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 - Supplementary Material

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

1. Computational costs

1.1. Negate-and-reflect

Pseudocode for our efficient rotation method implementation (which we will denote the *negation-and-reflection method*, since these are the two main steps taken) can be found in Algorithm 1. The comment next to each step in the pseudocode breaks down the operation cost of that step. Parentheses are used in one instance to denote order of operations, as a different order would result in a different operation count. Specifically, $(\eta\,\mathbf{h}^T)\,\mathbf{r}_k$ would involve 2 additions and 6 multiplications, whereas $\eta\,(\mathbf{h}^T\,\mathbf{r}_k)$ involves 2 additions and 4 multiplications.

In these calculations, we follow [WJZL07] and treat subtractions as equivalent to additions when assessing operation costs. When doing so, it is clear that every dot product involves three multiplications (one for each component) and two multiplications (for adding the three products together). Similarly, cross product calculation requires six multiplications and three additions. Then, adding up the values in the comments, one can see that we end up with 13 additions, 16 multiplications, and one division in total for each frame that is constructed.

For the cross-product in the last step of Algorithm 1, we assume that the initial frame is right-handed and that all following frames should be as well. Obviously, it would be trivial, and identical in computation cost, to modify the code for the left-handed case.

1.2. General algorithm

A similar step-by-step analysis of the general algorithm, making no assumptions about $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ and $\mathbf{g}_{i,k}$, is given in Algorithm 2. The pseudocode is similar to that in the main paper, though because the purpose of this version is to illustrate computational costs rather than understanding the method, some lines have been altered with this goal in mind. As in our negate-and-reflect analysis, parentheses are used to illustrate the order of operations on lines where multiple options yielding the same final result with different operation counts exist. Because the choice of reference frame and σ are flexible and thus implementation-dependent, we must omit their costs from the analysis. It is possible that choosing the reference frame may require no computation, such as if the same reference frame is used each time or if previous frame $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$ is used as a reference frame. Similar is true for choosing σ. We also do not count multiplication by the variables ω, σ, or ρ, since these are either -1 or $+1$ and therefore the "multiplication" would at most require negation.

In our comments, we also note that $\rho = -\omega\sigma$ (or $\rho = \omega\sigma$) can be computed using bitwise operations. Since $\omega, \sigma \in \{-1, +1\}$, one simply needs to obtain the sign bit from each of the variables, bitwise XOR them together to obtain a new sign, use bitwise OR to apply the sign to the value 1, and possibly negate the final result.

Disregarding the bitwise and unknown computations, we can see that the computation of each frame involves $26+8n$ additions, $38+12n$ multiplications, one division, and one square root. Comparing with Section 1.1, it is clear that significant efficiency can be gained when the nature of the $\mathbf{f}^{\mathbf{R}}_{i,k+1}$ and $\mathbf{g}_{i,k}$ can be assumed rather than kept as generalized as possible.

2. Further visuals

In this section, we include further visuals regarding the sample applications of our generalized frame tracking to computer graphics. As in our main paper, frames are generated to minimize

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

and all examples use $n = 3$, $\mathbf{f}^{\mathbf{R}}_{1,k+1} = \mathbf{r}_{k+1}$, $\mathbf{f}^{\mathbf{R}}_{2,k+1} = \mathbf{s}_{k+1}$, $\mathbf{g}_{1,k} = \mathbf{r}_k$, $\mathbf{g}_{2,k} = \mathbf{s}_k$, and $w_1 = w_2$. This means that $\mathbf{g}_{3,k}$, with a weight of w_3, is used to incorporate our additional objective; we will set $\mathbf{f}^{\mathbf{R}}_{3,k+1} = \mathbf{s}_{k+1}$ as the in-frame vector to align the additional objective with.

In Figure 1, we revisit the roller coaster example. For our roller coaster, we calculate passengers' acceleration minus gravity, so that the cart floor's normal force opposes gravity in addition to accelerating the passenger. We then normalize this vector to obtain its direction and assign the value to $\mathbf{g}_{3,k}$, whose influence is controlled by w_3. In the figure, we can see the effect that different values of $w_1 = w_2$ (the weight of the RMF-approximating vectors) and w_3 (the weight of the acceleration vector) have on the shape of the track. As the weight of the RMF-approximating component relative to the acceleration component increases, the track gradually begins to resemble the result of using only RMF approximation.

Algorithm 1 Negation-and-reflection method (our efficient rotation method implementation)

Input: Unit tangent vectors $\mathbf{t}_k$ for $k = 1,2,\ldots,m$ and initial frame unit vectors $\mathbf{r}_1$ and $\mathbf{s}_1$.
Output: Frames $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$, $k = 1,2,\ldots,m$

for k **from** 1 **to** $m-1$ **do**
  $\mathbf{h} = \mathbf{t}_k + \mathbf{t}_{k+1}$ ▷ 3 additions
  $\eta = 2/(\mathbf{h}^T\mathbf{h})$ ▷ 2 additions, 3 multiplications, 1 division
  $\mathbf{r}_{k+1} = \mathbf{r}_k - (\eta(\mathbf{h}^T\mathbf{r}_k))\mathbf{h}$ ▷ 5 additions, 7 multiplications
  $\mathbf{s}_{k+1} = \mathbf{t}_{k+1} \times \mathbf{r}_{k+1}$ ▷ 3 additions, 6 multiplications
end for

Algorithm 2 Frame Tracking (computational cost breakdown)

Input: Unit tangent vectors $\mathbf{t}_k$, vectors $\mathbf{f}_{i,k+1}$ and $\mathbf{g}_{i,k}$, and weights w_i for $k = 1,2,\ldots,m$ and $i = 1,2,\ldots,n$. 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 $[\mathbf{t}_k, \mathbf{r}_k, \mathbf{s}_k]$, $k = 1,2,\ldots,m$

for k **from** 1 **to** $m-1$ **do**
  $\mathbf{c}_k, \mathbf{a}_k, \mathbf{b}_k = \text{getReferenceFrame}(...)$ ▷ Depends on implementation (may just be lookup)
  $\omega = 1$ **if** $\mathbf{a}_k \times \mathbf{b}_k = \mathbf{c}_k$, **else** -1 ▷ 3 additions, 6 multiplications
  $\sigma = \text{chooseSigma}(...)$ ▷ Depends on implementation (may just be lookup)
  $\mathbf{h} = \mathbf{c}_k - \sigma\mathbf{t}_{k+1}$ ▷ 3 additions
  $\eta = 2/(\mathbf{h}^T\mathbf{h})$ ▷ 2 additions, 3 multiplications, 1 division
  $\tilde{\mathbf{r}}_{k+1} = \mathbf{r}_k - (\eta(\mathbf{h}^T\mathbf{r}_k))\mathbf{h}$ ▷ 5 additions, 7 multiplications
  $\tilde{\mathbf{s}}_{k+1} = \mathbf{s}_k - (\eta(\mathbf{h}^T\mathbf{s}_k))\mathbf{h}$ ▷ 5 additions, 7 multiplications
  $\rho = -\omega\sigma$ **if** final frames should be right-handed, **else** $\omega\sigma$ ▷ Can be done with bitwise operations.
  $\text{sum} = [0,0]^T$
  for i **from** 1 **to** n **do**
    $\text{sum} \mathrel{+}= w_i \left(\begin{bmatrix} f^{\mathbf{r}}_{i,k+1} & \rho f^{\mathbf{s}}_{i,k+1} \\ -\rho f^{\mathbf{s}}_{i,k+1} & f^{\mathbf{r}}_{i,k+1} \end{bmatrix} \left(\begin{bmatrix} \tilde{\mathbf{r}}^T_{k+1} \\ \tilde{\mathbf{s}}^T_{k+1} \end{bmatrix} \mathbf{g}_{i,k} \right) \right)$ ▷ 8 additions, 12 multiplications
  end for
  $\text{sumNorm} = \sqrt{\text{sum}^T\text{sum}}$ ▷ 2 additions, 3 multiplications, 1 sqrt
  $[\alpha,\beta]^T = \text{sum}/\text{sumNorm}$ ▷ 1 division
  $\mathbf{r}_{k+1} = \alpha\tilde{\mathbf{r}}_{k+1} + \beta\tilde{\mathbf{s}}_{k+1}$ ▷ 3 additions, 6 multiplications
  $\mathbf{s}_{k+1} = -\rho\beta\tilde{\mathbf{r}}_{k+1} + \rho\alpha\tilde{\mathbf{s}}_{k+1}$ ▷ 3 additions, 6 multiplications
end for

In Figure 2, we model a road through hilly terrain using a sweep surface. The road has uniform width along its extent and each cross-section normal to the curve is a flat rectangle; the only way the the road changes to accommodate the shape of the hills is in its tilt. Specifically, we set $\mathbf{g}_{3,k}$ to point in the direction that is normal to the nearest point on the hill's mesh. This makes the road "hug" the hill, as if it were draped over it or carved out of it. This is not physically realistic when compared to a real road, where turns are banked to allow the normal force to supply the necessary centripetal acceleration for the turn. However, having the road aligned with the hill normals is common in cartoon images and can yield stylistic, visually pleasing images. As seen in Figure 2, using our generalized frame tracking prevents the road from being upended at various points along its path, which is what happens with a pure RMF, while also lessening sharp twists that occur if we were to only rely on the hill's surface normals.

References

[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. 1