Structure and Synthesis of Robot Motion

Introduction: Some Concepts in Dynamics and Control

Subramanian Ramamoorthy
School of Informatics

23 January, 2012
How do we describe motion?

From the previous lecture:

- Describe what needs to happen in the task space (e.g., trace a line)
- Use kinematics calculations to figure out what this means for joints (c-space)

Next question:

**Dynamics:** *when* should each c-space point be visited?
Motion of a Particle

Consider the case of a single particle of mass $m$ that moves in a world $W = \mathbb{R}$.

The applied force is a scalar $f \in \mathbb{R}$.

Let $q$ denote the state (position) of the particle in $W$ at time $t$.

Then, $\ddot{q} = \frac{f}{m}$

If you fix mass, say $m = 1$, and force, say $\ddot{q} = u$, then the motion is completely described by two variables:

$$x_1' = x_2$$

$$x_2' = \frac{u}{m}$$
Motion of a Lunar Lander

There are three thrusters and gravity acts on the vehicle. We ignore rotations.

We model this as a particle of mass $m$ in a 2-dim world.

We can write equations of motion as ($u_i \in [0, 1]$),

$$m\ddot{q}_1 = u_l f_l - u_r f_r$$

$$m\ddot{q}_2 = u_u f_u - mg$$

Using 2 positions and velocities, we could write this as 4 first-order equations.
How to Move Beyond Point Masses?

• Clearly, we want to be able to model more complex robots as they appear in practice

• Within the Newton-Euler formalism, one can try to derive further conservation laws, e.g., momentum

• This yields additional equations that act as differential constraints on the state space
One ‘Modern’/General Method: Lagrangian Mechanics

• Based on calculus of variations – optimisation over the space of paths
• Motion is described in terms of the minimisation of an action functional:

$$\Phi(x) = \int_T L(x(t), \dot{x}(t), t) dt$$

Example: Shortest-path functional

$$L(x, \dot{x}, t) = \sqrt{1 + \dot{x}^2}$$

Optimization is over possible small perturbations in functional form
Recap from Calculus: Extrema of Functions

**Extreme Value Theorem**
Suppose $f(x)$ is continuous on interval $[a, b]$ then there are two numbers $a \leq c, d \leq b$ such that $f(c)$ is an absolute maximum and $f(d)$ is an absolute minimum for the function.

**Fermat’s Theorem**
If $f(x)$ has relative extrema at $x = c$ and $f'(c)$ exists then $x = c$ is a critical point of $f(x)$. In fact it will be a critical point such that $f'(c) = 0$

So, $f(x)$ is a continuous function on the interval $[a, b]$,

- Find critical points by solving $f'(x) = 0$
- Check which ones are absolute maxima/minima

Confirming maxima or minima: Second Derivative
- $f''(x) > 0$: minima
- $f''(x) < 0$: maxima
Consider all possible paths joining two points A and B in the plane. Suppose a particle is allowed to move along any of these paths and let the particle have a definite velocity $v(x, y)$ at each point $(x, y)$. Then one can define a functional associating each path with the time taken.

Based on this we can ask about absolute maxima/minima for paths (shortest time).

We seek $y(x)$ so that $J = \int_{a}^{b} \sqrt{1 + y'^2}$ is minimized.
An Interesting Question about Paths

*Brachistochrone* problem (Johann Bernoulli, 1696):

Let \( P_1 = (0, 0) \) and \( P_2 = (1, -1) \) be two points in the plane. Connect them by a differentiable curve \( y = f(x) \). Suppose a bead is allowed to slide, frictionless, along these curves then what is the curve that minimizes the time taken by the bead?

Bead’s motion: \( y(t) = f(x(t)) \)

\[
v_1(t) = \dot{x}(t) \quad \text{and} \quad v_2(t) = f'(x(t))\dot{x}(t)
\]

\[
|v(t)| = \sqrt{1 + [f'(x(t))]^2}\dot{x}(t)
\]

Conservation of energy: \( \frac{1}{2}mv^2 = -mgy(t) \)

\[
\frac{\sqrt{1+[f'(x(t))]^2}}{\sqrt{-2gf(x(t))}}\dot{x}(t) = 1
\]

\[
T = \int_0^1 \frac{\sqrt{1+[f'(x(t))]^2}}{\sqrt{-2gf(x(t))}}\,dx
\]

This is to be minimized, over all paths \( f(x(t)) \).
Q.: What is the Derivative of a Functional?

• From basic calculus, we know what to do with functions:

\[ y = f(x); \quad x, y \in \mathbb{R} \]

\[ f'(x) = \frac{dy}{dx} \]

– A statement about how \( y \) changes for “small” changes in \( x \)

• But now, instead of \( x \) & \( y \), we have:

\[ T = \int_0^1 \frac{\sqrt{1+[f'(x(t))]^2}}{\sqrt{-2gf(x(t))}} \, dx \]

• How does \( T \) change with respect to “small” changes in \( f(x(t)) \)
  – What does it mean to make a “small” change in \( f(x(t)) \)?
Euler-Lagrange Equation

**Differentiability of a functional:**
If you have a functional $I : \chi \rightarrow \mathbb{R}$, it is differentiable at $f$ if there is a continuous linear functional $T : \chi \rightarrow \mathbb{R}$ such that $\forall g \in \chi$,

$$T[g] = \lim_{t \to 0} \frac{I[f + tg] - I[f]}{t}$$

If $I[y] = \int_{a}^{b} L(x, y(x), \dot{y}(x)) \, dx$ then $\forall z \in \chi$,

Through a derivation to work out when $\delta I$ (term within limits) vanishes,

$$\frac{\partial L}{\partial y} - \frac{d}{dx} \left( \frac{\partial L}{\partial y'} \right) = 0$$
Solving the *Brachistochrone* Problem

Functional is defined in terms of:

\[ L = \frac{\sqrt{1 + [f'(x(t))]^2}}{\sqrt{-2gf(x(t))}} \]

With \( y = f(x(t)) \), apply

\[ \frac{\partial L}{\partial y} - \frac{d}{dx}\left(\frac{\partial L}{\partial y'}\right) = 0 \]

The solution is a family of *cycloids*:

\[ x = r(\theta - \sin \theta) + c \]
\[ y = r(1 - \cos \theta) \]

A ball will roll down the cycloid faster than the straight line!
Deriving Equations of Motion from Lagrangians: A Planar Robot

Begin with setting up the Lagrangian:

\[ L = \frac{1}{2} m (\dot{q}_1^2 + \dot{q}_2^2) + \frac{1}{2} I \ddot{q}_3^2 - m a_g q_2 \]

\[ u_1 = \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_1} - \frac{\partial L}{\partial q_1} = \frac{d}{dt} (m \ddot{q}_1) - 0 = m \ddot{q}_1 \]

\[ u_2 = \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_2} - \frac{\partial L}{\partial q_2} = \frac{d}{dt} (m \ddot{q}_2) - m a_g = m \ddot{q}_2 - m a_g \]

\[ u_3 = \frac{d}{dt} \frac{\partial L}{\partial \dot{q}_3} - \frac{\partial L}{\partial q_3} = \frac{d}{dt} (I \ddot{q}_3) - 0 = I \ddot{q}_3 \]

External forces balance accel. terms
This is a more complex example, so we need to write each term separately.

KE of first-link, \( K_1(q, \dot{q}) = \frac{1}{2} m_1 v_1^2 + \frac{1}{2} I_1 \omega_1^2 \)

PE of first-link, \( V_1(q) = m_1 a_g r_1 \sin q_1 \)

KE of second-link, \( K_2(q, \dot{q}) = \frac{1}{2} m_2 v_2^2 + \frac{1}{2} I_2 \omega_2^2 \)

PE of first-link, \( V_2(q) = m_2 a_g q_2 \sin q_1 \)

The complete Lagrangian is \( L = K_1 + K_2 - V_1 - V_2 \)

\[
L = \frac{1}{2} \left( (I_1 + I_2 + m_1 r_1^2 + m_2 q_2^2) \dot{q}_1^2 + m_2 \dot{q}_2^2 \right) - a_g \sin q_1 (m_1 r_1 + m_2 q_2)
\]
The complete Lagrangian is $L = K_1 + K_2 - V_1 - V_2$

$L = \frac{1}{2}((I_1 + I_2 + m_1 r_1^2 + m_2 q_2^2)\dot{q}_1^2 + m_2 \dot{q}_2^2) - a_g \sin q_1 (m_1 r_1 + m_2 q_2)$

From this, we can extract equations of motion as,

$u_1 = (I_1 + I_2 + m_1 r_1^2 + m_2 q_2^2)\ddot{q}_1 + 2 m_2 q_2 \dot{q}_1 \dot{q}_2 + a_g (m_1 r_1 + m_2 q_2) \cos q_1$

$u_2 = m_2 \ddot{q}_2 - m_2 q_2 \dot{q}_1^2 + a_g m_2 \sin q_1$
Question

- The equations of motion (and corresponding geometric/kinematic descriptions) will tell you what will happen given initial conditions, forces, etc.
- You want exactly the opposite – what should you tell the robot so it will go through the states you are interested in?!
The Optimal Control Problem

• Given a dynamical system with states and controls
• Find policy or sequence of control actions $u(t)$ up to some final time
• Forcing the state to go from an initial value to a final value
• While minimizing a specified cost function

The resulting state trajectory $x(t)$ is an optimal trajectory

Remarks:
• Certain combinations of cost functions and dynamical systems yield analytical solutions (most need to be solved numerically...)
• Often, the control policy can be described as a feedback function or control law
The Optimal Control Problem

We are looking for an optimal control $u^*(t) \leq t \leq t_f$ that minimizes

$$J = \phi[x(t_f), w(t_f), p(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), w(t), p(t), t]dt$$

where $x$ is the state, $u$ is the control, $w$ are disturbances, $p$ are physical parameters. Often, $w$ and $p$ are fixed and/or known.

So, the cost becomes,

$$J = \phi[x(t_f), t_f] + \int_{t_0}^{t_f} L[x(t), u(t), t]dt$$

Note that there are two distinct terms to the cost - a final term and a running term.
The Optimal Control Problem

This optimal control, \( u^*(t) \) \( t_0 \leq t \leq t_f \), acts on a dynamic system

\[
\dot{x}(t) = f[x(t), u(t), t]
\]

whose solution trajectory is,

\[
x(t) = x_0 + \int_{t_0}^{t_f} f[x(\tau), u(\tau), \tau] d\tau
\]

Often, one uses a general quadratic cost function in applications,

\[
J = \int_{t_0}^{t_f} \left\{ (x'(t)u'(t)) \left( \begin{array}{cc} Q & M' \\ M & R \end{array} \right) \left( \begin{array}{c} x(t) \\ u(t) \end{array} \right) \right\} dt
\]

\[
J = \int_{t_0}^{t_f} x'(t)Qx(t) + 2x'(t)Mu'(t) + u'(t)Ru(t) dt
\]

Such a cost function would penalize control effort and deviation of the distance from the desired state. The final cost term acts as a soft constraint on the end-point goals.
Classical Solution:
Necessary Conditions for Optimality

The solution to the optimal control problem can be captured by three necessary conditions:

\[
\begin{align*}
\lambda'(t) &= \left\{ -\frac{\partial H[\cdot]}{\partial x} \right\}' \quad t_0 \leq t \leq t_f \\
\lambda(t_f) &= \left\{ \frac{\partial \phi[\cdot]}{\partial x} \right\}' \\
\left\{ \frac{\partial H[\cdot]}{\partial u} \right\} &= \frac{\partial L}{\partial u} + \lambda' \frac{\partial f}{\partial u}
\end{align*}
\]

where \( H[x(t), u(t), \lambda(t), t] = L[x(t), u(t), t] + \lambda'(t)f[x(t), u(t), t] \)
Some Observations

• In robotics, the variability across tasks is exceptionally large
  – Robot morphologies differ
  – Task specifications differ
  – Disturbance classes and sources of uncertainty differ

• We do have certain coarse classification of types of tasks – manipulation, locomotion, area coverage, etc.

• But we are very far from being able to standardize our applications down to modules that can be entirely abstracted and dealt with using traditional program design approaches

• Let us look at one single (relatively old, but instructive) system to understand what the needs are ...
Handey

- System created by Thomas Lozano-Perez and many collaborators at MIT AI Lab
- Goal was to explore what it takes to program pick-and-place robots with certain basic properties:

  The word robot should conjure up the image of a system with (at least) three generic capabilities:

  - The ability to perceive its environment and to locate objects of interest.
  - The ability to act on its environment.
  - The ability to plan actions to achieve its goals.
State of the Art, in 1992

- When systems like Handey were being built, robots were being programmed at a very primitive level (“move to 1.2, 2.3, 3.4”) – like very primitive machine code
- Now, programming a pick-and-place task for a specific object, in a specific environment, with a specific robot, and to a specific destination is not that difficult.
  - The difficulty resides in achieving some degree of generality (i.e., autonomy)
- Research Goal: Manipulate a wide class of objects in a wide class of environments using a wide class of robots
What Handey Could Do

• Input: Object models for parts to be manipulated; High-level Move commands involving objects and destinations

• What Robot Does: Locates parts, picks a grasp, plans a path and moves towards goals (in cluttered environment)
What Problems might the Robot Face? Unexpected Changes

Consider the simplest issue:

- If you pre-program movements then what happens if A or B are not exactly where you thought they would be?
- In cases where you have sensory feedback, how might you account for still larger scale unexpected events (e.g., missing B)?
What Problems might the Robot Face?
Workspace Obstacles

If you pre-program movement patterns (e.g., ways to grasp) then what happens when obstacles impede your path?
What Problems might the Robot Face?
Computing Collision-Free Paths

How to compute the gross motion of getting from start pose to destination pose, respecting all constraints along the way?
What Problems might the Robot Face? Uncertainty

How to compute the gross/fine motion of the robot and objects, when none of the coordinates are known exactly (e.g., due to sensor/motor noise)?
What Problems might the Robot Face?
Error Detection and Recovery

Arguably, one of the most important issues – what should the robot do after something has done wrong?

e.g., part has slipped, environment has significantly changed w.r.t. models
Also Need to Think of Interactions between Constraints

Stable grasp vs. Collisions

Good grasp vs. Feasible Gross Motion Plans
Many High-Level Concerns

Broadly speaking, even with this simple setup, we can identify patterns of high-level concerns:

- Gross motion planning
- Grasp/Regrasp planning
- Multi-arm coordination
Gross Motion in Handey

- The c-space is represented in terms of slices
- The free-space is then represented using a tree data structure that utilizes the joint information in the slices
- In those days, these computations were implemented in parallel on a Connection Machine
Grasp Planning in Handey

Broadly, involves the following calculations:

- Choose faces to grasp
- Project obstacles/constraints onto grasp plane
- Characterize grasp and decide on approach/departure
- Search c-space for grasp pose, approach pose, departure pose, approach path, departure path
Multi-Robot Coordination in Handey

- Concept of a temporal coordination diagram
- A monotonically increasing line corresponds to sequence of poses
- Collisions can be represented as well and hence constraints can be specified