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 14, 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

-orientation error
- p = v w
- describe constants - shorten

In the development of our method, a pivotal element is the integration of a velocity profile that effectively guides the end effector (EE) toward its designated target position and orientation. Our methodology represents a deviation from the paradigm established by Khatib. Instead of quantifying the primary task's error within the joint coordinate framework, we opt for a Cartesian coordinate representation. This aspect is important when the manipulator exhibits redundancy, characterized by multiple or potentially infinite configurations that satisfy the primary task's requirements and constraints. In such scenarios, it can becomes impractical, if not impossible, to predetermine the optimal goal joint configuration. Utilizing solely the gradient of the squared distance would result in a high initial velocity when distant from the target, followed by a progressively decreasing velocity upon nearing the target. Such a velocity profile is suboptimal, as the excessive initial velocity could potentially compromise lower priority tasks, such as obstacle avoidance and in leads to unnecessary slow speeds when approaching the target. Our objective is to maintain a consistent velocity profile for the majority of the trajectory, with a gradual deceleration as the goal is approached to avoid overshooting. To this end, we initially compute the unit vector of the distance from the target, yielding a normalized velocity in the required direction. However, this would lead to overshooting and instability near the target. To counteract this, we modulate the magnitude of this vector using a sigmoid function, specifically the atan function. This modulation ensures a constant movement for the majority of the time towards the target, with the sigmoid function effectively reducing the velocity in inverse relation when in near proximity to the target. The following equation encapsulates our approach:

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

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)

3.4.1 Kernel Selection

3.4.2 Interpolation

4 Implementation

5 Results

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