Robot Autom 1362–1369. https://doi.org/10.1109/ICRA.2017.7989162

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

Chociej M, Welinder P, Weng L (2019) ORRB -- OpenAI Remote Rendering Backend

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

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

Deschaintre V, Aittala M, Durand F, et al (2018) Single-image SVBRDF capture with a rendering-aware deep network. ACM Trans Graph 37:. 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. 1–16

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

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

Fonder M, Droogenbroeck M (2019) Mid-Air: A Multi-Modal Dataset for Extremely Low Altitude Drone Flights

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

Georgakis G, Mousavian A, Berg AC, Košecká J (2017) Synthesizing training data for object detection in indoor scenes. Robot Sci Syst 13:. https://doi.org/10.15607/rss.2017.xiii.043

Girdhar R, Ramanan D (2019) CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. 1–16

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 Berlin Heidelberg, Berlin, Heidelberg, pp 323–332

Hamarneh G (1999) Deformable Spatio-Temporal Shape Modeling

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 1524–1531. 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 8:. 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

Jiang C, Zhu Y, Qi S, et al (2017) Configurable, Photorealistic Image Rendering and Ground Truth Synthesis by Sampling Stochastic Grammars Representing Indoor Scenes

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 746–753. https://doi.org/10.1109/ICRA.2017.7989092

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

Jung J (2019) Synthetic data generation for camera-based perception

Kaneva B, Torralba A, Freeman WT (2011) Evaluation of image features using a photorealistic virtual world. Proc IEEE Int Conf Comput Vis 2282–2289. 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

Koenig N, Howard A (2004) Design and use paradigms for Gazebo, an open-source multi-robot simulator. 2004 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 Vision, WACV 2020 1803–1812. https://doi.org/10.1109/WACV45572.2020.9093339

Lange D (2020) Synthetic Data: A Scalable Way to Train Perception Systems

Li W, Pan C, Zhang R, et al (2019) AADS: Augmented Autonomous Driving Simulation using Data-driven Algorithms

Lin J, Guo X, Shao J, et al (2016) A Virtual Reality Platform for Dynamic Human-Scene Interaction. In: SIGGRAPH ASIA 2016 Virtual Reality Meets Physical Reality: Modelling and Simulating Virtual Humans and Environments. Association for Computing Machinery, New York, NY, USA

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 137–144. 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

Mnih V, Kavukcuoglu K, Silver D, et al (2013) Playing Atari with Deep Reinforcement Learning. 1–9

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

Nikolenko SI (2019) Synthetic Data for Deep Learning. 1–156

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

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 ed. 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 3362–3369. 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 1473–1480. https://doi.org/10.1109/CVPR.2011.5995574

Pomerleau D a (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

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, Hayder Z, Koltun V (2017) Playing for Benchmarks

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

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

Rosales R, Athitsos V, Sigal L, Sclaroff S (2001) 3D hand pose reconstruction using specialized mappings. Proc IEEE Int Conf Comput Vis 1:378–387. https://doi.org/10.1109/ICCV.2001.937543

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

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 1209–1216. 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. 1–10

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

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

Tarel JP, Hautière N, Cord A, et al (2010) Improved visibility of road scene images under heterogeneous fog. IEEE Intell Veh Symp Proc 478–485. 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 1–11. 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

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 2018-June:1082–1090. https://doi.org/10.1109/CVPRW.2018.00143

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 2018-June:2119–2122. https://doi.org/10.1109/CVPRW.2018.00275

Tremblay J, To T, Sundaralingam B, et al (2018c) Deep Object Pose Estimation for Semantic Robotic Grasping of Household Objects. 1–11

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

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. 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 adaptationfor 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

Veeravasarapu VSR, Rothkopf C, Ramesh V (2016) Model-driven Simulations for Deep Convolutional Neural Networks. 1–10

Veeravasarapu VSR, Rothkopf C, Visvanathan R (2017) Adversarially tuned scene generation. Proc - 30th IEEE Conf Comput Vis Pattern Recognition, CVPR 2017 2017-Janua: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

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 2019-June:8190–8199. https://doi.org/10.1109/CVPR.2019.00839

Wang Q, Zheng S, Yan Q, et al (2019c) IRS: A Large Synthetic Indoor Robotics Stereo Dataset for Disparity and Surface Normal Estimation

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

Wu Z, Song S, Khosla A, et al (2014) 3D ShapeNets for 2.5D Object Recognition and Next-Best-View Prediction

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

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