Lehrstuhl für Computergestützte Modellierung und Simulation
TUM School of Engineering and Design
Technical University of Munich

Bim-Based Progress Monitoring Using 2D Semantic Segmentation

Scientific work to obtain the degree

Master of Science (M.Sc.)

at the TUM School of Engineering and Design
of the Technical University of Munich.

Supervised by Prof. Dr.-Ing. André Borrmann
Fabian Pfitzner M.Sc.
Lehrstuhl für Computergestützte Modellierung und Simulation

Submitted by Felix Friedl ([redacted])
[redacted]

Submitted on 01. April 2024

Abstract

According to various statistics concerning digitization, the construction sector is listed at the bottom. Despite current developments in other industry sectors, only little advancements are made, resulting in risks of cost and time overruns. Multiple sources trace these problems back to progress monitoring in the complex and dynamic on-site environments. This thesis provides a novel approach towards automating this critical and time-consuming task. It involves creating a new dataset of annotated site images and training semantic segmentation models to recognize cast-in-place concrete walls, columns, and slabs in panel, rebar, and concrete phases. Continuous site images are segmented and automatically processed, including averaging techniques to detect discrete element-specific progress timestamps. These are coupled with a BIM model or digital twin using the element's GUID to provide the results for further computations.

As verification, in-depth case studies are performed. Hereby, 49 semantic segmentation models are trained, and the derived construction progress timestamps are compared against as-built information for selected elements, resulting in reliable accuracies. Based on a real-world project, monitoring data is coupled with a BIM model, producing promising results and uncovering incentives for further research. Overall, this thesis' novel approach provides potential improvements for the construction sectors digitization.

Keywords: *Progress Monitoring, Semantic Segmentation, BIM, Digitization*

Zusammenfassung

In Statistiken zur Digitalisierung landet der Bausektor oft auf den hinteren Plätzen. Trotz der aktuellen Entwicklungen in anderen Branchen schreitet die Digitalisierung im Baualltag kaum voran, was zu Risiken für Kostenexplosionen und Zeitüberschreitungen führt. Vielfach werden diese Probleme auf die Fortschrittskontrolle in der komplexen Umgebung von Baustellen zurückgeführt.
Diese Masterarbeit entwickelt einen neuartigen Ansatz zur Automatisierung dieser kritischen und zeitaufwändigen Aufgabe. Dieser beinhaltet die Erstellung eines neuen Datensatzes aus annotierten Baustellenbildern und das Trainieren von Semantic Segmentation Modellen, sodass diese Ortbetonwände, -stützen und -platten im Schalungs-, Bewehrungs- und Betonzustand erkennen. Kontinuierliche Baustellenbilder werden segmentiert und automatisch zusammen mit Mittelungstechniken verarbeitet, um elementspezifische diskrete Baufortschrittszeitpunkte zu bestimmen. Diese werden mit einem BIM-Modell oder einem digitalen Zwilling unter Verwendung der GUID des Elements gekoppelt, um die Ergebnisse für weitere Berechnung zur Verfügung zu stellen.
Zur Validierung werden eingehende Fallstudien durchgeführt. Dabei werden 49 Semantic Segmentation Modelle trainiert und die abgeleiteten Baufortschrittszeitpunkte mit As-Built-Informationen für ausgewählte Elemente verglichen, was zu verlässlichen Genauigkeiten führt. Zusätzlich werden die Überwachungsresultate eines realen Projektes mit einem BIM-Modell verknüpft. Dies erreichte gute Ergebnisse und zeigt Anstöße für weitere Forschung auf. Zusammenfassend liefert dieser Arbeit mögliche Verbesserungen für die Digitalisierung auf Baustellen.
Stichworte: *Baufortschrittskontrolle, Semantic Segmentation, BIM, Digitalisierung*

Contents

Acronyms

2D	two-dimensional
3D	three-dimensional
ANN	Artificial Neural Network
BIM	Building Information Modeling
CNN	Convolutional Neural Network
FCN	Fully Convolutional Neural Network
FPS	frames per second
GT	ground truth
GUID	globally unique identifier
IFC	Industry Foundation Classes
lr	learning rate
mAcc	mean accuracy
mIoU	mean intersection over union
ReLU	rectified linar unit

Chapter 1

Introduction

The construction industry has always been one of the least digitized industry sectors according to the digitization index by MGI or McKinsey [1], [2]. The data from 2015 in figure 1.1 shows that in nearly all areas of the study the construction sector was classified to have “relatively low digitization”. Although this research was carried out nine years ago, its findings remain true today. Despite the current developments towards Industry 4.0, including digital twinning, virtual reality, or additive manufacturing, the processes involved in planning and fabricating buildings often remain unchanged. Therefore, there has been little to no increase in productivity during the past decades. When looking at the construction sector in the US, the overall productivity is even lower than in 1968 [3]. However, multiple sources confirm that most people employed in the sector are eager to incorporate digital processes as they expect major enhancements to cost estimation, progress monitoring, or schedule reliability [4], [5]. These areas in particular need improvements as construction projects worldwide have to deal with increasing costs and delays. Said problems can be attributed to various factors, but project management has a key influence on them [6], [7].

Construction sites are inherently complex and dynamic environments involving multiple activities, materials, and personnel. Therefore, effective monitoring is needed to increase efficiency and productivity while, in turn, reducing the risk of cost and time overruns. However, despite the recent developments in research fields related to Building Information Modeling (BIM) or digital twinning, site monitoring is still done manually most of the time [3], [8]. These procedures often involve the project management personally visiting a construction site, checking each building element’s state to collect the necessary information, and comparing it to the required state listed in a schedule. Manual work like this is tedious and error-prone.

In some cases, surveillance cameras on construction sites are already in use, but the pictures and videos captured by them are then manually evaluated. Similar practices can only yield few improvements as automation approaches possible on this data are not utilized. According to different studies, this behaviour can be attributed to the general mindset of industry professionals towards more traditional methods. Additionally, next to the development of new monitoring frameworks, there also is a need for more information about the implementation of already existing software applications. Therefore, novel approaches towards site monitoring need to be proposed, which are easy to understand and can be use in everyday construction projects. [4], [9]

Researching and improving on automation approaches in this regard will lead to significant improvements in the current standard practice [4], [9]. Much innovation has already been achieved regarding BIM-based construction and digital twinning in combination

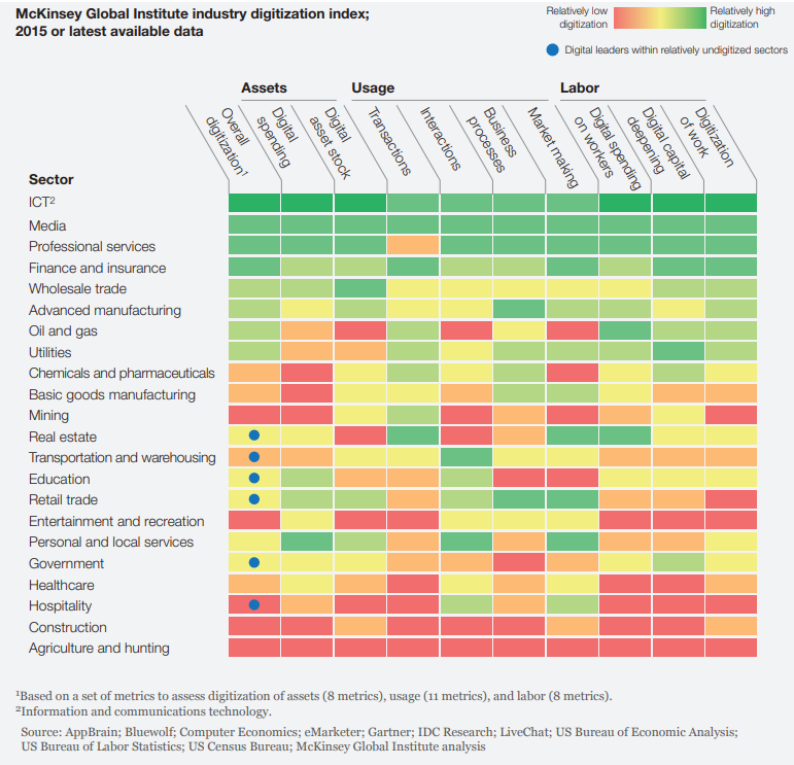


Figure 1.1: McKinsey Global Institute industry digitization index from 2015

with progress monitoring. In this context, a “Digital Twin” is defined as “a realistic digital representation of assets, processes, or systems in the built or natural environment” [10]. The term “BIM model” is frequently used synonymously, despite BIM-modelling describing only the three-dimensional (3D) geometry of a building enriched with semantic information. The significant detail, which distinguishes a digital twin from a general BIM model, is that it contains real-time information on the construction elements, the site, and all related processes [10]. The availability of this information for progress monitoring is an excellent motivation for incorporating digital twinning technologies [11].

Generating this kind of information by manual labour is very inefficient and not feasible. Therefore, automatic data-based processes are essential for a digital twin to effectively link the digital representation of a building to its real-world counterpart [10]. Numerous approaches towards collecting data on construction sites regarding various objects and purposes have already been developed and are stated in Section 3.1. However, these recent approaches lack monitoring functionalities for certain construction objects and show potential for improvements in regards to the integration of progress data into BIM models. Therefore, this thesis aims to research and provide a new approach for progress data collection on cast-in-place concrete elements (e.g. columns, walls and slabs), including the elements’ “panel”, “rebar”, and “concrete” phases. To detect those elements in site images, this thesis thoroughly investigates two-dimensional (2D) Semantic Segmentation within the context of construction progress monitoring. This computer vision technique allows the identification and classification of different objects in images and videos. Next to the segmentation part, a workflow for extracting element-specific progress data and for coupling these monitoring results with a BIM model or a digital twin will be discussed.

Chapter 2

2D semantic segmentation with convolutional neural networks

2.1 Convolutional Neural Networks

2.1.1 Fundamentals of artificial intelligence

The basis of the monitoring approach proposed by this thesis relies on automating the visual understanding of a construction site with the help of 2D semantic segmentation. To understand this discipline belonging to the field of computer vision, first some theoretical background on artificial intelligence and neural networks in general is needed.

The first mention of the term "Artificial Neural Network (ANN)" occurred in a paper from 1943 in the form of the so-called MP-Model. It was a proposal for a mathematical model of neurons that was heavily inspired by the behaviour of neurons in the human mind [12], [13]. This initial model was further refined to a single-layer perceptron model by F. Rosenblatt in 1958 and could later include the first stages of learning functionalities [14]. The last step towards modern ANNs was extending the model to multiple layers and including the so-called backpropagation. It is an algorithm that involves iteratively adjusting weights during a training phase to improve the models performance. These changes allow for better learning capabilities and the ability to solve nonlinear problems. [12]

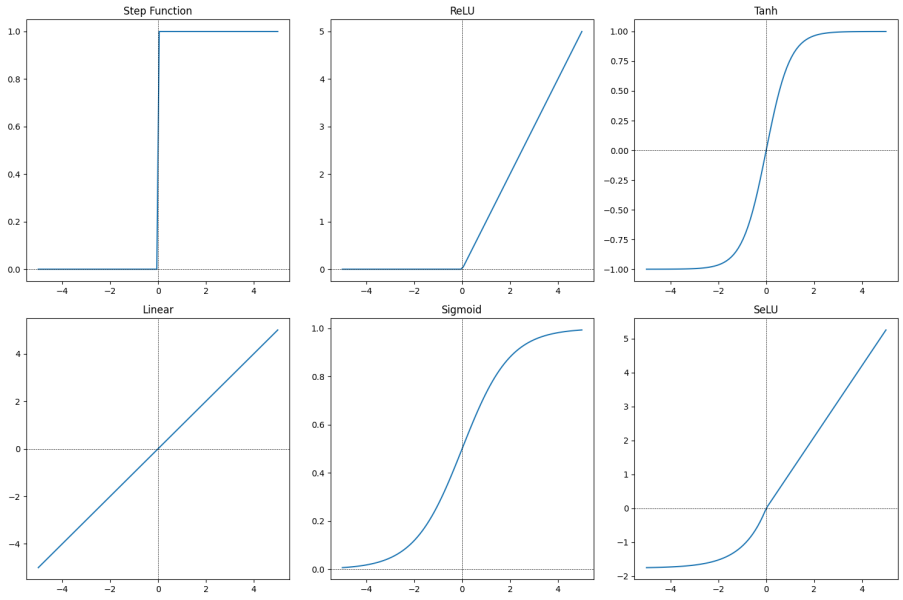


Figure 2.1: Different choice for activation functions

The basic functionality of the perceptron is still used today as the term itself is nearly synonymous with the term neuron. It takes all input values, multiplies each with a separate weight and adds them together, resulting in a weighted sum. While its values can lie within an arbitrary range, this sum can already be used as an output. However, it is usually beneficial to generate an output between 0 and 1, which is the reason for using an activation function. A popular choice within the domain of ANNs or Convolutional Neural Networks (CNNs) for this function is the rectified linar unit (ReLU) next to a threshold or step function. It returns a zero output if the input is equal to or less than zero and otherwise returns the input value. For other problems, choosing nonlinear activation functions like the Sigmoid or TanH function can be more helpful. Figure 2.1 provides an overview of multiple activation functions including the ones mentioned above. In conclusion, the activation function is necessary to represent nonlinear behaviour and convert the weighted sum to a normalized range. For some additional functionalities of neural networks, like the backpropagation algorithm, a critical trait of the activation function is its differentiability. In the case of the ReLU function, it is given from values upwards of zero. The last part of the perceptron structure is the bias. It functions as a threshold that needs to be reached for the perceptron to produce an output. A visual summary of the interaction of all parts is shown in figure 2.2. [13], [15], [16]

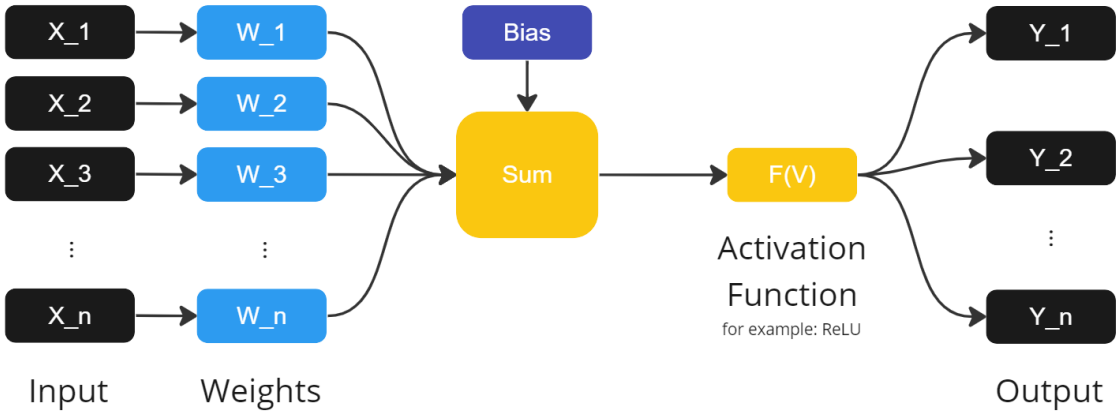


Figure 2.2: Functioning of a single perceptron

Modern ANNs consist of different layers, each housing a sometimes very high number of connected nodes called "neurons" or "perceptrons". The first layer is labeled "input layer" because it directly receives different values of a given problem as input. The last layer is called "output layer" and gives back the calculated solution to the problem. An arbitrary number of "hidden layers" can be allocated between these two containing multiple nodes, which are connected to all other nodes from the previous and following layer. They can change their internal parameters after each iteration in the training process to improve the calculated output from a given input. This behaviour is commonly called "learning" or "deep learning". [13]

The learning ability is based on the so-called backpropagation algorithm. Two passes through the model are performed over multiple epochs in the training phase. In each iteration, a forward pass through the network in performed, generating a solution from

the input, and a backward pass to optimize the model. In the forward pass each input is processed, and the resulting output is passed to a loss function. The goal of each iteration can be summarized as minimizing this function. Likewise to the activation functions, there are different choices for loss functions. The mean squared error or the mean absolute error function is often used for regression problems requiring the prediction of continuous values. In the context of classification problems, including image segmentation, the cross-entropy loss is frequently consulted. There are different sub-forms, but in general this function penalizes the model if it confidently predicts a wrong class label and rewards it for choosing the right one. The resulting loss values are then used in a backwards pass to optimize the weight and bias parameters. Applying the backpropagation algorithm, the model computes the gradient of the loss function with respect to the named parameters. Then it uses procedures like gradient descent to minimize the loss value by updating parameters in the direction shown by the gradient. The name "backpropagation" results from the computation of the gradient because it is first calculated with respect to the output of the last layer and then propagated backwards through the network. At each layer, the gradient is calculated again using the layer's inputs and the gradient from the previous layer. When the first layer is reached, all weights and biases of the model are updated and the next iteration of the training phase can begin. Different forms of learning can be distinguished by looking at the information given to the loss function. In unsupervised learning, the ANN does not receive a target output but only aims to decrease or increase an associated cost function as loss value in every iteration step. For image processing problems like semantic segmentation, supervised learning schemes emerged as best suited [13] and are used in this thesis. Hereby prelabeled inputs with a so-called "ground truth (GT)" are used as targets for the model's output. The loss function than quantifies the difference between the model's output and the expected output stemming from the GT. Minimizing this difference is then the training phases goal. [13], [17]

The magnitude of updates to weights and biases can be manually controlled with the learning rate (lr) hyper-parameter, which regulates the step size in which the parameters are changed after each iteration. This is used to influence the model's convergence and the training performance. Generally, it is the most useful and crucial parameter to change when experimenting with ANNs. A too-small learning rate results in very slow convergence and the possibility of getting stuck in local minima of the loss function. But too-big learning rates can make the model overshoot and never reach a convergence or get stuck in local maxima. Therefore, for each problem, an optimal setting needs to be found. Some training frameworks utilize learning rate scheduling, which enables dynamic adaption of the learning rate during the training process. Another significant hyper-parameter is the weight decay, which is added to the loss function to penalize huge weights. This reduces the risk of overfitting by discouraging the learning of very complex patterns that are only specific to the training data. Setting this parameter too high can result in the opposite behaviour of underfitting and rendering the model unable to grasp important patterns in the data. Both hyper-parameters can influence the training process and the resulting trained model crucially and, therefore, need to be chosen carefully. [18], [19]

2.1.2 CNN-based image processing

In the common configuration of an ANN, all nodes from one layer are connected to all nodes from the previous and following layer. These so-called "fully connected layers" result in a rapidly increasing number of modifiable parameters with larger input sizes. In the context of image processing, for a $28 \cdot 28$ pixel image with only black and white colours (see also MNIST database of handwritten digits [20]), a single neuron in the first hidden layer will contain $28 \cdot 28 \cdot 1 = 784$ weights, which poses a significant computational challenge. From this basic concept of ANNs, various models evolved to solve a multitude of different problems. This includes the aforementioned computational complexity problem concerning image processing in tasks like semantic segmentation, as shown in section 5.1.2. [13]

To remedy this, instead of fully connected layers, convolutional layers are introduced, which are only connected to a small fraction of the input values. This area is frequently called the receptive field of a neuron and can drastically reduce the computational complexity of the neural network model [13]. This effect can be seen clearly when looking at the number of weights required to process an arbitrary 4K RGB image. Typically, these images have a size of $3840 \cdot 2160$ pixels and 3 RGB channels. A node in a traditional ANN with fully connected layers would need to have $3840 \cdot 2160 \cdot 3 = 24.883.200$ weights in only the first hidden layer. In contrast to this, a node in a convolutional layer with a receptive field size of $12 \cdot 12$ would only need $12 \cdot 12 \cdot 3 = 432$ weights. This wast difference in computational complexity is the biggest advantage of CNNs over ANNs in image processing. In the context of autonomous vehicles, intelligent medical treatment or face recognition, CNNs can, therefore, successfully be employed as these cases are all special variants of image-processing questions. [12]

CNNs are feed-forward neural networks that can learn so-called convolutional kernels through the backpropagation algorithm [21]. This behaviour is inspired by visual perception, as each of these learned kernels represents a receptor, which can respond to different features in the input image. Activation functions and biases are in this context used to simulate a threshold that only allows signals of a specific strength to pass through these kernels. For the backpropagation algorithm and to enable the network's learning ability, loss functions in the form of cross-entropy loss or similar are needed [12].

The convolutional kernels are usually quite small in their spatial dimensionality but are used throughout the whole depth of the image. Each kernel is convolved over the input in each convolutional layer to produce a 2D activation map (also called activation or feature map) that represents a certain feature in the input image. From a mathematical perspective, convolution is a function that performs a sum of products between the kernel weights and input values while convoluting a smaller kernel over a larger image. A visual representation of the convolution process is shown in figure 2.3. The resulting features can range from simple vertical or horizontal lines to much more complex patterns. These can be seen as segmentation masks of object-specific features. The kernels or, strictly speaking, the weights inside them are the learnable parameters, which the model can change in each iteration to improve. Throughout the process, each convolution step generates a feature

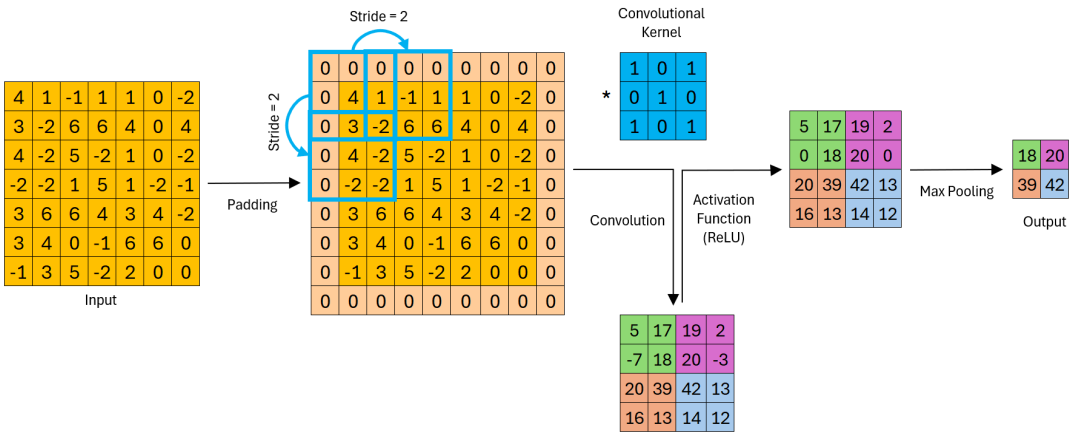


Figure 2.3: Procedure of a 2D CNN, based on [12]

map, which is stored as a multi-dimensional array, also called a tensor. Each dimension corresponds to an aspect of the input, such as the height, width and depth. The maps are passed onward through the network and refined within the training to finally generate the segmentation mask for the image. This final result is the output of the model and is called a "prediction". [12], [13]

One disadvantage stemming from the use of convolutional kernels is that the area perceived by each kernel can be relatively small. To increase it, dilated convolution was introduced, which enlarges the receptive field size of a kernel but, in turn, decreases the number of sampled points inside that area. Deformable convolution (or atrous convolution) can be used for irregularly shaped elements of interest to adapt the respective field dynamically to better resemble the element's geometry. [12], [13], [21]

The weights inside the convolutional kernels are shared between multiple nodes. This reduces computation time further and simultaneously utilizes the fact that activations are very likely to be useful in multiple areas of an image. Each individual kernel and its corresponding activation map is restricted to the same set of weights and biases, which results in each neuron directly producing an overall gradient in the backpropagation algorithm instead of node-specific ones like it is done in conventional ANNs. This gradient can be totaled across the whole depth of the input, meaning only a single set of weights needs to be updated in each iteration instead of every single weight individually. This massively reduces the number of learnable parameters and, therefore, the required training and computation time. [13], [22]

The main parts and algorithms found in most forms of a CNN can be seen in figure 2.3 as well as figure 2.4 and are the following: [12], [13], [22]

- Input Layer: This Layer is standard for every neural network and, in the case of semantic segmentation, holds the pixel values of the input image.
- Padding: Most of the time, padding is used to extend the image border with zero values to prevent the loss of information during the convolution process in these areas.

- Convolutional Layers: These are the main part of CNNs. Each node in the convolutional layer is connected to a particular subregion of the input and uses a learnable convolutional kernel. This kernel is convolved of the region and the scalar product between the kernel weights and the corresponding input values is stored as an output. The stride value determines the distance or speed at which the kernel moves over the input image. Most models utilize the ReLU or the sigmoid function to apply elementwise activation functions to the previous layer's output. This procedure can be seen in figure 2.3.
- Pooling Layers: Pooling reduces the spatial dimensions of the input volume for each depth level by taking the maximum value (max-pooling) or average value (general pooling) within the pooling window. This gradually reduces the dimensionality of the representation and the computational complexity of the model. The pooling layer operates on each activation map and usually has a size of $2 \cdot 2$ with a stride of 2, which would reduce the map size by 25%. There are other forms of pooling, but kernels with a size above three are usually not used due to the destructive nature of this process. In figure 2.3, a $2 \cdot 2$-sized max-pooling layer with a stride of 2 is used, which returns the maximum value from a $2 \cdot 2$ area of the previous layers activation map.
- Fully connected Layers: These Layers work the same way as in conventional ANNs and at least one fully connected layer is necessary in a CNN. It attempts to produce class scores using the activation maps from the convolutional layers. If there are multiple fully connected layers, ReLU is frequently used between these layers to improve the performance.

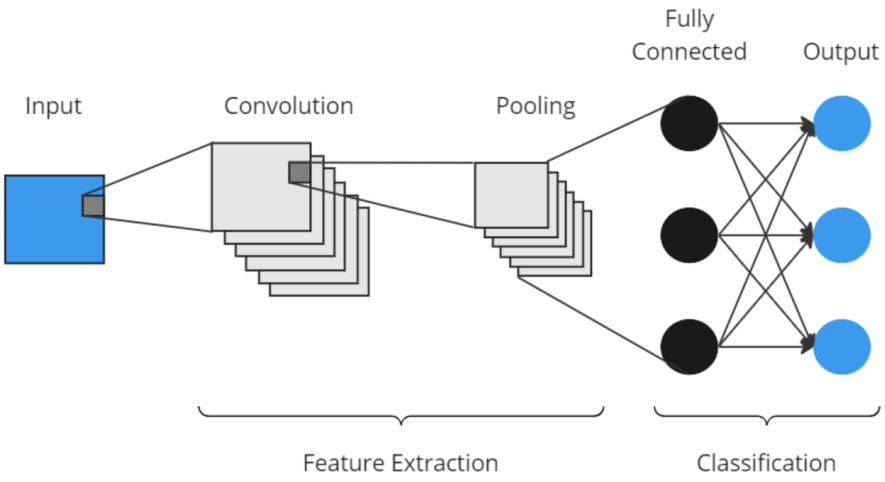


Figure 2.4: Different layers and operations of a 2D CNN, based on [12]

CNNs have three main advantages over conventional ANNs regarding image processing: [12]

- Local connections: Each neuron is not connected to every neuron on the next layer but only to a small number, which improves the training process by reducing parameters and increasing the convergence speed.

- Weight sharing: connected nodes share the same weights in order to further reduce the number of trainable parameters
- Down-sampling: Using pooling-layers, the dimensionality of images is reduced while retaining important information

Traditional ANNs are fully connected networks as they connect each neuron to every other neuron in the adjacent layers, which can lead to an explosion in the number of parameters, resulting in high computational costs. In contrast to this, CNNs leverage weight sharing and local connectivity through convolutional layers, reducing the number of parameters and enabling more efficient feature learning from images.

A further extension to CNNs are fully convolutional layers, which allow the model to work with arbitrary-sized image inputs. These replace the few remaining fully connected layers in CNNs (like the input or output layer). A so-called Fully Convolutional Neural Network (FCN) works by learning a mapping from pixel to pixel without computing region proposals beforehand. CNNs in contrast can only produce labels as outputs from specifically sized inputs because of the remaining fully connected layers and their fixed size. The main shortcoming with these types of neural networks is that the output is of shallow resolution because of the high number of convolution and pooling steps. To overcome this problem, many model architectures have included multilayer schemes and deconvolution layers with bilinear upsampling or alternative methods like deep deconvolutional networks [23], [24]. In this thesis, both CNNs and FCNs are tested because the used images are of a fixed size, which works for both network types.

CNNs in image processing can include aspects from other areas of applications like natural language processing. Some examples for this are the Swin-Large [25] or ViT [26] backbone using a vision transformer. A bockbone in this context describes the core algorithm of an ANN or CNN. The authors of the Swin backbone [25] state in their paper that transformers stem from network architectures used in natural language processing. They are designed for sequence modelling and transduction tasks excelling at depicting long-range dependencies. The use of transformers in image processing was hindered for a long time by the fact that word tokens as fundamental elements in language processing are of a small and fixed scale. In contrast, visual elements have bigger and varying sizes. This can directly be transferred to the resolution required to solve both problems. The smallest necessary element in language processing is a single word inside a passage of text. In contrast, image processing tasks like semantic segmentation need pixel-wise dense solutions, which would not be possible with standard transformers. Liu et al. [25] and other research teams proposed different solutions to those problems in the form of dedicated vision transformer backbones.

However, as for example described in a paper by W. Wang et al. [27], new takes on traditional CNN algorithms were proposed as an answer to the rise of the aforementioned vision transformers. Their InternImage backbone proved that classic convolutional networks can perform as well as or even better than networks with transformer backbones.

To achieve this, they reiterated over the deformable convolution algorithm and combined it with stem and downsampling layers as well as scaling rules.

There are multiple other approaches towards different segmentation algorithms and backbones. As this specific research field does not belong within the scope of this master's thesis, further elaboration is omitted at this point. Specific information concerning all semantic segmentation models used in the thesis can be found in their respective sources.

2.2 Semantic segmentation as a discipline of computer vision

Computer vision comprises various disciplines with one being image classification. It is a fundamental task in this genre with the goal of categorizing an entire image into one of several predefined classes or categories. This process looks at the image as a whole and tries to recognize patterns and features in it to assign it an appropriate label. Another discipline in this area is object detection. Hereby, objects inside an image are located and a bounding box is drawn around them. Image segmentation aims at partitioning an image into multiple segments or regions based on certain characteristics such as color, texture, or shape. This is considered a very challenging problem as, in contrast to object detection, no knowledge of the visual concepts of the objects to be segmented is known in advance [23]. This technique divides a given image into different reasonable sections of coherent pixels by assigning them a categorical label so that all pixels with the same label represent a distinct element inside the image. While the segmentation task only outputs which pixels belong together, semantic segmentation additionally classifies these groups into a finite set of predefined labels. [21], [23]

An ANN or CNN performing semantic segmentation can work autonomously after intensive training on the specific task, but this requires a large number of images with pixel-wise segmentation masks as so-called "ground truth". These are needed for the backpropagation algorithm within the context of supervised learning schemes. Creating this data often resembles the most time-consuming and expensive task in working with segmentation models. A possible workaround for this is the use of weakly supervised learning schemes utilizing only annotated bounding boxes or even image-level labels. Alternatively, the implementation of various data augmentation methods to artificially increase the dataset size can be helpful [28]. However, multiple segmentation approaches retain the problem of generality by only producing accurate solutions for tasks very similar to their training and fail to work successfully on different problems [23]. Therefore, better or utterly different training images must be created for new questions, as it is the case for this master's thesis.

During training and testing phases, the performance of a segmentation model needs to be evaluated. For this task, two significant metrics are used most of the time. The first and most important one is the "mean intersection over union (mIoU)" which provides a comprehensive assessment of the models segmentation performance across all classes comprised in one single metric. It is calculated as the ratio of the area of overlap between

the predictions and GT masks to the area of their union. Then the mean value of this metric over all classes is used to measure the models performance. [12], [13], [21]

$$\text{mIoU}_{class} = \frac{\text{area of overlap}}{\text{area of union}}$$

Additionally, the "mean accuracy (mAcc)" metric is often given, which provides a similar evaluation in comparison to the mIoU metric. But especially for datasets with imbalanced classes like the newly created one used in this thesis, the mAcc provides a more balanced summary of the models performance. This metric is calculated for each dataset entry as the proportion of correctly classified pixels compared to all pixels in the image. Than once again the mean value over all images is derived and used for the evaluation. [12], [13], [21]

$$\text{mAcc}_{class} = \frac{\text{correctly classified pixels}}{\text{all pixels}}$$

Both the mIoU and the mAcc metric can range from a value of 0 to 1. However, within this thesis, they always will be represented as percentage values. Meaning, a mIoU score of 0,7543 will be shown as 75,43%.

The segmentation task is a significant field of research due to its many real-world applications like satellite imagery (e.g. house number or street recognition), forensics (e.g. fingerprint, iris or face-recognition and voice recognition) or medical image processing (localizing aneurysms, tumours, cancer or specifics organs). Also cultural heritage preservation, image copy detection, on-the-fly visual search, human-computer interaction or surveillance and autonomous driving are important use cases [21]–[23]. Especially progress monitoring on construction sites can be enhanced using semantic segmentation, as shown by various researchers in the following chapter.

2.3 Existing datasets for semantic segmentation

For the purpose of evaluating a newly developed segmentation algorithm, large curated datasets exist on which those models can be trained and tested. This eliminates the need of creating a new dataset for every model and provides a platform on which all models can be easily compared to each other on a fair basis.

A very prominent example is the “Cityscapes” dataset for urban scene understanding [29]. It consists of 20.000 annotated images for training purposes and 1.525 testing images. All of them were taken by a car-mounted camera while participating in everyday urban traffic in different cities of germany and neighboring countries. The annotations consist of 19 classes including cars, pedestrian, and roads signs among other labels. The main focus of this set is to assist in developing segmentation models, that can understand scenes in urban traffic to help advance higher-level technologies like autonomous driving.

Furthermore, there are other large datasets that provide a less specific training and testing basis for segmentation models. The most common ones are the “Pascal Visual Object Classes (VOC) Challenge” [30], the “ADE20K Dataset” [31] or the “COCO-Stuff: Thing and Stuff Classes in Context” Dataset [32]. These consist of more than 10.000 Images each and contain scenes from indoors as well as outdoors with over 100 annotated classes, so that they can be used to evaluate a segmentation algorithm’s performance on a general basis.

For comparing models without a further specific use case in mind, these aforementioned datasets are very suitable. But for particular tasks, like in the present thesis, new datasets need to be put together which most of the time contain less images and annotated classes because they have a more limited scope that is tailored towards the research goal. A first example of this procedure and also an exception to the mentioned size limitations is a dataset of annotated construction-site images created by X. Luo et al. [33]. It contains 7.790 Images and 22 annotated classes organized in a tree like structure, focusing on activity-specific equipment, construction materials and construction workers. As the paper states, it is still one of the larger datasets but to achieve this, the researchers had to spend considerable resources and time on annotating the images. Others like Z. Wang et al.[34] created a dataset, which consists of 859 labeled images and 12 classes. In contrast to the aforementioned one, this set exclusively focuses on construction equipment like excavators or cranes as well as humans and therefore contains a far smaller number of classes and images alike. Following a similar approach as the present thesis, Z.Wang et al.[35] assembled a dataset of 660 Images that only focus on precast walls in order to train a segmentation model to recognize the timestamp at which the installation process of a precast element is completed.

Further explanation towards the research approaches behind these specific datasets as well as other scientific contributions in the research fields related to this thesis are provided in the following chapter.

Chapter 3

Scientific background

There are many applications of computer vision and digital twinning, as the theoretical background concerning artificial intelligence and convolutional neural networks established. Some of them also pose significant benefits to the construction sector, which have already been explored by different researchers. In the following paragraphs, short summaries of publications relevant to the proposed monitoring setup of this thesis are provided.

3.1 State of the art

3.1.1 Application of computer vision in construction

Within the research field of computer vision, convolutional neural networks are a well-established tool especially for image segmentation tasks. Multiple techniques for various particular problems in this domain often leverage deep learning approaches, as an overview by S. Ghosh et al. shows [21]. The segmentation task has gained a lot of prominence in computer vision because of its many applications in medical image processing, surveillance, or satellite imagery, and in robotics, or autonomous driving. Therefore, experts from those branches have already developed and improved on various deep learning-based algorithms and models for semantic segmentation. Numerous better-performing solutions rely on convolutional neural networks because of the advantages of convolutional layers stated in chapter 2. Many examples of said models are investigated in section 5.1.2 within the context of this masters thesis.

Progress monitoring

Next to other prominent use cases, the application of semantic segmentation models to aid progress monitoring tasks is often discussed within the construction sector. In this regard, Z. Wang et al. [35] proposed a method which is very similar to the procedure investigated in this master's thesis. The authors created and annotated a new image dataset from construction site pictures containing precast concrete wall elements in different installation steps. By training a Mask R-CNN model on this data, the researchers were able to automatically detect the installation timestamps for such wall elements. This paper also includes a self-written Revit plug-in, which attaches the results as attributes to the correct wall elements within a BIM model.

While the last mentioned paper from Z. Wang et al. [35] focuses on a very specific group of elements to monitor, the researchers Golpavar-Fard et al. [36] proposed a

more general method. In their approach, unordered daily photographs are combined with computer vision techniques to generate point cloud data enriched by dense and volumetric reconstruction. The new algorithms proposed for that task result in their processes coping very well with occlusions. Element detection and tracking are enabled by machine learning techniques so that overall automatic progress monitoring is possible. This paper also introduces the possibility of coupling the detected construction progress with a 4D BIM model for comparisons between as-built and as-planned timestamps and visual representations of the derived differences. This paper clearly states the benefits of the connection between progress monitoring automation and BIM workflows, which will also be explored within this thesis.

Another proof of a similar concept is provided by A. Pal et al. [37], who introduced an activity-level construction progress monitoring framework that also takes construction site images and a BIM-model as input. However, the aim of their approach is not a binary assessment of the current state of a building element but a percentage-based progression evaluation that can be used to dynamically update schedules. Because segmentation models in the form of instance segmentation are used within this paper, the authors also needed to create a custom dataset of annotated site images tailored to their needs. The trained models then process images that are transformed into an orthographic view in a preprocessing step. From the prediction images, a completion percentage is then calculated. With additional information from the BIM model, a 3D point cloud can be reconstructed to visualize the current construction progress within a modeling environment.

While the described approaches gather site data via images, laser scanning is another established solution for this task. One main issue with this technique is that a very high number of points is required to reach a sufficient level of accuracy, resulting in very long scanning and postprocessing durations. To mitigate this, S. El-Omari and O. Moselhi [38] combined the point cloud data from a laser scanner with additional images from a digital camera on site. The number of necessary scanning points could be decreased by using information retrieved from the pictures. The combination of both methods, especially in shaded and cluttered areas, resulted in an overall richer representation of the element in the final result. This paper proves that the solution for some problems lies within more than one specific data-capturing approach but in combining multiple suitable ones.

Next to the monitoring of building elements, photogrammetry and video analysis can also be used to evaluate productivity and progress in earthwork operations. A paper by Bügler et al. [39] established a framework for this task to improve security for costs and schedules on construction sites. The researchers combined photogrammetry for soil volume estimation and video analysis for construction activity statistics into one approach that focuses separately at progress and resource utilization tracking. The potential benefits of optimizing resource allocation based on productivity insights are highlighted by the authors. A real-world case study demonstrates various difficulties in correctly assessing productivity when dealing with potentially inaccurate computer vision results. Averaging

and normalization methods are implemented to retrieve reliable and precise productivity assessments, which is also a part of this thesis' methodology.

The last five papers all propose different approaches using multiple variants of input data for various sub-problems within construction progress monitoring. A summary of even more methods in this research field is provided in the work of Yang et al. [40]. The mentioned techniques take still images, time-lapse photos and video streams as input and use computer vision algorithms to implement 3D reconstruction, object recognition, semantic segmentation, or worker tracking. Next to progress monitoring, these processes can also be used to aid in health assessment, accident prevention, or carbon footprint calculation. While numerous approaches already exist, the authors emphasize the need for better collaboration between researchers and industry professionals to create real-world applications from the proposed theoretical concepts. As the publications by Golpavar-Fard et al. [36] and A. Pal et al. [37] already stated, Yang et al. [40] also explain the importance of combining multiple different sensors and data collection methods to ultimately gain real-time data on construction progress, which can be used in further evaluations.

Construction site safety

While progress monitoring is one of the most researched use-cases for segmentation models and computer vision in construction, applications in other construction related areas are possible. Q. Fang et al. [41] state a framework concerning on-site health and safety. Their proposal wants to improve the security of on-site work utilizing face- and trade-recognition. The researchers trained a neural network to detect and extract workers' faces from on-site video streams. Also, the model evaluates their movements and interactions with other site equipment to estimate the activity that they are currently performing. After combining the obtained information, a license database is consulted to assess whether a worker is qualified or non-certified for the concerned task.

Next to this specific approach, a more general take on on-site safety is chosen in a paper by Z.Wang et al. [34]. Once again, a new dataset is created for their work containing 12 labelled classes such as humans, excavators, loaders, cars or fences. The segmentation models trained on this data can then visually understand a construction site and its activities in general by recognizing borders and moving parts. This paper aims to aid in path-finding and obstacle-avoidance algorithms for mobile robotic systems or remote construction activities with this functionality. Those innovations can help reduce the workload of personnel and, in turn, mitigate on-site risk factors for them.

Damage assessment

Besides ensuring safety standards for workers on-site, deep learning approaches in computer vision can also aid in assessing the security of buildings and bridges. The researchers K.C. Laxman et al. [42] developed a monitoring tool based on CNN models for this matter to surveil concrete structures. Their approach can detect cracks and

measure their length, width and depth by segmenting and analyzing corresponding pictures taken by cell phones. Appropriate actions for each detected damage entry are provided automatically by the system based on the derived data so that the time spent manually evaluating the pictures is drastically reduced. High levels of accuracy could be achieved and the deployment of their proposed system is planned on recurring bridge examinations. Ultimately, it should enable the remote execution of those surveys.

Table 3.1: Summary of previous research

Source	**Content**	**Open questions**
	Progress Monitoring	
Z. Wang et al. [35]	Detection of precast wall installation from video with a Mask R-CNN Model	Extension of the dataset, Detection of other building elements
M. Golparvar-Fard et al. [36]	Construction progress monitoring and visualization with unordered daily images and 4D BIM	4D volumetric reconstruction, Progress detection on element surface level, Progress sequence knowledge
A. Pal et al. [37]	Activity-level completion percentage-based construction progress monitoring	Expansion of the dataset, Reality capture plan optimization, Performance improvement
S. El-Omari and O. Moselhi [38]	Combining image data collection with laser scanning	None stated in the paper
M. Bügler et al. [39]	Assessment of productivity in earthwork processes using photogrammetry and video analysis	Combination of data recorded by multiple video cameras on-site to decrease occlusions
J. Yang et al. [40]	Summary of construction performance monitoring via still images, time-lapse photos, and video streams	Combining multiple types of computer vision disciplines, Closing gaps between research and industrial application
	Construction Site Safety	
Q. Fang et al. [43]	Detection of non-certified work on site	Getting the software alerts to workers on site
Z. Wang et al. [34]	Segmentation of different construction objects for automation	More labelled data, Testing the limits of current segmentation models
	Damage Assessment	
K.C. Laxman et al. [42]	Automated crack detection in concrete bridges	None stated in the paper

3.1.2 Digital twinning in construction

The term "digital twin" within the context of construction is ill-defined as different technologies and processes are allocated to this concept by various researchers. Digital twinning generally builds on the existing ideas of BIM, automated data acquisition and artificial intelligence. Sacks et al. [10] proposed a set of four core information and control sets. These include at first a modelling and then a building phase. Third, monitoring and interpretation steps are allocated and followed by a last phase of evaluations and improvements. The key concept behind this formulation is that the researchers do not interpret "digital twinning" as a mere extension to BIM workflows by integrating sensing and monitoring techniques. While these data acquisition methods play a critical role, "digital twinning" in construction focuses on closing control loops to retain vital workflows for information storage, processing functionalities and monitoring technologies. As construction sites are inherently complex and dynamic environments, on-site monitoring to gain real-time information on the construction progress is crucial when working with digital twins.

However, to utilize the full potential of this concept in everyday monitoring, large amounts of data need to be collected. Schlenger et al. [44] provide a summary of the state of the art concerning data collection and information processing on different construction sites. The monitoring frameworks include image-based ones, and laser scanning, or Bluetooth Low Energy based approaches. The paper also highlights the importance of structuring and evaluating the wast amounts of available data. Only then can it successfully be used for higher-level monitoring tasks or other evaluations. However, the authors concluded that, in addition to other elements, generic methods for management and evaluation processes are still lacking.

Pfitzner et al. [45] propose a knowledge graph-based approach to enable insight into various construction processes. This contribution can be directly linked to the need for better data management approaches, as described by Schlenger et al.'s work. With their object detection-driven framework, the researchers achieved high accuracies in automatically processing image data. Using a graph database, numerous data analysis methods are possible. An example of investigating the correlation between temperatures and worker activities was showcased in a case study. In line with previous contributions, tha authors state, that the lack of open-source datasets hinders the development of further on-site investigations.

Another monitoring approach for data mining and understanding specific on-site activities was proposed by the same authors Pfitzner et al. [46]. In their work, they again emphasize the need for automated monitoring techniques and combining diverse data sources to gain insights into construction processes. The researchers focused on modern data analysis methods to process data from various sources like BLE sensors, crane cameras, or laser scanners. The publication aims at capturing detailed information and knowledge about construction activities and processes. For this task, they present a data pipeline to integrate different data sources in the context of digital twin construction. The authors could verify these claims in case studies at construction sites in Germany. However, they

also emphasize the need for further research into integrating additional data sources so that a comprehensive understanding of the as-performed construction process can be built.

A primary image-based source of information in monitoring frameworks is semantic segmentation, making it an essential part of digital twin construction. In the modelling phase, elements are represented as accurately as possible and ultimately need to be linked to their real-world counterparts by real-time data. Combining artificial intelligence and big data analytics, H.H. Hosamo and M.H. Hosamo [47] showed a workflow of automatically creating a digital model of simple bridges based on points clouds stemming from laser scanning. The specific aim of this approach was chosen because bridges are the most crucial element of the road infrastructure, and their continuous maintenance is essential. Digital twins of bridges would be very beneficial in enabling efficient decision-making over the whole life-cycle of the structure. However, due to the high average age of most of them, next to no digital data is available, so alternative ways of gathering the required information for modelling them are necessary. With the possibility of automatically generating a digital model from raw point cloud data, maintenance and surveillance information stemming from Laxmans [42] or other monitoring systems can easily be integrated into a digital twin of the bridge.

In its most helpful form, a digital twin should contain not only the building itself but also the surroundings, including the construction site before, during and after the manufacturing phase. Only comprehensive information in this regard enables other new deep learning-based applications. M. Kamari and Y. Ham [48] developed an AI-based approach working on this information to evaluate constructing site disaster preparedness. Their proposed framework was researched in response to the high damages hurricanes inflicted on construction sites. The workflow relies on image-based scene reconstruction and the semantic segmentation of site images to detect potential wind-borne debris at a 2D level. With this information as a basis, risk-associated heatmaps can be generated by projecting the calculated results into the 3D BIM model. This can eliminate the need for construction workers to manually check the whole site before hurricane events, resulting in time and especially cost savings.

Table 3.2: Summary of previous research concerning digital twin construction

Source	Content	Open questions
Digital Twinning		
J. Schlenger et al. [44]	Review of data collection frameworks, data structuring and processing	generic data evaluation algorithms; relations between workers and materials or equipment
Pfitner et al. [45]	Object Detection Based Knowledge Graph Creation	Lack of open datasets; on-site activity investigation
Pfitner et al. [46]	Heterogeneous data acquisition and fusion for data mining on construction sites; Integration of additional data sources	
H. H. Hosamo and M. H. Hosamo [47]	Creating a digital twin model for bridges based on 3D point clouds	Focus on Building inspection, modelling and data management with digital twins
M. Kamari and Y. Ham [48]	Construction site disaster preparedness	Further systematic assessment of other disasters apart from hurricanes

3.2 Research gap and contribution of this thesis

This thesis aims to revisit the semantic segmentation-based approaches proposed by Z. Wang et al. [35], A. Pal et al. [37] or Laxman et al. [42]. These papers proved the viability of such monitoring frameworks but lacked the required functionality to work with cast-in-place concrete slabs, walls and columns. As the installation of these elements consists of multiple steps needing different personnel and equipment, efficient monitoring promises to significantly increase on-site productivity and reduce delays.

Therefore, in the first stage, it needs to be tested, whether 2D image segmentation is possible with site pictures captured by crane-mounted cameras. The goal is to extract accurate predictions regarding the aforementioned concrete elements and their different construction phases. The accuracy in detecting those three states, including a panel, rebar and lastly a concrete phase for each element type needs to be studied with sufficient granularity. A continuous stream of such segmented images is then utilized in the second stage to evaluate element-specific construction progress timestamps in real-time. This seamless monitoring framework can work faster and less computationally extensive compared to 3D laser scanning-based approaches. In contrast to other, more established ways of on-site data collection, this is the main advantage of segmentation-based approaches. With the high speed, real-time data agglomeration, combined with less computational complex-

ity, is possible. As this is a crucial part of successfully working with digital twin-based approaches, like J. Yang [40] described, a possible workflow for coupling the progress monitoring results with a BIM model is presented in the third stage of this thesis. The main focus lies on this part of the monitoring framework, as the connection between on-site monitoring and digital project representations poses multiple benefits stated in papers by M. Golparvar-Fard et al. [36] and Z. Wang et al. [35].

To test the proposed algorithms and workflows of all three stages, in accordance with other approaches, first, a new dataset of annotated images tailored to the segmentation of cast-in-place concrete slabs, walls, and columns is put together. Different semantic segmentation models that have already been established and perform well on curated datasets are trained and tested. Secondly, algorithms for extracting element-specific construction progress timestamps, including two averaging functionalities, are implemented. The accuracy of them is evaluated in a case study with six building elements from two real-world construction sites. Also, the proposed workflow for coupling the monitoring results with a BIM model is thoroughly tested on one project. Finally, the whole monitoring framework is deployed on a real-world construction site. In this case study, the required computation time of all necessary steps is recorded. Additionally, the accuracy of the monitoring results is evaluated by comparing them against as-built timestamps of selected elements.

In summary, this thesis contributes a new dataset for the segmentation of cast-in-place building elements as well as workflows for extracting element-specific progress timestamps from segmented site images, that are coupled with a BIM model of the project.

Chapter 4

Proposed approach

This master's thesis aims to provide a novel approach towards automating construction progress monitoring. This includes using semantic segmentation models and further processing of the resulting prediction images. The goal is to automatically detect element-specific construction progress timestamps and ultimately couple them with a BIM model or other digital twin representation of the project. Figure 4.1 provides an overview of this general outline and shows that the proposed approach consists of three main components. The first of them handles the segmentation of site pictures by creating a new dataset of annotated images and the training of segmentation models on it. In the second part, element-specific construction progress timestamps are extracted from a continuous stream of segmented site images. Finally, in the third part, a possible workflow for coupling these timestamps with a project's BIM model is presented.

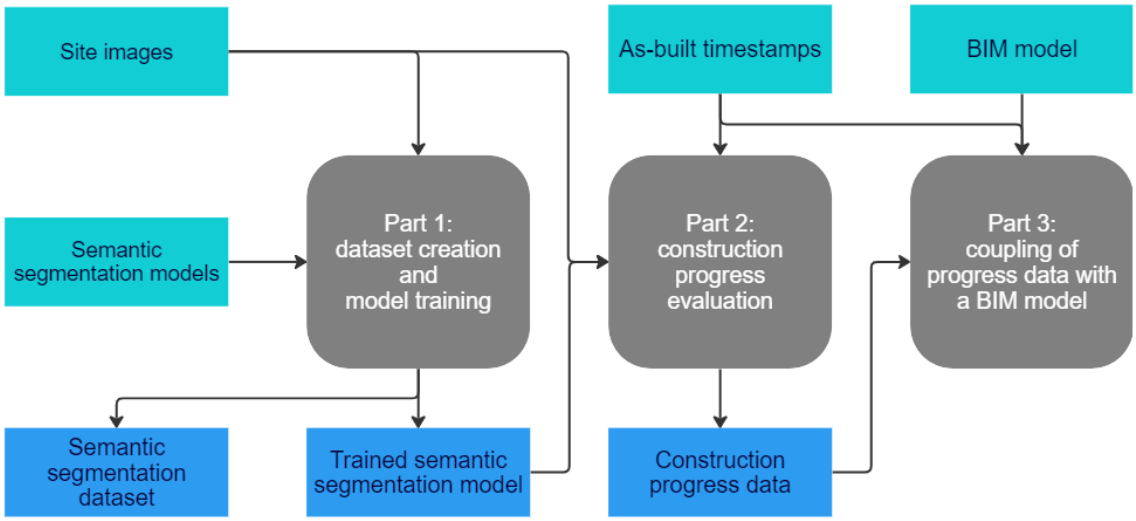


Figure 4.1: General workflow proposed in this thesis

4.1 Dataset creation and model training

In the first part of the intended framework the segmentation of construction site images is dealt with. The underlying process for it is further subdivided into different steps shown in figure 4.2. Because no trained segmentation models for this exact task are available, a custom semantic segmentation dataset as training and testing basis must be created at the start of this research.

The required pictures for it are sourced from an existing monitoring framework continuously capturing images of construction sites. For the dataset to be used for semantic segmen-

tation models, a so-called "ground truth" is created for each image during an extensive annotation phase. Multiple different semantic segmentation algorithms are then sourced for the task at hand. To find the best suited one available, all of them are trained on the newly created dataset and their performance is evaluated. Next to the finished dataset, which can be used to further aid in developing new segmentation algorithms, multiple trained models capable of confidently recognizing cast-in-place concrete elements are the result of this part of the proposed framework.

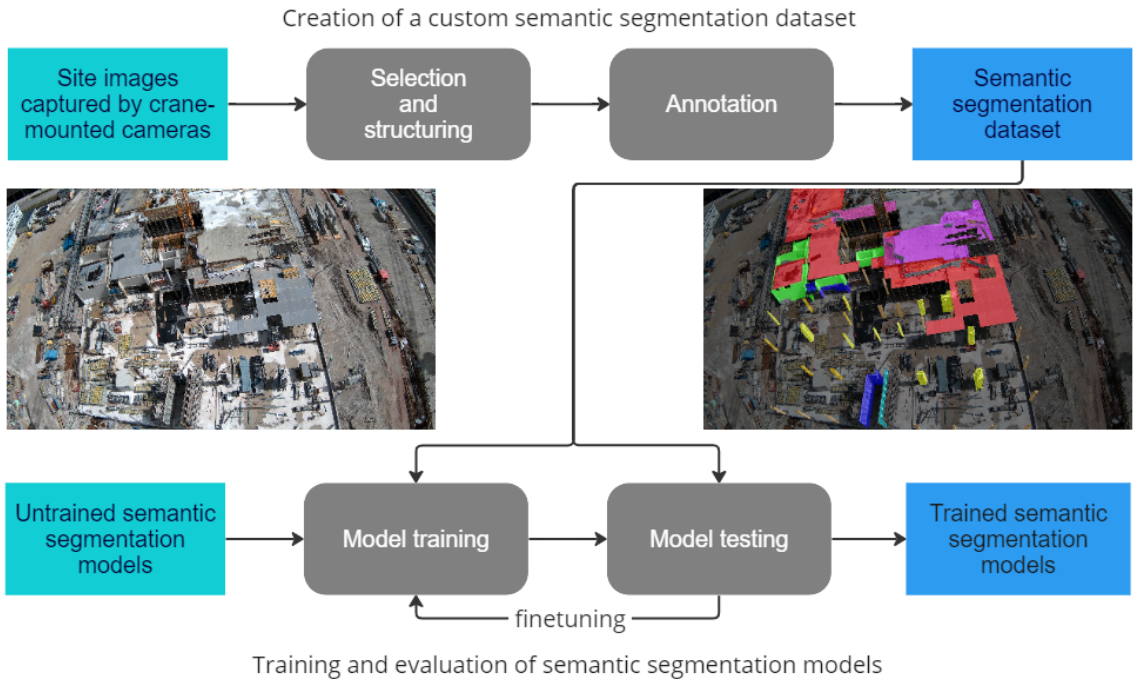


Figure 4.2: Part 1: Creation of a semantic segmentation dataset as well as training and testing of segmentation models

4.2 Construction progress monitoring

The trained semantic segmentation models are used to create prediction images from a continuous stream of construction site pictures. These images are captured by multiple crane-mounted cameras and closely resemble the data used in the custom dataset created in the first step. Additionally, a key attribute of those images is, that they retain a fixed camera viewpoint over the construction site during the whole monitored timespan. The goal of this part is to extract element-wise construction progress data from the prediction files.

For the proposed process shown in figure 4.3, additional GT style images are needed, which only contain one element of interest as annotated mask. In the figure an example is provided in the form of a single slab element, whose construction progress shall be evaluated. As the cameras viewpoint doesn't change, a single GT style annotation per camera is sufficient. The annotations are created manually for each element in this part of the thesis.

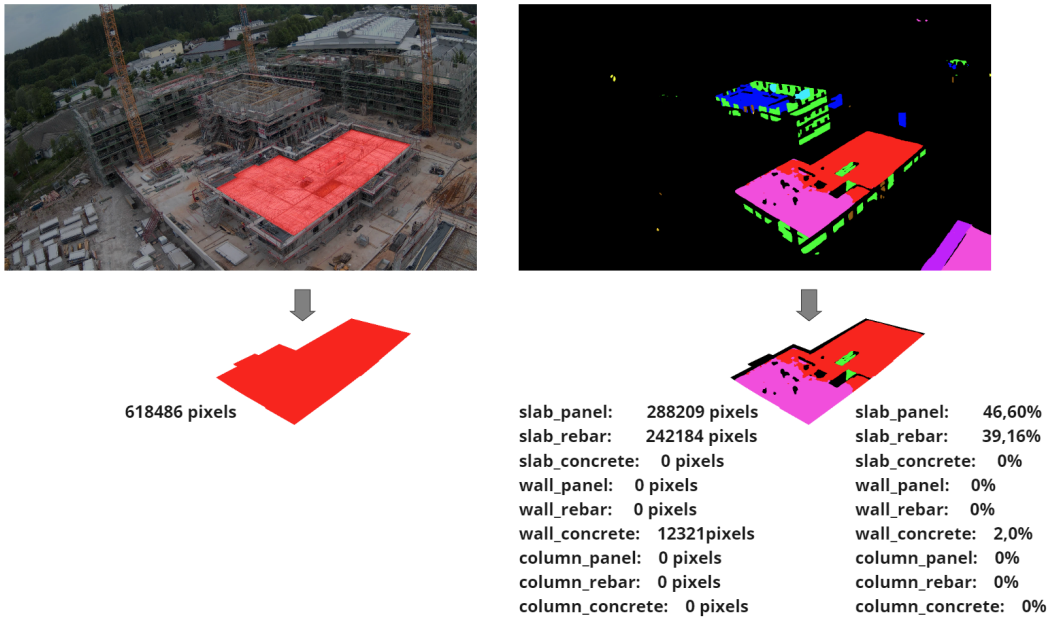


Figure 4.3: Proposed procedure for the extraction of element-specific construction progress from site images

In the evaluation process, each prediction image is compared against one element-specific GT style image and the segmented pixels within the mask area of the element are class-wise summed up. Before that, the bounding box of the GT mask area is calculated so that the comparison function only has to loop over the reduced bounding box area instead of the whole image to save on computational resources and time. The following code snippet as well as figure 4.3 show the intermediate result of this part of the evaluation:

```
{...
        "Reolink-013_00_20230513074712_RGB": {
                "background": 240314,
                "column_concrete": 7014, "column_panel": 0, "column_rebar": 0,
                "slab_concrete": 0, "slab_panel": 0, "slab_rebar": 0,
                "wall_concrete": 327520, "wall_panel": 12425, "wall_rebar": 0},
        "Reolink-013_00_20230513074742_RGB": {
                "background": 257643,
                "column_concrete": 6406, "column_panel": 0, "column_rebar": 0,
                "slab_concrete": 0, "slab_panel": 0, "slab_rebar": 0,
                "wall_concrete": 311945, "wall_panel": 11279, "wall_rebar": 0}
        ...}
```

For each prediction image one data-entry is generated holding the class-wise count of detected pixels within the elements' mask area. The proposed framework calculates the percentage-based share of perceived pixels for all classes in relation to the whole GT area's pixels. This intermediary result is also shown in figure 4.3. Then, it applies a user-chosen threshold to decide, at which point a particular construction progress milestone has been reached. In general, there are six timestamps evaluated including a starting point, which is flagged after reaching a low threshold value, and an ending point marked with a high threshold for the three construction states of a building element being "panel", "rebar" and "concrete".

Because the on-site cameras take a new picture every 30 seconds, a new data-entry needs is created at this frequency. Everyday construction progress could, therefore, be evaluated with very high accuracy in 30-second intervals. However, achieving this granularity is not necessary for determining construction progress in everyday monitoring, as evaluating timestamps in larger intervals of several minutes still provides sufficiently more data than manual monitoring. Additionally, evaluating every frame on its own may lead to incorrect conclusions as errors within the image generation or segmentation can not be compensated in this way. Therefore, a vital part of this framework is the incorporation of two different averaging techniques.

In a first step, all entries within a specified interval (called "averaging interval" in the upcoming case studies) are collected and a mean value for each class's pixel count over the interval is used in the evaluation. This decreases the number of necessary evaluations and the tendency to produced errors.

The initial evaluation generating the raw data shown in the code snippet needs to be performed separately for each camera because of the fixed but different viewpoints. The results of the final evaluation, however, can be produced by taking all cameras into consideration at once. Suitable functionality for this is also included within the proposed approach as a second averaging technique. A test of this procedure with six examples is provided in section 5.2.2.

A visual summary of the proposed process is shown in figure 4.4. It depicts the segmented site images and element-wise annotated ground truth pictures being used as inputs for the mapping process shown in figure 4.3. The raw data and progress timestamps received from this frame-wise evaluation are than passed along to averaging functions to produce accurate and at best error-free construction progress timestamps. Additionally, a potential evaluation of this framework is shown, which consist of comparing the derived timestamps against as-built ones for selected elements.

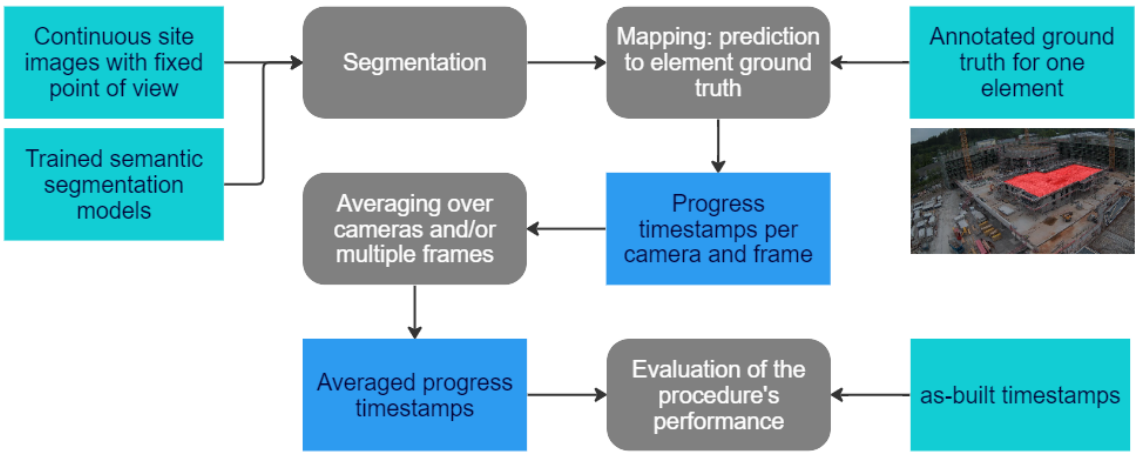


Figure 4.4: Part 2: Extraction from construction progress data from a site-image input

4.3 Coupling of resulting data with a BIM model

The monitoring workflow proposed in the second part of the intended framework is capable of evaluating element-specific construction progress timestamps from continuous site images but needs annotated GT style pictures for each element of interest. The third part of this thesis focuses on a way of automatically generating these GT style images from a BIM model. Then the evaluation results can be coupled with the digital representation of the building object. An overview of the proposed method is depicted in figure 4.5.

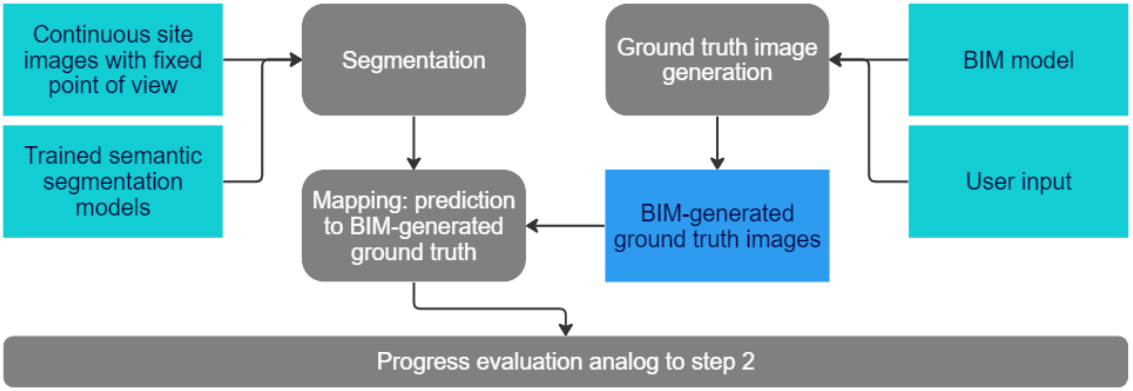


Figure 4.5: Part 3: Coupling of the construction progress data with a BIM model

Generation of GT images from a BIM model

For generating these GT style images, the workflow in figure 4.6 is proposed. The open-source Python library IFCOpenShell [49] is used in combination with a configuration file. The user specifies data paths as well as other parameters for the upcoming calculations, including the building floor, camera name and the Industry Foundation Classes (IFC)-type of interest within this file. At first, geometric information for all model objects from the selected building floor is extracted from the specified IFC-file containing the project's BIM model. In order to only work on elements, that match the criteria on which the segmentation models are trained, all objects within the IFC model are filtered to exclusively allow slabs, walls and columns to be processed. This is possible with BIM conform projects, as the model contains semantic information in the form of attributes attached to the geometric model objects [50].

To choose the elements of interest for this thesis, a selection can be performed by only allowing the IFC-types "IfcSlab", "IfcWall" and "IfcColumn". Additionally, the resulting subset of model elements is filtered using the following regular expression "stb" within their name attribute. In this way, aluminium or steel walls and columns as well as other elements not matching this prerequisite are eliminated.

The geometric information of all remaining model objects is plotted element-wise in a top-down perspective, resulting in only the ground floor area of the element being coloured red in an otherwise black image. The element's rotation in the plot is directly extracted

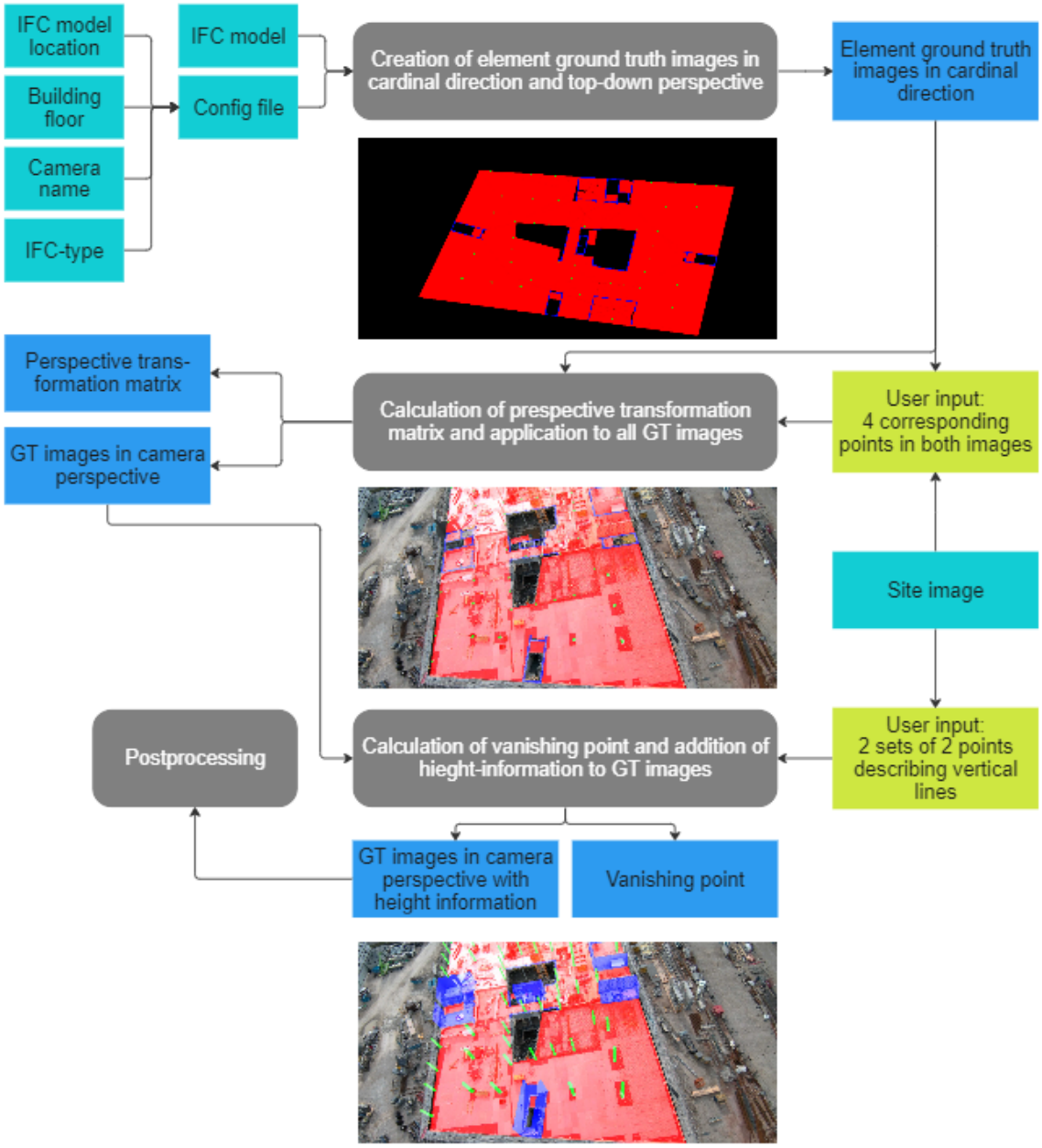


Figure 4.6: Overview of the model-based GT image generation

from the IFC file. This corresponds to the global cardinal directions of the building, which in most cases has the north direction facing upwards.

The size of those pictures matches the size of the site images to enable a pixel-wise comparison later on. Additionally, the bounding box of the BIM model is calculated and invisible markers for each corner of it are included in the element GT images. With this procedure, all elements are plotted true to their location relative to each other and can be transformed at once in a single step further down the line.

For the final coupling between progress data and the BIM model, another essential attribute called globally unique identifier (GUID) is used. As the name states, this attribute is unique to each model element and can be used to confidently identify it. For this reason, each element-specific GT image is stored with the corresponding elements GUID as filename.

Perspective transformation of the images

The GT images must be transformed to match the site cameras perspective, so that they can be used successfully in the proposed monitoring framework. However, the site pictures must be slightly preprocessed at first. Because of camera-specific parameters like focal length, optical center and lens distortion, the captured images contain some perspective and radial distortion, resulting in straight lines not appearing completely straight in the picture. Those parameters can be obtained by geometric calibration processes and then be used in correction functions to undistort and rectify the images [51], [52].

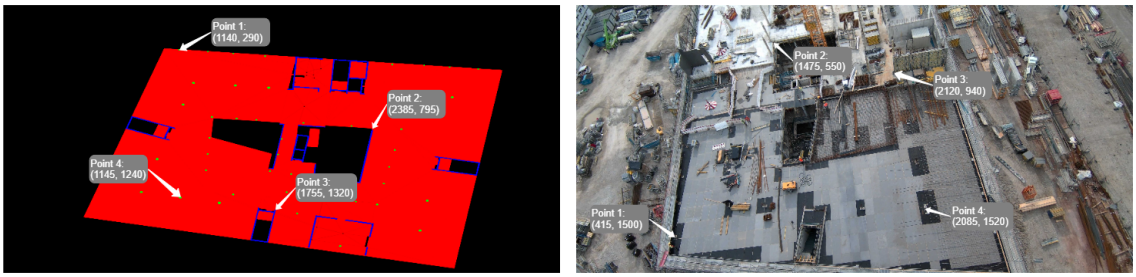


Figure 4.7: BIM model generated GT style image on the left, site image on the right; corresponding points for the calculation of the transformation matrix are marked in both

Separately for each building level and camera, the proposed framework needs user-input in the form of four corresponding points in the model-generated GT image and a random site image taken by the crane-mounted cameras. An example of this is provided in figure 4.7. With this input, a perspective transformation matrix is calculated. Hereby, at first, the pixel coordinates are transformed into homogeneous coordinates so each 2D point is represented by a 3D vector:

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Then, the transformation matrix H is constructed with these points using the least squares approach or the Direct Linear Transformation algorithm shown here [51]. At first, the points are collected in two matrices with X denoting the original and X' the transformed points. Often, a normalization step is included at this point to scale and shift points to improve the algorithm's accuracy.

$$\mathbf{X} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{bmatrix}, \quad \mathbf{X'} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \\ x_4 & y_4 & 1 \end{bmatrix}$$

A linear equation system can be constructed and solved from those two sets of points to obtain the transformation matrix.

$$\begin{bmatrix} x'_1 &= h_{11}x_1 + h_{12}y_1 + h_{13} \\ y'_1 &= h_{21}x_1 + h_{22}y_1 + h_{23} \\ x'_2 &= h_{11}x_2 + h_{12}y_2 + h_{13} \\ y'_2 &= h_{21}x_2 + h_{22}y_2 + h_{23} \\ x'_3 &= h_{11}x_3 + h_{12}y_3 + h_{13} \\ y'_3 &= h_{21}x_3 + h_{22}y_3 + h_{23} \\ x'_4 &= h_{11}x_4 + h_{12}y_4 + h_{13} \\ y'_4 &= h_{21}x_4 + h_{22}y_4 + h_{23} \end{bmatrix}, \quad \mathbf{X'} = \mathbf{HX}$$

This transformation matrix can then be applied to each pixel of an element's GT image, thus transforming it to match the camera's perspective. For that, each pixel needs to be multiplied with H and then normalized:

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \mathbf{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad x_{final} = \frac{x'}{w'}, \quad y_{final} = \frac{y'}{w'}$$

This procedure needs to be performed once per building level and camera to cover all possible perspectives, but the resulting matrices can be used for all elements in the IFC file. The collection of the required user input as well as the necessary calculations are performed within the open-source OpenCV python library [53].

Including the third dimension

The resulting transformed GT images were created by plotting the element geometry in a top-down perspective. Therefore, information concerning the element's height still needs to be included. However, the GT images are already usable for slab elements because their height (or thickness) is much smaller than the length and width. As a result, the inclusion of this third dimension within the GT images of slabs can be omitted at this point.

The proposed approach needs to decouple the geometry on the XY-plane from the third dimension so that the proposed perspective transformation can be applied in 2D space instead of the 3D space. This requires at least two sets of six points from user input and results in more complicated computations with higher impact of inaccuracies. Therefore, the element's height is added to the transformed GT images in a second step. For that, the vanishing point of each camera is calculated first. This point is defined as the horizon at which parallel lines appear to converge inside of a pictures [51]. Within this thesis, the computation again relies on user input in the form of two sets of two points with each one describing a line, which is vertical in reality and thus converges at the vanishing point in the image. An example of this procedure is provided in figure 4.8.

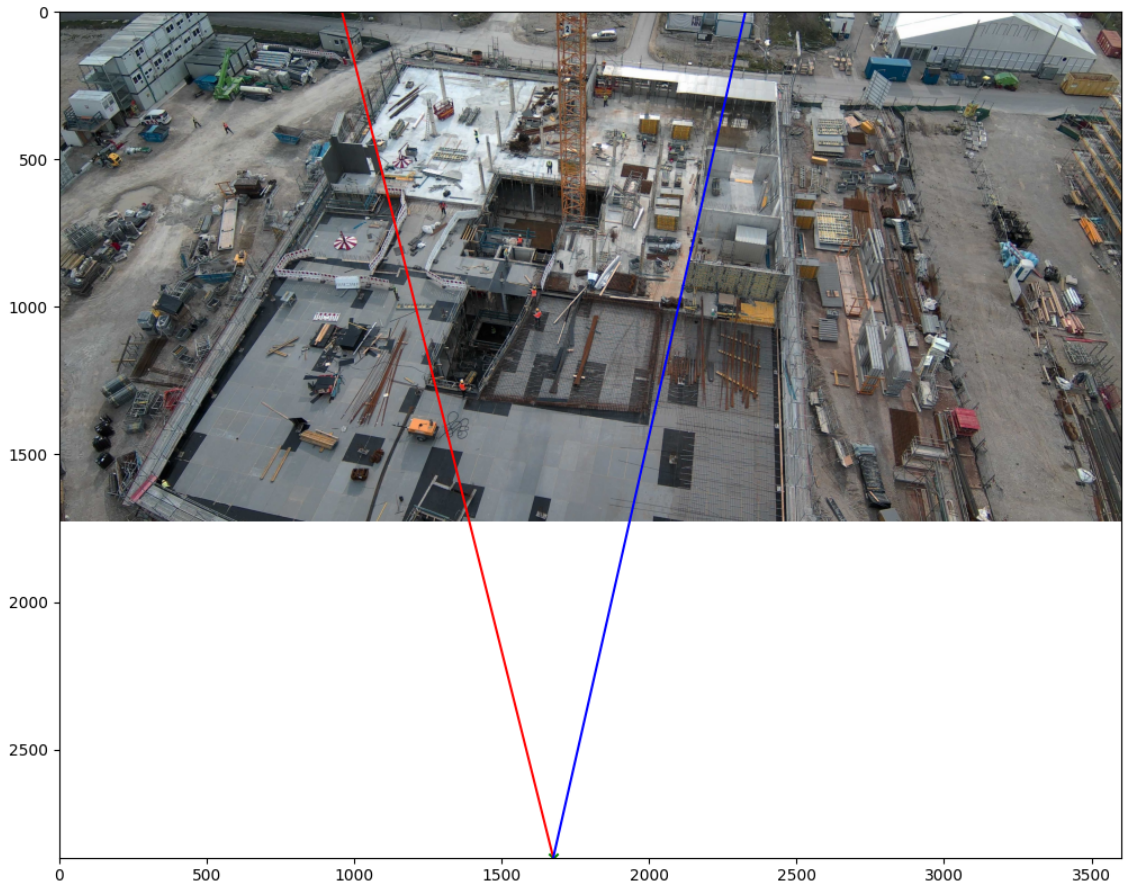


Figure 4.8: Calculation of the vanishing point

The element height is extracted from the IFC file using IFCOpenShell [49]. As these values are in metric units, a conversion step is needed to retrieve the height in pixels. For this purpose, a reference value is defined for each camera, which is then used to convert all element-height attributes. In combination with the vanishing point, this information is used to convert the element's GT image. The proposed workflow iterates over each pixel within the image and draws a line if the current pixel is coloured red and, therefore, belongs to the elements segmentation mask. The direction of this line is computed as a vector between the vanishing point and the current pixel so that the resulting line is parallel to other vertical lines within the image in reality. The length of the line is determined as the element height in pixels.

After this last step, the element GT style images are used in the progress evaluation algorithm proposed in the second part of this thesis and the resulting data is coupled to the BIM model or a digital twin using the elements GUID saved as the images filename. An example of these images before and after this lasts step is shown in figure 4.9

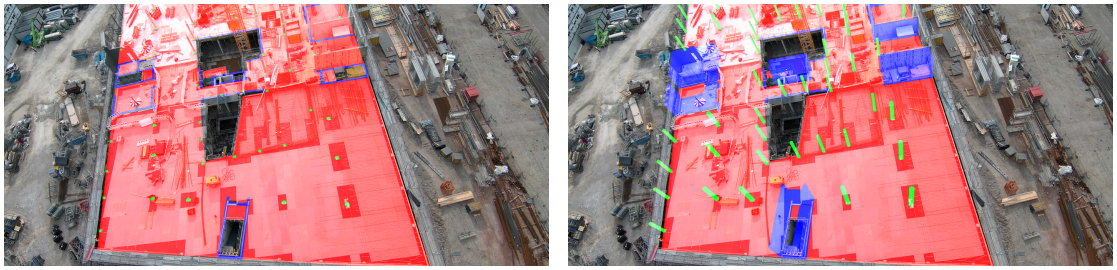


Figure 4.9: left: Transformed BIM model generated GT style image, right: additional site image as background

For all three parts of the proposed methodology in-depth case studies are performed in chapter 5 to evaluate all aspects, including the overall computation time and accuracy on a real-world construction site.

Chapter 5

Implementation

5.1 Dataset and model training

The proposed monitoring framework in this thesis aimed at automatically extracting element-specific construction progress timestamps from continuous site images. A sample picture together with the expected and actual segmentation result is provided in figure 5.1. This task started with the search for the best suited model capable of segmenting these images.

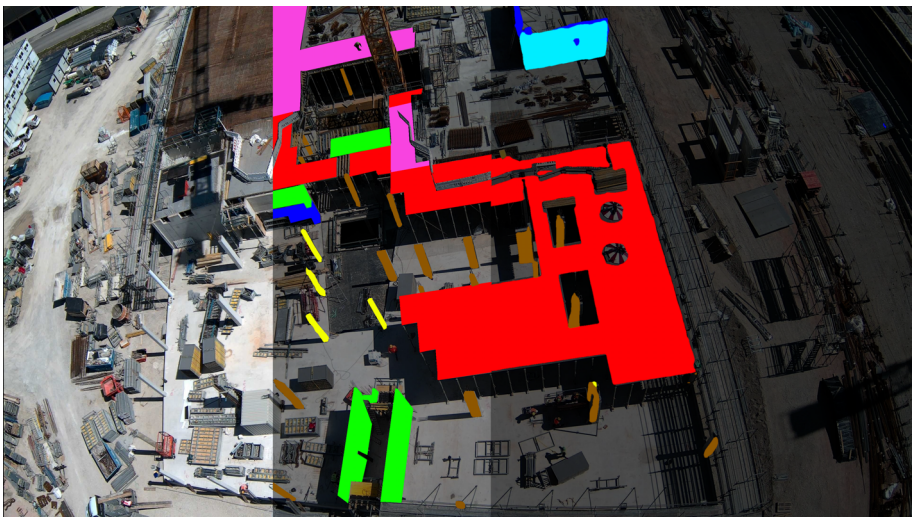


Figure 5.1: Sample site image with annotated ground truth in the middle and segmentation result on the right

However, to train said models for the task at hand and determine, which algorithms performed best, a curated dataset was needed. These typically contain a training and testing split, so that the model can be trained only on a certain subset of the data. After changing its internal parameters and thus improving its performance, it can be tested and evaluated on a different subset of unknown data. For both processes the program needed not only the image files to be segmented as an input but also a so called GT files containing the correct results. The latter was used to determine, how close the models predicted result came to the desired one. Based on that difference, metrics were calculated that can later be used to compare the overall performance to those of other models.

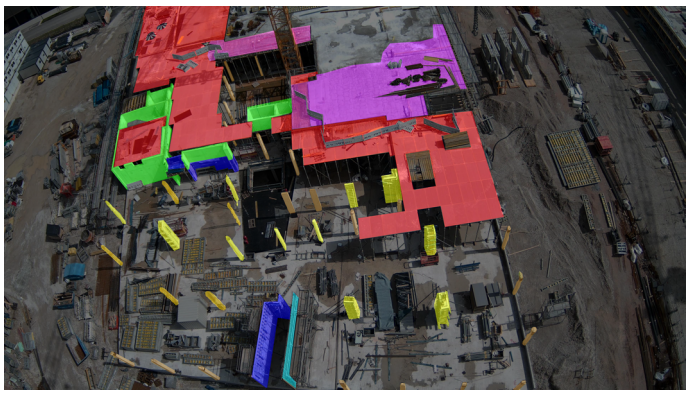
5.1.1 Dataset creation

The thesis at hand aimed at training and finding the most suitable semantic segmentation model to identify cast-in-place concrete elements in their respective construction phases. Because no other available research in this field has dealt with this exact subset of construction elements, in accordance to the mentioned papers in chapter 3, a new dataset needed to be created. It than functioned as training and testing basis for different segmentation algorithms.

The first step for creating a dataset was to gather raw data. In this case, existing data from a publication by F. Colins et al. [54] was used. In their described framework, up to three cameras were be mounted on each available crane on a construction site and RGB pictures with a 4K resolution of $3840 \cdot 2160$ pixels were captured in a continuous 30 second interval. Characteristics of these photos were, that the cameras view point didn't change apart from smaller movements during crane operations or stronger winds. Also a large portion of the construction site was viewable within one image because of the high mounting point of the cameras. This methodology has been tested on multiple projects and, therefore, large amounts of site pictures were available. For this masters thesis, images from a total of two larger and two smaller construction sites were used. Pictures from the larger projects called "Project A" and "Project B" were collected within the training and testing set. Additionally, fewer images form Projects "C" and "D" were used in an alternative second test set.

From all projects, images from two months of monitoring were initially considered. This large amount of data was then first reduced automatically by selecting 1% of all images for each monitored day in a normal distribution. In this way, more images from the middle of each day were preserved and less ones from the morning and evening hours. In a second selection step, viable images were sorted out manually so that a dataset was formed, which was comprised of a good distribution of images from diverse stages of each construction project containing different states of building elements, lighting conditions and camera angles.

To be able to use this dataset in an end-to-end trainable CNN framework with supervised learning, the "ground truth" for every image was created. This process is commonly referred to as "annotation" and can be done in two ways. Some datasets, like parts of the cityscapes dataset [29], use very coarse annotations containing only vague masks or bounding boxes for elements, whereas others contain pixel-wise precise masks, so that each pixel inside the image has a correct class label associated to it. The first approach is considerably faster than the second but yields worse results during model training which is often compensated with a larger number of annotated images. This thesis followed the second approach of precise annotations but, therefore, resulted in a overall smaller dataset considering the limited research time. For the creation of the annotation files, the open-source program called CVAT.ai was used [55]. The annotations were saved and structured in close relation to the Pascal VOC Dataset [30] as well as the ADE20k dataset [31].



Class Label	Color
column_concrete	
column_panel	
column_rebar	
slab_concrete	
slab_panel	
slab_rebar	
wall_concrete	
wall_panel	
wall_rebar	

Figure 5.2: Sample image with class labels and corresponding colors

In total, nine different classes were annotated within the images. Those consist of three states named "panel", "rebar" and "concrete" for each of the three different element types "slabs", "walls" and "columns". All classes and the corresponding colors chosen for them are listed in figure 5.2 together with an annotated sample picture. For each site image, all visible elements were annotated while following the subsequent guidelines, so that a consistent GT basis was established:

1. Small strips of walls or columns that were visible between the formwork of slab-elements were annotated as part of the “slab_panel” class.
2. Concrete slab elements were only marked as “slab_concrete” during or immediately after the concreting process. As soon as they were used as storage or working area and were highly occluded, they were no longer annotated and belonged to the “background” class.
3. Connected elements of all classes that were intermittently obscured were marked as one instance.
4. Connection-reinforcement-elements were annotated as “column_rebar”, “slab_rebar” or “wall_rebar” if they appeared dense enough in the image. Rebar elements that were only barely visible were therefore not annotated and belonged to the “background” class.
5. Formwork elements on walls and columns that were not removed after concreting were still marked as “column_panel” or “wall_panel”.
6. Stockpiled formwork was not annotated.
7. Larger elements like fall-protection or waste containers that were resting on formwork elements of panels were left out of the “slab_panel” mask (see figure 5.3).

Figure 5.3: Annotation of a slab with larger elements on top of it

Figure 5.4: Annotation of precast wall elements

8. Pre-cast wall elements were not annotated as “wall_concrete” elements, because only cast-in-place elements were of interest for this dataset. Figure 5.4 provides an example.
9. Elements heavily obscured by scaffolding were not annotated and belonged to the “background” class (see figure 5.5).

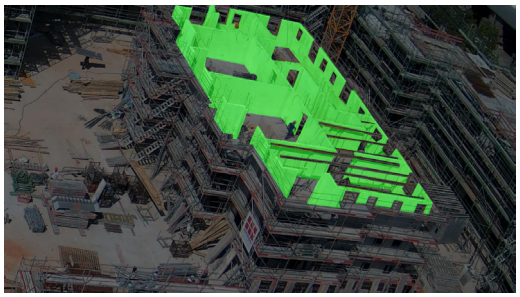


Figure 5.5: Annotation of obscured elements

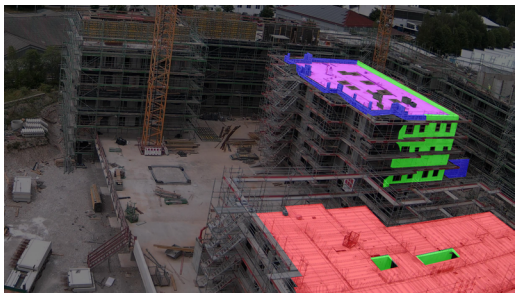


Figure 5.6: Annotation of buildings in the background

10. Only elements of buildings in the foreground were annotated. Buildings too far in the background were counted towards the “background” class, as shown in figure 5.6.
11. Oftentimes multiple vertical elements comprising columns and walls were cast together in one large wall formwork and therefore were annotated with the “wall_panel” label. However, after the formwork has been removed, slim column elements were annotated as “column_concrete” and walls as “wall_concrete”, because their load-bearing behavior and geometry still resembled that of the respective element. An example is provided in figure 5.7.

Figure 5.7: Annotation of elements, that are cast together in one formwork during their panel phase (on the left) and their concrete phase (on the right)

The last step of the dataset creation was to structure all images into different subsets. From Projects A and B a total 390 images were annotated and divided into a training subset with 350 images as well as a testing subset with 40 images which closely resembles a 1:10 split. Other established datasets often use a 1:20 split but considering the limited amount of annotated images, a smaller size for the testing set was chosen so more data was available for training purposes. To evaluate the segmentation models performance on unknown construction sites, 30 additional images from Projects C and D were annotated and formed a second alternative testing subset. The resulting distribution of annotated instances and the average mask area per class are shown in figure 5.8 separately for each subset of the dataset.

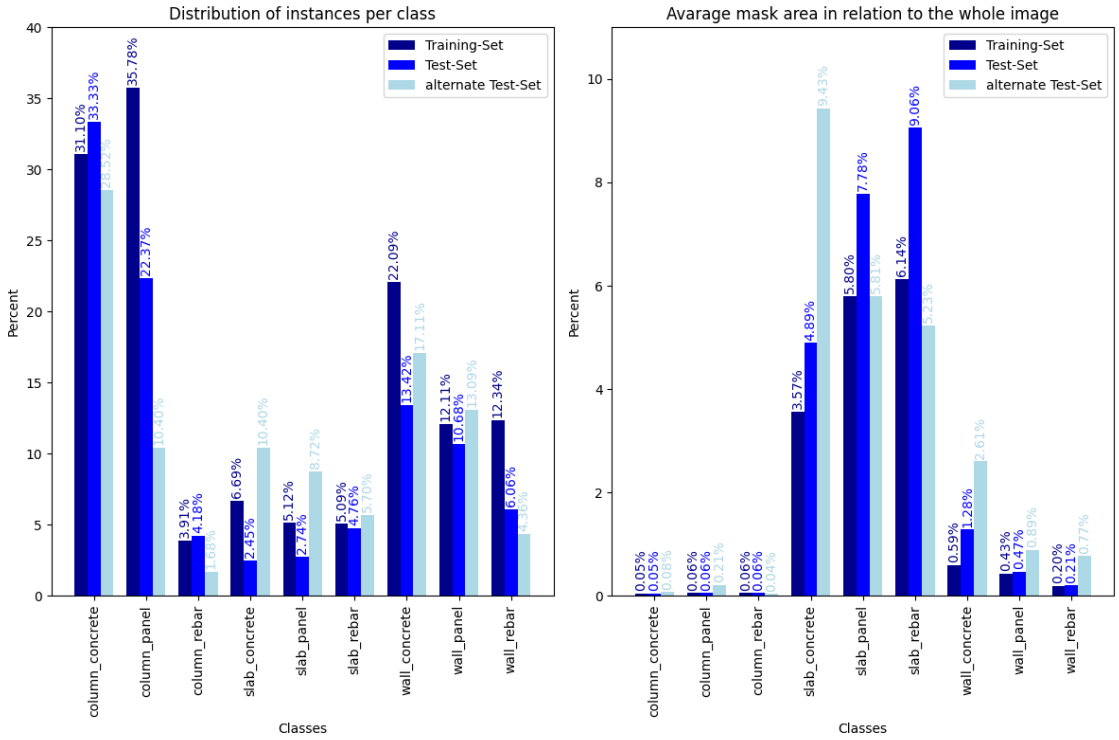


Figure 5.8: Statistics of the custom dataset

5.1.2 Training and evaluation of various semantic segmentation models

Training phase

The newly created dataset with pixel-wise annotated GT information was used to train and test various semantic segmentation models. But for this task, a pipeline was needed, which loaded the model architectures, batch-wise fed images to the models and compared the model's output against the expected result using the GT annotations. It also needed to calculate the error (or loss) after each iteration for the backpropagation algorithm to change the necessary parameters in the segmentation model. The pipeline used here was created on top of the "MMSegmentation" repository [56], which provides a customizable framework for training different segmentation models on standard datasets mentioned in section 2.3. To enable using the newly created image dataset within the "MMSegmentation" structure, new configuration files for the dataset itself as well as its interaction with different models were created.

In total, 49 models performing well on other standard datasets were trained and tested on the new dataset. Apart from models with an "Internimage", "BeiT" and "ViT" backbone, all architectures were directly accessible within the "MMSegmentation" library. For the other backbones mentioned above, additional repositories were used [57], [58].

The training procedure was performed on one Nvidia Quadro RTX 8000 GPU in a virtual docker container running on a Linux server with an AMD Ryzen Threadripper 3990X 64-Core Processor. Each model was trained over 40.000 Iterations while using the data augmentation methods "resize", "random crop", "random flip" and "photometric distortion"[56] with a crop size of 512x512 pixels. The learning rate and weight-decay parameters used for all models resembled a configuration that has already worked well for the specific algorithm and backbone combination during the evaluation of different datasets. Changes to these parameters were only made in a second training phase to finetune two very promising model architectures.

All in the evaluation included semantic segmentation models are shown in table 5.1 including their sources as well as their best scores on the "Cityscapes" [29] and "ADE20K Dataset" [31], if the respective data was available. Additionally, the learning rate and weight decay parameters are stated for each model next to the score of their best validation epoch, with validations performed in 4.000 iteration intervals.

During the training and later in the testing phase, two important metrics were evaluated for each model. As stated in section 2.2, the "mIoU" and "mAcc" metrics were used to comprise the segmentation performance over all classes into singular values. Both metrics are listed in tables 5.1, 5.2 and 5.3.

Table 5.1: Training parameters and validation results

Model		other datasets		Parameters		Validation	
Algorithm	**Backbone**	**CC [29]**	**ADE [31]**	**lr**	**decay**	**mIoU**	**mAcc**
APCNet [59]	R50-D8 [60]	78.02	42.20	0.01	0.0005	48.65	53.53
DaNet [61]	R101-D8 [60]	80.52	43.64	0.01	0.0005	86.51	92.04
	R50-D8 [60]	78.74	41.66	0.01	0.0005	86.18	92.05
DeepLabV3plus [62]	R101-D8 [60]	80.21	44.60	0.01	0.0005	44.34	47.61
	R50-D8 [60]	79.61	42.72	0.01	0.0005	46.54	53.63
	S-101-D8 [63]	79.62	46.47	0.01	0.0005	49.62	54.80
DPT [64]	ViT-Base [26]	-	46.97	0.0006	0.01	76.49	81.49
EMANet [65]	R101-D8 [60]	79.10	-	0.01	0.0005	52.04	60.06
	R50-D8 [60]	77.59	-	0.01	0.0005	48.28	55.01
ENCNET [66]	R101-D8 [60]	75.81	42.11	0.01	0.0005	52.76	62.57
	R50-D8 [60]	75.67	39.53	0.01	0.0005	51.00	58.02
FastFCN [67]	ENC [66]	79.92	40.88	0.01	0.0005	47.33	54.17
	PSP [68]	79.26	41.40	0.01	0.0005	48.27	57.63
FCN [69]	HRNetV2p-W18 [70]	77.19	36.27	0.01	0.0005	48.83	56.72
	HRNetV2p-W48 [70]	80.65	42.02	0.01	0.0005	49.75	58.03
	R101-D8 [60]	75.45	39.61	0.01	0.0005	41.27	50.93
	R50-D8 [60]	72.25	35.94	0.01	0.0005	50.12	60.02
FPN [71]	Twins-Large [72]	-	46.55	0.0001	0.0001	86.63	91.51
	Twins-Small [72]	-	43.26	0.0001	0.0001	89.41	93.72
ISANET [73]	R101-D8 [60]	79.58	43.51	0.01	0.0005	51.42	57.69
	R50-D8 [60]	78.49	41.12	0.01	0.0005	54.74	61.78
Knet [74] + DeepLab	R50-D8 [60]	-	45.06	0.0001	0.0005	43.61	49.59
Knet [74] + UperNet	Swin-Large [25]	-	52.05	0.00006	0.0005	74.32	81.26
Mask2Former [75]	InternImage-H [27]	-	62.6	0.00001	0.05	85.26	91.5
	R50-D8 [60]	80.44	47.87	0.0001	0.05	46.48	59.34
	Swin-Large [25]	83.52	52.44	0.0001	0.05	79.72	85.35
OCRNet [76]	HRNetV2p-W18[70]	77.72	37.79	0.01	0.0005	45.33	51.51
	HRNetV2p-W48[70]	80.58	43.00	0.01	0.0005	53.16	62.13
	R101-D8 [60]	80.30	-	0.01	0.0005	52.49	59.83
PIDNet [77]	PIDNet-Large [77]	80.89	-	0.01	0.0005	84.09	89.08
	PIDNet-Medium [77]	80.22	-	0.01	0.0005	82.39	88.70
	PIDNet-Small [77]	78.74	-	0.01	0.0005	86.95	91.18
PSANet [78]	R50-D8 [60]	79.31	41.67	0.01	0.0005	46.06	53.61
PSPNet [68]	R50-D8 [60]	79.59	42.48	0.01	0.0005	51.36	57.55
Segformer [79]	MIT-B0 [79]	76.54	37.41	0.00006	0.01	77.09	81.94
	MIT-B5 [79]	82.25	49.62	0.00006	0.01	85.52	91.00
Segmenter [80]	ViT-Base [26]	-	49.60	0.001	0.00	58.47	66.71
SegNeXT [81]	MSCAN-Base [81]	-	48.03	0.00006	0.01	73.00	79.50
SETR [82]	MLA [82]	77.00	47.39	0.001	0.0	50.46	57.93
UPerNet [83]	Beit-Base [84]	-	56.33	0.00003	0.05	24.62	32.36
	ConvNeXt-Base [85]	-	52.13	0.0001	0.05	48.69	52.31
	InternImage-L [27]	-	53.90	0.00002	0.05	79.84	85.08
	InternImage-T [27]	-	47.90	0.00006	0.05	77.88	83.44
	MAE [86]	-	48.13	0.0001	0.05	51.27	61.08
	R50-D8 [60]	79.39	42.05	0.01	0.0005	77.54	85.52
	Swin-Base [25]	-	50.13	0.00006	0.01	69.69	75.93
	Twins-Large [72]	-	49.65	0.00006	0.01	83.56	88.38
	Twins-Small [72]	-	46.04	0.00006	0.01	76.69	81.64
	ViT-Base [26]	-	48.13	0.00006	0.01	86.97	91.94

Testing phase

All models were tested once on the standard test set comprised of unknown images from the construction sites of Projects A and B, on which the models were trained, and once on an alternative test set consisting of images from two other construction sites. Additionally, the inference speed of each model was tested. It resembles a value for the time needed to segment one image, calculated as an average out of 25 Iterations over a batch size of 30 images taken from the test set. All resulting scores are stated in table 5.2 and figure 5.9.

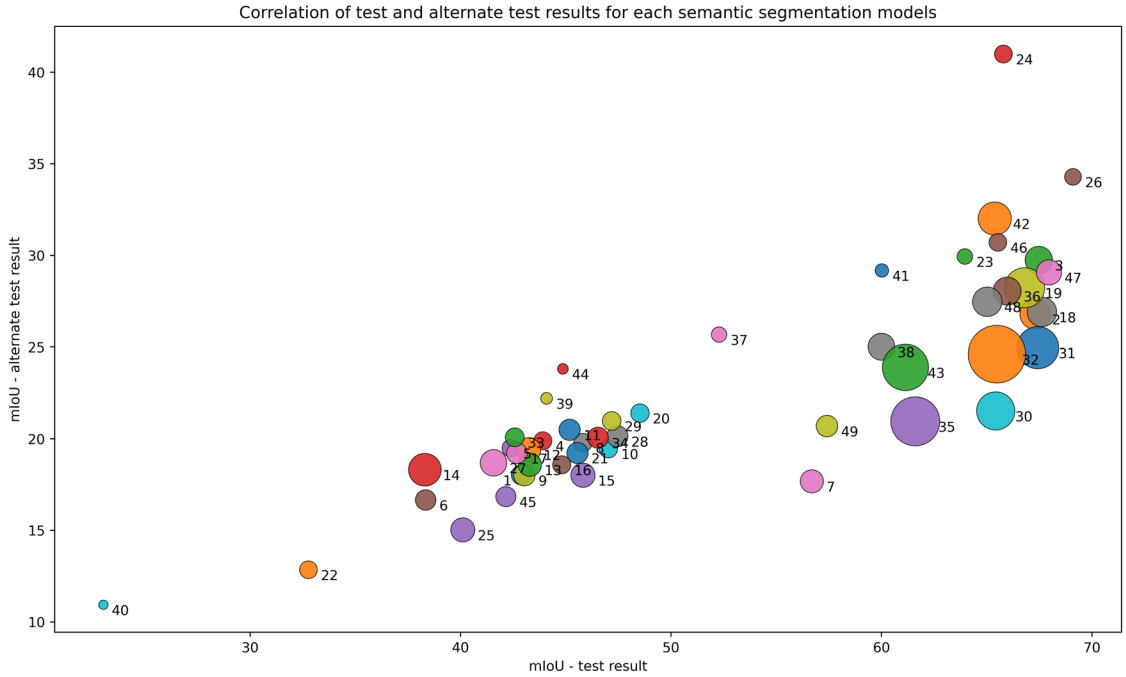


Figure 5.9: Correlation of the results on the test-set and alternate test-set; bubble size depends on the FPS; for the model names refer to table 5.2

Parameters for the learning rate and weight decay were set in accordance to each models performance on other datasets in the first training procedure. But because this can have significant influence on the models performance, for two promising models these parameters were closer studied in a second training and testing iteration. Most of the results in figure 5.9 showed a correlation of results in the standard and alternate testset. However, the combination of the Mask2Former [75] algorithm with the InternImage-H [27] backbone (Index 24) poses an exception with its very good performance on the alternate testset. Additionally, the same algorithm in combination with the Swin-Large [25] backbone was closer examined, because this model managed to achieve the overall best result in the standard test set (see also index 26 in figure 5.9).

Table 5.2: Test results

Model		Index	Test		Alternative Test		Speed
Algorithm	**Backbone**		**mIoU**	**mAcc**	**mIoU**	**mAcc**	**FPS**
APCNet [59]	R50-D8 [60]	1	42.88	48.23	18.03	21.06	0.75
DaNet [61]	R101-D8 [60]	2	67.29	75.37	26.80	32.95	2.75
	R50-D8 [60]	3	67.47	75.46	29.75	35.56	2.04
DeepLabV3plus [62]	R101-D8 [60]	4	43.91	49.47	19.89	24.81	0.58
	R50-D8 [60]	5	42.38	49.74	19.50	23.90	0.53
	S-101-D8 [63]	6	38.34	41.92	16.66	20.05	0.85
DPT [64]	ViT-Base [26]	7	56.69	63.26	17.68	20.11	1.23
EMANet [65]	R101-D8 [60]	8	45.81	54.18	19.80	27.33	0.62
	R50-D8 [60]	9	43.03	49.70	18.01	24.03	0.97
ENCNET [66]	R101-D8 [60]	10	47.02	57.10	19.46	28.44	0.62
	R50-D8 [60]	11	45.18	52.51	20.50	25.49	0.95
FastFCN [67]	ENC [66]	12	43.23	52.73	19.41	26.84	1.39
	PSP [68]	13	43.29	50.26	18.58	25.25	1.24
FCN [69]	HRNetV2p-W18 [70]	14	38.30	47.91	18.32	26.18	3.46
	HRNetV2p-W48 [70]	15	45.81	55.62	18.02	28.47	1.41
	R101-D8 [60]	16	44.80	53.24	18.58	24.84	0.59
	R50-D8 [60]	17	42.67	50.13	19.22	24.49	0.87
FPN [71]	Twins-Large [72]	18	67.62	75.18	26.91	31.16	2.55
	Twins-Small [72]	19	66.80	73.88	28.23	32.53	6.43
ISANET [73]	R101-D8 [60]	20	48.52	56.08	21.41	28.11	0.62
	R50-D8 [60]	21	45.55	52.08	19.23	22.74	0.94
Knet [74] + DeepLabV3	R50-D8 [60]	22	32.77	44.48	12.85	20.78	0.55
Knet [74] + UperNet	Swin-Large [25]	23	63.69	71.50	29.94	39.17	0.37
Mask2Former [75]	InternImage-H [27]	24	65.79	74.09	41.00	46.11	0.56
	R50-D8 [60]	25	40.10	53.18	15.03	24.81	1.39
	Swin-Large [25]	26	69.10	76.04	34.29	41.57	0.46
OCRNet [76]	HRNetV2p-W18 [70]	27	41.56	48.04	18.69	23.96	1.85
	HRNetV2p-W48 [70]	28	47.45	55.34	20.14	27.10	0.93
	R101-D8 [60]	29	47.18	56.49	20.98	28.53	0.65
PIDNet [77]	PIDNet-Large [77]	30	65.42	72.99	21.53	26.51	5.52
	PIDNet-Medium [77]	31	67.42	74.72	24.97	29.05	7.26
	PIDNet-Small [77]	32	65.48	74.20	24.62	32.17	18.68
PSANet [78]	R50-D8 [60]	33	42.57	50.31	20.09	25.12	0.66
PSPNet [68]	R50-D8 [60]	34	46.52	53.21	20.07	25.24	0.91
Segformer [79]	MIT-B0 [79]	35	61.60	68.12	20.96	24.05	11.66
	MIT-B5 [79]	36	65.97	72.81	28.05	32.22	2.18
Segmenter [80]	ViT-Base [26]	37	52.28	61.05	25.68	33.31	0.36
SegNeXT [81]	MSCAN-Base [81]	38	60.00	68.12	25.02	33.28	1.88
SETR [82]	MLA [82]	39	44.08	51.85	22.21	29.11	0.16
UPerNet [83]	Beit-Base [84]	40	23.02	31.25	10.94	13.48	0.08
	ConvNeXt-Base [85]	41	60.01	67.21	29.18	36.39	0.24
	InternImage-L [27]	42	65.38	72.04	32.01	35.51	3.63
	InternImage-T [27]	43	61.14	68.43	23.09	26.79	9.75
	MAE [86]	44	44.86	54.73	23.82	30.65	0.12
	R50-D8 [60]	45	42.15	46.50	16.85	20.33	0.81
	Swin-Base [25]	46	65.53	72.47	30.71	38.80	0.55
	Twins-Large [72]	47	67.96	75.44	29.07	34.48	1.58
	Twins-Small [72]	48	65.02	71.89	27.47	31.18	2.53
	ViT-Base [26]	49	57.41	63.87	20.70	23.44	0.99

Fine-tuning phase

Table 5.3: Training parameters and validation results

Parameters			Validation results		Test results		Alternate test results	
Schedule	**learning rate**	**decay**	**mIoU**	**mAcc**	**mIoU**	**mAcc**	**mIoU**	**mAcc**
Mask2Former [75] & Swin-Large [25]								
40.000	0.00005	0.05	81,36	87,54	69,50	77,59	32,93	42,00
40.000	0.0001	0.05	79,72	85,35	69,10	76,04	34,29	41,57
40.000	0.0002	0.05	85,52	90,28	70,35	77,87	34,52	43,11
40.000	0.0004	0.05	50,02	60,57	44,64	54,80	21,50	31,29
40.000	0.0008	0.05	20,24	32,95	19,80	33,01	11,63	21,08
Mask2Former [75] & InternImage-H [27]								
40.000	0.00001	0.05	85,26	91,50	65,79	74,09	41,00	46,11
40.000	0.00002	0.05	86,55	98,86	70,26	78,60	36,70	41,41
40.000	0.00004	0.05	87,23	98,93	70,21	79,01	39,88	44,09
40.000	0.00008	0.05	87,39	98,94	66,89	75,59	34,16	37,77
40.000	0.0001	0.05	86,60	91,78	67,53	76,02	34,22	38,07

The model architecture consisting of the Mask2Former [75] algorithm and the Swin-Large [25] backbone achieved the overall best result in the standard test set with a mIoU score of 70,35. This success can be traced back to the core element of this backbone, which is their general-purpose transformer generating hierarchical feature maps with linear computational complexity in relation to image size. This is achieved by partitioning the image into windows with a fixed number of patches inside them, resulting in linear complexity. Between consecutive self-attention layers, these window partitions are shifted so that they bridge the windows of the preceding layer. Those connections among them significantly enhance the modelling power [25]. The resulting feature maps resolution is the same as in other prominent CNN backbones like ResNet [60] making the new transformer backbone easily integratable into existing pipelines. However, despite that, the model's performance on the given dataset was by far better, with a mIoU score of 70.35 for the Swin-Large backbone [25] compared to a score of 40.10 for the ResNet backbone [60], while both combinations used the same algorithm.

The Mask2Former [75] algorithm was also coupled with an InternImage-H [27] backbone in this case study, which resulted in a slightly worse score on the standard test-set but, more importantly, in the best score on the alternative test-set. This considerably higher score could imply that, given more diverse training data, this model can evolve into a tool capable of correctly segmenting images from unknown construction sites. However, more testing is needed to generate a confident opinion on this matter.

5.2 Extraction of construction progress data

5.2.1 Implementation

In the second part of this thesis, the trained segmentation models were used to process a continuous stream of images. These stemmed from the monitoring setup described by F. Colins et al. [54] resulting in images taken every 30 seconds by multiple crane-mounted cameras on two exemplary construction sites. The reviewed sites were also the sources for all images used in the training and standard testing subset of the newly created dataset in section 5.1. For the evaluated elements in the following subsection, a time window of two months was considered, resulting in approximately 163.000 images from three cameras at the site of project A and 178.000 images from two cameras of project B. All pictures were segmented by the PidNet-S model [77] (Index 32 in table 5.2) because of its exceptionally high frames per second (FPS) value and promising mIoU score on the test set. This resulted in a very high computation speed in combination with good prediction results.

For each monitored day at Project A, site images were available from 06:00 a.m. to 9:00 p.m. and at Project B from 00:00 a.m. to 9:00 p.m. Not all of those images could be considered in the upcoming evaluations. While the segmentation models could produce usable results in varying weather conditions, they still relied on apt lighting so that images taken during nighttime were not segmented correctly. Figure 5.10 shows an image taken at roughly 06:18 a.m. and its corresponding prediction results to illustrate the problem. In reaction to this, only pictures taken between 07:00 a.m. and 07:00 p.m. were taken into account for the evaluation of construction progress timestamps.

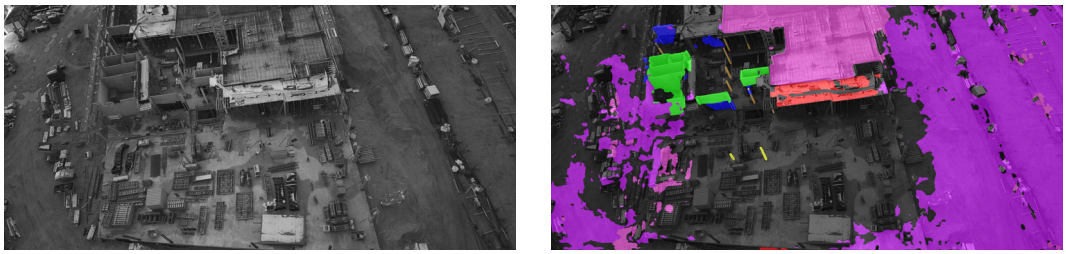


Figure 5.10: Image from Project A taken at 06:18:28 (left) and segmentation (right)

As stated in the elaboration on the proposed framework of this thesis in section 4.2, next to the trained segmentation model, some additional input was required for this part. A GT style image for each building element and camera was needed in order to evaluate the raw prediction images. One of those pictures for each upcoming example is shown at the start of each subsection with a suitable site image as background.

The segmented pictures were compared against those element-specific GT style image generating class-wise counts of detected pixels within the elements' mask area. Applying user-chosen thresholds, the program evaluated this raw data to produce start- and end-points for "panel", "rebar" and "concrete" states of the surveilled element. Within this

evaluation, the proposed framework included two averaging techniques introduced in section 4.2. These first combined raw data from singular images over longer time-frames and secondly over multiple cameras to reduce the number of necessary evaluations and improve the final result.

In the following examples the described methodology was tested on a total of six elements including two for each element type. As-built timestamps were compared against the derived timestamps to evaluate the procedure's accuracy. Because no other sources were available, the as-built timestamps were collected manually be visual inspection of the construction site image data at hand. The difference between the two recorded dates was calculated purely as a time interval not considering working hours or weekends. A negative entry corresponded to the derived timestamp being earlier than the as-built timestamp and a positive one to the derived timestamp being later than the as-built one. The error was also converted into frames corresponding to the specified averaging interval. This generated a better representation of the procedures performance in some cases. For example, when a certain timestamps was detected 15 minutes after its as-built pendant, the error as time value would be 15 minutes. However, if an averaging interval of 20 minutes was used to produce this result, the program could only output results in 20-minute intervals. As this deviation of 15 minutes then resides within one averaging interval, the program technically generated the best possible result with an error of zero frames. At the end of each example, the absolute values of errors for all timestamps were summed up to get an overall resulting accuracy.

For all six elements, the proposed averaging techniques were tested. Therefore, the whole evaluation process was performed once for each camera separately and once using the combined data from all cameras. For each scenario, the lower threshold for starting points was varied between 0% and 50%, and the threshold for endpoint was changed between 50% and 100%, both with a step size of 5%. As shown in figure 5.11, this resulted in $11 \cdot 11 = 121$ possible combinations. All of them were tested for each averaging interval with its duration changed from 0 to 30 minutes in one-minute intervals. A 0 minute setting corresponded to not using this technique at all. In total, varying these three parameters resulted in $11 \cdot 11 \cdot 31 = 3751$ possible combinations. For each one, the total error over all evaluated timestamps was recorded. However, in the following tables and plots only the best-scoring combination is shown.

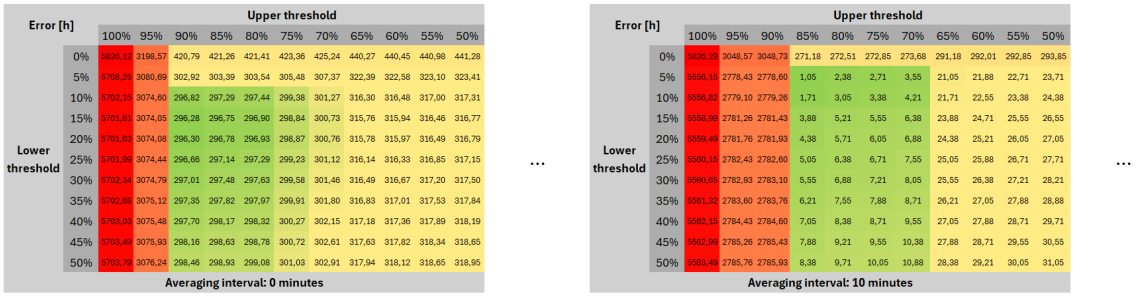


Figure 5.11: Visualization of tested threshold values and averaging intervals

5.2.2 Examples

Slab - Project A

Table 5.4: As-built timestamps

Formwork	Start	16.05.2023 11:18:02
	End	17.05.2023 11:43:32
Rebar	Start	17.05.2023 16:05:25
	End	23.05.2023 09:10:38
Concrete	Start	23.05.2023 09:10:38
	End	23.05.2023 15:23:16

Table 5.5: Evaluation for the slab of Project A

		Camera 1	Camera 2	Camera 3	All cameras
Parameters					
Averaging Interval		1min	20min	11min	10min
Upper threshold		90%	85%	85%	85%
Lower threshold		5%	5%	5%	5%
Results					
Formwork start		16.05. 11:25:00	16.05. 11:20:00	16.05. 11:22:00	16.05. 11:20:00
Error	time	6min 58sec	1min 58sec	3min 58sec	1min 58sec
	frames	6 frames	0 frames	0 frames	0 frames
Formwork end		17.05. 11:46:00	17.05. 11:40:00	17.05. 12:00:00	17.05. 11:40:00
Error	time	2min 28sec	- 3min 32sec	16min 26sec	- 3min 32sec
	frames	2 frames	0 frames	1 frame	0 frames
Rebar start		17.05. 16:14:00	17.05. 16:00:00	17.05. 16:11:00	17.05. 16:10:00
Error	time	8min 35sec	- 5min 25sec	5min 35sec	4min 35sec
	frames	8 frames	0 frames	0 frames	0 frames
Concrete start		23.05. 09:33:00	23.05. 09:40:00	23.05. 09:33:00	23.05. 09:40:00
Error	time	22min 22sec	29min 22sec	22min 22sec	29min 22sec
	frames	22 frames	1 frame	2 frames	2 frames
Concrete end		23.05. 15:05:00	23.05. 15:00:00	23.05. 15:00:00	23.05. 15:00:00
Error	time	- 18min 16sec	- 23min 16sec	- 23min 16sec	- 23min 16sec
	frames	18 frames	1 frame	2 frames	2 frames
Overall Error					
time		58min 39sec	1h 03min 33sec	1h 11min 37sec	1h 02min 43sec
frames		56 frames	2 frames	5 frames	4 frames

From the evaluation of this first slab element, a problem came to light that persisted for all other elements alike. A timestamp for completing rebar-works could only be detected by the segmentation models as the point at which the visible element area was fully covered with rebar elements. This is the case as soon as only a single layer of rebar has been constructed. However, four layers are usually necessary, consisting of two directional layers for each side of the element. As this is the only useful timestamp for the endpoint of rebar works and the stated problem within the context of a segmentation-based approach

could not be solved within this framework, the timestamp "rebar end" was not further considered in this and the upcoming examples.

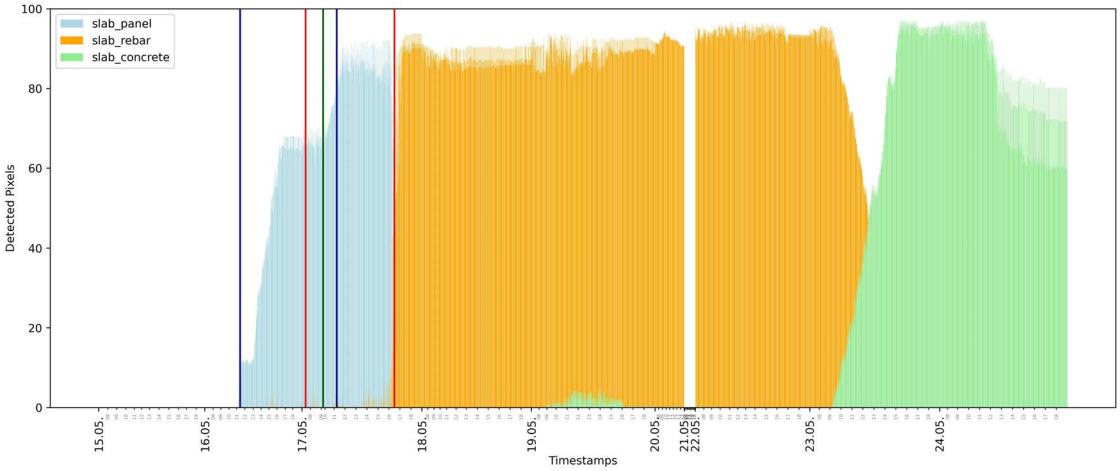


Figure 5.12: Results of all cameras without averaging

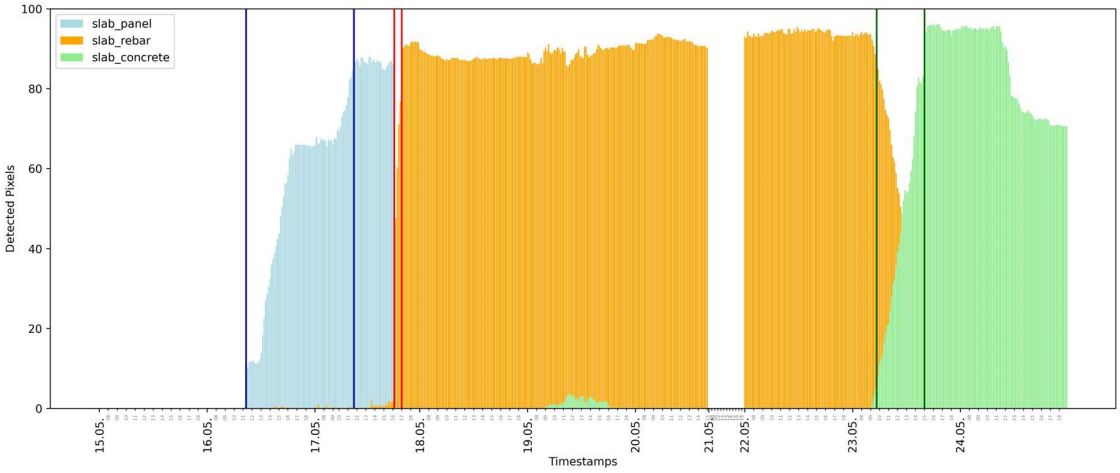


Figure 5.13: Results of all cameras with a averaging interval of 10min

When comparing the detected timestamps based on the evaluation of the combined data of all cameras using an averaging interval of 10min (figure 5.12) and without using an averaging interval at all (figure 5.13), the start- and endpoints for concrete work and the endpoint for rebar works were detected differently. The latter stemmed from minor occlusions of the panel elements due to them being used as storage space for rebar elements. The segmented rebar pixels in few frames could not be compensated without averaging and resulted in the detection of the end-timestamp for rebar works. The problem concerning the concrete progress estimation could be traced back to one singular image, which got corrupted sometime between the generation at the construction site and the processing steps for this thesis. The image in question is shown on the left in figure 5.14 with its corresponding prediction on the right. Due to the missing bottom half, the segmentation model categorized all its pixels as "slab_concrete", which resulted in the lower and upper threshold for concreting works being reached at once. This and the other

problem mentioned first could be compensated using a reasonably high averaging interval, as the plot in figure 5.13 proofs.

Faulty images like this could also be sorted out with additional checks performed before the segmentation process. However, this example shows, that the proposed framework for the evaluation of construction progress can handle such errors, when choosing a sufficiently high value for the averaging functionality.

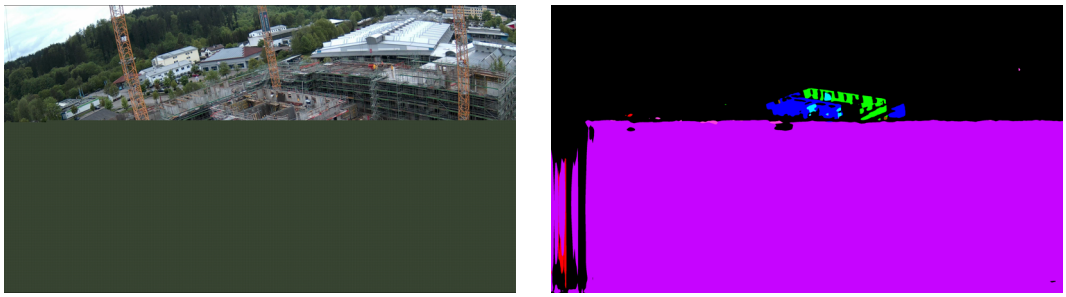


Figure 5.14: original (left) and segmented picture (right)

Slab - Project B

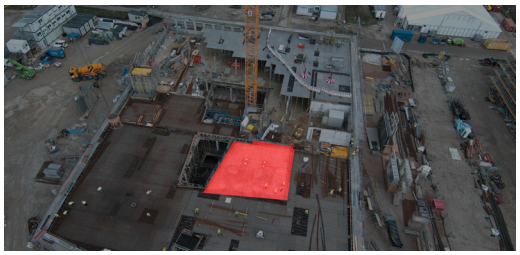


Table 5.6: As-built timestamps

Formwork	Start	prior to recording
	End	26.04.2022 12:18:53
Rebar	Start	27.04.2022 07:34:26
	End	28.04.2022 09:57:57
Concrete	Start	04.05.2022 12:23:20
	End	04.05.2022 18:30:26

The panel phase for this element started before the considered timeframe of this case study. Therefore, if the program detected this timestamp as the first entry in the surveilled window, the result was classified as correct. Furthermore, the endpoint of panel and concrete works in this example perfectly illustrated the importance of selecting adequate thresholds for the detection of timestamps. Right after the concreting process, the slab area was at first covered to aid the curing process of the concrete and later used as storage space for rebar elements rending the corresponding timestamp very hard to detect correctly. In figure 5.15, the resulting plot of the first camera with an averaging interval of 24 minutes and thresholds of 5% and 55% is shown along with the combined result of all cameras averaged over 10min with 5% and 90% thresholds in figure 5.16. The low upper threshold of 55% in evaluating the first camera's data enabled the detection of the endpoint of concreting works. However, it produced a wrong timestamp for the endpoint of the panel state. Contrary to that, the high threshold of 90% for processing the combined data generated a better estimation for the panel phase but did not allow the endpoint of the concreting phase to be picked up. Generally, a moderate upper threshold lead to more usable results because for most cases, generating an incorrect estimation for a relevant

event was less destructive to superior processes of progress monitoring than missing a notable milestone entirely.

Table 5.7: Evaluation for the slab of Project B

		Camera 1	Camera 2	All cameras
Parameters				
Averaging Interval		24min	5min	10min
Upper threshold		55%	90%	90%
Lower threshold		5%	5%	5%
Results				
Formwork start		01.04. 07:00:00	01.04. 07:00:00	01.04. 07:00:00
Error	time	0sec	0sec	0sec
	frames	0 frames	0 frames	0 frames
Formwork end		21.04. 15:48:00	26.04. 12:00:00	26.04. 12:00:00
Error	time	- 4d 20h 30min 53sec	- 18min 53sec	- 18min 53sec
	frames	291 frames	3 frames	1 frames
Rebar start		27.04. 09:48:00	27.04. 09:45:00	27.04. 09:50:00
Error	time	2h 13min 34sec	2h 10min 34sec	2h 15min 34sec
	frames	5 frames	26 frames	13 frames
Concrete start		04.05. 14:00:00	04.05. 14:00:00	04.05. 14:00:00
Error	time	1h 36min 40sec	1h 36min 40sec	1h 36min 40sec
	frames	4 frames	19 frames	9 frames
Concrete end		04.05. 18:24:00	-	-
Error	time	- 6min 26sec	-	-
	frames	0 frames	-	-
Overall Error				
time		5d 0h 27min 33sec	4h 06min 07sec *	4h 11min 07sec *
frames		300 frames	48 frames *	23 frames *

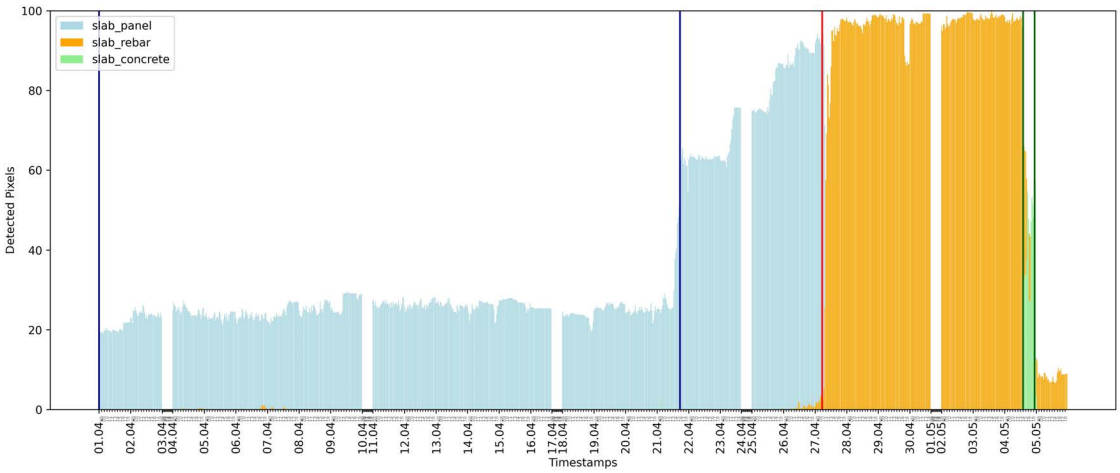


Figure 5.15: Results of camera 1

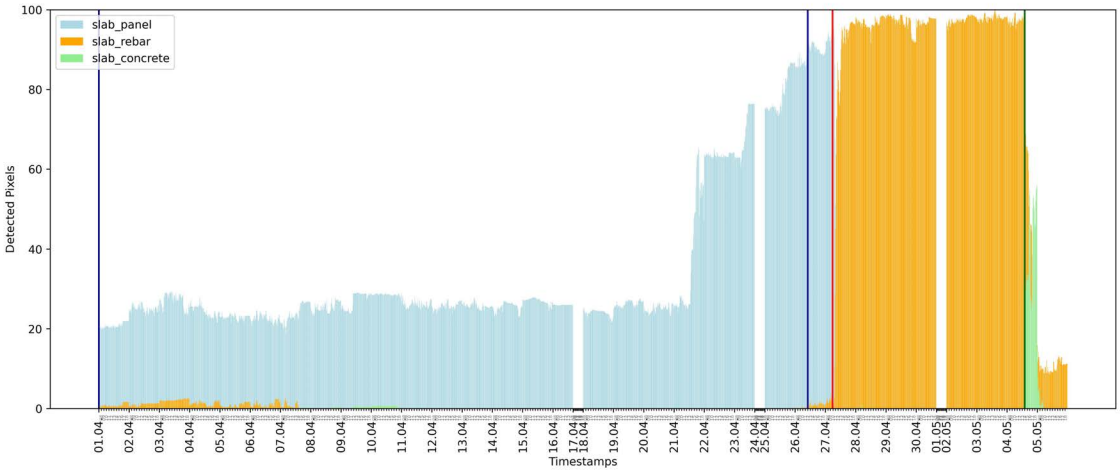


Figure 5.16: Results of all cameras

Wall - Project A

Table 5.8: As-built timestamps

Rebar	Start	24.05.2023 12:16:12
	End	24.05.2023 13:24:28
Formwork	Start	26.05.2023 09:09:09
	End	26.05.2023 13:50:23
Concrete	Start	27.05.2023 08:28:12
	End	27.05.2023 08:33:15

While the concreting process was easily detectable for most slab elements, it was impossible for wall or column elements within this approach, because for those vertical elements, the concrete itself was only visible in a small opening at the top of the formwork. Full visibility and therefore a confident detection within the segmentation-based approach only was possible after the removal of all formwork elements. This discrepancy became very apparent, when looking into other percentage-based monitoring approaches like A. Pal [37] proposed. While this thesis does not aim to include such a functionality, the examples provided in figure 5.17 illustrated its connection to the stated problem. For slab elements or horizontal elements in general, a gradual change between the states of rebar and concrete was visible inside the resulting plots. The left image in figure 5.17 shows an excerpt from the resulting plot of a slab element from the day of its concreting to illustrate this. It clearly depicts a procedural transition from pixels being labeled as slab_panel to slab_concrete. Contrary to that, in the wall or column examples, only a sudden state change was recorded, shown in the right image of figure 5.17 with a zoomed-in section of the resulting plot for the following wall element. Therefore, because the detection of concrete pixels only happened when the formwork was removed several days after the actual concreting, this action was used as the target timestamp in the evaluation of the subsequent wall and column examples.

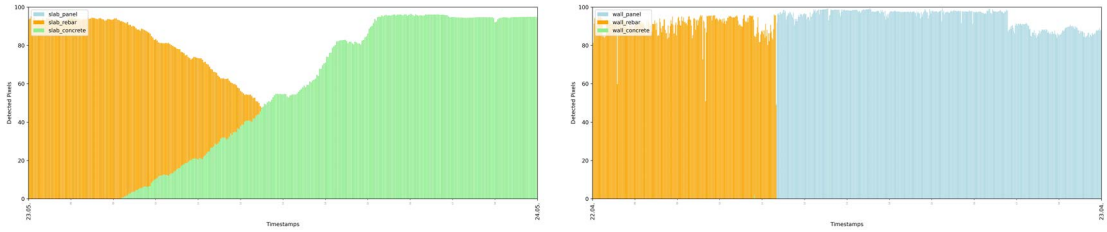


Figure 5.17: Comparison of the behavior when changing between two different states

Table 5.9: Evaluation for the wall of Project A

		Camera 1	Camera 2	Camera 3	All cameras
Parameters					
Averaging Interval		1min	13min	15min	7min
Upper threshold		55%	80%	65%	50%
Lower threshold		5%	5%	5%	5%
Results					
Rebar start		24.05. 12:16:00	24.05. 12:13:00	24.05. 12:15:00	24.05. 12:14:00
Error	time	- 12sec	- 3min 12sec	- 1min 12sec	- 2min 12sec
	frames	0 frames	0 frames	0 frames	0 frames
Rebar end		24.05. 12:36:00	24.05. 13:00:00	24.05. 12:45:00	24.05. 12:35:00
Error	time	- 48min 28sec	- 24min 28sec	- 39min 28sec	- 49min 28sec
	frames	48 frames	1 frame	2 frames	7 frames
Formwork start		26.05. 09:25:00	26.05. 09:26:00	26.05. 09:30:00	26.05. 09:28:00
Error	time	15min 51sec	16min 51sec	20min 51sec	18min 51sec
	frames	15 frames	1 frame	1 frame	2 frames
Formwork end		26.05. 13:46:00	26.05. 13:52:00	26.05. 13:45:00	26.05. 13:49:00
Error	time	- 4min 23sec	1min 37sec	- 5min 23sec	- 1min 23sec
	frames	4 frames	0 frames	0 frames	0 frames
Concrete start		27.05. 08:28:00	(29.05. 07:00:00)	(29.05. 07:00:00)	27.05. 08:28:00
Error	time	- 12sec	-	-	- 12sec
	frames	0 frames	-	-	0 frames
Concrete end		27.05. 08:37:00	(29.05. 07:00:00)	(29.05. 07:00:00)	27.05. 08:35:00
Error	time	3min 45sec	-	-	1min 45sec
	frames	3 frames	-	-	0 frames
Overall Error					
time		1h 12min 51sec	46min 08sec *	1h 06min 54sec *	1h 13min 51sec
frames		70 frames	2 frames *	3 frames *	9 frames

Because there was no data from the 27th for camera 2 and camera 3, the correct timestamps of the concreting processes could not be detected while only using data from one of those cameras as the example in figure 5.19 shows. Including data from the first camera remedied this problem resulting in the correct detection of this timestamp when using data from this camera alone or from all cameras combined, which is depicted in figure 5.18.

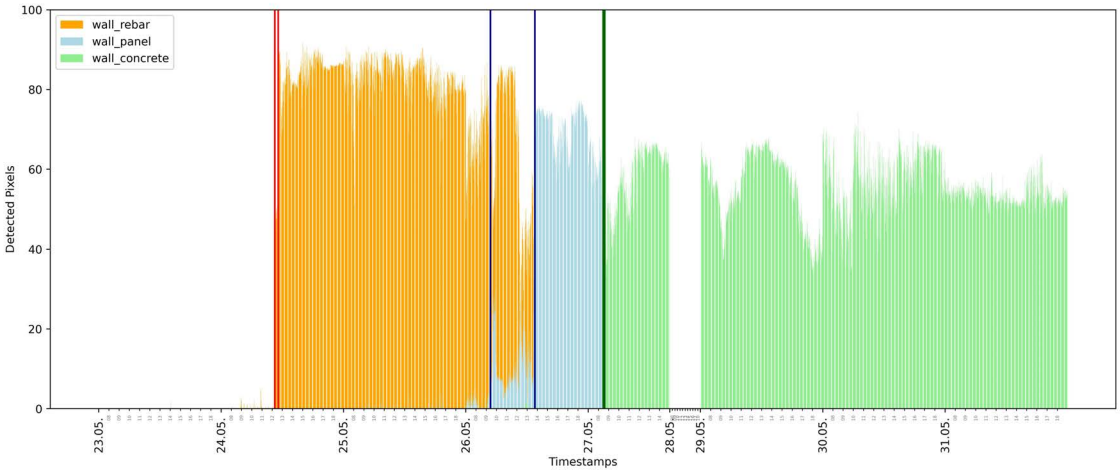


Figure 5.18: results of camera 1

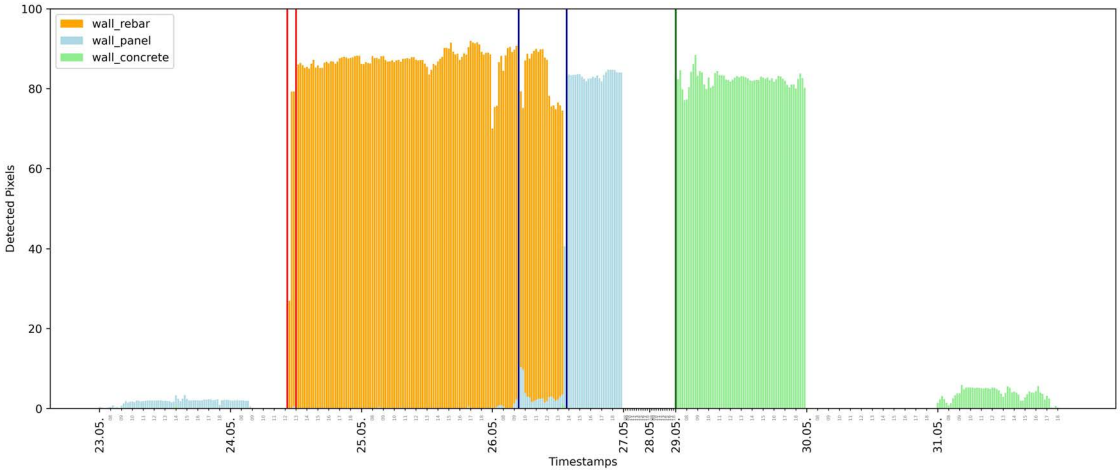


Figure 5.19: results of camera 2

Wall - Project B

Table 5.10: As-built timestamps

Formwork first side	Start	11.04.2022 16:05:37
	End	12.04.2022 08:50:10
Rebar	Start	12.04.2022 10:57:06
	End	13.04.2022 16:57:33
Formwork second side		22.04.2022 11:21:47
Concrete		26.04.2022 08:17:15

Table 5.11: Evaluation for the wall of Project B

		Camera 1	Camera 2	All cameras
Parameters				
Averaging Interval		2min	1min	1min
Upper threshold		85%	90%	85%
Lower threshold		10%	10%	10%
Results				
Formwork 1st side start		11.04. 17:00:00	11.04. 15:31:00	16.05. 16:53:00
Error	time	54min 23sec	- 34min 37sec	47min 23sec
	frames	27 frames	34 frames	47 frames
Formwork 1st side end		12.04. 09:00:00	12.04. 09:00:00	12.04. 09:48:00
Error	time	9min 50sec	9min 50sec	57min 50sec
	frames	4 frames	9 frames	57 frames
Rebar start		12.04. 10:56:00	12.04. 10:57:00	12.04. 10:57:00
Error	time	- 1min 06sec	- 6sec	- 6sec
	frames	0 frames	0 frames	0 frames
Formwork 2st side		22.04. 11:22:00	22.04. 11:21:00	22.04. 11:21:00
Error	time	13sec	- 47sec	- 47sec
	frames	0 frames	0 frames	0 frames
Concrete		26.04. 08:24:00	26.04. 08:32:00	26.04. 08:32:00
Error	time	6min 45sec	14min 45sec	14min 45sec
	frames	3 frames	14 frames	14 frames
Overall Error				
time		1h 12min 17sec	1h 00min 05sec	2h 00min 57sec
frames		34 frames	57 frames	118 frames

This second wall example did not only perform very well, but also highlighted the possibility of introducing more then one phase of a certain construction stage including a start and end point. In this case, two panel phases were evaluated because the rear formwork elements were placed and detected before the installation of rebar elements. All remaining formwork in front of the element was positioned later shortly before concreting. Concerning the evaluation algorithm, this change was accomplished by introducing a rule, that allows the program to search for a third and possibly fourth timestamps of one class once at least one timestamp of another class has been found. In the example at hand, the evaluation of the timestamp for the second formwork installation was possible from the point in time on, at which the starting point of the rebar works has been detected.

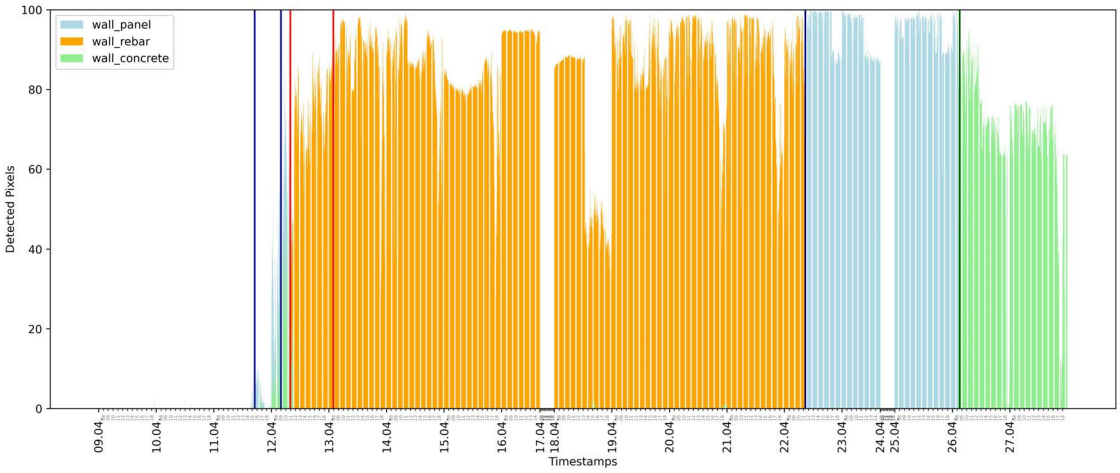


Figure 5.20: Results of camera 2

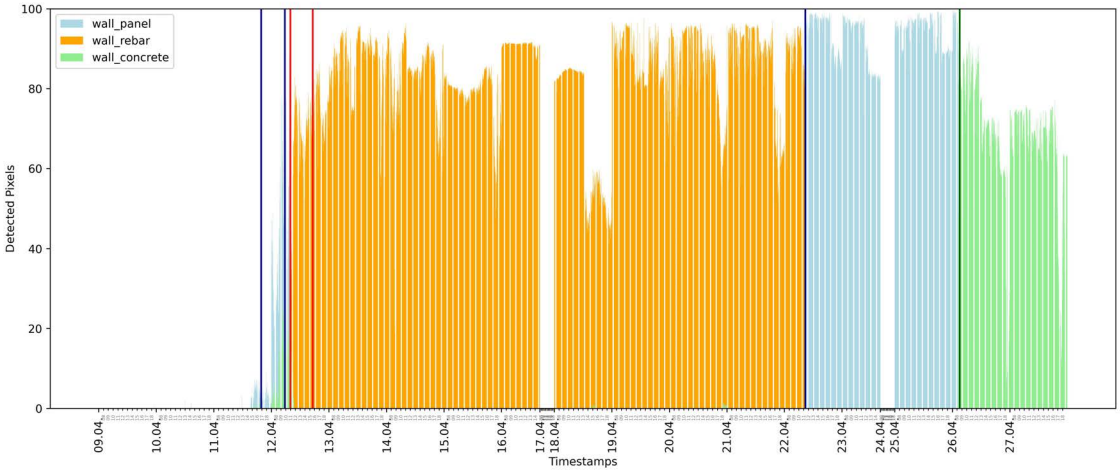


Figure 5.21: Results of all cameras

Column - Project A

Table 5.12: As-built timestamps

Rebar	Start	24.05.2023 12:25:17
	End	24.05.2023 15:00:00
Formwork	Start	25.05.2023 08:10:28
	End	25.05.2023 14:40:46
Concrete		26.05.2023 08:31:45

As described in the wall example of Project A, the correct timestamp for the end point of concreting works for this and the next column could not be derived correctly due to the segmentation approach only detecting concrete-labeled pixels after the removing of formwork. Therefore, again this action was used as correct timestamp for the upcoming evaluations.

Because column elements in general have a smaller volume compared to walls or slabs, they also contain less pixels in their GT mask used in the progress estimation algorithm. Because of slight variances in the camera angle due to the swaying of the crane while the GT images does not change, more fluctuations in the resulting data could be observed. Looking back at the first example of a slab element, there was not much of a difference concerning inconsistencies in the data between using one camera as shown in figure 5.12 and using all cameras combined, which is depicted in figure 5.13. Contrary to that, figures 5.22 and 5.23 demonstrate notable improvements when switching from using one camera to multiple ones.

Table 5.13: Evaluation for the column of Project A

		Camera 1	**Camera 2**	**Camera 3**	**All cameras**
Parameters					
Averaging Interval		2min	5min	1min	2min
Upper threshold		65%	50%	75%	60%
Lower threshold		10%	5%	5%	5%
Results					
Rebar start		24.05. 12:26:00	24.05. 13:45:00	24.05. 12:25:00	24.05. 12:26:00
Error	time	43sec	1h 19min 43sec	- 17sec	43sec
	frames	0 frames	15 frames	0 frames	0 frames
Rebar end		24.05. 15:04:00	24.05. 15:35:00	24.05. 14:56:00	24.05. 15:04:00
Error	time	4min 00sec	35min 00sec	- 4min 00sec	4min 00sec
	frames	2 frame	7 frames	4 frames	2 frame
Formwork start		25.05. 08:10:00	25.05. 08:10:00	25.05. 08:10:00	25.05. 08:10:00
Error	time	- 28sec	- 28sec	- 28sec	- 28sec
	frames	0 frames	0 frames	0 frames	0 frames
Formwork end		25.05. 13:06:00	25.05. 14:35:00	25.05. 14:39:00	25.05. 14:38:00
Error	time	- 1h 34min 46sec	- 5min 46sec	- 1min 46sec	- 2min 46sec
	frames	47 frames	1 frame	1 frame	1 frame
Concrete		26.05. 08:30:00	26.05. 08:30:00	26.05. 08:31:00	26.05. 08:30:00
Error	time	- 1min 45sec	- 1min 45sec	- 45sec	- 1min 45sec
	frames	0 frames	0 frames	0 frames	0 frames
Overall Error					
time		1h 41min 42sec	2h 02min 42sec	7min 16sec	9min 42sec
frames		49 frames	23 frames	5 frames	3 frames

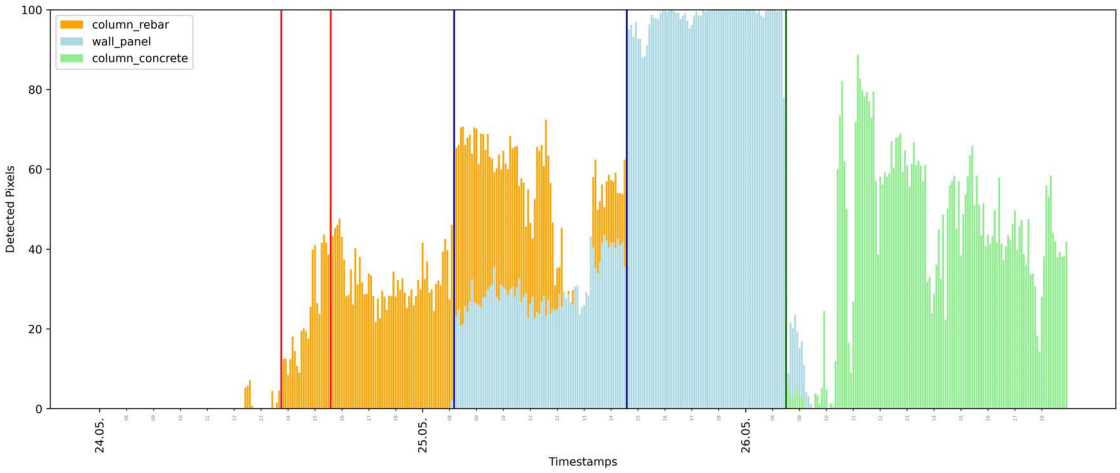


Figure 5.22: Results of camera 2

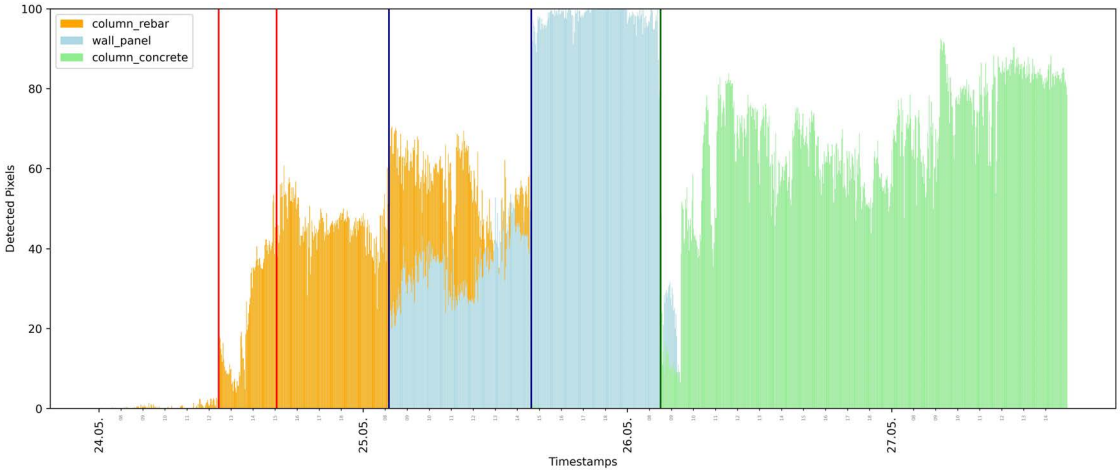


Figure 5.23: results of all cameras

Column - Project B

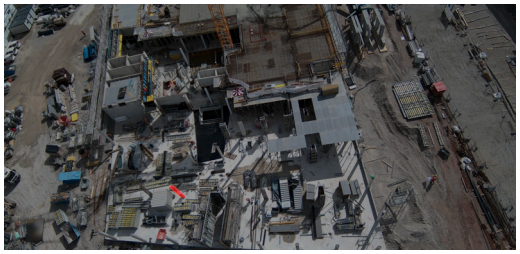


Table 5.14: As-built timestamps

Rebar	Start	06.04.2022 14:43:17
	End	06.04.2022 15:01:29
Formwork	Start	12.04.2022 12:24.02
	End	12.04.2022 13:27:46
Concrete		13.04.2022 07:24:41

In contrast to the column example from Project A, which was mainly viewed by all cameras from the side, this column from Project B was monitored nearly from a top-down perspective. This reduced the number of detectable pixels for this element and, therefore, increased the influence of inaccuracies due to movements of the crane-mounted cameras. Additionally, the first column was only obstructed by stationary scaffolding in an otherwise

uncluttered environment, while the second one was located in the middle of a crowed storage space oftentimes occluded by moving parts on the construction site. The result plots in figures 5.24 and 5.25 reflect this by showing high inconsistencies in the detected pixels and corresponding timestamps. For the second camera, these effects were of less influence because it was mounted lower than the first camera and thus had a slightly higher number of detectable pixels in the elements GT mask. The overall error in the evaluation process showed the notably better performance when using only the second camera. Also only a slight decline in accuracy occurred, when combining the data from both cameras, which proofed the importance of the implemented averaging techniques.

Table 5.15: Evaluation for the column of Project B

		Camera 1	**Camera 2**	**All cameras**
Parameters				
Averaging Interval		0min	2min	2min
Upper threshold		60%	55%	50%
Lower threshold		45%	40%	40%
Results				
Rebar start		06.04. 15:11:36	06.04. 14:46:00	06.04. 15:00:00
Error	time	28min 19sec	2min 43sec	16min 43sec
	frames	56 frames	1 frame	8 frames
Rebar end		06.04. 16:41:37	06.04. 15:00:00	06.04. 15:06:00
Error	time	1h 40min 08sec	- 1min 29sec	4min 31sec
	frames	200 frames	0 frames	2 frames
Formwork start		09.04. 13:38:01	12.04. 13:28:00	12.04. 13:28:00
Error	time	- 2d 22h 46min 01sec	1h 03min 58sec	1h 03min 58sec
	frames	8.492 frames	31 frames	31 frames
Formwork end		12.04. 13:29:17	12.04. 13:28:00	12.04. 13:28:00
Error	time	1min 31sec	14sec	14sec
	frames	3 frames	0 frames	0 frames
Concrete		13.04. 07:32:16	13.04. 07:24:00	13.04. 07:32:00
Error	time	7min 35sec	- 41sec	7min 19sec
	frames	15 frames	0 frames	3 frames
Overall Error				
time		3d 01h 03min 34sec	1h 09min 05sec	1h 32min 45sec
frames		8.766 frames	32 frames	44 frames

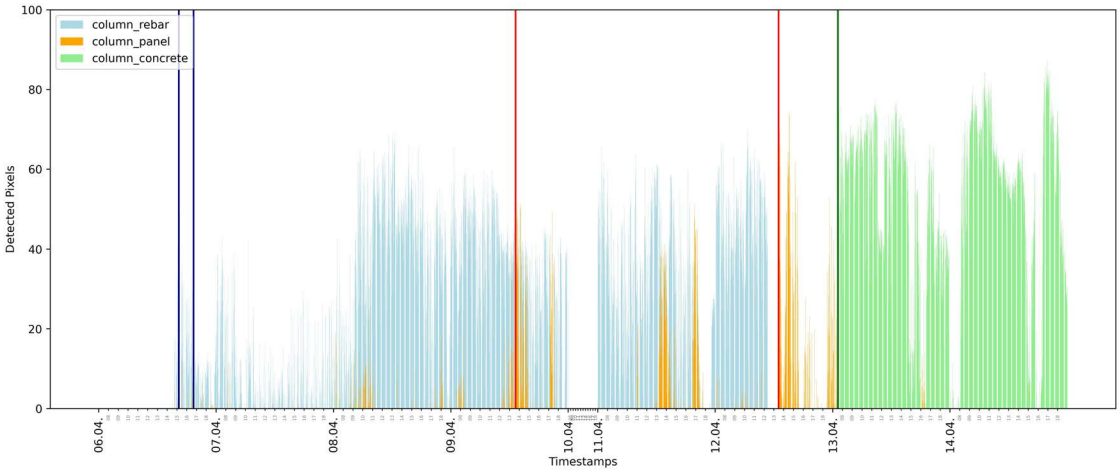


Figure 5.24: Results of camera 1

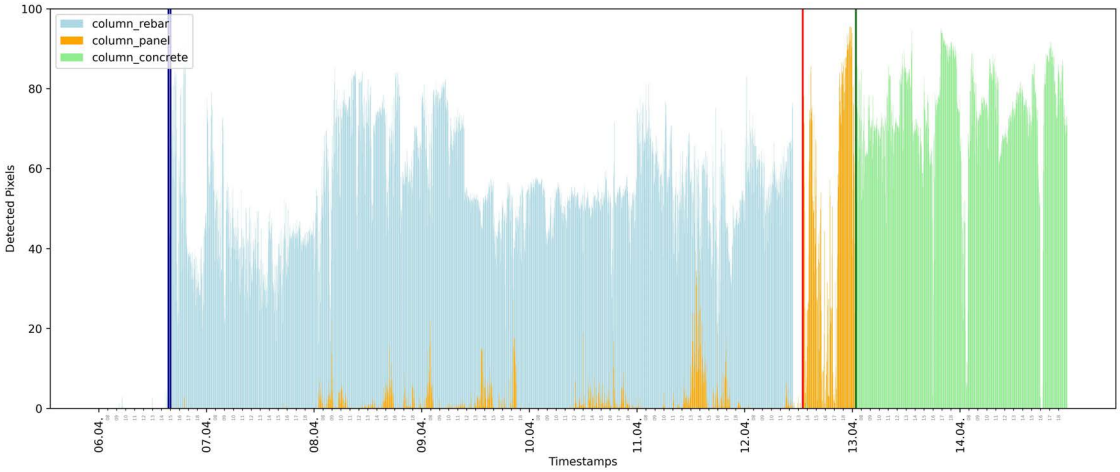


Figure 5.25: Results of camera 2

Summary

Table 5.16: Summary of the case study examples

		Camera	Averaging Interval	Upper threshold	Lower threshold
Slab	Project A	Camera 1	1min	90%	5%
		Camera 2	20min	85%	5%
		Camera 3	11min	85%	5%
		All cameras	10min	85%	5%
	Project B	Camera 1	24min	55%	5%
		Camera 2	5min	90%	5%
		All cameras	10min	90%	5%
Average			12min	85%	5%
Wall	Project A	Camera 1	1min	55%	5%
		Camera 2	13min	80%	5%
		Camera 3	15min	65%	5%
		All cameras	7min	50%	5%
	Project B	Camera 1	2min	85%	10%
		Camera 2	1min	90%	10%
		All cameras	1min	85%	10%
Average			6min	75%	5%
Column	Project A	Camera 1	2min	65%	10%
		Camera 2	5min	50%	5%
		Camera 3	1min	75%	5%
		All cameras	2min	60%	5%
	Project B	Camera 1	0min	60%	45%
		Camera 2	2min	55%	40%
		All cameras	2min	50%	40%
Average			2min	60%	20%
Overall average			7min	75%	10%

The examples showed, that in general a smaller averaging interval of 5 to 10 minutes generated the best results, because errors like the one shown in figure 5.14 could be eradicated but a certain level of accuracy was maintained. The applied upper threshold varied highly between the different studied elements. Considering, that a high number of them were at least partially occluded for example by scaffolding as it was the case with the column and wall example from project A, a conservative threshold seemed more practical. A possible example is the median of all examples which amounts to 75 percent. With the last column sample as an exception, all other studied elements worked very well with a small lower threshold of 5-10 percent. The parameters shown in the last row of table 5.16 pose a likely combination to work on arbitrary construction elements within this scheme. An other test of this hypothesis is conducted in section 6.

5.3 Coupling of the extracted data with a BIM model

5.3.1 Generation of ground truth images

The proposed frameworks ability to evaluate element-specific construction progress from segmented site images has been successfully tested in the last section. Therefore, according to the method elaborated in chapter 4, only the intended functionality to couple the monitoring results with a BIM model remained to be tested. Hereby, a BIM model and images of the construction site for Project B were used. As this project also was the source of a large part of images contained in the newly annotated dataset in section 5.1, good segmentation results of other unseen images from this site could be expected from the trained models.

Following the workflow of this part of the procedure proposed in chapter 4, the BIM model was opened with the open-source Python library IFCOpenShell [49] in combination with a configuration file. The necessary specifications inside it included the building floor, camera name and the IFC-type of interest. For this study, two different floors, two cameras and the three IFC-types "IfcSlab", "IfcWall" and "IfcColumn" were considered. The geometric information of all filtered model objects was first plotted element-wise in a top-down perspective. A combination of all element images from one floor is shown in figure 5.26. Notably, this image has been recoloured to show slabs in the original red, wall elements in blue and columns in green for better readability. Although, each singular GT image only contained the element's ground floor area in red pixels for all IFC-types alike. As stated in chapter 4, the orientation of elements was extracted from the IFC file corresponding to the global cardinal directions of the building, with the north direction facing upwards.

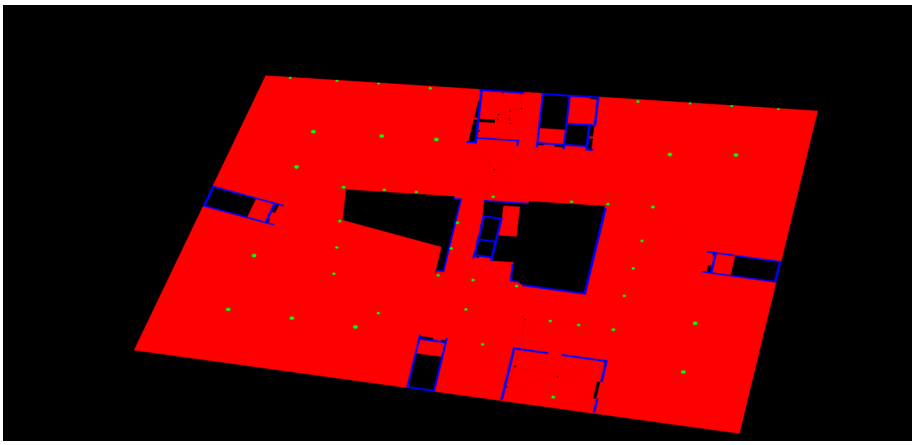


Figure 5.26: Combined element GT images

For the transformation of these GT style images, the collected site images were corrected before the segmentation process to remove perspective and radial distortions. Then, a perspective transformation matrix was calculated for each camera and building floor. According to the explanation in chapter 4, two sets of four points in the image shown in

figure 5.26 and one site image were collected for each one. One example of them and the computed transformation matrix H are shown in the following equations:

$$\mathbf{X} = \begin{bmatrix} 1140 & 290 & 1 \\ 2385 & 795 & 1 \\ 1755 & 1320 & 1 \\ 1145 & 1240 & 1 \end{bmatrix}, \quad \mathbf{X'} = \begin{bmatrix} 415 & 1500 & 1 \\ 1475 & 550 & 1 \\ 2120 & 940 & 1 \\ 2085 & 1520 & 1 \end{bmatrix}, \quad \mathbf{X'} = \mathbf{HX},$$

$$\mathbf{H} = \begin{bmatrix} 1.86472808 & 3.9259376 & -2345.9302138 \\ -1.13685646 & 0.08580612 & 4590.58451609 \\ 0.00105986 & 0.00001629 & 1.00 \end{bmatrix}$$

The third row of the transformation matrix should contain the values $(0, 0, 1)$ if the selected points in both images correspond perfectly. Because slight inaccuracies are unavoidable with this form of user input, the first two values were only close to zero. When applying the matrix to all pixels within the untransformed GT images, the newly created picture did not match the camera-taken site images exactly but very closely. This new combined image is shown in figure 5.27 and an evaluation of the achieved accuracy of this process is conducted later on.

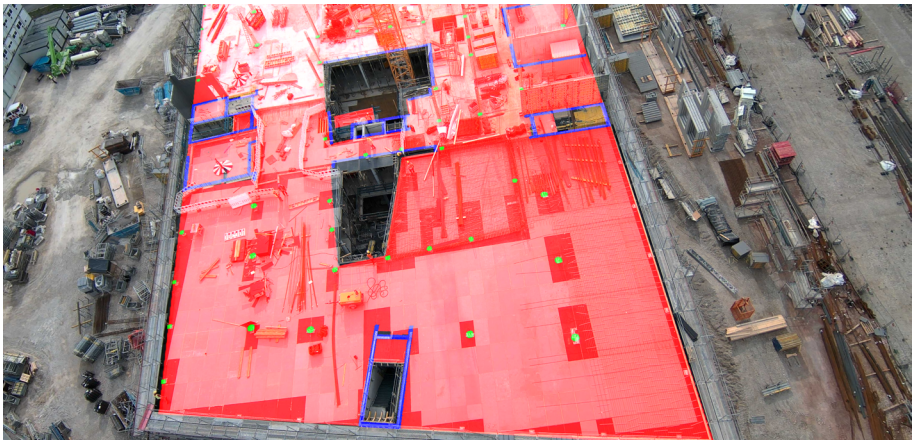


Figure 5.27: Combined transformed element GT images

As stated in the proposed approach of this thesis, the elements' geometry in the XY plane and the height are considered separately in this workflow to simplify the calculations and reduce the effect of inaccuracies. Therefore, the elements height information now had to be added to the transformed GT images for walls and columns.

After computing each cameras vanishing point from user input and extracting the height information from the IFC file, all element images were processed accordingly. For the unit conversion from meters to pixels, fixed reference values were used. Concerning the second camera of the surveilled project, this value was set to 25.5 pixels/m resulting in an $3.58m$ high element appearing 92 pixels tall in the image.

With this step, finished GT images have been created, which were used in the progress estimation processes proposed in section 5.2.1. A combination of all element images from one building floor is shown in figure 5.28.

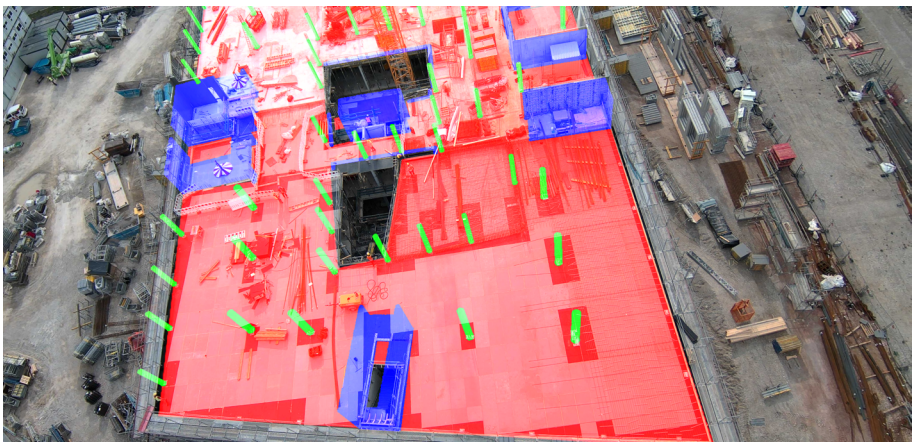


Figure 5.28: Combined transformed element GT images included height information

While the finished images were compatible with the estimation algorithm for a single element implemented in section 5.2.1, the process was relatively slow when applied to evaluate multiple elements at once. To reduce the computation time, a postprocessing step was added at this point, which converted the GT images into arrays containing the coordinates of all red pixels. All resulting arrays were saved in one JSON file per IFC-type so that during the evaluation process only three files had to be read per prediction image instead of one image file per element. This reduced the overall computation time from 30 seconds per prediction image to 4.5 seconds when evaluating 290 elements simultaneously. Further elaboration on the durations of different processes is provided in section 6.2. The resulting timestamps of the progress evaluation were coupled with the elements GUIDs saved within the GT images filenames. With this information, the deducted data could be saved as element attributes within a BIM model or used in other computations in the context of a digital twin [87].

5.3.2 Validation

After implementing the proposed methodology, the element GT images were created for all elements of the first and second building floor for the construction site of Project B. In this section, the model-generated images of all second-floor elements were compared to manually created ones to validate this approach's accuracy. In figures 5.29, 5.30 and 5.31, the model-generated results were stacked on top of the manual ones with common pixels coloured in red. Then, two different kinds of errors were marked within the images. All pixels that were part of the manually drawn segmentation masks but not the model-generated ones were called "missing pixels" and coloured in green. If pixels were marked within the model-generated pictures but not the manual ones, they were labelled "wrong pixels" and coloured in blue. The value of both errors was calculated as a fraction of wrong to correct pixels in percent.

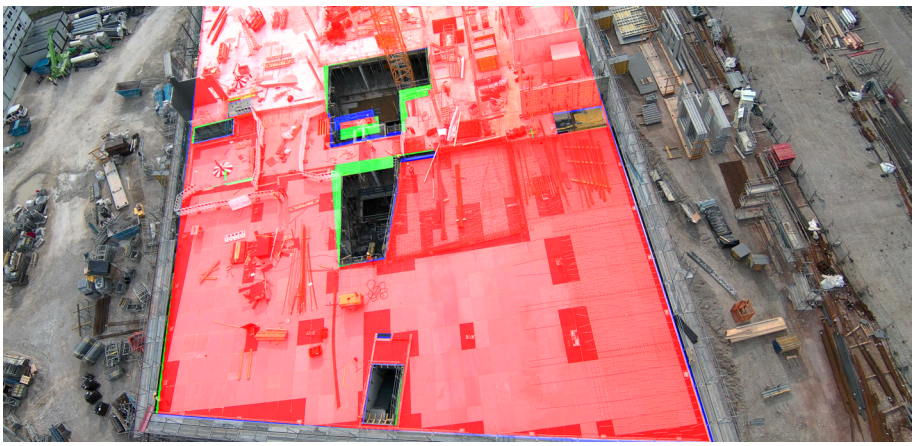


Figure 5.29: Slab elements: 1,67% missing pixels, 1,86% wrong pixels

The combined overlay of all slab elements in figure 5.29 showed a very high accuracy with only 1,67% missing and 1,86% wrong pixels. However, there was one exception in the form of a small slab in the middle staircase of the building. Because this element was the middle platform at the turning point of the stairs, it was not on the same height level as all other slabs. As the perspective transformation technique applied in the image generation works on a 2D basis, this difference in height resulted in a distorted display of the element in the final GT image.

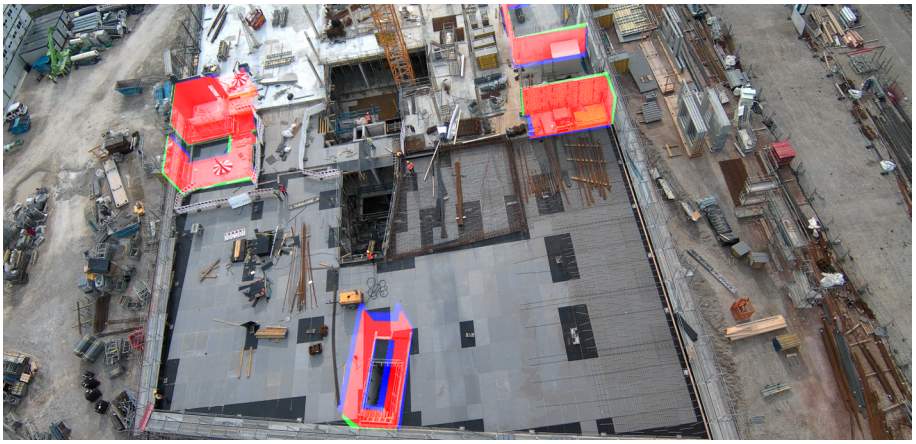


Figure 5.30: Wall elements: 5,06% missing pixels, 13,45% wrong pixels

The overall comparison of wall elements shown in figure 5.30 resulted in acceptable errors of 5,06% missing and 13,45% wrong pixels. A high fraction of the errors could be located within the walls at the bottom of the image, which were drawn too high in the model-generated segmentation masks. A possible cause for this problem was the implementation of the conversion from metric units to pixels using a fixed reference value. In reality, this relation between the actual dimension and the corresponding number of pixels in the picture varied over the image based on the different distances of objects towards the camera. For this thesis, a more dynamic approach to this problem has not been implemented as it often requires complex camera calibrations, which were unavailable for this data [51]. Furthermore, a fraction of the pixels attributed to the error of

wrongfully marked ones could be attributed to another part of the algorithm for generating the element's height. Because this process worked on every pixel of the ground floor area, openings for doors or windows could not be included, as this would have required further postprocessing of the images.

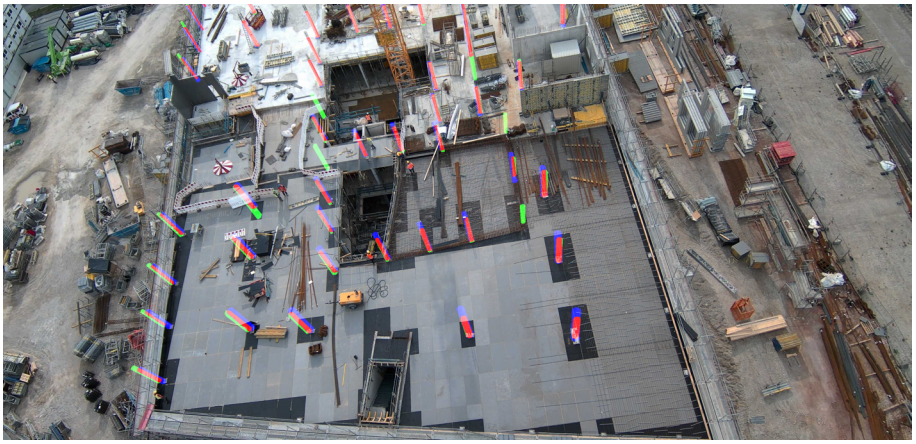


Figure 5.31: Column elements: 23,73% missing pixels, 37,39% wrong pixels

When comparing both segmentation mask generation methods for column elements shown in figure 5.31, considerably higher errors of 23,73% missing and 37,39% wrong pixels were detected. While a certain fraction of those could be attributed to the same problems concerning height generation stated in context with the wall example, the remaining portion of the error laid in the fact that a column's mask area is vastly smaller than of a slab or wall element. Therefore, inaccuracies due to the swaying of the crane or stemming from the transformation and height generation processes resulted in a higher fractions of wrong pixels. Additionally, many of the missing pixels did not result from the computation process proposed in this thesis but from modelling errors in the IFC-file. As figure 5.31 shows, a total of five columns were wholly coloured in green. They were entirely missing in the model-generated images, because they were either not contained in the BIM-model or had wrong attributes. Therefore, these elements have not been considered by the filter conditions stated at the beginning of this section.

Chapter 6

Results of an end-to-end approach

6.1 Data and methods

The proposed monitoring framework consists of multiple different parts working together, with the accuracy of each one being separately evaluated in the respective section in this thesis. An overall quantification of the framework's accuracies can, however, not be produced. Therefore, in this chapter, the interaction of all components, including the element-wise achievable accuracy and the required computation speed, was evaluated. For this task, another real-world case study was performed on the construction site of Project B, whose site images were already used in the newly created semantic segmentation dataset.

The data at hand for this project consisted of a stream of continuous site images taken by the camera setup proposed by F. Collins et al. [54]. Two months from the overall monitored period were extracted and used for the case study, comprising images of two cameras. While both were mounted on the same crane, one had a higher vantage point over the construction site than the other. Both cameras monitored 47 days out of the total of 61 days within the two months. However, as figure 6.1 shows, the surveilled days of both cameras dd not overlap completely. Approximately 163.000 pictures were taken during the two-month period and were used for the evaluation at hand.

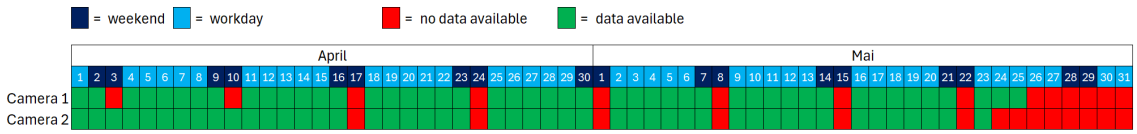


Figure 6.1: Available data from each camera for the case study

Additionally, an IFC-conform BIM model was available for the building of Project B. Because just a portion of all elements was constructed within the set timespan of two months, only the first and second floor were of interest for this case study. From all model objects within these two levels, 290 elements were selected by applying the filter conditions stated in section 5.3 concerning their IFC-type and name attribute.

As-built information was needed to compare the accuracy of the proposed setup in detecting construction progress timestamps. Because no other sources were available for this data, the required information was extracted via a manual inspection of the site images. As this process was considerably time-consuming, an evaluation of the procedure's accuracy was not conducted on all 290 elements but on eight representative ones later on.

In the first step of the case study, all 163.000 images were preprocessed outside of the scope of this master's thesis to remove all radial and perspective distortions, as it is also described in section 5.3. The corrected images were then segmented by the PidNet-S [77] model and the resulting prediction files were further processed. Before that, the projects BIM model was used to generate GT images for all 290 elements. Because those were spread over two building floors and the image data at hand stemmed from two cameras, this process included the calculation of four different perspective transformation matrices and two vanishing points from user input. As a result, 580 individual GT images were created. With these, the construction progress extraction algorithm was performed, which was already elaborated in section 5.2. For this part, the parameters retrieved in section 5.2.2 were used, resulting in an averaging interval of 7 minutes and thresholds of 75% and 10%, respectively. The evaluation process included the optimization step described in section 5.3 to save on computational resources and time. Lastly, the generated progress timestamps of eight selected elements were compared against their manually derived as-built timestamps to verify the overall accuracy of the combined framework. But before that, the durations of all included steps were recorded and are discussed in the following section.

6.2 Computation time

In this section, the computation time needed for each part was evaluated, apart from all steps needed in the creation of the custom dataset reviewed in section 5.1.1 as well as the training and testing of different semantic segmentation models shown in section 5.1.2. These parts have been omitted, as they were performed only once and generated a universal segmentation tool as output intended to be used on multiple different construction sites. However, the goal of this section was to evaluate the monitoring speed of the proposed setup. Hence, only the individual computation steps needed for every construction site were considered in this summary.

The procedure was tested on the construction site of Project B on all elements from the first and second building floors. For the upcoming evaluation, the available data stated in section 6.1 including all 290 elements over the whole timespan of two month was used. All operations in the first three steps stated in table 6.1 can be summarized as preparation procedures and were performed on a Surface Laptop Studio with a 11th Gen Intel Core i7 processor with 16 GB RAM and a Nvidia GeForce RTX 3050 Ti Laptop GPU. Summing all of those parts together, the total computation time amounted to 1h 29min 46sec for two building floors, two cameras and 290 elements. For this sum, a duration of five minutes was allocated for the collection of user input within the calculation of the perspective transformation matrix and two minutes for the computation of the vanishing point. Also, the 9.2 seconds needed to generate the finished GT images within the third step only needed to be applied to 270 elements as slabs were excluded from this processing part. Although this method needed some time to be set up initially, all those operations can be scheduled and carried out before work on the construction site begins. In conclusion,

Table 6.1: Summary of computation durations

Process	Duration	Frequency
Step 1: Generating ground truth images in cardinal direction		
Calculation of the IFC models bounding box	3min 28sec	per project
Generation of single element images in cardinal direction	3.3sec	per element
Combination of all images into one for the next steps	36sec	per project
Step 2: Transforming images to match the cameras perspective		
Calculation of the transformation matrix (user input)	ca. 5min	per camera and floor
Transformation of the single element images	0.2sec	per element and camera
Step 3: Adding height to images for column and wall elements		
Calculation of the vanishing point (user input)	ca. 2min	per camera
Generation of the finished GT image	9.2sec	per element and camera
Conversion of images into arrays of coloured pixels	0.5sec	per element and camera
Step 4: Evaluation of construction progress		
Segmentation of the site image (18.68 FPS)	5.4sec	per frame
Processing a prediction image to obtain raw data	0.016sec	per frame and element
Calculation of progress timestamps	0.001sec	per frame and element

only the durations of algorithms in the fourth step stated in table 6.1 were of interest in order to determine, how fast the proposed workflow could analyze construction progress. For the problem at hand, each segmented prediction image was compared against all 290 element GT images and the resulting raw data was processed in order to generate the required construction progress timestamps. The following values were calculated in reference to one processed frame:

$$0.016\text{sec} \cdot 290\text{elements} + 0.001\text{sec} \cdot 290\text{elements} + 5.4\text{sec} = 10.33\text{sec}$$

Evaluating one image in summary took close to 10 seconds on average. When using the image capturing setup described by by Colins F., Pfitzner F. and Schlenger J. [54] in section 5.1 every 30 seconds a new picture is taken by each camera. Assuming that some of the required processes in the fourth step can slightly overlap, three cameras could be used simultaneously to process all images in real-time.

6.3 Accuracy

Summary of findings

Before the accuracy of the proposed approach could be thoroughly evaluated, some data clean-up had to be carried out. This involved excluding 84 elements from the list of 290 objects, because they were installed before or after the two-month surveillance period. Therefore, their construction progress was not represented within the used image data. Furthermore, four elements were removed because they were not visible within the

viewpoint of either of the two cameras. Seven more elements needed to be excluded, as their bounding box was occluded by cranes or other construction site equipment during the whole observation timespan. The remaining 195 elements were processed accordingly. Some walls and columns did not fit into the proposed method while passing the element filters applied in section 5.3 as well as the clean-up process. This was due to errors within the BIM model because a total of 17 elements were classified as IfcWalls while having a very small cross-section and a high length. These would better be categorized as IfcBeams, resembling their load-bearing behaviour and optical appearance. Therefore, the segmentation model could not recognize those elements, which is the reason for excluding them from this evaluation. Additionally, two further elements had been removed because their height attribute was not set correctly, resulting in their GT image only containing the ground floor area.

After these clean-ups, 176 samples remained to be evaluated. With closer inspection of the program's output, twelve did render unusable results, while the underlying errors can be traced back to the element's GT images. For ten columns and two wall elements, the inaccuracies of all preceding processing steps combined with additional ones due to the swaying of the crane in some frames resulted in the GT map of these elements being shifted too far away from the actual element area. Additional elaboration on this phenomenon is provided later in this chapter. Figure 6.2 shows four columns from this subset of the twelve elements, whose GT images do not overlap with the correct element area at all. While the frame shown in the figure was an extreme case of the problem that did not represent the whole number of evaluated frames, this clearly shows that sometimes the sum of inaccuracies could exceed a certain threshold, above which a successful evaluation of progress timestamps was no longer possible.

Figure 6.2: Four columns of the first building floor whose GT maps were generated very badly

For multiple wall and column elements, a problem arose that did not have any effect in the evaluations of section 5.2.2. Here, all elements were monitored over the whole timespan of two months in contrast to a duration roughly matching the time of their construction.

This resulted in some walls and columns on the second floor having pixels attributed to them, which should correctly be counted towards other elements. The reason for this could be found within their GT images again. Because all pixels overlapping with the mask area of each element were counted within every frame, the program could not distinguish between different elements or building levels, as this information was not contained within the segmented prediction files. Figure 6.3 depicts one of the wall elements in question, showing many concrete labelled pixels in the first half of the evaluation. However, these pixels belonged to wall elements below and behind the actual one, constructed before the duration of the surveillance. This occlusion problem could happen for every element on the second floor, as actions happening on the first floor prior to the construction of the element in question were compared with its GT mask area. Currently, no solution is implemented within this thesis's scope, but possible approaches are provided in this section and chapter 7.

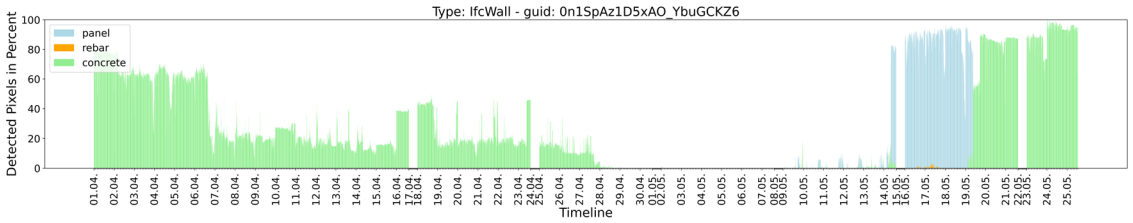


Figure 6.3: Evaluation of a wall in the second building floor

The proposed methodology stated in chapter 4 evaluated the construction progress of each building object in the BIM model fitting the selection filters. Within this workflow, the complete area of each singular model element was evaluated at once, as the whole element geometry was present in the model-generated GT image. Especially for slabs, this posed a problem when big elements were modelled as one object but were then constructed in multiple concreting steps. For this reason, the proposed methodology assumed that the BIM model only contains elements that have been installed in one part. However, some slabs were modelled contrary to this assumption within the surveilled samples. Figure 6.4 shows one slab element on the second floor as an example of this problem.

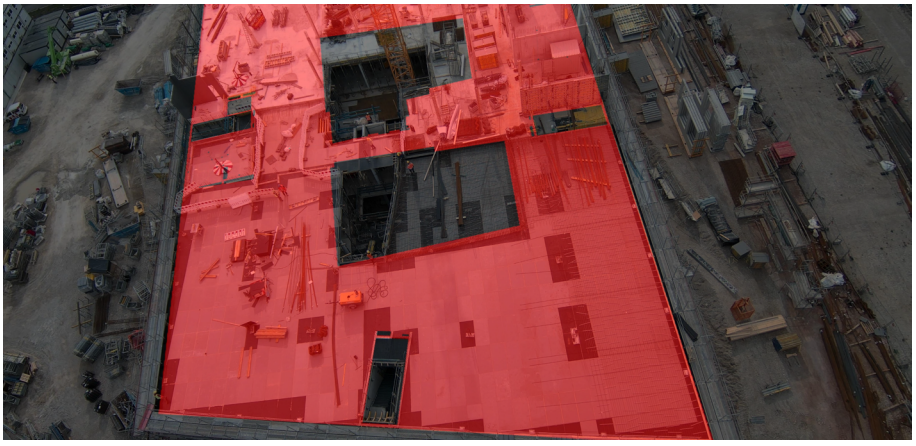


Figure 6.4: Slab element of the second building floor covering most of the floor area

When looking at slab elements like that, the proposed algorithm resulted in wrong conclusions for the construction progress timestamps, as the whole element area was evaluated at once. The procedure then failed to categorize the slab into a "panel", "rebar" or "concrete" class. As the shown slab element spans most of the building's floor area, a certain portion of the slab was entirely constructed and even occluded by numerous elements that are already installed on top of it, while other areas were only in the "rebar" or "panel" phase. The resulting evaluation shown in figure 6.5 is therefore very noisy and no reasonable evaluation for the whole element could be deducted by the program.

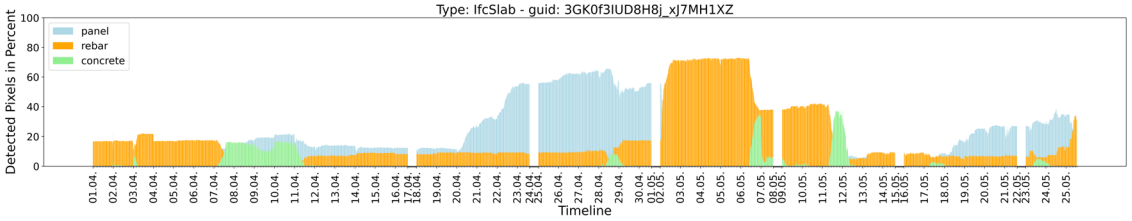


Figure 6.5: Evaluation of a slab in the second building floor with noisy results

As stated at the beginning of this section, no sources for as-built timestamps were available other than manually extracting them from the site images. Therefore, eight representative building elements were thoroughly inspected and their derived construction progress timestamps were compared against the as-built ones. The calculated differences are stated in table 6.2 and the result plots can be seen in appendix A. While a segmentation-based approach could often produce no correct estimate for the endpoint of the rebar installation due to reasons stated in section 5.2.2, the results were still included in the upcoming evaluation.

Table 6.2: Examples for determining the accuracy

	Panel Start	Panel End	Rebar Start	Rebar End	Concrete Start	Concrete End
Slab 1: 1zKYFF6k97s9zxpqalau9j, 2nd floor, plot in figure A.1						
as-built	27.04. 15:54:57	28.04. 07:20:13	04.05. 17:22:49	04.05. 17:52:40	11.05. 10:58:14	11.05. 11:05:50
derived	27.04. 16:06:00	27.04. 16:54:00	28.04. 07:00:00	04.05. 17:44:00	03.05. 08:06:00	11.05. 11:06:00
error	00:11:03	-14:36:13	$>$ 1 day	-00:08:40	$>$ 1 day	00:00:10
Slab 2: 19fYLvxgf8suiIdEnYtMFv, 2nd floor, plot in figure A.2						
as-built	01.04. 07:00:00	26.04. 12:18:53	27.04. 07:34:26	28.04. 09:57:57	04.05. 12:23:20	04.05. 18:30:26
derived	01.04. 07:00:00	23.04. 11:46:00	27.04. 10:42:00	27.04. 11:18:00	04.05. 14:04:00	04.05. 14:04:00
error	00:00:00	$>$ 1 day	03:07:34	-22:39:57	01:40:40	–
Wall 1: 0$t85hqrT7yOpjG8x4qx6K, 2nd floor, plot in figure A.3						
as-built	10.05. 12:48:54	10.05. 12:48:54	occluded	occluded	16.05. 13:41:20	16.05. 13:41:20
derived	01.04. 09:48:00	10.05. 12:50:00	11.05. 12:40:00	–	01.04. 07:00:00	16.05. 13:42:00
error	> 1day	00:01:06	–	–	> 1day	00:00:40
Wall 2: 3UuBFrtqLBgRqko$UrITge, 2nd floor, plot in figure A.4						
as-built	12.05. 12:39:39	12.05. 12:48:45	occluded	occluded	19.05. 09:29:10	19.05. 09:40:17
derived	28.04. 08:44:00	12.05. 12:40:00	27.04. 12:50:00	–	01.04. 07:00:00	19.05. 09:48:00
error	> 1day	-00:08:45	–	–	> 1day	00:07:43
Wall 3: 2rbZufm2L6SvoLNj_lChwC, 2nd floor, plot in figure A.5						
as-built	11.04. 16:06:08	12.04. 08:50:10	12.04. 10:57:06	13.04. 08:16:46	26.04. 08:17:45	26.04. 08:17:45
derived	11.04. 19:56:00	12.04. 09:48:00	12.04. 10:56:00	12.04. 11:50:00	12.04. 08:32:00	26.04. 08:32:00
error	03:50:08	00:57:50	-00:01:06	-20:26:46	> 1day	00:14:15
Column 1: 3Qex4CRQj7zBu0Bf66FEnU, 1st floor, plot in figure A.6						
as-built	12.04. 12:24:04	12.04. 13:57:06	06.04. 14:43:47	06.04. 14:51:22	13.04. 07:24:41	13.04. 07:31:15
derived	04.04. 16:44:00	–	06.04. 14:44:00	–	13.04. 07:24:00	–
error	> 1day	–	-00:00:13	–	-00:00:41	–
Column 2: 3Qex4CRQj7zBu0Bf66FEpZ, 1st floor, plot in figure A.7						
as-built	07.04. 12:02:25	07.04. 12:02:25	06.04. 11:21:58	06.04. 11:37:09	13.04. 11:58:44	14.04. 08:15:44
derived	06.04. 12:00:00	07.04. 12:18:00	06.04. 11:28:00	–	13.04. 11:58:00	–
error	-23:57:45	00:15:35	00:06:02	–	-00:00:44	–
Column 3: 0G8O_QWTj0oB1llP5cRKJC, 2nd floor, plot in figure A.8						
as-built	21.04. 14:50:24	21.04. 15:03:33	21.04. 14:42:20	21.04. 14:47:22	16.05. 16:39:48	16.05. 16:39:48
derived	21.04. 13:06:00	28.04. 15:58:00	21.04. 12:36:00	–	16.05. 16:38:00	–
error	-01:43:36	> 1day	02:06:20	–	-00:01:48	–

Analysis of the outcomes

The accuracy in both slab examples did not match the results of the studies in section 5.2.2. While construction progress milestones could be detected reasonably accurate for both previous examples, except for the concrete phase for the last one, the results from these samples varied heavily. This was especially unexpected, as the element for the second example in this case study was the same one used in section 5.2.2. While the endpoint of the concrete state could not be detected in both tests, the program deducted the end timestamp for the panel phase correctly in section 5.2.2. This could be attributed to the higher value for the upper threshold of 85% in the first test compared to the 75% used here.

When looking at the first slab sample of this case study and its result in figure A.1, it is not apparent, why the timestamps for the beginning of rebar and concrete works was detected in the middle of numerous panel pixels. Consulting the construction site images directly also revealed no apparent reasons for this behaviour. However, the prediction images contained small patches of "panel_rebar" and "panel_concrete" labelled pixels at these

timestamps. Figure 6.6 shows one image and its corresponding prediction result for the time of the detected endpoint for the concrete phase. These wrong segmentation results only occurred for one camera and a small number of frames. However, at these two points, the implemented averaging techniques were not able to mitigate its effects on the final evaluation. The reason for this is, that the ground floor area of the slab in question is only 6.2m^2. Seemingly small patches with incorrectly segmented pixels could, therefore, not be compensated by the few remaining correct pixels.

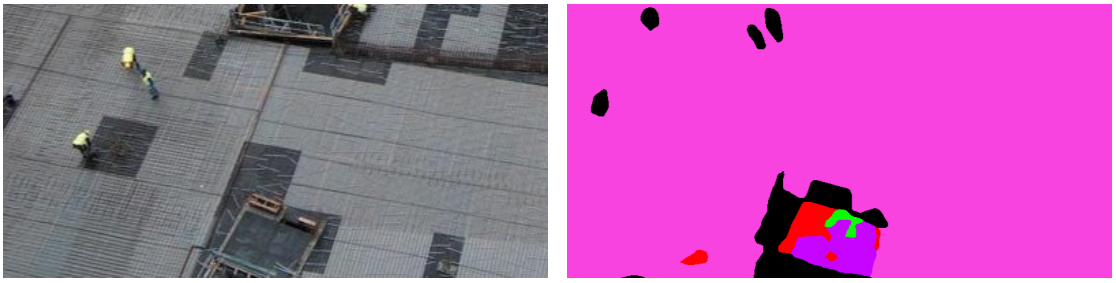


Figure 6.6: Examples of a wrong segmentation result

Both the first and second wall examples posed the same problems. First, the starting point of panel and concrete works was placed at the beginning of the considered period because of the occlusion problem stated in the beginning. Here, pixels from the walls of the first floor were counted towards the wall of interest on the second floor, fulfilling the 10% threshold of both classes immediately. In contrast to that, the ending timestamp for both classes was correctly derived with only little inaccuracies. Secondly, all rebar works should not be visible in the plot, considering that the formwork side facing the camera was placed first, thereby occluding all rebar elements of the object. The detected start of rebar works in the first example resembled the point in time at which the first rebar pieces were put into place in the wall in front of the wall element of interest. Because the segmentation approach did not differentiate between multiple instances of walls, these errors stemming from occlusions could not be avoided in the current state of the proposed framework.

In contrast to both examples, the last wall sample generated better results. Only the detected starting point of concrete works contained a higher error, with all other points derived reasonably accurate. The only exception was the ending timestamp for rebar works. However, this problem was already discussed in section 5.2.2 and its cause lied within the fact that a segmentation approach will always detect an area as fully filled with rebar elements when one layer is finished. But actually, the rebar process is only complete when all four layers are completed, which can hardly be detected by a segmentation-based approach. Hence, this timestamp is not as relevant to the overall accuracy as others.

The first column example revealed an interesting problem in relation to the number of used cameras and the overlap of monitored days. When switching from the 09the to the 10th of April, in figure A.6 it seems as if the detected pixels for rebar works jumped suddenly by about 20 percent. As this change occurred directly within the first frame of this monitored day, it could hardly be attributed to site progress. Instead, the reason for this is shown in figure 6.1 as only the second camera recorded any data on the 10th of April.

```
{"timestamp":"20220409195600", "panel":0.0792, "rebar":33.0471,
        "concrete":0.0, "total_mask":4538.0}
{"timestamp":"20220409195800", "panel":0.3651, "rebar":29.2958,
        "concrete":0.0, "total_mask":4538.0}
{"timestamp":"20220410070000", "panel":0.0, "rebar":55.5423,
        "concrete":0.0, "total_mask":4416.0}
{"timestamp":"20220410070200", "panel":0.0, "rebar":67.6630,
        "concrete":0.0, "total_mask":4416.0}
```


This excerpt of the raw data generated by the evaluation algorithm shows not only the sudden change in the percentage of detected rebar pixels, but also different total mask areas for the first and last shown entries. This alone indicated a change in the used cameras, as the GT mask area of the element was different for each camera because of their alternate viewpoints. The jump in the detected rebar pixels could be attributed to a different quality in the segmentation results for each camera. The images in figure 6.7 show the overlap of each camera's predicted pixels (yellow for column_rebar) coloured in orange with the element's GT map (red). A significantly smaller overlap was achieved for the first camera than for the second camera. If data for both was available, like on the 9th of April, the average of both values were taken into account for the evaluation. However, as one camera did not record images on the 10th of April, no averaging was performed. A higher percentage of detected pixels was calculated, resulting in the sudden jump shown in figure A.6.

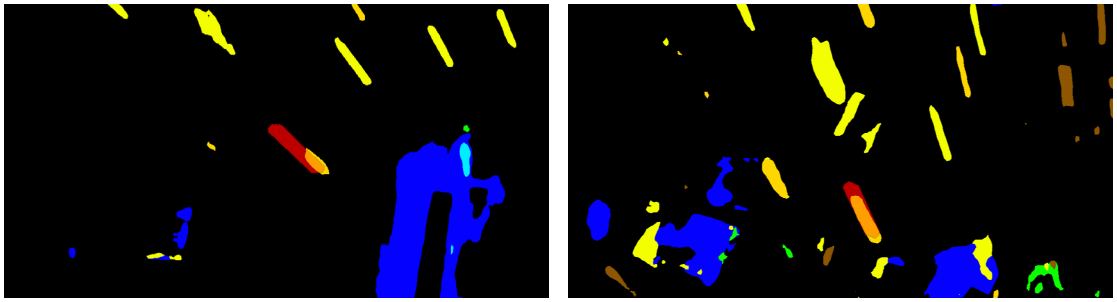


Figure 6.7: Left: Prediction image for camera 1, Right: Prediction image for camera 2

While this observation is interesting, it did not influence the other estimated progress timestamps for the surveilled samples. The missing or only very inaccurately detected results for all column examples could traced back to another problem. Its root was that column elements are very slim and, therefore, contained fewer pixels in their GT masks compared to walls or slabs. This increased the risk of the GT images no longer overlapping with the predicted element pixels when multiple sources of inaccuracies combined. The case study in section 5.3 showed that the model-generated GT images could already contain considerable defects compared to manually drawn ones. Within the evaluation process, another source of inaccuracies was then added. While the cameras retained a fixed point of view over the surveillance period, they could move slightly due to the swaying of the crane in stronger winds or during different crane operations. To understand this factor, 10% of all images taken on one day were manually evaluated. In every picture, the

same point was marked and its movement over time was recorded. Figure 6.8 illustrates the results for two points relativity close to the camera location. Figure 6.9 shows one point further away from the crane. The included plots of deviations from a middle value for each point show considerable variations throughout the day. Moreover, the studied point further away from the camera in the right image shows a nearly twice as strong maximum movement compared to the points closer to the camera.

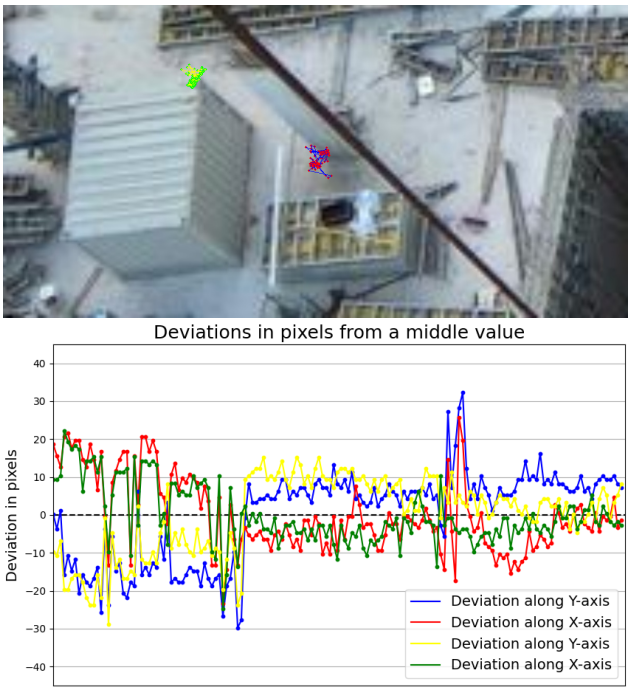


Figure 6.8: 1st example of camera viewpoint movement

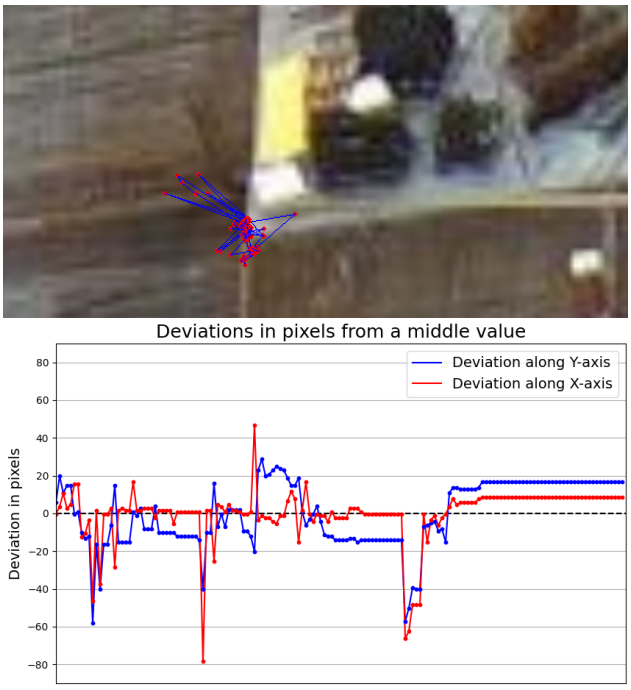


Figure 6.9: 2nd example of camera viewpoint movement

When construction progress was evaluated in times of high crane movement, the GT mask of an element could, therefore, be shifted up to 40 pixels away from its original location. This can result in no pixels being detected for the slim column elements, as the GT masks did not overlap at all with the actual element in the prediction image. Therefore, the resulting plot for the third column example A.8 shows multiple spikes and valleys, making a reasonable progress estimation impossible.

Limitations

The success of the proposed workflow is limited by some factors with one of them lying within the BIM model. As the derived algorithms to compute the model-generated GT images work with several attributes, namely the IFC-type, height, and object name, the attributes must be set correctly. Therefore, corresponding guidelines for the model generation need to be included in projects, on which the proposed workflow is intended to be tested. Another addition to these modelling rules is based on the assumption that all elements modelled as one object are also installed in one part. As every singular model object's real-world construction progress is evaluated as a whole at once, slabs or walls that are concreted in multiple sections also need to be divided into the respective parts in the BIM model.

While these limitations can be easily overcome by following the stated modelling rules, others need further research. One of those is the problem of occlusion occurring with elements in higher building floors. Their GT masks overlap with other elements constructed in floors below or behind and in front of them. Pixels belonging to those objects are then counted towards wrong elements. Because this results in incorrect conclusions for some progress milestones, a solution for this limitation would be beneficial. A possible approach is the implementation of checks, which, for example, only enable the monitoring a wall element as soon as the slab underneath it is finished. However, as this requires knowledge of the sequence in which all elements are constructed, this approach is not part of the scope of this master's thesis.

The same conclusion can be transferred to the accuracy problem caused by crane movements. The extensive influence of this on different points within the site images is shown in figures 6.8 and 6.9. However, an approach towards solving this problem can not be presented within this thesis. A possible way to remedy this limitation is the inclusion of markers on the construction site, which are also referenced in the BIM model. As this approach can eliminate the need for user input in calculating the model-generated GT images, these markers can be used to not only generate a perspective transformation matrix once but dynamically for each singular site image. This or other dynamic approaches within the transformation process could mitigate the crane-induced movements and, in turn, increase the accuracy of the generated element GT images. However, this lies beyond the scope of this master's thesis.

A further shortcoming of the proposed methodology in creating the model-generated GT images is that no openings of model elements can be included. The reason for this can

be found within the geometric representation of openings in the IFC file structure. These are stored in the form of separate model objects linked to the respective building element. The derived algorithm adds the height information for walls and columns homogeneously over the complete ground floor area of the element. Openings for doors, windows or other installations would then need to be cut out in a second step, which is not implemented in the proposed framework. As a result, this approach adds an additional source for inaccuracies within the GT images, which becomes more prominent with a higher proportion of openings in an element. However, while this particular influence was expected to be seen in the case study, no problems seem to have come to light in the surveilled samples. Nonetheless, further research and optimization in this regard can be done by other researchers, as this is not part of this work's scope.

The stated limitations, especially considering occlusions and inaccuracies due to crane movements, reduce the proposed approach's performance, as seen in the reviewed examples within this section. Besides the highlighted issues, the presented approach consistently yields strong results across most cases, demonstrating robust performance. A thorough discussion is presented in the following chapter to summarize the successful achievements of all parts of the proposed workflow, contrast them with their shortcomings, and provide incentives for further research.

Chapter 7

Discussion

The implementation of the in chapter 4 proposed methodology and its thorough part-wise and combined testing are complete and many potential contributions to aid in advancing digitization within the construction sector have been achieved. However, next to these successful achievements, some shortcomings of the elaborated procedures have come to light. Both are contrasted to each other in this chapter.

Dataset creation

The first part of this master's thesis focused on creating a new semantic segmentation dataset tailored to the needs of this research approach. After the required raw data in the form of site images captured by crane-mounted cameras had been collected and sorted, pixel-wise dense annotations of nine labelled classes were authored. These included three construction phases with a panel, rebar and concrete state for slabs, walls and columns. All resulting GT images were structured into one training and two testing subsets. Pictures from two real-world construction sites were used in all of them apart from the second testing set, which included images from two different projects. The dataset can be used to train segmentation models to confidently detect the listed cast-in-place concrete elements in environments similar to those found on the projects used in the training subset. Additionally, the alternate second testing set helps to determine the model's performance on other unknown construction sites that do not resemble the training projects' conditions.

However, due to the limitation of available research time within a master's thesis, the dataset consists of only 420 annotated images summed up over all training and testing sets. These numbers lack behind those of other datasets created in papers by Z. Wang et al. [35], X. Luo et al. [33] or by A. Pal et al. [37] with 660, 7.790 and 2.458 images respectively. Nevertheless, the small dataset still enabled the trained segmentation models to perform reasonably well on data from unknown and known construction sites.

Adding new reviewed and annotated data would render the dataset more homogeneous, balancing the class-wise metrics (number of instances per class and annotated mask area per class) shown in figure 5.8. If these images additionally stem from different construction sites, the trained models' performance on unknown projects can be significantly increased. The metrics on the alternate test set in correlation with the standard test set shown in table 5.2 currently suggest a significantly worse performance on data from unknown sites. However, it can not be confidently stated, how many new images and sites need to be included in the dataset and whether segmentation models trained on it can ultimately

evaluate images from arbitrary construction sites confidently. These questions remain to be solved by future research.

The dataset can already be used in combination with other discipline-specific datasets for detecting all building elements as it is proposed in different papers [34], [35], [37]. In this regard, the dataset contributed by this thesis can aid in generating a semantic segmentation model capable of understanding and detecting all elements present on a construction site and, therefore, evolve into a monitoring tool usable for all forms of on-site work.

Training and testing of semantic segmentation models

Further, within the first part of the proposed methodology, a training pipeline was created based on the MMSegmentation repository [56], and the newly conceived dataset was integrated into its workflow. With this proposition, the well-readable MMSegmentation documentation can be consulted, making this pipeline easily accessible and extendable to new datasets or segmentation models. In total, 49 different combinations of backbones and algorithms as model architectures have been trained on the new dataset with fixed learning rate and weight decay parameters. After an extensive evaluation of all models on both available testing sets including the computation of the segmentation speed, additional fine-tuning was conducted on two top-performing architectures. The resulting trained semantic segmentation models achieved good accuracies with a maximum mIoU metric of 70,35 on the standard and 41,00 on the alternate test set.

The list of evaluated models presents a comprehensive but not finite overview of segmentation approaches, leaving room for further testing of other model architectures or developing new ones. Consulting the evolution of best scores for established datasets like the Cityscapes or ADE20k dataset, a continuous increase in performance over the last years emerges [29], [31]. When transferred to the dataset of this thesis, future segmentation algorithms can possible achieve even better performances than the 49 already tested ones. Additionally, all included algorithm and backbone combinations are trained using a fixed combination of hyper-parameters, which worked well on other curated datasets. While a closer inspection of those parameters was not possible within the time limit of this thesis, other research approaches can focus on this very part. However, only little improvements can be expected from such efforts, as the fine-tuning section in this thesis shows, that the evaluated metrics only increase slightly when changing the learning rate. Therefore, the list of tested models provides a close insight into which fusion of currently available algorithms and backbones works best concerning the segmentation of cast-in-place concrete elements on construction sites.

Evaluation of construction progress timestamps

Using the predictions stemming from a trained semantic segmentation model as raw data, algorithms were proposed within this thesis to evaluate a continuous stream of images

and extract element-specific construction progress timestamps. A case study on two real-world construction projects proved the viability of this approach alongside its ability to compute accurate progress estimations for sample elements. For each of the six evaluated examples, optimal parameters for the detection thresholds and averaging techniques were derived. From these results, an overall applicable combination of settings was extracted to be used for observing other construction sites. Nevertheless, the values provided in table 5.16 clearly show that highly varying setting combinations for different element types must be used to get optimal results. Because of the inhomogenity inherent to construction sites and their elements, the chosen average of the three parameters is unlikely to work perfectly on every monitored element, as the studies in chapter 6 proved.

Alongside these surveilled building objects, the importance of using multiple cameras at once and implementing averaging techniques to combine a set number of frames emerged (see figure 5.12 and 5.13). These methods are also proven to be capable of handling faulty images and wrong segmentation results reasonably well.

While these tests obtained good results, one deficit of the segmentation-based approach came to light. The prediction results are only usable when generated from images with apt lighting conditions. Therefore, the proposed framework can not be used in projects with shift operation, as images taken during the night would not be segmented correctly. Hence, only frames between 07:00 a.m. and 07:00 p.m. were taken into consideration for the evaluations within this thesis.

Furthermore, the elaborated methodology could not detect two other progress events correctly. The first one in question is the completion milestone of rebar works because a segmentation-based approach will only identify an area as 100% completed in this regard as soon as one whole layer has been installed. However, naturally four different layers with two directional ones per side are needed on top of each other to complete the installation for one object. The second event concerns the concreting stage for walls and columns. This one is not detectable because the emerging concrete areas are not visible within the segmentation-based approach, as the proposed camera setup mainly looks at the vertical formwork sides of those elements. After the concreting is finished, some concrete-labelled pixels are visible at the top of the element, but this is insufficient to guarantee a confident detection of the event. A possible solution for this and the latter problem is the combination of a segmentation-based approach with other surveillance techniques like activity monitoring or infrared sensors as proposed by J. Yang [40] among other researchers. In a paper by Pfitzner et al. [45], the monitoring of concrete pouring with graph-enhanced computer vision is proposed. Their approach combines a YOLOv8 model for objects detected to monitor progress as well as activities with an ontological model for process reasoning. This method's fusion of computer vision and knowledge graphs achieves high accuracy rates in identifying concrete pouring states, as demonstrated in a case study from an active construction site in Munich. Therefore, this approach, combined with the one proposed in this thesis, could possibly remedy the problem of confidently detecting concrete pouring stages for walls and columns. Similar procedures

are conceivable for evaluating the installation of rebar elements to mitigate the elaborated problem.

Coupling of the progress data with a BIM model

After successfully evaluating the site images to get element-specific progress timestamps, the next step in the proposed procedure was linking this data to a digital representation in the form of a BIM model or digital twin. This was achieved by automatically generating the GT style images needed for each element to extract its progress advancements. The necessary information was obtained from a IFC-file containing a BIM conform representation of the building geometry. Using one of the projects included in the new segmentation dataset as an example, a case study was performed and the resulting accuracy was evaluated. For bigger slab and wall elements, excellent results emerged, but for smaller objects, the inaccuracies mainly stemming from utilizing user input and from modelling faults in the BIM model, added up to more significant errors in the resulting GT images. Furthermore, openings for windows, doors or other installations in model elements could not be represented with the proposed algorithms so that they posed an additional source of inaccuracies.

A primary area in need of further research in this part of the proposed methodology lies in the generation of the 2D transformation matrix and the inclusion of the element's height in the GT images, as both steps need user input in the current configuration. But there are possibilities for automating this part. Most promising is the introduction of markers on the construction site, which are also referenced within the BIM model and could replace the points now given by user input. This method and similar approaches are already widely used for point cloud registration in the context of laser scanning [88] but need to be investigated for segmentation-based approaches.

End-to-end approach in real-world scenarios

In the last part of this thesis, the procedure was tested as a whole to determine overall accuracy and computation time. The resulting durations of all necessary steps, excluding the generation of the dataset and the training of segmentation models, amounted to 10.33 seconds. Therefore, this performance is sufficient to run a setup with three cameras under the assumption that the singular processes can overlap with each other slightly. However, according to the proposed monitoring framework by F. Collins et al. [54], more cameras on different cranes are needed, to fully surveil a construction site and reduce the risk of occlusions for some elements. To achieve this, it is possible to run more cameras simulatinously while outsource the necessary computation to a server running 24h per day, because construction progress is usually only advanced during working hours from 06:00 am to 19:00 pm. Another way to remedy this problem is the implementation of parallelization techniques, as the derived computation durations are calculated with the program using one CPU core. So for every additional used core, the number of cameras

can increase accordingly. Nonetheless, these proposed features are past the scope of this theses and therefore not implemented.

To determine the accuracy, all 290 elements of two building floors from an example project were surveilled over a two-month period. Alongside multiple excellent results, some elements were found, which could not be processed due to errors and circumstances that were out of the control of this thesis. A small number of those elements can be included in the workflow if specific modelling rules are followed within the BIM model. For example, one slab object could not be monitored successfully because the digital element was not modelled according to the actual schedule of its construction as concreting sections were not included in the model. Also, elements with incorrect height and wrong IFC-type attributes rendered false results since the algorithm for generating the GT images can not process those elements correctly.

Furthermore, a small number of elements received wrong progress timestamps because other elements below or in front of them were constructed beforehand. Pixels belonging to those objects were then attributed to the surveilled elements in question. Some of these occlusion-based problems can be remedied by implementing certain checks based on the as-planned construction schedule proposed by M. Golparvar-Fard et al. [89]. Then, the evaluation of a particular object would only start if the required preceding elements are finished. An example of this is to check for the completion of a slab element before evaluating the wall constructed on top of it. Another approach worth pursuing is combining semantic segmentation with additional instance segmentation to enable the segmentation model to allocate pixels to the correct element automatically.

Another problem, that came to light in this section, concerns lacking surveillance data. For one column example, a sudden change in detected pixels was recorded as one of two used cameras did not capture images for one day. Ultimately this lead to the detection of a wrong progress timestamp. A possible way to remedy this situation is to either ensure, that all available cameras a switched on and off simulatinously or accomplish more redundancy by using a higher number of cameras overall. The latter would also benefit the overall accuracy of the proposed framework, as more available viewpoints could help with problems concerning occlusion. Further studies in this regard are not contained within the scope of this thesis and can be conducted in real-world deployments of the proposed monitoring framework.

Chapter 8

Conclusion

The overall goal of this master's thesis was to provide a novel approach towards automating progress monitoring of cast-in-place concrete slabs, walls, and columns to help raise the low digitization level of the construction sector. For this task, a custom-made dataset of annotated site images and trained semantic segmentation models have been created, which can confidently segment said elements in pictures from different sites. Additionally, algorithms for evaluating element-specific construction progress timestamps and procedures for coupling the obtained information with digital building models have been proposed. With the integration of different averaging approaches, robust performance with high levels of accuracy could be proven in case studies concerning the specific parts of the elaborated framework and the whole end-to-end approach. Also, the viability of the proposed methodology on real-world construction sites has been established in those studies. Next to those contributions, additional incentives for future research to enhance specific parts of the procedure were demonstrated.

This proposed tool to aid the construction monitoring processes needs to be tested during real-world construction projects to gain insights into its abilities to help in everyday surveillance tasks. Its desired influence on reducing cost and time overruns can be evaluated in a test run like that. Nonetheless, this master's thesis managed to produce a vital contribution to elevate the digitization level of the construction sector, with the hope that it will no longer stay at the bottom end of nearly every survey regarding this problem. After all, the benefits of digitized processes reported in other industry sectors can only help boost the stagnated productivity of the construction branch with innovations and research proposals as well as their application to real-world projects.

Appendix A

Plots for the end-to-end approach

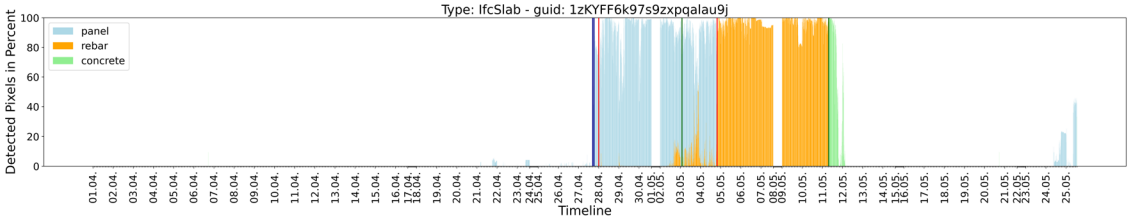


Figure A.1: Results for the slab with guid: 1zKYFF6k97s9zxpqalau9j

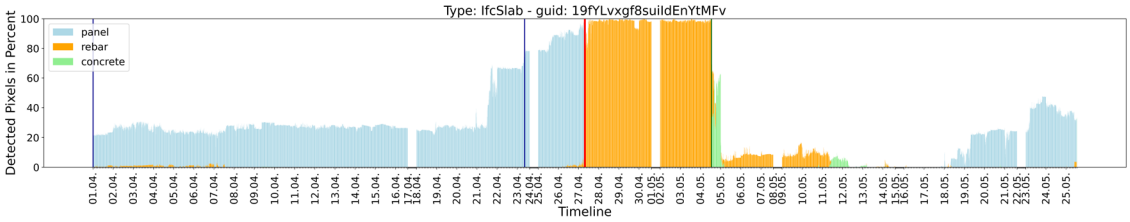


Figure A.2: Results for the slab with guid: 19fYLvxgf8suiIdEnYtMFv

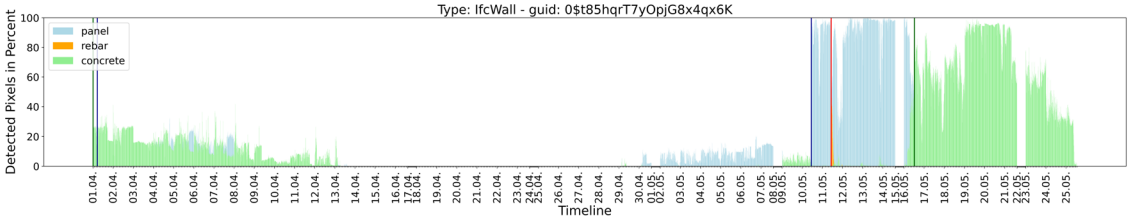


Figure A.3: Results for the wall with guid: 0$t85hqrT7yOpjG8x4qx6K

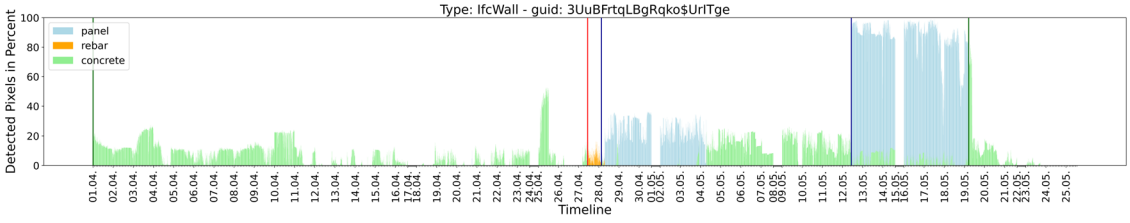


Figure A.4: Results for the wall with guid: 3UuBFrtqLBgRqko$UrlTge

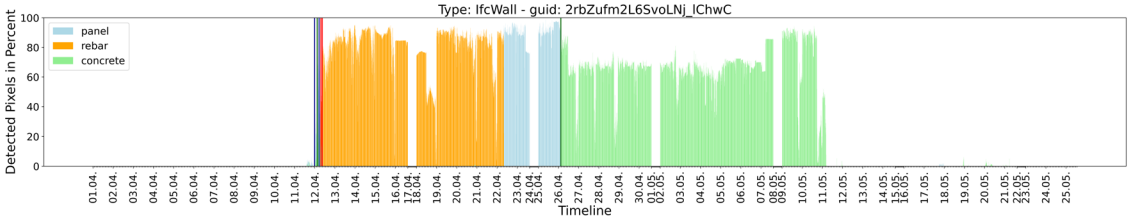


Figure A.5: Results for the wall with guid: 2rbZufm2L6SvoLNj_IChwC

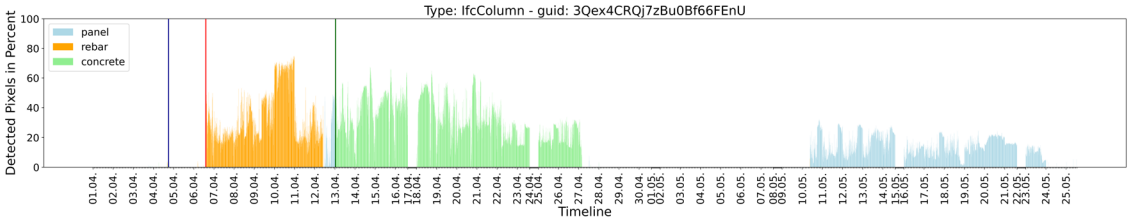


Figure A.6: Results for the column with guid: 3Qex4CRQj7zBu0Bf66FEnU

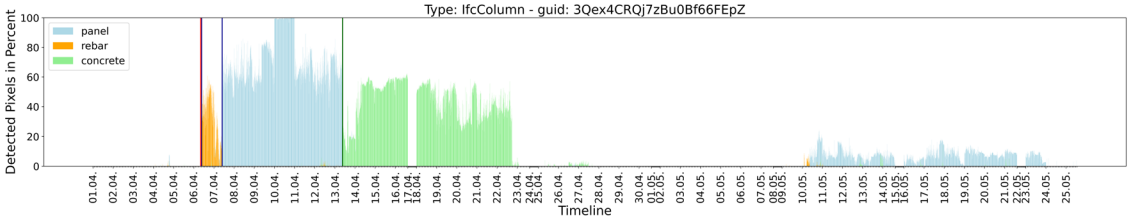


Figure A.7: Results for the column with guid: 3Qex4CRQj7zBu0Bf66FEpZ

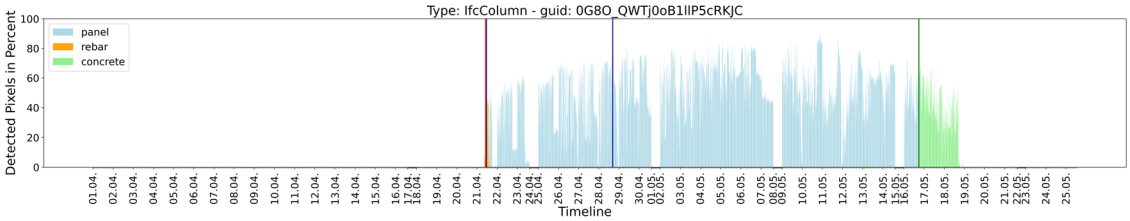


Figure A.8: Results for the column with guid: 0G8O_QWTj0oB1lIP5cRKJC

Bibliography

[1] A. A. Sezer, M. Thunberg, and B. Wernicke, “Digitalization index: Developing a model for assessing the degree of digitalization of construction projects,” *Journal of Construction Engineering and Management*, vol. 147, no. 10, 2021, ISSN: 0733-9364. DOI: 10.1061/(ASCE)CO.1943-7862.0002145.

[2] R. Agarwal, S. Chandrasekaran, and M. Sridhar, *Imagining construcion’s digital future*, Singapur, Jun. 2016. [Online]. Available: https://www.mckinsey.com/~/media/mckinsey/business%20functions/operations/our%20insights/imagining%20constructions%20digital%20future/imagining-constructions-digital-future.pdf (visited on 02/16/2024).

[3] K. Mostafa and T. Hegazy, “Review of image-based analysis and applications in construction,” *Automation in Construction*, vol. 122, 2021, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2020.103516.

[4] A. Sawhney and A. Knight, *Digitalisation in construction report 2023*, London, Jun. 2023.

[5] European Construction Sector Observatory, *Digitalisation in the construction sector: Analytical report*, Apr. 2021.

[6] M. Dlamini and R. Cumberlege, “The impact of cost overruns and delays in the construction business,” *IOP Conference Series: Earth and Environmental Science*, vol. 654, no. 1, 2021, ISSN: 1755-1307. DOI: 10.1088/1755-1315/654/1/012029.

[7] S. Sharma and A. K. Gupta, “Analysis of factors affecting cost and time overruns in construction projects,” in *Advances in Geotechnics and Structural Engineering*, ser. Lecture Notes in Civil Engineering, S. Kumar Shukla, S. N. Raman, B. Bhattacharjee, and J. Bhattacharjee, Eds., vol. 143, Singapore: Springer Singapore, 2021, pp. 55–63, ISBN: 978-981-33-6968-9. DOI: 10.1007/978-981-33-6969-6-6.

[8] M. Sami Ur Rehman, M. T. Shafiq, and F. Ullah, “Automated computer vision-based construction progress monitoring: A systematic review,” *Buildings*, vol. 12, no. 7, 2022. DOI: 10.3390/buildings12071037.

[9] A. Hannan Qureshi, W. Salah Alaloul, W. Kai Wing, *et al.*, “Automated progress monitoring technological model for construction projects,” *Ain Shams Engineering Journal*, vol. 14, no. 10, 2023, ISSN: 20904479. DOI: 10.1016/j.asej.2023.102165.

[10] R. Sacks, I. Brilakis, E. Pikas, H. S. Xie, and M. Girolami, “Construction with digital twin information systems,” *Data-Centric Engineering*, vol. 1, 2020. DOI: 10.1017/dce.2020.16.

[11] D.-G. J. Opoku, S. Perera, R. Osei-Kyei, and M. Rashidi, “Digital twin application in the construction industry: A literature review,” *Journal of Building Engineering*, vol. 40, 2021, ISSN: 23527102. DOI: 10.1016/j.jobe.2021.102726.

[12] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, “A survey of convolutional neural networks: Analysis, applications, and prospects,” *IEEE transactions on neural networks and learning systems*, vol. 33, no. 12, pp. 6999–7019, 2022. DOI: 10.1109/TNNLS.2021.3084827.

[13] K. O’Shea and R. Nash, *An introduction to convolutional neural networks*, Nov. 26, 2015. [Online]. Available: https://arxiv.org/pdf/1511.08458.pdf.

[14] F. Rosenblatt, *The perceptron: A probabilistic model for information storage and organization in the brain*, 1958.

[15] A. Bhardwaj, *What is a perceptron? basics of neural networks*, 2020. [Online]. Available: https://towardsdatascience.com/what-is-a-perceptron-basics-of-neural-networks-c4cfea20c590 (visited on 02/19/2024).

[16] J. Brwonlee, *How to choose an activation function for deep learning*, 2021. [Online]. Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/ (visited on 02/19/2024).

[17] S. P. Siregar and A. Wanto, “Analysis of artificial neural network accuracy using backpropagation algorithm in predicting process (forecasting),” *IJISTECH (International Journal Of Information System & Technology)*, vol. 1, no. 1, 2017. DOI: 10.30645/ijistech.v1i1.4.

[18] K. Nakamura and B.-W. Hong, “Adaptive weight decay for deep neural networks,” *IEEE Access*, vol. 7, 2019. DOI: 10.1109/ACCESS.2019.2937139.

[19] Y. Ding, “The impact of learning rate decay and periodical learning rate restart on artificial neural network,” in *2021 2nd International Conference on Artificial Intelligence in Electronics Engineering*, A. f. C. Machinery, Ed., New York NY: Association for Computing Machinery, 2021, pp. 6–14, ISBN: 9781450389273. DOI: 10.1145/3460268.3460270.

[20] Y. LeCun, C. Cortes, and C. Burges, *The mnist database of handwritten digits*. [Online]. Available: https://yann.lecun.com/exdb/mnist/.

[21] S. Ghosh, N. Das, I. Das, and U. Maulik, “Understanding deep learning techniques for image segmentation,” *ACM Computing Surveys*, vol. 52, no. 4, pp. 1–35, 2020, ISSN: 0360-0300. DOI: 10.1145/3329784.

[22] T. Liu, S. Fang, Y. Zhao, P. Wang, and J. Zhang, *Implementation of training convolutional neural networks*, Jun. 3, 2015. [Online]. Available: https://arxiv.org/pdf/1506.01195.pdf.

[23] Y. Guo, Y. Liu, T. Georgiou, and M. S. Lew, “A review of semantic segmentation using deep neural networks,” *International Journal of Multimedia Information Retrieval*, vol. 7, no. 2, pp. 87–93, 2018, ISSN: 2192-6611. DOI: 10.1007/s13735-017-0141-z.

[24] P. Wang, P. Chen, Y. Yuan, *et al.*, “Understanding convolution for semantic segmentation,” in *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2018, pp. 1451–1460, ISBN: 978-1-5386-4886-5. DOI: 10.1109/WACV.2018.00163.

[25] Z. Liu, Y. Lin, Y. Cao, *et al.*, *Swin transformer: Hierarchical vision transformer using shifted windows*, Mar. 25, 2021. [Online]. Available: https://arxiv.org/pdf/2103.14030.pdf.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, *et al.*, *An image is worth 16x16 words: Transformers for image recognition at scale*, Oct. 22, 2020. [Online]. Available: https://arxiv.org/pdf/2010.11929.pdf.

[27] W. Wang, J. Dai, Z. Chen, *et al.*, *Internimage: Exploring large-scale vision foundation models with deformable convolutions*, Nov. 10, 2022. [Online]. Available: http://arxiv.org/pdf/2211.05778v4.

[28] R. Ma, P. Tao, and H. Tang, "Optimizing data augmentation for semantic segmentation on small-scale dataset," in *Proceedings of the 2nd International Conference on Control and Computer Vision*, ser. ACM Digital Library, New York,NY,United States: Association for Computing Machinery, 2019, pp. 77–81, ISBN: 9781450363228. DOI: 10.1145/3341016.3341020.

[29] M. Cordts, M. Omran, S. Ramos, *et al.*, *The cityscapes dataset for semantic urban scene understanding*, Apr. 6, 2016. [Online]. Available: https://arxiv.org/pdf/1604.01685.pdf.

[30] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[31] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba, "Scene parsing through ade20k dataset," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017.

[32] H. Caesar, J. Uijlings, and V. Ferrari, *Coco-stuff: Thing and stuff classes in context*, 2018. arXiv: 1612.03716.

[33] X. Luo, H. Li, D. Cao, F. Dai, J. Seo, and S. Lee, "Recognizing diverse construction activities in site images via relevance networks of construction-related objects detected by convolutional neural networks," *Journal of Computing in Civil Engineering*, vol. 32, no. 3, 2018, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000756.

[34] Z. Wang, Y. Zhang, K. M. Mosalam, Y. Gao, and S. L. Huang, "Deep semantic segmentation for visual understanding on construction sites," *Computer-Aided Civil and Infrastructure Engineering*, vol. 37, no. 2, pp. 145–162, 2022, ISSN: 1467-8667. DOI: 10.1111/mice.12701.

[35] Z. Wang, Q. Zhang, B. Yang, *et al.*, "Vision-based framework for automatic progress monitoring of precast walls by using surveillance videos during the construction phase," *Journal of Computing in Civil Engineering*, vol. 35, no. 1, 2021, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000933.

[36] M. Golparvar-Fard, F. Peña-Mora, and S. Savarese, "Automated progress monitoring using unordered daily construction photographs and ifc-based building information models," *Journal of Computing in Civil Engineering*, vol. 29, no. 1, 2015, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000205.

[37] A. Pal, J. J. Lin, S.-H. Hsieh, and M. Golparvar-Fard, “Activity-level construction progress monitoring through semantic segmentation of 3d-informed orthographic images,” *Automation in Construction*, vol. 157, 2024, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2023.105157.

[38] S. El-Omari and O. Moselhi, “Integrating 3d laser scanning and photogrammetry for progress measurement of construction work,” *Automation in Construction*, vol. 18, no. 1, pp. 1–9, 2008, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2008.05.006.

[39] M. Bügler, A. Borrmann, G. Ogunmakin, P. A. Vela, and J. Teizer, “Fusion of photogrammetry and video analysis for productivity assessment of earthwork processes,” *Computer-Aided Civil and Infrastructure Engineering*, vol. 32, no. 2, pp. 107–123, 2017, ISSN: 1467-8667. DOI: 10.1111/mice.12235.

[40] J. Yang, M.-W. Park, P. A. Vela, and M. Golparvar-Fard, “Construction performance monitoring via still images, time-lapse photos, and video streams: Now, tomorrow, and the future,” *Advanced Engineering Informatics*, vol. 29, no. 2, pp. 211–224, 2015, ISSN: 1474-0346. DOI: 10.1016/j.aei.2015.01.011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/s1474034615000233.

[41] Q. Fang, H. Li, X. Luo, *et al.*, “A deep learning-based method for detecting non-certified work on construction sites,” *Advanced Engineering Informatics*, vol. 35, pp. 56–68, 2018, ISSN: 1474-0346. DOI: 10.1016/j.aei.2018.01.001.

[42] K. C. Laxman, N. Tabassum, L. Ai, C. Cole, and P. Ziehl, “Automated crack detection and crack depth prediction for reinforced concrete structures using deep learning,” *Construction and Building Materials*, vol. 370, 2023, ISSN: 09500618. DOI: 10.1016/j.conbuildmat.2023.130709.

[43] W. Fang, L. Ding, B. Zhong, P. E. Love, and H. Luo, “Automated detection of workers and heavy equipment on construction sites: A convolutional neural network approach,” *Advanced Engineering Informatics*, vol. 37, pp. 139–149, 2018, ISSN: 1474-0346. DOI: 10.1016/j.aei.2018.05.003.

[44] J. Schlenger, F. Pfitzner, A. Braun, S. Vilgertshofer, and A. Borrmann, “Digitaler zwilling baustelle – baustellenüberwachung zur automatisierten zeit– und kostenkontrolle,” *Bautechnik*, vol. 100, no. 4, pp. 190–197, 2023, ISSN: 0932-8351. DOI: 10.1002/bate.202300005.

[45] F. Pfitzner, A. Braun, and A. Borrmann, “Object detection-based knowledge graph creation: Enabling insight into construction processes,” in *Computing in Civil Engineering 2023*, Y. Turkan, J. Louis, F. Leite, and S. Ergan, Eds., Reston: American Society of Civil Engineers, 2024, pp. 186–193, ISBN: 9780784485224. DOI: 10.1061/9780784485224.023.

[46] E. Hjelseth, S. F. Sujan, and R. J. Scherer, Eds., *ECPPM 2022 - eWork and eBusiness in architecture, engineering and construction: Proceedings of the 14th European Conference on Product & Process Modelling (ECPPM 2022), 14-16 Spetember 2022, Trondheim, Norway*, 1st. Boca Raton: CRC PRESS, 2023, ISBN: 1000925544.

DOI: 10.1201/9781003354222. [Online]. Available: https://www.taylorfrancis.com/books/9781003354222.

[47] H. H. Hosamo and M. H. Hosamo, “Digital twin technology for bridge maintenance using 3d laser scanning: A review,” *Advances in Civil Engineering*, vol. 2022, pp. 1–15, 2022, ISSN: 1687-8086. DOI: 10.1155/2022/2194949.

[48] M. Kamari and Y. Ham, “Ai-based risk assessment for construction site disaster preparedness through deep learning-based digital twinning,” *Automation in Construction*, vol. 134, 2022, ISSN: 0926-5805. DOI: 10.1016/j.autcon.2021.104091.

[49] T. Kristensen and G. Uva, *IFC openshell*, https://github.com/IfcOpenShell/IfcOpenShell, 2024. (visited on 02/22/2024).

[50] *Industry foundation classes*, BuildingSMART International, 2022. [Online]. Available: https://www.buildingsmart.org/standards/ifc/ (visited on 02/22/2024).

[51] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*, 2. Auflage. Harlow: Pearson Education, 2012, ISBN: 0-273-76414-4.

[52] E. Bayro-Corrochano and B. Rosenhahn, “A geometric approach for the analysis and computation of the intrinsic camera parameters,” *Pattern Recognition*, vol. 35, no. 1, pp. 169–186, 2002, ISSN: 00313203. DOI: 10.1016/S0031-3203(00)00182-5.

[53] G. Bradski and A. Kaehler, *OpenCV – open source computer vision library*, 2000. [Online]. Available: https://opencv.org/ (visited on 02/22/2024).

[54] F. C. Collins, F. Pfitzner, and J. Schlenger, Eds., *Scalable construction monitoring for an as-performed progress documentation across time*, 2022.

[55] B. Sekachev, N. Manovich, M. Zhiltsov, *et al.*, *Opencv/cvat: V1.1.0*, version v1.1.0, Aug. 2020. DOI: 10.5281/zenodo.4009388. [Online]. Available: https://doi.org/10.5281/zenodo.4009388.

[56] Contributors, *MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark*, https://github.com/open-mmlab/mmsegmentation, 2020.

[57] W. Wang, J. Dai, Z. Chen, *et al.*, “Internimage: Exploring large-scale vision foundation models with deformable convolutions,” *arXiv preprint arXiv:2211.05778*, 2022.

[58] Z. Chen, Y. Duan, W. Wang, *et al.*, “Vision transformer adapter for dense predictions,” *arXiv preprint arXiv:2205.08534*, 2022.

[59] J. He, Z. Deng, L. Zhou, Y. Wang, and Y. Qiao, Eds., *Adaptive Pyramid Context Network for Semantic Segmentation*, 2019.

[60] K. He, X. Zhang, S. Ren, and J. Sun, *Deep residual learning for image recognition*, Dec. 10, 2015. [Online]. Available: https://arxiv.org/pdf/1512.03385.pdf.

[61] J. Fu, J. Liu, H. Tian, *et al.*, *Dual attention network for scene segmentation*, Sep. 9, 2018. [Online]. Available: https://arxiv.org/pdf/1809.02983.pdf.

[62] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, *Encoder-decoder with atrous separable convolution for semantic image segmentation*, Feb. 7, 2018. [Online]. Available: http://arxiv.org/pdf/1802.02611v3.

[63] H. Zhang, C. Wu, Z. Zhang, *et al.*, *Resnest: Split-attention networks*, Apr. 19, 2020. [Online]. Available: https://arxiv.org/pdf/2004.08955.pdf.

[64] R. Ranftl, A. Bochkovskiy, and V. Koltun, *Vision transformers for dense prediction*, Mar. 24, 2021. [Online]. Available: https://arxiv.org/pdf/2103.13413.pdf.

[65] X. Li, Z. Zhong, J. Wu, Y. Yang, Z. Lin, and H. Liu, *Expectation-maximization attention networks for semantic segmentation*, Jul. 31, 2019. [Online]. Available: https://arxiv.org/pdf/1907.13426.pdf.

[66] H. Zhang, K. Dana, J. Shi, *et al.*, *Context encoding for semantic segmentation*, Mar. 23, 2018. [Online]. Available: https://arxiv.org/pdf/1803.08904.pdf.

[67] H. Wu, J. Zhang, K. Huang, K. Liang, and Y. Yu, *Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation*, Mar. 28, 2019. [Online]. Available: https://arxiv.org/pdf/1903.11816.pdf.

[68] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, *Pyramid scene parsing network*, Dec. 4, 2016. [Online]. Available: http://arxiv.org/pdf/1612.01105v2.

[69] E. Shelhamer, J. Long, and T. Darrell, *Fully convolutional networks for semantic segmentation*, May 20, 2016. [Online]. Available: http://arxiv.org/pdf/1605.06211v1.

[70] J. Wang, K. Sun, T. Cheng, *et al.*, *Deep high-resolution representation learning for visual recognition*, Aug. 20, 2019. [Online]. Available: https://arxiv.org/pdf/1908.07919.pdf.

[71] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection*, Dec. 9, 2016. [Online]. Available: https://arxiv.org/pdf/1612.03144.pdf.

[72] X. Chu, Z. Tian, Y. Wang, *et al.*, *Twins: Revisiting the design of spatial attention in vision transformers*, Apr. 28, 2021. [Online]. Available: https://arxiv.org/pdf/2104.13840.pdf.

[73] L. Huang, Y. Yuan, J. Guo, C. Zhang, X. Chen, and J. Wang, *Interlaced sparse self-attention for semantic segmentation*, Jul. 29, 2019. [Online]. Available: https://arxiv.org/pdf/1907.12273.pdf.

[74] W. Zhang, J. Pang, K. Chen, and C. C. Loy, *K-net: Towards unified image segmentation*, Jun. 28, 2021. [Online]. Available: https://arxiv.org/pdf/2106.14855.pdf.

[75] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, *Masked-attention mask transformer for universal image segmentation*, Dec. 2, 2021. [Online]. Available: https://arxiv.org/pdf/2112.01527.pdf.

[76] Y. Yuan, X. Chen, X. Chen, and J. Wang, "Segmentation transformer: Object-contextual representations for semantic segmentation," *ECCV*, [Online]. Available: https://arxiv.org/pdf/1909.11065.pdf.

[77] J. Xu, Z. Xiong, and S. P. Bhattacharyya, *Pidnet: A real-time semantic segmentation network inspired by pid controllers*, Jun. 5, 2022. [Online]. Available: https://arxiv.org/pdf/2206.02066.pdf.

[78] H. Zhao, Zhang, Yi and Liu, Shu, J. Shi, C. C. Loy, D. Lin, and J. Jia, Eds., *PSANet: Point-wise Spatial Attention Network for Scene Parsing*, 2018.

[79] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez, and P. Luo, *Segformer: Simple and efficient design for semantic segmentation with transformers*, May 31, 2021. [Online]. Available: https://arxiv.org/pdf/2105.15203.pdf.

[80] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, *Segmenter: Transformer for semantic segmentation*, May 12, 2021. [Online]. Available: https://arxiv.org/pdf/2105.05633.pdf.

[81] M.-H. Guo, C.-Z. Lu, Q. Hou, Z. Liu, M.-M. Cheng, and S.-M. Hu, *Segnext: Rethinking convolutional attention design for semantic segmentation*, Sep. 18, 2022. [Online]. Available: https://arxiv.org/pdf/2209.08575.pdf.

[82] S. Zheng, J. Lu, H. Zhao, *et al.*, *Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers*, Dec. 31, 2020. [Online]. Available: https://arxiv.org/pdf/2012.15840.pdf.

[83] T. Xiao, Y. Liu, B. Zhou, Y. Jiang, and J. Sun, *Unified perceptual parsing for scene understanding*, Jul. 26, 2018. [Online]. Available: https://arxiv.org/pdf/1807.10221.pdf.

[84] H. Bao, L. Dong, S. Piao, and F. Wei, *Beit: Bert pre-training of image transformers*, Jun. 15, 2021. [Online]. Available: https://arxiv.org/pdf/2106.08254.pdf.

[85] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, *A convnet for the 2020s*, Jan. 10, 2022. [Online]. Available: https://arxiv.org/pdf/2201.03545.pdf.

[86] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, *Masked autoencoders are scalable vision learners*, Nov. 11, 2021. [Online]. Available: https://arxiv.org/pdf/2111.06377.pdf.

[87] A. Braun, "Automated bim-based construction progress monitoring by processing and matching semantic and geometric data," Dissertation, Technische Universität München, München, 2020.

[88] L. Cheng, S. Chen, X. Liu, *et al.*, "Registration of laser scanning point clouds: A review," *Sensors (Basel, Switzerland)*, vol. 18, no. 5, 2018. DOI: 10.3390/s18051641.

[89] M. Golparvar-Fard, F. Peña-Mora, and S. Savarese, "Automated progress monitoring using unordered daily construction photographs and ifc-based building information models," *Journal of Computing in Civil Engineering*, vol. 29, no. 1, 2015, ISSN: 0887-3801. DOI: 10.1061/(ASCE)CP.1943-5487.0000205.

Declaration

I hereby affirm that I have independently written the thesis submitted by me and have not used any sources or aids other than those indicated.

Location, Date, Signature