Installation Order in Automatic Fabrication of Reinforcement Rebar Cages

Johan Relefors[1], Mahdi Momeni[2], Lars Pettersson[3], Alessandro V. Papadopoulos[2], and Thomas Nolte[2]
[1]Solving Robotics Sweden AB, Västerås, Sweden
[2]Mälardalen University, Västerås, Sweden
[3]Skanska Sweden AB, Stockholm, Sweden

***Abstract*—Despite the significant development of automation in the manufacturing industry, the construction industry has not yet comparably gained much from automated processes. Fabrication of reinforcement rebar cages is one good example where automation has a limited application. Several challenges have to be tackled to introduce and take advantage of the automatic fabrication of reinforcement rebar cages. One important challenge is how and in what order the rebars should be installed one after another so that the fabrication of a reinforcement rebar cage is feasible. In this paper, we present our ongoing work towards proposing a method that gives a solution to finding the installation order of rebars.**

I. Introduction

The manufacturing industry has gained a lot of benefits through employment and the application of automation and robots in their production lines. The majority of the robots are used in the automotive, electronics, and metal machinery industries[1]. While the interest in employment and application of robots in the manufacturing industry is increasing on a daily basis, the construction industry has not yet witnessed major applications of robots to automate construction. Even though there are initiatives towards automation in construction domain both in research [1]–[5], and in industry, e.g., Built robotics[2], MX3D[3], IronBot or TyBot[4], the overall trend has not yet been properly reflected in the construction industry. One reason for the lack of reflection could be the difference(s) between the construction industry and the manufacturing industry which makes the employment of robots in the construction industry, which are perhaps expected to be on-site, a challenging task. One key difference between the construction industry and manufacturing industry is that construction projects are known to be one of a kind. As a matter of fact, automated solutions require special and careful considerations. A particular example, which is also of interest in this research, would be the fabrication of reinforcement rebar cages which is done manually even to this date. In addition to the required time, such manual work is often strenuous which can also happen under hazardous and harsh conditions, which in return is also reflected in a high risk for work accidents. The above-mentioned, as well as other issues, create a huge potential need for utilizing automation techniques in the construction industry, with a promise for increased safety and efficiency, as well as reduced costs.

In this work, we address the automated fabrication of reinforcement rebar cages. In particular, we consider proposing a solution for the problem of finding an installation order by which the rebars can be placed one after another so that the rebar cage is eventually fabricated.

This work was supported by the Knowledge Foundation with the Automation Region Research Academy (ARRAY) and SBUF.

[1]https://ifr.org/
[2]https://www.builtrobotics.com/
[3]https://mx3d.com/
[4]https://www.constructionrobots.com/

II. Problem Statement and Mission

The overall idea of the robotic fabrication of reinforcement rebar cages, including simulation, a proof-of-concept installation, and a description of some of the challenges ahead, was introduced in [1]. The idea is to automate the pre-fabrication of rebar cages. Upon completion, each rebar cage is transported to its final position before pouring concrete. Importantly in the context of this work, rebar cages are often one of a kind

Part of automating the generation of programs for the fabrication of rebar cages is to automate the process of *finding* an installation order for the rebars in a given rebar cage. There is also another important related part which is to answer the question of whether there *exists* a valid installation order for a given rebar cage. A fast method for either finding or showing the existence of an installation order would be most helpful when designing rebar cages. In this paper, we focus on our ongoing work on the question of *finding* an installation order. Or stated differently, the problem we wish to solve is:

Problem. *Given a 3D model of a reinforcement rebar cage, find an order in which the rebars should be installed in order to produce the rebar cage using the setup described in [1].*

When examining this problem we are working under the following assumptions:

Assumption 1. *The rebars are rigid enough, i.e., they do not deflect or twist because of their weight.*

Assumption 2. *The rebars are not deformed during the process before/after the cut and bend machine, i.e., their theoretical geometry is the same as their actual geometry.*

Assumption 3. *While lifting rebars out of the cage, two movements are allowed for the rebar: a straight line on*

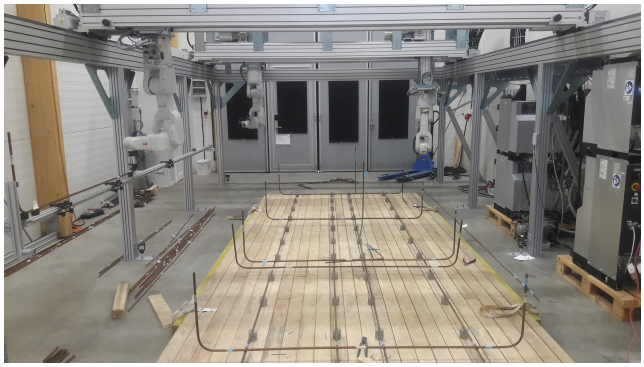

Fig. 1. Gantry robot systems

a horizontal plane or along the vertical axis. Horizontal movements could be parallel to either the x- or y-axis of the reference frame while the vertical movement is parallel to the z-axis of the reference frame.

The aim of this paper is then to outline our ongoing efforts and intention to tackle the problem of how to automatically generate an ordering of rebars which enables the robots to fabricate a cage by mounting and tying the rebars of different types, such as "A", "B", and "C"-bars, one after another. The installation order must allow for collision-free paths and the cage needs to be stable throughout the entire installation process, as well as when moving the rebar cage to its final location.

The problem of finding a valid installation order is a task planning problem. The input to the problem is then an unordered set of rebars $R_{in} = \{A, B, C, \ldots\}$ where each rebar has geometric information of shape, position, and orientation associated with it. The desired output is an ordered set R_{out} giving the installation order of all the rebars in R_{in} together with additional generated information on where to grip the rebars and where to tie the rebars into the already installed rebars. Note that we are not currently looking for an ordering which is optimal under some criteria, but rather an ordering which is feasible.

Another way of looking at the problem is to assign each rebar to a node in a graph. And letting the directed edges in the graph correspond to an ordering relation. Any edge in the graph might then be allowed or not allowed. This view highlights an interesting feature of the problem which is that the same edge can be allowed or not allowed depending on which of the nodes have already been visited. As an example, a rebar A might be blocking rebar B but rebar C might not block either of them. In this case, the ordering CAB is not allowed while the ordering BCA is allowed. Note that both orderings contain the sub-ordering CA which would correspond to an edge in the graph. This edge is allowed if B has already been placed but otherwise it is not allowed. This means that it is not possible to simply map the edges of the graph and then find a way through the graph.

From the discussion in the previous paragraph, we conclude that in order to solve the ordering problem we need the ability to determine whether a given state of the cage allows for a particular rebar to be installed. Given that rebars $\{A, B, C, \ldots\}$ have been placed, rebars $\{a, b, c, \ldots\}$ have yet to be placed, and we want to determine whether rebar a can be placed we need to determine the function

$$f(A, B, C, \ldots, a, b, c, \ldots, a) = \begin{cases} true \text{ if } a \text{ can be placed,} \\ false \text{ otherwise.} \end{cases} \tag{1}$$

One method for determining the output of f for installing a rebar in a given situation is to try to compute a path for the rebar, and then check that the rebar is not blocking the installation of any other rebars which are yet to be installed. However, producing a path is time-consuming and, in a case where there are many rebars to install, the combinatorics of the problem will lead to many evaluations of f.

The naive method described above will most likely not be fast enough for use in the process of designing rebar cages. Reducing the time that it takes to compute f, or reducing the number of times that f needs to be computed, are two ways of speeding up the computation of a valid installation order. In this paper, we propose two methods that we are working on evaluating.

III. Methodology

In the ideal case, f only needs to be evaluated as many times as there are rebars in the cage, completely removing the combinatorial aspect of the problem. To find such an algorithm, we would need to formulate precedence constraints on which rebars could potentially be installed in a partially assembled cage. Note that for a given partially assembled cage, there could be many rebars that are possible to install, introducing some arbitrariness in the installation order.

If an algorithm that never needs to evaluate f to $false$ is not found, the number of evaluations of f should be minimized. In this section, we briefly investigate how to compute an installation order by starting with no rebars installed, and moving forward until completion, as well as starting from a fully built cage, and disassembling it until no rebar is present, and using the reversed disassembly order. Furthermore, since it is most likely that f will sometimes evaluate to $false$, we should aim for computing f in a way that terminates early in the process when a rebar cannot be installed. Therefore, before turning to the ideas for rebar installation order, we first briefly look at computing f and how these ideas can help in reducing the combinatorial nature of the problem.

A. Computing f

At least for now, the only way that we can guarantee that a rebar can be installed is to compute paths for placing and tying the rebar in the cage. However, many ways of determining when a rebar cannot be installed exist.

When computing the installation path we start with the rebar in its position in the cage and try to remove it with the constraint that the rebar should be possible to remove using a

simple set of movements, see Assumption 3. This means that one way of figuring out whether a rebar can be installed or not is to check whether the simple movements are enough for removing the rebar from the cage. Below we list a few ways that installation of a rebar can fail:

- The rebar cannot be removed using simple movements of Assumption 3.
- The rebar is blocking the path of the remaining yet-to-be-installed rebars in an unavoidable obstacle.
- There are not enough valid grip locations on the rebar (considering only the tools and not the gantry robot systems).
- There are too few available tie points for the cage to remain stable after placing and tying the rebar (again considering only tools and not the gantry robot systems).
- The grip positions cannot be reached by the gantry robot systems.
- The tie positions cannot be reached by the gantry robot systems.

These ways of failing are computationally less expensive to check than the computation of a path (they are actually necessary when computing a path). This means that negative evaluations of f can terminate faster than positive ones.

Another way to speed up the computation of f is to save and keep track of the evaluations of f as well as the reasons for failures as a reference in future evaluations. For example, if a certain rebar cannot be installed in the cage because it needs additional support, we note which rebars could provide support for the rebar and don't consider the rebar for installation again until enough support is added.

B. Pre-computing parts of f

Valid installation orders can be found by removing the invalid ones. The preceding discussion on computing f hints at ways of removing some invalid orders by creating precedence constraints from the different ways f evaluates to $false$.

Precedence constraints can for example be based on which rebars are tied together. Starting from the rebars in the bottom of the cage, only rebars that tie into these can be considered possible to install. Then, one can consider the cage after those rebars have been added, and see which rebars can be installed then. Another way of incorporating constraints from tying is to loop through all rebars in the cage and establish for each rebar, which other rebars are necessary for stability. Similar things can be done by checking which other rebars are in the way of a specific rebar when trying to remove that rebar.

Precedence constraints created in this way amount to a partial pre-computation of f. The pre-computation is used for reducing the combinatorial complexity of the problem. This can be useful as long as the pre-computation itself does not give a large combinatorial factor and the evaluation of the part f that is considered is fast enough.

C. Proposed Solutions

Ignoring precedence constraints for the moment and looking at ways to create an installation order, we find two ways of thinking. One is starting with no cage and adding rebars, one by one. The other is starting from the final state of the cage and removing rebars one by one and finally reversing the uninstallation order to get an installation order. We call these methods the forward and backward methods, respectively.

1) Forward: In the forward method the general idea is to look at the cage and identify which rebars tie into the cage in a stable way given the cage's current state. Among the rebars we then start by trying to install the rebar with the tie points with the lowest "Z-"coordinate. In cases where this does not fully determine the order, we start with the rebar which is closest to the origin which we take to be in one of the corners of the cage[5]. This has to be done in a way that makes sure that the rebar does not block the installation of any other yet-to-be-installed rebars.

2) Backward: In the backward method, we start from the final state of the cage and identify rebars that can be removed to find an uninstallation order. Choosing which rebar to try to uninstall is not as clear as choosing which rebar to try to install in the forward method. It has to be a rebar that can be removed without compromising the stability of the rest of the cage. Possibly choosing one with the fewest tie points into the cage or some similar metric. While the uninstallation of a rebar is done, it should not affect the uninstallation of subsequent rebars. However, uninstalling a rebar might remove support for other rebars which could render the cage in a state where no rebar can be removed without affecting the stability of the cage.

3) Precedence constraints: Turning to the precedence constraints, the idea is to try to capture the positive sides of both types of algorithms. The natural order of installation and certainty of support imposed by the forward method can be captured as precedence constraints. These can hopefully be merged with the certainty that installing a rebar does not block the installation of another rebar. Looking at evaluations of f, the forward and backward method capture different aspects of the evaluation.

IV. Discussion

To generate the installation order, two methods have been proposed in this paper. In the $Backward$ method, the installation order is found by reversing the deconstruction process while in the $Forward$ method, the rebar cage is fabricated straight away. The $Forward$ method is inspired by how skilled workers fabricate a cage.

In the $Forward$ method, the general idea is to start with no rebars installed and finding an installation order by placing and tying rebars in a stable way. In the $Backward$ method, the idea is to start with the fully built cage and remove rebars one by one from the cage without causing any instability, and then reversing the order to get an installation order.

In the $Forward$ method, the only rebars which can be installed are the ones that tie into the already installed rebars.

[5]Moving the origin would produce different installation orders, which is something we intend to try.

This limits the search space at any given state of the cage. On the other hand, in the $Backward$ method, we have not found a simple way of limiting which rebars that are possible to remove. This means that the $Forward$ method has fewer branching possibilities, thereby reducing the combinatorial difficulty of the problem.

In the $Forward$ method, installing a rebar could give a situation where one of the yet-to-be-installed rebars cannot be installed. To avoid blocking yet-to-be-installed rebars, checks involving both the state of the cage and the yet-to-be-installed rebars need to be performed. In the $Backward$ method, removing a rebar is most likely not going to hinder the removal of other rebars in the cage. That means that the state of the cage can be treated separately from the uninstalled rebars.

V. Summary and Work-in-progress

In this paper, we presented our ongoing work towards automatic generation of the installation order for robotic fabrication of rebar cages given a 3D CAD model of the rebar cage. We defined a function, f which evaluates to $true$ or $false$ depending on whether a given rebar can be installed at a given position in a yet unfinished rebar cage. We then defined and analyzed our problem using properties of f.

Using a few different ways in which f evaluates to $false$ we can derive precedence constraints among the rebars of a given cage. The justification for focusing on the $false$ evaluations is that the needed computations are often much faster since $true$ evaluations require an actual path to be generated. This partial evaluation of f reduces the combinatorial difficulty of the problem.

We also briefly examined a forward and backward method for computing an installation order and subsequently analyzed the properties of the methods in terms of precedence constraints. In our ongoing work, we are working on developing an algorithm that captures the positive sides of both the forward and backward methods, by removing invalid orderings. Part of our future work will include showing that our proposed method(s) will find a solution for the installation order given that such solution exists. We are also aiming to find an optimal solution for the automatic generation of the installation order, to minimize the required time as well as gantry-robot motion to fabricate a cage.

Acknowledgements

The authors wish to acknowledge the team at Robotdalen, both current and previous members, especially Ingemar, Anders, and Erik.

References

[1] J. Relefors, M. Momeni, L. Petterson, E. Hellström, A. Thunell, A. V. Papadopoulos, and T. Nolte, "Towards automated installation of reinforcement using industrial robots," in *24th IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 1595–1598.

[2] B. Dolinšek and J. Duhovnik, "Robotic assembly of rebar cages for beams and columns," *Automation in Construction*, vol. 8, no. 2, pp. 195–207, 1998.

[3] A. Mirjan, F. Augugliaro, R. D'Andrea, F. Gramazio, and M. Kohler, *Building a Bridge with Flying Robots*. Springer, 2016.

[4] H. Ardiny, S. J. Witwicki, and F. Mondada, "Are autonomous mobile robots able to take over construction? A Review," *Int. J. Robotics*, vol. 4, no. 3, pp. 10–21, 2015.

[5] R. Bogue, "What are the prospects for robots in the construction industry?" *Industrial Robot*, vol. 45, no. 1, pp. 1–6, 2018.