A VOXEL-BASED NUMERICAL METHOD FOR COMPUTING AND VISUALISING THE WORKSPACE OF OFFSHORE CRANES

Lars I. Hatledal∗
Filippo Sanfilippo
Yingguang Chu
Houxiang Zhang
Department of Maritime Technology and Operations
Aalesund University College
Postbox 1517, 6025 Aalesund, Norway
Email: [laht, fisa, yich, hozh]@hials.no

ABSTRACT
Workspace computation and visualisation is one of the most important criteria in offshore crane design in terms of geometry dimensioning, installation feasibility and operational performance evaluation. This paper presents a numerical method for the computation and visualisation of the workspace of offshore cranes. The Working Load Limit (WLL) and the Safe Working Load (SWL) can be automatically determined. A three-dimensional (3D) rectangular grid of voxels is used to describe the properties of the workspace. Firstly, a number of joint configurations are generated by using the Monte Carlo method, which are then mapped from joint to Cartesian space using forward kinematics (FK). The bounding box of the workspace is then derived from these points, and the voxels are distributed on planes inside the box. The method distinguishes voxels by whether they are reachable and if they are on the workspace boundary. The output of the method is an approximation of the workspace volume and point clouds depicting both the reachable space and the boundary of the workspace. Using a third-party software that can work with point clouds, such like MeshLab, a 3D mesh of the workspace can be obtained. A more in-depth description and the pseudo-code of the presented method are presented. As a case study, the workspace of a common type of offshore crane, with three rotational joints, is computed with the proposed method.

INTRODUCTION
The workspace of a robot manipulator is defined as the set of all points the robot can reach [1]. In the maritime field, the workspace of an offshore crane often relates to the terms of Working Load Limit (WLL) or Safe Working Load (SWL) [2]. WLL is the maximum working load of lifting, while SWL is the value of WLL applied with a safe factor. Each crane has a load chart that specifies the crane’s capabilities -detailing its features and how its lift capacity varies in different positions. Such a chart is usually realised when the design phase is completed. In particular, a posteriori estimation of the crane load capacity is performed according to the given dimensions of the crane links and to the decided crane joints load capabilities. In this way, the crane’s WLL ends up to be just an indirect consequence of a priori design choice. This makes it really difficult to decide the crane characteristics according to the application scenario during the design stage.

To overcome these challenges, a numerical method for the computation and visualisation of the workspace of different offshore cranes is presented in this work. The WLL and the SWL can be automatically determined. This algorithm can be used as a trial and error designing tool during the preliminary design phase of a crane, when a complete solid model is not ready or not as easy to be modified. In detail, this method can be used to quickly generate the workspace in order to find the proper links length and corresponding cylinder’s size. A 3D rectangular grid of voxels is dynamically generated to describe the properties of the considered crane’s workspace. A voxel represents a value on a regular grid in 3D space. Firstly, a number of joint configurations are generated by using the Monte Carlo method, which

∗ Address all correspondence to this author.
are then mapped from joint to Cartesian space using FK. Then, a bounding box that encloses all these points is obtained. Successively, a set of voxels are equally distributed inside the box. All the voxels are labeled according to whether they are reachable by the crane and whether they lie on the workspace boundary or not. Next, a point cloud is generated by considering the center of each voxel. This approximation is justified by the fact that it allows for a cleaner visualisation of the points, and it also significantly reduces the number of points needed to be drawn on screen. This approach makes it possible to easily incorporate the characteristics of the derived workspace into data-driven documents. Furthermore, as the size and location of each voxel is known, they can be used for collision detection. Also, if a set of upper bound joint torques have been defined for the WLL and SWL, the algorithm will check whether or not these values are being violated, and assign colors to the voxels accordingly. While this paper mainly targets the workspace computation of crane structures, the method can be applied to get a volume approximation and 3D visualisation of any type of serial kinematic chain. However, the WLL- and SWL visualisation requires a non-redundant structure to make proper sense.

The paper is organized as follows. A review of the related research work is given, followed by the description of the proposed numerical method for the computation and visualisation of the workspace of different offshore cranes. Successively, a case study is presented considering the computation of the workspace of a common type of offshore crane, with three rotational joints. Finally, conclusions and future works are outlined.

RELATED WORKS
There have been limited attempts in literature addressing the workspace computation of offshore cranes. However, several related works can be found regarding robotic manipulators. In general, cranes are larger in size, but have less degrees of freedom (DOFs) than robotic arms or other manipulators. Previous studies on the workspace of robotic manipulators can be identified in three main categories: analytical, geometric, and numerical methods. Analytical and geometric methods determine the boundary surfaces of the manipulators by using mathematical equations and geometric principles. Concerning this approach, an algorithm for determining the workspace of a parallel kinematic machine was presented in [3]. Similarly, a numerical formulation for the placement and reachability of open-loop robotic manipulators was introduced in [4]. In this last study, the boundary curves or surfaces of the workspace were yielded from the singularities of the end-effector’s position vector and constrained by the joint limits. Analogously, a method based on the development of analytical criteria for determining singular behaviors of a robotic manipulator was outlined in [5]. In this last case, the workspace is generated when all the singularities are obtained and formulated into a set of boundary functions. Generalizing the concept of workspace generation and extending the idea to any kind of kinematic chain, a diffusion-based algorithm for workspace generation of hyper-redundant (snake-like) manipulators was introduced in [6]. This diffusion equation is a partial differential equation defined on the motion Special Euclidean group in an N-dimensional space, SE(N), and describes the evolution of the workspace density function, depending on manipulator length and kinematic properties. However, geometric and analytical methods are usually not easy to implement, due to the complexity in finding the boundary expressions and matrix manipulation involved in the manipulator’s kinematics. In addition, most of these methods can only handle certain specific manipulator models. Consequently, for manipulators and robot with multiple DOFs, it is rather difficult and impractical to obtain the boundary expressions using geometric or analytical methods.

Numerical methods have been increasingly investigated in recent years, as they are relatively simple and more flexible to be adapted to different types of manipulators. Besides, these methods are better suited for automatic calculations with computers. In [7], an integrated approach based on numerical calculation and solid modeling software- was presented to create a 3D robot workspace. Firstly, a numerical method based on the Monte Carlo [8] method was used to generate the planar boundary curves of a spatial robot in its main working plane. Then, the 3D shape and volume of the robot workspace were generated by using a computer-aided design software, by importing the boundary curves and adopting some simple operations. However, this method may result in errors in some cases when connecting the points to form the boundary contour curve. In [9], the method was improved by searching the boundary points in two dimensions instead of one. This numerical method gives approximations of the boundary, which means accuracy can be an issue for certain applications. For this reason, a further improvement of this approach was presented in [10] by using the Beta distribution [11], in generating random variables of the joints in order to obtain more accurate boundary curves.

In the case of offshore crane operations, accuracy is not the major issue considering the size of the workspace of offshore cranes. What is more significant is the workspace relating to the WLL and to the SWL. To the best of our knowledge, a general method that allows for automatically generating the workspace and to visualise both the WLL and the SWL for different cranes has not been released yet.

A NUMERICAL METHOD FOR THE COMPUTATION OF THE WORKSPACE OF DIFFERENT OFFSHORE CRANES
In this section, the proposed numerical method for the computation and visualisation of the workspace of different offshore cranes is outlined.

The flowchart of the proposed algorithm is shown in Fig. 1.
Acquire manipulator model
Generate joint configurations
Calculate bounding box
Assign voxels to grid
Calculate voxel properties
Generate point clouds
Generate mesh
End

Legend
Essential step
Optional step

FIGURE 1: FLOWCHART OF THE PROPOSED ALGORITHM

It should be noted that the last two steps are optional operations, meaning that they only affect the visuals. Fig. 2 illustrates some of the key steps, with the visible points marking the center location of a voxel. Initially starting with a three-dimensional grid of voxels that fills the bounding box in Fig. 2a, the points are reduced to only those that are reachable by the end-effector in 2b. Successively, only the points that represents the boundary are left in Fig. 2c. The algorithms also computes the normal vector of these points, which are used to generate the solid 3D mesh seen in Fig. 2d. In the following, the key steps are further described.

Acquire Manipulator Model The main iteration loop starts acquiring the FK equations describing the location of each DOF and the end-effector of the desired manipulator model. Multiple equations are needed in order to build the Jacobian matrices required to calculate how the self weight of the crane structure affects the joint efforts. In particular, the kinematics equations of a serial chain of \( n \) links, with joint parameters \( \theta_i \), are given by [1]:

\[
T_A = ^n_0 T = \prod_{i=1}^{n} t_i^{-1} T(\theta_i),
\]

where \( t_i^{-1} T(\theta_i) \) is the general homogeneous transformation matrix from the frame of link \( i \) to link \( i - 1 \).

Generate Random Joint Configurations Joint configurations are generated using the Monte Carlo method. The final number of configurations are given by the control variable, \( \theta_{num} \). As a random sampling numerical method, the Monte Carlo method is relatively simple and flexible to apply, and hence is suitable for workspace generation. All the generated configurations are within the limits of the rotational angles for revolute joints and/or metric displacement for prismatic joints. For each uniformly random set of variables in the joint space, a Cartesian position in the workspace of the robot’s end-effector can be calculated using FK. This process is repeated a preset number of times, in order to obtain a good approximation of the unknown workspace distribution. A large number of points are required for getting good results in subsequent steps of the method.

Calculate Bounding Box The bounding box of the workspace is then derived from these points. The minimum or smallest bounding or enclosing box is the box with the smallest volume measure within which all the points lie.

Assign Voxels to Grid Successively, the bounding box is partitioned into a three-dimensional grid of voxels, where the number of voxels along each coordinate axis is given by a scalar \( n \). The total number of voxels in the grid is then \( n^3 \). When assigning a voxel to a location on the grid, this location is stored as a property of the voxel along with its volume.

Calculate Voxel Properties In addition to the position and volume information found in the previous step, the voxel properties also includes information on whether or not the voxel is reachable by the end-effector and whether or not it lies on the workspace boundary. If it does lie on the boundary, the normal vector is calculated as seen in Fig. 3. Calculating the normals are very important in order to get a good result when creating a mesh from the boundary point cloud at a later stage, because a surface reconstruction algorithm can use the information to correctly orient the mesh faces. Additionally, the algorithm checks if a singularity could occur when moving the end-effector within the enclosed volume of a voxel.

The pseudo-code for finding the reachable voxels, calculating the joint efforts when the end-effector resides within those voxels and the total workspace volume, is given in Algorithm 1. How to determine which voxels lie on the boundary, and successively calculate the normal vectors, is shown in Algorithm 2. It should be noted that \( n \) determines the precision to display, meaning that a larger number of voxels produces better results. When increasing \( n \), \( \theta_{num} \) should increase as well, in order to avoid false
Algorithm 1 Compute voxels.

Require: \( n \geq 1 \) \{determines how many voxels to use along each coordinate axis\}

Require: \([\hat{\theta}, \hat{X}]\) \{random joint configurations and corresponding Cartesian coordinates\}

Require: \( \tau_{\text{max}} \) \{max joint torques\}

1: \( \text{voxels} \leftarrow \emptyset \)
2: \( \text{volume} \leftarrow 0 \)
3: \( \text{bb} \leftarrow \text{computeBoundingBox(points)} \)
4: \( \mathbf{x}, \mathbf{y}, \mathbf{z} \leftarrow \text{linspace}(\text{bb.min}, \text{bb.max}, n) \)
5: for \( i = 1 \) to \( n - 1 \) do
6:   for \( j = 1 \) to \( n - 1 \) do
7:     for \( k = 1 \) to \( n - 1 \) do
8:       \( \text{voxel} \leftarrow \text{a new voxel with bounding box given by the minimum vector} [x[i], y[j], z[k]] \) and maximum vector \( [x[i+1], y[j+1], z[k+1]] \)
9:       for \( X \in \hat{X} \) do
10:          if \( \text{voxel} \) contains \( X \) then
11:             mark \text{voxel} as reachable
12:             \( \theta \leftarrow \) the configuration that led to \( X \) (\( \theta \in \hat{\theta} \))
13:             if \( \det(J(\theta)) = 0 \) then
14:                mark \text{voxel} as containing a singularity
15:          end if
16:          \( \tau = J^T F_1 + J^T F_2 + \ldots + J^T F_n \)
17:          check if \( \tau \) violates WLL or SWL
18:     end if
19:   end for
20: end for
21: if \( \text{voxel} \) is reachable then
22:     \( \text{volume} \leftarrow \text{volume} + \text{volume of voxel} \)
23: end if
24: assign color to the \text{voxel} based on WLL and SWL
25: \( \text{voxels}[i][j][k] \leftarrow \text{voxel} \)
26: end for
27: end for

Algorithm 2 Locate boundary voxels.

Require: \( \text{voxels} \)

1: for \( i = 1 \) to \( n - 1 \) do
2:   for \( j = 1 \) to \( n - 1 \) do
3:     for \( k = 1 \) to \( n - 1 \) do
4:       \( \text{voxel} \leftarrow \text{voxels}[i][j][k] \)
5:       if \( \text{voxel} \) is reachable then
6:         mark the \text{voxel} as a boundary voxel if a neighbor
7:         is not reachable or if it is located on an edge
8:       end if
9:     end for
10: end for
11: end for
voids inside the workspace. Naturally, the computational demand related to the computation of the workspace increases with higher precision. The steps in the 3rd layer for-loop in Algorithm 1 and 2 may be parallelised in order to decrease the computation time. How the volume approximation and computing time changes with regards to the parameters \( n \) and \( \theta_{\text{num}} \) is shown in Table 1. The data was collected using a multi-threaded Java implementation of the algorithm, running on a Intel i7-4770 CPU. The resulting meshes can be viewed in Fig. 4.

In order to find the WLL- and SWL, the principle of virtual work [1] is applied. In particular, the principle of virtual work shows that the Jacobian provides a relationship between joint torques and the resultant force and torque applied by the crane’s end-effector. According to this principle, the following equation is valid:

\[
J^T F = \tau
\]
FIGURE 3: BOUNDARY NORMALS

TABLE 1: WORKSPACE ESTIMATION

<table>
<thead>
<tr>
<th>( \theta_{num} )</th>
<th>( n )</th>
<th>Volume ( [m^3] )</th>
<th>Time ( [ms] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>50000</td>
<td>15</td>
<td>3307</td>
<td>221</td>
</tr>
<tr>
<td>150000</td>
<td>25</td>
<td>2967</td>
<td>442</td>
</tr>
<tr>
<td>300000</td>
<td>35</td>
<td>2810</td>
<td>765</td>
</tr>
<tr>
<td>500000</td>
<td>45</td>
<td>2728</td>
<td>3473</td>
</tr>
<tr>
<td>1000000</td>
<td>55</td>
<td>2679</td>
<td>8377</td>
</tr>
<tr>
<td>1800000</td>
<td>65</td>
<td>2665</td>
<td>29058</td>
</tr>
</tbody>
</table>

where \( J \) is the Jacobian matrix of the crane, \( \mathbf{F} \) is the force vector exerted from the crane’s end-effector to the environment and \( \tau \) is the static joint torques. The above equation does, however, not take into account the self-weight of the crane structure. In order to do so the equation is expanded to include the forces exerted from the attached links:

\[
J_1^T \mathbf{F}_1 + J_2^T \mathbf{F}_2 \ldots J_n^T \mathbf{F}_n = \tau \tag{3}
\]

where \( J_n \) is the Jacobian matrix describing the point where the force \( \mathbf{F}_n \) is applied. \( \tau_{\text{max}} \) is known from the given characteristics of the crane actuators and is dependent on the cylinder’s size and length, and is used to describe the WLL- and SWL by applying a safe factor: \( \tau_{\text{max}} \alpha \), where \( \alpha \in [0, 1] \) is a real number.

The algorithm asserts the result of equation 3 against the maximum static joint torques given by the WLL- and SWL and assigns colors to the voxels accordingly. I.e a voxel is assigned the color green when the resultant static joint torques are within the SWL limits, yellow when they are within the WLL ones and red otherwise. This step should be recalculated whenever the weight of the end-effector load changes.

Generate Point Clouds The output of this step are the point clouds depicting both the reachable space and the boundary of the workspace. Each point in the point cloud corresponds to the center position of a voxel. In the case of the boundary point cloud, the normal vector for each point has been computed. Once the point clouds have been extracted from the voxels, they can be visualised or saved in a common point cloud format for further processing in any compliant tool.

Generate Mesh A solid model of the crane’s workspace may help in achieving better comprehension of the workspace shape or volume in 3D space. Using the oriented boundary point cloud found in the previous step and saving it in the common .PLY format for point clouds, the point cloud can be imported into a third party tool, such as MeshLab, and successively the 3D mesh may be obtained using a surface reconstruction algorithm.

CASE STUDY

In this section, a common type of offshore hydraulic crane is considered as a case study. In particular, a knuckle boom crane, which is shown in Fig. 5, is studied. The crane has three rotational joints that are actuated by a hydraulic motor and cylinders. According to the frame assignments in Fig. 5, the Denavit-Hartenberg (D-H) table [1] of the considered model is shown in Table 2. In this case study, the maximum joint torques, \( \tau_{\text{max}} \), was assigned the values \( \tau_1=100000 \, \tau_2=220000 \, \tau_3=70000 \) N·m. The weight of the considered links are \( L_1=1500 \, L_2=2200 \, L_3=1300 \) kg. As a simplification the center of gravity is thought to be at the center of each link. The weight of the actuators are not considered. The weight of the payload was set to 1000 kg.

The proposed method has been implemented in the Java programming language as part of a larger framework for simulation and visualisation of offshore cranes further described in [12]. The framework has been designed with the Model-View-Controller (MVC) pattern in mind, so that the visualisation

\[
\begin{array}{cccccc}
 i & \alpha_{i-1} & a_{i-1} & d_i & \theta_i \\
 1 & 0 & 0 & L_1 & \theta_1 \\
 2 & \frac{\pi}{2} & 0 & 0 & \theta_2 \\
 3 & 0 & L_2 & 0 & \theta_3 \\
 4 & 0 & L_3 & 0 & \theta_4 \\
\end{array}
\]

\( D\)-\( H \) TABLE OF THE KNUCKLE BOOM CRANE
module is independent and interchangeable from the simulation. Currently, WebGL is used as the visualisation platform, and the output of the proposed algorithm is demonstrated within this environment in Fig. 6, where the WLL- and SWL for the previously mentioned knuckle boom crane has been visualised. As described earlier, a green dot corresponds to the SWL, a yellow one to the WLL and a red is outside the capacity of the crane. The solid workspace mesh seen in the context of the simulation environment in Fig. 7 is retrieved using the third party tool MeshLab, which takes the oriented point cloud given by the boundary voxels as input- and outputs a .OBJ 3D model that can be viewed in most common 3D tools.

CONCLUSION AND FUTURE WORK

In this paper, a novel and effective method has been presented for the computation and visualisation of the workspace of different manipulators. Given an end-effector load, the WLL and the SWL can be automatically determined. This algorithm can be used as a trial and error designing tool during the preliminary design phase of a crane, when a complete solid model is not ready or not as easy to be modified. This approach allows for a fast generation of the crane workspace, so that the proper link’s length and the corresponding cylinder’s size can be efficiently determined according to the design requirements. In the future, the presented method can be coupled with some kind of optimisation method, so that the trial and error design approach can be tackled in a more efficient way. Any of the artificial intelligence methods implemented in [13] could be integrated with the proposed method to realise a reverse engineering method that allows for determining the crane link’s length and the corresponding cylinder’s size in a dynamic way as suggested in [14]. Furthermore,
using the known boundary points and normals, the algorithm could be improved to also handle the surface reconstruction in 3D, instead of having to rely on third-party software.

Acknowledgment

The project work of this paper is financially sponsored by a grant from the Norwegian National Research Foundation (Innovation Projects for the Industrial Sector MAROFF, 525564).

REFERENCES


