3D Voxel Grid Based Path Planning for Robotic Manipulators using Matrix Multiplication Technique

Alternatives:
- Efficient distance calculation / Efficient repulsive field calculation technique
- Matrix Multiplication-Driven Repulsive Fields for 3D Voxel-Based Robotic Manipulator Path Planning
- Robotic Manipulator Path Planning Optimization Using Matrix-Derived Repulsive Fields Based on 3D Voxel Grid

Jakob Baumgartner, Gregor Klančar

November 15, 2023

Abstract

1 Introduction

2 Background

1-2 PAGES

- PRESENT DISTANCE CALCULATION FOR MANIPULATORS (sensors, lidar, ir, bounding boxes)
- PRESENT PATH PLANNING METHODS FOR MANIPULATOR (optimization, sampling, biological, learning)
- similar to Distance Transform (a kind of inverted distance transform)
- method was inspired by Khatib APF (however, it evolved into a different method)
- different existing APF manipulator implementation articles
- VFH

3 Methodology

3 PAGES

3.1 Optimization Algorithm

- optimization algorithm
- robot kinematics (include the exact-reduced method)
- primary task of distance goal
- secondary task of repulsive field
- damped least squares
- task slowdown option
- secondary task of manipulation measure

3.2 Task Constraints

-equations of occupied and empty space

3.3 Attractive Velocity

-equation references
- citations
- check if quat log equation is correct

Our method ~~innovatively~~ employs inverse kinematics approach (IK) to guide the end effector (EE) towards its target, marking a departure from Khatib's joint coordinates approach in favor of a Cartesian coordinates framework. This is particularly beneficial in scenarios involving redundant manipulators, where determining an optimal goal joint configuration in advance is challenging.

When calculating translational velocity, we avoid the conventional gradient of the squared distance approach, which leads to high initial velocities and subsequently slow speeds near the target. Our aim is a consistent velocity throughout the trajectory, with controlled deceleration near the goal. This is achieved by first calculating the unit vector towards the target for direction, then modulating its magnitude using a sigmoid function, specifically the arctangent function, to prevent overshooting and ensure stable approaching motion.

$$\vec{v} = \frac{\vec{x}_{EE} - \vec{x}_g}{||\vec{x}_{EE} - \vec{x}_g||} \times \frac{\arctan(k_{sigm}\ ||\vec{x}_{EE} - \vec{x}_g||)}{pi/2} \tag{1}$$

In the above equation (eq. 1), $\vec{v}$ represents the end effector's translational velocity towards the target, combining direction and magnitude. The terms $\vec{x}_{EE}$ and $\vec{x}_g$ denote the current and goal positions of the EE, respectively, in Cartesian coordinates. The unit vector calculation, $\frac{\vec{x}_{EE} - \vec{x}_g}{||\vec{x}_{EE} - \vec{x}_g||}$, ensures motion directed towards the target. Finally, the sigmoid function, particularly the arctangent component, modulates this velocity to avoid overshooting, balancing speed and precision. The constant k_{sigm} allows us to set how close to the goal does the robot EE start slowing down.

The rotational velocity error of the EE is needed for ensuring goal orientation of the EE. In our approach, orientations are depicted using rotation matrices. Specifically, R represents the current EE orientation, while gR signifies the goal

EE orientation. The disparity between these orientations is encapsulated by the relative rotation matrix dR. This matrix is formulated by multiplying the goal orientation matrix gR with the transpose of the current orientation matrix cR^T. To ensure that it represents a pure rotation without any scaling we than normalize the so gotten matrix .

$$dR = \frac{gR \cdot cR^T}{||gR \cdot cR^T||} \tag{2}$$

The relative rotation matrix value is converted into a quaternion, which is then logarithmically transformed to represent the rotational error vectorially:

$$dR \mapsto dQ = a + b\,i + c\,j + d\,k \tag{3}$$

$$dQl = 2 \cdot \log(dQ) = al + bl\,i + cl\,j + dl\,k \tag{4}$$

The components of this quaternion, excluding the real part, form the rotational error vector ω:

$$\vec{\omega} = \begin{bmatrix} bl \\ cl \\ dl \end{bmatrix} \tag{5}$$

To get the full velocity of the end effector (EE), velocityEE, we combines translational and rotational velocities, which we scale using proportional gains k_p and k_r.

$$\vec{\text{velocity}}\text{EE} = \begin{bmatrix} k_p \times \vec{v} \\ k_\omega \times \vec{\omega} \end{bmatrix} \tag{6}$$

3.4 Repulsive Velocity

- object detection is done in task domain and not c-space (more logical)
- repulsive field calculation - matrix "convolution" method
- matrix size and shape selection
- equation for repulsive kernel values (non-linear)
- PLOT: (ERK) kernel graphics
- PLOT: (ERK) linear kernel graphics
- PLOT: kernel field shape
- interpolation of the repulsive field
- what if there are obstacles behind wall (usually not the case, depth sensors show only thin walls, some noise doesnt matter, non-linear kernel, possible additional pre-convolution to convert obstacle grid to edges)
- effecient calculation in dinamic environments, lacking prediction capabilities (MPC)

3 PAGES

- include execution times
- PLOT: kernel on robot graphics

6 Discussion

1 PAGE

- add the limitations of such method (already mentioned by Khatib)
- the limitations of local search
- number of parameters that need to be tuned (are there actually that many?)

7 Conclusion