The International Journal of Advanced Manufacturing Technology
https://doi.org/10.1007/s00170-023-10996-z

ORIGINAL ARTICLE

Generation of rule-adhering robot programs for aluminium welding automatically from CAD

Tuan Anh Tran[1] · Eirik Bjørndal Njåstad[2] · Ole Terje Midling[3] · Morten Bjelland[4] · Andrei Lobov[1]

Received: 20 September 2022 / Accepted: 24 January 2023

Abstract
This paper presents a method to automatically generate robot welding programs from CAD to address the ever-constant demand for product customisation. Furthermore, to ensure that proper welding operations and structural integrity are met, the generated programs also consider the welding conditions and requirements. These welding conditions and requirements are defined by the weld direction and face relative to gravity and surrounding geometry, which has not been observed in the present research sphere. To achieve this, the approach leverages information that can be extracted from a topological analysis of tessellated geometry local to the weld joint in conjunction with available CAD API functions. Finally, an implementation of the method using Siemens NX and the Robotics Toolbox for Python is presented and tested on three geometrically different node configurations and a stiffener piece provided by industrial collaborators. In all, the proposed system was able to correctly generate programs adhering to allowed welding operations as long as a solution existed. For the more complex node configurations (which require reorientation when welded by humans), 32 weld path programs out of 42 were generated based on the given criteria. For the least complex node, a total of 20 out of 24 were generated with the same criteria. All 14 weld programs were generated for the stiffener representation.

Keywords Computer-aided design · Robotic welding · Automatic offline programming

1 Introduction

To achieve a competitive edge, companies are constantly on the lookout for ways to offer a higher level of customisation of their products at increasingly higher quantities and with a shorter lead time [1–3]. This, in turn, calls for more flexible and intelligent development processes. Robotic automation has the potential to improve quality, flexibility, and save costs for manufacturers with shorter production runs and facilitate higher production uptime [4]. However, traditional robot programming methods and transitional phases between Computer-aided Design (CAD) and Computer-aided Manufacturing (CAM) still suffer from inefficient and time-consuming processes [5, 6]. Additionally, these inefficiencies become an even more significant hurdle when applied to the high demand for product customisation. Finally, when coupled with the initial expenses and required expertise linked with the implementation of robotic manufacturing, many Small- and Medium-sized Enterprises (SMEs) struggle to justify the adaptation of such systems [7]. Automatic generation of robot welding programs can be challenging and complex, depending on the geometry, especially when the geometry is parametric and constantly adapted to user demands.

In addition to higher flexibility and robotic implementation, utilising aluminium alloys in marine structures is also becoming increasingly desirable [8]. Aluminium alloys, with their high strength-to-weight ratio, allow for more lightweight designs, which can be useful for marine applications. This, unfortunately, comes with a price of several challenges in its manufacturing to reach adequate levels of structural integrity [9]. With regard to welding, these challenges are reflected as requiring higher levels of consideration of welding conditions and techniques to ensure proper fusion. In addition, the reflective nature of aluminium surfaces makes it more difficult for visual sensing systems, and the alloys are also more susceptible to thermal distortion.

This paper presents a framework and approach for generating robot welding programs starting from CAD.

✉ Tuan Anh Tran
tuan.a.tran@ntnu.no

Extended author information available on the last page of the article.

Published online: 08 March 2023

The particular distinction between the presented approach and earlier work is the focus towards high-mix-low-volume products, which can vary highly in geometry. Thus, the generation algorithm should be generalised with respect to geometry. Additionally, the program generation also considers requirements for particular welding operations in order to ensure sufficient structural integrity for the resulting welds. Such considerations become even more critical when manufacturing heat-sensitive materials such as aluminium.

The method utilises a combination of approaches for Automatic Offline Programming (AOLP), Automatic Feature Recognition (AFR), and a Knowledge Base (KB) to (1) automatically extract potential welds in addition to other relevant information from the CAD environment, (2) classify the different welding operations, and finally (3) generate rule-adhering robot welding programs. The developed system is incorporated into Siemens NX, using the NXOpen Python API to allow CAD functions to be accessed programmatically. It should be noted that while Siemens NX is used as a CAD-starting point in this paper, other CAD tools, such as Open Cascade and SolidWorks, can be used similarly [10, 11]. While the framework is developed with Python robotics toolbox as the underlying collision and kinematic functionality, the methods and algorithms described are not dependent on any particular tool, given that the same functionality is available.

The paper is structured in the following way. In Section 2, a literature review is conducted, describing current methods and presenting the identified gaps that are approached in this paper. In Section 3, the approach and system architecture is described in addition to the underlying methods for local wall extraction. Following, in Section 4, is a case study using several aluminium helideck nodes and a stiffener presentation as input for the automatic weld program generation. The parts and models used were provided by industrial project partners Marine Aluminium AS and Leirvik AS and are based on actual use cases. Section 5 then discusses the method's viability and potential further work, going through the implementation. Finally, the conclusions are presented in Section 6.

2 Literature review

Present-day robotic programming methods are mainly comprised of two different programming modes; online and offline [12]. Online programming is done using teach pendants, relying on the expertise and skill of the robot operator to determine the robot's behaviour [13]. While this allows for a more "hands-on" and intuitive problem-solving environment, the potential for automation is also more limited. Additionally, modifying and changing the product designs often leads to a manual reprogramming of parts or the entirety of the program while also requiring significant downtime of the cell during programming [14]. Offline Programming (OLP) relies on a digital environment to effectively simulate the robot programming process with spatial and kinematic models of the manufacturing environment, decreasing the required downtime [15]. This digital representation, in turn, enables robot programming and decision-making to be approached programmatically and with more intelligent algorithms, leading to so-called automated offline programming (AOLP).

AOLP is when algorithms, functions, and intelligent systems are layered on top of the path-planning capabilities of offline programming in a simulated manufacturing system, resulting in an automated process. Such automated functionality is appealing to industries that have a high demand for customisation and variation, which has resulted in an increase in research interest over time [13].

Many approaches have been presented for generating robot programs, with varying levels of autonomy, flexibility, and stability. For simplified setups, basic AOLP systems extending from CAD can be derived through the utilisation of in-built APIs for weld path extraction in combination with transformation calculations to invoke basic robot operations from the controller and assumptions about the setup [16–18]. However, manual steps are still required to provide viable tool orientations. In the case of [17], the setup enabled orientations to calculated with less attention to collision and singularities.

Other approaches utilise knowledge-based engineering principles and semantic knowledge models to automate decision-making for the generation of robot programs [19, 20]. While this enables the automatic generation of viable welding programs, current efforts utilising knowledge-driven systems have shown limited display regarding welding path complexity and flexibility in product geometries [21].

While these approaches are capable of solving the tasks they were intended for, further improvements are necessary for adaptation into industrial settings. For one, the program generation should be flexible and be able to solve based on product topology. Secondly, the output of the programs should contain solutions that are viable in ensuring structural integrity, as shown in the knowledge-driven methods. Larkin et al. demonstrated that using greedy search algorithms to find solutions from decomposed tasks- and configuration spaces is a viable topology-driven approach for welding operations [7]. With welding tool operations for these spaces established, path planning algorithms such as probabilistic roadmap (PRM)

and rapidly exploring random trees (RRT) were further shown to be capable of solving the remaining via-point motions. The benefit of this approach is the flexibility and independence from deriving rule sets that only work on particular product geometries. Instead, only the collision topology and simulated manufacturing cell are considered baseline elements.

Recent advances in AOLP within existing welding research still need more diversity in targeted product geometry and managing the output programs to ensure viable welding. For welding operations, this includes consideration of gravity and welding requirements. While the aforementioned knowledge-driven approaches target this domain, their flexibility towards product geometry is limited. Therefore, the approach presented in this paper aims at providing a welding program generation method capable of generating solutions for diverse geometries, given that solutions obeying the given criteria exist. Furthermore, while extending from the CAD environment and extracting useful welding information, the presented method incorporates feature recognition to automatically extract potential welds and useful weld-related information in the CAD environment to classify and generate robot welding programs [22].

3 Approach

This section describes the underlying approach to drive the automatic generation of rule-adhering robot welding programs from CAD. The general idea is that an autonomous CAD-to-robot program generation can be enabled in the CAD environment by utilising available APIs, similar to [16–18]. However, in addition to extracting and transferring weld information throughout the program generation system, the proposed approach also layers topology analysis and feature recognition to this stage to automatically extract potential welds and useful information for both program generation and classification down the line [22]. These welds are then transferred to a manufacturing simulation environment, where several proposed algorithms are used to generate collision-free, singularity-free, and rule-adhering robot welding programs. Figure 1 depicts the proposed approach's overall steps and information flow, starting from a CAD model and resulting in so-called generalised program tasks. These generalised program tasks contain information about collision-free and allowed welding operations and the corresponding classifications, which can be used to determine appropriate welding parameters or procedures. The idea is that a local robot controller can

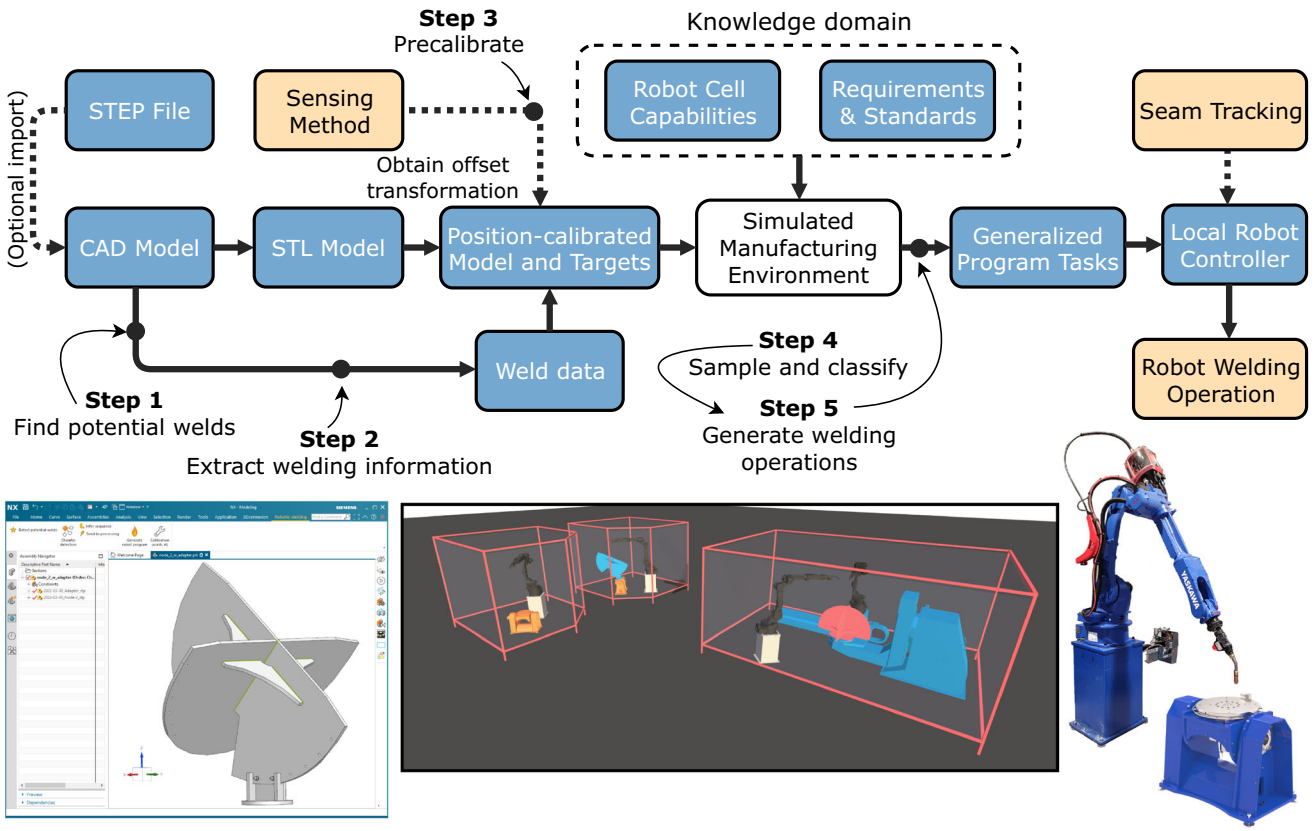


Fig. 1 Approach overview, where the blue boxes represent digital information flow as the CAD data moves to generalised program tasks, and the orange boxes represent physical systems

be used as a final parser to generate updated trajectories based on these generalised tasks in combination with other specialised scanning-driven welding policies. The depicted system consists mainly of five essential steps for the program generation:

1. CAD-integrated feature recognition to detect all the potential weld paths in the CAD environment. Tessellated geometry is generated as part of the process for analysis and use in collision models in the CAM environment.
2. Topological analysis of scanning of potential welds to extract local wall vectors and calculation of welding faces and initial approach vectors for the welding gun.
3. Transfer weld package and collision models to the CAM environment and determine the transformation offset of the workpiece relative to the robot manufacturing setup.
4. Classify welds for the potential welding operations depending on workpiece orientation and welding directions.
5. Generate generalised welding tasks with classified requirement-adhering welds.

The program generation approach proposed in this paper was developed using the CAD software Siemens NX along with the NXOpen Python API, which allows CAD functions to be accessed programmatically. Using the NXOpen API also allowed for the seamless transfer of information between the different systems, as client-server relationships could be deployed. Peter Corke's Robotics Toolbox Python was used to simulate the manufacturing environment and generate robot trajectories [23]. Currently, the robotics toolbox implements basic collision handling from PyBullet [24]. However, meshes are only represented by their convex hull. As such, all models, or assemblies,

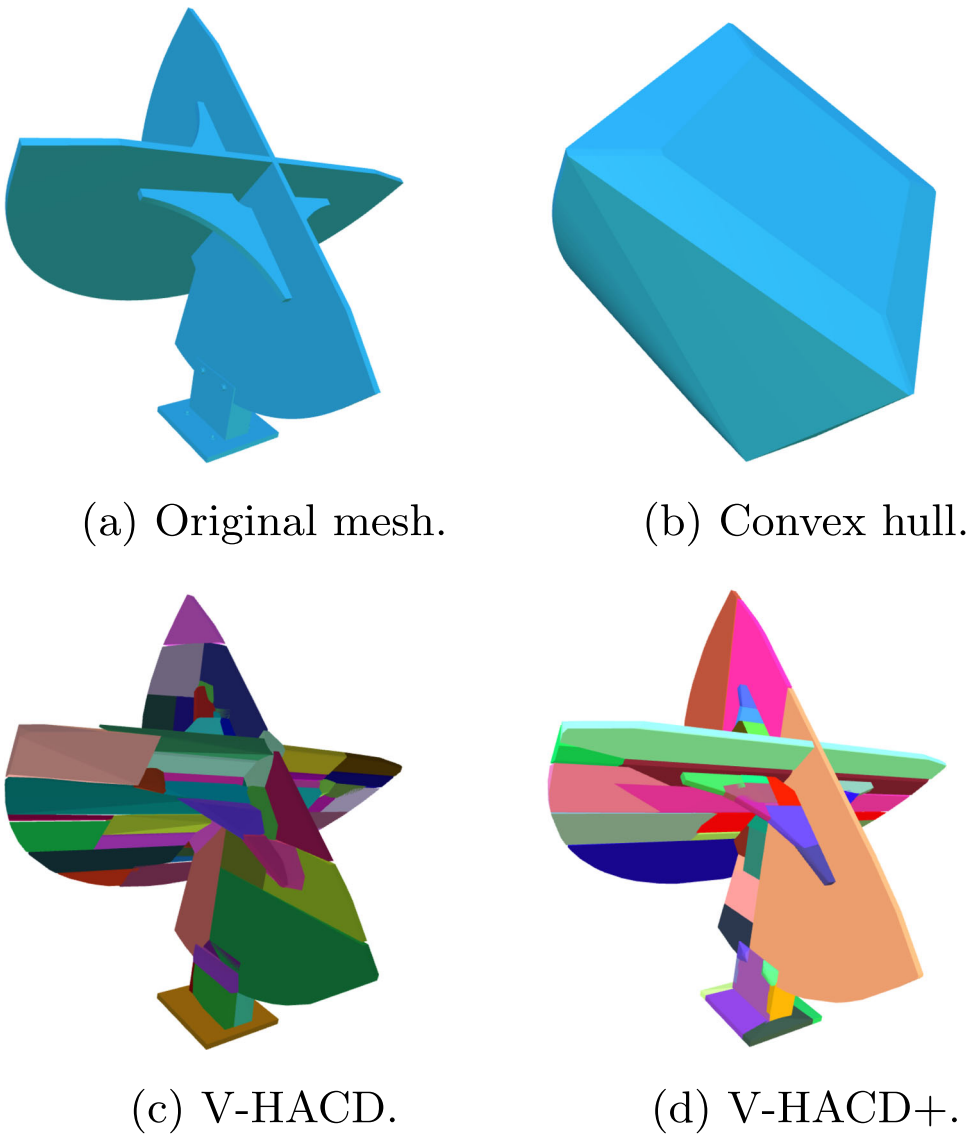


Fig. 2 Original mesh and collision models. Each colour represents a separate convex shape

were decomposed into parts and further decomposed into convex hull approximations with the open-source library Volumetric-Hierarchical Approximate Convex Decomposition (V-HACD) [25]. By using this approach, the final collision boundaries can be seen in Fig. 2, along with the unprocessed convex hull and original mesh. Each convex hull is represented as a different colour. The rationale for the initial part decomposition is due to the potential artefacts that V-HACD, in some cases, can introduce to the collision model [26], as can be seen in Fig. 2c. In the case of plate geometry, the decomposition into separate parts prior to V-HACD seemed more stable and avoided creating these artefacts. The browser-based visualisation module Swift offered by the robotics toolbox, was used to visualise the robot environment. By modifying the module to allow sockets over the web and integrating with the Python Flask module [27], a web-based simulation service for the manufacturing environment was deployed.

3.1 Topological extraction of relevant welding information

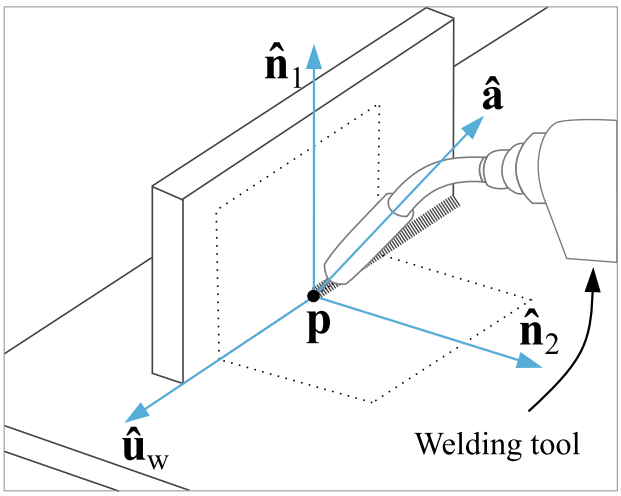
The robot program generation method used in this paper requires base information about the welds and local path topology to function in addition to the robot and manufacturing specifications. This local path topology comprises the welding points that constitute the lines or discretised to curve shapes, making up the welding operation. In addition, each point contains information about the directions of the walls closest to the weld (local walls). The local wall vectors are essentially the normal vectors of the faces intersecting with the weld but are easier to discretise for more complex shapes such as rounded surfaces. By knowing these, an initial approach vector can be calculated for the welding tool, allowing the sampling method to determine feasible robot welding operations. The extraction of local wall vectors is necessary in order to determine the face of the weld in cases where this is not defined manually in CAD. The main steps of the topological extraction are summarised as follows:

1. Collect intersection lines occurring between solid body elements in the CAD environment.
2. Segment solid body intersection lines that intersect with each other into separate lines.
3. Perform multi-directional slicing to obtain cross-sections for all potential welds.
4. Filter out welds that are assumed to be not accessible based on the angle of visibility from the weld face in the cross-section.
5. Calculate the local wall vectors and accurate groove angles.

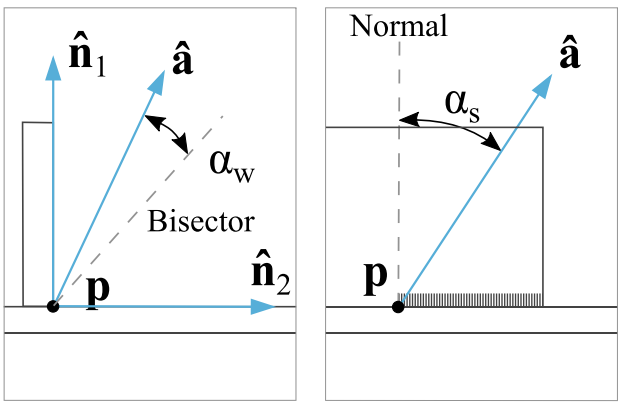
Figure 3 depicts the underlying vector and angle definitions used to generate the welding procedures. The wall vectors denoted as $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$, which is used to calculate an initial approach vector $\hat{\mathbf{a}}$, are shown in Fig. 3a. In certain cases, the weld gun needs to be shifted towards either wall to accommodate collision-free trajectories or in the welding direction $\hat{\mathbf{d}}_w$ to allow proper material fusion. These angles are shown in Fig. 3b and are defined as stick angle R_s and drag angle R_d. It should be noted that the local wall vectors are sensitive to features such as chamfers and fillets, as they are derived from tessellated models.

The bisector of $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$ is calculated simply as:

$$\hat{\mathbf{a}}_{\text{init}} = \hat{\mathbf{n}}_1 + \hat{\mathbf{n}}_2 \tag{1}$$



(a) Approach vector $\hat{\mathbf{a}}$, defined as the welding tool direction shown relative to wall vectors $\hat{\mathbf{n}}_1$, $\hat{\mathbf{n}}_2$, and welding direction $\hat{\mathbf{d}}_w$.



(b) Work angle α_w, and stick angle α_s defined relative to normals and bisector.

Fig. 3 Definition of welding parameters and vectors. Welding axis γ_a, and welding face γ_f

Shifting of the approach angle by the work- and stick-angles α_s and α_w can then be done with the following equation:

$$\hat{\mathbf{a}}_d = \cos(\alpha_d)\hat{\mathbf{d}}_w + \sin(\alpha_d)\hat{\mathbf{a}}_{\text{init}} \tag{2}$$

where $\hat{\mathbf{d}}_w$ is the welding direction and $\hat{\mathbf{a}}_{\text{init}}$ is the initial approach angle based on the bisector. Finally, the final approach vector $\hat{\mathbf{a}}_{d,s}$ with added rotation around the welding direction can be calculated with Rodrigues' rotation equation as:

$$\begin{aligned}\hat{\mathbf{a}}_{d,s} &= \cos(\alpha_w)\hat{\mathbf{a}}_d + \sin(\alpha_w)(\hat{\mathbf{d}}_w \times \hat{\mathbf{a}}_d) \\ &+ (1 - \cos(\alpha_w))(\hat{\mathbf{d}}_w \hat{\mathbf{a}}_d)\hat{\mathbf{d}}_w\end{aligned} \tag{3}$$

The automatic extraction of the local wall vectors is a crucial element for the presented approach to obtain the initial approach vector and, consequently, the automatic program generation. To extract the local wall vectors, the weld cross-sections have to be generated through a multi-directional slicer [22]. In this paper, the multi-directional slicing is done using Eq. 4, which is simply a plane equation corresponding to the slicing plane of interest, where the slicing planes are defined using an arbitrary point $\mathbf{p}$ on the potential welds along with the welding direction $\vec{\mathbf{n}}$ with their subsequent coordinate point values in x, y, and z. Variable v_i corresponds to one of three vertex points making up each triangle in the model. This is calculated for each vertex point making up the 3D tessellated assembly model:

$$\begin{aligned}Q_i &= \vec{\mathbf{n}}_x(\mathbf{p}_x - v_{i,x}) + \vec{\mathbf{n}}_y(\mathbf{p}_y - v_{i,y}) \\ &+ \vec{\mathbf{n}}_z(\mathbf{p}_z - v_{i,z})\end{aligned} \tag{4}$$

Six possible states are defined to determine whether a vertex intersects with the slicing planes. Three of these states occur when two vertex points of a triangle are located on each side of the plane. This corresponds to the Q variable being positive for one and negative for the other. This is illustrated in Fig. 4, with v_0 and v_1 being on opposite sides, as well as v_2 and v_1. The remaining three cases occur when one triangle vertex point is located precisely on the plane. In these cases, Q is equal to zero.

Finally, local wall vectors in CAD can be extracted from the cross-sections with Algorithm 1. The algorithm works by assuming that there are three possible scenarios where a local wall vector is connected to the weld. The first two scenarios occur when a given line pair consisting of 3-dimensional points p_a and p_b on the cross-section plane contains a point that occurs close to the weld path. A given tolerance, γ_t, should be defined for models that contain tolerance gaps in the model. The third scenario is where the corresponding line passes through or is close enough to the welding path. As the correct directions of the wall vectors relative to the weld are still uncertain, they have to be checked in a second loop. This is done by calculating the bisector and performing a point-in-polygon check, eliminating the inward pointing vectors. The bisectors with the corresponding wall vector set that occur outside a polygon can be assumed to be the correct solution.

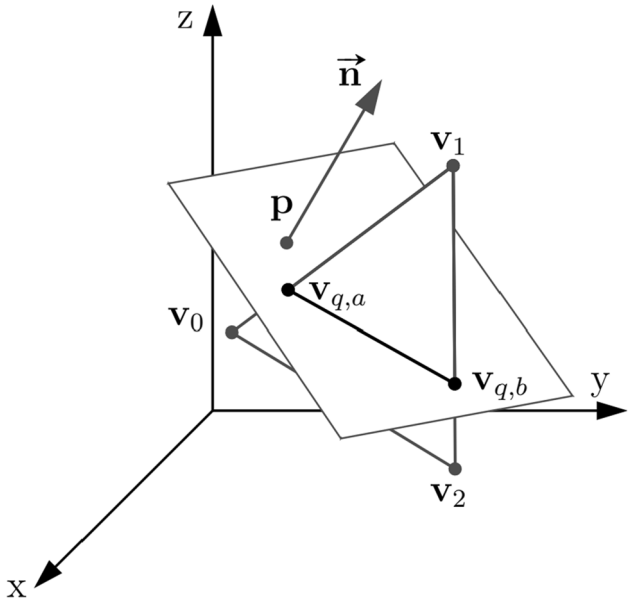


Fig. 4 Arbitrary slicing plane [22]

Input: Weld cross-section slices S_{pairs} defined as point pairs in 3D, cross-section center point P_c as point on weld of which cross-section was derived, scan tolerance γ_t
Output: List of local walls vectors L_{pairs}

for each pair$[p_a, p_b]$ in S_{pairs} **do**
  if p_a is close to p_c **then**
    $p_{\text{walls}} \leftarrow p_b$ ▷ Add potential wall vector points.
  else if p_b is close to p_c **then**
    $p_{\text{walls}} \leftarrow p_a$
  else if p_c is on line and between pair **then**
    $p_{\text{walls}} \leftarrow p_a, p_b$
  end if
end for
for each point p_{wall} in p_{walls} **do**
  $v_p \leftarrow p_{\text{wall}} - p_c$ ▷ Calculate potential wall vector.
end for
$v_{\text{wall sets}} \leftarrow \text{combinations}(v_{\text{wall}}, 2)$ ▷ Generate potential wall sets
for each $v_{\text{wall set}}$ in $v_{\text{wall sets}}$ **do**
  $a \leftarrow v_a + v_b$ ▷ Calculate bisector
  if a not inside solid **then**
    $L_{\text{pairs}} \leftarrow L[v_a, v_b]$ ▷ Add wall set to solution pool
  end if
end for

Algorithm 1 Cross-section scanning of local wall vectors $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$.

Fig. 5 Visualisation of wall vectors (blue) and initial approach vectors (yellow) automatically scanned and generated from CAD

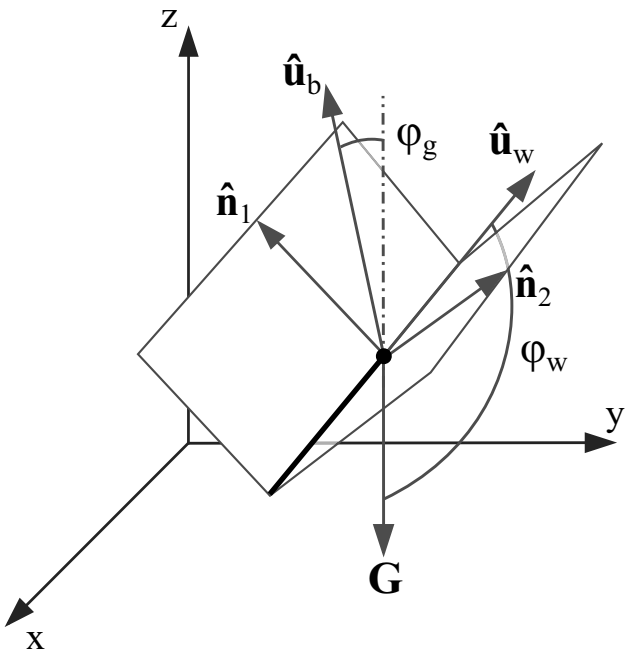


Fig. 6 Vectors for defining welding positions and operations. Welding axis $\hat{\mathbf{u}}_w$, and welding face $\hat{\mathbf{u}}_b$

Figure 5 shows the automatically extracted wall vectors $\hat{\mathbf{n}}_1$ and $\hat{\mathbf{n}}_2$ in blue and corresponding bisectors $\hat{\mathbf{a}}_i$ in yellow, scanned iteratively throughout the welds of a part assembly.

3.2 Classification and requirement adherence for welding operations

The classification of welding operations is used to determine whether the operation is suitable for the given material or if approved welding parameters exist. This is necessary to ensure that only valid operations are produced, meeting defined requirements and proper fusion. Additionally, the classification of operations also allows methods such as KBE to be implemented for rule-based reasoning.

The classification of the welds is done with respect to so-called welding positions, of which standards and procedures have already been established. These welding positions can be defined by applying minimum and maximum ranges to two parameters used to characterise welds: the angle ϕ_g, defined as the angle between the welding face $\hat{\mathbf{u}}_b$, and the gravity vector G, and the angle ϕ_w, defined as the angle between the welding direction $\hat{\mathbf{u}}_w$ and between the welding face $\hat{\mathbf{u}}_b$, and gravitational vector G, around the welding direction $\hat{\mathbf{u}}_w$. The representation of vectors and angles used to define welding positions can be seen depicted in Fig. 6.

The weld direction to gravity angle ϕ_g can be calculated with the following equation:

$$\phi_g = \cos^{-1}\left(\frac{v_w \cdot G}{\mid v_w \mid\mid G \mid}\right) \tag{5}$$

where v_w is the welding direction vector, and G is defined as gravity vector of the given manufacturing environment. The relative weld face vector $\hat{\mathbf{u}}_b$ to gravity G around the welding direction $\hat{\mathbf{u}}_w$, ϕ_w can then be calculated as:

$$\phi_w = \text{atan2}(u_b \cdot G, v_w \cdot (u_b \times G)) \tag{6}$$

Using these angles, the welding operations in the manufacturing environment can then be automatically classified with Algorithm 2.

Welding positions are a set of predefined welding configurations based on relative weld slope angles as well as the orientation of the weld face relative to a horizontal plane, of which standardised procedures can be developed. The classification is calculated based on the assumption that parameters R_w and R_g are sufficient for defining all types of welding operations in a relevant manner. R_g is the welding direction for the weld in relation to gravity $g = [0, 0, 1]$.

Each l_{req} in L_{reqs} consists of limit values $\phi_{\min}$ and $\phi_{\max}$ for both ϕ_w and ϕ_g. If the corresponding values of ϕ_w and ϕ_g are found to be within the definition of a classification, it is subsequently added to the weld operation object. It is important to note that it is possible to obtain several classifications for a single operation depending on the chosen values.

3.3 Generation of robot welding programs

The generation of the robot welding programs is based on selecting potential poses, trajectories, and operations that

Input: Sequenced weld transformation targets t_s, corresponding bisector vectors a, Classification limit definitions $L_{req}(R_w, R_g)$, global gravity vector $G \leftarrow [0, 0, 1]$
Output: Classifications types $l_c \in L_c$

$v_w \leftarrow t_s[1] - t_s[0]$ ▷ calculate welding direction
$\phi_g \leftarrow \cos^{-1}\left(\frac{v_w \cdot G}{|v_w||G|}\right)$ ▷ calculate welding angle relative to gravity
$\phi_w \leftarrow \text{atan2}(a \cdot G, v_w \cdot (a \times G))$ ▷ calculate angle between bisector and G around v_w
for each l_{req} in L_{reqs} **do**
  $g_{\text{limits}} \leftarrow l_{\text{req}}[\phi_g]$ ▷ get ϕ_g definition limits for each classification
  $w_{\text{limits}} \leftarrow l_{\text{req}}[\phi_w]$ ▷ get ϕ_w definition limits for each classification
  if $g_{\min} < \phi_g < g_{\max}$ **then**
    if $w_{\min} < \phi_w < w_{\max}$ **then**
      $l_c \leftarrow l_{\text{req}}$ ▷ add classification
    end if
  end if
end for

Algorithm 2 Classification of welding operations.

comply with requirements. As such, the program generation has to make sure that the selected solutions are possible with respect to the manufacturing cell configuration, i.e., joint limits, singularities, and collisions. Additionally, the generated welding operations have to comply with the restrictions defined by the expert system, meaning that only classifications approved for the given context are allowed, as elaborated in Section 3.2.

Applying these restrictions, a greedy search algorithm is used to sample through a discretised weld task space, using the initial approach vector $\hat{\mathbf{a}}_{\text{init}}$. The calculation of potential weld tool positions and orientations used for the discretised welding tool space sampling is done with Algorithm 3 using the equations described in Section 3.

Input: Initial approach $\hat{\mathbf{a}}_i$, welding direction $\hat{\mathbf{d}}_w$, work angle α_w, stick angle α_s
Output: Shifted T_{shift} (SE3) according to provided discretisation parameters

procedure SHIFTAPPROACH($\hat{\mathbf{a}}_i, \hat{\mathbf{d}}_w, \alpha_w, \alpha_s$)
  $\hat{\mathbf{a}} \leftarrow \text{Shift}(\hat{\mathbf{a}}_i, \hat{\mathbf{d}}_w, \alpha_w)$ ▷ Calculate shift (Eq. 2)
  $\hat{\mathbf{a}} \leftarrow \text{Rotate}(\hat{\mathbf{a}}, \hat{\mathbf{d}}_w, \alpha_s)$ ▷ Calculate combined rotation shift (Eq. 3)
  $T_{\text{shift}} \leftarrow \text{SE3}(\mathbf{p}, \hat{\mathbf{o}}, \hat{\mathbf{a}})$ ▷ Construct matrix
  output T_{shift}
end procedure

Algorithm 3 Calculation of discretised tool positions along the weld.

The greedy search can then be defined with Algorithm 4, where the aim is to create a list of homogeneous transformation matrices T (SE3). These can be defined with a point in 3D $\hat{\mathbf{p}}$ and robot weld tool frame vectors $\hat{\mathbf{a}}$, $\hat{\mathbf{n}}$, $\hat{\mathbf{o}}$, which are depicted with respect to the robot in Fig. 7.

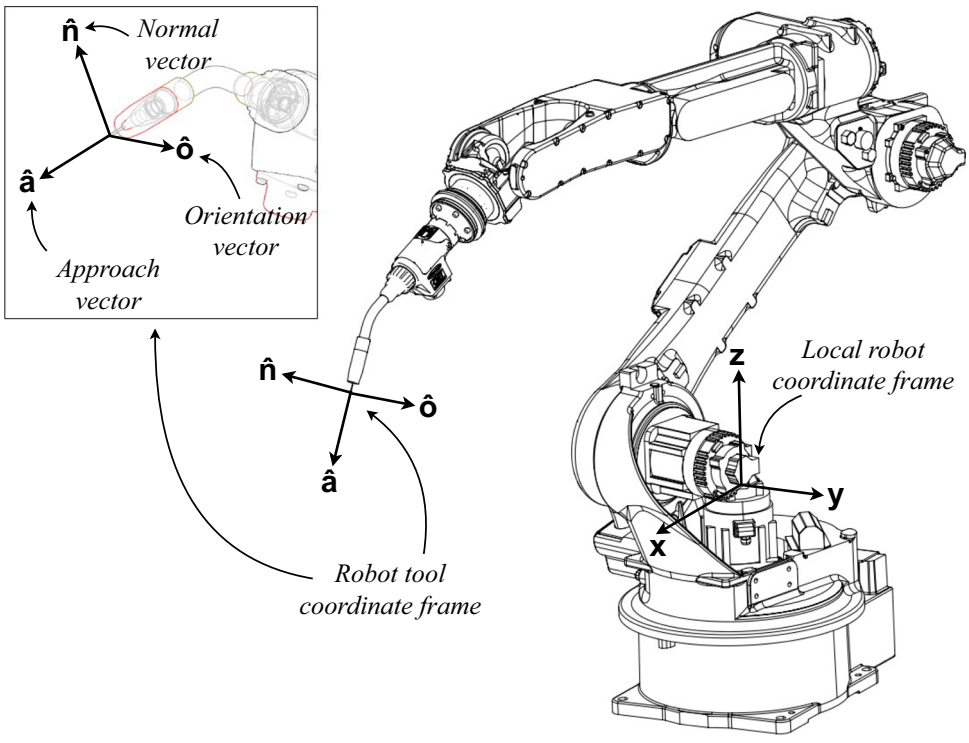


Fig. 7 Robot coordinate frame definitions. Local robot coordinate frame, tool frame definitions (from welding tip): approach vector $\hat{\mathbf{a}}$, orientation vector $\hat{\mathbf{o}}$ (reversed in depiction), and normal vector $\hat{\mathbf{n}}$

The normal vector $\hat{\mathbf{n}}$ is simply defined as the perpendicular $\hat{\mathbf{a}} \times \hat{\mathbf{o}} = \hat{\mathbf{n}}$. Additionally, robot poses q, and corresponding movement types and operations are included, resulting in a complete welding operation when interpreted by a local robot controller.

Input: Sequenced weld transformation target points $\mathbf{t}_s$, corresponding initial approach (bisector) vectors $\hat{\mathbf{a}}_s$, list of optimum and limit parameters σ_{opt}, discretisation step values γ_{step} in degrees, relevant collision objects c_{col}
Output: Generalised weld operations comprised of target poses defined as SE3 objects and robot poses $q_i(1 \times n_{\text{joints}})$

for each $(\mathbf{t}, \mathbf{a})$ in zip$(\mathbf{t}_s, \mathbf{a}_s)$ **do**
  while True **do**
    if $i < 1$ **then** ▷ First iteration
      Use optimum welding parameters σ_{opt} for α_w and α_s
    else if $i <$ lim(σ) **then** ▷ Still below search limit
      Pick less optimal parameters iteratively for α_w and α_s from σ_{opt}
    else ▷ No solution
      output False
    end if
    $\hat{\mathbf{a}}_{\text{shift}} \leftarrow$ ShiftApproach$(\hat{\mathbf{a}}_i, \hat{\mathbf{d}}_w, \alpha_w, \alpha_s)$
    Create o_{samples} array of orientations in circle perpendicular to $\hat{\mathbf{a}}_{\text{shift}}$ discretised by γ_{step}
    for each o_{step} in o_{samples} **do**
      $T \leftarrow$ SE3$(t, o_{\text{step}}, a_{\text{shift}})$
      $\text{q}_{\text{target}} \leftarrow$ ValidatePose(T, c_{col})
      if target$_q$ is not None **then**
        operation $\leftarrow$ Save$(T, \text{q}_{\text{target}})$
        success $\leftarrow True$
        break
      end if
    end for
    if success **then**
      break
    else
      $i++$ ▷ Sample iteration
    end if
  end while
end for

Algorithm 4 Generation of welding operations.

Essentially, the algorithm generates a discretised space of parameters that allows for calculating all the relevant robot poses throughout the welding operations. All that is left is to explore the solutions for each weld and check whether they comply with the manufacturing restrictions through a for-loop. The *ValidatePose* procedure is described in Algorithm 5. The algorithm loops through all the collision objects deemed relevant to the manufacturing, such as the part assembly decomposed convex hulls and simplified dummy shapes of surrounding structures. As the system is built around the robotics toolbox open-source library, the corresponding potential robot poses are calculated using a SciPy stiffness cost-minimiser function *ikine_min* [23]. Additionally, by setting the *qlim* flag to *True*, robot joint limitations can be considered. The final connecting movement motions can then be solved using PRM and RRT as outlined in Section 2.

Input: Transformation matrix T, collision objects c_{cols}
Output: Robot pose $q(1 \times n_{\text{joints}})$ if True; None if False.

procedure VALIDATEPOSE(T, c_{cols})
  $q_{\text{pose}} \leftarrow$ ikine_min$(T, qlim \leftarrow True)$ ▷ Calculate robot pose with joint limits
  if q_{pose} **then** ▷ Valid joint pose exists for T
    if Reached(q_{pose}, T) **then**
      if not isCollision$(q_{\text{pose}}, c_{\text{cols}})$ **then**
        output pose$_q$
      end if
    end if
  end if
  output None
end procedure

Algorithm 5 Validation of given poses.

The ikine_min function will return false if no solution is found within the minimiser limits. The final generalised program output of the robot program generation contains the following elements:

- Movement type
- Movement speed
- Target transformation, T
- Target robot pose, q
- Arc control.

where the movement type provides information about whether the robot movement should be considered a straight line in Cartesian space or joint space. Movement speed determines the robot's motion speed. This information, along with the transformation target T and pose q, is used by the robot controller to generate the corresponding trajectories. Finally, the arc control activates the welding torch and weld source program.

4 Implementation

In order to assess the viability and characteristics of the presented approach, the assemblies depicted in Fig. 8 were used as subjects for the program generation. The models were provided by industrial partners in the Norwegian maritime industry based on real aluminium use cases currently manufactured with manual welding.

4.1 Experimental setup

The presented automatic program generation method was implemented into a Siemens NX CAD environment using the Siemens NXOpen Python API. The offline programming environment, or manufacturing simulation server, was developed using the robotics toolbox for Python along with the Python Flask server module, elaborated in Section 3. The simulation manufacturing environment depicted in Fig. 1 was modelled based on one of the welding cells available at Manulab NTNU shown in Fig. 9. The cell consists of a Yaskawa Motoman GP25 equipped with a Fronius WF60I Robacta Drive CMT welding torch along with an MT1-500 positioner table. The joint kinematics between the robot welding arm and positioner was not considered for the scope of this paper. Instead, a set of pre-selected orientations for the positioner and part was used. Adding this functionality should not affect the results developed in this paper but rather result in faster and more efficient programs.

Fig. 9 The robot cell comprising a Yaskawa Motoman GP25 robot manipulator and a MT1-500 workpiece positioner

4.2 Welding program generation requirements

In addition to collision-free and singularity-free robot operations, the welding specifications are also part of the program generation. Based on the ISO6947:2019 standard, the allowed welding operations used for the implementation were set to PA, PB, PD, and PF to ensure the proper fusion of the aluminium joints. The allowed deviations from the work angle α_w were $\pm 5°$, and a stick angle α_s of 15° (optimal) to 40°.

4.2.1 Results

Table 1 tabulates the number of successful welding operations generated based on the four models comprised of

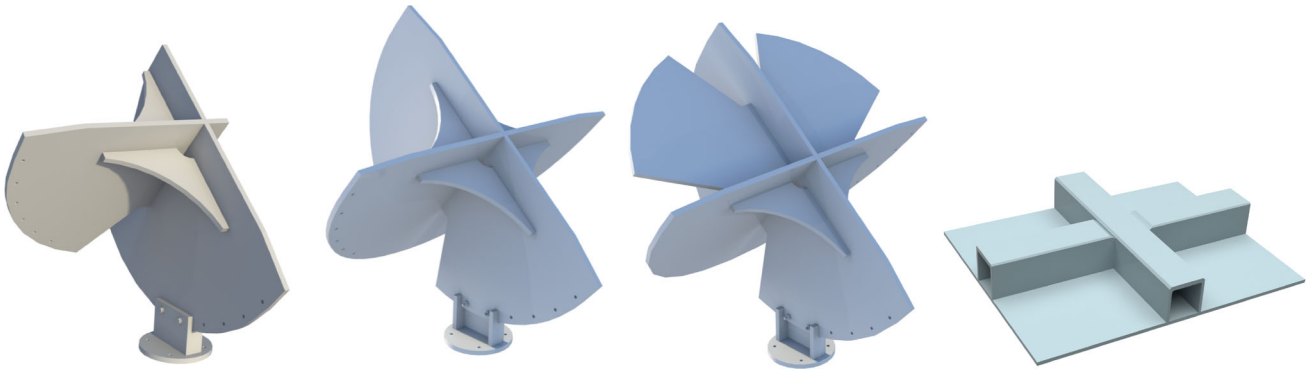


Fig. 8 Implementation assemblies: helideck aluminium nodes (a) configuration A, (b) configuration B, (c) configuration C, and (d) aluminium assembly stiffener representation

Table 1 Program generation for the implementation assemblies depicted in Fig. 8

Model	Parts	Generated welds	Total welds	Generation time
Node configuration A	4	20[a]	24	6 min 18 s
Node configuration B	6	32[ab]	42	9 min 29 s
Node configuration C	6	32[ab]	42	12 min 22 s
Stiffener geometry	4	14	14	3 min 25 s

[a]Obstructed by hole geometry as shown in Fig. 10

[b]A complete solution was not found with the given cell setup and weld position requirements

three different helideck node configurations and a stiffener representation, shown in Fig. 8. The table contains the total number of parts along with successfully generated welds versus the total number of located existing welds. The final column shows the total time taken for the generation. As will be elaborated in Section 5, the system is capable of generating rule-adhering programs for all welds, given that a solution exists for the defined criteria. For the helideck nodes, which require manual repositioning in actual production, several welds did not have solutions with the static fixture.

5 Discussion

As seen in Section 4, the system was able to generate generalised programs for all of the models provided, which contained different geometries with varying complexity. The experimental results using the assemblies depicted in Fig. 8 are tabulated in Table 1. The generated programs contained collision-free motions between a sequence of welds, where each weld had been characterised and classified based on the given weld topology and part orientation. While the presented approach was able to generate programs for each of the parts presented, the weld programs for the helideck nodes, Fig. 8a, b, and c do not contain valid welding operations for all of the possible welds, as simply no solutions adhering to the given rules existed for the given experimental setup shown in Fig. 9. These parts currently require repositioning several times in actual production with human welders in order to reach all the welds while obeying the specified allowed welding positions. However, the most problematic spots for the program generation were the holes in the structural corner plates bridging the main plates together, which contain welds on the inside, as shown in Fig. 10. From the experiments, it can be observed that the geometrical constraints of the holes lead to a loss of four welds for node A and eight welds for nodes B and C. One way to mitigate this issue could be by accommodating robotic welding accessibility at the design phase of the nodes in future design iterations. In essence, when excluding prohibited welding operations and other generation criteria, the system is capable of finding all solutions. The remaining welds could be accommodated by improving the weldability of the part in terms of design, modifying the manufacturing cell, or changing the criteria.

Another observation based on the results in Table 1 is that the generation speed for the programs performs relatively slow when compared to earlier implementations such as Larkin et al. [7]. The probable cause of this is attributed to the lower efficiency of the code used in the pose validation shown in Algorithm 5, which handles the generation of robot joint poses as well as checks for singularities, joint limits, and collisions. In theory, improving this aspect could improve the generation time significantly. However, to focus on the rule-adhering aspects of the automatic weld generation, this aspect was left to be studied in further work.

An important note is that the presented welding program generation method in its current state does not consider

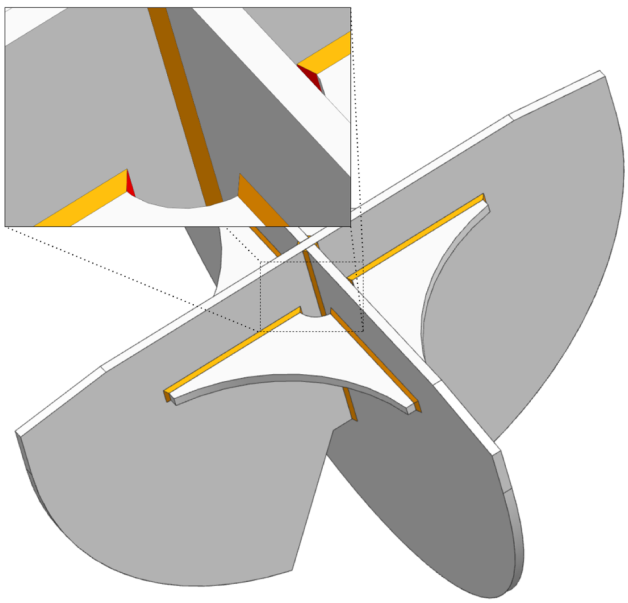


Fig. 10 Inaccessible welds located in holes (marked in red) with given restrictions and setup

the joint kinematics between the robot arm and positioner table. Instead, predetermined positions of the robot table based on experience were used for the algorithms. This was left to be studied in future works in order to focus on the rule-adhering aspects of the program generation.

The system presented in this paper mainly focuses on generating generalised robot programs consisting of base-level information so that a local robot control parser can incorporate it into different systems. Transitioning into fully automatic systems requires including a robot sensor system, such as a vision system, to account for the inaccuracies of real-world conditions. Additionally, an intelligent vision system for application during welding could compensate for material heat deformation and improve multi-pass welding quality. Such implementations have been proposed in Bedaka, Vidal, and Lin [28], and Rider-Marco et al. [29].

Additionally, as mentioned in Section 3, prior to program generation, a pre-calibration step should be performed. This ensures that the discrepancy between the generated program and the actual setup is manageable for the down-the-line adjustment and sensing systems. When adjusting this discrepancy, if too large, the local robot controller may run into joint limits or singularities if following the program targets blindly. However, it is possible to automate this step using vision-based sensing systems for initial part localisation, such as 3D cameras. The authors used a simple four-point measurement in the experiments to determine the offset transformation.

6 Conclusion

A framework and approach for generating rule-adhering robot welding programs from CAD aimed at products with varying geometries have been developed and presented. The aim of generating the rule adherence of welding programs is to ensure that the system output of the CAD to program generator corresponds to viable welding operations that can be used for critical structures. In order to evaluate the system performance, three geometrically different node configurations and a stiffener piece provided by industrial collaborators were used as input, along with allowable welding orientations and operations. For the more complex node configurations (which require reorientation when welded manually), 32 weld path programs out of 42 were generated based on the given criteria and boundaries. For the least complex node, 20 out of 24 welds were generated with the same criteria. All 14 weld programs were successfully generated for the stiffener representation. The incomplete results for some node configurations are attributed to the non-existence of solutions given the allowed operations, manufacturing cell setup, and the geometry of the part itself. Thus, the proposed system can correctly generate programs adhering to allowed welding operations as long as a solution exists.

Funding The research is supported by the Robotic Welding for Shipbuilding with Aluminium Hulls project (project number 295138), funded by the Research Council of Norway.

Data Availability The approach and results presented are developed on NXOpen API and supported to the knowledge of the authors with relevant sources.

Code availability The code is not available as it is also a part of the project deliverables. However, the relevant algorithms and methods which were used for the software implementation have been described in the article.

Declarations

Conflict of interest The authors declare no competing interests.

References

1. Verhagen WJC, Bermell-Garcia P, Dijk RECV, Curran R (2012) A critical review of knowledge-based engineering An identification of research challenges. Adv Eng Inform 26(1):5–15
2. Al-wswasi M, Ivanov A, Makatsoris H (2018) A survey on smart automated computer-aided process planning (ACAPP) techniques. Int J Adv Manufact Technol 97(1):809–832
3. Kuss A, Dietz T, Ksensow K, Verl A (2017) Manufacturing task description for robotic welding and automatic feature recognition on product cad models. Procedia Cirp 60:122–127
4. Epping K, Zhang H (2018) A sustainable decision-making framework for transitioning to robotic welding for small and medium manufacturers. Sustainability 10(10):3651
5. Shen W, Hu T, Zhang C, Ye Y, Li Z (2020) A welding task data model for intelligent process planning of robotic welding. Robot Comput Integr Manuf 64:101934
6. Fang W, Ding L, Tian X, Zheng F (2022) A robot welding path planning and automatic programming method for open impeller. The International Journal of Advanced Manufacturing Technology 1–12
7. Larkin N, Short A, Pan Z, Duin SV (2016) Automatic program generation for welding robots from cad. In: 2016 IEEE International conference on advanced intelligent mechatronics (AIM). IEEE, pp 560–565
8. Lundberg S (2016) Use of aluminium structures in the offshore industry. In: Key engineering materials, vol 710. Trans Tech Publ, pp 22–31

9. Wang X, Amdahl J, Egeland O (2022) Numerical study on buckling of aluminum extruded panels considering welding effects. Mar Struct 84:103230
10. Bedaka AK, Lin C-Y (2020) Cad-based offline programming platform for welding applications using 6-dof and 2-dof robots. In: 2020 International conference on advanced robotics and intelligent systems (ARIS). IEEE, pp 1–4
11. Sarivan Ioan-Matei, Madsen O, Waehrens BV (2021) Towards automatic welding-robot programming based on product model. In: Towards sustainable customization: Bridging smart products and manufacturing systems. Springer, pp 174–181
12. Pedersen MR, Nalpantidis L, Andersen RS, Schou C, Bøgh S, KrÜGer V, Madsen O (2016) Robot skills for manufacturing: From concept to industrial deployment. Robot Comput Integr Manuf 37:282–291
13. Pan Z, Polden J, Larkin N, Duin SV, Norrish J (2012) Recent progress on programming methods for industrial robots. Robot Comput Integr Manuf 28(2):87–94
14. Zheng C, An Y, Wang Z, Wu H, Qin X, Eynard B, Zhang Y (2022) Hybrid offline programming method for robotic welding systems. Robot Comput Integr Manuf 73:102238
15. Bottazzi VS, Cruz Fonseca JF (2005) Off-line robot programming framework. In: Joint international conference on autonomic and autonomous systems and international conference on networking and services-(icas-isns' 05). IEEE, pp 71–71
16. Neto P, Mendes N, Araújo R, Norberto Pires J, Moreira AP (2012) High-level robot programming based on cad: dealing with unpredictable environments. Industrial Robot: An International Journal
17. Ferreira LA, Figueira YL, Iglesias IF, Souto MÁ (2017) Offline cad-based robot programming and welding parametrization of a flexible and adaptive robotic cell using enriched cad/cam system for shipbuilding. Procedia Manufacturing 11:215–223
18. Alexander Z (2021) Programming of welding robots in shipbuilding. Procedia CIRP 99:478–483
19. Hillbrand C, Frank G (2012) Knowledge-based automated programming of welding robots for lot-size one products. In: Engineering systems design and analysis, vol 44878. American Society of Mechanical Engineers, pp 27–36
20. Zheng C, Xing J, Wang Z, Qin X, Eynard B, Li J, Bai J, Zhang Y (2022) Knowledge-based program generation approach for robotic manufacturing systems. Robot Comput Integr Manuf 73:102242
21. Li J, Lu Y, Shen N, Fan J, Qian H (2022) A knowledge-based method for tool path planning of large-sized parts. Expert Systems with Applications p 117685
22. Tran TA, Lobov A, Kaasa TH, Bjelland M, Midling OT (2021) CAD Integrated automatic recognition of weld paths. Int J Adv Manufact Technol 115(7):2145–2159
23. Corke P, Haviland J (2021) Not your grandmother's toolbox–the robotics toolbox reinvented for python. In: IEEE International conference on robotics and automation
24. Coumans E, Bai Y (2016) Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org
25. Mamou K, Lengyel E, Peters A (2016) Volumetric hierarchical approximate convex decomposition. In: Game engine gems 3. AK Peters, pp 141–158
26. Mu T, Ling Z, Xiang F, Yang D, Li X, Tao S, Huang Z, Jia Z, Su H (2021) Maniskill: Generalizable manipulation skill benchmark with large-scale demonstrations. arXiv: 2107.14483
27. Grinberg M (2018) Flask web development: developing web applications with python. " O'Reilly Media Inc"
28. Bedaka AK, Vidal J, Lin C-Y (2019) Automatic robot path integration using three-dimensional vision and offline programming. Int J Adv Manufact Technol 102(5):1935–1950
29. Marco-Rider J, Tran TA, Njaastad EB, Egeland O, Lobov A (2022) Supporting robotic welding of aluminium with a laser line scanner-based trigger definition method. In: 2022 IEEE 20Th international conference on industrial informatics (INDIN). IEEE, pp 399–406

Affiliations

Tuan Anh Tran[1] · Eirik Bjørndal Njåstad[2] · Ole Terje Midling[3] · Morten Bjelland[4] · Andrei Lobov[1]

Eirik Bjørndal Njåstad
eirik.njaastad@sintef.no

Ole Terje Midling
ole.terje.midling@m-a.no

Morten Bjelland
morten.bjelland@leirvik.com

Andrei Lobov
andrei.lobov@ntnu.no

[1] Department of Mechanical and Industrial Engineering, Norwegian University of Science and Technology, Richard Birkelands vei 2b, Trondheim, 7034, Trøndelag, Norway

[2] Department of Production Technology, SINTEF Manufacturing AS, Trondheim, Norway

[3] Marine Aluminium AS, Avaldsnes, Norway

[4] Leirvik AS, Stord, Norway