Potential Functions
Configuration Space

• A key concept for motion planning is a configuration:

  a complete specification of the position of every point in the system

• A simple example: a robot that translates but does not rotate in the plane:

  – what is a sufficient representation of its configuration?

The space of all configurations is the configuration space or C-space.

C-space formalism:
Lozano-Perez ‘79
Three Fundamental Issues with C-space

• How do we describe configurations?
  – What is the fundamental space of configurations for a mechanisms?
  – How we relate the configuration space to the workspace? Kinematics

• How are configurations related to controllable inputs to the system?
  – Holonomic vs. Non-holonomic systems
  – Effect of dynamics

• How are obstacles represented in C-space
  – Explicit mapping via Inverse kinematics
  – Implicit in algorithm
Configuration Space

Topology of Configuration Space

\[ S^1 \times S^1 \]

Cartesian product of two circles (2D torus)

The Workspace

\[
\begin{bmatrix}
  x \\
  y \\
\end{bmatrix} = \begin{bmatrix}
  L_1 \cos(q_1) \\
\end{bmatrix} + \begin{bmatrix}
  L_1 \cos(q_1 + q_2) \\
\end{bmatrix}
\]

\[(x, y) = (L_1 \cos(q_1), L_1 \cos(q_1 + q_2))\]
Mobility

n: Number of moving rigid bodies in space
N: Number of rigid bodies, including the fixed body (n+1)
D: Number of free-space degrees of freedom (i.e. 3, 6)

What is the total number of degrees of freedom?

\[ M = 6 \times (4-1) = 18 \]
Joints: Degrees of Freedom

Rigid (no motion)  Prismatic (1)  Revolute (1)  Parallel Cylinders (2)

Cylindrical (2)  Spherical (3)  Planar (3)  Edge Slider (4)

Cylindrical Slider (4)  Point Slider (5)  Spherical Slider (5)  Crossed Cylinders (5)
Gruebler’s Formula

- Each joint constrains the degrees of freedom of rigid bodies
- Subtract the number of degrees of freedom taken away from joints

\[ M = Dn - \sum_{i=1}^{J} (D - f_i) \]

\( J \): Number of Joints
\( f_i \): Number of DOF of joint i
Open vs. Closed Chain

Simple Open Chain \((J=n)\)

\[
M = Dn - \sum_{i=1}^{J} (D - f_i)
= \sum_{i=1}^{J} f_i
\]

Simple Closed Chain \((J=n+1)\)

\[
M = Dn - \sum_{i=1}^{J} (D - f_i)
= \sum_{i=1}^{J} f_i - D
\]
Configuration Space

Where do we put B?

Torus
(wraps horizontally and vertically)
Configuration Space

An obstacle in the robot’s workspace

Where do we put \( \bullet \) ?

Torus
(wraps horizontally and vertically)
Configuration Space

Reference configuration

An obstacle in the robot’s workspace

The C-space representation of this obstacle…

How do we get from A to B?
Two Link Path

Thanks to Ken Goldberg
Some Other Examples of C-Space

- A rotating bar fixed at a point
  - what is its C-space?
  - what is its workspace (the set of points it can reach?)

- A rotating bar that translates along the rotation axis
  - what is its C-space?
  - what is its workspace

- A two-link manipulator
  - what is its C-space?
  - what is its workspace?
  - Suppose there are joint limits, does this change the C-space?
  - The workspace?
Potential Field: The Basic Idea

A really simple idea:

- Suppose the configuration goal is a point \( q_{goal} \in \mathbb{R}^2 \)
- Suppose the robot is at configuration \( q_{start} \in \mathbb{R}^2 \)
- Think of a “spring” drawing the robot toward the goal and away from obstacles:
- Can also think of using same/opposite electric charges to repeal/attract

Principles of Robot Motion
Another Idea

- Think of the goal as the bottom of a bowl
- The robot is a ball on the rim of the bowl
- What will happen?

mechatronics.eng.buffalo.edu
The General Idea

• Both the bowl and the spring analogies are ways of storing potential energy
• The robot aims to move to a lower energy configuration
• A potential function is a function \( U: \mathbb{R}^M \rightarrow \mathbb{R} \)
• Energy is minimized by following the negative gradient of the potential energy function:

\[
\nabla U(q) = \begin{bmatrix}
\frac{\partial U}{\partial q_1} \\
\vdots \\
\frac{\partial U}{\partial q_M}
\end{bmatrix}
\]

• We can now think of a vector field over the configuration space
  – At any given point in time, the robot looks at the vector corresponding to its current configuration and moves in that direction
Attractive/Repulsive Potential Field

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

- \( U_{\text{att}} \) is the “attractive” potential --- move to the goal
- \( U_{\text{rep}} \) is the “repulsive” potential --- avoid obstacles
Artificial Potential Field Methods: Attractive Potential

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

\( \zeta \) is a parameter used to scale the attractive potential

Principles of Robot Motion

Quadratic Potential

\[ U_{\text{att}}(q) = \frac{1}{2} \zeta d^2(q, q_{\text{goal}}), \]
\[ \nabla U_{\text{att}}(q) = \nabla \left( \frac{1}{2} \zeta d^2(q, q_{\text{goal}}) \right), \]
\[ = \frac{1}{2} \zeta \nabla d^2(q, q_{\text{goal}}), \]
\[ = \zeta (q - q_{\text{goal}}), \]
Artificial Potential Field Methods: Attractive Potential

Combined conic and quadratic potential functions

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

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

In some cases, it may be desirable to have distance functions that grow more slowly to avoid huge velocities when far from the goal.

one idea is to use the quadratic potential near the goal (< d*) and the conic farther away.

One minor issue: what?
The Repulsive Potential

\[ D(q): \text{distance to the closest obstacle} \]
\[ Q^*: \text{maximum repulsive distance} \]
\[ \eta: \text{repulsive field gain} \]

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

whose gradient is

\[ \nabla U_{\text{rep}}(q) = \begin{cases} 
\eta \left( \frac{1}{Q^*} - \frac{1}{D(q)} \right) \frac{1}{D^2(q)} \nabla D(q), & D(q) \leq Q^*, \\
0, & D(q) > Q^*,
\end{cases} \]
Total Potential Function

\[ U(q) = U_{\text{att}}(q) + U_{\text{rep}}(q) \]
Gradient Descent

• A simple way to get to the bottom of a potential function $U(q)$ is to follow the gradient $-\nabla U(q)$
• At each configuration $q$, evaluate $-\nabla U(q)$ and take a “small step” $\Delta q$ in that direction to reach a new configuration $q + \Delta q$
• At a critical configuration $q^*$, $\nabla U(q^*) = 0$
Gradient Descent

• For a 1-d function, how do we know we are at a minimum (or maximum)?
  – 1st derivative tells us if we have a critical point (maximum, minimum, saddle point)
  – 2nd derivative tells us the kind of critical point
  \[ f''(x^*) < 0 \] then \( x^* \) is a maximum
  \[ f''(x^*) > 0 \] then \( x^* \) is a minimum

• For a M-d function the Hessian \( H_f(x) \) is the \( M \times M \) matrix of second derivatives

• If the Hessian is nonsingular \( \det(H) \neq 0 \), the critical point is a unique point
  – if \( H_f(x^*) \) is positive definite \( (x^*^TH_f(x^*)x^* > 0) \), is a minimum (positive eigenvalues)
  – if \( H_f(x^*) \) is negative definite \( (x^*^TH_f(x^*)x^* < 0) \), is a maximum (negative eigenvalues)
  – if \( H_f(x^*) \) is indefinite, \( x^* \) is a saddle point

www.mathsisfun.com

Wikipedia.org
The Hessian

- The Hessian is the $M \times M$ matrix of second derivatives

$$H_U(q) = \begin{bmatrix}
\frac{\partial^2 U}{\partial q_1^2} & \cdots & \frac{\partial^2 U}{\partial q_1 q_M} \\
\vdots & \ddots & \vdots \\
\frac{\partial^2 U}{\partial q_M q_1} & \cdots & \frac{\partial^2 U}{\partial q_M^2}
\end{bmatrix}$$

- The Hessian is the Jacobian of the gradient

$H_U(q) = J_{\nabla U}(q)$
Gradient Descent

Algorithm

1. You need:
   A start configuration: \( q_{\text{start}} \)
   Gradient of a potential function: \( \nabla U(q) \)

2. \( q_0 = q_{\text{start}} \)

3. \( i = 0 \)

4. while \( \nabla U(q_i) \neq 0 \) do

5. \( q_{i+1} = q_i - \alpha_i \nabla U(q_i) \)

6. \( i = i + 1 \)
Potential Functions Question

How do we know that we have only a single (global) minimum?

We have two choices:
- not guaranteed to be a global minimum: do something other than gradient descent (what?)
- make sure only one global minimum
Potential Fields on Non-Euclidean Spaces

• Thus far, we’ve dealt with points in $\mathbb{R}^n$, which is suitable for mobile robots. What about manipulators?
• Recall we can think of the gradient vectors as virtual forces that attract/repeal a point. The basic idea is to define forces in the workspace (which is $\mathbb{R}^2$ or $\mathbb{R}^3$)

Force $f$ acting at a point $p$ in the robot’s workspace
Torques $\tau$ acting in the robot’s configuration to move the robot
$v = J \dot{q}$, where $J = \frac{\partial f}{\partial q}$
$\tau^T \dot{q}$: power in the configuration space
$f^T v$: power in the work space

Power is conserved!

$\begin{align*}
    f^T J \dot{q} &= \tau^T \dot{q} \\
    f^T J &= \tau^T \\
    J^T f &= \tau
\end{align*}$

Jacobian (again!)
Potential Fields on Non-Euclidean Spaces

- Recall we can think of the gradient vectors as virtual forces.
- The idea is to define virtual forces in the workspace that will be applied at specific points on the robot arm.
  - We have a virtual force $f$ applied in the workspace (i.e. $\mathbb{R}^2$ or $\mathbb{R}^3$) and $\tau$ is in the configuration space (i.e. $\mathbb{R}^M$).
  - Using $J^T f = \tau$ we can define virtual forces in the workspace and then apply the resulting real torques in the configuration space.
- Example: our two-link manipulator.
Potential Fields on Non-Euclidean Spaces

Example: Our two-link manipulator

\[
\begin{bmatrix}
x(q_1, q_2) \\
y(q_1, q_2)
\end{bmatrix} = \begin{bmatrix}
L_1 \cos(q_1) + L_2 \cos(q_1 + q_2) \\
L_1 \sin(q_1) + L_2 \sin(q_1 + q_2)
\end{bmatrix}
\]

Compute and evaluate \( J(q_1, q_2) \)

Suppose \( \mathbf{w}_{goal} = [L_2, L_1]^T \), then \( \mathbf{f} = \mathbf{p}_{goal} - [x \quad y]^T \)

\( \mathbf{\tau} = J^T(q_1, q_2) \)

Then apply the joint torques \((\tau_1, \tau_2)\) to the actuators. These torques will result in a force \( \mathbf{f} \) at the end effector that will move the robot toward \( \mathbf{p}_{goal} \).

If joint “i” is prismatic, then \( \tau_i \) is a force.
Add Obstacles

- Pick several points on the manipulator
- Compute attractive and repulsive virtual forces for each
- Transform these forces into the configuration space and add them
- Use the resulting torques to move the robot (in its configuration space)

Be careful to use the correct Jacobian! Thus far we have been using the Jacobian that concern the coordinate frame of the end-effector.
Forces representing the gradient of a potential function in Cartesian space

Initial gradient is \([-1 \ 1]^T\) indicating that the end-effectors needs to move along \(-X\) and \(+Y\) with equal magnitude

Joint torques that will generate the desired forces
Computing 2D Distance: Use a Grid

• For mobile robots, the distance between the robot and obstacles can be approximated with a discrete representation of the space
  – The Brushfire algorithm is one way to do this
    • Need to define a grid on space
    • Need to define connectivity (4/8)
    • Squares occupied by obstacles are assigned a value of 1 in the grid; free squares are assigned a value of 0

<table>
<thead>
<tr>
<th>n1</th>
<th>n2</th>
<th>n3</th>
</tr>
</thead>
<tbody>
<tr>
<td>n4</td>
<td>n5</td>
<td>n6</td>
</tr>
<tr>
<td>n7</td>
<td>n8</td>
<td>n9</td>
</tr>
</tbody>
</table>

4 neighbors: n₂, n₄, n₆, n₈ are neighbors of n₅

<table>
<thead>
<tr>
<th>n1</th>
<th>n2</th>
<th>n3</th>
</tr>
</thead>
<tbody>
<tr>
<td>n4</td>
<td>n5</td>
<td>n6</td>
</tr>
<tr>
<td>n7</td>
<td>n8</td>
<td>n9</td>
</tr>
</tbody>
</table>

8 neighbors: n₁, n₂, n₃, n₄, n₆, n₇, n₈, n₉ are neighbors of n₅

www.societyofrobots.com