Computation of the Upper Bound of Motion

Xinyu Zhang

October 6, 2006

The derivation of μ on page 5 in the paper (Eq. 5) is incorrect. In this document, we give a correct derivation and re-measure the CCD query performance for all the benchmarks based on the corrected derivation.

Motion Projection

Given the interpolating motion $\mathbf{M}(t)$ and a moving convex object $\mathscr{A}$ and a fixed convex object $\mathscr{B}$, we wish to calculate the tight upper bound μ of motion. Let $\mathbf{p}_i$ be a point on $\mathscr{A}$, and as $\mathscr{A}$ undergoes $\mathbf{M}$, $\mathbf{p}_i$ will trace out a trajectory $\mathbf{p}_i(t)$ in 3D Euclidean space. Consider its velocity $\dot{\mathbf{p}}_i(t) = \mathbf{v} + \omega \times \mathbf{r}_i(t)$ and notice that $\mathbf{v}, \omega$ are constant between the time interval of $[0,1]$ in our case. Then, the *directed motion bound* μ of $\mathscr{A}$ can be expressed as:

$$\mu \quad = \quad \max_i \int_0^1 |\dot{\mathbf{p}}_i(t) \cdot \mathbf{n}| \, dt$$

where $\mathbf{n}$ is the closest direction vector between $\mathscr{A}$ and $\mathscr{B}$.

Here, we give two equations to calculate the upper bound of motion. The first equation is a tighter upper bound of motion than the second one, but involves the extremal vertex query (EVQ) and so more computation is needed than the second.

1. Equation with EVQ

The first equation of calculating the upper bound of motion μ is derived as follows:

$$\begin{aligned}
& \max_i \int_0^1 |\dot{\mathbf{p}}_i(t) \cdot \mathbf{n}| \, dt \\
\leq \quad & \max_i \int_0^1 |\mathbf{v} \cdot \mathbf{n} + \omega \times \mathbf{r}_i(t) \cdot \mathbf{n}| \, dt \\
\leq \quad & |\mathbf{v} \cdot \mathbf{n}| + \max_i \int_0^1 |\omega \times \mathbf{r}_i(t) \cdot \mathbf{n}| \, dt \\
\leq \quad & |\mathbf{v} \cdot \mathbf{n}| + \int_0^1 \max_i (|\omega \times \mathbf{r}_i(t) \cdot \mathbf{n}|) \, dt \\
= \quad & |\mathbf{v} \cdot \mathbf{n}| + \int_0^1 \max_i (|\mathbf{n} \times \omega \cdot \mathbf{r}_i(t)|) \, dt
\end{aligned}$$

Let $\mathbf{c} = \mathbf{n} \times \omega$ and $\mathbf{r}_i(t) = \mathbf{R}(t)\mathbf{r}_i(0)$. The rotation $\mathbf{R}(t)$ is performed about the fixed axis ω by an angle θ. The rotation in the above equation can also be considered as $\mathbf{c}$ rotating about a fixed vector with $\mathbf{r}_i(t)$ fixed. However, this new rotation denoted by $\mathbf{R}_\mathbf{c}(t)$, is about the fixed axis ω by an angle $-\theta$. Notice that $\mathbf{R}_\mathbf{c}(t) = \mathbf{R}^{-1}(t) = \mathbf{R}^T(t)$. For simplicity, we denote that $\mathbf{R}_\mathbf{c}(t)\mathbf{c} = \mathbf{c}(t)$ and $\mathbf{r}_i = \mathbf{r}_i(0)$. Therefore, the second term of the above equation can be rewritten as:

$$\begin{aligned} & \int_0^1 \max_i(|\mathbf{n} \times \omega \cdot \mathbf{r}_i(t)|)\, dt \\ = & \int_0^1 \max_i(|\mathbf{c} \cdot \mathbf{R}(t)\mathbf{r}_i|)\, dt \\ = & \int_0^1 \max_i(|\mathbf{R}^T(t)\mathbf{c} \cdot \mathbf{r}_i|)\, dt \\ = & \int_0^1 \max_i(|\mathbf{c}(t) \cdot \mathbf{r}_i|)\, dt \end{aligned} \tag{1}$$

The problem of evaluating Eq. 1 is equivalent to performing the extremal vertex query for a set of directions $\mathbf{c}(t)$ where $t \in [0,1]$, instead of a single direction. Moreover, notice that $\mathbf{c}(t)$'s are co-planar (the plane's normal is ω). Therefore we have $\mathbf{c}(t) \cdot \mathbf{r}_i = \mathbf{c}(t) \cdot \mathbf{r}_i^c$ where $\mathbf{r}_i^c$ is the projection onto the plane by $\mathbf{r}_i$. For any given $\mathbf{r}_i$, we have

$$|\mathbf{c}(t) \cdot \mathbf{r}_i^c| = \begin{cases} |\mathbf{c}(t)||\mathbf{r}_i^c| & \text{if } (\mathbf{c}(0) \times \mathbf{r}_i^c) \cdot (\mathbf{c}(1) \times \mathbf{r}_i^c) > 0 \\ \max_i(\mathbf{c}(0) \cdot \mathbf{r}_i^c, \mathbf{c}(1) \cdot \mathbf{r}_i^c) & \text{if } (\mathbf{c}(0) \times \mathbf{r}_i^c) \cdot (\mathbf{c}(1) \times \mathbf{r}_i^c) \leq 0 \end{cases} \tag{2}$$

where $|\mathbf{r}_i^c| = |\frac{\omega}{|\omega|} \times \mathbf{r}_i|$.

A simple way to evaluate Eq. 1 is to visit all the vertices (i.e., all $\mathbf{r}_i$'s) of a convex polytope and find $\mathbf{r}_i$ to maximize Eq. 2. A more sophisticated way of evaluating Eq. 1 would be the use of quadratic programming since $|\mathbf{r}_i^c|$ contains quadratic terms

2. Equation without EVQ

The second equation of calculating the upper bound of motion μ is derived as follows:

$$\begin{aligned} & \max_i \int_0^1 |\dot{\mathbf{p}}_i(t) \cdot \mathbf{n}|\, dt \\ \leq & \max_i \int_0^1 |\mathbf{v} \cdot \mathbf{n} + \omega \times \mathbf{r}_i \cdot \mathbf{n}|\, dt \\ \leq & |\mathbf{v} \cdot \mathbf{n}| + \max_i \int_0^1 |\omega \times \mathbf{r}_i \cdot \mathbf{n}|\, dt \\ \leq & |\mathbf{v} \cdot \mathbf{n}| + \int_0^1 \max_i(|\omega \times \mathbf{r}_i \cdot \mathbf{n}|)\, dt \\ \leq & |\mathbf{v} \cdot \mathbf{n}| + \int_0^1 \max_i(|\mathbf{n} \times \omega \cdot \mathbf{r}_i|)\, dt \\ \leq & |\mathbf{v} \cdot \mathbf{n}| + |\mathbf{n} \times \omega| \max_i(|\mathbf{r}_i|) \\ = & \mu \end{aligned} \tag{3}$$

Note that $\max_i(|\mathbf{r}_i|)$ can be calculated as preprocess.

3. Performance Re-measurement

We re-tested all the benchmarks and provided the CCD query performance. The query performance by using EVQ (i.e., Eq. 1) is labelled with **FAST-EVQ** and the query performance without using EVQ (i.e., Eq. 3)is labelled with **FAST-NEVQ** (No Extremal Vertex Query). For some selected benchmarking scenarios, we also provide the average number of iterations. The following table summarizes the performance measured using the new equations (labelled as Timing with and without EVQ) and compared it against the one presented in the original paper (labelled as Timing old).

Benchmarks	Timing (old)	Timing (with EVQ)	Timing (without EVQ)
Santa vs Thin Board	0.0647	0.419	0.0194
Bunny vs Bunny	8.00	9.55	9.10
Torusknot vs Torusknot (2.8K)	0.948	0.983	0.937
Torusknot vs Torusknot (11K)	2.01	2.98	2.50
Torusknot vs Torusknot (34K)	5.32	5.88	5.38
RBD for Bunnies	0.643	0.728	0.501
RBD for Rings	0.737	2.07	1.89

Table 1: Performance remeasurement of all benchmarks.

These results show that the algorithmic performance based on the motion bound equation without EVQ (Eq. 1) is better than the one with EVQ (Eq. 3) in our benchmarks. The main reasons are: (1) visiting all the vertices in a polytope and performing EVQs on them can be costly and (2) the motion bound without EVQ is reasonably tight in practice, even though a bit looser compared to the one with EVQ. As a result, we have decided to use Eq. 1 without EVQ in practice, especially for complex non-convex models.

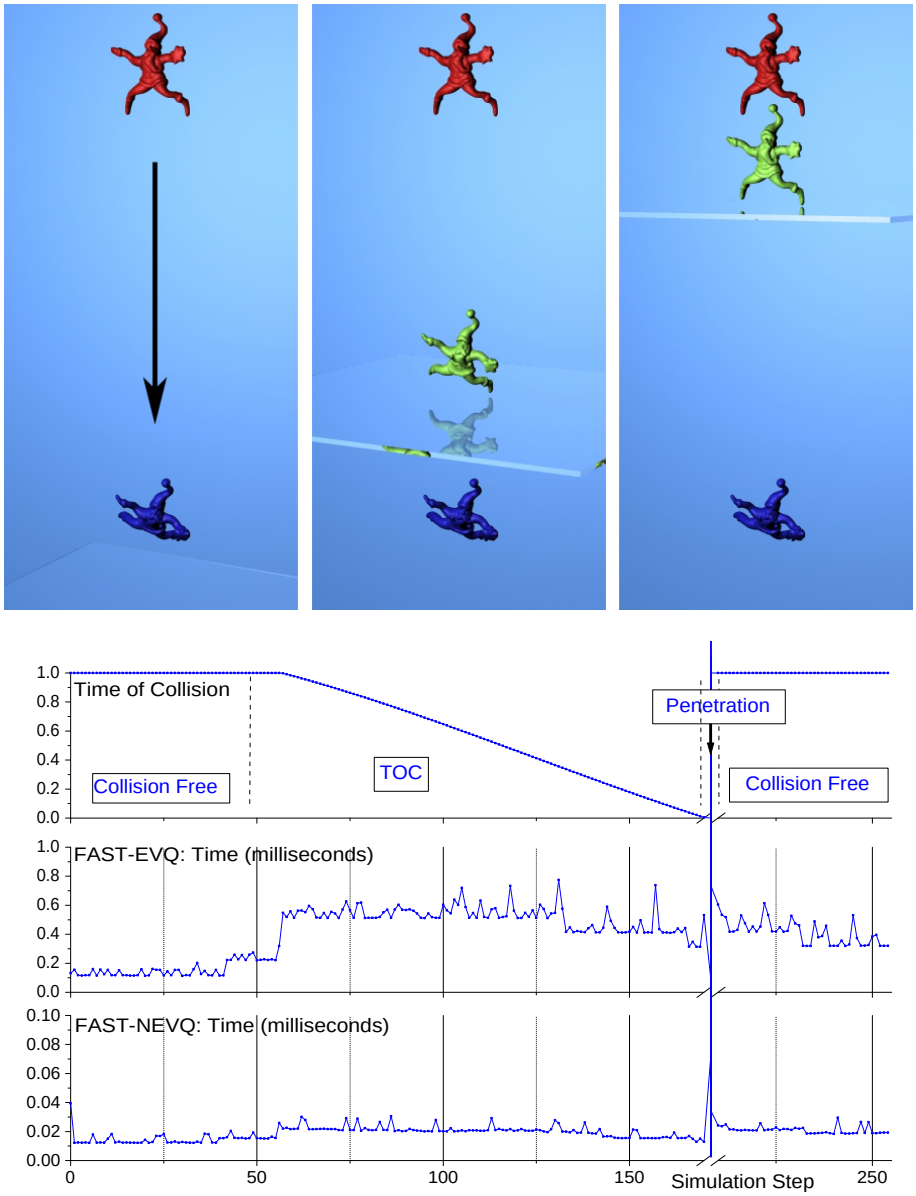

Figure 1: Benchmarking 1: Santa vs Thin Board. The average CCD query time with EVQ is 0.419 ms (2439 FPS) and The average CCD query time without EVQ is 0.0194 ms (51,546 FPS).The average number of iterations with EVQ is 3.64 and The average number of iterations without EVQ is 3.68.

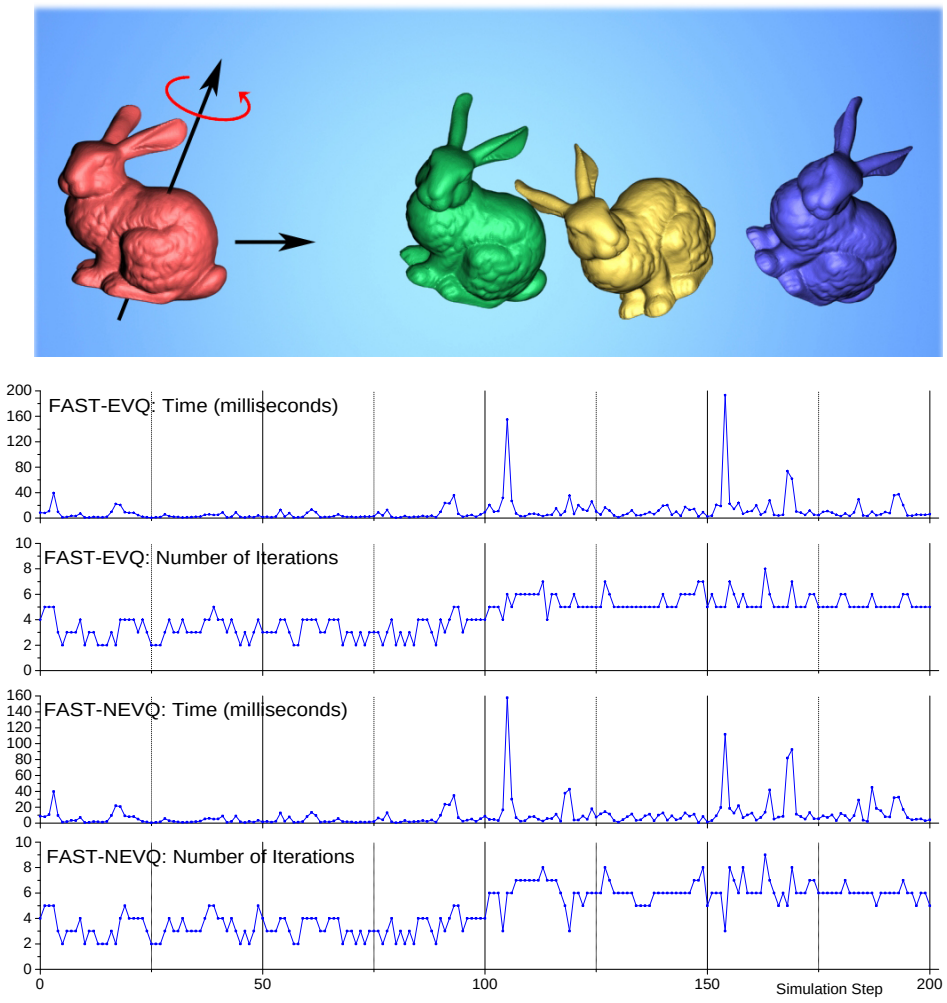

Figure 2: Benchmarking 2: Bunny vs Bunny. The average CCD query time with EVQ is 9.55 ms (105 FPS) and The average CCD query time without EVQ is 9.10 ms (110 FPS). The average number of iterations with EVQ is 4.32 and The average number of iterations without EVQ is 4.7

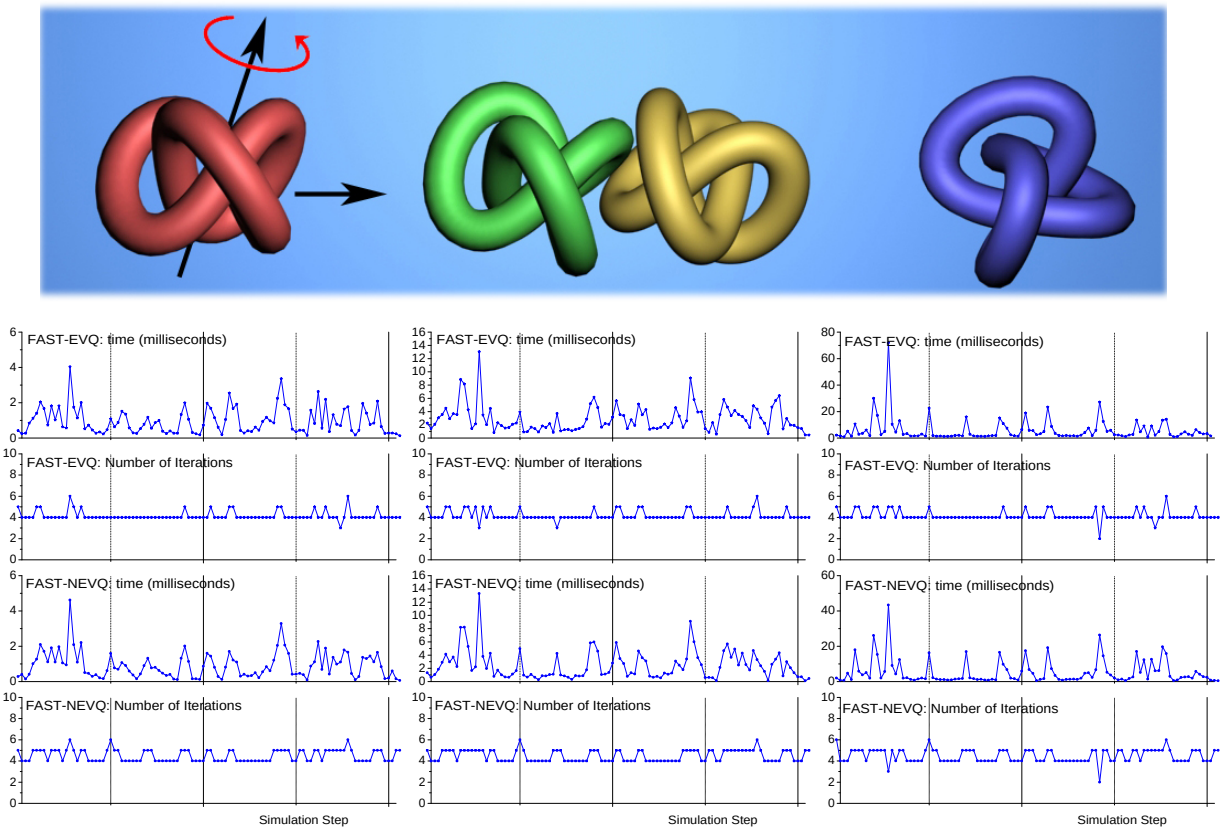

Figure 3: Benchmarking 3: Torusknots vs Torusknots in different Complexities. 2.8K, 11K, 34K triangles from left to right. The average CCD query times with EVQ are 0.983ms (1017 FPS), 2.98ms (336 FPS), 5.88ms (170 FPS), respectively. The average CCD query times without EVQ are 0.937ms (1067 FPS), 2.50ms (400 FPS), 5.38ms(186 FPS), respectively. The average number of iterations with EVQ are 4.16, 4.18, 4.16, respectively. The average number of iterations without EVQ are 4.49, 4.49, 4.46, respectively.

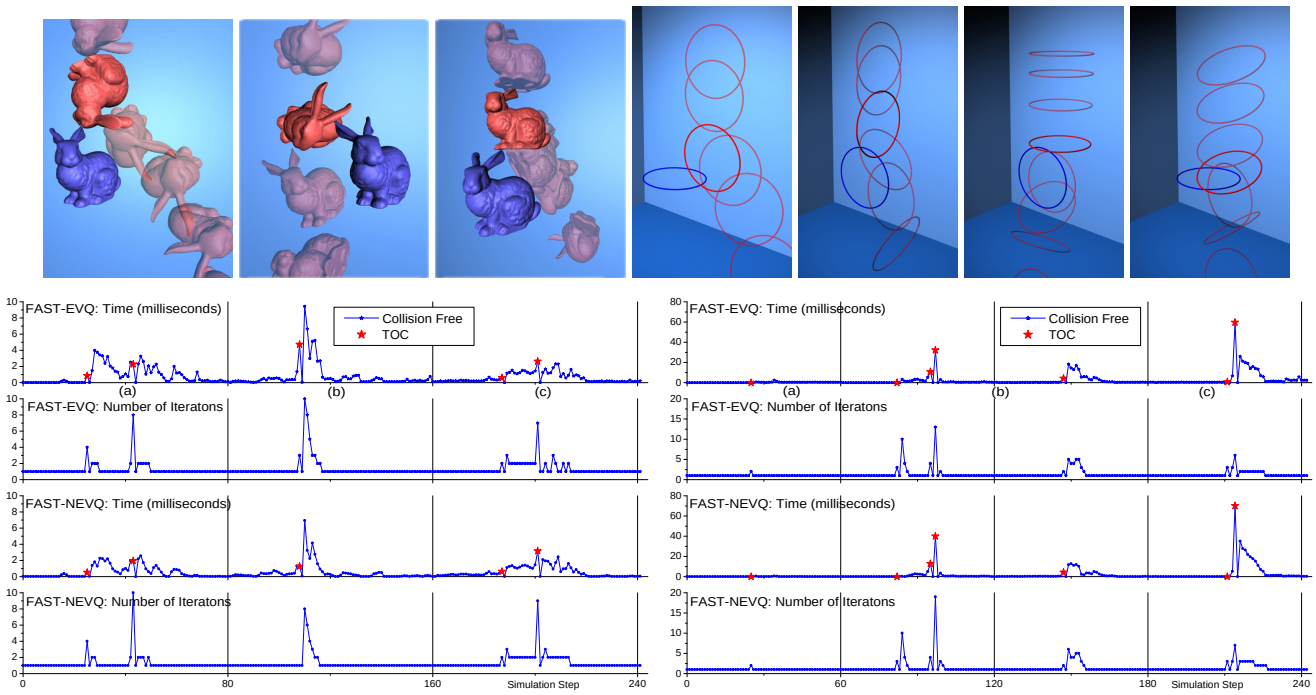

Figure 4: Benchmarking 4 and 5: Rigid Body Dynamics for Bunnies and Rings. For the bunny benchmark, the average CCD query time with EVQ is 0.728ms (1374 FPS) and the average CCD query time without EVQ is 0.501ms (1996 FPS). For the ring benchmark, the average CCD query time with EVQ is 2.07ms (483 FPS) and the average CCD query time without EVQ is 1.89ms (529 FPS).