Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Elias Mohammed Elfarri

Digital Twin of a Building Powered by Artificial Intelligence and Demonstrated in Virtual Reality

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2022

NTNU
Norwegian University of Science and Technology

Elias Mohammed Elfarri

Digital Twin of a Building Powered by Artificial Intelligence and Demonstrated in Virtual Reality

Master's thesis in Cybernetics and Robotics
Supervisor: Adil Rasheed
June 2022

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

NTNU
Norwegian University of
Science and Technology

Digital Twin of a Building Powered by Artificial Intelligence and Demonstrated in Virtual Reality

Elias Mohammed Elfarri

NTNU
Norwegian University of Science and Technology

Department of Engineering Cybernetics — NTNU 2022,
Faculty of Information Technology and Electrical Engineering

Abstract

A digital twin is defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making. Unfortunately, the term remains vague and says little about its capability. Recently, the concept of capability level has been introduced to address this issue. The concept basically states that a digital twin, based on its capability, can be categorized on a scale from zero to five, referred to as standalone, descriptive, diagnostic, predictive, prescriptive, and autonomous, respectively. The current thesis aims to physically demonstrate the concept with a built environment as a use case. It does so by combining the strengths of advanced sensor technologies, artificial intelligence, and virtual reality. The end products of this thesis are a modular and extendable digital twin based on the capability level concept, a minimum value product consisting of a virtual reality headset, and recommendations for further improvements of the work. This thesis is also accompanied with a video recorded in the virtual reality environment linked in the relevant chapters. A live demonstration in virtual reality can be provided on request.

Sammendrag

En digital tvilling er definert som en virtuell representasjon av et fysisk objekt, oppnådd gjennom data og simulatorer for sanntidsprediksjon, optimalisering, overvåking, kontroll og forbedret beslutningstaking. Dessverre er begrepet fortsatt vagt og sier lite om den digitale tvillingens evner. Nylig har evnenivå rammeverket for digitale tvillinger blitt introdusert for å løse dette problemet. Rammeverket sier i utgangspunktet at en digital tvilling, basert på dens evne, kan kategoriseres på en skala fra null til fem, referert til som henholdsvis frittstående (standalone), beskrivende (descriptive), diagnostisk (diagnostic), prediktiv (predictive), preskriptiv (prescriptive) og autonom (autonomous). Denne masteroppgaven tar sikte på å demonstrere evnenivå rammeverket for digitale tvillinger ved bruk av et eksisterende bygg som et use case. Dette gjøres ved å kombinere styrkene til avanserte sensorteknologier, kunstig intelligens og virtuell virkelighet (VR). Sluttproduktene av denne oppgaven er en modulær og skalerbar digital tvilling basert på evnenivå rammeverket, et minimumsverdiprodukt bestående av et VR-headset, og anbefalinger for ytterligere forbedringer av arbeidet. Et video opptak som demonstrerer evnenivåene følger også med, tatt opp i VR-miljøet. En fysisk demonstrasjon i VR kan også vises etter ønske.

Preface

I would like to thank my supervisor, Adil Rasheed, for his contribution of knowledge, guidance, resources, and time. Also, for providing his house, in the pursuit of science, to be portrayed as an example of a digital twin application.

Additionally, I would like to thank my parents, Leila Eljeroudi and Abdelkarim Elfarri, my significant other Ive Johanne Tøreki Lundsveen, as well as my extended family. Without their love and support, I wouldn't have been able to dedicate as much time and effort to my studies.

Much of the setup developed in this project would not be possible had it not been for companies' available resources. For instance, Disruptive Technologies provided us with over 40 temperature, proximity, humidity, and water detection sensors. These sensors were a crucial part of showcasing what is possible to do in the next generation of digital twins for the built environment.

I would also like to thank the contribution of NTNU's technology transfer office (TTO) for helping with guidance concerning the innovative potential of the work. With this in mind, I pursued and demonstrated aspects of digital twins that existing solutions do not contain, resulting in a minimum value product (MVP). Finally, in the same spirit, I would like to thank Spark* NTNU for nominating me to pitch my work to a panel of investors.

Last but not least, I would like to thank the Norwegian Society of Graduate Technical and Scientific Professionals (Tekna) for providing me with a stipend to raise the quality of my work. The fund was invested into an Oculus Quest 2 virtual reality headset to demonstrate the VR aspect of the project.

Trondheim, June 2022
Elias Mohammed Elfarri

Contents

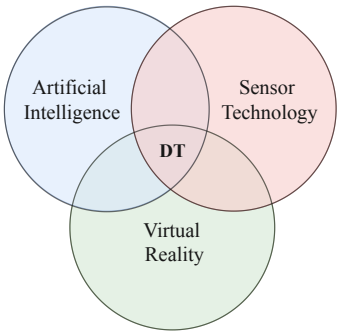
List of Figures

List of Tables

Nomenclature

AI	Artificial Intelligence
ANN	Artificial Neural Network
$ARIMA$	Autoregressive Integrated Moving Average
BIM	Building Information Modeling
CAD	Computer Aided Design
DDM	Data-driven Modeling
DT	Digital Twin
GRU	Gated Recurrent Unit
$HDRP$	High Definition Render Pipeline
$HVAC$	Heating, Ventilation and Air Conditioning
IoT	Internet of Things
$LSTM$	Long Short Term Memory
ML	Machine Learning
MSE	Mean Squared Error
PBM	Physics-based Modeling
PCA	Principal Component Analysis
$RMSE$	Root Mean Squared Error
$RMSLE$	Root Mean Squared Log Error
RNN	Recurrent Neural Network
$UBCF$	User Based Collaborative Filtering
UI	User Interface
URP	Universal Render Pipeline
VR	Virtual Reality

1. Introduction



The need for digitalization has achieved even higher importance during the ongoing corona pandemic. The ability to control physical assets remotely under the "work from home" situation is more relevant than ever. As the world is slowly but surely recovering from this unfortunate situation, one can expect to see a rapid expansion of technologies like digital twins. The current work aims at demonstrating how the immense potential of artificial intelligence, sensor technologies, and virtual reality, can be fused together to build highly capable digital twins.

In this chapter a short description of digital twins and their capability levels is provided. This is followed by the motivation for choosing built environment as the application area. Relevant work and industrial state-of-the-art are presented next, leading to research objectives and questions which guide the research conducted in this master's thesis. Lastly, the structure of the thesis is presented.

1.1. Digital Twins and their Capability Levels

A digital twin (DT) is defined as a virtual representation of a physical asset enabled through data and simulators for real-time prediction, optimization, monitoring, controlling, and improved decision making [1].

The concept can be described using Figure 1.1. On the top right side of the figure, we have the physical asset of which we want to build a DT. The asset is equipped with a diverse class of sensors that provides bigdata in real-time. However, this data has a very coarse spatio-temporal resolution and does not describe the future state of the asset. Therefore, to complement the measurement data, models are utilized to bring physical realism to the digital representation of the asset. Provided that the same information can be gained from the DT as can be from the physical asset, it can be utilized for more informed decision-making and optimal control of the asset. All the green arrows in the figure represent real-time data exchange and analysis. However, one might be interested in risk assessment, what if ? analysis, uncertainty quantification, and process optimization. These can be realized by running the DT in an offline setting for scenario analysis. The concept is then known as digital siblings. The box and arrows in the grey represent the digital sibling. Additionally, the DT predictions can be archived during the

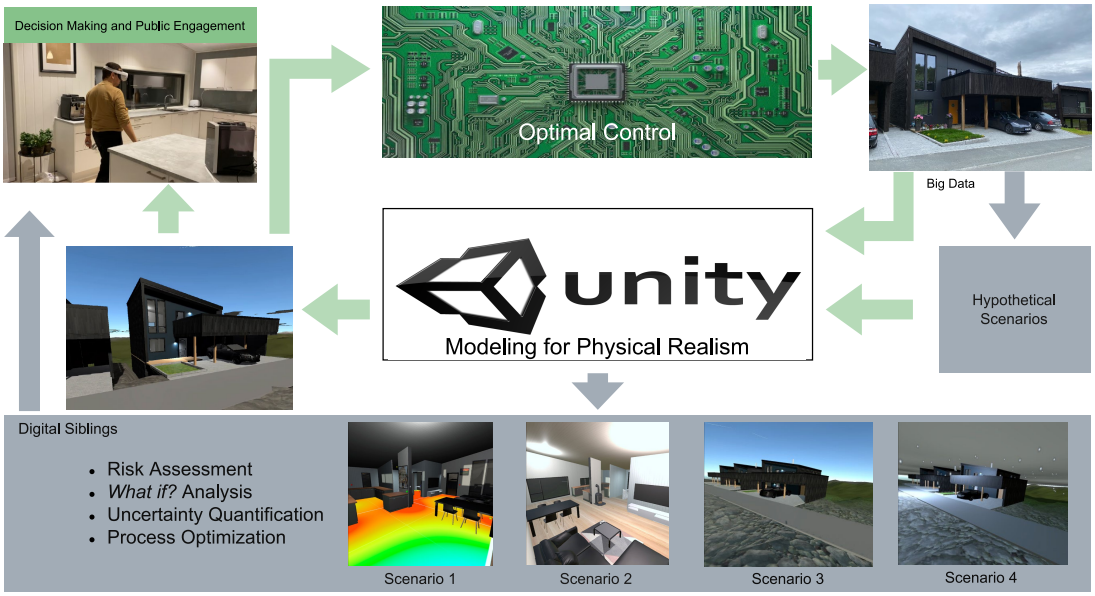


Figure 1.1.: Example of a fully fledged digital twin pipeline.

lifetime of the asset and can be used for designing a next generation of assets, in which the concept is referred to as digital threads [2].

1.1.1. Capability Levels of Digital Twins

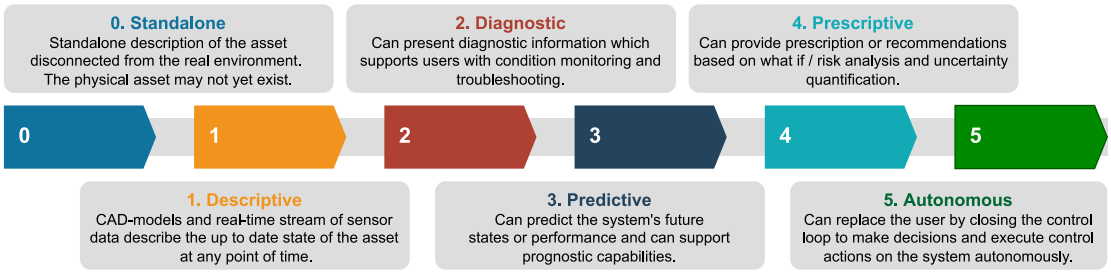


Figure 1.2.: Description of capability levels of a digital twin.

The authors in [3] present a DT capability level scale adapted from a DNV GL report that divides a DT into six distinct levels [4]. These levels are 0-Standalone, 1-Descriptive, 2-Diagnostic, 3-Predictive, 4-Prescriptive and 5-Autonomous (Figure 1.2). Standalone DT can exist even before the asset is built and can consist of solid models. When the asset is in place and is equipped with sensors, data can be streamed in real-time to create a descriptive DT, giving more insight into the state of the asset. When analytics tools are applied to the incoming data stream to diagnose anomalies, the DT advances to a diagnostic level. At the first three levels, the DT can provide information/insight only about the past and present.

However, a predictive DT can describe the future state of the asset. Using the predictive DT, one can do scenario analysis to provide recommendations to push the asset to the desired state. This is then referred to as prescriptive level. Lastly, the asset updates the DT at the autonomous level, and the DT controls the asset autonomously. We will refer to this setting as the capability level framework for DTs from now onwards.

The concept of DTs has been in use in different application domains with varying levels of detail. For example, in the context of buildings, Building Information Models (BIM) which is a specialized modeling tool that provides access to actual 3D objects and their composition to represent real-world components, i.e. walls, doors, and windows, for 3D modeling of a house, are considered DT. Additionally, the buildings can be equipped with Internet of Things (IoT) devices to improve physical realism. However, with the evolved definition of DTs, groundbreaking work in artificial intelligence, sensor technologies, and virtual reality, as we will see, makes the existing state-of-the-art DTs of buildings appear very primitive.

1.2. Motivation

Although the focus of this work is to demonstrate how a general DT of any asset can be constructed and what values can be generated from it, we focus on buildings for the following main reasons:

- *Environmental benefits:* The Norwegian government has an ambitious goal of reducing its carbon footprint by 50% to 55% within 2030 as part of the Paris climate agreement [5]. Newly constructed commercial buildings are one of the biggest contributors to carbon emissions within the building industry. It has been shown in pilot projects that technologies that help make modern buildings more energy efficient could indeed reduce the building's carbon emissions by half [6]. Big state-owned companies in Norway such as Statsbygg, are extremely forward-leaning and positive to adapting technologies and digitalization that prove environmentally beneficial. Therefore demonstrating how to generate an advanced DT, is a contribution towards improving efficiency of buildings.

- *Immature field with huge potential:* Construction projects can be broken down into four different phases; Design, pre-construction, construction and "management, operation and maintenance" [7]. BIM models benefit architects and engineers during the design and pre-construction phases. However, it is known to only be partly used during the construction and unused during management, operation, and maintenance [7] phases. Interestingly it is the last phase that is the most prolonged duration of the entire lifecycle of any building. Most of the DTs of buildings as they exist today in the construction industry are obsolete during the final phase and raises questions as to how one could bring the existing 3D model of the building into this phase of the asset's lifecycle [8, 9, 10].

- *Focus on digitalization of the built environment:* For highly advanced DTs the construction industry is the most mature space for implementing such a technology. The reason is because there has been a digitalization movement in the construction industry in Norway to create BIM models of all modern constructions and big commercial buildings [7]. This movement does indeed include existing buildings, and thus many big buildings have 3D models as well as being equipped with massive amount of sensor data in the form of temperature, humidity, water leakage and so on. The most recent example of an existing building being supported by both a BIM model and sensors is the Opera House of Oslo [11]. These resources make big commercial buildings easy targets for creating a high capability level DT in the future.

- *Ease of data collection and usage:* For the purpose of a demonstration project, it is relatively easier to use a private house, equip it with a variety of sensors, collect huge amount of data, and still have full control over the data privacy and security issues. This way we can focus only on the technical and scientific part of the technologies. Of course in order to scale the technology or to use in other application areas, issues related to data privacy, ownership, and security will have to be handled [12, 13, 14, 15].

Based on the need for this technology there is plenty of motivation behind trying to demonstrate the technological capabilities of a DT for buildings, and establishing a state-of-the-art DT of a building with unprecedented level of granularity.

1.3. Relevant Work

Table 1.1.: Comparing Master's thesis setup with industry DT services.

Comparing Digital Twin Solutions					
Capability	Standalone	Descriptive	Diagnostic	Predictive	Prescriptive
Thesis Setup	X	X	X	X	X
Matterport	X	-	-	-	-
Openspace	X	-	-	-	-
Revit	X	-	-	-	-
Invicara	X	X	-	-	-
TwinView	X	X	-	X	-

Within the field of DTs there is no consensus as to what qualifies as a DT application [10]. Organizations and sectors operate with different definitions of DTs that are simply too vague and generic to provide any indication of the current capabilities of the DT [16, 17, 18]. Furthermore within the built environment, a BIM of a building relates somehow to the DT of the asset, but many still struggle to see the liaison between the two [19]. The capability level framework described earlier gives a way to compare different existing DT solutions available in the built

environment market. A brief overview of the capability of existing solutions in the market is presented in Table 1.1.

Table 1.1 shows a lack of high capability level DTs in the industry. The connection between the DT capability level and the service that each company provides depends on what the focus of the provider is. For instance Matterport and Openspace are virtual tour services that focus on providing 360 degrees photogrammetry for building management [20, 21], and therefore only qualify as a standalone DT service. Note that there are many other similar standalone services in the market [22, 23]. Revit is a BIM modelling software and is therefore only limited to the construction of 3D models of a building [24]. Invicara and TwinView has an ambition of allowing customers to combine BIM models with IoT sensor data integration qualifying both companies for descriptive DT, with the addition of Twinview providing predictive DT capabilities [25, 26]. None of the above mentioned applications qualify as diagnostic or prescriptive DTs, which are properly covered by this thesis in Chapter 4 and Chapter 6. For companies such as TwinView or Invicara that provide services for maintenance and operations of buildings, reaching a high capability level DT might prove extremely useful for their customers.

To gain a better insight into the state-of-the-art of DT technologies within the Norwegian industry, a survey was conducted in the ongoing NorthWind project [27]. It turned out that the diagnostic and prescriptive capabilities are the ones the industries are most excited about [28]. However, they can already start generating value from standalone capabilities too. When it comes to automation, predictive might be most important. It was not clear if the world is still ready for autonomous DTs. Furthermore, three major challenges related to data (privacy, ownership and security), models (utilization of their full potential and legacy knowledge), and industrial acceptance were identified.

1.4. Research Objective and Research Questions

The challenges mentioned above have guided the setting up of the research objectives, and questions which are presented next.

1.4.1. Main Objective and Milestones

Primary objective:

- The primary objective of this work is to combine the power of Artificial Intelligence, advanced Sensor Technology, and Virtual Reality to demonstrate a fully functional DT of a real house.

Secondary objectives:

- To adapt the definitions of DT and its capability levels in the context of built environment and demonstrate it through a working prototype consisting of a VR headset.

- Demonstrate the standalone, descriptive, diagnostic, predictive, and prescriptive capabilities of DT.
- Provide insight into what needs to be done to build fully autonomous DTs and port the technology to other fields.

1.4.2. Research Questions

In order to realize the objectives mentioned in the previous section, the following research questions will be answered:

- How should one approach constructing a high capability level DT of a house?
- Which software workflow is most effective in terms of creating a 3D model Standalone DT of a house, that also enables a gateway to high capability DT?
- What types of states can be represented in a descriptive DT capability level of a house, and how can the 3D model be informed of these states in real-time?
- What types of diagnostic information is it possible to display such that the DT supports condition monitoring and troubleshooting?
- What types of data-driven and physics-driven forecasting models can be built for a predictive DT given the sensor data in the house, to support prognostic capabilities?
- What types of recommendation systems can be built to support user recommendations, what-if analysis and uncertainty quantification for prescriptive capabilities?

1.5. Outline of the Thesis

The report is outlined such that each capability level is presented in its relevant chapter. Chapter 2 presents the theory and detailed methodology for generating a standalone DT, followed by each subsequent Chapter, from 3 to 6, sequentially presenting relevant theory and methodology for descriptive DT, diagnostic DT, predictive DT and prescriptive DT respectively. Chapter 7 gives a short conceptual introduction to autonomous DT. Given the nature of the thesis, the results for each capability level are included in the respective chapters as demonstrations. However, in Chapter 8 a more involved discussions of those results are presented. Finally, Chapter 9 concludes the work and presents some future research directions. This thesis is also accompanied with a video recorded in the virtual reality environment linked in the relevant chapters. A live demonstration in VR can be provided on request.

2. Standalone Digital Twin

0 A house buyer is interested in buying a property that does not yet exist. A real estate company provides the buyer with a tour of the construction site and many sketches showing the 2D plan of different levels of the house to be constructed. Unfortunately, the documents presented give absolutely no idea of how the neighborhood would look. However, the buyer commits to buying the house based on the little information provided. Later on, the buyer is invited for customization of the house. Once again, the buyer makes choices without insight into how the customization would look and feel in the real world. It is evident that in such a situation, a virtual tour using a standalone DT could make the decision more informed and communication between the seller and buyer more effective.

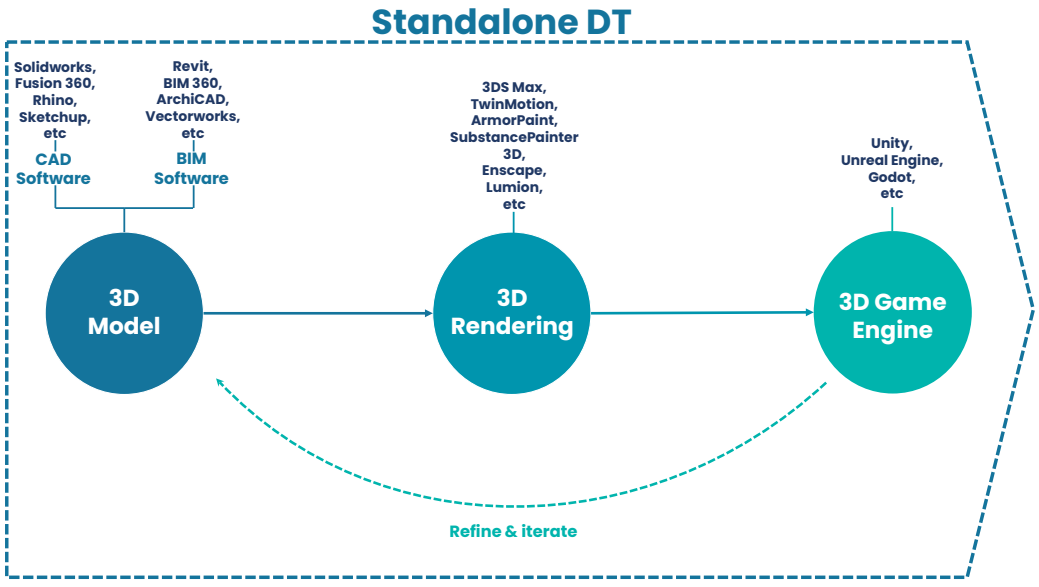


Figure 2.1.: Description of concrete workflow to produce a standalone DT capability level for a building using a combination of 3D modeling, 3D rendering and 3D Game Engine software. Note that one should aim to choose between one of the mentioned programs for each part of the pipeline to have an efficient workflow. In the specialization project the combination of Revit, 3DS Max and Unity were found to give the best workflow results.

This chapter will now describe the setting up of a standalone DT, and demonstrate

its utility.

As mentioned earlier a standalone DT is a virtual description of the asset, disconnected from its physical counterpart. The physical asset might not even exist at the inception of a standalone DT. This capability of a DT is the original definition coined two decades ago [29]. A standalone DT can give a feel of the asset to the stakeholders, and enable them to make informed decisions.

In the infancy of a construction project, an architectural 2D plan of buildings are made. In recent times the 3D BIM models are also becoming more popular. These model hold specific information about the buildings such as materials, characteristics and cost. The BIM models are rarely used on the already constructed buildings, and in many cases they might never be used after the design and construction phase is accomplished [7]. In the current age of digitalization there is great motivation and incentive to develop and identify technologies that makes it attractive for different stakeholders to utilize the full potential of the models during the entire lifecycle of the buildings. With this in mind, a three step methodology as visualized in Figure 2.1 is proposed [2], to enable this technology to evolve beyond the standalone DT with a goal of making it more accessible.

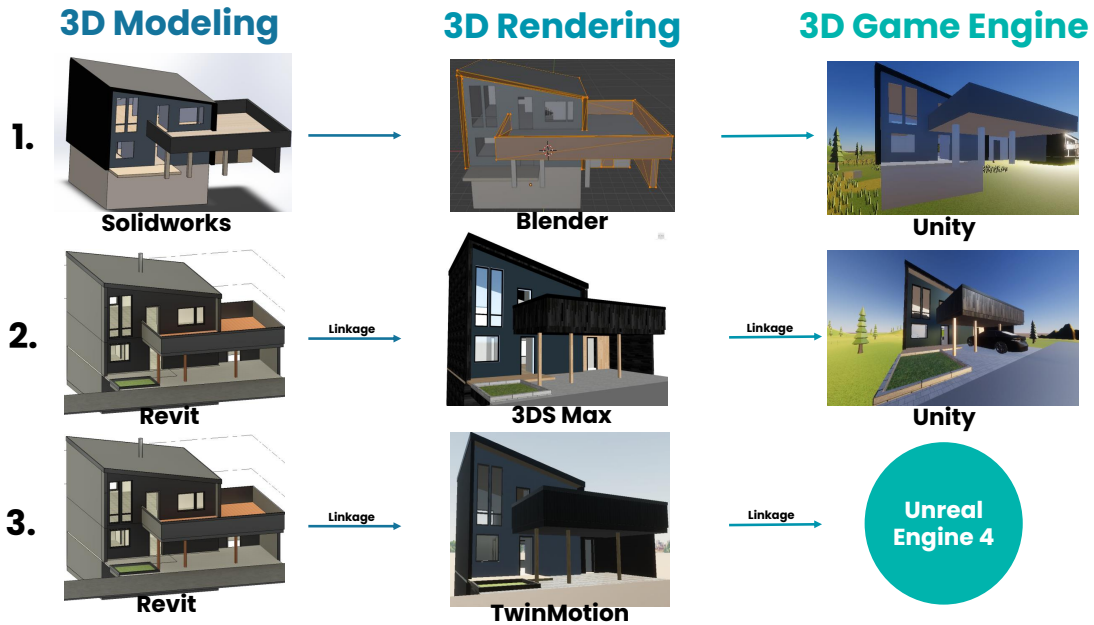


Figure 2.2.: Attempted workflows in the specialization project. The final workflow used for the master thesis is workflow two; Revit, 3DS Max and Unity.

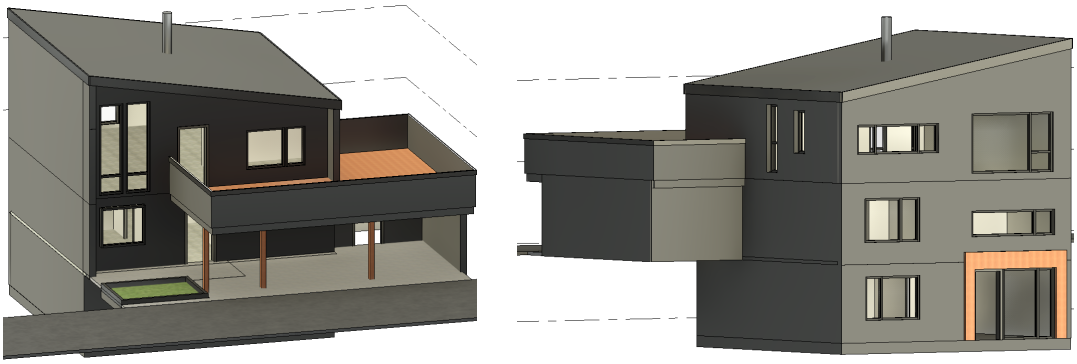
The proposed methodology in Figure 2.1 describes a workflow that ultimately enables a low complexity standalone DT as a starting point for developing more capable DTs. In the current work, this is achieved by bringing the 3D model into a 3D Game Engine environment that hosts a variety of technologies to make games, such as physics-based simulations, API data connectivity, and a scriptable coding environment. Such software also fits the purpose of generating a high capability

level DT. This workflow also identifies specific software compatibilities that allow for linkages [2], meaning that any changes made in the original 3D model in the 3D modeling software will propagate the changes into the 3D rendering as well as the 3D Game Engine software without the need for any additional effort. A cross-platform communication enables a workflow pipeline that focuses on refining important details and limits time spent worrying about inconsistencies across software platforms [30, 31].

We now give a brief overview of the structure of this chapter. Section 2.1 to 2.3 is about the setup and methodology of the standalone DT developed in this project. Section 2.4 demonstrates results related to the standalone DT.

2.1. 3D Modeling

The term 3D modeling is self-explanatory and is the concept of constructing a 3-dimensional model to represent a specific object [2]. For the specialization project, two different 3D modeling software were utilized. To begin with, Solidworks was used to create a CAD model. However, after much experimentation, it was concluded that Autodesk Revit is better suited for creating a 3D model of a building as it supports the best workflow pipeline. Therefore, Autodesk Revit version 2022 was used to generate the model in Figure 2.3.



(a) Front view. (b) Back view.

Figure 2.3.: BIM model designed in Autodesk Revit.

The methodology for creating a 3D model of the house is based on drawing on top of the 2D floor plans of the house directly. Autodesk Revit is simple to get proficient with and allows to handle each floor separately, ensuring that each floor is extruded on top of the other in the correct order. The windows and door frames were also added simultaneously, as they can be identified within the 2D floor plan. Finally, some finishing touches were done, such as adding inbuilt Revit textures to different house parts. Note that while the texturing would be redone in a 3D renderer, it is essential to give different parts of the house the correct materials. This is because the external rendering software will differentiate between the house sections based on the textures given in Revit.

It is important to note that multiple excursions to the actual building were done to verify the dimensions of uncertain parts, such as the elevation of windows, the thickness of door frames, and the walls. Furthermore, the homeowner has redone the basement, such that the original floor plans did not reflect the rework. For that reason, the basement needed to be physically measured to be accurately represented.

2.2. 3D Rendering

(a) Textures.

(b) Rendered house.

Figure 2.4.: Some patches of wood, grass, stone and concrete used to texture the uv-maps of the 3D house model in 3DS Max.

3D rendering is a computer graphics process in which surfaces of a 3D model are overlayed with textures such that the 3D object achieves photo-realism [32]. 3D rendering can be broken down into three steps; visual texturing, lighting, and detailing. Visual textures refer to the visual perception of a spatial surface with a variety of details such as color, orientation, intensity, size, shape, and density [33, 34]. Lighting, reflection, and shadows can all be generated in a 3D rendering software, and the way a material absorbs light can be integrated into a texture. Detailing is the last step and requires a designer to carefully sculpt wear and tear, imperfections, dents, and other details into the surfaces, giving the model a more lifelike impression. For the textures seen in Figure 2.4 to be read in the Unity Game Engine or any external environment, it is essential to add the textures to the uv-map of the 3D model. Uv-mapping is projecting a 2D image onto a 3D surface, where the "uv" part refers to the axes of the image projection. Most 3D rendering software supports manipulating the uv-map property of 3D models. Note that uv-mapping cannot be achieved on a 3D model that has non-orientable surfaces [2].

Three different 3D renderers were experimented with during the specialization project: Blender, Twinmotion, and Autodesk 3DS Max. Arguably Twinmotion provided the best texturing job for creating a fast high-quality house design [35]. However, the caveat was that it required to be used with the Unreal Game Engine as it was not compatible with the Unity Game Engine used in this project. On the

other hand, Autodesk 3DS Max comes from the same software suite as Autodesk Revit, making the project linkable. Version 2022 of the software was used in this setup. It is worth noting that after importing a Revit model into 3DS Max, one has to choose "Combine By Revit Material." This allows the 3D model to be split up into parts based on how they were textured in Revit. Finally, the model inside 3DS Max has to be converted in the "Scene Converter" such that it is compatible with Unity.

All textures used in 3DS Max to achieve the final results seen in Figure 2.4 are retrieved from external sources. Architextures is a library of high quality textures created for architectural 3D models. Textures on this site can be used for educational purposes for free [36]. There are many other open-sourced textures created by other authors that could be equally useful in the texturing process [2, 37, 38, 39]. These textures are imported to 3DS Max and dragged onto the uv-maps of walls, floors, etc. to give the building its final look.

2.3. 3D Game Engine

(a) Virtual house. (b) Real house.

Figure 2.5.: Front display of the target house as it stands in Unity. From being built in Revit and then textured in 3DS Max and finally integrated into Unity compared to the real asset. The angles and position of the cameras are not the same for that reason some parts of the virtual model might look a bit larger or smaller in this comparison. They are nevertheless dimensionally aligned based on the 2D floor plans.

A 3D Game Engine is a development environment with built-in tools for developing interactive 3D experiences and games rapidly. The Unity Game Engine supports a C# .NET programming environment, 3D rendering, accurate physics engines, and optimized features that allow developers to create games faster. A Game Engine works very well for the use case of creating a high capability level DT and

has a history of being used for creating DT experiences for academic purposes [40].

While Revit and 3DS Max are linked together out of the box, making Unity linkable with 3DS Max requires some additional setup. Unity supports linkage with 3DS Max through a middleware called FBX Exporter [31]. This is a joint effort by Unity and Autodesk to make it possible to have workflow between their programs. Once all programs are installed, the FBX Exporter plugin is added to the Unity project. When the 3DS Max model is exported into Unity, it is crucial to do some final setup by selecting the 3D model in Unity, going under "Materials" in the Unity inspector, and checking off "Use External Materials" under "Location". This allows the textures from 3DS Max to be imported. Finally, click on "Generate Colliders" under the model to make Unity treat the house as a physical object. This will make it interact with other physical objects imported into Unity in compliance with Newtonian mechanics.

2.3.1. Unity Setup

Unity Version 2020.3.16f1 is used with the Universal Render Pipeline version 10.5.1 (URP) for this project. Unity can be set up with one of three different graphics rendering pipelines; the standard pipeline is for general-purpose use and is limited in features and options. URP is customizable and gives optimized graphics across a wide range of platforms. High Definition Render Pipeline (HDRP) focuses on creating stunning high-end graphics that are only supported by the cutting-edge graphics card and machinery [41]. URP works best here as the goal is to run on low-end android VR platforms such as the Oculus Quest 2, to achieve diagnostic DT capabilities down the line.

Additionally the project is setup with the following add-on packages from the Unity Asset Store:

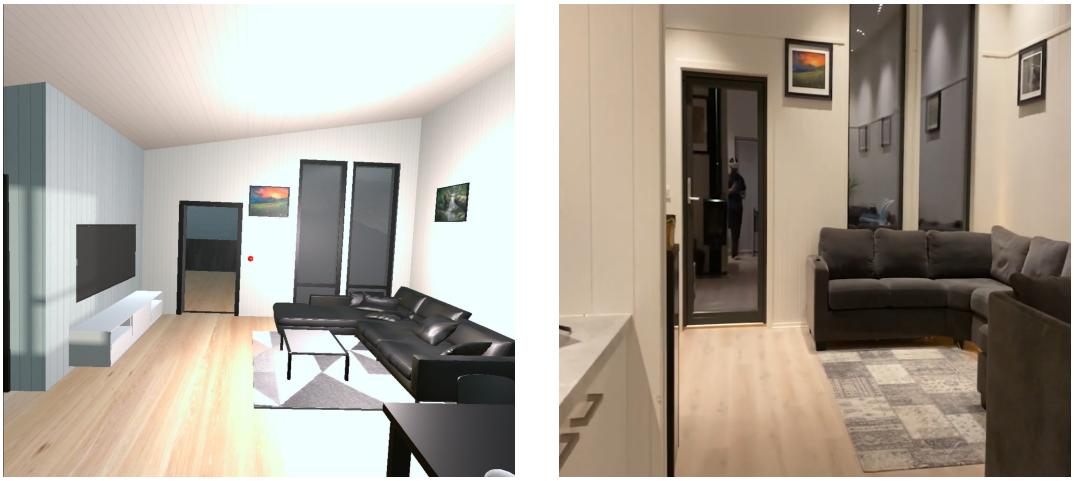
- FBX Exporter version 4.1.1
- Oculus XR Plugin 1.9.1
- Post Processing 3.1.1
- Test Framework 1.1.27
- TextMeshPro 3.0.6
- Timeline 1.5.2
- Toolchain Win Linux x64 1.0.0
- Unity UI 1.0.0
- XR Legacy Input Helpers 2.1.8
- XR Plugin Management 4.1.0

- NuGet 3.0.2

Each of which supports some particular part of the project, for example the Oculus XR Plugin makes it easier to integrate the application in Unity with the Oculus Quest 2 with minimal setup. While NuGet is used to manage .NET specific libraries. A new Unity project comes with very few features, allowing developers to only include libraries that are necessary for the program one is writing with the added freedom of writing one's own libraries if that suits the development environment better.

2.3.2. 3D Models

Unity can host a variety of 3D models in the same scene. For example, the building in Figure 2.5 shows the house as well as a 3D model of a Tesla car in the same scene. The house is naturally the one developed externally. However, the Game Engine allows the possibility of bringing different 3D models together to live inside the same environment. This could be a car, or furniture, as seen in Figure 2.6 or otherwise. Note that when importing models to Unity, it is crucial to be aware of the relative scale that has been used to create the model, i.e., meters, inches, etc.



(a) Virtual living room. (b) Real living room.

Figure 2.6.: Comparison of furnished living room.

The paintings on the walls were created using 3DS Max to create an image frame and images of the actual paintings. Other furniture such as the carpet, TV table, or sofa is downloaded from open-source websites such as Polantis. This website provides free CAD and BIM 3D objects resembling, for instance, IKEA furniture [42]. Additionally, this article enumerates a list of resources for finding open-source multi-purpose use 3D models as well as textures [39]. Note that it is essential to use 3D models with low polygon count, as it could create a performance bottleneck if not taken into account. A possible solution for high polygon 3D models is to use 3DS Max to reduce the polygon count of the downloaded 3D model without ruining the appearance of the original model. It is also important to note that Unity 3D objects work on a meter scale; however external 3D models might be generated

using cm, inches, and other units of measurements. Therefore, it is crucial to be aware of this and communicate to Unity the correct system for which the model is created such that Unity can convert it.

The Unity environment also simulates a day-night-cycle, including a highly accurate algorithm for the sun, which will be elaborated on in a later chapter. Here compass angle and altitude of the house are essential to synchronize the house's position relative to the sun simulation. For the angle, the house needed to be rotated 203° from the cardinal north direction of a compass. This was to align the front of the house correctly with the sun movement. Since the setup of the standalone DT also includes a correct height map terrain of the outside environment in Trondheim, the house was elevated to the correct altitude, which is 211m.

2.3.3. Terrain Height Map

(a) Visualization in unity. (b) Google map.

Figure 2.7.: Terrain height map in unity with the red circle showing where the 3D model virtual house lives in that map, zooming in towards the house will be equivalent to looking at 3D model of the house in Figure 2.5. There are no blue colors as the rivers and lakes have not been populated by water. Textures have been generated using a custom script that procedurally draws either mud, grass, stone or snow based on the altitude of each pixel.

Creating terrain in Unity based on a height map of Trondheim is not a straightforward task. However, the Norwegian government provides accurate height maps of Norwegian terrain with up to one-meter precision. One can apply for this data at Kartverket website, by selecting the specific region one wants to extract from the map of Norway and sending a request [43]. Other free APIs provide height map data, such as TerrainParty and SkyDark. However, these only provide an accuracy of 8km and 17.28km, respectively, while the detail level of Kartverket is at a 1m precision [44, 45]. It is worth noting that the data retrieved from Kartverket

needs to be preprocessed from its original file format GeoTIFF to RAW, which is the input format that Unity supports for height maps. Here is an enumerated instruction list to convert heightmap data to a format usable in Unity:

1. Apply for heightmap data from Kartverket.
2. Import all the GeoTIFF files to the software QGis.
3. Merge the GeoTIFF files into one common one using QGis merge option (Raster → Miscellaneous → Merge) use output data type as UInt16 and click run.
4. Export file by converting it to a BIL file, which is a variant of the RAW file type. This is done by navigating to (Raster → Convert → Translate).
5. Click on Advanced Parameters, and add the following commands "-ot UInt16 -outsize 2049 2049" into "Additonal command-line parameters [optional]", note outsize needs to be scaled to be in bit format +1 i.e. 1025,2049,4097 etc.
6. Click on "Converted → Save to File" and change "save as type" to BIL files (*.bil).
7. Click run, go to the folder where you saved the BIL file, create a copy and change filetype to ".raw".
8. Finally upload the extracted RAW file into a Unity Terrain object using the height map property of the object.

Note that the 3D model of the house is placed in the location seen in Figure 2.7, based on its real longitude and latitude. Where the top left corner of the terrain height map functions as the origin in a Unity grid, such that the house can be precisely positioned accordingly.

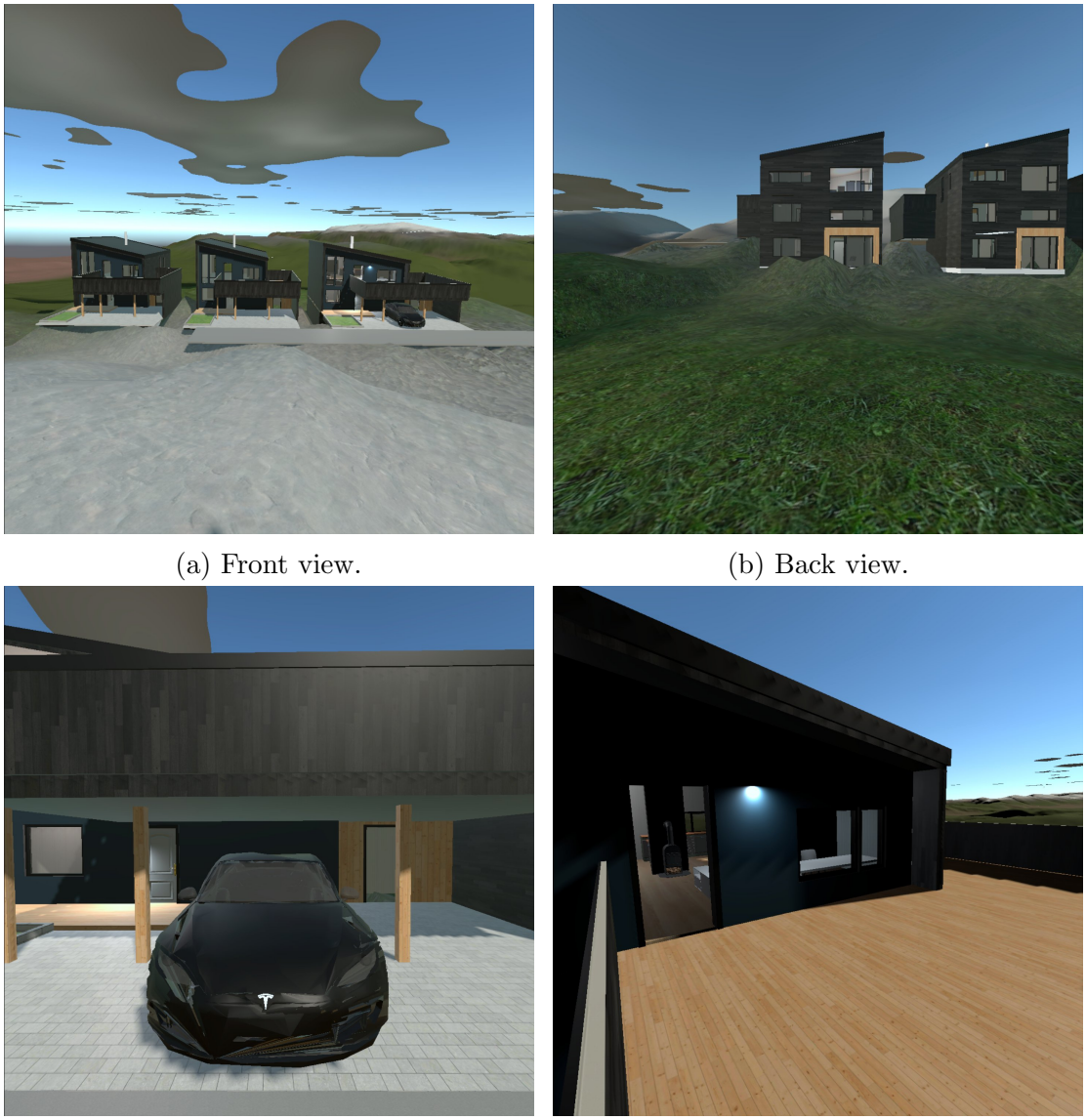
2.4. Demonstration of Standalone Digital Twin

In this section, we demonstrate the value of the standalone DT. For the particular house considered here, Figure 2.8 was provided by the real estate agent to give an idea of the site before construction. It is clear that not much can be imagined based on such an image. However, with a standalone DT, it is possible to experience the internal as well as the exterior environment of the building. The experience can help the buyer, for, e.g. customize the interior, estimate the solar potential, and feel the recreational activities in the area.

Figure 2.8.: A visit to the site before the deal of the house was finalized.

It should be self-evident that Figures 2.9 and 2.10 is more informative compared to Figure 2.8. Figures 2.9a and 2.9b gives a better idea about the surroundings of the house after it is built. It can also help in the choice of the external material for building to still keep it in harmony with nature. Other issues like the space in the driveway can help in planning which category of cars can be parked, or if there should be a change in design of the driveway to accommodate bigger cars (Figure 2.9c). The lighting simulation in the balcony (shown in Figure 2.9d) gives an idea about the availability of sun light for any day and time in a year. Likewise visualization of the internal environment (Figure 2.10) can help in more optimal placement of lights, choice of wall colors and flooring, furniture, and other such objects.

The standalone DT comes with a complementary demonstration video in this link *https://youtu.be/xXt5FFaVokQ* (0:00-1:50). A live demo of the concept similar to the one shown in the gallery and video can also be done onsite.



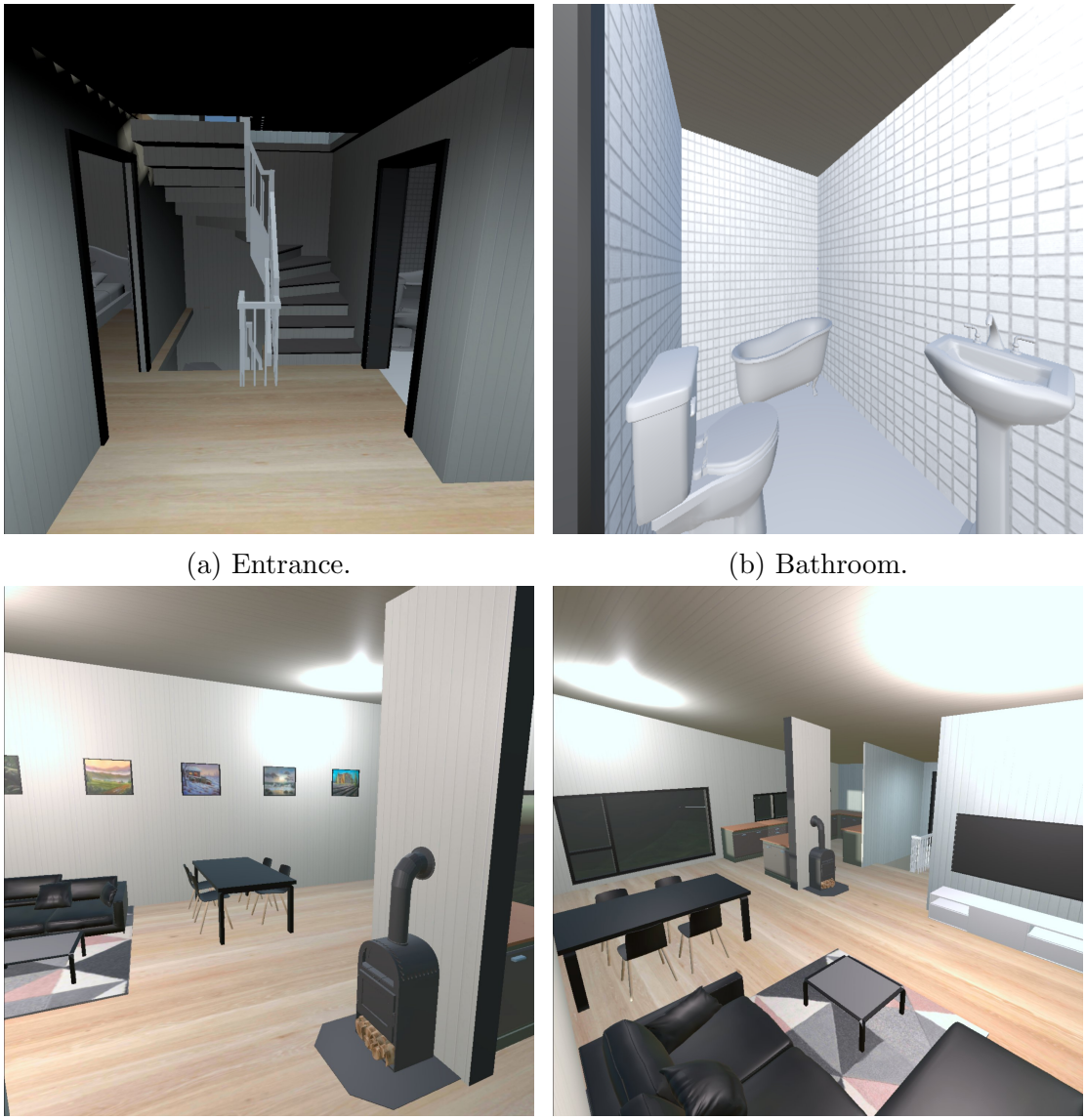
(a) Front view.

(b) Back view.

(c) Driveway.

(d) Balcony.

Figure 2.9.: Demonstration of the external environment using a standalone DT.



(a) Entrance.

(b) Bathroom.

(c) Living room.

(d) Living room.

Figure 2.10.: Demonstration of the internal environment using a standalone DT.

2.4.1. Summary

On an ending note, the demonstrations of the standalone DT provide significant value for homebuyers who wish to commit to a property. The ability to virtually experience the house, customize furniture, color schemes, and materials can effectively upsell the property. Realtors can present their customers with presets of the location before it is built, with the added opportunity to experience the virtual space first hand. A standalone DT of a neighborhood can also be a powerful tool in the hands of municipalities to better develop the area.

3. Descriptive Digital Twin

1 Provided that a standalone digital twin of the house existed, the buyer took an informed decision and got their house customized to their liking. With the digitalization in mind, they got advanced sensors and controllers installed. These sensors can provide real-time information about the house in terms of indoor air quality, water leakage, state of the doors, presence of people, break-ins, and external weather conditions to name a few. A descriptive digital twin can then process this information in real-time. The homeowner, if away, can then keep an eye on their house from the remote location. Furthermore, while they are residing inside the house, they can still benefit from the visualization of the measured quantities which are not visible to the naked eyes.

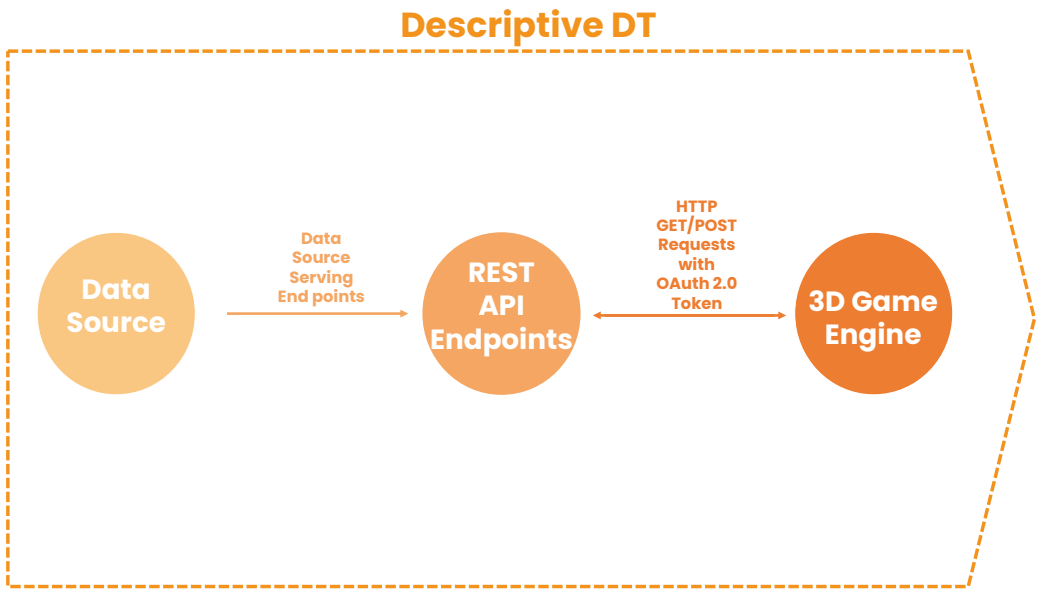


Figure 3.1.: Visualization of the descriptive DT. In our case the data sources are sensors that serve data to REST API endpoints provided by the sensor providers, for which can be accessed in the Unity Game Engine through C# authenticated API requests.

This chapter describes how a descriptive DT can be built upon the standalone DT developed in the previous chapter. The starting point for developing a descriptive DT is to equip the house with a wide array of sensors, stream the sensor data into

a virtual environment, and visualize them without the application of any analytics.

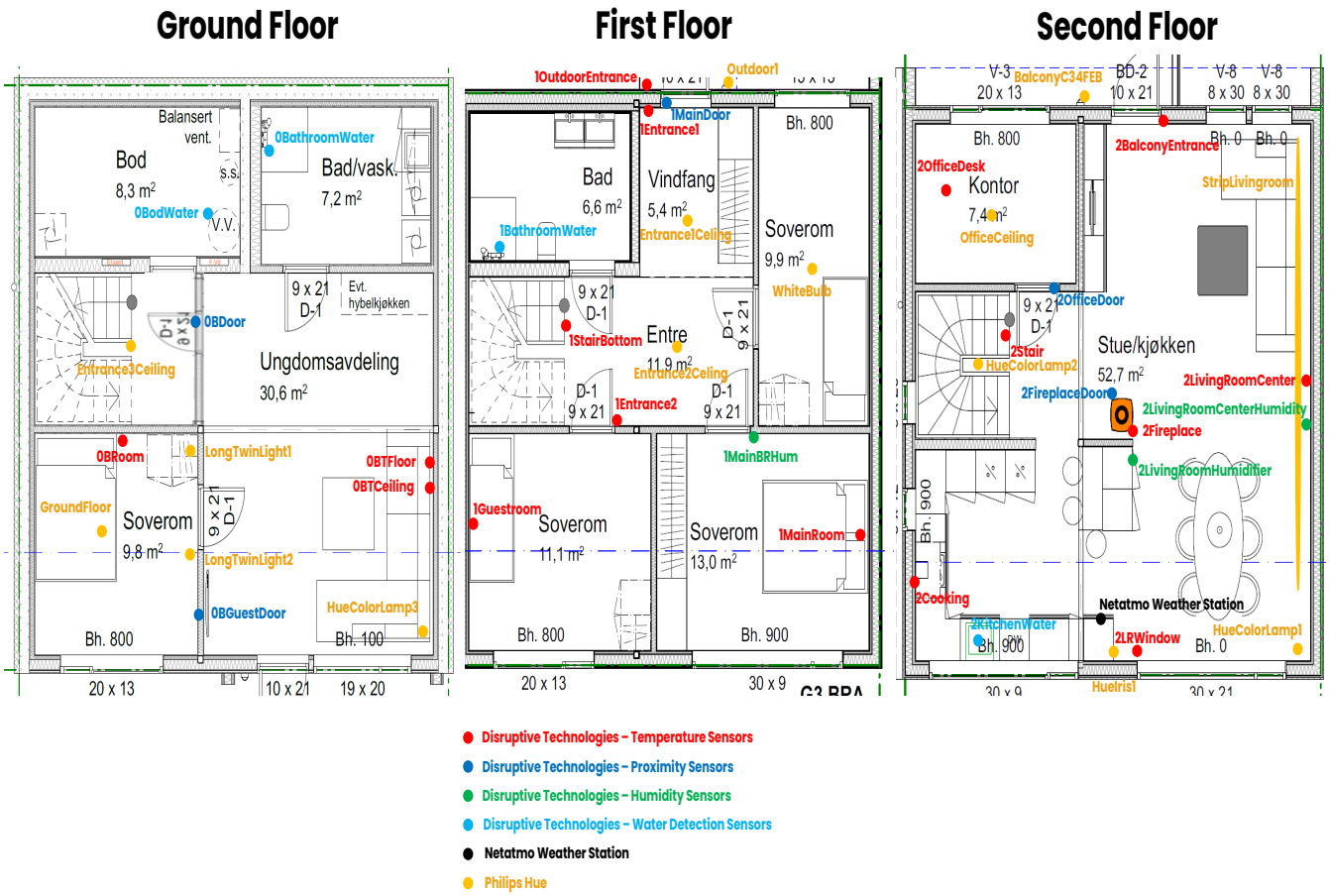


Figure 3.2.: 2D floor plans of the target house with all the sensors and devices setup for the project. All sensors are identified by their label and color as there are temperature, proximity, humidity, water detection and light sensors. All sensors are integrated into the Unity application environment where the BIM model of the house lives.

The descriptive DT is at the first level. Here 3D models and real-time stream of sensor data describe the up to date state of the asset at any point of time. In recent years there has been a boom in IoT and high quality, accurate and cheap sensors with long lasting battery life [46]. These sensors have created a lot of value for maintenance workers in big commercial buildings [47]. Seen from the perspective of DT capability levels, these buildings have existing BIM 3D models as well as real-time sensor data. Given that the BIM and the sensor data were to be integrated within the same interface that would qualify as a descriptive DT. A specific example of such a building is the Royal Opera House of London that both has a

BIM and recently been instrumented with Disruptive Technologies sensors [47, 48].

Creating a high capability level DT does not have a clear value for commercial real estate owners. How ever just by combining a BIM model and sensor data in one environment, has been shown to give stakeholders a better idea of the carbon footprint of their buildings. Therefore giving them a better aim at being able to reduce the energy consumption from different components and systems. Nordic BIM Group claims that sensor data combined with a BIM model has shown to reduce costs by 50% in a year's horizon as work is being prioritized based on sensor data [49]. Showing how the Unity Game Engine can be utilized to lift a standalone DT to a descriptive DT level has huge environmental benefits, and the motivation for pursuing a descriptive DT setup is to communicate to companies and real estate owners that it is possible to create a high capability level DT environment using existing technologies.

In the specialization project the descriptive DT was dependant on an external backend server written in javascript, that would upon request from the Unity client, fetch data from the sensor's REST API endpoint and send it to Unity. This created a layer of complexity that was necessary at the time, as the focus was creating a proof of concept descriptive DT [2]. This has been reworked in the thesis such that this logic is being handled directly in the Unity Game Engine using C#, which makes the application pipeline easier to work with. As seen in Figure 3.1, the only part that has changed is that HTTP fetch requests and Oauth 2.0 authentication is being done in the Unity Game Engine directly. Note that the sensor providers have created these data endpoints to adhere to REST API protocols, which stands for Representational State Transfer and is a set of architectural constraints for data flow between a client and server [50]. since the REST API server is written by the sensor providers it is not necessary for us to worry about how they work internally. How ever what is important to point out is that each sensor provider might expect a different type of HTTP request in order to get access to the data that we need. While there exists external libraries in python and javascript that handles these technicalities [51, 52, 53], this has to be done as part of the setup in C# to make it functional in Unity.

We now give a brief overview of the structure of this chapter. Section 3.1 to 3.4 is about the setup and methodology of the sensors/equipment related to the descriptive DT. Section 3.5 demonstrates results related to the descriptive DT.

3.1. Netatmo Weather Station

Image Source: netatmo.com

The Netatmo Weather Station is setup in the living room as seen in Figure 3.2, and updates its data once every five minutes [54]. The station measures temperature, humidity, CO2 concentration, noise levels and pressure. The weather station can be customized and equipped with many more sensors, but is only equipped

to monitor the mentioned data.

All Netatmo endpoints are located at “https://api.netatmo.com”. Before being able to access the data it is important to first do a POST request for an access token by adding “/oauth2/token” endpoint to the Netatmo API link . The request body has to have grant_type, client_id, client_secret, username, password and scope included [55]. Once the request is successful Netatmo grants an access token which can be used to get access to the current states of the Netatmo sensors. Here another POST request needs to be done to the ‘/api/getstationsdata” endpoint, where in the request body the granted access token has to be given as part of the request. The two step method to access sensor data is similar for the two other sensor providers, how ever the difference is in what they require to be a part of their access token authentication process. Note that data from Netatmo gets updated every five minutes.

3.2. Philips Hue

Image Source: philips-hue.com

The Philips Hue sensors are located all across the house as seen in Figure 3.2 and are identified by the yellow dots in the Figure. There are in total 16 Philips Hue lights used in the project, and it is worth noting that there are other analog light sources in the house which are not connected to the DT. The Philips Hue light bulbs provide various information about the brightness, state of the light (on/off), colors and any other relevant information about the light bulbs.

The authentication process for Philips Hue is a lot more complicated, as it requires that the initial authentication request for an access token needs to be done in the same local network. Once that access token expires one can use the refresh token that stays valid for a hundred and twelve days [56, 57]. Note that the refresh token is granted at the same time as the access token. The Philips Hue API that hosts the endpoints is ”https://api.meethue.com”, where to refresh an access token you have to provide a valid refresh token as part of the request body and do a POST request to the ”/v2/oauth2/token” endpoint. Assuming a valid access token one can get access to the light bulb data by doing a GET request to the ”/bridge/<USER_ID>/lights” endpoint, where USER_ID is an id granted by Philips Hue when an account is registered. In the same way as for the Netatmo, the access token has to be provided as part of the GET request to get the data.

3.3. Disruptive Technologies

Image Source: disruptive-technologies.com

The Disruptive Technologies sensors are located in all the floors where totally twenty eight sensors are being used. Amongst these the majority are temperature sensors as seen in Figure 3.2. There is approximately the same amount of humidity, proximity and water detection sensors. Note that the humidity sensors also register temperature data just like the temperature sensors. Furthermore all the Disruptive Technologies sensors can be used as a touch sensor, meaning that one could program an event trigger if that sensor has been pressed. Obviously this opens up a lot of freedom for experimentation where pressing a certain touch sensor could trigger an email, or something inside the Unity DT and so on.

The authentication process for Disruptive Technologies is somewhat similar to Netatmo, but with an extra layer of security. To get an access token you have to encrypt the email and key_id using the secret, all of which are provided when registering an account with the sensor provider. The encryption has to be done with a HS256 hash algorithm. Then the encrypted token has to be sent using a POST request to the "https://identity.disruptive-technologies.com/oauth2/token" endpoint with an assertion and grant_type [58]. Given one provides a valid access token in a GET request, one can get various data using the following endpoint "https://api.disruptive-technologies.com/v2" with parameter variations provided in the documentation [59]. Note that data gets updated on a frequency of fifteen or five minutes, or every time that a drastic change happens. The frequency of data sampling is dictated by internal logic of the proprietary hardware of the sensors.

3.4. Weather Conditions

3.4.1. Wind Data

Local wind data such as wind speed and wind direction for the location of the house are being requested from the "api.met.no", which is the Norwegian Meteorological Institute's weather forecast API. There are few endpoints one could use, but for our use-case the "Nowcast 2.0" endpoint is being requested. The request requires a User-Agent identity to identify the reason behind the request [60]. Note that the API only updates the weather once per hour.

To visualize the wind direction and speed, a visual effect vignette is made in Unity to appear around the house. This way when peeking outside one can see a wavy white line moving across the sky, which is supposed to illustrate the direction of the wind as well as its velocity. One such wind effect can be visibly seen in Figure 3.5f. The visual effect has no physical property but is valuable in terms of relaying speed and direction of the wind at that particular moment.

3.4.2. Sky Condition

A weather system is created to help monitor real-time weather conditions in the Game Engine [61]. The system will adjust cloud thickness, altitude and movement as well as animate rain or snow if such a weather event occurs. The cloud system is based on the shader technology of Unity, which is the same used to render temperature heat maps in Chapter 4. The clouds interact with sun light, creating shadow and blocking the sun such that it gives the environment a sense of being dynamic and alive. How ever the forming of the clouds beyond being animated and abiding to certain algorithmic rules, are not based on any physical properties. The clouds are simply random procedurally generated in a way that mimics the natural phenomenon. The clouds' movements depends on wind speed and direction. Also for simplicity sky and weather conditions have have been boiled down to five presets i.e. rain, snow, fog, cloudy and clear sky.

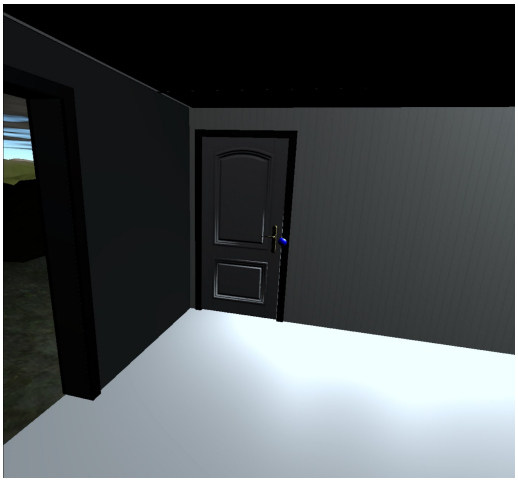
3.5. Demonstration of Descriptive Digital Twin

The descriptive DT comes with a complementary demonstration video in this link *https://youtu.be/xXt5FFaVokQ?t=110* (1:50-7:20). A live demo of the concept similar to the one shown in the demonstration and video can also be done onsite.

3.5.1. Sensor States

(a) Virtual office door. (b) Real office door.

Figure 3.3.: Real-time states of 2OfficeDoor, 2Stair, HueColorLamp2 located in the second floor, observed 21.02.2022. Note that the color of the spheres follows the same standards set in Figure 3.2 and their positions represent the 3D position of the sensor in the physical asset.

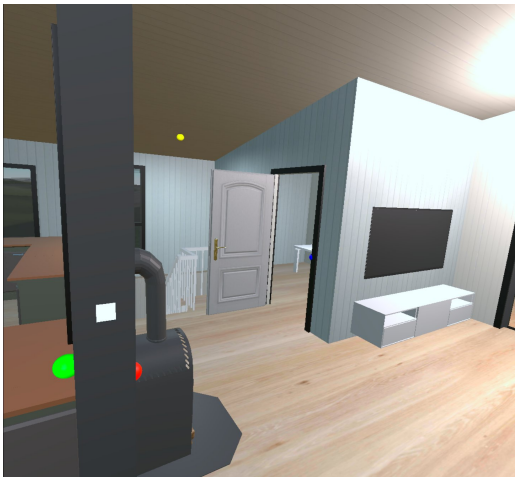


(a) 0BGuestDoor.

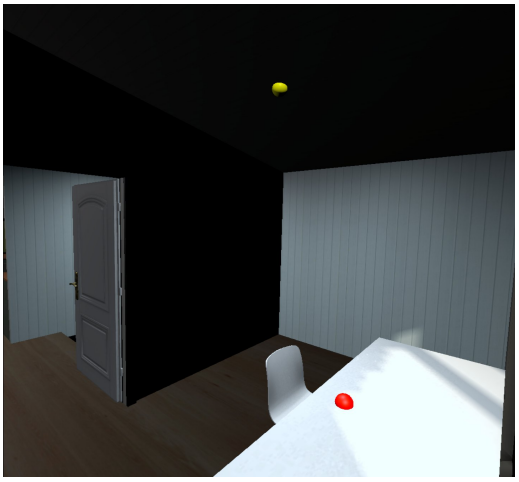
(b) 0BDoor.

(c) 1MainDoor, Entrance1Ceiling.

(d) 1MainDoor, Outdoor1.



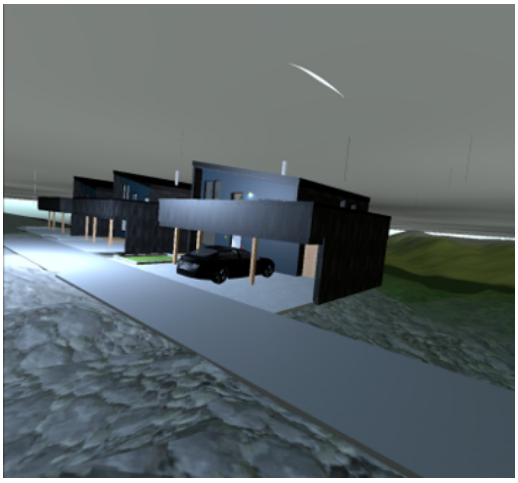
(e) 2OfficeDoor, HueColorLamp2.



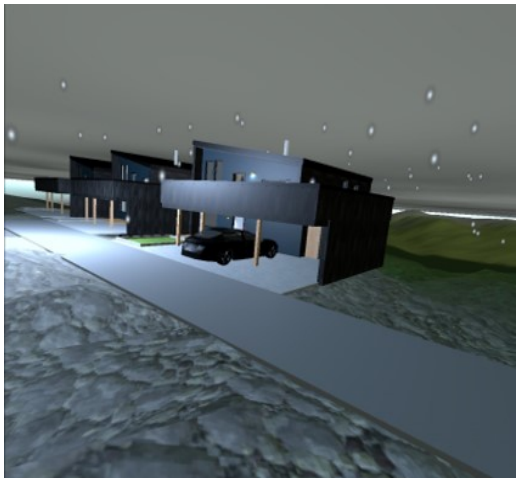
(f) 2OfficeDoor, OfficeCeiling.

Figure 3.4.: Demonstration of various door and light sensor states in the descriptive DT. Observations recorded throughout the day on 21.05.2022.

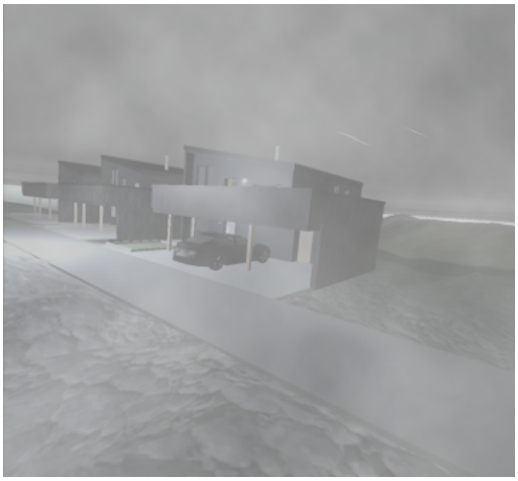
3.5.2. Weather Conditions



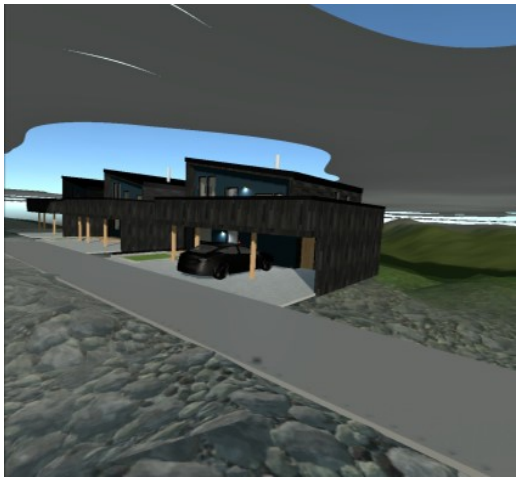
(a) Rain observed 24.03.2022.



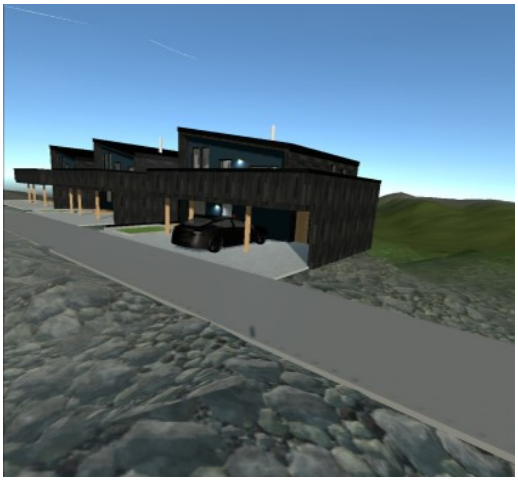
(b) Snow observed 03.04.2022.



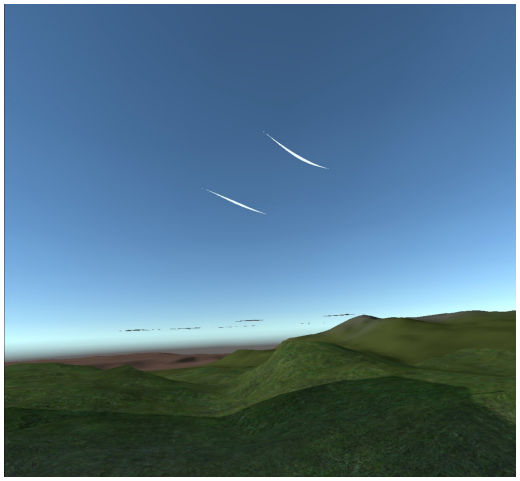
(c) Fog observed 25.04.2022.



(d) Cloudy observed 03.05.2022.



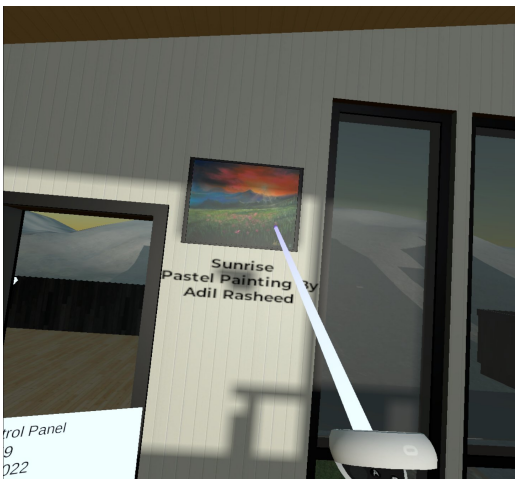
(e) Clear sky observed 21.04.2022.



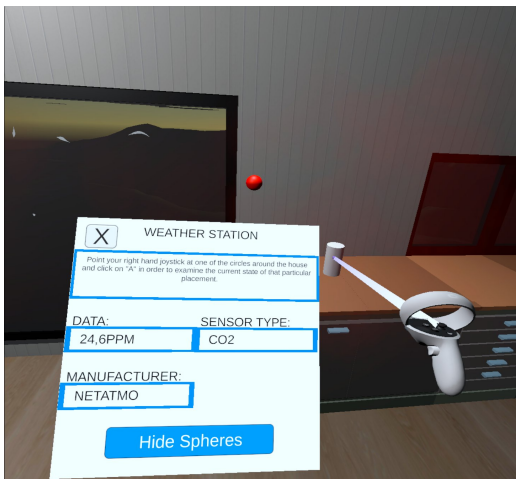
(f) Wind effect.

Figure 3.5.: Demonstration of weather conditions taking place in the descriptive DT.

3.5.3. Remaining Sensors and Furniture



(a) Pastel painting.



(b) Real-time CO2 data.

(c) Real-time temperature/humidity data.

(d) Another pastel painting.

Figure 3.6.: Information from the remaining sensors and other objects, obtained using a virtual reality interface.

Figure 3.3 and 3.4 shows the real-time states of the asset, coming from Netatmo, Disruptive Technologies and Philips Hue. Note while sensors such as humidity, temperature and CO2 are inside the Game Engine at this stage these are not visualized. As such only states that can directly be physically altered is represented, while other sensor data is first represented visually in the diagnostic DT covered in Chapter 4. Figure 3.5 shows the weather conditions that the descriptive DT is capable of displaying.

Figure 3.6 shows real-time data from paintings, CO2, humidity and temperature that otherwise is not possible to represent visually in the descriptive DT. Data coming from objects in the house is stored in a database and regularly updated by the home owner in order for the up-to-date state of said object to be contained within the DT. An example of such an object is the paintings. If the home owner

decided to change the name of the selected painting from “Sunrise” to something else, this will also be updated in the descriptive DT. Monitoring the name of a painting might not seem very significant, but the idea of the demonstration can be translated into monitoring more important states such as how much fire wood there is in the fireplace to reflect the inventory of the physical house.

3.5.4. Summary

For much of human history a house has only been seen as a place of residence, how ever the modern house experience can be embedded with unfathomable amounts of sensors that are cheap, small, long lasting and available. This makes it possible to have a property where the virtual asset is a reflection of the physical asset. Real-time updates of states and environment allows the homeowner the value of experiencing their home exactly as they left it within the virtual space. Along with the added benefit of staying informed about the property at all times. Today a home is not only a residence but a rich untapped data source, and using the available real-time data is only scratching the surface of what can possibly be achieved through the analysis of historical data.

4. Diagnostic Digital Twin

2 Imagine two distinct situations, one in which the homeowners are inside the house and another in which they are remotely located and do not have physical access to the house. Since the house is equipped with a multitude of sensors, not only can the residents be updated with the current state of the house, as in the case of a descriptive digital twin but also diagnose critical changes in the house. While the owner is present physically inside, they can utilize the diagnostic digital twin to gain more insight into the internal environment. For example, using virtual reality to visualize air quality, noise, or temperature maps, which are invisible in actual reality. Otherwise at a remote location, the diagnostic digital twin can provide analytics about the current situation in the house such as temperature increase, room occupancy and other insight by fusing data from different sensor sources.

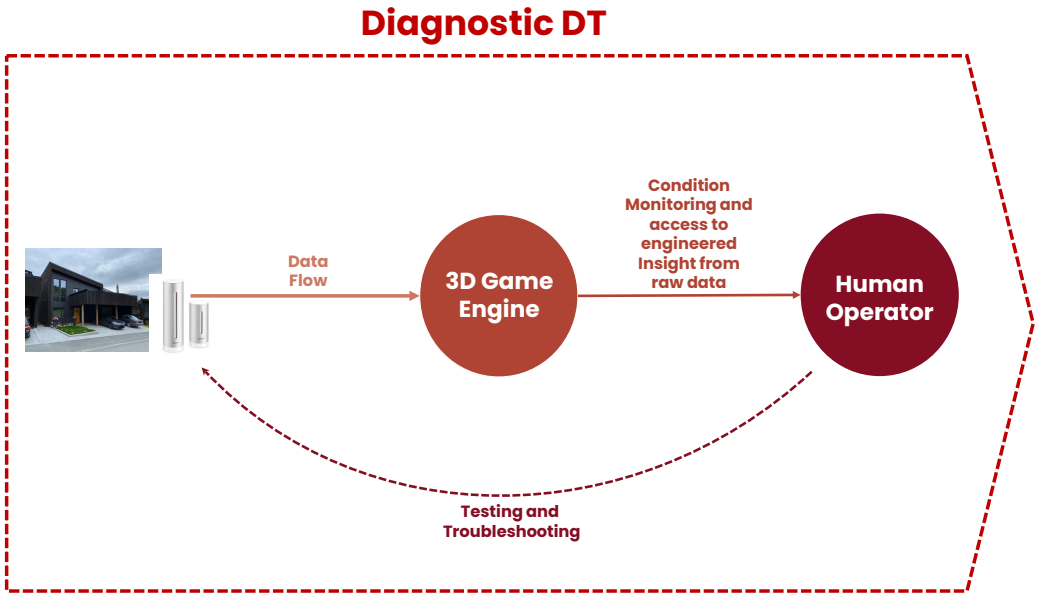


Figure 4.1.: Visualization of the diagnostic DT. As the human operator will be observing a DT that has reached this capability level, the DT will be able to provide the user with an adequate UI that prioritizes notifying the human operator about critical conditions if they appear.

This chapter is about the setup and methodology used to achieve the diagnostic DT. The Game Engine application is developed to include a VR setup that comes accompanied with an informative user interface (UI). The UI displays information

and triggers different visualizations for condition monitoring purposes at the user's request.

After the standalone and descriptive DTs, with the addition of analytics tools, comes the diagnostic DT within the framework, as explained earlier. At this level, the DT can present diagnostic information that supports users with condition monitoring and troubleshooting. The diagnostic DT integrates all the available data in real-time from multiple sources, fuses them together, generates insights, and presents the findings in a user-friendly format on-demand using an intuitive interface. In the event of critical changes the DT should capture the user's attention [9, 62].

In Chapter 3 sensor/equipment data was integrated into the DT. However, these sensors/equipment could only be monitored or controlled through the proprietary UI and libraries created by the sensor/equipment providers. Thus despite having a multitude of sensors from different vendors operating simultaneously in the same house, they could not communicate with each other. Moreover, the out-of-the-box UI systems were very limited in their functionality. For example the spatio-temporal resolution of the data was very coarse and irregular. Therefore, in the context of diagnostic DT, the first task undertaken was to unify the data from diverse sources and structure them for further analysis. Also, as the number of sensors scales with time, analyzing and inspecting them individually will be sub-optimal. Through the implementation of analytics tools to create continuous high-resolution spatio-temporal maps and immersive visualization in virtual reality within the DT framework, gives an intuitive and real-time feel. In particular, allowing users to delve into the DT through VR provides a captivating experience that emphasizes the importance of creating high fidelity DTs that feels as real as their physical counterpart with the added bonus of being able to interact with data that otherwise cannot be visualized in the real house.

The communication illustrated in Figure 4.1 between the human operator and the 3D Game Engine can partly be broken down to what is called a "Game+" experience, also known to be a serious gaming experience [63, 64]. Game+ is coined by professor Alf Inge Wang and is the field of creating game-like experiences that have other motivations beyond that of gaming itself. Most of the theory is beyond the scope of this project and do not apply for a DT application, but the UI that comes with a diagnostic DT can be justified from what is a good or bad UI in the Game+ philosophies [65, 66]. Using the intuition of the theoretical framework, a simple yet robust left wrist UI system is developed to provide users with an intuitive virtual reality UI system that provides users with vital information literally at their fingertips.

We now give a brief overview of the structure of this chapter. Section 4.1 to 4.3 is about the setup and methodology related to the diagnostic visual effects and VR equipment in the diagnostic DT. Section 4.4 demonstrates results related to the diagnostic DT.

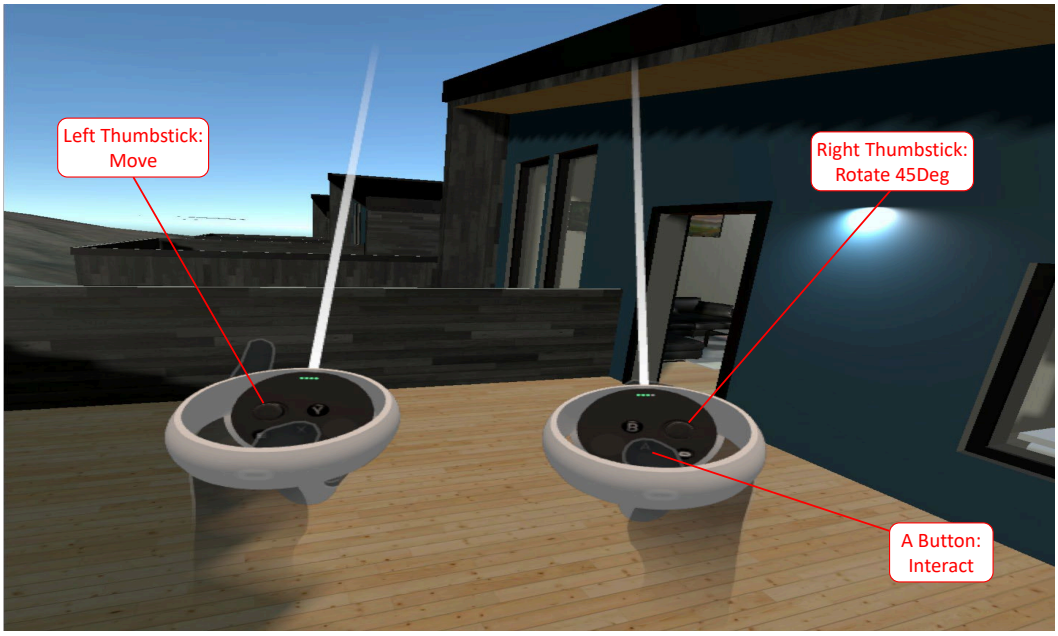


Figure 4.2.: How to use the Oculus controllers in VR. If the play area is big enough, one is also able to physically move around or rotate without using the thumbsticks. Pointing at an interactable object with the right hand controller and clicking on the "A" button triggers events.

4.1. Virtual Reality User Interface

Image Source: oculus.com

The VR setup uses an Oculus Quest 2, where a UI "tablet" is implemented on the user's left hand. This way, the user can move the menu in and out of sight as they see fit. The main focus of the UI system is to give the user a sense of empowerment, curiosity, and, most importantly, feedback, which is all ideas derived from the Octalysis Framework for Gamifaction, which is a serious gaming framework [66]. The empowerment and feedback come directly from the real-time control that a user feels when they can, for example, slide the time of the day and see the weather dynamically move as a consequence. Furthermore, this responsive system triggers an exploratory curiosity in the user, making them want to seek out the remaining contents of the UI system. Recall that the diagnostic DT is mainly about monitoring and troubleshooting, meaning that sparking the user's interest in seeking diagnostic information is as important as presenting the information itself. Therefore a sound UI system in this environment boils down to empowerment, curiosity, and feedback, as mentioned earlier.

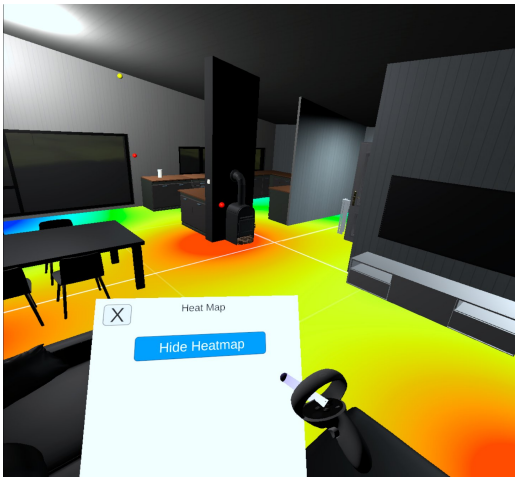
In Figure 4.3 one can see the enumeration of images featuring different navigation panels of the UI system. Figure 4.3a is the main menu which works as the center for all the other monitoring and troubleshooting systems for the diagnostic DT. Figures 4.3b and 4.3c feature the Sensor Inspector, providing critical information

(a) Main Menu.

(b) Sensor Inspector.

(c) Sensor Inspector.

(d) Sun Panel.



(e) Heat Map.

(f) Weather Controller.

Figure 4.3.: Navigating UI panels in Oculus Quest 2 VR.

related to condition monitoring. By pointing at spherical probes around the house and scanning them with the A button of the right hand controller, the Sensor Inspector will display relevant information relating to that probe. Real-time data

from the specific sensor implemented for the descriptive DT is then revealed in the Sensor Inspector panel. Note that the probe positions are supposed to reflect the real three-dimensional positions of the sensors from Figure 3.2. Figure 4.3d shows the UI for the sun control panel, presenting information about the diagnostic time and date. This has the added possibility of predicting future positions of the sun by sliding the time of the day, which is covered in Chapter 5. Figure 4.3e shows combining all temperature probes of a certain floor to create a heat map of the relatively warmest to coldest areas. This gives better contextual information to the user, which indeed could raise the quality of the monitoring capabilities in facility management [67]. Finally, Figure 4.3f displays the possibility a weather controller provides, as wind speed, wind direction, and weather conditions can be observed and the possibility to troubleshoot by initializing hypothetical weather conditions to see their effects on the existing environment.

4.2. Temperature Heat Map

The temperature heat map is a visualization aspect of the diagnostic DT, that presents valuable information in the most convenient way. The heat map do not represent the temperature on the floor of the room, but the temperature distribution of the temperature probes around the room. The heat map in Figure 4.6 assumes that the heat distributes itself in a plane, and therefore is using an euclidean distance equation (Equation 4.1) for path-finding to render the temperature gradient radially outwards onto a Unity shader with the path limited to a reach based on $\frac{\alpha}{2}$. The shader is an object that communicates how to correctly color pixels onto a material in Unity [68]. The implemented algorithm essentially takes the current warmest and coldest temperature sensor measurements in a room, and uses that interval to weight the radial output of the temperature gradient. The color mapping from temperature to color is the same as the one in Figure 4.8, i.e. if the minimum temperature is $16.5°C$ and maximum $25.7°C$ then that would be the blue and red colors respectively and the other temperatures would fall within the color space in between.

$$d\left(p,q\right) = \frac{\alpha}{2}\sqrt{\sum_{i=1}^{n}\left(q_i - p_i\right)^2} \tag{4.1}$$

4.3. Fog Particle Effects

Another possible way to monitor the real-time data coming in to Unity from the various data sources is by using the particles system in Unity. In the specialization project such effects were developed to visualize CO2 concentration (ppm), and a singular temperature [2]. It was argued that the fog could be given air flow properties in the future, to visually represent the CO2. In the project setup, the red fog has its opacity adjusted based on if the real-time CO2 concentration is

within a certain ppm interval, while the temperature is being mapped to a color to represent the heat.

The CO2 and temperature representations in Figure 4.7 and 4.8 are implemented to visualize the Netatmo Weather Station placed in the living room, seen in Figure 3.2. The temperature reading is first transformed into an HSV value as to be mapped into a color, and then from HSV into an RGB that is finally displayed as fog in Unity. It is important to note that the temperature intervals can be redefined, but for this example the temperature interval is set to be between zero and 40 degrees Celsius. As to not overlap, only temperature or CO2 concentration can be visualized at a particular instance [2].

4.4. Demonstration of Diagnostic Digital Twin

The diagnostic DT comes with a complementary demonstration video in this link *https://youtu.be/xXt5FFaVokQ?t=440* (7:20-11:10). A live demo of the concept similar to the one shown in the demonstration and video can also be done onsite.

4.4.1. From Remote Location

From a remote location a person can diagnose if a room is occupied or not by analyzing temperature and door sensors related to a specific room. For example Figure 4.4 clearly shows a temperature rise in the office at the same time the door was opened. As such one can diagnose that the room was occupied around 11:00-14:00. On the other hand, Figure 4.5 is a lot more varied but it still shows a temperature rise after 12:00 when the office door was opened. Which starts to look like a hypothetical work routine of the homeowner in the office. While the homeowners could be interested in diagnosing their own behavior, a more realistic use case is that the homeowner uses this information to know if a room is occupied or not while they are not at home. Note that 2OfficeDoor represents two states, open and closed as zero and one respectively.

4.4.2. From Inside The House

From inside the house the home owner can visualize diagnostic information of temperature maps and CO2 density as seen in Figures 4.6, 4.7, and 4.8. This information can be used to present issues relating to different parts of the house while monitoring the house.

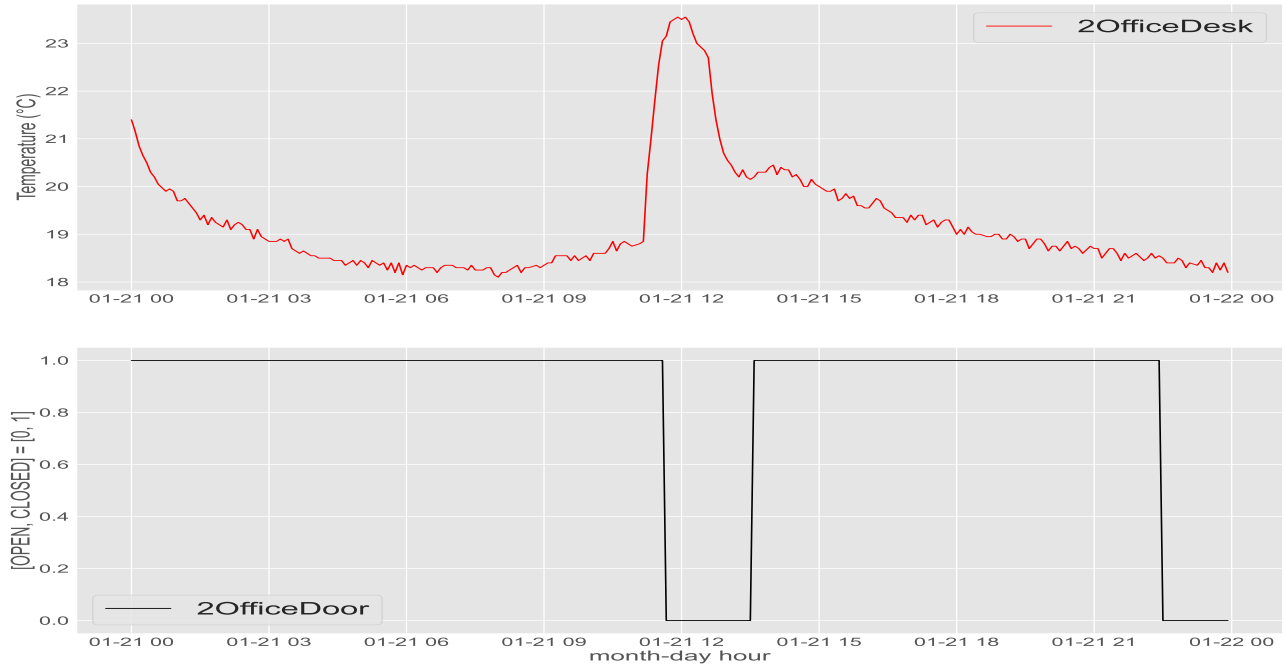


Figure 4.4.: Diagnostic information from fusing office temperature and office door proximity sensor. Scenario one recorded 21.01.2022.

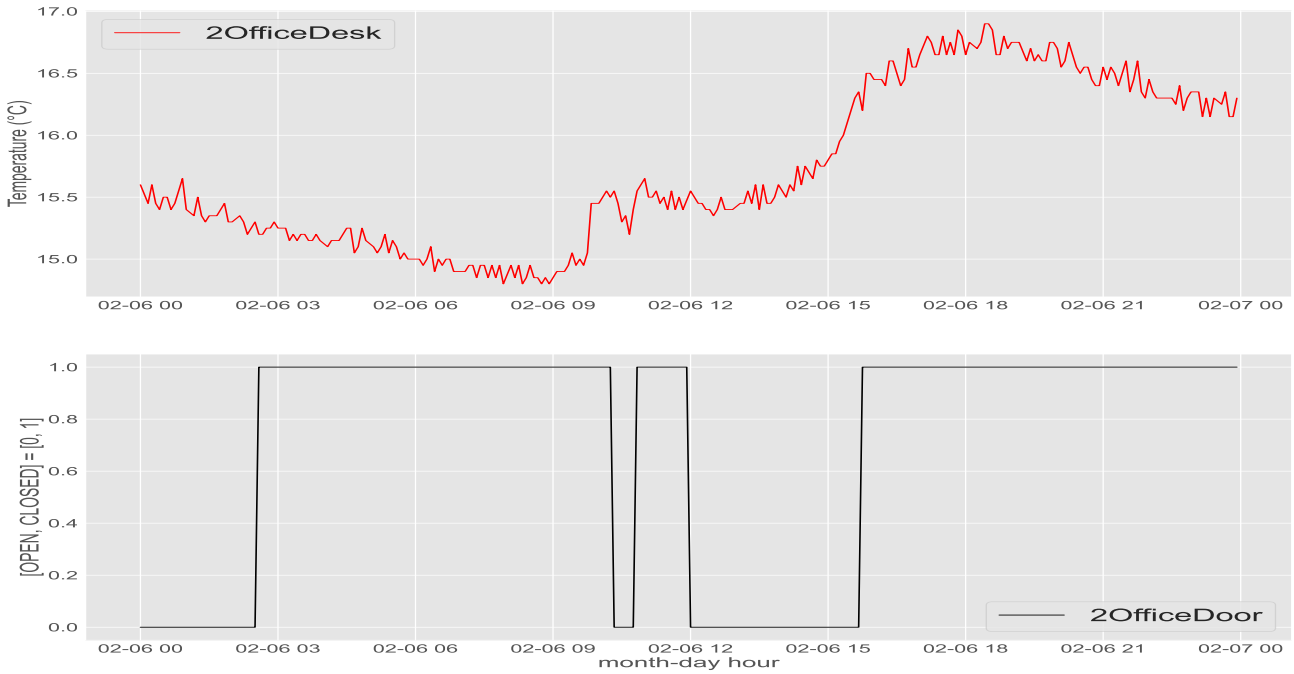
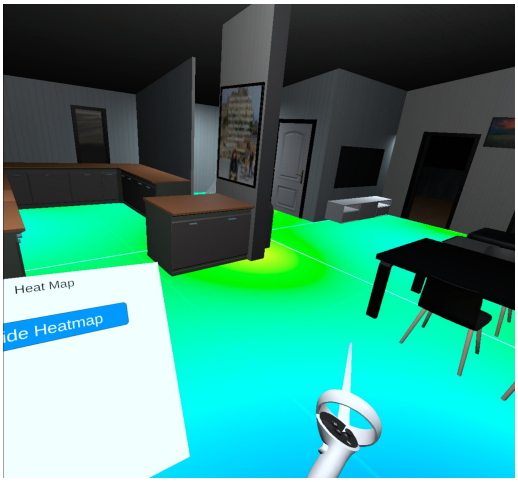
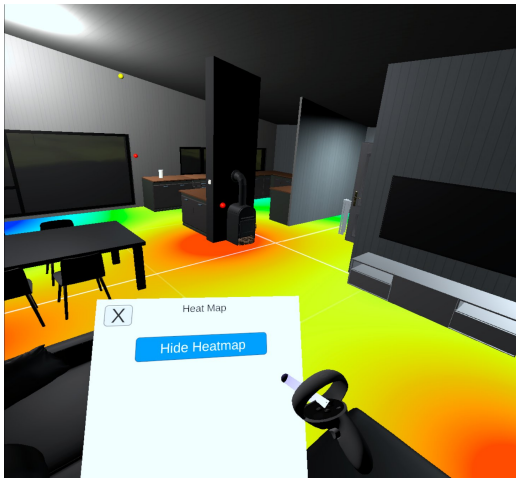


Figure 4.5.: Diagnostic information from fusing office temperature and office door proximity sensor. Scenario two recorded 06.02.2022.

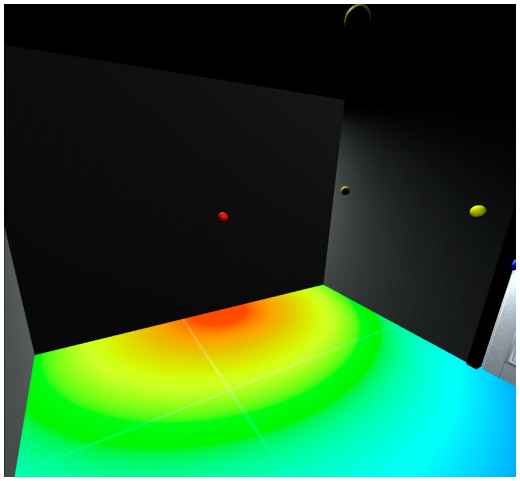


(a) 05.03.2022, second floor.

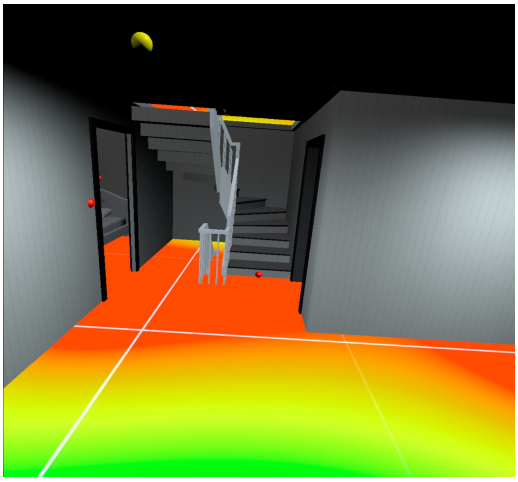


(b) 24.03.2022, second floor.

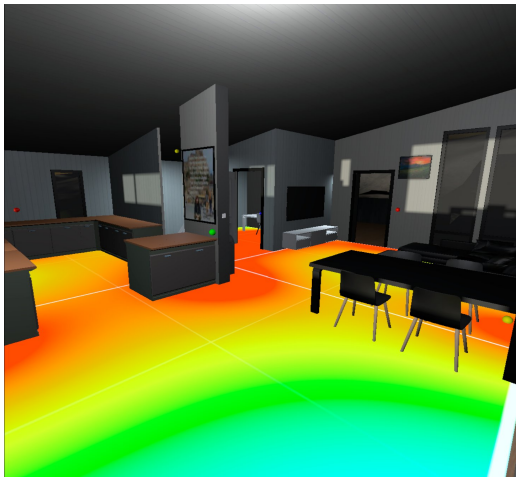
(c) 14.04.2022, second floor.



(d) 24.05.2022, ground floor.

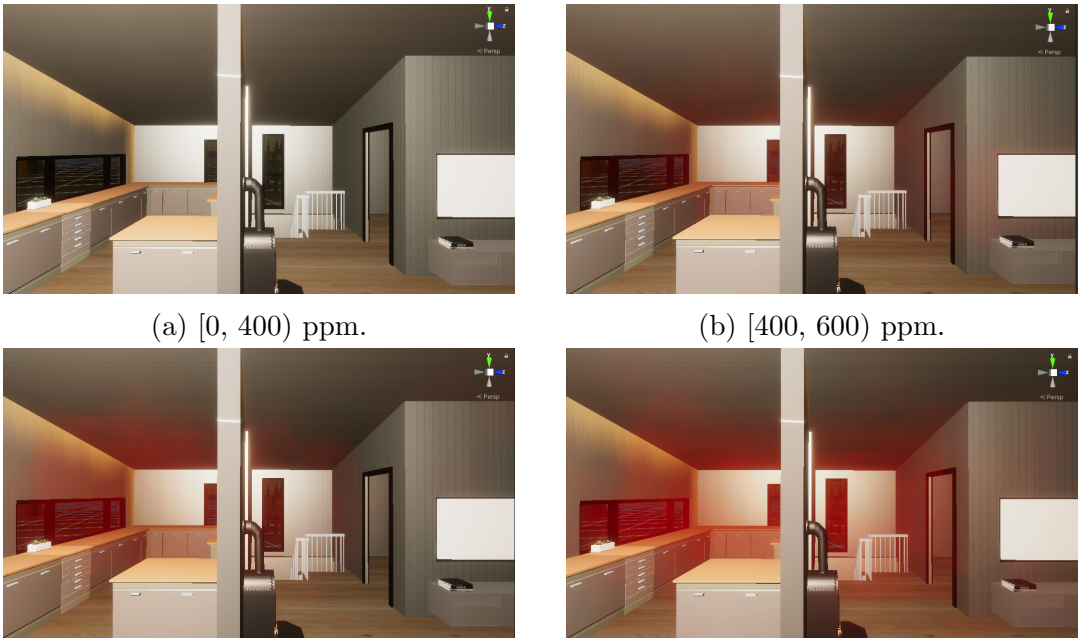


(e) 24.05.2022, first floor.



(f) 24.05.2022, second floor.

Figure 4.6.: Heat map observations on different dates in the entire house.The heat distribution can be seen manifesting itself in different areas of the room, showcasing the warmest areas. For example (b) Was recorded on a cold rainy day that required heating of the fireplace, and (c) was a sunny day where the high temperatures are coming from sunlight through the office window.



(a) [0, 400) ppm.

(b) [400, 600) ppm.

(c) [600, 800) ppm.

(d) [800, ∞) ppm.

Figure 4.7.: Visualizing CO2 concentration from Netatmo Weather Station as fog in the Unity Game Engine based on 4 predefined intervals. Recorded 27.11.2021.

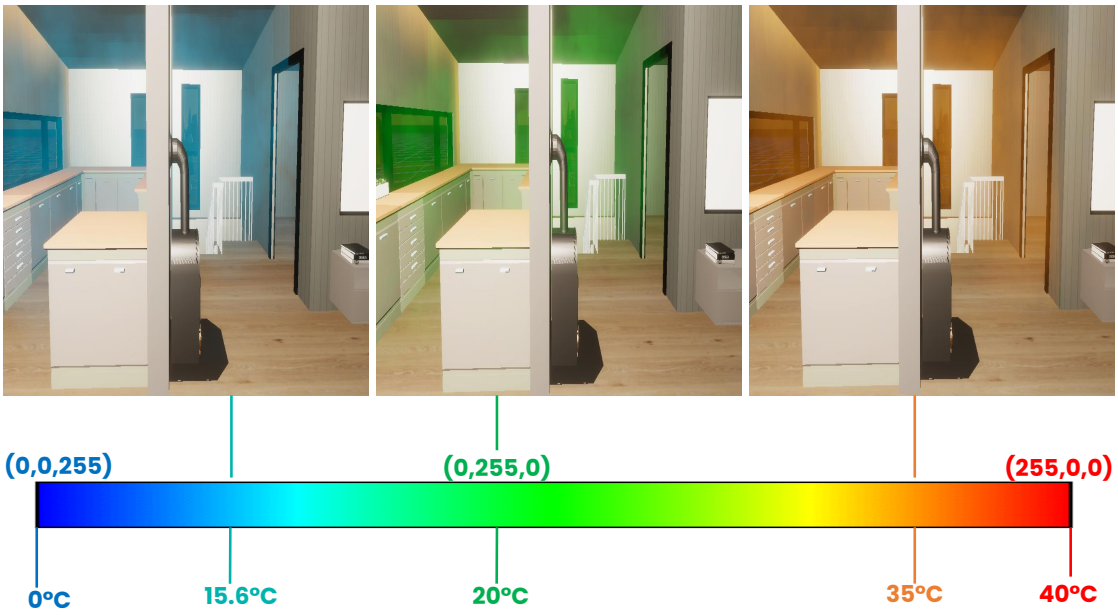


Figure 4.8.: Visualizing indoor temperature from Netatmo Weather station as fog by converting temperature in Celsius to an RGB color representation. 15.6°C was observed 30.11.2021 at 07:30 AM, 20°C at the same date 10:30 AM and 35°C was not an observation but a simulated scenario to display how that would look like in the event of such an occurrence.

4.4.3. Summary

The two distinct situations presented in the start of the diagnostic chapter is demonstrated in Figure 4.4 and 4.5 for remote location diagnostics and Figure 4.6, 4.7 and 4.8 for inside the house diagnostics. The ability to make informed decisions about the house, based on the fusion of different sensors grants the home owner a valuable analytics tool that is enabled only by the installation of sensor data. This shows that real-time aspect of sensor data is only the tip of the iceberg, as the diagnostic DT provides the homeowner with valuable insights that will influence their understanding about the house and themselves. The fact that the entire virtual asset can also be physically experienced as a VR experience, encourages another layer of inquisitiveness because of its novelty.

5. Predictive Digital Twin

3 At this stage, the homeowner has access to a considerable amount of high-quality sensor data both from the past and present. However, they are interested in knowing the future state of the house that will enable better planning for more efficient and cost-effective utilization of energy. They might also be interested in the availability of natural sun light because of the potential development foreseen in the future.

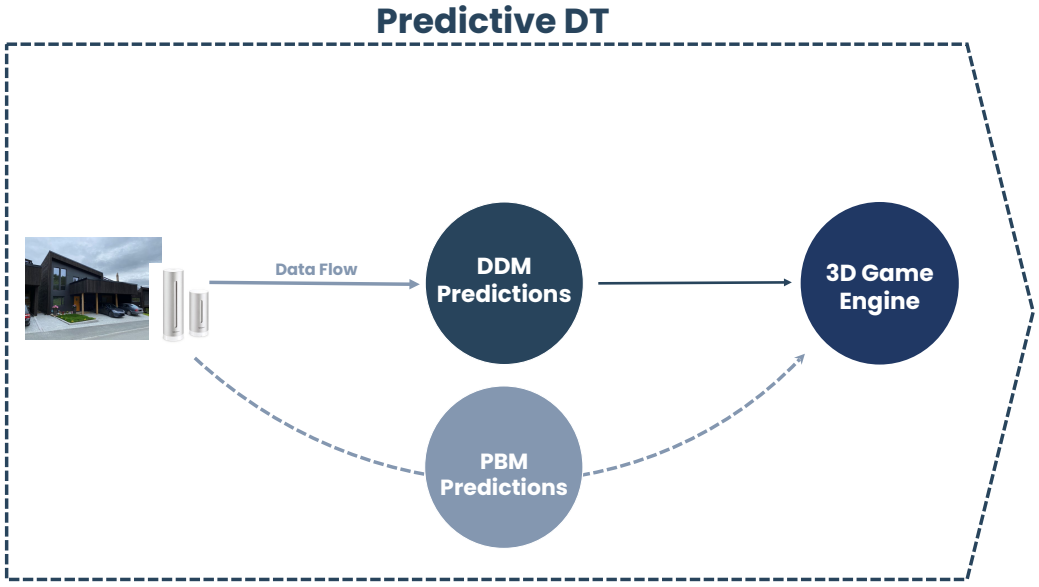


Figure 5.1.: Visualization of the predictive DT. Data streams both into prediction models as well as directly into the Game Engine where future scenarios are displayed.

This chapter is about the setup and methodology used to achieve a predictive DT. At this level, the DT can predict the system's future states or performance and support prognostic capabilities. To demonstrate the DT's predictive capability, we consider two cases one related to the prediction of the inner state of the house in terms of the indoor temperature and another related to the external state in terms of the available solar potential. The reason for choosing the first case is that knowing the evolution of inside temperature can help develop cost and energy-effective control strategies. The second case is of great relevance for a country like Norway, where complex terrain can significantly affect solar exposure. Knowing

this apriori can be crucial in deciding to invest in a real estate.

This chapter also demonstrates two entirely different approaches to modeling that are relevant for DT-like technologies. One is pure data-driven modeling (DDM), while the other is physics-based modeling (PBM). DDM is effective when physics governing a process is not entirely known, is computationally expensive to solve in real-time, or the values of physical parameters are not known accurately. PBM is more effective when the physics is known, or there is a need for generalization in unseen situations for which no data exists. We will use a DDM to predict the indoor second floor temperature based on past experiences because the state-of-the-art building simulation model can not precisely describe the dynamics of the buildings, and the exact composition of the built material is unknown. On the other hand, we will use a PBM for sun position prediction because the equations governing the sun's movement are well known and can be used to deterministically simulate any situation.

We now give a brief overview of the structure of this chapter. Sections 5.1 to 5.4 are exhaustive theory sections on the DDM used to construct the temperature forecasting models. 5.5 and 5.6 are sections on the DDM setup and methodology. Section 5.7 is the theory and setup of the physics-based sun prediction model. Finally the results and demonstrations for both the DDM and PBM are presented.

5.1. Time Series Regression

Time series data is all around us in our everyday endeavors. From intuition, one would probably identify stock prices, weather data, and sensor data as good examples of data that constitute a time series. A time series can be defined to be sequenced data that consists of real-valued continuous numerical observations that are a function of time [69]. The data collected in the current work comes from sensors sampled at regularly spaced intervals; thus, the data can be viewed as continuous-valued but discrete in time. Thus, time series regression models are used for predictions as they are compatible with the dataset's numerical nature.

5.1.1. Time Series Prediction Model

Time series predictions and time series forecasting, while being slight variations of the same thing, can often be confused to mean the same thing [70]. In the context of machine learning (ML) and this work, a time series predictions model will refer to a regression model capable of predicting unknown or unseen values based on present information. On the other hand, a time series forecasting model is a regression model capable of making future predictions based on learned trends and seasonality, amongst other things. The importance of understanding the difference cannot be emphasized enough as one may run the risk of having data leakage if one confuses the two methods.

Given an arbitrary trained prediction model given by Equation 5.1 with $\mathbf{X}$ as an input matrix, if $p = 1$ then this model has a univariate input otherwise if $p > 1$ then the input is multivariate. Similarly depending on if $q = 1$ or $q > 1$ then the output would be either univariate or multivariate respectively. Many existing regression model libraries are capable of handling univariate or multivariate inputs to produce univariate or multivariate outputs of specific targets. In ML this problem would be referred to as a supervised learning algorithm, where input features and ground truth information are used to make predictions about unknown data that has the same input and output structure as the training data.

$$\begin{gathered} \mathbf{model}(\mathbf{X}) = \hat{\mathbf{Y}} \\ \mathbf{X} \in \mathbb{R}^{n \times p}, \hat{\mathbf{Y}} \in \mathbb{R}^{n \times q} \end{gathered} \tag{5.1}$$

An example of a multivariate input and multivariate output model given by Equation 5.2 can be seen where $\mathbf{a}$, $\mathbf{b}$ up until the last arbitrary vector $\mathbf{z}$ is the input for each iteration up until row n, to predict outputs $\hat{\boldsymbol{\alpha}}$, $\hat{\boldsymbol{\beta}}$ and up until $\hat{\boldsymbol{\omega}}$. The predictions can be evaluated against ground truth vectors $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$ and up until $\boldsymbol{\omega}$ which are provided during fitting and validation.

$$\begin{aligned} \mathbf{model}(\begin{bmatrix} a_1 & b_1 & \dots & z_1 \end{bmatrix}) &= \begin{bmatrix} \hat{\alpha}_1 & \hat{\beta}_1 & \dots & \hat{\omega}_1 \end{bmatrix} \\ \mathbf{model}(\begin{bmatrix} a_2 & b_2 & \dots & z_2 \end{bmatrix}) &= \begin{bmatrix} \hat{\alpha}_2 & \hat{\beta}_2 & \dots & \hat{\omega}_2 \end{bmatrix} \\ &\vdots \\ \mathbf{model}(\begin{bmatrix} a_n & b_n & \dots & z_n \end{bmatrix}) &= \begin{bmatrix} \hat{\alpha}_n & \hat{\beta}_n & \dots & \hat{\omega}_n \end{bmatrix} \end{aligned} \tag{5.2}$$

5.1.2. Time Series Forecasting Model

Time series forecasting models can be divided into incremental forecasting models and multi-output forecasting models. Traditional forecasting models such as ARIMA use the incremental method, while a recurrent neural network (RNN) would be capable of both being an incremental or a multi-output model depending on the setup. Incremental models use an input sequence to forecast a single output step. To make an incremental multi-step output, the previous forecasted output is fed as part of the input vector while the head of the input is shifted to remain in the same dimensions until the entire forecast horizon is satisfied. The model given by Eqution 5.3 demonstrates an incremental multi-step where n and m denote sizes of the input sequence and forecast horizon, respectively. Note that the model is simplified, and other features can be included to forecast the future as long as these are accessible before the future happens, e.g. features like day of the week, weather forecasts, and celestial bodies positions. On the other hand a multi-output forecasting model would be able to use the input sequence from the test set to output the entire forecast as seen in Equation 5.4.

$$\begin{aligned}
\mathbf{model}_{inc}(\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^\intercal) &= \hat{y}_1 \\
\mathbf{model}_{inc}(\begin{bmatrix} x_2 & \dots & x_n & \hat{y}_1 \end{bmatrix}^\intercal) &= \hat{y}_2 \\
&\vdots \\
\mathbf{model}_{inc}(\begin{bmatrix} x_n & \hat{y}_1 & \dots & \hat{y}_{m-1} \end{bmatrix}^\intercal) &= \hat{y}_m
\end{aligned} \tag{5.3}$$

$$\mathbf{model}_{multi}(\begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^\intercal) = \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \dots & \hat{y}_m \end{bmatrix}^\intercal \tag{5.4}$$

In forecasting, the data may not always be predictable, and using the random walk hypothesis, unpredictable data can only be best described using the last known value i.e $x_n = \hat{y}_1 = \ldots = \hat{y}_n$. A random walk seen in Equation 5.5 is a noise function initialized with the desired value $g(t-1)$ that has the same probability of incrementing or decrementing by one for each timestep [71]. Through the central limit theorem, this noise will have a gaussian distribution making the last known value x_n the mean of that distribution assuming that $g(t-1) = x_n$. Therefore, given that the time series dataset is unpredictable, it is crucial to observe if a model is fitted to a straight line based on the random walk hypothesis. This indicates that the best forecast in this situation would simply be the last observed value.

$$\begin{aligned}
g(t) &= g(t-1) + e(t) \\
g(t) &\sim \mathcal{N}(g(t-1), \sigma^2) \\
e(t) &\in \{-1, +1\}
\end{aligned} \tag{5.5}$$

5.1.3. Stationarity

Looking at a distribution from a subset of random variables from the fireplace sensor, the temperature will not change over time. I.e. the mean and variance will stay roughly the same, which means that the temperature data is stationary rather than trending up or down over time. Formally this is known as a weak-sense stationarity and is described by Equation 5.6, understanding the internal theory behind these equations is outside of the scope of this project. The equations are used to detect weak stationarity by allowing to check that the mean stays the same given a shift τ and that the covariance staying the same over a given distance t_1, t_2 [72].

$$\begin{aligned}
\mu_y(t) &= \mu_y(t+\tau), \quad \forall \tau \in \mathbb{R} \\
\kappa_{yy}(t_1, t_2) &= \kappa_{yy}(t_1 - t_2, 0), \quad \forall t_1, t_2 \in \mathbb{R}
\end{aligned} \tag{5.6}$$

A less theoretical way to check for stationarity is to use the Augmented Dickey–Fuller test. This statistical test can be written algorithmically where the test outputs a p-value. The null hypothesis is that the data is non-stationary and the alternative

is that it is stationary. Given that the p-value is statistically significant then one can loosely conclude that the data is stationary. The strength of the conclusion will obviously be relative to what is the chosen threshold for statistical significance.

Using the Augmented Dickey-Fuller Test, the fireplace temperature data labeled "2Fireplace" in Figure 3.2 collected from 07.01.2022 to 26.05.2022, can be viewed as stationary with a significance threshold of 5% and a p-value of less than 10^{-22}. The indication of stationarity of the temperature data means in theory that it is possible to construct a single model to forecast some repeatable patterns in the future.

Nonstationary data can be made stationary by applying a suitable transformation. Depending on the dataset some transforms might not work as these assume that the dataset is strictly positive or nonzero. The most common transforms are Power, Log and Box-Cox for regression tasks. For time series dependant regression differencing is used, which is a method to calculate the n-th discrete difference to control the auto-correlations. This method can be inversely transformed after predictions by using the cumulative sum, hence restoring the data to its original format [73]. Equation 5.7 is an example of a first order differencing applied to an arbitrary input vector and ground truth $\mathbf{X}, \mathbf{Y}$. This makes the model predict a differenced output $\boldsymbol{\Delta}\hat{\mathbf{Y}}$, that has to be inversely transformed to obtain the results.

$$\begin{aligned} \Delta x_t &= x_t - x_{t-1} \\ \boldsymbol{\Delta}\mathbf{X} &= [\Delta x_1, \Delta x_2, \ldots, \Delta x_t] \\ \boldsymbol{\Delta}\hat{\mathbf{Y}} &= [\Delta \hat{y}_1, \Delta \hat{y}_2, \ldots, \Delta \hat{y}_t] \end{aligned} \tag{5.7}$$

5.1.4. Evaluation Metrics

An evaluation metric is a way to monitor the performance of a ML model. This is not the same as a loss function, as it is used to measure the performance of a fitted model. A loss function is used as part of the gradient descent back-propagation to fit the model during training. In classification tasks, measuring accuracy is quite binary as the model either is correct or incorrect about its predictions. On the other hand, regression tasks evaluation metrics do not measure direct accuracy of the model but rather to what degree some of the data remain unpredictable for the regression model, and therefore for most regression metrics, a lower error relative to the scale of the dataset means that the model is performing better.

Table 5.1.: Different point estimation methods for evaluation of regression model performance.

MSE	**RMSE**	**RMSLE**
$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$

To evaluate the performance of a regression model, there are many different types of evaluation metrics. Each evaluation metric has its specific use-case depending on the behavior of the time series dataset, and so no specific metric is better than the other. Studying the data could give us indications of which evaluation metric might be useful. For example the mean squared error (MSE), coincides with maximizing the Gaussian likelihood, such that this is a good error metric to minimize if the errors are normally distributed. For many regression problems this is a common assumption, hence MSE or other variants of it such as the root mean squared error (RMSE) or root mean squared log error (RMSLE) from Table 5.1 can be good starting points.

RMSE and RMSLE are very commonly used evaluation metrics for regression problems. RMSLE is robust to outliers, such that the predicted error doesn't explode given that one prediction is overshooting. Instead the error increases but relatively to all the other prediction errors. It also punishes underestimation of values much more than overestimation. RMSE on the other hand gives a high weight to large singular errors which can result in exploding errors. For tasks such as price predictions where a slight overestimation is more accepted than an underestimation and where prices might vary a lot, then it makes sense to use a RMSLE. For price predictions, RMSE evaluation would force the model to become very good at predicting high priced items and only capable of predicting averages of low priced items. For temporal data of room temperatures where the data is stationary due to being bounded by its physical properties, RMSE fits such a task as it will not be affected by the metric's shortcomings. Furthermore it has the added benefit of being interpretable, because the error metric is the same dimensionality as the temperautre data [74]. RMSE penalizes large absolute errors, which is helpful for this particular case.

Cross validation is an application of the evaluation metric, such that one gets a more accurate picture of the regression model performance on the test set. The most commonly used method is K-fold cross validation where the original dataset is partitioned into K equal size subsamples called folds, where $K = 5$ is common. For each one of K total iterations, one fold is left out as the test set and the model is trained on the $K - 1$ remaining folds. The measured error becomes the average of all K trials to get the total effectiveness of the model. The benefits of this is that every observation from the original dataset has the chance to appear both in the training and test set which significantly reduces bias and variance of the model, and enables models to become better at generalizing patterns.

5.2. Traditional Time Series Regressors

5.2.1. AutoRegressive Integrated Moving Average

Autoregressive integrated moving average (ARIMA) is a statistical modeling approach used to predict or forecast data. Autoregressive refers to the multidimensional linear regressor of the model that takes p past time series data as input as

seen in Equation 5.8 where w_i are weights and b is the bias term. Essentially this model resembles a specific case of the generic incremental model given by Equation 5.3 described for time series forecasting.

$$\begin{aligned} \hat{y}_1 &= w_1 x_1 + w_2 x_2 + \ldots + w_p x_p + b \\ &\vdots \\ \hat{y}_m &= w_m x_m + w_{m+1} x_{m+1} + \ldots + w_{m+p} x_{m+p} + b \end{aligned} \tag{5.8}$$

The moving average part of the ARIMA model describes a linear regression of the past errors. This is used to smooth out noise of the predictions, where q describes the size of error terms. For simplicity only one error term equation is shown. Note this part of the ARIMA model is not the same as a simple moving average, which takes the mean of a subsample of previous data to predict the next value in the time series.

$$\gamma_1 = \omega_1 \epsilon_1 + \omega_2 \epsilon_2 + \ldots + \omega_q \epsilon_q + \beta \tag{5.9}$$

To create ARIMA, the autoregressive and the moving average models are summed together which generates ARMA [75]. Finally becoming ARIMA(p,d,q) by taking the current time step model and differencing it with the previous timestep, where d describes order of differencing. Note that a random walk in Equation 5.5 is just a special case of ARIMA as seen in 5.10. This says that future data essentially cannot be forecasted using previous values or previous noise terms and is completely unpredictable aside from the mean of the current error term which is the data from the previous timestep.

$$\begin{aligned} \textit{ARIMA(0,1,0)} &= \Delta \hat{y}_t = \epsilon_t \\ \hat{y}_t - x_{t-1} &= \epsilon_t \\ \hat{y}_t &= x_{t-1} + \epsilon_t \end{aligned} \tag{5.10}$$

5.2.2. Prophet

The Prophet forecasting framework is an easy to setup forecasting model created by facebook, optimized for business related data. The model integrates seasonality at multiple scales such as hourly or daily seasonalities, as well as cyclical but not exactly periodic seasonalities such as holidays. Having these properties as a core of its functionality, it works very well for forecasting future sales. This model may also be a good fit for forecasting high frequency temperature sensor data, with multiple trends and seasonalities as Prophet is capable of handling big datasets. A high level overview of the formal definition can be broken down into four enumerated terms as seen in Equation 5.11; Trend, seasonality, holiday and noise respectively.

$$
\begin{aligned}
y(t) &= g(t) + s(t) + h(t) + \epsilon(t) \\
g(t) &= (k + \mathbf{a}(t)^\intercal \boldsymbol{\delta})t + (m + \mathbf{a}(t)^\intercal \boldsymbol{\gamma}) \\
s(t) &= \sum_{n=1}^{N} (a_n \cos\left(\frac{2\pi n t}{P}\right) + b_n \sin\left(\frac{2\pi n t}{P}\right) \\
h(t) &= [\mathbf{1}(t \in D_1), \ldots, \mathbf{1}(t \in D_L)]\boldsymbol{\kappa}, \quad \boldsymbol{\kappa} \sim \mathcal{N}(0, \nu^2)
\end{aligned}
\tag{5.11}
$$

The trend term tries to model the nonperiodic changes in the time series as a combination of linear slopes intercepted at time t. Seasonality tries to model the periodic changes as a Fourier series, where P represents the period e.g. $P = 7$ for weekly seasonality given that time in the dataset is scaled in days. The holiday term accounts for irregular seasonality that is not captured by the first two terms where it consists of a one-hot encoded matrix. Lastly a noise term constitutes what is considered the unpredictable nature of the dataset [76, 77]. Note one might be tempted to conclude that the model is autoregressive such as AR in Equation 5.8, rather it is Bayesian as all model parameters have priors and is fitted with these in mind as a function of time [78].

5.3. Recurrent Neural Networks

Recurrent neural networks (RNN) improves upon the shortcomings of traditional artificial neural network (ANN) by providing previous events to inform current predictions, allowing information to persist [79]. Some important keywords to be addressed before explaining the intricacies of RNN, are neurons, activation functions, forward and back propagation. Simply each node of Figure 5.2 is a neuron, where the input data is fed through the input neurons and progressively moves through the network to produce the outputs in the output layer. For regression tasks each link between a neuron and its successor is accompanied by a weight and bias term, making linear regression the building block of the network. Looking from the input layer, the successor is a neuron in the hidden layer which squishes the weighted input neurons using an activation function σ as seen in Equation 5.12. These operations combined are termed forward propagation, which end up producing the wanted outputs. Back propagation on the other hand happens during training, where weights at each link of the neural network are tuned using gradient descent of a loss function to make the model fit its predictions to the ground truth data. In other words this an example of a classical supervised learning algorithm, that can both be tweaked for regression and classification tasks alike.

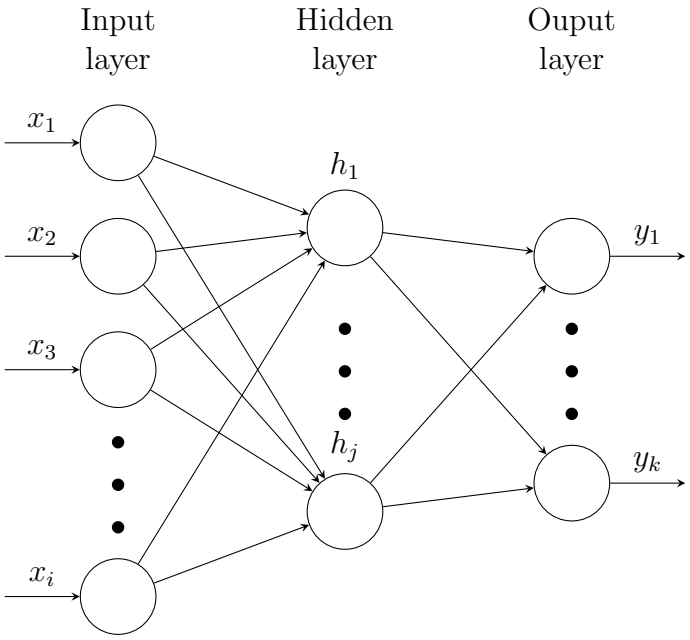


Figure 5.2.: Example of a feed-forward ANN with multi input and output. Total tunable weights and biases are $j(i+k)$ as there are $i \cdot j$ connections between the input and hidden layer and $j \cdot k$ connections between the hidden and output layer. Note that bias term is also represented as an input layer node for simplification.

Note the representation in Equation 5.12 can easily be generalized in terms of matrices. Therefore based on which activation function is used i.e. Sigmoid, ReLU or etc, one can create a general term to describe the back propagation of the network. This will not be demonstrated as the brief introduction into the topic as well as the conceptual elaboration is sufficient to derive the remaining properties of ANNs which are extendable to RNNs.

$$h_n = \sigma(\begin{bmatrix} w_{1n} & w_{2n} & \dots & w_{in} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_i \end{bmatrix}), \quad \forall n \in \{1, 2, \dots, j\} \tag{5.12}$$

5.3.1. Simple Recurrent Neural Networks

One of the main assumptions of ANNs is that a neural network moves only in one direction towards the output. This makes the network architecture restricted to only use a small portion of the information it has learned to make decisions. For a simple Elman neuron, suppose that for the entire sequence of input data is applied to make a prediction. The difference is that instead of just using the current input data to make predictions, the hidden neurons from the previous timestep are also fed into the current neuron [80]. This way the current output is dependant on the current hidden layers, and effectively all the hidden layers from previous timesteps

as seen in Equation 5.13. This helps the model with retention of information, and the ability to use previous contexts with the current prediction [81].

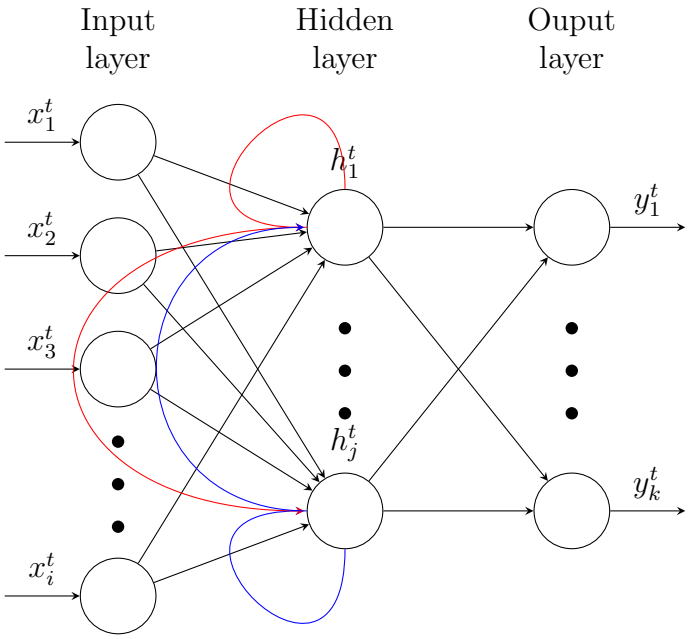


Figure 5.3.: Elman RNN where the red and blue loops represent the previous hidden value states $\{h_1^{t-1}, \ldots, h_j^{t-1}\}$ fed into current hidden neuron. Equation 5.14 shows an alternative representation.

$$h_n^t = \sigma((\mathbf{w}_h^{t-1})^\intercal \mathbf{h}^{t-1} + b_h^{t-1} + (\mathbf{w}_x^t)^\intercal \mathbf{x}^t + b_x^t) \tag{5.13}$$

$$\mathbf{h}^t = \sigma(\begin{bmatrix}\mathbf{W}_x & \mathbf{W}_h\end{bmatrix} \begin{bmatrix}\mathbf{x}^t \\ \mathbf{h}^{t-1}\end{bmatrix} + \mathbf{b}_h), \quad \mathbf{h}^t \in \mathbb{R}^{j\times 1} \tag{5.14}$$

5.3.2. Long Short Term Memory Networks

Although the long short term memory (LSTM) unit is a different cell than the Elman neuron, the way they are incorporated into a RNN remains the same. One of the main pitfalls of an Elman RNN is that it only has the capacity to learn from the most recent previous information. This comes as a direct result of Elman RNNs being vulnerable to the vanishing gradient problem. For an RNN in timestep t, the earliest inputs will be deeply nested which leads to more multiplications to find the gradient term, as a consequence of the chain rule. Therefore this causes the degradation of early memory, making it unable to learn long term dependencies. While this can be solved by specific calibrations and tailored architectural modifications, the fact remains which is that the length of memory retention is not indefinite in such a network leading to the inevitability of the vanishing gradient.

Hence the better solution is to use gated recurrent units (GRU) or LSTMs as they directly address this specific problem [82].

$$\begin{aligned}
\mathbf{f}^t &= \sigma(\mathbf{W}_{xf}\mathbf{x}^t + \mathbf{W}_{hf}\mathbf{h}^{t-1} + \mathbf{b}_f) \\
\mathbf{i}^t &= \sigma(\mathbf{W}_{xi}\mathbf{x}^t + \mathbf{W}_{hi}\mathbf{h}^{t-1} + \mathbf{b}_i) \\
\mathbf{o}^t &= \sigma(\mathbf{W}_{xo}\mathbf{x}^t + \mathbf{W}_{ho}\mathbf{h}^{t-1} + \mathbf{b}_o) \\
\mathbf{c}^t &= \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \sigma(\mathbf{W}_{xc}\mathbf{x}^t + \mathbf{W}_{hc}\mathbf{h}^{t-1} + \mathbf{b}_c) \\
\mathbf{h}^t &= o^t \odot \sigma(\mathbf{c}^t)
\end{aligned} \tag{5.15}$$

The five Equations in 5.15 make up the LSTM unit where $\mathbf{f}^t, \mathbf{i}^t, \mathbf{o}^t$ are logistic regression neurons that are used as binary classifiers and therefore are referred to as forget, input and output gates. Furthermore $\mathbf{c}^t$ is called the cell state and can be seen either as another hidden state that can be fed to the next timestep or just an intermediary function to finding $\mathbf{h}^t$. The first term of $\mathbf{c}^t$ can be interpreted as how much of the previous memory we wish to discard, controlled by the forget gate. The second term resembles a modified version of an Elman hidden neuron in Equation 5.14, where how much of it is kept is weighted by the input gate. Finally the output gate controls how much of the current cell state is passed to the current hidden neuron. Note that the activation function σ is not to be confused with the sigmoid, but rather any arbitrary activation function of choice. Also the term t is not an exponential but a label of the current timestep.

The LSTM gives the network the flexibility to recall previous features that are not in the immediate vicinity of short term memory, and by extension a deduced prediction concatenated from long and short term memory of available data. One of the drawbacks of this unit is that the explicit memory adds many more weights to each neuron, which increases dimensionality and makes it harder to find an optimal solution during fitting.

5.3.3. Multi Input Multi Output RNN

What is unique about the RNN compared to other forecasting methods is its capability to generate instant multi-outputs given in Equation 5.4. This can be very useful in real-time applications as predictions are all generated simultaneously, while incremental models depend on generating one prediction at a time. Assuming a multivariate input and a univariate multi output, a training process of a hypothetical RNN can be viewed in Equation 5.16 where there are n rows in a single multivariate input, m total inputs and a single multi output of size p.

$$
\begin{aligned}
\mathbf{RNN}(\begin{bmatrix} a_1 & b_1 & \dots & z_1 \\ a_2 & b_2 & \dots & z_2 \\ a_3 & b_3 & \dots & z_3 \\ \vdots & \vdots & \vdots & \vdots \\ a_n & b_n & \dots & z_n \end{bmatrix}) &= \begin{bmatrix} \hat{y}_1 & \hat{y}_2 & \dots & \hat{y}_p \end{bmatrix}^\intercal \\
\mathbf{RNN}(\begin{bmatrix} a_2 & b_2 & \dots & z_2 \\ a_3 & b_3 & \dots & z_3 \\ a_4 & b_4 & \dots & z_4 \\ \vdots & \vdots & \vdots & \vdots \\ a_{n+1} & b_{n+1} & \dots & z_{n+1} \end{bmatrix}) &= \begin{bmatrix} \hat{y}_{p+1} & \hat{y}_{p+2} & \dots & \hat{y}_{2p} \end{bmatrix}^\intercal \qquad (5.16) \\
&\vdots \\
\mathbf{RNN}(\begin{bmatrix} a_{m-3} & b_{m-3} & \dots & z_{m-3} \\ a_{m-2} & b_{m-2} & \dots & z_{m-2} \\ a_{m-1} & b_{m-1} & \dots & z_{m-1} \\ \vdots & \vdots & \vdots & \vdots \\ a_m & b_m & \dots & z_m \end{bmatrix}) &= \begin{bmatrix} \hat{y}_m & \hat{y}_{m+1} & \dots & \hat{y}_{mp} \end{bmatrix}^\intercal
\end{aligned}
$$

5.4. Ensemble Learning

In real world applications ensemble learning for regression problems tends to perform better than a single model. Instead of just depending on a single model like ARIMA, prophet or LSTM, the contributions of several additional models are assembled together in a specific way. Ensemble learning will always make better predictions than any single contributing model if the models are less correlated. The idea is that using uncorrelated models may solve a weakness some other model might have, and with their combined strength this will result in a better overall prediction [83].

5.4.1. Gradient Boosting Machines

Gradient Boosting is a very popular ML algorithm developed to utilize a combination of weak learners to create a good predictor [84]. Gradient boosting machines was one of the first sequential ensemble methods of its kind created by Friedman, and has evolved to many of the state-of-the-art tree-based algorithms such as XGBoost, CatBoost and LightGBM [85, 86, 87].

Gradient boosting machines rely on building decision trees. The algorithm for fitting decision trees tries to build itself recursively and greedily from the root and branches out by splitting up the input data into left and right branch nodes based on different conditions or "if-statements". The algorithm tries different splitting conditions and measures the success based on trying to minimize MSE for the

current step. The goal is to get leaf/end nodes in the tree that are "pure", or in other words, the leaf node should only represent one type of data. This all happens in the fitting stage. The aim of gradient boosting is to train an ensemble of trees. This technique is called boosting, as it is expected that an ensemble will work better than a single estimator.

The gradient boosting algorithm at each iteration, constructs a tree d_{tree}^{n+1} as seen in Equation 5.17. This tree is then trained to optimize the residual, which is the ground truth subtracted by the predictions by the existing sequential ensemble. Finally it is added to the ensemble and the iterative process continues till a set amount of trees are added to the ensemble [88].

$$\begin{aligned} \textbf{ENSEMBLE}^n(X) &= d_{tree}^1(X) + d_{tree}^2(X) + \ldots + d_{tree}^n(X) \\ d_{tree}^{n+1}(X) &= y - \textbf{ENSEMBLE}^n(X) \end{aligned} \tag{5.17}$$

5.4.2. Stacking

Stacking uses output predictions of base models as input to a second-level model, usually called the meta-learner. However one cannot simply train the base models on the full training data, generate predictions on the full test set and then output these for the second-level training. This would not lead to the benefits that stacking provides. Instead K-folds of the dataset are created, recall Section 5.1.4 on cross validation. Then each model is fitted on K-1 of the training set and predicts only on $\frac{1}{K}$ of the data set, this is done for K iterations until all data appears on the test set. All K predictions of a single model is concatenated into the size of the original test set vector, this is done for each of the models. Finally all these vectors are fed together as features to the meta-regressor, which produces the final predictions [89, 90]. Empirical evidence show that model stacking makes the model more robust to changes in the data set, allowing for better generalization [91]. This is because the stacking deduces the bias in a model on a particular data set, and then corrects said bias in the meta-learner [92].

5.4.3. Weight Averaging

A simple but powerful way to create a strong predictor is by using parallel ensembling. One way to think of parallel ensemble learning is weighting the predictions of multiple different models. Another way of parallel ensemble predictions is to have multiple models of the same type e.g. LSTM, but each LSTM model has different hyperparameters or fed with different features and then their predictions are weighted to get the final prediction. One can combine and experiment with endless types of parallel ensembling, as each dataset might work very well with a specific kind. In Equation 5.18 assuming predictions from N different models, then the final prediction $\hat{\mathbf{y}}_f$ is the weighted average of all the models. The applied weights are based on the calculated validation metric, where the Equation assumes a validation RMSE based weight as an example. Note that p can be chosen to

be any value greater than zero, from experiments tuning p can result in a better prediction.

$$\hat{\mathbf{y}}_f = \frac{\sum_{i=1}^{N} \hat{\mathbf{y}}_i w_i}{\sum_{i=1}^{N} w_i}$$
$$w_i = \frac{1}{(\sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k^{val} - \hat{y}_k^{val})^2})^p} \tag{5.18}$$

5.5. Time Series Forecasting Model

First the raw time series data from Disruptive Technologies sensors are preprocessed, by downsampling the data into a frequency of five minutes. This is because all the sensors are sampled within five, ten or fifteen minute intervals as well as every time a significant change happens, which makes each sensor have a different time series. The data is downsampled by using the mean value of time series within a five minute interval, and compressing that interval into one row of time series data. Since all temperature sensors are within a five minute or less frequency, no loss of data occurred. The final resulting data is then split into 22 763 training, 288 validation and 288 test data. Note 288 rows of recorded data represents a full day of data with a frequency of five minute, which is added to the dataset everyday.

As seen in Figure 5.4, a weight averaging ensemble is used as the final model. Gradient boosting machines and the stacked model worked very well with differencing transform of the target, while the LSTM model performed better with standard data normalization. For the Prophet model, providing extra features increased performance. Also based on the RMSE validation score, hyperparameters of each single model is optimized using the Optuna Python library. The final forecasts on the validation and test set are achieved by using the weighted average based on the validation RMSE of each model. All the forecast models except for LSTM are setup as incremental forecasting models as seen in Equation 5.3, mainly because none of them support an instant multi-output target. LSTM on the other hand performed better as a sequence to sequence model and is therefore setup as a multi-output model as seen in Equation 5.4 rather than an incremental model.

5.5.1. Machine Learning Setup

The training of the ML models is done on a computer with the following specifications: Intel(R) Xeon(R) Gold 6140M CPU @ 2.30GHz for a CPU and NVIDIA Quadro RTX 5000 for GPU. The GPU is mainly used to speed up training of the LSTM utilizing NVIDIA's CUDA library [93, 94].

5.5.2. Forecasting Model Performance

Given only 24 hours of past fireplace data, Figure 5.5 and 5.6 shows two different forecasting scenarios.

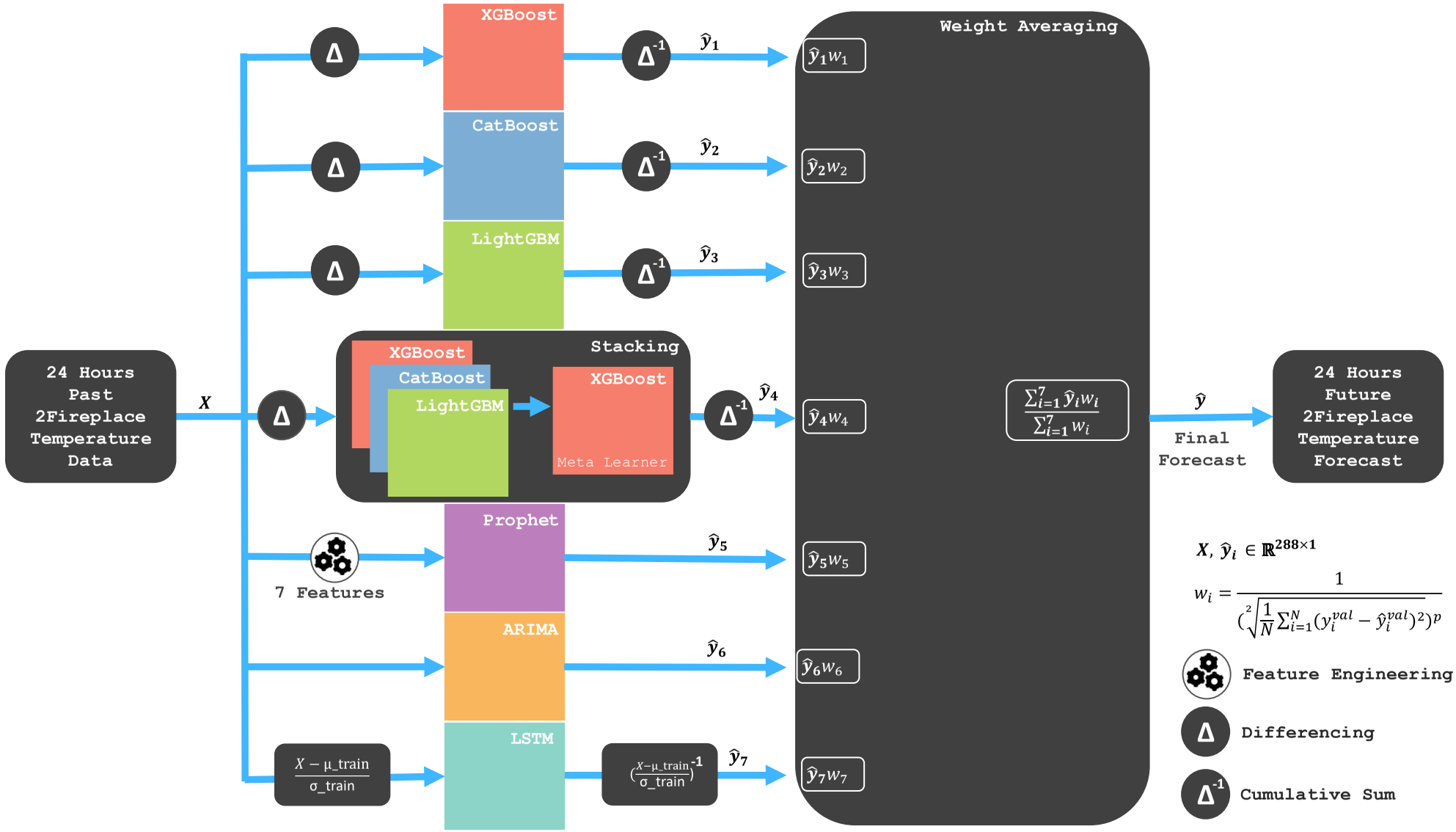


Figure 5.4.: High level view of the forecasting pipeline.

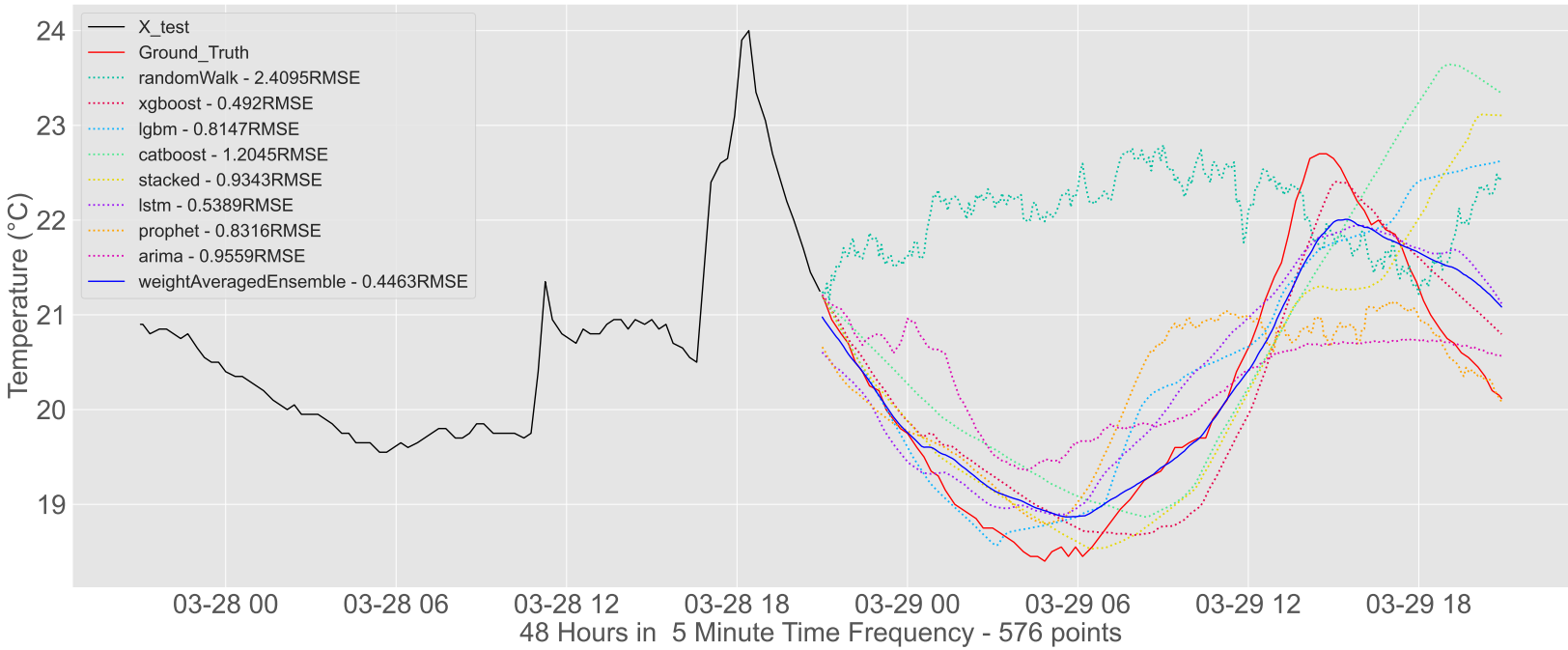


Figure 5.5.: 2Fireplace temperature forecast. Given 24 hours of past fireplace sensor data from Disruptive Technologies temperature sensor "2Fireplace" (black graph), these models are constructed to forecast 24 hours into the future of said sensor. The best forecasting model is the weight averaging ensemble (blue graph), weighing the contribution of each model seen in Figure 5.4. Note that the random walk is not part the forecasting model, but merely a way to demonstrate that the models can learn a pattern better than a baseline coin flip forecast. The ground truth is the red graph where the models are compared to that.

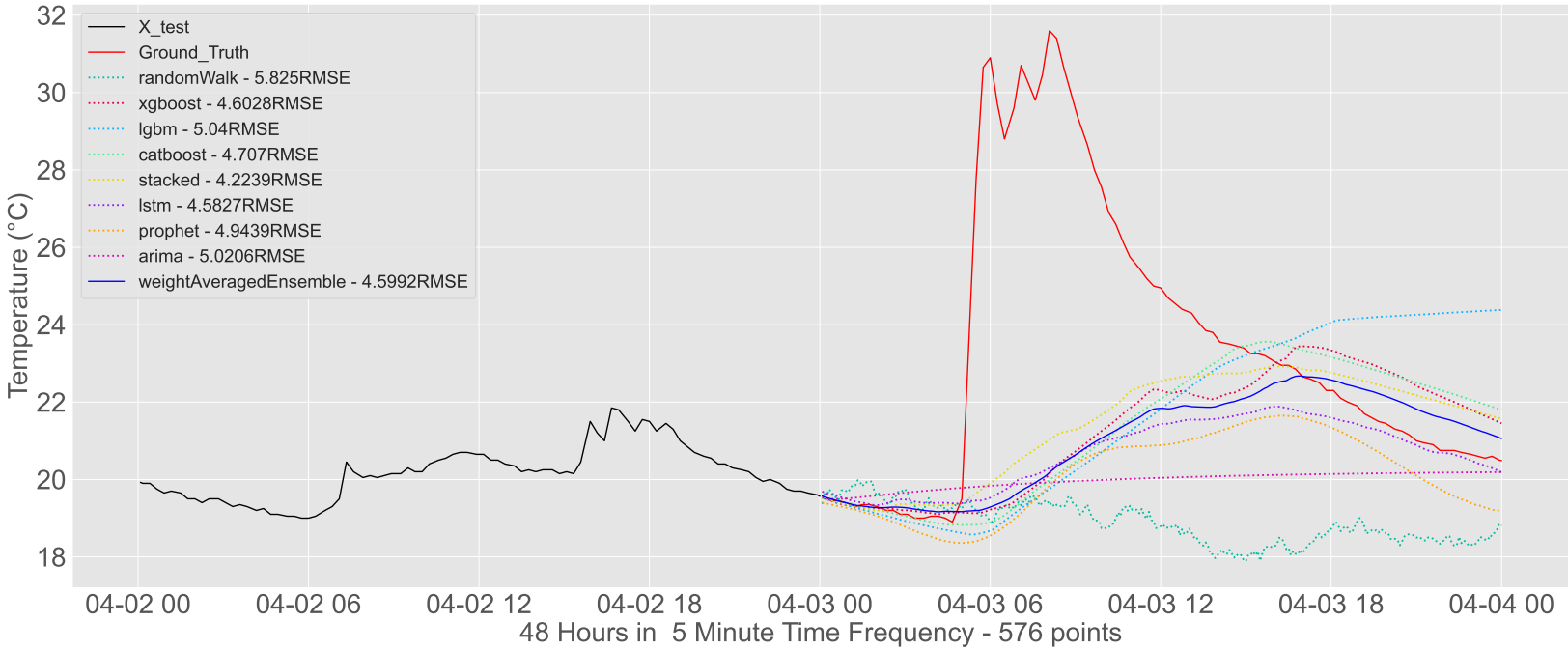
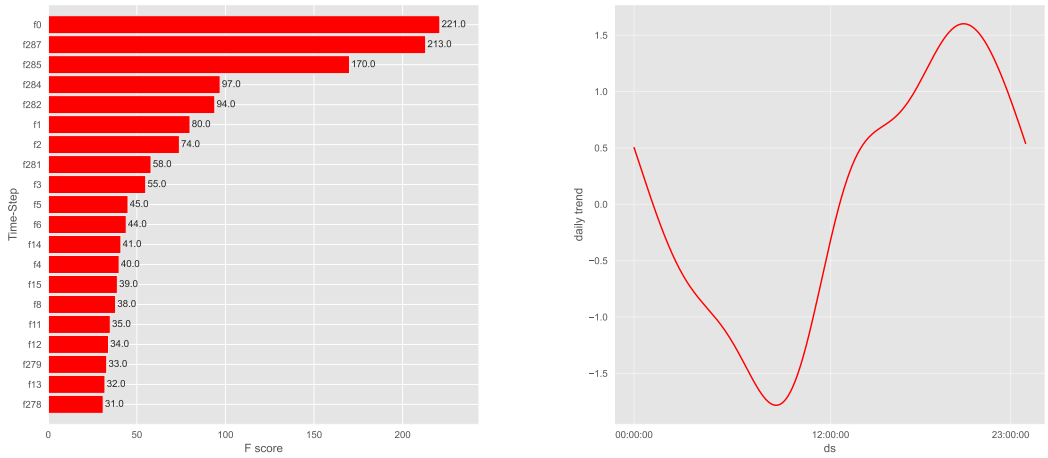


Figure 5.6.: 2Fireplace temperature forecast. This is a harder forecasting scenario where the fireplace is turned on at 05:00 in the morning. The way the model is constructed, there is no way the model can know if the fireplace is going to be turned on or not. 03.04.2022 is a day where it snowed in Trondheim, making it a particularly cold day in the spring season. In this case even model from Figure 5.4 struggles.

5.5.3. Forecasting Model Interpretation



(a) XGBoost forecaster feature importance. (b) Prophet Forecaster daily trends.

Figure 5.7.: Model interpretation of two of the forecasting models used in the ensemble in scenario two from Figure 5.6. XGBoost values the first time step and the last time step in the input data when making its forecasts in the future. Prophet bases its forecasts on an identified daily seasonality in the room temperature.

5.6. Time Series Prediction Model

In the prediction model, the data is preprocessed in a similar fashion to the forecasting model. In practice the prediction pipeline in Figure 5.8 can be used in two ways. First it can be used to predict the potential state of an out of commission sensor, assuming that the fireplace sensor in the room is still operational with data $\mathbf{X}$. Or it can be used to fill in the eight forecasted temperature sensors using the forecasted output $\hat{\mathbf{y}}$ of Figure 5.4, where in the context of Figure 5.8 the forecasted fireplace data is denoted as input $\hat{\mathbf{X}}$. Therefore assuming a temperature sensor deployed in the second floor dies after a few years, instead of replacing it, these sensors can be accurately predicted based on the fireplace sensor. Otherwise the prediction model can be seen as an extension of the forecasting model. Note that the distinction between forecasting and prediction model is established in Section 5.1.

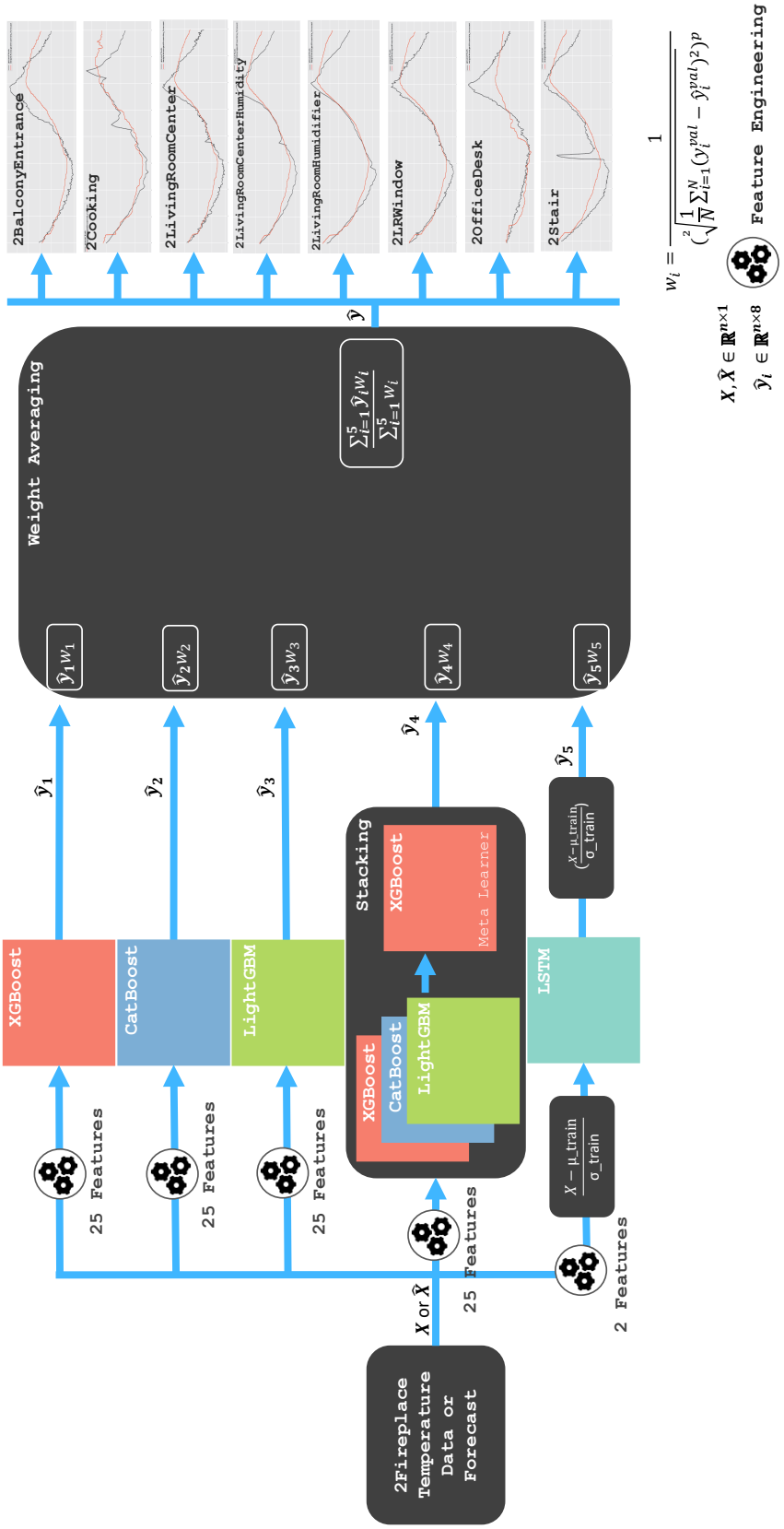


Figure 5.8.: High level view of the prediction pipeline.

5.6.1. Feature Engineering

This section goes through an exhaustive list of features used for the Prophet model in the forecasting pipeline as well as all the models in the prediction pipeline. The most important traits of engineered features is that these features need to be available for the forecasting interval. In other words, features that include information about the future which are unknown to us at the time of forecasting, cannot be included as a feature.

Time-series data have core components like seasonality, trend and cycles. Temperature data in the second floor has daily seasonality, for example the second floor is always colder during the nights than during the day. This can allow us to reasonably forecast the vicinity of where the next 24 hours of temperature might lie. Furthermore extracting date-based features and providing them as cyclic features to the model, may help the models discover new learning patterns in the data. These are categorical features in the form of integers of minutes, hours, days, weeks, months, seasons and years. The day is also split into a categorical feature that separates between night, morning, afternoon and evening. Notice that these features are known to us in the future.

Transformation features is also a simple form of feature engineering, that allows models to pick up details that otherwise are ignored. Simply taking the input data and transforming it into a log, square root and squared power are all possible features that lower the RMSE. This gives the model a possibility to take nonlinear data and add linear patterns for which is much easier for the model to learn.

Additionally domain knowledge based features are also quite significant to lower the error of the model. In this case adding historical weather observations from previous years that include outside temperature, humidity and sky conditions are all relevant [95]. This year's weather forecasts is the most significant feature for the model. Information about sunrise and sunset, based on the determinstic sun position prediction model. Other data that could be derived from weather data is dew point and frost point, where approximative equations exist assuming an interval for which satisfies the properties of room temperature [96, 97]. These are all features that are somehow related to the predictable parts of the inside temperature.

5.6.2. Data Leakage

The model from Figure 5.8 is, during training exposed to data leakage, as one of the features provided to the model is observed outside weather temperature from this year. For future forecasts one needs to fill this feature based on weather forecasts, but the training and validation data is based on historical observations rather than historical forecasts. Furthermore finding a long span of past historical temperature forecasts for our specific region proved to be challenging, and should have been recorded as part of the data accumulation process. This is not detrimental to the performance of the model, as the feature is fed with weather forecasts for real test

set scenarios. It is however important to be aware of this leakage regardless of its insignificance, and ideally exchange the feature with historical forecasts.

5.7. Sun Position Prediction Model

The approximate algorithm that is used in the current implementation of the predictive DT is inspired by [98] which is taken from Montenbruck's book on algorithms about astronomical phenomena. The precision of these calculations are in the range of 01.03.1900 till 28.02.2100 as stated by the author [99]. The resulting sun position algorithm in Unity as seen in appendix A, is accurate enough that it can be used for external lighting simulations on the house. From Figure A.1 in the appendix one can see that the azimuth in Unity is the same as the algorithms from two different external sources. There are how ever small deviations in the altitude compared to the NOAA and SunCalc sun position algorithms [100, 101]. The variations of the altitude are small enough that they are negligible for the purposes of this project. Note that the NOAA and SunCalc sun position algorithms are also predictive algorithms and as such they are susceptible to errors, but they still function as a good sanity check for knowing if the implemented algorithm is accurate enough [2].

Table 5.2.: Symbols used in the sun position algorithm.

Symbols	Description
C	Sun's center
JD	Julian Date
JC	Julian Century
L_0	Sun's Mean Longitude
M_0	Sun's Mean Anomaly
λ	Sun's Ecliptic Longitude
β	Sun's Ecliptic Latitude
Ω	The Earth's Obliquity of the Ecliptic
α	Right Ascension
δ	Declination
t_{UT}	Universal Time
t_{SR}	Sidereal Time Greenwich
t_{SRUT}	Sidereal Time Greenwich for Universal time
t_{LSR}	Local Sidereal Time
HA	Hour Angle of Object
L	Input Longitude
B	Input Latitude
ϕ	Azimuth from North
θ	Altitude
$D/M/Y$	Input Day/Month/Year

Most of the theory regarding the astrophysical phenomena is outside the scope of

this project. Therefore only a short description of the different variables is going to be made. JD, JC, t_{UT}, t_{SR}, t_{SRUT} and t_{LSR} are all used for the simple purpose of converting the input date to the correct date format for calculating the azimuth and altitude of the sun. L_0 and M_0 are used to calculate the ecliptic coordinates of the sun, λ and β, where the ecliptic coordinate system is used to represent the apparent positions of any solar system object. Furthermore the ecliptic coordinates of the sun together with Ω relate to the equatorial coordinates α and δ. This way it is possible to calculate the position of the sun relative to the earth, where the earth is in the origin of the equatorial coordinate system. Finally these coordinates can be converted into an azimuth and altitude angles, ϕ and θ, such that the output sun coordinates are relative to a stationary point on the earth's surface. In our case, the stationary point is the longitude and latitude positions of the house.

First it is important to find the number of days relative to the reference date and time of 1.January.2000, 12h Universal Time (J2000). This is also known as the Julian Date. Note all symbols for these equations are described in Table 5.2.

$$JD = 367Y - \frac{7}{4}(Y + \frac{9}{12}M) + \frac{275}{9}M + D - 730531.5$$

This relates to the Julian Century

$$JC = \frac{JD}{36525}$$

Furthermore using Julian Century, Sidereal time (given in hours) can be defined. Which is the meridian of Greenwich at midnight (00h) for a given date as well as the conversion from sidereal time of the Greenwich meridian for Universal Time can be calculated as follows.

$$t_{SR} = 6.6974 + 2400.0513JC$$
$$t_{SRUT} = t_{SR} + \frac{366.2422}{365.2422}t_{UT}$$

Where 365.2422 is the length of a tropical year given in days. Finally the local sidereal time for a geographical longitude L can be found as such.

$$t_{LSR} = t_{SRUT} + L$$

Local sidereal time will come in handy later when calculating the altitude and azimuth of the sun. Firstly numbers of days d from J2000 for the input date is given as.

$$JD_d = JD + \frac{t_{UT}}{24}$$

Which is needed to calculate the relative centuries from the reference time.

$$JC_d = \frac{JD_d}{36525}$$

Using the Julian century of the input date the sun's mean longitude and its mean anomaly can be calculated.

$$L_0 = 280.466 + 36000.770JC_d$$

$$M_0 = 357.529 + 35999.050JC_d$$

The sun's equation of center C is given as

$$C = (1.915 - 0.005JC_d)\sin(M_0) + 0.020\sin(2M_0)$$

Using the Equation of center of the sun the ecliptic longitude can be found. Note that the ecliptic latitude is approximately zero ($\beta \approx 0$).

$$\lambda = L_0 + C$$

There are many intermediate calculations that needs to be done in order to get the correct position of the sun relative to where a persons relative geographical longitude and latitude. Consequently in order to find the azimuth and altitude it is important to find the Sun's equatorial coordinates, right ascension α and declination δ which are both relying on the obliquity of the ecliptic Ω.

$$\Omega = 23.439 - 0.013JC_d$$

$$\alpha = \arctan(\tan(\lambda)\cos(\Omega))$$

$$\delta = \arctan(\sin(\alpha)\sin(\Omega))$$

Now it is possible to proceed to find the horizontal coordinates for the sun for a given input of geographical longitude L and latitude B. First the hour angle of the object is given as.

$$HA = t_{LSR} - \alpha$$

Resulting in the final Equations for the altitude of the sun θ and the azimuth ϕ respectively, where the algorithmic version of the azimuth ϕ is calculated using Arctan2 as suggested by [102].

$$\theta = \arcsin(\sin(B)\sin(\delta) + \cos(B)\cos(HA))$$

$$\phi = \arctan\left(\frac{-\sin(HA)}{\tan(\delta)\cos(B) - \sin(B)\cos(HA)}\right)$$

Naturally there is a lot of theory behind all these calculations, but the comprehensive and detailed theory as to why all of these calculations have been done in a certain way will not be motivated in this project. The nature of this simulation is a significant piece in the implementation of a DT for buildings, as it provides another layer of realness to the virtual environment and arguable a step closer to relaying a 1-to-1 experience of the DT. The value of this algorithm comes from how it can significantly improve the experience of the virtual house, by utilizing the position of the sun to simulate accurate external lighting on the asset [2].

5.8. Demonstration of Predictive Digital Twin

The predictive DT comes with a complementary demonstration video in this link *https://youtu.be/xXt5FFaVokQ?t = 668* (11:10-16:00). A live demo of the concept similar to the one shown in the demonstration and video can also be done onsite.

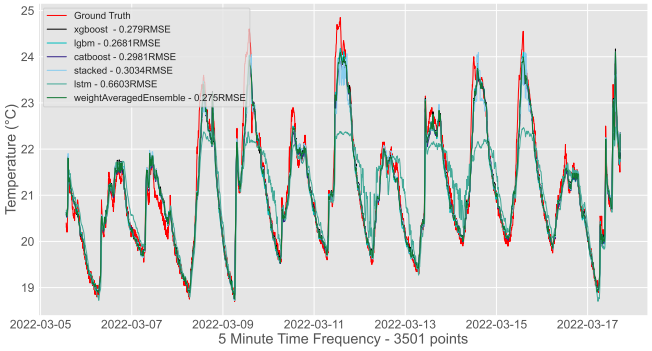
Section 5.8.1 and 5.8.2 are related to the DDM model for the second floor temperature forecasting model. Section 5.8.3 is a demonstration of the PBM model.

5.8.1. Temperature Predictive Model Performance

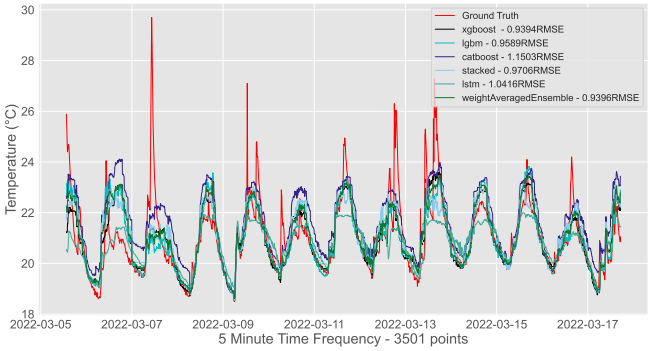
Figure 5.9 shows how the predictive model pipeline, can be used to fill out missing values for the eight other temperature sensors, given that only the fireplace sensor is available. That makes it possible to still keep track of the temperature in the room even though some of the sensors might go out of commission in the future.

5.8.2. Temperature Forecasting Model Performance

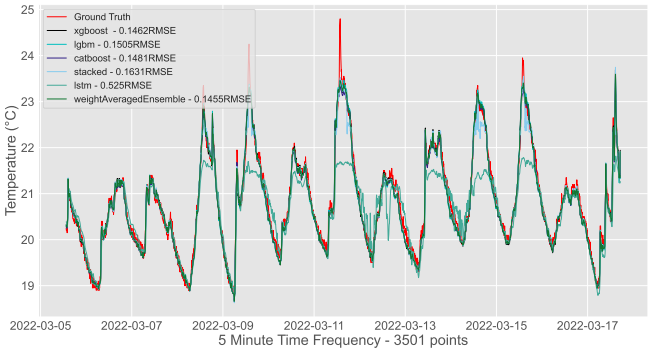
Figures 5.10 and 5.11 are forecasting scenarios presented in a graphical sense for showcasing performance of the forecasting model. To integrate the forecasting pipeline with the Unity VR setup, Pythonanywhere is used to host the python script and models where the forecasting task is being executed every 24 hours. Then the forecasted output is stored in the Dweet RESTful API, where Unity uses HTTP requests to access it. The endpoints are 'https://dweet.io/get/latest/dweet /for/{sensor_name}", where {sensor_name} is a temperature sensor from the second floor following the convention set in Figure 3.2. Note that Pythonanywhere was also used for the sensor data collection and storage. Ideally a better setup would be to have the Python scripts be built into the VR build, but this was a more convenient solution for quick demonstrations of the concept, as seen in Figure 5.12.



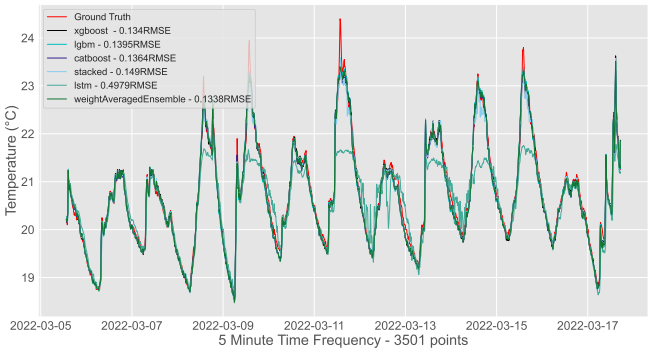
(a) 2BalconyEntrance temperature prediction.



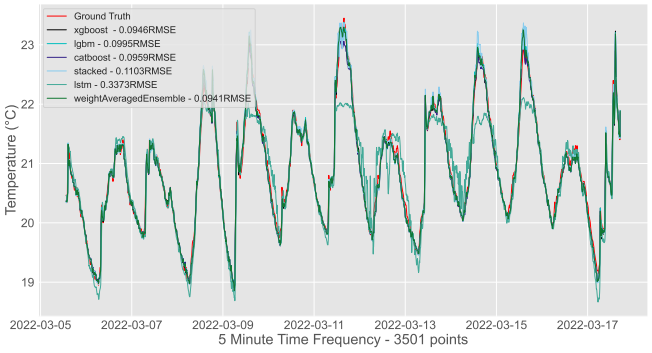
(b) 2Cooking temperature prediction.



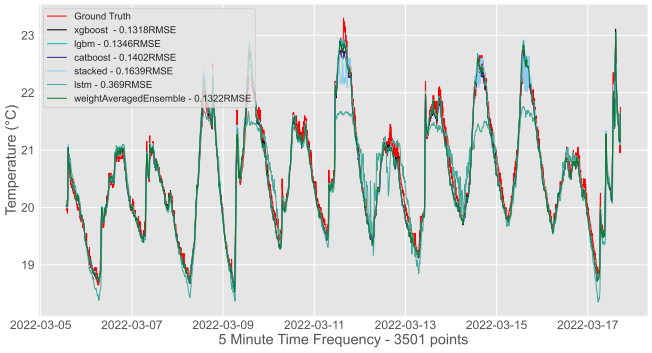
(c) 2LivingRoomCenter temperature prediction.



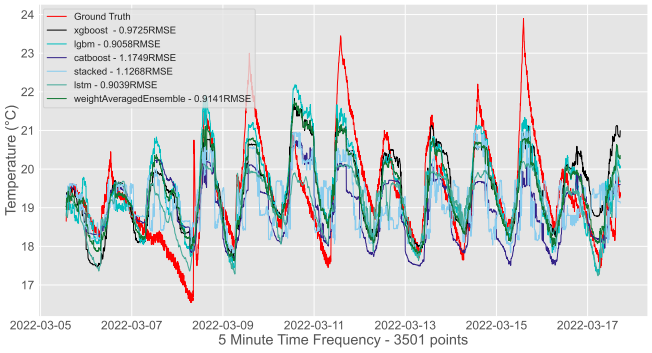
(d) 2LivingRoomCenterHumidity temperature prediction.



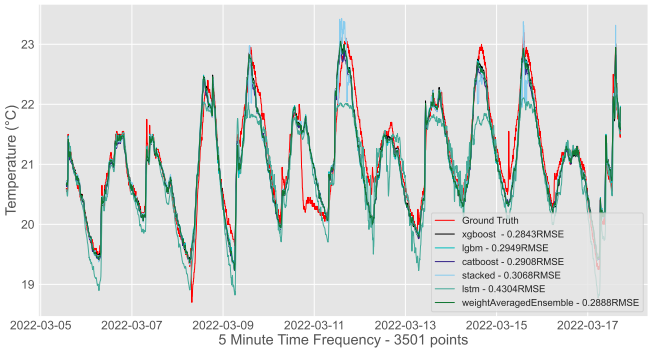
(e) 2LivingRoomHumidifier temperature prediction.



(f) 2LRWindow temperature prediction.

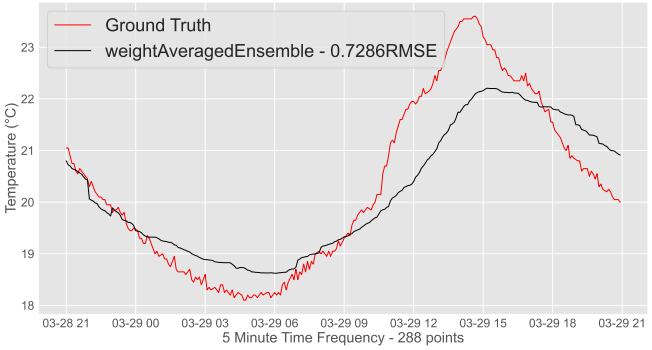


(g) 2OfficeDesk temperature prediction.

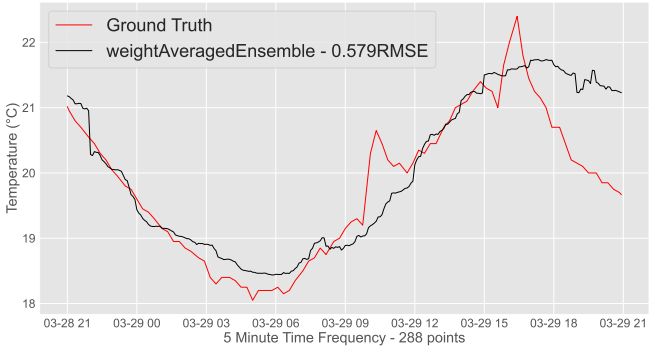


(h) 2Stair temperature prediction.

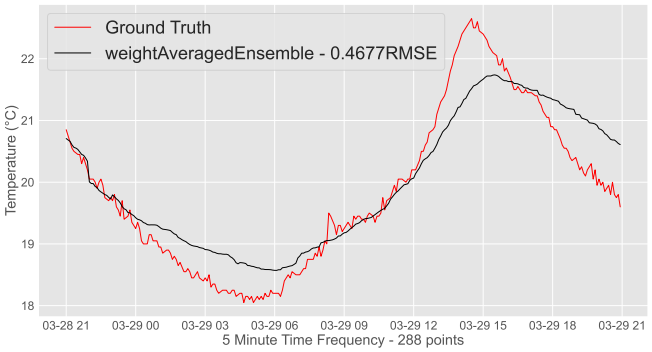
Figure 5.9.: Given that the fireplace data sensor data is known but the remaining temperature sensors of second floor have missing values, where this Figure shows the performance of different regression models. The best model is the weight averaging ensemble from Figure 5.8, which weighs the contribution of each model based on their validation set RMSE.



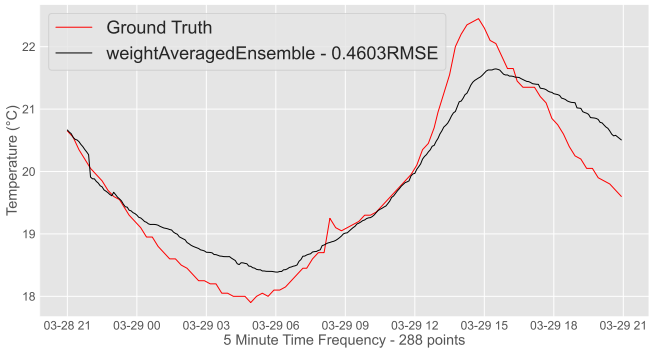
(a) 2BalconyEntrance temperature forecast.



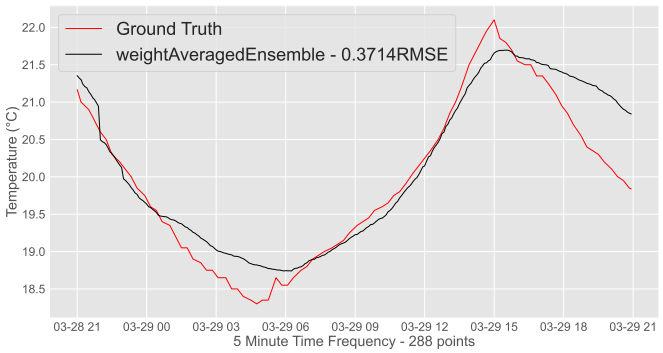
(b) 2Cooking temperature forecast.



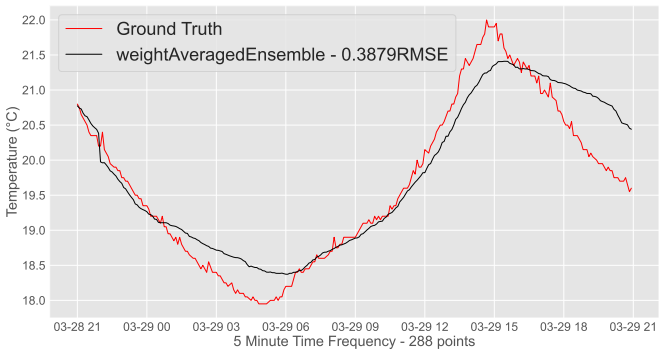
(c) 2LivingRoomCenter temperature forecast.



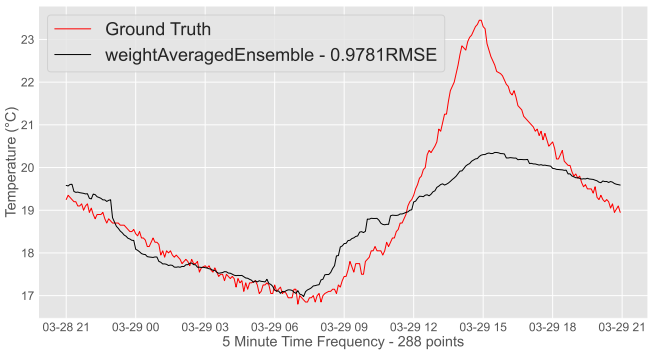
(d) 2LivingRoomCenterHuidity temperature forecast.



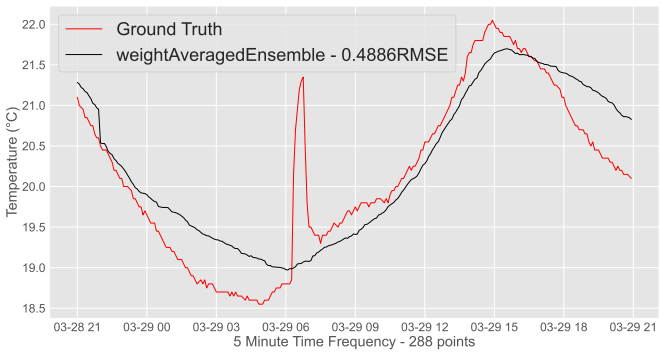
(e) 2LivingRoomHumidifier temperature forecast.



(f) 2LRWindow temperature forecast.

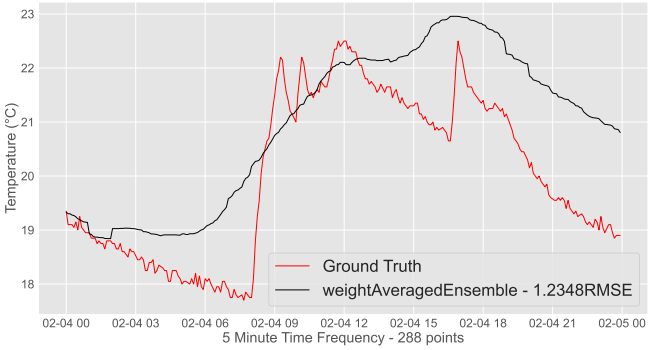


(g) 2OfficeDesk temperature forecast.

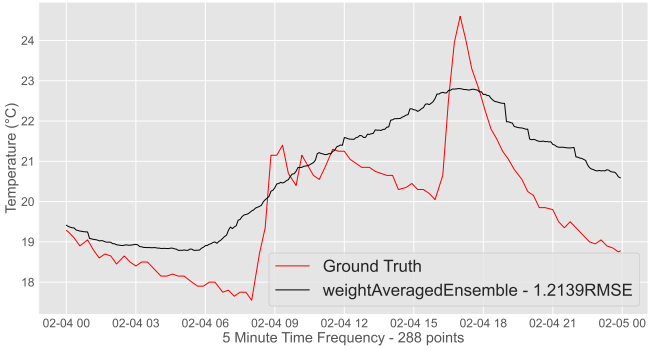


(h) 2Stair temperature forecast.

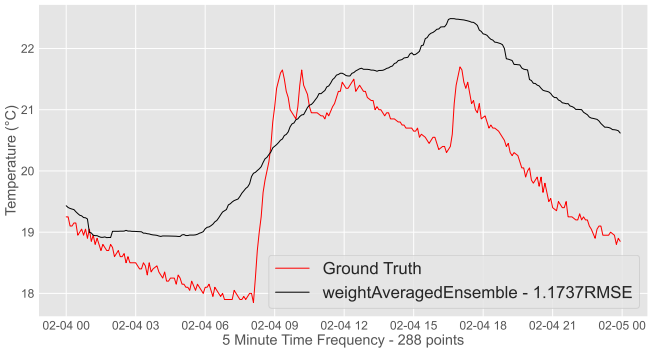
Figure 5.10.: Using the output of the fireplace forecaster in Figure 5.5, we can feed it to the weighted average prediction model from Figure 5.8, similarly to the one used to output Figure 5.9 and generate forecasts for the remaining second floor temperatures.



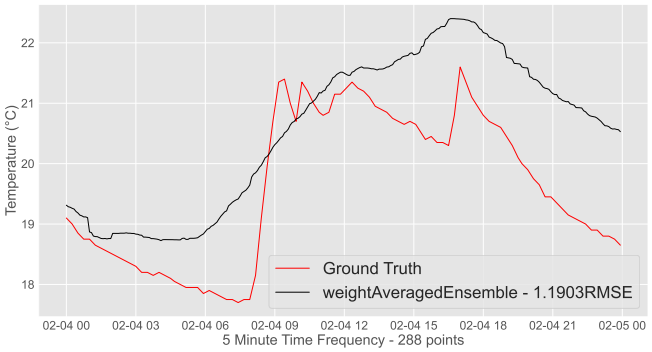
(a) 2BalconyEntrance temperature forecast.



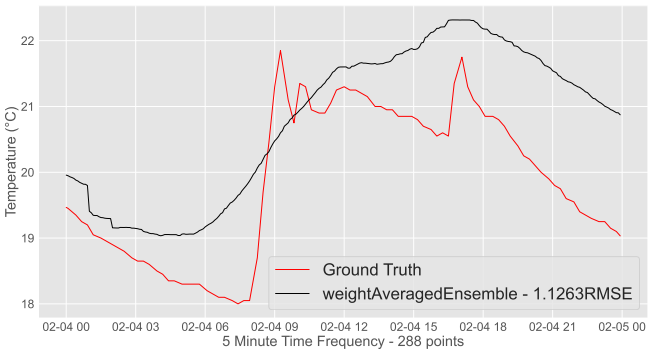
(b) 2Cooking temperature forecast.



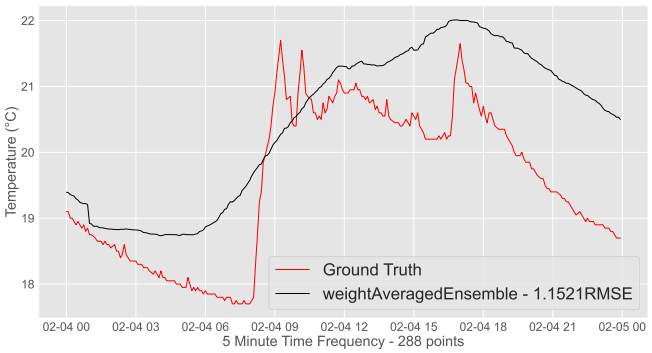
(c) 2LivingRoomCenter temperature forecast.



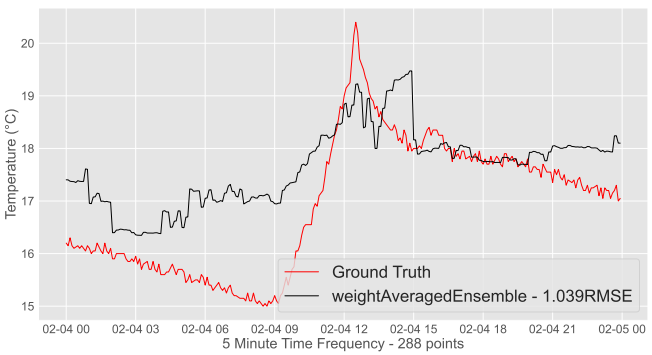
(d) 2LivingRoomCenterHumidity temperature forecast.



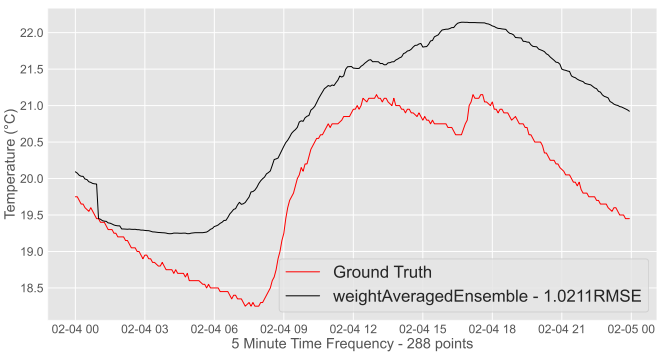
(e) 2LivingRoomHumidifier temperature forecast.



(f) 2LRWindow temperature forecast.

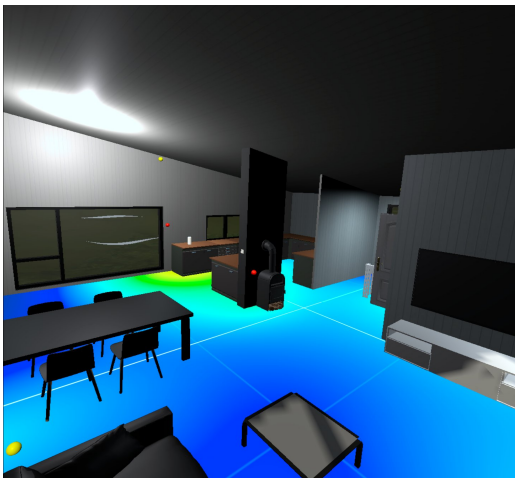


(g) 2OfficeDesk temperature forecast.

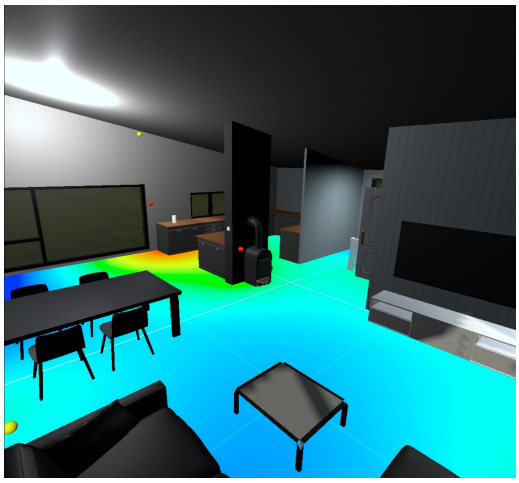


(h) 2Stair temperature forecast.

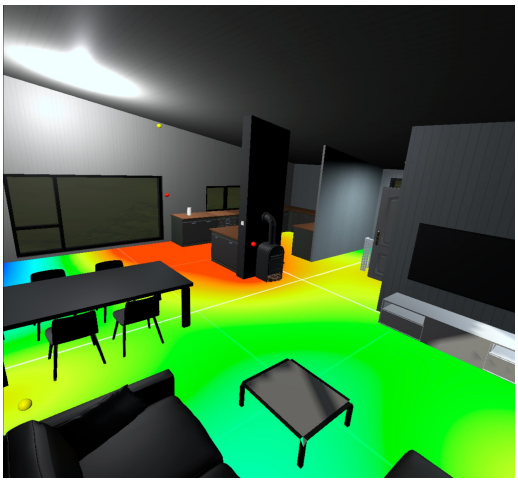
Figure 5.11.: Using the output of the fireplace forecaster in Figure 5.6, we can feed it to the weighted average prediction model from Figure 5.8, and generate forecasts for the remaining sensors. As seen here, if there is a big forecasting error in the fireplace then this greatly affects the remaining forecasts.



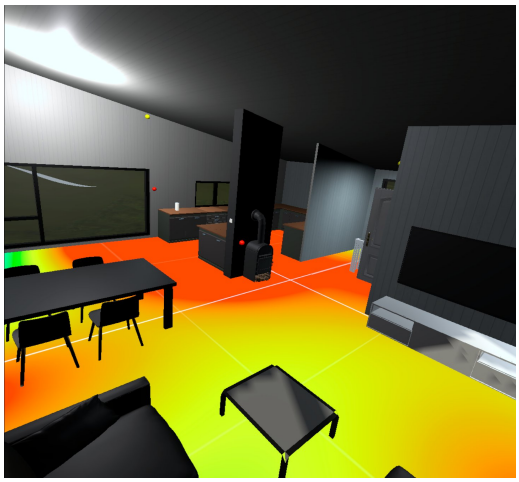
(a) 00:00.



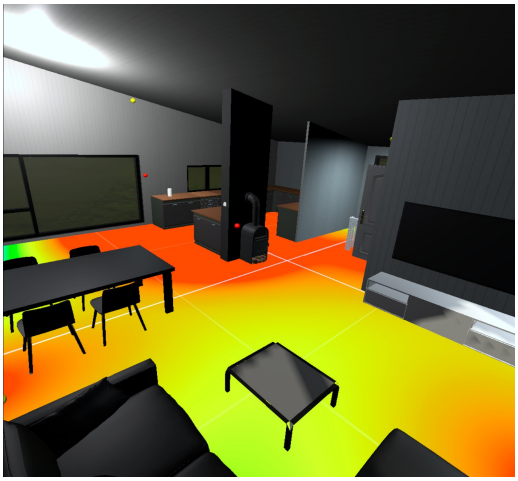
(b) 04:00.



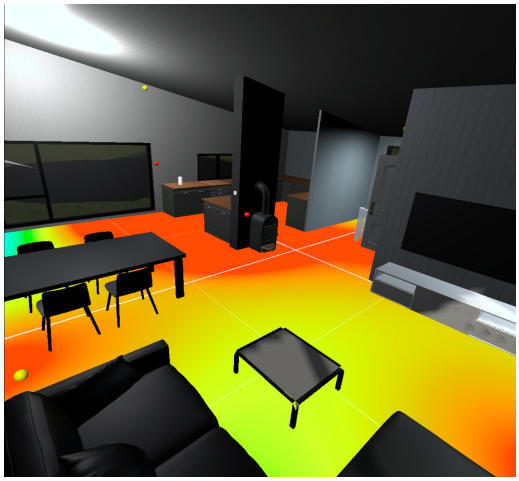
(c) 10:00.



(d) 12:00.



(e) 16:00.



(f) 20:00.

Figure 5.12.: Using the forecasting model to forecast the next 24 hours in the future (24.04.2022), visualized in the temperature heat map made for the diagnostic DT. The visualization shows how the model believes that the temperature in the second floor will develop the next day.

5.8.3. Sun Position Prediction Model Performance

Figure 5.13 shows the predictive sun model for each month, for informing decision making around sun hours for the entire year. Furthermore Figure 5.14 and 5.15 shows the same algorithm used in VR.

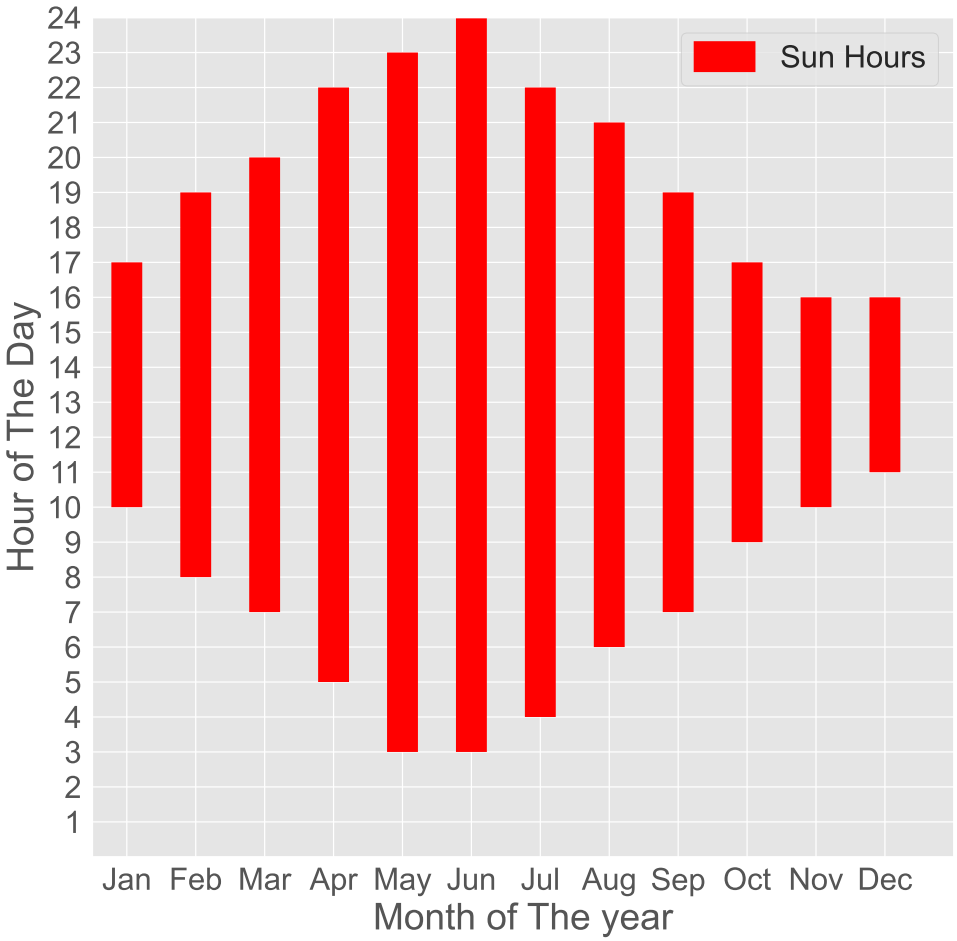
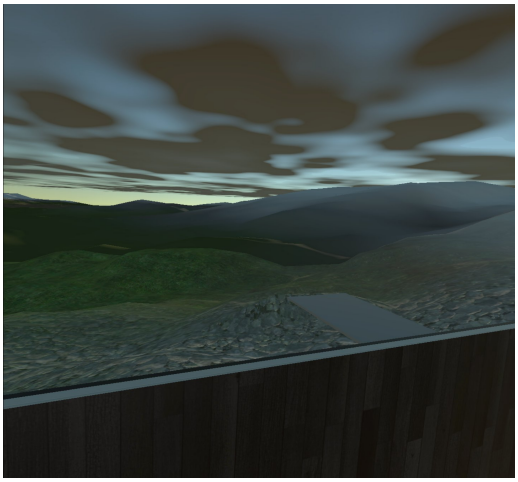
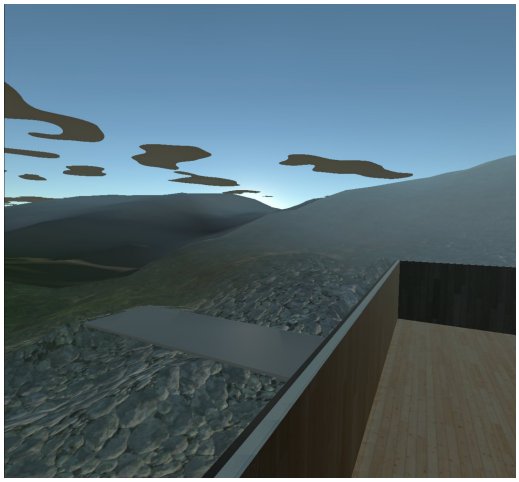


Figure 5.13.: Predictive sun position model used to give range of sun light for last day of each month in the year 2022 from the location of the house.

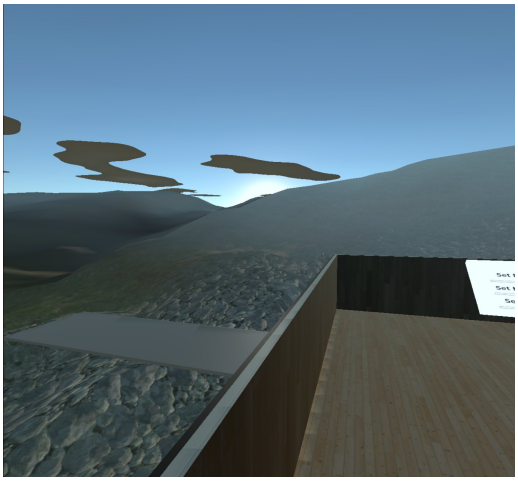
The significance of knowing about how many sun hours, a house gets in the span of a day is a key factor for any potential homeowner. Therefore having a PBM that accurately displays that future prediction both as a bar plot of the entire year as well as viewing a specific day visually within the VR setup, allows for better decision-making around the biggest investment that majority of people go through. While the bar plot does show you the sun hours, the value of the visual demonstration is that it fills in the gaps by showing exactly which part of the house will be exposed in the sun. For instance if the terrace or balcony does not get a lot of sun exposure, some buyers would become disinterested with the property.



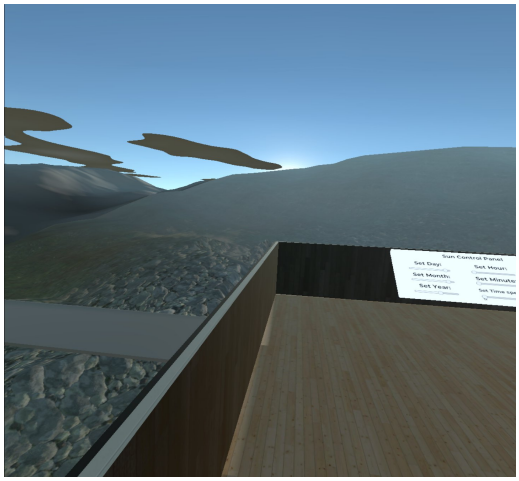
(a) 09:15.



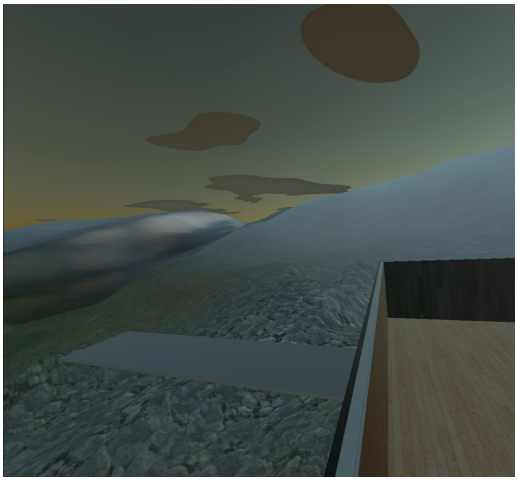
(b) 10:00.



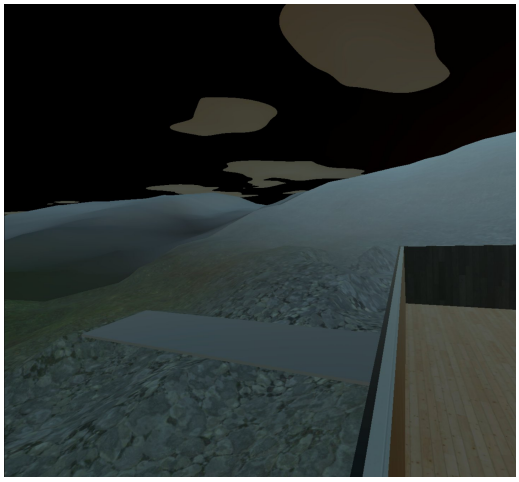
(c) 11:00.



(d) 12:00

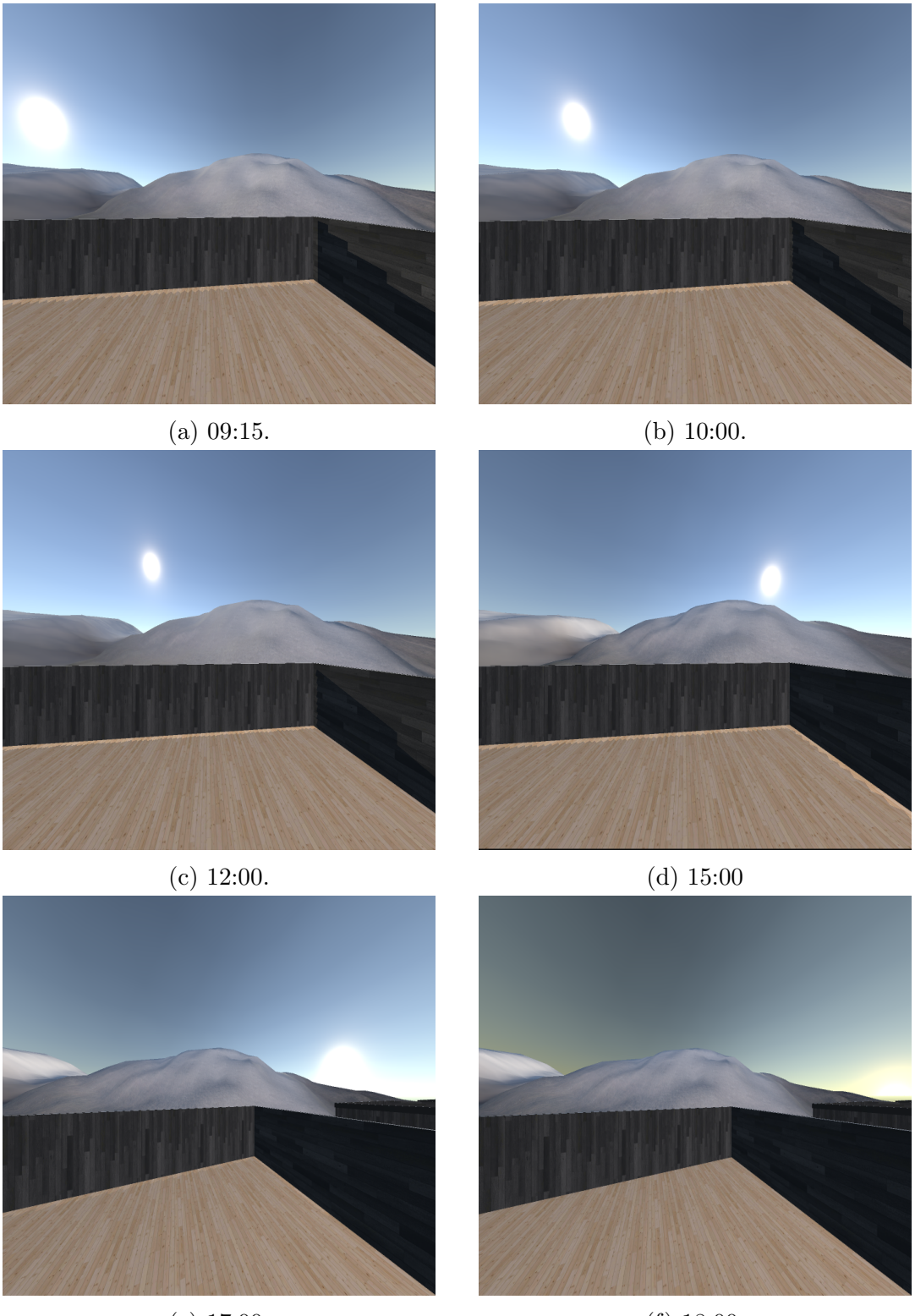


(e) 14:00.



(f) 15:00.

Figure 5.14.: Sunlight simulation in Unity using the sun position algorithm from appendix A. This observation was made originally at 29.11.2021, the scenario have been reconstructed in VR. The algorithm follows the real-time sun position, unless it is overridden with particular hour and minute values.



(a) 09:15. (b) 10:00.

(c) 12:00. (d) 15:00

(e) 17:00. (f) 18:00.

Figure 5.15.: Sunlight simulation in Unity using the sun position algorithm. This observation was made at 07.03.2022. Note that the view is from the balcony of the virtual house placed in the correct altitude, rotation and geographic location on a terrain generated based on a Trondheim height map from Kartverket.

5.8.4. Summary

As the installed sensors continue to generate high-quality sensor data, it sets up the asset for the possibility of a DDM predictive DT. This is demonstrated both graphically and in VR. Being able to track forecasted temperatures of the second floor allows the homeowner to better understand the temperature of the house, and make informed decisions before events take place. For example if it is forecasted to be cold the next day, then it would prepare the resident for turning on the fireplace.

On the other hand for PBM predictive DT, the approach depends on the complexity of the physics and how much existing research and technologies exist to support the simulations. The sun position is something that humans have been tracking for millennia, and our fascination with the sun has lead to a very precise scientific understanding of the concept. Thanks to these advances, using the Game Engine Unity to generate a sun simulation is possible. With the capabilities of being able to forecast sun positions, the homeowner can for example understand how they should install sun panels on their roofs to get maximum efficiency out of them. Another valuable use case is if a home buyer could make more informed decisions about property purchase, given that they are presented with the sun position simulation. This is as sun exposure is important to consider in a country such as Norway, and many homeowners value long exposure of sunshine as this can be scarce in some seasons of the year.

6. Prescriptive Digital Twin

4 Now that the homeowner has a predictive digital twin, they can foresee possible future scenarios based on past data or physics simulations. However, they are not only interested in the forecasts but specific recommendations in the house based on their past behavior. Such as what time of day is it recommended for the user to turn on the fireplace, given the temperature profile of tomorrow. Perhaps there is also a cluster of neighbors with a digital twin setup, and the recommendations can be supported by all of them if they share a similar behavioral pattern.

This chapter is about the setup and methodology used to achieve the prescriptive DT. The prescriptive DT, as seen in Figure 6.1, is the fourth capability of a DT pipeline. In this level, the DT is able to make recommendations based on what-if? scenarios, and through that support uncertainty quantification. What-if scenarios in our case can be potential weather forecasts, or historical weather conditions in the past that may have led to certain behaviors.

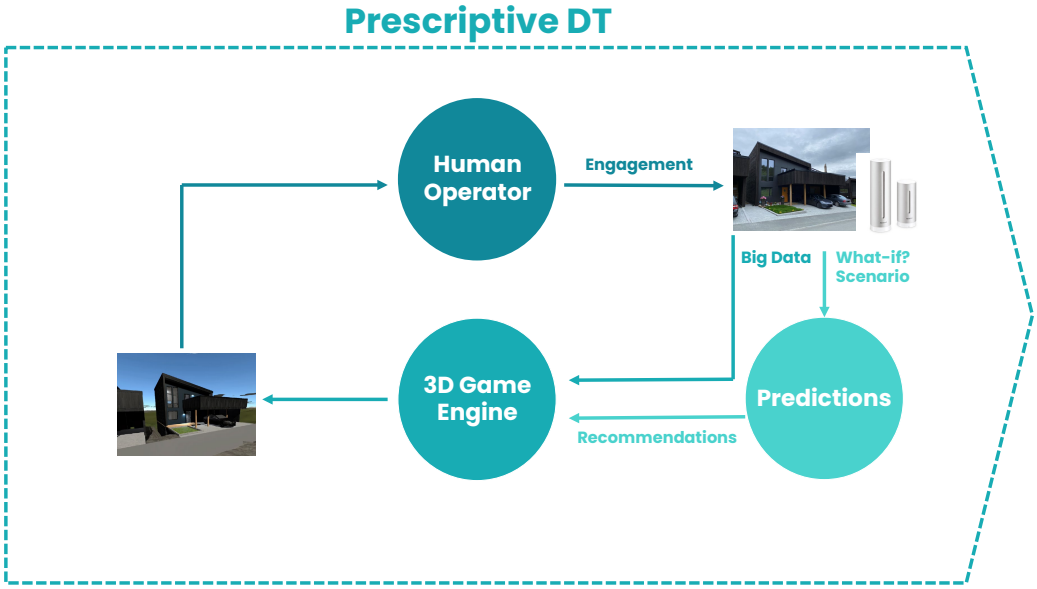


Figure 6.1.: Visualization of the prescriptive DT. It is similar to the diagnostic DT with a human operator in the loop, but now predictions are also integrated into it.

Behaviours can certainly be used to motivate future recommendations, given if such a situation would again occur sometime in the future. The recommendations

can be used to give experts a decision support system, but for our case such a system might be used for uncertainty quantification of the unpredictable parts of the future forecasts, that have been modeled and highlighted in the predictive DT.

The current second floor future forecasts struggle with not being able to model temperature spikes caused by the fireplace, as seen in Figure 5.6. This can be solved by predicting the time of day that it is most likely that the fireplace is going to be turned on, given a certain outdoors temperature profile. The behaviour for when the fireplace is turned on is certainly linked to cold weather, which compels the residents of the house to switch on the fireplace. Hence being able to pinpoint this behaviour based on similar experiences from the past, will allow for uncertainty quantification of the predictive DT forecasting model and prompt recommendations for when to rekindle the fireplace. Furthermore another motivation, while this is not going to be pursued, one could broaden the recommendations by including how much firewood one needs and the price of the amount firewood. Though this would certainly be possible, this would require data about the wood that is being used in the fireplace, the energy effeciency of the fireplace, calorific values of the firewood and a thermodynamics model of the room, in order to give good recommendations about estimated cost of firewood to reach desired temperatures.

We now give a brief overview of the structure of this chapter. Section 6.1 is about the theory, setup and methodology related to the prescriptive recommendation system in the prescriptive DT. Section 6.2 demonstrates results related to the prescriptive DT.

6.1. Cast The Wood Into The Fire!

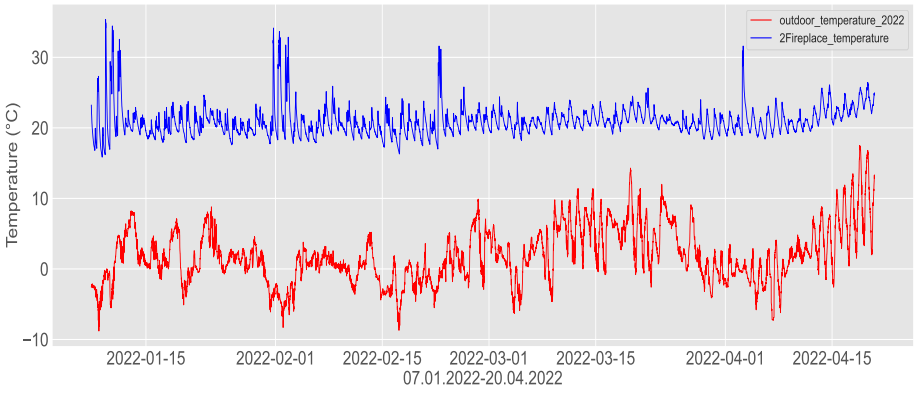


Figure 6.2.: Fireplace temperature and outdoors temperature comparison. Intuitively the outdoors temperature, has an affect on if the fireplace is lit on a particular day.

In Lord of The Rings when Elrond told Isildur to throw the One Ring into the

fires of Mount Doom, Isildur refused. This clearly must be because Elrond did not have a high level prescriptive DT that can prescribe the ideal timing for casting the Ring into the volcano. Analagous to this, is recommending the ideal timing to cast the firewood into the fireplace. For any homeowner wanting to maximize the ideal conditions for using the fireplace, this can be very valuable. Such a system would also be capable of prescribing the amount of wood needed for each firing event. In this section the theory and setup of the time of day recommendation system, for when the fireplace should be lit is presented.

6.1.1. Data Preprocessing

First the data is split into a 24 hour format, and then filtered for the days where there are temperature spikes greater or equal to 30 degrees Celsius. For the current cold season, the fireplace was used for warming up the room only on eight different occasions, as seen in Figure 6.3. Note for the prescriptive DT setup, the data ranges from 07.01.2022 to 19.04.2022. There are 102 full days with 288 timesteps each, where day zero corresponds to 08.01.2022 and day 102 correspond to 19.04.2022.

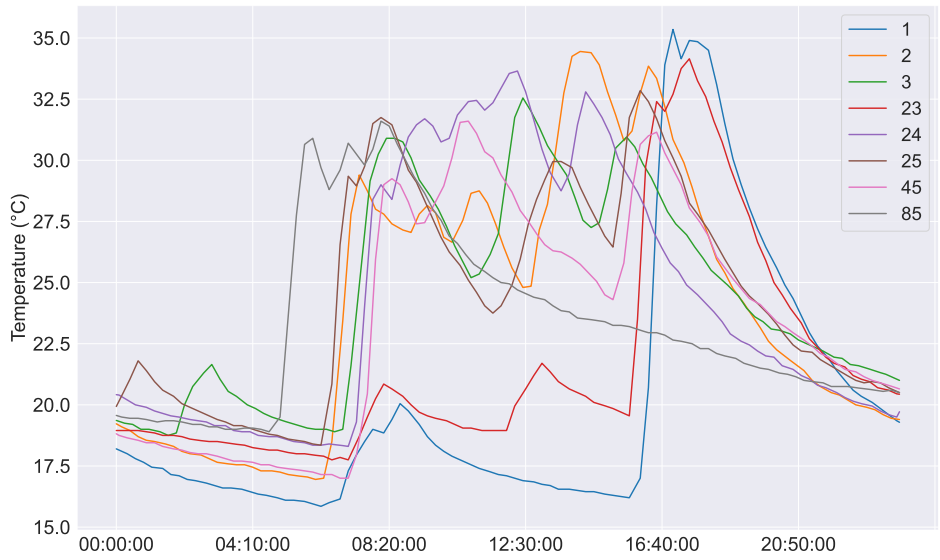


Figure 6.3.: Fireplace temperature data where each number is a days since 08.01.2022 where the fireplace has been turned on. This data is pre-processed in this fashion for the recommender system setup. Note the dataset is quite lacking as these eight scenarios are the only fireplace events in the entire collected dataset.

6.1.2. User-Based Collaborative Filtering

Collaborative filtering is an information filtering technique for making predictions about user A's interest or behaviour by collecting data from many users. The assumption is that if person A behaves similarly to person B in a specific context, then person A might behave similarly to person B in another context than a randomly chosen person from the population. Recommender systems are heavily utilized by Amazon, Netflix, YouTube and other services which tries to learn what you like based of other users that behave similarly to yourself [103]. This approach makes it possible to give each user unique recommendations based entirely on their behaviour, derived from other users.

The main challenge with this approach is that it requires a lot of data about the specific behaviour, not necessarily by the user in question but in general from the entire user-base from which the data is collected. Furthermore, there is the cold start problem, which is in this case that a new user registers and has not provided any interaction yet, therefore it is not possible to provide personalized recommendations [104]. At the beginning, the algorithm is not very accurate but becomes more precise as more data is collected while the user is active.

Homeowner (user) \ Days (items)	08.01.2022	09.01.2022	10.01.2022	11.01.2022	•••	19.04.2022
	Fireplace Not Used	16:20:00 (196)	06:40:00 (80)	07:15:00 (87)	⋮	Fireplace Not Used
	15:35:00 (187)	16:30:00 (198)	07:55:00 (95)	Fireplace Not Used	⋮	Fireplace Not Used
⋮	•••	•••	•••	•••	⋱	⋮
	Fireplace Not Used	Fireplace Not Used	Fireplace Not Used	20:50:00 (250)	•••	Fireplace Not Used

Figure 6.4.: Collaborative filtering matrix. Note that other than the house that we have collected data from in row one, the remaining houses and their data is artificial for the sake of demonstrating user based collaborative filtering.

Other project specific challenges apply here. As seen in Figure 6.3, there are only eight recorded fireplace events for 102 days. This is a very small dataset to work from, making it hard to conclude anything in particular about the user's fireplace lighting behaviour. Furthermore, no data has been collected from other

homeowners and there are therefore no other users, from where we can obtain our recommendations to the specific user. Hence there is insufficient data, as the sparsity of data comes from that most of the days the fireplace is simply not used. The motivation for demonstrating this particular method comes from its potential to become valuable, as more data is collected from this house and other locations.

The collaborative filtering for fireplace lighting events are set up with **U** homeowners (users) as the rows and **V** days (items) as columns. Note each day has specific information about the outdoor temperature of that certain day. In the collaborative filtering matrix from Figure 6.4, the fireplace timesteps for each single day is extracted and used as a row for a specific user. The timestep for when the fireplace was approximately turned on is inferred by finding the biggest difference between two points in a specific day, using differencing described in theory Section 5.1.3 Equation 5.7. The timestep for a lit fireplace for a specific day can be seen in the first row, as a red vertical line in Figure 6.4.

The user based collaborative (UBCF) pipeline as implemented is visualized in Figure 6.5. Given that an unknown input outdoor temperature forecast or scenario is given to the pipeline, first the RMSE between the input and each column specific outdoor temperature is calculated to find the most similar scenario to our input. For simplicity all users are assumed to have experienced the same outdoors temperature in Trondheim, therefore for users outside of Trondheim, RMSE specific to that user needs to be calculated. The remaining pipeline would still work the same way. Resulting in the RMSE weight vector $\mathbf{w}_{RMSE}$ seen in Equation 6.1.

The recommendation pipeline seen in Figure 6.5 is supposed to be for the first user. In parallel with finding $\mathbf{w}_{RMSE}$ for the user, the Pearson correlation of all the fireplace lighting event actions of said user is calculated against the other users in the population, assuming these exist. Then the highest correlated users based on some threshold are extracted as a weight vector $\mathbf{w}_{pearson}$ seen in Equation 6.1. Where χ_{pq} is the set of days both users p and q that have fireplace behavioral data.

Knowing the most similar scenarios to the input scenario and the most correlated users to the user we wish to make recommendations for, now we extract a matrix denoted **T**. This matrix is the region where both the green and red rectangles cross as seen in Figure 6.5. Each row of said matrix is weighted by the RMSE score of the specific user in a weighted average. This gives the predicted fireplace lighting behaviour of each individual user, given that specific scenario. That particular prediction vector $\mathbf{t}_{predicted}$ is then weighted by the most similar users to the particular user we are recommending for, to produce the final recommendation time to light the fireplace t_{n+1}.

$$\mathbf{w}_{RMSE} = \left[\frac{1}{(\sqrt{\frac{1}{N}\sum_{k=1}^{N}(T_k^{item1} - T^{input})^2})} \quad \cdots \quad \frac{1}{(\sqrt{\frac{1}{N}\sum_{k=1}^{N}(T_k^{itemV} - T^{input})^2})} \right] \tag{6.1}$$

$$\mathbf{w}_{pearson} = \left[\frac{\sum_{j\in\chi_{1i}}(t_{1j}-\bar{t}_1)(t_{ij}-\bar{t}_i)}{\sqrt{\sum_{j\in\chi_{1i}}(t_{1j}-\bar{t}_1)^2}\sqrt{\sum_{j\in\chi_{1i}}(t_{ij}-\bar{t}_i)^2}} \quad \cdots \quad \frac{\sum_{j\in\chi_{1U}}(t_{1j}-\bar{t}_1)(t_{Uj}-\bar{t}_U)}{\sqrt{\sum_{j\in\chi_{1U}}(t_{1j}-\bar{t}_1)^2}\sqrt{\sum_{j\in\chi_{1U}}(t_{Uj}-\bar{t}_U)^2}} \right]$$

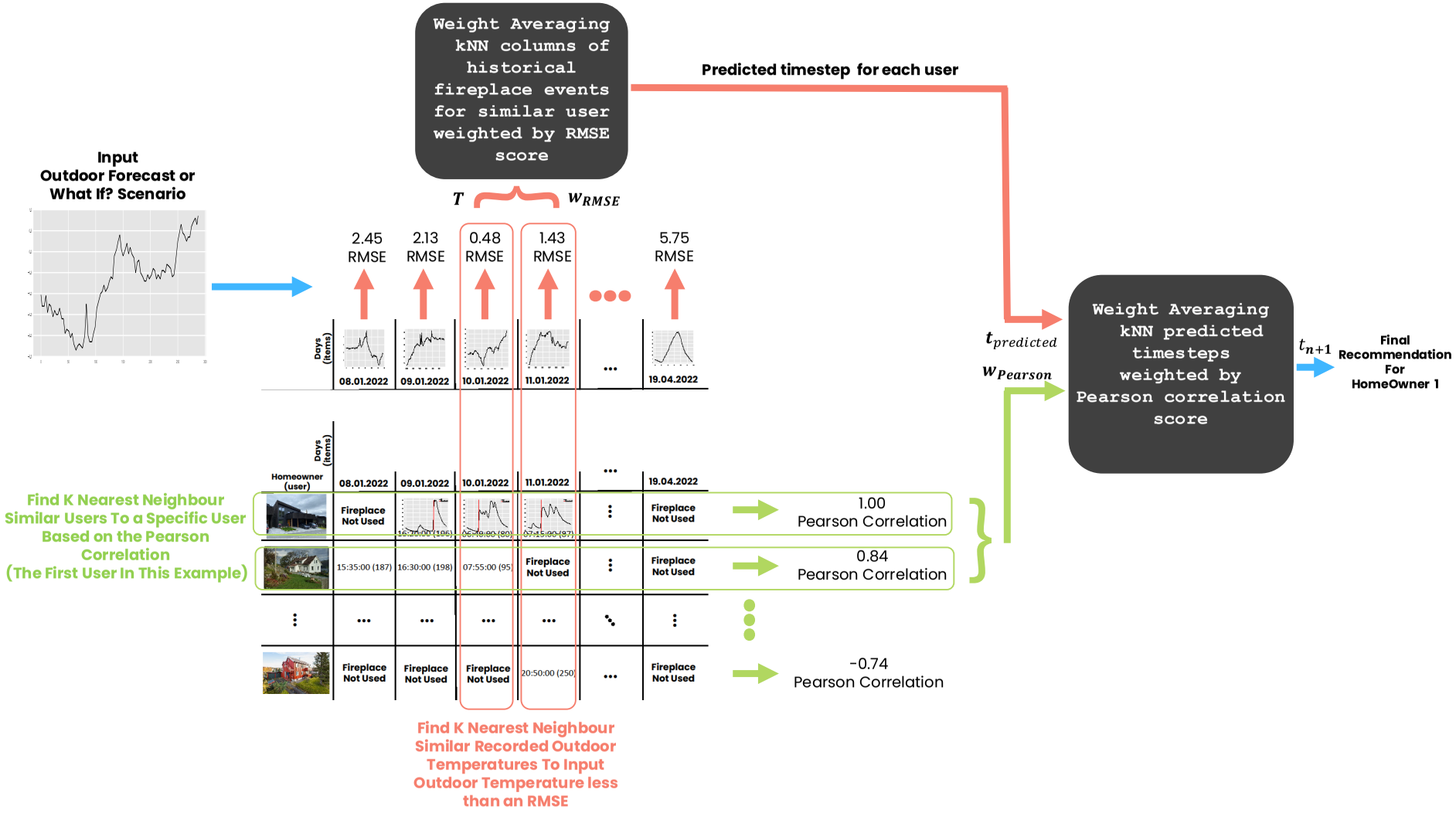


Figure 6.5.: User based collaborative filtering pipeline.

6.2. Demonstration of Prescriptive Digital Twin

In Figure 6.6 we are trying to predict day 23 of the 102 registered days in the dataset, using the seven other existing scenarios where the fireplace is turned on. The RMSE threshold is set to 1.5, and the days with RMSE under 1.5 are then day one, two and 25, which resemble the outside weather of day 23 the most out of the existing samples. There isn't any other user data, such that the final recommendation is only based on the user's weighted average of other similar days, and not the Pearson correlation. Therefore the recommendation uses only the the RMSE part of the UBCF pipeline in Figure 6.5. The recommendation suggests to turn on the temperature at timestep 118 (09:50:00), which is very far of from the ground truth of 189 (15:45:00). This is due to the lack of data, making the method inconclusive at the time of implementation. The inconclusiveness can be clearly seen when analyzing when the fireplace is turned on for day one, two and 25, seen in Figure 6.3. There day one and 23 have very similar fireplace temperature profiles, while two and 25 look more alike. This shows the potential formation of data clustering, which would be well defined with an excess of data available.

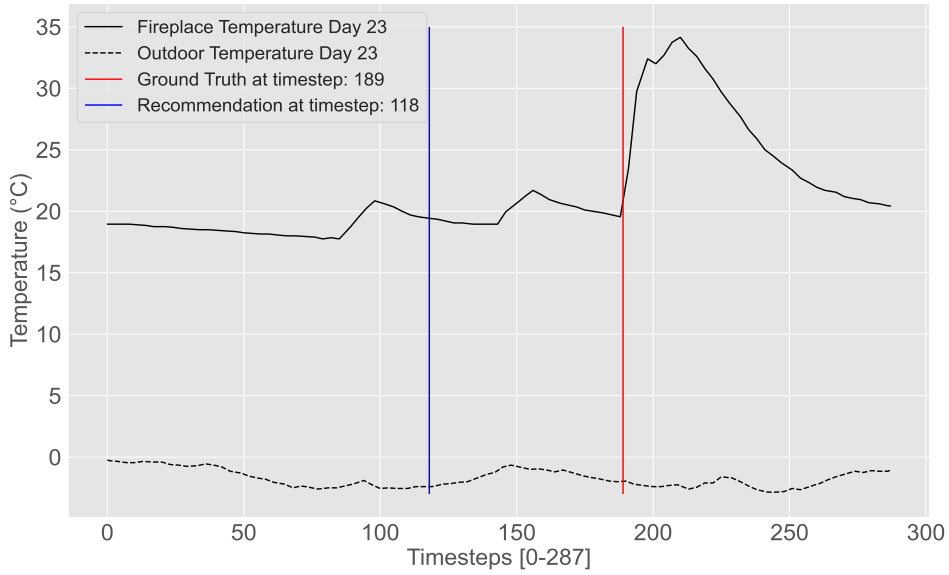


Figure 6.6.: Recommendation to turn on the fireplace. Performance for day 23 of the dataset.

6.2.1. Summary

The UBCF from sensor data is a novel field, and has yet to be explored to its fullest potential. This is mainly due to the lack of existing big data sets, as is the one of the pitfalls of this demonstration as well. Usually UBCF datasets have millions of data about specific user behaviours, which can be used to understand new users.

Since the dataset used for our case is not big enough, as we only have a total of eight acts of turning on the fireplace, it is simply inconclusive. How ever the setup itself is quite promising, and has the potential to support many prescriptive DTs of different homes. Furthermore a recommender system demonstrated in this chapter is also beneficial in supporting the unpredictable parts of the temperature forecasting presented in the predictive DT. In the future, recommendation systems will not only be something used in social media, streaming services and online marketplaces but also in smart homes that have big and rich datasets to support recommendations for various tasks, such as turning on the fireplace, vacuuming the house or making dinner because the homeowner's blood sugar levels are low.

7. Autonomous Digital Twin

5 Now imagine that the homeowner has an accurate and computationally efficient model for predicting the future state of the house and its surroundings under the influence of input changes that the homeowner can affect. These models in combination with advanced control algorithms can be used to get the humans completely out of the loop. The asset can continuously update the digital twin while the latter can control the asset to push it towards a preset optimal operating condition.

The autonomous DT will not be pursued in this project, given that the scope is already ambitious enough, and there are some practical challenges in the demonstration at this level within the timeframe of this work. However, for the sake of completion, it is worth appreciating the complexity of setting up a DT at this capability level. For this reason, this chapter is written more as a concept and not as a demonstration of the capability as has been done in the preceding chapters.

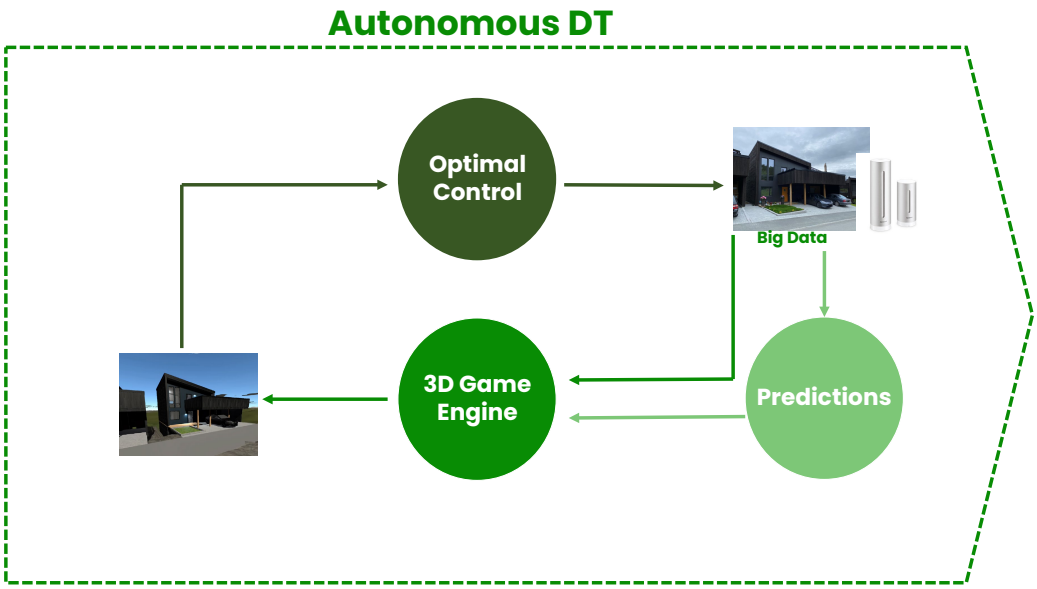


Figure 7.1.: Visualization of the autonomous control loop for the last DT capability level.

Autonomous DT can replace the user by closing the control loop to make decisions and execute control actions on the system autonomously. In Figure 7.1 sensor data and prediction models of the data both flow into the 3D Game Engine, where

the DT is visualized for the observer. The Game Engine then makes automatic decisions to request changes on some specific states based on the monitoring, predictions, and other assessments. It finally makes a controlled change in the real house through API calls to the backend server to the actuators, closing the control loop. Note that the human operator is taken entirely out of the decision-making process, which for some processes has the benefits of reaching the desired output much sooner than with human intervention.

We now give a brief overview of the structure of this chapter. Sections 7.1 and 7.2 are about the conceptual setup and demonstration related to the autonomous DT.

7.1. Cybernetics

A way to satisfy the autonomous DT is to combine data-driven insight such as temperature sensors, and physics-based modeling to create a temperature control system. This would mean modeling the room temperature, and using actuators in a specific floor to reach that desired temperature. It would be necessary to have actuators that makes it possible to both increase and decrease temperature in the the room. The most natural place for this to take place would be the second floor, which already has a fireplace. There is how ever currently no specific way to cool down the room with actuators, which would need air conditioning to be installed.

7.1.1. Linear Control System

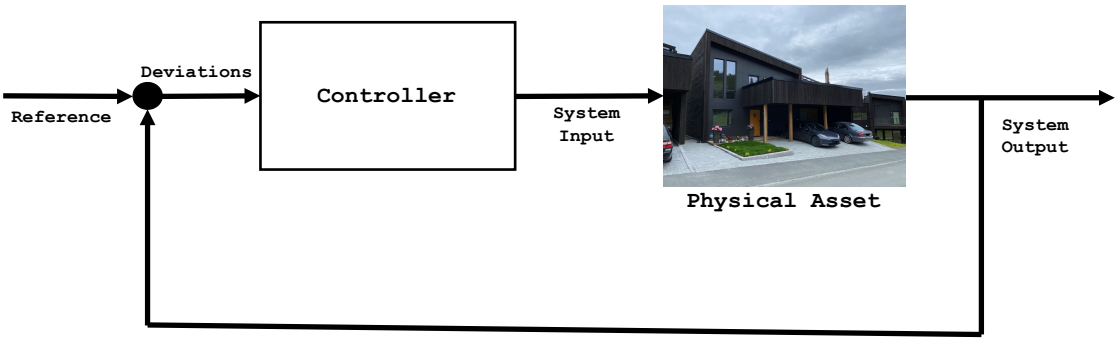


Figure 7.2.: High level view of a control loop.

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{u} \\ \mathbf{y} &= \mathbf{C}\mathbf{x} \end{aligned} \tag{7.1}$$

The accurate representation of room temperature would require the design of a thermodynamics model, which can be represented as a generic state space system as seen in Equation 7.1. A state space model is a way to compress systems of ordinary differential equations into a state vector $\mathbf{x}$, input vector $\mathbf{u}$ and output

vector $\mathbf{y}$ [105]. Where the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ are filled with coefficients that are either constant or time-varying. Then linear control system theory can be applied such as PID, pole placement, model predictive control or other classical control methods to optimize the room temperature [106]. Since the second floor is almost an open space between the living room and the kitchen, one could assume that it is a cube and simplify the modeling process there after. That way the model would also assume that the office door is open. Figure 7.2 displays a high level of the concept. This would be a simple PBM approach, which has its own set of challenges related to modeling assumptions and errors. How ever, since the room is instrumented with temperature sensors on all the walls, we can resort to utilizing big data cybernetics. This can be done by building a model with a hybrid modeling approach, that results in an informed control strategy that can stabilize the room temperature to the desired temperature reference. This method is motivated further in Section 7.1.2.

7.1.2. Big Data Cybernetics

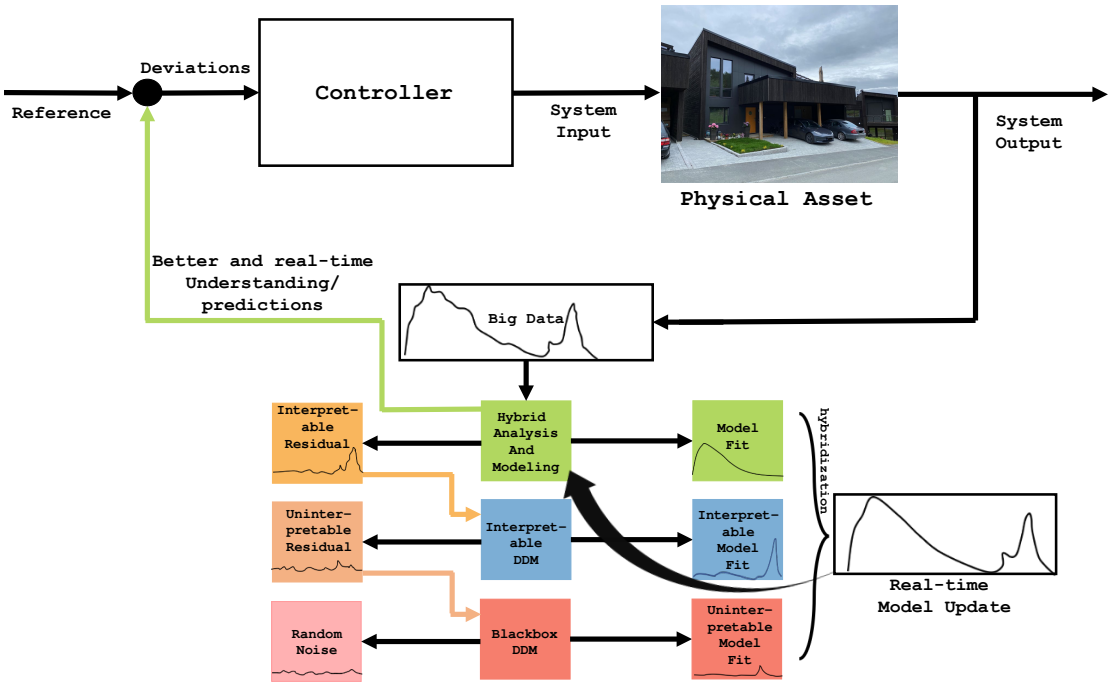


Figure 7.3.: Big Data Cybernetics control loop.

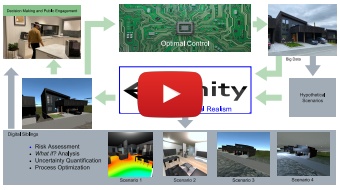
The role of cybernetics in the context of autonomous DTs is to steer the house towards an optimal set point. In order to do so, the output of the system is continuously monitored and compared against a reference. The difference, called the error signal, is applied as feedback to the controller, which generates a system input to bring the output set-point closer to the reference. With the availability of more and more sensors and communication technologies, increasingly larger volumes of data (in fact, big data) are getting available in real-time. Moreover, there is no guarantee that the quantity of interest can be measured directly. The challenge

is then to develop a better understanding of the big data and infer the quantity of interest before it can be used for control purposes. The newly established field of Big Data Cybernetics, an adaptation of the concept first conceived in [107] at the Norwegian University of Science and Technology, offers to address this challenge in a real-time control context. In the first step, the big data is interpreted using well-understood PBM based on known physics. The difference between the observation and the PBM is called the interpretable residual. This, in the second step, is modeled/analyzed using interpretable data-driven modeling approaches. After the second step, uninterpretable residual remains. This is modeled using more complex black box models like Deep Neural Networks. The remaining residual is generally noise which can be discarded. The three steps result in a better understanding of the data and improved models, provided new approaches can be developed to combine existing PBM with interpretable and non-interpretable big data DDM. This approach is a new field of research called Hybrid Analysis and Modeling (HAM). The steps are continuously looped with the availability of new data (i.e., see Figure 7.3), resulting in ever-adapting and improving models compared to the simple linear controller presented in Section 7.1.1.

7.2. Demonstration of Autonomous Digital Twin

Despite the ability to develop control algorithms and the availability of remotely controllable equipment like the balance ventilation system and heat pump, no effort was made to demonstrate this capability. There are several reasons for that, which we briefly mention here. Firstly, this was a short-term project focused on demonstrating the DT and its capability levels concept. Due to lack of time, all the algorithms and modeling tools presented could not be rigorously tested. Furthermore, using black-box neural network-based methods for predictions complicates the matter. Unless the model's working is humanly interpretable, it is not wise to use them in a controlled setting. Doing so could risk the safety of the inhabitants or at the least make the manufacturer's guarantee on the equipment null.

8. Discussion of Results



The results obtained from different levels of digital twins have already been presented in the respective chapters in the demonstration section as well as in the videos (click the thumbnail to the left) and VR demo. In this chapter we discuss those results in greater detail, and also reflect on the weaknesses in our approach that were realized only after having completed the thesis.

8.1. Standalone Digital Twin

The main result for the standalone DT is the final 3D model in Figure 2.5 as well as Figures 2.9 and 2.10. It is important to stress that even the external environment of the asset is part of the standalone, as was done in the current case. At this level, there is no real-time information exchange between the building and the DT. The internal description of the building is not an innovation and is well established within architectural design. Therefore, the only contribution of this work is a streamlined workflow to setup a standalone DT. The proposed workflow ensures that any future changes in the geometric construction of the asset can be quickly affected in the standalone DT. Then the change will propagate to higher levels automatically. Although Unity was utilized in this work due to ease of usage and larger user support, in hindsight, the Unreal Engine could have been a better choice as they are more photo-realistic appearance-wise, and have out-of-the-box support for NVIDIA Omniverse which is tailored towards generating real-time digital twin environments [108]. However, the software have a steep learning curve, compared to Unity. All in all, the impact of the choice of the Game Engine was only limited to the standalone DT.

Additionally, one of the weaknesses of the current workflow is the loss of information from BIM software such as Revit. Usually, Revit can hold information about wall material, cost, and other detailed information that is important during a construction project. However, for our setup, none of these are recorded in the first place. Still, for a professional workflow, it might be essential to migrate over to the Game Engine where the DT asset lives. For Revit or any Autodesk software, this could be achieved by utilizing the Forge API provided by Autodesk to integrate this information with the DT [109].

8.2. Descriptive Digital Twin

Descriptive DT was constructed by establishing a real-time data stream between the asset and the standalone DT. The real-time data, except for the meteorological data (Figure 3.5), came from the installed sensors (Figure 3.3). The real-time data without any processing gave a real-time status of the asset (Figure 3.4). It is clear how having a high-quality standalone DT eased its upgrade to a descriptive DT using RESTful API endpoints provided by the sensor manufacturers. Since the Game Engine supports the C# scripting language, the data was used to dynamically update the DT with the evolution of the asset. Working with data coming from diverse sources, several challenges related to data access was realized. While some data, like the information about the paintings, were static and straightforward to communicate, other sensor data had to be accessed using the APIs provided by the sensor manufacturers. The ease of data acquisition varies across the sensors, with the Philips hue light proving to be the most challenging. The main issue with the Philips Hue lights is that these lights can only be remotely represented if an access token is generated using a local network. Additionally, this token is only valid for seven days. This is mainly a security measure, as the states of these lights can also be manipulated remotely using the access token. This creates an additional hurdle of creating an individual centralized API that securely stores the refresh token and automatically updates it when it expires. This way, all the remote systems that require obtaining the real-time states only have access to the access token, not the refresh token. This is both for security reasons and because if the access token is refreshed with a refresh token, the refresh token becomes invalid as a new refresh token is generated. Hence centralizing the refresh token is necessary to keep track of it and not run the risk of having it being lost.

Another challenge at this level is the visualization of meteorological data. For example the movement clouds in the descriptive DT is based on real wind conditions. However, the formation/distribution of clouds as seen in Figure 3.5 does not bear any similarity to reality. The purpose is just to create a feeling of different kinds of weather. Positioning clouds based on their real clouds counterpart could be a cumbersome task. A solution could be to use real time satellite imagery to construct an approximation of at least the clouds' position, and connect it to the existing cloud renderer. This would position the clouds correctly, but the shapes of them may still be incorrect. Nevertheless all the data at this level are dynamically archived on an external server which makes it possible to not only probe the current state but also the state long time in the past.

8.3. Diagnostic Digital Twin

Diagnostic DT results seen in Figures 4.3,4.6,4.7, and 4.8 are all examples of interactivity that provides user with condition monitoring and troubleshooting. VR allows all sorts of diagnostic information to be relayed to the user, whilst also providing a rather immersive game-like experience. Instead of analyzing individual digits and graphs, one sees the data intuitively as it would exist in the world.

Additionally data that usually cannot be perceived, is funneled through custom made visual effects created in the Game Engine.

In the specialization project only one temperature from the Netatmo weather station was being measured, meaning that it made sense at the time to represent the temperature as a particle system. However the only useful aspect of that was that the color represented a certain temperature. Once multiple temperature sensor measurements were setup, it made a lot more sense to generate a temperature heat map.

The Disruptive Technologies temperature sensors are all approximately placed one meter high from the floor, which justifies the temperature heat map being a 2D plane. If many different temperature sensors were placed at the same length and breadth positions at different heights, then that would allow a more detailed 3D heat map of the rooms. This is more costly, and an alternative is to place the sensors in the corners and middle of the room to optimize the accuracy of a linear interpolation between them, hence generating a 3D heat map that way. Either way a 2D heat map provides more than enough information about the warmest areas in a room, with the added benefit of being the easiest and cheapest to setup.

The current heat map implementation makes certain simplifications that makes the temperature gradients less accurate around walls and doors. The heat map algorithm calculates the temperature distribution radially outwards from the point of measurement and assumes an open space, without walls or doors. A better solution could be put in place including inverse distance weighting, where the temperature gradient is bound to traverse only within the enclosed area, corners and open door spaces [110]. This way the gradient will be even more accurate. As the rooms in the house are quite open spaced the errors are negligible, making the approximating visuals still quite useful in terms of perceiving relative temperatures in the room.

While the heat map algorithm implemented can be improved for temperature, the use-case of it allows for monitoring for any type of spatio-temporal data, such as energy consumption, humidity, registered human interactions, etc. This is quite valuable monitoring capabilities for facility management workers of a commercial building.

8.4. Predictive Digital Twin

Predictive DT results seen in Figures 5.5, 5.6, 5.10 and 5.11 show the performance of forecasting 24 hours of nine future temperatures in the living room. As presented in Chapter 5, a good forecast of the fireplace will propagate to good forecasts of the remaining second floor temperatures. This is demonstrated when predicting the eight sensors using only the known fireplace as input in Figure 5.9, as well as when comparing forecasts from the model to a day when the fireplace gets turned on seen in Figures 5.6 and 5.11. Essentially the main goal of the forecasting model,

is to capture the predictable parts of the room temperature. On the other hand it is not surprising that the model is incapable of adding temperature increases based on fireplace spikes, as that information is certainly not available at the time of forecasting. Furthermore with improvements to the models, one could observe in Figure 6.2 that there is a negative correlation between when it is cold and the fireplace temperature rising. Meaning that based on outside temperature forecasts, the model could be reconstructed in a way that takes in weather forecasts to determine if there is a probability that the fireplace will be turned on 24 hours in the future. Thus with a more complicated model there is a potential of embedding the spikes coming in from the fireplace into the model.

The temperature forecasting model relies on a weighted average ensembling method, described in theory Section 5.4. The motivation behind this is that it provides a more reliable and robust model than relying on an individual model. As each model is built with its own set of biases, if one diverges the other models would be able to pad that forecast as all their contributions are weighted. For example LSTM in specific circumstances outputs poor forecasts [111], which makes relying on a single model in general unsuitable for real-time forecasting in a predictive DT setting. For our case this can be observed in Figure 5.6. Where even though the LightGBM outputs a bad forecast, the remaining models are able to give a weighted forecast that is stable. Note this does not mean that the weighted average will outperform all the other single models in every particular situation, as in this case LSTM performs better, but it will certainly in most forecasting scenarios outperform the single model forecasts. This can be further motivated by looking at the model interpretation of XGBoost and Prophet seen in Figure 5.7. XGBoost weighs the beginning and end of the input more than the rest of the sequence when trying to forecast the next day, on the other hand Prophet has identified a daily trend that it assumes exists for all future forecasts. Combining these together provides the advantages of both models, while averaging out their imperfections.

Another reason to use an ensemble of simple but generally uncorrelated models, is to create a more interpretable, maintainable and robust model. The pipeline approach from Figure 5.4 and 5.8 tries to focus mainly on a data-centric rather than a model-centric approach, where the idea is first presented by Andrew Ng [112]. Rather than spending a lot of time constructing the perfect LSTM deep learning architecture, customized to forecast each one and specific second floor temperature sensor. Instead the objective is to create a forecasting pipeline that uses a lot of simple, and in some cases unoptimized models, to generate a streamlined forecast for all the sensors. Where the improvements of prediction accuracy comes from feature engineering and data analysis. The main challenge of the current pipeline is that there is only a few months of data from the winter season, which makes it hard for the model to learn the patterns of spring, summer and fall. But some challenges can already be addressed, for example by cleverly indicating to the model if the fireplace is likely to be turned on the next day. Other additional insight might appear in a few years, in the form of yearly seasonality which also could provide a layer of precision.

8.4.1. Temperature Predictive Model

To emphasize the discussion on a data-centric approach, an incremental cross validation RMSE improvements by building on an a base XGBoost model can be seen in Table 8.1. Note that these results are similar to prediction results from Figure 5.9, as they are based on known fireplace input not a forecasted fireplace input.

Table 8.1.: Summary of incremetal RMSE improvements of predictions of second floor indoor temperature regression model, based on a $k = 5$ kfold cross validation score.

Addition	**RMSE**
Base XGBoost	0.98
Cyclical Features	0.81
Transformative Features	0.78
Lagged Features	0.65
Historical Weather Temperature	0.64
Domain Specific Feature Engineering	0.63
Target Transformation	0.61
Stacked Model Ensemble	0.54
Weight Averaging Ensemble	0.42
Hyperparameter Optimization	0.36

From the Figure 5.9, one can see that the performance is not very different for most of the sensors, except for 2OfficeDesk and 2Cooking. Where the 2OfficeDesk sensor is placed, the office can be opened or closed with a door and also has a computer and window on the side of where the sun shines the most. Both the computer and the sun rays are unpredictable factors that contribute to a higher temperature in that room. 2Cooking temperature sensor is placed over the stove top in the kitchen, which also is a source of temperature increases. These are also second floor sensors that are the least exposed to temperature from the fireplace as seen in Figure 3.2. Therefore a higher error is to be expected on these sensors due to their unpredictable nature, how ever with a bigger dataset these spikes coming from cooking and other activities could be dealt with through effective feature engineering.

8.4.2. Temperature Forecasting Model

The forecasting pipeline in Figure 5.4 is a double-edged sword, because the performance in the eight other sensors is bottlenecked by the fireplace temperature forecast. Currently how the pipeline is setup, it has a lot of potential to be further improved by the addition of other features that are not used in the forecasting pipeline. Additionally as seeing that XGBoost do not use the entire sequence of input data to create its forecasts, the usage of principal component analysis (PCA) or other dimensionality reduction methods could be worth investigating for transforming the data in meaningful ways to the model.

8.4.3. Sun Position Prediction Model

The sun position algorithm as demonstrated in Figures 5.13, 5.14, and 5.15 works splendidly within the Game Engine environment. Since the lighting simulation is delegated to the Unity Engine itself, it is worth debating as to how realistic it is representing light. The lighting physics of the Game Engine might work as intended, but it is unclear to what degree it interprets the material textures correctly in terms of light absorption and reflection. These physics are in general very expensive to render in real-time for an Oculus Quest VR device, and so the quality of the lighting physics depends on which rendering pipeline is used i.e. standard, URP or HDRP (recall Section 2.3.1), where HDRP generates the most realistic lighting physics. While this is not directly related to how the sun position algorithm intrinsically works, the way Unity renders the sun is worth being aware of.

8.5. Prescriptive Digital Twin

Due to the insufficient data about user behaviour, the results relating to the UBCF is inconclusive at the time of setup as discussed in regards to Figure 6.6. This is to be expected as the algorithm has a cold start [104], and the quality of the results improve as both more data and users are acquired. In terms of contribution, the proposed UBCF method for detecting fireplace lighting behavior, seen in Figure 6.4 and 6.5, are interesting findings. Recommender systems are still uncommon for sensor-based data, where many of the challenges comes from quantifying user behavior from these sources [113]. Being able to quantify user behavior based on for instance fireplace activity, can provide useful insight that will allow quantification of the unpredictable parts in the predictive DT forecasts as seen in Figure 5.6. Therefore there is a great motivation to keep gathering more data, and revisiting these methods in the future to evaluate the performance of the recommender system.

8.6. Autonomous Digital Twin

Results from the autonomous DT were never achieved, as it has only been set up on a conceptual level. Therefore for the sake of continuity, this section will follow the discussion on the same conceptual basis. The homeowner has access to a heat pump, adjustable lights, and control of electricity flowing in the outlets. These allow controlling temperature, luminosity, and energy optimization. This could be a valuable and relevant contribution in terms of energy optimization and how electricity prices have been getting increasingly more expensive. Although it is hard to say which aspects one should include in such a control loop, as the system should not be invasive on the humans using the asset. Instead, one can speculate that such a system needs to intelligently cut off energy usage not currently used by humans and prioritize what is directly necessary. For instance, automatically turn off all the lights at night to prioritize charging the car. This would require an advanced control system using the concept of big data cybernetics.

9. Conclusion and Future Work

We have already seen that a DT is created from a patchwork of various enabling technologies, as seen illustrated in Figure 9.1. In this chapter we will conclude the work described in this thesis and then present a detail of future research directions to be taken to lift the presented DT framework to an altogether different level.

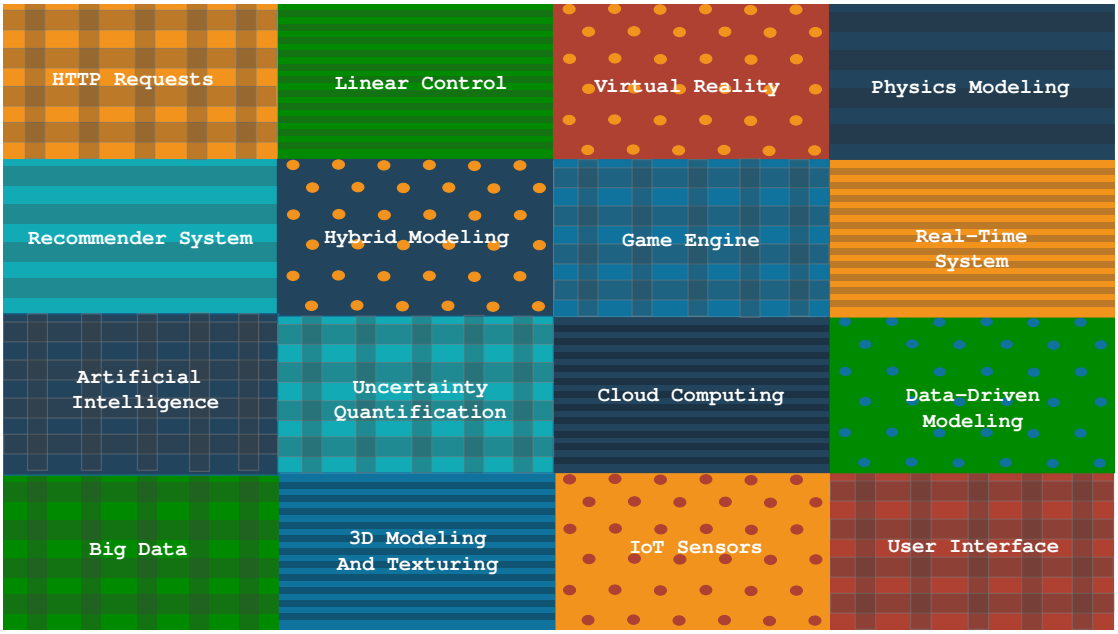


Figure 9.1.: A DT is a patchwork of enabling technologies working together to create a high capability virtual representation.

9.1. Conclusion

In this work, we exploited the power of artificial intelligence, advanced sensor technologies, and virtual reality to develop a fully functional and highly capable DT of a modern house. The work involved creating a realistic 3D model of the house that is not only good for visualization but also for conducting engineering simulations. This corresponded to a standalone DT. The physical house was then equipped with a diverse class of sensors, and the corresponding digital representation of the house was updated accordingly. A real-time data acquisition pipeline was established to update the state of the DT with any changes in the state of the physical house, resulting in a descriptive DT. Analytics tools were applied to the incoming data to detect critical changes, resulting in a diagnostic DT. These three levels of DT were not capable of giving any information about the future state as they all

relied on the incoming data. At the next level, i.e., the predictive DT, an ensemble of pure data-driven time series forecasting models was built and trained. A novel approach to fuse the results from the trained model was implemented to predict the future state of the house accurately. In addition, a physics-based modeling approach to predict the sun's movement and its obstruction by the local terrain was also implemented to predict the solar potential for any time in the future. At the prescriptive DT level, with the insufficient data available, it was shown how collaborative filtering could be used to benefit from the data available from the other similar houses in the neighborhood. It was also demonstrated how a prescriptive DT could learn about the user's behavior to support recommendations, by doing so also eliminating what is perceived as unpredictable to the forecasting model in the predictive DT. Finally, the autonomous DT weaves all the subsystems together by closing the control loop, taking all the insight from each capability level, and executing it without any human intervention. By doing so all the objectives set in the beginning of this work is realized as follows:

- The concept of DT, and its capability level as defined in [1], and [3] was successfully demonstrated using the use case of a modern house. Moreover, the framework was designed in a way that is fully modular and hence extendable. To the best of my knowledge this is first framework of its kind.
- The developed DT can be run on a standalone virtual reality headset and hence can be used to communicate the real potential of the technology to potential end-users. The choice of a modern house as a use case also helps in letting the users better relate to the technology.
- During the implementation, it was realized that data-related issues like data privacy and security, models that can exploit prior knowledge and combine it with artificial intelligence, and industrial acceptance are the key to the success of this technology. In the current project, we did not address the first issue but demonstrated that if those issues are adequately handled, then it is indeed possible to develop highly capable DTs.

Having highlighted some of the achievements of this work, it is important to highlight some of the areas of improvement in the following section.

9.2. Future Work

The modular nature of the DT framework developed in this work offers the opportunity to extend and improve individual capabilities without rendering the other dysfunctional. We enumerate them as follows:

- *Standalone digital twin:* At this level, a considerable amount of time was spent creating a 3D model of the house and the objects, like pieces of furniture manually. This can be a bottleneck in scaling the DT technologies to a small neighborhood with many houses. Fortunately, image-based photogrammetry has already solved this issue and can be implemented within

the framework discussed in the current work. Furthermore, the solid models are generally represented by textured tessellated polygon surfaces whose numbers should be reduced without an observable degradation in quality to make the DT run on low-end affordable computing devices like those in Oculus Quest 2.

- *Descriptive digital twin:* Geometric change detection [114] could be implemented to keep track of the geometric changes inside the house in a descriptive DT using just an RGB camera, limited communication bandwidth and storage. Likewise, the external environment like the cloud cover could be more accurately described using the real-time satellite data available from the meteorological site.

- *Diagnostic digital twin:* Intrusion detection based on dynamic mode decomposition and compressed sensing. Principal component analysis or autoencoder for detecting deviations from the norm in the heterogeneous multivariate data can also be useful for detecting anomalies. The other sensor data like temperature, humidity, noise, and air quality are measured on only a few discrete locations. At the moment a very crude way of interpolation scheme has been implemented to create visually pleasing heatmaps. The method can be improved by sensitizing the inverse distance weighting [110] with door states and wall corner locations. It would also be worth evaluating the potential of advanced techniques like compressed sensing for creating such high resolution maps.

- *Predictive digital twin:* In the current project, either a purely physics-based or data-driven model has been used to predict the external and internal state of the house, respectively. However, both these approaches have inherent weaknesses as discussed in [3] Recently in [115] the authors have shown how a hybrid modeling approach can be used to address these weaknesses and make accurate and more certain predictions. The approach is ideal for modeling physics that is not completely known or when the parameters required to run the model are not known accurately, the kind of problem which will be even more pronounced for existing old buildings.

- *Prescriptive digital twin:* The recommender system created for Chapter 6 is a simple UBCF. One way this could be made more advanced is using matrix factorization machines, which is simply another method for creating recommendations based on user and item features. The matrix factorization approach also struggles from a cold start, and has therefore been left unexplored for this thesis. Another method that could be investigated is a restricted Boltzmann machine for collaborative filtering. Once extensive fireplace activity is generated, and the user can be understood better, it would be interesting to compare a simple UBCF against these approaches. However, the main priority for the prescriptive DT, is collecting fireplace activity data from multiple users over a span of a winter season. That has the potential to support conclusive results of these methods, and should be done first before setting of to explore the recommender system methodologies.

Another recommender system that can be implemented using the existing sensor data, is a Bayesian ranking system. Suppose one wants to rank room visitations based on door proximity sensor data, then the rooms that most likely are going to be visited at a certain point in time could be given as recommendations to the user. The advantage of this method compared to UBCF, is that it does not struggle with a cold start and will adapt ranking according to the existing data. Furthermore there is also a lot more behavior that can be derived from the existing proximity sensors, that is currently unavailable with the fireplace data. There is no immediate value to tell the user which rooms are the most popular in a residential house. How ever, for big commercial buildings being able to track this information could allow for better understanding of how the building is utilized during the lifecycle of the prescriptive DT.

- *Autonomous digital twin:* Due to a fear of voiding the guarantee on equipment like the balance ventilation system, the potential of a fully autonomous DT could not be practically demonstrated. The smart lights could be controlled remotely, but since no data was ever recorded for the kind of research undertaken in this work, a control strategy could not be implemented. It would be interesting to record data regarding the lighting taste of the occupants and develop a controller to satisfy the taste. Alternatively, research on the psychological effects of lighting on the occupants can be integrated in the autonomous DT. Another challenge that is worth addressing before realizing a fully autonomous DT is to make the models on which decisions are made humanly interpretable.

The DT concept is fast evolving and the achievement of this project, as well as the future research directions proposed, are only minuscule in the context of the bigger picture. However, the project has resulted in an extendable DT framework that can be used for pedagogical purposes and for testing new methods capable of making the DT indistinguishable from its physical counterpart.

Bibliography

[1] Adil Rasheed, Omer San, and Trond Kvamsdal. Digital Twin: Values, Challenges and Enablers from a modeling perspective. *IEEE Access*, 8:21980–22012, 2020.

[2] Elias Mohammed Elfarri. Digital twin of smart housing: An initial setup of a digital twin using the capability levels framework. *TTK4550 Specialization Project*, 2021.

[3] Omer San, Adil Rasheed, and Trond Kvamsdal. Hybrid analysis and modeling, eclecticism, and multifidelitycomputing toward digital twin revolution. *GAMM Mitteilungen*, page e2021000071, 2020.

[4] DNV GL. Practice dnv-rp-a204: Qualification and assurance of digital twins. 01 2020.

[5] Regjeringen. Klimaendringer og norsk klimapolitikk. https://www.regjeringen.no/no/tema/klima-og-miljo/innsiktsartikler-klima-miljo/klimaendringer-og-norsk-klimapolitikk/id2636812/, 2021. [Online; accessed 06.03.2022].

[6] Byggalliansen. Klimakur for bygg og eiendom. https://byggalliansen.no/kunnskapssenter/publikasjoner/infopakkeklimakjempen/#1610543721239-1fec4ebb-a64b, 2020. [Online; accessed 06.03.2022].

[7] Erik Lindsay Griffin and Thomas Oscar-Andersen. Gevinstrealisering med BIM – barrierer og muligheter. https://www.standard.no/nyheter/nyhetsarkiv/bygg-anlegg-og-eiendom/2020/-bim-rapport-om-barrierer-og-muligheter-lansert/, 2020. [Online; accessed 19.02.2022].

[8] Manish K Dixit, Varusha Venkatraj, Mohammadreza Ostadalimakhmalbaf, Fatemeh Pariafsai, and Sarel Lavy. Integration of facility management and building information modeling (bim). *Facilities.*, 37(7/8):455–483, 2019.

[9] Siavash H. Khajavi, Naser Hossein Motlagh, Alireza Jaribion, Liss C. Werner, and Jan Holmström. Digital twin: Vision, benefits, boundaries, and creation for buildings. *IEEE Access*, 7:147406–147419, 2019.

[10] Muhammad Shahzad, Muhammad Tariq Shafiq, Dean Douglas, and Mohamad Kassem. Digital twins in built environments: An investigation of the characteristics, applications, and challenges. *Buildings*, 12(2), 2022.

[11] Teknisk Ukeblad. Hver krik og krok av Operaen bimmes med robot-skanner på hjul. https://www.tu.no/artikler/hver-krik-og-krok-av-operaen-bimmes-med-robot-skanner-pa-hjul-br/441612, 2018. [Online; accessed 02.03.2022].

[12] Guodong Shao and Moneer Helu. Framework for a digital twin in manufacturing: Scope and requirements. *Manufacturing Letters*, 24:105–107, 2020.

[13] Azad M. Madni, Carla C. Madni, and Scott D. Lucero. Leveraging digital twin technology in model-based systems engineering. *Systems*, 7(1), 2019.

[14] Sarah Jones. What Is a Digital Twin? How Intelligent Data Models Can Shape the Built World. https://redshift.autodesk.com/what-is-a-digital-twin/, 2021. [Online; accessed 14.03.2022].

[15] The Institute of Engineering and Technology. Digital Twins for the built environment. https://www.theiet.org/impact-society/sectors/built-environment/built-environment-news/2019-news/digital-twins-for-the-built-environment/, 2019. [Online; accessed 21.03.2022].

[16] Aidan Fuller, Zhong Fan, Charles Day, and Chris Barlow. Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8:108952–108971, 2020.

[17] Werner Kritzinger, Matthias Karner, Georg Traar, Jan Henjes, and Wilfried Sihn. Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11):1016–1022, 2018. 16th IFAC Symposium on Information Control Problems in Manufacturing INCOM 2018.

[18] Michael Batty. Digital twins. *Environment and Planning B: Urban Analytics and City Science*, 45(5):817–820, 2018.

[19] Dean Douglas, Graham Kelly, and Mohamad Kassem. Bim, digital twin and cyber-physical systems: crossing and blurring boundaries. *CoRR*, abs/2106.11030, 2021.

[20] Matterport. Matterport Website. https://investors.matterport.com/, 2022. [Online; accessed 07.03.2022].

[21] Openspace. Openspace. https://www.openspace.ai/, 2021. [Online; accessed 30.11.2021].

[22] Ocean Engineering. CONVRS - Showcase your company and products anytime and anywhere without worrying about space, freight cost, travelling and waste of precious time. https://oceanvisioneering.com/, 2022. [Online; accessed 09.04.2022].

[23] OnSiteViewer. OnSiteViewer Experience Realism. https://www.onsiteviewer.com/, 2022. [Online; accessed accessed 14.04.2022].

[24] Autodesk. Multidisciplinary BIM software for higher-quality, coordinated designs. https://www.autodesk.eu/products/revit/overview, 2022. [Online; accessed 22.03.2022].

[25] TwinView. TwinView Website. https://www.twinview.com/, 2018. [Online; accessed 25.01.2022].

[26] Invicara. Invicara Website. https://invicara.com/about, 2022. [Online; accessed 23.03.2022].

[27] NorthWind. NorthWind. https://www.northwindresearch.no/, 2022. [Online; accessed 23.03.2022].

[28] Florian Stadtmann, Adil Rasheed, Kjetil Andre Johannessen, Omer San, Trond Kvamsdal, John Olav Tande, and Industry Partners. Digital twin and asset management for wind energy. *To be submitted*, 2022.

[29] Michael Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. Springer International Publishing, Cham, 2017.

[30] Sahay Rina and Devadas Viveka. Unity'ed We Stand with Revit and BIM 360. https://www.autodesk.com/autodesk-university/class/Unityed-We-Stand-Revit-and-BIM-360-2019, 2021. [Online; accessed 04.11.2021].

[31] Unity Technologies. Working with the FBX Exporter Setup and Roundtrip - 2019.1, 2019.2. https://learn.unity.com/tutorial/working-with-the-fbx-exporter-setup-and-roundtrip-2019-1, 2021. [Online; accessed 17.11.2021].

[32] Sung-eui Yoon, Enrico Gobbetti, David Kasik, and Dinesh Manocha. 2008.

[33] Dongxiao Zhou. Texture analysis and synthesis using a generic markov-gibbs image model. 01 2006.

[34] R.M. Haralick. Statistical and structural approaches to texture. *Proceedings of the IEEE*, 67(5):786–804, 1979.

[35] Tianbo An, Peng Lian, Jian Zhao, Qing Liu, Yuze Sun, and Weiqi Wang. Application of 3d rendering in twinmotion. In *2020 International Conference on Virtual Reality and Visualization (ICVRV)*, pages 280–281, 2020.

[36] Architextures. Free Textures For Educational Purposes. https://architextures.org/, 2021. [Online; accessed 22.11.2021].

[37] sketchupTextures. Textures. https://www.sketchuptextureclub.com/, 2021. [Online; accessed 22.11.2021].

[38] cgTextures. Textures. https://www.textures.com/, 2021. [Online; accessed 22.11.2021].

[39] John Manfredy. Where to find good materials for Revit... https://www.revitforid.com/the-revit-resource/where-to-find-good-materials-for-revit, 2021. [Online; accessed 22.11.2021].

[40] The investigation on using unity3d game engine in urban design study. *Journal of ICT research and applications.*, 3(1):1, 2009.

[41] Unity Documentation. Render pipelines introduction. https://docs.unity3d.com/Manual/render-pipelines-overview.html, 2021. [Online; accessed 22.11.2021].

[42] Polantis. Free CAD and BIM Objects. https://www.polantis.com/ikea, 2021. [Online; accessed 23.11.2021].

[43] Kartverket. Height Map Data - Kartverket. https://hoydedata.no/LaserInnsyn/, 2021. [Online; accessed 21.October.2021].

[44] TerrainParty. Height Map Data - TerrainParty. http://terrain.party/, 2021. [Online; accessed 21.10.2021].

[45] skydark. Height Map Data - Skydark. https://heightmap.skydark.pl/, 2021. [Online; accessed 21.10.2021].

[46] Wilfrid Donkers and Kathy Feucht. Deloitte real estate predictions 2021. 01 2021.

[47] Pippa Boothman. Legacy Meets Technology: Saving Hundreds of Maintenance Hours at The Royal Opera House. https://www.disruptive-technologies.com/blog/preserving-the-royal-opera-house-with-sensor-technology, 2021. [Online; accessed 27.02.2022].

[48] Case Studies. Having successfully built a strong working relationship with Playfords Ltd, Rolton Group were appointed to provide mechanical and electrical engineering services at the Royal Opera House in Covent Garden, London. https://www.rolton.com/case-studies/browse/royal-opera-house, 2021. [Online; accessed 17.03.2022].

[49] Nordic BIM Group. TWINVIEW. https://www.nordicbim.com/no/loesninger/twinview, 2022. [Online; accessed 09.03.2022].

[50] David Herron. *Node Web Development: Second Edition*. Packt Publishing, 2013.

[51] Disruptive Technologies. Disruptive Technologies Python Client. https://github.com/disruptive-technologies/python-client, 2022. [Online; accessed 01.03.2022].

[52] Philippelt. Netatmo Python Client. https://github.com/philippelt/netatmo-api-python, 2022. [Online; accessed 01.03.2022].

[53] quentinsf. Philips Hue Python Client. https://github.com/quentinsf/qhue, 2022. [Online; accessed 01.03.2022].

[54] Netatmo. Smart Weather Station. https://www.netatmo.com/en-us/weather/weatherstation, 2021. [Online; accessed 16.11.2021].

[55] Netatmo. Netatmo Documentation - Authentication. https://dev.netatmo.com/apidocumentation/oauth, 2021. [Online; accessed 16.03.2022].

[56] Philips. Philips Hue - Authentication. https://developers.meethue.com/develop/get-started-2/, 2021. [Online; accessed 01.02.2022].

[57] Mohamed Ashiq Faleel. Control your Philips Hue Lights from Microsoft Power Platform and .NET. https://ashiqf.com/tag/philips-hue-remote-api/, 2021. [Online; accessed 02.02.2022].

[58] Disruptive Technologies. A guide on how to implement an OAuth2 flow for authenticating our REST API. https://developer.disruptive-technologies.com/docs/authentication/oauth2, 2021. [Online; accessed 23.01.2022].

[59] Disruptive Technologies. REST API Reference. https://developer.disruptive-technologies.com/api, 2021. [Online; accessed 25.01.2022].

[60] Metrologisk Institutt. Nowcast 2.0 . https://api.met.no/weatherapi/nowcast/2.0/documentation, 2022. [Online; accessed 22.03.2022].

[61] Metrologisk Institutt. Possible Weather Conditions From API . https://api.met.no/weatherapi/weathericon/2.0/documentation, 2022. [Online; accessed 22.03.2022].

[62] Xinghua Gao and Pardis Pishdad-Bozorgi. Bim-enabled facilities operation and maintenance: A review. *Advanced Engineering Informatics*, 39:227–247, 2019.

[63] Alf Inge Wang. IT3021 - Game+. https://www.ntnu.edu/studies/courses/IT3021#tab=omEmnet, 2021. [Online; accessed 20.08.2021].

[64] Tarja Susi, Mikael Johannesson, and Per Backlund. Serious games : An overview. 2007.

[65] Sebastian Deterding, Dan Dixon, Rilla Khaled, and Lennart Nacke. From game design elements to gamefulness: Defining "gamification". In *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments*, MindTrek '11, page 9–15, New York, NY, USA, 2011. Association for Computing Machinery.

[66] Yu kai Chou. The Octalysis Framework for Gamification & Behavioral Design. https://medium.com/@yukaichou/the-octalysis-framework-for-gamification-behavioral-design-fe381150f0c1, 2021. [Online; accessed 11.11.2021].

[67] Erik Fossum Færevaag. Why Heatmaps Are A Powerful Tool for Facilities Management . https://www.disruptive-technologies.com/blog/why-heatmaps-are-a-powerful-tool-for-facilities-management, 2020. [Online; accessed 25.03.2022].

[68] Unity. Unity Documentation - Manual: Shaders introduction . https://docs.unity3d.com/Manual/shader-introduction.html, 2022. [Online; accessed 02.03.2022].

[69] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, DMKD '03, page 2–11, New York, NY, USA, 2003. Association for Computing Machinery.

[70] Matthias Döring. Prediction vs Forecasting. https://www.datascienceblog.net/post/machine-learning/forecasting_vs_prediction/, 2022. [Online; accessed 05.04.2022].

[71] Burton. G. Malkiel. *A Random Walk Down Wall Street*. Norton, New York, 1973.

[72] K.I. Park. *Fundamentals of Probability and Stochastic Processes with Applications to Communications*. Springer International Publishing, 2017.

[73] Prashant Banerjee. Complete Guide on Time Series Analysis in Python. https://www.kaggle.com/code/prashant111/complete-guide-on-time-series-analysis-in-python/notebook, 2020. [Online; accessed 12.04.2022].

[74] Sharoon Saxena. What's the Difference Between RMSE and RMSLE? https://medium.com/analytics-vidhya/root-mean-square-log-error-rmse-vs-rmlse-935c6cc1802a, 2019. [Online; accessed 06.04.2022].

[75] Xichu Zhang. Deep understanding of the ARIMA model. https://towardsdatascience.com/deep-understanding-of-the-arima-model-d3f0751fc709, 2021. [Online; accessed 07.04.2022].

[76] Facebook. Prophet — Forecasting at scale. https://facebook.github.io/prophet/, 2017. [Online; accessed 07.04.2022].

[77] Sean J Taylor and Benjamin Letham. Forecasting at scale. preprint, PeerJ Preprints, September 2017.

[78] Arindam. What is the Prophet model? https://www.kaggle.com/code/arindamgot/eda-prophet-mlp-neural-network-forecasting, 2022. [Online; accessed 08.04.2022].

[79] Christopher Olah. Understanding LSTM Networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/, 2015. [Online; accessed 12.04.2022].

[80] Jeffrey L. Elman. Finding structure in time. *Cogn. Sci.*, 14:179–211, 1990.

[81] Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015. [Online; accessed 13.04.2022].

[82] Alex Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385. 01 2012.

[83] Hendrik Jacob van Veen, Le Nguyen The Dat, and Armando Segnini. Kaggle Ensembling Guide. https://web.archive.org/web/20181103114010/https://mlwave.com/kaggle-ensembling-guide/, 2015. [Online; accessed 13.04.2022].

[84] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, June 1990.

[85] Tianqi Chen and Carlos Guestrin. XGBoost. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, aug 2016.

[86] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *NIPS*, 2017.

[87] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features, 2017.

[88] Xgboost. Introduction to Boosted Trees. https://xgboost.readthedocs.io/en/latest/tutorials/model.html, 2021. [Online; accessed 17.11.2021].

[89] Casper Hansen. Stack machine learning models: Get better results. https://developer.ibm.com/articles/stack-machine-learning-models-get-better-results/, 2022. [Online; accessed accessed 13.04.2022].

[90] ANISOTROPIC. Introduction to Ensembling/Stacking in Python. https://www.kaggle.com/code/arthurtok/introduction-to-ensembling-stacking-in-python/notebook, 2022. [Online; accessed accessed 13.04.2022].

[91] Bertrand Clarke. Comparing bayes model averaging and stacking when model approximation error cannot be ignored. *J. Mach. Learn. Res.*, 4(null):683–712, dec 2003.

[92] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[93] NVIDIA. CUDA Toolkit Archive. https://developer.nvidia.com/cuda-toolkit-archive, 2022. [Online; accessed accessed 18.04.2022].

[94] NVIDIA. NVIDIA cuDNN. https://developer.nvidia.com/cudnn, 2022. [Online; accessed accessed 18.04.2022].

[95] Norsk Klimaservicesenter. Observasjoner og værstatistikk. https://seklima.met.no/observations/, 2022. [Online; accessed accessed 12.04.2022].

[96] paroscientific. MET4 and MET4A calculation of dew point. https://archive.ph/20120526034637/http://www.paroscientific.com/dewpoint.htm, 2022. [Online; accessed accessed 25.03.2022].

[97] A. W. T. Barenbrug. *Psychrometry and psychrometric charts*. Chamber of Mines of South Africa, Johannesburg, 3d ed edition, 1974.

[98] Bjørn H Granslo. Sun Position in C#. http://guideving.blogspot.com/2010/08/sun-position-in-c.html/, 2021. [Online; accessed 17.10.2021].

[99] S. Dunlop, O. Montenbruck, R.M. West, and T. Pfleger. *Astronomy on the Personal Computer*. Springer Berlin Heidelberg, 2013.

[100] NOAA. National Oceanic and Atmospheric Administration(NOAA) Sun Position Calculator. http://www.geoastro.de/astro/suncalc/index.htm, 2021. [Online; accessed 29.11.2021].

[101] sunCalc. Sun Position Calculations. https://https://www.suncalc.org/, 2021. [Online; accessed 29.11.2021].

[102] Taiping Zhang, Paul W. Stackhouse, Bradley Macpherson, and J. Colleen Mikovitz. A solar azimuth formula that renders circumstantial treatment unnecessary without compromising mathematical rigor: Mathematical setup, application and extension of a formula based on the subsolar point and atan2 function. *Renewable Energy*, 172:1333–1340, 2021.

[103] Harald Steck, Linas Baltrunas, Ehtsham Elahi, Dawen Liang, Yves Raimond, and Justin Basilico. Deep learning for recommender systems: A netflix case study. *AI Magazine*, 42(3):7–18, Nov. 2021.

[104] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Jesús Bernal. A collaborative filtering approach to mitigate the new user cold start problem. *Knowledge-Based Systems*, 26:225–238, 2012.

[105] Chi-Tsong Chen. *Linear System Theory and Design*. Oxford University Press, Inc., USA, 3rd edition, 1998.

[106] Jozef Kurilla. Temperature control of multidimensional system using decoupled mpc controllers. In *2017 21st International Conference on Process Control (PC)*, pages 351–357, 2017.

[107] Harald Martens. Big Data Cybernetics. https://folk.ntnu.no/martens/?BigDataCybernetics, 2015. [Online; accessed 08.06.2022].

[108] NVIDIA. Develop with NVIDIA Omniverse. https://developer.nvidia.com/nvidia-omniverse-platform, 2022. [Online; accessed 05.04.2022].

[109] Autodesk. Forge - A cloud-based developer platform from Autodesk. https://forge.autodesk.com/, 2022. [Online; accessed 19.04.2022].

[110] Disruptive Technologies. Generating a Room Temperature Heatmap . https://developer.disruptive-technologies.com/docs/other/application-notes/generating-a-room-temperature-heatmap, 2020. [Online; accessed 25.03.2022].

[111] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. Statistical and machine learning forecasting methods: Concerns and ways forward. *PLOS ONE*, 13(3):1–26, 03 2018.

[112] Andrew Ng. AI Doesn't Have to Be Too Complicated or Expensive for Your Business. https://hbr.org/2021/07/ai-doesnt-have-to-be-too-complicated-or-expensive-for-your-business, 2021. [Online; accessed 22.04.2022].

[113] May Altulyan, Lina Yao, Xianzhi Wang, Chaoran Huang, Salil S Kanhere, and Quan Z Sheng. Recommender systems for the internet of things: A survey, 2020.

[114] Tiril Sundby, Julia Maria Graham, Adil Rasheed, Mandar Tabib, and Omer San. Geometric change detection in digital twins. *Digital*, 1(2):111–129, 2021.

[115] Sindre Stenen Blakseth, Adil Rasheed, Trond Kvamsdal, and Omer San. Deep neural network enabled corrective source term approach to hybrid analysis and modeling. *Neural Networks*, 146:181–199, 2022.

A. Sun Position Prediction Model

Algorithm 1: Simplified algorithm for finding position of celestial bodies used to simulate external light for the Digital Twin.

Input: Y/M/D, B, L
Output: ϕ, θ
$H \leftarrow TimeNow.Hour$
$JD \leftarrow 367Y - \frac{7}{4}(Y + \frac{9}{12}M) + \frac{275}{9}M + D - 730531.5$
$JC \leftarrow \frac{JD}{36525.0}$
$t_{LSR} \leftarrow (6.6974 + 2400.0513JC + \frac{366.2422}{365.2422}H) \cdot 15 + L$
$M_0 \leftarrow c_1 + c_2 \cdot JC$
$L_0 \leftarrow c_3 + c_4 \cdot JC$
$\lambda \leftarrow f(L_0, M_0, JC)$
$\beta \leftarrow g(L_0, M_0, JC)$
$\Omega \leftarrow 23.439 - 0.013JC$
$\alpha \leftarrow \arctan 2[(\sin(\lambda)cos(\Omega) - tan(\beta)sin(\Omega)), cos(\lambda)]$
$\delta \leftarrow \arcsin[\sin(\Omega)\cos(\beta)\sin(\lambda) + \cos(\Omega)\sin(\beta)]$
$HA \leftarrow t_{LSR} - \alpha$ **if** $HA > \pi$ **then**
  $HA \leftarrow t_{LSR} - \alpha - 2\pi$
$\phi \leftarrow \arctan 2[-sin(HA), (cos(\beta)tan(\delta) - sin(\beta)cos(HA))]$
$\theta \leftarrow \arcsin[\sin(\beta)\sin(\delta) + \cos(\beta)\cos(\delta)cos(HA)]$

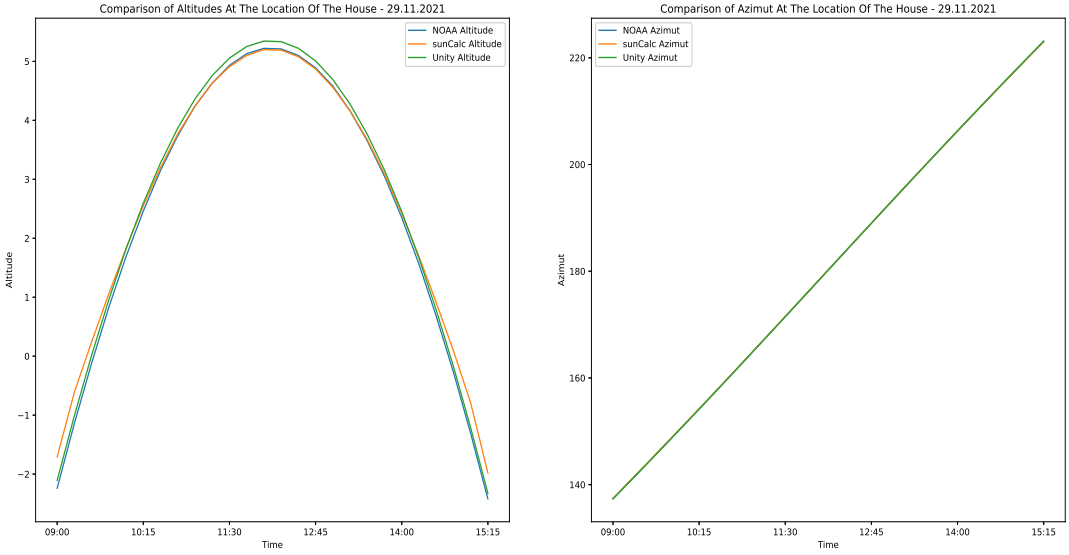


Figure A.1.: Comparison of my sun position algorithm with NOAA and sunCalc, given the longitude and latitude of the house for a specific day spanning sunrise till sunset. There is a small but negligable difference in the altitude for the Unity algorithm.

Elias Mohammed Elfarri

Digital Twin of a Building

NTNU
Norwegian University of
Science and Technology