DOI: 10.1111/cgf.14979

Pacific Graphics 2023
R. Chaine, Z. Deng, and M. H. Kim
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 42 (2023), Number 7

Balancing Rotation Minimizing Frames with Additional Objectives

C. Mossman[1], R. H. Bartels[2], and F. F. Samavati[1]

[1]Department of Computer Science, University of Calgary, Canada
[2]Department of Computer Science, University of Waterloo, Canada

Abstract

When moving along 3D curves, one may require local coordinate frames for visited points, such as for animating virtual cameras, controlling robotic motion, or constructing sweep surfaces. Often, consecutive coordinate frames should be similar, avoiding sharp twists. Previous work achieved this goal by using various methods to approximate rotation minimizing frames (RMFs) with respect to a curve's tangent. In this work, we use Householder transformations to construct preliminary tangent-aligned coordinate frames and then optimize these initial frames under the constraint that they remain tangent-aligned. This optimization minimizes the weighted sum of squared distances between selected vectors within the new frames and fixed vectors outside them (such as the axes of previous frames). By selecting different vectors for this objective function, we reproduce existing RMF approximation methods and modify them to consider additional objectives beyond rotation minimization. We also provide some example computer graphics use cases for this new frame tracking.

CCS Concepts

• ***Computing methodologies*** → ***Shape modeling; Procedural animation;*** • ***Mathematics of computing*** → *Mathematical optimization;*

1. Introduction

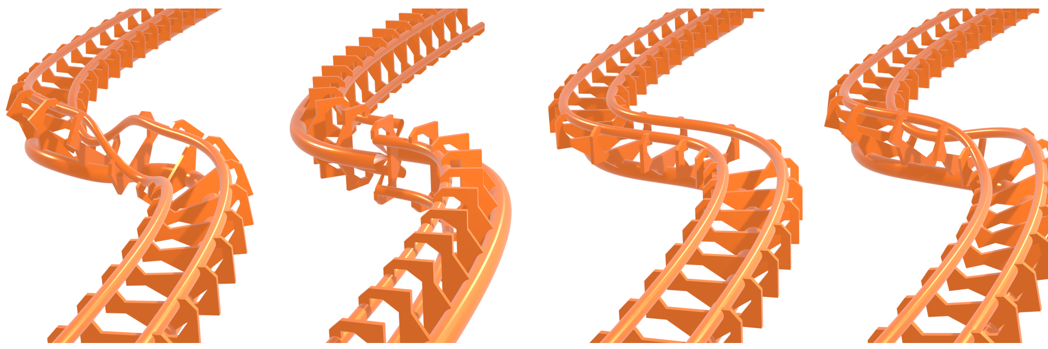
In stepping along a curve, we are interested in the transition of the *orthonormal frame* at some point to the orthonormal frame at a neighboring point. Specifically, for a differentiable curve $\mathrm{P}(u)$, we are interested in using the orthonormal frame at $\mathrm{P}_k = \mathrm{P}(u_k)$ to construct another at P_{k+1}. For our purposes, the orthonormal frame at each point that we visit has the following three orthonormal vectors: a distinguished vector $\mathbf{t}$, of some particular interest to us (usually the curve tangent), and two subsidiary vectors, $\mathbf{r}$ and $\mathbf{s}$, that span the plane perpendicular to $\mathbf{t}$.

Though $\mathbf{t}$ is fixed, unlimited orthonormal choices for $\mathbf{r}$ and $\mathbf{s}$ exist within their plane. In many applications, we would like to choose these vectors so that neighbouring frames are as similar as possible. For example, in sweep surfaces, a 2D cross section is swept along $\mathrm{P}(u)$ and its orientation at each u_k is defined by the respective local frame. If we generate frames in a manner that can create sharp disparities between consecutive ones, as with Frenet-Serret frames, then unappealing surfaces with sharp twists result, as in Figure 1.

One way to increase neighbouring frames' similarity is minimizing the rotation between them. A *rotation minimizing frame* (RMF) with respect to $\mathbf{t}$, also called a *Bishop frame* or *parallel transport frame*, was first proposed by Bishop [Bis75]. In RMFs, every value of u has an associated orthonormal frame $[\mathbf{t}(u), \mathbf{r}(u), \mathbf{s}(u)]$ where the derivative of the vector field $\mathbf{r}(u)$ is parallel to $\mathbf{t}(u)$, and likewise for $\mathbf{s}(u)$. Finding $\mathbf{r}(u)$ and $\mathbf{s}(u)$ exactly requires solving a system of first-order differential equations and, thus, may not always be possible or practical. In this work, we primarily reference a set of works that calculate discrete approximations of RMFs instead. We refer to these as the *projection method* [CW96], *rotation method* [Blo90], and *double reflection method* [WJZL08], to be consistent with the naming used in [WJZL08].

However, some of these approximations, such as the rotation method, become unstable when points are sampled too densely. Additionally, one may sometimes have additional objectives beyond the similarity of neighbouring frames; for example, if one wishes to generate a roller coaster animation for a curve, it may be desirable to not only have the cart's orientation change gradually along the curve but to also have the acceleration felt by passengers point in the direction of its floor as much as possible. These RMF approximation works do not explore how to balance rotation minimization with other constraints or objectives. Finally, approximating RMFs is not the only way to consider "similarity" between neighbouring frames; we propose to minimize some *distance* metric between neighbouring $\mathbf{r}$ and $\mathbf{s}$ vectors instead. By also minimizing distances between other pairs of vectors, this technique enables us to include the additional objectives beyond neighbour similarity.

A high-level visual overview of our work can be found in Figure 2. Our work starts (in Section 4) by exploring the minimization of such a distance metric. The process begins with a frame $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$ at point $\mathrm{P}(u_k) = \mathrm{P}_k$, having unit tangent $\mathbf{t}_k$ at P_k. We construct a following, *trial frame* $\left[\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}\right]$, at point $\mathrm{P}(u_{k+1}) = \mathrm{P}_{k+1}$ where $\tilde{\mathbf{t}}_{k+1}$ is aligned with the unit tangent $\mathbf{t}_{k+1}$; that is,



(a) Acceleration (b) Frenet-Serret (c) RMF (d) Ours

Figure 1: *A comparison of our generalized system compared to other curve tracking options. For a roller coaster track example, frames based on only acceleration (a) and Frenet-Serret frames (b) have sharp twists. Meanwhile, with RMFs (c), passengers' acceleration relative to the cart has no effect on the shape. Our generalized frame tracking (d) allows us to create a "compromise" track between (a) and (c).*

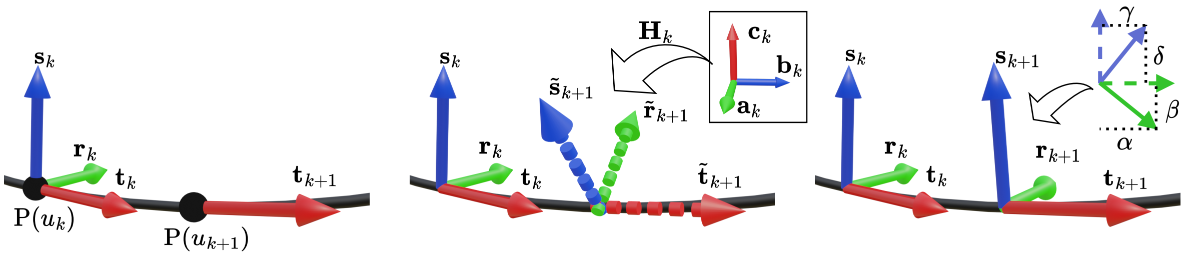


Figure 2: *An overview of our technique. When calculating the frame at* $\mathrm{P}(u_{k+1})$, *we may assume that we know its tangent* $\mathbf{t}_{k+1}$ *and that we know the previous frame* $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$ *at* $\mathrm{P}(u_k)$. *Using reference frame* $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$ *and Householder transformation* $\mathbf{H}_k$ *(justified in Section 3), we construct a trial frame* $\left[\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}\right]$ *aligned with* $\mathbf{t}_{k+1}$, *as discussed in Section 4.1. Then, in an optimization step described in Section 4.2, we find appropriate* α, β, γ, *and* δ *and set* $\mathbf{r}_{k+1} = \alpha\tilde{\mathbf{r}}_{k+1} + \beta\tilde{\mathbf{s}}_{k+1}$ *and* $\mathbf{s}_{k+1} = \gamma\tilde{\mathbf{r}}_{k+1} + \delta\tilde{\mathbf{s}}_{k+1}$. *The effects of* $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$, $\mathbf{H}_k$, α, β, γ, *and* δ *on the handedness of* $[\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}]$ *are discussed in Section 4.3.*

$\tilde{\mathbf{t}}_{k+1} = \pm\mathbf{t}_{k+1}$. We embody the sign chosen for $\tilde{\mathbf{t}}_{k+1}$ by the factor $\sigma = +1$ or $\sigma = -1$, so that $\tilde{\mathbf{t}}_{k+1} = \sigma\mathbf{t}_{k+1}$. As described in Section 4.3, σ determines the handedness of the trial frame, and as described in Section 4.1, changing the sign of σ can quickly produce a more numerically stable trial frame in certain situations. One way to obtain trial frames entails choosing any orthonormal frame $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$ to serve as *reference* and constructing the *Householder matrix* $\mathbf{H}_k$ that maps $\mathbf{c}_k$ onto $\tilde{\mathbf{t}}_{k+1}$ and taking $\mathbf{H}_k\mathbf{a}_k$ as $\tilde{\mathbf{r}}_{k+1}$ and $\mathbf{H}_k\mathbf{b}_k$ as $\tilde{\mathbf{s}}_{k+1}$. Properties of Householder transformations relevant to the construction of these trial frames are discussed in Section 3. It has been shown that, compared to other techniques, Householder transformations produce these orthonormal frames more efficiently without sacrificing numerical robustness or accuracy [LSA13].

From the trial frame, we produce a preferred frame $[\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}]$ at P_{k+1}. The preferred frame's subsidiary vectors are chosen to be orthonormal linear combinations of the $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$ that they replace. If we wish to minimize the shift from $\mathbf{r}_k$ to $\mathbf{r}_{k+1}$ and $\mathbf{s}_k$ to $\mathbf{s}_{k+1}$, we can minimize, for example, $\|\mathbf{r}_{k+1} - \mathbf{r}_k\|_2^2 + \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2$. More generally, we can include other objectives and minimize the weighted squared distance sum $\sum_{i=1}^{n} w_i \|\mathbf{g}_{i,k} - \mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2$, where each $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ is a vector within the u_{k+1} frame (such as $\mathbf{r}_{k+1}$), each $\mathbf{g}_{i,k}$ is a fixed vector outside of it (such as $\mathbf{r}_k$ or something else like acceleration), and $w_i \in \mathbb{R}$ is the weight. Worded another way, we solve a modified form of Wahba's problem [Wah65] where the axis of rotation is predetermined. An example $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ and $\mathbf{g}_{i,k}$ pair is shown in Figure 3. After solving this subproblem, we finish up by setting the correct sign to $\mathbf{t}_{k+1} = \sigma\tilde{\mathbf{t}}_{k+1}$ so that we have the frame $[\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}]$.

The above process enables the following productions, which compose the main contributions of our work:

- New derivations of existing discrete RMF approximation methods (Section 5).
- A more efficient and stable calculation of the rotation method (Section 5.1).
- Optimization of frames along a curve that balances RMF approximation with additional objectives (when considering Section 4 and Section 5 together).

Lastly, we produce sweep surfaces and animations that utilize such optimization, demonstrating this contribution's possibilities when applied to computer graphics (Section 6).

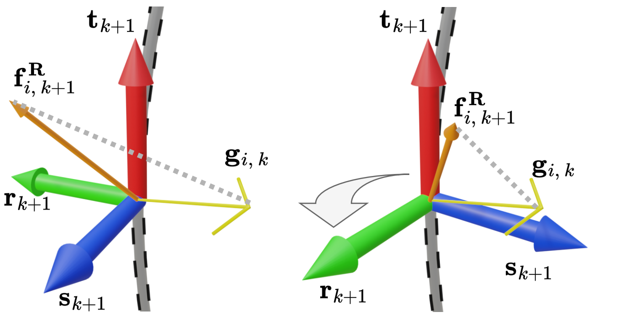


Figure 3: *Our work involves multiple pairs of $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ and $\mathbf{g}_{i,k}$ vectors that we want to minimize the squared distance between. The $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ vectors are part of the current frame we wish to calculate; if the frame were to rotate about the tangent, $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ would as well. Meanwhile, the $\mathbf{g}_{i,k}$ vectors are fixed in world space and would not.*

2. Related works

2.1. RMF approximation

Frequently, smooth tracking of frames along curves utilizes RMFs. Since RMFs are defined in terms of an ordinary differential equation (ODE) [Bis75], such frame tracking may rely on an RMF approximation for cases where it cannot be assumed that the differential equation is solvable, or at least not in the time required. One may approximate the RMF by numerically solving the ODE using the Runge-Kutta method, but as Wang *et al.* describe [WJZL08], this is inefficient and requires a curve's second derivatives as input. This last point is a disadvantage because the RMF can still be defined for curves that are only C^1 rather than C^2 and because other approximation methods require fewer inputs (for example, just the tangents). Another approach is to approximate $\mathrm{P}(u)$ using simpler curves whose RMFs can be computed exactly, such as circular arcs [WJ97] or Pythagorean-Hodograph (PH) curves [JM99].

Discrete RMF approximation methods assign a frame to each of a finite set of points sampled along the curve. Multiple discrete approximation methods exist. In what we refer to as the projection method, Chung and Wang [CW96] minimize the angle between $\mathbf{r}_k$ and $\mathbf{r}_{k+1}$ by projecting $\mathbf{r}_k$ into the plane normal to $\mathbf{t}_{k+1}$; the normalization of the projection becomes $\mathbf{r}_{k+1}$. Meanwhile, the rotation method minimizes the magnitude of the single rotation that transforms the frame at u_k into the frame at u_{k+1} [Blo90]. The axis of the resulting rotation is $\mathbf{t}_k \times \mathbf{t}_{k+1}$, meaning the rotation method is unstable when $\mathbf{t}_k \approx \mathbf{t}_{k+1}$, such as if points are sampled densely or curvature is low. The double reflection method by Wang *et al.* [WJZL08] transforms frames at u_k into the ones at u_{k+1} using two reflections such that, if the curve were a spherical curve, the RMF computation would be exact. The double reflection method has fourth order global approximation error, improving upon the second order error of the projection and rotation methods, and the authors provide a workaround for handling dense sampling. Other discrete approximation methods also exist, such as the *discrete distance minimizing frame* by Chung and Wang [CW96], a technique by Klok that projects sweep surface cross sections along the line segments of a curve's polyline approximation [Klo86], and a method by Krajnc and Vitrih utilizing quintic PH curves for interpolation [KV12]. For the most part, none of these discrete approximation methods provide means to balance RMF approximation with other objectives, though a modification for closed curves to ensure the frames at the endpoints line up is provided in [WJZL08].

There can be multiple ways to derive or calculate the same approximation method. For example, Bischof *et al.* [BGK17] apply the constant step-size backward Euler method [HW10] to the ODE definition of RMFs and derive the projection method as the least-squares solution. Meanwhile, Chung and Wang [CW96] derive the projection method by looking for the method that produces an $\mathbf{r}_{k+1}$ whose angle with $\mathbf{r}_k$ is minimized. Yoon *et al.* [YNS12] reframe the double reflection method to use quaternions instead of reflections. Sometimes, different ways of deriving the approximation methods can reveal relationships between them, such as how Chung and Wang [CW96] derive the rotation method by trying to minimize the smallest angle one can produce when comparing $\cos(\theta)\mathbf{r}_{k+1} + \sin(\theta)\mathbf{s}_{k+1}$ and $\cos(\theta)\mathbf{r}_k + \sin(\theta)\mathbf{s}_k$ for all $\theta \in \mathbb{R}$, which is similar to how they derive the projection method by only considering the angles between $\mathbf{r}$ vectors. Other times, there may be performance benefits; it can be shown that the double reflection method by Wang *et al.* is an alternative way of representing, and more efficient way of calculating, the *global minimum distance intersecting frame* by Chung and Wang [CW96]. Likewise, Poston *et al.* [PFL95] demonstrate how to compute the rotation method using fewer operations than in previous works. In a similar vein, our generalized method provides alternative derivations for multiple discrete approximation methods, relating them together, and allows the rotation method to be computed with greater efficiency and stability than before.

2.2. Non-RMF frame tracking

Other works have also explored ways to track frames along curves without using or approximating RMFs, taking considerations beyond just rotation minimization about the tangent into account. Carroll *et al.* [CKS13] define what they call the Beta frame, whose normal is always parallel to the Frenet-Serret normal (when it exists) but may face the opposite direction to maintain smoothness. Meanwhile, Yılmaz and Turgut [YT10] introduce a frame similar to RMFs but that shares the Frenet-Serret binormal, rather than its tangent, and is rotation-minimizing with respect to the binormal instead of the tangent. Similarly, Farouki and Giannelli [FG09] use frames that are rotation-minimizing with respect to $\mathrm{P}(u)/\|\mathrm{P}(u)\|$ to define smoothly varying camera orientations. Meanwhile, Huang and Ju [HJ16] produce extrinsically smooth direction fields to produce frames for curves and surfaces that balance the goals of smoothness and shape conformity. While these works all offer alternatives or modifications to RMFs, they only take the original curve as input and, beyond that, do not offer any fine-tuned control or customization of the output frames.

2.3. Householder transformations and frame tracking

Previous works have also discussed the role Householder transformations could play in curve tracking. Lopes *et al.* discuss ways to apply Householder transformations to multiple problems in computer graphics and suggest that frame tracking would be another possible use, though they do not demonstrate how this could be done [LSA13]. Wang *et al.* apply Householder transformations directly to $\mathbf{t}_k$ to get $\mathbf{t}_{k+1}$, and these Householder transformations are determined uniquely from the inputs [WJZL08]. Meanwhile, we use Householder transformations to get trial frames and then project onto them; because we are only using the Householder transformations to create a preliminary coordinate frame at each point, we have the freedom to choose between different reference frames, and thus different Householder transformations, when creating them. Details will follow in Sections 3 and 4.

3. Householder transformation preliminaries

As seen in Figure 2, we need to construct orthogonal trial frames aligned with the curve tangents. We can create these frames using Householder transformations (an approach justified by works such as [LSA13]). In this section, we will quickly cover some key properties of Householder transformations that our work relies on in preparation for Section 4.1, where our technique is described in depth.

3.1. Definition and basic properties

A *Householder transformation* can be expressed in matrix format as:

$$\mathbf{H} \equiv \mathbf{I} - \frac{2}{\mathbf{h}^T\mathbf{h}}\mathbf{h}\mathbf{h}^T \tag{1}$$

for identity matrix $\mathbf{I}$ and some vector $\mathbf{h} \neq \mathbf{0}$.

It is immediate from (1) that

1. $\mathbf{H}$ is *symmetric*; that is, $\mathbf{H}^T = \mathbf{H}$.
2. $\mathbf{H}$ is impervious to the *scaling* of $\mathbf{h}$; that is, $\mathbf{h}$ and $\alpha\mathbf{h}$ yield the same $\mathbf{H}$ for any scalar $\alpha \neq 0$.
3. $\mathbf{H}$ is *orthogonal*; that is $\mathbf{H}^T = \mathbf{H}^{-1}$, since $\mathbf{H}^T\mathbf{H} = \mathbf{H}\mathbf{H} = \mathbf{I}$
4. Item 3. guarantees that $\mathbf{H}$ preserves norms and angles.
$$\|\mathbf{v}\| = \sqrt{\mathbf{v}^T\mathbf{v}} = \sqrt{\mathbf{v}^T\mathbf{H}^T\mathbf{H}\mathbf{v}} = \|\mathbf{H}\mathbf{v}\|$$
$$\cos(\theta) = \frac{\mathbf{v}_1^T\mathbf{v}_2}{\|\mathbf{v}_1\|\|\mathbf{v}_2\|} = \frac{\mathbf{v}_1^T\mathbf{H}^T\mathbf{H}\mathbf{v}_2}{\|\mathbf{H}\mathbf{v}_1\|\|\mathbf{H}\mathbf{v}_2\|}$$
Importantly for us, if $[\mathbf{v}_1,\mathbf{v}_2,\mathbf{v}_3]$ is an orthonormal frame, then the frame $[\mathbf{H}\mathbf{v}_1,\mathbf{H}\mathbf{v}_2,\mathbf{H}\mathbf{v}_3]$ will also be orthonormal.
5. Every Householder matrix has a determinant equal to -1.
6. Items 1. 3. and 5. imply that the Householder transformation acts on a cross product of any two vectors $\mathbf{v}$ and $\mathbf{w}$ as follows:
$$(\mathbf{H}\mathbf{v} \times \mathbf{H}\mathbf{w}) = \det(\mathbf{H})\left(\mathbf{H}^T\right)^{-1}(\mathbf{v} \times \mathbf{w}) = -\mathbf{H}(\mathbf{v} \times \mathbf{w}) \tag{2}$$

It can also be seen that Householder transformations act as reflections across the hyperplane that passes through the origin and is orthogonal to $\mathbf{h}$. The double reflection method [WJZL08] utilizes Householder transformations to perform its reflections.

3.2. Formats and computational analysis

Instead of explicitly forming and storing the matrix $\mathbf{H}$, it is more efficient in storage space and computation time if we merely need to apply $\mathbf{H}$ to one or more vectors, $\mathbf{w}$, in the following way:

$$\mathbf{H}\mathbf{w} \equiv \mathbf{w} - (\eta\mathbf{h}^T\mathbf{w})\mathbf{h} \tag{3}$$

Here we simply form $\eta \leftarrow \frac{2}{\mathbf{h}^T\mathbf{h}}$ and retain it and $\mathbf{h}$ as the essential part of $\mathbf{H}$.

Both (1) and (3) are computationally stable and accurate under standard floating-point arithmetic. This fact has been given in numerous papers in computational linear algebra as a strong recommendation for employing Householder transformations wherever they can be used. Higham provides a comprehensive analysis of the stability and roundoff properties of Householder transformations [Hig02]. Numerous computational uses of Householder transformations can be found in [GL13].

3.3. Mapping

The vector $\mathbf{h}$ can be constructed to *map* a multiple of any given vector $\mathbf{p}$ onto some multiple of any other vector $\mathbf{q}$. In this work, we are primarily interested in the case where $\mathbf{p}$ and $\mathbf{q}$ are unit vectors, since we are working exclusively with frames that are triples of mutually orthogonal unit vectors. With unit vectors, the choice of multiples for both $\mathbf{p}$ and $\mathbf{q}$ simplifies to either $+1$ or -1 alone. In this case:

$$\mathbf{h} = \mathbf{p} - \sigma\mathbf{q} \implies \mathbf{H}\,\mathbf{p} = \sigma\mathbf{q} \tag{4}$$

where σ is either plus or minus 1 and represents the $\mathbf{p}$ and $\mathbf{q}$ multiples divided together onto the right.

4. Tracking

A visual overview of our method appears in Figure 2. Given the frame $[\mathbf{t}_k,\mathbf{r}_k,\mathbf{s}_k]$ at the point $\mathrm{P}_k = \mathrm{P}(u_k)$ of a curve, and having generated the unit tangent $\mathbf{t}_{k+1}$ at $\mathrm{P}_{k+1} = \mathrm{P}(u_{k+1})$, we first establish an orthonormal trial frame $\left[\tilde{\mathbf{t}}_{k+1},\tilde{\mathbf{r}}_{k+1},\tilde{\mathbf{s}}_{k+1}\right]$ at P_{k+1}. Vector $\tilde{\mathbf{t}}_{k+1}$ is aligned with $\mathbf{t}_{k+1}$: $\tilde{\mathbf{t}}_{k+1} = \sigma_k\mathbf{t}_{k+1} = \pm\mathbf{t}_{k+1}$ and the remaining two vectors, $[\tilde{\mathbf{r}}_{k+1},\tilde{\mathbf{s}}_{k+1}]$ are basis vectors for the plane perpendicular to $\mathbf{t}_{k+1}$, hence also to $\tilde{\mathbf{t}}_{k+1}$. The details of constructing a trial frame using a Householder transformation, $\mathbf{H}_k$ employing $\sigma_k = \pm1$, are discussed in subsection 4.1.

We next wish to replace $[\tilde{\mathbf{r}}_{k+1},\tilde{\mathbf{s}}_{k+1}]$ by

$$\begin{aligned}\mathbf{r}_{k+1} &= \alpha_{k+1}\tilde{\mathbf{r}}_{k+1} + \beta_{k+1}\tilde{\mathbf{s}}_{k+1}\\ \mathbf{s}_{k+1} &= \gamma_{k+1}\tilde{\mathbf{r}}_{k+1} + \delta_{k+1}\tilde{\mathbf{s}}_{k+1}\end{aligned} \tag{5}$$

with the coefficients α_{k+1}, β_{k+1}, γ_{k+1} and δ_{k+1} chosen so that $[\mathbf{r}_{k+1},\mathbf{s}_{k+1}]$ are closest in some chosen measure to $[\mathbf{r}_k,\mathbf{s}_k]$, they are constrained to have unit norms, and $\mathbf{r}_{k+1}^T\mathbf{s}_{k+1} = 0$. From now on, we may drop the subscripts on these coefficients when they can be inferred. We discuss the optimization and finding formulas for the coefficients in subsection 4.2.

Another potentially useful condition is to require the frame at u_{k+1} to have the same *handedness* as that at u_k. When $\mathbf{r} \times \mathbf{s} = +\mathbf{t}$, we call the frame *right-handed*, and with a negative sign on $\mathbf{t}$, we call the frame *left-handed*. Subsection 4.3 shows how our choice of reference frame handedness and σ_k, as well as α_{k+1}, β_{k+1}, γ_{k+1} and δ_{k+1}, determine the handedness of $[\mathbf{t}_{k+1},\mathbf{r}_{k+1},\mathbf{s}_{k+1}]$, and we also show how the tracking process can be formulated *a priori* so that the handedness is enforced, automatically and beforehand, to that which is desired. The handedness discussion is also summarized in Figure 5.

4.1. Trial frame

To determine a trial frame where one axis is parallel to $\mathbf{t}_{k+1}$, we first choose any *Reference Frame* $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$ and then apply a Householder transformation to create our trial frame. Compared to other techniques for creating orthonormal frames, using Householder transformations is efficient without sacrificing numerical robustness or accuracy [LSA13]. To produce the Householder transformation, we set $\mathbf{h}_k = \mathbf{c}_k - \sigma_{k+1}\mathbf{t}_{k+1}$, where $\sigma_{k+1} = \pm 1$, and construct $\mathbf{H}_k$ as in (3). The frame $[\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}] = [\mathbf{H}_k\mathbf{c}_k, \mathbf{H}_k\mathbf{a}_k, \mathbf{H}_k\mathbf{b}_k]$ is *aligned* with $\mathbf{t}_{k+1}$ in the sense that $\tilde{\mathbf{t}}_{k+1} = \sigma_k\mathbf{t}_{k+1}$ and $[\tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}]$ are two orthonormal vectors that define/span the plane perpendicular to $\mathbf{t}_{k+1}$.

While the Reference Frame has subscript k, this does not require that one change the Reference Frame with every tracking step. Indeed, to parallelize the computation of trial frames, one could use $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k] \equiv [\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3]$, where $\mathbf{e}_i$ is the i^{th} column of the identity matrix, that is, the i^{th} unit vector, for all k.

Two possible uses of σ or $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$ might be mentioned here. There is always a concern that the vectors $\mathbf{c}_k$ and $\tilde{\mathbf{t}}_{k+1}$ which comprise the vector $\mathbf{h}_k$ that defines the Householder matrix $\mathbf{H}_k$ might be nearly equal, causing a problem with $\mathbf{h}_k \approx \mathbf{0}$ and the division by $\mathbf{h}_k^T\mathbf{h}_k$. In a typical curve tracking use case, where one would not expect $\mathbf{t}_k \approx -\mathbf{t}_{k+1}$, this could likely be avoided by always choosing $\sigma = -1$ and $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k] = [\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$. However, if this choice cannot be used (for example, if computing the different frames in parallel, $\mathbf{r}_k$ and $\mathbf{s}_k$ would be unavailable), then one can instead fix the $\mathbf{h}_k \approx 0$ problem by reversing the sign of σ or calling forth an alternative Reference Frame. Neither action changes the final outcome of the tracking step, since the $\mathbf{r}_{k+1}$ and $\mathbf{s}_{k+1}$ vectors of the Final Frame are always found through our formulas defining α_{k+1}, β_{k+1}, γ_{k+1} and δ_{k+1} as the optimal vectors for the k to $k+1$ step. Additionally, two different reference frames with the same handedness produce trial frames of the same handedness; such frames can be generated, for example, by reversing the sign of $\mathbf{c}_k$ and interchanging $\mathbf{a}_k$ and $\mathbf{b}_k$. However, reversing the sign of σ results in the reversal of trial frame handedness that needs to be corrected subsequently. A more detailed explanation is given in subsection 4.3.

4.2. Optimization

Our next step is to determine coefficients α_{k+1}, β_{k+1}, γ_{k+1} and δ_{k+1} as in (5). To ensure $\mathbf{r}_{k+1}$ and $\mathbf{s}_{k+1}$ are indeed orthonormal, we require that $\alpha^2 + \beta^2 = 1$, $\gamma^2 + \delta^2 = 1$, and $\alpha\gamma + \beta\delta = 0$. With these constraints, if we are given the values of α and β, we only have two possibilities for γ and δ: we may choose $\gamma = -\beta$ and $\delta = +\alpha$, or we may choose $\gamma = +\beta$ and $\delta = -\alpha$. We encode which of these two possibilities for γ and δ we use via the variable $\rho = \pm 1$, so that we may say $\gamma = -\rho\beta$ and $\delta = \rho\alpha$. The choice of ρ influences how the handedness of the trial frame and final frame compare, as discussed in Section 4.3.

We then find the α and β that best minimize the distances between pairs of vectors of interest. Let $n \geq 1$ be the number of vector pairs to consider and let $\mathbf{R}_{k+1} = [\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}]$ represent a matrix that takes a vector expressed in our coordinate frame at u_{k+1} into world space. For $1 \leq i \leq n$, let $\mathbf{f}_{i,k+1} = [f^{\mathbf{t}}_{i,k+1}, f^{\mathbf{r}}_{i,k+1}, f^{\mathbf{s}}_{i,k+1}]^T \in \mathbb{R}^3$ be vectors of interest in our coordinate frame at u_{k+1}, making them fixed relative to said coordinate frame. Meanwhile, let $\mathbf{g}_{i,k} \in \mathbb{R}^3$ be the corresponding target vectors in world space and $w_i \in \mathbb{R}$ the weighting for the corresponding objective. We want to pick the α and β that determine this final frame such that the distances between $\mathbf{R}_{k+1}\mathbf{f}_{i,k+1}$ and corresponding vectors $\mathbf{g}_{i,k} \in \mathbb{R}^3$ are as small as possible. That is, we wish to find the $\mathbf{R}_{k+1}$ (determined by α and β) that minimizes:

$$\sum_{i=1}^{n} w_i \|\mathbf{g}_{i,k} - \mathbf{R}_{k+1}\mathbf{f}_{i,k+1}\|^2 \tag{6}$$

We may note the similarity to Wahba's problem [Wah65]. The only differences are that Wahba's problem states $n \geq 2$, whereas we allow $n = 1$, and that $\mathbf{R}_{k+1}$ can be any rotation matrix in Wahba's problem, whereas in ours the $\mathbf{t}_{k+1}$ axis is fixed.

For convenience, we can define $\mathbf{f}^{\mathbf{R}}_{i,k+1} := \mathbf{R}_{k+1}\mathbf{f}_{i,k+1}$, which turns (6) into:

$$\sum_{i=1}^{n} w_i \|\mathbf{g}_{i,k} - \mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2 \tag{7}$$

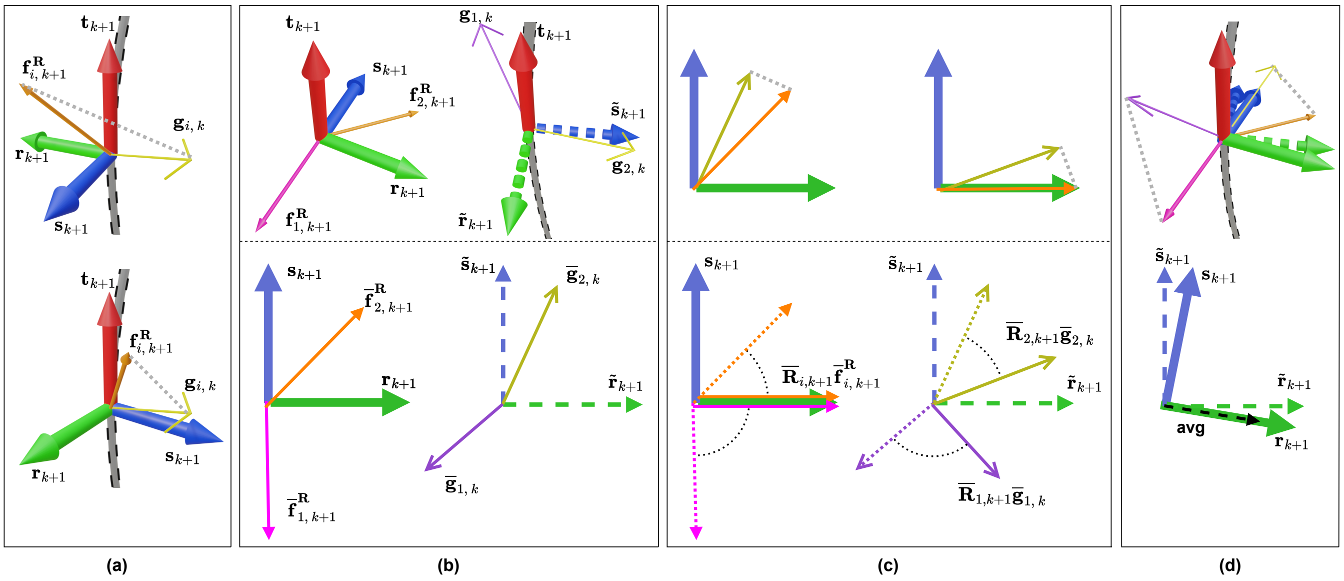
To solve this minimization problem, we observe the following:

$$\begin{aligned}
\text{Error} &= \sum_{i=1}^{n} w_i \|\mathbf{g}_{i,k} - \mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2 \\
&= \sum_{i=1}^{n} w_i \left(\|\mathbf{g}_{i,k}\|^2 - (2\mathbf{f}^{\mathbf{R}}_{i,k+1})^T \mathbf{g}_{i,k} + \|\mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2 \right) \\
&= \sum_{i=1}^{n} w_i \left(\|\mathbf{g}_{i,k}\|^2 + \|\mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2 - 2 \begin{bmatrix} f^{\mathbf{t}}_{i,k+1} \\ f^{\mathbf{r}}_{i,k+1} \\ f^{\mathbf{s}}_{i,k+1} \end{bmatrix}^T \begin{bmatrix} \mathbf{t}^T_{k+1} \\ \mathbf{r}^T_{k+1} \\ \mathbf{s}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k} \right) \\
&= \sum_{i=1}^{n} w_i \left(\|\mathbf{g}_{i,k}\|^2 + \|\mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2 - 2 f^{\mathbf{t}}_{i,k+1}\mathbf{t}^T_{k+1}\mathbf{g}_{i,k} \right) \\
&\quad - 2\sum_{i=1}^{n} w_i \left(\begin{bmatrix} f^{\mathbf{r}}_{i,k+1} \\ f^{\mathbf{s}}_{i,k+1} \end{bmatrix}^T \begin{bmatrix} \alpha & \beta \\ -\rho\beta & \rho\alpha \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{r}}^T_{k+1} \\ \tilde{\mathbf{s}}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k} \right) \\
&= \ldots - 2\sum_{i=1}^{n} w_i \left(\begin{bmatrix} \alpha \\ \beta \end{bmatrix}^T \begin{bmatrix} f^{\mathbf{r}}_{i,k+1} & \rho f^{\mathbf{s}}_{i,k+1} \\ -\rho f^{\mathbf{s}}_{i,k+1} & f^{\mathbf{r}}_{i,k+1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{r}}^T_{k+1} \\ \tilde{\mathbf{s}}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k} \right) \\
&= \ldots - 2\begin{bmatrix} \alpha \\ \beta \end{bmatrix}^T \sum_{i=1}^{n} w_i \begin{bmatrix} f^{\mathbf{r}}_{i,k+1} & \rho f^{\mathbf{s}}_{i,k+1} \\ -\rho f^{\mathbf{s}}_{i,k+1} & f^{\mathbf{r}}_{i,k+1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{r}}^T_{k+1} \\ \tilde{\mathbf{s}}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k}
\end{aligned}$$

The value of $\sum_{i=1}^{n} w_i \left(\|\mathbf{g}_{i,k}\|^2 + \|\mathbf{f}^{\mathbf{R}}_{i,k+1}\|^2 - 2f^{\mathbf{t}}_{i,k+1}\mathbf{t}^T_{k+1}\mathbf{g}_{i,k} \right)$ is constant with respect to the choice of α and β. Therefore, to minimize the error, we want to maximize:

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix}^T \sum_{i=1}^{n} w_i \begin{bmatrix} f^{\mathbf{r}}_{i,k+1} & \rho f^{\mathbf{s}}_{i,k+1} \\ -\rho f^{\mathbf{s}}_{i,k+1} & f^{\mathbf{r}}_{i,k+1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{r}}^T_{k+1} \\ \tilde{\mathbf{s}}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k} \tag{8}$$

This is the dot product of two 2D vectors. Since $[\alpha, \beta]^T$ is of unit length, this dot product is maximized if $[\alpha, \beta]^T$ points in the same direction as the sum on the right. This means that the solution to



$\mathbf{t}_{k+1}$
$\mathbf{f}^{\mathbf{R}}_{i,k+1}$
$\mathbf{g}_{i,k}$
$\mathbf{r}_{k+1}$
$\mathbf{s}_{k+1}$
$\mathbf{f}^{\mathbf{R}}_{2,k+1}$
$\mathbf{f}^{\mathbf{R}}_{1,k+1}$
$\mathbf{g}_{1,k}$
$\mathbf{g}_{2,k}$
$\tilde{\mathbf{s}}_{k+1}$
$\tilde{\mathbf{r}}_{k+1}$
$\bar{\mathbf{f}}^{\mathbf{R}}_{2,k+1}$
$\bar{\mathbf{f}}^{\mathbf{R}}_{1,k+1}$
$\bar{\mathbf{g}}_{2,k}$
$\bar{\mathbf{g}}_{1,k}$
$\overline{\mathbf{R}}_{i,k+1}\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}$
$\overline{\mathbf{R}}_{2,k+1}\bar{\mathbf{g}}_{2,k}$
$\overline{\mathbf{R}}_{1,k+1}\bar{\mathbf{g}}_{1,k}$
avg
(a)
(b)
(c)
(d)

Figure 4: *A visualization of the optimization step. In (a), we see how frame choice affects distances between $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ and $\mathbf{g}_{i,k}$ vectors; we want to minimize the squared sum of these distances. In (b), we consider a specific example; on the right are values whose global positions are already known and on the left are vectors whose positions will depend on the final calculated frame. Because the distance between $\mathbf{g}_{i,k}$ and $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ along $\mathbf{t}_{k+1}$ cannot change, we can project these vectors into the plane spanned by $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$ and work with corresponding 2D vectors $\bar{\mathbf{g}}_{i,k}$ and $\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}$. For simplicity, in this example, we choose $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ such that $\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}$ is of unit length and we assume that our final and trial frames have the same handedness. In (c), since jointly rotating a pair of vectors preserves the distance between them, we can rotate $\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}$ by $\overline{\mathbf{R}}_{i,k+1}$ such that $\overline{\mathbf{R}}_{i,k+1}\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}$ is aligned with the $\mathbf{r}_{k+1}$ axis and rotate $\bar{\mathbf{g}}_{i,k}$ by the same $\overline{\mathbf{R}}_{i,k+1}$. The minimization problem then becomes the problem of finding $\mathbf{r}_{k+1}$ in the coordinate frame with axes $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$ closest to these $\overline{\mathbf{R}}_{i,k+1}\bar{\mathbf{g}}_{i,k}$. The solution is the unit length normalization of the weighted average of the $\overline{\mathbf{R}}_{i,k+1}\bar{\mathbf{g}}_{i,k}$, as shown in (d).*

the generalized form is the normalized weighted average of the **un-normalized** individual solutions for each i.

To better visualize how this method works, as in Figure 4, we can observe the following. Let us denote $[\tilde{\mathbf{r}}^T_{k+1}, \tilde{\mathbf{s}}^T_{k+1}]^T\mathbf{g}_{i,k}$ by $\bar{\mathbf{g}}_{i,k} \in \mathbb{R}^2$, $[\tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}]^T\mathbf{f}^{\mathbf{R}}_{i,k+1}$ by $\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1} \in \mathbb{R}^2$. and $[\tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}]^T\mathbf{r}_{k+1} = [\alpha, \beta]^T$ by $\overline{\mathbf{r}_{k+1}} \in \mathbb{R}^2$. These represent the projections of $\mathbf{g}_{i,k}$, $\mathbf{f}^{\mathbf{R}}_{i,k+1}$, and $\mathbf{r}_{k+1}$, respectively, into the plane spanned by $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$ using a 2D coordinate system with axes $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$. Because distances along $\mathbf{t}_{k+1}$ are fixed, we can minimize $\sum w_i\|\bar{\mathbf{g}}_{i,k} - \bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}\|^2$ instead of (7). Now, we may observe that:

$$\begin{bmatrix} f^{\mathbf{r}}_{i,k+1} & \rho f^{\mathbf{s}}_{i,k+1} \\ -\rho f^{\mathbf{s}}_{i,k+1} & f^{\mathbf{r}}_{i,k+1} \end{bmatrix} = \overline{\mathbf{R}}_{i,k+1}\left\|\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}\right\| \tag{9}$$

for some 2D rotation matrix $\overline{\mathbf{R}}_{i,k+1}$. Then we can rewrite (8) as:

$$\overline{\mathbf{r}_{k+1}}^T \sum_{i=1}^{n} \overline{\mathbf{R}}_{i,k+1}\|\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}\|\bar{\mathbf{g}}_{i,k} \tag{10}$$

Using steps similar to the ones we used to find (8), one can show that maximizing (10) is the same as minimizing

$$\sum w_i\left\|\left(\overline{\mathbf{R}}_{i,k+1}\left\|\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}\right\|\bar{\mathbf{g}}_{i,k}\right) - \overline{\mathbf{r}_{k+1}}\right\|^2 \tag{11}$$

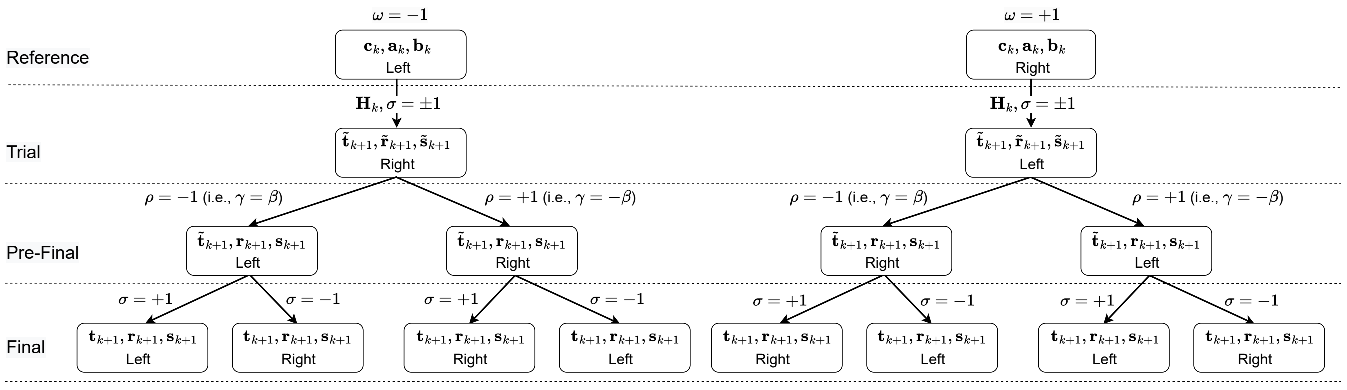
That is, we want to find $\mathbf{r}_{k+1}$ that, in terms of $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$, is as close as possible to our various rotated and scaled $\overline{\mathbf{R}}_{i,k+1}\left\|\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}\right\|\bar{\mathbf{g}}_{i,k}$ vectors. This observation allows us to produce the visualization shown in Figure 4. In the figure, for the sake of simplicity, we only selected $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ such that we have $\left\|\bar{\mathbf{f}}^{\mathbf{R}}_{i,k+1}\right\|^2 = 1$ and we also label $\overline{\mathbf{r}_{k+1}}$ as just $\mathbf{r}_{k+1}$. This figure also ignores the $\rho = -1$ case and how one in general would ensure consistent frame handedness, which is the focus of subsection 4.3 and Figure 5.

4.3. Handedness

In this subsection, we discuss how to determine and control the handedness of final frames. First, we can show that the handedness of the trial frame $\left[\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}\right] = [\mathbf{H}_k\mathbf{c}_k, \mathbf{H}_k\mathbf{a}_k, \mathbf{H}_k\mathbf{b}_k]$ is the opposite of the the handedness of the reference frame $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$. This comes from (2), which shows that $\tilde{\mathbf{r}}_{k+1} \times \tilde{\mathbf{s}}_{k+1} = \mathbf{H}_k\mathbf{a}_k \times \mathbf{H}_k\mathbf{b}_k = -\mathbf{H}_k(\mathbf{a}_k \times \mathbf{b}_k)$. If, similar to our use of ρ and σ, we use the variable $\omega = \pm 1$ to encode the reference frame's handedness, then:

$$\mathbf{a}_k \times \mathbf{b}_k = \omega\mathbf{c}_k \implies \tilde{\mathbf{r}}_{k+1} \times \tilde{\mathbf{s}}_{k+1} = -\omega\tilde{\mathbf{t}}_{k+1} \tag{12}$$

Next comes the impact of ρ. To save space, the below omits the



$\omega = -1$
Reference
$\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k$ Left
$\mathbf{H}_k, \sigma = \pm 1$
Trial
$\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}$ Right
$\rho = -1$ (i.e., $\gamma = \beta$)
$\rho = +1$ (i.e., $\gamma = -\beta$)
Pre-Final
$\tilde{\mathbf{t}}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Left
$\tilde{\mathbf{t}}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Right
$\sigma = +1$
$\sigma = -1$
Final
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Left
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Right
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Right
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Left
$\omega = +1$
$\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k$ Right
$\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}$ Left
$\tilde{\mathbf{t}}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Right
$\tilde{\mathbf{t}}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Left
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Right
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Left
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Left
$\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}$ Right

Figure 5: *Handedness influence of various tracking options, as discussed in Section 4.3.*

subscripts from $\mathbf{r}_{k+1}$, $\mathbf{s}_{k+1}$, $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$. Using properties of the cross product, we may observe that:

$$\mathbf{r} \times \mathbf{s} = (\alpha\tilde{\mathbf{r}} + \beta\tilde{\mathbf{s}}) \times (-\rho\,\beta\,\tilde{\mathbf{r}} + \rho\,\alpha\,\tilde{\mathbf{s}}) = \cdots = \rho(\tilde{\mathbf{r}} \times \tilde{\mathbf{s}}) \tag{13}$$

Thus, the handedness of the pre-final frame $\left[\tilde{\mathbf{t}}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}\right]$ is the same as that of the trial frame $\left[\tilde{\mathbf{t}}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}\right]$ if $\rho = +1$ (that is, $\gamma = -\beta$) and the opposite if $\rho = -1$ (that is, $\gamma = \beta$).

Finally comes the impact of σ. Since $\tilde{\mathbf{t}}_{k+1} = \sigma\mathbf{t}_{k+1}$, it is clear that the handedness of the final frame $[\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}]$ is the same as the handedness of the pre-final frame $\left[\tilde{\mathbf{t}}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}\right]$ if $\sigma = +1$ and the opposite if $\sigma = -1$.

The above is summarized visually in Figure 5. Using our variables σ, ρ, and ω, we can also produce the following summary:

$$\mathbf{r}_{k+1} \times \mathbf{s}_{k+1} = \rho(\tilde{\mathbf{r}}_{k+1} \times \tilde{\mathbf{s}}_{k+1}) = \rho(-\omega)\tilde{\mathbf{t}}_{k+1} = -\rho\,\omega\sigma\mathbf{t}_{k+1} \tag{14}$$

4.4. Pseudocode implementation

We have compiled the details of our frame tracking approach as pseudocode in Algorithm 1. It is important to note two things about this pseudocode. First, as mentioned before and as we will demonstrate explicitly in Section 5, for some values of i, the $\mathbf{g}_{i,k}$ may be functions of $\mathbf{r}_k$ and $\mathbf{s}_k$ rather than constants; this enables RMF approximation. Secondly, the pseudocode covers the most general case; more efficient code is possible in specific cases, such as our rotation method calculation in Section 5.1.

Supplementary material contains an operation cost breakdown.

5. Deriving and calculating discrete RMF approximation methods

With our generalized technique, we can derive multiple preexisting discrete RMF approximation methods. This reveals relationships between them. Additionally, the proof that we can derive the rotation method also showcases a new method of computing it that is more efficient and stable than previous ones.

5.1. Deriving and calculating the rotation method

Let us minimize (7) with $n = 2$, $\mathbf{f}^{\mathbf{R}}_{1,k+1} = \mathbf{r}_{k+1}$, $\mathbf{f}^{\mathbf{R}}_{2,k+1} = \mathbf{s}_{k+1}$, $\mathbf{g}_{1,k} = \mathbf{r}_k$, $\mathbf{g}_{2,k} = \mathbf{s}_k$, and $w_1 = w_2 = 1$.

That is, we wish to minimize:

$$\|\mathbf{r}_{k+1} - \mathbf{r}_k\|_2^2 + \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2 \tag{15}$$

From (8), we can see that

$$\begin{bmatrix}\alpha \\ \beta\end{bmatrix} = \frac{1}{\left\| \begin{bmatrix} \mathbf{r}_k^T\tilde{\mathbf{r}}_{k+1} + \rho\,\mathbf{s}_k^T\tilde{\mathbf{s}}_{k+1} \\ \mathbf{r}_k^T\tilde{\mathbf{s}}_{k+1} - \rho\,\mathbf{s}_k^T\tilde{\mathbf{r}}_{k+1} \end{bmatrix} \right\|_2} \begin{bmatrix} \mathbf{r}_k^T\tilde{\mathbf{r}}_{k+1} + \rho\,\mathbf{s}_k^T\tilde{\mathbf{s}}_{k+1} \\ \mathbf{r}_k^T\tilde{\mathbf{s}}_{k+1} - \rho\,\mathbf{s}_k^T\tilde{\mathbf{r}}_{k+1} \end{bmatrix} \tag{16}$$

We can show that (16) generates the same frames as the rotation method [Blo90], and we can also show how these frames can be calculated in fewer steps and with more stability than in previous works. For this proof, we assume right-handed $[\mathbf{t}, \mathbf{r}, \mathbf{s}]$ frames; the proof for left-handed frames would be nearly identical. Since both methods are coordinate invariant, we can choose to work in a coordinate system where $\mathbf{t}_k$ and $\mathbf{t}_{k+1}$ lie in the xy-plane and $\mathbf{t}_k = [1, 0, 0]^T$. Then

$$\mathbf{r}_k = \begin{bmatrix} 0 \\ \cos(\theta) \\ \sin(\theta) \end{bmatrix}, \quad \mathbf{s}_k = \begin{bmatrix} 0 \\ -\sin(\theta) \\ \cos(\theta) \end{bmatrix}, \quad \mathbf{t}_{k+1} = \begin{bmatrix} \cos(\phi) \\ \sin(\phi) \\ 0 \end{bmatrix}$$

for some angles $\theta, \phi \in \mathbb{R}$.

The rotation method rotates frame k to frame $k+1$ by the smallest angle possible. If $\mathbf{t}_k = \mathbf{t}_{k+1}$ (which, in our setup, happens if $\phi = 0$), then the matrix representing such a rotation is just $\mathbf{I}$. Otherwise, the method calculates an axis of rotation $\frac{\mathbf{t}_k \times \mathbf{t}_{k+1}}{\|\mathbf{t}_k \times \mathbf{t}_{k+1}\|}$ and an angle of rotation $\arccos(\mathbf{t}_k \cdot \mathbf{t}_{k+1})$ and then performs this rotation on $\mathbf{r}_k$ and $\mathbf{s}_k$. In our setup, the axis of rotation is clearly $[0, 0, 1]^T$ and the angle of rotation is $\arccos(\mathbf{t}_k \cdot \mathbf{t}_{k+1}) = \phi + 2\pi m$ for some $m \in \mathbb{Z}$. Therefore, regardless of the value of ϕ used to construct $\mathbf{t}_{k+1}$, we can represent the rotation method with the matrix

$$\begin{bmatrix} \cos(\phi) & -\sin(\phi) & 0 \\ \sin(\phi) & \cos(\phi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{17}$$

Algorithm 1 Frame Tracking

Input: Unit tangent vectors $\mathbf{t}_k$, vectors $\mathbf{f}_{i,k+1}$ and $\mathbf{g}_{i,k}$, and weights w_i for $k = 1,2,\dots,m+1$ and $i = 1,2,\dots,n$. Additionally, a choice of final frame handedness. Finally, $\mathbf{g}_{i,1}$ may rely on an initial frame unit vectors $\mathbf{r}_1$ and $\mathbf{s}_1$, making them inputs as well.

Output: Frames $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$, $k = 1,2,\dots,m+1$

```
 1: for $k$ from 1 to $m$ do
 2:     $\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k = \text{getReferenceFrame}(\dots)$        ▷ Choice of reference frame is flexible (see Section 4.1).
 3:     $\omega = 1$ if $\mathbf{a}_k \times \mathbf{b}_k = \mathbf{c}_k$, else -1        ▷ Stores $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k]$ handedness.
 4:     $\sigma = \text{chooseSigma}(\dots)$        ▷ Either -1 or +1; also flexible (see Section 4.1).
 5:     $\mathbf{h} = \mathbf{c}_k - \sigma \mathbf{t}_{k+1}$        ▷ Determines Householder transformation (i.e., reflection) plane.
 6:     $\tilde{\mathbf{r}}_{k+1} = \mathbf{r}_k - (2/\mathbf{h}^T\mathbf{h})(\mathbf{h}^T\mathbf{r}_k)\mathbf{h}$        ▷ Set trial frame vectors via Householder transformation.
 7:     $\tilde{\mathbf{s}}_{k+1} = \mathbf{s}_k - (2/\mathbf{h}^T\mathbf{h})(\mathbf{h}^T\mathbf{s}_k)\mathbf{h}$
 8:     $\rho = -\omega\sigma$ if final frames should be right-handed, else $\omega\sigma$        ▷ Logic for choice of $\rho$ comes from Section 4.3.
 9:     $\text{sum} = [0,0]^T$
10:     for $i$ from 1 to $n$ do        ▷ Performs summation described in Section 4.2.
11:         $\text{sum} \mathrel{+}= w_i \begin{bmatrix} f^{\mathbf{r}}_{i,k+1} & \rho f^{\mathbf{s}}_{i,k+1} \\ -\rho f^{\mathbf{s}}_{i,k+1} & f^{\mathbf{r}}_{i,k+1} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{r}}^T_{k+1} \\ \tilde{\mathbf{s}}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k}$        ▷ Components of $\mathbf{f}_{i,k+1}$ are denoted $f^{\mathbf{t}}_{i,k+1}$, $f^{\mathbf{r}}_{i,k+1}$, and $f^{\mathbf{s}}_{i,k+1}$.
12:     end for
13:     $[\alpha, \beta]^T = \text{sum}/\|\text{sum}\|$        ▷ Normalize the weighted sum variable.
14:     $\mathbf{r}_{k+1} = \alpha\tilde{\mathbf{r}}_{k+1} + \beta\tilde{\mathbf{s}}_{k+1}$        ▷ We now calculate the final frame vectors using the trial frame ones.
15:     $\mathbf{s}_{k+1} = -\rho\beta\tilde{\mathbf{r}}_{k+1} + \rho\alpha\tilde{\mathbf{s}}_{k+1}$        ▷ Could also calculate $\mathbf{s}_{k+1}$ via cross-product from $\mathbf{t}_{k+1}$ and $\mathbf{r}_{k+1}$.
16: end for
```

Then it is clear that the rotation method generates vectors

$$\mathbf{r}_{k+1} = \begin{bmatrix} -\cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \\ \sin(\theta) \end{bmatrix}, \quad \mathbf{s}_{k+1} = \begin{bmatrix} \sin(\theta)\sin(\phi) \\ -\sin(\theta)\cos(\phi) \\ \cos(\theta) \end{bmatrix} \tag{18}$$

For our approach, we must first choose a reference frame and σ to produce $\tilde{\mathbf{r}}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1}$. Any choice produces the same result (so long as we choose the correct ρ, based on the other choices' impact on handedness). For this discussion, we choose $\sigma = -1$ and a reference frame of $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$ and we will show that these choices result in $\tilde{\mathbf{r}}_{k+1} = \mathbf{r}_{k+1}$ and $\tilde{\mathbf{s}}_{k+1} = \mathbf{s}_{k+1}$, a fact one can use to save computation steps when calculating the rotation method compared to previous works. These choices do not work if $\mathbf{t}_k = -\mathbf{t}_{k+1}$, as then we would have $\mathbf{h} = 0$, but the steps below could in that case be replicated with a different reference frame so that all cases for $\mathbf{t}_{k+1}$ are covered. Our choice of reference frame means that $\mathbf{h} = \mathbf{t}_k - \sigma\mathbf{t}_{k+1} = \mathbf{t}_k + \mathbf{t}_{k+1} = [1+\cos(\phi), \sin(\phi), 0]^T$. Therefore, using (3) and simplifying, we have:

$$\begin{aligned} \tilde{\mathbf{r}}_{k+1} &= \begin{bmatrix} 0 \\ \cos(\theta) \\ \sin(\theta) \end{bmatrix} - \left(\frac{2\cos(\theta)\sin(\phi)}{2+2\cos(\phi)}\right) \begin{bmatrix} 1+\cos(\phi) \\ \sin(\phi) \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ \cos(\theta) \\ \sin(\theta) \end{bmatrix} - \begin{bmatrix} \cos(\theta)\sin(\phi) \\ \cos(\theta)\frac{(1+\cos(\phi))(1-\cos(\phi))}{1+\cos(\phi)} \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} -\cos(\theta)\sin(\phi) \\ \cos(\theta)\cos(\phi) \\ \sin(\theta) \end{bmatrix} \end{aligned}$$

We can see that $\tilde{\mathbf{r}}_{k+1}$ then matches the $\mathbf{r}_{k+1}$ given in (18). A similar set of steps shows that the $\tilde{\mathbf{s}}_{k+1}$ for this reference frame and σ also matches the $\mathbf{s}_{k+1}$ in (18).

Because our reference frame is right-handed and $\sigma = -1$, we know, from Section 4.3 (or the diagram in Figure 5), that we need to use the $\gamma = -\beta$ (that is, the $\rho = +1$) coefficients. By plugging the necessary values into (16) and performing some simple trigonometric simplification, we see that $\alpha = 1, \beta = 0, \gamma = 0, \delta = 1$, meaning $\mathbf{r}_{k+1} = \tilde{\mathbf{r}}_{k+1}$ and $\mathbf{s}_{k+1} = \tilde{\mathbf{s}}_{k+1}$ for this case. Therefore, when using $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k] = [\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$ and $\sigma = -1$, we could omit the last, redundant "projection" step and just use $[\mathbf{t}_{k+1}, \mathbf{r}_{k+1}, \mathbf{s}_{k+1}] = [\mathbf{t}_{k+1}, \tilde{\mathbf{r}}_{k+1}, \tilde{\mathbf{s}}_{k+1}]$ as our final frame. Let us call this the *negation-and-reflection method*.

While our approach and the typical way of computing the rotation method generate the same results in theory, when working with floating-point representations of the inputs, our approach can be more robust when densely sampling a curve. Behaviour under dense sampling is important to consider because, in theory, increasing the number of samples along a continuous curve improves discrete approximation of its RMFs. Specifically, let $u = 0$ to $u = L$ represent the parameter range of a curve and suppose the curve is evenly sampled so that $u_k = (k-1)\ell$ for $k = 1,2,\dots,m+1$ and $\ell = L/m$. Then prior work shows that the double reflection method's error is $\mathcal{O}(\ell^4)$ [WJZL08] and the rotation method's error is $\mathcal{O}(\ell^2)$ [PFL95]. The idea that error would decrease as sampling increases is not surprising. For example, consider using only two samples: the start and end of the curve. This would be highly inaccurate, because many different curves have identical endpoint locations and tangents, but discrete RMF approximation would assign them all one common pair of frames unless more samples are taken.

However, because of finite floating-point precision, the rotation method lacks robustness when $\mathbf{t}_k$ and $\mathbf{t}_{k+1}$ are too similar, such as when sampling a curve densely enough (especially in low curvature areas). If $\mathbf{t}_k \approx \mathbf{t}_{k+1}$, then $\mathbf{t}_k \times \mathbf{t}_{k+1} \approx 0$, making the axis of rotation ambiguous. Meanwhile, if $\mathbf{t}_k \approx \mathbf{t}_{k+1}$, then our method just involves a reflection across span$\{\mathbf{h}\}^{\perp}$, where $\mathbf{h} = \mathbf{t}_k + \mathbf{t}_{k+1} \approx 2\mathbf{t}_k$, which is well-defined.

To observe this, we can construct an example using a Pythagorean-hodograph (PH) curve, for which exact RMFs may be computed via the process in [Far02]. We create a cubic Bézier PH curve from desired endpoints $\mathbf{P}_0 = [0,0,0]^T$ and $\mathbf{P}_3 = [23,266,1]^T$ and respective unit tangents $[0,1,0]$ and $[23/265, 264/265, 0]^T$, chosen so that the curve has low, but still nonzero, curvature for most of its length. We can use [JM99] to find the correct middle control points $\mathbf{P}_1 = [0,2,0]^T$ and $\mathbf{P}_2 = [0,2,1]^T$. To rewrite the curve in the form required by [Far02], we follow [FGS15]. Figure 6 shows how, compared to those generated by the rotation method, frames generated by the negate-and-reflect method for this PH curve are closer to the exact frames found using [Far02] at high resolutions.

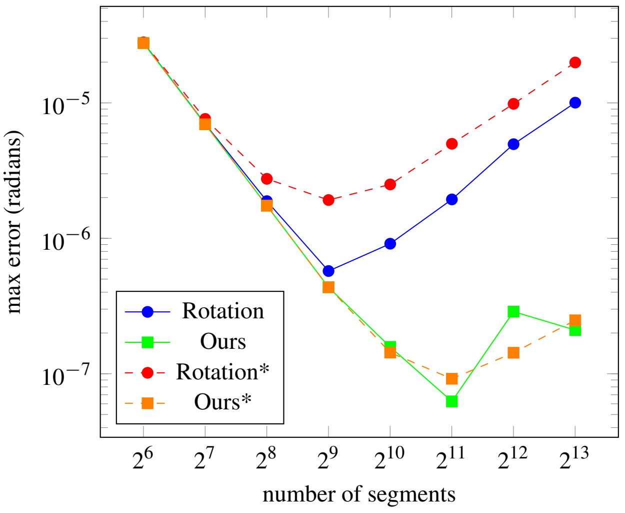


Figure 6: *Results of comparing the negate-and-reflect method (ours) and rotation method for our sample curve. Error represents the max angle (in radians) between exact $\mathbf{r}_k$ generated by [Far02] and approximate $\mathbf{r}_k$ across all $1 \le k \le m+1$, where m is the number of curve segments in our discrete approximation. We used 32-bit floats for our computations. The plot uses "Rotation*" and "Ours*" to show what happens if we take the extra step of normalizing coordinate frame vectors before calculating the next frame. With or without this extra correction, the negate-and-reflect method produces lower error as the number of segments n increases.*

Another benefit of our calculation over previous ones is the number of steps required. An efficient implementation of the rotation method, which we used to generate Figure 6, can be found in [PFL95], and an analysis of this implementation in [WJZL07] shows that, when written as efficiently as possible, it uses 26 additions, 36 multiplications, and one division to compute $\mathbf{r}_{k+1}$ and $\mathbf{s}_{k+1}$. In comparison, if one uses our negation-and-reflection method with $\sigma = -1$ and $[\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k] = [\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$, then our method only requires only 13 additions, 16 multiplications, and one division (one first calculates $\mathbf{r}_{k+1} = \tilde{\mathbf{r}}_{k+1}$ as described in (3) and then computes $\mathbf{s}_{k+1}$ via cross product; this paper's supplementary material has a detailed breakdown). This is summarized in Table 1.

Table 1: *Step counts for the efficient rotation method implementation in [PFL95] versus our negation-and-reflection method.*

	Additions	Multiplications	Divisions
Rotation	26	36	1
Ours	13	16	1

5.2. Deriving the projection method

We have shown that minimizing $\|\mathbf{r}_{k+1} - \mathbf{r}_k\|_2^2 + \|\mathbf{s}_{k+1} - \mathbf{s}_k\|_2^2$ produces the rotation method. If we simply minimize $\|\mathbf{r}_{k+1} - \mathbf{r}_k\|_2^2$ (or, equivalently, minimize $\|\mathbf{r}_{k+1} - \mathbf{r}_k\|_2$), then we produce the projection method. To show this, substitute $n = 1$, $w_1 = 1$, $\mathbf{f}^{\mathbf{R}}_{1,k+1} = \mathbf{r}_{k+1}$, and $\mathbf{g}_{1,k} = \mathbf{r}_k$ into (8).

Note that, in general, there is nothing special about minimizing the distance between $\mathbf{r}$ vectors compared to minimizing the distance between $\mathbf{s}$ vectors. One could choose to instead project $\mathbf{s}_k$ into the plane orthogonal to $\mathbf{t}_{k+1}$ and normalize to obtain $\mathbf{s}_{k+1}$ and then obtain $\mathbf{r}_{k+1}$ via cross product. One could even alternate between the two: for example, one could calculate $\check{\mathbf{r}}$ as the normalized projection of $\mathbf{r}_k$ onto span$\{\mathbf{t}_{k+1}\}^{\perp}$, calculate $\check{\mathbf{s}}$ as the normalized projection of $\mathbf{s}_k$ onto span$\{\mathbf{t}_{k+1}\}^{\perp}$, and then choose to set $\mathbf{r}_{k+1} = \check{\mathbf{r}}$ or $\mathbf{s}_{k+1} = \check{\mathbf{s}}$ based on how $\|\mathbf{r}_k - \check{\mathbf{r}}\|$ and $\|\mathbf{s}_k - \check{\mathbf{s}}\|$ compare.

Chung and Wang [CW96] extend this last idea to derive the rotation method. Rather than just considering the difference between $\mathbf{r}$ and $\mathbf{s}$ vectors, however, they aimed to find the $\mathbf{r}_{k+1}$ and $\mathbf{s}_{k+1}$ that would minimize the smallest possible angle (directly related with distance for angles less than π radians) between any $\cos(\theta)\mathbf{r}_k + \sin(\theta)\mathbf{s}_k$ and $\cos(\theta)\mathbf{r}_{k+1} + \sin(\theta)\mathbf{s}_{k+1}$ in the consecutive frames. The smallest possible angle between such two vectors is the angle between the two planes they would lie in, which is also the angle between $\mathbf{t}_k$ and $\mathbf{t}_{k+1}$, and so the rotation method provides the solution. Our version of the rotation method in (16) can also be seen as extending the projection method to derive the rotation method; there, the final α and β can be interpreted as the normalized weighted average of the α and β we would receive by using the projection method on $\mathbf{r}$ and the projection method on $\mathbf{s}$, with the weights being the lengths of the respective projections.

5.3. Approximation methods that utilize frame origins

All of the RMF approximation methods we have re-derived so far only consider the distances between vectors, but consecutive frames along a curve do not share the same origin; thus, considering the positional offset of the frames in the approximation is also worth considering. Chung and Wang, for example, consider minimizing the distance between $\mathrm{P}(u_k) + \mathbf{r}_k$ and $\mathrm{P}(u_{k+1})$ in what they call their *discrete distance minimizing frame* [CW96]. Using Lagrange multipliers, their solution is the normalized projection of the vector $\mathbf{r}_k - \mathrm{P}(u_{k+1}) + \mathrm{P}(u_k)$ into the span$\{\mathbf{t}_{k+1}\}^{\perp}$ plane. While we omit the steps to show this, as they are straightforward, one can see that our generalized setup provides the same solution when setting $n = 1$, $w_i = 1$, $\mathbf{f}^{\mathbf{R}}_{1,k+1} = \mathbf{r}_{k+1}$, and $\mathbf{g}_{1,k} = \mathbf{r}_k - \mathrm{P}(u_{k+1}) + \mathrm{P}(u_k)$, which, as required, minimizes

$$\|\mathbf{r}_{k+1} - (\mathbf{r}_k - \mathrm{P}(u_{k+1}) + \mathrm{P}(u_k))\|^2$$
$$= \|(\mathrm{P}(u_{k+1}) + \mathbf{r}_{k+1}) - (\mathrm{P}(u_k) + \mathbf{r}_k)\|^2$$

Of course, depending on the application, one may want to minimize multiple such distances. One case that gives an interesting result is minimizing the total squared distance between $\mathrm{P}(u_k) + \mathbf{v}_k$ and $\mathrm{P}(u_{k+1}) + \mathbf{v}_{k+1}$ for the $n = 4$ cases $\mathbf{v} = \pm\mathbf{r}$ and $\mathbf{v} = \pm\mathbf{s}$. In other words, for a sweep surface with cross section vertices $\mathrm{P}(u_k) \pm \mathbf{r}_k$ and $\mathrm{P}(u_k) \pm \mathbf{s}_k$ (four vertices arranged in a square), this would be the sum of the squared lengths of the edges between the cross sections. By plugging the values into Equation (7) and simplifying, one can see that the given error is minimized if and only if the error in Equation (15) is minimized, meaning the same solution as in (16), and thus the rotation method, applies. In fact, the result holds for any polygonal cross-section with rotational symmetry of order four.

Lastly, to a more limited degree, we can relate our method to the double reflection method [WJZL08]. In that method, one computes two Householder transformations, namely $\mathbf{H}_0$, with $\mathbf{h} = \mathrm{P}(u_{k+1}) - \mathrm{P}(u_k)$, and $\mathbf{H}_1$, which maps $\mathbf{H}_0\mathbf{t}_k$ to $\mathbf{t}_{k+1}$. Then one sets $\mathbf{r}_{k+1} = \mathbf{H}_1\mathbf{H}_0\mathbf{r}_k$ and $\mathbf{s}_{k+1} = \mathbf{H}_1\mathbf{H}_0\mathbf{s}_k$. Our negation-and-reflection method from Section 5.1 is identical except for replacing $\mathbf{H}_0$ with a different reflection, one taking $\mathbf{t}_k$ to $\tilde{\mathbf{t}}_{k+1} = \sigma\mathbf{t}_k = -\mathbf{t}_k$ without changing $\mathbf{r}_k$ or $\mathbf{s}_k$, as they lie in the hyperplane that is reflected about. Alternatively, one can say that the double reflection method is the same as the rotation method if $\mathrm{P}(u_{k+1}) - \mathrm{P}(u_k)$ is estimated by $\mathbf{t}_k$.

As for actually calculating the double reflection method, consider an orthonormal frame $[\mathbf{t}_k^L, \mathbf{r}_k^L, \mathbf{s}_k^L] = [-\mathbf{H}_0\mathbf{t}_k, \mathbf{H}_0\mathbf{r}_k, \mathbf{H}_0\mathbf{s}_k]$, which we know has the same handedness as $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$ because of (2). Now suppose one minimizes the following:

$$\|\mathbf{r}_{k+1} - \mathbf{H}_0\mathbf{r}_k\|^2 + \|\mathbf{s}_{k+1} - \mathbf{H}_0\mathbf{s}_k\|^2 \tag{19}$$

We know from Section 5.1 that we can minimize this by finding the Householder transformation $\mathbf{H}'$ that maps $-\mathbf{t}_k^L$ to $\mathbf{t}_{k+1}$ and setting $\mathbf{r}_{k+1} = \mathbf{H}'\mathbf{r}_k^L$ and $\mathbf{s}_{k+1} = \mathbf{H}'\mathbf{s}_k^L$. But since $-\mathbf{t}_k^L = \mathbf{H}_0\mathbf{t}_k$, we can conclude that minimizing (19) yields the same result as the double reflection method. This adjustment of Section 5.1 performs the exact same computations as the original double reflection method, meaning it offers no additional computational stability or efficiency. However, this reframing of double reflection, in addition to the above insight into how it compares to the rotation method, allows augmentation with additional objectives (similar to how we augment the rotation method in Section 6).

6. Example applications to computer graphics

In this section, we demonstrate some sample applications of the generalized frame tracking from Section 4.2 to computer graphics. Because of the ability to balance multiple objectives, and because objectives are provided in a way that would be easy for artists to understand and visualize (vector fields along the curve), we imagine there could be many applications beyond what we showcase here. In all figures referenced in this section, the RMF approximation method that we augment with additional objectives is the rotation method, meaning that frame origins do not impact the calculation as they do with methods like double reflection; we only rely on curve tangents. Therefore, like in (15) for the rotation method, $\mathbf{f}^{\mathbf{R}}_{1,k+1} = \mathbf{r}_{k+1}$, $\mathbf{f}^{\mathbf{R}}_{2,k+1} = \mathbf{s}_{k+1}$, $\mathbf{g}_{1,k} = \mathbf{r}_k$, $\mathbf{g}_{2,k} = \mathbf{s}_k$, and $w_1 = w_2$. Then we include $\mathbf{g}_{3,k}$ as our additional objective, use w_3 as its weight, and set $\mathbf{f}^{\mathbf{R}}_{3,k+1} = \mathbf{s}_{k+1}$ as the in-frame vector to align the additional objective with. More visuals, including animated video, can be found in this paper's supplementary materials.

When simulating a roller coaster, one generally wants to have the cart's floor aligned so that its normal force opposes the gravity and provides the acceleration of the passengers. This way, passengers are pulled down into their seats. However, in artistic situations where exact realism is not required, one might want to smooth out the sharp twists that can result from this choice of normal vector, such as the twist seen in Figure 1a. At the same time, a pure RMF might capture none of the acceleration information, as seen in Figure 1c. Our version in Figure 1d offers a compromise between the two, with the contribution of each factor weighted by the user. Here, $\mathbf{f}^{\mathbf{R}}_{3,k+1} = \mathbf{s}_{k+1}$ and $\mathbf{g}_{3,k}$ is aligned with the direction of passengers' acceleration minus gravity. Acceleration can be calculated numerically by assigning the cart a fixed speed at the highest point of the coaster, using conservation of energy to compute the speed at each point, using the tangent vectors to compute velocities, and then using the velocities to approximate acceleration. A figure showing the effect of different values for the $w_1 = w_2$ influence of the RMF-approximating component can be found in this paper's supplementary material.

In Figure 7 we model a snake as a sweep surface. In this case, we set $\mathbf{f}^{\mathbf{R}}_{3,k+1} = \mathbf{s}_{k+1}$ and $\mathbf{g}_{3,k}$ to point directly upwards in world-space. This allows us to enforce that all parts should generally be ground-level (i.e., that $\mathbf{r}_{k+1}$ should be parallel to the ground plane), which prevents the head from turning upside-down as it would with a pure RMF approximation. We use the same configuration to animate a drone's flight, as seen in Figure 8.

One thing should be noted about closed curves. Currently, our work does not ensure that the first and last frames generated are equivalent. The solution by Wang *et al.* [WJZL08] for pure RMFs – to add the angle difference between the two, in an accumulating fashion, to all frames along the curve so that the last two line up – is not ideal for our proposed applications. For example, along lengthy straight sections of a roller coaster, the track would keep twisting as this difference is distributed gradually along it, whereas the carts could be pointing straight upwards to both align with acceleration and reduce the differences between frames. A workaround for now could be to apply this technique over a segment of the curve where such effect is not noticeable.

7. Future work

Currently, our work offers a solution to a modified version of Wahba's problem where rotation is constrained about a single axis and thus has only one degree of freedom. Meanwhile, in the original Wahba's problem, the rotation being optimized has three degrees of freedom. It would be interesting to consider the case where

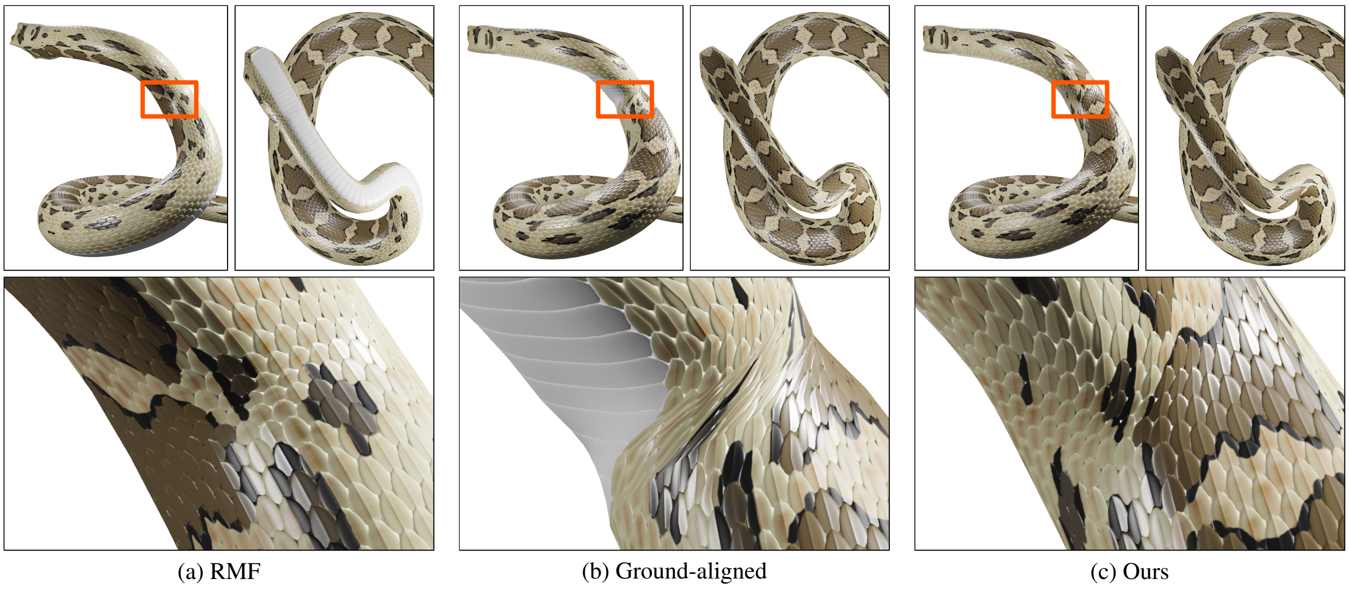

(a) RMF (b) Ground-aligned (c) Ours

Figure 7: *A snake sweep surface example demonstrating three frame tracking options. For each, the top row contains a side view and a birds-eye view of the render, while the bottom row contains a larger view of the twisting region highlighted in the side view. Using only RMF approximation (a), if one chooses the starting frame so that the belly lies on the ground, then the head is upside-down. Meanwhile, if we simply choose the coordinate frames so that one of the axes is parallel to the ground (b), then sharp twists can appear. Our version (c) allows one to balance smoothness with ground alignment.*

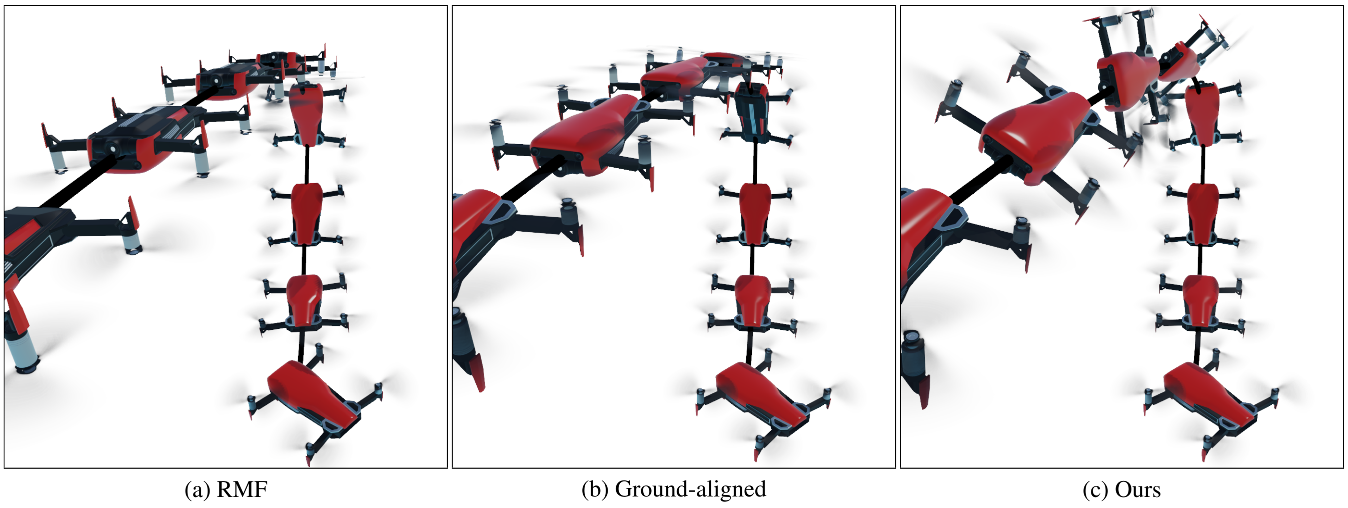

(a) RMF (b) Ground-aligned (c) Ours

Figure 8: *A drone example. Using only RMF approximation (a), the drone flies upside-down for a substantial period. Meanwhile, if we simply choose the coordinate frames so that one of the axes is parallel to the ground (b), the drone's orientation rapidly flips during its climb. Our version (c) smooths out the transition.*

there is exactly two degrees of freedom. Part of the motivation here would be for the sake of completeness, though it could still be relevant to considering different ways that RMF approximations can be derived and calculated, as Chung and Wang derive the rotation and double reflection method by minimizing the distance between a single pair of vectors where each vector has the freedom to rotate about its respective tangent, creating a problem with two degrees of freedom. In comparison, the future work we are proposing would be figuring out whether one can accommodate arbitrary pairs of vectors in these two frames.

In this work, we have focused on optimizing distances only between consecutive frames. However, one may instead be interested in minimizing the general error in (7) globally, as we imagine there may be situations in graphics, robotics, and more where it is this global error, rather than local error, that one wants to minimize. This may also offer us a better approach to handle closed curves in our generalized optimization.

Finally, another possibility for future work is to use the freedom offered by our generalized technique to track coordinate frames across entire surfaces rather than just along curves. As with the issue our current method faces for closed curves, one primary challenge here would be closed surfaces.

8. Conclusion

In this work, we have shown how one can track frames along curves by considering the distances between vectors in consecutive coordinate frames as well as vectors representing other objectives. We also demonstrated how to represent some existing RMF approximation techniques using this method. The benefits of this are relating the techniques together in new ways, deriving a new efficient and stable way to compute the rotation method, and allowing existing RMF approximation methods to be balanced with other objectives by adding other distances to minimize into the objective function. Finally, we have discussed some examples of how the last benefit can be applied to create 3D models and animations.

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC), Alberta Innovates, and Alberta Advanced Education. We wish to thank George Labahn, Director of the Symbolic Computation Group and Professor at the David R. Cheriton School of Computer Science of the University of Waterloo for providing us a copy of `Maple 2018`. We also wish to thank Professor Przemyslaw Prusinkiewicz, Department of Computer Science, University of Calgary, for asking how our work compared to the rotation method, inspiring us to perform the work in Section 5.1. We thank the referees for their helpful suggestions in the review phase. Finally, we thank Lakin Wecker, Mohammad Hameed, Jeffrey Layton, and Andrew Owens (students in the Department of Computer Science, University of Calgary); Camilo Talero (student in the Department of Computer Science, University of British Columbia); and Professor Adam Runions (Department of Computer Science, University of Calgary) for insightful discussion informing the examples in Section 6.

References

[BGK17] Bischof B., Glück T., Kugi A.: Combined path following and compliance control for fully actuated rigid body systems in 3-d space. *IEEE Transactions on Control Systems Technology 25*, 5 (2017), 1750–1760. 3

[Bis75] Bishop R. L.: There is more than one way to frame a curve. *The American Mathematical Monthly 82*, 3 (1975), 246–251. 1, 3

[Blo90] Bloomenthal J.: *Calculation of Reference Frames along a Space Curve*. Academic Press Professional, Inc., USA, 1990, p. 567–571. 1, 3, 7

[CKS13] Carroll D., Köse E., Sterling I.: Improving frenet's frame using bishop's frame. *Journal of Mathematics Research 5* (11 2013). 3

[CW96] Chung K., Wang W.: Discrete moving frames for sweep surface modeling. In *Proceedings of pacific graphics* (1996), vol. 96, pp. 159–173. 1, 3, 9

[Far02] Farouki R. T.: Exact rotation-minimizing frames for spatial pythagorean-hodograph curves. *Graphical Models 64*, 6 (2002), 382–395. 9

[FG09] Farouki R. T., Giannelli C.: Spatial camera orientation control by rotation-minimizing directed frames. *Computer Animation and Virtual Worlds 20*, 4 (2009), 457–472. 3

[FGS15] Farouki R. T., Giannelli C., Sestini A.: Identification and "reverse engineering" of pythagorean-hodograph curves. *Computer Aided Geometric Design 34* (2015), 21–36. 9

[GL13] Golub G. H., Loan C. F. V.: *Matrix Computations*, fourth ed. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013. 4

[Hig02] Higham N. J.: *Accuracy and Stability of Numerical Algorithms*, second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002. 4

[HJ16] Huang Z., Ju T.: Extrinsically smooth direction fields. *Computers & Graphics 58* (2016), 109–117. Shape Modeling International 2016. 3

[HW10] Hairer E., Wanner G.: *Stiff and differential-algebraic problems. Solving ordinary differential equations II.* Springer, Berlin, 2010. 3

[JM99] Jüttler B., Mäurer C.: Cubic pythagorean hodograph spline curves and applications to sweep surface modeling. *Computer-Aided Design 31*, 1 (1999), 73–83. 3, 9

[Klo86] Klok F.: Two moving coordinate frames for sweeping along a 3d trajectory. *Computer Aided Geometric Design 3*, 3 (1986), 217–229. 3

[KV12] Krajnc M., Vitrih V.: Motion design with euler–rodrigues frames of quintic pythagorean-hodograph curves. *Mathematics and Computers in Simulation 82*, 9 (2012), 1696–1711. 3

[LSA13] Lopes D. S., Silva M. T., Ambrósio J. A.: Tangent vectors to a 3-d surface normal: A geometric tool to find orthogonal vectors based on the householder transformation. *Computer-Aided Design 45* (2013), 683–694. 2, 4, 5

[PFL95] Poston T., Fang S., Lawton W.: Computing and approximating sweeping surfaces based on rotation minimizing frames. In *Proceedings of the 4th International Conference on CAD/CG* (Wuhan, China, 1995). 3, 8, 9

[Wah65] Wahba G.: A least squares estimate of satellite attitude. *SIAM Review 7*, 3 (July 1965), 409–409. 2, 5

[WJ97] Wang W., Joe B.: Robust computation of the rotation minimizing frame for sweep surface modeling. *Computer-Aided Design 29*, 5 (1997), 379–391. 3

[WJZL07] Wang W., Jüttler B., Zheng D., Liu Y.: *Computation of Rotation Minimizing Frame in Computer Graphics*. Tech. rep., Department of Computer Science, University of Hong Kong, 07 2007. 9

[WJZL08] Wang W., Jüttler B., Zheng D., Liu Y.: Computation of rotation minimizing frames. *ACM Trans. Graph. 27*, 1 (Mar. 2008), 1–18. 1, 3, 4, 8, 10

[YNS12] Yoon D., Narduzzi M., Shen J.: Computing rotation minimizing frames using quaternions. *Computer-Aided Design and Applications 9*, 5 (2012), 679–690. 3

[YT10] Yilmaz S., Turgut M.: A new version of bishop frame and an application to spherical images. *Journal of Mathematical Analysis and Applications 371*, 2 (2010), 764–776. 3