BlendGen: A Blender Add-on for General RGB-D Databases Generation

Cedric Maron, Virginie Fresse, Karynn Morand, Hubert Konik

International Journal of Computer Information Systems and Industrial Management Applications.
ISSN 2150-7988 Volume 15 (2023) pp. 332-340

Received: 03 November 2022; Accepted: 15 May, 2023; Published: 23 June, 2023

BlendGen: A Blender Add-on for General RGB-D Databases Generation

Cedric Maron[1,2], Virginie Fresse[1], Karynn Morand[2] and Hubert Konik[1]

[1] Hubert Curien Laboratory, Jean-Monnet University,
18 Rue Professeur Benoît Lauras Bâtiment F, 42000 Saint-Étienne, France
cedric.maron@univ-st-etienne.fr
virginie.fresse@univ-st-etienne.fr
hubert.konik@univ-st-etienne.fr

[2] SEGULA Technologies
1 Rue des Combats du 24 Août 1944, 69200 Venissieux, France
cedric.maron@segula.fr
karynn.morand@segula.fr

***Abstract*: With the growing usage of convolutional neural networks in image classification[1], [2], detection and segmentation tasks, the need for image databases has also increased. Image database creation from real scenes is a time-consuming process. It requires capturing several thousand images of objects that are not necessarily easy to access in various contexts. In addition, the labeling process also takes a lot of time and human labeling precision is far from perfect. With time, numerous image databases have been made available. However, these databases rarely match new application requirements in terms of image resolution, labeling quality, object specificity and quantity. They can also lack information such as depth images. Creating synthetic image databases using database generation tools is a solution that makes image database creation easier and faster. In this paper, a new database generation tool named BlendGen is proposed. Database generation tools enable to create image databases with various image resolutions, perfect labeling quality, highly specific objects and high context variety. However, the existing database generation tools require users to use APIs to set up scenes and database parameters. The APIs usage can become heavy and less convenient for scene setups when the camera and numerous objects are following complex trajectories. BlendGen on the other hand is a Blender[3] add-on that enables RGB-D image database generation via a complete and intuitive graphical user interface (GUI). BlendGen GUI enables the creation of whole synthetic RGB-D databases including complex scenes, camera movements and lighting environments.**

***Keywords*:** Blender, Image Database Generation, RGB-D images database, CGI

I. Introduction

Databases are a key element of applications based on neural networks. Image databases are used to train and test neural networks to perform different tasks such as classification, detection, or segmentation.

The creation of an image database from real scenes is a very time-consuming process. First, it requires to have access to the objects which is not always easy or possible. For example, in the industry, it is not always possible to interact with a production line to acquire an image database. Secondly, it is necessary to acquire images with various contexts such as various object positions, rotations, textures, sizes and background environments. Finally, it requires labeling the images by hand. The human labeling precision is enough to classify images, but it becomes almost impossible for a human to label pixel-accurately detection and segmentation database.

With time numerous image databases have been made available such as ImageNet[4] or Cityscapes[5]. The problem is that these existing databases are generally not adapted for new application needs. The image resolution of these databases is often low. For example, the image resolutions of the COCO[6] and KITTI[7] databases are respectively 640x480p and 1,382x512p, therefore they are not adapted for applications processing high-resolution images such as 2K, 4K and more. The variety of objects present in available databases is usually wide. For example, the dataset PASCAL-Context[8] and NYU-Depth V2[9] contain respectively 460 and 1000 classes which means only a fraction of these databases can be useful for applications requiring to detect only few classes. The context variety is also either too narrow or too wide in terms of background environment, object positions, rotations and textures. The label precision is also far from perfect and the acquisition precision for images other than RGB such as depth images is usually below application requirements. For example, in the NYU-Depth V2 database, depth images were acquired using Microsoft Kinect which leaves blind spots on depth images.

Synthetic image database generation tools have been created to address the previously listed problems. These tools enable synthetic image database generation from 3D models addressing the problem of access to real objects. They make it possible to choose an image resolution that suits application needs. They enable perfect labeling precision. They also enable full control of context variety in terms of camera/object positions, rotations, textures, background

environments and lighting conditions.

However, existing image database generation tools require users to use APIs which can become heavy to create complex scenes. Complex scenes refer to having the camera and objects following complex paths combined with changing textures, background environments and lighting conditions. Moreover, some tools partially lack interesting functionalities such as camera path following, point of interest tracking, depth acquisition range control and the use of 360° images as background environment.

Our contribution is the creation of BlendGen, a Blender add-on that provides fast and easy RGB, Depth and RGB-D labeled databases generation via a simple graphical user interface. The add-on integrates multiple features designed to assist users in realistic database creation.

This paper presents BlendGen starting with its design principle, its interface and then its workflow. Finally, a concrete BlendGen use case to generate a high-resolution bus back-end image database is presented.

II. Related Work

The largest image databases became the benchmark for general classification, recognition and segmentation applications. For example, ImageNet became one of the benchmarks for classification models[10]. NYU-Depth V2, SUN RGB-D[11] and Cityscapes became benchmarks for object recognition and object segmentation models[12]–[14]. However, these large databases are too general and therefore, they cannot respond to the needs of specific applications in terms of image resolution, presence of specific objects in the database, labeling quality (hand-labeled databases are not 100% accurate), depth acquisition quality (sometimes depth is acquired using poor acquisition systems). Using database generation tools is a solution to generate high-quality databases for dedicated applications.

Database generation tools can be separated into two categories. The application-specific ones are dedicated to generating one specific database type such as a small electric motor database [15] and the general ones can generate databases based on all kinds of objects like cars, human body parts, animals, furniture, buildings and so on. Database generation tools are usually made using existing graphical computing frameworks[16]. Others use 3D computer graphics engines such as Unity [17], Unreal Engine [18] or Blender. Blender is often used since it is an open-source software with an API that allows to control its interface. Blender also continuously improves thanks to the help of its highly invested community.

However, most existing database creation tools require APIs usage to configure scenes and database generation parameters. APIs usage can become heavy when the scene complexity increases. Objects can be moving while the camera follows a given trajectory and, at the same time, tracks a point of interest that is also moving. For example, a camera attached to a robotic welding arm that is performing complex movements around a car moving along an assembly line with parts being assembled as it goes.

BlendGen is an alternative solution to the existing image database generation tools. BlendGen generates image databases directly through the powerful Blender GUI. This enables quick and intuitive scene setup, which enables faster database generation. BlendGen's GUI also integrates multiple tools to ease the configuration of realistic scenes. Finally, BlendGen enables image rendering using either ray tracing or rasterization rendering methods. Rasterization is several times faster than ray tracing, but ray tracing enables to get more photorealistic images. Most existing tools enable to use only one rendering method as presented in Table 1.

Name	Rendering methods		API	GUI
	Rasterization	Ray tracing		
BlenderProc [19]		x	x	
Blendtorch [20]	x		x	
ZPY [21]		x	x	
Kubric [22]		x	x	
BlendGen	x	x	x	x

Table 1. Rasterization/Ray-tracing : the majority of the existing tools use either one or the other; API: all solutions use API to generate databases, BlendGen can also be used using the Blender API named Bpy; GUI: Only BlendGen enables the use a GUI to setup scenes and database generation parameters.

BlendGen GUI tools have been compared to other existing image database generation tools under 3 tables which resume depth functionalities, camera functionalities and objects/background functionalities.

Concerning depth functionalities, most existing tools enable depth acquisition but none of them enables to modify depth acquisition resolution nor enables different depth acquisition methods usage as presented in Table 2.

Name	Depth		
	Depth acquisition	changeable depth acquisition resolution	Different depth acquisition methods
BlenderProc	x		
Blendtorch			
ZPY	x		
Kubric	x		
BlendGen	x	x	x

Table 2. Depth functionalities available in the different solutions

For the camera functionalities, most tools enable object/point of interest tracking. Only one existing tool enables camera path following and another enables random camera position and rotation in given ranges while BlendGen enables all three functionalities as presented in Table 3.

Name	Camera		
	Object/Point of Interest tracking	Camera path following	Random camera pos/rot in given ranges

BlenderProc	x	x	
Blendtorch			
ZPY	x		x
Kubric	x		
BlendGen	x	x	x

Table 3. Camera functionalities available in the different existing tools

Finally, only one existing tool enables random object position and rotation in given ranges. Two existing tools enable the use of 360° images as background environments. Only BlendGen and ZPY enable these two functionalities as presented in Table 4.

Name	Random object pos/rot in given ranges	360° image as background environment
BlenderProc		
Blendtorch		
ZPY	x	x
Kubric		x
BlendGen	x	x

Table 4. Object positioning and background functionalities available in the different existing tools

III. BlendGen architecture

BlendGen architecture is detailed starting with its design principle and motivations. Then its user interface is detailed. Finally, the general workflow to generate a database using BlendGen will be presented.

A. Motivation and design principle

The motivation behind the BlendGen creation was to create a tool that enables RGB-D database generation efficiently and intuitively.

Quick installation BlendGen is a Blender add-on, therefore the BlendGen installation only requires installing Blender and adding Blender to the used add-ons list. Blender is a free and open-source 3D creation suite. It is one of the most popular applications in the world for 3D artists.

Simplicity. BlendGen GUI enables quick and intuitive database generation. It makes possible for users to set up every key aspect of RGB-D image database generation by only using the BlendGen GUI. Its tools enable to quickly set up background environment, complex objects and camera positioning, depth acquisition parameters as well as image-related parameters.

Rendering flexibility. BlendGen enables to use either ray tracing or rasterization render engine. Ray tracing has the advantage to produce highly realistic images but takes a significant amount of time to render. Rasterization rendering quality is often lower than ray-tracing rendering, but it is a good compromise between quality and rendering speed.

B. Global user interface architecture

The global user interface architecture depicted in Figure 1 is composed of 4 blocks. The first block, *General Parameters*, contains the high-level database parameters such as the image resolution, the number of images to generate and the types of images to generate (RGB, depth, label).

The second block, *Background Properties*, makes it possible to add a 360° background image that will be used to create a photorealistic background environment.

The third block, *Camera Properties* enables the selection of two different depth acquisition methods and defining a depth acquisition range. The camera positioning can be controlled in this block by either positioning it in random ranges or making it follow complex trajectories and track objects.

Finally, the last block named *Objects Properties* configures all the object parameters such as their positions, their rotations, their label values as well as their visibilities during the different rendering stages (RGB, depth, label).

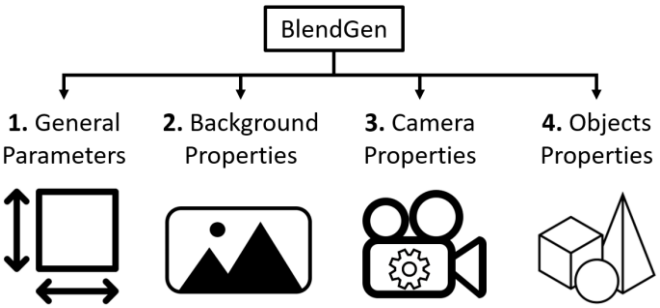

Figure 1. BlendGen global architecture

1) General parameters

The *General parameters* block controls the high-level parameters relative to the database generation. As depicted in Figure 2, it is possible to modify/change the image resolution, the number of images to generate and the types of images to generate (RGB, depth, label). Two generation types are proposed, *Random* and *Animation*. The *Random* generation type gives the possibility to set the camera and objects using random position and rotation values inside given ranges. The *Animation* generation type uses the animation system present in Blender. This second type enables the creation of more complex object and camera trajectories.

The *General parameter* block also gives the possibility to choose between two rendering engines for the RGB images. The first rendering engine named *Cycles* uses raytracing to render images. Raytracing produces photorealistic results but takes a good amount of time to render. The second rendering engine named *Eevee* uses rasterization to render images. Rasterization is a way to convert the 3D scene into a 2D image that is faster than raytracing but produces less photorealistic results.

All the images generated are saved in the .PNG format with a 0% compression rate. The RGB images are saved in an 8bits RGB format. The depth and label images are saved in a 16bits grayscale format.

Finally, there are different paths for the different image types and the metadata file to be saved. The metadata file contains information such as object label values, the generated data number, the camera position/rotation, the 360° background image view angle if there is one. Objects positions/ rotations are also provided.

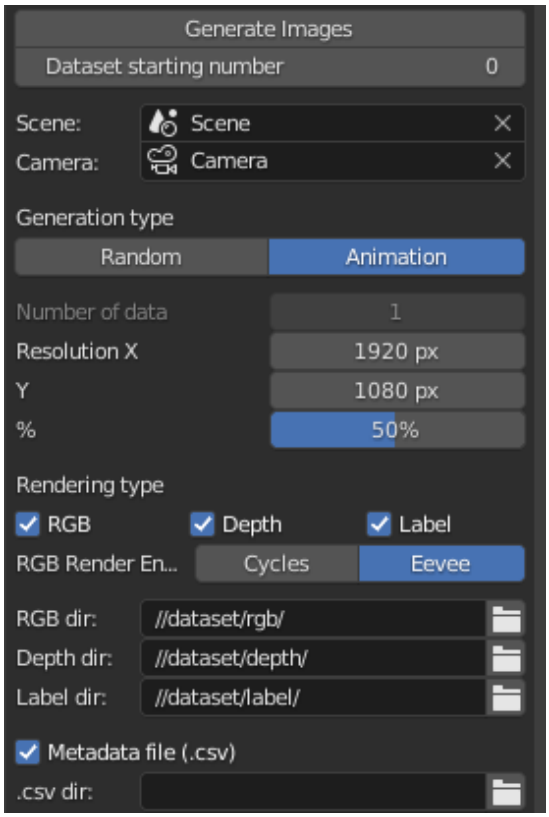

Figure 2. General parameters properties GUI

2) Background properties

The *Background properties* block controls parameters related to the 360° background image use. In addition to obtaining a photorealistic background, background image use simplifies the background environment creation. Moreover, Blender calculates the light emitted by the background image, avoiding manual set up numerous lights to illuminate the objects in the scene. Nevertheless, BlendGen can be used without using background images by adding 3D models for the background and setting up a light environment.

With the block Background properties, the user can select or not the use of a background image. If the user chooses not to use a background image, only the background label value variable is accessible. In case the user wishes to use a background image, all the parameters are accessible as depicted in Figure 3.

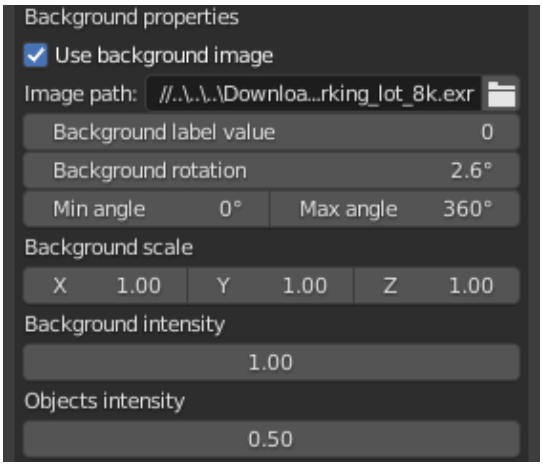

Figure 3. Background properties GUI

To obtain the best possible rendering, it is advisable to choose a background image resolution that is adapted to the generated images resolution. Since the images rendering is done using a small background image area, it is necessary that the background image has a resolution several times higher than the resolution generated images resolution If the background image resolution is too low, the background in the generated image will be pixelated.

Once the background image is chosen, the user needs to add the path to the background image. Then it is possible to choose the label value associated with the background image. It is also possible to rotate the background image around the scene vertical axis using the variable named *Background rotation*. The variables *min angle* and *max angle* indicate the range in which the background rotation will be set randomly during the generation database generation. The vector *Background scale* allows to modify the background image proportion.

Finally, it is possible to modify separately the backgrounds light intensity and the light intensity received by scene objects. This gives to the user the possibility to perform fine tuning to obtain the most realistic light environment. The *Objects intensity* is useful to create a match between the light generated by the background and the light received by the object. The *Background intensity* can be used to create various light environment conditions. Figure 4 shows different *Object intensity* values using a fixed *Background intensity*. Figure 5 shows different *Background intensity* values using for each a relatively adapted *Objet intensity*.

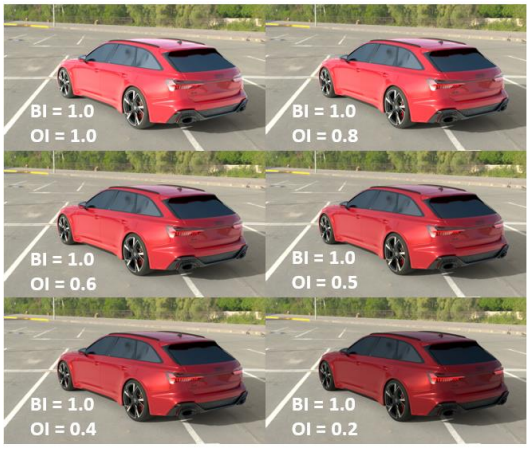

Figure 4. Different object intensities (OI) with a fixed background intensity (BI)

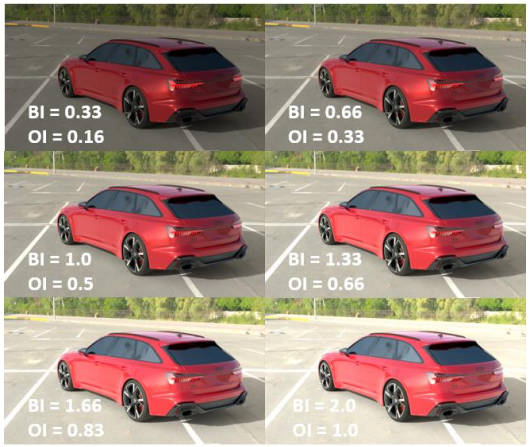

Figure 5. Different background intensities (BI) with adapted object intensity (OI)

1) Camera properties

The block *Camera Properties* controls the depth acquisition and the camera positions/rotations. Two depth acquisition methods are available. The first method is called *Front-Axis.* It acquires the depth for each camera pixel by calculating the distance between the scene 3D points seen by the camera and their projection on the camera plane. The Figure 6 depicts the method schema and the Figure 7 corresponds to its application in Blender.

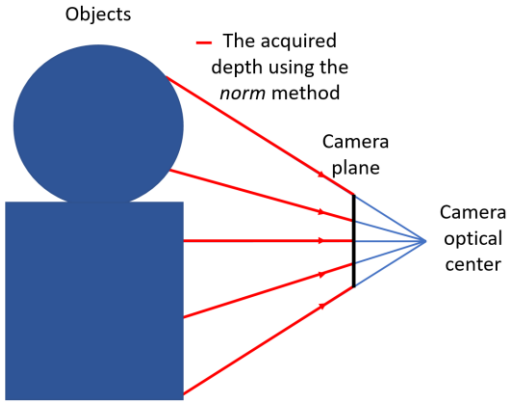

Figure 6 Depth acquisition scheme using the front-axis method

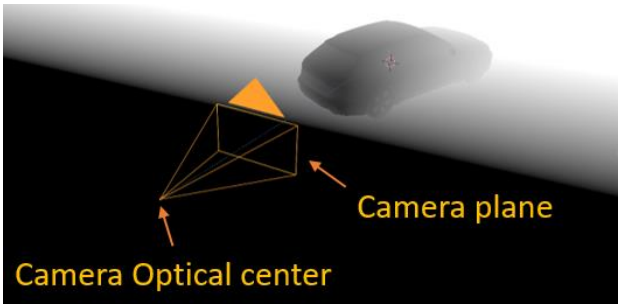

Figure 7. Depth acquisition using the camera front-axis

The second method is called *Norm.* It acquires the depth for each camera pixel by calculating the distance between the scene 3D points and the intersection with the camera plane. Figure 8 depicts the method schema and the Figure 9 corresponds to its application in Blender.

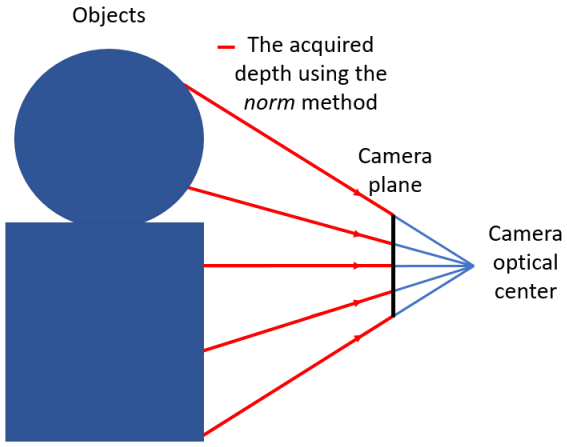

Figure 8 Depth acquisition scheme using the norm method

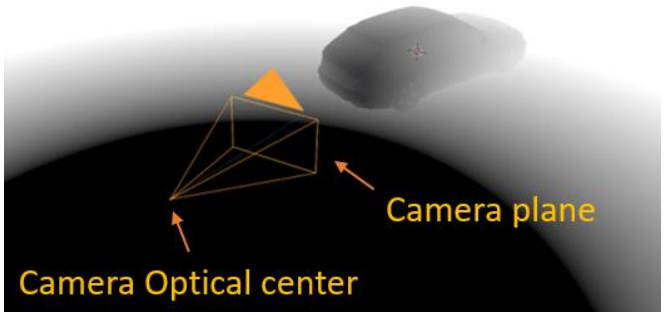

Figure 9. Depth acquisition using the norm between objects and the camera

It is also possible to define the depth acquisition range. A resolution indication is given in meters to help the user to select an acquisition range that suits his constraints.

For the camera position and rotation control, different options are available depending on the generation type chosen in the *General parameters* block. In case the user chooses *Random* as generation type, it is possible to define for the three scene axes X, Y, Z, a position and rotation range as depicted in Figure 10 (X and Y correspond to scene horizontal plane and the Z axis is the vertical axis). Regarding the rotation, it is also possible to choose to track an object present in the scene. The object tracking allows to make sure that the tracked object is always in the image center. The buttons *Copy Camera Pos* and *Copy Camera Rot* are used to copy respectively the current camera position and rotation to the min and max vector under it.

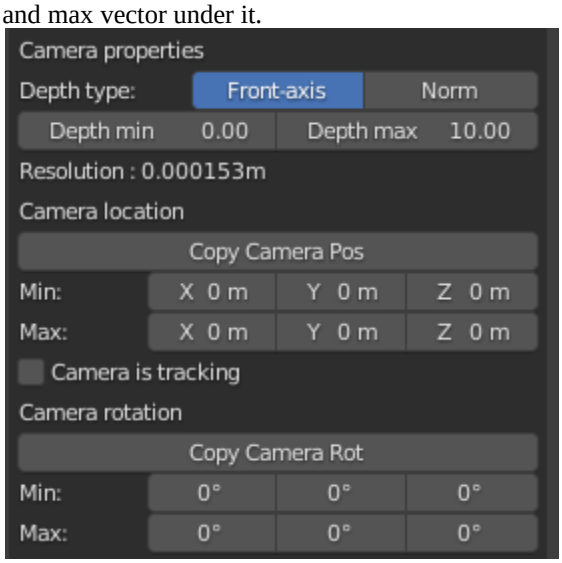

Figure 10. Camera properties GUI when the generation type chosen is *Random*

If the user chooses *Animation* as generation type, the database generation will be done using the animation system present in Blender. The camera can be animated in different ways. The first way is to define several key frames that will be used to define the camera position and/or rotation at key animation moments. Blender then interpolates between the different key frames to calculate a trajectory passing through the different animation key frames. The other option is to create a trajectory that will be followed by the camera. To do this it is necessary to create a curve typed object and model it until the desired trajectory is obtained. Then the user can add the curve as a target to follow via the interface present in Figure 11 and press the button *Generate Camera Animation*. This button samples the curve according to the total image

number in the animation so that the camera follows the curve throughout the animation. The camera can at the same time track an object and follow a given trajectory as depicted in Figure 12 where the camera follows a circle shaped trajectory and track an object positioned in the car center.

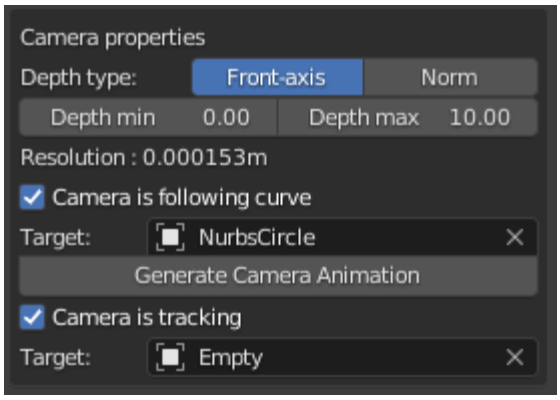

Figure 11. Camera properties GUI when the generation type is animation

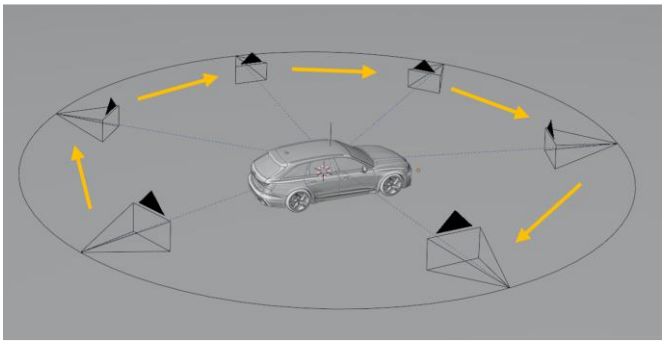

Figure 12. Camera following a circle shaped curve while tracking a car

2) Object properties

The block *Object properties* controls parameters related to the selected objects as depicted in Figure 14.

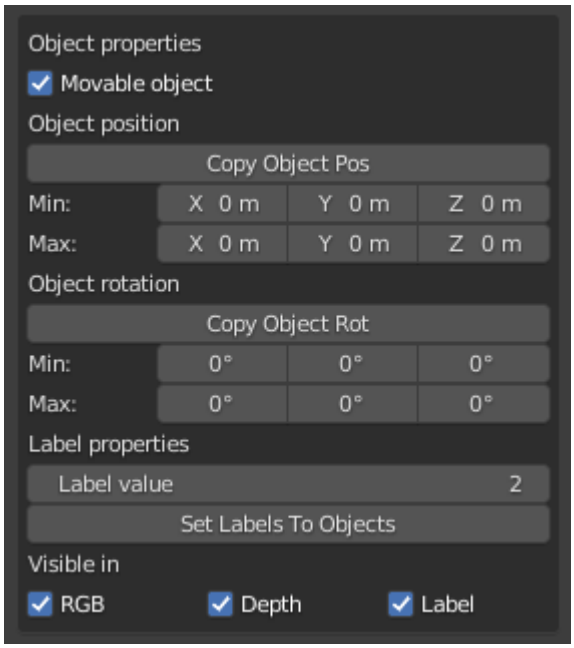

Figure 13. Object properties GUI

The first parameter is a checkbox used to define if the object is either mobile or not. If the object is mobile and the generation type selected is *Random*, it is possible to define for each scene axis, position and rotation ranges in which the object will be positioned randomly. If the generation type selected is *Animation*, this enables to use the Blender animation system. Therefore, it becomes possible to animate objects using key-frames. This enables to create complexes scenes and scenarios. Then a label value can be associated to the selected object. The button *Set Labels To Objects* sets a label value to every object in the scene starting by the value 1 up to the objects number. Finally, it is possible to select the object visibility during the different rendering types (RGB, depth or label).

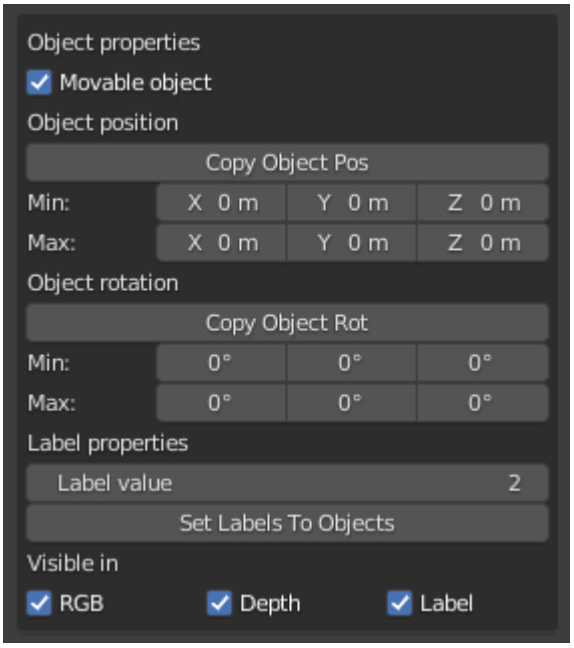

Figure 14. Object properties GUI

C. BlendGen general Workflow

The general BlendGen database generation workflow consists in 7 steps. The workflow is depicted in Figure 15. The first step is the creation or acquisition assets such as 3D models, textures or 360° environment images. The second step is the database parameters set-up which is done in the block *General properties.* The third step is the background environment set-up using either a 360° background image or using 3D models to create a background environment. The fourth step consists in positioning the mains assets into the scene by either using the animation system or using random ranges and setting up their label values. The fifth step is the camera positioning using either the animation system or using random ranges. The sixth step consists in adjusting the light environment by either modifying background parameters in the block *Background properties* if a 360° background image is used or by adding light objects if the background environment is 3D models composed. The last step is the database generation once all the previous steps are done.

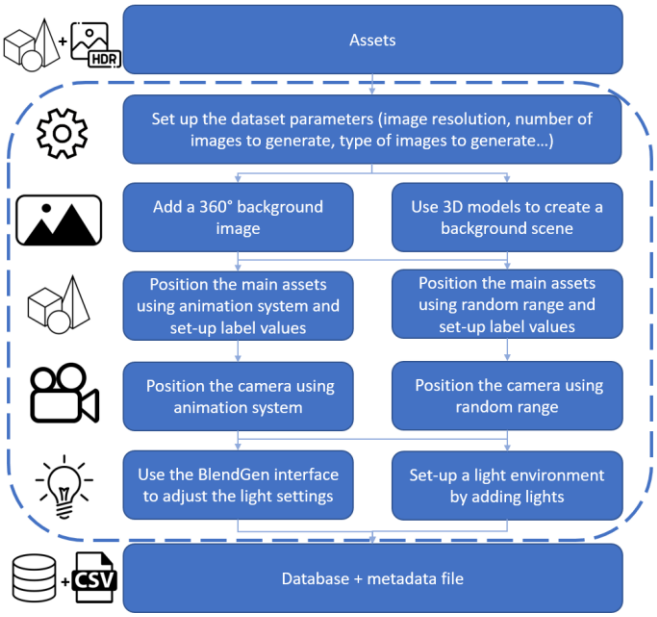

Figure 15 BlendGen database generation workflow

IV. Real BlendGen use case to generate high resolution bus rear faces RGB-D image database

BlendGen was used to generate a photorealistic RGB-D bus backs image database for an industrial application requested by Segula Technologies. The purpose of the application was to segment different sizes of body parts ranging from parts in the order of centimeters to meters using high resolution RGB-D images. The application constraints required to have few thousand RGB-D photorealistic buses backs images, with body part labeled and high resolution (1 pixel < 1mm²). For example, if the bus rear faces dimensions are 2.5 meters wide and 4 meters tall, its image should have at least 2,500x4,000 pixels. Because the bus will never perfectly fit the entire image, its image should have an even higher resolution. The final chosen resolution was 3,600x4,800 pixels. The application needed that some body parts were positioned with slight shift and rotation. Finally, the application also needed a depth resolution below the millimeter.

To generate a database as realistically as possible, a bus model was used. An environment close to a body part assembly site was needed. Therefore, a 360° background garage image was used as it is an indoor environment composed of light condition close to a body part assembly site as depicted in Figure 16. The background label value has been set to 0.

Figure 16 360° background image chosen to generate the bus backend database

Then the bus rear faces body parts were added to the scene as depicted in Figure 17. A rectangle shape without much detail has been added to simulate the rest of the bus. When using the Cycles rendering engine, Blender enables to set an object as *shadow catcher* in the Blender *Object properties*. When this option is selected, the object will display only the shadow that is supposed to catch. To add realism a plane with the option *shadow catcher* has been added under the bus to catch shadows that the bus is supposed to create. As required in the application constraints, some body parts have been set with slight shift and rotation. Finally, objects label values have been set.

Figure 17 Bus back-end 3D model positioning without materials into the scene.

Because the 3D models procured had no materials, 3 different materials have been created from scratch using Blender for each body part types which are glass, light and body. The different materials are depicted in the Figure 18.

Figure 18 Bus back-end with materials.

Some textures have been generated to add diversity in the database. These textures have been applied on the bus rear faces using UV mapping. UV mapping is a process that creates a 2D representation of a 3D object. The UV mapping and its result are depicted in the Figure 19 below

.

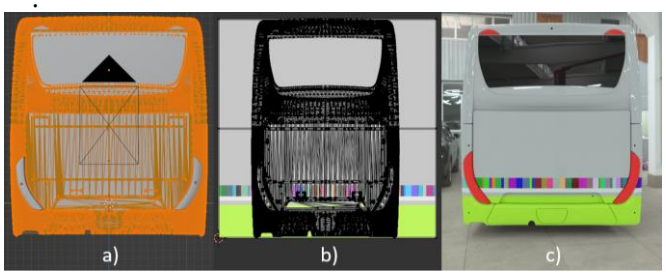

Figure 19 UV mapping applied on the body part. a) Selected body part. b) UV mapping result with the applied texture in the background. c) RGB image rendering using Cycles rendering engine.

The camera location ranges have been set to be between 4.5 and 5.5 meters behind the bus with a -0.5 to +0.5 meters right/left shifting and a height between 0.8 to 1.2 meters. The camera positioning is depicted in Figure 20. The camera rotation has been set to track an empty time object which was positioned in the bus back-end center. These parameters have been set to always have the bus back-end entirely visible.

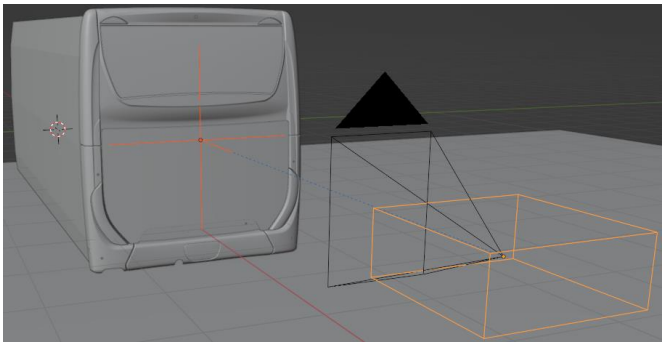

Figure 20 Camera positioning. The orange box represents the volume in which the camera can be randomly located. The orange axis on the bus back-end is the object tracked by the camera.

Then the depth type has been set to *Front-axis* as the application needed this depth acquisition type. The depth range has been set between 2 meters and 8.5535 meters to get a 0.1 millimeter depth resolution. The depth acquisition preview is visible in the Figure 21.

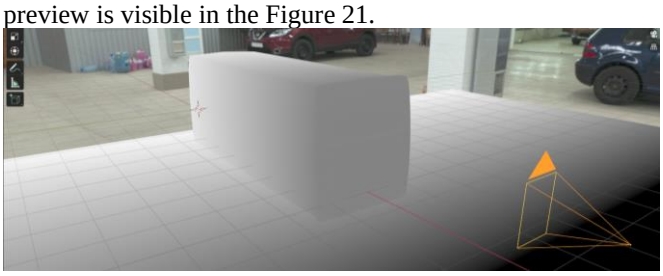

Figure 21 The bus back-end depth acquisition preview

The object intensity has been lowered to 0.6 to match the light reflected by the other vehicles in the background images. Finally, the database has been generated giving the result in the Figure 22 below. The database of 1000 RGB-D labeled images have been generated using the Cycles engine with GPU compute in less than 24 hours. The PC configuration used to generate the database is composed of a i7-9700 CPU, 32Go RAM, Nvidia Quadro P5000 16Go GPU with windows 10 and Blender 3.1.

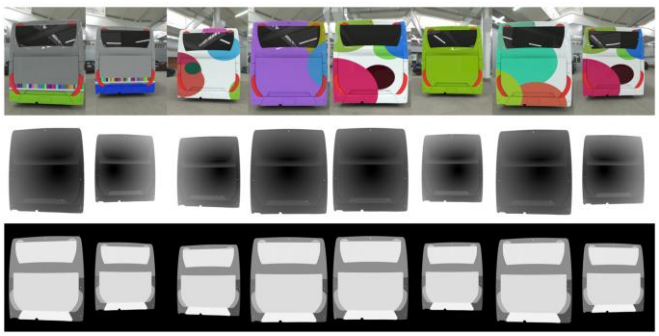

Figure 22. Dataset containing few bus back-ends RGB-D images generated with BlendGen using the ray tracing render engine named Cycles

V. Conclusion

We present BlendGen, a Blender add-on that enables photorealistic RGB-D database generation for general and specific industrial applications via a simple and powerful GUI. BlendGen is an alternative to the existing database generation tools that enables databases generation uniquely through APIs which can make the set-up of complexes scenes less convenient. BlendGen GUI tools are designed to make the scenes set-ups as quick and intuitive as possible therefore enabling fast databases generation. BlendGen enables parameters control such as image resolution, background environment set-up using 360° images, camera positioning, depth acquisition resolution, objects positioning and labeling. An industrial BlendGen use case has been detailed through a high resolution (3,600x4,800p) RGB-D bus backs database generation. This database composed of 1,000 images have been generated using Blender ray-tracing rendering engine in less than 24 hours using a i7-9700 CPU, 32Go RAM and a Nvidia Quadro P5000 16Go GPU. BlendGen was created hoping it will help the community by making image database generation easier. BlendGen code is available on https://gitlab.univ-st-etienne.fr/labhc/dep-isi/image/blendgen.

Acknowledgment

This work is funded by the Association Nationale de la Recherche et de la Technologie (ANRT) under the industrial agreement (CIFRE contract) number 2022/0190.

References

[1] N. Srisook, O. Tuntoolavest, P. Danphitsanuparn, V. Pattana-anake, and F. J. Joseph, “Convolutional Neural Network Based Nutrient Deficiency Classification in Leaves of Elaeis guineensis Jacq,” *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 14, pp. 19–27, 2022.

[2] M. Elleuch and M. Kherallah, “Off-line Handwritten Arabic text recognition using convolutional DL networks,” *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 12, pp. 104–112, 2020.

[3] B. Foundation, “blender.org - Home of the Blender project - Free and Open 3D Creation Software,” *blender.org*. https://www.blender.org/ (accessed May 22, 2023).

[4] O. Russakovsky *et al.*, “ImageNet Large Scale Visual Recognition Challenge,” *ArXiv14090575 Cs*, Jan. 2015, Accessed: Sep. 20, 2022. [Online]. Available: http://arxiv.org/abs/1409.0575

[5] “Cityscapes Dataset – Semantic Understanding of Urban Street Scenes.” https://www.cityscapes-dataset.com/ (accessed Sep. 20, 2022).

[6] T.-Y. Lin *et al.*, “Microsoft COCO: Common Objects in Context,” *ArXiv14050312 Cs*, Feb. 2015, Accessed: Sep. 28, 2022. [Online]. Available: http://arxiv.org/abs/1405.0312

[7] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013, doi: 10.1177/0278364913491297.

[8] “PASCAL-Context Dataset.” https://cs.stanford.edu/~roozbeh/pascal-context/#introduction (accessed Oct. 24, 2022).

[9] “NYU Depth V2 « Nathan Silberman.” https://cs.nyu.edu/~silberman/datasets/nyu_depth_v2.html (accessed Sep. 20, 2022).

[10] "Papers with Code - ImageNet Benchmark (Image Classification)." https://paperswithcode.com/sota/image-classification-on-imagenet (accessed Sep. 20, 2022).
[11] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 567–576. doi: 10.1109/CVPR.2015.7298655.
[12] "Papers with Code - NYU Depth v2 Benchmark (Semantic Segmentation)." https://paperswithcode.com/sota/semantic-segmentation-on-nyu-depth-v2 (accessed Sep. 20, 2022).
[13] "Papers with Code - SUN-RGBD Benchmark (Semantic Segmentation)." https://paperswithcode.com/sota/semantic-segmentation-on-sun-rgbd (accessed Sep. 20, 2022).
[14] "Papers with Code - Cityscapes val Benchmark (Semantic Segmentation)." https://paperswithcode.com/sota/semantic-segmentation-on-cityscapes-val (accessed Sep. 20, 2022).
[15] C. Wu *et al.*, "MotorFactory: A Blender Add-on for Large Dataset Generation of Small Electric Motors," *Procedia CIRP*, vol. 106, pp. 138–143, Jan. 2022, doi: 10.1016/j.procir.2022.02.168.
[16] J. McCormac, A. Handa, S. Leutenegger, and A. J. Davison, "SceneNet RGB-D: 5M Photorealistic Images of Synthetic Indoor Trajectories with Ground Truth." arXiv, 2016. doi: 10.48550/ARXIV.1612.05079.
[17] S. Borkman *et al.*, "Unity Perception: Generate Synthetic Data for Computer Vision," *ArXiv210704259 Cs*, Jul. 2021, Accessed: Sep. 20, 2022. [Online]. Available: http://arxiv.org/abs/2107.04259
[18] W. Qiu *et al.*, "UnrealCV: Virtual Worlds for Computer Vision," in *Proceedings of the 25th ACM international conference on Multimedia*, New York, NY, USA, Oct. 2017, pp. 1221–1224. doi: 10.1145/3123266.3129396.
[19] M. Denninger *et al.*, "BlenderProc," *ArXiv191101911 Cs*, Oct. 2019, Accessed: Sep. 20, 2022. [Online]. Available: http://arxiv.org/abs/1911.01911
[20] C. Heindl, L. Brunner, S. Zambal, and J. Scharinger, "BlendTorch: A Real-Time, Adaptive Domain Randomization Library," *ArXiv201011696 Cs*, Oct. 2020, Accessed: Sep. 20, 2022. [Online]. Available: http://arxiv.org/abs/2010.11696
[21] H. Ponte, N. Ponte, and S. Crowder, "zpy: Synthetic data for Blender.," *GitHub. Note: https://github.com/ZumoLabs/zpy*, vol. 1. 2021.
[22] K. Greff *et al.*, "Kubric: A scalable dataset generator," *ArXiv220303570 Cs*, Mar. 2022, Accessed: Sep. 20, 2022. [Online]. Available: http://arxiv.org/abs/2203.03570

Author Biographies

Cédric Maron graduated with an electrical engineering degree in 2021 from Polytech Clermont-Ferrand, France. Since 2022 he is a Ph.D. student in Computer science working at the Hubert-Curien Laboratory in Saint-Etienne, France in collaboration with Segula Technologies. His research interests are artificial intelligence and computer vision.

Virginie Fresse is an associate professor in the Jean Monnet University, in Saint Etienne, France. She got her PhD degree in Electrical Engineering in INSA Rennes in 2001 and got a post-doctorate position in the University of Strathclyde, in Glasgow from 2001-2003. Her reserarch projects are on embedding image processing algorithms on embedded systems containing FPGA, DSP or embedded CPU devices put on cloud and edge infrastructure. The integration of CNN models for videos and images is also an actual research project with an industrial partner.

Karynn Morand is currently a Research and Development Manager in SEGULA Technologies, Lyon. She graduated with a mechanical and electrical engineering degree in 2001 from ECAM Lyon, France. Her area of work includes artificial intelligence, computer vision and materials.

Hubert Konik received the Ph.D. degree in computer science from Université Jean Monnet, in 1995. He is currently an Associate Professor with Télécom Saint-Etienne and a member of Image Science and Computer Vision team, Laboratoire Hubert Curien, Saint-Etienne, France. His research interests are focused on image processing and analysis, more particularly content aware image processing for new services and usages and eXplainable Artificial Intelligence.