Intelligent Human Systems Integration (IHSI 2023), Vol. 69, 2023, 896–904
https://doi.org/10.54941/ahfe1002915

Towards Knowledge-Based Generation of Synthetic Data by Taxonomizing Expert Knowledge in Production

Oliver Petrovic, David Leander Dias Duarte, Simon Storms, and Werner Herfs

Laboratory for Machine Tools WZL, RWTH Aachen University, 52074 Aachen, Germany

ABSTRACT

Synthetic data is a promising approach for industrial computer vision because it can enable highly autonomous production processes. However, this potential is not fulfilled by current software for synthetic data generation, which usually requires a programmer to create new datasets. To overcome this, we are proposing a framework for more autonomous synthetic data generation, formalizing user roles relevant to such systems. A central aspect of our framework is that domain experts can easily influence the generation of synthetic data by entering knowledge via user interfaces. To get a better idea of what such knowledge could be, we have systematically collected examples of knowledge types for synthetic data generation in production and combined them into a taxonomy with almost 300 nodes. Using this taxonomy as the basis for analyses, we derive six implications for our framework, such as knowledge being not only passed on by domain experts but also by the designer of the user interfaces and generation algorithms. We plan to incorporate these findings to further refine and implement our framework in future research.

Keywords: Synthetic data generation, Expert knowledge, Machine learning, Computer vision, Sim2Real transfer, production, Industry 4.0

INTRODUCTION

While machine learning (ML) has enabled many advances in industrial computer vision (Zhou et al. 2022), modern ML algorithms often require big datasets, which are expensive to create. The reason for these high costs is that in computer vision, datasets are usually created manually by taking photos with a camera and then labeling one after another with a computer program. A more cost-efficient approach is the use of synthetic data, which can be generated significantly quicker in a computer simulation.

Synthetic data is especially promising in the context of production because 3D models, which are required for the simulations, often already exist in the form of CAD models (Eversberg and Lambrecht 2021). Furthermore, synthetic data could enable highly autonomous production processes because new datasets can be automatically generated to retrain ML models when a production task changes (Alexopoulos et al. 2020). However, the use of synthetic data in production has so far been hindered by two aspects. First, ML models trained on it often underperform when applied in the real world. There are several methods that try to overcome this "sim2real gap". For instance, a process expert's knowledge about a particular task can be used to build more realistic simulations, which is the approach we're focusing on in this paper. Secondly, current synthetic data APIs often require programming to create new datasets (e.g., Denninger et al. 2019; Frolov et al. 2022; Morrical et al. 2021), which limits the potential for autonomy.

To increase the usability of synthetic data for production contexts, we're working on a new software system that streamlines the integration of expert knowledge in the generation process. As part of this work, we have developed a novel, human-centered framework, which we're presenting in this paper. This framework introduces three user roles and defines their tasks within the software system.

Because we want our system to be expandable to a large variety of production tasks, it should also support a broad range of knowledge types. However, to the best of our knowledge, there currently exists no overview of knowledge types for synthetic data generation in production. Thus, to get a better idea of what our system needs to be capable of, we have created such an overview in the form of a taxonomy. This taxonomy is the result of collecting and unifying many examples of expert knowledge and includes over 290 nodes to give a broad overview of this field. Following the presentation of our taxonomy, in the last part of this paper we're using it to derive practical implications for our framework and synthetic data software systems in general. These implications concern the roles within our framework and the way that data can be stored in the system.

A FRAMEWORK FOR KNOWLEDGE-BASED SYNTHETIC DATA GENERATION

The aim of the software system that we're working on is to simplify the generation of synthetic data in production contexts. Expert knowledge plays an important part in our system. On the one hand, this knowledge defines the task itself, for instance which objects should be recognized. On the other hand, the knowledge is also used to create more realistic synthetic data to help overcome the sim2real gap. A focus of our system is on how this knowledge is entered into it. While current synthetic data APIs often require programming, our system uncouples programming from the input of process knowledge. This way, process experts from many fields of production can directly enter their knowledge without needing specific programming expertise. A last key characteristic of our system is that it is being designed as a general system that can be used for a broad range of tasks. To achieve this, it should be extendable with modular add-ons that add features for additional tasks.

As a first step towards the development of the system, we have created a framework that introduces user roles relevant for such a system and formalizes the ways that they interact with it. Overall, as can be seen in Figure 1, our framework includes three user roles, which are described in the following paragraphs.

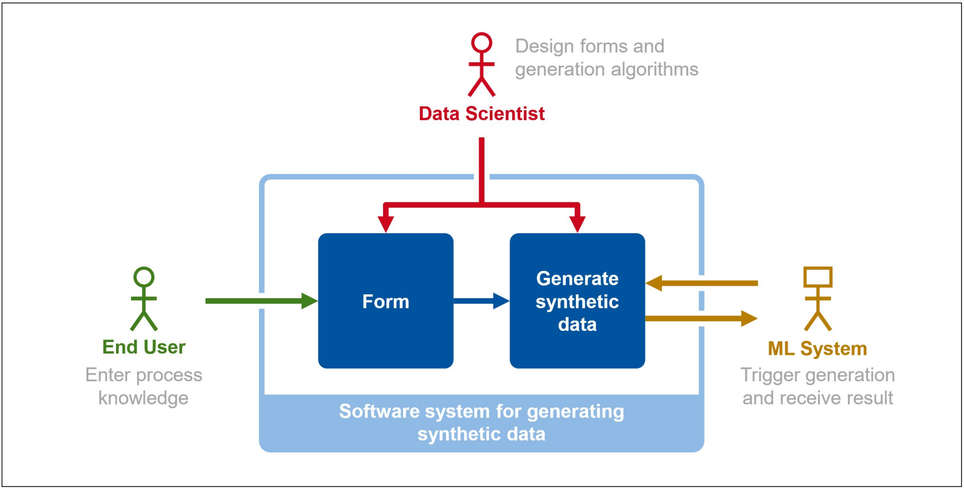
Design forms and generation algorithms
Data Scientist
End User
Enter process knowledge
Form
Generate synthetic data
ML System
Trigger generation and receive result
Software system for generating synthetic data

Figure 1: Overview of our framework for knowledge-based generation of synthetic data. The framework introduces the three user roles "End User", "Data Scientist" and "ML System" that software systems following the framework would interact with.

End Users are human users that hold knowledge about the production process. We want to enable these users to directly enter their knowledge into the system without requiring a programmer. To achieve this, form-based user interfaces are used in which knowledge can be easily entered. A broad range of people have knowledge relevant for production processes and could thus fill this role. Specifically, in the context of an industry 4.0 process, "smarter operators" are expected to become process experts with sufficient technical training for such tasks (Margherita and Bua 2021).

Data Scientists are also human users. Their task is to create the forms in which End Users enter their knowledge and the generation algorithms that determine how the entered knowledge is used to generate synthetic data. To make sound decisions for these tasks, the Data Scientist role needs to work out requirements for a project with a process expert. Once the Data Scientist is finished, End Users can use the created forms on their own without further help from the Data Scientist. Data Scientists need programming skills to fulfill their tasks. Not only people with the job title "data scientist" can fulfill this role, but many types of engineers.

The last role in our framework is the **Machine Learning System**. This role is not a human role but an external software system that controls ML processes and uses our software as one of several modules for their tasks. Due to this relationship, our system only needs to focus on generating synthetic data, while all other ML-related tasks, such as augmentation or inference, are outside of its system boundaries. In our framework, the ML System triggers the generation of synthetic data and receives the generated dataset at the end of this process. When triggering the generation process, the ML System can also send parameters, which is crucial for highly autonomous production processes. For example, consider robots handling parts based on a computer vision algorithm. When a new production process with a never-before-seen part has to be implemented, the ML System could send the CAD model from this task as a parameter to our system, which would generate a synthetic dataset for it. This dataset could then be used by a different module of the ML System to retrain ML models. Finally, these models could be deployed on the robots' controls, which would then automatically have learned to handle the new part without any human intervention. Furthermore, parameters sent by the ML System could also be used for more realistic simulations, e.g., by containing information about the current factory environment collected via sensors. Because the parameters could also come from somebody who saved them in a previous process step (e.g., a constructor), the ML System may also be seen as a pipeline inducing knowledge from other process experts who are unaware of their relationship to the synthetic data system.

METHODOLOGY

To get a better idea of what our system needs to be capable of and what kinds of knowledge the form-based interfaces might need to support, we wanted to concretize the term expert knowledge. To do this, we have systematically created a taxonomy of knowledge types for synthetic data generation in production. First, we have collected examples of knowledge types from several sources, including internal work, research papers concerning industrial computer vision tasks, and by examining a software system for rendering synthetic data. Each source offered a unique perspective. For instance, the papers yielded more practically relevant examples, such as common object arrangements, whereas the software system led to technical examples, like rendering attributes that define how a material looks. After this collection phase, we combined all examples in a single list and unified them, e.g., by choosing consistent terms. Finally, we attempted to find a unifying structure for all examples by clustering them into logical groups.

TAXONOMY OF KNOWLEDGE TYPES FOR SYNTHETIC DATA GENERATION

The result of the described approach is a taxonomy with 298 nodes that are structured in 9 levels. Our taxonomy covers a wide variety of topics, such as scene arrangements, light properties, label types, and many more. The aim of our taxonomy is to give an overview of the many types of knowledge that can be utilized in the generation of synthetic data. Figure 2 shows the first four levels of it. The entirety of our taxonomy can be found online as a spreadsheet and visualized as a big figure: https://zenodo.org/record/7270630. In the following, we'll describe a few key aspects of our taxonomy.

At the second level, our taxonomy is divided into two branches. "Modelling Reality" encompasses expert knowledge about the real scene to be simulated, e.g., its physical appearance or what number of objects in it are probable. "Settings", on the other hand, includes expert knowledge that cannot be deduced from how the scene looks but are decisions made for other reasons. For instance, such decisions are the rendering method, the number of

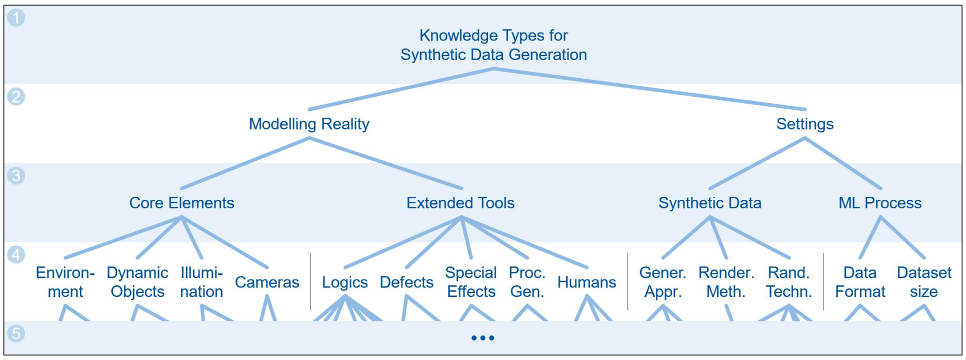

Figure 2: The first four levels of our taxonomy of knowledge types for synthetic data generation in production.

images to generate, and whether to employ unrealistic randomization methods to make ML models trained on the data more robust. They may still influence how the simulation looks but cannot be derived by looking at a real scene.

Within the "Modelling Reality" branch of our taxonomy, the next division is into "Core Elements" and "Extended Tools". "Core Elements", such as "Objects", "Environment" and "Illumination", are relevant to most projects. "Extended Tools", on the other hand, are entire categories only useful for specific contexts, like for "Defect Detection" tasks or rendering "Humans" in a scene.

One pattern found at various places in our taxonomy is a division into "Properties" and "Selection Method" branches. "Properties" include all the attributes that can be modelled for a specific parent category, like "Roughness", "Sheen" and "Transmission" if the parent category is "Material". The other branch, "Selection Method", specifies how these properties are set by an expert. For instance, an expert could adjust a material by directly setting the value for many rendering properties. However, this would be a difficult task because an expert on a metal processing application probably wouldn't know what combinations of rendering properties yield a realistic metal look. Therefore, a potentially more fitting selection method might be an interface offering only a few predefined categories, out of which the expert selects a fitting one. As shown in Figure 3, in the context of "Material" such categories could be "Plastic", "Wood" and "Steel". The values for the rendering properties would then automatically be set in the background according to the chosen category. Beyond these two examples, more selection methods are conceivable.

One last thing to note about our taxonomy is that it displays at parts different kinds of relationships. There are generalization relationships, like a "Light Source" being either a "Dome Light" or a "Mesh Light", but never both. Then there are attribute relationships, like a "Sky" having an "Atmosphere Thickness" argument that determines its look. And there are category relationships, which are a special form of attributes, like the division into "Environment", "Object", "Illumination" and "Camera" knowledge types in the fourth layer.

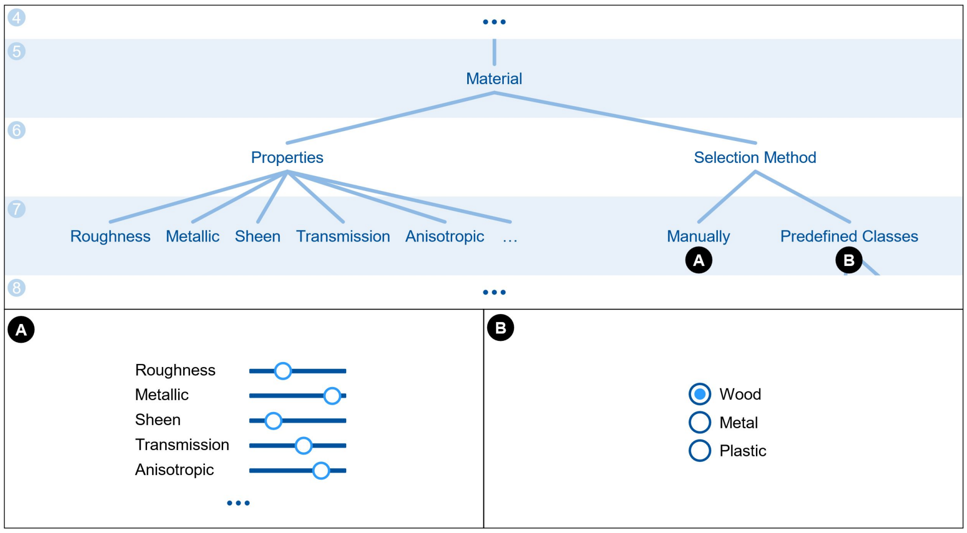

Figure 3: Top: Cutout of our taxonomy with "Properties" in one branch and "Selection Methods" defining how those properties are set by an End User in the other. Bottom: Exemplary interfaces for the two "Selection Methods". When a category in (B) is selected, the properties from (A) would still be set, but automatically in the background with predefined values yielding a realistic look.

IMPLICATIONS FOR OUR FRAMEWORK

Our aim in creating the taxonomy was to better understand what our software system needs to be capable of. To do this, we have used the taxonomy as the basis for analyses. For instance, we looked at where in the taxonomy similar data types can be found, and we tried to answer which of the knowledge types in the taxonomy could be entered by which of the three user roles of our framework. These analyses have yielded six implications for our software system that we found especially relevant.

1. **The Data Scientist role also enters own expert knowledge that influences the generation of the synthetic data.** As was seen in Figure 3, the design of the form, in which the End User enters their knowledge, determines what can be entered at all. For instance, whether there are sliders or more simple-to-understand classes, and if the latter, then which classes there are and which values they set in the background. In our framework, the Data Scientist role is responsible for creating the forms. Thus, we deduce that the Data Scientist is also entering knowledge when doing this preselection of what the form looks like. This has two consequences. First, if one wants to record what knowledge led to the generation of a synthetic dataset, then beyond looking at what the End User entered, one also has to take into account the knowledge entered implicitly by the Data Scientist. Secondly, this deduction underscores the importance of the Data Scientist working together with a process expert in the requirements engineering of a new interface.

2. **The task of the End User can be simplified by focusing on modelling the real world.** At the top, our taxonomy is divided into "Modelling Reality" and "Settings". While most End Users could enter "Modelling Reality"

knowledge types, only End Users with ML knowledge can enter "Settings" types. We thus think that it is in most cases easier if the task for the End User is limited to modelling reality. For instance, instead of explaining that an unrealistically high variance might improve the performance of synthetic data (cf. domain randomization, Tobin et al. 2017), a form should only ask the End User to enter realistic values. If the Data Scientist wants to use domain randomization, then they themselves could add additional variance on top of the End User's values as part of their generation algorithm.

3. **Software systems don't need to differentiate between knowledge coming from an End User or from the ML System.** When trying to answer for a few of the knowledge types in our taxonomy which user role could enter them, we found that when one could be entered by End Users, it usually could also come as a parameter from the ML System. This makes sense because the ML System can induce expert knowledge that it received in previous steps of the production process.

4. **Some knowledge types are only relevant for a user-friendly experience.** For instance, a rotation can be saved as a Euler rotation or, more visually, as being defined by the center of the object to rotate and a second point to which it should be orientated. One could transform the latter to a Euler rotation so that the generation algorithm only needs to support one data type. However, the points would still need to be saved for a friendly user experience. The reason for this is that if a user wants to edit their input later on, they again want the simple representation that they used when entering it (e.g., the two points). A software system could differentiate between data used for the rendering process and data only used for user-friendly forms to get more focused data structures for both.

5. **Some knowledge types can have different representations.** For instance, for color values there are the two equal representations of RGB or HSV values. A data structure could realize this differentiation via inheritance schemes with general parent classes and more specific child classes.

6. **Some knowledge types can be found at multiple places, such as a position.** A unified structure for these types can save effort. Sometimes it might even be beneficial to use the same knowledge entered once by an expert at more than one place. For example, the same type of material could be used by many different objects in a scene.

DISCUSSION

The taxonomy enabled implications directly relevant to our framework. As was the aim, concretizing expert knowledge led to new understandings, like forms better focusing on modelling reality, or the idea of also observing knowledge added implicitly by the Data Scientist. These findings can be used in the further refinement and development of our framework and software system.

Beyond enabling implications for a software system, our taxonomy can also be used for many other applications. Because examples from many different sources were used to create it, the taxonomy gives a diverse overview of knowledge types relevant for synthetic data generation in production. Possible applications include using it as an overview for teams deciding which kinds of synthetic data knowledge to use in a project or as a basis for systematically researching the impact of the different knowledge types in it on the performance of ML models.

Despite these applications, our taxonomy should be seen as a proposal. The focus in creating it was on combining a broad set of data into one structure. To achieve this, opinionated decisions were made that could also have been made differently. Thus, there are other ways to taxonomize expert knowledge and there are also knowledge types not yet included. Future research projects can extend or adapt our taxonomy to fit their tasks, or they can also use it as a baseline to discuss improvements or different structures altogether.

Back to our own use of the taxonomy, our implications can be further worked out. On the one hand, they were derived rather broadly by using the taxonomy to answer questions. These analyses could be repeated more systematically to gain further confidence and a more detailed understanding of the implications. On the other hand, the implications can be tested as hypotheses in dedicated experiments. For instance, it could be tested whether End Users from different fields really find it easier to just model reality than if they also had to mind things like unrealistic variance for domain randomization. Such systematic testing can show if the implications can hold and, if they do, then to what extent.

Lastly, the breadth of knowledge types found shows that an add-on system indeed appears useful because a system could hardly support all knowledge types from the get-go. The categories in the taxonomy show that knowledge types can be divided into distinct clusters. These clusters could become an orientation for how add-ons could be grouped, e.g., one add-on focused on "Humans", one on "Defects", and so forth.

CONCLUSION

In this paper, we have attempted to concretize several aspects of synthetic data generation in production. First, we presented a novel, human-centered framework that introduces three key user roles for knowledge-based synthetic data generation systems. Then, we attempted to concretize what kinds of knowledge such systems can be extended with by systematically collecting examples of knowledge and combining them into a unified taxonomy with almost 300 nodes. This taxonomy is the first of its kind that we are aware of and can be a basis for the further development of such overviews. Finally, we have used our taxonomy to derive implications for our framework and for synthetic data generation systems in general.

We have published the entirety of our taxonomy online to further advance progress in the field of knowledge-based synthetic data generation. Based on the work presented in this paper, we have refined our framework and developed a prototype software system. We plan to share more about these progresses in a future, more practicality-oriented follow-up paper.

REFERENCES

Alexopoulos, K.; Nikolakis, N.; Chryssolouris, G. (2020): Digital twin-driven supervised machine learning for the development of artificial intelligence applications in manufacturing. In *International Journal of Computer Integrated Manufacturing* 33 (5), pp. 429–439. DOI: 10.1080/0951192X.2020.1747642.

Denninger, M.; Sundermeyer, M.; Winkelbauer, D.; Zidan, Y.; Olefir, D.; Elbadrawy, M. et al. (2019): BlenderProc. Available online at http://arxiv.org/pdf/1911.01911v1.

Eversberg, L.; Lambrecht, J. (2021): Generating Images with Physics-Based Rendering for an Industrial Object Detection Task: Realism versus Domain Randomization. In *Sensors (Basel, Switzerland)* 21 (23). DOI: 10.3390/s21237901.

Frolov, V.; Faizov, B.; Shakhuro, V.; Sanzharov, V.; Konushin, A.; Galaktionov, V.; Voloboy, A. (2022): Image Synthesis Pipeline for CNN-Based Sensing Systems. In *Sensors (Basel, Switzerland)* 22 (6). DOI: 10.3390/s22062080.

Margherita, E.G.; Bua, I. (2021): The Role of Human Resource Practices for the Development of Operator 4.0 in Industry 4.0 Organisations: A Literature Review and a Research Agenda. In *Businesses* 1 (1), pp. 18–33. DOI: 10.3390/businesses1010002.

Morrical, N.; Tremblay, J.; Lin, Y.; Tyree, S.; Birchfield, S.; Pascucci, V.; Wald, I. (2021): NViSII: A Scriptable Tool for Photorealistic Image Generation. Available online at http://arxiv.org/pdf/2105.13962v1.

Tobin, J.; Fong, R.; Ray, A.; Schneider, J.; Zaremba, W.; Abbeel, P. (2017): Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World. Available online at http://arxiv.org/pdf/1703.06907v1.

Zhou, L.; Zhang, L.; Konz, N. (2022): Computer Vision Techniques in Manufacturing. In *IEEE Trans. Syst. Man Cybern, Syst.*, pp. 1–13. DOI: 10.1109/TSMC.2022.3166397.