Path and Trajectory specification

Robots are built to accomplish complex and difficult tasks that require highly non-linear motions.

Specifying the desired motion to achieve a specified goal is often a difficult task in itself.

Path: A path is a curve in pose space specifying position and attitude. A path specification contains no time specification.

Trajectory: A trajectory is a time-parameterised curve in pose space specifying position and attitude and time. Implicitly, a trajectory specifies a velocity profile of motion along a path in pose space.

Paths or trajectories are usually specified for the end effector of a robotic manipulator or for the body attached frame of a mobile robot.

Requirements of a trajectory

A trajectory should be specified relative to the station frame:

1. Allows generalisation to moving station frames without significant problems. Examples are moving conveyer belts.
2. Specification of a trajectory or path is task dependent and should be independent of robot.

A trajectory should be smooth (first and possibly second derivatives should be smooth)

1. Avoids wear on joints, motors and gears.
2. Reduces impulsive forces applied to payload. Avoids difficulties with a gripper losing grip on a payload and damage to tools.

A trajectory should satisfy the temporal requirements of the task.

1. Many tasks require fairly specific rates of motion or a time critical in completion. For example, a conveyer belt may be stationary only for a short period of time to allow a weld to be made, or in spray painting applications the movement of the tool piece is more important than its placement.

Trajectory generation

There are a number of steps to planning a trajectory of a robotic manipulator or mobile robot.

1. Specification of initial and goal configurations for a given motion.
2. Specification of intermediate 'way-points' or 'via-points' if required. This specification is often done to avoid obstacles in the work space.
3. Conversion of initial and final set points and way-points into joint configurations (optional - required if we choose to generate trajectory in joint space - use inverse kinematics).
4. Generation of a smooth path beginning at the initial goal, terminating at the final goal and passing through (or close) to intermediate way-points.
5. Generation of a suitable velocity profile for the path and consequently specification of the trajectory.
6. *Calculation of joint forces and torques along specified trajectory and verification of manipulator performance.*
7. Generation of set point data for robot controller.

From step 3 onward, the method details depend on whether our implementation is in joint or Cartesian space, and we will present these separately.

Initial, Goal, and Way-point generation

The specification of initial and goal configurations is generally comes *fait accompli* with the specification of the task.

Automatically generating a path that avoids obstacles is one of the more difficult questions in robotics.

There are three methods that are most commonly used

METHOD I:
Form a connected graph representation of the free space and search exhaustively for a collision free path. *Can be computationally prohibitive to implement although dynamic programming methods can be used to improve the search performance. Requires full knowledge of the environment in advance.*

METHOD II:
Assign every obstacle a potential and then apply a gradient descent path that minimises distance to the goal subject to the repelling potentials of the obstacles. *Tends to lead to local minima when multiple obstacles are present.*

METHOD III:
Specify suitable way-points based on *a-priori* knowledge or generated by an expert system. *Un-modelled changes in the work space may lead to undesired interaction of the robot with its local environment.*

Most applications with robotic manipulators are undertaken in controlled conditions where it is a trivial matter to assign suitable way-points to avoid obstacles.

Applications with mobile robots can often lead to extremely difficult and unsolvable obstacle avoidance problems.

Smooth Path Generation

A smooth path may be generated either in *joint space* or *Cartesian space*.

Joint space paths: To generate a joint space path the way-points must be converted into joint coordinates using the inverse kinematics of the manipulator (this was the optional step above). Trajectories are specified for each joint independently. The actual Cartesian position of the end effector is only known at the specified initial, goal and way-point positions, ie. a "straight-line" path in joint space will not produce a straight-line path of the end-effector in the workspace.

Cartesian paths: Cartesian paths are easy to specify, and the tip motion of the manipulator is completely specified. Joint motion is obtained via a repeated application of the inverse kinematics at every point along the trajectory. Since trajectories are not generated in joint space care must be taken that the path lies in the reachable work space and does not pass through singularities.

In either case the initial specification of the route is given in terms of way-points in Cartesian space

Let ${}^S_T T(k)$ for $k = 1, \ldots, N$ denote the initial pose, way-points and final goal.

Here ${}^S_T T(1)$ is the initial pose and ${}^S_T T(N)$ is the final goal.

Joint Space Schemes

1. For $k = 1, \ldots, N-1$.
2. Compute the way-points in joint space $\{\theta(k)\}$.

The path chosen in joint space is simply direct interpolation between joint variables.

$$\theta_i : \theta_i(1) \rightarrow \theta_i(2) \rightarrow \cdots \rightarrow \theta_i(N-1) \rightarrow \theta_i(N)$$

The relative values of the $\{\theta_i\}$ in the intermediary path regions depends on the rate at which each θ_i moves along the path between $\theta_i(k)$ and $\theta_i(k+1)$.

It is easiest to specify this relationship by directly specifying the trajectory in joint space.

3. Specify an initial time t_k and final time t_{k+1} for the trajectory segment $\{k, k+1\}$ between each way-point.
4. A smooth time trajectory in each θ_i joint space is calculated based on the initial and final data $\{\theta_i(k), t_k, \theta_i(k+1), t_{k+1}$ for each trajectory segment.

Note that the Cartesian path from the resulting trajectory is unknown.

Since the way-points were calculated using the inverse kinematics the Cartesian position of the robot is known at these points.

Choosing the Interpolating curves

The interpolating curves $\theta_i(t) : [t_k, t_{k+1}]$ can be chosen in a number of ways.

The three most common are

1. Polynomials in time. Cubic polynomial. Splines in time.
2. Linear interpolation with smoothing. Linear function with parabolic blends.
3. Optimal control methods. The shooting method.

The shooting method is a method of pushing the rates of acceleration and deceleration along the path close to maximum rates achievable for the mechanism.

Cubic polynomials

Given we want to fit a trajectory between a start and end point, 4 constraints on the trajectory curve are evident.

Let θ_0 and θ_f denote the initial and final conditions of the curve chosen for a given joint variable.

Let $\dot{\theta}_0$ and $\dot{\theta}_f$ denote the initial and final velocities. These velocities are often set to zero, although in general they need not be.

These four constraints can be satisfied by a polynomial of at least third degree.

Therefore, set the trajectory curve to be described by

$$\theta(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3$$

so that

$$\dot{\theta}(t) = a_1 + 2a_2 t + 3a_3 t^2$$

and

$$\ddot{\theta}(t) = 2a_2 + 6a_3 t$$

so that we can write our four desired constraints as four equations in four unknowns

$$\begin{aligned} \theta_0 &= a_0 \\ \theta_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\ 0 &= a_1 \\ 0 &= a_1 + 2a_2 t_f + 3a_3 t_f^2 \end{aligned}$$

Solving for the four unknown $\{a_0, \ldots, a_3\}$ gives

$$\begin{aligned} a_0 &= \theta_0 \\ a_1 &= 0 \\ a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) \\ a_3 &= \frac{2}{t_f^3}(\theta_f - \theta_0) \end{aligned}$$

Example 7.1:
(Craig Example 7.1)

Assume a simple single link robot where we require a smooth trajectory starting motionless a $\theta = 15$ degrees and ending at $\theta = 75$ in 3 seconds. Use a cubic polynomial to describe the trajectory.

Solution

simply need to plug data into above formula for coefficients to get $a_0 = 15$, $a_1 = 0$, $a_2 = 20$, $a_3 = -4.44$.

And so the equations for angle, velocity and accelerations are

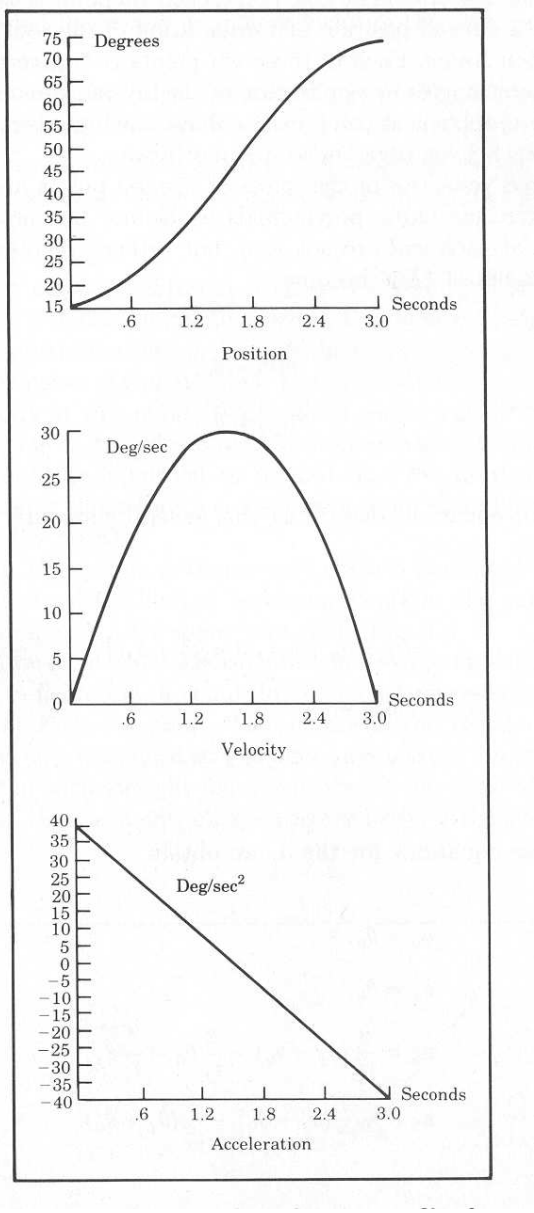
$$\theta(t) = 15 + 20t^2 - 4.44t^3$$

so that

$$\dot{\theta}(t) = 40t - 13.33t^2$$

and

$$\ddot{\theta}(t) = 40 + -26.66t$$



Cubic polynomials with non-zero velocity constraints

When you think about it, the cubic polynomial determined above will be of little use in finding practical interpolating curves $\theta_i(t) : [t_k, t_{k+1}]$. The robot will "stop" at each via point.

It is more practical if the trajectory derived passes through intermediate via points with continuous velocity and possibly even continuous acceleration. In this latter case, the trajectory is called a *spline*.

To be able to specify an arbitrary start and end velocity, our four equations are

$$\begin{aligned}
\theta_0 &= a_0 \\
\theta_f &= a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 \\
\dot{\theta}_0 &= a_1 \\
\dot{\theta}_f &= a_1 + 2a_2 t_f + 3a_3 t_f^2
\end{aligned}$$

and solving these equations gives

$$\begin{aligned}
a_0 &= \theta_0 \\
a_1 &= \dot{\theta}_0 \\
a_2 &= \frac{3}{t_f^2}(\theta_f - \theta_0) - \frac{2}{t_f}\dot{\theta}_0 - \frac{1}{t_f}\dot{\theta_f} \\
a_3 &= \frac{2}{t_f^3}(\theta_f - \theta_0) + \frac{1}{t_f^2}(\dot{\theta}_f - \dot{\theta_0})
\end{aligned}$$

Our problem then becomes one of specifying the required velocities at each via point.

A number of options are available; we can specify them manually, but this is tedious.

Alternatively we can come up with some heuristic to assign velocities. For example, if we plot a simple linear interpolation between each θ_k and θ_{k+1}, we might choose the velocity as the slope of the line orthogonal to the bisector of incoming and outgoing paths at the via point.

Cubic Splines

When fitting a cubic as our interpolating curve $\theta_i(t) : [t_k, t_{k+1}]$, we saw that 4 constraints were required. We previously chose the initial and final positions and velocities.

Clearly, the 2 position constraints need to be retained (we want the via points in the path!), however to minimise jerk we may want a continuous acceleration profile at via points - ie. a spline

So add the remaining two constraints that at the via point (i) velocity needs to be continuous and (ii) acceleration needs to be continuous.

If the first cubic is

$$\theta(t) = a_{10} + a_{11}t + a_{12}t^2 + a_{13}t^3$$

and the second is

$$\theta(t) = a_{20} + a_{21}t + a_{22}t^2 + a_{23}t^3$$

and, if θ_0, θ_v and θ_f are the initial, via and final positions respectively, the eight constraint equations are (2 for each cubic)

(position constraints)

$$\begin{aligned}
\theta_0 &= a_{10} \\
\theta_v &= a_{10} + a_{11}t_{f1} + a_{12}t_{f1}^2 + a_{13}t_{f1}^3 \\
\theta_v &= a_{20} \\
\theta_g &= a_{20} + a_{21}t_{f2} + a_{22}t_{f2}^2 + a_{23}t_{f2}^3
\end{aligned}$$

(initial and final velocities are zero)

$$\begin{aligned}
0 &= a_{11} \\
0 &= a_{21} + 2a_{22}t_{f2} + 3a_{23}t_{f2}^2
\end{aligned}$$

(acceleration and velocity are continuous at via point)

$$\begin{aligned}
a_{21} &= a_{11} + 2a_{12}t_{f1} + 3a_{13}t_{f1}^2 \\
2a_{22} &= 2a_{12} + 6a_{13}t_{f1}
\end{aligned}$$

which have solutions (writing t_{f1} and t_{f2} as t_f)

$$\begin{aligned}
a_{10} &= \theta_0 \\
a_{11} &= 0 \\
a_{12} &= \frac{12\theta_v - 3\theta_g - 9\theta_0}{4t_f^2} \\
a_{13} &= \frac{-8\theta_v + 3\theta_g + 5\theta_0}{4t_f^3} \\
a_{20} &= \theta_v \\
a_{21} &= \frac{3\theta_g - 3\theta_0}{4t_f} \\
a_{22} &= \frac{-12\theta_v + 6\theta_g + 6\theta_0}{4t_f^2} \\
a_{23} &= \frac{8\theta_v - 5\theta_g - 3\theta_0}{4t_f^3}
\end{aligned}$$

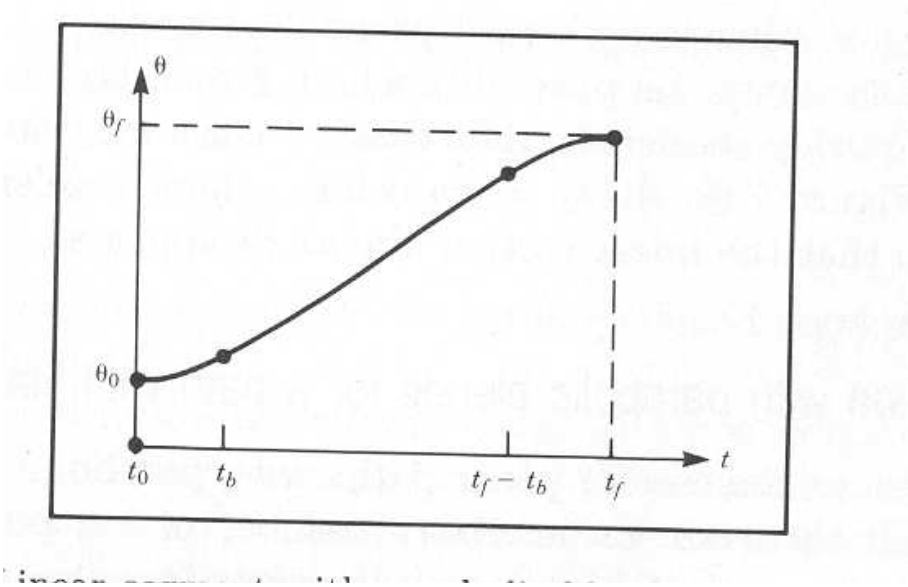
We are usually interested in the more general case when we have n adjacent interpolating curves. Craig provides a reference for a solution methodology in this case.

Linear functions with parabolic blends

A simpler possible interpolation scheme to polynomial is linear. Problem is that at via points the velocity and acceleration will be discontinuous.

Solution is to add a "parabolic blend" section at the via point to interface the two interpolating curves.

During the blend portion of the curve, we choose the blend so that constant acceleration is used to change the velocity smoothly.



Depending on the value of acceleration chosen, and on the change in velocity required between adjacent linear sections, the blend region will extend further or less into the linear region.

For the case where we consider only one interpolating curve in isolation , and we come to rest at each via point

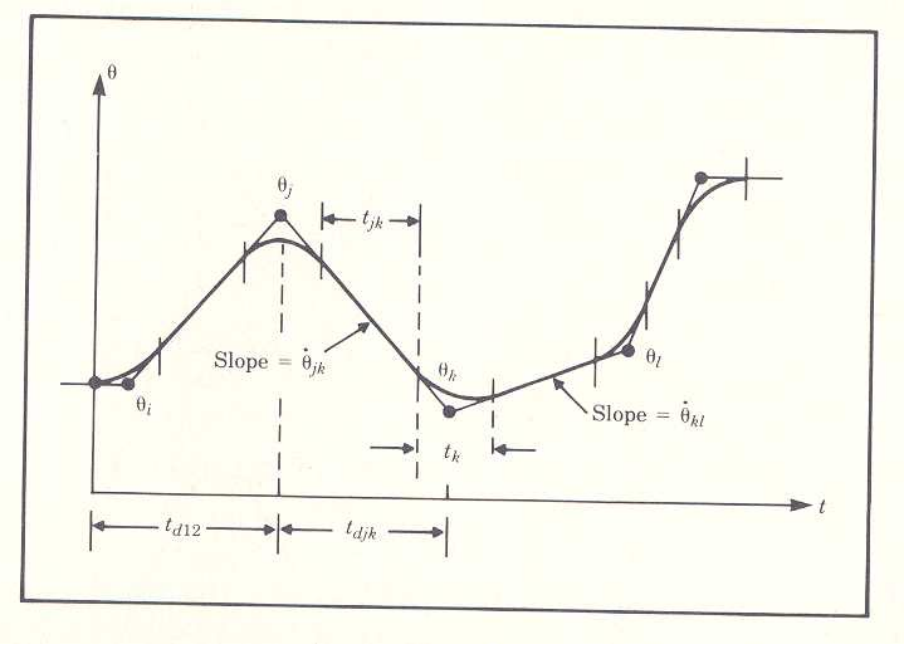
$$t_b = t/2 - \frac{\sqrt{\ddot{\theta}^2 t^2 - 4\ddot{\theta}(\theta_f - \theta_0)}}{2\ddot{\theta}}$$

where $\ddot{\theta}$ is the constant acceleration applied in the blend, and t is the time duration between the start and end of the interpolation.

The constraint on acceleration so that the linear section of the interpolation doesn't disappear is

$$\ddot{\theta} \geq \frac{4(\theta_f - \theta_0)}{t^2}$$

The same strategy can be used for the case of start and end points with multiple via points (detailed in full in Craig). Note that the trajectory may not pass exactly through the via points.



Validating trajectories for force and torque

A second phase of trajectory verification is usually carried out for commercial robot arms.

Given a trajectory ${}^S_T T(t)$

$$ {}^S_T T(t) : [t_i, t_f] \rightarrow SE(3) $$

1. Compute the link forces and torques for the trajectory specified.
2. Rescale the time access for all joints if necessary to satisfy the torque limitations on the joints.

Generating set point trajectories for controller

All robots require a control system of some form or other.

The most common control architecture for industrial robotic systems is a decoupled joint based PID control architecture.

A joint trajectory profile is used as a time varying reference input to the control design.

Load applied to the end effector and changing mechanical configuration of the manipulator act as disturbances on the set point tracking of the control system.

Even if the joint trajectories are exactly calculated the actual trajectory followed by the robot will depend on the control design and quality of the feedback.

Controller joint trajectories

The joint trajectory profile needs to be supplied to the controller in form that it can manage. Typical formats are:

A sequence of set-points: These set-points are chosen as a finely digitised sequence of points lying along the trajectory. They should not be confused with the way-points used to generate the trajectory - typically a trajectory may have 2-4 way points including initial and final conditions while there are hundreds of set point used to specify the trajectory.

The control system uses the position of the set-points to generate a trajectory profile. Usually, the trajectory time between set-points is set to a constant value. Thus, setting the set-points close together leads to slow precise motion, setting the set-points far apart leads to fast approximate motion.

Parameterised curves: Some controllers accept specific curve parameters. For example the trajectory given by a cubic spline interpolation between two way-points can be specified by the data

$$\{\theta(k), \theta(k+1), (t_{k+1} - t_k), \dot{\theta}(k), \dot{\theta}(k+1)\}$$

Typically, this data would be internally digitised into a sequence of set-points, usually at the sample frequency of the controller and used as the reference signal during the motion.

Cartesian Trajectories

Cartesian trajectories are the best representation of the actual motion of the end effector of the robot.

However, Cartesian motion of the robot does not map trivially to a trajectory in joint space.

The trajectory resulting from a Cartesian path is generally more complex in joint space than a direct interpolation and can lead to high actuation requirements on individual joints.

On the plus side, for Cartesian trajectories the motion of the end effector is usually smooth and natural. Consequently there are reduced inertia and gyroscopic disturbances on the manipulator due to load carried by the end effector.

Smooth Path Generation - Cartesian Space Schemes

Let ${}^S_T T(k)$ for $k = 1, \ldots, N$ denote the initial pose, way-points and final goal. ${}^S_T T(1)$ is the initial pose, ${}^S_T T(N)$ is the final goal.

1. For $k = 1, \ldots, N-1$.

2. Choose a smooth path in $SE(3)$ that connects the way points ${}^S_T T(k)$ and ${}^S_T T(k+1)$
3. Check to verify that the initial and terminal conditions on each path segment $\{k, k+1\}$ match up. That is that there is smooth transition between path segments.
4. Choose a velocity profile along the complete path to produce a trajectory.

Choosing path segments

Specifying a trajectory on $SE(3)$ involves separately specifying the position and attitude of the end effector.

Position: The position coordinate of the $SE(3)$ is specified as a vector form of the interpolation schemes discussed in the joint schemes.

The way points are ξ_k and ξ_{k+1} with initial and final velocities $\dot{\xi}_k$ and $\dot{\xi}_{k+1}$.

Cubic spline interpolation: Set

$$\xi(t) = w_0 + w_1 t + w_2 t^2 + w_3 t^3 \tag{1}$$

Note each unknown $w_i \in \Re^3$ is a vector. Equation 1 is a vector equation in three coordinates.

Solve for the four unknown vectors $\{w_0, \ldots, w_3\}$ using the initial and final conditions.

Linear interpolation with parabolic blend:
Smooth paths can also be specified using linear interpolation. Draw linear interpolation between set points.

1. Choose a transition region and set the boundary conditions.
2. Use a constant acceleration to transition between linear sections.
3. Interpolate using a polynomial function.

The process is entirely analogous to that undertaken for the joint space trajectories.

Choosing path segments for attitude

The attitude path has to be specified as a trajectory on $SO(3)$. This is not a straightforward process since rotation matrices are an over-parameterisation of $SO(3)$.

Can't simply interpolate elements of a rotation matrix because we may produce a non-rotation matrix; remember the special properties of a rotation matrix!

The way points for a segment of path are R_k and R_{k+1} with initial and final velocities Ω_k and Ω_{k+1}.

Cubic interpolation:

STEP I:
Choose a local representation of the rotation matrices.

Let $R(t)$ denote the attitude at a point $\{t\}$ between via points $\{k\}$ and $\{k+1\}$. Thus,

$$R(t) = {}^S_t R$$

where S is the station frame.

Euler angles around R_k. The first stage is to write down the local rotation from $\{k\}$ to $\{t\}$

$${}^k_t R = {}^k_S R {}^S_t R = R_k^T R(t)$$

The second stage is to write ${}^k_t R$ in terms of Euler angles about the reference frame $\{k\}$

$${}^k_t R = R_k^T R(\alpha, \beta, \gamma) R_k$$

Finally, combining these representations one has

$${}^S_t R = R(\alpha, \beta, \gamma) R_k$$

Quaternion representation: Use a Quaternion representation ϵ_k and ϵ_{k+1} of the rotation matrices R_k and R_{k+1}.

STEP II:
Find the initial and terminal set points for the path segment in the local parameterisation of the path. For Euler angles

$$(\alpha_k, \beta_k, \gamma_k) = (0, 0, 0)$$
$$R(\alpha_{k+1}, \beta_{k+1}, \gamma_{k+1}) = R_{k+1} R_k^T$$

STEP III:
If velocity interpolation is required then the initial and terminal angular velocities Ω_k and Ω_{k+1} need to be used to calculate the Euler angle velocities

$$(\dot{\alpha}_k, \dot{\beta}_k, \dot{\gamma}_k), \quad \& \quad (\dot{\alpha}_{k+1}, \dot{\beta}_{k+1}, \dot{\gamma}_{k+1}).$$

this can be done using the velocity Jacobian

STEP IV:
Cubic spline interpolation can now be used on the local coordinate representation of the rotation. The way points are

$$\eta_k = \begin{pmatrix} \alpha_k \\ \beta_k \\ \gamma_k \end{pmatrix}, \eta_{k+1} = \begin{pmatrix} \alpha_{k+1} \\ \beta_{k+1} \\ \gamma_{k+1} \end{pmatrix},$$

with velocities $\dot{\eta}_k$ and $\dot{\eta}_{k+1}$.

Set

$$\eta(t) = u_0 + u_1 t + u_2 t^2 + u_3 t^3$$

Solve for the four unknown vectors $\{w_0, \ldots, w_3\}$ using the initial and final conditions in η.

The rotation obtained is

$$R(t) = R(\eta(t))R_k$$

An analogous approach can be undertaken using the Quaternion representation.

Linear interpolation with parabolic blend:
Smooth paths can also be specified using linear interpolation.

What is a linear interpolation between two attitudes.

What is the relative rotation between $\{k\}$ and $\{k+1\}$

$${}^k_{k+1}R = {}^k_S R^S_{k+1}R = R^T_k R_{k+1}$$

STEP I: Generate an analogy of linear interpolation.
Working relative to the basis $\{k\}$ this rotation can be represented in angle-axis form

$$R({}^k v, \theta_k) = {}^k_{k+1} R = R^T_k R_{k+1}$$

where ${}^k v$ is the rotation axis in $\{k\}$ and θ is the rotation angle.

Setting

$${}^k_t R = R({}^k v, t), \quad 0 \leq t \leq \theta_k$$

is the shortest distance in $SO(3)$ between $\{k\}$ and $\{k+1\}$.

Thus,

$$R(t) = R_k \, {}^k_t R = R_k R({}^k v, t)$$

is the analogy of a linear interpolation between R_k and R_{k+1}.

Steps II and III of linear interpolation and blends requires a representation in which two linear interpolations can be blended at a way point.

Considering that the rotation $R(t)$ is effectively expressed in angle-axis form the best representation to use is the Quaternion representation. Using a Quaternion representation one completes

STEP II:
Choose a transition region and set the boundary conditions.

STEP III:
Use a constant acceleration to transition between linear sections.

in an analogous manner to earlier developments.

Generating set point trajectories for controller

After a Cartesian trajectory

$$ {}^{S}_{t}T \in SE(3), \quad t = [t_i, t_f] $$

is generated it must still be converted into input for the robot controller.

Typically, this trajectory is converted into a sequence of set-points for the manipulator via the inverse kinematics.

Thus, we digitise the time axis

$$ t_s = t_i + s\delta, \quad s = 1, \ldots, N, \quad \delta = \frac{t_f - t_i}{N}. $$

and set way-points

$$ T_s = {}^{S}_{t_s} T = {}^{S}_{t_i + s\delta} T $$

The way-points are typically converted into joint set-points

$$ \theta_s = K^{-1} \left({}^{S}_{t_i + s\delta} T \right), \quad s = 1, \ldots, N, $$

where K^{-1} denotes the inverse kinematics.

Singularities and Reachable Workspace

When Cartesian planning algorithms are used it is always possible that the planned trajectory may pass through singularities, or it may pass outside the manipulators reachable workspace.

In practice one must:

1. Provide sufficient way-points on trajectories to avoid singularities in the manipulator kinematics, and to keep inside the workspace

2. Use joint space trajectory planning for large displacement moves.
3. Use Cartesian planning for local precise movements.

Summary

Moving the robot end effector means generating a trajectory to produce set-points for the controller.

The first step is to identify the start and end points, and any via points if required, in the Cartesian workspace.

Next, we must decide if we want to generate the trajectory in joint space of in the Cartesian workspace.

If decide on joint space, then must map start, end and via points into joint space and generate smooth trajectories between these points.

If decide on Cartesian space, them we apply our smooth trajectory generating schemes directly to the start, and and via points.

Advantages and disadvantages with both approaches; joint space schemes will not produce a "known" trajectory in Cartesian space between via points. However, no problems with with singularities or reachable workspace.

Two main smooth trajectory generating schemes are fitting polynomials and linear interpolation with parabolic blending.

Both are applicable to both joint space, and Cartesian space trajectory generation domains.

Fitting a cubic spline (velocity and acceleration continuous at via points) one of the most popular "polynomial" fitting approaches

Linear interpolation with parabolic blend sees a constant acceleration being applied between the two linear sections intersecting at via points. Simpler than fitting a polynomial, but less precise, eg. may not pass exactly through specified via points.