Proceedings of the HPI Research School on Service-oriented Systems Engineering 2020 Fall Retreat

Christoph Meinel, Jürgen Döllner, Mathias Weske, Andreas Polze, Robert Hirschfeld, Felix Naumann, Holger Giese, Patrick Baudisch, Tobias Friedrich, Erwin Böttinger, Christoph Lippert, Christian Dörr, Anja Lehmann, Bernhard Renard, Tilmann Rabl, Falk Uebernickel, Bert Arnrich, Katharina Hölzle

Technische Berichte Nr. 138

des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam

Universität Potsdam

HPI Hasso Plattner Institut
Digital Engineering · Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam

Technische Berichte des Hasso-Plattner-Instituts für
Digital Engineering an der Universität Potsdam | 138

Christoph Meinel | Jürgen Döllner | Mathias Weske | Andreas Polze | Robert Hirschfeld | Felix Naumann | Holger Giese | Patrick Baudisch | Tobias Friedrich | Erwin Böttinger | Christoph Lippert | Christian Dörr | Anja Lehmann | Bernhard Renard | Tilmann Rabl | Falk Uebernickel | Bert Arnrich | Katharina Hölzle

Proceedings of the HPI Research School on Service-oriented Systems Engineering 2020 Fall Retreat

Universitätsverlag Potsdam

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet via http://dnb.dnb.de/.

Universitätsverlag Potsdam 2023
http://verlag.ub.uni-potsdam.de/

Am Neuen Palais 10, 14469 Potsdam
Phone: +49 (0)331 977 2533 / Fax: 2292
Email: verlag@uni-potsdam.de

The series **Technische Berichte des Hasso-Plattner-Instituts für Digital Engineering an der Universität Potsdam** is edited by the professors of the Hasso Plattner Institute for Digital Engineering at the University of Potsdam.

ISSN (print) 1613-5652
ISSN (online) 2191-1665

Layout: Tobias Pape
Print: docupoint GmbH Magdeburg

ISBN 978-3-86956-513-2

Also published online on the publication server of the University of Potsdam:
https://doi.org/10.25932/publishup-50413
https://nbn-resolving.org/urn:nbn:de:kobv:517-opus4-504132

Contents

The Complex Road to a Verisimilitudinous Environment Using Virtual City Models

Andreas Fricke

Computer Graphics Systems Group
Hasso Plattner Institute for Digital Engineering
andreas.fricke@hpi.de

This report describes my recent activities and research work on Service-oriented Systems Engineering in the HPI Research School as a continuation of my Fall Report 2019 and my Spring Report 2020.

1 Overview and objectives of this doctoral project

This doctoral project deals with a question relevant to both science and everyday reality: How can functionally rich virtual multidimensional city models be generated, derived, visualised and made usable in an application-oriented way in the face of insufficient basic data (lack of reference)? On the one hand, questions of acquisition, processing (geometric as well as semantic context), function assignment and visualisation have to be addressed. On the other hand, the mentioned sub-areas are to be integrated into a generic, service-oriented overall process, which is to run largely automatically.

1.1 About the previous reports

The last reports illustrated how geographical problem solving can be achieved with modern service-based techniques and procedures, using the example of a database for the digital representation of a virtual spatial model. Furthermore, it was explained which problems exist in the spectrum of digital twins and spatial databases with regard to the handling of ubiquitous spatial data, which require the conception of a new service-oriented approach. A special focus is on the database as a central element for a virtual representation of an urban environment. Ubiquitous geodata require meaningful tools for integration and harmonisation, which are also integrated as part of a modular process chain. In this approach, procedures and workflows based on generic data models and standard schemes are designed to ensure transparency and transferability both methodologically and spatially. A workflow describing data acquisition, data processing and data use will be presented and illustrated using the example of the Middle East agglomeration region of East Jerusalem.

1.2 About this report

This report presents a novel approach to constructing spatially referenced, multi-dimensional virtual city models from remotely generated point clouds for areas for

which no reliable geographical reference data is available. A multidimensional point cloud is an unstructured array of single, irregular points in a spatial 3D coordinate system with time stamp. If geospatial reference points are available, a point cloud is georeferenced. Georeferenced point clouds contain a high-precision reference data set. Point clouds can be used in a variety of applications. They are particularly suitable for the representation of surfaces, structures, terrain and objects. Point clouds are used here to create a virtual 3D city model that represents the complex, granular cityscape of Jerusalem and its centre, the Old City (see figure 2). The generation of point clouds is based on two data acquisition methods: active data acquisition by laser scanning and passive data acquisition by photogrammetric methods. In this case, very high-resolution stereo images in visible light and in the near-infrared range were systematically acquired by a flight campaign (see figure 1). The collected spatio-temporal data require further processing to extract the necessary geographical reference and semantic features at a specific resolution and scale. An insight into the processing of an unstructured point cloud is given in order to extract and classify the 3D city structure and to reconstruct its objects. Finally, tailor-made, precise and up-to-date geographical data sets can be provided for a defined region in a defined resolution and scale.

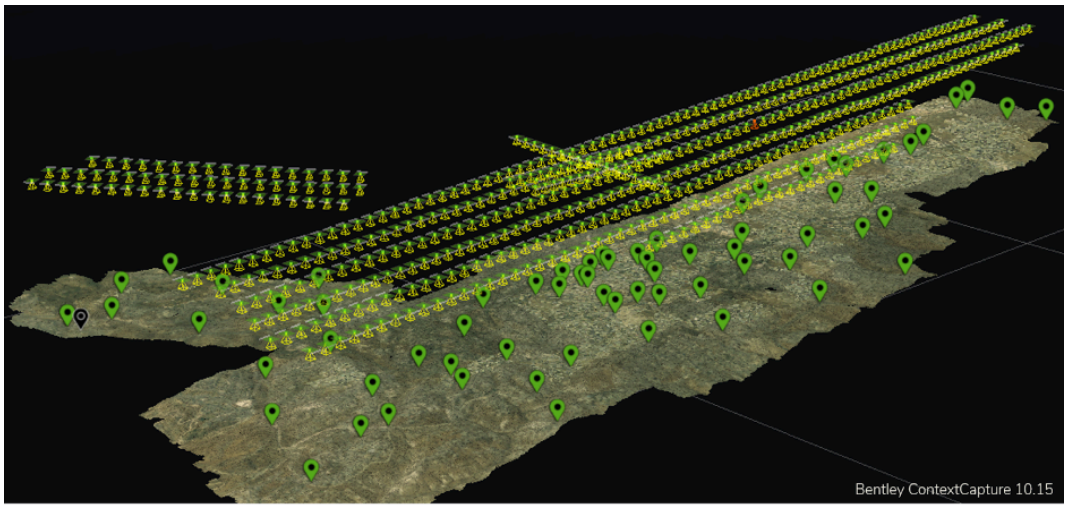

Figure 1: Overview of the East Jerusalem study area with camera positions (yellow markers, 571 images) and ground reference points (green markers, 69 dGPS points). In the central area of the picture, where the flight strips overlap, is the Old City of Jerusalem with the Templedome The area of the study area is a fair 190 sqkm.

1.3 Thematic scope and application context

In the context of an ongoing EU project, this approach will be applied and evaluated using the example of the East Jerusalem urban region. The aim of this R+D project is to provide East Jerusalem's citizens and decision-makers with a high-resolution three-dimensional city model with dedicated spatial data and system functions that enable civil society to perform its tasks. In many cases, suitable spatially related basic data can be used for such issues. This is not the case here. Due to the complex geopolitical situation, no usable up-to-date geodata are available for model and system construction. Consequently, the entire production process must cover the entire process chain from data acquisition, processing and visualisation to spatial evaluation or analysis. In this doctoral project I am therefore developing a flexible, largely automated modular process that includes all the sub-complexes mentioned above and is transferable to comparable spatial starting points. In fact, on a global scale, the lack of suitable basic geodata is the rule for the generation of three-dimensional urban spatial models. The latter have considerable relevance for modelling and monitoring large-scale urbanisation and the formation of megacities.

2 Approach

This approach deals with a novel way of constructing a georeferenced virtual city model of East Jerusalem from unorganised, unstructured point clouds. As we deal with a granular cityscape, a particular focus is on the classification and extraction of building models [7, 8, 11]. The novelty of our approach is the fact that we adapt the processing of invariant point data in terms of spatio-temporal reference to the construction of multidimensional city models. In that way, a generic processing (and acquisition) technique is made available for the construction of both geospatial cityscapes and landscapes in areas where no reliable reference data are available. During the construction process, an elastic grid is generated by means of voxels in different levels of resolution and hierarchy, which represents and maps the point cloud [2, 18]. Applying established methods of computer graphics, object surfaces are then projected and constructed using iterative resampling methods [1, 14]. Particular attention is paid to the point consolidation process, with respect to uniform point distribution as well sufficient sampling density at minimum distance from point to voxel. In addition, the requirements point clouds need to comply with in order to perform the process in multidimensional point clouds, are checked during pre-processing [17]. The solution approach outlined above is developed within the scope of a wider R+D effort that deals with the with the creation, (re-)construction, administration and maintenance of virtual 3D city models in constantly changing urban environments [4, 5, 6]. An essential prerequisite for this work is the volume and quality of the geospatial source data. The more precise the source data record or map geographical reality, i.e. spatio-temporal object features or geometric and multi-spectral resolution, the higher the degree of the pseudo-realistic visualisation of the resulting city model can be [13].

2.1 Point clouds as a comprehensive basic data set

To date, remotely sensed point clouds can best meet the above requirements. Point clouds can also easily capture the spatio-temporal dynamics typical of urban environments. Point clouds easily map 3D surfaces and objects by a densely spaced sequence of points storing the detected geometry of objects and surfaces point-wise. Thus the totality of points, the point cloud, is a versatile geospatial data representation of complex terrestrial surfaces, such as cityscapes [13]. Because of their geometric nature, point clouds are particularly well-suited for the derivation and subsequent visualisation of geospatial phenomena and artefacts (see figure 2). Other than classical vector and raster data, point clouds facilitate the extraction of a regional, spatially invariant, geometric base structure [3, 15]. Any processing of point cloud data requires to solve a twofold problem: How can addressable objects from any kind of geodata be connected in a meaningful way, and how can these objects be analysed for their potential added value? Solving these R+D issues is considered a major challenge in the interdisciplinary field of spatial sciences [8, 12]. Based on the lowest common denominator, the space-time reference, two possible solutions can be identified to model high to very-high resolution 3D geo-objects: schematisation and reconstruction.

Figure 2: Section of a point cloud of the Old City of Jerusalem in true colour. In total, the invariant mass data set comprises almost 20 billion points, with a storage requirement of almost 500 GB, a point spacing between 0.511-0.023 cm and an average horizontal point density of 119 pts/sqm.

2.1.1 Schematisation

This research field deals with the procedures required to schematise available geodata. To link existing data with our without a spatial reference a scheme is essential to

do this in a meaningful way. A common scheme is therefore mandatory to facilitate successful harmonisation and fusion of data [15]. Schematisation of different input data is hampered by the data reference which is both spatial and semantic. When source data have different spatial as well as semantic references, which generally is the case, schematisation may result in a loss of reference accuracy. In addition, data schematisation, like, e.g., interpolation, is a one-way process. As a result, the original source data are irreversibly altered. This is compensated by the fact that geospatial data different sources can successfully be processed jointly and stored in uniform datasets. Today, structured schematisation, harmonisation and integration of different data forms and formats is implemented in the quasi-standard Feature Manipulation Engine (FME) or in the INSPIRE Guidelines of the EU. In contrast, schematisation of references is a somewhat disregarded R+D field as it directly affects the accuracy of geodata.

2.1.2 Reconstruction

This research field investigates the use of computational engineering methods to approximate an artefact or semantics by constructing both virtually. Hence, an artificial reconstruction of the original data is carried out without directly manipulating them, thus creating new data while leaving the source data unaltered. This is the rationale behind machine learning and artificial intelligence in general. Although both schematisation and reconstruction are roughly based on the same principle, they differ in their handling of source data. This is apparent when it comes to the application of 3D building models which have massively increased in importance far beyond simple visualisations in times of so-called digital twins [5]. One driving factor behind this development is the rapidly growing availability of high-resolution input data. What is lacking, however, is an executable generic process applicable to all current domains and solutions. For the time being, only domain-specific approaches can be found. Consequently, the level of abstraction is currently limited to the domain and therefore increasingly schematised.

3 Generation of geo-referenced point clouds

3.1 Characteristics

A multidimensional point cloud is a 3D or 4D data set of a single, irregular and unstructured point array in a spatial 3D coordinate system defined by x, y, z coordinates and the time component [17]. The result is a registered point cloud. If reference points are available, a point cloud is geo-referenced. Assuming a spatial reference, point clouds contain an high-precision reference dataset, since every single point has a very high position accuracy and fidelity compared with classical vector and raster spatial models. Therefore, referenced point clouds are often used for secondary referencing of other geodata. Point clouds can be employed in a variety of applications. They are particularly suitable for the representation of surfaces, structures, terrain and objects [16]. Point clouds are used for documentation purposes or fur-

ther processing in, e.g., CAD, BIM or in 3D rendering software for 3D modelling or 3D visualisation [13].

3.2 Data acquisition, database

Implementing the approach detailed above, aerial imagery obtained from a flight campaign is evaluated photogrammetrically to generate a point cloud for the study area of East Jerusalem. A total of 571 RGBI (red, green, blue, near infrared) nadir aerial images are available acquired in two flight campaigns of 23.09.2019 with a ground resolution of (minimum) 10 cm GSD, taken with a camera system of the Z/I DMC 250 II e type (see figure 1). The first campaign covers the entire investigation area of East Jerusalem with 525 images (forward overlap 80 percent, side overlap 60 percent). The imagery covers an area of nearly 190 square kilometres, corresponding to a data volume of almost 135 gigapixels. The second campaign covers the Old City of Jerusalem only with 46 images (forward overlap 80 percent, side overlap 80 percent) encompassing an area of nearly 13.5 square kilometres, which corresponds to a data volume of almost 11 gigapixels. This campaign is located within the area covered by the first Campaign. Corresponding pixels have been recorded in at least 10 different aerial images in the entire study area, with exceptions in the peripheral areas. In the area of the Old City, corresponding pixels are present in at least 30 different images due to the overlay of the flight campaigns. A total of 70 ground reference points (differential GPS), equally distributed across the study area, are available for georeference. All aerial images are referenced by GPS and INS. Bentley's ContextCapture (v14) software system is used to perform referencing of the input imagery as well as of 69 out of 70 ground points, and is also employed for bundle block adjustment (see figure 1). Essential parameters are the collinearity equations at pixel level and to maximally increase the quality of matching. The processing is performed on a mini-cluster consisting of two identical workstations. An AMD Ryzen 9 3900X 12-core processor, 64 GB, NVMe memory and NVIDIA GeForce RTX 2080Ti are used for the calculation. The mini-cluster allows, among other things, parallel computation and simultaneous use of several graphics cards. It shows that a very consistent dataset or bundle block is provided, since the global error with ground reference points is about half a pixel; the median is even slightly below. Turning to the point clouds generated, it can be seen that these cover massive volume of data and, with more than 100 points per square meter, represent the full pixel density of the source data. Note that this number refers to a 2D reference surface [sqm] only. However, it includes all points of the surface, i.e. the 3D points, too. also. The theoretical maximum for photogrammetric processing in the reference plane X & Y is 100 points per square meter (image data 10 cm GSD), or 91 points in the computed block.

3.3 Structuring of point clouds

Within the framework described above, 3D respectively 4D point clouds are photogrammetrically generated in the pre-processing. Subsequently, the data space is subdivided by hierarchical methods, such as octree or n-tree. Octree, for example,

defines a hierarchy that can be used to inspect large point clouds and interact with them with high performance [13, 15]. An octree is a data structure for indexing three-dimensional data. It extends the concept of binary trees and quadtrees, which structure 1D or 2D data. Each node of an octree represents a volume as a cuboid. Furthermore, these cubes are often aligned with axes of the coordinate system. Each octree node has up to eight subnodes. If a node has no subnodes, the corresponding cube can be represented uniformly and no further subdivision is necessary. Each node of the spatial system with associated subnodes is characterised by the presence or absence of points. In the representation of volumetric data, such as building objects, a further subdivision of a node is not necessary if the mapped volume is completely uniform and a reference plane is known. Nodes that have points will become part of a spatially adaptive, elastic hierarchical voxel grid in the further process. Elasticity includes the possibility to use different resolutions in a grid in an advantageous way.

Volume graphics is a different method of modeling as a voxel data set [1]. Here a value is recorded at a single point from the object at regular intervals. The result is a cube-shaped 3D grid of voxels. These voxel models can easily be converted into images, which, depending on the minimum resolution, can be extremely pseudo-realistic and detailed [2]. This type of modelling requires significantly higher resources in direct comparison to triangle-based polygonal meshes and has a performance loss in visualisation compared to a polygonal surface model of similar size. In addition, the manipulation of these data is much more complex [1]. The ultimale goal of all algorithms briefly touched on here is to model 3D objects [4]. This is usually done, even on the most modern graphics hardware, by means of simple triangular structures, which in mass reproduce and represent complex polygonal surfaces [14]. Also, despite the inclusion of complex 3D textures in the volume graphics, the hardware acceleration is still focused on triangle computation. Hence a combination of both approaches proves to be expedient, as the example of the Marching Cubes algorithm from 1987 in imaging medicine proves [10]. For the first time it was possible to approximate ineffcient volume models by effcient polygonal surface models, to visualise them effectively and to combine the advantages of both approaches [2].

3.4 Generation of urban 3D objects using the Marching Cube algorithm

The core idea of Marching Cubes is to decompose a given voxel model of an object into small cubes and then march from one cube to the next and determine how the surface of the object intersects the respective cube [10]. A selected threshold value regulates which parts of the relevant cube lie inside or outside the object. The distinction between solid and transparent is fundamental to the procedure. It affects the normal calculation of the surface, since the slope of the scalar field at each voxel point is also the normal vector of a hypothetical iso surface running from that point . There are 256 possibilities of how an arbitrarily shaped surface can divide a voxel into interior and exterior areas, as, based on combinatorics, there are 2 to the power of 8 possibilities to divide the eight corners of a voxel into two disjoint sets inside and outside [10]. Due to symmetry effects, however, the number

is reduced to only 15 different variants. The so-called Triangle Lookup Table contains all these possibilities. The runtime of the classical marching cube algorithm depends significantly on the number of voxels considered. An optimisation is based on the hierarchical investigation beforehand, as described, to use only voxels as input that contain points and thus represent an object [9, 18]. It emerges that the Marching Cube algorithm represents an effective technique for the calculation of iso-surface and object modelling in 3D [18]. Consequently, a combination of volume graphics and triangle-based polygonal meshes based on a well-established algorithm can be used to approximate and reconstruct the surface and thus the hull of an object [9]. Finally, this geometry requires to be characterised and extracted as an object in order to enrich it with spatially referenced semantics for use in a spatial information system of the entire study area.

The essential work steps in this context include a meaningful subdivision of a point cloud [17]. This has to be manageable for algorithms working on it. The subdivision can be based on either the data structure, as described, using a hierarchical tree structure, or with spatial or structural filtering within the process. Similar to a segmentation, this reduction represents an adjustment factor, since an object can occur in several areas, nodes or tiles [15]. An overlapping of areas to be processed also needs to be considered so that no break between regions occurs. Similarly, it has to be made sure that an object has identical characteristics across regions. In other words, voxels or horizontal slices of a point cloud, as familiar from imaging medicine, are fed into an algorithm [2, 4]. The algorithm reconstructs an object from them bit by bit, as one would do with, say, with a finite number of small Lego bricks. Depending on the resolution, i.e. how high the layers and the voxel are, an object image, for example of a building, is gradually created. If the process is repeated and parameters of resolution are changed, there is also an incremental approach that can model an object in its entirety or in a spatio-temporal manner. This results is a volume model of an object as well as a polygonal triangular model that can be stored in a database. Depending on the process, different levels of detail are conceivable for one and the same object [4].

4 Implementation strategy

In a nutshell, the methodological concept of this approach addresses the research question of how accurately and precisely multidimensional virtual spatial models correspond to the complex reality they present. It challenges the widespread approach to different virtual models by different levels of detail (LOD) [15]. Since the availability of high-resolution output data such as point clouds has increased significantly, it can be assumed that the concept of static LOD degrees is less appropriate to represent very-high resolution complex objects. In addition, discretisation of the source data is always applied when preparing a LOD inventory. It is important to consider the geometric tolerances of existing LODs with regard to quality assurance. In this context, discretisation of the source data to known LODs is possibile but not mandatory. Even without discretisation, the full data depth available can be utilised.

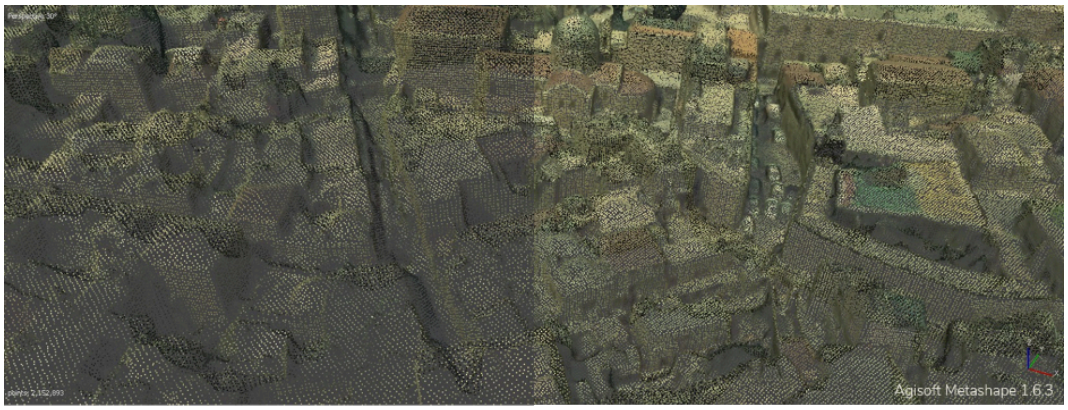

Figure 3: Comparison of two resolution levels of a highly accurate point cloud of the Old City of Jerusalem. Certain objects (such as buildings) can be visually recognised even at low resolution. One focus of the methodical approach of the work is thus also to efficiently assemble the data set with regard to performance without causing a loss of detail.

The above issues need further analysis and evaluation. Tests have shown that the approach outlined here includes some open issues that need to be addressed. Figure 4 shows derived and extracted two-dimensional rings generated with a provisional version of the layer algorithm. It is important to note that this layer is based on the terrain's reference plane only. In the area of each outline, mean heights for the individual object areas can be derived from the point cloud and added to the data as a semantic feature. As shown in figure 4, the result is a 2.5 D data set containing extruded building outlines, which can be semantically augmented with further attributes.

Another issue to be addressed when implementing the algorithm is the reduction of objects to two dimensions. Without exact segmentation and classification of a 2D slice of a point cloud's reference plane, it is sometimes problematic to derive the exact extent and position of an outline generated [7]. This may result in a wrong rotation or over-extraction of an object. A point cloud can also have a too high resolution for an algorithm (see figures 1, 2). Likewise, photogrammetric point clouds of building facade elements may contain few or no points due to missing corresponding pixels. Contrary to that, the Marching Cube algorithm requires a consistent database. Hence, no values can be generated where no values are available. At this point, one option is the reconstruction of buildings by means of voxels. Buildings usually have a cubic or cuboid shape, hence it can be assumed that facades can be reconstructed in a simple geometric way (see figure 4). In contrast, if an object is represented in a point cloud of a very high resolution, and this resolution is not required to represent this object distinctly, resources are wasted (see in contrast figures 2, 4). It is therefore a matter of the correct dimension and an incremental and interactive approach to extract the most realistic object from the point cloud. Potential applications are, for example, the analysis of the deviation between vector, raster and point cloud representations that map a building object. Differences can reveal geometric accuracies between the

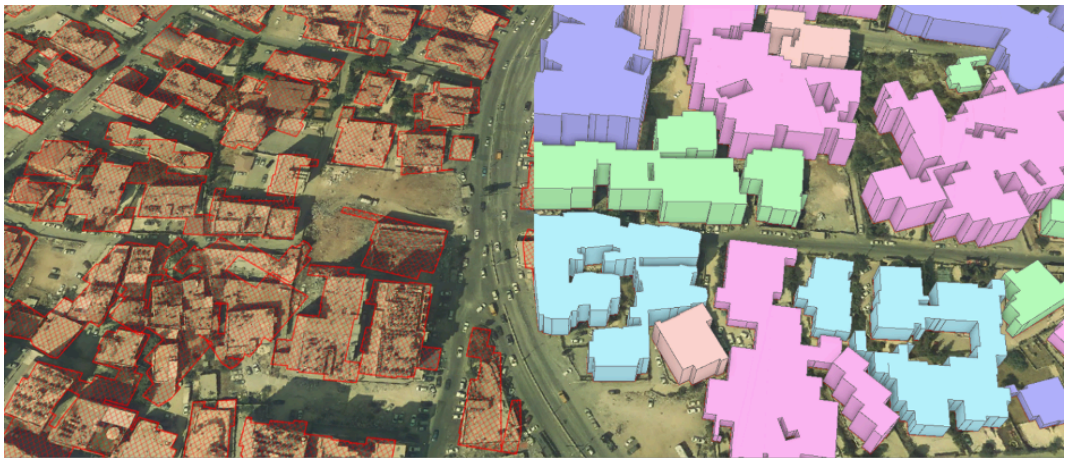

Figure 4: Prototypical implementation and presentation of derived two-dimensional building rings (left) and extruded real building heights with semantic colour coding (right). Challenges lie in the demarcation between individual buildings and the correct orientation in space. This partial aspect characterises the current state of work.

different representations, but also show structural changes. Typically, objects are analysed by algorithms, especially in machine learning, with the help of similarities, and are successively derived and constructed. Consequently, a building model is broken down into its components (i. e. walls and roof) and then, by means of trained similarities, it is determined which part of the object is to be represented and how [3, 7, 11]. This requires a substantial number of potential object components that can be compared to automatically derive the required object component. Overall, the mathematical morphology of point clouds is highly suitable to be processed bin this context, since the most versatile objects from a comprehensive, invariant 4D data set are available in a complete and simple form. With machine learning methods, as with the marching cube algorithm, considerable added value can be gained from geodata with an ever-present link to the original data.

5 Conclusion remarks

Within this doctoral project an innovative approach to construct spatially referenced multidimensional virtual city models from photogrammetric point clouds for areas where reliable geographical reference data are not available is presented.. The work presented here is part of a larger international R+D project that investigates the generation, (re-)construction, management and maintenance of virtual 3D city models in a constantly changing urban environment. To date, the project work carried out has shown that the generation and processing of point clouds can provide quality geodata to create and maintain a uniform, fully referenced 3D geodata base in a limited period of time, if not on an ad-hoc basis. In the study area of East Jerusalem,

the lack of precise, up-to-data geospatial data necessitates the airborne acquisition of geospatial point data conforming to the key requirements of resolution, data quality and topicality. The resulting high-resolution point cloud can be used for a variety of applications. One relevant usage is the extraction of virtual building models from high-resolution point clouds. The use of point clouds is of crucial importance in East Jerusalem, where no primary data source is able to provide geospatial data representing the complex urban reality and its spatio-temporal dynamics. Because of their purely geometric content, point clouds are especially suitable for the derivation and visualisation of geospatial phenomena and artefacts. To extract building models from point clouds, a procedure originating from the field of imaging medicine is adapted, allowing for the modelling of 3D objects in (very) high resolution. Drawing on methods of volume graphics (voxel grids) and classical computer graphics (polygonal triangular meshes), this novel approach facilitates the effective processing of point clouds. In that context, the structuring of the source data is given special attention. Preliminary applications of the approach discussed here prove that extruded building outlines can successfully be extracted and visualised.

5.1 Future work

According to the current work status, the sub-complexes *data acquisition* and *raw data processing* have been completed. In Q3/4 2019, an aerial survey of the area under personal supervision was carried out in accordance with the data acquisition specifications I had developed (2). From the high-resolution georeferenced flight data, mass data in the form of invariant 3D point clouds were then generated using stereo-photogrammetric methods. A comprehensive, jointly rectified data set is now available in different resolutions (see figure 3). Geometrically, this point cloud represents a reference data set which is the basis for all further processing steps. Currently, I am working on the automated modelling of uniquely addressable planar geo-objects (e.g. buildings) from point cloud data (object formation, see figure 4). Due to the fact that the point cloud data are semantically invariant at first, reliable, reproducible extraction procedures are largely missing. This work shall contribute to this problem. A methodical approach to the formation of complex 3D objects is offered by imaging medical procedures for the three-dimensional visualisation of organs (e.g. Marching Cube Algorithm). The concept of levels of detail of a spatial model will be examined in comparison to the concept of continuous, data-dependent levels of detail. A related research problem I am currently working on is the dimensional assignment of semantic attributes to previously extracted geo-objects (object enrichment). Here, too, there are research gaps, and this work contributes to reducing these gaps. Thus, the doctoral project goes beyond the now mainly established "visualisation" of complex multidimensional urban structures (see figure 2).

5.2 Work plan and outlook

Two issues have directly influenced the original time and work plan. Firstly, the delayed realisation of the flight led to a delay in the provision of raw data (Plan

Q1 2019, Real Q3/4 2019). Secondly, the partial aspect of 'object extraction' and 'addressing' from mass data reveals previously unresolved research questions that need to be addressed. Furthermore, it becomes apparent that the implementation of the generic and modular process chain involves individual problems that can have an effect on the implementation of a process that is as automated as possible. The outstanding aspects of 'functionalities' and 'visualisation' will be adapted within the service-oriented approach. The work plan is constantly compared with the real situation.

5.3 Acknowledgements

The work discussed here is part of a larger R+D project on East Jerusalem with Palestinian, East Jerusalem, and NGO partners funded by the European Union. Part of this research work is supported by a PhD grant from the HPI Research School for Service-Oriented Systems Engineering at the Hasso Plattner Institute for Digital Engineering, University of Potsdam. The funding of both institutions is gratefully acknowledged.

References

[1] T. Akenine-Möller, E. Haines, and N. Hoffman. *Real-time rendering*. Crc Press, 2019.

[2] J. Ashburner and K. J. Friston. "Why voxel-based morphometry should be used". In: *Neuroimage* 14.6 (2001), pages 1238–1243.

[3] C. Brenner. "Building reconstruction from images and laser scanning". In: *International Journal of Applied Earth Observation and Geoinformation* 6.3-4 (2005), pages 187–198.

[4] A. Fricke and H. Asche. "Constructing Geo-Referenced Virtual City Models from Point Cloud Primitives". In: *International Conference on Computational Science and Its Applications*. Springer. 2020, pages 448–462.

[5] A. Fricke and H. Asche. "Geospatial Database for the Generation of Multidimensional Virtual City Models Dedicated to Urban Analysis and Decision-Making". In: *International Conference on Computational Science and Its Applications*. Springer. 2019, pages 711–726.

[6] A. Fricke, J. Döllner, and H. Asche. "Servicification–Trend or Paradigm Shift in Geospatial Data Processing?" In: *International Conference on Computational Science and Its Applications*. Springer. 2018, pages 339–350.

[7] N. Haala and M. Kada. "An update on automatic 3D building reconstruction". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 65.6 (2010), pages 570–580.

[8] T. Hall and H. Barrett. *Urban geography*. Routledge, 2018.

[9] M. Huang, P. Wei, and X. Liu. "An Efficient Encoding Voxel-Based Segmentation (EVBS) Algorithm Based on Fast Adjacent Voxel Search for Point Cloud Plane Segmentation". In: *Remote Sensing* 11.23 (2019), page 2727.

[10] W. E. Lorensen and H. E. Cline. "Marching cubes: A high resolution 3D surface construction algorithm". In: *ACM siggraph computer graphics* 21.4 (1987), pages 163–169.

[11] H. Mayer. "Object extraction in photogrammetric computer vision". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63.2 (2008), pages 213–222.

[12] S. Nedkov, G. Zhelezov, N. Ilieva, M. Nikolova, B. Koulov, K. Naydenov, and S. Dimitrov. *Smart Geography*. Springer, 2020.

[13] R. Richter, S. Discher, and J. Döllner. "Out-of-core visualization of classified 3d point clouds". In: *3D Geoinformation Science*. Springer, 2015, pages 227–242.

[14] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[15] V. Tolpekin and A. Stein. *The core of GIScience: A process-based approach*. University of Twente, Faculty of Geo-Information Science and Earth, 2012.

[16] G. Vosselman, S. Dijkman, et al. "3D building model reconstruction from point clouds and ground plans". In: *International archives of photogrammetry remote sensing and spatial information sciences* 34.3/W4 (2001), pages 37–44.

[17] M. Weinmann. "Preliminaries of 3D point cloud processing". In: *Reconstruction and Analysis of 3D Scenes*. Springer, 2016, pages 17–38.

[18] Y. Zhou and O. Tuzel. "Voxelnet: End-to-end learning for point cloud based 3d object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pages 4490–4499.

Testing Distributed Application in Co-Simulated Environments

Arne Boockmeyer

Operating Systems and Middleware Group
Hasso Plattner Institute for Digital Engineering
arne.boockmeyer@hpi.de

Distributed applications are widely established in all kinds of daily life. Even in critical areas like the German railway system, distributed systems exist. For these, there are higher requirements for testing.

Testing of distributed applications is mostly done by field tests or laboratory tests. But both methods are limited to a small set of conditions. To test applications in more diverse scenarios, simulation-based testbeds are one solution. These testbeds simulate an execution environment and place the application on virtual or hardware nodes in that environment. This enables realistic test scenarios in many diverse situations. Cohydra is such a testbed which also includes techniques for fault injection testing.

1 Motivation & Use-Case

Nowadays distributed applications are more and more used in research and industry. Even in a conservative company like the Deutsche Bahn the usage of distributed applications is increasing. One example of such a distributed application was one use-case of the research in the project Rail2X: A future switch can record maintenance data about its state. This data is useful for predicting outages of the switch which helps to prevent train delays or blocked tracks. But in structurally lagging regions it is difficult to transmit the recorded data to the cloud-servers where the evaluation of the data happens. A researched solution is shown in Figure 1. The basic idea is that the switch is transmitting the data to a train which passes the switch. The train is collecting the data and transmit it to the cloud-servers in the next train station with a good internet connection.

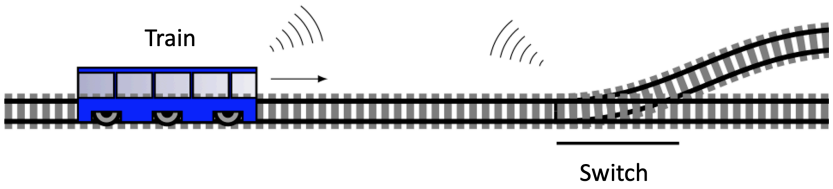

Figure 1: One possible solution to transmit the data from the switches is to collect the data with passing trains and deliver it to the cloud in the next train station

2 Distributed Applications

In this report the definition of distributed application from Tanenberg and Steen is used [5]:

> "A distributed system is a collection of independent computers that appears to its users as a single coherent system."

This means the described use-case contains a distributed application. Each infrastructure element – the train and the switch – need a part of the software system and these parts communicate over some communication channel. The usual communication channel in the area of mobility is the Vehicle-To-X (V2X) standard which bases on Wifi 802.11p.

Besides that, for the user, the application appears like a single coherent system because the user can access the data from a single interface.

3 Testing Distributed Applications

Before such an application can be used in practice, it has to be tested and verified. But testing a distributed application especially with a focus on mobility is difficult. The software-system has to operate without software fails under several different conditions. These are for example:

- Different train speeds: The speed of the train defines how long the two parts of the software can communicate. When the train moves slowly the communication time is longer compared with a faster train. Depending on the data size this can have an impact on the success of the transmission.
- Different train schedules: Assumed that the switch records the data and stores it locally, a different train schedule can increase or decrease the amount of stored data. So, it is possible that the application crashes because of an out of memory failure.
- Different weather conditions: Since the software-system communicates over a Wifi-connection the weather can have a serious impact. The communication can work perfectly when the weather is good, but it can have outages when it's raining or snowing outside.
- Different environments: Depending on the vegetation and infrastructure around the switch the communication can be disturbed as well. It's common knowledge, that buildings have a negative impact on communication ranges of wifi-signals. This can have an impact on the success of the transmission.

When testing distributed applications, the tester has to keep these different conditions in mind. To test these applications, testers mostly use two different methods: Field tests and laboratory tests. Both are briefly explained in the next two subsections.

3.1 Field Tests

The first idea to test distributed applications are field tests. Having the software system and the hardware requirements (like the hardware system in the train and the future switch) it is the first idea to install the software on the hardware and test the application by moving the train over the switch. On the one hand, for the single test case, this is the most realistic way to test the application. All conditions which would also apply in the practical usage have the same impacting during testing the application. On the other hand, performing a field test covers just one specific set of conditions. The parameter of the train (like speed or schedule) is limited to a small set of variations and the environment is mostly static. Besides that, a field test is expensive compared to other testing methods [4]. Additionally, it is unlikely to have a repeatable testing result because in field tests it is hard to repeat the test with exactly the same conditions. Furthermore, a field test can only take place at one of the last steps of the development because all parts of the system (hardware- and software-requirements) have to be fully developed. This contradicts the common software-pattern to test as early as possible.

3.2 Laboratory Tests

To especially address the test early pattern, another method to test distributed applications is a laboratory test. A laboratory test means to test the software on common devices in a controllable environment. Figure 2 shows such a laboratory test with EL-20 nodes that are able to communicate with Wifi 802.11p.

Testing distributed application with this method have a big advantage that tests are early possible and compared to field tests cheap since the infrastructure elements are not required. Because of the fully controlled environment in a laboratory, the tests are also repeatable. But this method of tests loses the realism of the tests. Since it is not in the field or some comparable environment it's unclear if the results are applicable for the reality.

4 Simulation based testing in testbeds

To summarize, field tests are realistic since they run the tests in real environments. But they are limited to a small set of conditions. Laboratory tests are also limited to a small set of conditions especially in the area of the environment. But compared to field tests they can be applied in the early stages of development. Since both methods have their disadvantages, a third method is required, which combines the advantages and zero out the disadvantages. This method needs to place the software system in a controllable but realistic environment. Besides that, the testing should be cheap and possible in the early stages of development. Simulation-based testbeds are one possible solution [1]. Like it is shown in Figure 3, a testbed simulates an environment and places the application in that specific environment. Therefore, it

Figure 2: A laboratory test means testing the software in a controllable environment. These tests are easier to realize, but less realistic.

uses a huge set of different technologies like different simulations, virtualization, and the connection of real hardware (so called hardware-in-the-loop) [6, 7].

The following subsections describe how to realize such a testbed.

4.1 Simulating the environment

The first important step is simulating the environment. Simulation in the case means, that a model of the real-world system (like the traffic) was created and will be used in the simulation. The realism of the simulation depends on the quality of the model [4]: When the model behaves like the real world in most aspects, it is realistic and can be used for the simulation. Otherwise, the results would be not applicable to the real world. Having such a model gives the opportunity to behave like in the real world and try the same scenario under a lot of different conditions. it is also possible to replay a scenario with exactly the same conditions. In particular, it is also cheap since the environment will only be simulated.

In the given use-case the focus lays on two different simulations. The first simulation simulates the network connection between the nodes (the train and the switch). Another simulation simulates traffic and vehicle behavior. Since there are now two simulations from different domains involved, it is unlikely that a single model covers both aspects.

To solve this issue, a technique called co-simulation can be used. Figure 4 shows the idea of a co-simulation. Two standalone simulations are working together to simulate a joint result. In the described use-case it is useful that the traffic simulation

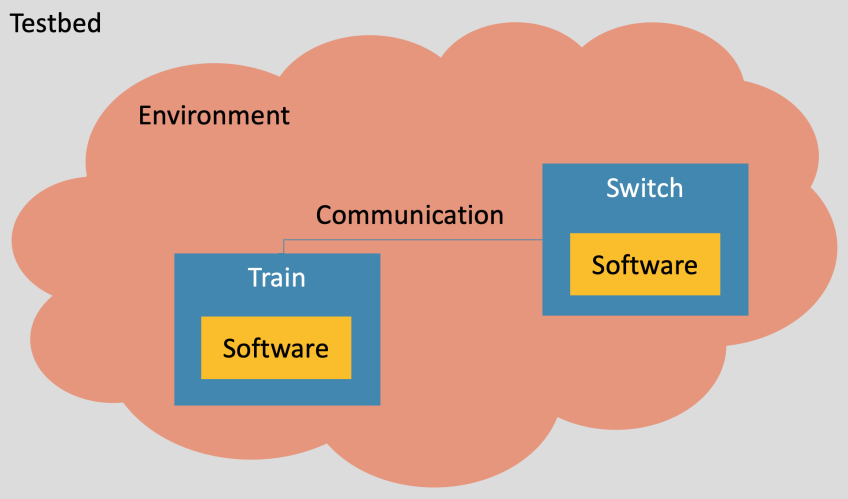

Figure 3: The structure of a testbed

simulates the movement of the vehicles and sends the vehicle positions to the network simulation. The network simulation uses vehicle positions to calculate the quality of the network connection. To manage the data exchange between the simulation an orchestrator is used.

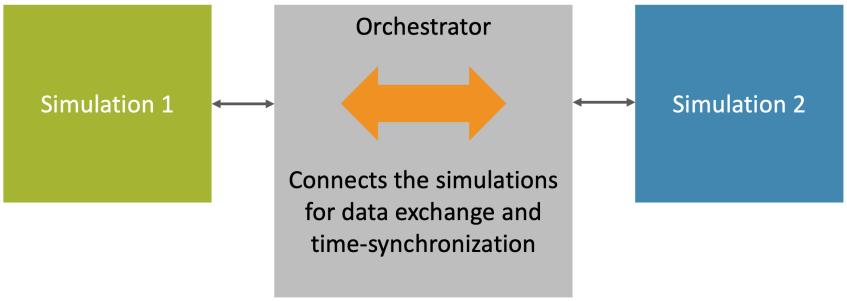

Figure 4: The structure of a co-simulation. The orchestrator controls both connected simulations

Having these two simulations in a co-simulation, two aspects of the simulation-based testbed are realized: The environment and the network connections are implemented.

4.2 Execution Environment for the software system

The next required step is to integrate the software system in the simulation. Therefore a runtime container for the software is required. In general, there are three different options:

- Simulate the software as part of the simulation: In some simulation environments, especially in the area of network simulation, it is possible to implement a simulation of the software as well. This simulated execution of the software gets a simulated (and abstracted) hardware system, operating system, and internet stack. This solution has the big disadvantage that realism is unclear because of the simulation. Since the other two options are more realistic, this option will not be followed up.
- Run the software on virtualized nodes: Running the software in some kind of virtualized containers (like Docker, LXD, or similar) real operating systems and internet stacks are used. Besides that, virtualized nodes are easily scalable. This is a pro when talking about huge simulations. On the other hand, the containers will run on laboratory computers. That means the future target hardware is not involved.
- Run the software on hardware nodes: To compensate for the disadvantage of virtualized nodes the software can be executed on the target hardware system. This results in the most realistic execution environment for the software system. But this option requires that the target hardware system exists and is available. Besides that, having real hardware as part of the test execution the costs for the execution are increasing again.

A good testbed would need to combine the last two methods to have a cheap and scalable option as well as a realistic option with real hardware. Having this combination opens the option to test parts of the hardware as well. For example, in the case the hardware of the train exists, that hardware could be tested while the hardware of the switch does not exist.

4.3 Hybrid Testbeds

Testbeds that combines simulated environments with virtualized nodes and hardware nodes are called hybrid testbeds [6, 7]. Figure 5 shows the structure of a hybrid testbed. Compared to Figure 3 the representation of the nodes and the simulation is more detailed. The simulation host is the runtime environment of the simulation. It executes both simulations and all virtual nodes. it is also connected to the hardware nodes and integrates them into the simulation.

Hybrid testbeds fulfill the requirements:

- They allow early-stage testings since the environment is simulated and virtual nodes can be used [2]. Both also make the test environment flexible enough to cover a lot of different conditions for the software system.
- They allow cheap large-scale tests since virtualized nodes are included.
- They allow realistic tests since real hardware systems can be included.

How good the requirements are fulfilled depends on the balance between hardware and virtual nodes. If many virtual nodes are used and because of the simulation, the

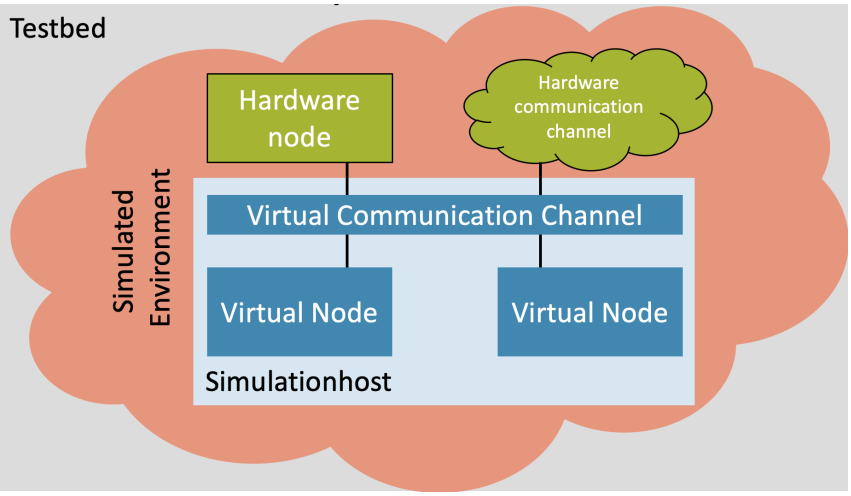

Figure 5: The more detailed structure of a hybrid testbed

quality of the results also depends a lot on the quality of the models [4, 6]. To verify this is an important task of the testbed creator.

5 Cohydra

In my master thesis, I developed the concept of a hybrid testbed addressing this kind of use-cases. Thanks to the Bachelorprojekt AP 19/20 an extensive implementation was created earlier this year. The resulting hybrid testbed is called cohydra[1].

It follows the structure which was explained above. Technically it orchestrates the network simulator ns-3[2] and the traffic simulator SUMO[3] As virtual nodes Docker and LXD nodes are supported, QEMU support is in development. All kinds of hardware nodes are supported when they support IP over ethernet. Besides that, it is also possible to integrate hardware networks in the simulation as shown in Figure 5. Furthermore, cohydra contains a framework to create fault injection scenarios to evaluate the reliability of the tested software system [3].

6 Future Work

The concept above and the implementation cohydra are still under development. Conceptual next steps are including more environment simulations to increase the realism of the testbed and to support more use-cases. Potential ideas are weather

[1] https://github.com/osmhpi/cohydra
[2] https://www.nsnam.org/
[3] https://www.eclipse.org/sumo/

simulations and water-flow simulations. Especially the last one is part of a cooperation with the TU Berlin in a project called Marvis. Technically a current research topic is the distribution of the simulation to scale-out the simulation host. Besides that, scheduling virtual nodes and the distribution of the software system are part of our research.

References

[1] A. Boockmeyer, J. Beilharz, L. Pirl, and A. Polze. "Hatebefi: Hybrid Applications Testbed for Fault Injection". In: *22nd International Symposium on Real-Time Distributed Computing (ISORC)*. IEEE. 2019, pages 97–98.

[2] K. Dooley. "Simulation Research Methods". In: *The Blackwell Companion to Organizations*. John Wiley and Sons, Ltd, 2017. Chapter 36, pages 829–848. ISBN: 978-1-4051-6406-1. DOI: 10.1002/9781405164061.ch36. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/9781405164061.ch36.

[3] C. Giuffrida, A. Kuijsten, and A. S. Tanenbaum. "EDFI: A Dependable Fault Injection Tool for Dependability Benchmarking Experiments". In: *IEEE 19th Pacific Rim International Symposium on Dependable Computing*. Dec. 2013, pages 31–40. DOI: 10.1109/PRDC.2013.12.

[4] C. Sommer, R. German, and F. Dressler. "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis". In: *IEEE Transactions on Mobile Computing* 10.1 (Jan. 2011), pages 3–15. ISSN: 2161-9875. DOI: 10.1109/TMC.2010.133.

[5] A. S. Tanenbaum and M. van Steen. *Distributed Systems: Principles and Paradigms, 2. Auflage*. Prentice Hall, 2007. ISBN: 978-0-13-239227-3.

[6] A. Zimmermann, M. Gunes, M. Wenig, U. Meis, and J. Ritzerfeld. "How to Study Wireless Mesh Networks: A hybrid Testbed Approach". In: *21st International Conference on Advanced Information Networking and Applications (AINA '07)*. May 2007, pages 853–860. DOI: 10.1109/AINA.2007.77.

[7] A. Zimmermann, M. Gunes, M. Wenig, J. Ritzerfeld, and U. Meis. "A Hybrid Testbed for Wireless Mesh Networks". In: *1st Workshop on Operator-Assisted (Wireless Mesh) Community Networks*. Sept. 2006, pages 1–9. DOI: 10.1109/WOACN.2006.337189.

Causal and Sequential Decision Models of Imperfect Fault Understanding

Christian Adriano

System Analysis and Modeling Group
Hasso Plattner Institute for Digital Engineering
christian.adriano@hpi.de

I investigate the phenomenon of imperfect fault understanding, which consists of incorrect or incomplete explanations about the code problem that is causing a software failure. This phenomenon impacts the cost of debugging and hinders the adoption of debugging tools. My approach is to build causal and sequential decision models to identify the causal mechanism that affect the accuracy and the efficiency of fault understanding produced by programmers. Fault understanding is captured by a set of attributes of code inspection tasks on real bugs from popular open source software projects. The results consists of a set of causal and sequential models that allow to make predictions about the accuracy of a given fault understanding and make interventions to improve the efficiency of producing a correct fault understanding. I discuss these results with respect to future work and the implications on the processes and tools supporting on-boarding, bug triage, and debugging.

1 About This Report

This is my fourth Fall report. In the next two sections I review my previous results and the results obtained through the year of 2020.

In the previous reports, I investigated prediction factors of fault understanding and how these factors could be used to decide which program statement to inspect next. (Figure 1).

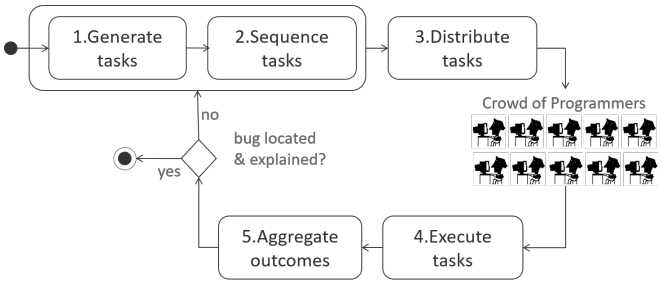

Figure 1: Experimental design

2 Motivation

Software Debugging Programmers investigate the causes of a software failure by debugging software. This involves generating and refining hypothesis [16] about the location of the problem in source code (the software fault), relate it to the failure, and devise fixes [22] for the fault.

Cost Software debugging is a tedious and labor-intensive process. Programmers spend 3% to 50% of their time debugging software [10, 20] 50% [2, 16], which was estimated to cost from $59 to $312 billions a year [2, 20].

Speed Quickly resolving software failures has been part of the practice of agile teams, which since the agile manifesto[1] became widespread by achieving the goldilocks of shorter software release cycles without sacrificing quality, cost, or overloading the development teams [4]. Nonetheless, software teams have come under renewed pressure by two external factors: the disruption from failures that happen only in later stages of testing and the need to deploy software multiple times a day. A recent industry report [5] shows that top performing businesses are deploying on average 32 times a day and keeping lead times below one hour, i.e., the time between committing the source code and running it on production. Hence, while teams used to have days to resolve failures, now many teams have only a few hours. To alleviate this pressure on teams, many approaches have been investigated, e.g., improve continuous integration environments, mandatory code reviews, and, finally, make failure resolution quicker to perform (my research focus).

Automated Methods In response to more agility, software engineering researchers developed various automated methods. Static analysis tools like PMC and FindBugs are executed during compilation time to quickly weed out errors that can become potential software failures. Continuous integration tools run unit tests to expose failures before they are moved into production. Spectra-based fault localization [15] methods run hundreds unit tests to identify the program statements that the most probable root-cause of a software failure. Besides creating a safety net to catch software failures introduced during short development cycles, automated methods are also the pillar of many tools that automatically repair root-causes of failures [13]. Hence, agile teams, high frequency deployment, and automated program repair tools depend on the methods that quickly identify root-causes of software failures.

Perfect Fault Understanding Assumptions However, research has shown that identifying a root-cause is not a sufficient condition to resolve a failure. Even when programmers are given the root-cause, there is no guarantee that programmers will unequivocally recognize the root-cause. Parnin and Orso [14] described this as the "perfect fault understanding assumption".

[1] http://agilemanifesto.org

Difficulty to guarantee fault understanding is also an issue when fixes are automatically suggested. Research showed [17, 19] that programmers still need to be involved to evaluate if the suggested fixes are not over-fitting the unit tests ,i.e., making the failing test pass but not generalizing to other test input data. Research also confirmed the need for patches explanations [11], but this is hindered by the difficulty of using "understandability" as a metric to select of patches [12].

In my research I investigate the assumption that fault understanding can be made explicit in written explanations. This assumption is corroborated by the studies on the performance of senior and junior programmers on debugging tasks. Senior programmers are shown to require fewer [7] hypotheses (i.e., explanations) to correctly identify a root-cause of a failure. Moreover, senior programmers' hypotheses have shown present better quality [9] than the ones from junior peers.

Therefore, my research focuses on both the traditional problem of fault localization (identifying the lines of code that need to be fixed) and how to obtain the explanations for the corresponding software failures. I named this dual outcome as a failure resolution process, which I investigate with respect to its efficiency (speed), efficacy (positive impact on bug fixing), and predictability.

2.1 Related Work

Factors that influence fault understanding were investigated in the context of fault localization [6], code reviews [1], debugging tools [9], static analysis tools [14, 21], However, these studies have two shortcomings: (1) they did not uncovered causal factors (only on correlational ones) and (2) they did not investigate the efficiency of generating fault understanding explanations.

The reliance on correlational factors implies that confounding factors might inadvertently affect the outcomes of program comprehension [18], which is part of the fault understanding task. The lack of knowledge about confounding factors reduces the ability of current methods and tools to influence fault understanding. The reason is that unknown confounding prevents tool designers to determine the which factors can be manipulated (intervened) to improve fault understanding for a certain type of bug and programming skill group.

We aim to mitigate the effect of confounding by building causal and sequential decision models. We use these models to compare different intervention mechanisms and their effects (potential outcomes). The strength of these potential outcomes can inform cost benefit analysis of new features in methods and tools for better onboarding of new programmers, delegation of bugs to programmers, and debugging.

2.2 Investigation Approach

My method obtains explanations to the failure root-causes by the execution of human-judgment tasks. The method was inspired on a MapReduce [3] model that has been successfully combined with crowdsourcing [8] to partition complex work into smaller and independent tasks (microtasks), distribute them, and aggregate their individual outcomes into one failure resolution result.

Experimental Design To build the causal inference models we performed two large field experiments and one lab-controlled experiment with programmers inspecting and suggesting fixes for real bugs from popular open source software projects. In the first experiment, we partitioned work by instantiating template questions about ten failing Java methods from popular open source projects. We recruited 777 programmers on the Mechanical Turk (MT) platform, who executed 4476 microtasks. The aggregated outcomes of these microtasks correctly predicted and explained six out of the ten root-causes of the software failures. In the second experiment, we focus on investigating ways to improve the efficiency of the method. We recruited 654 programmers who performed 2580 microtasks to work on different set of eight software failures from popular open source projects. In the third experiment we have a controlled setting where experienced programmers received different types of fault understanding explanations (including no explanation at all) and we asked these programmers to suggest fixes to the failures. Programmers were randomly assigned to different sets of tasks that involved different bugs and different types of explanations. Since we had 3 types of explanation (including control) and three types of bugs, and we wanted to have at least 3 person by condition, we needed 18 programmers. This allowed to allowed to perform both between and within-subject comparisons.

3 Research Problems

Problem 1: Prediction Fault understanding tasks have multiple attributes that can be combined in different cause-effect relations, i.e., an exponential number of graph models (or mechanisms). We express this problem in the following research questions: What are the models that can predict if a software fault was correctly recognized? How do these models compare to each other? How much replication is necessary to correctly recognize a software fault? How do fault understanding explanations impact the accuracy and speed of bug fixes?

Problem 2: Aggregation In order to identify fault, a programmer has to weigh the many possible suspicious program statements, which corresponds to aggregating the outcomes of multiple tasks. This is particularly problematic when there are multiple plausible causes and explanations for bug (by plausible we mean they can be successfully used to suggest a valid bug fix). Ultimately, our problem is to learn how to aggregate tasks outcomes that correctly resolve the failure. For that, we selected the following research questions: How to compute the probability that competing causes and explanations are plausible? How to learn aggregation models that maximize precision and recall while minimizing the number of source lines to inspect? Our approach was to use different voting mechanisms and estimate the number of votes needed for an aggregation model to correctly identify a bug with a certain level of precision.

Problem 3: Exploration and Exploitation The efficiency of fault understanding corresponds to the number of tasks necessary to correctly identify and explain a software failure. This is difficulty because at any given moment we have the option to obtain more results for the same task (exploit) or execute a different task (explore), or decide to stop executing certain tasks (expire). Hence as questions like: How to decide how many times execute a tasks? (exploitation), How to decide which task to execute next? (exploration), How to decide when to stop executing a certain task? (expiration).

4 Method

I investigated these problems with a method that automatically partitions work in small independent tasks (microtasks) and distributes them to programmers (crowd) recruited on a crowdsourcing platform (Mechanical Turk). Microtasks are automatically generated from template questions that cover the program statements of the source code that failed its unit test (Figure 2).

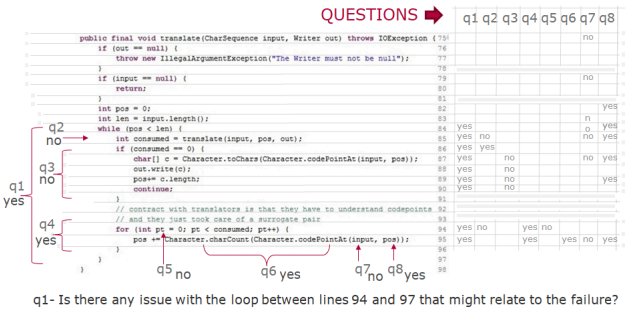

Figure 2: Questions and answers for each source code fragment

For each microtask, the programmers provide their beliefs on a root-cause of given a software failure and justify their beliefs with a written explanation. Their beliefs are expressed in an answer to a question about the possible relationship between a program statement and the software failure. The answers are either YES, NO, or I DON'T KNOW. Programmers are also required to provide their confidence on their answers. To answer these questions, programmers receive the following information: the unit test assertion, the failure message, the source code of the method that failed the test. This information is provided to programmers on web-based interface (Figure 3) that also collects programmers' answers.

The outcomes of microtasks are automatically aggregated by different voting methods. The method selects the best performing subcrowds. This is done by prediction models that were trained with attributes of programmers and tasks.

Although I automated the processes of partitioning, distribution, aggregation, and subcrowd selection, I did not apply any automated debugging method. I purpose-

Figure 3: Web-based GUI for the microtask

fully did so as a first study because I wanted to evaluate a method that is independent of automated debugging tools. This way I sought to avoid possible tool biases and produce a baseline for future automation.

5 Contributions

1. I showed the efficacy of my method for two different designs of microtasks and two sets of real software failures from 16 different popular open source software projects. In the first design, I performed an experiment with 777 programmers, who executed 5705 microtasks. The best group of programmers (subcrowd) located six out of the ten root-causes of software failures, but with high levels of false positives (57% precision and 70% recall). For the second microtask design, I performed an experiment with 654 programmers who performed 2580 microtask. The most effective subcrowd comprised 133 programmers, who answered 836 questions in 33 minutes to resolve all eight software failures with 94% precision and 65% recall at the program statement level. Therefore, my method achieved a competitive efficiency in the second experiment. Efficiency was measured in terms of the number of microtasks, programmers, cost per failure, and the time needed to locate the root-causes of all software failures.

2. I showed that models can be built to predict the accuracy of tasks. Models were built with features (attributes) from programmers and their answers to the microtasks. These predictive models were evaluated across different types of software failures and source code with different sizes and complexities.

3. The aggregation mechanisms based on cardinal voting method performs better than majority voting, proportional voting, and absolute threshold voting. We validated this for both microtask designs and different software failures.

4. I showed that the selected explanations were meaningful and effective to help programmers to suggest bug fixes. Meaningful because explanations were categorized in distinct classes. Effective because programmers with access to the explanations presented higher accuracy in their bug fixes.

5. The answers produced by the crowd of programmers consist of one of the largest publicly available datasets on code inspection tasks for bug finding.

6 Latest Results

6.1 Task selection model

To increase the speed of failure resolution I investigated how to minimize the number of questions asked. My approach was to prioritize questions by the perceived utility to the programmer. I experimented with two formulations: user perceived difficulty and confidence. To evaluate these utility functions I designed an iterative algorithm that samples, aggregates, and ranks questions (Figure 4)

1: Start with one answer per question
2: Loop (**iterations**=10)
3: Aggregate answers
4: Compute question ranking (**classification frontier**)
5: Select top **R questions** as predictors of the fault location
6: Compute statistics (Precision, Recall, LOC)
7: Rank question by **utility value**
8: Select top **U questions to sample**
9: Sample **A** answers from the **U questions**
10: End loop

Figure 4: Algorithm to evaluate the utility-based selection of microtasks. Hyper-parameters of the algorithm in bold font

I explored the hyper-parameter space for a utility function based on the answer confidence. The best results were obtained for the following parameterization: sample one answer per question (A1), rank only the top question to be answered (U1), and select the top three questions (R3) as covering the failure root-causes. This enable to locate and explain all software failures with more than 90% precision, 100% recall (at program statement level) and requiring less than 20% of all questions asked (Figure 5).

6.2 Practical application

In a third experiment I evaluated the impact of the explanations on the perfect fault understanding assumption. The experiment involved 21 software programmers and

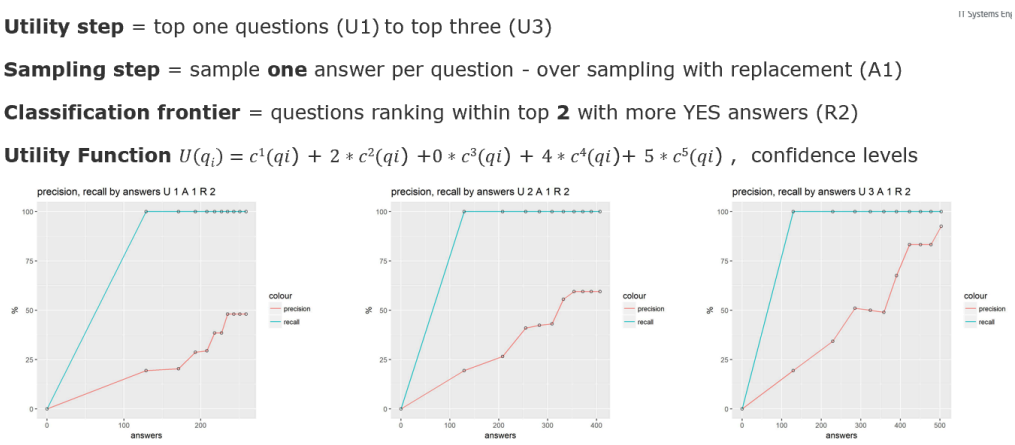

Figure 5: Precision and recall for various hyper-parameter values

three real software failures. These programmers were asked to suggest bug fixes based on the failure resolution produced in the second experiment by the crowd of programmers. I evaluated the correctness of the bug fixes in the presence and the absence of the explanations for the root-causes identified in the second experiment. I found that programmers who relied on explanations were more accurate in their bug fixes. The effect of explanations on bug fix accuracy was statistically significant (two-tailed t-test, p-value=0.0014) and had large effect (Cohen's D = 1.13).

6.3 Publications

I obtained feedback on these results in various venues, for instance, three Dagstuhl Seminars, two FutureSoc Symposium, International Symposium for Empirical Software Engineering, and two SAP workshops in Berlin and Waldorf. Besides these I published the following papers.

- *"Collective Risk Minimization via a Bayesian Model for Statistical Software Testing" published at SEAMS 2020.* In this paper I investigate together with my co-authors the use of a Bayesian model to generate predictions on future needs of software testing. While in this paper we used Binomial distribution and assume a single cycle of update, in my thesis I use Hypergeometric distribution and multiple cycles of belief update.

- *"Learning Utility Changes for Rule-based Self-Adaptive Systems" published at ICAC 2018.* In this paper I developed and evaluated a method to compare rankings produced by competing prediction models. My method consists of an ensemble of three types of similarity metrics that count the difference in ranking positions. This is a generic method to compare the outcome of different task prioritization models.

- *"Microtasking Fault Localization" published at IDOESE 2018.* In this paper I described the design of the three experiments that I performed. The paper also discusses the implications for future work.

References

[1] A. Bacchelli and C. Bird. "Expectations, outcomes, and challenges of modern code review". In: *35th International Conference on Software Engineering (ICSE)*. IEEE. 2013, pages 712–721.

[2] T. Britton, L. Jeng, G. Carver, P. Cheak, and T. Katzenellenbogen. "Reversible Debugging Software". In: *Judge Bus. School, Univ. Cambridge, Cambridge, UK, Tech. Rep* (2013), page 1.

[3] J. Dean and S. Ghemawat. "MapReduce: simplified data processing on large clusters". In: *Communications of the ACM* 51.1 (2008), pages 107–113.

[4] T. Dingsøyr, S. Nerur, V. Balijepally, and N. B. Moe. "A decade of agile methodologies: Towards explaining agile software development". In: *Journal of systems and software* 85.6 (2012), pages 1213–1221.

[5] N. Forsgren, G. Kim, J. Humble, A. Brown, and N. Kersten. *2017 State of DevOps Report*. Puppet, 2017.

[6] Z. P. Fry and W. Weimer. "A human study of fault localization accuracy". In: *IEEE International Conference on Software Maintenance*. IEEE. 2010, pages 1–10.

[7] L. Gugerty and G. M. Olson. "Comprehension differences in debugging by skilled and novice programmers". In: *Papers presented at the first workshop on empirical studies of programmers on Empirical studies of programmers*. Ablex Publishing Corp. 1986, pages 13–27.

[8] A. Kittur, B. Smus, S. Khamkar, and R. E. Kraut. "Crowdforge: Crowdsourcing complex work". In: *Proceedings of the 24th annual ACM symposium on User interface software and technology*. ACM. 2011, pages 43–52.

[9] A. Ko and B. Myers. "Debugging reinvented". In: *Software Engineering, 2008. ICSE'08. ACM/IEEE 30th International Conference on*. IEEE. 2008, pages 301–310.

[10] T. D. LaToza, G. Venolia, and R. DeLine. "Maintaining mental models: a study of developer work habits". In: *Proceedings of the 28th international conference on Software engineering*. ACM. 2006, pages 492–501.

[11] C. Le Goues, S. Forrest, and W. Weimer. "Current challenges in automatic software repair". In: *Software quality journal* 21.3 (2013), pages 421–443.

[12] M. Monperrus. "A critical review of automatic patch generation learned from human-written patches: essay on the problem statement and the evaluation of automatic software repair". In: *Proceedings of the 36th International Conference on Software Engineering*. ACM. 2014, pages 234–242.

[13] M. Monperrus. "Automatic software repair: a bibliography". In: *ACM Computing Surveys (CSUR)* 51.1 (2018), page 17.

[14] C. Parnin and A. Orso. "Are automated debugging techniques actually helping programmers?" In: *Proceedings of the 2011 international symposium on software testing and analysis*. ACM. 2011, pages 199–209.

[15] S. Pearson, J. Campos, R. Just, G. Fraser, R. Abreu, M. D. Ernst, D. Pang, and B. Keller. "Evaluating and improving fault localization". In: *Proceedings of the 39th International Conference on Software Engineering*. IEEE Press. 2017, pages 609–620.

[16] M. Perscheid, B. Siegmund, M. Taeumel, and R. Hirschfeld. "Studying the advancement in debugging practice of professional software developers". In: *Software Quality Journal* 25.1 (2017), pages 83–110.

[17] Z. Qi, F. Long, S. Achour, and M. Rinard. "An analysis of patch plausibility and correctness for generate-and-validate patch generation systems". In: *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. ACM. 2015, pages 24–36.

[18] J. Siegmund and J. Schumann. "Confounding parameters on program comprehension: a literature survey". In: *Empirical Software Engineering* 20.4 (2015), pages 1159–1192.

[19] E. K. Smith, E. T. Barr, C. Le Goues, and Y. Brun. "Is the cure worse than the disease? overfitting in automated program repair". In: *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*. ACM. 2015, pages 532–543.

[20] G. Tassey. "The economic impacts of inadequate infrastructure for software testing". In: *National Institute of Standards and Technology, RTI Project* 7007.011 (2002), pages 429–489.

[21] X. Xia, L. Bao, D. Lo, and S. Li. ""Automated Debugging Considered Harmful" Considered Harmful: A User Study Revisiting the Usefulness of Spectra-Based Fault Localization Techniques with Professionals Using Real Bugs from Large Systems". In: *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE. 2016, pages 267–278.

[22] A. Zeller. *Why programs fail: a guide to systematic debugging*. Elsevier, 2009.

Discovering Data Models from Event Logs

Dorina Bano

Business Process Technology Group
Hasso Plattner Institute for Digital Engineering
dorina.bano@hpi.de

I have joined the SSE (Service-Oriented System Engineering) research school last October in 2019. Hence, this is my first Fall report, where I am shortly describing my research during the first year. The research area that I am currently focused on is *Process Mining*, a sub-discipline of *Business Process Management*. Process mining is important for understanding the behaviour aspects of any given organization. In a process mining project, process experts are tasked with discovering or improving the operational business processes. They do so by analyzing event logs, the starting point of any process mining endeavor.

My research is focused on understanding the implicit contextual information during process discovery. To this end, I have published a paper in *International Conference on Conceptual Modeling* (*ER 2020*) called *Discovering Data Models from Event Logs*. In addition, I have submitted a report in the *International Business Process Intelligence Challenge* which is still under review. In this report I am mostly focused on these research contributions.

1 Motivation

Some examples of process mining techniques include: process discovery, which aims at discover a process model from the recorded executions of a process; conformance checking, which intends to compare event data with a given process model in order to find deviations; and process improvement, where a business process model is enriched with additional details about its performance [3].

The starting point of any process mining technique is an event log, which is purely a collection of events [2]. In many real-world scenarios such event logs are extracted from data warehouses of a given organizations [4]. After the extraction process takes place the event logs are made available to business process mining experts.

Since the log is tailored to discovering and improving a business process the data perspective is usually overlooked. Therefore the process mining experts are left with an event log that does not provide explicit information about the context the data it was extracted from. We argue that the data perspective is an important aspects that complements the process mining procedure with useful information. It plays an important role in the understandability of the event logs and consecutively the process model.

We have introduced a semi-automatic two-step approach for discovering a complementary UML data model from an event log which is tailored for process mining.

The discovered data model provides additional insights regarding the domain specific information in the log. In addition, it can be used to enrich the mined process with data objects, therefore, improving its readability.

2 Approach

Deriving a data model from the event log implies systematically deriving the individual UML language [8] (our language of choice) constructs: classes, class attributes, and the associations between classes. To this end, we will follow a two-step approach as depicted in Figure 1. In the first automatic step we introduce a intermediate representation that captures the access relation between all activities and all attributes from the event log. This representation is called Activity-Attribute relationship (A2A) diagram, which is inspired from [1]. In the second step, a set of generic rules are applied to the A2A diagram resulting in the target data model. Afterwards, we leave the choice to the user of the approach to review the generated data model.

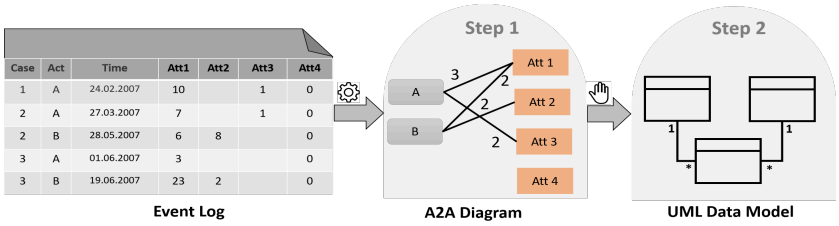

Figure 1: Overview of the data model discovering approach

2.1 Derivation of the A2A Diagram

The starting step for constructing such a diagram is to identify the activities and attributes from the event log. Therefore, we first derive the set of all activities (at the model level). Second, all attributes, except the meta-attributes (like case identifier, activity name and timestamp), are identified.

The next step for construction of the A2A diagram is to identify the access relation between activities and attributes. If an activity *A* writes a value to an attribute *Att1*, then there is an access relation between activity *A* and attribute *Att1* (see Figure 1). The access occurrence number (depicted over the access arrow in Figure 1, Step 2) represent the number of times an access relation between an activity and attribute holds in the event log independently of the case.

2.2 Data Model Discovering

The data model generation consist in generating the data model classes with their attributes and the associations between the classes. For the data model classes generation we look at the relations between two or more attributes in the A2A diagram and consider whether they belong to the same data model class. After exhaustively going through all the attributes and grouping them into UML classes, we identify the UML associations between those classes. Defining the UML associations entails specifying their multiplicity.

A set of rules (see Figure 2) are constructed and applied to the A2A diagram for grouping the attributes into data model classes. These rules are organized based on two aspects: not/isolated attributes and not/isolated activities. An attribute is called isolated if all the activity that access it do not access other attributes. Likewise, an activity is isolated if all the attributes it accesses are not accessed by any other activity.

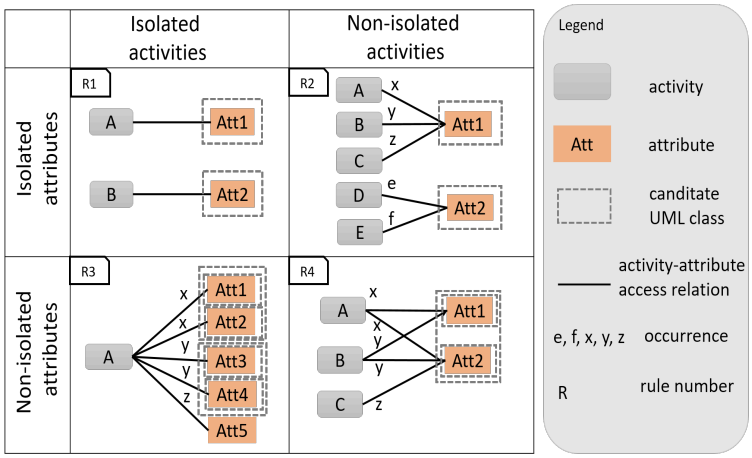

Figure 2: Rules for deriving data model classes

- **isolated attributes, isolated activities** – we identify the isolated access relations in the A2A diagram, in that an attribute is accessed only by a single activity and the activity accesses only the said attribute. In this case, the rule is to assign all isolated attributes to separate independent UML classes. At this stage, there is no other information in the A2A diagram that can give insights about how the generated UML classes could be related.

- **isolated attributes, non-isolated activity** – we search the A2A diagram for isolated attributes that are accessed by non-isolated activities. Similar to rule R1, the attributes are isolated and, thus, there is no additional information on how they can be grouped into classes. Hence, the isolated attributes will be each assigned to a separated class.

- **non-isolated attributes, isolated activities** – if at least one common activity accesses two or more attributes, then the attributes are said to be related. We are looking specifically for related attributes that may belong to the same class. We argue that if an activity accesses two or more attributes with the same occurrence then these attributes are highly likely to be contained in a single class. Therefore, we group these attributes based on common occurrences. However, we cannot deduce from the A2A diagram alone whether the attributes are accessed simultaneously by the activity. It may happen that in total these attributes are accessed the same amount of time by the activity but never in the same event. This means that the attributes are highly likely to not belong to the same class as they seem to be accessed independently. To counter this problem we offer the following solution.

  Let E_{A1} be a set of events from the event log where activity A accesses attribute 1. $\mid E_{A1} \mid$ denotes the access occurrence. Similarly, we define E_{A2} as the set of all events where the activity A accesses attribute 2 with occurrence $\mid E_{A2} \mid$, where $\mid E_{A2} \mid \geq \mid E_{A1} \mid$. The decision of whether attribute 1 and 2 belong to the same class is made based on the following function:

$$rel(E_{A1}, E_{A2}) = \begin{cases} \text{one class} & \text{, if } E_{A1} \cap E_{A2} = E_{A1} \\ \text{independent classes} & \text{, if } E_{A1} \cap E_{A2} = \emptyset \\ \text{dependent classes} & \text{, if } 0 < \mid E_{A1} \cap E_{A2} \mid < \mid E_{A1} \mid \end{cases} \quad (1)$$

  Attribute 1 and 2 belong to the same class if set E_{A1} is a subset of E_{A2} because anytime the activity A accesses attribute 1 it also accesses attribute 2. Both attributes define a new UML class, however, attribute 1 is marked as optional because it is not always accessed when attribute 2 is accessed.

  If the two sets are disjoint (i.e., $E_{A1} \cap E_{A2} = \emptyset$), then attribute 1 and 2 are not in the same class and, moreover, these classes have no association between them. This is due to attribute 1 and 2 happening independently of each other.

  Finally, there are events in which attribute 1 and attribute 2 are accessed simultaneously except the first case. This means that there are some events where the attribute 1 and 2 are accessed by the same activity A but this number of events is not the same as $\mid E_{A1} \mid$. In this case, the attributes are placed in different classes, but the classes are still related via an bidirectional association. The multiplicity of the association is 0..1 to * from the class containing attribute 1 to the class containing attribute 2.

  In a more general case, where the number of attributes which share the same activity with the same occurrence is more than two, we apply the above function for every pair of attributes to determine the resulting classes.

- **non-isolated attributes, non-isolated activities** – every relation that cannot be expressed by the previous rules is captured by this rule. Activities and attributes are non-isolated, which mean that an attribute is accessed by several activities and each activity accesses several attributes.

After removing the attributes and activities that satisfied the previous three rules we are left with an A2A diagram that contains one or more disconnected subgraphs (i.e., interconnected activities and attributes) which we are referring to as islands. In Figure 2 R4, there is only one island, but it can happen that another set of non-isolated activities and attributes, which has no relation with the first set, can be left in the A2A diagram. That is why we call these sets islands.

To group the attributes into UML classes each island is decomposed into smaller A2A diagram fragments for each activity. This means that the number of the fragments is the same with the number of activities in an island. The attributes that are accessed by the activity are represented in the respective fragment. Hence, an attribute may appear in one or more fragments (see Figure 3).

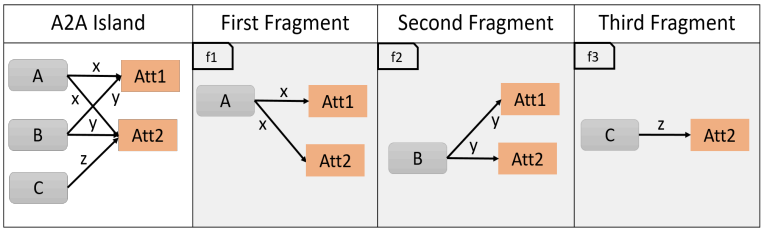

Figure 3: Decomposed A2A diagram into three fragments after applying the fourth rule

The resulting fragments can satisfy either rule 1 or rule 3 but not rule 2 because the fragments contain only isolated activities. The grouping of the attributes, then, follows the rule 1 or 3. However, since attributes may belong to two or more distinct fragments there is a conflicts that needs to be resolved. For example, it might happen that the same attribute is grouped either in a standalone UML class or in a class with some other attributes depending on the grouping results from each fragment. In this case, we leave the choice to the user of the approach to make a decision that better fits the overall result.

3 Evaluation

The approach presented in this report is evaluated based on two real-life event logs, namely: Road Traffic Fine Management (RTFM) [7]; and Sepsis event log [6].

However, for sake of writing space, we describe the evaluation of our approach based on RTFM event log, which is taken from the information systems of the Italian police. The event log contains information regarding the road-traffic fines and includes 150.370 cases (561.470 events) that are processed by the municipality over a three-years time period (January 2010 – June 2013). To provide a behavioral overview

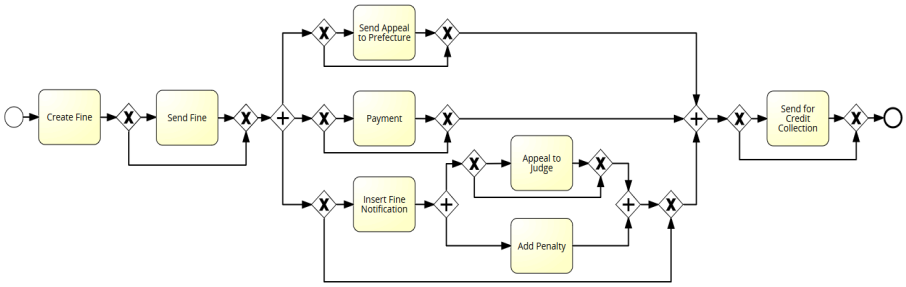

Figure 4: BPMN model discovered from the RTFM event log

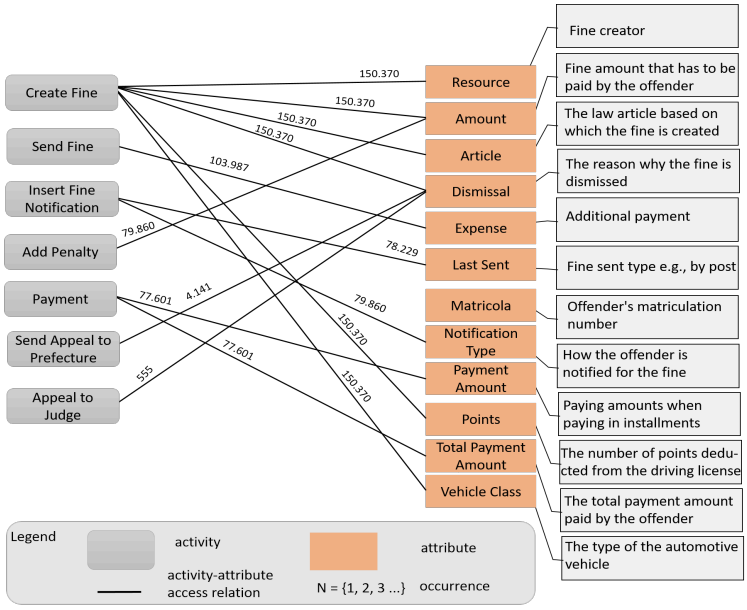

Figure 5: The RTFM A2A diagram

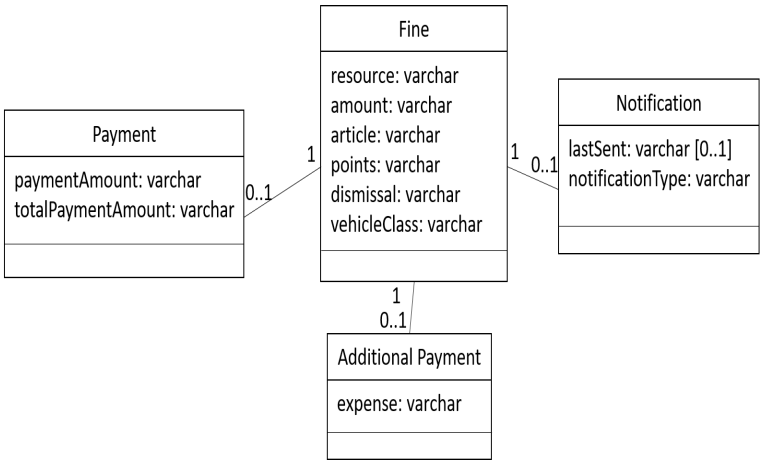

Figure 6: The discovered Data Model from RTFM event log

of the event log, we show in Figure 4 the process model (represented as BPMN [9]) that is discovered by applying the Inductive Miner algorithm [5]. Some activities that do not access any attribute are excluded from the process model without breaking its meaning.

The process starts with *Create Fine* activity. After the fine is created it can be send to the offender via *Send Fine*. The offender has the option to pay the fine immediately after it is handed over to him (*Payment*). If this is not the case, the date when the offender receives the fine is registered (*Insert Fine Notification*). If the payment will not take place (i.e., within 60 days) then a penalty (*Add Penalty*) is added to the fine. The offender has the option to appeal against the fine through the Judge (*Appeal to Judge*) or Prefecture (*Send Appeal to Prefecture*). If the appeal is successful then the process ends. Otherwise the fine is sent for credit collection (*Send for Credit Collection*) marking the process terminations.

Following the rules presented above the A2A diagram (Figure 5) and the discovered Data Model (Figure 6) are illustrated below.

4 Future Work

Currently I am working on discovering the business process architecture from several event logs which are extracted from the same domain. The main idea is to discover a set of patterns which can explain how several business process are related together.

Another work in progress is related with the event log extraction from the redo logs. The redo logs are part of the database recovery mechanism and provides the information regarding the transactions made upon. We are aiming to extract an event log by just considering as input the Redo logs. The motivation behind this work is to discover the process from the Redo logs without having access to the database schema or a deep understanding of it. In addition, we are aiming to discover the schema from the Redo logs as well.

Furthermore, I am planning to extend the approach presented in this report to support more than one event log from the same organization and to derive a common data model that spans many discoverable business processes.

References

[1] W. M. P. van der Aalst. *Object-Centric Process Mining: Dealing With Divergence and Convergence in Event Data*. EasyChair Preprint no. 2301. EasyChair, 2020.

[2] W. M. P. van der Aalst. "Process Mining in the Large: A Tutorial". In: *Business Intelligence - Third European Summer School, eBISS 2013, Dagstuhl Castle, Germany, July 7-12, 2013, Tutorial Lectures*. Volume 172. Springer, 2013, pages 33–76. DOI: 10.1007/978-3-319-05461-2_2.

[3] W. M. P. van der Aalst. *Process Mining: Data Science in Action*. 2nd edition. Heidelberg: Springer, 2016. ISBN: 978-3-662-49850-7. DOI: 10.1007/978-3-662-49851-4.

[4] K. Diba, K. Batoulis, M. Weidlich, and M. Weske. "Extraction, correlation, and abstraction of event data for process mining". In: *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* 10.3 (2020). DOI: 10.1002/widm.1346.

[5] S. J. J. Leemans, D. Fahland, and W. M. P. van der Aalst. "Process and Deviation Exploration with Inductive Visual Miner". In: *Proceedings of the BPM Demo Sessions 2014 Co-located with the 12th International Conference on Business Process Management (BPM 2014)*. CEUR-WS.org, 2014.

[6] F. Mannhardt. "Sepsis cases-event log". In: *Eindhoven University of Technology, Eindhoven* (2016). https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

[7] F. Mannhardt and M. de Leoni. "Road traffic fine management process". In: *Eindhoven University of Technology, Eindhoven* (2015). https://doi.org/10.4121/uuid:270fd440-1057-4fb9-89a9-b699b47990f5.

[8] J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2nd Edition)*. Pearson Higher Education, 2004. ISBN: 978-0-321-24562-5.

[9] M. Weske. *Business Process Management - Concepts, Languages, Architectures, Third Edition*. Springer, 2019. ISBN: 978-3-662-59431-5. DOI: 10.1007/978-3-662-59432-2.

Towards Joint Design-Time and Run-time Verification of Complex Systems

He Xu

System Analysis and Modeling Group
Hasso Plattner Institute for Digital Engineering
he.xu@hpi.de

This report describes a new approach that combines the backward design time and forward run-time checking techniques. The backward checking is used to find out all counterexample paths that will lead to an unsafe state and establish an unsafe area with a radius of k. At run-time, the forward model checking will periodically perform n steps checking from the current state. Associated with the verification result from the development phase, systems can perceive potential hazards $k + n - 1$ steps before their happening. This approach moves the most computationally intensive task to the design time and obtains enough confidence and reactive time using constrained time and resources. Our work provides a complete verification tool that can help improve the system, generate the unsafe areas towards unsafe states, and cope with the failure at run-time.

1 Motivation

When we concern ourselves with designing a complex system, especially those who interact with the environment and other systems a lot, e.g. a self-driving car system, the most crucial problem is how to guarantee assurances during operation. Run-time internal failures and external hazards may lead to severe consequences and some of them may lead to fatal accidents.

Indeed, for some systems, we can prove their correctness at design time, but for complicated systems, that would be extremely expensive, and sometimes it is impractical. For instance, acquiring an accurate model at design time for some systems is infeasible[4]. Specific to the self-driving car system, we cannot anticipate all uncertainties in the environment and system at design time, and the system may also change its behavior (e.g., update its model) at run-time, which means in most cases there is no way to design a absolute safe system.

On the other hand, run-time verification can provide systems assurance when they face changes from the environment or system themselves. However, due to the limitation of resources and time, run-time approaches cannot provide an accurate and comprehensive guarantee to the system. Hence, it is necessary to explore a seamless verification method working in both the design phase and run-time phase.

In this approach, we develop a joint verification method, which combines advantages of both design time and run-time verification. That is at design time the verification process will do as much as possible to ensure the safety, reliability, and

robustness of the system. These works will cover the most resource and time consumption parts of the verification of the system. At the same time, the design time verification will also prepare the data, strategies, etc. for the run-time verification. At run-time, the system will monitor the system itself and the context. When potential risks have been detected or the prediction of hazards occurs, the system will invoke the coping mechanism.

2 Approaches

2.1 Introduction

In this approach, I will use two checking methods correspond to run-time and design time verification. The backward checking will be used at design time to establish the unsafe areas around the inevitable unsafe states and to help engineers pick the proper countermeasures for each of these failures.

At run-time, the forward checking process only searches to states that are reachable from the current state within a fixed number of transitions. Once it detects the boundary of the unsafe area, it can execute the emergent mechanism to cope with the adverse situation. Combining with results from design time backward checking, the run-time forward checking can hence give the system enough time to prepare for potential failures or hazards. At the same time, it will reduce the time and resource consumption of time and resource at run-time.

2.2 Use Case

Before presenting details of the work, We will first introduce the use case.

During the operation, the self-driving car will face a variety of uncertainties. These uncertainties may come from the environment, other participants on the road, or from the system itself and some of these uncertainties may trigger foreseeable or unforeseeable problems.

Potential hazards at run-time:

System failure

- Power system failures.
- Electronic system failures.
- Control system failures.

Context disturbance

- Dangerous behaviors of other participants.
- Bad weather (resulting in some sensor degradation).

Besides, there also exist plenty of hazards inside or outside the system, which are difficult to anticipate or identified before deployment. How to deal with these problems is another topic.

For those foreseeable failures, some of them can be excluded before deployment by good design, verification, and validation. However, during the design time, for the other failures, engineers can only either reduce its harm or design some emergent mechanism to prevent its happening. For instance, engineers can design a perfect control system for the self-driving car that can avoid any collision when other participants behave properly (e.g., every vehicle adopts the same control strategy and all in good condition)[5]. However, in most instances, it is impossible. Different cars have different mechanical capacity, that means they have different control strategies. Even for the same type of cars, after learning the driver's habit, they also perform differently.

In this work, we establish a simple traffic system to help describe the approach. As depicted in Figure 1, in this system, there are three cars, one traffic lane and one emergency lane. It is easy to see that, for the target car (the central blue car), it is impossible to ensure absolute safety when the front and rear participants are too close.

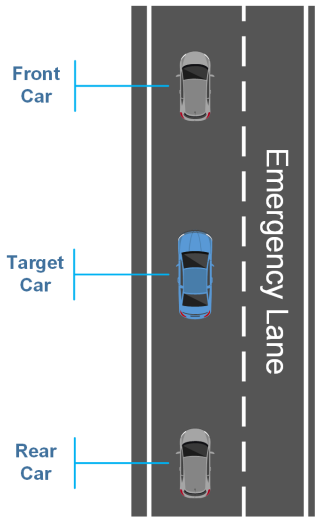

Figure 1: In this scenario, the *target car* can not ensure absolute safety without using the emergency lane

Based on the scenario discussed above, we can establish a simple traffic system model. It has three templates:

- Car template
  The car template is a timed automaton that describes the target car that is instantiated as "*Target*" in the system. As depicted in figure 2, it has three states, which are *Normal*, *Approaching*, and *Danger*. In the system, normally, the target car is in a "*Normal*" state that means the self-driving system can avoid potential

accidents by regular driving actions(do not need to steer to emergency lane). When neither the front car nor the rear car(instances of participant template, see figure 3) is at *"Far"* state and not both of them are at "*Close*" state, then the state of target car moves from "*Normal*" through *Alert*/*Acc_Needed*/*Brk_Needed* (which are committed states), and then to "*Approaching*". When both the front and rear cars are in the "*Close*" state, the state will move from "*Approach*" to "*Danger*".

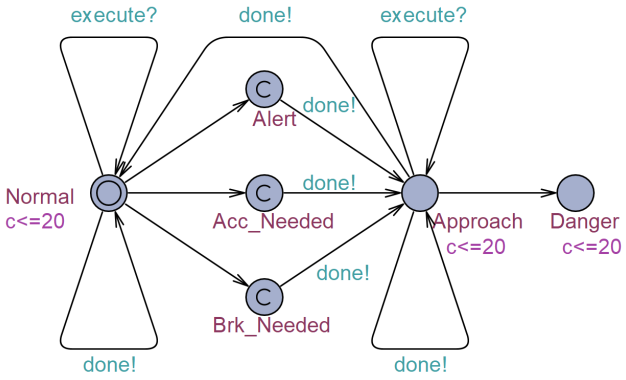

Figure 2: Template: Car

- Participant template
  The participant template is instantiated twice in the system(*Car_f* and *Car_r*), which describes the front car and rear car in the traffic system, respectively. As depicted in figure 3, it has three states, which are *Far*, *Near*, and *Close*. Every transition will vary the distance between the front/rear car and the target car. When specific distance values are reached(e.g., the threshold between *Close* and *Near* is 100), the transition between states will take place.

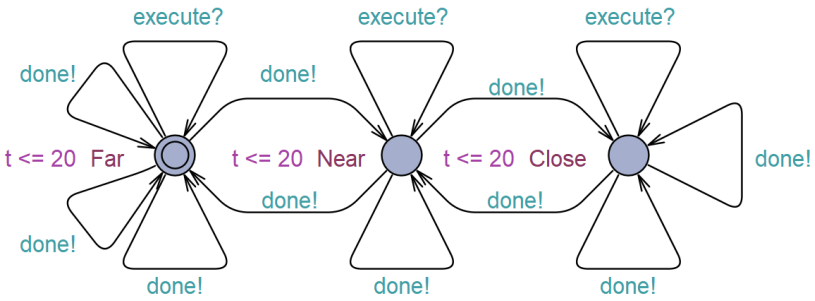

Figure 3: Template: Participant

- Order template
  The order template arranges the execution order of all objects in the system(Figure 4).

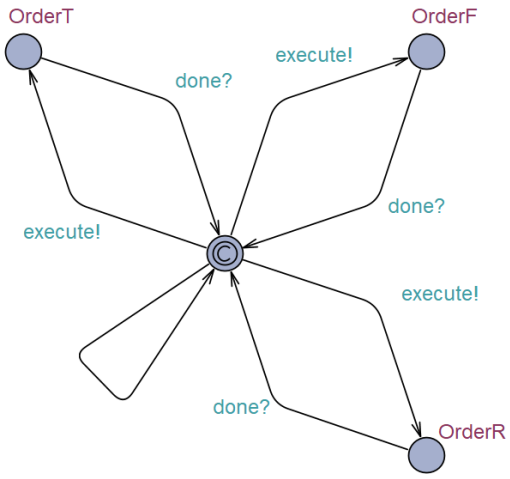

Figure 4: Template: Order

2.3 Design Time Backward Checking

In this section, the backward checking method and some associated concepts are laid out. The backward checking is used to find out all counterexample paths that will lead to the unsafe state. It brings two advantages. First, the backward checking process may find out some violations (unexpected failures) in a short time, especially when they are hidden "deeply". These counterexamples will help the improvement of system design. Second, as the initial state in the reversed model is the unsafe state, the backward checking process can traverse all paths in k steps (or in a acceptable longer $k + m(m > 0)$ steps, in order to find out states fulfill the definition 5.3) that will lead to this unsafe state and choose and store all states that fulfill certain requirements (Figure 5).

The checking step k is derived from the requirements of the system emergency behaviors. For instance, the emergency brake or steering needs some time to observe and estimate the traffic situation to make sure the subsequent actions will not result in another accident.

On the other hand, if states that are detected on k steps also found on $k - r$ $(0 < r \leqslant k)$ steps, then the system's design might be imperfect.

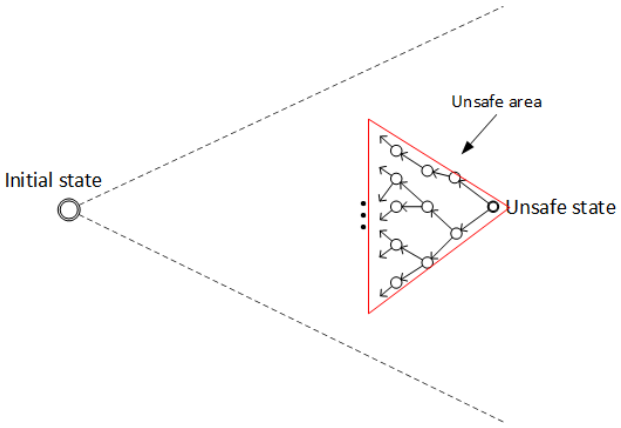

Figure 5: Design Time Backward Checking and Established Unsafe Area

2.3.1 Backward Model Checking

Before conducting the backward model checking, first, it is necessary to identify states that need to be concerned about. State suitable for this approach has following features:

- For the target system, it is unsafe.
- Under certain circumstance, it is inevitable.
- It can be identified and well defined during development.

For a specific property ϕ, unsafe states of a system can be defined as follows.

Definition 5.1. Unsafe State
Let ϕ be the property that capture a specific hazard, and let S be the set of all states of a system model. S_ϕ is the subset of S that all states in S_ϕ fulfill property ϕ. *Unsafe state* is the state $s_n \in S_\phi$.

In the system we mentioned above, the unsafe state is $Target.Danger$. Based on the scenario, there is only one traffic lane, when the front car and rear car are both too close to the target car, then it cannot guarantee the absolute safety under regular manipulation.

Once an unsafe state has been identified, a reverse model of the original automaton can be built. In the reverse model, the initial state is an unsafe state $s_n \in S_\phi$, and the backward checking procedure then searches for all path fragments in k steps.

In our case study, all templates in the system are reversed (Figure 6 and 7). In the instantiation, the initial state of target car becomes $Target.Danger$, and accordingly, the initial states of the front and rear car are $Car\_f.Close$ and $Car\_r.Close$.

The backward model checking is defined as follows.

Definition 5.2. Backward Model Checking(BMC)
Let S_ϕ be the set of unsafe states. M is a state-transition system and $\overline{M}$ is its reverse model. *Backward model checking* is to find all k steps initial path fragment π_k in $\overline{M}$. Here, the initial state s_0 is the state $s \in S_\phi$.

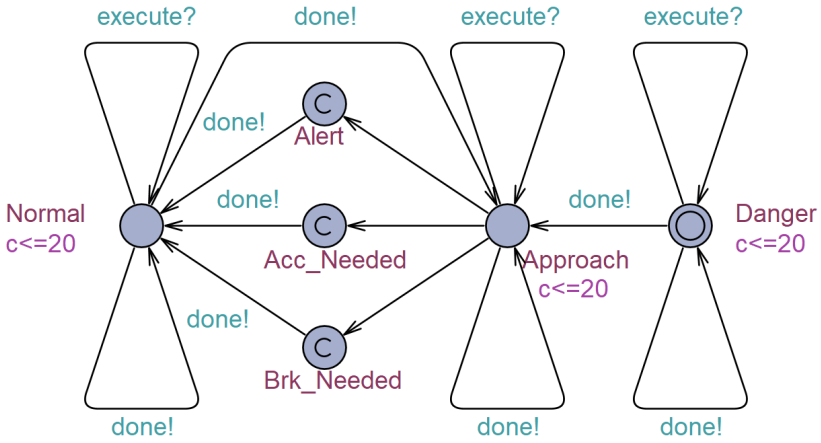

Figure 6: Reversed Car Template

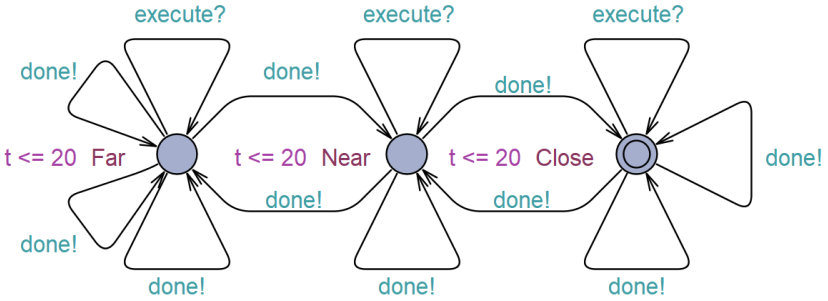

Figure 7: Reversed Participant Template

The backward checking process on the reverse model generates a series of path fragments, by which can define the unsafe boundary of a specific unsafe state. The unsafe boundary is a subset of all system states, and all elements in it can reach the unsafe state in k steps. The unsafe area can also be established, which is the union of all unsafe boundaries less than k and the unsafe state itself.

Definition 5.3. Unsafe Boundary
The *unsafe boundary*$(BMC(S_\phi, k))$ of a system model is defined as follows:

$$BMC(S_\phi, k) = \{s \in S \mid s' \in S_\phi : (s' \rightarrow_k s) \wedge (s' \nrightarrow_r s)\} \qquad 0 < r < k$$

in which, $S_\phi \in S$ is the set of all unsafe states.

Definition 5.4. Unsafe Area
$BMC(S_\phi, k)$ is the *unsafe boundary*. Unsafe area of k steps$(U(k))$ can be defined as follows:

$$U(k) = \bigcup_{n=1}^{k} BMC(S_\phi, n) \cup S_\phi.$$

2.4 Run-time Forward Checking

The classic model checking technique explores the whole state space in a brute-force manner [1]. At design time and for some specific system models, it works. Although it will consume considerable time and computational resources, it is still acceptable. However, for run-time use, it is another story. In some scenarios, for example, the system updates its model at run-time, we cannot "frozen" the system and wait for the generation of the new state space and the termination of exhaustive verification.

There are several different approaches to conduct the run-time checking. First, in [3], a run-time statistical model checking component has been integrated into a self-adaptive system. This method enables the system to verify properties at run-time and adjust its performance by the tradeoff of confidence it provides and the time and resources it requires. Second, [6] offers another perspective, that is we can migrate the computational intensive task, i.e., model checking, to the cloud, and offer the verification as a service. Both approaches work well in their own scenarios. However, the first method may be too heavy to the self-driving system, and the second method may suffer from network latency or uncertain QoS.

In this work, we restrict the search to states that are reachable from the current state within fixed number n of transitions(Figure 8). The number n is chosen at design time based on the requirement of emergency actions, or modified at run-time based on the tradeoff of accuracy and resources.

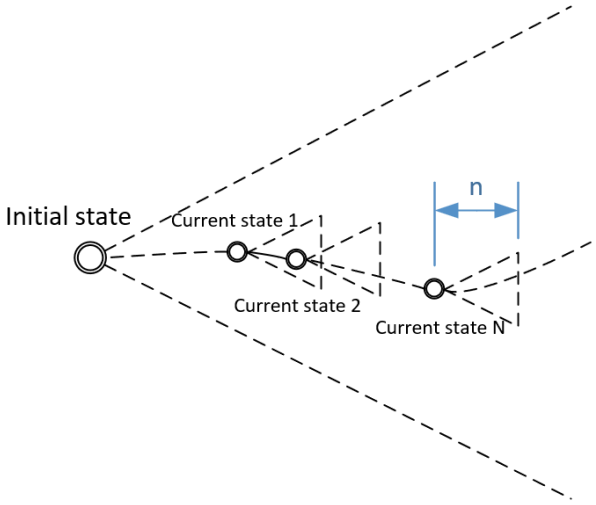

Figure 8: Run-time Forward Checking.

2.4.1 Bounded Forward Checking

At run-time, the model checker will periodically perform the checking process from the current state (Figure 9). Once it detects states $s \in BMC(S_\phi, k)$ that lead to an specific unsafe state, then it can invoke emergent actions according to this situation(e.g., emergent brake or steer to the emergency lane). Combining with results from design time backward model checking, the run-time forward model checking can give the system the ability that let the system predict the potential failure in the near future and enough time to react (depends on the checking bound). We define the run-time

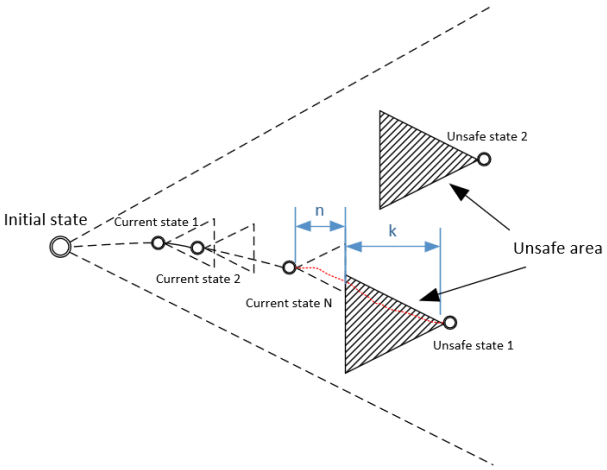

Figure 9: Combining Forward and Backward Checking

forward model checking as follow.

Definition 5.5. Forward Model Checking(*FMC*)
Let s_c be the current state and also a safe state of a system, M be the system model, and $FMC(s_c, n)$ is the subset of S that all states in it can be reached in n steps (not less than n steps).

$$FMC(s_c, n) = \{s \in S \mid (s_c \rightarrow_n s) \wedge (s_c \nrightarrow_r s)\} \qquad 0 < r < n$$

Forward model checking is to find out all $s \in FMC(s_c, n)$ such that

$$FMC(s_c, n) \cap BMC(S_\phi, k) \neq \emptyset.$$

The backward checking on the reverse model generates unsafe areas with a radius of k. During operation, the system search from the current state for n steps periodically (i.e., from every current state). This assures that states $s \in FMC(s_c, n)$ can only be reached at n steps. Therefore, during operation, the system can be alerted to the unsafe state in $n + k - 1$ steps.

Theorem 5.1. Combination of BMC and FMC
Combining the result of k steps design time backward model checking and n steps run-time model checking can acquire checking result of $n + k - 1$ steps at run-time.

$$FMC(s_c, n) \cap BMC(S_\phi, k) \neq \emptyset \iff FMC(s_c, n + k - 1) \cap S_\phi \neq \emptyset.$$

Remark. $FMC(s_c, n + k - 1) \cap S_\phi \neq \emptyset$ equals that the hazard can be reached from current (and safe) state in $n + k - 1$ steps.

In the case study, there is an additional template(*emergency*) that executes the emergency maneuver when *Target Car* detects the anticipated unsafe state. It needs some time to aware of the situation, make the decision, and perform the emergency behaviors (Figure 10).

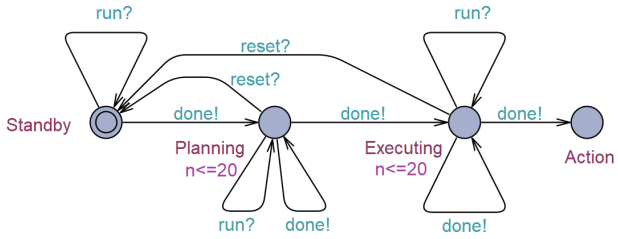

Figure 10: Template: Emergency

During operation, once the *Target Car* detects the boundary of unsafe area, it will invoke the emergency process to cope with the potential hazard. This process will consume T time unit to reach the final state "*Action*". In the case study, $n \times t \leqslant T < (n + k - 1) \times t$, where t is the maximum number of time units that needed by each transition.

3 Evaluation

As discussed above, we use a simplified traffic system to elaborate our approach. In this work, we use UPPAAL to establish models of this traffic system. UPPAAL is a toolbox for modeling, verification (via automatic model-checking) and validation (via graphical simulation) of systems modeled as networks of timed automata. it is developed by Uppsala University and Aalborg University, and the first version was released in 1995 [2].

In the simulation[1], first, we use the regular model checking method, and set different initial values (distance from the target car, corresponds to a certain number of

[1]Configuration: 64-bit Operating system (Windows 10), Intel Core i7-6700HQ, 40GB RAM, UPPAAL 4.1.19(32-bit).

checking steps) to the participants, then evaluate the time and memory consumption towards a specific property(generating the shortest path that leads to the unsafe state "*Target.Danger*"). After that, we use the reversed model to conduct the backward checking, and count the time and memory consumption under different checking steps. In the end, we simulate the run-time forward checking process.

In this traffic system, the *Target Car* can only cruise or execute emergency action. For *Participants* (front car and rear car), in every transition, they have three options that are shortening, keeping, or extending the distance with the target car.

Our results of checking the whole system is shown in Table. 1.

Table 1: Time and Memory Usage with Different Initial Distance.

Initial Distance	Steps [2]	E<> Target.Danger	
		Time(s) [3]	Memory(KB) [4]
250	910	8.672	251,852
280	1090	12.297	351,324
310	1270	17.063	482,540
340	1450	21.875	611,320
370	1630	27.172	771,556
400	1810	34.547	963,612

According to the results in Table 1, it is apparent that it is hard to check the whole system during operation. First, it cost to much time to check the whole system. Second, this case study only focuses on a simple scenario, if there are more participant or more complex traffic condition, the usage of time and memory will increase considerably.

In the following experiment, the performance of backward checking is evaluated. Due to the limitation of the model checker, in the system model, we test different checking bounds(k) by setting different thresholds of *Near Distance*. This parameter closely relates to the checking steps that are needed to find out the *unsafe boundary*. In this experiment, the unsafe boundary is:

$$BMC(S_\phi, k) = \{Target.Alert, Target.Acc\_Needed, Target.Brk\_Needed\}$$

Table 2 is the time and memory usage of the backward checking.

From the table above, we can see, the backward checking process also costs an amount of time and resources. However, since it is executed at design time, these costs to establish the unsafe area is acceptable.

[2] **Search Order:** Breadth First; **Diagnostic Trace:** Shortest.

[3] Average elapsed time of ten experiments.

[4] Average memory usage peaks of ten experiments.

Table 2: Time and Memory Usage of Backward Checking.

Threshold of Near Distance	Steps(k)	Time(s)	Memory(KB)
200	1209	15.62	444,348
230	1389	20.597	567,744
260	1569	25.591	706,816
290	1749	32.724	909,284
320	1929	38.642	1,077,080
350	2109	46.427	1,277,124

In the following experiments, different forward checking steps (n) are evaluated. In practical application, the parameter n should be chosen according to the demand of safety requirements, and here we only test performances of several certain numbers.

The evaluation of run-time forward checking is shown in Table 3.

Table 3: Time and Memory Usage of Run-time Forward Checking.

Steps(n)	Time(s)	Memory(KB)
39	0.038	26,968
69	0.105	27,600
99	0.205	28,740
129	0.339	30,456
159	0.502	32,812
189	0.721	35,044

From the result of the forward checking simulation, we can see that combining with the result from backward checking, system acquires the same confidence and the reaction time in a much shorter checking time and cost fewer resources, comparing with regular run-time forward checking.

4 Discussion and Conclusion

This report describes a new approach that combines the backward design time and forward run-time checking techniques. It provides a combined verification method that can help improve the system, generate unsafe areas, and cope with the failure at run-time.

This method can be used in many systems. For example, for some systems, they are either too complex to obtain a model with all details, or some crucial parameters can only acquire from other systems or environments at run-time. Besides, some systems will also change their configuration or behaviors when adapting to some disturbance from the environment or system itself (e.g., self-adaptive system).

In the next step of my research, I want to enhance the design time checking method, that is, not only generate the unsafe paths (counterexamples) but also generate solutions (e.g., paths can avoid the unsafe transitions) automatically based on these counterexamples. At run-time, the system can use these solutions directly when a potential hazard is detected.

References

[1] C. Baier and J.-P. Katoen. *Principles of model checking*. MIT press, 2008.

[2] G. Behrmann, A. David, and K. G. Larsen. "A tutorial on Uppaal 4.0". In: *Department of computer science, Aalborg university* (2006).

[3] M. U. Iftikhar and D. Weyns. "Towards runtime statistical model checking for self-adaptive systems". In: *CW Reports* (2016).

[4] J. Rinast. "An online model-checking framework for timed automata". PhD thesis. Technische Universität Hamburg, 2015.

[5] S. Shalev-Shwartz, S. Shammah, and A. Shashua. "On a formal model of safe and scalable self-driving cars". In: *arXiv preprint arXiv:1708.06374* (2017).

[6] A. M. Sharifloo and A. Metzger. "Mcaas: Model checking in the cloud for assurances of adaptive systems". In: *Software Engineering for Self-Adaptive Systems III. Assurances*. Springer, 2017, pages 137–153.

Survey on Failure Propagation, Detection and Control in Microservices Architecture

Iqra Zafar

System Analysis and Modeling
Hasso Plattner Institute for Digital Engineering
Iqra.Zafar@hpi.de

This report describes my research activities in the System Analysis and Modeling group at HPI Research School during the last six month. As a preliminary research, we investigate different aspects of the microservice architecture,including its characteristics, challenges and state-of-the-art solutions. We also consider the concept of anomaly detection from a propagation and failure controlling perspective in microservice architecture (MSA).

1 Overview

Microservices (or microservices architecture) are cloud native architectural style that composes many loosely coupled and independently deployable smaller components, or services that are separated via bounded context in a single application. These services have their own database and communications with each other over a combination of REST APIs, event streaming, and message brokers. Microservice Architecture appears poised to replace SOA as the dominant industry architecture. Number of internet applications are using this approach due to its flexibility and clear logic. The stability of microservice is thus vitally important for these applications' quality of service. The performance quality of microservice is of vital importance to the Internet company, because a microservice failure can degrade the user experience and bring economic loss. Therefore, an efficient root cause localization of online failures, which enables rapid service recovery and loss mitigation, becomes increasingly more important for microservices. During manual diagnosis, experts target certain kinds of metrics for troubleshooting according to microservices characteristics. In such a case where performance degradation occurs due to anomalies, anomaly detection and localization can help in capturing and tracking the anomalous behavior that deviates from the normal behavior of Microservice functionalities.In production operation, service deficiencies create anomalies as high latency, low throughput and unacceptable usability which can propagate throughout the network and finally crash the overall system. It is challenging to find root cause due to complexity of microservices based system even for experienced Site Reliability Engineer (SRE) [7].

2 State-of-the-Art

As a preliminary study, we investigate different research papers i.e journal, conference and extract proposed approaches, open challenges with provided solution and future work in microservice architecture. Recently, several works have been proposed to understand how a failure is propagated across microservices and try to localize the root cause microservice that leads to this failure.

In research work by Baylov et. al.[1], reference architectural approach is introduced in order to tackle microservice system problems.It is based on Autonomic computing.This reference model allows services to search or register itself for self-adaptation mechanisms when they need to respond to external environment changes. Existing SOA reference models become base for this proposed solution and the idea that in order to handle complexity of system and achieving high-level of objectives, self-adaptive systems can manage themselves with minimal or no human intervention.

A tool(MicroART) based approach is proposed by Di Francesco et.al. [2]. Microservice architecture models are generated with the help of proposed solution that can be managed by software architects for multiple purposes.By proposing this solution, author aims to get the overall picture of MSA based system during runtime involvement of microservices. This tool supports modeling, documenting, analysis, support to development and deployment for designing microservice-based architectures.

In research work by Ma, Meng, et al.[3, 4], self-adaptive root cause diagnosis framework is proposed. Main goal of research work is to identify anomalous services based on multiple kinds of collected metrics from frontend and backend services. Focus of proposed solution is how to combine domain knowledge with their solution and demonstrate its effectiveness. Instead of specifying certain kind of metrics to services, author designed self-adaptive mechanism named MS-Rank to update weight of metrics and dynamically select them for diagnosis according to their history about root cause diagnose precision. Proposed technique creates separate graph for each kind of metric to reduce complexity instead of considering the inter-relationship between multiple metrics. MS-Rank use constructed impact graph based on latency metrics to run other algorithms because calling topology is difficult to obtain. AutoMAP [5] is the tool implementation of proposed framework with few modification.

Samir, et al. [7], proposed a Detection and Localization systems for Anomalies (DLA) that detects and locates the anomalous behavior of MSA based on response time observation by using Hierarchical Hidden Markov Models (HHMM). The proposed system is 3 fold: (1) Monitoring that collects different data such as CPU, memory, and network metrics; that is responsible for the performance of services, containers, nodes and Virtual Machine (2) Detection that detects anomalous behavior which is observed in response time of a component; (3) Anomaly injection which simulates anomalous performance data representing normal and abnormal conditions.

In research work by Wu, Li, et al. [8], presents MicroRCA, a system to locate root causes of performance issues in microservices. Without using any application instruments, real time root cause analysis is being done by MicroRCA by correlating application performance symptoms with corresponding system resource uti-

lization.Attributed graphs become the base for root cause localization that model anomaly propagation across services and machines.

In this paper by Qiu, Juan, et al.[6], author propose an approach for mining causality and diagnosing the root cause that uses knowledge graph technology and a causal search algorithm. He verified the proposed method on a classic cloud-native application and found that the method is effective.

2.1 Open Challenges and Existing Solutions

In this section, we summarized challenges that are being faced during root cause localization in microservices based systems. These are also discussed in literature and researchers proposed different solution to tackle them using different performance metrics. Table 1 summarize those challenges and proposed solution.

Table 1: List of Challenges, Approach and Solution for Root Cause Analysis in Microservices

Challenges	Solution	Approach	Metrics	Graph	Ref
Service deficiencies	Self-adaptive	MS-Rank	Resource consumption and latency	Completed partially directed acyclic graph (CPDAG) i-e Impact Graph	[3, 4]
Performance degrade	self-contained probabilistic model	DLA	CPU, Memory, and Network	(HHMM)	[7]
Locate root cause	Correlation method	Graph based	Response time, CPU,Memory	Attributed graph	[8]
Causality Mining	Knowledge graph technology	dependency graph-based method	Cache, Memory usage, Response Time, Latency time	Causality Graph- Directed acyclic graph (DAG)	[6]

3 Future Plans

We are planning to carried out a Systematic Literature Review (SLR) in the next few months in the area of anomaly detection,its propagation and control its negative effects in microservices architectures. We will find out how MDE contribute in the life cycle of cloud-agnostic microservice application development. As a result of this SLR, we can be able to extract our research problem, existing solution and current research gap in microservices performance analysis.

References

[1] K. Baylov and A. Dimov. "Reference architecture for self-adaptive microservice systems". In: *International Symposium on Intelligent and Distributed Computing*. Springer. 2017, pages 297–303.

[2] P. Di Francesco. "Architecting microservices". In: *IEEE International Conference on Software Architecture Workshops (ICSAW)*. IEEE. 2017, pages 224–229.

[3] M. Ma, W. Lin, D. Pan, and P. Wang. "MS-Rank: Multi-Metric and Self-Adaptive Root Cause Diagnosis for Microservice Applications". In: *2019 IEEE International Conference on Web Services (ICWS)*. IEEE. 2019, pages 60–67.

[4] M. Ma, W. Lin, D. Pan, and P. Wang. "Self-Adaptive Root Cause Diagnosis for Large-Scale Microservice Architecture". In: *IEEE Transactions on Services Computing* (2020).

[5] M. Ma, J. Xu, Y. Wang, P. Chen, Z. Zhang, and P. Wang. "AutoMAP: Diagnose Your Microservice-based Web Applications Automatically". In: *Proceedings of The Web Conference 2020*. 2020, pages 246–258.

[6] J. Qiu, Q. Du, K. Yin, S.-L. Zhang, and C. Qian. "A Causality Mining and Knowledge Graph Based Method of Root Cause Diagnosis for Performance Anomaly in Cloud Applications". In: *Applied Sciences* 10.6 (2020), page 2166.

[7] A. Samir and C. Pahl. "Dla: Detecting and localizing anomalies in containerized microservice architectures using markov models". In: *2019 7th International Conference on Future Internet of Things and Cloud (FiCloud)*. IEEE. 2019, pages 205–213.

[8] L. Wu, J. Tordsson, E. Elmroth, and O. Kao. "MicroRCA: Root Cause Localization of Performance Issues in Microservices". In: *NOMS 2020-2020 IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2020, pages 1–9.

Image-Based Classification of Transport Infrastructure in Mobile Mapping 3D Point Clouds

Johannes Wolf

Computer Graphics Systems Group
Hasso Plattner Institute for Digital Engineering
johannes.wolf@hpi.de

Capturing urban areas and transport infrastructure for automated analysis processes becomes ever more important. Laserscanning and photogrammetry are used for scanning the environment in highly detailed resolution. This report presents techniques for the semantic classification of 3D point clouds from mobile mapping scans based on rendering and analyzing images. Objects of interest are primarily road markings and railroad tracks. The approach renders 3D point cloud input data into images for which U-Net as an established image recognition convolutional neural network is used for the semantic classification. The results of the classification are projected back into the 3D point cloud. An automated extraction of vector data can be applied for detected objects to generate detailed maps. The automatically generated shape files created by the presented process can be further used in various GIS applications. The results of implemented out-of-core techniques show that the approach can efficiently be applied on large datasets.

1 Overview

3D point clouds are widely used for representing geospatial information. They have proven to be a valuable data source for analyses as they are easy to capture even for large areas and hold great detail of the scanned environment [34]. They are a discrete representation of the real world and can be used for any environment without the need for specific configuration [10]. 3D point clouds can be updated with low effort and are well-suited for an automated analysis. Technically, they are stored as an unordered collection of measurement points each featuring three-dimensional coordinates and additional attributes, e. g., intensity values when being measured via LiDAR [24]. The unordered and unstructured points of a 3D point cloud usually require a semantic classification for successive usage [19]. Semantic classification is the process of assigning each object an additional attribute that describes the type of the object, such as "Car", "Lamp post" or "Traffic sign – turn right". Once individual objects and their semantic classes have been identified, they can be used for, e. g., road cadastre creation or renewal [5], clearance area checks [18], and 3D modeling [32]. Typical semantic classes enable a basic distinction between ground, vegetation, and buildings, but many additional and very detailed classes might be required for individual use cases, such as cars, road markings, traffic signs, or curbstones

[20]. Figure 1 shows a 3D point cloud with classification results whereas all points belonging to the same class are highlighted with the same color.

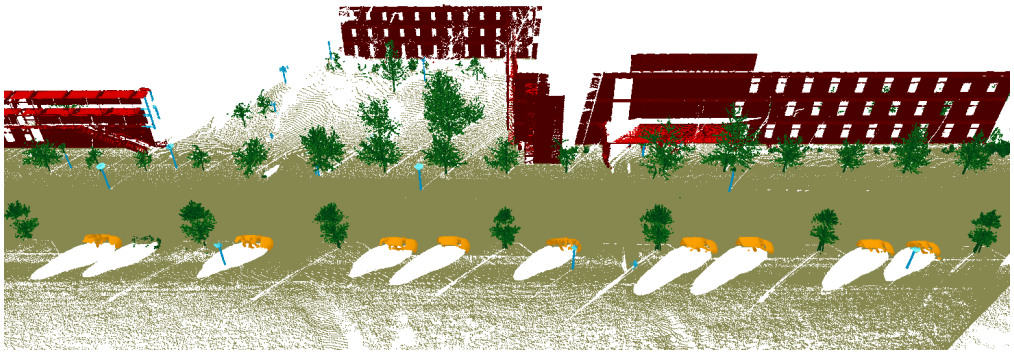

Figure 1: 3D point cloud colored based on semantic class: Ground (brown), vegetation (green), buildings (red), vehicles (orange), pole-like structures (blue)

Current information about the condition of traffic infrastructure is of great interest to municipalities and traffic offices. Road markings are essential for the regulation of traffic flow, especially in crossing situations. When planning construction sites and after the recreation of the previous state after road works, road construction offices require detailed information about the location of road markings. In everyday applications, navigation systems use a map with information about the number of lanes and which lane must be used to turn into a certain direction [3]. For autonomous driving, cars need to continuously detect road markings to keep the car in its lane and also use a base map with lane information for anticipating the course [17].

Similarly, railroad track courses must be permanently maintained, which is an expensive process [27]. It is important to detect, e.g., small changes in the track height caused by wear or erosion. More and more data is to be digitized in order to accelerate processing the information [2].

Thus, there is a need for efficiently capturing information about roads and railroad tracks in order to automatically create maps and statistic data. Mobile mapping scans are an established data source for the required information, 3D point clouds being measured via LiDAR include valuable intensity data for the localization of ground level objects and they can easily be captured in urban regions and on railroad tracks. The intensity value represents the strength of the reflection of the scanning laser, enabling conclusions about the structure of the surface or its material. As shown in Figures 2 and 3, road markings by design have a much higher intensity value than the surrounding pavement.

This report presents approaches for the semantic classification of road markings and railroad tracks using convolutional neural networks (CNNs) for visual recognition in images. Convolutional neural networks are a class of networks used in machine learning that were inspired by biological processes and find use especially in the automated analysis of image data [14]. In the past, many techniques for the

Figure 2: 3D point cloud of a crossing with road markings. Intensity values are represented in grayscale, lighter colors have higher intensity values.

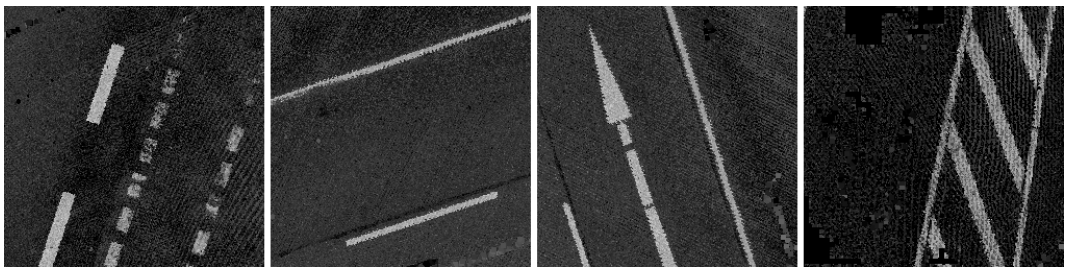

Figure 3: Examples of top-down rendered images from a 3D point cloud, showing different types of road markings. Intensity values are represented in grayscale, lighter colors have higher intensity values.

automated analysis of images have been made and popular frameworks have been developed [21]. We show that those can also be used for the classification of 3D point clouds in some use cases.

Approach The approaches use the abilities of image object detection algorithms to automatically classify objects in 3D point clouds. Road markings have already been analyzed in detail, railroad track detection will here only be described as a concept that is not yet completely developed.

Road Markings Road markings are clearly visible in a top-down view of the 3D point cloud data as shown in Figure 3. The implementation is based on a pipeline concept capable of automatically rendering large datasets, detecting markings in the created images, and mapping the results back into the original 3D point clouds.

Afterwards, shapes are created for individual markings. The resulting shape file can be used for further processing in GIS applications.

First, all input 3D point clouds are filtered in various preprocessing steps such as ground detection and outlier removal. Second, a renderer creates square images of these filtered 3D point clouds. Third, the rendered images are classified using the previously trained neural networks and the results are mapped back into the 3D point cloud. Finally, vector data is created for the individual markings.

Railroad Tracks In theory, the same approach could be used for railroad tracks. However, complex track switch configurations and the shadowing effects on parallel tracks lead to a low detection accuracy. Instead, upright cross sections in the direction of travel can be used.

Laser scanners mounted on trains usually scan in so called "scan profiles": The laser of the scanner rotates several times a second in a circle to accurately scan a cross-section of the surrounding area. If the scanner is moved continuously, the entire environment can be recorded. A scan profile, such as shown in Figure 4, corresponds to one of these cross sections. If the scanner is moved, a scan profile is not actually a closed circle, but a section of a Helix. When being cut in parts, each of these parts can be considered being roughly located within a disk. Thus, in railroad context, the rails on which the measuring train moves must be included in each of the scan profiles.

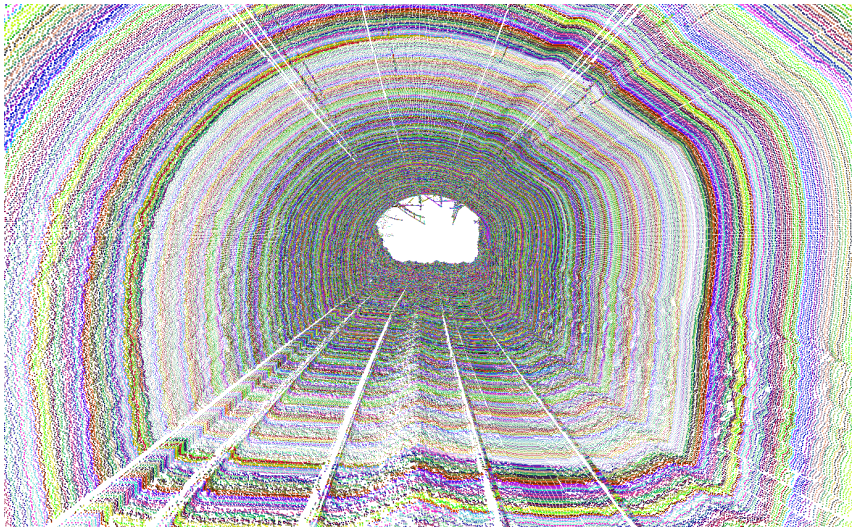

Figure 4: 3D point cloud of two parallel railroad tracks in a tunnel with visualized scan lines. All points on the same scan line are shown in the same color.

Each of the scan profiles can individually be rendered as an image which can then be used for railroad track detection. Either, a U-Net classification similar to the approach for road markings could be used, or the images are analyzed with

predefined characteristics for track height and the distance between the parallel rails.

Vector Data Generation Shape files are used to describe vector-based geospatial data. They can contain different types of shapes, such as points (positional data) and polygons (areal data). Each shape can have arbitrary attribute-value pairs, describing additional information available for this specific shape.

The presented approach for road markings creates polygonal shapes for all detected road markings, showing their precise location, size and orientation. Each shape gets an additional attribute describing the semantic class of the road marking in this position, which can then be used for, e. g., coloring all road marking types with different colors. Depending of the type of road marking, different approaches for the shape creation are used, as explained below.

The resulting files can be used in various GIS applications for subsequent tasks. Figure 5 shows the rendering of an automatically created shape file.

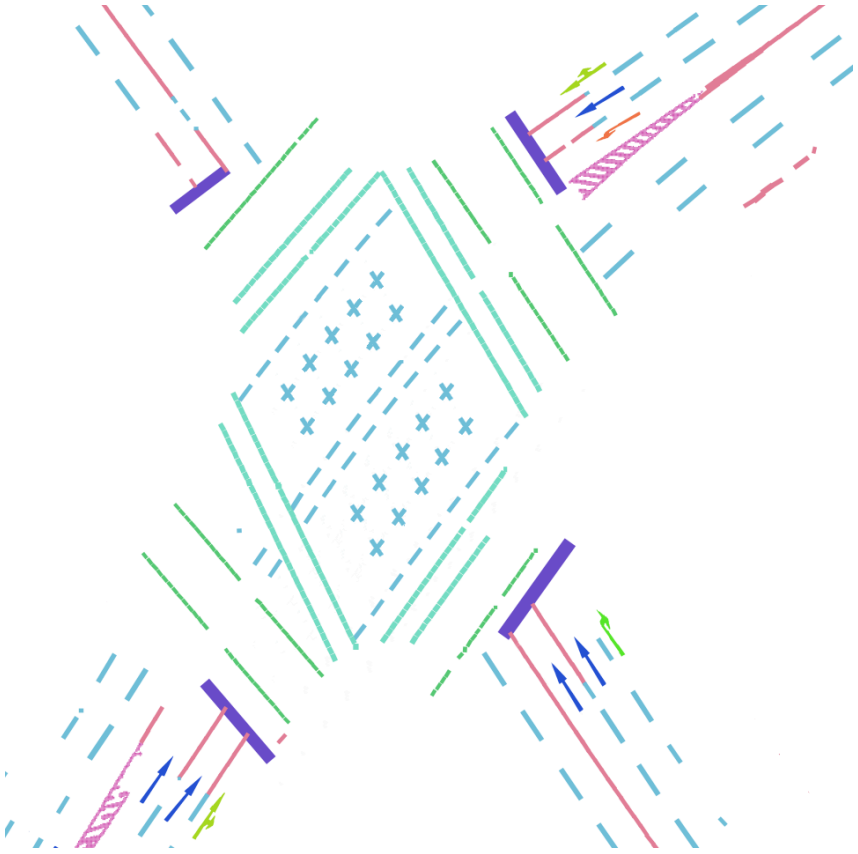

Figure 5: Automatically created shape file with shapes for individual road markings. Different colors represent different semantic classes.

When creating convex hulls for rectangular road markings, these often have rough edges, resulting in a noisy visualization. A better approach is therefore the representation via oriented rectangles, concerning width, height and orientation of the road marking.

To create shapes for lines consisting of individual parts, where neighboring line segments should have the same orientation, larger clusters are created. Lines from the same type are collected into a combined cluster, as long as the distance to the next neighboring line is smaller than a given threshold and the orientation does only differ by a small amount. This prevents taking lines at corners which are oriented perpendicular to each other into the same cluster. For each of these larger clusters, the orientation can now be determined on the points of all road markings that are part of this cluster. All of the rectangles in the cluster will then be oriented in the calculated direction.

Arrows on the road are used to show which lane must be used for which direction at a crossing. There is only a relatively small number of arrows that are used in almost all situations, namely those pointing left, straight, right, and any combination of those. For this reason, templates of arrow shapes can be used. These templates are placed on the position where an arrow marking was found. The orientation is determined as best-fit in a way that they are covering the largest number of points of the detected road marking. Using templates results in clearly shaped arrows in the final result.

For other markings, such as barred areas and intersecting lines, the shapes are constructed based on a grid measuring which grid cells contains points of the road marking. Those cells containing points define the area that should be spanned by the created shape. A polygon is generated by iterating over all outer cells at the border of the marking, the respective outermost point will be used as a vertex for the generated polygon, resulting in a shape fitting closely to the detected road marking. The Douglas-Peucker algorithm [7] is used for shape simplification.

Related Work 3D point clouds commonly serve as base data to automatically derive 3D city models and landscape models [28] for many different use cases in urban planning for local authorities, companies, or individuals [33]. As showcased by several existing tools, cadastral data can be visualized in combination with 3D point clouds to provide additional context and further facilitate the visual analysis and task-oriented exploration of captured data sets [1]. High density point information can be analyzed and creating large models gets possible, requiring only minimal manual effort [25]. Besides, aerial captures of 3D point clouds, mobile mapping techniques are widely used [16]. Mobile mapping scans can be used to, e. g., automatically extract road networks, or to analyze road surfaces [11], as well as for the reconstruction of building facades.

For many use cases the automated semantic analysis of 3D point clouds is a mandatory preparation. Examples include ground detection, tree analysis, cadastre comparison, or automated 3D model generation. Semantic classification can be performed by two fundamentally different approaches: Semantic per-point surface category infor-

mation can be derived by analyzing a 3D point cloud's topology [6] or by applying deep learning concepts [4].

Traditionally, explicit rules are defined to distinguish semantic classes by geometric attributes [8]. 3D point clouds can be segmented into local groups of points with, e. g., similar surface directions [23]. Each of these segments can then be analyzed with regard to their size and orientation. Large, vertical surfaces would, for example, be identified as building facades, whereas groups of points whose corresponding surface normals are pointing into many different directions usually represent part of vegetation [35]. An alternative approach uses machine learning techniques to identify the semantic classes of objects by using previously trained neural networks [39]. Such networks use an already classified dataset for training and learn to predict the semantic class for individual points or groups of points in new, unknown datasets. In recent years, using the internal structure of the 3D point clouds themselves has become increasingly popular, as exemplified by PointNet and similar networks [22]. However, those approaches often focus on small datasets of separated objects and require large training datasets.

Detecting objects in image data is a relevant research field for many application fields, such as face recognition, license plate identification or medical imagery analysis. Viola et al. present an image object detection algorithm which can be used to detect, e. g., faces in images [31]. U-Net, originally developed in a medical context, is nowadays widely used for image segmentation [26]. It enables the automated detection of specific areas in images, such as cancer cells but also roads in aerial images [38]. The extraction of road marking information from images taken from a car is discussed by Vacek et al., who detect lanes and arrows markings [29]. Veit et al. present an approach to evaluate the performance of algorithms for the road marking detection in images in general [30].

Yang et al. use the reflective properties of road markings to extract them from LiDAR point clouds [36]. Similarly, Guan et al. present how mobile laser scanning data can be used for an extraction of road markings [9]. They perform a curb-based road extraction, followed by rendering intensity images of the 3D point cloud and a final extraction step, segmenting the areas containing road markings. Yu et al. take this approach one step further by distinguishing seven specific types of road markings by five classification methods [37].

Multiple authors describe possible approaches for railroad track detection in mobile mapping scans [12, 13, 15]. Image-based classification and the usage of characteristics in individual scan lines have not yet been examined in detail to the knowledge of the author.

Future Work Besides road markings, the implementation can also be used to identify manhole covers and similar structures on the road, if labeled data sets with training data for the neural network are available.

The approach for railroad tracks is still to be implemented, allowing for the automated identification of rails and maybe also ties and balises, as well as the extraction of generalized rail network plans which are essential for the maintenance and surveillance of such infrastructure.

In the future, we envision the ever increasing affordability of 3D scanning technology to result in scans being conducted more and more regular and by a larger variety of sensor systems, eventually leading to data sets that get updated every few minutes. The size of those data sets would dwarf that of today's scans, thus further necessitating the use of efficient classification approaches like the one described in this report.

References

[1] K. Aringer and R. Roschlaub. "Bavarian 3D building model and update concept based on LiDAR, image matching and cadastre information". In: *Innovations in 3D Geo-Information Sciences*. Springer, 2014, pages 143–157.

[2] Z. Ž. Avramović, D. M. Marinković, and I. T. Lastrić. "Digitalization of Railways – ICT Approach to the Development of Automation". In: *Journal of Information Technology & Applications* 9.1 (2019).

[3] D. Bétaille and R. Toledo-Moreo. "Creating enhanced maps for lane-level vehicle navigation". In: *IEEE Transactions on Intelligent Transportation Systems* 11.4 (2010), pages 786–798.

[4] A. Boulch, B. L. Saux, and N. Audebert. "Unstructured point cloud semantic labeling using deep segmentation networks". In: *Proceedings of 3DOR*. Volume 2. 2017, page 1.

[5] G. Caroti, A. Piemonte, and B. Pucci. "Terrestrial Laser Scanning as Road's Cadastre Revision And Integration Support". In: *ISPRS Workshop Italy-Canada 2005 3D Digital Imaging and Modeling: Applications of Heritage, Industry*. Volume 1. CIRGEO. 2005, pages 1–3.

[6] D. Chen, R. Wang, and J. Peethambaran. "Topologically aware building rooftop reconstruction from airborne laser scanning point clouds". In: *IEEE TGRS* 55.12 (2017), pages 7032–7052.

[7] D. H. Douglas and T. K. Peucker. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature". In: *Cartographica: the international journal for geographic information and geovisualization* 10.2 (1973), pages 112–122.

[8] E. Grilli, F. Menna, and F. Remondino. "A Review of Point Clouds Segmentation and Classification Algorithms". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 42 (2017), page 339.

[9] H. Guan, J. Li, Y. Yu, C. Wang, M. Chapman, and B. Yang. "Using mobile laser scanning data for automated extraction of road markings". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 87 (2014), pages 93–107.

[10] N. Haala, M. Peter, J. Kremer, and G. Hunter. "Mobile LiDAR mapping for 3D point cloud collection in urban areas, A performance test". In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 37 (2008), pages 1119–1127.

[11] A. Jaakkola, J. Hyyppä, H. Hyyppä, and A. Kukko. "Retrieval algorithms for road surface modelling using laser-based mobile mapping". In: *Sensors* 8 (2008), pages 5238–5249.

[12] Y. Jwa and G. Sonh. "KALMAN FILTER BASED RAILWAY TRACKING FROM MOBILE LIDAR DATA." In: *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences* 2 (2015).

[13] J. Kremer and A. Grimm. "The RailMapper – A dedicated mobile LiDAR mapping system for railway networks". In: *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci* 39 (2012), page 477.

[14] Y. LeCun, K. Kavukcuoglu, and C. Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE. 2010, pages 253–256.

[15] M. Leslar, G. Perry, and K. McNease. "Using mobile lidar to survey a railway line for asset inventory". In: *Proceedings of the ASPRS 2010 Annual Conference, San Diego, CA, USA*. 2010, pages 26–30.

[16] R. Li. "Mobile mapping: An emerging technology for spatial data acquisition". In: *Photogrammetric Engineering and Remote Sensing* 63.9 (1997), pages 1085–1092.

[17] M. Maurer, J. C. Gerdes, B. Lenz, H. Winner, et al. *Autonomous driving*. Berlin: Springer, 2016.

[18] S. Mikrut, P. Kohut, K. Pyka, R. Tokarczyk, T. Barszcz, and T. Uhl. "Mobile Laser Scanning Systems for Measuring the Clearance Gauge of Railways: State of Play, Testing and Outlook". In: *Sensors* 16.5 (2016), page 683.

[19] J. Niemeyer, F. Rottensteiner, and U. Soergel. "Conditional Random Fields for LiDAR Point Cloud Classification in Complex Urban Areas". In: *ISPRS annals of the photogrammetry, remote sensing and spatial information sciences* 1.3 (2012), pages 263–268.

[20] S. Pu, M. Rutzinger, G. Vosselman, and S. O. Elberink. "Recognizing Basic Structures from Mobile Laser Scanning Data for Road Inventory Studies". In: *ISPRS Journal of Photogrammetry and Remote Sensing* 66.6 (2011), pages 28–39.

[21] K. Pulli, A. Baksheev, K. Kornyakov, and V. Eruhimov. "Real-time computer vision with OpenCV". In: *Communications of the ACM* 55.6 (2012), pages 61–69.

[22] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. "Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pages 652–660.

[23] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann. "Segmentation of Point Clouds Using Smoothness Constraint". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences* 36.5 (2006), pages 248–253.

[24] R. Richter, M. Behrens, and J. Döllner. "Object Class Segmentation of Massive 3D Point Clouds of Urban Areas Using Point Cloud Topology". In: *International Journal of Remote Sensing* 34.23 (Dec. 2013), pages 8408–8424.

[25] R. Richter and J. Döllner. "Concepts and techniques for integration, analysis and visualization of massive 3D point clouds". In: *Computers, Environment and Urban Systems* 45 (2013), pages 114–124.

[26] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical image computing and computer-assisted intervention*. Springer. 2015, pages 234–241.

[27] M. Safa, A. Sabet, K. Ghahremani, C. Haas, and S. Walbridge. "Rail corrosion forensics using 3D imaging and finite element analysis". In: *International Journal of Rail Transportation* 3.3 (2015), pages 164–178.

[28] E. Schwalbe, H.-G. Maas, and F. Seidel. "3D building model generation from airborne laser scanner data using 2D GIS data and orthogonal point cloud projections". In: *Proceedings of ISPRS WG III/3, III/4* 3 (2005), pages 12–14.

[29] S. Vacek, C. Schimmel, and R. Dillmann. "Road-marking Analysis for Autonomous Vehicle Guidance." In: *EMCR*. 2007, pages 1–6.

[30] T. Veit, J.-P. Tarel, P. Nicolle, and P. Charbonnier. "Evaluation of road marking feature extraction". In: *11th International IEEE Conference on Intelligent Transportation Systems*. IEEE. 2008, pages 174–181.

[31] P. Viola, M. Jones, et al. "Rapid object detection using a boosted cascade of simple features". In: *CVPR (1)* 1.511-518 (2001), page 3.

[32] G. Vosselman. "3D Reconstruction of Roads and Trees for City Modelling". In: *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Dresden, Germany* 34 (2003), page 3.

[33] G. Vosselman, E. Dijkman, K. W. B. Reconstruction, L. Altimetry, and H. Transform. "3D Building Model Reconstruction from Point Clouds and Ground Plans". In: *Int. Arch. of Photogrammetry and Remote Sensing* XXXIV, Part 3/W4 (2001), pages 37–43.

[34] G. Vosselman, B. G. Gorte, G. Sithole, and T. Rabbani. "Recognising Structure in Laser Scanner Point Clouds". In: *International archives of photogrammetry, remote sensing and spatial information sciences* 46.8 (2004), pages 33–38.

[35] J. Wolf, R. Richter, and J. Döllner. "Techniques for Automated Classification and Segregation of Mobile Mapping 3D Point Clouds". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. 2019, pages 201–208.

[36] B. Yang, L. Fang, Q. Li, and J. Li. "Automated extraction of road markings from mobile LiDAR point clouds". In: *Photogrammetric Engineering & Remote Sensing* 78.4 (2012), pages 331–338.

[37] Y. Yu, J. Li, H. Guan, F. Jia, and C. Wang. "Learning hierarchical features for automated extraction of road markings from 3-D mobile LiDAR point clouds". In: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 8.2 (2014), pages 709–726.

[38] Z. Zhang, Q. Liu, and Y. Wang. "Road Extraction by Deep Residual U-Net". In: *IEEE Geoscience and Remote Sensing Letters* 15.5 (2018), pages 749–753.

[39] Y. Zhou and O. Tuzel. "Voxelnet: End-to-end Learning for Point Cloud Based 3D Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pages 4490–4499.

Transcalibur: A 2D Weight Shifting Virtual Reality Controller for Shape Rendering

Jotaro Shigeyama

Human-Computer Interaction Group
Hasso Plattner Institute for Digital Engineering
jotaro.shigeyama@hpi.uni-potsdam.de

Humans can estimate the shape of a wielded object through the illusory feeling of the mass properties of the object obtained using their hands. Even though the shape of hand-held objects influences immersion and realism in virtual reality (VR), it is difficult to design VR controllers for rendering desired shapes according to the perceptions derived from the illusory effects of mass properties and shape perception. We propose Transcalibur, which is a hand-held VR controller that can render a 2D shape by changing its mass properties on a 2D planar area. We built a computational perception model using a data-driven approach from the collected data pairs of mass properties and perceived shapes. This enables Transcalibur to easily and effectively provide convincing shape perception based on complex illusory effects. Our user study showed that the system succeeded in providing the perception of various desired shapes in a virtual environment.

1 Introduction

Intensive developments in computing power and the increasing resolution of displays have enabled virtual reality (VR) to become accessible for everyone and at any location. With the growing demand for rich immersion and realism in VR along high audiovisual fidelity, there is a significant need for haptic feedback techniques since interaction between virtual object and users is essential for VR. Thus, the number of haptic devices for VR has been proposed that are mainly focused on providing force [2, 3, 8, 10, 12] or texture or tactile feedback [17]. As for immersion, the sensation of the shape of the wielded object in VR is also important. Because conventional VR controllers cannot render the shape of virtual objects, they thus lack immersion and realism.

We propose a weight shifting controller for dynamic 2D haptic shape rendering. The resulting device *Transcalibur* has a weight shifting mechanism and angle mechanism that alters mass properties in a 2D planar area, which allows for rendering various shape sensation.

Although psychological studies have revealed how the mass properties or inertia of the wielded object affect the perceived shape of an unseen object, deterministic mapping between the mass property and the perceived shape cannot be derived owing to the nonlinearity of human perception characteristics. This makes it difficult to manually adjust and configure Transcalibur's weight positions. In this paper, we

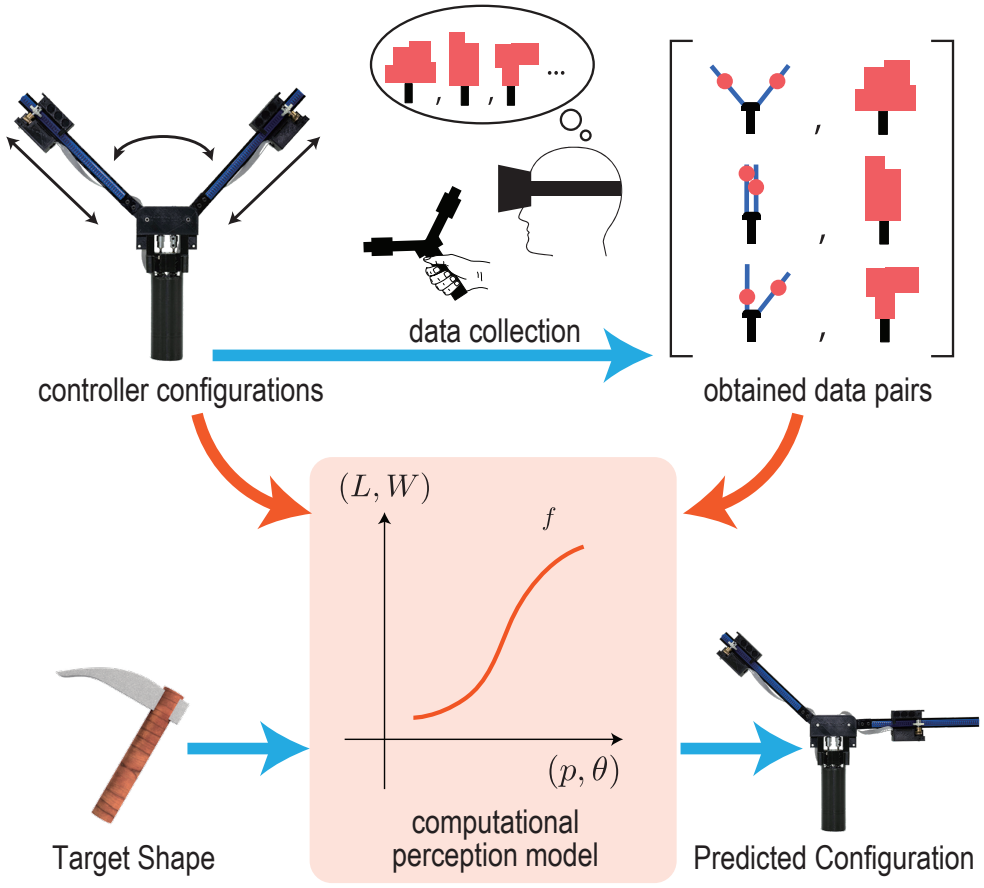

Figure 1: Transcalibur renders shape using the computational perception model obtained by data collection. The collected data is fit to the regression model, which predicts the optimal configuration of the controller for the target shape in VR.

also applied a data-driven method to determine the best physical state of Transcalibur for the desired object to render in VR: namely the Computational Perception Model.

Figure 1 shows our approach. Based on the object shown in VR, the shape of the controller that users grasp is dynamically changed so that its shape perception matches the target object.

2 Transcalibur

Transcalibur is a hand-held VR controller that dynamically and illusorily changes the perceived shape of a 2D object. Figure 2 shows the hardware overview of the Transcalibur. The weight moving mechanisms on the controller changes its 2D mass properties, which provides various shape perceptions for the user in VR. The perceived shape can be rendered based on a perception model which is built by precol-

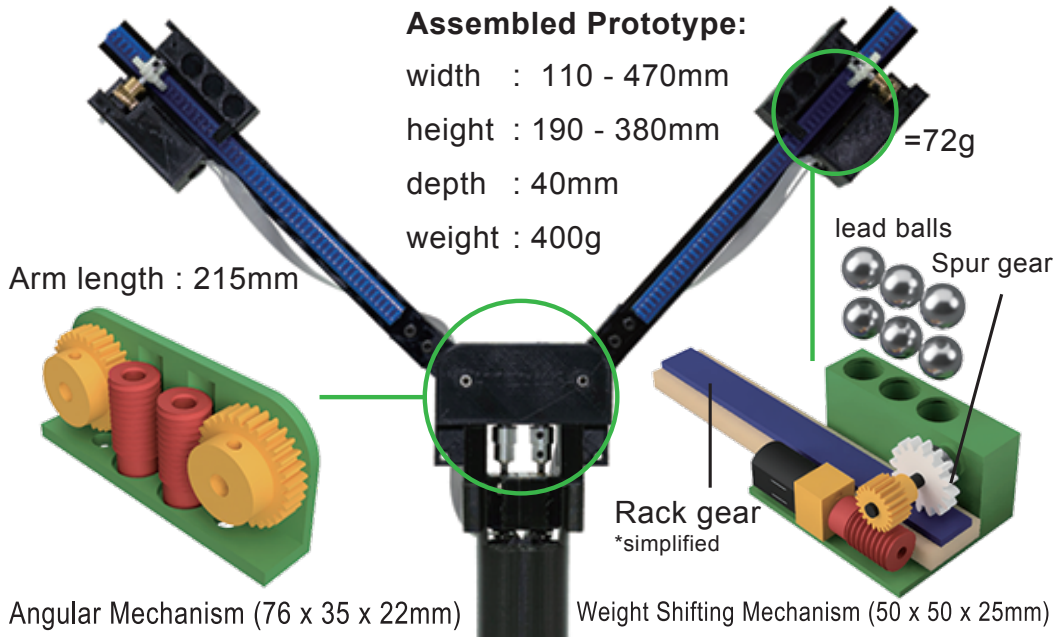

Figure 2: Mechanical design of the proposed device. The angular mechanism (left bottom) and weight shifting mechanism (right bottom) are actuated by a motor through the worm (red) and the worm wheel (yellow).

lected perception data through our experiment. In this section, we describe the core concept of Transcalibur.

2.1 Computational Perception Model on Haptic Shape Illusion

When we grab and wiggle an object, we feel the object through haptic sensation: the skin on the hand in contact will be stretched and the muscles and tendons of the arm will contract. This means that humans can guess what s/he is holding even when their eyes are closed. Researchers in psychology have revealed that the mass properties of an object, such as rotational inertia and center of gravity, affect the perception of what shape of the object people are wielding [9, 11, 15, 16]. This occurs even when the actual shape differs from the perceived shape. That being said, we can illusorily present various sensations of wielded objects through changes in the mass properties of the object. We call this Haptic Shape Illusion. By utilizing this effect, we can simulate the shape of a wielded object without using the actual shape of a targeted object.

This will allow the system to reduce the spatial cost when provided with a large object such as a longsword or an axe in the game as it is important to provide various haptic sensations at a low cost in conventional VR systems for the public. Our approach is to simulate haptic feedback through the illusion. Transcalibur aims to provide such an effect dynamically using just a single controller.

Haptic shape illusion involves utilizing the perceptual illusion existing between the mass properties of a wielding object and shape perception. However, in most cases, this relation makes it difficult to predict how or how much of the illusion effect could occur, especially for a VR experience designer, owing to the nonlinearity of human perception, which means that what one perceives is not always consistent with what one is actually exposed to. Even if the type of sensory input and perceptual phenomenon are clarified through previous psychology studies, their mapping must be restudied and reoptimized when a device or an environment that we assume is different from the one observed in previous studies. Even though this procedure is necessary for constructing this type of interface, finding the mapping through psychophysics studies that repeatedly presents different sensory stimuli to determine the occurrence of the perceptual phenomenon requires considerable time and effort.

To overcome this problem, we apply the Computational Perception Model to this device. This approach allows providing haptic feedback based on actual human perception data. We collect data pairs of physical configurations of Transcalibur (positions of weight modules on the controller) and the shapes displayed in VR for users and fit to the machine learning model. Then, from the desired shape that we want to display in VR, we derive Transcalibur configurations from the trained model. In this manner, we can easily and efficiently design and provide haptic shape experiences for various users.

3 Related Work

3.1 Haptic Display for VR

There are numerous projects that deal with increasing immersion and realism in VR by providing various types of haptic feedback. A force feedback system with mechanical linkages is used to simulate contact against a human body. Linkages on the hand [3, 6, 18] or fingers [2, 13] are used to simulate grasping virtual objects. Collisions on the fingers are simulated as well [13]. Actuators are attached to a conventional VR controller to combine tracking and button input capability with a force feedback system [14]. These force feedback systems require a number of actuators, especially for 6-DoF force feedback, that have to be grounded, and this makes the mechanism more complicated and expensive. Propeller for aerobotics or drones are applied [7, 8, 12] to provide force feedback as well, but they emit considerable noise and wind while being used, which is distracting for an immersive VR experience.

3.2 Shape perception, Displays, and Applications for VR

Researchers in the psychological field explored the underlying mechanisms of the ability of estimating size and shapes of the wielded object. “Dynamic Touch” named by Turvey et al. [15] has been investigated through various psychological experiments

[16]. Based on these results, it has been observed that the principal component of the object's inertia tensor plays a significant role in perceiving shape.

Several haptic displays for shape perception have been proposed. As for a Tabletop-type, shape displays using a 2D array of linear actuators are proposed [4] to render the geometric surface of the virtual object. shapeShift enables to track and freely move the array shape display and renders the shape of a spatially registered object. Zhao et al. [20] proposed a self-assembling multi-robot system to present a physical equivalent of the target object. However, these types of devices require several motors and mechanical components. Benko et al. [1] utilized a displacement on a fingertip to render the shape of a virtual object in VR. However, these types of devices require several motors and mechanical components.

Shifty [19] is a VR controller that simulates the length of the object by moving the weight on the stick, and the motion is tracked by a VR system so that the users can feel its change along the length in VR. This device also enables thickness rendering, but the variety of the shape rendered by this device is constrained in a 1D symmetric shape.

4 Perception Model Design

For computational derivation of shape of the hand-held controller for desired shape feedback, we conducted data collection using our prototype. Then, the collected data pairs are fitted as a linear regression model that provides the predicted shapes that matches the target object in VR. In this section, we explain how we build a perception model that maps the physical state of Transcalibur to the users' shape perception in VR.

4.1 Mapping from Controller to Virtual Shape

As a previous work on computational fabrication of hand-held controller [5] proposed, we assumed a perception model f that maps the physical configuration of the controller ϕ to the perceived shape of the wielded object in VR ψ:

$$f : \phi \mapsto \psi \tag{1}$$

Transcalibur has four configuration parameters: positions of the weight module on the arms, and angles of the angular mechanism of the arms. In this manner, Transcalibur can change its inertia moment and center of gravity, then alter shape perception of an hand held object. Since we want to give a perceived object mapped to a mechanical configuration of Transcalibur instead of directly calculating inertia tensor, we represent this as $\phi = (p_r, \theta_r, p_l, \theta_l)$ and ϕ_i as i-th obtained through the data collection experiment.

On the contrary, we represent the perceived object using its 2D boundary box of the object and its center of gravity (CoG) as $\psi = (H, W, G_x, G_y)$, where an obtained i-th data is written ψ_i as well. Our goal is to provide the best estimate of the shape

of the controller ϕ from the model f, given the target perceived object ψ in VR. We describe the manner in which we collected data, built the model, and estimated the values.

4.2 Collecting Perceived Shape Data

4.2.1 Experiment Set

In the data collection experiment, we provided the participants with various shapes of the controller and asked them to report the perceived shapes in VE. This generates matched pairs of (ϕ_i, ψ_i), which are used to build a regression model for the training data. To collect various shapes over a wide range within the constraints of the user study, we manually defined the number of shapes to provide in VE. The mass properties of the controller presented and the exact values used for configurations are shown in Figure 3. Because we consider those shapes that will remain the same when the controller is flipped, $49 - (49 - 7)/2 = 28$ different configurations of the controller are presented to the participants.

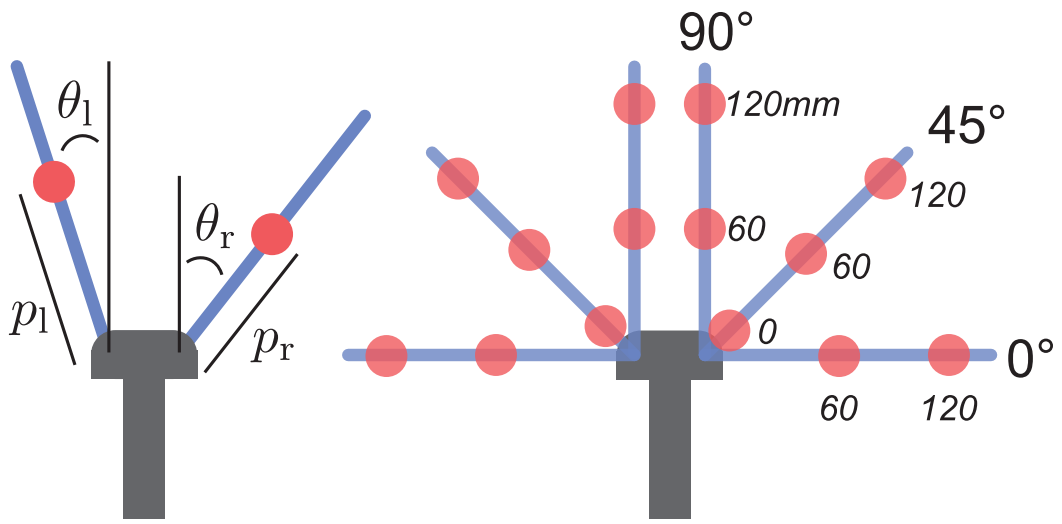

Figure 3: Illustrated configuration variables of Transcalibur $(p_r, \theta_r, p_l, \theta_l)$. The red circles illustrate the positions of the weight module (left). Group of variables used in the experiment (right).

4.2.2 VR System

In the data collection experiment, participants wore a HTC Vive head-mounted display (HMD) that was tracked by its motion capture cameras (VIVE lightroom). The VE is run on Unity game engine, and deployed using a gaming PC (iiyama LEVEL infinity, Intel Core i7-6700 CPU @ 3.40 GHz, NVIDIA GeForce GTX 1080), and a joypad (Sony Playstation DUALSHOCK4) is utilized for user input. A Vive tracker is attached to Transcalibur for motion tracking and wireless communication.

In the VE, participants can see a virtual handle in the exact same size and position as the actual Transcalibur and can move/operate it naturally. On the top of the handle, four red rectangular plates are mounted, and users can adjust their heights and widths using the joypad.

$$d_i = (H_{ru}, W_{ru}, H_{rd}, W_{rd}, H_{lu}, W_{lu}, H_{ld}, W_{ld})_i \tag{2}$$

These eight variables are converted into four variables representing a rectangular bounding box the shape and a Center of Gravity (CoG) coordinates (H, W, G_x, G_y). As described in the section "Mapping from Controller to Virtual Shape", this description makes simpler to represent the mass property of the perceived object, and easier for mathematical model to fit the training data. Participants were then asked to adjust these eight variables with the trigger button and arrow keys of the joypad such that the virtual appearance matched the haptically perceived shape. We also presented hardware having an acrylic square plane identical to the initial size of the VR object as a reference controller to control the assumptions for the material displayed in VR, which may otherwise differ among participants. We consider that the weights of the VIVE tracker, which is attached to Transcalibur, and acrylic square plane controller affect the perception of the shape; therefore, the 3D model of the VIVE tracker is also visually presented in VR (Figure 4).

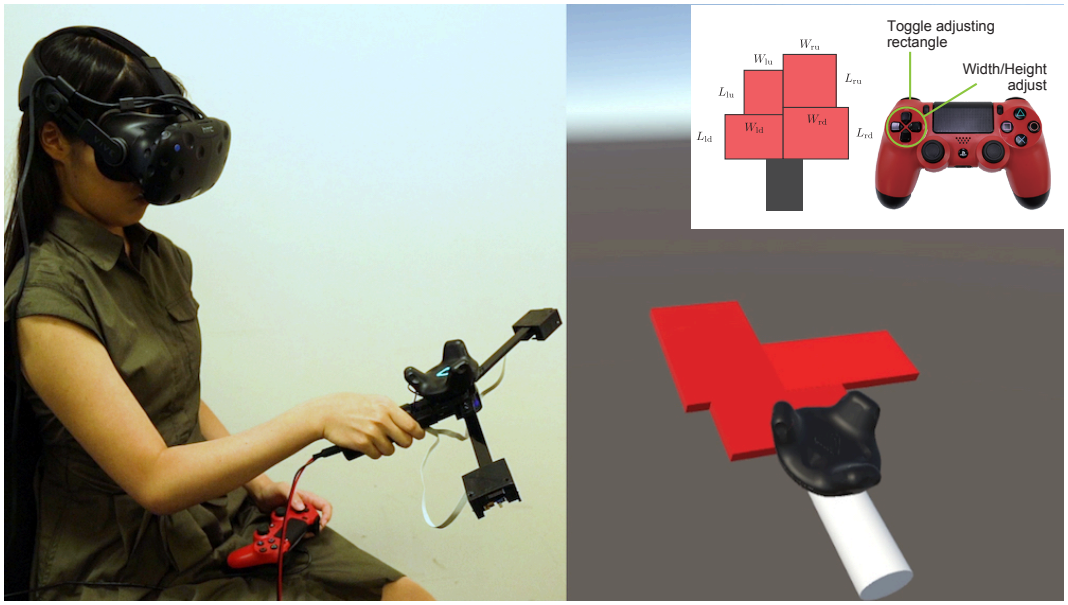

Figure 4: In the data collection experiment, a participant grabs and wiggles the Transcalibur with their dominant hand and uses joypad with their non dominant hand. In VE (right) the participants can see the adjustable shape. Participants can use the gamepad to adjust the shape parameters (upper-right).

4.3 Regression Models and Results

Using the obtained data pairs, we performed regression analysis to build a map f from the configurations of the controller onto the perceived shapes. For each parameter in ϕ, we used a linear regression model and trained model $\psi = f(\phi; A, b)$ to describe the perceived shape $\psi = \phi A + b$, where $A \in R^{4\times4}, b \in R^{1\times4}$ as parameter and $\phi, \psi \in R^{1\times4}$ as an input and output data.

$$E = E_H + E_W + E_{G_x} + E_{G_y} \tag{3}$$

where each cost can be described using the perceived height H, width W and CoG (G_x, G_y) and the target height H^{target}, width W^{target}, and CoG $({G_x}^{target}, {G_y}^{target})$.

$$\begin{aligned} E_H &= ||f_H(p_r, \theta_r, p_l, \theta_l) - H^{target}||^2 && (4)\\ E_W &= ||f_W(p_r, \theta_r, p_l, \theta_l) - W^{target}||^2 && (5)\\ E_{G_x} &= ||f_{G_x}(p_r, \theta_r, p_l, \theta_l) - {G_x}^{target}||^2 && (6)\\ E_{G_y} &= ||f_{G_y}(p_r, \theta_r, p_l, \theta_l) - {G_y}^{target}||^2 && (7) \end{aligned}$$

To obtain an optimal controller shape ϕ^*, we defined the following optimization problem with the hardware constraints of the controller.

$$\begin{cases} \phi^* = argmin_\phi E \\ s.t. \quad 0 \le p_r, p_l \le 125, 0 \le \theta_r, \theta_l \le 90 \end{cases} \tag{8}$$

We used COBYLA optimizer for solving the linear regression prediction and executed the solution in the Python Scikit-learn environment. Figure 5 shows examples of the predicted shapes of the controller calculated by the optimizer, which are later used in the evaluation experiments.

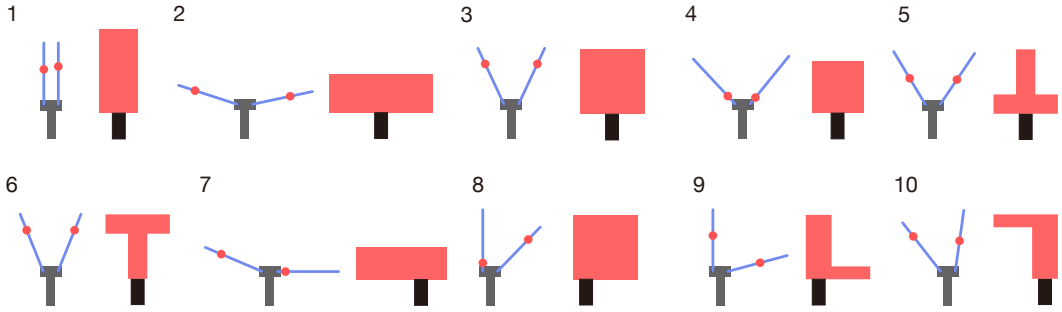

Figure 5: Ten pairs of physical properties of Transcalibur and their corresponding virtual shapes provided in the experiment. Each configuration of Transcalibur is predicted from the linear regression model.

5 Model Evaluation

To measure the validity of the perception model, we conducted validation experiments. We prepared ten shapes to present in the VE, which can be described using the same parameter format d_i as that of the data collection experiment (Figure 5). The ten virtual shapes were manually determined such that variations in height, width, symmetricity, and asymmetricity of the target shapes could be evaluated.

5.1 Results

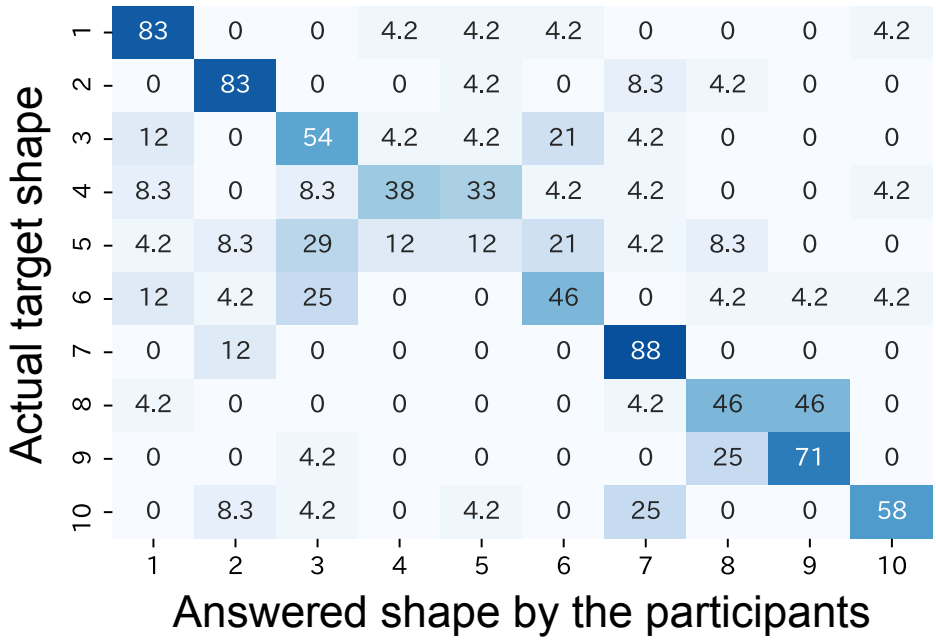

Figure 6: Confusion matrix derived from the results of the experiment. Each row shows ratio [%] of the actual target shape answered as the test shape in VR.

Figure 6 shows the confusion matrix derived from the accuracies of the selected shapes of the object.

Overall, our perception model succeeded in providing various target shapes in VR for Transcalibur, leaving a few shapes with confusions on shapes 4 and 5 or 8 and 9.

6 Conclusion

We introduced Transcalibur: the weight moving VR controller for 2D haptic shape illusion. We implemented a hardware prototype, which can change its mass property in 2D planar space, and applied data-driven methods to obtain maps between mass property and perceived shape. Based on the demonstration and experiment, we succeeded in rendering various shape perceptions through the controller based on pre-computed perception model. As a future work, we further investigate details on time factor of shape changing in VR, and we aim to develop a simpler design and yet maximizes range of rendering shape.

References

[1] H. Benko, C. Holz, M. Sinclair, and E. Ofek. "NormalTouch and TextureTouch". In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*. 2016, pages 717–728. ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984526.

[2] I. Choi and S. Follmer. "Wolverine: A Wearable Haptic Interface for Grasping in VR". In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. UIST '16 Adjunct. Tokyo, Japan: ACM, 2016, pages 117–119. ISBN: 978-1-4503-4531-6. DOI: 10.1145/2984751.2985725.

[3] I. Choi, E. Ofek, H. Benko, M. Sinclair, and C. Holz. "CLAW: A Multifunctional Handheld Haptic Controller for Grasping, Touching, and Triggering in Virtual Reality". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (2018), pages 1–13. DOI: 10.1145/3173574.3174228.

[4] S. Follmer, D. Leithinger, A. Olwal, A. Hogge, and H. Ishii. "inFORM". In: *Proceedings of the 26th annual ACM symposium on User interface software and technology - UIST '13*. 2013, pages 417–426. ISBN: 978-1-4503-2268-3. DOI: 10.1145/2501988.2502032.

[5] E. Fujinawa, S. Yoshida, Y. Koyama, T. Narumi, T. Tanikawa, and M. Hirose. "Computational design of hand-held VR controllers using haptic shape illusion". In: *Proceedings of the 23rd ACM Symposium on Virtual Reality Software and Technology - VRST '17*. 2017, pages 1–10. ISBN: 978-1-4503-5548-3. DOI: 10.1145/3139131.3139160.

[6] Y. M. Han, C. J. Kim, and S. B. Choi. "A magnetorheological fluid-based multifunctional haptic device for vehicular instrument controls". In: *Smart Materials and Structures* 18.1 (2009). ISSN: 09641726. DOI: 10.1088/0964-1726/18/1/015002.

[7] S. Heo, C. Chung, G. Lee, and D. Wigdor. "Thor's Hammer: An Ungrounded Force Feedback Device Utilizing Propeller-Induced Propulsive Force". In: *Chi* (2018), pages 1–11. DOI: 10.1145/3170427.3186544.

[8] S. Je, H. Lee, M. J. Kim, and A. Bianchi. "Wind-blaster: A Wearable Propeller-based Prototype That Provides Ungrounded Force-feedback". In: *ACM SIGGRAPH 2018 Emerging Technologies*. SIGGRAPH '18. Vancouver, British Columbia, Canada: ACM, 2018, 23:1–23:2. ISBN: 978-1-4503-5810-1. DOI: 10.1145/3214907.3214915.

[9] I. Kingma, R. Van De Langenberg, and P. J. Beek. "Which Mechanical Invariants Are Associated With the Perception of Length and Heaviness of a Nonvisible Handheld Rod? Testing the Inertia Tensor Hypothesis". In: *Journal of Experimental Psychology: Human Perception and Performance* 30.2 (2004), pages 346–354. ISSN: 0096-1523. DOI: 10.1037/0096-1523.30.2.346.

[10] K. Minamizawa and S. Fukamachi. "Gravity grabber: wearable haptic display to present virtual mass sensation". In: *ACM SIGGRAPH 2007 emerging technologies* (2007), page 8. DOI: 10.1145/1278280.1278289.

[11] C. C. Pagano, P. Fitzpatrick, and M. T. Turvey. "Tensorial basis to the constancy of perceived object extent over variations of dynamic touch". In: *Perception & Psychophysics* 54.1 (1993), pages 43–54. ISSN: 00315117. DOI: 10.3758/BF03206936.

[12] T. Sasaki, R. S. Hartanto, K.-H. Liu, K. Tsuchiya, A. Hiyama, and M. Inami. "Leviopole: Mid-air Haptic Interactions Using Multirotor". In: *ACM SIGGRAPH 2018 Emerging Technologies*. SIGGRAPH '18. Vancouver, British Columbia, Canada: ACM, 2018, 12:1–12:2. ISBN: 978-1-4503-5810-1. DOI: 10.1145/3214907.3214913.

[13] S. B. Schorr and A. M. Okamura. "Fingertip Tactile Devices for Virtual Object Manipulation and Exploration". In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems - CHI '17*. 2017, pages 3115–3119. ISBN: 978-1-4503-4655-9. DOI: 10.1145/3025453.3025744.

[14] E. Strasnick, C. Holz, E. Ofek, M. Sinclair, and H. Benko. "Haptic Links: Bimanual Haptics for Virtual Reality Using Variable Stiffness Actuation". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (2018), pages 1–12. DOI: 10.1145/3173574.3174218.

[15] M. T. Turvey. "Dynamic Touch". In: *American Psychologist* 51.11 (1996), pages 1134–1152. ISSN: 0003-066X. DOI: 10.1037/0003-066X.51.11.1134.

[16] M. T. Turvey, G. Burton, E. L. Amazeen, M. Butwill, and C. Carello. "Perceiving the Width and Height of a Hand-Held Object by Dynamic Touch". In: *Journal of Experimental Psychology: Human Perception and Performance* 24.1 (1998), pages 35–48. ISSN: 00961523. DOI: 10.1037/0096-1523.24.1.35.

[17] E. Whitmire, H. Benko, C. Holz, E. Ofek, and M. Sinclair. "Haptic Revolver: Touch, Shear, Texture, and Shape Rendering on a Reconfigurable Virtual Reality Controller". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18* (2018), pages 1–12. DOI: 10.1145/3173574.3173660.

[18] V. Yem, R. Okazaki, and H. Kajimoto. "FinGAR: Combination of Electrical and Mechanical Stimulation for High-Fidelity Tactile Presentation". In: *ACM SIGGRAPH 2016 Emerging Technologies on - SIGGRAPH '16*. 2016, pages 1–2. ISBN: 978-1-4503-4372-5. DOI: 10.1145/2929464.2929474.

[19] A. Zenner and A. Kruger. "Shifty: A Weight-Shifting Dynamic Passive Haptic Proxy to Enhance Object Perception in Virtual Reality". In: *IEEE Transactions on Visualization and Computer Graphics* 23.4 (2017), pages 1312–1321. ISSN: 1077-2626. DOI: 10.1109/TVCG.2017.2656978.

[20] Y. Zhao, L. H. Kim, Y. Wang, M. Le Goc, and S. Follmer. "Robotic Assembly of Haptic Proxy Objects for Tangible Interaction and Virtual Reality". In: *Proceedings of the 2017 ACM International Conference on Interactive Surfaces and Spaces*. ISS '17. Brighton, United Kingdom: ACM, 2017, pages 82–91. ISBN: 978-1-4503-4691-7. DOI: 10.1145/3132272.3134143.

Multi-Tenancy for FPGA Accelorator Designs

Lukas Wenzel

Operating Systems and Middleware Group
Hasso Plattner Institute for Digital Engineering
lukas.wenzel@hpi.de

FPGA accelerators offer important performance and efficiency benefits in increasingly heterogeneous compute infrastructures. Domains like cloud or service-oriented computing could benefit from this, but their multi-tenant nature limits the effective use of accelerator resources: The simultaneous execution of tasks from different tenants on a the same FPGA usually lacks firm isolation guarantees as well as per-task customizability.

My current research is focused on adapting established resource management and isolation concepts from operating systems for general-purpose compute platforms to the unique characteristics of FPGA accelerators.

1 Overview

The ever increasing volume and performance demands of compute workloads, coupled with physical and architectural limitations of general-purpose compute platforms to satisfy them, has led many application domains to shift focus towards accelerators in heterogeneous system architectures. Especially FPGA accelerators promise a very efficient computation, because they can be customized to accomodate a specific workload and avoid overheads present in fixed-function accelerators.

In certain domains like service-oriented and cloud computing, a single compute platform can serve multiple requests, users or organizations (tenants) simultaneously. On general-purpose platforms, sharing between tenants is facilitated by well established resource abstraction mechanisms on the hardware and operating system level. In contrast, FPGAs do not offer such mechanisms by default. If required for a certain use-case, they must be built into the FPGA design just like other parts of the application logic.

Consequently, resource management facilities in FPGAs are specific to a particular design, and might change when the FPGA is reconfigured. Furthermore, the correctness of their implementation depends on application developers and thus on a wider and less specialized circle than with fixed function hardware. With fixed function hardware, resource management mechanisms are usually standardized, implemented by small and specialized teams of engineers and under the scrutiny of large user groups.

In addition, advanced resource management mechanisms must contend with the actual application logic for a limited amount of FPGA resources. Thus the cost of design features required to share the FPGA can even defeat the expected utilization and throughput improvements by limiting the application logic performance.

At the current state of the art, the combination of these factors results in FPGA accelerators being usually operated in a single-tenant model. This limits the scalability of FPGA solutions in large scale cloud computing scenarios and also sacrifices utilization of valuable accelerator resources.

1.1 Context

Multi-Tenancy The term multi-tenancy originates from the domain of cloud computing and describes a Software-as-a-Service product, where multiple customers share access to a single application instance [2, 14]. In contrast to the more heavyweight approach of providing isolated application instances per customer, multi-tenancy can acheive better utilization and resource sharing. However, without further provisions, multi-tenancy enables interactions between tenants, which might be undesirable. In this context, Tsai et. al. [24, Sec. 3.1] identified three major design criteria:

- Resource sharing.
- Isolation.
- Customization.

In a more general sense, multi-tenancy can be applied outside the original SaaS context: In this report I will use the term whenever a resource (e.g. application, execution environment, hardware unit) is shared between multiple independent tenants, without being partitioned according to each tenant. Thus, multi-tenancy on a hardware level can encompass running multiple threads on the same CPU core using simultaneous multithreading, where each thread is considered a tenant. Similarly, on the operating system level, most systems support multi-tenancy through the concurrent execution of multiple independent processes on the same machine.

This relates multi-tenancy to the concept of virtualization, which according to Singh [21] means the division of the resources of a single machine into multiple execution environments. In this sense it might apply to an operating system, which offers each process its own execution environment. Nevertheless, the current use of the term usually implies an additional layer between the hardware and an operating system, running multiple virtual machines with individual operating systems on a single hardware platform. In both meanings, virtualization can be considered a technique to realize multi-tenancy. However, an application would not be considered multi-tenant just because it serves multiple tenants through individual instances running virtualized on the same machine.

Accelerators Accelerators are a crucial component in heterogeneous system architectures: In contrast to conventional general-purpose compute resources like CPUs, special-purpose accelerators are tailored to execute only one particular workload class efficiently. This limited applicability enables workload-specific optimizations, so that the accelerator can outperform general-purpose CPUs in terms of throughput, latency, or energy efficiency.

Usually, accelerators operate within a host system, which also features general-purpose compute resources and an operating system managing the entire platform. Processes can access accelerator functionality by attaching the corresponding accelerator device. In most cases, this operation only enables the process to control the accelerator, but with coherently attached accelerators, it also gives the accelerator access to the processes address space [23].

The most widely deployed class of accelerators are GPUs, which are tuned for floating-point intensive computations with limited control flow complexity. Besides computer graphics, this characteristic is useful for many scientific computing and machine learning applications. Recently, with the significant increase in the volume of machine learning workloads, even more specific accelerators like TPUs [7, Ch. 7.4] have been designed to match their finer characteristics.

This dynamic reveals a central issue with accelerator development in contrast to general-purpose hardware: Due to very high hardware development costs, an accelerator design must be general enough to match a minimum volume of workloads, in order to make its development economical. FPGA accelerators can offer a solution to this dilemma. Though being themselves general-purpose compenents benefitting from large production volumes, FPGAs can be configured to realize a custom accelerator design. They consist of a configurable fabric with logic primitives and an interconnect, and can thus implement any digital circuit within the fabric capacity. FPGA accelerators can be reconfigured at runtime with latencies in the low seconds to milliseconds range, enabling the accelerator to adapt to changing workload profiles.

1.2 Concept

Multi-tenancy in an FPGA accelerator means, that it can be attached to multiple independent processes and process their computation tasks simultaneously. This gives rise to analogous design requirements as for the SaaS interpretation of multi-tenancy:

Resource Sharing Tasks from multiple tenants must be handled by the FPGA simultaneously. Nevertheless there could be an upper limit to the amount of simultaneously processed tasks, depending on the overhead introduced by each single task. Most accelerated workloads involve waiting for FPGA-external resources besides computation, including accesses to various memory tiers or communication links. Therefore, it is usually efficient to implement true sharing of FPGA fabric resources instead of just partitioning them between tenants. This however requires maintaining the association between tenants and their data items while they are being processed, which constitutes an overhead with respect to a statically partitioned architecture.

Isolation Tasks from different tenants must not be able to interfere with each other. There is a wide variety of finer interpretations for this requirement. The simplest take on the point is that no tenant should be able to affect the correctness of the computations performed for a different tenant. A stricter version, which is also commonly

employed between operating system processes, requires that a tenant can neither influence the computation results of another tenant nor gain any information about it.

In addition, some degree of performance isolation might be desirable. In the extreme case this means that the execution performance for any single tenant should be completely independent from the activity of other tenants. This extreme interpretation is not practical due to the finite nature of compute resources and the difficulty of predicting task behavior accurately. More relaxed versions of the performance isolation criterion like fair resource allocation between tenants are of practical importance.

Customization The execution environment should be adaptible to the specific requirements of each tenant. Offering tenants the full configurability of the FPGA fabric, though possible by means of partial reconfiguration [12], would either involve a static partitioning between tenants including the associated suboptimal fabric utilization, or else incur significant task transition overheads [16, Sec. 5.1]. In most use cases, the workload characteristics of each tenant are closely related, resulting in a much smaller configuration space. This can be accomodated within a single fixed FPGA design by providing a set of runtime configuration options. Nevertheless, this set of parameters must be maintained per tenant and applied to each compute facility with each transition between tenants. Depending on the nature of the compute facilities and size of the configuration space, this might involve significant overheads, opening an interesting field of tradeoffs between configurability and efficiency.

A realization of the aforementioned requirements can not be achieved within an FPGA design alone. Though the design could make suitable provisions for multi-tenancy, it requires support from the host system to allow multiple processes to attach the FPGA accelerator. In the most convenient case, the host operating system tracks all attached processes and automatically adds a trustworthy process identification to every accelerator interaction. Alternatively, if the operating system does not support this feature or if only a single process can attach the accelerator exclusively at any time, a host-side middleware layer is necessary. A special daemon process can bind the accelerator on behalf of multiple actual tenants, forward their interactions and perform the role originally ascribed to the operating system. Though offering more design flexibility, this approach adds a level of indirection, which might cause a significant performance bottleneck.

Furthermore, the FPGA-side implementation of multi-tenancy offers several choices. The simplest case would be a monolithic FPGA design, which includes custom resource abstraction mechanisms. This approach can be very efficient in terms of FPGA fabric utilization, as only a minimum set of mechanisms can be included as required by the particular application. On the other hand, the correctness of the implementation and thus the potential for isolation breaches depends on an application developer. For this reason and also to reduce development costs, most practical designs rely on the re-use of usually expert-designed IP-blocks [25]. This approach, comparable to using a library in the software context, supports application developers by

offering predefined resource abstraction mechanisms they can employ in a custom design. Nevertheless, it does not guarantee, that all shared resources are exclusively accessed through such abstraction mechanisms, as application developers might still access some resources directly. FPGA overlays [10] are an answer to this problem, as they wrap all available resources in a global abstraction layer, so that any access by the application design necessarily goes through the overlay. Similar to using a framework in the software context, an overlay can assure the correct use of the shared resources and also provide a more convenient interface for application developers. Nevertheless, overlays still leave some potential for isolation breaches. The application design usually maintains some internal state, which can not be controlled by the overlay and thus might leak between tasks of different tenants. Consequently, a suitable overlay architecture can significantly reduce the development effort for a robust multi-tenant FPGA accelerator design, but any firm guarantees still depend on application developers.

2 Approach

I will base a concrete multi-tenancy implementation on the currently single-tenant Metal FS framework. It is a research prototype we have developed at our group [20].

2.1 Metal FS

The Metal FS framework is a hardware-software codesign, incorporating an FPGA overlay architecture as well as a middleware layer for the host system.

Metal FS is built on top of the CAPI SNAP framework [26], which is maintained by the OpenPOWER foundation. It wraps the coherent host memory access and control mechanisms provided by the underlying CAPI interface into a set of industry standard AXI interfaces, which offers sufficient abstraction to ensure easy portability to other emerging coherent accelerator interconnect technologies.

Metal FS realizes a dataflow model [15], where computations are orchestrated by the flow of data between operations. Logically, a Metal FS task is a stream of data elements, that passes through a pipeline of operators. This is a coarse-grained interpretation of the dataflow model, because operators are not limited to primitive, single cycle operations, but can realize complex calculations. An operator is an application specific hardware module with a single input and a single output stream, and operates within the Metal FS overlay.

The Metal FS overlay provides the data stream infrastructure, including a runtime-configurable stream routing mechanism between operators, as well as DMA facilities to read or write data streams to memory buffers.

Figure 1 illustrates the architecture of the Metal FS overlay. The central Stream Switch implements the routing mechanism and is configured via control registers accessible to a controlling host process. In addition, there is at least one built-in Host Memory DMA operator, to supply a data stream from reading successive host mem-

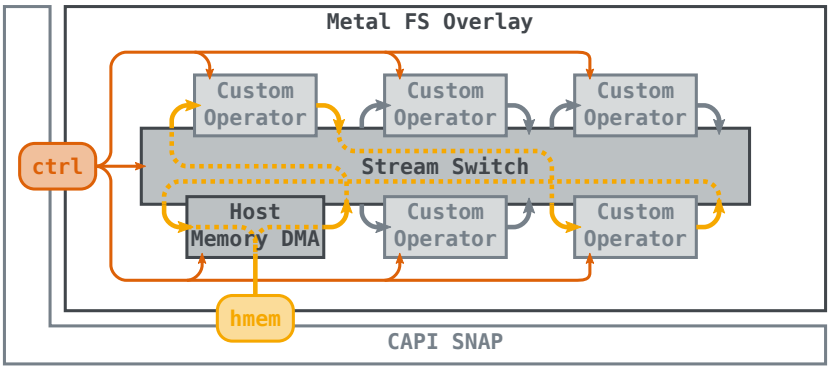

Figure 1: Metal FS overlay architecture. `hmem` and `ctrl` represent the CAPI SNAP host memory and control register interfaces respectively. An example pipeline configuration with two operators is shown.

ory locations or terminate a stream by writing it back to host memory. Each operator can expose custom configuration parameters through the control register interface.

The host side middleware offers different levels of access to the accelerator design. The foundation is formed by a library, which encapsulates the discovery of available operators, configuration of operator pipelines, and the initiation of data stream transformations. It can be used by a single application to interact with the accelerator on a high level of abstraction.

Older versions of the CAPI interface do not support attaching the same accelerator design to multiple processes. Therefore Metal FS offers a daemon process, which binds the FPGA card and presents the available operators in a virtual filesystem. Users on the host system can interact with the accelerator by piping data through the placeholder binaries in the Metal FS filesystem.

Currently, both the overlay and the host software support only a single active pipeline configuration at any given time. If multiple commandline users initiate a Metal FS task independently, these tasks are serialized by the host software.

2.2 Design Space

Multi-tenancy is a desirable feature for the Metal FS framework. Depending on the application domain, the complete set of operators deployed in a Metal FS design is likely to be larger than the set of active operators in an average pipeline configuration. Consequently, a higher utilization can be achieved by allowing multiple concurrently active tasks. Each task gets a customization space including the number and order of operators deployed in the pipeline, as well as the configuration parameters offered by each operator. Isolation between tasks means that no task can access items from another task's data stream, as well as configuration parameters and operator state specific to another task.

These requirements open wide design space of valid implementations and different tradeoffs between performance and complexity.

For certain applications, a valid solution can be implemented within the host software alone: Many operators like filters or functional transformations do not maintain their state across the entire data stream, but only over a fixed logical unit within that stream. If the entire application design is comprised of such operators, the host software can implement a time multiplex scheme between tasks without support from the FPGA design. The data streams beloning to each task are divided into blocks and processing of blocks from different tasks can be interleaved. The host software must take care to apply the correct pipeline and operator configuration to the Metal FS design whenever a new block is initiated. Though simple to implement, this scheme is severely limited to stateless operators and can hardly qualify as true multi-tenancy, as there is never any simultaneous activity from multiple tenants.

A slightly more advanced approach offers true resource sharing on the overlay but not operator level: If the built-in DMA mechanism of the Metal FS overlay is expanded to handle multiple independent data streams, tasks requiring non-overlapping sets of operators can be scheduled simultaneously. As each task is processed to completion, this scheme supports stateful operators. The efficiency of this solution depends however on the specific operator composition of the FPGA design as well as the average pipeline structures. To ensure a minimum operator overlap between tasks, it might be necessary to include additional copies of frequently used operators, which is limited by FPGA fabric constraints.

Especially complex, stateful operators, which possibly access external resources do not achieve complete utilization by processing a single data stream. This additional utilization potential could not be realized by the previously discussed resource sharing scheme. Instead, the operators themselves must support multi-tenancy, whereas with the previous approach such issues would be handled within the overlay and host software, invisible to application developers.

A multi-tenancy aware operator could simply support saving and restoring its current processing state to implement a context switch. In this way, the overlay gains a significant amount of scheduling flexibility. Combined with a stream buffer mechanism, different operator characteristics in terms of throughput and latency can be balanced. For example, an operator waiting for data on its input stream could be temporarily reassigned to a task where data is already available. This scheme still places much of the scheduling complexity in the overlay, whereas application developers only need to implement the operator context transfer.

In contrast, the most flexibility but also highest complexity from an application developer's point of view would be operators with native support for multiple data streams without explicit context switches. In this way, operators can interleave their internal operations between different streams and thus balance out single-stream idle times. The overlay retains the role of injecting and and forwarding data from different tasks as required by each operator. This functionality as well as the associated flow control and bandwidth distribution mechanisms belong to the domain of Network on Chip design. Considering the limited resources of the FPGA fabric, this considerable increase overlay complexity might defeat the benefits from an increased

operator utilization.

Each of the three outlined multi-tenancy schemes offers a unique tradeoff between performance and design complexity, and may prove superior for a specific application. Even though I am interested in exploring as wide a design space as possibly to determine suitable application characteristics and implementation costs empirically, my primary focus will be on approaches which minimize the implementation complexity for application developers. Besides the potential for a more widespread applicability, such approaches have the advantage of leaving much of the isolation responsibility with the overlay.

3 Related Work

Applications of FPGA accelerators in a cloud computing environment can be found as early as 2014 with the Catapult project by Putnam et. al. [18]. However, they focus on implementing a scalable multi FPGA architecture for a single application on a cluster system. This use-case allows sufficient assumptions about individual task behavior, that no external mechanisms for multi-tenant isolation are required. With the introduction of Amazon EC2 F1 instances in 2017, FPGA accelerators became available in a multi-tenant environment, even though for security reasons the FPGAs themselves remain dedicated to a single instance each [19, Sec. 2.A].

Nevertheless, the overall trend sparked much recent interest in the multi-tenant operation of FPGAs. Authors like Ramesh et. al. [19], Mahmoud et. al. [17], Krautter et. al. [13] and Elnaggar et. al. [5] envision a multi-tenant architecture, where each tenant gains complete control over a partition of the FPGA via partial reconfiguration. They focus on the severe security threat, an arbitrary FPGA design can impose on co-located FPGA applications via side-channel attacks. This is valuable evidence against the partial-reconfiguration based multi-tenancy approach. It also underlines, that the isolation guarantees of a multi-tenant FPGA design can only be determined from the complete design, as any user defined part could exploit a side-channel vulnerability to break isolation.

Studies of multi-tenant architectures that do not rely on partial reconfiguration are less frequent. István et. al. [9] are an example, which matches the concept outlined in section 1.2 very well. They describe an FPGA implementation of a Key-Value Store, which can process requests from multiple tenants. The design is not partitioned and replicated according to a predefined set of tenants, but can handle multiple tenants within the same application logic. Another interesting point is the use of mechanisms like request queueing and token-bucket flow control to realize performance isolation between tenants. Even though the work describes architectural choices that are relevant to other applications, it does not provide a generalized development framework or overlay architecture.

Apart from the multi-tenant application scenario, there has been a longstanding interest in FPGA operating systems and the application of established software re-

source management mechanisms to a hardware context. Such mechanisms can be employed to realize multi-tenancy capable FPGA designs.

There is a considerable variety in the conception of an FPGA operating system. Earlier approaches by Danne [4], So et. al. [22] and Jozwik et. al. [11] take the operating system role more literally, implementing mechanisms to schedule hardware tasks and providing classic software operating system facilities like virtual memory. Some approaches integrate with and rely on the support of a host operating system, while others are designed for standalone operation. The actual deployment of a hardware task is usually realized through partial reconfiguration, so the FPGA operating system makes no assumptions about the internal architecture a task.

Later designs like the M3 [1] and Leap [6] operating systems take a more abstract view and focus on providing application designers with a stable and convenient runtime environment. They support application developers by providing a framework of communication and memory primitives. This conception matches the spirit of Metal FS more closely.

Finally, multi-tenant operation of other accelerator platforms like GPUs has also recently come into focus. There are some proposals of GPU virtualization techniques, for example by Xiao et. al. [27] or Iserte et. al. [8]. These are based on sharing local GPU resources with remote nodes in a cluster setup. This is usually achieved by forwarding of API calls via network to the destination node. Even though they offer a level of GPU virtualization, these approaches do not match the requirements for multi-tenancy. GPUs themselves have long been unable to guarantee isolation between concurrent tasks and the aforementioned approaches assume all nodes which share GPU resources to be controlled by a single tenant.

However, recent developments in GPU architecture are beginning to offer firm isolation guarantees between multiple tenants. The Volta generation of NVidia GPUs supports the Multi-Process Service [3, Sec. 1.1.1], a runtime environment allowing multiple processes from different tenants to share a single GPU.

References

[1] N. Asmussen, M. Völp, B. Noethen, H. Härtig, and G. Fettweis. “M3: A Hardware / Operating-System Co-Design to Tame Heterogeneous Manycores”. In: *ACM SIGPLAN Notices*. Volume 51. 4. ACM. 2016, pages 189–203.

[2] C.-P. Bezemer, A. Zaidman, B. Platzbeecker, T. Hurkmans, et al. “Enabling multi-tenancy: An industrial experience report”. In: *IEEE International Conference on Software Maintenance*. IEEE. 2010, pages 1–8.

[3] N. Corporation. *Multi-Process Service*. June 2020.

[4] K. Danne. “Operating systems for fpga based computers and their memory management”. In: *ARCS Organic and Pervasive Computing, Workshop Proceedings*. Volume 41. 2004, page 10.

[5] R. Elnaggar, R. Karri, and K. Chakrabarty. "Multi-tenant FPGA-based reconfigurable systems: Attacks and defenses". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2019, pages 7–12.

[6] K. Fleming and M. Adler. "The LEAP FPGA Operating System". In: *FPGAs for Software Programmers*. Springer, 2016, pages 245–258.

[7] J. L. Hennessy and D. A. Patterson. *Computer Architecture - A Quantitative Approach*. 6th edition. Morgan Kaufmann, 2018. ISBN: 978-0-12-811905-1.

[8] S. Iserte, F. J. Clemente-Castelló, A. Castelló, R. Mayo, and E. S. Quintana-Ortí. "Enabling GPU Virtualization in Cloud Environments." In: *CLOSER (2)*. 2016, pages 249–256.

[9] Z. István, G. Alonso, and A. Singla. "Providing multi-tenant services with FPGAs: Case study on a key-value store". In: *28th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE. 2018, pages 119–1195.

[10] A. K. Jain. "Architecture centric coarse-grained FPGA overlays". PhD thesis. 2017.

[11] K. Jozwik, S. Honda, M. Edahiro, H. Tomiyama, and H. Takada. "Rainbow: An Operating System for Software-Hardware Multitasking on Dynamically Partially Reconfigurable FPGAs". In: *International Journal of Reconfigurable Computing* 2013 (2013), page 5.

[12] D. Koch. *Partial Reconfiguration on FPGAs: Architectures, Tools and Applications*. Volume 153. Springer Science & Business Media, 2012.

[13] J. Krautter, D. R. Gnad, and M. B. Tahoori. "Mitigating electrical-level attacks towards secure multi-tenant FPGAs in the cloud". In: *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 12.3 (2019), pages 1–26.

[14] R. Krebs, C. Momm, and S. Kounev. "Architectural Concerns in Multi-tenant SaaS Applications." In: *Closer* 12 (2012), pages 426–431.

[15] E. Lee and T. Parks. "Dataflow process networks". In: *Proceedings of the IEEE* 83.5 (1995), pages 773–801.

[16] M. Liu, W. Kuehn, Z. Lu, and A. Jantsch. "Run-time partial reconfiguration speed investigation and architectural design space exploration". In: *International Conference on Field Programmable Logic and Applications*. IEEE. 2009, pages 498–502.

[17] D. Mahmoud and M. Stojilović. "Timing violation induced faults in multi-tenant FPGAs". In: *Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE. 2019, pages 1745–1750.

[18] A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, et al. "A reconfigurable fabric for accelerating large-scale datacenter services". In: *ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. IEEE. 2014, pages 13–24.

[19] C. Ramesh, S. B. Patil, S. N. Dhanuskodi, G. Provelengios, S. Pillement, D. Holcomb, and R. Tessier. "FPGA side channel attacks without physical access". In: *IEEE 26th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE. 2018, pages 45–52.

[20] R. Schmid, M. Plauth, L. Wenzel, F. Eberhardt, and A. Polze. "Orchestrating Near-Data FPGA Accelerators Using Unix Pipes". In: *Seventh International Symposium on Computing and Networking Workshops (CANDARW)*. IEEE. 2019, pages 125–128.

[21] A. Singh. "An Introduction to Virtualization". In: *kernelthread.com* (Jan. 2004).

[22] H. K.-H. So and R. W. Brodersen. *BORPH: An operating system for FPGA-based reconfigurable computers*. University of California, Berkeley, 2007.

[23] J. Stuecheli, B. Blaner, C. Johns, and M. Siegel. "CAPI: A coherent accelerator processor interface". In: *IBM Journal of Research and Development* 59.1 (2015), pages 7–1.

[24] W.-T. Tsai, X. Sun, and J. Balasooriya. "Service-oriented cloud computing architecture". In: *Seventh international conference on information technology: new generations*. IEEE. 2010, pages 684–689.

[25] *Vivado Design Suite User Guide - Designing IP Subsystems Using IP Integrator*. 2018.3. UG994. Xilinx, Inc. 2018.

[26] L. Wenzel, R. Schmid, B. Martin, M. Plauth, F. Eberhardt, and A. Polze. "Getting Started with CAPI SNAP: Hardware Development for Software Engineers". In: *European Conference on Parallel Processing*. Springer. 2018, pages 187–198.

[27] S. Xiao, P. Balaji, Q. Zhu, R. Thakur, S. Coghlan, H. Lin, G. Wen, J. Hong, and W.-c. Feng. "VOCL: An optimized environment for transparent virtualization of graphics processing units". In: *Innovative Parallel Computing (InPar)*. IEEE. 2012, pages 1–12.

fastForce: Real-Time Reinforcement of Laser-Cut Structures

Muhammad Abdullah

Human Computer Interaction Group
Hasso Plattner Institute for Digital Engineering
muhammad.abdullah@hpi.de

We present fastForce, a software tool that detects structural flaws in laser cut 3D models and fixes them by introducing additional plates into the model, thereby making models up to 52x stronger. By focusing on a specific type of structural issue, i.e., poorly connected sub-structures in closed box structures, fastForce achieves real-time performance (10^6x faster than finite element analysis). This allows fastForce to fix structural issues continuously in the background, while users stay focused on editing their models and without ever becoming aware of any structural issues. In our survey we found 286 of 402 relevant models in the kyub [1] model library to contain such flaws. We integrated fastForce into a 3D editor for lasercutting (kyub) and found that even with high plate counts fastForce achieves real-time performance.

1 Introduction

Recently, researchers presented software systems that help users design 3D models for lasercutting, such as FlatFitFab [7], which helps users create interlocking cross sections. Kyub [1] builds on ideas from FlatFitFab and extends it towards structures that withstand 1000x larger loads. Kyub achieves this by producing "closed box structures" held together by box joints (finger joints if not perpendicular). As a result, kyub allows users to make chairs that people can sit on, even from comparably thin building materials. Unfortunately, as the kyub authors acknowledge, in order to allow them to take such high loads, some models require "reinforcement", i.e., they require additional internal plates to be added to the model. We found 286 of 402 relevant user generated models in the kyub repository to contain such flaws (see survey). Since users cannot figure out the need for reinforcement, reinforcement needs to be created automatically.

At first glance, such reinforcement functionality could be integrated into the user's workflow as a separate step after the design workflow. However, it is generally understood that such a design-first-then-engineer workflow leads to suboptimal results.

In this paper, we address these issues by generating reinforcement (1) automatically and (2) in real-time. While we started by exploring the traditional approach of finite element analysis, we found it to be 6 orders of magnitude too slow for interactive use. Thus we have developed a custom algorithm that finds and fixes points of failure in real-time by formulating them as a graph connectivity problem. As shown in Figure 1b and c, our software tool fastForce runs in the background while users

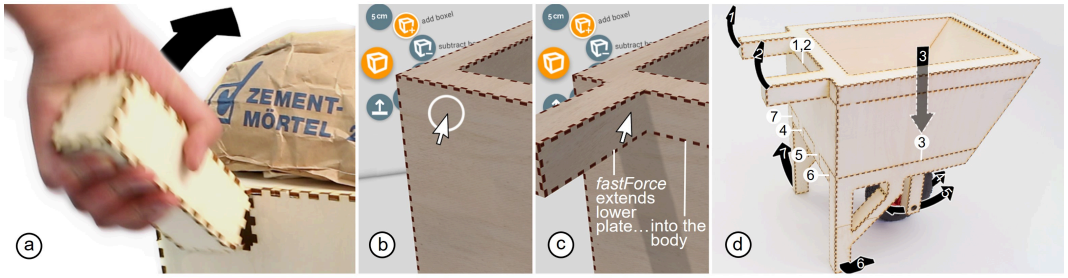

Figure 1: (a) The handle of this laser-cut wheelbarrow breaks, when the user is trying to lift a heavy load. (b) Our software tool fastForce addresses this. The moment the user clicks in the editor to create the handle, (c) not only the handle appears, but fastForce also detects the resulting point of failure and reinforces it in real-time (d) fastForce detects additional points of failure (black arrows 2-7) and generates corresponding reinforcement (white 2-7).

are editing their 3D model (here in kyub [ref]). The moment the user starts modeling, here adding a handle for the wheelbarrow, fastForce simultaneously adds the necessary reinforcement, here by extending the handle's lower plate into the interior of the wheelbarrow. (d) Since fastForce only modifies the wheelbarrow's interior, the outer shape of the model is preserved while still allowing to lift a load that is 45.6 times higher. fastForce generates reinforcement for the other six points of failure similarly (here labeled 2-7).

2 Understanding the problem space

Closer inspection revealed that this category of "critical" failures originate when laser-cut closed box structures have deteriorated into poorly connected sub-structures. Figure 2 illustrates this. Consider the second case labeled "E1"; it represents the wheelbarrow's handle. It breaks because only one of its plates is rigidly connected to the rest of the model; the other three plates, in contrast, are connected by means of finger joints only. We will refer to these as concave finger joints. When subjected to tension, the fingers are pulled out, resulting in the single remaining plate supporting the entire load. This causes it to break even under very small loads (here as small as 9.8N).

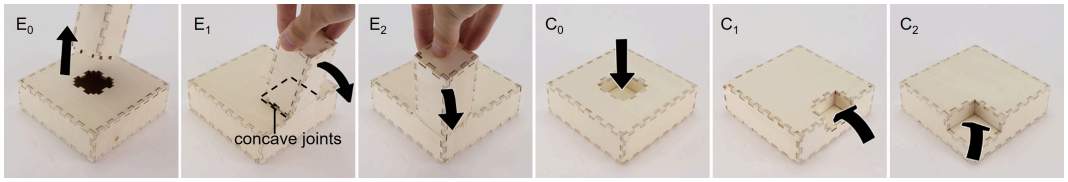

Figure 2: The six main types of structural issues of closed box structures.

Similar points of failure arise almost anywhere where two sub-structures of different sizes connect, or where sub structures are offset with respect to each other and Figure 2 classifies the six main resulting cases, which we label by E for extrusions and C for cavities and then index by the number of remaining connecting plates.

Based on this list of cases to search for, we propose an algorithm that would focus on detecting only these points of failure. This turned out to be the key to solving the real-time requirement, as it allowed us to reformulate the analysis as a graph connectivity problem making it computationally tractable.

3 The Building Blocks of Reinforcement

We have manually devised reinforcement for the six points of failure cases, we based our principles on structural engineering, particularly on beam design [6].

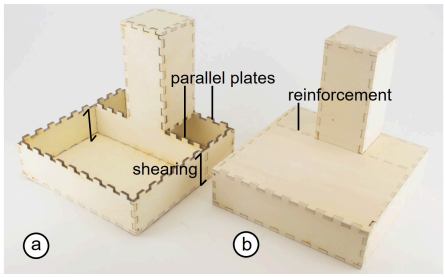

Figure 3: (a) A reliable connection is formed by at least two parallel plates extending internally and hooking into the sides. (b) The outer hull of the object remains the same.

As illustrated by Figure 3 at the example of the E1 failure type, all types of reinforcements we designed create a reliable connection between the sub-structures using at least two parallel through plates. As shown, the reinforcement plates extend internally and hook into the sides while respecting the hull/envelope of the model. By hooking into the sides the reinforcement plates allow finger joints to be loaded on compression and shearing (which they are strong against), counteracting tension force.

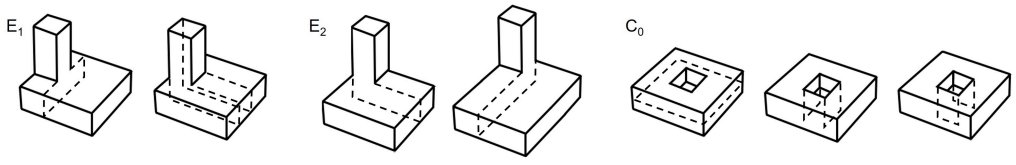

Figure 4: Some of the manually designed sets of reinforcement options.

Figure 4 shows some of the reinforcement options we generated for the six individual failure cases.

4 The fastForce Algorithm

We now present the fastForce algorithm. fastForce detects all points of failure in the model, places appropriate reinforcement automatically and in real-time, and fine-tunes the reinforcement so as to guarantee assemblability.

fastForce achieves real-time performance by looking for weakly connected components in a graph representation of the model which point to weakly connected sub-structures. Since the graph structure is at a higher level of abstraction than 3D geometry, a model that would become highly complex if represented in FEA continues to be represented by a small number of nodes and edges.

Step 1: fastForce identifies critical points of failure fastForce starts by locating points of failure based on a simplified graph representation, here the wheelbarrow model from Figure 1. The undirected graph represents each 2D plate in the model as a node, while finger joints are represented as edges between nodes. Figure 5a the graph distinguishes between normal and concave finger joints (red edges), as they can be subjected to tension, causing potential failure. fastForce simulates their failure by removing all concave joints (i.e., all red edges) from the graph. As shown in Figure 10a and b this causes the graph to fall apart into four disconnected subgraphs. This corresponds to E0 (or C0) type points of failure i.e. these sub-structures were only connected by concave finger joints. This means that the corresponding physical model, the wheelbarrow, will also disconnect into four parts when subjected to minor forces.

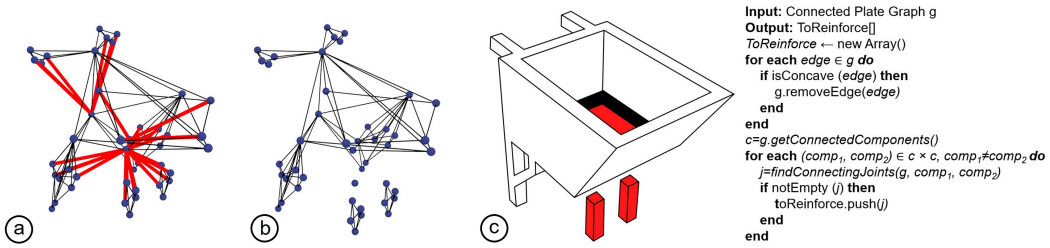

Figure 5: (a) When removing the red edges representing concave joints (b) this graph decomposes into four subgraphs, suggesting that (c) this model of a wheelbarrow will disconnect into four parts based on E0 (or C0) points of failure.

E1 (or C1) points of failure occur when sub-structures are only connected by a single plate. As illustrated by Figure 6, removing the red edges created three subgraphs connected by a single node. fastForce locates the problem by performing a one-node

cut on all the disconnected subgraphs. This results in a set of single nodes, which, if removed, will disconnect the subgraphs.

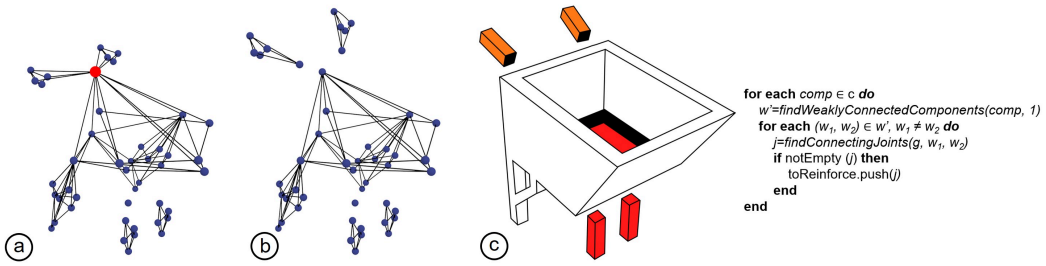

Figure 6: (a) When removing isolated nodes that connect subgraphs (b) this model graph decomposes into disconnected subgraphs, suggesting (c) that these parts are subject to E1 (or C1) points of failure.

Figure 7 shows that fastForce identifies E2 (or C2) points of failure by performing a two-node cut on the remaining subgraphs. This means that the back legs of the wheelbarrow are only held in place by two plates each. Specifically, for E2 (or C2), fastForce checks if these two plates are actually connected to each other, which forms the basis of the corner connection. If they are not connected and are parallel this already fulfills our notion of a reliable connection and will not benefit from further reinforcement.

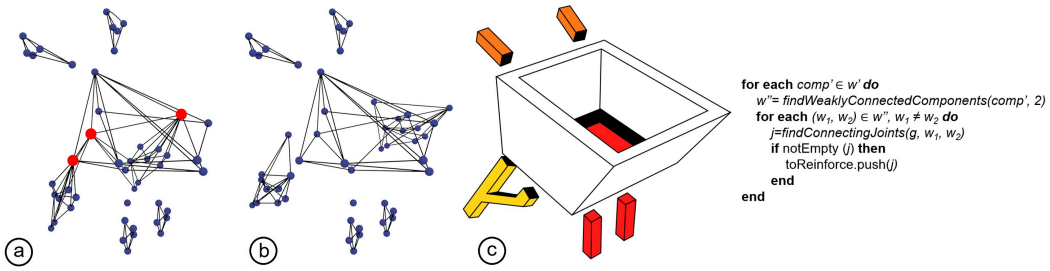

Figure 7: (a) When removing pairs of nodes that connect subgraphs (b) this model graph decomposes further, suggesting (c) that these parts are subject to E2 (or C2) points of potential failure.

The algorithm has a linear complexity of O(n+e), where n are the number of nodes (plates) and e are the number of edges (joints). Node-cuts are based on the Hopcroft Tarjan Algorithm [4].

Step 2: fastForce adds reinforcement fastForce adds reinforcement sequentially starting with the weakest points of failure E0 (and C0). After selecting a point of failure, fastForce identifies the set of plates that can be extended for reinforcement.

fastForce achieves this by revisiting the removed graph edges (concave joints) that originally belonged to the disconnected subgraph. These edges connect to nodes, which point to their respective plates represented as polygons at the mesh level. These plates connect the two sub-structures together and are extended to reinforce them.

Figure 8 show the process of reinforcing the "tray" of the wheelbarrow. (a) After choosing a plate to extend, a slice plane is generated which intersects with the surrounding geometry. (b) This forms a closed intersection path which is used to create an extension to the original plate. This new reinforcement plate is added to the mesh. (c) fastForce creates joints, where the extended plate intersects with other plates. (d) fastForce then repeats this process for each point of failure identified in the first part of the algorithm.

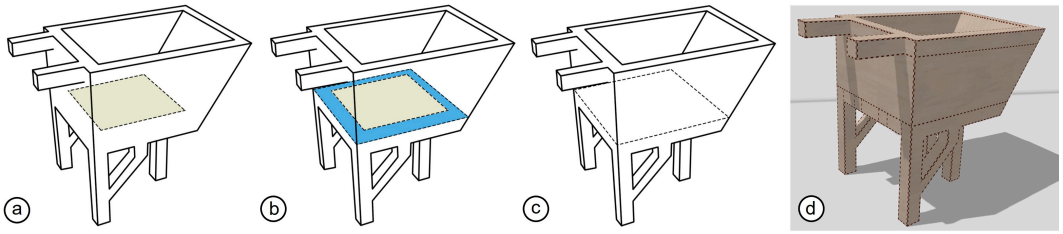

Figure 8: (a) fastForce reinforces the "tray" by (b) extending the chosen plate using a slice plane. (c) The plate is extended and joints are created where it intersects the other plates. (d) fastForce, sequentially reinforces the remaining points of failure.

5 Survey: User created content is subject to points of failure

We collected user generated models from an online repository hosted by kyub, some of which are shown in Figure 9. The color corresponds to the type and severity of weakness with red being the most severe.

We started out with 1067 models. Eliminating structurally trivial models (prisms), left us with 402 models that we analyzed. Out of these 402 models, 286 models were subject to one or more critical point of failure and would benefit from fastForce's reinforcement.

Out of the remaining 116 models, 88 already had a reliable connection with more than two plates connecting the sub structures. While 28 models either required living hinges to be extended for reinforcement or cutouts/decorative elements were present in the locations where the reinforcements would hook in.

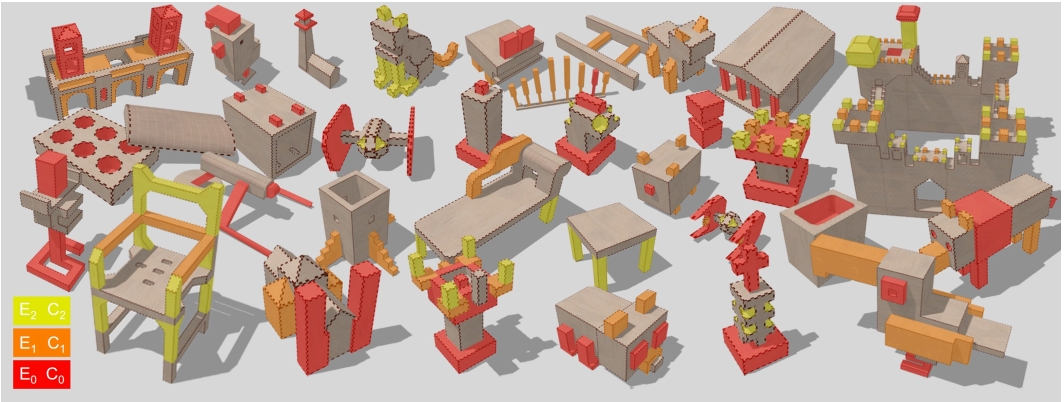

Figure 9: A selection of user generated models (kyub.com) that would benefit from reinforcements generated by fastForce.

6 Integration and performance evaluation

We integrated fastForce into a 3D editor for lasercutting (kyub [1]), allowing fastForce to be used interactively. Figure 10 illustrates the system diagram of kyub. The key contribution is that there is no fastForce-specific user interface. Reinforcement shows up without any specific user interaction.

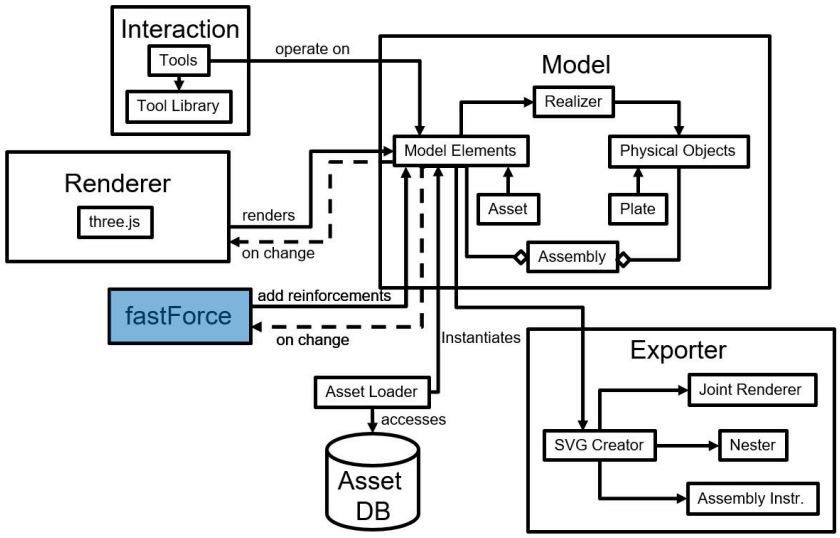

Figure 10: fastForce is called automatically whenever the user edits the model.

We benchmarked the runtime performance of the kyub 3D editor after integrating fastForce. The editor was running on a lightweight laptop computer (X1 Carbon, 4-core machine, 10 GB of ram). Eight different models representing different plate counts were used. We simulated user interaction by performing a basic kyub com-

mand (push/pull) to modify the model and measured the response time. The results are summarized in table 1.

Table 1: Response times for different models.

Model	**Number of plates**	**Response time (ms)**
Wheelbarrow (Figure 1)	30	10
Chair (Figure 9: bottom)	46	21
Cathedral	52	11
Chess Queen	68	68
Bear	72	34
Chess King	86	44
Toucan	95	47

The results show that even for models with high plate counts integrating fastForce still allows good response times, allowing the user to smoothly use the editor.

7 Validation: Testing structural strength

To evaluate the effect reinforcement created by fastForce, we tested the structural strength of 6 models (shown in Figure 11) after reinforcing them using fastForce.

We manufactured all models from standard 4mm 3-layer poplar plywood. We then used the custom rig with force sensor forceX 2.30 made from aluminum profiles. In case of the extrusion models, we applied a bending moment from the side to the top of the extrusion. For models with a cavity, we applied a force from above the cavity to push them inwards. In both cases this results in tension forces acting at the finger joints susceptible to tension.

Results Figure 11 shows the results. For the reinforced primitives for the six basic types of failures (E0-C2). While reinforcement made the reasonably connected substructures E2 and C2 even stronger by 3x, as expected reinforcement using fastForce showed its biggest effect on the C0 and E0 designs, where they led to improvements of 50x and 52x respectively.

8 Related works

Finding structural weaknesses Stress-relief [11] is a system that uses FEA to automatically fix structural problems in 3D models. Breuß et al. [2] use a skeletonization approach to enhance the stability of 3D printed structures and validate their results using FEM. Some scholars have utilized other techniques to circumvent the non-interactive rates of finite elements. Specifically for trusses, Markis et al. [5] present a

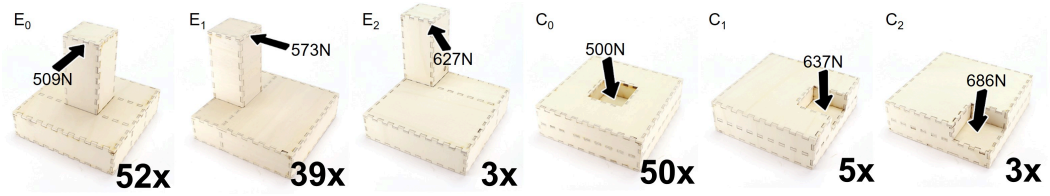

Figure 11: The reinforced test objects. Numbers indicate the factor of increase in strength after fastForce's reinforcement.

real-time algorithm that identifies structural weaknesses. SketchChair [9] for example achieves interactive rates by employing a ragdoll physics model running in the background. Umetani and Schmidt [12] leverage the Euler Bernoulli method to find critical weaknesses in models. Langlois et al.

(Dis)assembly in fabrication Fu et al. [3] developed interlocking furniture. To do this, they developed an algorithm that checks whether an object can be disassembled by taking parts out of the assembly piece by piece. Fabrication-aware Design with Intersecting Planar Pieces [10] leverages that method as well. Both approaches are similar in their logic (finding out how to gradually disassemble a model), which inspired our general method of finding points of potential failure as well.

Fabrication based on closed box structures In kyub [1], Baudisch et al. used the underlying strength of closed box structures to make sturdy laser-cut objects. The system uses plates connected together using finger joints to create self-supporting facades that can resists loads close to 4900N. Box like structures are also being introduced in 3D printing. In Boxception [8], Sajadi et al. propose 3D printing objects using box like cells to create impact resistance objects.

9 Conclusion and future work

We presented fastForce, a software tool designed to identify and remove structural weaknesses in laser-cut objects based on closed box structures. We presented an algorithm called fastForce that detects points of failure automatically and reinforces them in real-time. We integrated fastForce into the interactive 3D editor kyub, where fastForce's real-time ability allows it to run in the background without interfering with users' modeling activity. As future work, we plan on extending fastForce's capability of real-time reinforcement to non-closed box structures as well.

10 Publications

- **Muhammad Abdullah**, Martin Taraz, Yannis Kommana, Shohei Katakura, Robert Kovacs, Jotaro Shigeyama, Thijs Roumen, and Patrick Baudisch."fast-

Force: Real-time Reinforcement of Laser-Cut Structures." submitted to CHI 2021, under review.

- Thijs Roumen, Ingo Apel, Jotaro Shigeyama, **Muhammad Abdullah**, and Patrick Baudisch. "Kerf-Canceling Mechanisms: Making Laser-Cut Mechanisms Operate Across Different Laser Cutters." In Proceedings of the 33nd Annual ACM Symposium on User Interface Software and Technology. 2020.

References

[1] P. Baudisch, A. Silber, Y. Kommana, M. Gruner, L. Wall, K. Reuss, L. Heilman, R. Kovacs, D. Rechlitz, and T. Roumen. "Kyub: A 3D Editor for Modeling Sturdy Laser-Cut Objects". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pages 1–12.

[2] M. Breuß, J. Buhl, A. M. Yarahmadi, M. Bambach, and P. Peter. "A Simple Approach to Stiffness Enhancement of a Printable Shape by Hamilton-Jacobi Skeletonization". In: *Procedia Manufacturing* 47 (2020), pages 1190–1196.

[3] C.-W. Fu, P. Song, X. Yan, L. W. Yang, P. K. Jayaraman, and D. Cohen-Or. "Computational interlocking furniture assembly". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pages 1–11.

[4] C. Gutwenger and P. Mutzel. "A linear time implementation of SPQR-trees". In: *International Symposium on Graph Drawing*. Springer. 2000, pages 77–90.

[5] M. Makris, D. Gerber, A. Carlson, and D. Noble. "Informing Design through parametric integrated structural simulation: iterative structural feedback for design decision support of complex trusses". In: (2013).

[6] D. B. Marghitu. *Mechanical engineer's handbook*. Elsevier, 2001.

[7] J. McCrae, N. Umetani, and K. Singh. "FlatFitFab: interactive modeling with planar sections". In: *Proceedings of the 27th annual ACM symposium on User interface software and technology*. 2014, pages 13–22.

[8] S. M. Sajadi, P. S. Owuor, R. Vajtai, J. Lou, R. S. Ayyagari, C. S. Tiwary, and P. M. Ajayan. "Boxception: impact resistance structure using 3D printing". In: *Advanced Engineering Materials* 21.8 (2019).

[9] G. Saul, M. Lau, J. Mitani, and T. Igarashi. "SketchChair: an all-in-one chair design system for end users". In: *Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*. 2010, pages 73–80.

[10] Y. Schwartzburg and M. Pauly. "Fabrication-aware design with intersecting planar pieces". In: *Computer Graphics Forum*. Volume 32. 2pt3. Wiley Online Library. 2013, pages 317–326.

[11] O. Stava, J. Vanek, B. Benes, N. Carr, and R. Měch. "Stress relief: improving structural strength of 3D printable objects". In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), pages 1–11.

[12] N. Umetani and R. Schmidt. "Cross-sectional structural analysis for 3D printing optimization." In: *SIGGRAPH Asia Technical Briefs*. Citeseer. 2013.

Virtualizing Physical Space

Sebastian Marwecki

Human Computer Interaction Group
Hasso Plattner Institute for Digital Engineering
sebastian.marwecki@hpi.de

The topic of my dissertation is to allow arbitrary virtual reality experiences to be run in arbitrary tracking volumes and with arbitrary physical objects. To achieve this, I created an abstraction between VR applications and the space and physical objects they are using. Instead of accessing space and physical objects directly, applications express their needs in an abstract way, which my systems then maps to the actual available physical space and physical objects. This allows VR applications to run on a wide range of installations. This may have substantial commercial impact, as the proliferation of real walking VR is currently hindered by developers' reluctance to require users to have space and objects. I here summarize four select projects I have presented throughout the last years in the *HPI Research School of Service-oriented Systems Engineering* and provide further detail on their technical background.

1 Introduction

The topic of my dissertation is to create an operating system for virtual reality, in order to enable real walking.

Operating systems are a key component of modern computer systems. They allow applications to run on arbitrary computers and architectures by creating an abstraction of the physical hardware, an API, that prevents applications from accessing the hardware directly, which is what happens in VR today. VR applications directly access the physical hardware, which is space, and, in some cases physical objects and props contained within. This is a problem, because it prevents VR applications from running in any space that does not offer at least the required amount and shape of space.

I dissected this problem within various projects. *VirtualSpace* provides an abstraction to physical space, so that multiple users can co-use that space for real walking in VR. *Scenograph* adds to this so that users can enjoy more complex applications with multiple storylines across arbitrary tracking volumes. *Stuff-Haptics* extends this notion further to also include passive haptic props and room geometry. I worked on multiple other projects (e.g. *Mise-Unseen*) that extend the toolbox that solves the issue of limited space and props.

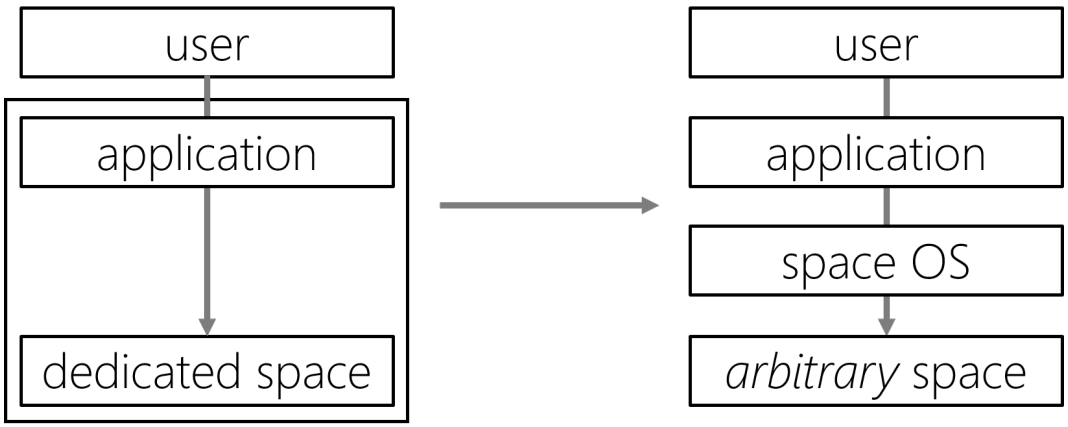

Figure 1: Applications for virtual reality access space directly, preventing them from running in arbitrary spaces. I present multiple systems to sketch out what an operating system for virtual reality should entail, to allow such applications to run anywhere and thus enable real walking in virtual reality.

I have presented reports in the *HPI Research School of Service-oriented Systems Engineering* that provide background to these projects. I here want to provide further technical background.

2 Virtual Space – Abstracting Space Away From Its Use

Virtual Space is a system that allows multiple real walking VR users to share the same physical space without being aware of each other. Virtual Space is designed to give each user the illusion of being in possession of the entire physical space.

Figure 2 shows an example. Two users share the same 4x4m physical space, while being tracked using a VR tracking system. Both users are in their own, separate virtual environments. (a) The green user is immersed in a badminton app, while (b) the blue user experiences a Pac-man game.

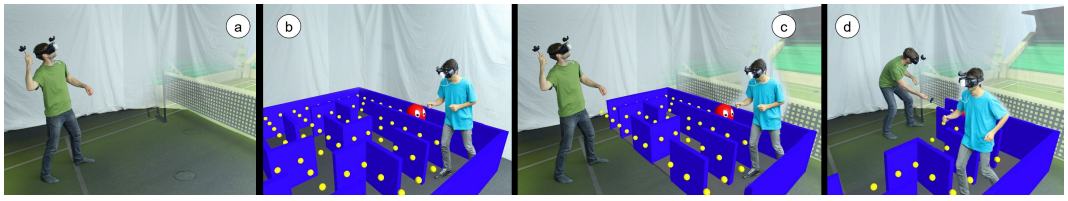

Figure 2: Multiple users share the whole physical space without needing to be aware of one another.

To keep users from running into each other, Virtual Space limits each app to non-overlapping tiles at any given time. Client apps handle this in a way transparent to their user. The badminton app, for example, always makes the user's virtual opponent return the ball to locations inside the tile currently assigned to the app. By reassigning tiles frequently, Virtual Space moves users across the entire space, thereby seemingly allowing for unrestricted walking.

The key point is that both apps are mapped to the entire physical space, i.e., Virtual Space allows each app to be designed under the assumption that the user has physical access to the entire 4x4m space. And that is true, albeit not necessarily at every particular moment, as Virtual Space limits each app to a different non-overlapping tile of space. Client apps handle this in a way transparent to their user. The badminton app, for example, makes the user's virtual opponent returns the ball always to locations inside the tile currently assigned to this app.

To allow users to still complete their story arc and to prevent users from noticing that the system is confining them, Virtual Space employs what we call maneuvers.

Apps achieve the ability to run in Virtual Space by implementing the Virtual Space API (Figure 3).

```
void SetProbabilityDistribution(
        List<Vector2> walkableArea,
        List<Vector2> preferredArea)

List<Valuation> ValuatePossibleManeuvers(
        List<Maneuver> maneuvers)

void AdaptToTimedManeuver(
         List<TimedManeuver> timedManeuvers)

TimedManeuver(Maneuver m, float delay, float
duration}

Maneuver{ Tile start, Tile end, List<Tile>
animation}

Tile{ List<Vector2> area}

Valuation{
        float weight,
        Maneuver maneuver,
        List<Tuple<float, float>> timeInfo}
```

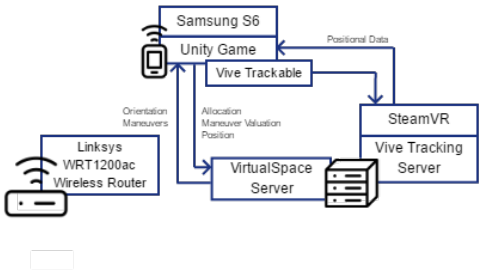

Figure 3: (left) Virtual Space API, (right) System Setup

The system decides which maneuver to invoke. It adds up the utility for each maneuver reported by each app and picks the maneuver that maximizes utility across apps. It also tries to avoid maneuvers subject to timing mismatches between apps. In addition, the system considers past maneuvers so as to ensure fairness over time: a focus maneuver for the first app, for example, implies three corresponding defocus maneuvers for the other users.

The system now informs apps about the upcoming maneuver. It does so by sending a sequence of what we call frames, i.e., a sequence of images that show the space distribution at a given moment in time. To keep tiles convex at all times, the system computes tiles as Voronoi tessellations.

Apps respond to maneuvers by placing rewards and obstacles inside their virtual environment, thus guiding the user to the next tile.

Figure 3(left) summarizes our algorithm in the form of a resulting API. For an app to participate in Virtual Space, it must implement this API. We have implemented four apps to run with Virtual Space: Whac-A-Mole, Badminton, Pac-man, Space Invaders.

The applications were developed in Unity3D. The backend uses C# with the libraries Clipper, Protobuf and Gurobi.

The space is tracked using the Vive Lighthouse system with eight trackers. The tracking information is forwarded via UDP to head-mounted displays (four GearVRs with Samsung S6 running Android) as shown in Figure 3 (right).

3 Scenograph – Enabling Complex Storylines

Virtual Space allows for simultaneous space sharing of different applications. However, those applications are limited to somewhat arcade-style experiences. Scenograph pushes this idea further by mapping complex experiences onto space. By making real-walking experiences independent of any particular tracking volume, Scenograph provides a crucial component for making real-walking experiences available to consumers.

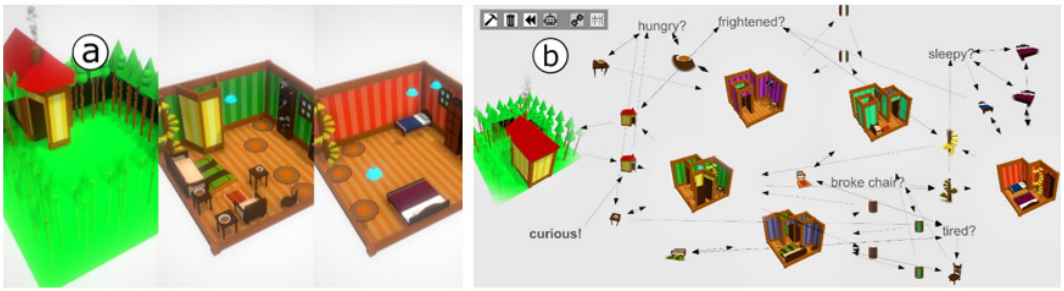

Figure 4: (a) This rendition of the fairy tale 'Goldilocks' consists of three 25m² locations filled with interactive assets. Unfortunately, specifying the tracking volume prevents the experience from running on smaller tracking volumes. (b) Here we used Scenograph to map 'Goldilocks' to an L-shaped 8m² space. While maintaining the narrative structure, it splits the three locations into six smaller ones, each fitting the new tracking volume.

Scenograph is a software system that offers a tracking volume-independent representation of real-walking experiences. Instead of designing for a tracking volume of specific size and shape, Scenograph lets designers specify an experience independent of the tracking volume. The virtual world is then automatically generated by Scenograph.

Internally, Scenograph represents the experience as a bipartite graph, a petri-net, as such it has transitions and nodes. The instantiated graph reflects the arrangement of virtual scenes and objects. Nodes are either spatial or logical. The spatial nodes are the scenes of the experience. The logical nodes are the states that enable story progression. The two kinds of nodes are connected by transitions, the virtual objects in the scenes. Transitions pass tokens between input and output nodes.

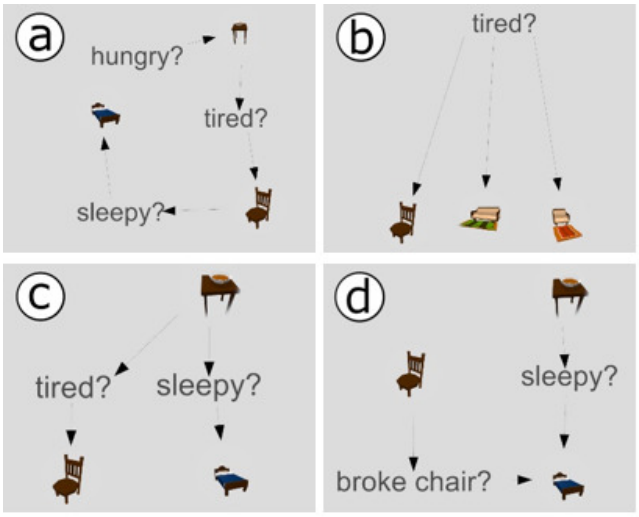

Figure 5: Here are examples of different narrative arrangements, which the petri-net representation allows. (a) Sequence: the 'porridge', 'chair' and 'bed' are accessed in a set order. (b) Conflict: the user needs to decide between one of the 'chairs'. (c) Concurrency: eating the 'porridge' allows using the 'chair' and the 'bed'. (e) Synchronization: sitting on the 'chair' and eating the 'porridge' is necessary before sleeping in the 'bed'. Other progressions are also possible ('confusion', 'merging', etc.).

Scenograph adapts the scenes to the available space by splitting the nodes into multiple instances – this is the core value of the system. As seen in Figure 4, limiting the tracking volume from 25m² to 8m² results in splitting the 'home' node into four nodes. Figure 6 shows in detail how transitions get re-linked to maintain narrative structure. Scenograph takes the petri-net and the available tracking volume as input and transforms them into the new layout.

The system requires a specification of the given tracking volume in the form of a polygon, as well as the resolution into which the space gets virtualized. In our lab setup we have 5m x 5m available, and our example applications is designed for a resolution of 1m x 1m so that Scenograph tessellates the space into 5 x 5 tiles. This specification can be provided by a range of tracking technologies. The designer provides the volume-independent representation of the experience. The end-user has no knowledge of Scenograph's data structure.

Figure 6: Scenograph splits the 'home' node into two as the designed for 25m² get reduced to an L-shaped 12m². (a) This node has six transitions (three porridges followed by three chairs). (b) The porridges are placed in the first (upper) node, the chairs in the second, as they are interacted with after the porridges.

The system offers different packing algorithms, such as "best-fit" using simulated annealing). In Figure 6 we cannot pack three 'porridges' and three 'chairs' into L-shaped 12m², thus Scenograph splits the 'home' node into two.

To determine the number of splits per node and the distribution of transition onto split nodes Scenograph uses divisive hierarchical clustering. The distance computation between transition pairs, required for our hierarchical clustering, uses a semi-decision technique. Scenograph now needs to evaluate which clustering to take for each node. Scenograph cannot linearly iterate through each node separately to find the right clustering, as some virtual objects need to be instantiated on the same tiles in more than one scene (transitions with different spatial nodes as input and output, such as a door). This means that Scenograph needs to consider all possible clusterings for all nodes in parallel, making the packing problem 3-dimensional (width, depth, occurrence in nodes). Scenograph iterates through all possible clustering in an informed manner. The number of clusters and therefore the potential connected scenes to consider is exponential in the number of virtual objects. Our hierarchical clustering does not reduce this amount, it merely sorts all potential clusterings based on the conceptual distance between transitions. The number of virtual objects and thus of potential clusters is different for each scene. For example, our three nodes have one transition ('forest' has a door leading to 'home'), eight transitions ('home' has 3 'porridges', 3 'chairs', 1 'door', 1 'stairway') and three transitions ('upstairs bedroom' has 2 'beds', 1 'stairway'), leaving $2^0 + 2^7 + 2^2 = 512$ possibilities. Each possibility corresponds to a certain clustering depth per node, which we represent using a mixed radix numeric system (e.g., $1_1 2_6 1_3$ corresponds to splitting the second node in two). We iterate through this numeric system linearly first based on the checksum of this clusterings number (to reduce the number of nodes) and then on its order within the hierarchical clustering (maximizing proximity of virtual objects that are also conceptually close).

Scenograph loads and unloads scenes dynamically depending on the users' interaction with the virtual objects or where they walk. Scenograph uses corridors similar to "impossible spaces" to connect scenes, which serve as portals or locks. The

L-shaped 12m² space in Figure 6 can thus be used twice to fit the 'porridges' as well as the 'chairs'. Less overlap makes the technique less perceptible. However, since we focus on small spaces, Scenograph here fully overlaps the scenes.

The system is implemented in C#, the example application and editor interface in Unity3D. ALGLIB is used for clustering.

4 Stuff-Haptics – enabling complex virtual scenography on 3D geometry

Virtual Space allows co-use of space and Scenograph allows complex storylines on arbitrary spaces. Stuff-Haptics is a software tool that provides consistent storylines not only across different spaces, but across limited, uncurated sets of physical props. Stuff-Haptics lays out the virtual objects onto an annotated scan of the stuff in users homes. Given limited sets of props, Stuff-Haptics offers different solutions to best preserve the experience, either through re-use of props or by pruning the storyline to still fit the most important story elements.

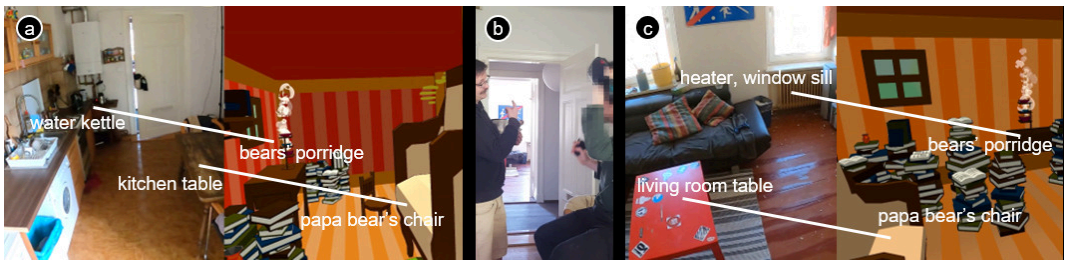

Figure 7: (a) The user invokes a virtual reality experience, 'Goldilocks and the three bears' (from [31]), in their regular living environment. Our software tool Stuff-Haptics provides passive haptics to the experience by automatically lining up relevant virtual objects with physical props of matching haptic qualities, such as the 'bears' three bowls of porridge' with the physical water kettle, and 'papa bear's chair' with the physical table. (b) When a flatmate needs to use the kitchen, our user moves on to another room. (c) Stuff-Haptics redesigns the virtual experience on the fly. It now places the porridges onto the physical window sill above the physical heater and the chair onto the physical living room table.

We fuse Scenograph's petri-net structure together with parametric model descriptions together with a constraint solver to place virtual content. We integrate the story structure in the process. Also we added material properties (surface elasticity, heat emission) for a better haptic experience.

Our algorithm takes in an annotated scan of the physical environment and simplifies it by extracting only relevant (big enough) surfaces. In a pre-processing step

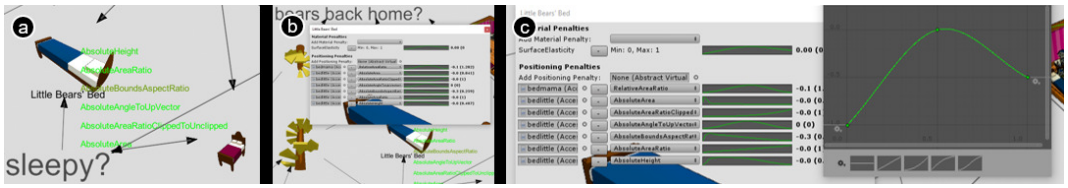

Figure 8: (a) By hovering over 'little bear's bed' we preview its parametric description. (b) This popup window details all parameters and current fitness values for 'little bear's bed'. (c) We change the valuation for 'surface elasticity' to get a somewhat soft 'bed'. The 13 virtual objects in Goldilocks have a total of 91 rules.

for the mapping, for every virtual object, our algorithm first dismisses any surface which would not provide a fit.

Algorithm 1: Map virtual objects to physical surfaces

Input: S – scan of the physical environment (as annotated surfaces), P – storyline as petri-net (contains nodes, contains transitions as virtual objects with placement rules)
Output: mapping M from P onto S

$V \leftarrow P.GetVirtualObjects()$
for each v_i in V **do**
  $M(v_i) \leftarrow RandomSolution(v_i, S)$
  $o_i \leftarrow CreateOptimizer(v_i, S)$
end for
$A \leftarrow AmountNodesChanged(P, V)$
for each o_i **where** $o_i.StillOptimizing()$ **do**
  $s_i \leftarrow o_i.SampleNewSolution(S)$
  $f \leftarrow Fitness(v_i, s_i) =>$
    $L \leftarrow \{\}$
    $R_i \leftarrow v_i.GetPlacementRules()$
    for each r in R_i **do**
      $eval \leftarrow r.Evaluate(s_i)$
      $v_r \leftarrow r.GetRelativeVirtualObject()$
      $relevance \leftarrow A(v_r) / Max(A)$
      $L.add(eval * relevance)$
    end for
    $Fitness(v_i, s_i) \leftarrow LogSum(L)$
  if $f > Fitness(v_i, M(v_i))$ **then**
    $M(v_i) \leftarrow s_i$
  end if
end for
for each v_i in V **do**
  if $Fitness(v_i, M(v_i)) < v_i.Thres(M(v_i))$ **then**
    $M(v_i) \leftarrow \{\}$
  end if
end for

Algorithm 2: Re-use props

Input: P – storyline as petri-net (contains nodes, contains transitions as virtual objects with placement rules), M – mapping of P onto current physical environment
Output: scene splits based on P and M

$N \leftarrow GetSpatialNodes(P)$
for each n in N **do**
  $V \leftarrow P.GetVirtualObjectsOfNode(n)$
  $d \leftarrow P.GetDistanceMatrix(V)$
  for each v_a, v_b **in** V **with** $a \neq b$ **do** *
    $R_a \leftarrow v_a.GetPlacementRules()$
    for each r in R_a **do**
      if $v_b = r.RelativeVirtualObject()$ **then**
        $d(a, b) \leftarrow 0$
      end if
    end for
  end for
  $g \leftarrow GenerateDendrogram(d, V)$
  $C \leftarrow GetPossibleClustersSorted(g)$
  while $M.VirtualObjectsOverlap(V, P)$ **do** *
    $c \leftarrow C.Next()$
    $N_c \leftarrow n.SplitNode\ (c)$
    for each n_i in N_c **do**
      $V_i \leftarrow P.GetVirtualObjectsOfNode(n_i)$
      if $M.VirtualObjectsOverlap(V_i, P)$ **then**
        $P.ReplaceNode(n, N_c)$
      end if
    end for
  end while
end for
for each n in $GetSpatialNodes(P)$ **do**
  $GenerateScene(n)$
end for

* These are the main modifications to the original algorithm

Figure 9: Core algorithms.

In each processing step Stuff-Haptics calculates a new fit for each virtual object using a stochastic method, the covariance matrix adaptation evolution strategy (CMA-ES [4]). Note that each virtual object set is really comprised of a set of objects. The optimization runs separately for every virtual object set, but it runs simultaneously to account for interdependencies between object sets. Our algorithm uses the graph-based story structure to weight virtual objects more heavily in the global optimization, which are more relevant for story progression. Virtual objects are ranked by the amount logical states they affect.

In practice, a lack of matching physical props prevents mixed reality experiences from running everywhere.

As a first solution, we re-use props. 'Goldilocks' requires three virtual beds, but it may be that fewer physical props exist that fit the parametric design, say just one sofa. That prop then can be assigned to all beds. This contradicts some placement rules, but it maps all beds onto the environment. Stuff-Haptics automatically splits the 'bedroom' into three connecting ones by modifying the Scenograph algorithm (Figure 9).

Creators might not want to split the virtual environment. Stuff-Haptics offers to automatically remove less relevant story objects to free up props and space for more relevant objects. This relevance is again measured by its impact on the story graph.

While this solution allows the experience to still run, some basic shaped props should exist to be able to instantiate enough relevant parts of the experience. For Goldilocks this is a horizontal surface of certain size for the 'beds' and 'chairs' and a heat source for the 'porridge'. All the objects in the narrative structure have an importance for the story, which, if removed, decrease the overall fitness value. This overall value is also reduced by the valuations of each mapping. The software tool compares this total fitness value to a threshold set by the application to decide whether or not to run that application in the given physical environment.

5 Mise Unseen

The aforementioned systems generate complex experiences onto arbitrary spaces. While these systems also allow for runtime changes and regeneration of the experience, these changes might be perceivable to the user.

Application scenarios are manifold (1), specifically it allows re-use of space and props for passive haptics and real walking.

Mise-Unseen is a software system that applies covert changes to a virtual scene that can occur inside the us-er's field of view. Mise-Unseen processes gaze data to prevent the user from noticing a change. Specifically it prevents observation of the change when it is happening, recall after it has happened and anticipation when it is about to happen (Figure 11). To achieve this, Mise-Unseen uses five models of user attention together with visual masking techniques. Each model incorporates the knowledge of a specific field of perception theory.

Prevent observation: focus for attention, pupillometry and saccades and cognitive load We compute the convex hull of a projection of an object's geometry onto the view plane. If the angular distance between the convex hull and the currently gaze point falls below a threshold, the object is considered to be in the user's focus. This threshold depends on the precision and reliability of the eye-tracking data: we use a minimum angular threshold of 7° to be above the 5° angle of foveal vision [7], factoring in possible tracking errors. We implemented the Index of Pupillary Activity [2] as a measure of cognitive load. The advantage of using pupil dilation over absolute pupil size is that no calibration is required to establish a baseline. This

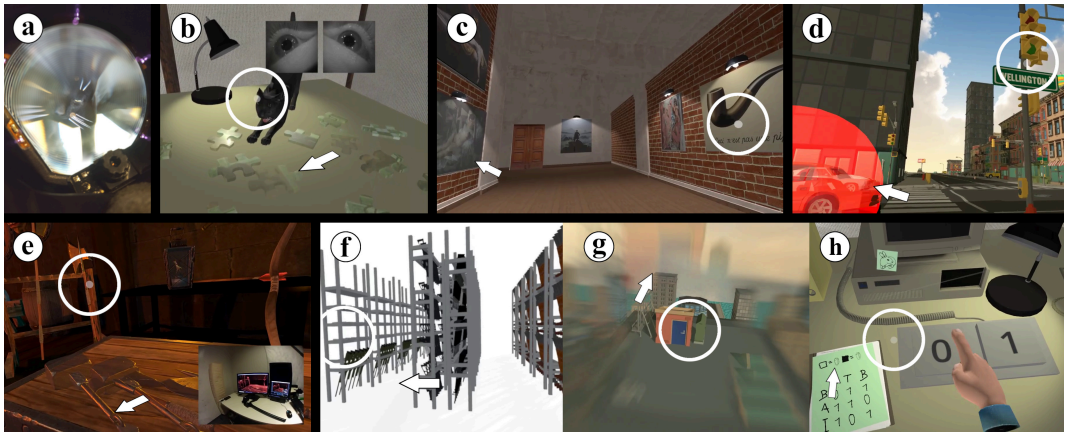

Figure 10: (a) Mise-Unseen unnoticeably changes the scene as the user is focusing elsewhere (circle): (b) cross-facing pieces together (arrow) to help the user solve this puzzle, (c) swapping the gallery painting to adapt to the user's detected interest in modern art, (d) hiding the low fidelity of this "explosion", (e) matching the virtual axe's position to the haptic prop following the detected user interest, (f) shifting storage racks while walking to adapt to a lack of physical space, (g) reducing motion sickness during teleportation by blending static images outside the fovea, and (h) updating hints to prevent the user from solving this riddle too quickly.

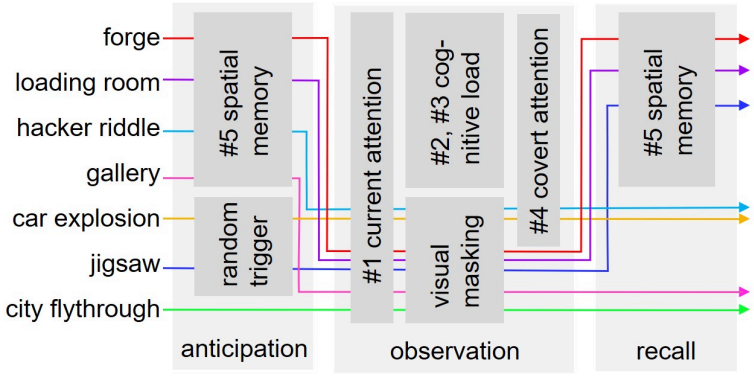

Figure 11: Mise-Unseen prevents anticipation, observation, and recall of a scene change using five attention models together with visual masking techniques.

measure uses fast wavelet transforms on the pupil diameter, counts local peaks of that value, and finally normalizes that count. We additionally factored in the scene illumination (various methods on the influence of luminance on pupil size exist [8]). Our model runs the same measure (fast wavelets) over the changing value for scene illumination. If pupillary activity occurs shortly after a spike in scene illumination (here 200ms), which could have triggered it, its signal is removed from the final count. We also implemented a model that uses saccades as a measure of cognitive load (based on [3, 6]). We count the number of saccades over time and normalize this number (using a maximum of five saccades per five seconds).

Prevent recall: dwell times for spatial memory We implemented a weighted directed graph to represent the user's spatial memory, i.e., the distances (spatial and angular) between relevant objects in the scene the user has internalized. Figure 10e shows the forge as an example. Nodes represent all scene objects, e.g. weapons in the forge, as well as the user themself. The edges and their weights represent the internalized distance be-tween pairs of objects (allocentric), and objects and user (egocentric). The egocentric weights increase when the user looks at an object, the allocentric weights increase when the user looks at any two objects in a row. The weights decrease when the user looks away (here after 1 second). As computational cost is exponential in the number of nodes, pairing all objects can be computationally costly. We reduce cost by working with a subset of relevant objects.

Mise-Unseen reduces the probability of a user noticing a change by recommending new positions and rotations for objects unconnected to the change. For example, the weapon in the forge that is not picked (Figure 10e) moves together with the weapon that should be matched onto the physical prop. Mise-Unseen computes the difference of all internalized distances (edge weights) before and after the change, normalized by a baseline distance and baseline rotation. This difference quantifies the violation of spatial memory and is our predictor for attention. Mise-Unseen recommends a new position and rotation for each object. This results in in less of a difference, less violation of spatial memory and thus lower probability of detection.

Prevent anticipation: unpredictable triggers A user should not be able to anticipate a change by tying it to a specific, overt action: covert changes need covert triggers. We measured user intent as such a random trigger by capturing and thresholding dwell times on objects (or sets of objects). This follows existing implementations that use eye-movement patterns to predict user actions [5, 10]. The hacker riddle uses user understanding as a trigger. Once the user has understood the hint the riddle may be solved, else the hint changes. We measure user understanding by following related work [1, 9] and match the user's exhibited gaze pattern onto a predefined scan path (linear or tree). Here the user's gaze must follow the hidden code sequence on a sheet of paper (Figure 10h). Our implementation uses an variation of attention model five (spatial memory) with faster increasing and decreasing edge weights, similar to a short term memory. Something is deemed understood once the edge weights closely match a set of predefined values.

6 Conclusion

Virtual Space enables sharing of space as a resource for real walking. Scenograph enables more complex experiences. Stuff Haptics enables more complex experiences for arbitrary room geometry. Mise Unseen enables changes to those experiences at runtime.

7 Core Publications

- Sebastian Marwecki, Maximilian Brehm, Lukas Wagner, Lung-Pan Cheng, Florian 'Floyd' Mueller, and Patrick Baudisch. 2018. VirtualSpace - Overloading Physical Space with Multiple Virtual Reality Users. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18). Association for Computing Machinery, New York, NY, USA, Paper 241, 1–10. DOI:https://doi.org/10.1145/3173574.3173815.
- Sebastian Marwecki and Patrick Baudisch. 2018. Scenograph: Fitting Real-Walking VR Experiences into Various Tracking Volumes. In Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology (UIST '18). Association for Computing Machinery, New York, NY, USA, 511–520. DOI:https://doi.org/10.1145/3242587.3242648.
- Sebastian Marwecki, Andrew D. Wilson, Eyal Ofek, Mar Gonzalez Franco, and Christian Holz. 2019. Mise-Unseen: Using Eye Tracking to Hide Virtual Reality Scene Changes in Plain Sight. In Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19). Association for Computing Machinery, New York, NY, USA, 777–789. DOI:https://doi.org/10.1145/3332165.3347919.

References

[1] A. Bochynska and B. Laeng. "Tracking down the path of memory: eye scanpaths facilitate retrieval of visuospatial information". In: *Cognitive processing* 16.1 (2015), pages 159–163. ISSN: 1612-4782.

[2] A. T. Duchowski, K. Krejtz, I. Krejtz, C. Biele, A. Niedzielska, P. Kiefer, M. Raubal, and I. Giannopoulos. "The index of pupillary activity: Measuring cognitive load vis-à-vis task difficulty with pupil oscillation". In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. ACM, 2018, page 282. ISBN: 1-4503-5620-6.

[3] L. Elazary and L. Itti. "Interesting objects are visually salient". In: *Journal of vision* 8.3 (2008), page 3. ISSN: 1534-7362.

[4] N. Hansen. *The CMA Evolution Strategy: A Comparing Review*. Technical report.

[5] R. Singh, T. Miller, J. Newn, L. Sonenberg, E. Velloso, and F. Vetere. "Combining Planning with Gaze for Online Human Intention Recognition". In: *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. 2018, pages 488–496.

[6] J. R. Tole, A. T. Stephens, M. Vivaudou, A. R. Ephrath, and L. R. Young. "Visual scanning behavior and pilot workload." In: (1983).

[7] H. K. Walker, W. D. Hall, and J. W. Hurst. *Clinical Methods: The History, Physical and Laboratory Examinations*. 1990.

[8] A. B. Watson and J. I. Yellott. "A unified formula for light-adapted pupil size". In: *Journal of Vision* 12.10 (2012), page 12.

[9] Y. Yamazaki, H. Hino, and K. Fukui. "Sensing Visual Attention by Sequential Patterns". In: *ICPR '14 Proceedings of the 2014 22nd International Conference on Pattern Recognition*. 2014, pages 483–488.

[10] M. Zank and A. M. Kunz. "Eye tracking for locomotion prediction in redirected walking". In: *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. 2016, pages 49–58.

A Software System for Designing Jigs to Produce Small Batches

Shohei Katakura

Human-Computer Interaction Group
Hasso Plattner Institute for Digital Engineering
shohei.katakura@hpi.de

We introduce a work-in-progress project, a software system for non-engineers to design customized tools (jig and fixture) for small-batch production. The design of such tools for repetitive tasks depend on processing accuracy, processing speed and human factors. Since such design space and trade-off impose design iteration including fabrication and test, it is hard to design without domain experts. In this report we present the work-in-progress software system for designing such customization, especially fixtures for drills with linear sliders. This software enable to design fixture that consider insert-ability (easy-to-operate) and hold-ability (accurate process).

1 Introduction

How to make 500 copies of a 3D model? Such small batches play a role, for example, when farming a physical device out for beta testing or when sending out the first batch of devices of a successfully backed kickstarter project. So how to produce such a "small batch?"

On the one end of the spectrum, we find "personal fabrication" a subfield of human computer interaction and graphics since about 2011, with several hundreds of publications ranging from designing mechanical objects to fabricating interactive matter using digital fabrication machine. Even though this process is commonly referred to as "personal fabrication, [2]" , it is a bit of a misnomer. "Personal Prototyping" would probably be a better description, as generally only a single copy is being made. When trying to make a batch of 500, however, 3D printers, for example, tend to take too long and the per-piece price of tens or hundreds of dollars, tends to become a showstopper when multiplied by 500.

On the other end of the spectrum, we find mass production at a factory, as studied not so much in HCI, but in mechanical engineering and industrial engineering. Here the main objective is to bring the per-piece cost down and making huge amounts of exact copies. Machines like injection molding, machining center, robotic arm and conveyor belts play a major role. Much work is done automatically by those machines, and humans do a few works that those machines cannot. Hundreds to thousands of hours go into optimizing a product design (DFMA [3]), optimizing the timing of each production step to allow the belt to run perpetually, renting space, and setting up production. The cost of this set-up and DFMA tends to go into the lower millions – again, not what we want.

We argue that such small batches should be made how they have traditionally been made: using technique from "manu-facturing (made by hand in latin)", i.e., the processes developed 1800 – 1920 for creating porcelain, e.g., in Höchster Porzellan-Manufaktur. While such manufacturers do not use mass production technologies, such as the aforementioned machining center and belt conveyor, they did and do use means for creating efficiency, such as molds and jigs, and they are subject to the division of labor. This process is used to date for objects created in small numbers, such as to manufacture anything from high-end musical instruments (e.g., Höfner Guitars) to technical equipment, such as laser cutters (e.g., Trotec).

A key role in such manufacturing processes play efficient tools that allow producing custom objects quickly, yet that are faster to make than mass production tools: such as molds, stencils, fixtures, and jigs. These tools bring the manufacturing time per piece often down to a few seconds per piece and allow them to make exact copies.

Unfortunately, such hand-made tools tend to be made and optimized by tradition and experience. In order to design such tools, non-manufacturers such as software engineers and designers need the know-how of manufacturers and manufacturing processes(knowledge of mechanical engineering and industrial engineering).

This report presents tools for designing such customization tools. Since this research is still work-in-progress, and at present, it is an introduction to the design software for fixtures for drills with linear slider. The key to this software enable to design fixture that consider insert-ability (easy-to-operate) and hold-ability (accurate process).

2 Related Work

2.1 computational design of functional object

To make functional objects in the physical world (furniture, etc.), designers need to think about the material characteristics, structure, and manufacturability. It is difficult for people without knowledge of, for example, mechanical engineering.

Researchers at computer graphics and HCI proposed software systems for designing such physical objects without engineering knowledge. AutoConnect [6] can generate a 3D printable connector that connects two physical objects. Instead of a typical friction physics simulation, they use data-driven physics that uses data that measures and parameterizes the frictional force of the actually manufactured connectors. Umetani et al. have proposed a suggestive user interface CAD system that allows designers to design shapes and verify the physical validity of furniture simultaneously [10]. They also proposed Pteromys [11] that allows users to design well-flying paper planes of any shape through high-speed machine learning-based physics simulations using proprietary wing theory.

2.2 human-work-aware fabrication

In fabricating large objects or an object containing a complicated mechanism, it is sometimes necessary to assemble, positioning manufactured parts. Typically, humans do such work. Several large-scale personal fabrication research has supported these human tasks within computer-supported system.

JigFab[8] system helps non-skilled person to cut (or mill) wood plate into designed shape without mesureing and annotation. This system generates constraints of power tools for customized wooden furniture, such as shelves and desks. Yoshida et.al propsed projector-based guiding system for large-scale chopstick architecture [12], and also audio-visual guiding system for positioning unique branches in a CNC machine [7]. Rivers et al. propose position correcting tools for 2D complex shape [9]. A user of the tool coarsely positions a desired path, while the device adjusts the position of the tool within the frame to correct the user's positioning error in real time. Devendorf et al. worked on an experimental study that considers humans as machines. They builded a manufacturing guide system for human using laser pointers [4].

2.3 design customized tools for production efficiency and accuracy

There is research designing customized tools for production in the domain of industrial manufacturing. In machining, the fixture design is essential. What designers should consider is positioning and constraint. The most famous method is the 3-2-1 principle is one of a well-known method for designing fixtures [5]. This design principle allows for the stable placement of the workpiece with minimal contacts. If additional restraints are required, the workpiece is fixed with a clamp device. The point of this principle are that it is easy for humans to insert, and in the case of drills, prevent swarf stay. However, this principle is only applicable to relatively large, standardized shapes (e.g., rectangular metal with a flat mounting surface). For example, cylinder shapes, a V-shaped fixture is a better option.

Customized tools in manufacturing need to be designed with consideration for humans to perform their jobs efficiently without injury. In the motion and time study, Barnes summarized that from tool design to placement concerning human movement efficiency [1].Therblig analyzed and factorized human movements in its smallest form and is used to design workspaces.

3 Jig Design and Software System

We use a slide-type drill tool as an example (Figure 1). Our software enables a user to design the part of the fixture that holds the object in the figure. Here, we have not considered clamping (because we use hand). The most straightforward fixture is a half mold, but in this case, there is a problem that swarf piles on the fixture (accurateness decreases), and sometimes it is difficult to take in and take out. Another fixture design has a 3-6 support point pillars that support the installation

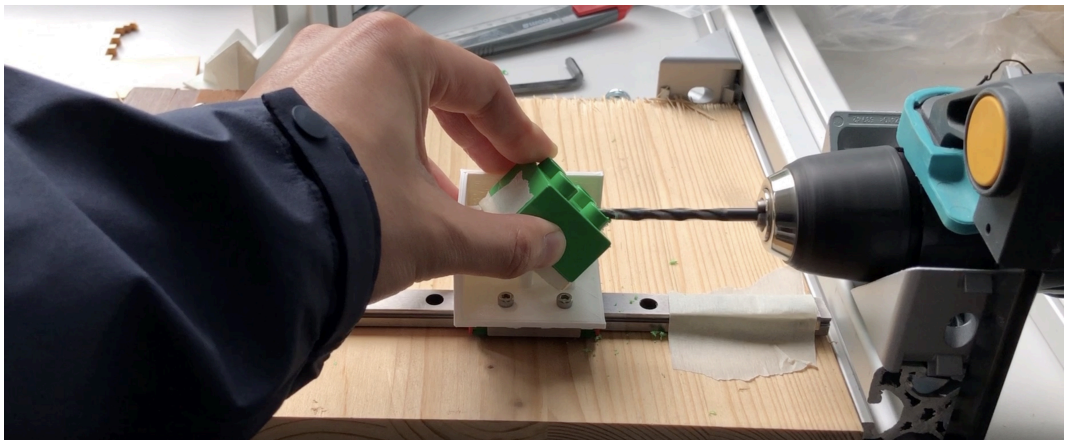

Figure 1: To drill workpiece using linear slider.

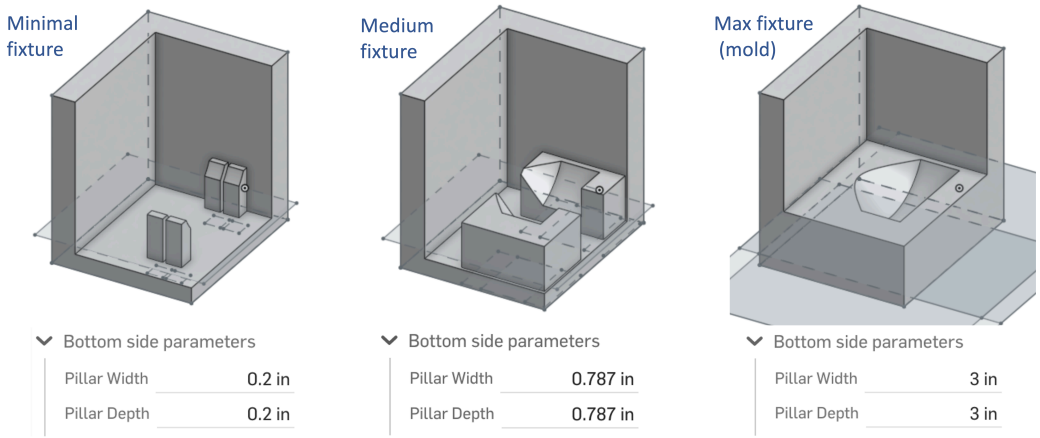

Figure 2: Result of the generated fixture. The left one is minimal support pillars; the right one is half mold. The middle one is an interpolation between the two.

surface, the same as a 3-2-1 principle. In that case, it is possible to prevent swarf from accumulating, and it is easy to put in and take out. However, it may be inserted at the wrong angle (non-foolproof design). We try to design a software that allows humans to consider these trade-offs.

The following three design consideration for fixture in our software (refer from "Jig and Fixture Design" [5]): insert-abirity, hold-abirity, and foolproof.

Figure 2 shows result of generated fixtures. A user can interpolate half mold and minimal points. We implemented this system on OnShape CAD using FeatureScript.

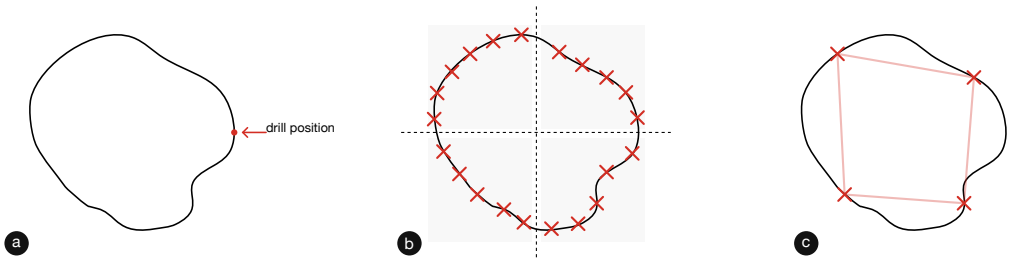

Figure 3: (a) Map 3D object to 2D from top. (b) Divide 4 region. The boundary line is a direction of drill and orthogonal to the direction at center of mass. (c) Our algorithm (See Figure 4) pick best support point.

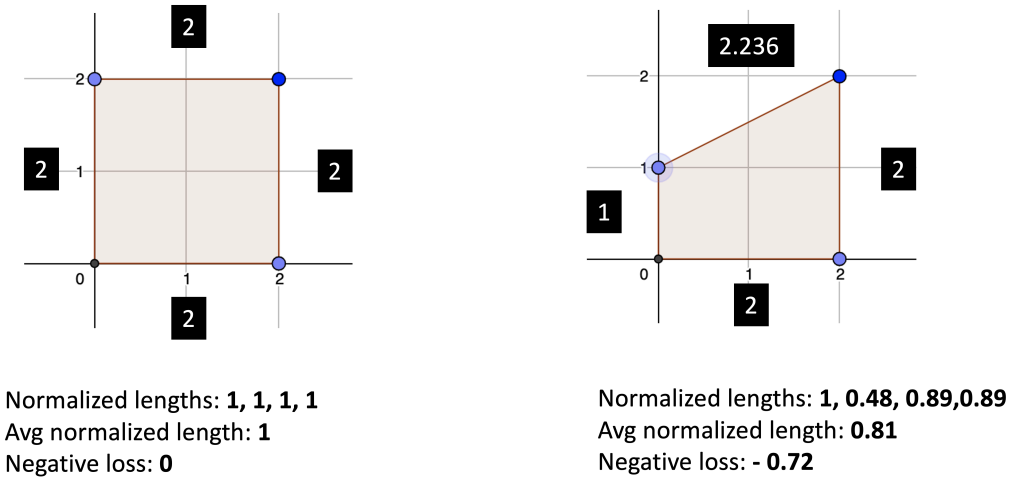

Figure 4: Visualization of algorithm 1

4 Find Best Support Points

Here we will show the algorithm behind the software. The important algorithm is finding the best support point for arbitrary objects. Figure 3 shows what the algorithm does visually. First, a user input drill position on a CAD (Figure 3A). A 3D object is mapped to 2D from the top and divided into four segments; then, the contour line is divided into an arbitrary number of points (Figure 3B). Finally, the best combination of 4 points are picked up using our equilateral fitness function (Figure 3C). The function is described Algorithm 1 (Figure 4).

5 Conclusion and Future Works

In this report, we introduced a work-in-progress system to design customized tools for small batch production. As future work, we understand the proper formulation of

Algorithm 1 Calculate equilateral fitness

```
L[4]:edge length between points
F = total loss of fitness
for k = 0 to 3 do
    F += L[k] - avg(L)
end for
return F
```

customization tools and finding a good example. Currently, it is a tool for a particular process of particular objects. Customized tools have various requirements depending on the manufacturing method and process, but it is necessary to understand and formulate them.

References

[1] R. M. Barnes. *Motion and time study*. Wiley, 1949.

[2] P. Baudisch, S. Mueller, et al. "Personal fabrication". In: *Foundations and Trends® in Human–Computer Interaction* 10.3–4 (2017), pages 165–293.

[3] G. Boothroyd and L. Alting. "Design for assembly and disassembly". In: *CIRP annals* 41.2 (1992), pages 625–636.

[4] L. Devendorf and K. Ryokai. "Being the machine: reconfiguring agency and control in hybrid fabrication". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 2015, pages 2477–2486.

[5] E. Hoffman. *Jig and fixture design*. Cengage Learning, 2012.

[6] Y. Koyama, S. Sueda, E. Steinhardt, T. Igarashi, A. Shamir, and W. Matusik. "AutoConnect: computational design of 3D-printable connectors". In: *ACM Transactions on Graphics (TOG)* 34.6 (2015), pages 1–11.

[7] M. Larsson, H. Yoshida, and T. Igarashi. "Human-in-the-loop fabrication of 3D surfaces with natural tree branches". In: *Proceedings of the ACM Symposium on Computational Fabrication*. 2019, pages 1–12.

[8] D. Leen, T. Veuskens, K. Luyten, and R. Ramakers. "JigFab: Computational Fabrication of Constraints to Facilitate Woodworking with Power Tools". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. 2019, pages 1–12.

[9] A. Rivers, I. E. Moyer, and F. Durand. "Position-correcting tools for 2D digital fabrication". In: *ACM Transactions on Graphics (TOG)* 31.4 (2012), pages 1–7.

[10] N. Umetani, T. Igarashi, and N. J. Mitra. "Guided exploration of physically valid shapes for furniture design." In: *ACM Trans. Graph.* 31.4 (2012).

[11] N. Umetani, Y. Koyama, R. Schmidt, and T. Igarashi. "Pteromys: interactive design and optimization of free-formed free-flight model airplanes". In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), pages 1–10.

[12] H. Yoshida, T. Igarashi, Y. Obuchi, Y. Takami, J. Sato, M. Araki, M. Miki, K. Nagata, K. Sakai, and S. Igarashi. "Architecture-scale human-assisted additive manufacturing". In: *ACM Transactions on Graphics (TOG)* 34.4 (2015), pages 1–8.

Intrinsic Editing of RGB-D Images on Smartphones

Sumit Shekhar

Computer Graphics Systems Group
Hasso Plattner Institute for Digital Engineering
sumit.shekhar@hpi.de

This report describes my recent activities on Service-oriented Systems Engineering between April 2020 and October 2020 with the HPI Research School, and summarizes my research and teaching activities.

1 Overview

On a bright sunny day, it is quite easy for us to identify objects like a wall, a car, or a bike irrespective of their color, material or whether they are partially shaded. This remarkable capacity of human visual system (HVS) to disentangle visual ambiguities due to color, material, shape, and lighting is a result of many years of evolution [7]. Replicating this ability for machine vision – to enable better scene understanding – has been a widely researched topic, but ever has been challenging because of its *ill-posed* and *under-constrained* nature.

The physical formation of an image involves various unknowns at macroscopic and microscopic levels, and decomposing them altogether makes it ill-posed. A more relaxed approximation is given by the *Dichromatic Reflection Model* where an image (I) is assumed to be composed of the sum of specular (I_s) and diffuse (I_d) components (at every pixel location $\boldsymbol{x}$) [31]:

$$I(\boldsymbol{x}) = I_d(\boldsymbol{x}) + I_s(\boldsymbol{x}). \tag{1}$$

The diffuse component (I_d) can be further expressed as the product of *albedo* (A) and *shading* (S) [5]:

$$I_d(\boldsymbol{x}) = A(\boldsymbol{x}) \cdot S(\boldsymbol{x}). \tag{2}$$

However, even this approximation is under-constrained as three unknowns, namely $A(\boldsymbol{x})$, $S(\boldsymbol{x})$ and $I_s(\boldsymbol{x})$ need to be solved given only the image color $I(\boldsymbol{x})$. We propose a novel method to extract intrinsic layers of albedo, shading and specularity. The computed layers, apart from offering better scene understanding, facilitate a range of Augmented Reality (AR) applications such as recoloring, relighting, appearance editing etc.. To this end, we perform the above decomposition on a high-end smartphone with a built-in depth sensor. The specularity removal is carried out as a pre-processing step followed by a depth-based energy minimization for computing the other two layers.

One limitation for most of the previous work is the assumption of a complete diffuse reflection. In general, the decomposition of an image into diffuse reflectance (albedo) and shading is referred to as Intrinsic Image Decomposition (IID). The existing IID algorithms can be broadly classified into two categories:

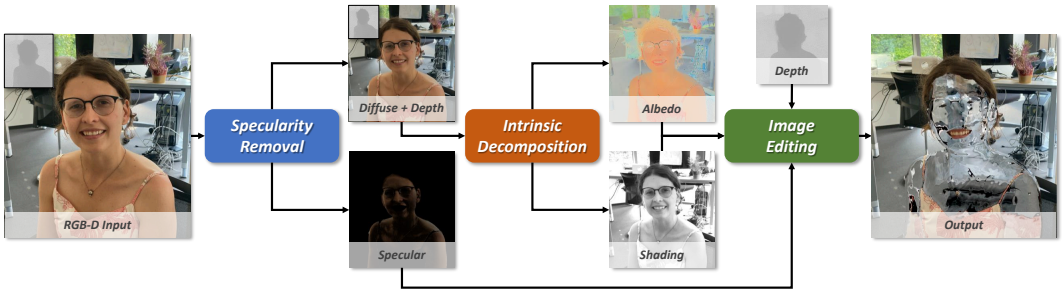

Figure 1: Flowchart of our complete framework showing extraction of intrinsic layers followed by image editing

***Learning*-based methods:** the priors on albedo and shading are incorporated as loss functions, and the decomposition is learned by training. In the past few years – with the significant improvement in deep-learning technology – such methods have become quite popular [11, 21, 40]. However, capturing real-world training data for IID is challenging and the existing datasets might not be sufficient [15, 32]. Unsupervised learning does not require any training data, however, the results are not satisfactory [26, 28]. Moreover, it is challenging to adapt neural-network based methods for smartphones as they require significant computational resources and often are limited to low image resolutions.

***Optimization*-based methods:** a cost function based on priors is minimized to find an approximate solution. Initial techniques use simplistic priors, which are not suitable for real-world scenes [36]. More complex priors improve the accuracy at the cost of associated computational complexity [4, 7, 39]. The advent of readily available depth sensors fostered depth-based methods for IID [10, 19]. However, such sensors need to be attached to a desktop/laptop making them unsuitable for outdoor scenes. Moreover, a few years ago, it was not feasible to deploy any of the above methods on a mobile phone due to their limited visual processing capacity. We leverage the computing capabilities of current high-end consumer smartphones with a dedicated GPU and real-time capture of RGB-D data for our purpose.

As an additional constraint, only a few previous methods perform both IID and specularity extraction together. Innamorati *et al.* [18] and Shi *et al.* [34] employ a learning-based technique: both of them train and test for single objects but do not consider a realistic scene with many objects. The algorithms by Alperovich *et al.* [1] are designed for light-fields but cannot be used for a single image. The method of Beigpour *et al.* [6] is applicable for a single image and, like ours, removes specularities in a pre-processing step. However, for specularity extraction, they do not consider chroma channels leading to artifacts in highly saturated image regions. Moreover, their method is an order of magnitude slower than ours. Unlike most of the previous standalone specularity removal methods, we showcase our results on a broad range of realistic images [2]. We treat high-frequency and low-frequency specularities differently, thereby achieving seamless outputs. Furthermore, our specularity

removal step performs at interactive frame rates, achieving improved performance and quality over the current state-of-the-art [13]. Overall, we propose the following contributions: (See Figure 1)

- An algorithm for intrinsic decomposition of RGB-D images on smartphones,
- A novel, interactive specularity removal method that treats high-frequency and low-frequency specularities differently, and that is well-suited for real-world images,
- A variety of mobile-based applications using the intrinsic layers of depth, albedo, shading, and specularity.

2 Related Work

2.1 Intrinsic Image Decomposition

The term intrinsic decomposition was introduced in the literature by Barrow and Tenenbaum [5]. The Retinex theory by Land and McCann proved to be a crucial finding, which became part of many following algorithms as a prior [24]. In the course of previous decades, intrinsic decomposition algorithms have been proposed for image [4, 7, 28, 36, 39], video [9, 30, 38], multiple-views [12, 23] and light-fields [1, 6, 14]. A survey covering many of these algorithms is provided by Bonneel *et al.* [8]. A particular class of algorithms use depth as additional information for IID. Lee *et al.* [25] use normals to impose constraints on shading and also use temporal constraints to obtain smooth results. Chen and Koltun [10] further decompose shading into direct and indirect irradiance; the authors use depth to construct position-normal vectors for regularizing them. Hachama *et al.* [17] use a single image or multiple RGB-D images to construct a point cloud. The normal vectors along with low dimensional global lighting model is used to jointly estimate lighting and albedo. Similarly, we use depth information to impose constraints on shading. However, unlike previous methods, a pre-processing step of specularity removal makes our method robust against specular image pixels.

2.2 Specularity Removal

Some of the earliest methods for specularity removal were based on color segmentation, thus they were not robust against textures [3, 22]. Mallik *et al.* [29] introduce a partial differential equation (PDE) in the SUV color space that iteratively erodes the specular component. A class of algorithms use the concept of specular-free image based on chromaticity values [33, 35]. Yang *et al.* [37] use a similar approach, and achieve real-time performance by employing parallelized processing. Kim *et al.* [20] use a dark channel prior to obtain specular-free images, followed by an optimization framework. Guo *et al.* [16] propose a sparse low-rank reflection model and use a L_1

norm constraint in their optimization to filter specularities. A broad survey of specularity removal methods is provided by Artusi *et al.* [2]. Recently, Li *et al.* [27] also use image and depth data for removing specularity from human facial images. Most of these methods, however, employ specific object(s) or scene settings to evaluate their methods and do not consider generic real-world images. A recent method by Fu *et al.* [13] aims to address this issue; the authors assume that specularity is generally sparse and the diffuse component can be expressed as a linear combination of basis colors. They present a wide range of results, however, the optimization solving is comparably slow. By contrast, our method is aimed for generic real-world images with interactive performance on mobile devices.

3 Applications

A perfect, physically accurate editing of a photo would require full inverse rendering with high precision. However, we can achieve convincing material and volumetric media editing even without the above. The intrinsic decomposition output can also be effectively used for enhancing image stylization results. A mobile-based intrinsic decomposition, as provided in this work, could be used for various other photo-realistic image editing in Augmented Reality applications. As part of future work, we aim to relax some of the existing assumptions and address image scenes with multi-color illuminant and indirect illumination effects.

4 Teaching and Other Activities

Apart from participation within the Research School and my PhD research work, I have also provided assistance for the following teaching activities of our group,

- Teaching assistant for the *Techniques for Image and Video Analysis* (M.Sc) lecture (summer term 2020).
- Master thesis supervision titled *Real-time Optic Flow for Mobile Devices.*

References

[1] A. Alperovich and B. Goldluecke. "A Variational Model for Intrinsic Light Field Decomposition". In: *Asian Conference on Computer Vision (ACCV), November 20-24*. Volume 10113. Lecture Notes in Computer Science. 2016, pages 66–82. DOI: 10.1007/978-3-319-54187-7_5.

[2] A. Artusi, F. Banterle, and D. Chetverikov. "A Survey of Specularity Removal Methods". In: *Computer Graphics Forum* 30.8 (2011), pages 2208–2230. DOI: 10.1111/j.1467-8659.2011.01971.x.

[3] R. Bajcsy, S. W. Lee, and A. Leonardis. "Detection of Diffuse and Specular Interface Reflections and Inter-Reflections by Color Image Segmentation". In: *International Journal of Computer Vision* 17.3 (Mar. 1996), pages 241–272. ISSN: 0920-5691. DOI: 10.1007/BF00128233.

[4] J. T. Barron and J. Malik. "Shape, Illumination, and Reflectance from Shading." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.8 (2015), pages 1670–1687. DOI: 10.1109/TPAMI.2014.2377712.

[5] H. Barrow and J. Tenenbaum. *Recovering intrinsic scene characteristics from images*. Technical report. Artificial Intelligence Center, SRI International, 1978, page 15.

[6] S. Beigpour, S. Shekhar, M. Mansouryar, K. Myszkowski, and H.-P. Seidel. "Light-Field Appearance Editing based on Intrinsic Decomposition". In: *Journal of Perceptual Imaging* 1.1 (2018), page 15. DOI: 10.2352/J.Percept.Imaging.2018.1.1.010502.

[7] S. Bell, K. Bala, and N. Snavely. "Intrinsic Images in the Wild". In: *ACM Transactions on Graphics* 33.4 (July 2014). ISSN: 0730-0301. DOI: 10.1145/2601097.2601206.

[8] N. Bonneel, B. Kovacs, S. Paris, and K. Bala. "Intrinsic Decompositions for Image Editing". In: *Computer Graphics Forum* 36.2 (May 2017), pages 593–609. ISSN: 0167-7055. DOI: 10.1111/cgf.13149.

[9] N. Bonneel, K. Sunkavalli, J. Tompkin, D. Sun, S. Paris, and H. Pfister. "Interactive Intrinsic Video Editing". In: *ACM Transactions on Graphics* 33.6 (Nov. 2014). ISSN: 0730-0301. DOI: 10.1145/2661229.2661253.

[10] Q. Chen and V. Koltun. "A Simple Model for Intrinsic Image Decomposition with Depth Cues". In: *IEEE International Conference on Computer Vision (ICCV)*. USA, 2013, pages 241–248. ISBN: 978-1-4799-2840-8. DOI: 10.1109/ICCV.2013.37.

[11] L. Cheng, C. Zhang, and Z. Liao. "Intrinsic Image Transformation via Scale Space Decomposition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pages 656–665. DOI: 10.1109/CVPR.2018.00075.

[12] S. Duchêne, C. Riant, G. Chaurasia, J. L. Moreno, P.-Y. Laffont, S. Popov, A. Bousseau, and G. Drettakis. "Multiview Intrinsic Images of Outdoors Scenes with an Application to Relighting". In: *ACM Transactions on Graphics* 34.5 (Nov. 2015). ISSN: 0730-0301. DOI: 10.1145/2756549.

[13] G. Fu, Q. Zhang, C. Song, Q. Lin, and C. Xiao. "Specular Highlight Removal for Real-world Images". In: *Computer Graphics Forum* 38.7 (2019), pages 253–263. DOI: 10.1111/cgf.13834.

[14] E. Garces, J. I. Echevarria, W. Zhang, H. Wu, K. Zhou, and D. Gutierrez. "Intrinsic Light Field Images". In: *Computer Graphics Forum* 36.8 (2017), pages 589–599. DOI: 10.1111/cgf.13154.

[15] R. Grosse, M. K. Johnson, E. H. Adelson, and W. T. Freeman. "Ground truth dataset and baseline evaluations for intrinsic image algorithms". In: *International Conference on Computer Vision (ICCV)*. 2009, pages 2335–2342. DOI: 10.1109/ICCV.2009.5459428.

[16] J. Guo, Z. Zhou, and L. Wang. "Single Image Highlight Removal with a Sparse and Low-Rank Reflection Model". In: *European Conference on Computer Vision (ECCV), Munich, Germany, September 8-14*. 2018, pages 282–298. DOI: 10.1007/978-3-030-01225-0_17.

[17] M. Hachama, B. Ghanem, and P. Wonka. "Intrinsic Scene Decomposition from RGB-D Images". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pages 810–818. DOI: 10.1109/ICCV.2015.99.

[18] C. Innamorati, T. Ritschel, T. Weyrich, and N. J. Mitra. "Decomposing Single Images for Layered Photo Retouching". In: *Computer Graphics Forum* 36.4 (2017), pages 15–25. DOI: 10.1111/cgf.13220.

[19] J. Jeon, S. Cho, X. Tong, and S. Lee. "Intrinsic image decomposition using structure-texture separation and surface normals". In: *European Conference on Computer Vision (ECCV)*. 2014, pages 218–233. DOI: 10.1007/978-3-319-10584-0_15.

[20] H. Kim, H. Jin, S. Hadap, and I. Kweon. "Specular Reflection Separation Using Dark Channel Prior". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pages 1460–1467. ISBN: 978-0-7695-4989-7. DOI: 10.1109/CVPR.2013.192.

[21] S. Kim, K. Park, K. Sohn, and S. Lin. "Unified Depth Prediction and Intrinsic Image Decomposition from a Single Image via Joint Convolutional Neural Fields". In: *European Conference on Computer Vision (ECCV)*. 2016, pages 143–159. DOI: 10.1007/978-3-319-46484-8_9.

[22] G. J. Klinker, S. A. Shafer, and T. Kanade. "The measurement of highlights in color images". In: *International Journal of Computer Vision* 2.1 (June 1988), pages 7–32. ISSN: 1573-1405. DOI: 10.1007/BF00836279.

[23] P. Laffont, A. Bousseau, and G. Drettakis. "Rich Intrinsic Image Decomposition of Outdoor Scenes from Multiple Views". In: *IEEE Transactions on Visualization and Computer Graphics* 19.2 (2013), pages 210–224. DOI: 10.1145/2343045.2343113.

[24] E. H. Land and J. J. McCann. "Lightness and retinex theory". In: *Journal of the Optical Society of America* 61.1 (1971), pages 1–11. DOI: 10.1364/JOSA.61.000001.

[25] K. J. Lee, Q. Zhao, X. Tong, M. Gong, S. Izadi, S. U. Lee, P. Tan, and S. Lin. "Estimation of Intrinsic Image Sequences from Image+Depth Video". In: *European Conference on Computer Vision (ECCV)*. 2012, pages 327–340. DOI: 10.1007/978-3-642-33783-3_24.

[26] L. Lettry, K. Vanhoey, and L. Van Gool. "Unsupervised Deep Single-Image Intrinsic Decomposition using Illumination-Varying Image Sequences". In: *Computer Graphics Forum* 37.7 (2018), pages 409–419. DOI: 10.1111/cgf.13578.

[27] C. Li, S. Lin, K. Zhou, and K. Ikeuchi. "Specular Highlight Removal in Facial Images". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pages 2780–2789. DOI: 10.1109/CVPR.2017.297.

[28] Z. Li and N. Snavely. "Learning Intrinsic Image Decomposition from Watching the World". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pages 9039–9048. DOI: 10.1109/CVPR.2018.00942.

[29] S. P. Mallick, T. Zickler, P. N. Belhumeur, and D. J. Kriegman. "Specularity Removal in Images and Videos: A PDE Approach". In: *European Conference on Computer Vision (ECCV)*. 2006, pages 550–563. DOI: 10.1007/11744023_43.

[30] A. Meka, M. Zollhöfer, C. Richardt, and C. Theobalt. "Live Intrinsic Video". In: *ACM Transactions on Graphics* 35.4 (July 2016). DOI: 10.1145/2897824.2925907.

[31] S. A. Shafer. "Using color to separate reflection components". In: *Color Research & Application* 10.4 (1985), pages 210–218. DOI: 10.1002/col.5080100409.

[32] S. Shekhar, S. Beigpour, M. Ziegler, M. Chwesiuk, D. Palen, K. Myszkowski, J. Keinert, R. Mantiuk, and P. Didyk. "Light-Field Intrinsic Dataset". In: *British Machine Vision Conference (BMVC), Newcastle, UK, September 3-6*. 2018, page 120.

[33] H.-L. Shen and Q.-Y. Cai. "Simple and efficient method for specularity removal in an image". In: *Applied Optics* 48.14 (May 2009), pages 2711–2719. DOI: 10.1364/AO.48.002711.

[34] J. Shi, Y. Dong, H. Su, and S. X. Yu. "Learning non-lambertian object intrinsics across shapenet categories". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pages 1685–1694. DOI: 10.1109/CVPR.2017.619.

[35] R. T. Tan and K. Ikeuchi. "Separating Reflection Components of Textured Surfaces Using a Single Image". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.2 (Feb. 2005), pages 178–193. DOI: 10.1109/TPAMI.2005.36.

[36] M. F. Tappen, W. T. Freeman, and E. H. Adelson. "Recovering Intrinsic Images from a Single Image". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.9 (Sept. 2005), pages 1459–1472. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2005.185.

[37] Q. Yang, S. Wang, and N. Ahuja. "Real-Time Specular Highlight Removal Using Bilateral Filtering". In: *European Conference on Computer Vision (ECCV)*. 2010, pages 87–100. DOI: 10.5555/1888089.1888097.

[38] G. Ye, E. Garces, Y. Liu, Q. Dai, and D. Gutierrez. "Intrinsic Video and Applications". In: *ACM Transactions on Graphics* 33.4 (July 2014). DOI: 10.1145/2601097.2601135.

[39] Q. Zhao, P. Tan, Q. Dai, L. Shen, E. Wu, and S. Lin. "A Closed-Form Solution to Retinex with Nonlocal Texture Constraints". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.7 (July 2012), pages 1437–1444. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2012.77.

[40] T. Zhou, P. Krahenbuhl, and A. A. Efros. "Learning data-driven reflectance priors for intrinsic image decomposition". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pages 3469–3477. DOI: 10.1109/ICCV.2015.396.

Point Cloud Analytics: Proposed Future Work

Vladeta Stojanovic

Computer Graphics Systems Group
Hasso Plattner Institute for Digital Engineering
vladeta.stojanovic@hpi.de

Automated processing, semantic-enrichment and visual analytics methods for point clouds are often use-case specific for a given domain (e.g. for Facility Management (FM) applications). Currently, this means that applicable processing techniques, semantics and visual analytics methods need to be selected, generated or implemented by human domain experts, which is a rather error-prone, subjective and non-interoperable process. An ontology-driven analytics approach could be used to solve this problem by creating and maintaining a dynamic knowledge structure and utilizing an ontology for inferencing optimal selection of automated processing and analytics techniques of point cloud data.

1 Introduction

The digital representation and analysis of the built environment is gaining paramount importance as humanity progresses on living in the fourth industrial revolution. Maintenance of smart buildings, homes, offices and cities, as well as key infrastructure (e.g., roads, bridges, tunnels, etc), is becoming increasingly dependent on the use of digital counterparts, known as a Digital Twin (DT). DTs enable assessment and forecasting of current and future states of built environment objects they represent, making use of historic and real-time data sources. A key source of data that can be used by DTs for representing and analyzing the current physical state of the built environment are point clouds.

However, since point clouds are ambiguous, they require further processing and semantic enrichment in order to make them useful for analysis and decision making. Semantic enrichment of point clouds has in the past decade focused almost exclusively on using either supervised or unsupervised machine-learning methods for generating and injecting useful semantics. As such, these semantic-enrichment methods for point clouds are often use-case specific for a given domain (e.g., for Facility Management (FM) applications). This means that for every new point cloud dataset, the appropriate semantics generation methods need to be defined, implemented and tuned by human domain experts. This process is usually error prone, subjective and non-interoperable.

The idea of smart building and cities is to anthropomorphize the built environment into intelligent cyber systems that can learn, make decisions and predict future states. In order for a system to learn and make decisions concerning its current and future

state, a cognitive solution is required. Such solutions can rely on knowledge discovery and ontology building and inferencing, which over time can create intelligent and autonomous systems capable of processing key data using optimally configured parameters. An ontology can be used to define the relationships between entities, methods, data, semantics, and processes for a given domain. The use of an ontology could thus enable cognitive-based decision making for DT representations of the built environment.

More importantly, an ontology can be used to set up and maintain a Knowledge Graph (KG) for inferring new decisions based on previously learnt knowledge. Therefore, the optimal configuration of relevant parameters used for semantic enrichment of point clouds could be inferred using KG, without domain-dependent human intervention (e.g., automated selection of processing parameters for semantic segmentation of a point cloud).

2 Foundations and Related Work

Point clouds can be used to visually inspect and assess the current state of the built environment, can help to track construction-related or refurbishment-related changes over time, and can be used as base-data for the generation of as-is and as-built Building Information Models (BIM) [11]. A point-cloud based representation of indoor environments within the context of interactive 3D visualization enables enhanced stakeholder engagement and communication [14].

Since point clouds do not contain any other information besides spatial distribution in 3D space and possibly color and/or intensity values, they need to be enriched with semantics to effectively support the various FM-related tasks. This process can be time-consuming when performed manually, introduces errors in decisions due to incorrect observations. Generation and injection of semantics into point clouds is either based on associating each segmented point cluster with either metric [1], domain expertise [12], or probabilistic deep-learning-based processes and their outputs [2].

An ontology-driven approach for semantic enrichment and analysis of digital built environment representations has recently become a topic of interest in the Architecture, Construction, Engineering, Owner and Occupant (AECOO) research community.

[3] describe the development of a prototypical BIM Semantic Bridge system, that can map Industry Foundation Classes (IFC) semantics to an ontology representation using OWL (Web Ontology Language). They state that the main advantage developing a knowledge base using an ontology is that is allows experts from different AEC domains to access and exchange of knowledge during the design phase of a building (as BIM by default focuses on geometric representation of a building or structure).

[10] advocate the use of a multi-level semantics framework, in which the first level of semantics represents the point cloud data structure, the second represents the connected elements between the point cloud data structure and the spatial context, while the third level connects specific ontologies with the point cloud that is used by

domain experts for performing various semantic queries. All three semantic levels are connected within a feedback loop to a knowledge base (e.g., database system). The knowledge base is updated through analytic, device and domain expertise-based inputs.

[9] describe a fully semantically guided approach for the detection of objects in indoor point clouds. They propose the use of a constantly updated knowledge based that, in turn, is used to select and adapt the most appropriate processing algorithm based on the observed ontology. Sadeghineko et al. (2018) describe the generation of semantically rich BIM models from point cloud data, where each segmented region of a point cloud is given a unique annotation using the Resource Description Framework (RDF) specification, thus enabling relationships between BIM elements to be captured and queried.

2.1 Service-Oriented System Implementation

Approaches for semantic-enrichment, ontology generation and visual analytics can be implemented using an SOA, which can be used for decoupling of hardware and software requirements between the user and the processing system [6].

Such approaches can include methods for 3D annotations of both outdoor and indoor scenes [7], where the semantics can be processed on the server and streamed to the client that may use commodity hardware [4].

The use of SOA can also benefit integration and fusion of existing digital data (e.g., CAD models, floorplans, sensor data, etc) [13].

3 An Ontology-Driven Approach for Point Cloud Analytics

The problem arises when attempting to dynamically assign semantics or adapt processing algorithms for generalized use-cases. The process of semantic enrichment of point clouds is in most current situations unidirectional (e.g., point clouds are semantically enriched at a single time for a single purpose, with the semantics remain valid only for the current version of the point cloud). Therefore, in order to be able to dynamically generate and query semantics for point clouds, a feedback system using a *dynamic knowledge structure* (e.g., a Knowledge Base (KB) [8] or Knowledge Graph (KG) [5]) for semantics generation and updating is required (Figure 1).

3.1 Concept System

A concept system is presented and discussed, based on the SOA design approach (Figure 2).

The proposed system design is made up of six key components: (1) the Knowledge component (for simplicity it is refered to as a Knowledge Base, though it could be a Knowledge Graph), (2) DBMS (Database Management System) component, (3) Algorithm Library, (4) Analytics, the (5) the Processing component and (6) the

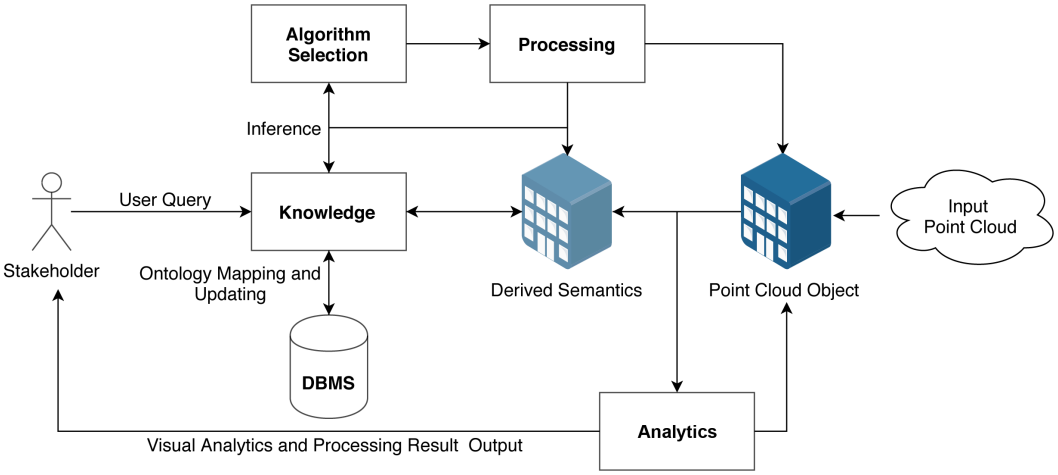

Figure 1: A high-level overview of the main components of the ontology-driven analytics architecture for indoor point clouds. The stakeholder, would interact with the point cloud and be provided with analysis outputs (e.g., visual analytics). User tasks, such as spatial or inventory queries, would be inferred from the *Knowledge* component that is derived and updated using the ontology-drive approach.

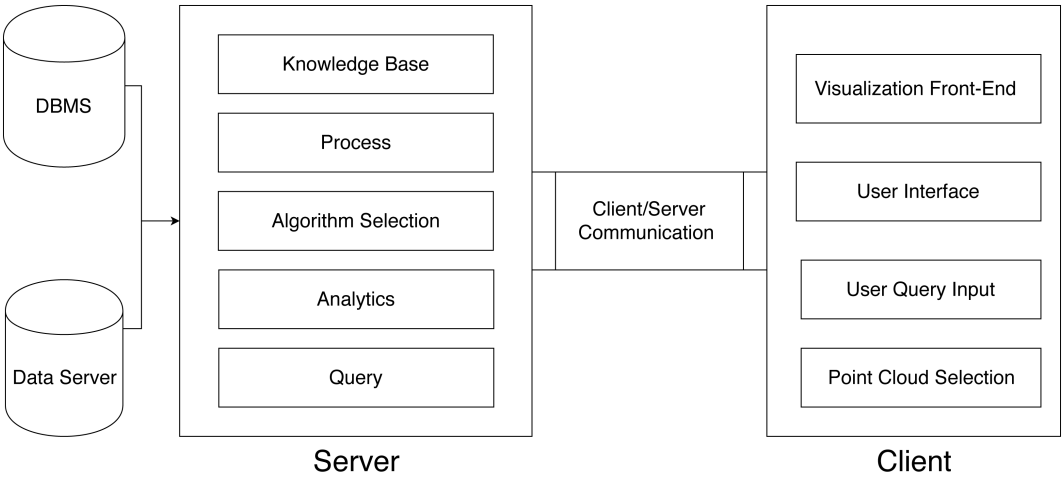

Figure 2: Conceptual design of an SOA. The server is responsible for the processing, inference, DBMS operations, and ontology updating tasks, while the client enables the users to select the initial point cloud they want to perform semantic queries on, to visualize the result of the query, as well as to use and update the ontology.

Query component. The six components work together in order to update the ontology, generate semantics and parameter values for selected processing algorithms. It is assumed that a core ontology is defined, which is then subsequently updated with through the introduction of new semantics, expert knowledge and existing digital documentation.

Initially, the Processing component would be used to filter the input point cloud (e.g., generate normal vectors, remove duplicate points, or sub-sample). The user would then be required to enter a new analysis task – a semantic query, that would be interpreted by the Query component. The Query component would use this query (translated into a machine-readable format), to infer a decision using the ontology represented by Knowledge Base component. In turn, the Knowledge Base component would then utilize the existing semantics objects accessed by the DBMS component to form a decision. In this case the decision would be selecting the appropriate matching algorithm for the required task from the Algorithm Library component.

The point cloud along with the selected algorithm would then be sent back to Processing component, where the result of the processing would semantically-enrich the point cloud. These associated semantics would then be sent to the DBMS component in the form of standardized semantics description object, where they would be once again utilized by the Knowledge Base component next time a new task is initiated by the user.

In an example scenario, we can say that a user wants to find all the chairs in a given office room (Figure 3). A user will load in a point cloud that will be filtered by the Processing component (e.g., normals generation, planar surface segmentation). Next, the user's semantic query for selecting all chairs will be sent to the Knowledge Base component that will compare all existing semantics obtained from semantics objects in the DBMS component, in order to decide what specific algorithm to use for detecting the objects via the Algorithm Library component. For example, it is known that chairs are often found in rooms with desks, and that each office has at least one computer desk, which in turn has specific dimensions for the room object that is evaluated from segmented planar clusters obtained from the initial processing of the point cloud.

In the given example, it would also be known that the point cloud has RGB values, and that it was captured using commodity mobile hardware (so it is a coarse representation of the real-world with a lot of noise). Based on this ontology, the Knowledge Base component could formulate an algorithm suggestion and request the Algorithm Library component for a classification algorithm with specifically tuned parameters based on the derived semantic relations. Once the selected classification algorithm detects and classifies the chairs, the associated resulting semantics that are injected and presented to the user via the Analytics component would also be sent to the DBMS component. This would make the whole ontology-driven system "smarter", as new semantics is introduced with each new user query and new relations are used to update the ontology that is mapped by the Knowledge component the next time it is used.

Other services could implement and advertise other complex computation tasks including, e.g., ML-based classification of point clouds. The proposed Algorithm

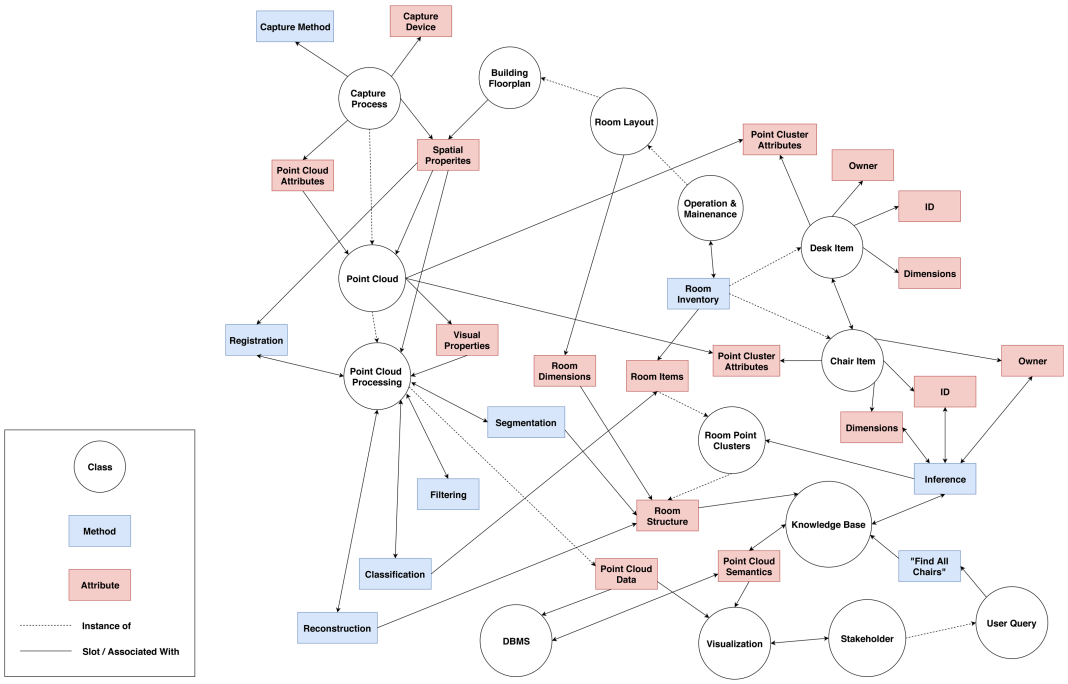

Figure 3: A core ontology example for using a point cloud for an O&M task - based on the concept of a Knowledge Graph. In this case the user query of finding a chair object in the given point cloud is presented as an example.

Library and Processing components will contain all of the related metric and deep learning-based algorithms used for processing, reconstructing, classifying and evaluating point clouds.

Also, the design of the Knowledge Base and DBMS components as separate services allows for running the generation and association of semantics as well as ontology management, on dedicated workstation computers with high-performance computing and storage capabilities, whilst rather lightweight clients can access this functionality through the service interfaces.

4 Conclusions

The proposed future work would attempt to answer a fundamental research question:

"How can an ontology-driven approach be used for creating and maintaining a Knowledge Graph for optimal selection and tuning of semantic enrichment algorithms for point cloud representations of the built environment?"

Furthermore, the formulation of the answer to the above question will require researching and developing additional answers to the following related research questions:

- How can the proposed approach be implemented within a service-oriented software engineering paradigm?

- How can an ontology for a built environment object be defined and formalized?
- How can point cloud representations of the built environment be linked to the defined ontology?
- How can various algorithms and their parameters for point cloud processing and semantic enrichment be selected using the defined ontology?
- What are the key requirements for a cognitive approach to decision making using the defined built environment object ontology?

The overall goal of the proposed future research would be to create a "Google for point clouds" - where a user could simply query the system with a task or a question, and automatically receive an answer (e.g., "Find all furniture in a given room scan", see Figure 4).

Such an approach could be used to benefit decision making within the realms of FM and Real Estate 4.0, and enable advancement of *Smart Building* and *Smart City* paradigms.

Overall, the proposed future work and outcomes would contribute original knowledge to the fields of computer science, software engineering, and Architecture, Construction, Engineering, Owner and Occupant (AECOO) domains.

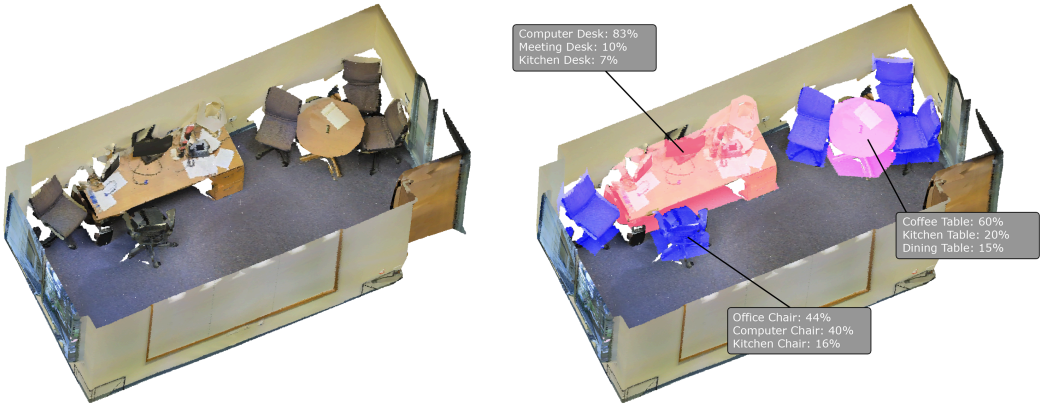

Figure 4: An example of a query task result, where a user wants to find all furniture in a given point cloud representation of a room. The ontology-driven approach would automatically process and classify the point cloud in order to come up with a possible answer, inferred from previous knowledge stored in a dynamic knowledge component i.e., a Knowledge Graph.

5 Related Publications

The following peer-reviewed publications, generated as part of the author's PhD studies, accompany the overview and case study in this report, and provide more in-depth discussion and evaluation of the presented methods and approaches.

1. **Stojanovic, V.**, Richter, R., Döllner, J., and Trapp, M. (2018). "Comparative visualization of BIM geometry and corresponding point clouds". In: *International Journal of Sustainable Development and Planning*, 13, 1, pages 12-23.

2. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2018). "A service-oriented approach for classifying 3D points clouds by example of office furniture classification". In: *Proceedings of the 23$_{rd}$ International ACM Conference on 3D Web Technology (Web3D '18)*, page 9.

3. **Stojanovic, V.**, Trapp, M., Richter, R., Hagedorn, B., and Döllner, J. (2018). "Towards the Generation of Digital Twins for Facility Management Based on 3D Point Clouds". In: *Gorse, C and Neilson, C J (Eds.), Proceedings 34th Annual ARCOM Conference*. Association of Researchers in Construction Management, pages 270–279.

4. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "Generation of Approximate 2D and 3D Floor Plans from 3D Point Clouds". In: *Proceedings of the 14th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications – Volume 1: GRAPP*, pages 177-184.

5. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "Classification of Indoor Point Clouds Using Multiviews". In: *Web3D '19: The 24th International Conference on 3D Web Technology (Web3D '19)*, page 9.

6. **Stojanovic, V.**, Trapp, M., Richter, R., Hagedorn, B., and Döllner, J. (2019). "Semantic Enrichment of Indoor Point Clouds: An Overview of Progress towards Digital Twinning". In: *Architecture in the Age of the 4th Industrial Revolution - Proceedings of the 37th eCAADe and 23rd SIGraDi Conference - Volume 2*. pages 809-818.

7. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "Service-Oriented Semantic Enrichment of Indoor Point Clouds using Multiview-Based Classification". In: *Graphical Models*. Elsevier.

8. **Stojanovic, V.**, Trapp, M., Hagedorn, B., Klimke, J., Richter, R., and Döllner, J. (2019). "Sensor Data Visualization for Indoor Point Clouds". In: *Advances in Cartography and GIScience of the ICA*, 2.

9. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2019). "A Service-oriented Indoor Point Cloud Processing Pipeline". In: *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42, pages 339-346.

10. Isailović, D., **Stojanovic, V.**, Trapp, M., Richter, R., Hajdin, R., and Döllner, J. (2020). "Bridge Damage: Detection, IFC-Based Semantic Enrichment and Visualization". In: *Automation in Construction*. Elsevier.

11. **Stojanovic, V.**, Trapp, M., Richter, R., and Döllner, J. (2020). "Comparison of Deep-Learning Classification Approaches for Indoor Point Clouds". In: *Publikationen der DGPF, Band 29, 2020*. pages 437-447.

12. **Stojanovic, V.**, Hagedorn, B., Trapp, M., and Döllner, J. (2020). "Ontology-Driven Analytics for Indoor Point Clouds". In: *RE: Anthropocene, Design in the Age of Humans - Proceedings of the 25th CAADRIA Conference – Volume 2*. pages 537-546.

6 Acknowledgments

This work was funded by the Research School on Service-Oriented Systems Engineering of the Hasso Plattner Institute, Faculty of Digital Engineering, University of Potsdam.

References

[1] I. Armeni, O. Sener, A. R. Zamir, H. Jiang, I. Brilakis, M. Fischer, and S. Savarese. "3d semantic parsing of large-scale indoor spaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pages 1534–1543.

[2] E. Che, J. Jung, and M. J. Olsen. "Object recognition, segmentation, and classification of mobile laser scanning point clouds: A state of the art review". In: *Sensors* 19.4 (2019), page 810.

[3] S. Cursi, D. Simeone, and U. M. Coraglia. "An ontology-based platform for BIM semantic enrichment". In: *Proceedings of the 35th eCAADe Conference*. Volume 2. 2017, pages 649–656.

[4] J. Doellner, B. Hagedorn, and J. Klimke. "Server-based rendering of large 3D scenes for mobile devices using G-buffer cube maps". In: *Proceedings of the 17th International Conference on 3D Web Technology*. 2012, pages 97–100.

[5] L. Ehrlinger and W. Wöß. "Towards a Definition of Knowledge Graphs." In: *SEMANTiCS (Posters, Demos, SuCCESS)* 48 (2016), pages 1–4.

[6] D. Hildebrandt, J. Klimke, B. Hagedorn, and J. Döllner. "Service-oriented interactive 3D visualization of massive 3D city models on thin clients". In: *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*. 2011, page 1.

[7] J. Klimke and J. Döllner. "Geospatial Annotations for 3D Environments and their WFS-based Implementation". In: *Geospatial Thinking*. Springer, 2010, pages 379–397.

[8] S. Krishna. *Introduction to database and knowledge-base systems*. Volume 28. World Scientific, 1992.

[9] J.-J. Ponciano, A. Trémeau, and F. Boochs. "Automatic Detection of Objects in 3D Point Clouds Based on Exclusively Semantic Guided Processes". In: *ISPRS International Journal of Geo-Information* 8.10 (2019), page 442.

[10] F. Poux, R. Neuville, P. Hallot, and R. Billen. "Model for reasoning from semantically rich point cloud data". In: *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 4 (2017), pages 107–115.

[11] T. Qu and W. Sun. "Usage of 3D point cloud data in BIM (building information modelling): Current applications and challenges". In: *Journal of Civil Engineering and Architecture* 9.11 (2015), pages 1269–1278.

[12] R. Sacks, A. Kedar, A. Borrmann, L. Ma, I. Brilakis, P. Hüthwohl, S. Daum, U. Kattel, R. Yosef, T. Liebich, et al. "SeeBridge as next generation bridge inspection: overview, information delivery manual and model view definition". In: *Automation in Construction* 90 (2018), pages 134–145.

[13] J. Teizer, M. Wolf, O. Golovina, M. Perschewski, M. Propach, M. Neges, and M. König. "Internet of Things (IoT) for integrating environmental and localization data in Building Information Modeling (BIM)". In: *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction*. Volume 34. IAARC Publications. 2017.

[14] J. Xu, K. Chen, F. Xue, and W. Lu. "3D point cloud data enabled facility management: A critical review". In: *The 23rd International Symposium on Advancement of Construction Management and Real Estate, CRIOCM2018*. 2018.

Current Technical Reports from the Hasso Plattner Institute

Vol.	ISBN	Title	Authors/Editors
137	978-3-86956-505-7	**Language and tool support for 3D crochet patterns : virtual crochet with a graph structure**	Klara Seitz, Jens Lincke, Patrick Rein, Robert Hirschfeld
136	978-3-86956-504-0	**An individual-centered approach to visualize people's opinions and demographic information**	Wanda Baltzer, Theresa Hradilak, Lara Pfennigschmidt, Luc Maurice Prestin, Moritz Spranger, Simon Stadlinger, Leo Wendt, Jens Lincke, Patrick Rein, Luke Church, Robert Hirschfeld
135	978-3-86956-503-3	**Fast packrat parsing in a live programming environment : improving left-recursion in parsing expression grammars**	Friedrich Eichenroth, Patrick Rein, Robert Hirschfeld
134	978-3-86956-502-6	**Interval probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese
133	978-3-86956-501-9	**Compositional analysis of probabilistic timed graph transformation systems**	Maria Maximova, Sven Schneider, Holger Giese
132	978-3-86956-482-1	**SandBlocks : Integration visueller und textueller Programmelemente in Live-Programmiersysteme**	Leon Bein, Tom Braun, Björn Daase, Elina Emsbach, Leon Matthes, Maximilian Stiede, Marcel Taeumel, Toni Mattis, Stefan Ramson, Patrick Rein, Robert Hirschfeld, Jens Mönig
131	978-3-86956-481-4	**Was macht das Hasso-Plattner-Institut für Digital Engineering zu einer Besonderheit? : Festrede zum Anlass des 20-jährigen Bestehens des Hasso-Plattner-Instituts**	August-Wilhelm Scheer

ISBN 978-3-86956-513-2
ISSN 1613-5652