



Maintaining Visibility of Dynamic Objects in Cluttered Environments Using Mobile Manipulators and Vector Field Inequalities

DOI:

[10.1109/iros55552.2023.10341763](https://doi.org/10.1109/iros55552.2023.10341763)

[10.1109/IROS55552.2023.10341763](https://doi.org/10.1109/IROS55552.2023.10341763)

Document Version

Accepted author manuscript

[Link to publication record in Manchester Research Explorer](#)

Citation for published version (APA):

DURSUN, FATIH., Adorno, B. V., Watson, S., & Pan, W. (2023). Maintaining Visibility of Dynamic Objects in Cluttered Environments Using Mobile Manipulators and Vector Field Inequalities. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2023* (pp. 6371-6378). (IEEE International Conference on Intelligent Robots and Systems). IEEE. <https://doi.org/10.1109/iros55552.2023.10341763>, <https://doi.org/10.1109/IROS55552.2023.10341763>

Published in:

2023 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2023

Citing this paper

Please note that where the full-text provided on Manchester Research Explorer is the Author Accepted Manuscript or Proof version this may differ from the final Published version. If citing, it is advised that you check and use the publisher's definitive version.

General rights

Copyright and moral rights for the publications made accessible in the Research Explorer are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

Takedown policy

If you believe that this document breaches copyright please refer to the University of Manchester's Takedown Procedures [<http://man.ac.uk/04Y6Bo>] or contact openresearch@manchester.ac.uk providing relevant details, so we can investigate your claim.



Maintaining Visibility of Dynamic Objects in Cluttered Environments Using Mobile Manipulators and Vector Field Inequalities

Fatih Dursun, Bruno Vilhena Adorno, Simon Watson and Wei Pan

Abstract—Vision-based perception has become prevalent in robotic applications, especially in those where the control loop relies on visual data, such as visual servoing. For those applications, ensuring that the features or target object remain visible to the camera is critical, necessitating visibility-aware control. In this paper, we propose a method to guarantee the visibility of a dynamic object using a constrained kinematic controller and Vector Field Inequalities (VFIs) to include a linear visibility constraint. Unlike existing methods, we introduce constraints into the kinematic controller to ensure the target’s visibility without needing a trajectory optimizer or local planner. Our method maintains the target object in the camera field of view (FoV) by representing the FoV with four infinite planes and maintaining the distance between the target object and each plane higher than a predefined distance. We evaluated the proposed approach using a mobile manipulator in two simulations involving cluttered environments: the first scenario involves a stationary target object, whereas the second scenario presents a more challenging workspace involving a moving target. Our results demonstrate that the proposed approach successfully maintains the target within the FoV while avoiding obstacles in the workspace, showing the potential of our method to improve the safety and reliability of visual-servoing-based robotic systems.

I. INTRODUCTION

Vision-based perception has become one of the prevailing onboard sensing approaches for robots with the progression of perception algorithms, the availability of cost-effective cameras, and the improved computational power of computers. Visual servoing, a commonly adopted perception-based technique, controls a robot by using visual feedback [1]. Various visual servoing methods have been proposed to tackle different robotic challenges, including object manipulation, position control, tracking, and landing for Unmanned Aerial Vehicles (UAVs) [2]–[6].

Visual servoing can be classified into two main categories: Position-Based Visual Servoing (PBVS) and Image-Based Visual Servoing (IBVS) [7]–[9]. PBVS techniques determine the three-dimensional pose of the target using acquired visual features and generate control actions based on this estimate. On the other hand, IBVS approaches generate control actions by computing the error between the target and current feature points on the image plane.

The visual servoing control loop relies on visual feedback, making it imperative that the features or objects remain

in continuous view. Loss of sight can lead to catastrophic consequences, especially when the target object is in motion, making maintaining the target’s visibility crucial.

Several methods have been proposed in the literature to ensure visibility, including trajectory optimisation techniques [3], [4], [10], [11] and control barrier functions [6]. Leonard *et al.* proposed a dynamic visibility check algorithm that checks if the 3D coordinates of the target collide with the frustum formed by the camera field of view (FoV) [12]. Nägeli *et al.* [13], [14] introduced a motion planning strategy for aerial videography to optimize the viewpoint of a camera mounted on an aerial robot to enhance the quality of video recordings. They employed a camera model represented by a cone and incorporated a cost term to prevent other drones from entering the field of view (FoV) during scenarios where multiple drones simultaneously capture measurements.

LaValle *et al.* [15] presented a method to ensure target visibility in the context of target following. Their approach employed trajectory optimization techniques within a 2D world involving mobile robots. Lozano *et al.* [16] proposed a vision-based planner for pursuit-evasion games, where the pursuer’s field of view (FoV) is considered. They used a sampling-based planner and considered the evader’s visibility in the tree’s nodes in a 2D environment. Oskam *et al.* [17] proposed a visibility-aware global planning strategy targeting virtual environments, computer games, or other interactive applications.

In this paper, we propose a simple and computationally efficient method based on Vector Field Inequalities (VFIs) that guarantees the target’s visibility at any time. We model the camera FoV with four infinite planes and employ the VFI method to constrain the distance between those planes and the object. We demonstrate the effectiveness of our method using a mobile manipulator in a highly cluttered environment, and our results show that our method can deal with obstacles while ensuring the visibility of the moving target.

Our proposed approach can potentially improve the safety and reliability of visual-servoing-based robotic systems. Unlike the existing literature, we focus on a computationally efficient method for visibility maintenance that can be used in highly cluttered environments and for various types of robots. The main contribution can be summarised as follows:

- We propose a novel method for enforcing visibility constraints that is simple to implement, computationally efficient, and has the potential to enhance the safety and reliability of visual-servoing-based robotic systems significantly.

*This work was funded by the Republic of Türkiye Ministry of National Education, Ankara, Türkiye

Fatih Dursun, Bruno V. Adorno and Simon Watson are with Department of Electrical and Electronic Engineering, and Wei Pan is with Department of Computer Science, and all the authors are with Manchester Centre for Robotics and AI, University of Manchester, UK. (for correspondence: fatih.dursun@postgrad.manchester.ac.uk)

- The method applies to a wide range of robots composed of a serial kinematic chain, including mobile manipulators.
- The proposed method is validated in a cluttered simulation scenario, demonstrating its effectiveness in ensuring the visibility of a moving target while respecting environmental constraints.

The rest of the paper is organized as follows: Section II provides the formulation of the kinematic controller, followed by a summary of the Vector Field Inequalities (VFIs) in Section III; Sections IV and V present the proposed visibility and collision constraints, respectively; Section VI is dedicated to the simulation results, and finally, Section VII concludes the paper.

II. KINEMATIC CONTROL

The differential kinematics of a serial kinematic chain is given by

$$\dot{x} = J\dot{q}, \quad (1)$$

where $q \triangleq q(t) \in \mathbb{R}^n$ is the vector of joint values, $x \triangleq x(q) \in \mathbb{R}^m$ is the vector of task-space variables, and $J \triangleq J(q) \in \mathbb{R}^{m \times n}$ is the task Jacobian matrix. The task-space variables can be specified for any frame in the robot's kinematic chain. For example, this variable could be position, orientation, or pose of the end-effector frame. Provided that a desired task vector $x_d \in \mathbb{R}^m$ with $\dot{x}_d = 0, \forall t$, and the task error $\tilde{x} \triangleq x - x_d$, a control vector $u \in \mathbb{R}^m$ that ensures a stable closed-loop error dynamics [18] is given by

$$u \in \arg \min \quad \|J\dot{q} + \eta\tilde{x}\|_2^2 + \beta \|\dot{q}\|_2^2 \quad (2)$$

subject to $W\dot{q} \leq w$

in which $W \triangleq W(q) \in \mathbb{R}^{r \times n}$, $w \triangleq w(q) \in \mathbb{R}^r$ and $\beta \in \mathbb{R}^{n \times n}$ is a positive definite gain matrix that penalises high joint velocities. The solution to (2) is found using a numerical solver to generate the control input at each time step.

III. VECTOR FIELD INEQUALITIES

The VFIs are differential inequalities that transform the nonlinear constraints in the robot configuration to linear ones in the control inputs [19], [20]. Originally, it was proposed to avoid collisions between pairs of geometric primitives.

Given a signed distance function $d(t) \in \mathbb{R}$ between two collidable entities, the Jacobian providing a relation between the joint velocities and the time derivative of the distance function in general form is given by

$$\dot{d}(q(t), t) = J_d \dot{q} + \zeta(t), \quad (3)$$

where $\zeta(t)$ is the residual that represents the change in distance unrelated to the joint velocities (e.g., distance change due to the moving objects)

A distance function is specified as $\tilde{d}(q(t), t) \triangleq d(q(t), t) - d_{\text{safe}}$ with $d_{\text{safe}} \in \mathbb{R}$ being the arbitrary safe distance. To keep the robot outside of a collision region, the following inequality must hold for all t :

$$-J_d \dot{q} \leq \eta_d \tilde{d}(q(t), t) + \zeta(t), \quad (4)$$

where $\eta_d \in [0, \infty)$ defines the approach velocity.

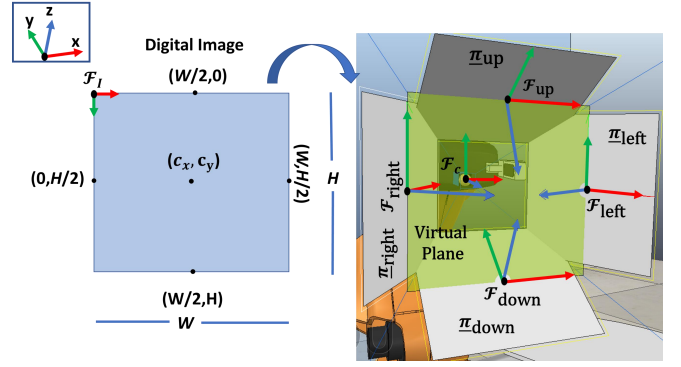


Fig. 1. The representation of the camera FoV using four infinite planes. A virtual plane is defined in front of the camera frame $\mathcal{F}_c$, and the centre points of the boundaries of the digital images are used to find the frames' position and orientation, which are assigned to infinite planes.

IV. VISIBILITY CONSTRAINT

Let the object of interest be at an arbitrary point in the robot workspace and let the camera FoV be represented by four infinite planes, as shown in Fig. 1. To keep the object in camera view, it must stay in the region enclosed by those four planes; the distance between the workspace point and each plane should be less than a threshold distance. The value of the threshold distance determines how close we want the object to be to the camera centre.

A. Representation of Camera FoV

The representation of a plane π^i with respect to (w.r.t.) a frame $\mathcal{F}_i$ in a dual quaternion space is given by [21]

$$\underline{\pi}^i \triangleq n_\pi^i + \varepsilon d_\pi, \quad (5)$$

where $n_\pi^i \in \mathbb{H}_p \cap \mathbb{S}^3$ is the normal to the plane, $d_\pi \in \mathbb{R}$ is the signed perpendicular distance between the origin of the reference frame $\mathcal{F}_i$ and the plane, and ε is the dual unit such that $\varepsilon \neq 0$ but $\varepsilon^2 = 0$ [22]. Given an arbitrary point p_π^i in the plane, the signed perpendicular distance is given by $d_\pi = \langle p_\pi^i, n_\pi^i \rangle$. In our setting, each plane is an entity attached to the robot end-effector, and a corresponding frame is assigned to the plane (Fig. 1). Given the origin of the assigned frames as the arbitrary points in the planes and the unit normal vector of the plane is the z axis of the assigned frames $\mathcal{F}_i$, those four planes w.r.t. $\mathcal{F}_i$ are defined as $\underline{\pi}_i^i = \hat{k}$, where $i \in \{\text{up, down, left, right}\}$.

To represent planes w.r.t. another frame (i.e., camera frame $\mathcal{F}_c$), the following transformation holds:

$$\underline{\pi}_i^c = \text{Ad}_\#(\underline{x}_i^c) \underline{\pi}_i^i, \quad (6)$$

where $\underline{x}_i^c = r_i^c + \varepsilon(1/2)p_{ci}^c r_i^c \in \mathcal{S}$ is the rigid transformation from $\mathcal{F}_c$ to $\mathcal{F}_i$,

$$\text{Ad}_\#(\underline{x}_i^c) \underline{\pi}_i^i \triangleq (\underline{x}_i^c)^\# \underline{\pi}_i^i (\underline{x}_i^c)^* \quad (7)$$

and $(\underline{x}_i^c)^\# = \mathcal{P}(\underline{x}_i^c) - \varepsilon \mathcal{D}(\underline{x}_i^c)$ with $\mathcal{P}$ and $\mathcal{D}$ being the primary and dual parts, respectively [21]. To find the rigid transformation from $\mathcal{F}_c$ to $\mathcal{F}_i$ ($\underline{x}_i^c$), we used a virtual plane placed in front of the camera (Fig. 1). According to the

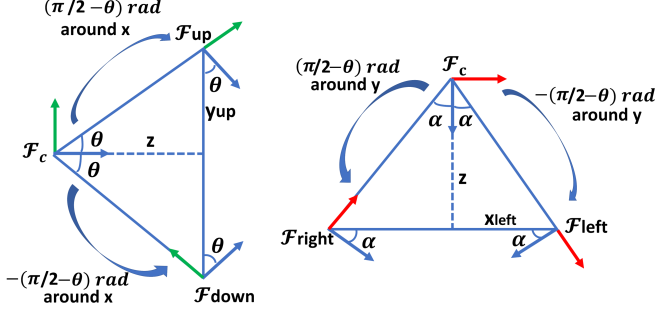


Fig. 2. The illustration of frame orientations w.r.t. the camera frame for π_{up} and π_{down} on the left, and π_{left} and π_{right} on the right.

pinhole camera model [23], any point on the digital image can be back-projected as a ray w.r.t. the camera frame by the following equation,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -(x' - c_x) \frac{z}{f_{\text{px}}} \\ -(y' - c_y) \frac{z}{f_{\text{py}}} \end{bmatrix}$$

where (x, y, z) is the point w.r.t. the camera frame, (x', y') is the point in the digital image (in pixels), (c_x, c_y) is the centre of the digital image, and f_{px} and f_{py} are the focal lengths of the camera in pixels in the x and y directions, respectively. An arbitrary z value (0.1m in our case) can be selected since the obtained point will be on the same plane.

Using the vertical and horizontal limits of the digital image, the origin of frames associated with each plane is given by,

$$p_{\text{up}}^c = -(W/2 - c_x) \frac{z}{f_{\text{px}}} \hat{i} - (0 - c_y) \frac{z}{f_{\text{py}}} \hat{j} + z \hat{k} \quad (8)$$

$$p_{\text{down}}^c = -(W/2 - c_x) \frac{z}{f_{\text{px}}} \hat{i} - (H - c_y) \frac{z}{f_{\text{py}}} \hat{j} + z \hat{k} \quad (9)$$

$$p_{\text{left}}^c = -(0 - c_x) \frac{z}{f_{\text{px}}} \hat{i} - (H/2 - c_y) \frac{z}{f_{\text{py}}} \hat{j} + z \hat{k} \quad (10)$$

$$p_{\text{right}}^c = -(W - c_x) \frac{z}{f_{\text{px}}} \hat{i} - (H/2 - c_y) \frac{z}{f_{\text{py}}} \hat{j} + z \hat{k}, \quad (11)$$

where W and H are the width and height of the digital image, in turn. Table I shows the camera and image-related parameters that we used in our work. The calculation of the frame orientations w.r.t. camera frame is given by

$$r_{\text{up}}^c = \cos((\pi/2 - \theta)/2) + \hat{i} \sin((\pi/2 - \theta)/2) \quad (12)$$

$$r_{\text{down}}^c = \cos(-(\pi/2 - \theta)/2) + \hat{i} \sin(-(\pi/2 - \theta)/2) \quad (13)$$

$$r_{\text{left}}^c = \cos(-(\pi/2 - \alpha)/2) + \hat{j} \sin(-(\pi/2 - \alpha)/2) \quad (14)$$

$$r_{\text{right}}^c = \cos((\pi/2 - \alpha)/2) + \hat{j} \sin((\pi/2 - \alpha)/2) \quad (15)$$

where $\theta = \arctan(y_{\text{up}}/z)$ and $\alpha = \arctan(x_{\text{left}}/z)$ (Fig. 2).

The representation of the planes w.r.t. world frame is found by utilizing the kinematic model of the robot as follows,

$$\pi_i = \text{Ad}_{\#}(\mathbf{x}_c) \pi_i^c, \quad (16)$$

TABLE I

THE CAMERA AND IMAGE-RELATED PARAMETERS.

f_{px}	(W, H)	z	(c_x, c_y)
236	(256, 256)	0.1m	(128, 128)

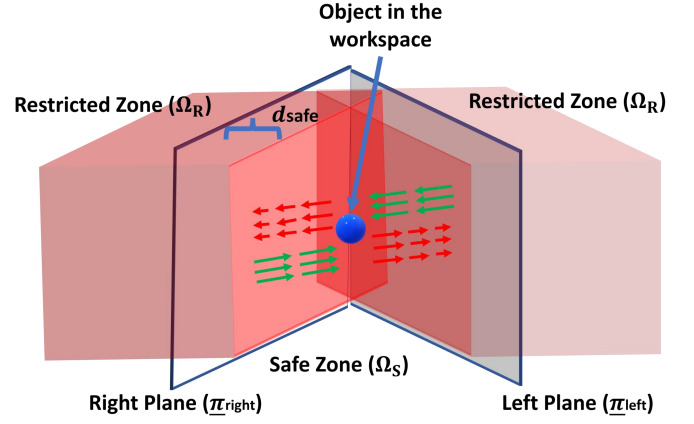


Fig. 3. An illustration of the safe (Ω_S) and restricted zones (Ω_R) formed by two planes. The object is in the area enclosed by planes, which is the safe zone. VFIs (vectors given by red and green arrows) keep the object away from the planes within a safe distance, d_{safe} .

where π_i is the representation of the plane w.r.t. world frame, $\mathbf{x}_c$ is the pose of the camera w.r.t. word frame, and π_i^c is the representation of plane w.r.t. camera frame.

B. Distance Functions and Jacobians

In our setting, the planes composing the safe area are attached to the robot end-effector, and the object is a point in the workspace. The distance between the planes and the point should always be greater than the predetermined safe distance. The area enclosed by planes is the safe region, and outside is the restricted region, as shown in Fig. 3 for two planes. The goal is to keep the object within a safe region. Let a function $d = d(q(t), t) \in \mathbb{R}$ encode the signed distance between each plane and the object, and a safe distance d_{safe} to keep the object outside the restricted region. Based on the distance, $\tilde{d}(q(t), t) = d(q(t), t) - d_{\text{safe}}$, safe regions (Ω_S), and restricted regions (Ω_R) are defined as:

$$\Omega_R = \{q(t) \in \mathbb{R}^n : t \in [0, \infty), \tilde{d}(q(t), t) < 0\}$$

$$\Omega_S = \{q(t) \in \mathbb{R}^n : t \in [0, \infty), \tilde{d}(q(t), t) \geq 0\}.$$

The distance dynamics between each plane and the object is given by

$$\dot{\tilde{d}} = \dot{d} - \dot{d}_{\text{safe}} \quad (17)$$

and, using (3),

$$\dot{\tilde{d}}_{\pi_i, p} = J_{\pi_i, p} \dot{q} + \zeta_{\pi_i, p} \quad (18)$$

holds because $\dot{d}_{\text{safe}} = 0$ for all t .

Using inequality (4), the constraint to keep the object in the area enclosed by four planes is given by

$$-J_{\pi_i, p} \dot{q} \leq \eta \tilde{d}_{\pi_i, p} + \zeta_{\pi_i, p} \quad (19)$$

where $i \in \{\text{up, down, left, right}\}$, and $\tilde{d}_{\pi_i, p} = d_{\pi_i, p} - d_{\pi, \text{safe}}$, with $d_{\pi, \text{safe}}$ being the safe distance and $d_{\pi, p}$ and $J_{\pi_i, p}$ are the plane-to-point distance and its Jacobian, respectively [20].

The proposed constraint affects the system in the following ways:

- If the object is in the safe region $\tilde{d}(q) \geq 0$, any increase in distance is allowed (plane and point separate from

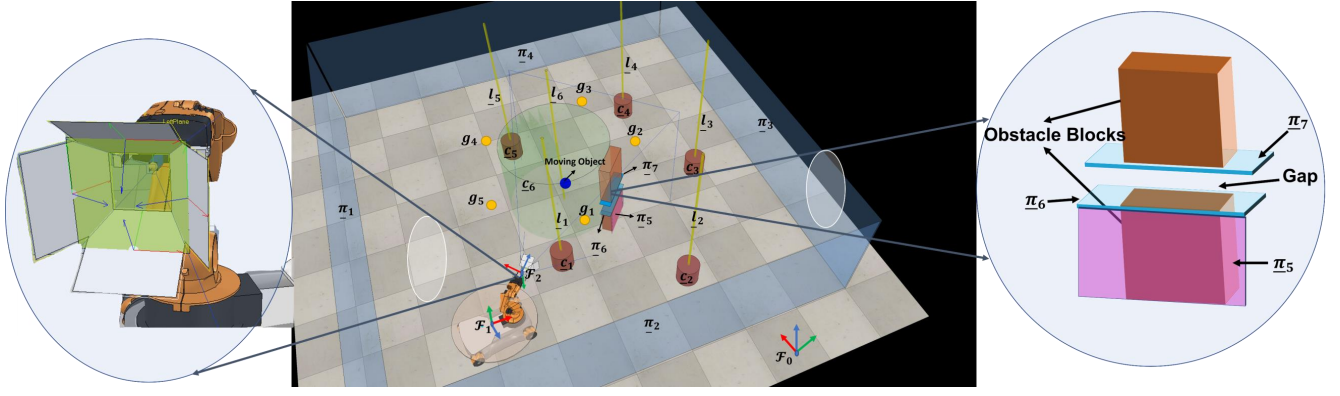


Fig. 4. Simulation scenario created in CoppeliaSim. The robot is supposed to keep the *Moving Object* (coloured in blue) in the camera FoV while traversing between five successive positions given as g_1, g_2, g_3, g_4, g_5 . In addition to visibility constraints, we included several constraints for both the robot arm and the base. These constraints are modelled as planes ($\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, \pi_7$) or cylinders ($\mathcal{C}(l_1, r_1), \mathcal{C}(l_2, r_2), \mathcal{C}(l_3, r_3), \mathcal{C}(l_4, r_4), \mathcal{C}(l_5, r_5), \mathcal{C}(l_6, r_6)$), where l_i is the line and r_i is the radius (which you can define as a safe distance).

each other), $\dot{\tilde{d}} \geq 0 \geq -\eta_d \tilde{d}$ (green arrows in Fig. 3). However, in our case, the increase will be limited by constraints introduced by other planes (red arrows in Fig. 3).

- If the object is in the safe region $\tilde{d}(q) \geq 0$ and the object moves towards the plane, the maximum rate of decrease in the distance rate is exponential and is given by $\dot{\tilde{d}} = -\eta_d \tilde{d}$, which means that as the plane and the object come closer, the maximum rate of approach decreases. Finally, when $\tilde{d} = 0$, the restriction becomes $\dot{\tilde{d}} = 0$; therefore, the object does not enter the restricted area.

V. COLLISION CONSTRAINTS

To avoid collisions with obstacles, we define several constraints for the mobile manipulator's arm and base. The objects in the environment are modelled as cylinders or planes as shown in Fig. 4. For the robot end-effector, we placed an obstacle between target points g_1 and g_2 , which has a small gap passable by the robot. To provide these constraints, we defined two planes (π_6 and π_7 in Fig. 4) to compose the upper and lower limit of the passable gap. Since the planes are infinite, we activate this constraint when the robot end-effector is close to these planes than a certain distance threshold. In addition, a constraint is provided to avoid collision between the end-effector and the workspace of the object represented by a cylinder $\mathcal{C}_6$ in Fig. 4.

Similarly, we included multiple constraints for the robot base, which are six cylinders (five of them are spread in the environment ($\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_4, \mathcal{C}_5$) and one enclosing the workspace ($\mathcal{C}_6$)) and five planes (four of them enclose the environment ($\pi_1, \pi_2, \pi_3, \pi_4$) and one belonging to the obstacle block between target points g_1 and g_2 , (π_5)).

The constraints between a frame in the robot kinematic chain and geometric primitives are given by

$$-J_{p, l_i} \leq \eta \tilde{d}_{p, l_i} \text{ and } -J_{p, \pi_j} \leq \eta \tilde{d}_{p, \pi_j} \quad (20)$$

for the cylinder and plane, respectively, where $i = 6$ and $j = \{6, 7\}$ for the end-effector while $i = \{1, 2, 3, 4, 5, 6\}$ and $j = \{1, 2, 3, 4, 5\}$ for the robot base. In addition, $\tilde{d}_{p, l_i} = d_{p, l_i} - d_{l, \text{safe}}$ and $\tilde{d}_{p, \pi_i} = d_{p, \pi_i} - d_{\pi, \text{safe}}$ with $d_{l, \text{safe}}$ and $d_{\pi, \text{safe}}$

being the safe distance from the robot end-effector or base to the line and planes. We utilised the point-line distance and its Jacobian for lines, and similarly, for planes, the point-plane distance and its Jacobian [20].

VI. SIMULATIONS AND DISCUSSION

We performed simulations using CoppeliaSim 4.1¹ and Python to validate the proposed method. The simulations were performed on a computer with an Intel i9-12900H processor running at 2.9 GHz, 32 GB of RAM, and Ubuntu version 20.04. We utilized DQ Robotics² [24] for all dual quaternion operations, robot modelling and control, and the description of geometric primitives. In addition, the PyRep library³ [25] is used for communication between CoppeliaSim and Python. We first validate the proposed method in a scenario where the target object is stationary. In the second scenario, we evaluate the method using a moving target that executes a circular path. We used quadratic programming provided by DQ Robotics to optimise the convex quadratic function with linear constraints given by Eq. 2.

A. Simulation Setup

The objective is to move the robot through five consecutive target positions while ensuring that a target object remains within the camera FoV. In addition, the robot must prevent collisions in the environment comprising obstacles for both the robot arm and base as shown in Fig. 4. The robot comprises a 5-DOF manipulator serially coupled to a 3-DOF holonomic base, totalling an 8-DOF. To maintain safe distances in the workspace, we have selected a distance of $d_{\pi, \text{safe}} = 0.2m$ for the planes used in the visibility constraint. The safe distances for the robot base are determined based on the radius of the base, $r_{\text{base}} = 0.35m$, while the safe distance for the end-effector is $d_{\text{eff}} = 0.06m$.

Furthermore, we employed a random environment generation strategy to assess both the computational time

¹<https://www.coppeliarobotics.com/>

²<https://github.com/dqrobotics>

³<https://github.com/stepjam/PyRep>

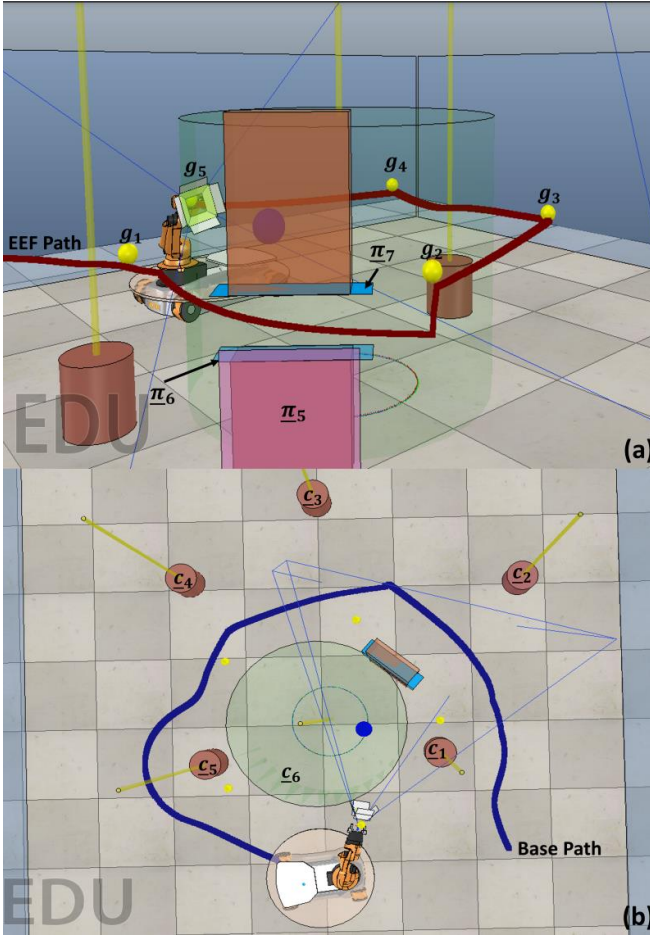


Fig. 5. The illustration of the paths followed by the end-effector (a) and base of the robot (b). The robot moves between target positions, $(g_1, g_2, g_3, g_4, g_5)$ and avoid all obstacles for the end-effector (π_6, π_7) and the base ($c_1, c_2, c_3, c_4, c_5, c_6$ and π_5).

required for generating control inputs and the generalization capability of the proposed method, as illustrated in Fig. 9. This approach enabled us to evaluate the performance of our methodology across a range of diverse scenarios.

B. Stationary Target Object

In the initial configuration, we assess the effectiveness of the proposed method when the target object remains stationary. We have defined a series of five sequential target positions $(g_1, g_2, g_3, g_4, g_5)$ surrounding a cylinder (c_6), containing the object, which the robot must reach while maintaining the object within the camera FoV. We assume that the 3D position of the object is provided to the robot. Additionally, to simulate a more realistic scenario, we have

TABLE II

MINIMUM SIGNED DISTANCE VALUES BETWEEN EACH PLANE AND THE OBJECT WHEN THE OBJECT IS STATIONARY.

	Up Plane	Down Plane	Left Plane	Right Plane
Distance (m)	0.215	0.2	0.2	0.230

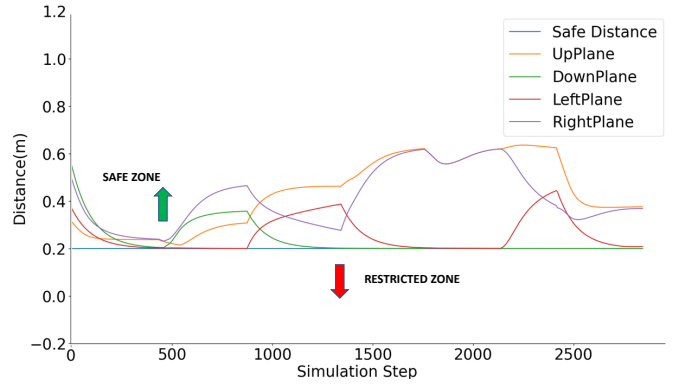


Fig. 6. The distance between each plane and the stationary object. The VFIs are employed to maintain the object within a designated safe zone enclosed by the planes. Specifically, the constraints ensure that the distance between each plane and the object always remains greater than a predefined safe distance, d_{safe} .

included several obstacles for the robot to navigate around while completing the task, as illustrated in Fig. 4.

Fig. 5 illustrates the paths of both the end-effector and the mobile manipulator's base. Throughout the movement between each position $(g_1, g_2, g_3, g_4, g_5)$, the robot respects to all the constraints specified for both the end-effector and the base.

To ensure that the object remains within a safe zone at all times, we have implemented constraints based on VFIs. These constraints ensure that the distances between the four planes and the object remain greater than a predefined safe distance of $d_{\pi, safe} = 0.2m$. Table II shows the minimum distances between the object and the four planes throughout the simulation.

C. Moving Target Object

In the second scenario, we introduce a moving object that executes a circular path with the same objective as in the previous setup. The simulation results depicted in Fig. 7 demonstrate that the proposed approach can maintain the object within the camera FoV while performing the primary task, even when the object is in motion. However, in some instances, the distance between the object and the planes falls below the predefined safe distance, as opposed to the stationary object in the first scenario.

This is an expected result since the object's movement is not considered in the VFIs. To solve this issue, one needs to provide the robot with the object's velocity, which is beyond the scope of this work. To mitigate the violation of the safe

TABLE III

MINIMUM SIGNED DISTANCE (DIST.) AND SAFE DISTANCE VIOLATION (VIO.) VALUES BETWEEN EACH PLANE AND THE OBJECT FOR $\eta_d = 1$ AND $\eta_d = 10$ WHEN THE OBJECT TRAVERSES ON A CIRCULAR PATH.

	Up		Down		Left		Right	
	$\eta_d=1$	$\eta_d=10$	$\eta_d=1$	$\eta_d=10$	$\eta_d=1$	$\eta_d=10$	$\eta_d=1$	$\eta_d=10$
Dist.	0.156	0.193	0.145	0.193	0.085	0.185	0.192	0.189
Vio.	0.044	0.007	0.055	0.007	0.115	0.015	0.008	0.011

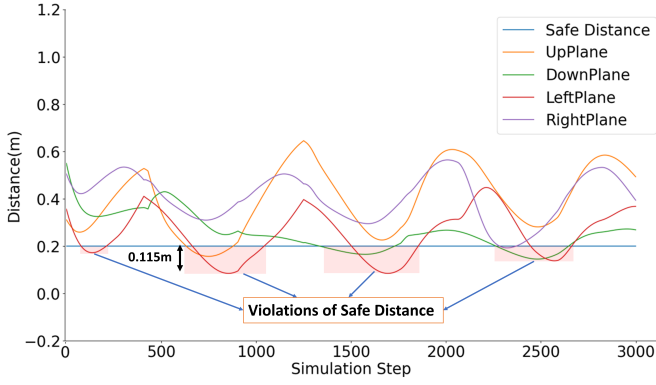


Fig. 7. The distance between each plane and the moving object for $\eta_d = 1$. The object constantly moves on a circular path in the region enclosed by a cylinder, $\mathcal{C}_G$. It stays in the camera FoV but with some violations of the safe distance, which is expected because the object velocities are not being considered in the VFIs.

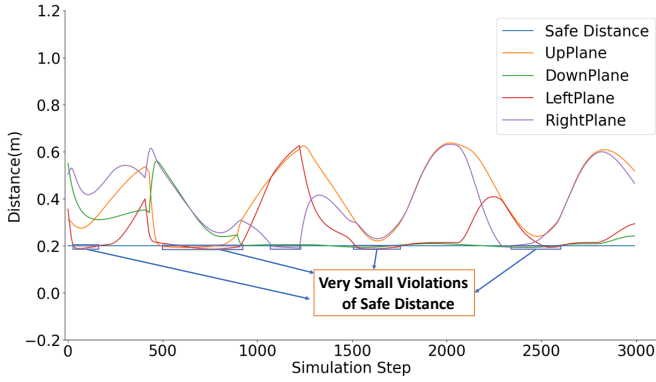


Fig. 8. The distance between each plane and the moving object for $\eta_d = 10$. The object constantly moves on a circular path in the region enclosed by a cylinder, $\mathcal{C}_G$. It stays in the camera FoV, and violations of the safe distance decreases significantly due to the increase in η_d .

distance, the convergence speed of the VFIs can be increased by increasing the parameter η_d ; that is, the higher the value of η_d , the faster the controller responds to changes in distance. The results illustrated in Fig. 8, after increasing η_d from 1 to 10, indicate a considerable reduction in the frequency of safe distance violations. Table III outlines the minimum distances between the planes and object for two η_d values of 1 and 10.

D. Computational Time and Generalization

To showcase the computational time and generalization capability of the proposed method, we devised a randomized environment as depicted in Fig. 9. We used rectangular regions to sample different robot base poses and obstacle and target positions, denoted as $\Omega_R, \Omega_O, \Omega_T$, respectively. The robot starts at a random pose in Ω_R , and moves to the target position in Ω_T while keeping the target within view. Additionally, the number of obstacles is randomly selected, ranging from 5 to 10, as well as their corresponding radii, varying between 0.1 m and 0.2 m. Lastly, the visibility target position was randomly sampled on the z -axis of the camera frame, with distances ranging from 2 m to 4 m.

Using the described strategy, we generated 100 random

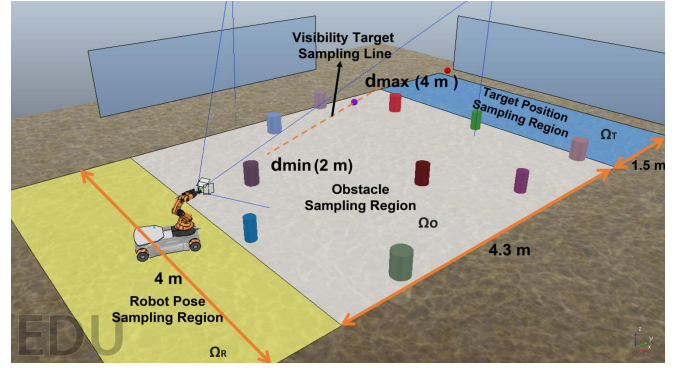


Fig. 9. Random environment that includes rectangular regions ($\Omega_R, \Omega_O, \Omega_T$) to sample the robot pose and obstacle and target positions. A visibility target is sampled on a line in the camera frame's z -axis direction.

scenarios. Half contained a traversable gap between obstacles, whereas the other half did not consider this distance limit (i.e., the scenario could end up with no traversable passages, preventing the robot from reaching the target position). Table IV provides the times required to issue one-step control inputs (base and joint velocities) and success rates for the first and second types of scenarios, as well as the overall outcomes, which account for all scenarios.

In terms of computational time, the results demonstrate that the proposed method efficiently generates control inputs, specifically velocities for the base and arm, with a worst-case time of 10 ms. The average computation time required for generating these inputs is only 2.9 ms. These results emphasise the method's computational efficiency.

The robot completed the main task, reaching the randomly generated target position in 46 out of 50 randomly generated environments (96% success rate) with traversable passages. However, when the obstacle generation did not ensure that traversable passages would exist, the success rate decreased to 86%, resulting in an overall success rate of 89%. This is expected because some environments might have clutter arranged in a way that no passage wide enough exists. Nonetheless, the robot never collided with obstacles and always kept the target within its FoV, highlighting the efficacy of the proposed constrained controller with visibility constraints. Fig. 10 illustrates four representative scenarios (a–d) that depict the failed cases, offering insights into the remaining unsuccessful instances.

TABLE IV
THE ELAPSED TIME TO CALCULATE ONE-STEP CONTROL ACTIONS (BASE AND JOINT VELOCITIES) AND THE ROBOT'S SUCCESS RATE IN RANDOM ENVIRONMENTS.

	Gap	No Gap	Overall
Max Time (ms)	7.9	10.0	10.0
Min Time (ms)	2.5	2.5	2.5
Avg Time (ms)	2.8	3.0	2.9
Standard Deviation	0.000128	0.000126	0.000128
Success Rate (%)	92	86	89

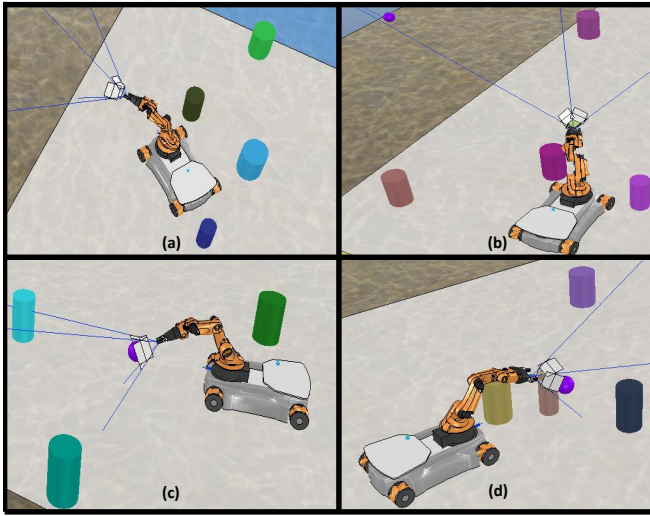


Fig. 10. Different failure cases. The robot fails to reach the target position due to obstacles (local minimum) in (a) and (b), and due to the proximity of the visibility target to the camera frame in (c) and (d).

Figs. 10 (a–b) show cases in which the controller failed to reach the target positions due to local minima. This outcome is expected, as the proposed controller operates locally, and obstacle avoidance is a hard constraint. To address this issue, a global planner could be considered to plan a path for the robot base while our constrained controller generates commands to follow the desired path and ensure visibility.

Fig. 10(c) presents a situation wherein the visibility target is too close to the camera frame, while the green obstacle restricts the robot’s manoeuvrability. The robot must navigate towards the visibility target to avoid the obstacle, but the proximity of the visibility target to the camera frame limits this action. Fig. 10(d) shows the robot stuck in a local minimum due to high clutter and the visibility target being in very close proximity. Since the target position is behind the visibility target, the robot would have to recede and go around the obstacles to reach the other side and continue moving to the target position. However, this manoeuvre would imply increasing the error to the target in the controller objective function, which is not allowed. Hence the robot stops. Finally, Fig. 11 illustrates the mean and standard deviation of the position error for all successful scenarios, showing the error convergence to zero.

VII. CONCLUSIONS

In this work, we proposed a computationally efficient visibility constraint method based on the Vector Field Inequalities (VFIs) framework to keep a target object in the camera field of view (FoV). We evaluated the proposed method in two simulation scenarios involving stationary and moving objects with a mobile manipulator. The results demonstrate that the method ensures the target’s visibility while traversing to five positions and avoiding collisions in the environment. The proposed method is general and can be employed for any robot with a serial kinematic chain, including mobile manipulators. It can be integrated into vision-based applications, such as visual-servoing, to

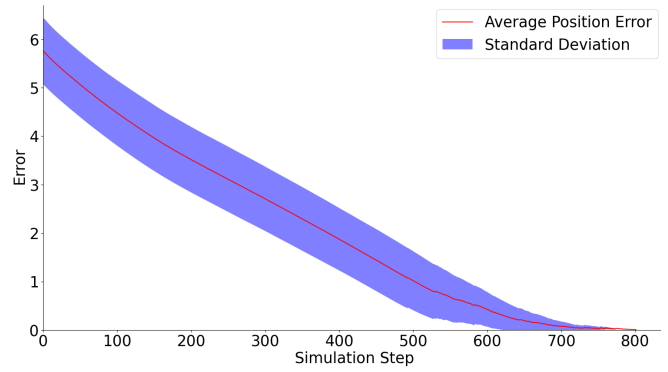


Fig. 11. The mean and standard deviation of the position error for all successful scenarios.

enhance the target or feature tracking performance of robotic systems in highly cluttered environments.

The proposed method assumes that only the end-effector position is controlled, ensuring that there is no conflict between the main task and the visibility constraint. However, a conflict may arise in other applications where both end-effector position and orientation are controlled. To address this issue, future work will focus on developing a strategy to disable the visibility constraint temporarily to allow the robot to perform the primary task.

ACKNOWLEDGMENT

This work was funded by the Republic of Türkiye Ministry of National Education, Ankara, Türkiye.

REFERENCES

- [1] F. Chaumette and S. Hutchinson, “Visual servo control. I. Basic approaches,” *IEEE Robot. Automat. Mag.*, vol. 13, no. 4, pp. 82–90, Dec. 2006.
- [2] V. Lippiello, J. Cacace, A. Santamaria-Navarro, J. Andrade-Cetto, M. A. Trujillo, Y. R. Rodríguez Esteves, and A. Viguria, “Hybrid visual servoing with hierarchical task composition for aerial manipulation,” *IEEE Robot. Automat. Mag.*, vol. 1, no. 1, pp. 259–266, Jan. 2016.
- [3] D. Falanga, P. Foehn, P. Lu, and D. Scaramuzza, “PAMPC: Perception-Aware Model Predictive Control for Quadrotors,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, Oct. 2018, pp. 1–8.
- [4] B. Penin, R. Spica, P. R. Giordano, and F. Chaumette, “Vision-based minimum-time trajectory generation for a quadrotor UAV,” in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, Sept. 2017, pp. 6199–6206.
- [5] D. Lee, T. Ryan, and H. J. Kim, “Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing,” in *Proc. IEEE International Conference on Robotics and Automation*, Saint Paul, MN, USA, May 2012, pp. 971–976.
- [6] D. Zheng, H. Wang, J. Wang, X. Zhang, and W. Chen, “Toward visibility guaranteed visual servoing control of quadrotor UAVs,” *IEEE/ASME Trans. Mechatron.*, vol. 24, no. 3, pp. 1087–1095, June 2019.
- [7] E. Altug, J. Ostrowski, and R. Mahony, “Control of a quadrotor helicopter using visual feedback,” in *Proc. IEEE International Conference on Robotics and Automation*, Washington, DC, USA, May 2002, pp. 72–77.
- [8] L. Mejias, P. Campoy, S. Saripalli, and G. Sukhatme, “A visual servoing approach for tracking features in urban areas using an autonomous helicopter,” in *Proc. IEEE International Conference on Robotics and Automation*, Orlando, FL, USA, May 2006, pp. 2503–2508.
- [9] N. Guenard, T. Hamel, and R. Mahony, “A practical visual servo control for an unmanned aerial vehicle,” *IEEE Transactions on Robotics*, vol. 24, no. 2, pp. 331–340, Apr. 2008.

- [10] M. Sheckells, G. Garimella, and M. Kobilarov, "Optimal visual servoing for differentially flat underactuated systems," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Daejeon, Korea (South), Oct. 2016, pp. 5541–5548.
- [11] C. Potena, D. Nardi, and A. Pretto, "Effective target aware visual navigation for UAVs," in *Proc. European Conference on Mobile Robots (ECMR)*, Paris, France, Sept. 2017, pp. 1–7.
- [12] S. Leonard, E. A. Croft, and J. J. Little, "Dynamic visibility checking for vision-based motion planning," in *Proc. IEEE International Conference on Robotics and Automation*, Pasadena, CA, USA, May 2008, pp. 2283–2288.
- [13] T. Nageli, L. Meier, A. Domahidi, J. Alonso-Mora, and O. Hilliges, "Real-time planning for automated multi-view drone cinematography," *SIGGRAPH*, vol. 36, no. 4, 2017.
- [14] T. Nageli, J. Alonso-Mora, A. Domahidi, D. Rus, and O. Hilliges, "Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization," *IEEE Robotics and Automation Letters*, vol. 2, no. 3, pp. 1696–1703, 2017.
- [15] S. LaValle, H. Gonzalez-Banos, C. Becker, and J.-C. Latombe, "Motion strategies for maintaining visibility of a moving target," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1, 1997, pp. 731–736.
- [16] E. Lozano, U. Ruiz, I. Becerra, and R. Murrieta-Cid, "Surveillance and collision-free tracking of an aggressive evader with an actuated sensor pursuer," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6854–6861, 2022.
- [17] T. Oskam, R. W. Sumner, N. Thuerey, and M. Gross, "Visibility transition planning for dynamic camera control," in *Proc. ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, New York, NY, USA, 2009, p. 55–65.
- [18] M. M. Marinho and B. V. Adorno, "Adaptive constrained kinematic control using partial or complete task-space measurements," *IEEE Transactions on Robotics*, vol. 38, no. 6, pp. 3498–3513, Dec. 2022.
- [19] B. Faverjon and P. Tournassoud, "A local based approach for path planning of manipulators with a high number of degrees of freedom," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 4, 1987, pp. 1152–1159.
- [20] M. M. Marinho, B. V. Adorno, K. Harada, and M. Mitsuishi, "Dynamic active constraints for surgical robots using vector-field inequalities," *IEEE Transactions on Robotics*, vol. 35, no. 5, pp. 1166–1185, Oct. 2019.
- [21] B. Vilhena Adorno, "Robot kinematic modeling and control based on dual quaternion algebra — Part I: Fundamentals," Feb. 2017, working paper or preprint. [Online]. Available: <https://hal.science/hal-01478225>
- [22] J. M. Selig, *Geometric Fundamentals of Robotics*, 2nd ed. Springer-Verlag New York Inc., 2005.
- [23] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.
- [24] B. V. Adorno and M. M. Marinho, "DQ robotics: A library for robot modeling and control," *IEEE Robot. Automat. Mag.*, vol. 28, no. 3, pp. 102–116, Sept. 2021.
- [25] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.