Matrix Multiplication-Driven Repulsive Fields for Manipulator Kinematic Obstacle Avoidance

Jakob Baumgartner[1] and Gregor Klančar[2]

***Abstract*— The article presents a new method for obstacle avoidance for robotic manipulators. It uses occupancy voxel grids and convolutional matrix kernels to calculate repulsive velocities based on the proximity of obstacles. This approach allows for safe and agile navigation in changing environments. Repulsive velocities generated through matrix multiplication are processed using tri-linear interpolation, resulting in smooth and efficient movement away from potential hazards. The integration of this method with inverse kinematic control enables manipulator motion planning and obstacle avoidance capabilities. Simulation results prove the effectiveness of this method in various scenarios, showcasing its potential for local motion planning and obstacle avoidance.**

I. INTRODUCTION

Motion planning [1] is a technique in robotics, that is used to calculate joint changes leading a manipulator from initial to a target configuration, while addressing constraints and optimization criteria. This task is facilitated by the manipulators' redundant degrees of freedom (DOF), enabling it to undertake various configurations to not only reach their End Effector (EE) target but also optimize for secondary objectives such as obstacle avoidance and minimizing joint torques [2].

While global motion planning techniques provide comprehensive solutions by examining the entire configuration space, they fall short in terms of smoothness and real-time application, making them less ideal for dynamic settings [1]. Conversely, local planning methods, particularly inverse kinematics [3] and quadratic programming [4], [5], [6], offer prompt, smooth trajectories suitable for real-time operations. However, these methods are prone to fall into local minima due to their incremental planning approach.

Often used approach is a combination [7] of global and local planning strategies, leveraging global planners for static environment navigation and local planners for adjusting to dynamic changes. This implementation enhances the manipulator's ability to navigate complex environments by combining the comprehensive pathfinding capabilities of global methods with the adaptability and efficiency of local techniques.

The Artificial Potential Field (APF) concept [8], employs repulsive forces for obstacle repulsion and attractive forces to guide the robot towards target. Different variations of the method have been proposed adapting APF for manipulator motion planning [9], [10], [11], [12], [2].

In addressing dynamic environments, technologies such as Euclidian Signed Distance Field (ESDF) grids, derived from sensor data or pre-established occupancy grids, play a crucial role in real-time obstacle avoidance [13], [14], [15].

The introduced approach utilizes occupancy voxel grids [16] to enable safe and dynamic navigation for robot manipulators in changing environments. A method for obstacle avoidance in robotic manipulators via matrix multiplication-driven repulsive fields is presented. By representing the workspace with voxels and calculating repulsive velocities based on proximity to obstacles, our method allows the robot to determine safe movement directions and velocities in real-time. Repulsive velocities gain strength as the robot approaches an obstacle, guiding it to move away from potential collisions, and weaken when the robot is at a safe distance, maintaining an optimal distance from hazards. This locally-calculated repulsive field ensures the robot moves safely and efficiently in dynamic environments.

Section II outlines the mathematical framework for generating repulsive velocities. Section III integrates this with inverse kinematic control, focusing on obstacle navigation. Section IV presents simulations demonstrating the method's effectiveness in various scenarios, highlighting its potential in robotic motion planning and obstacle avoidance.

II. REPULSIVE FIELD CALCULATION

The method leverages a voxel-based representation of the surrounding space, to dynamically assess potential collision threats and calculate directional repulsive velocities for smooth and safe navigation through complex environments.

Kernel multiplication is employed to ascertain the repulsive velocity components within Cartesian space. We select a segment from the occupation grid, that is centered in the point of interest (POI) on the robot, where we want to calculate the repulsive velocities. The segment needs to be of the same size as the directional matrix, for the cordinate axis in which we are calulating the repulsive velocity. This results in two identically sized 3D matrices: one serving as a "window" into the obstacle grid ($\mathbf{G_d}$) and the other as the directional kernel ($\mathbf{W_d}$).

Applying the Hadamard (element-wise) product to the segment of the occupation grid ($\mathbf{G_d}$) and the kernel ($\mathbf{W_d}$) results in a new matrix. By summing over all values in this resultant matrix, we calculate the repulsive velocity component for that specific direction. This process is independently conducted for each Cartesian coordinate, ultimately yielding the comprehensive repulsive velocity vector $\mathbf{v}_i = [v_{x_i}, v_{y_i}, v_{z_i}]$.

$$v_d = \sum_{i,j,k} (\mathbf{G} \odot \mathbf{W})_{ijk} = \sum_i \sum_j \sum_k g_{\Delta i \Delta j \Delta k} \; w_{\Delta i \Delta j \Delta k} \tag{1}$$

Where $g_{\Delta i \Delta j \Delta k}$ and $w_{\Delta i \Delta j \Delta k}$ denote i-th, j-th, k-th indexed elements of each matrix.

A. KERNELS AND THEIR MATRIX APPLICATIONS

The fundamental concept of convolutional directional kernels lies in computing the repulsive field individually for each direction within the Cartesian coordinate system.

Kernels are designed as three-dimensional matrixes with a primary kernel axis i aligned along a specific Cartesian direction, corresponding to the calculated repulsive velocity. The two secondary kernel axes j, k are orthogonal to the primary axis. The distribution of values along the primary axis is inversely symmetric, exhibiting positive values on one side and negative values on the other, with the zero valued cell in the center of the kernel, where the values jump from max negative to max positive magnitude. The function of the increase in magnitude along the primary axis of the kernel defines the shape of the repulsive velocity field, determining how the repulsive velocity changes as the agent approaches an obstacle. Moreover, it is essential for the magnitudes at the kernel's periphery to be minimal, promoting a smooth increase in repulsive velocity when approaching the obstacle rather than a sudden spike.

We introduce two distributions for the primary axis weights, namely Gaussian and linear. The selection of weight distribution is contingent on the desired profile of repulsive velocities within the Artificial Potential Field (APF).

First proposed function is mirrored normal or Gaussian distribution, the standard deviation (σ) modulates the rate at which field values escalate as obstacles are approached.

$$w_{\Delta i} = \begin{cases} e^{-\frac{\Delta i^2}{2\sigma^2}}/(\sigma\sqrt{2\pi}) & \text{if } \Delta i > 0 \\ 0 & \text{if } \Delta i = 0 \\ -e^{-\frac{\Delta i^2}{2\sigma^2}}/(\sigma\sqrt{2\pi}) & \text{if } \Delta i < 0 \end{cases} \tag{2}$$

Terms $\Delta i = c_i - i$, $\Delta j = c_j - j$, and $\Delta k = c_k - k$ denote the displacement index from the matrix's central field, where fields' components are calculated.

Second proposed distribution is mirrored linear, where the $\Delta i_{max} = \lfloor \frac{\text{range}}{\Delta \text{R}} \rfloor$ is the rounded down half length of the primary axis kernel.

$$w_{\Delta i} = \begin{cases} \frac{\Delta i_{max} - \Delta i}{\Delta i_{max}} & \text{if } \Delta i > 0 \\ 0 & \text{if } \Delta i = 0 \\ \frac{\Delta i_{max} + \Delta i}{\Delta i_{max}} & \text{if } \Delta i < 0 \end{cases} \tag{3}$$

The length i_{max} of the primary axis is critical, as it dictates the detection range for obstacles.

Since our need is to detect also obstacles that dont align perfectly along the cartesian direction, it is important that our matrixes have width (i axis) and height (j axis). However as we want bigger repulsive field when the obstacle is head on in the direction than when the obstacle is off the cardinal direction, we propose the following multiplicator, to account for the off direction obstacles.

The length $\Delta j_{max} = \lfloor \frac{\text{width}}{2\Delta \text{R}} \rfloor$ and height $\Delta k_{max} = \lfloor \frac{\text{height}}{2\Delta \text{R}} \rfloor$ of the orthogonal axes influences the peripheral detection range for obstacles.

When selecting the width and height of the kernel, we must consider the density of the neighboring points of interest on the manipulator, ensuring that the combination of kernels adequately cover the entire manipulator's surrounding area.

We propose a sinusoidal function for smooth orthogonal axis transitions when approaching obstacles.

$$w_{\Delta j} = \begin{cases} \sin(|\Delta j| \, \pi/(2\Delta j_{max})) & \text{if } \Delta j < 0 \text{ or } \Delta j > 0 \\ 1 & \text{if } \Delta j = 0 \end{cases} \tag{4}$$

Another option is to use linearly falling weights.

$$w_{\Delta j} = \begin{cases} \frac{\Delta j_{max} - \Delta j}{\Delta j_{max}} & \text{if } \Delta j < 0 \text{ or } \Delta j > 0 \\ 1 & \text{if } \Delta j = 0 \end{cases} \tag{5}$$

By multiplieing weights components for primary and both orthagonal axis we get matrix kernels. We calculate a different kernel for every cartesian direction.

$$w_{\Delta i \Delta j \Delta k} = w_{\Delta i} \cdot w_{\Delta j} \cdot w_{\Delta k} \tag{6}$$

B. 3D INTERPOLATION

The occupancy grid's representation of the robot's environment is inherently discrete, possessing finite resolution, in contrast to the continuous nature of Cartesian space. The approximation or rounding necessary to transition from Cartesian coordinates to discrete grid representations introduces discontinuities in the calculated repulsive field. To counteract this, tri-linear interpolation is utilized, facilitating a smooth transition to a continuous repulsive field value, as illustrated in Fig. 1.

The indexes of the eight surrounding cells $s = 1...8$ are calculated by scaling the selected POI coordinates x, y, z by grid resolution and than rounding the position to the nearest lower and upper integer positions (eq. (7)).

$$\vec{P} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \\ \mathbf{Z} \end{bmatrix} = \begin{bmatrix} \lfloor \mathbf{p}_{\text{POI}}(1) \times \Delta \text{R} \rfloor & \lceil \mathbf{p}_{\text{POI}}(1) \times \Delta \text{R} \rceil \\ \lfloor \mathbf{p}_{\text{POI}}(2) \times \Delta \text{R} \rfloor & \lceil \mathbf{p}_{\text{POI}}(2) \times \Delta \text{R} \rceil \\ \lfloor \mathbf{p}_{\text{POI}}(3) \times \Delta \text{R} \rfloor & \lceil \mathbf{p}_{\text{POI}}(3) \times \Delta \text{R} \rceil \end{bmatrix} \tag{7}$$

Once we got the indexes of the eight surrounding cells of our POI, we calculate the repulsive velocity vectors in each point using kernel matrix multiplication method (seperately for each cartesian direction) (eq. (8)).

$$\mathbf{V}rep_{xyz,nml} = \text{rep}(\mathbf{X}[n], \mathbf{Y}[m], \mathbf{Z}[l]) \quad \forall n, m, l \in \{1, 2\} \tag{8}$$

Trilinear interpolation method works on a 3-dimensional regular grid. Before we can start with the interpolation we need to calculate the distance between POI and smaller coordinates of the cells where we calculated the repulsive velocities (eq. (9)). The calculated repulsive velocity values

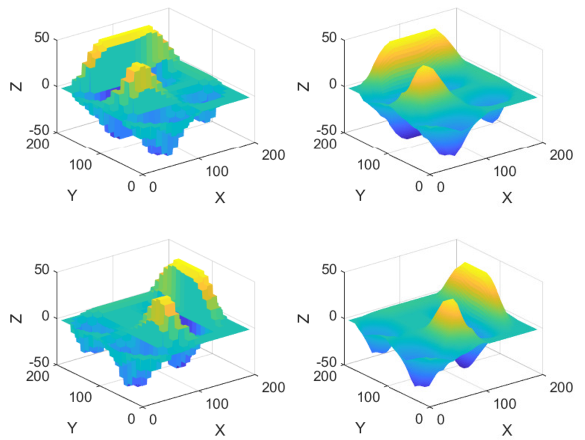

Fig. 1. Repulsive field generation via convolutional kernels. Left: Original repulsive fields in X (top) and Y (bottom) directions. Right: Corresponding fields using interpolation, showcasing enhanced smoothness.

are located at the centers of the cells. Therefore, before interpolation, we shift the values of the cells coordinates by half the resolution of the obstacle grid in each direction. That is, the cell coordinates are same as x, y, z, cell index and plus a half of grid resolution $\Delta \mathrm{R}$ step.

$$\begin{bmatrix} \Delta x \\ \Delta y \\ \Delta z \end{bmatrix} = \begin{bmatrix} \frac{\left(P_x - \left(\mathbf{X}(1) + \frac{1}{2}\Delta R\right)\right)}{(\mathbf{X}(2) - \mathbf{X}(1))} \\ \frac{\left(P_y - \left(\mathbf{Y}(1) + \frac{1}{2}\Delta R\right)\right)}{(\mathbf{Y}(2) - \mathbf{Y}(1))} \\ \frac{\left(P_z - \left(\mathbf{Z}(1) + \frac{1}{2}\Delta R\right)\right)}{(\mathbf{Z}(2) - \mathbf{Z}(1))} \end{bmatrix} \tag{9}$$

The result of the interpolation is agnostic of the order of the operations. We first interpolate along the x-axis, followed by along the y-axis and finally along z-axis.

$$\begin{aligned} \mathbf{V}rep_{xyz,jk} =& \mathbf{V}rep_{xyz,0jk}(1-\Delta x) \\ &+ \mathbf{V}rep_{xyz,1jk}\,\Delta x \quad \forall j,k \in \{1,2\} \end{aligned} \tag{10}$$

$$\begin{aligned} \mathbf{V}rep_{xyz,k} =& \mathbf{V}rep_{xyz,0k}(1-\Delta y) \\ &+ \mathbf{V}rep_{xyz,1k}\,\Delta y \quad \forall k \in \{1,2\} \end{aligned} \tag{11}$$

$$\mathbf{V}rep_{xyz} = \mathbf{V}rep_{xyz,0}(1-\Delta z) + \mathbf{V}rep_{xyz,1}\,\Delta z \tag{12}$$

The end product is a uniformly smooth repulsive velocity field vector.

III. MANIPULATOR KINEMATICS

A. INVERSE KINEMATIC CONTROL

The desired movement of the end-effector is achieved by using inverse kinematics velocity control scheme with task prioritisation.

$$\dot{\mathbf{q}} = \mathbf{J}^{+}\xi_p \mathbf{v}_{\text{att}} + \dot{\mathbf{q}}_{rep} \tag{13}$$

The damped Moore-Penrose pseudo-inverse, $\mathbf{J}^{+} = \mathbf{J}'(\mathbf{J}\mathbf{J}' + \sigma_{ee}\mathbf{I})^{-1}$, is utilized to mitigate singularity issues and improve numerical stability in inverse kinematics computations. ξ_p is the primary task execution slowdown constant. Finally $\dot{\mathbf{q}}_{rep}$ are the weighted sum of avoidance joint velocities, each transformed into the null space of primary velocities.

B. END-EFFECTOR VELOCITY

Prioritizing control over the end-effector's (EE) translational velocity is essential for a consistent and stable target approach. The commonly used method, where translational velocity is proportional to squared distance, leads to impractically high initial velocities, which can prevent obstacle avoidance and real-world execution, followed by disproportionately slow velocities when nearing the goal position.

$$\mathbf{v} = \frac{\mathbf{x}_{EE} - \mathbf{x}_g}{||\mathbf{x}_{EE} - \mathbf{x}_g||} \times \frac{\arctan(k_{sigm}\,||\mathbf{x}_{EE} - \mathbf{x}_g||)}{\pi/2} \tag{14}$$

In Eq. (14), $\mathbf{v}$ specifies the end-effector's (EE) translational velocity directed towards the target, integrating both its direction and magnitude. Here, $\mathbf{x}_{EE}$ represents the EE's current position, and $\mathbf{x}_g$ is the target position, both in Cartesian coordinates. The calculation $\frac{\mathbf{x}_{EE} - \mathbf{x}_g}{||\mathbf{x}_{EE} - \mathbf{x}_g||}$ generates a unit vector pointing towards the target, ensuring targeted movement. The velocity's modulation by the arctangent sigmoid function curtails overshooting by moderating speed in proximity to the target and saturating velocity at a predefined upper limit when distant. The parameter k_{sigm} in the arctangent function adjusts the curve's steepness, affecting how quickly the end-effector decelerates near the target. A higher k_{sigm} maintains speed until closer to the target for a sharp deceleration, while a lower value starts slowing down earlier for a gradual approach.

The current orientation of the end-effector (EE) and its goal orientation are encoded through rotation matrices, $\mathbf{cR}$ for the current state and $\mathbf{gR}$ for the goal state. The necessary rotational adjustment is identified through the relative rotation matrix $\mathbf{dR}$, which is transformed into a quaternion representation, where we than calculate the rotational error using method presented in the article [17].

To get the full velocity of the end effector (EE), $\mathbf{v}_{\text{att}}$, translational and rotational components are multiplied using proportional gains and combined.

$$\mathbf{v}_{\text{att}} = \begin{bmatrix} k_v \cdot \mathbf{v} \\ k_\omega \cdot \omega \end{bmatrix} \tag{15}$$

Utilizing the mapping into the null space of the primary velocity, it becomes evident that at high velocities of the primary task, there may be an insufficient number of degrees of freedom remaining for the secondary task to safely navigate around obstacles.

To mitigate the dominance of the primary task over the secondary task, a mechanism for the deceleration of the primary task's execution has been implemented. This strategy effectively decreases the velocity of the manipulator towards its primary objective, thereby allocating additional maneuverability for secondary task.

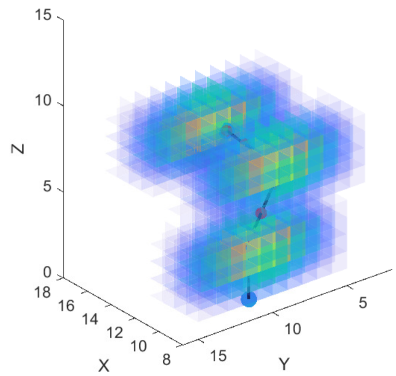

Fig. 2. Visualization of only Y repulsive direction kernels (X,Z not shown), for 3 seperate POI on the manipulator. The colours of the kernels display their absolute values. We can see, that the magnitude of the kernel values falls when approaching kernel edge.

$$\xi_p = \frac{1}{1 + \kappa_{\text{sec}}||v_{\text{Rmax}}||} \tag{16}$$

The deceleration factor (eq. (16)) is determined by the constant κ_{sec} and the magnitude of the maximum repulsive velocity vectors, which is contingent upon the robot's minimal distance from an obstacle v_{Rmax}. According to the equation, a small v_{Rmax} minimizes the deceleration effect, permitting the uninterrupted execution of the primary task. Conversely, large v_{Rmax} boosts the deceleration by reducing the ξ_p factor, thereby affording the secondary task increased opportunity for obstacles avoidance motion.

C. AVOIDANCE VELOCITIES

To enable the avoidance of obstacles by the manipulator within its operating environment, the manipulator is uniformly coated with virtual detection points (POIs). The density of these virtual points is dependent upon the dimensions of matrices. Given a matrix width w perpendicular to the primary direction, it is advantageous for these points to be spaced at intervals of $\Delta = \frac{w}{3}$. This spacing ensures partial overlap between matrices, thereby achieving a uniform distribution of repulsive vectors across the entirety of the robot.

During kinematic optimization, repulsive velocities for each point are computed following the methodology described in the previous section (sect. II). This is achieved by mapping each point into the task space, where the calculation of repulsive velocity is performed. The cartesian mapping for each point comprises a kinematic transformation from the robot's base to the beginning of the segment where the point is located, followed by an additional mapping to the specific point on the segment (eq. (17)). For each point, three components of cartesian velocity are obtained.

$$\mathbf{T}_{0\rightarrow\text{POI}} = \mathbf{T}_{0\rightarrow 1} \cdot \mathbf{T}_{1\rightarrow 2} \cdot \ldots \cdot \mathbf{T}_{(j-1)\rightarrow j} \cdot \mathbf{T}_{j\rightarrow\text{POI}} \tag{17}$$

Upon obtaining the repulsive velocity for each point, the second norm of each velocity is calculated. For a number of K points where the avoidance velocities are greatest, indicative of closest proximity to an obstacle, the resulting joint velocities that facilitate obstacle avoidance are computed using inverse kinematic equations for the selected points on the manipulator and transformed into the null-space velocities of the primary task (eq. (18)).

$$\mathbf{J}_{d_i}^{+} = \mathbf{N}\mathbf{J}_{d_i}'(\mathbf{J}_{d_i}\mathbf{N}\mathbf{J}_{d_i}' + \sigma_{rep}\mathbf{I})^{-1}, \quad \text{for } i = 1, 2, \ldots, K; \tag{18}$$

The translational velocity Jacobian is calculated for each of the K selected points on the robot. Velocity space can be restricted to just one dimension, on the velocity component that directs movement away from obstacles. Consequently, the Jacobian, correlating joint space velocities q with the directional velocity d_i, is represented as $\mathbf{J}_{d_i} = \mathbf{n}_i^T \mathbf{J}_i$. The normal vector, signifying the desired avoidance velocity, is essentially the unit velocity vector, derived from our matrices as $\mathbf{n}_i = \frac{\mathbf{v}_{poii}}{||\mathbf{v}_{poii}||}$. This approach streamlines computational processes by avoiding the necessity for complex matrix inversions, transforming the Jacobian from a matrix to a vector of dimensions $1 \times n$, where n signifies the count of joints preceding the POI, thus facilitating more straightforward calculations of repulsive velocities.

$$\dot{\mathbf{q}}_{rep} = k_r \sum_{i=1}^{K} \alpha_i \mathbf{J}_{d_i}^{+} \left(v_i - \mathbf{J}_{d_i}\mathbf{J}^{+}\xi_p \mathbf{v}_{att} \right) \tag{19}$$

The overall avoidance joint velocities are obtained through a weighted sum of the avoidance joint velocities for the selected K nearest points, taking into account the influence of the primary task of manipulator EE movement on the joint velocities.

IV. SIMULATION RESULTS

A. Repulsive Field Visualization

We introduce the resultant repulsive field on a two-dimensional map. While analogous principles apply to three-dimensional spaces, visualization of more dimensions is complex. Utilizing our repulsive potential field velocity calculation method, points were sampled across the entire obstacle map as depicted in Figure 3. The prescribed algorithm was executed at each sampled point to derive the repulsive fields along the X and Y axes, which were subsequently visualized as a composite vector field.

B. Manipulator Examples

The operation of the potential field is demonstrated on two different manipulator cases. The resolution of our voxel grid for obstacles is $R = 10$ cm. The use of interpolation allows us to have a relatively coarse voxel grid, which reduces the memory demand of the space grid and accelerates the computation. We selected $K = 7$ POI to observe the distance from obstacles and are uniformly distributed them along the segments and joints, starting from the second joint of the robot to the tip of the manipulator. This distribution

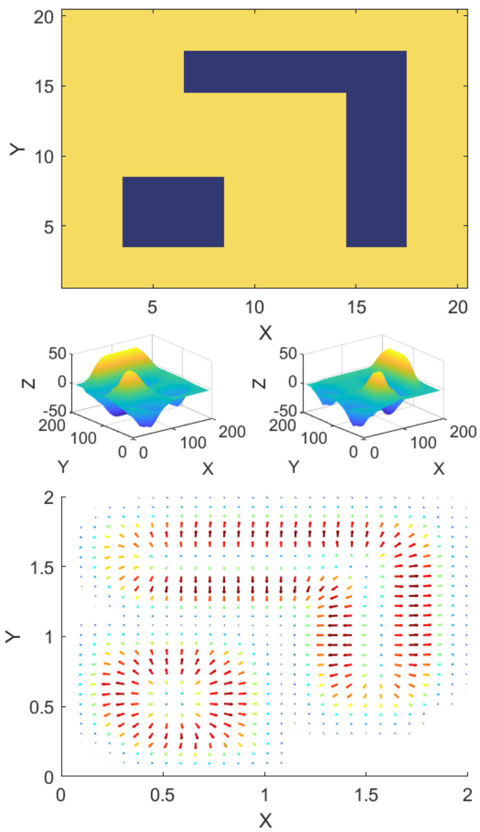

Fig. 3. Visualization of the potential field calculated for the entire map using the proposed approach. Top: Obstacle distribution via a 2D heatmap. Center: Induced repulsive fields along X (left) and Y (right) axes. Bottom: Composite vector field showcasing the resultant repulsive velocities for obstacle avoidance.

begins from the second joint because it is from this point onwards that the manipulator has the capability to avoid obstacles. POIs are indicated by dots on the manipulator in the figures. Near obstacles, vectors emanate from these points, depicting the calculated repulsive velocities at the locations. Throughout the simulation, Euler integration of the calculated joint velocities is performed with a step size of $T_{step} = 0.1$ s.

In the first case (Fig. 4, 5), the manipulator safely 'curls' or selects a path to a point located on the other side of a column, avoiding the obstacle with the potential field calculated by the proposed method. The constants chosen for the primary task are $k_v = 5$ and $k_w = 1.5$, for the avoidance task $k_r = 20$. The weighting constants for the individual POIs are $\alpha = \frac{[\frac{3}{9}, \frac{2}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}, \frac{1}{9}]}{10}$, where the biggest weight belongs to the point on the manipulator which is closest to the obstacle and so on. As the most direct path for the end-effector to the target passes straight through the column, an approach that would result in a collision, we implement a reduction of the primary speed in the vicinity of the obstacle, setting $\xi_p = 1$.

In the second scenario (Fig. 6, 7), our robot operates within

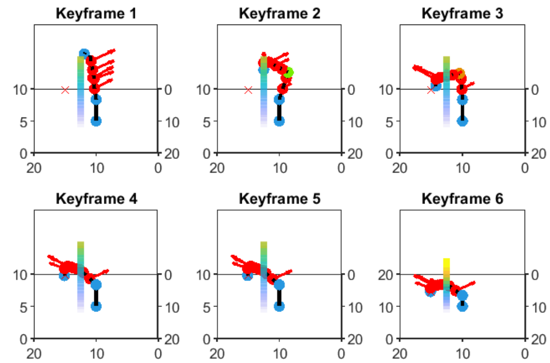

Fig. 4. Sequential keyframes demonstrating the manipulator's path planning and column obstacle avoidance strategy.

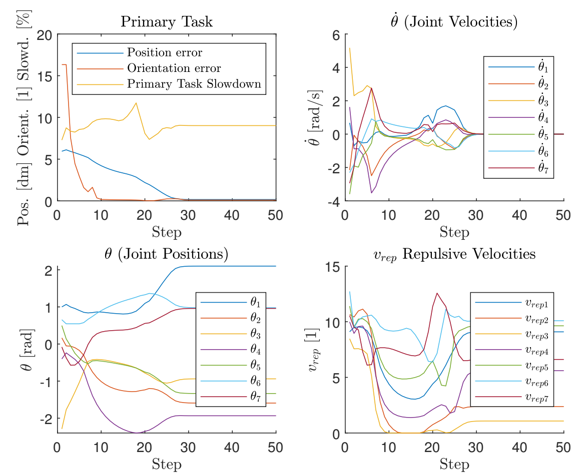

Fig. 5. Visualization of the manipulator's path around the column obstacle: (a) Primary Task errors and percentage of the primary speed after applying slowdown, (b) Joint Velocities $\dot{\theta}$, (c) Joint Positions θ, and (d) Norms of Repulsive Velocities v_{rep}.

a dynamic environment. The proposed method for calculating repulsive velocities proves to be a suitable choice when a local optimization approach is needed to accommodate dynamic changes. In this scenario, a ball moves as governed by the equation $y_{ball} = 0.4 + (1.4 - 0.4)\frac{N_{step}}{75}$, with $x_{ball} = 1.25$ and $z_{ball} = 0.7$. The primary task is now to maintain the robot's end-effector (EE) at a fixed point, disregarding EE orientation to enhance obstacle avoidance capability. This increases maneuverability, thereby eliminating the need for primary speed task slowdown ($\xi_p = 0$). Constants selected for the primary task are $k_v = 2$ and $k_w = 0$, and for the avoidance task $k_r = 3$. Weighting constants for the individual POIs are $\alpha = \frac{[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}]}{10}$.

V. CONCLUSION

This study introduces an innovative methodology for computing repulsive velocities to facilitate obstacle avoidance in the motion planning of robotic manipulators. It employs discretized voxel grids to model environmental obstacles and utilizes matrix multiplication for calculating repulsive field

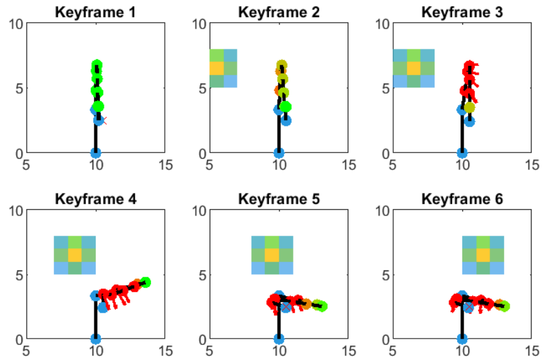

Fig. 6. Dynamic obstacle avoidance scenario: Sequential keyframes illustrating the robot's maneuvering in response to a progressively moving ball from left to right across the workspace.

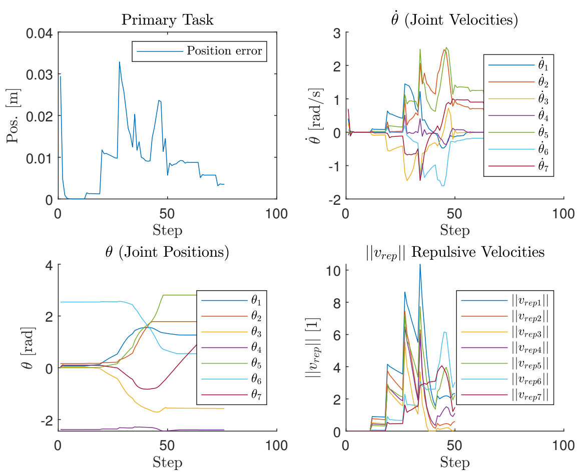

Fig. 7. Performance in a dynamic obstacle avoidance scenario: (a) Primary Task error (b) Joint Velocities $\dot{\theta}$, (c) Joint Positions θ, and (d) Norms of Repulsive Velocities v_{rep}.

values at voxel centers. Tri-linear interpolation is applied to achieve a continuous and smooth velocity field between voxels. The approach is further integrated with a modified inverse kinematics optimization strategy, which concurrently manages the end-effector's goal attainment and obstacle circumvention. The method's efficiency is heightened by the permissible largeness of voxel dimensions, which accelerates the computation of avoidance velocities and allows for a coarser grid resolution. The technique's efficacy and applicability for local path planning and dynamic environment adjustments were validated through simulation.

Future work will focus on refining the repulsive field generation by encapsulating the manipulator within a singular kernel, rather than relying on separate directional kernels at points of interest. Given the complexity of balancing multiple repulsive velocities, a learning-based approach is proposed for developing a unified controller for primary tasks and obstacle avoidance. Additionally, the exploration of a Model Predictive Control (MPC) strategy is anticipated, aiming to extend the manipulator's motion planning capabilities beyond immediate step considerations.

References

[1] M. G. Tamizi, M. Yaghoubi, and H. Najjaran, "A review of recent trend in motion planning of industrial robots," Int J Intell Robot Appl, vol. 7, no. 2, Art. no. 2, Jun. 2023. doi: 10.1007/s41315-023-00274-2.

[2] L. Žlajpah and T. Petrič, "Obstacle Avoidance for Redundant Manipulators as Control Problem," in Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization, S. Kucuk, Ed. InTech, 2012. doi: 10.5772/32651.

[3] Y. Nakamura, H. Hanafusa, and T. Yoshikawa, "Task-Priority Based Redundancy Control of Robot Manipulators," The International Journal of Robotics Research, vol. 6, no. 2, pp. 3–15, Jun. 1987. doi: 10.1177/027836498700600201.

[4] E. A. Basso and K. Y. Pettersen, "Task-Priority Control of Redundant Robotic Systems using Control Lyapunov and Control Barrier Function based Quadratic Programs," in IFAC-PapersOnLine, vol. 53, no. 2, pp. 9037–9044, 2020. doi: 10.1016/j.ifacol.2020.12.2024.

[5] H. Toshani and M. Farrokhi, "Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: A Lyapunov-based approach," Robotics and Autonomous Systems, vol. 62, no. 6, pp. 766–781, Jun. 2014. doi: 10.1016/j.robot.2014.02.005.

[6] J. Haviland and P. Corke, "NEO: A Novel Expeditious Optimisation Algorithm for Reactive Motion Control of Manipulators," IEEE Robot. Autom. Lett., vol. 6, no. 2, Art. no. 2, Apr. 2021, doi: 10.1109/LRA.2021.3056060.

[7] J. Amiryan and M. Jamzad, "Adaptive motion planning with artificial potential fields using a prior path," in 2015 3rd RSI International Conference on Robotics and Mechatronics (ICROM), Tehran, Iran, 2015, pp. 731-736, doi: 10.1109/ICRoM.2015.7367873. Keywords: Planning; Force; Motion segmentation; Vehicles; Path planning; Mobile robots; Artificial Potential Fields; Autonomous vehicle; Evolutionary Algorithms; Random Sampling; Reactive Motion Planning.

[8] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," in 1985 IEEE International Conference on Robotics and Automation Proceedings, Mar. 1985, pp. 500–505. doi: 10.1109/ROBOT.1985.1087247.

[9] G. Klančar, A. Zdešar, and M. Krishnan, "Robot Navigation Based on Potential Field and Gradient Obtained by Bilinear Interpolation and a Grid-Based Search," Sensors, vol. 22, no. 9, art. no. 3295, pp. 1-24, 2022. doi: 10.3390/s22093295.

[10] X. Xia et al., "Path Planning for Obstacle Avoidance of Robot Arm Based on Improved Potential Field Method," Sensors, vol. 23, no. 7, Art. no. 7, Apr. 2023. doi: 10.3390/s23073754.

[11] S.-O. Park, M. C. Lee, and J. Kim, "Trajectory Planning with Collision Avoidance for Redundant Robots Using Jacobian and Artificial Potential Field-based Real-time Inverse Kinematics," Int. J. Control Autom. Syst., vol. 18, no. 8, Art. no. 8, Aug. 2020, doi: 10.1007/s12555-019-0076-7.

[12] J. Baumgartner, T. Petrič, and G. Klančar, "Potential Field Control of a Redundant Nonholonomic Mobile Manipulator with Corridor-Constrained Base Motion," Machines, vol. 11, no. 2, art. no. 293, pp. 1-28, 2023. doi: 10.3390/machines11020293.

[13] H. Oleynikova, Z. Taylor, M. Fehr, R. Siegwart, and J. Nieto, "Voxblox: Incremental 3D Euclidean Signed Distance Fields for on-board MAV planning," in Proc. of the 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vancouver, BC, Sep. 2017, pp. 1366–1373. doi: 10.1109/IROS.2017.8202315.

[14] L. Han, F. Gao, B. Zhou, and S. Shen, "FIESTA: Fast Incremental Euclidean Distance Fields for Online Motion Planning of Aerial Robots," arXiv, Jul. 26, 2019. Accessed: Jan. 11, 2024. [Online]. Available: http://arxiv.org/abs/1903.02144

[15] X. Zhou, Z. Wang, H. Ye, C. Xu, and F. Gao, "EGO-Planner: An ESDF-Free Gradient-Based Local Planner for Quadrotors," IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 478–485, Apr. 2021, doi: 10.1109/LRA.2020.3047728.

[16] D. Han, H. Nie, J. Chen, and M. Chen, "Dynamic obstacle avoidance for manipulators using distance calculation and discrete detection," Robotics and Computer-Integrated Manufacturing, vol. 49, pp. 98–104, Feb. 2018, doi: 10.1016/j.rcim.2017.05.013.

[17] L. Žlajpah, "On orientation control of functional redundant robots," in Proc. of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, May 2017, pp. 2475–2482. doi: 10.1109/ICRA.2017.7989288.