Task Space Velocity Blending for Real-Time Trajectory Generation

Richard Volpe
The Jet Propulsion Laboratory
California Institute of Technology
Pasadena, California 91109

Abstract

This paper presents a new velocity blending approach to the problem of task space trajectory generation. To compare this technique with others, a generalized formulation for task space trajectory blending is also developed. It is shown that task space velocity blending provides a substantial simplification in both representation and computational complexity over previously proposed methods. While some residual orientation error is incurred by mathematical approximations, it is analytically shown that this error is small and a correction method is provided. Finally, examples are given, our real-time implementation is described, and implementational considerations are addressed.

1 Introduction

Just as manipulator control can be effectively accomplished in joint space or task space, trajectories for the manipulator can also be specified in joint or task space. Typically, the trajectory is specified in the same space in which the controller is working. However, conversion techniques can be used to translated the specified trajectory to the control space. For instance, inverse kinematics applied to a task space trajectory will provide setpoints to a joint space controller. Since task space trajectory specification is usually considered most useful (especially with task space control), the converse translation of a joint space trajectory to task space is uncommon.

Joint space trajectory generation is straightforward since each joint may be treated independently [8, 1, 4]. Typically, motion between specified joint values is dictated with a third, fourth, or fifth order polynomial. Some extension and optimization of this technique have been proposed [2, 14].

Task space trajectory generation has been addressed more extensively, because of the complexity inherent in it. Whitney proposed Resolved Rate control [15] to easily enable straight line motion or constant axis rotation of an end effector. However, this technique does not inherently address extended trajectory generation considerations. Foremost among these is the problem of blending changes in end effector orientation. Paul [8, 10] proposed blending of the *Euler angles* describing the relations of the initial and final frames to the intermediate one. This method blends one orientation to the next, but the path generated is not intuitively obvious. Worse, he proposes changing one Euler angle with a different blend profile from the others. Alternatively, Canny [3] utilizes *quaternions* to describe orientation. However, since he was adressing a different problem (collision detection), he does not discuss the issues of blending the quaternions. Craig [4] utilizes the similar *angle-axis* formulation, but represents the orientation of each via frame with respect to the world frame, not the previous frame as Paul had done. Thus, the blend of orientation parameters will produce a motion path that is dependent on the relation of the via frames to the world frame, not just their relation to each other. Finally, Lloyd and Hayward [6] developed an elegant method for creating variable position blend paths, but do not show an extension of the method for orientations.

As will be seen, Taylor [13] has proposed a scheme that provides smooth, intuitive, and repeatable position and orientation blends. Its major drawback is computational complexity. This paper presents a velocity based method that achieves the same results with a simpler formulation and significantly reduced computation time.

The next section presents the terminology employed for the solution description. Section 3 presents the proposed velocity blending formulation and describes possible blend profile functions. Section 4 quickly discusses position path blending. Orientation blending is extensively discussed in Section 5, where Taylor's method is reviewed, angular velocity blending is presented, and the second order difference between them is analyzed. Finally, Sections 6 and 7 discuss implementational considerations and computational costs associated with the algorithms and show why velocity blending is preferable.

2 Velocity Blending Terminology

A *task frame* is defined as the set containing the rotation matrix that specifies the end effector orientation, $\mathcal{R}$, the end effector position, $\mathbf{p}$, other scalar configuration control parameters (eg. arm angle, ψ [12]), and the transit time to this arm pose, T. Thus,

$$F_i \equiv \{\mathcal{R}_i, \mathbf{p}_i, \psi_i, T_i\} \tag{1}$$

Typically the end effector orientation is specified by a rotation matrix composed of the vectors defining the end effector orientation with respect to the stationary world frame [8].

$$\mathcal{R}_i = \left[\mathbf{n}_i^T, \mathbf{o}_i^T, \mathbf{a}_i^T\right] \tag{2}$$

1050-4729/93 $3.00 © 1993 IEEE

To specify a frame, rotation matrix, or vector with respect to another frame, the former is proceeded with a superscript. For instance, a frame, rotation, or vector with respect to the world frame is denoted by ${}^{w}F, {}^{w}\mathcal{R}, {}^{w}\mathbf{p}$.

In between two sequential frames, the desired linear velocity of the end effector is simply the difference in position over time:

$$v_i = \frac{\Delta \mathbf{p}}{\Delta t} = \frac{\mathbf{p}_i - \mathbf{p}_{i-1}}{T_i} \tag{3}$$

The angular velocity is obtained from the equivalent angle–axis formulation for a rotation from one frame to another [4]:

$$\begin{aligned} \omega_i &= \mathbf{k}_i\ \varphi_i / T_i & (4)\\ \mathbf{k}_i \sin\varphi_i &= \tfrac{1}{2}(\mathbf{n}_{i-1} \times \mathbf{n}_i\ +\ \mathbf{o}_{i-1} \times \mathbf{o}_i\ +\ \mathbf{a}_{i-1} \times \mathbf{a}_i) & (5)\\ \cos\varphi_i &= \tfrac{1}{2}(\mathbf{n}_{i-1} \cdot \mathbf{n}_i\ +\ \mathbf{o}_{i-1} \cdot \mathbf{o}_i\ +\ \mathbf{a}_{i-1} \cdot \mathbf{a}_i\ -\ 1) & (6) \end{aligned}$$

where motion at velocity ω for time t causes a rotation of:

$$\mathcal{R}(\omega\Delta t) = \mathcal{R}(\mathbf{k}, \varphi) =$$
$$\begin{bmatrix} k_x k_x V_\varphi + C_\varphi & k_x k_y V_\varphi - k_z S_\varphi & k_x k_z V_\varphi + k_y S_\varphi \\ k_x k_y V_\varphi + k_z S_\varphi & k_y k_y V_\varphi + C_\varphi & k_y k_z V_\varphi - k_x S_\varphi \\ k_x k_z V_\varphi + k_y S_\varphi & k_y k_z V_\varphi + k_x S_\varphi & k_z k_z V_\varphi + C_\varphi \end{bmatrix} \tag{7}$$

with $S_\varphi = \sin\varphi$, $C_\varphi = \cos\varphi$, and $V_\varphi = 1 - \cos\varphi$.

Finally, the velocity associated with scalar components is calculated as in Equation (3). Therefore, the frame velocity may be defined as:

$$\mathbf{v} \equiv [v, \omega, \dot{\psi}] \tag{8}$$

3 Segment Velocity Blending

To move smoothly from one segment to another, the velocities of the segments must be blended together. To achieve this, many strategies have been suggested [9, 13, 2, 14, 6, 7]. We will review these within a framework that utilizes the following convention:

$$\begin{aligned} \mathbf{v}_a &= \mathbf{v}_i & (9)\\ \mathbf{v}_b &= \mathbf{v}_{i+1} & (10)\\ s &= \frac{t - (t_i - \tau)}{2\tau} & (11)\\ t_i &= \sum_{j=0}^{i} T_j & (12) \end{aligned}$$

where 2τ is the blend period, dependent on the maximum allowed acceleration, as will be shown below. This implies that the normalized time parameter $s \in [0, 1]$.

To smoothly blend from $\mathbf{v}_a$ to $\mathbf{v}_b$ over the interval s, we employ a normalized blending function $f'(s) \in [0, 1]$. Utilizing this function, the velocity profile during the blend is:

$$\begin{aligned} \mathbf{v} &= \mathbf{v}_a(1 - f'(s)) + \mathbf{v}_b f'(s) & (13)\\ &= \mathbf{v}_a + (\mathbf{v}_b - \mathbf{v}_a) f'(s) & (14) \end{aligned}$$

and the acceleration is:

$$\begin{aligned} \mathbf{a} &= (\mathbf{v}_b - \mathbf{v}_a)\frac{df'(s)}{dt} & (15)\\ &= (\mathbf{v}_b - \mathbf{v}_a)\frac{df'(s)}{ds}\frac{1}{2\tau} & (16) \end{aligned}$$

Note that this formulation ensures zero acceleration for $\mathbf{v}_a = \mathbf{v}_b$. Also, there is spatial symmetry of the path for the case of $|\mathbf{v}_a| = |\mathbf{v}_b|$, because the acceleration vector is parallel to the difference of the two velocity vectors, and will therefore bisect them.

If the maximum allowed acceleration is specified, then the blend period may be determined:

$$2\tau = \frac{(\mathbf{v}_b - \mathbf{v}_a)}{|\mathbf{a}|_{max}} \frac{df'(s)}{ds}\bigg|_{s=\frac{1}{2}} \tag{17}$$

assuming that the derivative of $f'(s)$ is a symmetric function with a maximum value at $s = 0.5$.

There are several simple choices available for blend functions. These are provided below, along with the resultant form of the velocity, acceleration, and blend time.

Linear Velocity Blending [13]

$$\begin{aligned} f'(s) &= s & (18)\\ \mathbf{a} &= \frac{\mathbf{v}_b - \mathbf{v}_a}{2\tau} & (19)\\ 2\tau &= \frac{|\mathbf{v}_b - \mathbf{v}_a|}{|\mathbf{a}|_{max}} & (20) \end{aligned}$$

Third Order Polynomial Velocity Blending [9, 2]

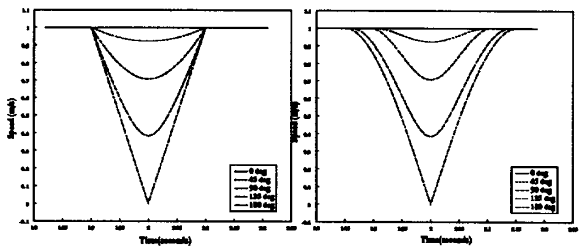
$$\begin{aligned} f'(s) &= -2s^3 + 3s^2 & (21)\\ \mathbf{a} &= \frac{(\mathbf{v}_b - \mathbf{v}_a)}{2\tau}(-6s^2 + 6s) & (22)\\ 2\tau &= \frac{|\mathbf{v}_b - \mathbf{v}_a|}{|\mathbf{a}|_{max}}\frac{3}{2} & (23) \end{aligned}$$

Cycloidal Velocity Blending [7]

$$\begin{aligned} f'(s) &= \sin^2 \tfrac{\pi}{2} s & (24)\\ \mathbf{a} &= \frac{(\mathbf{v}_b - \mathbf{v}_a)}{2\tau}\frac{\pi}{2}\sin \pi s & (25)\\ 2\tau &= \frac{|\mathbf{v}_b - \mathbf{v}_a|}{|\mathbf{a}|_{max}}\frac{\pi}{2} & (26) \end{aligned}$$

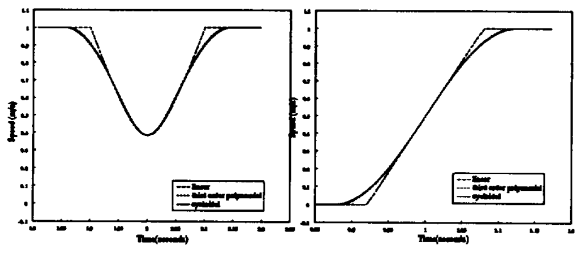
The cyclcoid has a functional form very close to that of the O(3) polynomial, but does not have a discontinuous *jerk* (the derivative of the acceleration). In turn, the O(3) polynomial is superior to the linear form since the latter has discontinuous acceleration (and infinite jerk). The strength of the linear form is that it requires the least time since the acceleration is applied constantly at the maximum value allowed. Finally, note that many other functions are possible; in particular, all odd order polynomials.

Figures 1 show the blend speed versus time for a spectrum of angles (0,45,90,135,180 deg) between the initial and final velocity vectors for the case of $|\mathbf{v}_a| = |\mathbf{v}_b|$, $|\mathbf{a}|_{max} = 10\ \text{m/s}^2$. Figure 1(a) shows the speeds for linear velocity blending. Figure 1(b) shows the speeds for third order polynomial blending. The profiles for cycloidal blending are extremely close to those shown in (b). The cusp in the plot for 180 degrees is due to a change in direction, and does not



(a) Linear (b) O(3) Polynomial

Figure 1: These graphs show the blend speed for a spectrum of angles (0,45,90,135,180 deg) between the initial and final velocities, for the case of $|\mathbf{v}_a| = |\mathbf{v}_b|$. See the text for a discussion.



(a) $|\mathbf{v}_a| = |\mathbf{v}_b|, \mathbf{v}_a \perp \mathbf{v}_b$ (b) $|\mathbf{v}_a| \neq |\mathbf{v}_b|, \mathbf{v}_a || \mathbf{v}_b$

Figure 2: These graphs provide a comparison of linear, third order polynomial, and cycloidal velocity blends. Figure (a) shows a transition between two velocities of equal magnitude at an angle of 135 deg. Figure (b) shows a transition between two velocities of unequal magnitude.

indicate a discontinuity in the acceleration. Also note that when the initial and final velocities are equal the speed is constant across the blend.

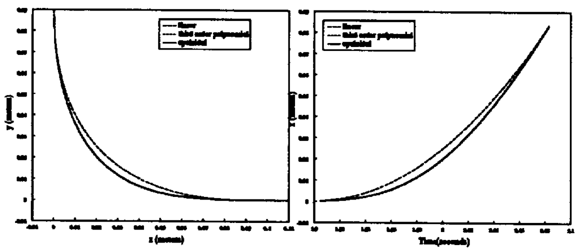
Figures 2 show a comparison of linear, third order polynomial, and cycloidal velocity blends, with $|\mathbf{a}|_{max} = 10$ m/s^2. Figure 2(a) shows the blend speed for a transition between two velocities of equal magnitude at an angle of 135 degrees. Figure 2(b) shows a transition between two velocities of unequal magnitude. In this figure, the initial velocity is zero, however the transition curve has the same shape for two non-zero parallel velocities. Further, Equation (14) shows that this form of the blending occurs for each component of the resultant velocity vector.

4 Blending the Position Trajectory

The blend of the end effector position (p) is described by direct integration of Equation (3). (Scalar quantities are handled in the same way.) This yields:

$$\mathbf{p} = \int v(s)dt = 2\tau \int v(s)ds \quad (27)$$

$$= \mathbf{p}_o + \mathbf{v}_a 2\tau s + (v_b - v_a)2\tau \int f'(s)ds \quad (28)$$



(a) Spatial path (b) Temporal path

Figure 3: These graphs show the spatial and temporal paths for a transition $\mathbf{v}_a \perp \mathbf{v}_b$ and $|\mathbf{v}_a| = |\mathbf{v}_b|$, with $|\mathbf{a}|_{max} = 10$ m/s^2.

$$= \mathbf{p}_o + \mathbf{v}_a 2\tau s + (v_b - v_a)2\tau f(s) \quad (29)$$

where $\mathbf{p}_o$ is the initial position as the blend is entered. The form of the integral of the blend function determines the spatial form traced by the path. For the three blend functions considered, we have:

$$\text{Linear}: \quad f(s) = \tfrac{1}{2}s^2 \quad (30)$$

$$\text{O(3) Polynomial}: \quad f(s) = -\tfrac{1}{2}s^4 + s^3 \quad (31)$$

$$\text{Cycloidal}: \quad f(s) = \tfrac{s}{2} - \tfrac{1}{2\pi}\sin \pi s \quad (32)$$

Equation (30) provides a second order polynomial, and the blend is parabolic. Equation (31) provides a fourth order polynomial, and the blend that is more steep (Higher order even polynomial functions will be increasingly steeper.) The cycloidal blend path remains sinusoidal, but has the addition of a linear term.

Figures 3 show the spatial and temporal paths for a transition between $\mathbf{v}_a$ and $\mathbf{v}_b$, such that $\mathbf{v}_a \perp \mathbf{v}_b$, $|\mathbf{v}_a| = |\mathbf{v}_b|$, with $|\mathbf{a}|_{max} = 10$ m/s^2. It is apparent from Figure 3(a) that tighter cornering can be accomplished with polynomial and cycloidal blending. However, this requires longer blend times (or larger acceleration, and therefore greater joint torques from the actuators). Figure 3(b) shows the positions as a function of time, which are essentially the integrals of the velocities shown in Figure 2(b). The form of these curves also represents the functional form of the position blend functions, Equations (30)–(32).

5 Blending the Orientation

Blending of the orientation is more complicated than position, since the angular velocities are nonholonomic. However, this section shows that a close approximation to analytic orientation blending can be obtained. This requires numeric integration of the rotations obtained from the instantaneous value of the blended angular velocity.

5.1 Rotation Matrix Blending for Orientation

In reference [13] Taylor proposed a method of blending orientation based on rotation matrices. A generalization of this method will be presented here. In this method, the

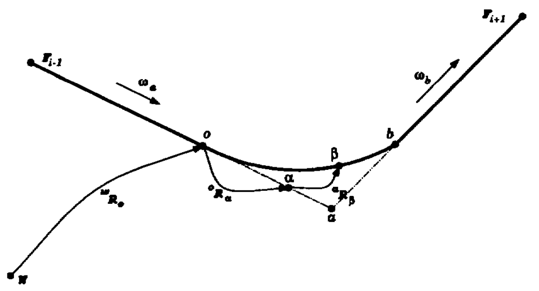


Figure 4: Graphical depiction of the blending described by Equation (33). See the text for a description.

amount of rotation contributed by each rotation matrix is scaled with the previously presented blend functions:

$$
\begin{aligned}
{}^w\mathcal{R}(s) &= {}^w\mathcal{R}_o {}^o\mathcal{R}_a [\omega_a 2\tau(s - f(s))]\, {}^a\mathcal{R}_b [\omega_b 2\tau f(s)] && (33)\\
&= {}^w\mathcal{R}_o {}^o\mathcal{R}_\alpha {}^\alpha\mathcal{R}_\beta && (34)
\end{aligned}
$$

Figure 4 provides a graphical depiction of this blending method. Prior to the blend there is motion away from the orientation of the previous frame, F_{i-1}, and toward the intermediate orientation, $a = F_i$. The constant angular velocity before the blend is ω_a, and the blend begins at orientation o. In this method, for each interval after o a rotation is constructed and applied according to the rotation matrix blending described by Equation (33) or (34). After the normalized blend time s has become unity, the commanded angular velocity will be ω_b, and the commanded orientation is b. After this time, the trajectory continues toward the next target frame, $F_i + 1$, at the constant angular velocity of ω_b. To avoid faceted motion through the blend, the normalized time must be incremented in infinitesimal intervals.

In reference [13], the formulation of this blending scheme is presented with respect to frame F_a, not F_o. This alternate representation can be seen by starting with Equation (33), and utilizing the identity:

$$
\begin{aligned}
{}^w\mathcal{R}_o {}^o\mathcal{R}_\alpha &= {}^w\mathcal{R}_a {}^a\mathcal{R}_o {}^o\mathcal{R}_\alpha && (35)\\
&= {}^w\mathcal{R}_a {}^a\mathcal{R}_o[\omega_a\tau]^o\mathcal{R}_\alpha && (36)\\
&= {}^w\mathcal{R}_a {}^o\mathcal{R}_a[-\omega_a\tau]^o\mathcal{R}_\alpha && (37)
\end{aligned}
$$

we have:

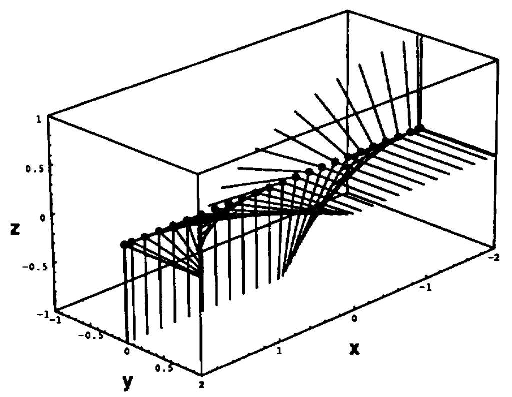
$$
\begin{aligned}
{}^w\mathcal{R}(s) &= {}^w\mathcal{R}_o {}^o\mathcal{R}_a [\omega_a 2\tau(s - f(s))]\, {}^a\mathcal{R}_b [\omega_b 2\tau f(s)] && (38)\\
&= {}^w\mathcal{R}_a {}^o\mathcal{R}_a [-\omega_a\tau]\, {}^o\mathcal{R}_a [\omega_a 2\tau(s - f(s))]\, {}^a\mathcal{R}_b [\omega_b 2\tau f(s)] && (39)\\
&= {}^w\mathcal{R}_a {}^o\mathcal{R}_a \left[\omega_a 2\tau(s - f(s) - \tfrac{1}{2})\right] {}^a\mathcal{R}_b [\omega_b 2\tau f(s)] && (40)
\end{aligned}
$$

Further, reference [13] only considers the linear blend case with $f(s) = \frac{1}{2}s^2$. This gives:

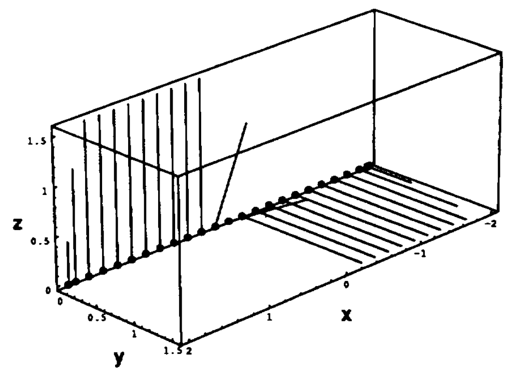
$$
{}^w\mathcal{R}(s) = {}^w\mathcal{R}_a {}^o\mathcal{R}_a \left[-\omega_a\tau(1-s)^2\right] {}^a\mathcal{R}_b \left[\omega_b\tau s^2\right] \qquad (41)
$$

Substituting Equations (4), (11), and (12) yields:

$$
\begin{aligned}
{}^w\mathcal{R}(t) &= {}^w\mathcal{R}_a {}^o\mathcal{R}_a \left[\mathbf{k}_a, -\frac{(\tau-(t-t_a))^2}{4\tau T_a}\varphi_a\right] \cdot \\
&\quad {}^o\mathcal{R}_a \left[\mathbf{k}_b, \frac{(\tau+(t-t_a))^2}{4\tau T_b}\varphi_b\right] && (42)
\end{aligned}
$$



(a) Spatial path of frames.



(b) Angular velocity vectors.

Figure 5: The spatial transition of the target frame and angular velocity vector, during an orientation blend utilizing Equation (33) with linear blending.

This is the form of the rotation blend presented in [13].

Figures 5 provides a graphical depiction of change in the target frame (a) and the direction of the angular velocity vector (b). (A constant spatial velocity has also be used, to spread out the vectors for pictorial clarity.) Figure 6 shows the change in the target frame basis vector **n** components under this transformation.

5.2 Incremental Rotation Blend Components

It is informative to look at the rotations that represent the individual incremental rotation between successive time increments when utilizing Equation (33). Consider the difference between successive frames depicted in Figure (7). The incremental rotation between successive orientations is:

$$
\begin{aligned}
{}^w\mathcal{R}_{\beta'} &= {}^w\mathcal{R}_\beta {}^\beta\mathcal{R}_{\beta'} && (43)\\
{}^\beta\mathcal{R}_{\beta'} &= {}^w\mathcal{R}_\beta^{-1}\, {}^w\mathcal{R}_{\beta'} && (44)\\
&= {}^\alpha\mathcal{R}_\beta^{-1}\, {}^o\mathcal{R}_\alpha^{-1}\, {}^w\mathcal{R}_o^{-1}\, {}^w\mathcal{R}_o\, {}^o\mathcal{R}_{\alpha'}\, {}^{\alpha'}\mathcal{R}_{\beta'} && (45)\\
&= {}^\alpha\mathcal{R}_\beta^{-1}\, {}^o\mathcal{R}_\alpha^{-1}\, {}^o\mathcal{R}_{\alpha'}\, {}^{\alpha'}\mathcal{R}_{\beta'} && (46)
\end{aligned}
$$

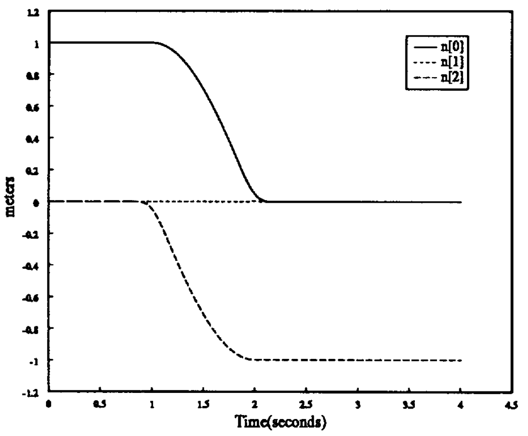


Figure 6: The component values of the unit vector $\mathbf{n}$ of the frame in in Figure 5(a).

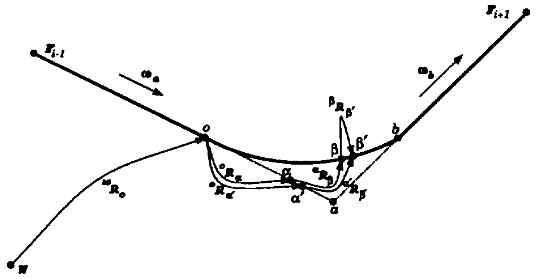


Figure 7: Graphical depiction of the incremental blending described by Equation (44). See the text for a description.

$$\begin{aligned}
&\cong {}^{\alpha}\mathcal{R}_{\beta}^{-1}\,(1+{}^{\alpha}\epsilon_{\alpha'})\,{}^{\alpha'}\mathcal{R}_{\beta'} && (47)\\
&\cong {}^{\alpha}\mathcal{R}_{\beta}^{-1}\,{}^{\alpha'}\mathcal{R}_{\beta'}+{}^{\alpha}\mathcal{R}_{\beta}^{-1}\,{}^{\alpha}\epsilon_{\alpha'}\,{}^{\alpha'}\mathcal{R}_{\beta'} && (48)\\
&\cong (1+{}^{\beta}\epsilon_{\beta'})+{}^{\alpha}\mathcal{R}_{\beta}^{-1}\,{}^{\alpha}\epsilon_{\alpha'}\,{}^{\alpha'}\mathcal{R}_{\beta'} && (49)\\
&\cong 1+{}^{\beta}\epsilon_{\beta'}+{}^{\beta}\epsilon_{\alpha'} && (50)\\
&\cong {}^{\beta}\mathcal{R}[\epsilon_{\beta'}+\epsilon_{\alpha'}] && (51)\\
&\cong {}^{\beta}\mathcal{R}[\omega_b(s_\beta)\Delta s+\omega_a(s_\beta)\Delta s] && (52)\\
&\cong {}^{\beta}\mathcal{R}[\omega(s_\beta)\Delta s] && (53)
\end{aligned}$$

where ϵ is the infinitesimal rotation operator [5]. This result indicates each incremental rotation of Taylor's scheme is equal, to first order, to the rotation provided by the instantaneous angular velocity. This implies that it is possible to blend the angular velocities utilizing Equation (14), and obtain the incremental rotations from the value of the instantaneous angular velocity.

5.3 Angular Velocity Blending for Orientation

As was discussed in the last section, the incremental rotations of an orientation blend may be approximated by uti-

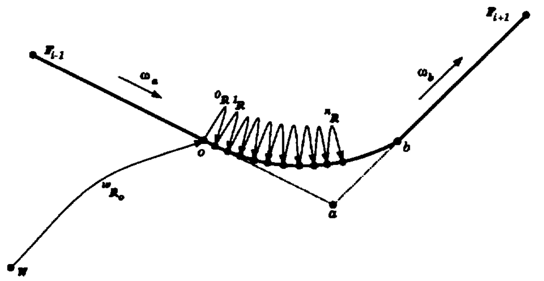


Figure 8: Graphical depiction of the blending described by Equation (54). See the text for a description.

lizing the instantaneous angular velocity provided by Equation (14). Thus, the orientation of the target frame can be computed by utilizing Equations (1), (4), (7), (8), and (14):

$$^{w}\mathcal{R}(s_m) = {}^{w}\mathcal{R}_o \prod_{n=0}^{m} {}^{n}\mathcal{R}\left[\omega(s_n)\Delta s\right] \qquad s_n = n/N, \quad \Delta s = 1/N \tag{54}$$

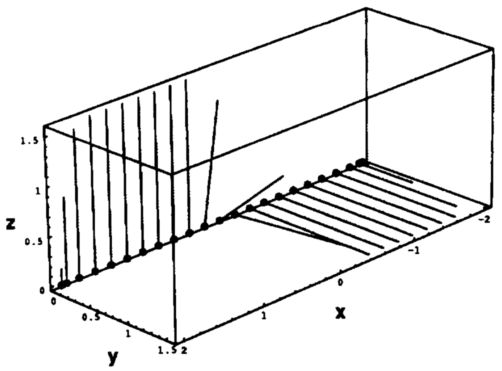
where N is the total number of steps for the complete blend. Figure 8 provides a graphical depiction of this blending method. Before the blend, there is motion away from the orientation of the previous frame, F_{i-1}, and toward the intermediate orientation, $a = F_i$. The constant angular velocity before the blend is ω_a. The blend begins at orientation o. For each interval after o, a rotation is constructed and applied according to the angular velocity blending provided by Equation (14). After the normalized blend time s has become unity, the commanded angular velocity will be ω_b. Ideally, the blend will be complete at the desired orientation, b, where the trajectory continues toward the next target frame, $F_i + 1$.

In practice, velocity-based blending can provide comparable blends to the rotation matrix method described previously. Figures 9 show the spatial transition of the angular velocity vector and the components of $\mathbf{n}$ utilizing third order polynomial angular velocity blending with $|\mathbf{a}|_{max} = 10$ m/s^2. A constant linear velocity is also utilized to spread out the origins of the frames for clarity. Comparing Figures 9 with Figures 5(b) and 6 shows that there is little difference between blending schemes, even when using different blending profiles.

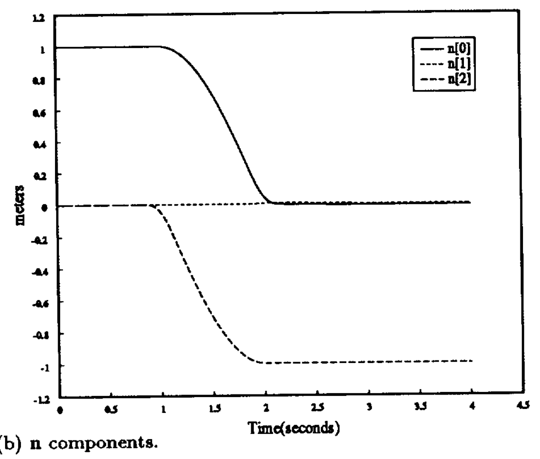
5.4 Compensation for Second Order Error from Angular Velocity Blending

Looking closely at Figure 9(b) it can be seen that there is some small residual error in the first and second components of $\mathbf{n}$ (they should both be zero). This error results from the second order error introduced by the infinitesimal rotation approximation in Section 5.2. This can be understood by considering how the angular velocity blending effects the rotation blending. Consider first the case of total completion of rotation by ω_a, before rotation by ω_b begins. In this case, the resulting rotation is exact:

$$^{o}\mathcal{R}_b = {}^{o}\mathcal{R}_a[\omega_a\tau]\,{}^{a}\mathcal{R}_b[\omega_b\tau] \tag{55}$$



(a) Angular velocity vectors.



(b) n components.

Figure 9: The spatial transition of the angular velocity vector and the changes in the components of n, during an orientation blend utilizing Equation (54) and third order polynomial blending.

$$= {}^o\mathcal{R}_a^1\,{}^o\mathcal{R}_a^2 \cdots {}^o\mathcal{R}_a^{N-1}\,{}^o\mathcal{R}_a^N\,{}^a\mathcal{R}_b^1\,{}^a\mathcal{R}_b^2 \cdots {}^a\mathcal{R}_b^{N-1}\,{}^a\mathcal{R}_b^N \tag{56}$$

where the rotations ${}^o\mathcal{R}_a$ and ${}^a\mathcal{R}_b$ have been divided into N parts. Blending the angular velocities is equivalent to changing the order of some of the rotations at the center of this chain. For instance, utilizing the infinitesimal rotation approximation [5]:

$${}^o\mathcal{R}_b \cong {}^o\mathcal{R}_a^1 \cdots {}^o\mathcal{R}_a^{N-1}\left(1+{}^o\epsilon_a{}^N\right)\left(1+{}^a\epsilon_b{}^1\right){}^a\mathcal{R}_b^2 \cdots {}^a\mathcal{R}_b^N \tag{57}$$

$$\cong {}^o\mathcal{R}_a^1 \cdots {}^o\mathcal{R}_a^{N-1}\left(1+{}^a\epsilon_b{}^1\right)\left(1+{}^o\epsilon_a{}^N\right){}^a\mathcal{R}_b^2 \cdots {}^a\mathcal{R}_b^N \tag{58}$$

This commutation of the infinitesimal rotations may be continued until the proper sequence is attained. However, second order errors arise from the initial approximation of $\mathcal{R} \cong (1+\epsilon)$ and from the disregard of the commutator (the difference between the sequence of the rotations):

$$[1+\epsilon_A, 1+\epsilon_B] = 2\epsilon_A\epsilon_B - 2\epsilon_B\epsilon_A \tag{59}$$

The lack of these second order terms explains the small error introduced by angular velocity based orientation blending.

The change in position of ${}^o\mathcal{R}_a{}^i$ and ${}^a\mathcal{R}_b{}^j$ operators in the sequence is reminiscent of *diffusion*. As the ${}^o\mathcal{R}_a{}^i$ 'diffuse' farther to the right, and the ${}^a\mathcal{R}_b{}^j$ 'diffuse' farther to the left, the changed in orientation becomes more blended. Since the infinitesimal rotations can be represented by their angular velocity equivalents, the diffusion profile is equivalent to the velocity blend profile. For instance, the shape of the cycloidal blend profile in Figure 2(b) indicates more diffusion than the linear one. Further smaller values of $|\mathbf{a}|_{max}$ also imply more diffusion, since they spread out these curves. More diffusion introduces second order error. Therefore, linear blends and high acceleration blends result in less residual error for a given value of $|\mathbf{a}|_{max}$. However, linear blends will result in more error if the blend time is fixed instead of the acceleration. This can be understood by lessening the slope of the linear blend line in Figure 2(b), thus introducing more diffusion.

To provide some quantitative description to this discussion, the following table shows the magnitude of the orientation error for the example previously considered.

blend type	$\lvert\mathbf{a}\rvert_{max} = 10$	$\lvert\mathbf{a}\rvert_{max} = 5$
linear	0.29°	1.16°
O(3) polynomial	0.39°	1.56°
cycloidal	0.41°	1.62°

It is apparent that these errors are small and may be corrected (as described below). Substantially larger errors are not possible since they would require much smaller accelerations which require longer blend times. Too large of a blend time multiplied by ω_a or ω_b would indicate a rotation greater than 180° in the initial or final legs. Such large rotations have been precluded by Equation (5).

While this small error introduced by one blend does not necessarily require compensation, the summation of this error over successive blends may become significant. To compensate for the residual error, we propose the use of a correction term which is calculated at the end of every velocity based blend of orientation. This term is the incremental rotation from the resultant frame to the desired frame at the end of the orientation blend:

$$\mathcal{R}[\mathbf{k}_{cor}, \varphi_{cor}] = \left({}^w\mathcal{R}_a{}^a\mathcal{R}_b[\omega_a\tau]\right)^{-1}\,{}^w\mathcal{R}(s_N = 1) \tag{60}$$

In practice, $\mathbf{k}_{cor}$ and φ_{cor} can be easily calculated by Equations (5) and (6). A correction velocity may then be calculated and applied to the leg of the trajectory being entered, for the time specified to the next via frame:

$$\omega_{cor} = \mathbf{k}_{cor}\varphi_{cor}/(T_{i+1} - \tau_i) \tag{61}$$

This correction term is directly added to the angular velocity ω_b. Since it is very small in magnitude, concerns about changing the value of ω_b have been ignored.

6 Implementation Considerations

6.1 Maximum Acceleration

Since the calculated trajectories are to be executed by real manipulators, the commanded acceleration must be limited to what is achievable. Further, the achievable task space acceleration of the arm depends on the configuration of the robot arm. In different parts of the workspace, different task space accelerations are possible. Therefore, two possibilities exist: 1.) limit all task space accelerations to the worst case acceleration of the arm, or 2.) create a complete map of the achievable task space accelerations, and limit the trajectory blending accordingly. For now we have chosen to work with the first, and simpler, of these two options.

Another consequence of limited acceleration is that it erodes the straight line leg segments of the trajectory between via frames. For a small enough acceleration, one blend will end as another begins. For accelerations smaller than this, one blend would have to begin before another ends. We do not permit this to occur. In this case, the acceleration is increased to the value needed for concatenated blending. If the increased level of acceleration is not achievable by the arm, then the via frames are not reasonably selected.

6.2 Minimum Blend Time

Due to the discrete nature of the computer implementation of these algorithms, it is necessary to specify a minimum number of iterations over which an acceleration is specified. From Equation (17) this quantity is the minimum allowed value of 2τ. If a minimum is not specified, the calculated blend time may become comparable to the algorithm cycle time. Thus, the calculated velocity and position will be discontinuous, providing poor input to the arm controller. We have empirically determined and utilized a minimum value of twenty iterations per blend. A direct consequence of this specification of $2\tau_{min}$ is that the maximum allowed acceleration is also limited. If more acceleration is desired, and the manipulator is capable of it, then $2\tau_{min}$ should be reduced. However, to keep the same number of iterations ber blend, the algorithm rate must be increased proportionally.

6.3 Velocity Summation

To be able to modify commanded trajectories with other control inputs, the commanded variable must be a velocity (a generalized flow variable), not a position [11]. Figure 10 shows our implementation. The trajectory generator is subject to modification by the input of a joystick or a proximity sensor monitor process.

Utilizing the velocity blending scheme proposed in this paper, velocity output is obtained directly. Alternatively, if analytic integration of position is used (as in Equation (29)), or if rotation matrix orientation blending is used (as in Equation (33)), then the velocity must be obtained by differencing the reference frames. As will be seen in the next section, this is computationally costly.

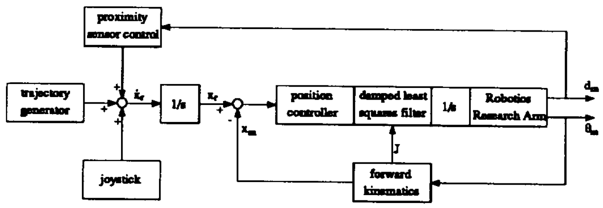


Figure 10: Block diagram of our experimental implementation of the proposed velocity based trajectory blending.

Algorithm Step	Eqns	Ops
Common		
$\mathbf{v}\Delta t = framedif(F_1, F_2) = D()$	1-6, 8	69, 2
$F_2 = frameinc(F_1, \mathbf{v}\Delta t) = I()$	1-4, 7-8	91, 3
calc $f(s)$ or $f'(s)$	14, 29, 33	variable
$\mathbf{v}_2 = vecscale(\mathbf{v}_1, func) = S()$	8	7,0
$\mathbf{a} < \lvert\mathbf{a}\rvert_{max}, \tau > \tau_{min}$	17	variable
Position / Orientation Blending Method		
blend		
calc $f(s)$	30, 32, 31	variable
$\mathbf{v}_\alpha = S(\mathbf{v}_a, s - f(s))$	33	6
$\mathbf{v}_\beta = S(\mathbf{v}_b, f(s))$	33	6
$F_a = I(F_o, \mathbf{v}_\alpha \Delta t)$	33	91,3
$F_b = I(F_a, \mathbf{v}_\beta \Delta t)$	33	91,3
$\mathbf{v} = D(F_o, F_b)/\Delta t_{ob}$	1-6, 8	69, 2
leg		
$F(t) = I(F_i, \{\mathbf{p}(t), \mathbf{k}\varphi(t), \psi(t)\})$	1-4, 7-8	91,3
$\mathbf{v} = D(F(t), F(t-\Delta t))/\Delta t$	1-6, 8	69, 2
transition		
$\mathbf{v}_b = D(F_i, F_{i+1})/T_{i+1}$	1-6,8	69, 2
$\mathbf{a} < \lvert\mathbf{a}\rvert_{max}, \tau > \tau_{min}$	17	variable
Velocity Blending Method		
blend		
calc $f'(s)$	18, 24, 21	variable
$\mathbf{v} = S(\mathbf{v}_a, 1 - f'(s))$	14	6
$\mathbf{v} += S(\mathbf{v}_b, f'(s))$	14	6
leg		
nothing, constant $\mathbf{v} = \mathbf{v}_a$		0,0
transition		
$\mathbf{v}_b = D(F_i, F_{i+1})/T_{i+1}$	1-6, 8	69, 2
$F'_b = I(F_i, \mathbf{v}_b \tau_i)$	60, 61	69, 2
$\mathbf{v}_b += D(F_b, F'_b)/(T_{i+1} - \tau_i)$	60, 61	69, 2
$\mathbf{a} < \lvert\mathbf{a}\rvert_{max}, \tau > \tau_{min}$	17	variable

Figure 11: Algorithm description and comparison. Under the operations column, the values are the number of standard math operations (+ − ∗/) and the number of trigonometric and other math operations (sin,cos,sqrt,etc.).

7 Computational Costs

Table 11 provides an outline of the computational steps and costs for both forms of blending. The equations involved in each step are also summarized. Finally, an estimate of the computational complexity is given by stating the number of additions, subtractions, multiplies, and divides required, as well as the trigonometric (and square root) operations needed.

The top section of the table reviews some common steps needed for both schemes. Of these, the frame differencing and frame incrementing are very costly. The calculation of $f(s)$ or $f'(s)$ is variable since it depends on the blend functions chosen.

The second and third sections of the table show the algorithmic differences between the position/orientation blending and the velocity blending methods. The most striking difference between the two formulations is the reduced computational cost of the velocity blending method. During a blend it requires only 12 operations, while the position/orientation method requires 263 operations plus eight costly trig or square root calls. The situation is much the same during the straight line leg segments of the trajectory, where the velocity based scheme requires zero operations, while a completely position based scheme requires 160 plus 5. The efficiency of the velocity based scheme is paid for by the overhead necessary during the transition from blend to leg segments. At this juncture, the velocity scheme must make 207 plus 6 operations, while the position/orientation scheme requires only 69 plus 2. However, this savings occurs only once for each via frame, compared to the hundred or thousands of iterations that occur for the blend or leg segment computations. Obviously, velocity blending introduces a significant computational savings.

It is important to note that some of the computational advantage of velocity blending is introduced by the assumption that the output of a trajectory generator must be a velocity. The position/orientation scheme must utilize a velocity calculation step during the blend and leg segments which costs 69 plus 2 operations. However, even without this step the velocity blending method is significantly faster. Further, it was shown in the last section why velocity output is necessary.

One other computational burden is introduced to the position/orientation method by the assumption that position, $[\mathbf{p}, \mathbf{k}\varphi, \psi]$, is specified as a function of time during the leg segment. Alternatively, the leg segment velocity could be precomputed and utilized directly as in the velocity blend method. Since $\mathbf{k}$ is constant during the leg segment, no errors would be introduced. Also, the leg velocity must be computed anyway if the maximum acceleration checks are to be performed (as is assumed).

8 Conclusion

This paper has presented a new formulation of trajectory generation based on velocity blending. First, a new formulation for trajectory blending was provided, allowing for the direct comparison and utilization of numerous blend functions. Second, a generalized version of the previously proposed orientation matrix blending formulation was reviewed. Third, it was shown how a first order approximation of this scheme leads directly to angular velocity blending for orientation change. Fourth, the residual error incurred was explained, quantized, and compensated. Finally, the implementational considerations such as acceleration limits, velocity summation requirements, algorithm computation rates and complexity were discussed at length. It was shown that the speed and simplicity of the velocity-blending formulation enable its ease of use in real-time manipulator control. As proof of this, we have implemented it on an Iris workstation for simulation, and on a VME based 68020 microprocessor for control of a 7 DOF Robotics Research Arm.

9 Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government or the Jet Propulsion Laboratory, California Institute of Technology.

References

[1] M. Brady and et al. (editors). *Robot Motion: Planning and Control.* MIT Press, Cambridge, MA, 1982.

[2] J. Canny. Collision Detection for Moving Polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2), March 1986.

[3] J. Craig. *Introduction to Robotics:Mechanics and Control.* Addison–Wesley, Reading, Massachusetts, 1986.

[4] H. Goldstein. *Classical Mechanics.* Addison-Wesley, Reading, Mass., 1980.

[5] C. Lin and P. Chang. Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots. *IEEE Transactions on Automatic Control*, 28(12):1066–1073, 1983.

[6] J. Lloyd and V. Hayward. Real-Time Trajectory Generation Using Blend Functions. In *IEEE International Conference on Robotics and Automation*, Sacramento, California, April 1991.

[7] M. Mujtaba. *Discussion of Trajectory Calculation Methods.* Stanford University, Artificial Intelligence Laboratory, AIM 285.4, 1977.

[8] R. Paul. *Robot Manipulators : Mathematics, Programming and Control.* MIT Press, Cambridge, MA, 1981.

[9] R. Paul. *Manipulator Cartesian Path Control*, pages 245–263. MIT Press, Cambridge, Mass., 1982.

[10] R. Paul and H. Zhang. Robot Motion Trajectory Specification and Generation. In *Second International Symposium on Robotics Research*, Kyoto, Japan, August 1984.

[11] R. Rosenberg and D. Karnopp. *Introduction to Physical System Dynamics.* McGraw-Hill, New York, 1983.

[12] H. Seraji and R. Colbaugh. Improved Configuration Control for Redundant Robots. *Journal of Robotics Systems*, 7(6), 1990.

[13] R. Taylor. *Planning and Execution of Straight Line Manipulator Trajectories*, pages 265–286. MIT Press, Cambridge, Mass., 1982.

[14] S. Thompson and R. Patel. Formulation of Joint Trajectories for Industrial Robots Using B-Splines. *IEEE Transactions on Industrial Electronics*, 34(2):192–199, 1987.

[15] D. Whitney. Resolved Motion Rate Control of Manipulators and Human Protheses. *IEEE Transactions on Man-Machine Systems*, 10(2):49–53, June 1969.