Simulation Environment for Neural Network Dataset Generation

Aleš Vysocký(✉), Stefan Grushko, Robert Pastor, and Petr Novák

Department of Robotics, Faculty of Mechanical Engineering,
VSB-Technical University of Ostrava, 70800 Ostrava, Czech Republic
ales.vysocky@vsb.cz

Abstract. We present a simulation setup in the robot simulation software CoppeliaSim which is used for a synthetic dataset generation for training the neural network. In the simulator we can generate either color and depth images which can be tuned according to the real cameras mounted to the robot or robotic workplace. Vision sensors capture the simulated scene which contains different environment features, obstacles and objects of interest which can be labeled automatically with another filtering vision sensor. Except static environment which can be imported in case of known setup or generated based on height-field or simple objects. We can simulate randomly or with a specific pose oriented and positioned objects which may appear in the field of view of the robot. As an output the system produce RGB or depth information which is stored as a RGB or a gray-scale image or a combined RGBA image including the RGB data extended by depth data stored in the alpha channel. Second product of the system is a label describing different detectable classes for the neural network. The simulator is able to generate large datasets in a short period of time and produce a highly customized learning base for the neural network.

Keywords: Simulation · Neural network · Synthetic dataset

1 Introduction

Synthetic dataset is a learning set of data for training a neural network which is created in an artificial simulated environment. Neural network based image recognition systems used in a cluttered environment require a large and highly diverse dataset. Manual creation of a such dataset is a time consuming and arduous work. In the first step the scene has to be set up according to the situation we want to include in the training base of the neural network next we need to capture this situation and label the regions of interest.

In a comparative study by Dandekar et al. [1] are described different techniques of synthetic dataset acquisition. There are already available datasets made by different research groups including RGB, depth and labeled ground truth images. SceneNet [2] is a collection of data of indoor scenes. Dataset SYNTHIA

J. Mazal et al. (Eds.): MESAS 2021, LNCS 13207, pp. 322–332, 2022.
https://doi.org/10.1007/978-3-030-98260-7_20

[3] includes millions of data samples of the urban environment simulated in an environment including roads, buildings, cars and thousands of different objects. Except rendering different objects in the scene under specific conditions, there might be incorporated also influencing factors extending the basic situations. This can be the weather or light conditions as is described in the work of Khan et al. [4] or in TartanAir dataset [5]. There is a difference between synthetic data generated in the simulation and real world data, in the case of image recognition also depending on the capturing device. Domain adaption [6] considers the differences between simulated and a real data.

In our research we focus on detection and localisation of hands and arms of the operator working with the collaborative robot. For the hand recognition there are datasets available based on real RGB-D data [7,8], special type is dataset with the RGB-D source but labeled automatically with the additional sensor [9], in this study is a comparison between different datasets and labeling methods. Another way is a synthetic dataset generated for RGB tracking [10] or systems using 3D data to recognize the pose of the hand [11]. These systems usually use a simplified estimation of the position of the hand, it uses either other detector to localise the specific region with the hand or methods based on specific shape or color detection. Use of depth image instead of RGB helps to focus more to the shape of the hand than the color which can be in the industrial environment more various in connection with using protective equipment such as gloves. Using RGB and depth together may bring a synergistic effect. There are also datasets including not only the hands but also tools and other objects which may be used during working operation [12] or during interaction with objects [13].

According to Pasieka [14] there is a significant evolution of building synthetic datasets. It can be stochastic, rule-based or generated by the artificial intelligence systems. For the generation we can use simulation environment like CoppeliaSim, game engines like Unity or Unreal Engine or custom software tools usually built on computer vision libraries. The simplest methods are based on combination of 2D images and different augmentations of its parts, more advanced methods are based on placing a 3D object to the 3D environment including the influence of other objects and environment like the occlusion, distortion or lights and ray casting if we need also the color image and not only the depth information.

In the Sect. 2 we describe the simulation environment and conditions for dataset generating. Description of the dataset image is provided. In the Sect. 3 we show the process of unification and tuning of the dataset with real conditions captured with an RGB-D camera. In the Conclusion section we depict further steps and usecase of the dataset.

2 Experiment Setup

In this section we describe conditions and the environment where we want to use the image recognition with the neural network based system. The simulation environment used in this research is CoppeliaSim software. CoppeliaSim provides

simulation environment for script-controlled multi-object scenarios as well as simulation of vision sensors which are used in this research. There is also powerful remote API available for interaction with the simulator from a custom software tools. We specify the vision sensor - simulated in the simulation environment and the real camera which will be used with the neural network based recognizer, then we describe the setup for the environment and the desired output.

2.1 Virtual Vision Sensor

In this experiment setup we use an Intel RealSense D435i RGB-D camera. This camera provides up to 1920×1080 pixel resolution 16 bit color image at maximal frame rate 30 FPS or 60 FPS for lower resolutions. Depth is calculated based on active stereo vision technology. Camera provides either the 1280×720 raw image from both infrared cameras used for stereo vision computation of the distance or the camera provides already calculated depth stream which is even refined with pattern emitted with laser projector. Resolution of the depth stream is 1280×720 at 30 FPS or even 90 FPS for lower resolutions. In the presented approach we try to simulate this depth stream. It is important to pay attention to the specific characteristics and limitations of the stereo vision technology. There is a shadow and invalid data band on one side of the image which are caused by unavailability of the data from both sensors (Fig. 1).

In the CoppeliaSim environment there is an object called vision sensor available which can provide either RGB image and depth. According to extrinsic parameters of the real camera we can use 2 vision sensors, one for RGB and second for depth which will be shifted because of the distance between sensors on the real camera. Setting the perspective angle and resolution unites the field of view of the real and virtual vision sensor. To display depth on the image we use intensity clipped to the minimum and maximum distance which is set equally with the real camera. The product is a gray-scale image (Fig. 1) which can be saved and used later for training of the neural network. In this study we do not use the RGB data, but we use a second vision sensor which also captures the depth. This sensor does not capture the whole scene but only a specific objects of interest, output of this image is transformed to binary producing the mask which is described later in the paper.

The main aim for a synthetic dataset generator is to produce depth images corresponding to the output of the depth camera scaled to the gray-scale image.

2.2 Working Environment

By specifying the obstacles directly according the environment where the system will be used we can set objects which should be detected, which should be recognized or ignored. There are three types of obstacles which are used in the generator:

– **Static environment** A static environment in the industrial applications may be the working table, frame of the workplace, several component supply

(a) (b)

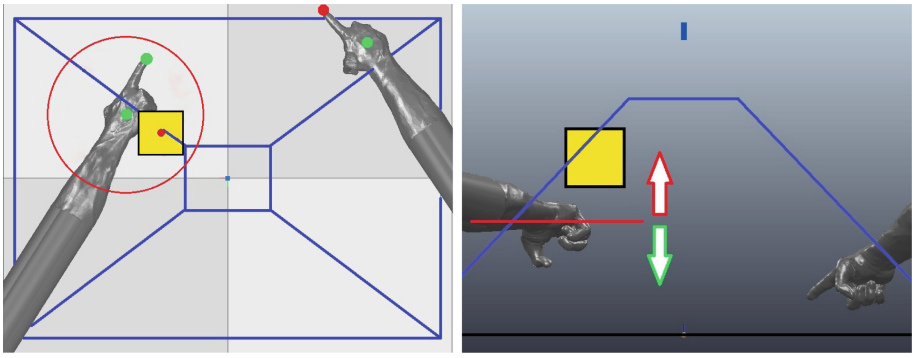
Fig. 1. Depth image colorized in gray-scale in a specific clipping distance: (a) Image captured by the depth camera with visible shadow which is a characteristic for the stereo vision method, (b) image captured in simulation.

devices, clamping units etc. This type of obstacles are usually available as a 3D model of the workstation. This model can be imported to the simulation scene as a mesh. We want to teach the neural network to ignore the static environment.

- **Objects of interest** Objects of interest are things or body parts which we want to detect, distinguish in the scene and localize. The goal is to label these objects in the dataset and train the neural network to recognize these objects.
- **Other objects** Object in the scene which may be similar to the objects of interest but we do not want to detect those objects and we need to teach the neural network to ignore those objects as well as the static environment. This may be the handled/manufactured parts, tools, free wires etc.

In our experiments we want to detect a hand and arm of the operator so we use a 3D model of the right hand in the simulation. It is not necessary to include left hand to the dataset generation because during training phase of the neural dataset we can add augmentations to the input image such as flips and rotations. In different setups we use hand with open palm gesture and pointing with index finger. During dataset generation the hand position is changing. This may be done randomly but we use a spatial linear pattern, where the fingertip of the index finger is shifted in every iteration in one direction, at the end of the area which is captured by the camera the fingertip is shifted in a perpendicular direction, when the finger finishes the layer it is shifted by the selected increment in the z-direction closer to the camera. With this pattern we ensure to capture the object of interest in all positions of the workplace. Moreover we set a semi-random orientation of the hand. Roll, pitch and yaw are limited in order to make the hand visible in obtainable positions according to the real environment. In our experiment we use limits $\pm 90°$ for yaw, $\pm 60°$ for pitch and $\pm 30°$ for roll of the hand. The combination of the position and the random orientation is checked during two tests (Fig. 2). First test is a check if the majority of the hand is visible within the camera field of view. Images with only fragment of the hand may confuse the neural network. We check if the fingertip of the index finger and

a point in the center of the palm are present in a truncated pyramid representing the camera work-space. Second test is a check if an obstacle does not cover the hand, this may occur when the obstacle position is closer to the camera than the hand. Because we use obstacles of simple shapes, we test if the centroid of the obstacle is within a certain distance in the X-Y plane to the point in the center of the palm, if this occurs we check the Z coordinate (the depth) of both of those points. When the obstacle is closer to the camera we move the obstacle more further. If any of those two tests records a problem situation, the scene is regenerated without capturing the image.



(a) (b)

Fig. 2. Tests for omission of invalid situations: (a) top view indicating the obstacle within a monitored area of the hand on the left and the fingertip of the right hand located outside the field of view of the camera, (b) side view indicating the position of the obstacle closer to the camera than is the hand.

Tests are demonstrated in the Fig. 2. There are two situations in the scene, hand on the left pass the first test because both fingertip and center of the palm are within the camera field of view represented with blue color. The hand on the right fails the first test because the fingertip is outside the truncated pyramid. The hand on the left fails during the second test because the centroid of the yellow cube representing the obstacle is within the observed distance and the cube is closer to the camera than the hand.

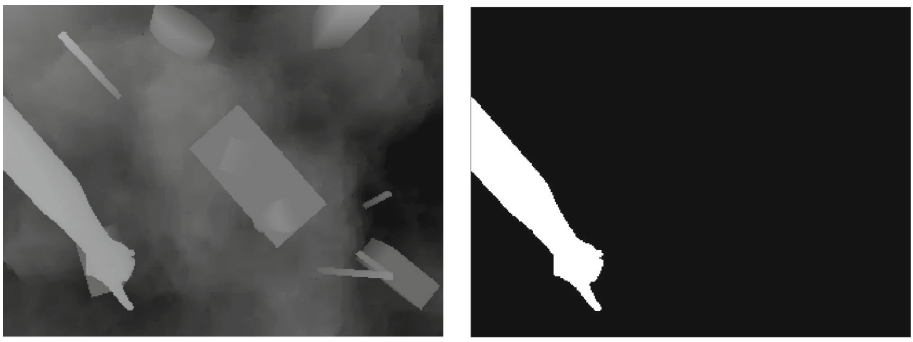
2.3 Dataset Image

Output of this generator is a set of image couples (Fig. 3). Depth image of the scene is saved as gray-scale image. 8 bit format allows to save the depth in resolution of 256 values, this is approximately 4 mm of depth resolution if a clipping distance is 1 m. This resolution may be sufficient for recognition of bigger objects, such as the hand. If we want to detect more details, the depth map may be stored in different format, we used 8 bit resolution in order to keep

the size of the file small. Depth image is stored as a single channel 8 bit depth gray-scale image.

Second image is a mask or ground truth for neural network training process. The mask is binary and it is captured with a vision sensor which can detect only objects of interest and the result of the depth sensor is transformed to binary image. The image may be stored as single channel 1 bit depth gray-scale image. Mask corresponds to the desired output of the neural network based recognizer.

Both images are saved directly from the CoppeliaSim software, further post-processing is done with custom software tools based on OpenCV library. Images are saved to separate folders with the same name for easy pairing in the training process.



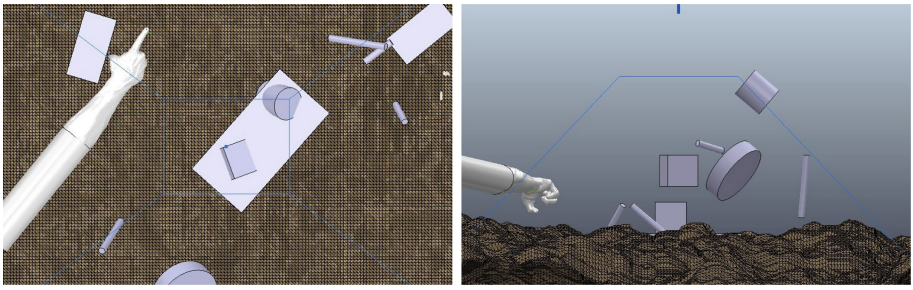
(a) (b)

Fig. 3. Output of the generator is a couple of depth image (a) and a binary label (b) where white are pixels of the object of interest and black is the rest.

3 Synthetic Dataset Generation

In the first simulation scenario (Fig. 4), there is no static environment and the camera is placed one meter above the plane surface. We simulate the floor or the surface below the camera with a height-field which represents uneven surface. In the real camera image there are surface irregularities but also some reflections which can cause false irregularities on the flat surface. The height-field is randomly generated. In our scenario we use a height-field which is wider than the camera field of view and by changing position of the height-field below the camera we make the background random. The height-field is generated from the gray-scale image based on the Perlin noise. Perlin noise is a gradient type noise which creates a very natural pseudo-random appearance. It is used in virtual landscape simulation.

Obstacles in this test are simple shape primitives such as cuboid boxes and cylinders. Long cylinders with a small diameter represent tools or pencils which may occur in the scene and the shape is close to fingers. Moving the obstacles to random positions and orientations creates a highly variable cluttered environment.



(a) (b)

Fig. 4. Simulation environment in CoppeliaSim software: (a) top view corresponding to the view from the camera, (b) view from the side with visible height-field illustrating the background noise.

Comparing the image made by real RGB-D camera (Fig. 5) to the dataset generator output (Fig. 3) shows, that synthetic and real data is similar which was intended in the experiment. Camera image was post-processed to remove the shadow caused by the depth capturing technology which was shown on the Fig. 1. We used a hole filling filter which fills in the missing values. Our custom filter use a static background environment which is captured before the moving objects appear in the scene, for different situations we can use different methods which are also provided by the Intel RealSense camera. Basic filters use the value based on the surroundings of the missing pixel. It can be either copied or calculated to best fit to the missing region.

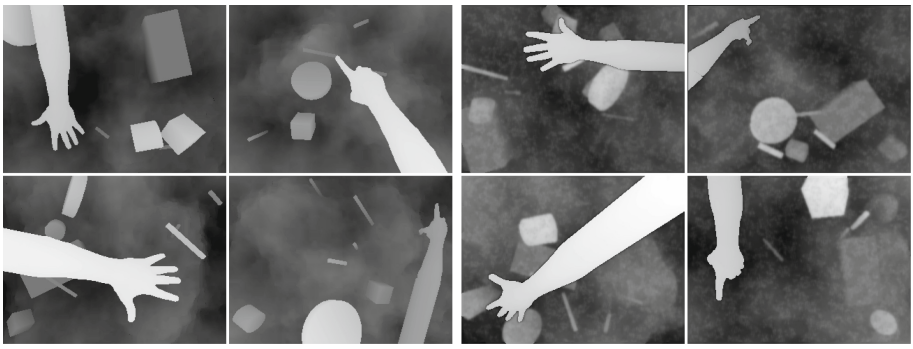
It is not necessary, that the hand, or the object of interest in general, is closer to the camera than other objects in the background. Some recognizers use this feature to separate the area with the hand from the background. In our system we don't take the object of interest as the object in the foreground but objects of interest are labeled with the labeling vision sensor. This makes the system more versatile but we need to handle the possible occlusions which may disrupt the learning process.

If it is necessary to use higher resolution of the output image, there may be problem with sharper edges in the simulation than in the real camera image, where reflections on the surface and other disruptive effect cause that the shape of the object is not as sharp as in the simulation. Further post processing, such as a blur filter or another noise added to the image helps with unification of the image.

In the second experiment (Fig. 7) we set the environment according to a real workplace which is available in our laboratory. There is a robot Universal Robots UR3 mounted to the table which is intended for human-robot cooperation during assembly. The robot is capable of handling the maximal payload of 3 kg and the operation radius is 500 mm. This robot is intended for collaborative operation next to the human operator and it is certified as safe for collisions under a specific circumstances (regulated velocity, safe tool and other safety precautions essential

(a) (b)

Fig. 5. Image captured with a real RGB-D camera mounted 1 m above the ground, scene includes objects of basic shapes and the hand: (a) RGB image of the scene, (b) depth image with applied hole filling filter.



(a) (b)

Fig. 6. Augmentations used in the dataset: (a) different gestures may be used in the simulation according to imported model of the hand, (b) selective blur filter applied to the image may help to blunt the edges of the simulated objects and additional noise represents different reflections and surface irregularities.

for the safe human-machine cooperation). The robot has six degrees of freedom and the body of the robot is made of aluminium cylinders and plastic covers. The plane of the working table is approximately one meter above the ground which means that the standing operator can freely operate with his hands on the table. One meter above the working plane is a platform where an RGB-D camera is mounted same as in the first experiment. 3D model of the construction of the workplace is imported to the simulation software and the virtual sensor is placed to the same place as the real camera is mounted.

(a) (b)

Fig. 7. Experimental workplace with the robot Universal Robots UR3: (a) Simulation environment CoppeliaSim with imported workplace including robot UR3. Figurine has a hand with pointing gesture mounted to the right arm. Output of the generator are depth (upper) and label (bottom) images, (b) image captured with real camera at the workplace in the laboratory.

Object of interest is a part of the figurine, hand and arm are visible for the labelling sensor. The hand and arm are moving in the same pattern as in the first experiment. Instead of random basic shape obstacles there is a robot which is moving to random valid positions. Lightweight industrial robot as UR3 may appear in the image very similar to the human arm. In this experiment we intend to teach the neural network based recognizer to recognize the arm but to ignore the robot during operation (Fig. 6).

4 Conclusion

Proposed synthetic dataset generator is able to generate large sets of labeled images in a short period of time. Moreover the CoppeliaSim simulation environment may be operated in multiple instances on a single computer or it can be run on more machines with different augmentations of the scene. In our experiment we store the images on the hard drive of the computer, so for better operation we try to minimize the size of the file. Size of the image 320×240 pixels which was used in the experiment was sufficient for the given scene and objects of interest. We used a post-processing for dataset split into train/validation and test parts. In this simulation environment this could be done by additional random process or by random save to different locations without additional post-processing of the dataset. By increasing the size of the dataset we can capture more details of the scene, this may be necessary if we want to detect screw heads or small parts, on the other hand bigger file takes up more space on hard drive which may be significant in datasets including millions of images. Secondly the training phase of the neural network gets longer with larger images. This also applies to the inference phase where we need the time for recognition as short as possible to ensure the operation without delays.

The scene based on the imported workplace and a 3D scan of the hand in the simulation is a source of images very similar to the images taken by the depth camera in the real workplace. Simple post-processing of the simulated image or the real camera image may further increase the similarity between those data sources. Simulation image can be blurred for refinement of sharp edges or we can add some additional noise to prepare the network for the real data. The noise and unclear segments are cleaned from the real camera image. The setup of the real camera and the properties of the post-processing of the generated images may be tuned to get the best combination for reaching the best similarity between generated images and real camera images in the specific environment.

With this generator we can simulate environment with some specific shapes and characteristics typical in the industrial area. The dataset is more specific and the neural network recognizer may be trained with stronger emphasis to the specific workplace. 3D model of the workplace is usually available from the design stage and is a part of documentation of the workplace. Therefore the process of setup of the environment in the simulation software is fast. First tests with the training of the neural network have very positive results and specification of the training phase and comparing to the existing recognition ways will be done in the further research. For industrial purposes we need to localise the hand and recognise the gesture for a natural human robot interaction, there is also a second purpose to use the localisation for safety reasons. In this case the system must be very robust for predicting the collision between the operator and a moving part of the machine and the operation must be without delays and if possible close to the real-time operation. Basic dataset we intend to use in semantic segmentation with hand area extraction. This may be used as an input to the OpenPose network which requires color image with specified hands locations. From the CoppeliaSim environment we can extract hands and fingers joints positions and gesture types and train network for detection.

Acknowledgment. This work was supported by the Research Platform focused on Industry 4.0 and Robotics in Ostrava Agglomeration project, project number CZ.02.1.01/0.0/0.0/17_049/0008425 within the Operational Programme Research, Development and Education.

References

1. Dandekar, A., Zen, R.A.M., Bressan, S.: A comparative study of synthetic dataset generation techniques. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) DEXA 2018. LNCS, vol. 11030, pp. 387–395. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98812-2_35
2. McCormac, J., Handa, A., Leutenegger, S., Davison, A. J.: SceneNet RGB-D: 5M photorealistic images of synthetic indoor trajectories with ground truth. arXiv (2016)
3. Ros, G., Sellart, L., Materzynska, J., Vazquez, D., Lopez, A.M.: The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016

4. Khan, S., Phan, B., Salay, R., Czarnecki, K.: ProcSy: procedural synthetic dataset generation towards influence factor studies of semantic segmentation networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, June 2019
5. Wang, W., et al.: TartanAir: a dataset to push the limits of visual SLAM. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (2020)
6. Loghmani, M.R., Robbiano, L., Planamente, M., Park, K., Caputo, B., Vincze, M.: Unsupervised domain adaptation through inter-modal rotation for RGB-D object recognition. arXiv preprint arXiv:2004.10016 (2020)
7. Tompson, J., Stein, M., Lecun, Y., Perlin, K.: Real-time continuous pose recovery of human hands using convolutional networks. ACM Trans. Graph. **33**, 1–10 (2014)
8. Qian, C., Sun, X., Wei, Y., Tang, X., Sun, J.: Realtime and robust hand tracking from depth. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1106–1113 (2014). https://doi.org/10.1109/CVPR.2014.145
9. Yuan, S., Ye, Q., Stenger, B., Jain, S., Kim, T.: BigHand2.2M benchmark: hand pose dataset and state of the art analysis. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017
10. Yang, D., Moon, B., Kim, H., Choi, Y.: Synthetic hands generator for RGB hand tracking. In: TENCON 2018–2018 IEEE Region 10 Conference (2018)
11. Malik, J., et al.: DeepHPS: end-to-end estimation of 3D hand pose and shape by learning from synthetic depth. arXiv (2018)
12. Shilkrot, R., Narasimhaswamy, S., Vazir, S., Hoai, M.: WorkingHands: a hand-tool assembly dataset for image segmentation and activity mining. In: BMVC (2019)
13. Hasson, Y., et al.: Learning joint reconstruction of hands and manipulated objects. In: Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR (2019)
14. Pasieka, M.: The evolution of synthetic data: a comparison of three data generation methods. Mostly AI. https://mostly.ai/2020/10/28/comparison-of-synthetic-data-types/. Accessed 21 Aug 2021