International Journal of Mechanical Engineering and Applications
2021; 9(6): 90-97
http://www.sciencepublishinggroup.com/j/ijmea
doi: 10.11648/j.ijmea.20210906.12
ISSN: 2330-023X (Print); ISSN: 2330-0248 (Online)

Real-time Simulation Methods for Robots with a Flexible Arm Based on Computer Graphics Technology

Hiroshi Seki[1, *], Katsunori Ueno[2], Katsuhiko Hirano[2]

[1]Research and Development Group, Hitachi, Ltd., Hitachi-shi, Japan

[2]Nuclear Engineering and Product Division, Hitachi-GE Nuclear Energy, Ltd., Hitachi-shi, Japan

Email address:
hiroshi.seki.mf@hitachi.com (H. Seki)

[*]Corresponding author

To cite this article:
Hiroshi Seki, Katsunori Ueno, Katsuhiko Hirano. Real-time Simulation Methods for Robots with a Flexible Arm Based on Computer Graphics Technology. *International Journal of Mechanical Engineering and Applications.* Vol. 9, No. 6, 2021, pp. 90-97.
doi: 10.11648/j.ijmea.20210906.12

Received: October 26, 2021; **Accepted**: November 23, 2021; **Published**: December 2, 2021

Abstract: A robot with a flexible arm that is controlled with a remotely operated water-pressure mechanism has been developed for dismantling objects that are heavily contaminated by radioactive materials during decommissioning of nuclear power plants. The objective of this research is to develop a process planning support system and method that can improve the accuracy of estimating the time required by the robot to complete its dismantling activity, support recovery from delays, and determine the feasibility of conducting a dismantling process in the reactor building. Since the flexible arm has a more complex mechanism and shapes those that of multi-axis robots, unique movable arm structures were modeled with a tool for three-dimensional (3D) computer graphics (CG) technology. The 3D CG model was used to make valid operation sequences for planning the motion of the robot. With the help of a prototype system, motion planning can perform to calculate the duration needed for the robot to complete its operation. The calculated duration is then used for updating the planned duration of a specific activity in a dismantling schedule. To plan the robot behaviors for complex dismantling processes, a simplified planning method with few virtual controllers based on a spline inverse kinematics (IK) with a non-uniform rational B-spline (NURBS) curve for an arm comprised of pistons and cylinders pressurized by water was studied. A prototype system for planning the behaviors of the robot was evaluated, and it was confirmed that the movement trajectory of the robot and the three-dimensional isometric display could be visualized using the mesh model generated from point-cloud data used to make the environment model of the robot. It was also confirmed that the operations involved in a specific activity of the robot could be completed within the duration determined in the simulation.

Keywords: Decommissioning, Computer Graphics, Forward Kinematics, Inverse Kinematics, Non-Uniform Rational B-Spline (NURBS) Curve

1. Introduction

In current decommissioning projects for high dose-rate areas in nuclear power plants (NPPs), it is necessary to reduce occupational exposure for workers; thus, portions of dismantling work must be done with remotely operated robots. It is necessary to dismantle radioactive waste safely and economically: amounts of such waste are estimated to be 3.6% by weight in commercial reactors [1]. Since the number of NPP decommissioning projects is likely to increase in the future, the development of robotic dismantling technology is becoming more important. The results of a cost analysis of NPP decommissioning projects show that cutting and dismantling account for a large portion of the cost [2]. So far, a system that visualizes calculated radiation dose rates in a virtual reality (VR) environment has been developed for mitigating high dose-rate environments inside an NPP [3]. Dose-rate distributions can be calculated with high accuracy based on the geometry of the facility, materials, and radioactive sources [4]. Moreover, computer simulations of the processes of cutting equipment and waste generation have been developed [5, 6]. In particular, the amount of waste generated can be predicted accurately on a volumetric basis

by using 3D CAD models [7, 8].

The radiation resistance of the devices making up the robot is an important issue. To address this issue, we have developed a robot arm with a flexible structure and a hydraulic drive mechanism. Since the flexible arm does not have an electronic controlling mechanism in the robot's body, it has high resistance to radiation [9]. This kind of robot was described in a survey paper on soft robotic manipulators, which analyzed structures ranging from multi-axis arms to redundant multi-axis arms, as well as arms with many joints and arms that can bend continuously like living organisms [10]. The flexible arm developed by Hitachi is positioned between a redundant multi-axis arm and an arm with many joints.

Autonomous and/or teleoperated control of robots in a certain environment was simulated with Unity 3D virtual reality software and MATLAB using shared memory for real-time feedback [11]. A dual-manipulator mobile robotic system was developed for nuclear decommissioning and inverse kinematics was used for controlling the system [12]. In particular, a multi-objective genetic algorithm was used to stabilize the motion of robots through inverse kinematics [13]. Visualizations of the motions of robots were simulated by an automated generation of the kinematics equations of the robots and analytical solving their motion planning equations subject to time-varying constraints, behavioral objectives, and modular configuration [14].

Section 2 below describes the issues to be resolved and the objectives of this research. Section 3 describes a CG robot model that is suitable for the visualization of kinematics. The methods for simulating robots with flexible arms in real-time are described in section 4, and the results of simulations are discussed in section 5. Section 6 is the conclusion.

2. Issues to Be Resolved and Objectives of the Research

The high dose rates encountered during decommissioning of NPPs have made it essential to deploy remotely operated robots. Here, it is important to plan how long it will take for a robot to perform the desired actions, such as cutting and removing obstacles or operating a manual valve handle on behalf of a human, in an environment that is similar to the worksite. In addition, when operating in the field, it is necessary to check whether the robot operates without collisions in an environmental 3D model that is created in advance of its operation. The following functions are required for a real-time robot simulation to meet these requirements (Figure 1 depicts them as a system configuration).

We have developed a function to import designed 3D models and point cloud data measured in the field and incorporate them in an environmental model for a VR simulator.

Simulation of kinematics to calculate the coordinates of the robot's crawler and the rotation angles of arm joints by the operation commands in the VR environment, or a function to visualize the posture and motion of the actual robot in real-time.

Calculation of the duration needed to complete an activity on schedule that is based on the results of a VR simulation.

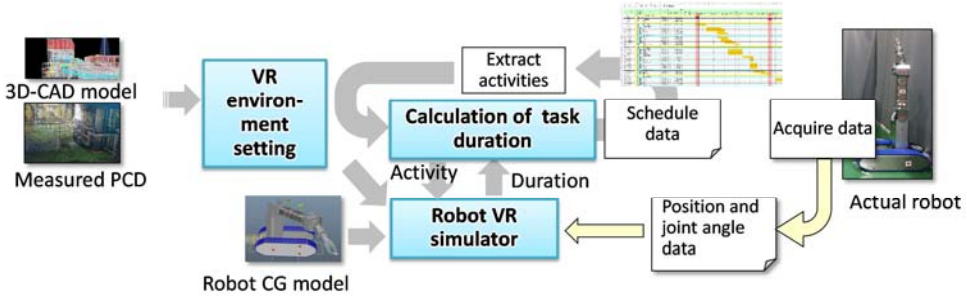

Figure 1. *System configuration of real-time robot simulation.*

3. Building a Robot CG Model Suitable for Visualization of Kinematics

3.1. Requirements for Real-Time Robot Simulation

This paper describes the research and development of an execution-time calculation of a single task using the VR environment described in (a) and the robot VR simulation described in (b) above. For the VR environment, it is necessary to address the following technical issues.

An efficient and rapid method for generating shape models that accurately reproduces in-situ 3D measurement data.

Determination of the usage conditions under which the models are generated, and the screen is updated with the complexity of the model, which depends on the upper limit of the number of constituent polygon meshes, so that the robot can be used interactively in a VR environment without a time delay.

In addition, the following technical issue regards the robot VR simulation function.

Development of a simple method to determine the robot's movable space and operation time.

Previous studies have investigated these issues and have set forth a development strategy [15]. When decommissioning an NPP, there are places where equipment and piping are not modeled in 3D CAD data. It is not always possible to prepare a VR environment in advance that matches the on-site conditions.

As for the geometric shape model in (a), mesh models can

be created from 3D measured point cloud data. The 3D point cloud data contain errors, and a method called corner-aware neighborhood (CAN) is used to generate a smoothed mesh shape in which the errors are suppressed by taking the normal vectors near the corners of the object [16]. However, this method cannot be used for plant and piping components whose shapes change significantly. In this research, we developed a method to connect meshes based on the similarity of normal vectors to mesh planes generated from neighboring point-cloud data to reproduce equipment and piping shapes.

The real-time usage conditions of the VR environment in (b) are described using Unity, a widely used VR software development tool [17]. Here, a mobile device can handle a mesh model with no more than 10^5 vertices, while a PC can handle a model with no more than 10^6 vertices.

We decided to express the geometric shape with fewer meshes than the upper limit for PCs so that we could use real-time processing.

A previous study examined a simple method of evaluating the robot's operational space and operation time (c); in particular, it studied how to automatically plan the movement route of the robot based on a self-position estimation on a 3D map [18]. However, the resulting plans of that method cannot account for complex operations such as the removal of obstructing objects or the opening and closing of valves. Hence, to ensure that the robot is operated in a way that conforms to the physical constraints of its environment, we devised a real-time collision checking process.

3.2. Controlling Postures of Flexible Arms with a Spline IK Method

To check collisions operation in the confined three-dimensional environment of a plant, it is necessary to recreate the detailed motions of each part of the robot. Figure 2 is an enlarged picture of the flexible arm parts of the robot. The arm parts are supported by plate springs made of metal and are reinforced so that the initial state is a posture on a straight line. Inside the plate spring, pistons are attached to the joints. The direction in which the piston is compressed or expanded is controlled by the pressure of the water in the cylinder, and the movements of the individual pistons affect the flexible posture of the entire arm. To estimate the posture of the individual parts from the overall movement, a virtual joint structure called a rig was created, and the states of components such as pistons were calculated and visualized.

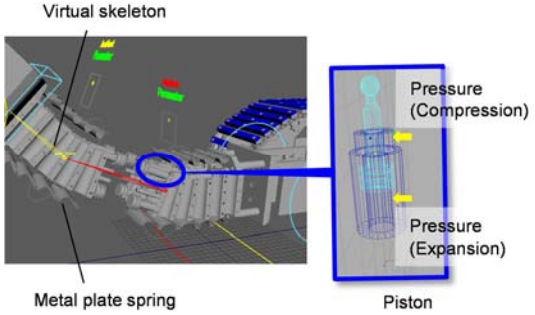

Figure 2. *Image of robot with piston movements enabled by water pressure.*

Inverse kinematics (IK) was used to accurately simulate the robot's behavior and to make it easy to operate. To simplify the discussion of the problem, a case where the behavior of each joint is simulated in a two-dimensional plane by calculating backward from the target position of the robot's hand in the case of a joint arm is shown in Figure 3.

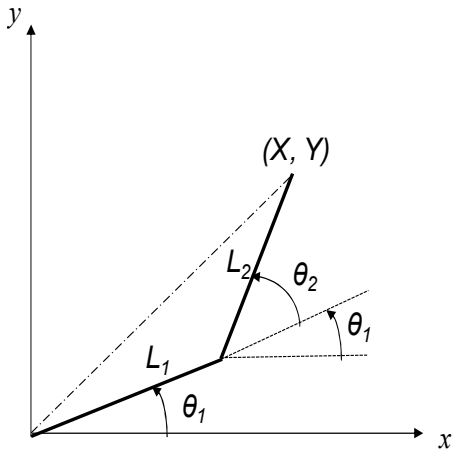

Figure 3. *Determination of angles from the location of the end-effector of an arm composed of two links.*

We assume that an arm has two links of length L_1 and L_2 and that the links are connected at angles θ_1 and $\theta_1 + \theta_2$ to the X-axis. The relationship between the time variation of the hand coordinates (X, Y) and the rotation angles of each joint (θ_1, θ_2) can be expressed as in equation (1), where their time derivatives of the individual coordinates are represented as dots.

$$\begin{pmatrix}\dot{X}\\ \dot{Y}\end{pmatrix} = \begin{pmatrix}L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2) & L_2\cos(\theta_1+\theta_2)\\ L_1\sin\theta_1 + L_2\sin(\theta_1+\theta_2) & L_2\sin(\theta_1+\theta_2)\end{pmatrix}\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix} \quad (1)$$

If $\begin{pmatrix}\dot{X}\\ \dot{Y}\end{pmatrix}$ is expressed as $\dot{X}$ and $\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix}$ as $\dot{\theta}$, the rotation angle of each joint can be obtained from the hand position, as shown in equation (2).

$$J^{-1}\dot{X} = \dot{\theta} \quad (2)$$

The matrix J shown in equation (3) is Jacobian, and to find the inverse matrix J^{-1}, we need to find its determinant det(J) and calculate equation (4).

$$J = \begin{pmatrix}L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2) & L_2\cos(\theta_1+\theta_2)\\ L_1\sin\theta_1 + L_2\sin(\theta_1+\theta_2) & L_2\sin(\theta_1+\theta_2)\end{pmatrix} \quad (3)$$

$$J^{-1} = \frac{1}{\det(J)}\begin{pmatrix}L_2\sin(\theta_1+\theta_2) & -L_2\cos(\theta_1+\theta_2)\\ -L_1\sin\theta_1 - L_2\sin(\theta_1+\theta_2) & L_1\cos\theta_1 + L_2\cos(\theta_1+\theta_2)\end{pmatrix} \quad (4)$$

When det(J) = 0, the relationship is as in equation (5).

$$L_1L_2\sin\theta_2 = 0 \quad (5)$$

The values of θ_2 that satisfy this condition are 0° and 180° and called the singular posture. Namely, when L_1 and L_2 form a straight line, IK cannot be used to control the arm. In the case of a normal multi-axis robot, the joints are slightly tilted in the direction of bending to avoid a singular posture.

However, in the case of the flexible arm used here, the initial posture is in a straight line, and the angle of each joint cannot be calculated with the usual IK. For this reason, a spline IK solver [19], in which the motion of the arm follows a non-uniform rational B-spline (NURBS) curve represented by the CG function to define the smooth shape, is used to represent the flexible arm bending angle. This enables the simulation of free bending of the arm in three dimensions by matching the joints on the NURBS curve. We decided to use it for the first time in the simulation of flexible arms.

The NURBS curve has the feature of determining the shape by using multiple control points, as shown in Figure 4. The curve is represented by a polynomial expression that follows the definition of a B-spline basis function, where the point closest to the control point is affected on the curve. Therefore, if the control point is an element, called a controller, for rotating the joints of the robot arm, the arm can be freely bent in three dimensions, unlike the method using IK, where the angle is strictly specified, and singular postures are avoided. The spline IK solver specifies the start and endpoints of a NURBS curve and makes each control point follow the movement of the endpoint. Although the solver is not a feature of conventional robot simulation tools, it is incorporated in several CG tools, so we decided to use it for planning the robot's posture instead of the usual IK.

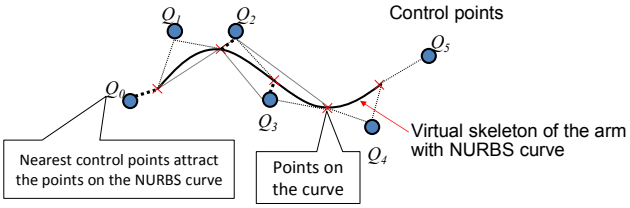

Figure 4. *Sample curve drawn by NURBS for the spline IK method.*

4. Real-Time Simulation of Robots with Flexible Arms

4.1. Building Environmental Mesh Objects from 3D Point-Cloud Data

If the results of 3D measurements can be imported into the VR environment on the fly, the efficiency of preparing environmental models for remote operation support can be improved. Here, a method for generating a mesh model for a VR environment in a few minutes in the field was developed based on 3D measured point-cloud data. Figure 5 shows the point cloud obtained from 3D measurements conducted in a test space enclosed by partitions that were used to verify the operation of the robot in a narrow area. It is necessary to subsample the point cloud to the order of 10^6 points because of the constraints of the development tool on its use in real-time in the VR environment. The figure compares the raw (total) point cloud data consisting of (a) $2x10^7$ points and (b) $1x10^6$ subsampled data points. Shapes common to both figures appear, but there is a gap between the points in the data of (b) such that the wall seems transparent. Therefore, it is necessary to generate a mesh model of the wall to prevent the other side from being seen when the robot is close to it and performs a collision check.

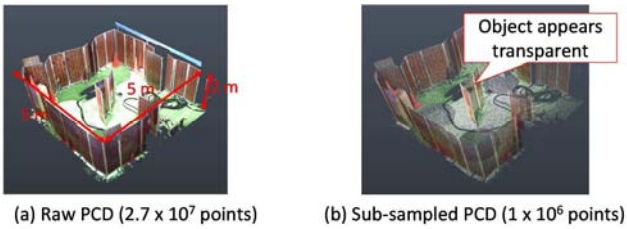

Figure 5. *Comparison between raw and sub-sampled point-cloud data for mesh modeling.*

A method to enlarge the faces of the mesh model is as follows: select three points and connect them with corresponding edges to form a face. Faces with similar normal vectors to those of adjacent faces are consolidated (Figure 6). To generate a mesh model, points of interest are selected from the point-cloud data (S-1). N points around the point of interest are chosen (S-2). For the chosen surrounding points, a weighting factor is given according to its distance from the point of interest (S-3). Then a face of the triangular mesh is set around the point of interest with maximum angle $\theta_{t,\,max}$, and minimum angle $\theta_{t,\,min}$ (S-4). After the faces are generated, the normal vectors of the faces are calculated (S-5), and if the angle θ_n between two normal vectors is less than $\theta_{n,\,max}$, the two normal vectors are considered similar, and the surfaces are merged (S-6). The above process is iterated to the maximum number of point-cloud data (S-7).

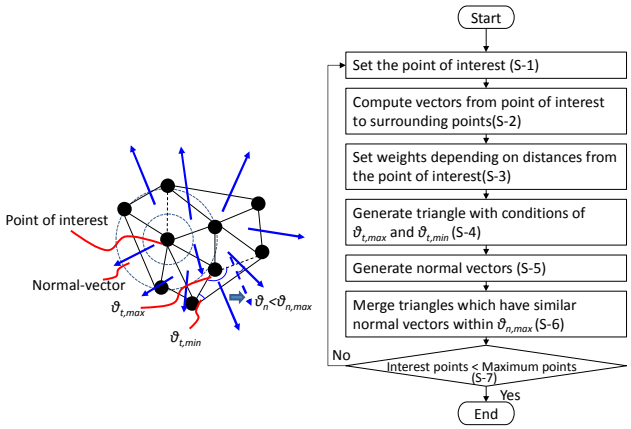

Figure 6. *Flowchart of modeling mesh from point-cloud data.*

4.2. Visualization of Moving Postures for Flexible Arms in a Virtual Reality Environment

A CG model of a robot was created using a commercially available CG tool (Maya, Autodesk Inc. [20]). In Figure 7, the metal plate springs that cover the flexible arms are removed, and the internal structure is illustrated. There are sets of hydraulically driven pistons. Each set consists of four pistons and cylinders and tubes that inject and discharge water. The flexible arm can be bent in three dimensions by adjusting the amount of water injected into the cylinders.

Each arm set can be bent up to 45 degrees relative to each of the x-, y-, and z-axes. Since the arm consists of an upper part and a forearm, it can bend up to 90 degrees relative to each axis. Therefore, by using a simple operation method with the spline IK solver, each piston is set to bend to the average of up to 45 degrees, and the ball joints connect pistons to bend two parts before and after each joint, it sets the constraint to be bent upwards to 22.5 degrees in each piston alone.

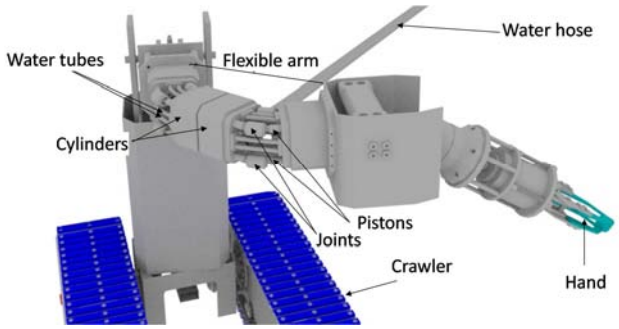

Figure 7. *Simulation of movement for pistons and water tubes in the flexible arm.*

4.3. Collision Detection of Arms Objects with Environmental Meshes

Figure 8 shows the model used to check for collisions between the robot and objects in the area that it will be working in. A real-time physics computation engine (open-source NVIDIA PhysX) is used for checking collisions [21]. Mesh with a geometric shape, called a mesh collider (the green wire frame part of the diagram) is used for detecting collision events. We decided that the mesh collider should be composed of simple geometric shapes to reduce the colliding portions and events because the processing becomes heavy when the mesh of the colliding body precisely matches the shape of part of the robot. Moreover, the robot's mesh collider was displaced 10 mm upward relative to the floor plane because the floor surface generated from the measured 3D point cloud data has an error of about several millimeters, and if the mesh collider remains in continuous contact with the floor, a state of collision persists. This robot VR application can be used for calculating the duration of the operation for the robot as well as recognizing the self-position of the actual robot from various three-dimensional views.

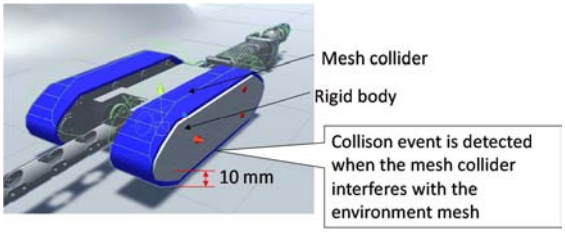

Figure 8. *Geometric shapes for detecting collisions with a mesh collider based on PhysX engine.*

The relationship between the CG model of the robot and the operational parts is shown in Figure 9. The flexible arm and column can move relative to the chassis of the robot as follows: swing 1 moves the column upwards through an angle, turn 1 rotates swing 1, swing 2 moves the flexible arm downwards through an angle, swing 3 moves the hand end effector downwards through an angle, suspension contracts the hand-end effector, and turn 2 rotates the hand end effector. These operational parts of the arm can be bent up to 90 degrees in four directions (bend_up, bend_down, bend_left, and bend_right).

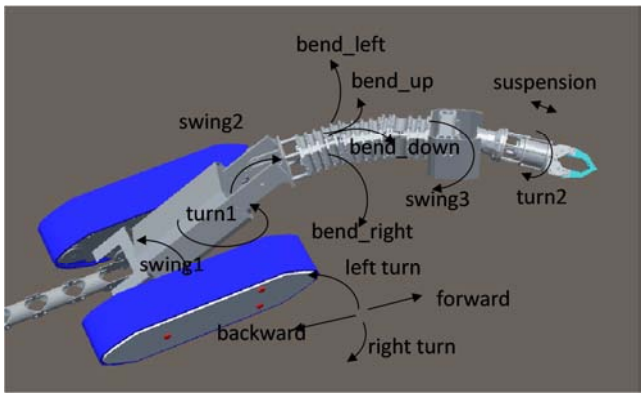

Figure 9. *Parameters for operating the robot with a flexible arm.*

5. Results and Discussion

5.1. Generating a Mesh Model from a Point-Cloud

As mentioned in section 4.1, the maximum number of meshes should be limited to 10^6 for a VR simulation to be carried out in real-time. To confirm this condition is practically achievable, the quality of the generated mesh models was evaluated in comparison with the processing time to generate the model.

An example of a mesh model generated from the point cloud data is shown in Figure 10. Figure 10 (a) shows the registration results of point-cloud data obtained by 3D laser scanning of three locations in a field test. It consists of 2.7×10^7 points. Figures 10 (b) and (c) show mesh models obtained from 10^6 and 10^7 of those points. In these models, the meshes of the wall do not appear transparent as the red cone in Figure 10 (a) when the robot is close to the obstacle and can be visualized as an obstacle. The number of mesh faces in each case is 1.9×10^6 and 1.8×10^7, and it is easy to see that the surfaces match those in the raw data.

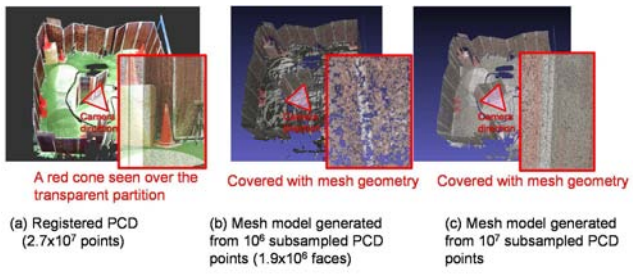

Figure 10. *Comparison of mesh models and corresponding point cloud data.*

Figure 11 shows the time it takes to generate the mesh model from the point cloud data. It took 2 minutes to complete the

processing of 10^6 points that were sub-sampled from the raw point-cloud data. When 10^5 points were obtained by the laser range finder (LRF) mounted on the robot, the model was completed in 20 seconds. When 10^7 points were used, it took 6 hours to make the model. Therefore, the processing time increased in proportion to the square of the number of points. If processing data can be done offline, 10^7 points could be used, and it would be possible to simulate in an environment model made of relatively smooth geometric shapes. However, in the field, it would be desirable to capture the mesh model within a few minutes of making a 3D measurement. Therefore, we consider that 10^6 points of point-cloud data are a practical limit regarding the processing time of the mesh model.

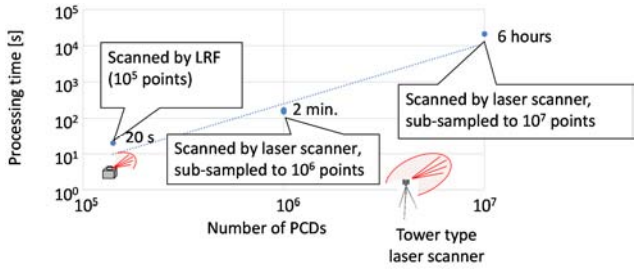

Figure 11. *Relationship between mesh model size from point-cloud data and its generation time.*

The screen update speed of the simulation was verified in the case of using a mesh model generated from large-scale point-cloud data. The verification used a PC equipped with an Intel Xeon ES-2603, 1.6 GHz 2CPU. The VR simulation of the mesh model generated from 10^7 points had a screen update speed of 1-2 frames per second; thus, a mesh model of this size is not suitable for real-time operation. It is said that the screen update speed for real-time operation should be 30 frames per second or more. When the mesh model was generated from 10^6 points, the screen update speed in the VR simulation was 60 frames per second. Consequently, 10^6 point-cloud data was determined to be adequate for the VR simulation.

If the number of points is decreased, the geometric shape of the resulting mesh model will be of inferior quality, as shown in Figure 10 (b). We thought that the quality of the geometric shapes could be improved by adjusting the parameters when the mesh is generated. Figure 12 shows the relationship between the number of meshes generated with K_s points to be searched in the vicinity of the point of interest at the time of calculation and the number of generated faces. The characteristics of a cone mesh model were examined for various values of K_s. The diameter of the bottom of the conical shape was 278 mm and it corresponds to the maximum size of small-bore piping components that would be cut and removed during decommissioning. Therefore, the cone was a reasonable size for this evaluation. K_s is a parameter that determines the complexity of the mesh shape that can be generated in the vicinity of a particular point. The number of generated geometric faces tends to increase as K_s increases, but the growth rate tends to saturate beyond a certain value of K_s. This tendency appears when there are fewer points in the neighborhood, and the resulting shape depends greatly on which points are chosen. When K_s is 20, the cone appears incomplete with numerous fractures. When K_s is 50 and 100, the cone is more complete and has fewer fractures. Moreover, it is not efficient to increase this K_s more than necessary; it takes 18 minutes or more when K_s is 1000 but only 4 minutes when K_s is 100. Thus, K_s should be set from 50 to 100 to generate mesh models of equipment or piping components with many changes in the generated shapes. With the developed method, we confirmed that it connects meshes based on the similarity of normal vectors to mesh planes generated from neighboring point-cloud data to reproduce equipment and piping shapes compared with the previous study [16].

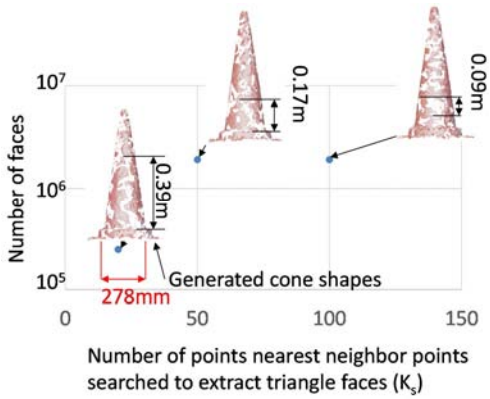

Figure 12. *Quality of generated meshes.*

5.2. Checking the Collision Detection Function of the Robot in the Mesh Model

Figure 13 shows the superimposed movement locus of the center coordinates of the crawler as blue points on the point-cloud data. The white square on the floor shows an area of 80 cm × 80 cm. This VR application can be used to confirm the coordinates 180 seconds after arrival at the endpoint of the center of the crawler, and 20 seconds after the start of the movement. Moreover, it can be used to confirm the posture of the robot with the 3D isometric display. Therefore, the collision detection function enables users to confirm the postures of the robot even if the camera views are occluded by obstacles.

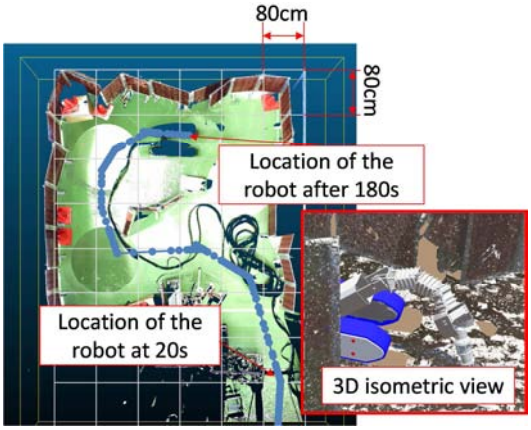

Figure 13. *Trajectory of the robot in the field test.*

Figure 14 shows how the robot hand detects rubble and its

contact situation. The situation at the moment of contacting the rubble was able to be detected from the posture. When a collision event was detected, the amount of processing temporarily increased; consequently, the screen was updated at about 30 frames per second, i.e., about half that of a situation with no collision. Therefore, it was found that 60 frames per second or more is a desirable update speed when no collision occurs. With the developed method, we confirmed that it connects meshes based on the similarity of normal vectors to mesh planes generated from neighboring point-cloud data to reproduce equipment and piping shapes compared with the previous study [16].

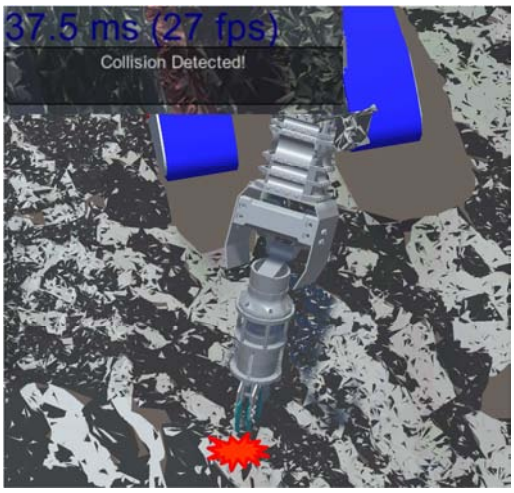

Figure 14. *Example posture of robot with flexible arm avoiding collisions against obstacles.*

5.3. Evaluation of Moving Time of the Robot

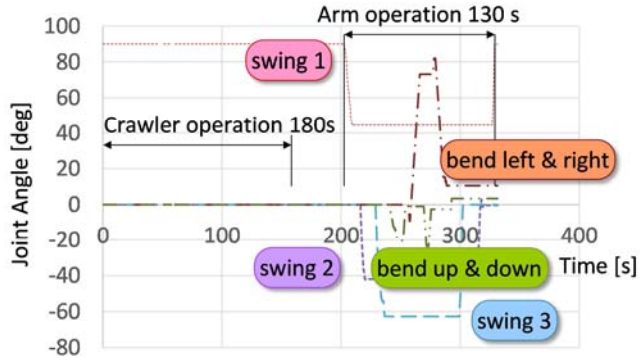

Figure 15. *Recorded time of the movement and the changes in angle of the flexible arm of the robot.*

Figure 15 shows the recorded times of movements and changes in the angle of the flexible arm. By changing the angle of the arm (swing 1 to swing 3, bend_left, up, down, and right) at a rate of 10 degrees per second, the manual operation took 130 seconds to grasp and remove rubble. It took a total of 13 operation steps to contact the rubble from the moment that the rubble was recognized, and the operator completed the movements (while looking at the screen) about three times faster than the actual robot's operation. This means that the planned operation can be checked more quickly with the VR application than with the actual robot hardware in a mockup field test. In addition, the difference between the start and the end of a series of operations can be assigned as the duration of the actual scheduled activity.

The mesh is very coarse when the number of points is small, as shown in Figure 10 (b), because the normal vectors of the triangular mesh formed by the edge between points and the surface are not smoothed or optimized in the VR environment. The simulation in the VR environment used such a coarse mesh model because its purpose was to detect collision events. In contrast, more detailed simulations that incorporate the mechanical stability of the center of gravity of the robot on unstable surfaces, lifting of the object to be removed, and dynamics such as multi-legged walking will require more sophisticated processing such as smoothing of the mesh, repairing splits in the mesh surface, or replacing the point-cloud based mesh with a 3D CAD model. In the future, we will build mesh models that can be used for dynamics simulations. In addition, we will use the calculated duration in this VR environment as input for the schedule optimization system. In this way, we will develop a schedule optimization technology that checks and updates the duration of an activity in the planned schedule corresponding to various operating scenarios.

6. Conclusions

A support system of a robot operated remotely for high dose-rate environments plays a crucial role during decommissioning of NPPs. Before deploying remote-controlled robots to a site, it is necessary to confirm that the operation plans are safe and efficient. In this research, we developed the technology necessary to calculate the working duration by checking that the robot operates without colliding with obstacles, such as equipment and piping components on floors and walls, in a virtual environment made of a mesh model built from 3D point-cloud measurements.

A method that enlarges face surfaces was investigated to generate mesh models efficiently. It generates triangular by selecting three points for each face and consolidating nearest surfaces which have normal vectors in similar directions as one face. The appropriate number of point cloud data and parameters used for generating mesh models were evaluated by examining the quality of the generated mesh model as well as the processing time. As for the point cloud data, a 5 m x 5 m test field enclosed by partitions for testing the behaviors of the robot in a narrow space was measured with a 3D scanner.

As a result of the evaluation, it was decided to generate mesh models from point cloud data consisting of 10^6 points, with which the operator's screen can be refreshed at 60 frames or more per second. The number of points to be searched in the vicinity of a point of interest should be set from 50 to 100 for generating faces with fewer cracks. In this case, the calculation time was less than two minutes.

As for detecting collisions, a CG model of a robot with a flexible arm was made and collisions with obstacles were detected by

simplifying the mesh shapes including that of the robot.

With the developed functions, we confirmed that the movement trajectory of the robot and a three-dimensional isometric display from an arbitrary camera angle can be visualized by using the mesh model as the environment model of the robot. In addition, we confirmed that the moment of contact with rubble can be detected and displayed. The speed at which the user operates while looking at the screen is faster than the actual robot's angular change. By using this VR application instead of an actual hardware robot, work planning can be substantially shortened. The difference between the start and end times of a series of operations conducted in the VR environment can be scheduled as the duration of the actual working activity.

Acknowledgements

I would like to thank Professor Satoshi Tadokoro of Tohoku University for his constructive suggestions.

References

[1] Schmittem, M. (2016). Nuclear decommissioning in Japan: Opportunities for European companies. EU–Japan Centre for Industrial Cooperation.

[2] Grossi, P., Segabinaze, R., Tello, C., and Daniška, V. (2013). Cost estimation for decommissioning of research reactors. 2013 International Nuclear Atlantic Conference. ISBN 978-85-99141-05-2.

[3] Szőke, I., Louka, M., Bryntesen, T., Edvardsen, S. and Bratteli, J. (2015). Comprehensive support for nuclear decommissioning based on 3D simulation and advanced user interface technologies. Journal of Nuclear Science and Technology, 2015, 52, 371–387.

[4] Ohga, Y., Fukuda, M., Shibata, K., Kawakami, T. and Matsuzaki, T. (2005). A system for the calculation of radiation field for maintenance support in nuclear power plants. Radiation Protection Dosimetry. 116, 592-596.

[5] Lee, J., Kim, G., Kim, I., Hyun, D., Jeong, K., Choi, B., and Moon, J. (2016). Establishment of the framework to visualize the space dose rates on the dismantling simulation system based on a digital manufacturing platform. Annals of Nuclear Energy, 95, 161-167.

[6] Kim, I., Choi, B., Hyun, D., Moon, J., Lee, J., Jeong, K. and Kang, S. (2016). A framework for a flexible cutting-process simulation of a nuclear facility decommission. Annals of Nuclear Energy, 2016, 97, 204-207.

[7] Nonaka, Y., Yamamoto, E., Oya, K., Enomoto, A. and Seki, H. (2016). Development of IT-driven power plant engineering work support systems. Hitachi Review, 65, 963-968.

[8] Seki, H., Imamura, M., and Nagase, H. (2020). Evaluating Precise Quantity of Decommissioning Waste by Cutting Virtual 3D Models of Large Equipment. Nuclear Science, 5, 36-43.

[9] Okada, S., Hirano, K., Kobayashi, R., and Kometani, Y. (2020). Development and Application of Robotics for Decommissioning of Fukushima Daiichi Nuclear Power Plant. Hitachi Review, 69, 562–563.

[10] Thuruthel, T., Ansari, Y., Falotico, E., and Laschi, C. (2018). Control strategies for soft robotic manipulators: A survey. Soft robotics, 5, 149-163.

[11] Andaluz, V., Chicaiza, F., Gallardo, C., Quevedo, W., Varela, J., Sánchez, J., and Arteaga, O. (2016). Unity3D-MatLab simulator in real time for robotics applications. International Conference on Augmented Reality, Virtual Reality and Computer Graphics, 9768, 246-263.

[12] Besset, P., and Taylor, C. (2014). Inverse kinematics for a redundant robotic manipulator used for nuclear decommissioning. UKACC International Conference on Control (CONTROL), IEEE, 56-61.

[13] Borboni A., Bussola R., Faglia R., Magnani P., Menegolo A. (2008). Movement optimization of a redundant serial robot for high-quality pipe cutting. J Mech Design, 130 (8): 0823011 1-6.

[14] Pin, F., Love, L., and Jung, D. (2004). Automated kinematic equations generation and constrained motion planning resolution for modular and reconfigurable robots. In Proceedings of the 227th ACS National Meeting.

[15] Shoji, K. (2017). Possibility of applying large-scale point cloud/mixed reality technology in decommissioning of nuclear facilities. Dekomisshoningu Giho, 55, 8-21.

[16] Li, T., Wang, J., Liu, H., and Liu, L. (2017). Efficient mesh denoising via robust normal filtering and alternate vertex updating. Frontiers of Information Technology and Electronic Engineering, 18, 1828-1842.

[17] Unity Technologies. (2020). Optimizing graphics performance, https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html.

[18] Ramanagopal, M., Nguyen, A., and Ny, J. (2017). A motion planning strategy for the active vision-based mapping of ground-level structures. IEEE Transactions on Automation Science and Engineering, 15, 356-368.

[19] Shi, X., Fang, H., and Guo, L. (2016). Multi-objective optimal trajectory planning of manipulators based on quintic NURBS. IEEE International Conference on Mechatronics and Automation, IEEE, 759-765.

[20] Autodesk, Inc., Maya | Computer Animation & Modelling Software | Autodesk, https://www.autodesk.com/products/maya/overview.

[21] NVIDIA, Inc., NVIDIA PhysX Software System, https://www.nvidia.com/en-us/drivers/physx/physx-9-19-0218-driver/.