Potential function

⇒ It is difficulty of explicitly representing the configuration space.

↳ An alternative is to develop search algorithms that incrementally "explore" free space while searching for a path.

⇒ A potential function is a differentiable real-valued function:

$$U : \mathbb{R}^m \to \mathbb{R}$$

⇒ The value of a potential function can be viewed as energy and hence the gradient of the potential is force.

⇒ The gradient is a vector $\nabla U(q) = DU(q)^T = [\frac{\partial U}{\partial q_1}(q), \ldots, \frac{\partial U}{\partial q_m}(q)]^T$

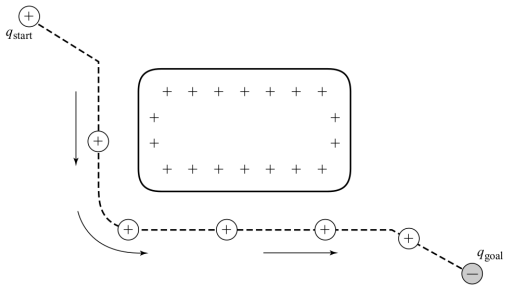
which points in the direction that locally maximally increases U.

↳ Gradient is used to define a vector field, which assigns a vector to each point on a manifold.

⇒ When U is energy, the gradient vector field has the property that work done along any closed path is zero.

⇒ The potential function approach directs a robot as if it were a particle moving in a gradient vector field.

- Robot => Positively charged
- Goal => Negatively charged
- Obstacles => Positively charged



↳ The combination of repulsive and attractive forces hopefully directs the robot from the start location to the goal location while avoiding obstacles.

⇒ Here, we mainly deal with first-order systems (i.e., we ignore dynamics)

↳ So we view the gradients as velocity vectors instead of force vectors.

⇒ The robot follows path of gradient descent.

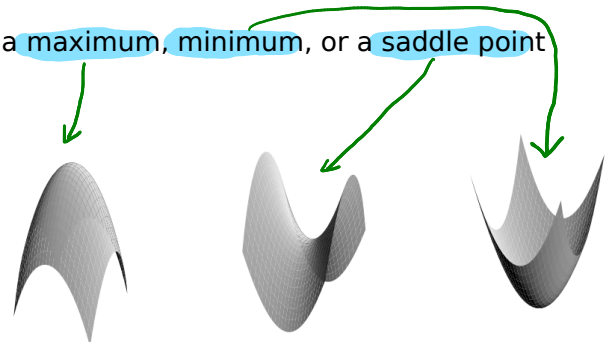
$$\dot{c}(t) = -\nabla U(c(t))$$

⇒ The robot terminates motion when it reaches a point where the gradient vanishes

$$q^* \text{ where } \nabla U(q^*) = 0$$

⇒ Such a point q^∗ is called a critical point of U .

↳ The point q^∗ is either a maximum, minimum, or a saddle point



⇒ One can look at the second derivative to determine the type of critical point.

- For real-valued functions, this second derivative is the Hessian matrix.

$$\begin{bmatrix} \frac{\partial^2 U}{\partial q_1^2} & \cdots & \frac{\partial^2 U}{\partial q_1 \partial q_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 U}{\partial q_1 \partial q_n} & \cdots & \frac{\partial^2 U}{\partial q_n^2} \end{bmatrix}$$

- When the Hessian is positive-definite, the critical point is a local minimum
- When the Hessian is negative-definite, then the critical point is a local maximum.
- When the Hessian is nonsingular the critical point is non-degenerate, implying that the critical point is isolated.

⇒ For gradient descent methods, we do not have to compute the Hessian because the robot generically terminates its motion at a local minimum, not at a local maximum or a saddle point.

- Since gradient descent decreases U, the robot cannot arrive at a local maximum, unless of course the robot starts at a maximum.
  - It is very unlikely, that the robot start at local maxima.
  - Even if the robot starts at a maximum, any perturbation of the robot position frees the robot, allowing the gradient vector field to induce motion onto the robot.
- Since we assume that the function is never flat, the set of maxima contains just isolated points, and the likelihood of starting at one is practically zero.
  - Arriving at a saddle point is also unlikely, because they are unstable as well.
- Local minima, on the other hand, are stable because after any perturbation from a minimum, gradient descent returns the robot to the minimum.

⇒ Unfortunately, all potential function suffer one problem:

- Existence of local minima not corresponding to the goal.

⇒ Two classes of approaches address this problem:

- Augments the potential field with a search-based planner.
- Defines a potential function with one local minimum, called a navigation function.

⇒ Both methods require full knowledge of the configuration space prior to the planning event.

★ Additive Attractive/Repulsive Potential

⇒ The simplest potential function in Q_free is the attractive/repulsive potential.

⇒ The potential function can be constructed as the sum of attractive and repulsive potentials.

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

The Attractive Potential

⇒ There are several criteria that the potential field U_att should satisfy:

↳ First, U_att should be monotonically increasing with distance from q_goal.

⇒ The simplest choice is the conic potential, measuring a scaled distance to the goal.

$$U(q) = \zeta d(q, q_{\text{goal}})$$

The ζ is a parameter used to scale the effect of the attractive potential.

⇒ The attractive gradient is

$$\nabla U(q) = \frac{\zeta}{d(q, q_{\text{goal}})}(q - q_{\text{goal}})$$

↳ The gradient vector points away from the goal with magnitude ζ at all points of the configuration space except the goal, where it is undefined.

⇒ When numerically implementing this method, gradient descent may have "chattering" problems since there is a discontinuity in the attractive gradient at the origin.

↳ For this reason, we would prefer a potential function that is continuously differentiable, such that the magnitude of the attractive gradient decreases as the robot approaches q_goal .

⇒ The simplest such potential function is one that grows quadratically with the distance to q_goal.

↳ $$U_{\text{att}}(q) = \frac{1}{2}\zeta d^2(q, q_{\text{goal}})$$

↳ $$\begin{aligned}\nabla U_{\text{att}}(q) &= \nabla\left(\frac{1}{2}\zeta d^2(q, q_{\text{goal}})\right) \\ &= \frac{1}{2}\zeta \nabla d^2(q, q_{\text{goal}}), \\ &= \zeta(q - q_{\text{goal}}),\end{aligned}$$

↳ It has a magnitude proportional to the distance from q to q_goal.

⇒ The gradient ∇U_att(q) converges linearly to zero as q approaches q_goal (which is a desirable property), it grows without bound as q moves away from q_goal .

↳ If q_start is far from q_goal , this may produce a desired velocity that is too large.

⇒ For this reason, we may choose to combine the quadratic and conic potentials

↳ So that the conic potential attracts the robot when it is very distant from q_goal and the quadratic potential attracts the robot when it is near q_goal

⇒ Such a field can be defined by

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

$$\nabla U_{\text{att}}(q) = \begin{cases} \zeta(q - q_{\text{goal}}), & d(q, q_{\text{goal}}) \le d^*_{\text{goal}}, \\ \frac{d^*_{\text{goal}}\zeta(q - q_{\text{goal}})}{d(q, q_{\text{goal}})}, & d(q, q_{\text{goal}}) > d^*_{\text{goal}}, \end{cases}$$

⇒ where d goal is the threshold distance from the goal where the planner switches between conic and quadratic potentials.

The repulsive potential

⇒ A repulsive potential keeps the robot away from an obstacle.

⇒ The strength of the repulsive force depends upon the robot's proximity to the an obstacle.

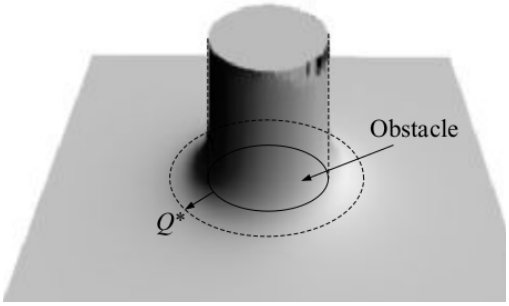
↳ The closer the robot is to an obstacle, the stronger the repulsive force should be.

⇒ Therefore the repulsive potential is usually defined in terms of distance to the closest obstacle D(q).

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

whose gradient is

$$\nabla U_{\text{rep}}(q) = \begin{cases} \eta\left(\frac{1}{Q^*} - \frac{1}{D(q)}\right)\frac{1}{D^2(q)}\nabla D(q), & D(q) \le Q^*, \\ 0, & D(q) > Q^*, \end{cases}$$



↳ where the Q^∗ ∈ R factor allows the robot to ignore obstacles sufficiently far away from it and the η can be viewed as a gain on the repulsive gradient.

↳ These scalars are usually determined by trial and error.

⇒ When numerically implementing this solution, a path may form that oscillates around points that are two-way equidistant from obstacles, i.e., points where D is nonsmooth.

⇒ To avoid these oscillations, instead of defining the repulsive potential function in terms of distance to the closest obstacle, the repulsive potential function is redefined in terms of distances to individual obstacles where d_i(q) is the distance to obstacle QO_i.

$$d_i(q) = \min_{c \in \mathcal{QO}_i} d(q, c)$$

↳ min operator returns the smallest d(q, c) for all points c in QO_i

⇒ It can be shown for convex obstacles QO_i where c is the closest point to x that the gradient of d_i(q) is

$$\boxed{\nabla d_i(q) = \frac{q - c}{d(q, c)}}$$

Now, each obstacle has its own potential function,

$$U_{\text{rep}_i}(q) = \begin{cases} \frac{1}{2}\eta\left(\frac{1}{d_i(q)} - \frac{1}{Q^*_i}\right)^2, & \text{if } d_i(q) \le Q^*_i, \\ 0, & \text{if } d_i(q) > Q^*_i, \end{cases}$$

where Q_i^* defines the size of the domain of influence for obstacle QO_i.

⇒ Then $U_{rep}(q) = \sum_{i=1}^{n} U_{rep_i}(q)$

⇒ Assuming that there are only convex obstacles or nonconvex ones can be decomposed into convex pieces, oscillations do not occur because the planner does not have radical changes in the closest point anymore.

★ Gradient Descent

⇒ Gradient descent is a well-known approach to optimization problems.

↳ Idea: Starting at the initial configuration, take a small step in the direction opposite the gradient.

↳ This gives a new configuration, and the process is repeated until the gradient is zero.

Algorithm 4 Gradient Descent

Input: A means to compute the gradient $\nabla U(q)$ at a point q

Output: A sequence of points $\{q(0), q(1), \ldots, q(i)\}$

1: $q(0) = q_{start}$
2: $i = 0$
3: **while** $\nabla U(q(i)) \neq 0$ **do**
4: $\quad q(i+1) = q(i) + \alpha(i)\nabla U(q(i))$
5: $\quad i = i + 1$
6: **end while**

⇒ The value of the scalar α(i) determines the step size at the i iteration.

⇒ It is important that α(i) be small enough that the robot is not allowed to "jump into" obstacles, while being large enough that the algorithm does not require excessive computation time.

↳ The choice for α(i) is often made on an ad hoc or empirical basis, perhaps based on the distance to the nearest obstacle or to the goal.

⇒ Finally, it is unlikely that we will ever exactly satisfy the condition ∇U(q(i)) = 0.

↳ For this reason, this condition is often replaced with the more forgiving condition $\|\nabla U(q(i))\| < \epsilon$.

↳ in which ϵ is chosen to be sufficiently small, based on the task requirements.

★ Computing distance for implementation in the plane

⇒ The attractive potential function is rather straight forward if the robot knows its current location and the goal location.

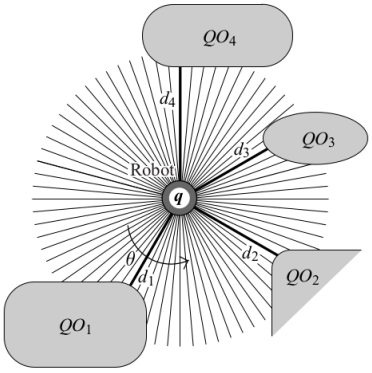
⇒ The challenge lies in computing the repulsive function because it requires calculation of distance to obstacles.

⇒ In this section, we discuss two different methods to compute distance:

↳ The first method deals with sensor-based implementation.

↳ The second method assumes the configuration space has been discretized into a grid of pixels and computes distance on the grid.

Mobile Robot Implementation



$f(q, \theta) \Rightarrow$ Will give distance to the closest obstacle along the ray from 'q' at an angle θ.

- $D(q) \Rightarrow$ Corresponds to global minim of raw distance function f.
- $d_i(q) \Rightarrow$ Corresponds to local minimum with respect to θ of f.

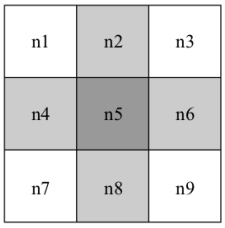
Breushfiere Algorithm (Method to Compute Distance on a Grid)

"Method for computing distance from a map representation called a grid"

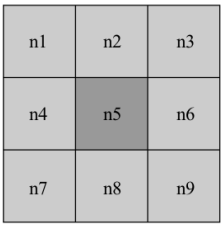
⇛ A pixel has a value of:

- Zero - If it is completely free of obstacles
- One - If it is completely or even partially occupied by an obstacle

⇛ Choice in determining the neighbor:



Four-point



Eight-point

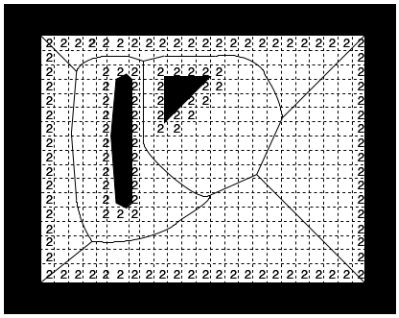
⇛ The input to the algorithm is a grid of pixels.

⇛ The output is a grid of pixels whose corresponding values each measure distance to the nearest obstacle.

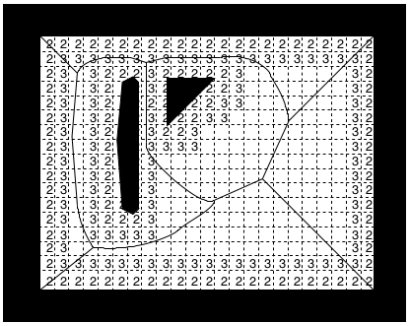
⇛ These values can then be used to compute a repulsive potential function, as well as its gradient.

Steps

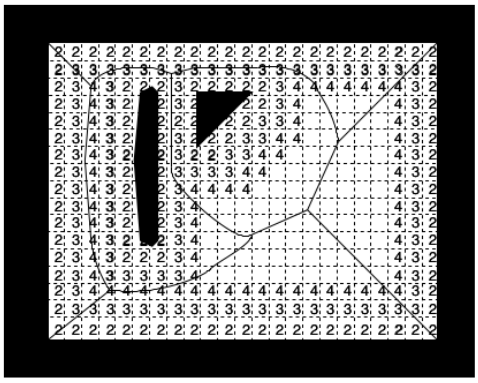
1. All zero-valued pixels neighboring one-valued pixels are labeled with a two.
   - The algorithm can use four-point or eight-point connectivity to determine adjacency.

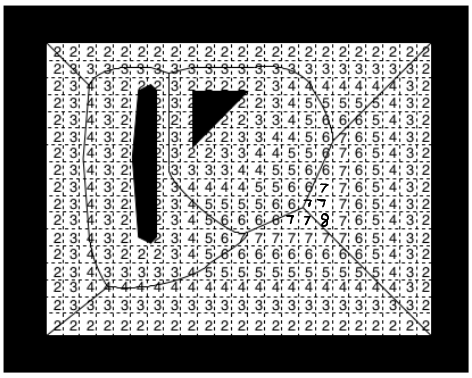


2. All zero-valued pixels adjacent to two's are labeled with a three.



3. This procedure repeats, until the fire consumes all of the free-space pixels.





⇛ The brushfire method produces a map of distance values to the nearest obstacle.

⇛ The gradient of distance at a pixel is determined by finding a neighbor with the lowest pixel value.

- The gradient is then a vector which points to this neighbor.
- Note that this vector points in either one of four or one of eight possible directions.

⇛ This method described here generalizes into higher dimensions where pixels then become volume elements.

● Local Minima Problem

⇛ The problem that plagues all gradient descent algorithms is the possible existence of local minima in the potential field.

⇛ Barraquand and Latombe developed search techniques other than gradient descent to overcome the problem of local minima present when planning with potential functions.

- Randomized Path Planner (RPP)

⇛ RPP followed the negative gradient of the specified potential function and when stuck at a local minimum, it initiated a series of random walks.

- Often the random walks allowed RPP to escape the local minimum and in that case, the negative gradient to the goal was followed again.

★ Wave-Front Planner

⇛ Simplest solution to the local minima problem, but can only be implemented in spaces that are represented as grids.

⇛ Initially, the planner starts with the standard binary grid of zeros corresponding to free space and ones to obstacles.

⇛ The planner also knows the pixel locations of the start and goal.

⇒ The goal pixel is labeled with a two.

1. In the first step, all zero-valued pixels neighboring the goal are labeled with a three.

2. Next, all zero-valued pixels adjacent to threes are labeled with four.

3. This procedure essentially grows a wave front from the goal where at each iteration, all pixels on the wave front have the same path length, measured with respect to the grid, to the goal.

4. This procedure terminates when the wave front reaches the pixel that contains the robot start location.

The planner then determines a path via gradient descent on the grid starting from the start.

Essentially, the planner determines the path one pixel at a time.

1. Assume that the value of the start pixel is 33.

2. The next pixel in the path is any neighboring pixel whose value is 32.

3. The next pixel is then one whose value is 31.

4 This procedure forms a path in the grid to the goal.

⇒ Just like the brushfire method, the wave-front planner generalizes into higher dimensions as well.

⇒ Implementation of the wavefront planner in higher dimensions becomes computationally intractable.

★ Navigation Potential function

"This applies to a limited class of configuration spaces"

DEFINITION 4.6.1 *A function* $\varphi : \mathcal{Q}_{\text{free}} \rightarrow [0, 1]$ *is called a* navigation function *if it*

- *is smooth (or at least* C^k *for* $k \geq 2$*),*
- *has a unique minimum at* q_{goal} *in the connected component of the free space that contains* q_{goal}*,*
- *is uniformly maximal on the boundary of the free space, and*
- *is Morse.*
  - ↳ A Morse function is one whose critical points are all non-degenerate.
    - This means that critical points are isolated, and if a Morse function is used for gradient descent, any random perturbation will destabilize saddles and maxima.

⇒ The navigation function approach represents obstacles as $\mathcal{QO}_i = \{q \mid \beta_i(q) \leq 0\}$

↳ In other words, β_i(q) is negative in the interior of QO_i, zero on the boundary of QO_i, and positive in the exterior of QO_i.

Sphere-Space

⇒ This approach initially assumes that the configuration space is bounded by a sphere centered at q_0 and has n dim(Q free)-dimensional spherical obstacles centered at q_1 , . . . q_n .

⇒ The obstacle distance functions are easy to define as:

* $\beta_0(q) = -d^2(q, q_0) + r_0^2,$
* $\beta_i(q) = d^2(q, q_i) - r_i^2,$

{where r_i is the radius of the sphere}

⇒ Instead of considering the distance to the closest obstacle or the distance to each individual obstacle, we consider

$$\beta(q) = \prod_{i=0}^{n} \beta_i(q)$$

⇒ This approach uses β to form a repulsive-like function.

⇒ The attractive portion of the navigation function is a power of distance to the goal

$$\gamma_\kappa(q) = (d(q, q_{\text{goal}}))^{2\kappa}$$

⇒ The function $\frac{\gamma_\kappa}{\beta}(q)$ is equal to zero only at the goal, and it goes to infinity as q approaches the boundary of any obstacle.

⇒ $\frac{\gamma_\kappa}{\beta}(q)$ has a unique minimum, but unfortunately it can have arbitrarily large values, making it difficut to compute.

⇒ Therefore, we introduce the analytical switch, which is defined as

→ $\sigma_\lambda(x) = \frac{x}{\lambda + x}, \quad \lambda > 0.$

→ σ_λ(x) is zero at x = 0

→ σ_λ(x) is one at x = inf.

⇒ we can use $\sigma_\lambda(x)$ to bound the value of the function $\frac{\gamma_\kappa}{\beta}$.

$$s(q, \lambda) = \left(\sigma_\lambda \circ \frac{\gamma_\kappa}{\beta}\right)(q) = \left(\frac{\gamma_\kappa}{\lambda\beta + \gamma_\kappa}\right)(q)$$

⇒ The function s(q, λ) has a zero value at the goal, unitary value on the boundary of any obstacle, and varies continuously in the free space.

↳ It has a unique minimum for a large enough κ.

⇒ However, it is still not necessarily a Morse function because it may have degenerate critical points.

→ So, we introduce another function that essentially sharpens s(q, λ) so its critical points become nondegenerate, i.e., so that s(q, λ) can become a Morse function.

→ This sharpening function is

$$\xi_\kappa(x) = x^{\frac{1}{\kappa}}$$

⇒ For λ = 1, the resulting navigation function on a sphere-world is then

$$\varphi(q) = \left(\xi_\kappa \circ \sigma_1 \circ \frac{\gamma_\kappa}{\beta}\right)(q) = \frac{d^2(q, q_{\text{goal}})}{[(d(q, q_{\text{goal}}))^{2\kappa} + \beta(q)]^{1/\kappa}},$$

which is guaranteed to have a single minimum at q goal for a sufficiently large κ

⇒ The drawback to this particular navigation function is that it is flat near the goal and far away from the goal, but has sharp transitions in between.

This makes implementation of a gradient descent approach quite difficult because of numerical errors.

Star-Space

⇒ The result of sphere-spaces is just the first step toward a more general planner.

⇒ A sphere-space can serve as a "model space" for any configuration space that is diffeomorphic to the sphere-space.

Once we have a navigation function for the model space, to find a navigation function for the diffeomorphic configuration space, we need only find the diffeomorphism relating the two spaces.

⇒ Here we consider star-spaces consisting of a star-shaped configuration space populated by star-shaped obstacles.

⇒ A star-shaped set S is a set where there exists at least one point that is within line of sight of all other points in the set

$\exists x$ such that $\forall y \in S, \qquad tx + (1-t)y \in S \quad \forall t \in [0, 1]$

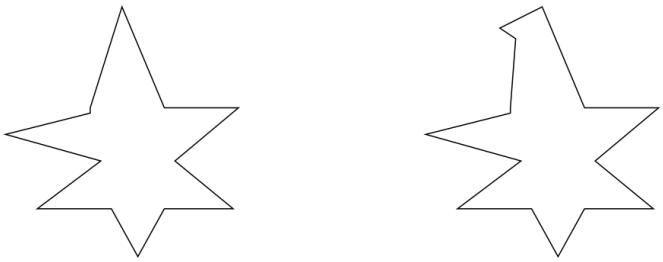


Figure 4.17 (Left) Star-shaped set. (Right) Not a star-shaped set.

⇒ All convex sets are star-shaped, but the converse is not true.

⇒ The approach is to map a configuration space populated by star-shaped obstacles into a space populated by sphere-shaped obstacles.

⇒ It can be shown that for two free configuration spaces M and F.

if φ : M → [0, 1] is a navigation function on M and there exists a mapping h : F → M which is a diffeomorphism.

then φ = φ ◦ h is a navigation function on F.

⇒ This diffeormorphism ensures that there is a one-to-one correspondence between critical points.

⇒The h mapping between the star- and sphere-spaces will be constructed using a translated scaling map

$$T_i(q) = \nu_i(q)(q - q_i) + p_i,$$

where

$$\nu_i(q) = (1 + \beta_i(q))^{1/2} \frac{r_i}{d(q, q_i)},$$

where q_i is the center of the star-shaped set, and p_i and r_i are, respectively, the center and radius of the spherical obstacle.

⇒For the star-shaped obstacle $\mathcal{QO}_i$, we define the analytical switch

$$s_i(q, \lambda) = \left(\sigma_\lambda \circ \frac{\gamma_\kappa \bar{\beta}_i}{\beta_i}\right)(q) = \left(\frac{\gamma_\kappa \bar{\beta}_i}{\gamma_\kappa \bar{\beta}_i + \lambda \beta_i}\right)(q),$$

where

$$\bar{\beta}_i = \prod_{j=0, j\neq i}^{n} \beta_j,$$

⇒We define a similar switch for the goal which is one at the goal and zero on the boundary of the free space

$$s_{q_{\text{goal}}}(q, \lambda) = 1 - \sum_{i=0}^{M} s_i$$

⇒Now, using the above switches and a translated scaling map, we can define a mapping between star-space and sphere-space as

$$h_\lambda(q) = s_{q_{\text{goal}}}(q, \lambda) T_{q_{\text{goal}}}(q) + \sum_{i=0}^{M} s_i(q, \lambda) T_i(q)$$

★ Potential function in Non-Euclidean Space

⇒ Another major challenge for implementing potential functions is when the configuration space is non-Euclidean and multidimensional.

⇒ In order to deal with this difficulty, we will define a potential function in the workspace, which is Euclidean, and then lift it to the configuration space.

⇒To do so, instead of thinking of gradients as velocity vectors, we will now think of them as forces.

⇒ We then establish a relationship between a workspace force and a configuration space force.

⇒ Then we apply this relationship to a single rigid-body robot.

⇒Finally, we apply this relationship to a multibody robot.

● Relationship between Forces in the Workspace and Configuration Space

⇒ Since the workspace is a subset of a low-dimensional space (either R2 or R3), it is much easier to implement and evaluate potential functions over the workspace than over the configuration space.

⇒ Naturally, workspace potential functions give rise to workspace forces

↳ but ultimately, we will need forces in the configuration space to determine the path for the robot.

⇒ Let x and q be coordinate vectors representing a point in the workspace and the configuration of the robot.

⇒ where the coordinates x and q are related by the forward kinematics x = φ(q).

⇒ Let f and u denote generalized forces in the workspace and the configuration space, respectively.

⇒ To represent a force f acting at a point x in the workspace as a generalized force u acting in the robot's configuration space, we use the principle of virtual work.

↳ says that work (or power) is a coordinate-independent quantity

⇒ This means that power measured in workspace coordinates must be equal to power measured in configuration space coordinates.

⇒ In the workspace, the power done by a force f is the familiar $f^T \dot{x}$.

⇒ In the configuration space, power is given by $u^T \dot{q}$.

⇒ We know that $\dot{x} = J\dot{q}$, where $J = \partial\phi/\partial q$ is the Jacobian of the forward kinematic map.

⇒ Therefore, the mapping from workspace forces to configuration space forces is given by

$$f^T J\dot{q} = u^T\dot{q}$$
$$f^T J = u^T$$
$$J^T f = u.$$

• Potential functions for Rigid-Body Robots

⇒ In the configuration space, potential function was conceptually simple because the robot was represented by a single point, which we treated as a point mass under the influence of a potential field.

⇒ In the workspace, things are not so simple; the robot has finite area in the plane and volume in three dimensions.

⇒ Evaluating the effect of a potential field on the robot would involve computing an integral over the area/volume defined by the robot, and this can be quite complex (both mathematically and computationally).

⇒ An alternative approach is to select a subset of points on the robot, called control points, and to define a workspace potential for each of these points.

⇒ We then use the relationship established in the previous subsection to convert the individual workspace forces to configuration space forces.

⇒ We then add them to get the total configuration space force.

⇒ As a result, we have approximately "lifted" the total workspace forces on the robot to a generalized force in the configuration space.

⇒ We need to pick control points {r_i} on the robot.

⇒ The minimum number of control points depends upon the number of degrees of freedom of the robot.

⇒ For example, for a rigid-body robot in the plane, we can fix the position of the robot by fixing the position of two of its points.

⇒ For each r_j , the attractive potential is

$$U_{\text{att},j}(q) = \begin{cases} \frac{1}{2}\zeta_i d^2(r_j(q), r_j(q_{\text{goal}})), & d(r_j(q), r_j(q_{\text{goal}})) \le d^*_{\text{goal}} \\ d^*_{\text{goal}}\zeta_j d(r_j(q), r_j(q_{\text{goal}})) - \frac{1}{2}\zeta_j d^{*2}_{\text{goal}}, & d(r_j(q), r_j(q_{\text{goal}})) > d^*_{\text{goal}}. \end{cases}$$

⇒ With this potential function, the workspace force for attractive control point r_i is defined by

$$\nabla U_{\text{att},j}(q) = \begin{cases} \zeta_i(r_j(q) - r_j(q_{\text{goal}})), & d(r_j(q), r_j(q_{\text{goal}})) \le d^*_{\text{goal}}, \\ \frac{d^*_{\text{goal}}\zeta_j(r_j(q) - r_j(q_{\text{goal}}))}{d(r_j(q), r_j(q_{\text{goal}}))}, & d(r_j(q), r_j(q_{\text{goal}})) > d^*_{\text{goal}}. \end{cases}$$

⇒ For the workspace repulsive potential fields, we use the same control points {r_j}, and define the repulsive potential for r_j as

$$U_{\text{rep}i,j}(q) = \begin{cases} \frac{1}{2}\eta_j \left(\frac{1}{d_i(r_j(q))} - \frac{1}{Q^*_i}\right)^2, & d_i(r_j(q)) \le Q^*_i, \\ 0, & d_i(r_j(q)) > Q^*_i, \end{cases}$$

⇒ The gradient of each U_{repi, j} corresponds to a workspace force

$$\nabla U_{\text{rep}i,j}(q) = \begin{cases} \eta_j \left(\frac{1}{Q^*_i} - \frac{1}{d_i(r_j(q))}\right) \frac{1}{d^2_i(r_j(q))} \nabla d_i(r_j(q)), & d_i(r_j(q)) \le Q^*_i, \\ 0, & d_i(r_j(q)) > Q^*_i. \end{cases}$$

⇒ Often the vertices of the robot are used as the repulsive control points, but it is important to note that this selection of repulsive control points does not guarantee that the robot cannot collide with an obstacle.

⇒ The total configuration space force acting on the robot is the sum of the configuration space forces that result from all attractive and repulsive control points

$$\begin{aligned} u(q) &= \sum_j u_{\text{att}\,j} + \sum_{ij} u_{\text{rep}ij} \\ &= \sum_j J_j^T(q)\nabla u_{\text{att}\,ij}(q) + \sum_i \sum_j J_j^T(q)\nabla u_{\text{rep}ij} \end{aligned}$$

↳ in which J_j(q) is the Jacobian matrix for control point r_j .

⇒ It is essential that the addition of forces be done in the configuration space and not in the workspace.

• Path Planning Algorithm

⇒ Having defined a configuration space force, which we will again treat as a velocity, we can use the same gradient descent method for this case.

⇒ There are a number of design choices that must be made:

- ζ_j controls the relative influence of the attractive potential for control point r_j.
- η_j controls the relative influence of the repulsive potential for control point r_j.
- Q_i^* We can define a distinct Q_i^* for each obstacle.