Utilizing Convolution for Obstacle Avoidance during Robotic Manipulator Movement

Jakob Baumgartner[1], **Gregor Klančar**[2]

[1]*Fakulteta za Elektrotehniko, Tržaška cesta 25, 1000 Ljubljana*
E-pošta: jakob.baumgartner@fe.uni-lj.si

Abstract

This study proposes a method for real-time obstacle avoidance in robotic arm control, leveraging a 3D voxel grid and 3D directional kernels. The approach integrates the benefits of convolutional techniques with the spatial understanding provided by voxel grid representation to efficiently calculate movement directions while evading obstacles. By focusing on kinematics and obstacle grid representation, the method efficiently utilizes the versatility of a 7-degree-of-freedom (DOF) robot arm, employing geometric and differential kinematics for movement calculations. The strategy doesn't require exhaustive training as with neural networks, making it deployable in various environments.

1 Introduction

Robotic manipulators, specifically those with a high degree of freedom (DOF), possess a crucial characteristic known as kinematic redundancy [1]. Kinematic redundancy refers to having more joints (i.e., degrees of freedom - DOF) than required to perform a task, thus allowing the robot to adopt multiple configurations to accomplish the same end-effector position and orientation. This redundancy provides a considerable advantage, making it possible for manipulators to optimize movement according to different criteria, such as energy efficiency, obstacle avoidance, or joint limit avoidance.

In this research, we specifically consider a 7-DOF Franka Panda Emika [2] manipulator. When controlling both the position and orientation of the end-effector in a 3D space, which requires six degrees of freedom (three for position and three for orientation), a 7-DOF manipulator such as the Panda Emika has one redundant degree. However, when controlling only the position, which requires three degrees of freedom, the manipulator possesses four redundant degrees. This redundancy grants the manipulator an essential characteristic: an infinite number of possible joint configurations to reach a specific end-effector (EE) point in space.

When dealing with redundant manipulators, task prioritization is crucial. In our approach, we distinguish between primary and secondary tasks. The primary task of the manipulator is usually defined as reaching a desired EE position and orientation or as following a trajectory made out of such points. In this work we instead use APF (Artificial Potential Field), to guide the EE towards the goal position, while avoiding the obstacles in the path of the EE. Operating in a real-world environment often requires considering secondary tasks, such as real-time obstacle avoidance, to ensure safe and uninterrupted operation of the manipulator.

This paper proposes a novel method for real-time obstacle avoidance of the manipulator arm. We use an obstacle grid combined with convolution techniques to assign avoidance directions to each manipulator segment, ensuring efficient and safe maneuvering within complex environments.

2 Background and Related Work

The execution of the secondary task of obstacle avoidance necessitates an understanding of the spatial relationship between the manipulator and surrounding obstacles, usually quantified as a distance or interpreted as a repulsive force. By conceptualizing obstacles as sources of repulsive forces, the manipulator can be guided away from potential collisions, akin to how two magnets of the same polarity repel each other.

check that repulsive - avoidance is well changed (READ TEXT)

Two commonly employed methods for obstacle avoidance include artificial potential field methods (APF) [3] and artificial neural networks. In potential field methods, the robot and obstacles are treated as charged entities. The robot is attracted to the target (goal) while being repelled by obstacles, resulting in a potential field. The robot navigates this field, moving along the path of steepest descent. While this method is intuitive and relatively easy to implement, it can sometimes result in local minima problems where the robot gets stuck in a position that is not the target but cannot find a path due to the repelling obstacles.

Artificial neural networks (ANNs) are another prevalent approach, offering potential solutions to the local minima problem. By learning to map sensor readings to appropriate actions, ANNs can be trained to successfully navigate complex environments. However, they require extensive training and can be computically intensive.

An advanced version of ANNs in the context of 3D

obstacle detection and avoidance is the use of 3D Convolutional Neural Networks (3D-CNNs) [4]. While not strictly a path planning method, 3D-CNNs can be trained to recognize obstacles and free space in 3D occupancy data. The convolutional layers in the network can learn spatial hierarchies and structures, and the final layers can output a direction or action to take. These methods, although effective, require extensive training data and computational resources, however it is capable to recognizing and avoiding some local minima.

The Vector Field Histogram [5] (VFH) method presents another approach. This method uses a 2D histogram grid as a spatial representation of the environment. The histogram cells are updated based on the proximity and angle of obstacles. Then a primary polar histogram is calculated from the grid cells, leading to a candidate valley for navigation. While originally developed for 2D, this method was later extended to 3D as VFH+ [6].

Our proposed approach uses a 3D occupancy or voxel grid and 3D directional kernels, akin to 2D Sobel filters, to perform kernel convolution and calculate the direction in which to move the robot arm to avoid obstacles. This method marries the benefits of convolutional approaches with the intuitive spatial reasoning provided by a voxel grid representation.

The primary strength of the proposed method is its ability to swiftly calculate directional guidance for the robot arm in real-time. By using a voxel grid, we can efficiently represent the 3D environment, and the use of 3D directional kernels allows for robust calculations of obstacle proximity and directions of free space. This is beneficial in dynamic environments where quick responses are required. Moreover, this approach does not require extensive training as with neural networks, making it easier to deploy in new environments.

3 Methodology

3.1 Kinematics

The manipulation capabilities of a robot are fundamentally determined by its kinematics, the mathematical description of motion without considering the forces that cause it. In our experiments, we employed a 7 degree-of-freedom (DOF) robot. The seven joints provide a high level of versatility, enabling the robot to move and orient its end-effector in a broad range of positions and directions. To calculate the movements and positions of the robot, we use both geometric and differential kinematics.

Geometric kinematics (also known as forward and inverse kinematics) focuses on the relationship between joint angles and the position and orientation of the end-effector. Forward kinematics involves calculating the position and orientation of the end-effector given the joint angles, while inverse kinematics is the process of determining the joint angles for a desired end-effector position and orientation. In the case or redundant manipulator, there isn't an unique mapping from EE coordinates to the joint coordinates as there exist an infinite number of different joint angles configurations that map to the same EE position.

Differential kinematics, also known as the robots Jacobian, forms the foundation of robot velocity analysis. The Jacobian matrix, often represented as J, encapsulates the relationship between the joint velocities and the end-effector linear and angular velocities. It's derived from the partial derivatives of the forward kinematics.

The Jacobian matrix is a key element in the control of robotic manipulators. It can also be used for computing the inverse differential kinematics problem. However, for a 7-DOF manipulator, the Jacobian matrix is not square and its inverse does not exist in a traditional sense. In such cases, we resort to the pseudo-inverse of the Jacobian, calculated using the Moore-Penrose method (eq. 1). The Moore-Penrose pseudo-inverse, denoted as J^{+}, provides an approximation of the inverse for non-square matrices.

Damping, represented by the constant term k_{damp} in the Moore-Penrose pseudo-inverse equation, is used to introduce stability and prevent potential numerical issues when computing the pseudo-inverse of the Jacobian for non-square matrices. It acts as a regularization term to ensure a well-conditioned solution.

$$J^{\dagger} = (J^T J + k_{damp}\, I)^{-1} J^T \tag{1}$$

The null space (eq. 2) of the Jacobian corresponds to those joint motions which produce no movement in the end-effector. The primary task for our robot is the desired end-effector movement. The secondary task of obstacle avoidance in our research takes advantage of this null-space motion [7]. Our method applies an avoidance velocity from detected obstacles in the environment. We calculate this velocities at certain points of interest (POI) that are positioned on the manipulator arm segments. We transform avoidance velocities into the joint space velocities using $J^{\dagger}_{poi}$ jacobian calculated in the specific POI. These velocities we than transform into the null-space using matrix N. The robot arm is thus able to carry out the primary task while simultaneously avoiding obstacles by appropriately utilizing its redundant DOF (eq. 3).

$$N = (I - J^{\dagger}(q)J(q)) \tag{2}$$

$$\dot{q} = J^{\dagger}(q)\dot{x}_{ee} + N J^{\dagger}_{poi}\dot{x}_{poi} + N\dot{q}_{mid} \tag{3}$$

We integrated a supplementary secondary task aiming to maintain the robot manipulator's joints as close to the midpoint of their range (between $q(i)_{max}$ and $q(i)_{min}$ for the i-th joint) as possible (eq. 4). This auxiliary task, designed to elevate the robot's "elbow", reinforced the robot's upright posture during the execution of the primary task and the obstacle avoidance task.

$$\dot{q}_{mid}(i) = w_m \left(\frac{q(i)_{max} - q(i)_{min}}{2} - q(i) \right) \tag{4}$$

3.2 Obstacle Grid

In order to safely and efficiently navigate a robot through a complex environment, it is essential to have an appropriate representation of the robots work space. In

our approach, we achieve 3D spatial understanding of the robot's surrounding environment by employing an obstacles grid, a method that transforms the environment into a three-dimensional grid of voxels (volumetric pixels).

Each voxel within this grid corresponds to a specific volume in real-world space and contains information about the occupancy of that volume. This provides a structured and spatially efficient representation of the 3D environment, including the locations and extents of any obstacles present. This 3D voxel grid can be built from sensor data, such as that obtained from a LiDAR, 3D camera, or depth sensor. Each reading from the sensor provides a point in 3D space, and these points are used to update the occupancy probability of the corresponding voxels in the grid.

3.3 Avoidance velocities

In our approach to robot arm obstacle avoidance, we apply a method of kernel convolution in conjunction with geometric transformations. This methodology allows us to calculate the necessary avoidance velocities for specific points of interest located on the manipulator.

The process commences with kinematic transformations, which facilitate the mapping of points of interest from the robot's configuration space to the Cartesian space, followed by their projection onto the 3D voxel grid.

Once the mapping of these points to the voxel grid is completed, we proceed to employ a specialized kernel convolution method. This method is tailored to calculate the constituent parts of the avoidance velocity vector in the Cartesian space, specifically for the x, y, and z directions. For each direction, we generate the corresponding kernel and extract a segment of the obstacle grid of matching size, centered at the point of interest, resulting in two comparable 3D matrices—one from the obstacle grid A_d and the other representing the kernel K_d.

By employing the Hadamard (element-wise) product (eq. 5) between the cutout segment of the obstacle grid A_d and the corresponding 3D convolutional kernel K_d for direction d, we derive the resultant matrix C_d, which can be expressed as:

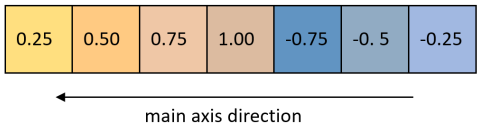
$$C_d = A_d \odot K_d \tag{5}$$

Subsequently, the avoidance velocity in the Cartesian coordinate system is obtained by summing all the values in the resulting matrix (eq. 6), which can be represented as:

$$\dot{x}_{poi} = \sum_i \sum_j \sum_k c_{dijk} \tag{6}$$

3.4 Convolution Kernel

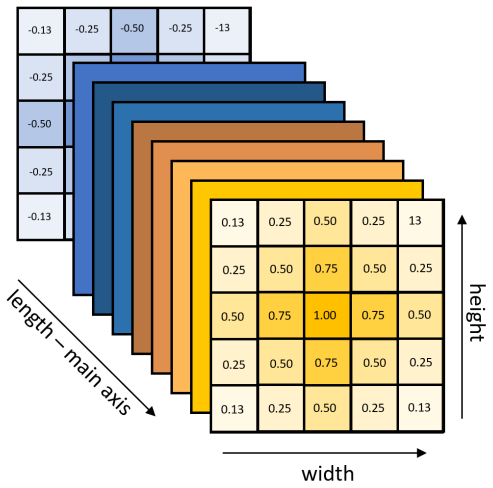
Our convolutional kernels are essentially discrete differential operators in 3D space, and their design reflects this. These kernels bear a resemblance to the Sobel kernels, which are widely used in computer vision for edge detection and approximation of spatial derivatives. Each kernel has a primary axis aligned with the direction in which the avoidance velocity is being calculated, with the other two axes orthogonal to it. The kernel values along the primary axis exhibit a symmetric pattern (Figure 1): the highest value lies in the center of the kernel, with the absolute values decreasing uniformly on either side, with negative polarity on one side. This decrease simulates the derivative-like behavior of the kernel.



Slika 1: The kernel values along the primary axis exhibit a pattern of symmetrical distribution with negative polarity on one side.

Contrary to the design of a classical Sobel operator, our custom kernels don't contain zeroes along the primary axis. Instead, we assign the maximum kernel values along this line. This choice is specifically designed to meet our goal of obstacle avoidance. By doing so, when our points of interest on the robot manipulator coincide with an obstacle - effectively being 'right on top' of the obstacle in the voxel grid - the convolution operation yields a maximal avoidance velocity. This encourages the robot to move away from the obstacle and ensures safer navigation in the presence of potential collisions.

However, the kernel values are not merely mirror images on each side of the kernel. While the absolute values are mirrored, the signed values are positive on one side and negative on the other. This sign difference is critical for calculating directional gradients.



Slika 2: In most instances, an equal width and height around the primary axis are desired, creating a symmetrical detection area.

In addition to the kernel values along the primary axis, the values along the orthogonal axes mimic the same pattern (Figure 2). These are derived from the primary axis values, decrease proportionally with the radial distance from the primary axis, and crucially, they maintain the same sign as dictated by the derivative along the primary axis. This symmetry in both magnitude and sign around

the main axis is a fundamental characteristic of our kernels.

By applying these specialized kernels to the voxel grid at the locations of the points of interest, we compute the gradients that guide the robot arm's movements. This approach enables the arm to continue its primary task while dynamically avoiding any potential collisions with obstacles.

The dimensions of the kernel, both along the primary axis and orthogonal to it, play crucial roles in determining the effective detection range and the nature of the avoidance velocity.

The length of the kernel along the detection axis is dependent on the desired detection range for potential obstacles. Longer kernels can detect obstacles further away from the robot, essentially extending the 'safety zone' around the robot. Additionally, the rate at which the absolute values decrease when moving away from the center of the kernel along the detection axis determines the avoidance velocity profile as we move away from the obstacle. We have experimented with both linear and Gaussian decay profiles. The Gaussian profile, with its bell-curve distribution, generates an avoidance velocity that decreases exponentially as we move away from the obstacle, creating a softer transition compared to the linear profile.

Orthogonal to the primary axis, the size of the kernel dictates the width of the detection area. In most instances, an equal width and height around the primary axis are desired, creating a symmetrical detection area. Narrow kernels detect a focused, 'beam-like' area directly ahead, while wider radial kernels can detect obstacles not exactly aligned with the primary axis but still within the potential path of the robot.

To optimize obstacle detection, we have explored the idea of reducing the kernel values orthogonal to the primary axis with increasing distance from it. The intent here is to minimize the impact of distant, off-axis obstacles while still ensuring they are detected. However, as the ideal kernel configuration depends on the grid resolution and the specific environment, further testing is required to refine this approach.

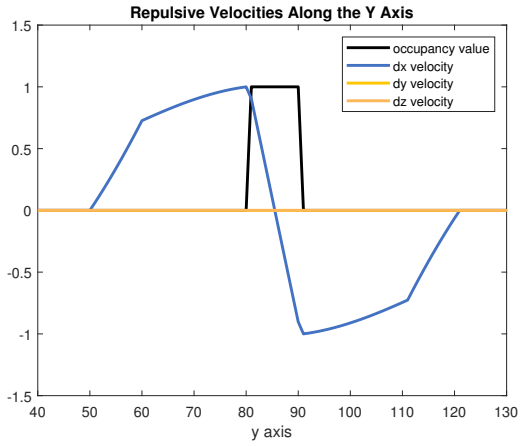
3.5 Attractive Field

In our end-effector motion planning approach, we leverage derivatives of the attractive field to facilitate efficient trajectory generation. Prior to commencing motion planning, we construct a distance matrix, analogous in size to the matrix representing the obstacles. This distance matrix encompasses distance values from our designated goal point, corresponding to each voxel in the workspace. Subsequently, we employ interpolation techniques to estimate the value at the current end-effector (EE) point by considering the distance field values of the eight neighboring voxels and the distances of the current EE pose relative to these eight values. By performing interpolation between two points, we effectively calculate the numerical derivative of distance change, thereby obtaining the EE velocity. This approach enables us to navigate the robotic manipulator smoothly towards the desired goal while considering the surrounding obstacle space.

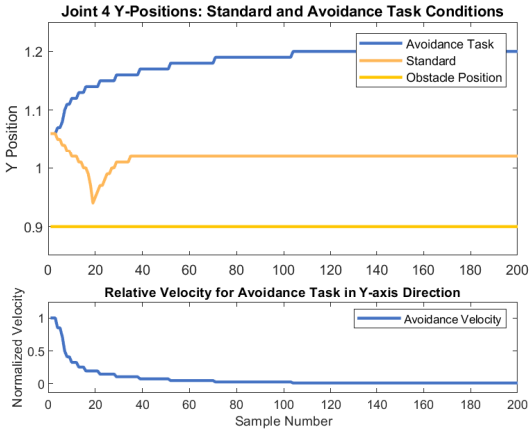
Additionally, we can augment our EE velocity with obstacles avoidance component, which is calculated using the convolutional kernel technique. This enables us to guide the end-effector around obstacles smoothly, ensuring safe and collision-free navigation while dynamically adapting to the environment. By integrating both the attractive and avoidance field contributions, our motion planning approach ensures efficient and effective trajectory generation, enabling the robotic manipulator to reach its desired goal while avoiding potential obstacles in the workspace.

- attractive field calculating equation

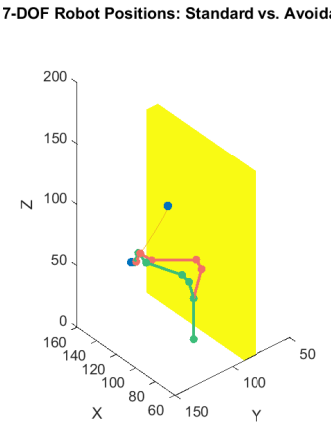
4 Experiment and Results



Slika 3: Plot depicting avoidance velocities (x, y, z) as calculated by our kernel along the y-axis, showing the robot's adaptive movement away from the wall obstacle.



Slika 4: Comparative visualization of the y-position of the robot's fourth joint with and without the secondary avoidance task, highlighting the impact of the avoidance velocity component.



Slika 5: 3D model displaying the final position of the robot, contrasting the scenarios with and without the implementation of joint 4 obstacle avoidance.

Our proposed algorithm was tested using a kinematic model of the 7 Degrees of Freedom (DoF) Panda Emika robot, we implemented within the MATLAB environment. This robot was selected for its operational versatility, thus serving to underscore the robustness and flexibility of our approach.

The end-effector's motion was controlled using the attractive field method, serving as the primary task. This was complemented by the avoidance field method implemented as a secondary task, with the specific aim of maneuvering a point on the fourth joint of the robot as far away from the obstacle - in this case, a simple wall - as possible.

To compute the joint positions, we used joint velocities derived through inverse kinematics, incorporating the aforementioned secondary tasks. We then applied numerical integration to these velocities to model the robot's motion in a simulated environment.

Figure 3 illustrates the avoidance velocities in the x, y, and z directions, computed by our kernel, when its center aligns with various points along the y-axis. Here, the y-coordinates are presented on the x-axis, and the corresponding velocities are plotted on the y-axis. As expected, the y-velocity is the only component exhibiting non-zero values. It effectively propels the robot away from the wall, moving in either a positive or negative direction, depending on the relative position to the wall. Conversely, the x and z velocities remain zero, reflecting the absence of obstacles along these axes. This aligns with our expectations given the presence of the wall obstacle is only along the y direction.

In Figure 4, we observe the variation in the y-position of the fourth joint, both with and without the implementation of the secondary avoidance task. The corresponding plot beneath the image illustrates the diminishing avoidance velocity as the robot moves further away from the wall obstacle. The rate and function of this decrement can be modulated by altering the dimensions and values within the kernel.

Figure 5 presents a 3D model depicting the final position of the robot, showcasing both scenarios—with the implementation of the joint 4 obstacle avoidance (illustrated in green) and without it (depicted in red).

The figure also highlights how the application of the avoidance algorithm prevents the fourth joint from moving closer to the wall, as compared to when the avoidance algorithm is not employed. The position of the joint, upon task completion, is noted to be approximately 20 cm further away from the wall.

Although these preliminary results demonstrate the efficacy of our proposed approach, further testing of the proposed method is needed to analyse its adaptability in more complex operational environments.

5 Conclusions and further development

This paper presented a novel approach to robotic motion planning, combining the application of kernel convolution for obstacle avoidance velocity calculations with an integration of attractive and avoidance velocities components for end-effector trajectory generation and robot joint obstacle avoidance. The method was validated using the kinematic model of a 7 DoF Panda Emika robot, demonstrating its operational versatility and adaptability in varying workspace conditions.

Notable advantages of our approach include its simplicity and the fact that we work directly in the obstacle space grid, negating the necessity for calculating the spatial primitives of shapes. Moreover, when implemented in parallel, our method exhibits high computational efficiency, making it a viable solution for real-time applications. The experiments conducted effectively showcased that the end-effector could move efficiently towards the target while successfully avoiding obstacles, specifically, a wall in this instance.

Our results affirmed the robustness of the proposed approach, as evidenced by the robot's fourth joint being approximately 20 cm further from the wall upon task completion when the avoidance algorithm was employed.

Despite its effectiveness, the approach could potentially fall into local minima traps. This limitation could be addressed by incorporating multiple convolutional kernels, though this would increase complexity and resemble convolutional neural networks.

Additionally, we acknowledge that our study was conducted under simplified conditions with a single type of obstacle. As such, future work should focus on validating the proposed method in more complex environments, with different obstacle types, shapes, and densities, to further ascertain its adaptability and robustness. It may also be beneficial to explore the integration of other optimization strategies and machine learning techniques to further enhance the overall efficiency and effectiveness of the trajectory planning process.

add a note about picking up multiple POI

add note that this is just a part of algorithm development

Literatura

[1] Nguyen, Charles C., Zhou, Zhen-Lei, Mosier, Gary E. (1991) Analysis and control of a kinematically redundant manipulator. *Computers & Electrical Engineering*, 17(3), pp. 147–161. doi:10.1016/0045-7906(91)90031-T.

[2] Franka Emika. (2021). Panda: Specification. *Available at: https://www.franka.de/panda*

[3] Khatib, O. (1985) Real-time obstacle avoidance for manipulators and mobile robots. In: *1985 IEEE International Conference on Robotics and Automation Proceedings*, 2, pp. 500–505. doi:10.1109/ROBOT.1985.1087247.

[4] M. Duguleana, F. G. Barbuceanu, A. Teirelbar, and G. Mogan, "Obstacle avoidance of redundant manipulators using neural networks based reinforcement learning,"*Robotics and Computer-Integrated Manufacturing*, DOI: 10.1016/j.rcim.2011.07.004, Sep. 2011.

[5] J. Borenstein and Y. Koren, "The vector field histogram-fast obstacle avoidance for mobile robots,"*IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278–288, Jun. 1991. DOI: 10.1109/70.88137

[6] I. Ulrich and J. Borenstein, "VFH+: reliable obstacle avoidance for fast mobile robots,"In *Proceedings. 1998 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1572–1577, May 1998. DOI: 10.1109/ROBOT.1998.677362

[7] *Obstacle Avoidance for Redundant Manipulators as Control Problem*, Leon Lajpah and Tadej Petri, In: *Serial and Parallel Robot Manipulators - Kinematics, Dynamics, Control and Optimization*, edited by Serdar Kucuk, InTech, March 2012, ISBN: 978-953-51-0437-7, DOI: 10.5772/32651, Language: English.