# An Approach to n-Dimensional Path Planning with Potential Functions

**OZGUR GULSUNA**, 2307668
Middle East Technical University, Ankara, TR

***Abstract***

**This study explores the intricacies of robot motion planning in higher dimensions, deliberately excluding dimensions below two for depth. Detailing the robot's interaction with an n-dimensional environment and complex obstacles, the proposed Potential Function Approach employs conic and quadratic functions for attraction and dynamic adjustments for repulsion. Solver selection integrates ODE45 for accuracy and a discrete iterative solver for flexibility. Implementation intricacies, including obstacle representation through bounding n-spheres, address computational trade-offs. Experimentation ranges from 2D scenarios to 3D, seeking insights into effective motion planning within a MATLAB simulation environment. The study offers a concise exploration of path planning challenges in multi-dimensional spaces.**

***Index Terms*—Motion Planning, Potential Functions, High Dimensions, Hypersphere, MATLAB, Simulation.**



## I. Introduction

This exploration delves into the challenges of robot motion planning in higher dimensions, deliberately focusing on dimensions greater than two for a nuanced investigation. It details the robot as a single-point entity within an n-dimensional environment, navigating static obstacles of varying complexities. The deliberate absence of repulsive systems from soft borders adds complexity to the robot's interaction with its environment, progressing from simple points to intricate high-dimensional geometric configurations.

The utilized approach introduces additive attractive and repulsive potentials, with conic and quadratic functions for attraction and repulsion based on obstacle proximity. Employing Gradient Descent for goal-reaching, the implementation integrates both ODE45 for accuracy and a discrete iterative solver for flexibility in solver selection, unraveling advantages and trade-offs across dimensions.

Implementation include obstacle representation through bounding n-spheres, accuracy trade-offs between exact and approximate representations. Experimentation spans 2D scenarios to eliminate local minima challenges and extends to 3D, visually representing potential functions in complex environments. The study seeks insights into effective robot motion planning in intricate spaces, offering a comprehensive exploration within a MATLAB simulation environment, aiming to deepen understanding of path planning challenges in multi-dimensional spaces.

## II. Problem Definition

The implementation begins with a precise definition of the robot, environment, and obstacles. These definitions are constructed to allow for mathematical concepts to be generalized seamlessly to higher spatial dimensions, requiring little to no modification in observations and interactions with the environment.

This work specifically engages with dimensions higher than two. This choice is deliberate, as navigating one-dimensional environments and obstacles presents a balance of being straightforward and complex simultaneously. Moreover, the notion of an "obstacle" itself poses a challenge to finding a "suitable path" solution in a single dimension. However, there exist a distinct research-area in the domain of list-type path planning tasks for deformable structures, as outlined in the literature [1]. Given these considerations, our primary focus is on higher dimensions, and dimensions lower than two are intentionally excluded from our current analysis.

### A. The Robot, Environment & Obstacles

The robot is as a **single** point entity that can navigate freely in any direction across all dimensions. Describing the position of the robot involves an n-dimensional vector,

expressed as $q_{robot} = \{q_1, q_2, ..., q_n\}$. In the process, the planning algorithm plans the path forehand, without necessitating active sensors. The robot can ideally depend on the preplanned path knowing its position in the environment. In achieving this, a basic odometry sensor proves to be a sufficient means for localization.

The environment is conceptualized as an n-dimensional free space, surrounded by soft borders that are only assessed when determining whether the robot is within or outside those boundaries. Importantly, these soft borders lack a defined repulsive system, thus exerting no influence on the robot. Both the starting point and the goal point, denoted as $q_{start}$ and $q_{goal}$ respectively, are single point entities within this n-dimensional environment. To be more precise, their coordinates can be represented as $q_{start} = \{s_1, s_2, ..., s_n\}$ and $q_{goal} = \{g_1, g_2, ..., g_n\}$.

The environment has static obstacles of different sizes placed at various locations. It's important to delve more into high-dimensional obstacles situated in more complex spaces. These obstacles, existing in higher dimensions, are intricate geometric structures. They can be defined in diverse ways, incorporating different regions like inside and outside. To develop a comprehensive understanding, we'll begin by exploring obstacles in simpler, lower dimensions and gradually build knowledge about the types of obstacles that can exist in more complex, higher dimensions.

The simplest obstacle is a point in free space, applicable across any dimension. Similarly, a line can be considered an obstacle, although in environments with more than two dimensions, it can be traversed with ease. Moving on to polygons, they are composed of points connected by line segments, with a defined inside and outside. In two dimensions, passing through a polygon can be challenging, but in three dimensions, it becomes more manageable. This pattern persists as we increase dimensions. Notably, polygons can exhibit varying complexities, having different topologies like convex or concave, even when constructed from the same set of points. This combination of entities becomes more pronounced and intricate in high-dimensional obstacles, presenting additional challenges on obstacle definitions as the dimensionality increases.

As the dimensionality increases, the perception of obstacles needs to evolve to handle the complexities introduced, like navigating the inside/outside of a 4D tesseract. To cope with these challenges, it becomes essential to simplify the representation of obstacles.

# III. Potential Function Approach

## A. Additive Attractive/Repulsive Potentials

The combination of attractive and repulsive potentials forms the simplest potential function with a straightforward concept: the goal point attracts the robot, while each obstacle repels it. The summation of these effects results in a potential field that guides the robot toward the goal.

$$U(q) = U_{att}(q) + U_{rep}(q)$$

To reach the goal, the robot should traverse from high potential to low potential. This involves defining a change in potential with respect to the dimensions, represented by the gradient. Following the negative gradient, a strategy known as *gradient descent*, leads the robot toward the goal.

The potential function approach offers significant advantages when dealing with high-dimensional problems. The potentials can be easily structured for any number of dimensions, making it a versatile approach. In the subsequent sections, all equations will be presented in their most generic form to accommodate various dimensionalities.

### 1. The Attractive Potential

There are several options available for defining an appropriate attractive potential for the goal. One such option is the conic potential, but it introduces a discontinuity at the point of origin. Alternatively, a quadratic potential can be utilized. This potential grows quadratically as the object approaches, providing a continuously differentiable function. Since it relies solely on distance definition, it can be applied seamlessly across any number of dimensions.

$$U_{att}(q) = \frac{1}{2}\zeta d^2(q, q_{goal}) = \frac{1}{2}\zeta (q - q_{goal})^\top (q - q_{goal})$$
$$\nabla U_{att}(q) = \zeta (q - q_{goal})$$

The norm squared, defined as the transpose multiplication of vectors, is utilized for calculating the potential that will be used to visualize the environment, to better understanding of the system's dynamics.

While the quadratic potential grows significantly for large distances, imposing a heavier computational burden, it becomes necessary to cap it. The transition is then made to the conic potential after reaching a specified distance, denoted as $d^*_{goal}$ distance. This transition forms a new combined field, which can be defined as:

$$U_{att}(q) = \begin{cases} \frac{1}{2\zeta} d^2(q_{\text{goal}}), & \text{if } d(q) \leq d^*_{\text{goal}} \\ d^*_{\text{goal}} \zeta d(q_{\text{goal}}) - \frac{1}{2}\zeta \left(d^*_{\text{goal}}\right)^2, & \text{if } d(q) > d^*_{\text{goal}} \end{cases}$$

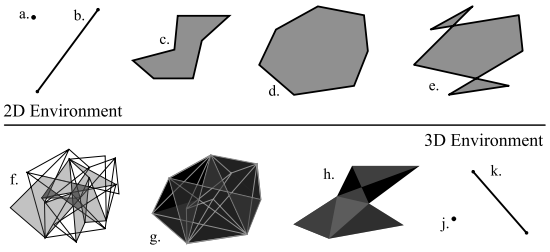



Fig. 1: **Different obstacle structures that can exist in 2D / 3D**
a) 2D point, b) 2D line, c) 2D convex polygon, d) 2D concave polygon, e) 2D convex polygon with same points, f) 3D projection of a 4D mesh, g) 3D mesh, h) 3D polygon, j) 3D point, k) 3D line.

### 2. *The Repulsive Potential*

A repulsive potential used to keep the robot at a distance from obstacles. The intensity of the repulsive force is contingent on the robot's proximity to an obstacle. The closer the robot is to an obstacle, the stronger the repulsive force should be. As a result, the repulsive gradient operates exclusively within a domain near the obstacle. The repulsive potential is commonly defined in terms of the distance to the nearest obstacle, denoted as $D(q)$. The repulsive potential and its gradient is explained in further detail in [2].

$$U_{\text{rep}}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{D(q)} - \frac{1}{Q^*})^2, & \text{if } D(q) \leq Q^*, \\ 0, & \text{if } D(q) > Q^*. \end{cases}$$

### 3. *Gradient Descent*

Gradient descent is a commonly used approach for searching solutions to generic optimization problems. The concept is straightforward: starting from the initial configuration, take a small step in the direction opposite to the gradient. This results in a new configuration, and the process iterates until the gradient becomes zero. Formally, a gradient descent algorithm can be defined in Algorithm. 1.

## IV. Implementation

In the MATLAB-based simulation environment, the implementation begins by creating a navigable space with uniformly expanded bounds. However, the obstacles are created manually. This introduces challenges, particularly with higher-order obstacles that lack direct physical representation and there is no dimension increase that is interconnected to the robot's changing configuration space. This disconnect poses difficulties in generating meaningful obstacles for algorithm testing. Addressing this challenge is crucial to experiment on high and low dimensional structures where real cases may be applicable to in the future. Obstacles that are consistent are essential to thoroughly test the path planning algorithm hence an approximation is to be made both helping in construction of obstacles and calculations in later steps.

**Algorithm 1** Gradient Descent

**Input:** Gradient $\nabla U(q)$at a point $q$
**Output:** A sequence of points $\{q(0), q(1), ..., q(i)\}$

$q(0) = q_{start}$
$i = 0$
**while** $\nabla U(q(i)) \neq 0$ **do**
  $q(i+1) = q(i) + \alpha(i)\nabla U(q(i))$
  $i = i + 1$
**end while**

### A. Obstacle Approximation

Beginning with planar environments, the introduction of polygons as obstacles are plausible and logical. Constructing these polygons is relatively straightforward, allowing for flexible placement and sizing that aligns with the requirements of the test environments. The incorporation of polygonal obstacles serves a dual purpose — not only do they facilitate the comparison of different approximations, but they also pose a challenging scenario for evaluating the effectiveness of the potential function approach in terms of path-finding.

In three-dimensional environments, utilizing a mesh as an obstacle is a suitable choice, although constructing these meshes can be a bit intricate. One approach is to generate primitive shapes, or they can be crafted using proprietary software, presenting a challenging yet manageable task.

However, when dealing with dimensions higher than three, there is a lack of tools and established procedures for construction. In such cases, using points on the obstacles becomes a viable alternative. N-dimensional point data known to define an obstacle can be employed to generate a high-dimensional structure that serves as an approximation.

The hyper-sphere approach proves valuable in this context. By utilizing an n-dimensional sphere consistent with the dimension of the environment, the creation of hyper-obstacles becomes relatively simple. Given a collection of points from the obstacle in any dimension, the diameter of the hyper-obstacle can be determined by the distance between the two most distant points, with the center of the hyper-obstacle located at the midpoint of these two extremities. Obstacle approximation is done once at the beginning, and the approximate map is retained for ongoing analysis during the simulation.

- **Exact Obstacles** : The exact obstacles/polygons is applicable strictly for 2D environments. In such cases, gradients of potentials can be calculated by considering planar distances from the robot's position to the edges of the obstacles. However, in higher dimensions, the edges transition into surfaces and volumes. This complicates distance calculations from the obstacle "surface" to the robot, posing challenges in determining accurate potentials.
- **Approximate Obstacles** : The approximate obstacles involve utilizing only the point data from the actual obstacles. This involves constructing "bounding n-spheres," ensuring that all points lie within the smallest hyper-sphere. This approximation is used in dimensions from three to N. In 2D environments, the approximations simplify to circles for the obstacles, offering a practical comparison setting for the exact and approximate obstacle presentations.

### B. Solver Selection

The computation of gradients has been previously explained, and the algorithmic components are straightforward. The path calculation from the gradients involves

two approaches. One approach is solving the differential equations more accurately using proprietary solvers, while the other approach involves discrete methods by iteratively taking small steps over the gradients. The implementation incorporates both approaches, allowing for experimentation with various aspects of these different calculation methods. This dual approach enables a comprehensive exploration of the trade-offs and advantages associated with accurate differential equation solvers and iterative discrete methods in the context of path planning. Moreover both of the algorithms are applicable to n-dimensions.

- **ODE45**: A continuous solver that is proven to provide highly accurate and smooth path solutions. It has an adaptive step size, it efficiently computes precise results, making it particularly effective for scenarios where precision is crucial, that is moving through a tight passage.
- **DISCRETE**: An iterative solver, easily implementable and allowing for the adjustment of step sizes, provides flexibility. However, it may lead to instabilities, including oscillations around minima without achieving convergence. Careful consideration is necessary.

The experimentation will include both approaches, comparing their efficacy across dimensions ranging from two to n.

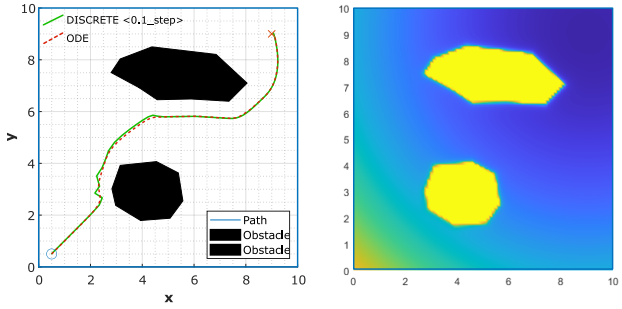



Fig. 2: Path from "ODE45" and "DISCRETE" approaches on the left and potential function shown on the right.

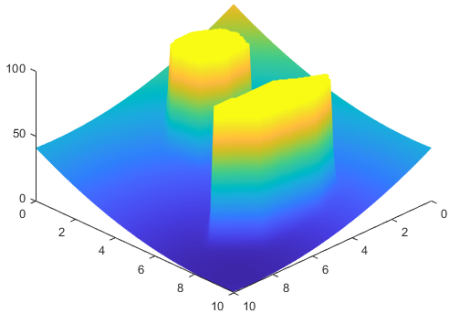


Fig. 3: 3D visualization of the potential function for the simple case.

## V. Experimentation

### A. 2D Experiments

The simple case serves as a distinct test to compare the discrete and ODE solutions. With uncomplicated obstacles, the path is easily found in both scenarios as the Fig. (2) laying out. The step size for the DISCRETE option was set at 0.1, leading to a jittery motion along the path. The resulting path also exhibits a similarity to the Greedy Bug and Bug-2 algorithms, wherein the robot follows the boundary until it can directly sense the direction of the goal. Concave obstacles were deliberately chosen to minimize the risk of encountering local minima. The upcoming test will focus specifically on scenarios designed to further eliminate the possibility of local minima.

The variables $\zeta$ and $\eta$ play a crucial role in modifying both attractive and repulsive potential fields, and their values are determined experimentally. Ensuring that adjustments to one variable do not adversely impact the other is a challenge, especially in cases where close or complex-shaped obstacles are involved. While calibration is straightforward in many instances, obstacles with intricate shapes or close proximity often require numerous tweaks to strike the right balance.

In the second test, the starting point and goal remain unchanged, but the obstacle approximation is modified. The 2-sphere approximation now encapsulates the entire obstacles, including the available paths around them. However, a notable issue arises as the new obstacles are positioned closer to each other, creating a local minima point in between. Consequently, the algorithm struggles to find a viable path.

This example highlights a couple of problems with the approximation. Notably, thin and long obstacles are approximated to circles, consuming a significant portion of the free space and affecting the algorithm's ability to navigate through complex configurations.

In the upcoming test, specific convex polygons are introduced, designed to act like cups that can trap the robot along its path to the goal. If these obstacles are strategically placed in the available path, it becomes challenging for the robot to escape from the resulting local minima. One potential solution could involve increasing the repulsive

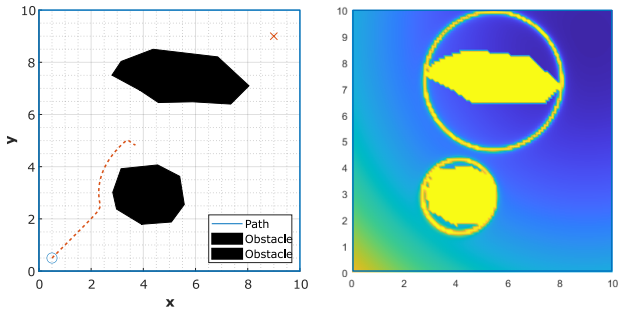



Fig. 4: Simple case with obstacles are approximated, the solution cannot be found.

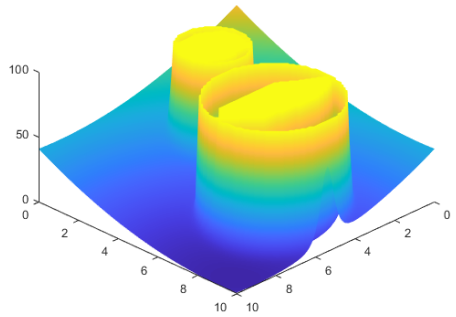


Fig. 5: Potential field for the simple case with obstacle approximation.

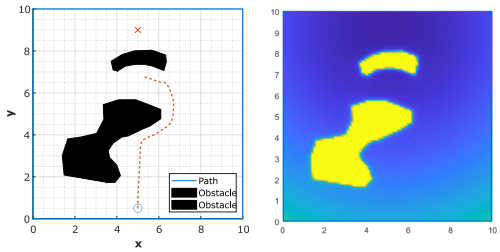



Fig. 6: Cup case with exact obstacles, the path cannot be found.

potential. However, this approach poses a dilemma, as increasing repulsion may cause the obstacle to resemble a point-like structure, necessitating a corresponding increase in the attractive potential. Striking the right balance between these potentials becomes crucial for successful path planning through environments with strategically placed convex obstacles.

The approximate hyper-sphere approach proves efficacy in transforming convex obstacles into concave hyperdimensional entities. This methodology contributes to eliminating local minima, thereby generating a more viable solution in complex environments.

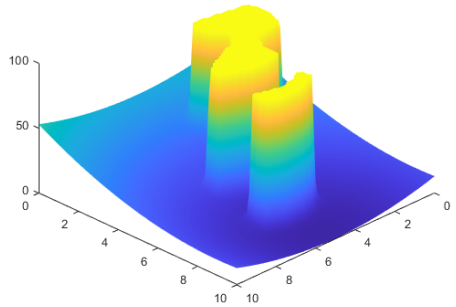


Fig. 7: Potential field representation in 3D for the Cup case.

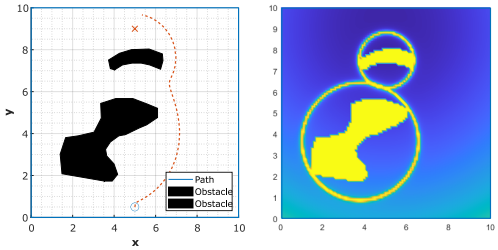



Fig. 8: Cup case with approximated obstacles, the solution is closer to the goal.

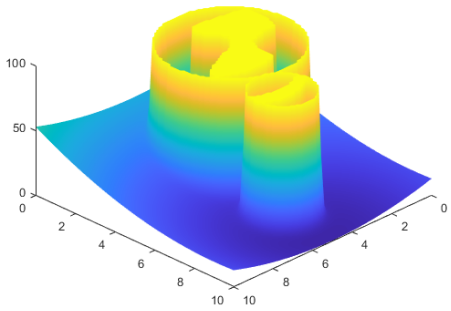


Fig. 9: The potential field plot for the approximated cup case.

## B. 3D Experiments

Displaying three-dimensional examples poses more challenges, but the path planning algorithm remains consistent. In these scenarios, the potential visualizations are volumetric, with the color intensity indicating the

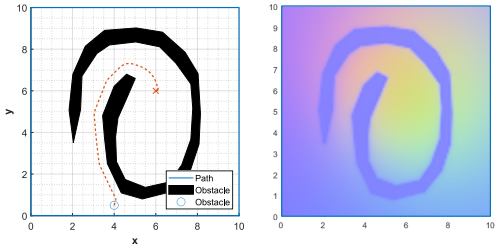
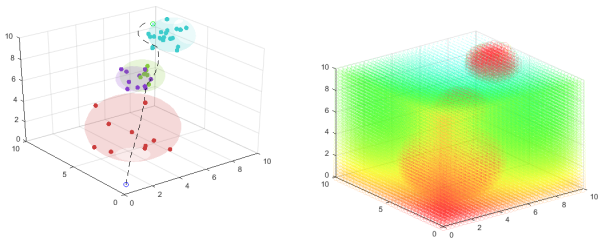
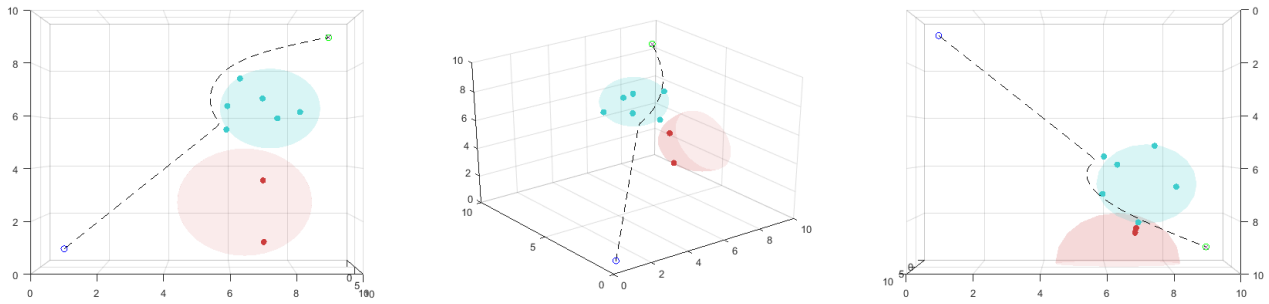



Fig. 10: Hard case as a single obstacle twirls.



(a) Three dimensional solution with obstacles are approximated and the path is found.

(b) The related potential function that is a volumetric scalar visualization.



(a) (b) (c)

Fig. 12: Three dimensional path finding using potential functions, path is found.

degree of repulsion; deeper shades of red signify higher repulsive forces, as evident in the figures.

## C. 4D Experiments

Extending the solution to four dimensions is as straightforward as adding one more dimension, although visualizations become more challenging. The figure below illustrates projections of the fourth-dimensional environment and the found path onto four distinct three-dimensional plots. Despite the complexity, the path is successfully found, and obstacles are approximated using high-dimensional spheres.

REFERENCES


[1] S. Javdani, S. Tandon, J. Tang, J. F. O'Brien, and P. Abbeel
Modeling and perception of deformable one-dimensional objects
In *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1607–1614.

[2] H. M. Choset
*Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge: MA: MIT Press, 2005.

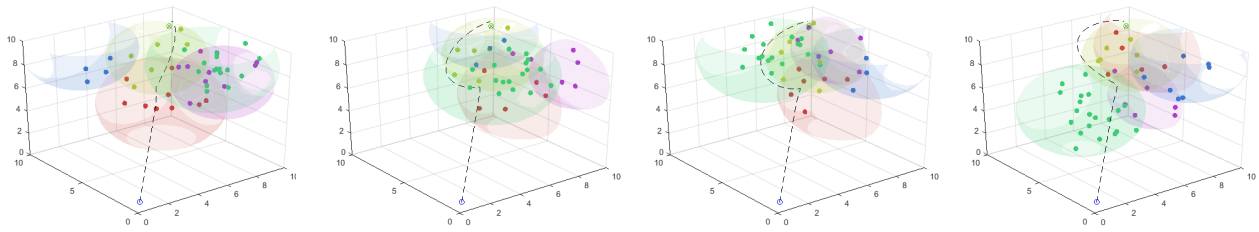


Fig. 13: Three dimensional projections of the fourth dimensional environments, the path is found consisting of a four dimensional vector. The figures above are orthogonal to each other.