AMR Lectures | part1

- Path and Trajectory planning

Before going ahead we'll remind what is differential flatness. Differential flatness is about to reconstruct inputs and states with flat outputs.

$$w = h(q) \quad (1)$$

$$q = \eta(w, \dot{w}, \dots, w^{(r)}) \quad (2)$$

$$u = \mu(w, \dot{w}, \dots, w^{(r)}) \quad (3)$$

$w \rightarrow$ flat outputs

η and μ are functions of w and its derivatives up to certain order (r) that reconstruct inputs and states. (2) and (3) are ALGEBRAIC RECONSTRUCTION for kinematic model. Notice that, if we analyze geometric model q and u would be reconstructed by following equations:

$$q = \eta(w, w', \dots, w^{(r)'}) \quad (4)$$

$$u = \mu(w, w', \dots, w^{(r)'}) \quad (5)$$

What are differences between $\dot{w}$ and w':

$$\dot{w} = \frac{dw}{dt} \quad (6)$$

$$w' = \frac{dw}{ds} \quad (7)$$

where t and s stand for time and path parameter, respectively.

In previous lecture we analyzed unicycle in approach of differential flatness. We can see from the Table 1 that flat outputs are same for also bicycle and tricycle

with trailer :

Model	Flat outputs
Unicycle	x, y
Bicycle	x, y
Tricycle with trailer	x, y

Table 1

In other words, inputs and states of unicycle, bicycle, tricycle with trailer can be reconstructed by using cartesian coordinates of contact point (x, y).

- Chained form is also differentially flat. We can show it as below:

→ The model of (2, 3) case:

$$\dot{z}_1 = v_1 \quad (8)$$
$$\dot{z}_2 = v_2 \quad (9)$$
$$\dot{z}_3 = z_2 v_1 \quad (10)$$

→ flat outputs for this case are z_1, z_3.

→ State reconstruction:

z_1 and z_3 are already states. We need to reconstruct z_2:

$$z_2 = \frac{\dot{z}_3}{v_1} = \frac{\dot{z}_3}{\dot{z}_1} \quad (11)$$

→ Input reconstruction.

$$v_1 = \dot{z}_1 \quad (12)$$
$$v_2 = \dot{z}_2 = \frac{d}{dt}\left(\frac{\dot{z}_3}{\dot{z}_1}\right) = \frac{\ddot{z}_3 \dot{z}_1 - \ddot{z}_1 \dot{z}_3}{\dot{z}_1^2} \quad (13)$$

★ Note 1: In general chained form in the form of (2, n), FLAT OUTPUTS are z_1 and z_n

Use of differential flatness for planning:
The basic idea is, if we know flat outputs we can always reconstruct states and inputs which are associated to the known flat outputs. This means that, flat outputs can evolve ARBITRARILY (i.e. there is no limitations on evolution of the flat outputs). Even if, the model is constrained, the evolution of flat outputs is limitless. Because, the reconstruction formulas will provide the associated trajectory which will be automatically feasible. Why is that automatically feasible? We generate trajectory according to the system equations (equations of states (inputs) which can be reconstructed by flat outputs

- Generic algorithm for path and trajectory planning using flatness.

Note 2: If the robot we are considering does not have any property of flatness, we can't apply this algorithm.

Algorithm:
Assume robot moves from q_i to q_f as shown in figure 1. We have following consider following assumptions (hypotheses):
- Robot is flat
- $\omega = h(q)$

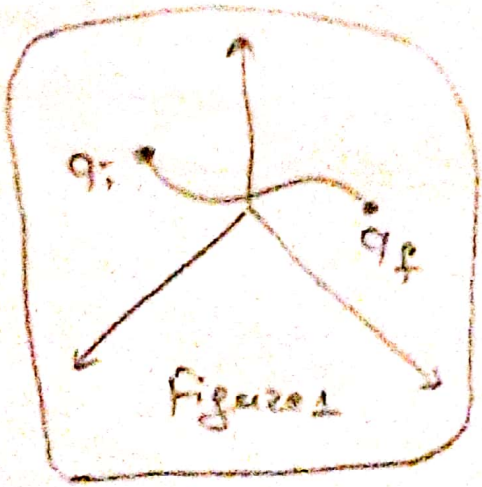


figure 1

Note 3: q_i → initial, q_f → final configurations

1. Compute initial and final values of flat outputs:

$$w_i = h(q_i) \quad (14)$$
$$w_f = h(q_f) \quad (15)$$

2. Generate a path (in s) or a trajectory (in t) from w_i to w_f, with the appropriate boundary conditions.

Notice that, if we generate path, we will work with path parameter (s), and if we generate trajectory, we will work with time (t).

We generate this path or trajectory, because we are not working with joint or configuration space. We are working with flat outputs. Thus, we need to generate them. Additionally, because of arbitrary choises of flat outputs, we can use any path or trajectory (e.g. sinusoidals, piecewise linear paths ...) as long as it is not interpolation problem.

3. Use the reconstruction formulas to compute the <u>states</u> and <u>inputs</u> path or trajectory. We do this step, in order to "transform" our computations from output space to state space.

Example

- Unicycle (path planning)

The model is unicycle is given as below:

$$x' = \tilde{v}' \cos\theta \quad (16)$$
$$y' = \tilde{v}' \sin\theta \quad (17)$$
$$\theta' = \omega \quad (18)$$

It's seen from (16), (17) and (18) we are using geometric model.

1. We can give initial and final configurations (q_i and q_f), and initial and final flat outputs (w_i and w_f) as following:

→ Initial → final configuration:

$$q_i = \begin{pmatrix} x_i \\ y_i \\ \theta_i \end{pmatrix} \longrightarrow q_f = \begin{pmatrix} x_f \\ y_f \\ \theta_f \end{pmatrix} \quad (19)$$

→ Initial → final outputs:

$$w_i = \begin{pmatrix} x_i \\ y_i \end{pmatrix} \longrightarrow w_f = \begin{pmatrix} x_f \\ y_f \end{pmatrix} \quad (20)$$

2. Now we can generate boundary conditions:

Boundary conditions will be generated according to the equations (16) and (17). Because initial and final x' and y' can't be arbitrarily different. On the other hand, they have to be in the relation with θ value. Thus, they can be given as below:

→ 2 conditions for initial point:

$$x'(s_i) = \tilde{v}(s_i) \cos\theta_i \quad (21)$$
$$y'(s_i) = \tilde{v}(s_i) \sin\theta_i \quad (22) \qquad | \quad \tilde{v}(s_i) = K_i \quad (23)$$

→ 2 conditions for final point:

$$x'(s_f) = \tilde{v}(s_f) \cos\theta_f \quad (24)$$
$$y'(s_f) = \tilde{v}(s_f) \cos\theta_f \quad (25) \qquad | \quad \tilde{v}(s_f) = K_f \quad (26)$$

Now we need to generate 2 interpolation functions (one for x and one for y) in order to generate the path.

→ x must go from x_i to x_f, satisfying (21) and (24) we'll use cubic polynomial for this purpose.

$$x(s) = s^3 x_f - (s-1)^3 x_i + \alpha_x s^2(s-1) + \beta_x s(s-1) \quad (27)$$

→ y must go from y_i to y_f, satisfying (22) and (25) we'll use cube polynomial for that, too:

$$y(s) = s^3 y_f - (s-1)^3 y_i + \alpha_y s^2(s-1) + \beta_y s(s-1) \quad (28)$$

Notice that in (27) and (28), $s \in [0, 1]$, where $s_i = 0$ and $s_f = 1$.

If we check (27), we'll see it already satisfies

$$x(s_i) = x(0) = x_i \quad (29)$$

$$x(s_f) = x(1) = x_f \quad (30)$$

On the other hand, we can choose α_x and β_x so as to satisfy (21) and (24).

$$\alpha_x = K \cos\theta_f - 3x_f \quad (31)$$

$$\beta_x = K \cos\theta_i + 3x_i \quad (32)$$

$v(s_i)$ and $v(s_f)$ can be chosen arbitrarily, thus we assumed in (31) and (32) that:

$$K = K_i = K_f \quad (33).$$

The same computations can be done for (28) as well.

3. As we have $x(s)$ and $y(s)$ interpretations ((27) and (28)) we can compute $x'(s), x''(s), \ldots$ and $y'(s), y''(s), \ldots$ up to certain order. Then by implementing reconstruction formulas we can reconstruct inputs and states, easily. (6)

- Chained form.

In this example, we are going to do path planning of (2,3) case of chained form. Because we'll do path planning, we are going to use geometric model. It can be written by following equations:

$$z_1' = \tilde{v}_1 \quad (34)$$

$$z_2' = \tilde{v}_2 \quad (35)$$

$$z_3' = z_2' \tilde{v}_1 \quad (36)$$

Flat outputs are z_1 and z_3, as we showed before.

1. Computation of flat outputs:

$$z_i = \begin{pmatrix} z_{1,i} \\ z_{2,i} \\ z_{3,i} \end{pmatrix} \longrightarrow z_f = \begin{pmatrix} z_{1,f} \\ z_{2,f} \\ z_{3,f} \end{pmatrix} \quad (37)$$

$$w_i = \begin{pmatrix} z_{1,i} \\ z_{3,i} \end{pmatrix} \longrightarrow w_f = \begin{pmatrix} z_{1,f} \\ z_{3,f} \end{pmatrix} \quad (38)$$

2. Boundary conditions

$$z_2(s_f) = \frac{z_3'(s_f)}{z_1'(s_f)} \quad (39)$$

$$z_2(s_i) = \frac{z_3'(s_i)}{z_1'(s_i)} \quad (40)$$

We can use the following polynomials in order to interpolate initial and final outputs

For w_1: $z_1(s) = z_{1,f}(s) - (s-1)\, z_{1,i}$ (41)

For w_2: $z_3(s) = z_{3,f}(s)s^3 - z_{3,i}(s-1)^3 + \alpha_z s^2(s-1) + \beta_z (s-1)^3$ (42)

In (41) and (42), $s \in [0,1]$ satisfies (38). On the other hand we can determine α_z and β_z so as to to satisfy (39) and (40). Notice that, z_1' is constant and non-zero value:

$$z_1' = z_{1,f} - z_{1,i} \neq 0 \quad (43)$$

α_2 and β_2 can be determined as below:

$$\alpha_2 = z_{2,f}(z_{1,f} - z_{1,i}) - 3z_{3,f} \quad (44)$$

$$\beta_2 = z_{2,i}(z_{1,f} - z_{1,i}) + 3z_{3,f} \quad (45)$$

Notice that, this scheme can't be applied when $z_{1,f} = z_{1,i}$. To handle this singular case, we can introduce via point:

$$q_v = \begin{pmatrix} x_v \\ y_v \\ \theta_v \end{pmatrix} \quad (46)$$

In that case, $\theta_i \neq \theta_v$ and we will solve the original path planning problem with solving 2 consecutive paths q_i to q_v and q_v to q_f.

- An alternative approach (not based on differential flatness)

This approach works for robot that can be transformed into <u>chained forms.</u>

Note: The robots which $n \leq 4$, are easily transformed into <u>chained forms.</u> $n > 4$ may or may not be transformed.

<u>Idea</u>: Chained forms can be easily integrable under <u>appropriate inputs.</u>

<u>Path planning.</u>

Geometric model for chained forms is described by the following equations:

$$\left.\begin{aligned} \dot{z}_1 &= \tilde{v}_1 \\ \dot{z}_2 &= \tilde{v}_2 \\ \dot{z}_3 &= z_2 \tilde{v}_1 \\ &\vdots \\ \dot{z}_n &= z_{n-1} \tilde{v}_1 \end{aligned}\right\} \quad (47)$$

Path can be described as in Figure 2.
In general, the space is given in Figure 2, is n-dimensional.

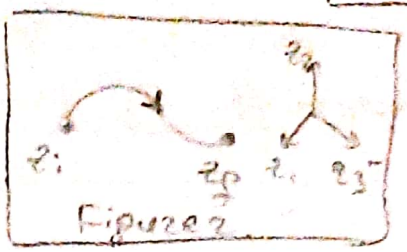


Figure 2

Idea of approach:

- Define Δ, such that is given with the difference between the final and initial values of the first coordinate, z_i and z_f can be given as below:

$$z_i = \begin{pmatrix} z_{1,i} \\ z_{2,i} \\ \vdots \end{pmatrix} (48) \; ; \; z_f = \begin{pmatrix} z_{1,f} \\ z_{2,f} \\ \vdots \end{pmatrix} (49)$$

Then, we can define Δ:

$$\Delta = z_{1,f} - z_{1,i} \quad (50)$$

We assume, Δ is nonzero ($\Delta \neq 0$).

- Choose $\tilde{v}_1$ as a sign of Δ:

$$\tilde{v}_1 = sign(\Delta) \quad (51)$$

In fact, $\tilde{v}_1$ is constant can be ± 1.

- Define $\tilde{v}_2$ as below:

$$\tilde{v}_2 = c_0 + c_1 s + \ldots + c_{n-2} s^{n-2} \quad (52) \; ; s \in [0, |\Delta|]$$

In (52), we have $n-1$ parameters ($c_0 \rightarrow c_{n-2}$). Because, we need $n-1$ parameter to place $n-1$ variables from z_2 to z_{n-1}.

By choosing c_0 to c_{n-2} parameters, we can compute $\tilde{v}_2$, which drives the robot from z_i to z_f.

Because, $\tilde{\sigma}_i$ is constant, z can be integrated easily (i.e. z because linear)

- Integrate chained form equations (can be done because $\tilde{\sigma}_i = \pm 1$) and impose that:

$$\begin{matrix} z_2(|\Delta|) = z_{2,f} \\ \vdots \qquad\qquad \vdots \\ z_n(|\Delta|) = z_{n,f} \end{matrix} \qquad (53)$$

(53) is a set of $(n-1)$ equations with $(n-1)$ parameters. Thus we can say that, the system is square.

Note: We use $|\Delta|$, because Δ can also be negative.

A linear system can be written as below:

$$D(\Delta) \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \end{pmatrix} = b(z_i, z_f, \Delta) \quad (54)$$

In (2,3) case $D(\Delta)$ and $b(z_i, z_f, \Delta)$ will be:

$$D(\Delta) = \begin{pmatrix} |\Delta| & \Delta^2/2 \\ \text{sign}(\Delta)\,\Delta^2/2 & \Delta^3/6 \end{pmatrix} \quad (55)$$

$$b(z_i, z_f, \Delta) = \begin{pmatrix} z_{2,f} - z_{2,i} \\ z_{3,f} - z_{3,i} - z_{3,i}\,\Delta \end{pmatrix} \quad (56)$$

$D(\Delta)$ is nonsingular coefficient matrix, if and only if $\Delta \neq 0$ (as we assumed).

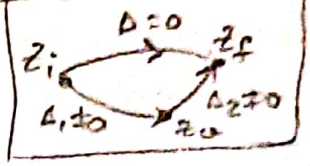
- What happens if $\Delta = 0$.

We know from (56) that, Δ is the difference between initial and final values of the i^{th} component of z. In order to solve this problem we'll add an intermediate point.

1. Add a "via point";

We can show it in Figure 3.

Figure 3



$$\Delta_1 = z_{1,0} - z_{1,i} \quad (57)$$

$$\Delta_2 = z_{1,f} - z_{i,2} \quad (58)$$

Using this we can solve path planning by dividing main path to 2 different path. We can solve them sequentially, in order to plan the path.

2. If z_i represents an angle we will use the following approach in order to solve the problem by adding artificial displacement:

$$z_{1,f}^{new} = z_{1,f}^{old} + 2\pi \quad (59)$$

(59) lets us avoid the computational issue, but in interpretational point of view it changes nothing.

- Choice of the timing law.

We have a path $q(s)$, where $s \in [s_i, s_f]$. We know that:

$$q_i = q(s_i) \quad (60)$$

$$q_f = q(s_f) \quad (61)$$

How do we choose timing law $s = s(t)$, in order to transform the given path to trajectory? Notice that, because we have already a path, in order to transform it to trajectory, we'll need to follow "decoupled" approach.

Note: A trajectory is needed to actually control the robot in time.

The timing law simply tells us the speed that robot moves along the path. We will generate timing laws according to some considerations. One of those typical considerations is velocity bound.

- Velocity bounds. (Unicycle)

For unicycle we will have some bounds on the driving and steering velocities, which can't exceed certain values:

$$|v(t)| \leq v_{max} \quad (62)$$

$$|\omega(t)| \leq \omega_{max} \quad (63)$$

A way to do this, to choose a uniform constant scaling between time (t) and path parameter.

$$t = \alpha s \quad (64)$$

When we apply (64), how can we choose timing laws that satisfies (62) and (63).

AME Lecture Part 4

1. The relation between geometric model is given as below (as we know):

$$v = \vec{v} \cdot \dot{s} \quad (65)$$

$$\omega = \vec{\omega} \cdot \dot{s} \quad (66)$$

Thus, we need to know how to define $\dot{s}$. From the equation (64):

$$\frac{d}{dt}(t) = \frac{d}{dt}(\alpha \cdot s) = \alpha \dot{s} = 1 \quad (67)$$

$$(67) \Rightarrow \dot{s} = \frac{1}{\alpha} \quad (68)$$

* Then we choose $\alpha = 1$ and compute $v(t)$ and $\omega(t)$ from (65) and (66), respectively.

Assume, after computations we will have such graphs as are demonstrated in the Figure 4.

By using these, we determine peak values for v and ω which are v_{peak} and ω_{peak}, respectively.

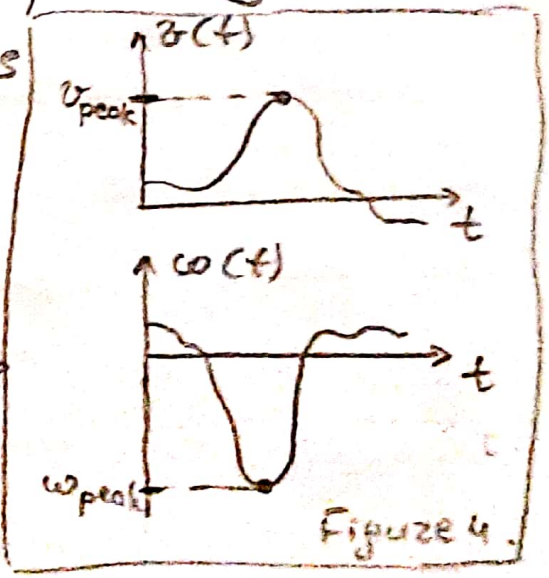


Figure 4.

2. We compare peak values with bounds. If there is no any violation of velocity bounds, by peak values, then $\alpha = 1$ is good choice for trajectory planning:

$$|v_{peak}| \leq v_{max} \quad (69)$$

$$|\omega_{peak}| \leq \omega_{max} \quad (70)$$

If (69) and (70) are satisfied, then $\alpha = 1$ is Ok!

3. If peak values violate the bounds, then we define ζ which is maximum violation.

$$\zeta = \max\left(\frac{|v_{peak}|}{v_{max}}, \frac{|\omega_{peak}|}{\omega_{max}}\right) \quad (71)$$

We compute (71), if (69) and (70) are not satisfied by peak values. Then, according to ζ we compute α as below:

$$\alpha = \frac{1}{\zeta} \quad (72)$$

Note: ζ is computed, no matter if 1 of 2 conditions was violated or both of them were violated by peak values.

By using (72), we scale the time in order to satisfy velocity bounds.

After applying this, we get velocities that satisfy the bounds:

$$|v| \leq v_{max} \quad (73)$$

$$|\omega| \leq \omega_{max} \quad (74)$$

Because we generate v and ω, according to the minimal scaling, (73) and (74) will be satisfied with (at least) one velocity saturating the bound at peak value.