International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:15, No:4, 2021

Obtaining High-Dimensional Configuration Space for Robotic Systems Operating in a Common Environment

U. Yerlikaya, R. T. Balkan

Abstract**—**In this research, a method is developed to obtain high-dimensional configuration space for path planning problems. In typical cases, the path planning problems are solved directly in the 3-dimensional (D) workspace. However, this method is inefficient in handling the robots with various geometrical and mechanical restrictions. To overcome these difficulties, path planning may be formalized and solved in a new space which is called configuration space. The number of dimensions of the configuration space comes from the degree of freedoms of the system of interest. The method can be applied in two ways. In the first way, the point clouds of all the bodies of the system and interaction of them are used. The second way is performed via using the clearance function of simulation software where the minimum distances between surfaces of bodies are simultaneously measured. A double-turret system is held in the scope of this study. The 4-D configuration space of a double-turret system is obtained in these two ways. As a result, the difference between these two methods is around 1%, depending on the density of the point cloud. The disparity between the two forms steadily decreases as the point cloud density increases. At the end of the study, in order to verify 4-D configuration space obtained, 4-D path planning problem was realized as 2-D + 2-D and a sample path planning is carried out with using A* algorithm. Then, the accuracy of the configuration space is proved using the obtained paths on the simulation model of the double-turret system.

Keywords**—**A* Algorithm, autonomous turrets, high-dimensional C-Space, manifold C-Space, point clouds.

I. Introduction

THE robotic manipulation is an established technology that is widely used in the industry [1]. Besides, the topic of Industry 4.0 has become popular among many companies, research centers, and universities. The transition to industry 4.0, the rise of unmanned and smart factories, smart production, machine-to-machine and advanced manufacturing have greatly increased the need for co-working robots and so that this situation has made robots with a certain degree of independence more attractive [1]-[4]. However, with the development of complex and intelligent operations, complex tasks in some processes cannot be completed effectively by only using a single manipulator. At the same time, it is known that the coordination of two manipulators increases the complexity of the task and improves the efficiency of the operation as well. However, when the two manipulators operate in a common workspace, it is possible to encounter collision between the manipulators and with the obstacles standing around, so the coordinated operation of the two manipulators becomes also a hot topic for researchers [5].

The number of dimensions of the configuration space comes from the degree of freedoms (DOFs) of the system that is interested in. In the literature, there are many studies about path planning in configuration space. They commonly used a sample 2 or 3-dimensional configuration space to verify their path planning approach [5]-[8]. However, unlike those studies, this study aims to construct ways to obtain high dimensional configuration spaces for real systems like robotic manipulators, turrets, etc. [9], [10].

A multiple cradle launcher named Jobaria in Fig. 1 can be an example for this study. 3 dependent turrets that have 2 DOFs in each one means that the whole system has 6 DOFs in total [9] with some constraints. It is impossible to rotate launchers independently because of the space restrictions. Likewise, an infantry fighting vehicle with two turrets which have 2 DOFs each can be considered as an example for this study as well [10]. Also, the configurations in Fig. 2 can be similar examples where two robotic manipulators or dual-arm robots are working together in the same environment which creates collision problems [5]. The motion of two dual-arm robots, each of which has 2 degrees of freedom in Fig. 2 (a), can be defined as 4-dimensional C-Space. Each manipulator in Fig. 2 (b) has 6-DOFs and satisfies the Pieper criterion, that is, the three consecutive axes of the robot intersect at one point, and therefore it is possible to model the last 3 DOFs of the manipulator as a sphere that covers all the motion of last 3 links. So, the motions of two manipulators can be expressed as a 6-dimensional configuration space which helps to conduct the path planning without colliding. To handle these difficulties and obtain high-dimensional configuration space, there are on-line and off-line methods to conduct path planning. These methods can provide advantages and disadvantages to the systems they are used. Because the online methods recognize objects in the workspace simultaneously, it constantly updates the configuration space.

Ümit Yerlikaya is with both Mechanical Engineering, Middle East Technical University and FNSS Defense Systems Inc., Ankara, Turkey (e-mail: umit.yerlikaya@metu.edu.tr).
R. Tuna Balkan is with Mechanical Engineering, Middle East Technical University, Ankara, Turkey (e-mail: balkan@metu.edu.tr).

Fig. 1 Jobaria multiple cradle launcher [9]

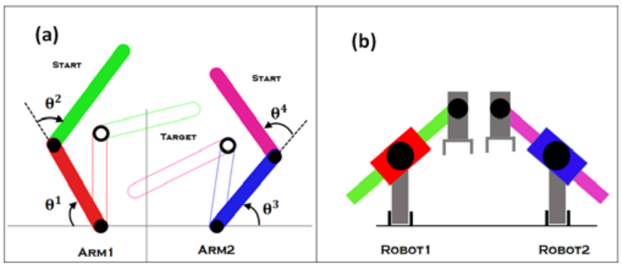


Fig. 2 (a) Collaborative dual-arm robots, (b) model of two manipulators

The continuous updating of the configuration space provides advantages for this method, while the electronic devices used for updating can make the system more expensive. As long as the workspaces of the robots which are working together do not change, off-line methods are cheaper.

II. Configuration Space

The motion planning problems can be directly formalized and solved in the 3-D workspace, generally by the potential field algorithms [11]. However, it cannot be easily handled the robots with various geometrical and mechanical restrictions by these workspace solutions. To solve these difficulties, path planning may be formalized and solved in a new space which is called configuration space [8], [12]-[14]. The complex geometric shaped robot in a 3-D workspace is mapped to a point robot, therefore the motion of the robot corresponds to a continuous curve in the high-dimensional configuration space as given in Fig. 3.

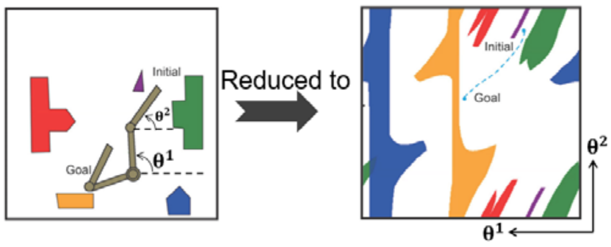


Fig. 3 Workspace and C-Space representations

We can solve the motion planning problem in two steps [8]. For robots or manipulators to be driven to desired locations without colliding, firstly it is required to obtain the configuration space and perform an optimization on the found C-Space. The configuration space is created by combining all the possible motions that cooperative systems can do. Besides, the mapping of the obstacles in configuration space may be expanded by a specified safety factor. The safety factor is determined according to the safe distance that must be between the co-operators. It can be done by "Minkowski" sum method [8] for the 2-D cases.

In this study, we will be interested in the configuration space of the systems which does not change frequently, in other words, focused on the off-line systems. If the workspace is changed, then the corresponding configuration space should be updated. During this study, we do not focus on path planning itself. Besides, there will be some path planning solutions to verify the obtained n-dimensional configuration space. There are two ways to obtain n-dimensional configuration spaces which will be input for n-dimensional path planning. One of them is to obtain n-dimensional configuration space for a system by using point clouds and the detection of the intersection of these clouds. The other is to handle 3-D models and DOFs by using simulation softwares. During this study, the configuration space will be obtained by using these two ways.

A. First Way: Obtain by Point Clouds

To obtain configuration space by using point clouds, firstly all the 3-D shapes are converted into point clouds. For this purpose, any finite element meshing software can be used. After creating a mesh of 3-D shapes, the point cloud representation of the double-turret system can be seen as in Fig. 4.

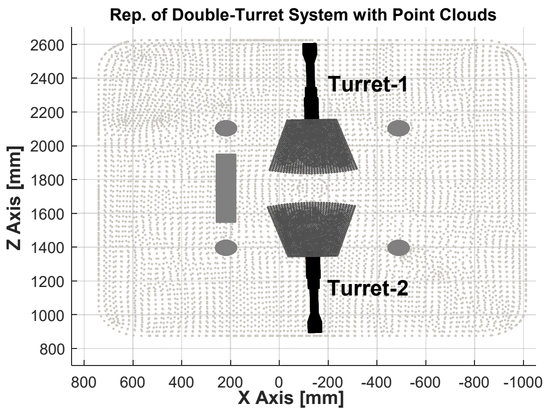


Fig. 4 Point cloud representation of double-turret system

The point clouds can be classified and represented as moving and fixed point clouds as seen in (1) and (2):

$$MPC = \{MPC_1 \quad MPC_2 \quad MPC_3 \quad .....\} \tag{1}$$

$$FPC = \{FPC_1 \quad FPC_2 \quad FPC_3 \quad .....\} \tag{2}$$

Each element of MPC and FPC can be represented as in (3):

$$MPC_i = [X_{in}\, Y_{in}\, Z_{in}] \quad FPC_k = [X_{km}\, Y_{km}\, Z_{km}] \tag{3}$$

where

$$X_{in}=\begin{bmatrix}x_{i1}\\x_{i2}\\ \cdot\\ \cdot\\x_{in}\end{bmatrix}\quad Y_{in}=\begin{bmatrix}y_{i1}\\y_{i2}\\ \cdot\\ \cdot\\y_{in}\end{bmatrix}\quad Z_{in}=\begin{bmatrix}z_{i1}\\z_{i2}\\ \cdot\\ \cdot\\z_{in}\end{bmatrix}$$
$$X_{km}=\begin{bmatrix}x_{k1}\\x_{k2}\\ \cdot\\ \cdot\\x_{km}\end{bmatrix}\quad Y_{km}=\begin{bmatrix}y_{k1}\\y_{k2}\\ \cdot\\ \cdot\\y_{km}\end{bmatrix}\quad Z_{km}=\begin{bmatrix}z_{k1}\\z_{k2}\\ \cdot\\ \cdot\\z_{km}\end{bmatrix}\qquad(4)$$

For a given point cloud MPC_1 $(n\times 3)$ in Cartesian coordinates, the translational and rotational operations can be represented as:

$$(MPC_1)_{translate}=MPC_1+I_u\times T,\qquad(5)$$

where I_u is unity column array with a length of n, and translation vector $T=[t_x\quad t_y\quad t_z]$. First, the rotation vectors are defined in three-axes as in (6):

$$RA_x=[0\quad r_y\quad r_z],\qquad RA_y=[r_x\quad 0\quad r_z],$$
$$RA_z=[r_x\quad r_y\quad 0]\qquad(6)$$

Rotations are defined by 3×3 transformation matrices. Rotation about X-axis by an angle of α, rotation about Y axis by an angle of β, and rotation about Z by an angle of γ are defined, respectively, as

$$R_x(\alpha)=\begin{bmatrix}1&0&0\\0&cos\alpha&-sin\alpha\\0&sin\alpha&cos\alpha\end{bmatrix}$$
$$R_y(\beta)=\begin{bmatrix}cos\beta&0&sin\beta\\0&1&0\\-sin\beta&0&cos\beta\end{bmatrix}$$
$$R_z(\gamma)=\begin{bmatrix}cos\gamma&-sin\gamma&0\\sin\gamma&cos\gamma&0\\0&0&1\end{bmatrix}\qquad(7)$$

Finally, rotated point clouds can be obtained as in (8):

$$(MPC_1)_x=[MPC_1-I_u\times RA_x]\times R_x(\alpha)+I_u\times RA_x$$
$$(MPC_1)_y=[MPC_1-I_u\times RA_y]\times R_y(\beta)+I_u\times RA_y$$
$$(MPC_1)_z=[MPC_1-I_u\times RA_z]\times R_z(\gamma)+I_u\times RA_z\qquad(8)$$

To find the distances between any point in MPC_i and any point in FPC_j, the matrix operations given below are used.

$$X=X_{in}-X_{km}^T\quad Y=Y_{in}-Y_{km}^T\quad Z=Z_{in}-Z_{km}^T\qquad(9)$$

X, Y and Z matrices are as follows.

$$X=\begin{bmatrix}x_{i1}-x_{k1}&x_{i1}-x_{k2}&\cdot&\cdot&x_{i1}-x_{km}\\x_{i2}-x_{k1}&x_{i2}-x_{k2}&\cdot&\cdot&x_{i2}-x_{km}\\ \cdot&\cdot&\cdot&\cdot&\cdot\\ \cdot&\cdot&\cdot&\cdot&\cdot\\x_{in}-x_{k1}&x_{in}-x_{k2}&\cdot&\cdot&x_{in}-x_{km}\end{bmatrix}$$

$$Y=\begin{bmatrix}y_{i1}-y_{k1}&y_{i1}-y_{k2}&\cdot&\cdot&y_{i1}-y_{km}\\y_{i2}-y_{k1}&y_{i2}-y_{k2}&\cdot&\cdot&y_{i2}-y_{km}\\ \cdot&\cdot&\cdot&\cdot&\cdot\\ \cdot&\cdot&\cdot&\cdot&\cdot\\y_{in}-y_{k1}&y_{in}-y_{k2}&\cdot&\cdot&y_{in}-y_{km}\end{bmatrix}$$

$$Z=\begin{bmatrix}z_{i1}-z_{k1}&z_{i1}-z_{k2}&\cdot&\cdot&z_{i1}-z_{km}\\z_{i2}-z_{k1}&z_{i2}-z_{k2}&\cdot&\cdot&z_{i2}-z_{km}\\ \cdot&\cdot&\cdot&\cdot&\cdot\\ \cdot&\cdot&\cdot&\cdot&\cdot\\z_{in}-z_{k1}&z_{in}-z_{k2}&\cdot&\cdot&z_{in}-z_{km}\end{bmatrix}\qquad(10)$$

Finally, distance matrix between two point clouds is found as in (11):

$$D=\sqrt{X^2+Y^2+Z^2}\qquad(11)$$

The condition to check collision between point clouds is given in (12) where δ is the safe distance defined in the definition of the collision. The value of δ may not be less than maximum mesh distance.

$$Collision=\begin{cases}1, if\ \ min(D)\le\delta\\0, if\ \ min(D)>\delta\end{cases}\qquad(12)$$

As mentioned before, the 3-D point clouds are created from 3-D shapes. It is assumed that the length of any mesh in 3-D shape is approximately equal and named as dm. The worst case of placing two point clouds and x-y and x-z views of the worst-case are given in Fig. 5 which will help to find the minimum required safe distance, δ.

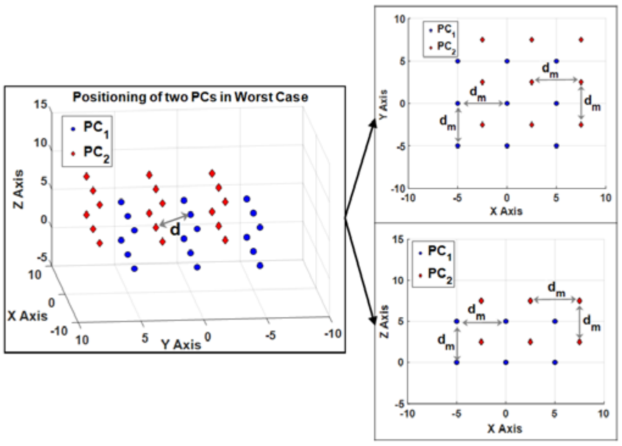


Fig. 5 Positioning of two PCs in worst case

According to Fig. 5, the safe distance should satisfy the criterion in (13) which is necessary and sufficient.

$$\delta\ge d=\frac{\sqrt{3}}{2}\ d_m\qquad(12)$$

The overall algorithm for obtaining the high-dimensional configuration space can be summarized as in Algorithm-1. There is given the information about a 3-D environment such as moving point clouds (MPC), fixed-point clouds (FPC), safe distance (δ), the number of variables (n), initial values of n variables (θ_{in}), final values of n variables (θ_{end}) and step angles of n variables ($\delta\theta$). There is n number of nested loops in the algorithm where each of them belongs to a variable, in order to be an example, during this study the number of variables (n) is kept as 4. As a result of the algorithm, n-dimensional configuration space which is named ConfMap is obtained. Lines between 1-4 and 13-16 vary according to number of variables. The location of MPC is updated according to instantaneous variable values in Line 5. In Line 6-7, it is checked whether there is collision between MPC and FPC, and in Line 8-9 between MPC and MPC by avoiding matching of same point clouds according to the safe distance (δ). Also, the Algorithm-1 calls Algorithm-2 named CheckCollision which is used to detect any collision between point cloud sets to fill configuration space map for corresponding variables. The Algorithm-2 expects the MPC, FPC and δ as inputs and also needs three different functions which are named CheckLap, RemovePoints and FindDistance. In Line 3, it is checked whether there is an intersection between bounding volumes (BV) of two point clouds.

Algorithm 1: ObtainConfSpace

```
     In: $MPC, FPC, \theta_{in}, \delta\theta, \theta_{end}, \delta$
     Out: $CSpace$
1    for $\theta^1 = \theta^1_{in} : \delta\theta^1 : \theta^1_{end}\ (s_1++)$
2      for $\theta^2 = \theta^2_{in} : \delta\theta^2 : \theta^2_{end}\ (s_2++)$
3        for $\theta^3 = \theta^3_{in} : \delta\theta^3 : \theta^3_{end}\ (s_3++)$
4          for $\theta^4 = \theta^4_{in} : \delta\theta^4 : \theta^4_{end}\ (s_4++)$
5            $update\ \boldsymbol{MPC}\ using\ current\ \theta^1, \theta^2, \theta^3\ and\ \theta^4$
6            if $\boldsymbol{CheckCollision}\ (MPC, FPC, \delta) = 1\ (true)$
7              $CSpace\ (s_1, s_2, s_3, s_4) = 1$
8            elseif $\boldsymbol{CheckCollision}\ (MPC, MPC, \delta) = 1\ (true)$
               $(matching\ same\ p.clouds\ is\ avoided)$
9              $CSpace\ (s_1, s_2, s_3, s_4) = 1$
10           else
11             $CSpace\ (s_1, s_2, s_3, s_4) = 0$
12           end if
13         end for
14       end for
15     end for
16   end for
17   return $CSpace$
```

Algorithm 2: CheckCollision

```
     In: $MPC, FPC, \delta$
     Out: $Collision$
1    for i ← number of P.clouds in MPC (i ++)
2      for k ← number of P.clouds in FPC (k ++)
3        $[Lap, IB] = \boldsymbol{CheckLap}\ (MPC_i, FPC_k)$
4        if $Lap = 1\ (true)\ \rightarrow P.clouds\ overlapped$
5          $MPC'_i = \boldsymbol{RemovePoints}\ (MPC_i, IB)$
6          $FPC'_k = \boldsymbol{RemovePoints}\ (FPC_k, IB)$
7          $d = \boldsymbol{FindDistance}\ (MPC'_i, FPC'_k)$
8          if $\min(d) \le \delta$
9            $Collision(i,k) = 1$
10           return $max(Collision)$
11         else
12           $Collision(i,k) = 0$
13         end if
14       else
15         $Collision(i,k) = 0$
16       end if
17     end for
18   end for
19   return $max(Collision)$
```

If there is no intersection, then it can be said that there is no collision between the two point clouds. In the case of an intersection, the point clouds are updated by getting rid of the points outside the intersection box and a distance matrix is obtained between them to see if the minimum distance is smaller than the safe distance (δ).

To effectively check collision between point clouds, it is advisable to approximate objects with bounding volumes (BV) [18], [19]. Various bounding volume types which are widely used in the literature are presented in Fig. 6.

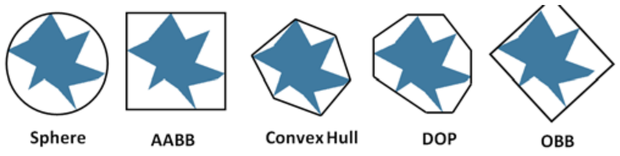


Fig. 6 The types of bounding volumes

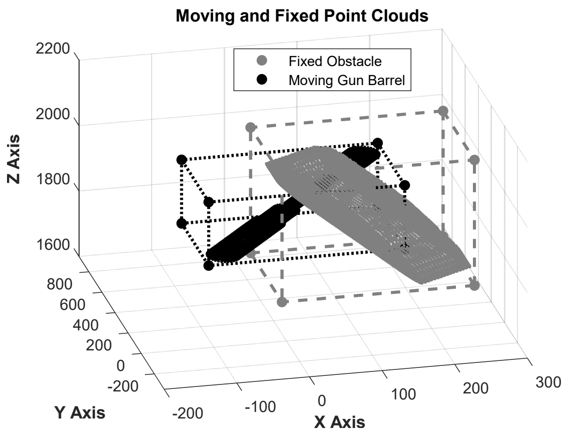


Fig. 7 Two point clouds and their AABBs in 3-D Space

The selection of a bounding volume type depends on the usage. To easily select correct bounding volume, the type of objects should be known beforehand. If no information is available for the object size or shape, the more general shape is always better. However, the more general bounding volume adds extra complexity and hence heavier computationally [15]-[21]. To overcome this difficulty and uncertainty, the sphere or axis-aligned bounding boxes (AABB) can be preferred. In this study, AABB is used for that reason. The following function checks whether the AABBs of two point clouds (PC1 and PC2) are intersected. If there exists an intersection between AABBs, the function gives information about the intersection box (IB), otherwise the overall algorithm returns to no-collision-detected state. Two point clouds and their AABBs are given in Fig. 7. As seen in this figure, there is an intersection between these two

AABBs. Also, the intersection box (IB) is given in Fig. 8.

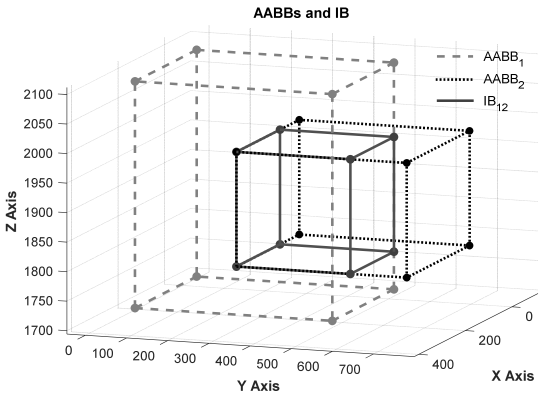


Fig. 8 AABBs of two point clouds and their intersection box (IB)

The following Function-1 named CheckLap allows to see if the AABBs of the two point cloud intersect, and also to obtain intersection box (IB) information in case of intersection.

Function 1: CheckLap

```
    In: $PC_1, PC_2$
    Out: $[Lap, IB]$
1   for $i = [1,2,3]$ (x: 1, y: 2, z: 3)
2       if $\min\{PC_1(i^{th}\ column)\} \geq \min\{PC_2(i^{th}\ column)\}$
3           $min^i = \min\{PC_1(i^{th}\ column)\}$
4       else
5           $min^i = \min\{PC_2(i^{th}\ column)\}$
6       end if
7       if $\max\{PC_1(i^{th}\ column)\} \leq \max\{PC_2(i^{th}\ column)\}$
8           $max^i = \max\{PC_1(i^{th}\ column)\}$
9       else
10          $max^i = \max\{PC_2(i^{th}\ column)\}$
11      end if
12      $IB(i,1) = min^i, \quad IB(i,2) = max^i$
13      if $min^i \leq max^i$
14          $Lap^i = 1$
15      else
16          $Lap^i = 0$
17      end if
18  end for
19  $Lap = Lap^1 \cdot Lap^2 \cdot Lap^3$
20  return $[Lap, IB]$
```

After determining the intersection box (IB) and providing point clouds and information about IB to the Function-2 named RemovePoints, now it is time to get rid of excess points that do not belong to IB. By using this function, the point clouds are modified and the points outside the IB are removed as seen in Fig. 9.

Firstly, the number of points is decreased by deleting the points outside the intersection box (IB) and then the modified MPC and FPC are created which only belong to the IB. Now, in order to decide whether there is a collision between point clouds, the distances between any combination of the points in MPC and FPC are measured. The following Function-3 named FindDistance measures g*h distances and gives g×h sized matrix where the PCs have g and h number of points, respectively.

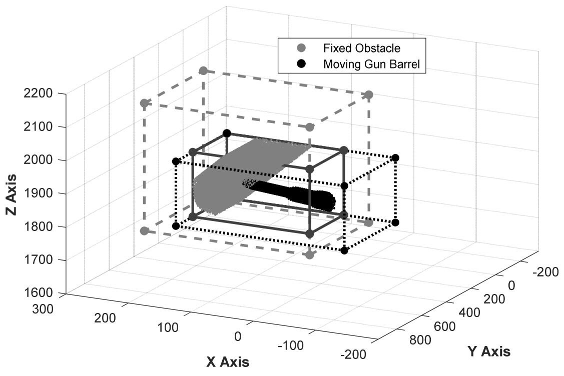


Fig. 9 Modified point clouds after removing points outside intersection box (IB)

Function 2: RemovePoints

```
    In: $PC, IB$
    Out: $PC'$
1   if $\{x\ of\ \forall\ points\ \in\ PC > IB(1,2)\ \textbf{OR} < IB(1,1)\}$
        $\textbf{OR}\ \{y\ of\ \forall\ points\ \in\ PC > IB(2,2)\ \textbf{OR} < IB(2,1)\}$
        $\textbf{OR}\ \{z\ of\ \forall\ points\ \in\ PC > IB(3,2)\ \textbf{OR} < IB(3,1)\}$
2           $\rightarrow remove\ \forall\ points\ of\ PC\ \notin\ IB$
3           $PC' = \forall\ points\ of\ PC\ \in\ IB$
4   else
5       $PC' = PC$
6   end if
7   return $PC'$
```

Function 3: FindDistance

```
    In: $PC_1, PC_2$
    Out: $d$
1   $x_1 = PC_1(1^{th}\ column) \rightarrow (m \times 1)$
2   $y_1 = PC_1(2^{nd}\ column) \rightarrow (m \times 1)$
3   $z_1 = PC_1(3^{rd}\ column) \rightarrow (m \times 1)$
4   $x_2 = PC_2(1^{th}\ column) \rightarrow (n \times 1)$
5   $y_2 = PC_2(2^{nd}\ column) \rightarrow (n \times 1)$
6   $z_2 = PC_2(3^{rd}\ column) \rightarrow (n \times 1)$
7   $X = x_1 - x_2^T \rightarrow (m \times n)$
8   $Y = y_1 - y_2^T \rightarrow (m \times n)$
9   $Z = z_1 - z_2^T \rightarrow (m \times n)$
10  $d = \sqrt{X^2 + Y^2 + Z^2} \rightarrow (m \times n)$
11  return $d$
```

After obtaining the distance matrix, now it is easy to decide if there is a collision at that moment. If we sort and draw the elements of the distance matrix, the distances between points can be obtained as in Fig. 10. If there is any distance lower than safe distance (δ), then we can decide that there is a collision for corresponding variables.

B. Second Way: Obtain by a Simulation Software

In the first way, obtaining the configuration space by detection of intersection of point clouds is explained in detail. In this way, the only collision detection algorithm is changed with the clearance function of the simulation software.

Clearance function measures the minimum distance between the surfaces of bodies and only focuses the surfaces of bodies so that the bodies are assumed to be hollow. During any contact between the bodies, the clearance returns to zero which shows that there is a collision. We can summarize the steps of this way as shown in Fig. 11.

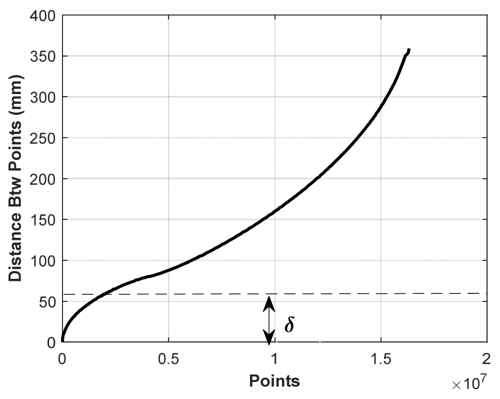


Fig. 10 Sorting and drawing distance matrix to check collision

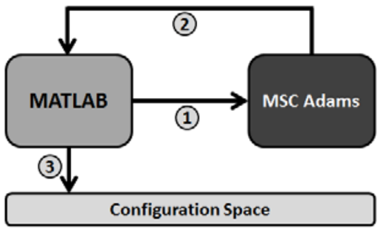
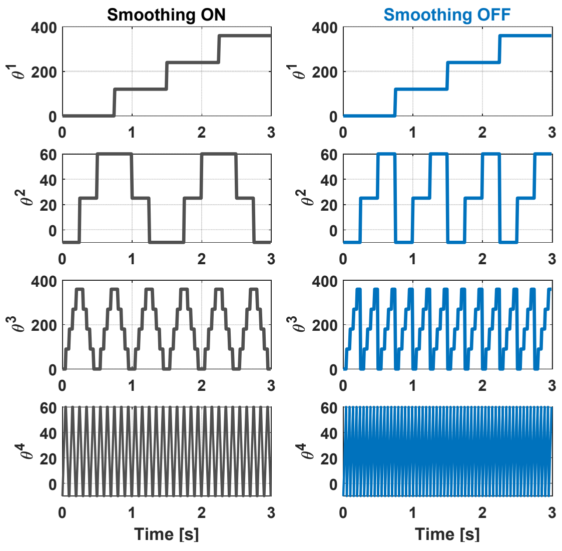


Fig. 11 The sequence of the second way

1. Create position profiles for variables by converting the nested for-loops to time-domain. As an example, the position profiles for four variables in time-domain are illustrated as in Fig. 12.

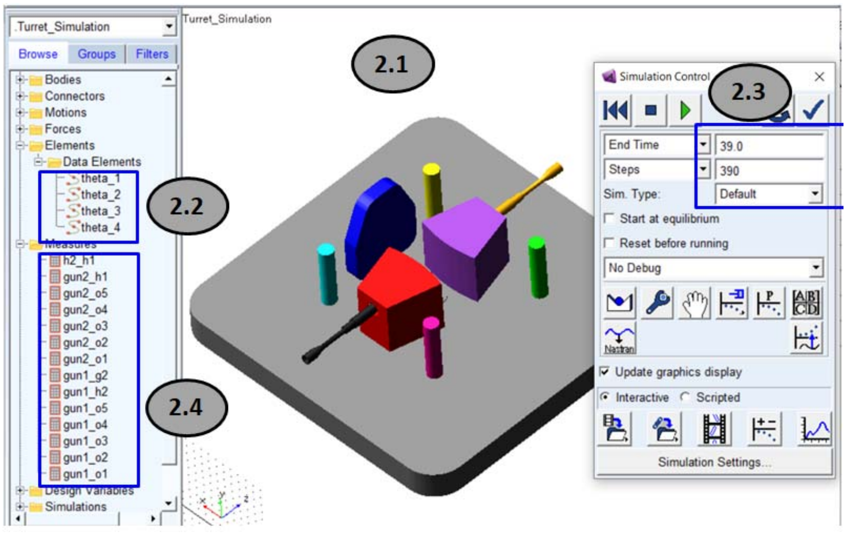


(a) smoothing on (b) smoothing off

Fig. 12 Converting nested for-loops into time-domain position profiles for variables

The position profiles are directly mapped from 4 nested for-loops with sharp edges as shown in Fig 12 (b). The hard transition from the maximum to minimum value causes some problems during solving kinematic equations in simulation software. Therefore, by doing a smoothing operation the position profiles are updated as shown in Fig. 12 (a) and so that the possible errors in simulation are prevented.

2. Create the exact simulation model of the system in MSC Adams. For instance, the simulation model of the 4-DOF double-turret system is given in Fig. 13.



Bodies
Connectors
Motions
Forces
Elements
Data Elements
theta_1
theta_2
theta_3
theta_4
Measures
h2_h1
gun2_h1
gun2_o5
gun2_o4
gun2_o3
gun2_o2
gun2_o1
gun1_g2
gun1_h2
gun1_o5
gun1_o4
gun1_o3
gun1_o2
gun1_o1
Design Variables
Simulations

End Time 39.0
Steps 390
Sim. Type: Default
Start at equilibrium
Reset before running
No Debug
☑ Update graphics display
● Interactive ○ Scripted

Fig. 13 The simulation model of a sample system

All joints, motions and clearances between stationary (obst-1 to 5) and moving bodies (hull and gun barrel) have to be defined. The position profiles in time-domain which are explained in the first step have to be imported and assigned to motions on the joints (θ_1, θ_2). According to steps and step time of position profiles, simulation is conducted.

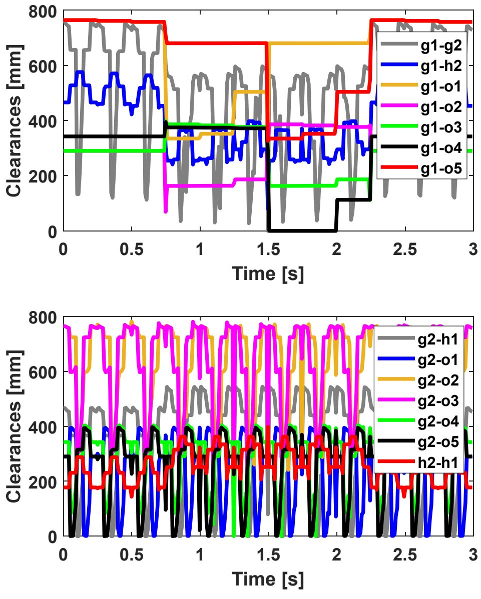


Fig. 14 The change of minimum clearances as a result of sample run

3. After completing the analysis, the change of clearances again in time-domain are obtained as in Table I and sent to other simulation software (MATLAB) in order to obtain configuration space by processing with position profiles inputs.

TABLE I
RESULTS OF SIMULATION

t [s]	θ^1[°]	θ^2[°]	θ^3[°]	θ^4[°]	Collision(δ)
0	0	-10	0	0	0
t_s	0	-10	0	20	0
$2t_s$	0	-10	0	40	0
….	….	….	….	….	….
t_{end}	358	0	358	0	0

III. Case Study

To compare 2 ways of obtaining configuration space, a double-turret system is considered as an example. The simulation and point cloud models of the double-turret system are given in Figs. 15 and 4, respectively.

The turret has two DOFs, one for traverse and the other for elevation. In this example, we have 4 DOFs because of two turrets in the same environment. Also, there are 5 stationary obstacles around these two turrets. Finally, the configuration space up to 4 dimensions can be obtained. Point numbers in the point clouds of all the elements used in the double-turret system are as in Table II. Within scope of this study, the capital or lowercase letters g, o and h represent guns, obstacles and hulls respectively. Also, the ground was not included in the calculations because it is known that no part interacts with the ground at maximum and minimum operating limits for faster results.

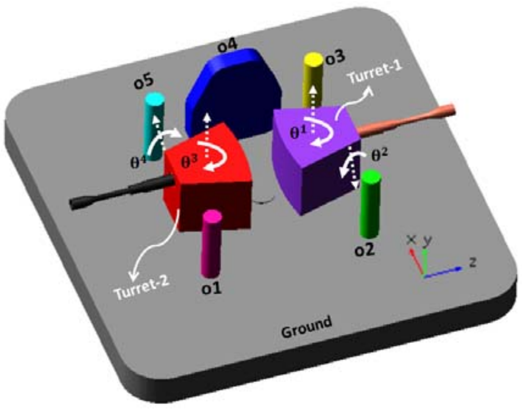


Fig. 15 The Simulation model of double-turret system with stationary obstacles

TABLE II
NUMBER OF POINTS

Parts	Number of Points (for each)
O1, O2, O3, O5	4065
O4	18831
H1 and H2	5614
G1 and G2	3624

The 4-D configuration space of the example is obtained. In this 4-D C-Space, the elevation axes (θ^2, θ^4) start from - 10°, end at 60° with a step size of 2°. The traverse axes (θ^1, θ^3) start from 0°, end at 358° with a step size of 2°. In the method of path planning on the C-Space, the grid number of C-Space has great effect on the algorithm itself. If the grid is too great, the precision of planning will decrease. If the grid is small, the calculation payload will increase. A reasonable grid decomposition should be based on some optimum criterion [25]. The 4-D C-Space can be represented by a certain number of 3-D configuration spaces. For this example, it can be obtained with 36 different 3-D C-Spaces for each value of elevation axis of turret-2 (θ^4) in the example of double-turret system. The 3-D representations of this 4-D C-Space are given for eight different θ^4 angles in Figs. 16 and 17. As can be seen from the figures, as the angle of θ^4 increases, the volume of the disabled area in the C-Space decreases.

The obtained 3-D configuration spaces of two ways (by point cloud, by MSC software) are almost the same. The difference between them is about 1% which comes from the method difference. In the first way, the clearances between bodies are measured from point to point. However, in the second way, they are measured from the surface to surface. In this example, the distance between points in the point clouds is about 5 mm which creates a 1% difference than the mentioned and also the safe distance is 5 mm. The difference between these two ways depends on the distance between points. Difference between methods decreases when the distance between points decreases.

Also, the first way is only created to obtain configuration space, however in the second one, we only use the power of clearance function of the MSC Adams simulation software in order to create an alternative for obtaining configuration space.

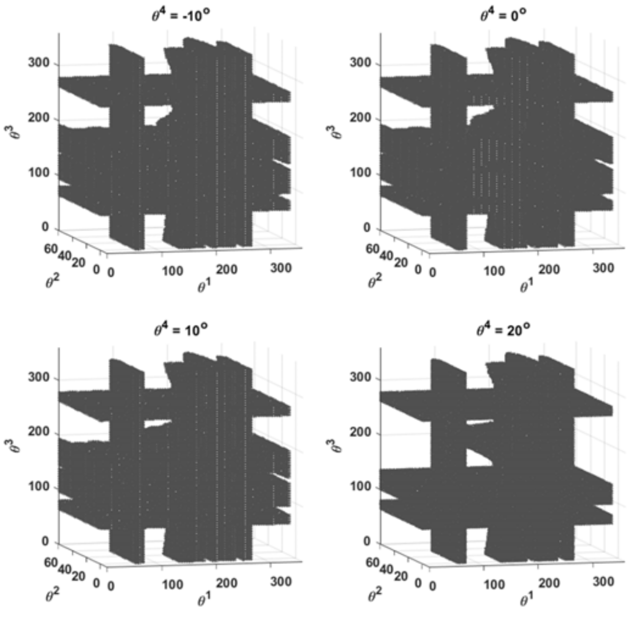


Fig. 16 3-D C-Space of double-turret system (for θ^4 equals -10°, 0°, 10° and 20°)

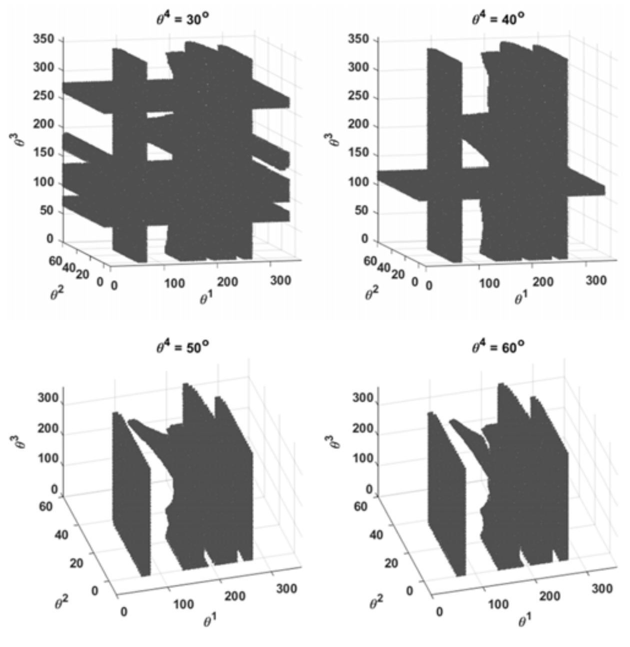


Fig. 17 3-D C-Space of double-turret system (for θ^4 equals 30°, 40°, 50° and 60°)

The computation times for obtaining one of the 3-D configuration space in Figs. 16 and 17 with first and second way are 298.5 and 2414.5 minutes, respectively. One of the 3-D C-Space has 180×180×36 = 1166400 configurations. Since 4-D C-Space consists of 36 3-D C-Spaces, it contains approximately 42 million different configurations. It took approximately 179 hours to calculate 4-D C-Space using the first method. Since this simulation of 42 million will take about a month with the second method, the performance comparison of the two methods was carried out over the time it took to obtain 3-dimensional C-Spaces. For that reason, it can be said that the first way is 8.1 times more efficient. Since C-Spaces do not change frequently in off-line systems, calculation times are quite reasonable.

IV. Verification of the Results

After obtaining the configuration space, the rest is related to path planning algorithms which can be A* [7,22], many variants of rapidly exploring random tree (RRT) [23], probabilistic roadmap (PRM) [24] and so forth. In order to verify obtained 4-D C-Space, 4-D path planning problem was realized as 2-D + 2-D by choosing two axis sets from four existing axes. Whichever two axes from four existing axes will be driven first, motion planning of these two axes is made by taking the 2-D section of the 4-D C-Space corresponding to the starting positions of the other two axes. In order to drive the second axes pair, motion planning is made by taking the 2-D section of the 4-D C-Space corresponding to the target position of the first two axes. For this example, first 1st and 2nd, then 3rd and 4th axes are driven simultaneously.

In order to plan a sample path, the starting and target positions are selected as in (14) and (15), respectively.

$$\theta_s = [\,\theta_s^1, \theta_s^2, \theta_s^3, \theta_s^4] = [40°, -10°, 160°, 16°] \tag{14}$$

$$\theta_t = [\,\theta_t^1, \theta_t^2, \theta_t^3, \theta_t^4] = [264°, 0°, 10°, -10°] \tag{15}$$

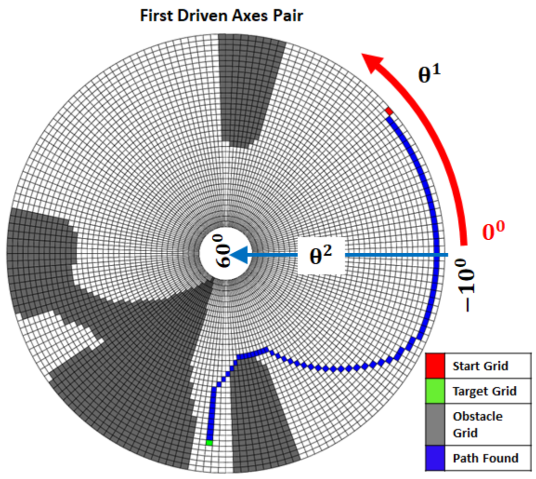


Fig. 18 2-D C-Space of double-turret system for first driven axis pair and path planning result

The configuration space of first driven axes pair and second driven axes pair are given in Figs. 18 and 19. The configuration space is shown in a circular instead of rectangular since the second and fourth axes can be rotated full. Each of the circular C-Spaces has 180×36 grids (axes are divided into 2° steps). A sample path is planned with the help of A * path planning algorithm using two circular C-Spaces as shown in Figs. 18 and 19.

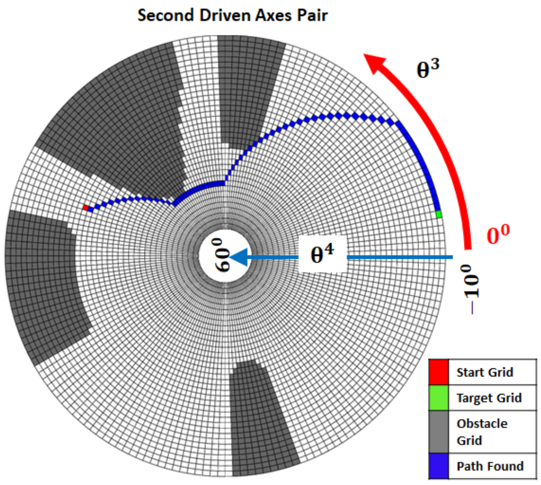


Fig. 19 2-D C-Space of double-turret system for second driven axis pair and path planning result

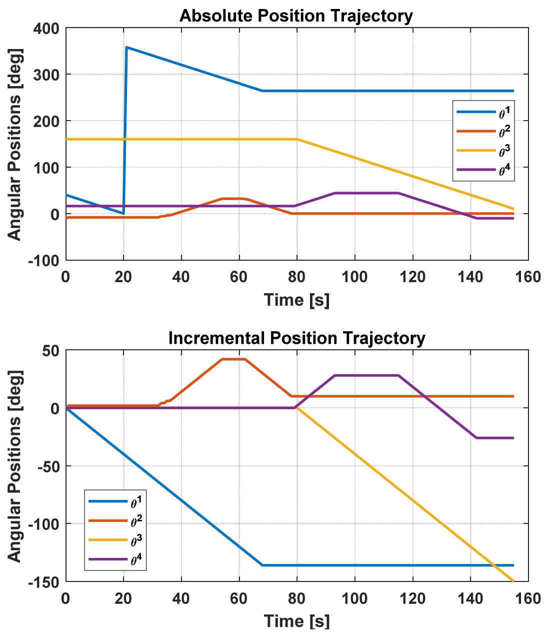


Fig. 20 The change of position profiles concerning path planning result space

The shortest path is calculated so that the turrets can rotate to the goal position. The turrets will reach the goal position without any collision if the system performs simultaneous traverse and elevation rotations as shown in Figs. 18 and 19. The traverse and elevation motion profiles should be created according to angular speed, acceleration or time requirements. If the angular velocities of the axes are set to 2 °/s, the time required to pass one grid is approximately 1 second. In this case, the system can reach the target position from the specified start position in 156 seconds. Thus, the traverse and elevation motion profiles are created as shown in Fig. 20. In order to drive double-turret system, the absolute changes of positions are converted into incremental position changes by resetting starting positions as zero.

Once the motion profiles are obtained, the rotations of the turret system were simulated on simulation model. As a result of this simulation, the changes of minimum clearances between bodies are obtained as shown in Fig. 21.

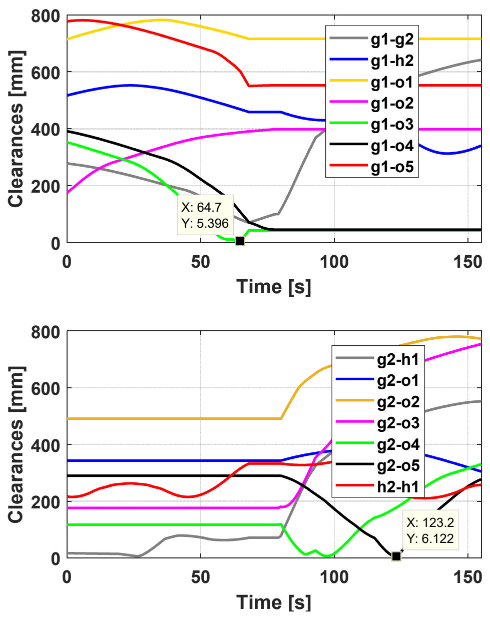


Fig. 21 The Change of minimum clearances between bodies

Accordingly, as the turrets are rotated to their goal positions, the minimum clearance is between g1-o3 which is 5.4 mm and no collision occurs as expected.

V. Conclusion

In this research, a method has been developed with two different ways to obtain a high-dimensional configuration space. One of the ways is obtaining by using point clouds and the second one is using only simulation software. In first way, the bodies in the system are meshed and converted into points and then the configuration space is obtained by using the method of intersection of point clouds. In the second way, this work is carried out using clearance function which measured minimum distance between the surfaces of defined bodies that are defined. A double-turret system is held in the scope of this study. 4-D configuration space of double-turret system is obtained by these two ways. As a result of this, the difference between these two ways is about 1% which depends on the point cloud density. As the point cloud density increases, the difference between the two ways decreases gradually. By the way, the first way is 8.1 times more efficient than the second way which is an advantage of the first way. The need to create a point cloud for each part is seen as a disadvantage of the first way. However, for the second way, 3-D cad models of parts and appropriate input profiles are sufficient to carry out the analysis. At the end of the study, a sample path planning with A* algorithm was made by using the previously obtained 4-D configuration space. Then, the accuracy of the configuration space was proved via using the obtained paths on the simulation model of the double-turret system. To fully verify the results of two different methods on the 3 or more dimensions, different path planning algorithms such as RRT and RRT* can be used or different approach like potential field methods can be proposed. Therefore, the focus should be on full verification of the methods for future work by conducting path planning studies on more cases.

Acknowledgment

The authors would like to thank FNSS Defense Systems Inc. and Scientific and Technological Research Council of Turkey (TÜBİTAK) for their support. They would also like to express sincere appreciation for the valuable editorial help of Mert Şen.

References

[1] M. Bahrin, M. Othman, N. H. N. Azli and M. F. Talib, “Industry 4.0: A review on industrial automation and robotic,” *Jurnal Teknologi (Sciences and Engineering)*, pp. 137– 143, 2016.
[2] Günther Schuh et al., “Collaboration Mechanisms to Increase Productivity in the Context of Industrie 4.0” *in Procedia CIRP*, 2014.
[3] M. Reuter, H. Oberc, M. Wannöffel, D. Kreimeier, J. Klippert, P. Pawlicki and B. Kuhlenkötter, “Learning factories’ trainings as an enabler of proactive workers participation regarding industrie 4.0,” *Procedia Manufacturing*, pp. 354– 360, 2017.
[4] F. Padula and V. Perdereau, “An on-line path planner for industrial manipulators,” *International Journal of Advanced Robotic Systems*, January 2013.
[5] J. Zhao, Y. Chao and Y. Yuan, “A cooperative obstacle-avoidance approach for two-manipulator based on A* algorithm,” *International Conference on Intelligent Robotics and Applications (ICIRA)*, pp. 16-25, 2019.
[6] H. Choset and J. Latombe, “Principles of robot motion: theory, algorithms, and implementations [Book Review],” *IEEE Robotics & Automation Magazine*, vol. 12, 2005.
[7] K. H. Kim, S. Sin and W. Lee, “Exploring 3D shortest distance using A* algorithm in unity3d,” *TechArt: Journal of Arts and Imaging Science*, vol.2, no. 3, pp. 81-85, August 2015.
[8] J. Pan and D. Manocha, “Efficient configuration space construction and optimization for motion planning,” *Engineering*, vol. 1, no 1, pp 46–57, 2015.
[9] “Multiple cradle launcher,” *Roketsan Missiles Inc.*, www.roketsan.com.tr/wpcontent/uploads/2013/05/ IDEX-1.pdf (last accessed: April-2021).
[10] Implementing multi-turret and twin-barrel support with a 3rd soviet heavy line, 2015, Retrieved from http://ritastatusreport.blogspot.com/ 2015/12/implementing-multi-turret-and-twin_16.html
[11] O. Khatib, “Real-Time Obstacle Avoidance for Manipulators and Mobile Robots” *in Proceedings - IEEE International Conference on Robotics and Automation*, 1985.
[12] T. Lozano-Pérez, M. A. Wesley, “An algorithm for planning collision-free paths among polyhedral obstacles,” *Commun of the ACM*, vol. 22, no. 10, pp. 560-570, 1979.
[13] T. Lozano-Pérez, “Automatic planning of manipulator transfer movements,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 11, no. 10, pp. 681–98, October 1981.
[14] G. K. Lin and T. Lozano-Perez, “Spatial Planning: A Configuration Space Approach,” *Ieee Transactions On Computers*, vol. 32, 1983.
[15] P. Jiménez, F. Thomas, and C. Torras, “3d collision detection: a survey,” *Computers & Graphics*, vol. 25, no. 2, pp. 269-285, 2001.
[16] T. Liski, *3-D collision checking for improving machine operator's spatial awareness (Master Thesis),* Aalto University-School of Electrical Engineering, Finland, 2014.
[17] W. Wu, H. Zhu, X. Zhuang, G. Ma and Y. Cai, “A multi-shell cover algorithm for contact detection in the three-dimensional discontinuous deformation analysis,” *Theoretical and Applied Fracture Mechanics*, vol. 72, no. 1, pp. 136–49, 2014.
[18] J. Klein and G. Zachmann, “Point cloud collision detection,” *Computer Graphics Forum*, vol. 23, no. 3, pp. 567-576, 2004.
[19] G. Zachmann, “Minimal Hierarchical Collision Detection,” *ACM Symposium on Virtual Reality Software and Technology, Proceedings, VRST*, 121–28, 2002.
[20] M. Figueiredo, J. Oliveira, B. Araújo, J. Pereira, “An efficient collision detection algorithm for point cloud models,” *20th International Conference on Computer Graphics and Vision, GraphiCon'2010 - Conference Proceedings*, 2010.
[21] J. Han, “An efficient approach to 3D path planning,” *Information Sciences*, vol. 478, pp. 318–30, April 2019.
[22] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, S. Thrun, “Anytime Dynamic A*: An Anytime, Replanning Algorithm,” *in ICAPS 2005 - Proceedings of the 15th International Conference on Automated Planning and Scheduling*, 2005.
[23] J. J. Kuffner, S. LaValle, “RRT-connect: An efficient approach to single-query path planning”, *In: Proceedings of IEEE International Conference on Robotics and Automation*, 995–1001, 2000.
[24] L. Kavraki, P. Svestka, J. C. Latombe and M. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE Trans. Robot. Autom.*, vol. 12, no. 4, pp. 566–580, August 1996.
[25] D. Henrich, C. Wurll, H. Worn, “Online path planning with optimal c-space discretization,” *Proceedings of the 1998 IEEE/RSJ International Conference on Robots and System*, Victoria, BC, Canada, pp. 1479–84, 1998.