5th Conference on Production Systems and Logistics

Method For Creating A Control Cabinet Model With Realistic Wires

Stefanie Bartelt[1], Matthias Bartelt[2], Bernd Kuhlenkötter[1]
[1]*Chair of Production Systems, Ruhr University Bochum, Bochum, Germany*
[2]*SCHUBS GmbH, Hameln, Germany*

Abstract

During the assembly of a control cabinet, a major time-consuming step is the wiring of the included components. Hence, automating this step will noticeably reduce production costs. According to the planning, wires are routed through wire ducts and connected to components. While a comprehensive digital twin can be computed for the included components, this twin is missing a proper modelling of the connecting wires. For these, only a rough route through the wire ducts is given. However, a physically plausible model is an important prerequisite to perform reliable path planning for automated assembly. The paper addresses this need for accurate wire path computation during automated cabinet assembly and introduces a method to compute realistic wire paths through the wire ducts. Different models with and without a fixed wire length are presented and compared. An evolutionary algorithm optimizes the corresponding variables of the models. As described, both approaches yield valid paths, although the fixed length model appears to be able to compute more realistic paths.

Keywords

Control Cabinet Assembly; Wire Modeling; Path Planning; Simulation; Automated Assembly

1. Introduction

Control cabinets are very common in industry. They contain several components, e.g., programmable logic controllers, power devices, or terminal blocks. The kind of components correspond to the functions that the control cabinet must fulfill. Therefore, there is a high variance in the design of control cabinets. Furthermore, the components are connected by wire that are routed through wire ducts. Typically, there are about 300 to 400 components and about 500 wires in a cabinet. The production of a cabinet starts with the definition of functions and the selection of required components. Using an appropriate software, the physical design of the cabinet is created, and the components are connected as required. With this planning, the assembly of the cabinet can be done. [1] gives a detailed overview on the setup and assembly procedure. However, most assembly steps are carried out manually because effects such as poor data quality, low lot size, or a lack of component design suitable for automation hinder automation. The components are usually not designed for an automated assembly. This causes a high level of complexity regarding the required technology. Especially the wiring, which consists of assembly and routing, is complex in terms of automation [2]. Furthermore, there is no holistic digital model of the control cabinet in a system-neutral exchange format that can be used is the software tools. As a result, important information to automate the process steps are missing and an appropriate solution is both time-consuming and cost intensive. However, solutions for automating the manufacturing process must be cost-effective, which is a challenging task, particularly for make-to-order production. There are two promising approaches to increase the automation level – reduction of complexity and a comprehensive data model.

DOI: https://doi.org/10.15488/15281
ISSN: 2701-6277

The reduction of complexity refers to the shape and connections of the components. Nowadays, both shapes and connectors are not designed for automation. Additionally, there is a high variance among manufacturers, which obstructs automated assembly. In terms of connections, there are already good variants, such as push-in technology [3]. Nevertheless, optimizing components is the responsibility of the manufacturers.

The second approach is the creation of a comprehensive data model of the control cabinet. Especially for individual projects, it is important that the data model can be derived solely from the planning tool. Furthermore, the model must provide information in a system-neutral format to support the development of automated manufacturing steps. Currently, the information that can be derived from the design is a descriptive 3D model of the cabinet including the position, size, and hierarchical structure. Additionally, wiring information can be derived in a second step. The wiring information can include the connected components, the wire ducts for the routing, and information to the wire like color or cross section. Even if the accessible information provides a good model, important supplementary information is still missing for more complex tasks, such as the calculation of the degree of filling of wire ducts, the distribution probability of wires within the ducts, or the automation of assembly steps.

The paper focuses on the latter approach and presents a method to create a comprehensive data model. It is structured as follows. The next section gives an overview of the accessible information as well as a general method to enrich the model with further information. By combining different sources, the basis of a digital twin for a control cabinet is created. Both the data model as well as the data sources providing relevant content to the model are discussed. As the chapter reveals, a realistic model for the wires in the control cabinet is missing. Section 3 elaborates on this issue and presents an approach for a proper modeling. Section 4 discusses the results and ends with a conclusion of the presented work.

2. Control cabinet model

As discussed above, a comprehensive data model of the control cabinet supports the optimization of the manufacturing process. The generation of the model should be accomplished with as little effort as possible, and the creation process should be automated as much as possible. This is particularly important for control cabinets that are only manufactured once.

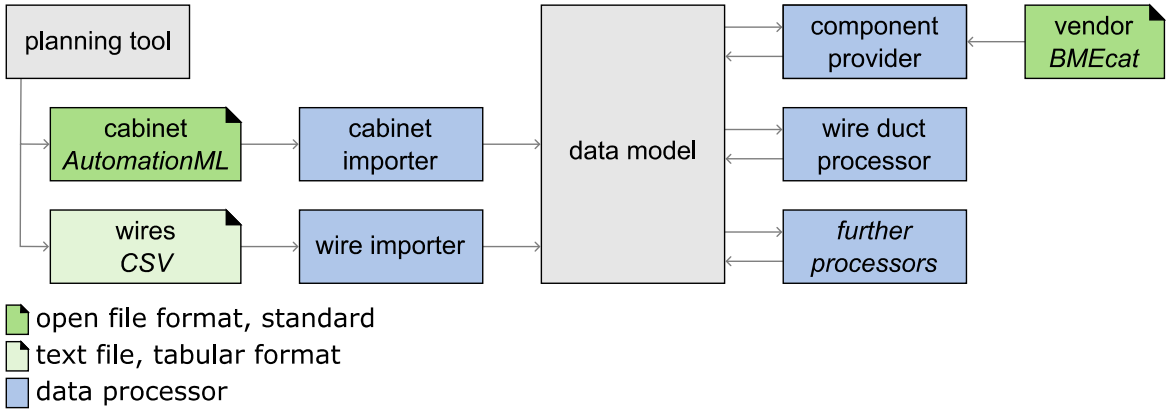


Figure 1: Concept for the creation and enrichment of the data model. The arrows indicate the flow of information.

In general, the effort to create the model can be divided into two tasks. The first task is to create all cabinet-specific information, which varies for each cabinet. The second task is the creation of component-specific information. This information is static for each component and must only be created once. Furthermore, it is possible to enrich the model by applying different algorithms or rules. Figure 1 depicts the corresponding concept to create a data model. All data sources are connected to the model via appropriate processors, while the planning tool is the only provider for cabinet-specific information.

2.1 Data sources

In general, the planning tool might provide a sufficient detailed set of cabinet-related as well as component-related information to build the data model. Nevertheless, there are several reasons, why the provided information is insufficient. For instance, the user may have planned the cabinet only partially, or the export of the tool lacks important details. Although the completeness of the exported data depends on the software used, it is most likely that more sources are required to achieve a holistic data model. While the planning tool must provide all information related to the specific setup of the cabinet, component-related information may can be acquired from other sources, i.e., the manufacturer of the components. Furthermore, there might be some information that is neither provided by the planning tool nor by the manufacturer in a machine-readable way. A further issue in collecting data are the data formats of the corresponding sources. Although there exist many standards for data exchange [4], proprietary data formats are still common.

Although proprietary data might be added to the model, e.g., by implementing adapters that transform the proprietary information to an appropriate exchange format as described in [5]. Nevertheless, standardized open file formats are preferred to achieve a vendor-independent solution. An open exchange format, which has a growing importance, is the AutomationML format [6]. Besides the exchange of arbitrary data, it offers the possibility to exchange component specific information in a standardized way, as described in [7]. As shown in Figure 1, further product-specific information can be added to the model by using the BMEcat format [8]. In addition, other data formats can be easily integrated by implementing appropriate processors as described in the next section.

The planning tool generates two files. The first one exports the cabinet as AutomationML document. This document contains a list of components as well as the hierarchical structure of the cabinet. Each component has a unique identifier that refers to the type of the component, i.e., if a certain component is assembled multiple times, all have the same identifier. Furthermore, a component has information about its local transformation, a physical extent, and meta-information. The latter one includes, for instance, the manufacturer name, or the component's function. The second file generated by the planning tool is a document including wiring information. Although it is exported in an open file format, the export is tool specific and must by handled by a specialized data processor.

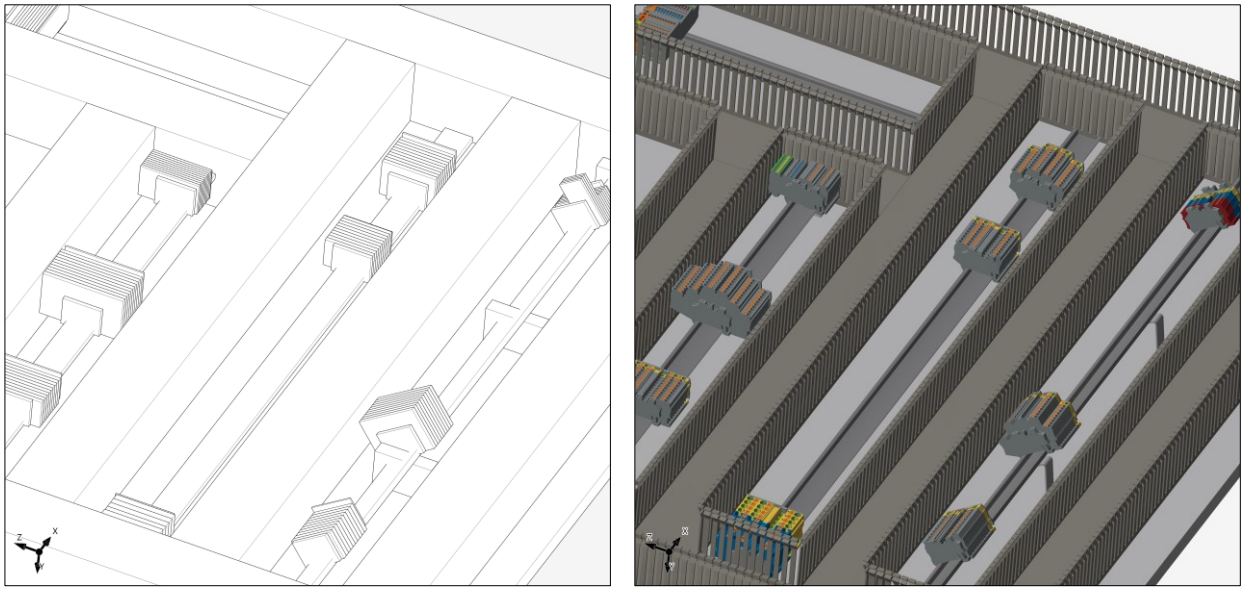
Further components-specific data is added via the BMEcat format. This data comprises the geometrical representation and attachment points that define where other components may be attached. For example, wires are mounted between two attachment points, or components can be assembled onto a DIN rail.

2.2 Data processors

Data processors can directly interact with the data model. They can access the stored information and they can add further information to the data model. With this, information provided by data sources is included, but a processor can also compute information that is not provided by any source. By successively running different processors, a holistic data model is achieved. In [9], a set of data processors are presented to compute the main cabinet model via a combination of several successive processors. They were required, because the planning tool only provides proprietary data as well as a small subset of the required information in a spreadsheet format. By applying different rules, missing information to build the model were computed. However, the model is consistent with the design only if the rules applied were followed in the planning. This results in additional requirements for the planning, making it more time-consuming. With the presented approach, these processors are not required anymore, which simplifies the planning process and enables the automatic generation of a consistent model.

As depicted in Figure 1, there are four processors right now: Two of them, the cabinet and wire importer, are importing data into the model. By processing the data provided by the planning tool, corresponding

components are created in the cabinet model. Figure 2 (a) depicts the state of the model after running both processors. As shown, all components are placed correctly, but only their extent is defined and no real geometrical representation. Furthermore, wires cannot be connected, as there is no information about attachments. The component provider processor increases this detail level of the model by implementing a local database with component-specific information. This database is filled by importing BMEcat files or manually, if no appropriate files are provided by the manufacturer. By traversing all created components, each component is enriched by its geometrical information and by further information, especially attachment points for wires. The wire duct processor improves the detail level without using external data. This processor searches for all wire ducts and creates breakouts at positions where other wire ducts are adjacent.



(a) (b)

Figure 2: Visualization of the model after importing the information provided by the planning tool (a) and after enrichment by subsequent processors (b).

Figure 2 (b) depicts the model after running all four processors. Not shown in the picture are the specified wires. For these, only the components that the cable connects as well as the wire ducts through which the cable is to be routed are defined. The integration of the wires into the model is discussed in the next section.

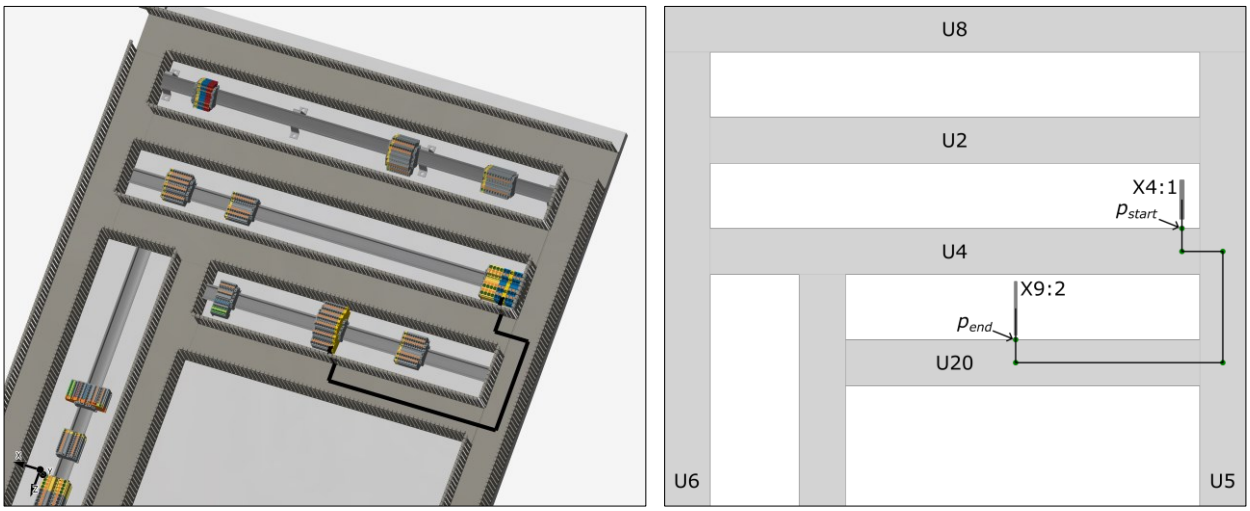
3. Modeling wires

This section discusses wire modeling methods. Figure 3 depicts an example of a routed wire. The data model specifies that the wire connects the components with item designation X1:4 and X9:2. Furthermore, it is specified that the length of the wire is 834 mm and that the wire should pass wire ducts U4, U5, and U20. For manual assembly, this information is sufficient. Nevertheless, to automate processes, a more detailed routing is required. For this, there are some boundary conditions that must be satisfied:

- Both start and end point are fixed. For routing, these points are placed at the feedthroughs to the wire ducts. These can be assumed to behave like a fixed point, and they are referred in the following as p_{start} and p_{end}.
- The length l of the wire between p_{start} and p_{end} is determined by the planning process. The wire is produced using this length before assembly. Hence, the wire length is constant for the path calculation.
- The wire between p_{start} and p_{end} must stay within any of the given wire ducts.

Paths that meet these conditions are referred to as valid. Paths that sufficiently match the physical properties of the wire are referred to as realistic.

The problem of finding a proper route for the wire correlates to the path planning for mobile systems. To solve the path finding problem for mobile systems, various algorithms have been developed [10–12]. However, path planning algorithms focus on finding the optimized path, e.g., the shortest one, and it is not possible to force a fixed length as required for the given problem. Of course, one can search a path and skip it if the length does not match the given length, but with this, the algorithms do not converge anymore, and a solution cannot be found. Another approach that is often used in simulation is an impulse-based model. The wire is approximated by small rigid segments that are connected by joints. For each simulation step, constant forces are assumed in the multi-body system (although these forces may be different in other time steps). By summing up forces, impulses can be computed that changes the velocities and angular moments of the single segment [13]. Further on, three different approaches are presented.



(a) (b)

Figure 3: Example of a routed wire from component X4:1 to component X9:2. The shown route is received by a linear routing as described in section 3.1.

3.1 Linear routing

The linear routing is a very simple algorithm. Starting at p_{start}, a linear path to the center of the wire duct with minimal length is created. From the end point, another linear path to the next duct with the same method is created. These steps are performed until the routing is complete. Finally, the end of the path is connected to p_{end}. Figure 3 shows the result of the routing. Although the algorithm is quite fast, it is not guaranteed that the wire length will match the specified length.

3.2 Minimal energy approach

The second approach to route is to compute the energy of the wire and minimize the energy. To find a global minimum, algorithms such as randomized minimization by Monte Carlo simulation, gradient descent method, or heuristic algorithms (cf. e.g. [14]) can be used. For the given problem, the wire with specified length l is considered as a sequence of n linear segments. To simplify the computation, all segments are assumed to be in the same plane. Each segment has a nominal length of $l_0 = l/n$. The first and last segment are connected to p_{start} and p_{end} respectively. The rotation of a segment i relative to its precursor is described by an angle α_i. Furthermore, there are angles between the first/last segment and the wire duct. With this, the bending energy E_B of a wire is

$$E_B = \sum_{i=0}^{n} \tfrac{1}{2} k_B \cdot \alpha_i^2. \quad (1)$$

k_B is the moment of direction. It depends on type of wire and is assumed to be constant. The requirement that the wires must be inside the wire ducts is achieved with an additional energy E_W. If a segment is within any wire duct, this energy is 0. Otherwise, a large value is set to move the segment into a wire duct. To find a global minimum, an evolutionary algorithms, e.g. as described in [15], is implemented. Thereby, the wire segments are assumed to be a straight line, and they can move freely as far as they are not. With this, any bending of the wire will create a force in the opposite direction.

In the following, two different models to compute a realistic wire path are presented. An evolutionary algorithm is implemented to find parameters resulting in minimal energy. For this, a population of wires is created with an initial path. This path is referred as initialized path. The path as computed by the linear routing is referred as planned path. It will be used to optimize the initial population. The population evolves into a new epoch by computing a set of best wires. With this set, new wires are computed until the initial size of the population is reached. The algorithm is implemented in C# and includes as dedicated processor in the data model as described in the previous section.

The first model does not force a certain length of the wire. It rather assumes a string between two successive joints. This results into a new energy E_L with

$$E_L = \sum_{i=1}^{n} \tfrac{1}{2} k_L \cdot (l_i - l_0)^2. \qquad (2)$$

k_L is the (virtual) spring constant. The initial population of 100 contains wires with arbitrary segments. In each epoch, 70% of the wires with lowest energy survive. Until the population reaches the initial size, an offspring is generated with two parents by selecting an arbitrary value p with $1 \leq p \leq n+1$. Segments 1 to $p-1$ are taken from the first parent, and segments from p to n are taken from the second. $p = 0$ is a copy of the second parent, and $p = n+1$ is a copy of the first parent. Both parents are determined via a roulette wheel selection from the last population. For this selection, the energy of the wire is anti-proportional to the probability to be selected.

In a next step, wires are mutated with a probability of 0.1. If a wire mutates, an arbitrary segment is selected. Furthermore, m subsequent segments are selected. m is normal distributed with a standard deviation of 2.0.

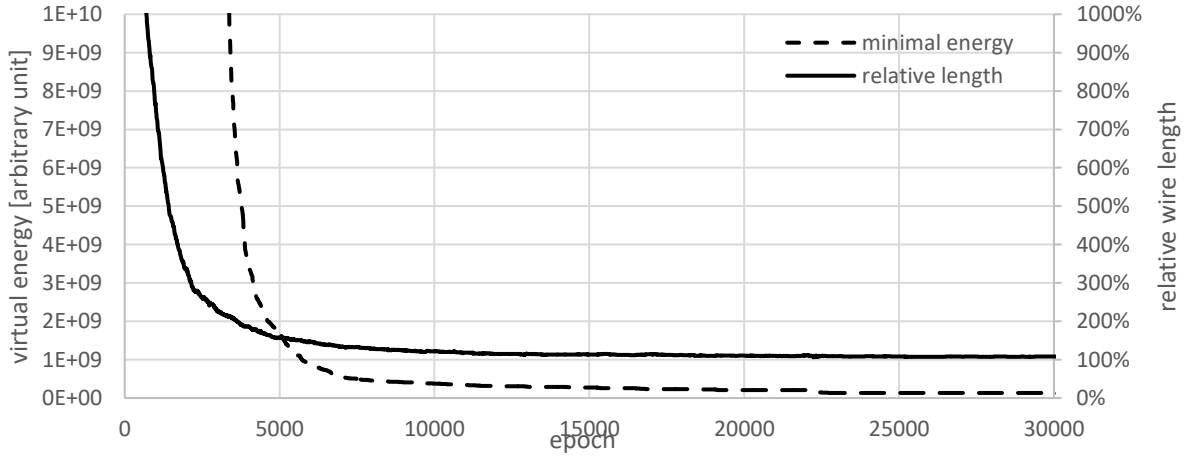
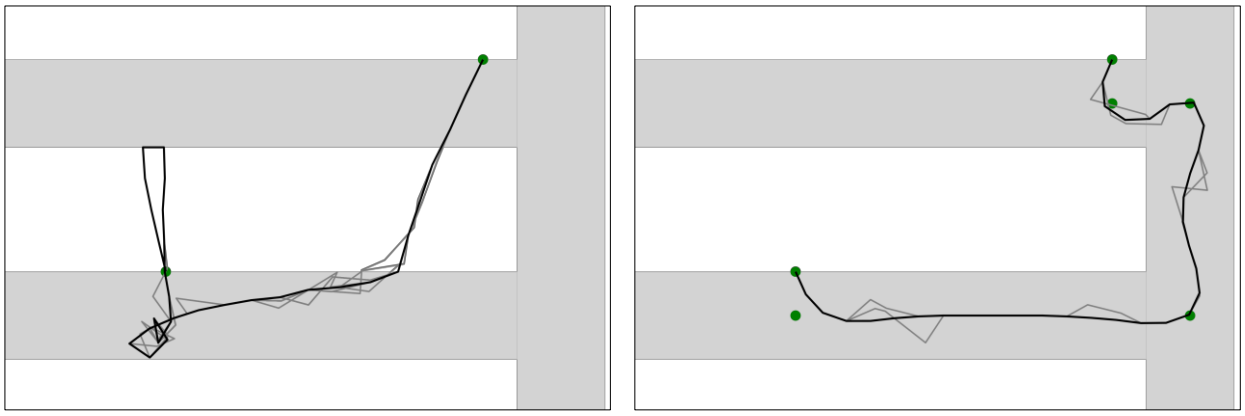


Figure 4: Minimal energy of the population (dotted line) and the corresponding wire length (solid line). The length of the wire is given as relative length compared to the planned length.

In each epoch, the wire with the lowest energy is selected as best approach. For this wire, also the length is computed. Figure 4 depicts the energy as well as the length over 30 000 epochs. As shown, the algorithm converges. Also, the length is about the specified length. However, there are several local minima, and it is not ensured that the algorithm converges to a valid wire. Figure 5 (a) shows the corresponding computed path. Although a local minimum is reached, the wire is obviously not valid. Additionally, it was found that

loops in a wire are very likely to remain. To improve the results, the wires are not generated randomly, but they are created along the planned. Due to this initialization, similar results are obtained even with different starting conditions. Figure 5 (b) presents the resulting wire. Although valid solutions are computed, these solutions are very similar to the initialized path.

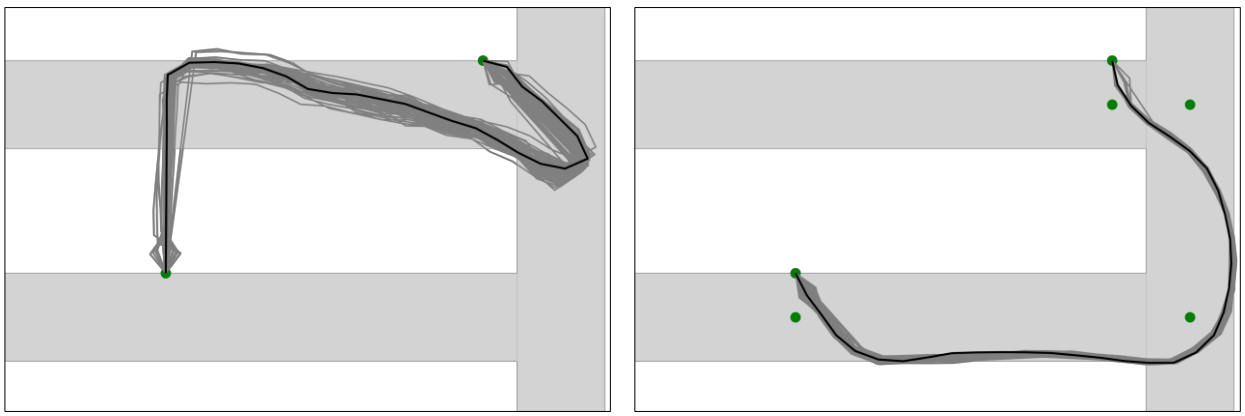


(a) (b)

Figure 5: Result with a randomly initialized set of wires (a) and when initializing the set along the planned path (b). The black wire indicates the wire with the lowest energy, while the other wires are drawn gray. However, most wires are similar and not visible in the figure.

The second model implements a fixed length. For this, segments with constant length l_0 are connected. Due to the fixed end points, the segments are not arbitrary, but each segment must be chosen in such way that subsequent segments can reach the end point. 50 wires are generated from the start to the end point and 50 wires are generated vice versa to improve the routing. The combination of two wires to a new offspring must also be changed to meet the fixed segment length. As before, segments are combined from both parents. However, the segments of the second parent are used, but they must be adjusted so that the wire reaches the end point.

Like the recombination, the mutation must be changed accordingly. An arbitrary segment is changed randomly. Either all subsequent or all preceding segments are then changed only to the extent that the end point is still reached. Figure 6 (a) depicts the result of the computation. Like the first model, the model converges but does not yield a valid solution. To improve the results, the intermediate points of the given planned path are used. With this, random sub-paths are created between two consecutive points. The sub-paths are combined to the wire, whereby the algorithm ensures that the number of segments are the same for all wires. As before, the paths are generated in both directions, from start to end point and from end to start point. Figure 6 (b) presents the resulting wire.



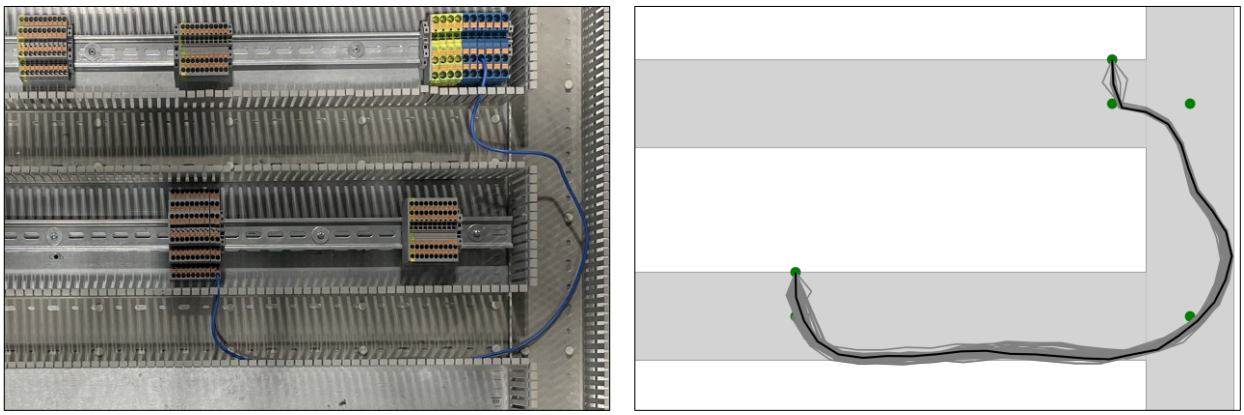
(a) (b)

Figure 6: Result with a randomly initialized set of wires (a) and when initializing the set along the intermediate points of the planned path (b).

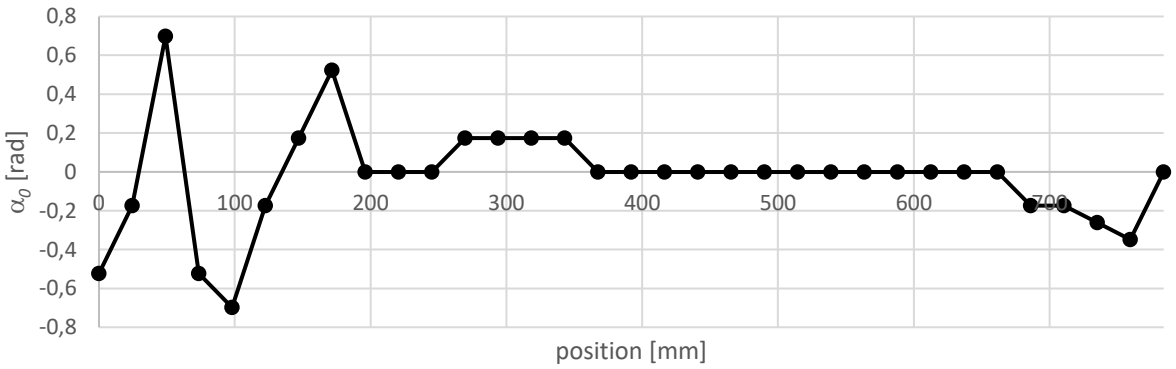
Wires are assumed to be unbent before assembly. However, the bending energy E_B can be modified to

$$E_B = \sum_{i=0}^{n} k_B \cdot \left(\alpha_i - \alpha_{0,\mathrm{i}}\right)^2. \tag{3}$$

The parameters $\alpha_{0,i}$ indicate a preferred direction of the wire at the corresponding position. Although the parameters need not be equal in general, it is assumed that a constant value $\alpha_{0,i} = \alpha_0$ can be achieved via a suitable wire pusher mechanism. With this, bends in the wire can be included in the calculation. As a result, is should be possible to find parameters $\alpha_{0,i}$ that leads to a computed wire matching a real wire. Figure 7 (a) shows the real mounting plate that corresponds to the configuration shown in Figure 3. The length of the wire matches to the length of the wire used for the computations. Based on this image, parameters $\alpha_{0,i}$ are changed stepwise by 5°. For the computation of the path, the fixed-length model with initialization along the planned path is used. For each parameter set, the result is compared to the real image. However, this comparison is made only visual, i.e., there is no measure applied. Figure 7 (b) shows the best-found combination. The corresponding parameters are given in Figure 7 (c).



(a) (b)



(c)

Figure 7: Photo of the real setup (a) and the result of computation (b) that is using the pre-bend angles α_0. The used values for the angles are shown in (c).

3.3 Physics simulation

The third approach to compute a wire model is the usage of a physics engine. The wire is modeled as kinematic chain with small segments with a length of 2 mm. The initial shape of the wire is set to the linear routed path that is computed as described in section 3.1. The physical computation is done by the BEPUphysics v2 library [16]. Figure 8 shows the resulting wire. Although the shape of the wire is fine, it is quite difficult to achieve a proper design of the wire. Especially defining physical properties such as the stiffness of the wire is complicated. Depending on the selected parameters, instabilities are very likely to occur, resulting in invalid wires.

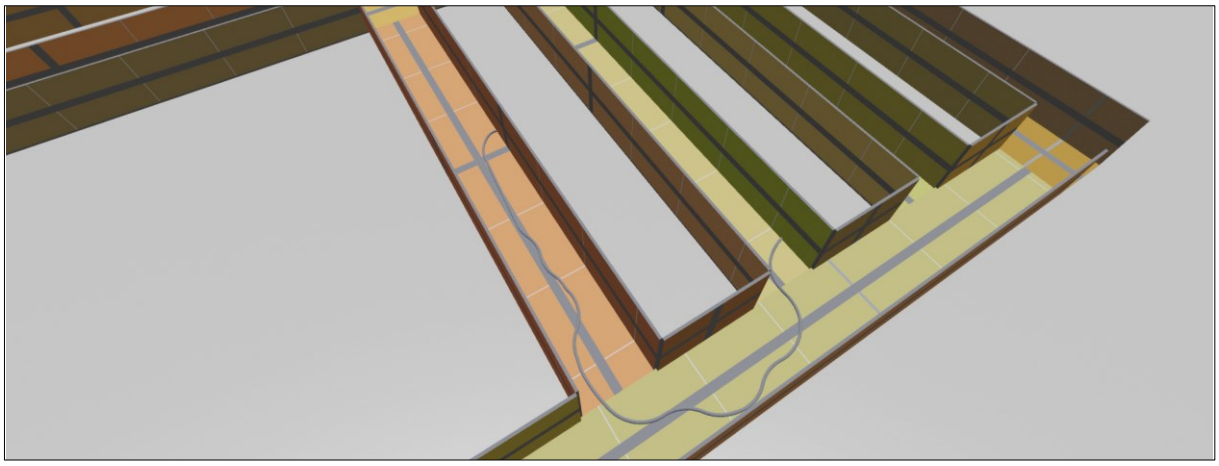


Figure 8: Simulation of the wire using a physics engine.

4. Conclusion and future work

In the paper, an approach to combine different data sources to build a comprehensive model of a control cabinet was presented. The resulting model is an important prerequisite, for example, to generate a digital twin that can be used to realize path planning for automatic wiring. After combining external data sources, data processors are utilized, particularly to compute proper wire models. For the latter, three different methods were implemented. The linear routed path is a fast and robust algorithm. The resulting path does not match a real wire, but it might be sufficient for simple tasks. The minimal energy and the physic engine approach can compute realistic wires. In comparison with a real assembled wire, pre-bend parameters can be determined that result in a good match to the real wire. Nevertheless, currently only a single wire is considered. An extension of the algorithm from the calculation of wires in the plane to a computation in three dimensions, as well as the treatment of multiple wires, will be done in subsequent works. The utilization of a physics engine is suitable for a dynamic simulation. Although multiple wires can be simulated together, their behavior in terms of stiffness is not realistic right now.

In future work, real assembled wires will be digitized, and a corresponding path will be computed. With this, a measure can be defined, and the deviation of the model with the real wire can be computed. By mounting the wire several time, the variance of the assembly process will be analyzed. Additionally, parameters $\alpha_{0,i}$ can be computed automatically by minimizing the deviation to the real wire. As soon as the assembly process is automated with a wire pusher, the assumption that $\alpha_{0,i}$ is influenced by the wire reel and the pusher mechanism can be validated. The findings will be used to improve the presented models.

With respect to a future simulation of the assembly, the models will be examined for their suitability. In this context, also further models, e.g., an impulse based approach, will be implemented and compared to the presented ones.

Acknowledgements

This Project is supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK) on the basis of a decision by the German Bundestag (funding number: KK5055201MS0).

References

[1] Spies, S., Bartelt, M., Hypki, A., Kuhlenkötter, B., 2019. Robot Automation in Control Cabinet Assembly, in: COMA'19 Proceedings, pp. 272–277.

[2] Busi, M., Cirillo, A., Gregorio, D. de, Indovini, M., Maria, G. de, Melchiorri, C., Natale, C., Palli, G., Pirozzi, S., 2017. The WIRES Experiment: Tools and Strategies for Robotized Switchgear Cabling. Procedia Manufacturing 11, 355–363.
[3] Edelmann, M., 2021. How to make wiring more efficient from start to finish: Push-in technology Whitepaper.
[4] Khan, A.A., Abonyi, J., 2022. Information sharing in supply chains – Interoperability in an era of circular economy. Cleaner Logistics and Supply Chain 5, 100074.
[5] Winkler, D., Biffl, S., Bergsmann, J., 2019. Software Quality: The Complexity and Challenges of Software Engineering and Software Quality in the Cloud. Springer International Publishing, Cham.
[6] Lüder, A., Schmidt, N., 2020. AutomationML in a Nutshell, in: Hompel, M. ten, Vogel-Heuser, B., Bauernhansl, T. (Eds.), Handbuch Industrie 4.0. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 1–48.
[7] AutomationML consortium, 2020. Whitepaper AutomationML: Part 6: AutomationML Component.
[8] ETIM International, 2019. Guideline for suppliers to provide product data according to BMEcat Version 2005. http://www.etim.at/fileadmin/user_upload/ETIM_BMEcat_Guideline_V4-0-2_changes.pdf. Accessed 4 March 2023.
[9] Spies, S., Bartelt, M., Kuhlenkötter, B., 2019. Wiring of Control Cabinets using a Distributed Control within a Robot-Based Production Cell, in: 2019 19th International Conference on Advanced Robotics (ICAR). 2019 19th International Conference on Advanced Robotics (ICAR), Belo Horizonte, Brazil. 02.12.2019 - 06.12.2019. IEEE, pp. 332–337.
[10] Campbell, S., O'Mahony, N., Carvalho, A., Krpalkova, L., Riordan, D., Walsh, J., 2020. Path Planning Techniques for Mobile Robots A Review, in: 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE). 2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE), Barcelona, Spain. 12.02.2020 - 15.02.2020. IEEE, pp. 12–16.
[11] Injarapu, A.S.H.H.V., Gawre, S.K., 2017. A survey of autonomous mobile robot path planning approaches, in: 2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE). 2017 International Conference on Recent Innovations in Signal processing and Embedded Systems (RISE), Bhopal. 27.10.2017 - 29.10.2017. IEEE, pp. 624–628.
[12] Karur, K., Sharma, N., Dharmatti, C., Siegel, J.E., 2021. A Survey of Path Planning Algorithms for Mobile Robots. Vehicles 3 (3), 448–468.
[13] Schotte, W., Ovtcharova, J., 2006. Improved impulse-based simulation of cables for virtual assembly environments, in: , International PACE Forum Collaborative Visualization.
[14] Gautam, B., 2021. Energy Minimization, in: Trindade Maia, R., Maciel de Moraes Filho, R., Campos, M. (Eds.), Homology Molecular Modeling - Perspectives and Applications. IntechOpen.
[15] Yu, X., Gen, M., 2010. Introduction to evolutionary algorithms. Springer, London, 418 pp.
[16] Bepu Entertainment LLC. BEPUphysics. https://www.bepuentertainment.com/. Accessed 2 August 2023.

Biography

Stefanie Bartelt (*1991) is a researcher at the Chair of Production Systems at Ruhr University Bochum. She conducts research in the field of industrial robotics and is currently working on a project for the automated handling and quality control of bendable components for the wiring of control cabinets.

Matthias Bartelt (*1978) is head of research and development at SCHUBS GmbH. He is working in the field of control cabinet manufacturing. Thereby, he develops solutions to optimize assembly processes and to achieve consistent use of data from the planning phase up to the delivery of the cabinet.

Bernd Kuhlenkötter (*1971) is Professor and head of the Chair of Production Systems (LPS) at Ruhr University Bochum. His research focuses on the planning, simulation and implementation of production systems and smart product-service systems. Prof. Kuhlenkötter is, among others, a member of the Scientific Society for Production Engineering (WGP) and the Scientific Society for Assembly, Handling and Industrial Robotics (MHI) as well as scientific director of the Research Center for the Engineering of Smart Product-Service Systems (ZESS).