Cascading upper bounds for triangle soup
Pompeiu-Hausdorff distance: supplemental material

Leonardo Sacht¹ and Alec Jacobson²,³

¹Universidade Federal de Santa Catarina, Brazil
²University of Toronto, Canada
³Adobe Research, Canada

We present here details omitted or presented in a summarized way in the paper: pseudocode for our method (Section 1) and extended data for some benchmarks (Section 2).

1. Pseudocode
Algorithms 1 to 6 are pseudocode for all functions discussed or mentioned in Section 4 of our paper. Numbers are represented by lowercase or Greek letters, while the other structures (vectors, points, matrices, AABB tree, and queue) are represented by capital letters.

Algorithms 1 and 2 were discussed in detail in Section 4 of the paper. Algorithms 3, 4, 5, and 6 are pseudocode for the four upper bounds used by our method. For better comprehension, we suggest the reader complement the reading of the code with the illustrations provided in Figures 6 to 10 from the paper.

Functions EdgeLengths (in Algorithms 3 and 4), DecideCase (in Algorithm 5), TsVsDs (in Algorithm 5), ExactPompeiuHausdorff (in Algorithm 5), EdgeBisectorIntersect (in Algorithms 5 and 6), and ShortestEdge (in Algorithm 6) are basic geometric operations for which we refer to the code at https://github.com/leokollersacht/pompeiu_hausdorff.

2. Extended benchmark data
2.1. Upper bound order benchmark
Table 1 shows the number of shared wins of every upper bound ordering tested on the benchmark presented in Figure 11 and discussed at the beginning of Section 5 of the paper.

2.2. Benchmark by Kang et al. [KKYK18]
This section contains details of the benchmark presented in Figures 13 and 14 and Table 2 of our paper.

Table 2 in this supplemental material shows the timings and memory usage of the three methods: the one by Kang et al. [KKYK18], the one by Zheng et al. [ZSL∗22], and ours. Timings are averages of running each method 100 times. The first row of each table contains the name of the model, preparation time, and the maximum number of triangles in the queue ever reached during the process. Preparation stands for the construction of a volumetric hierarchy for fast distance computation: the method of Kang et al. [KKYK18] uses a uniform grid, the method of Zheng et al. [ZSL∗22] uses an AABB specifically coded for their work, and we use libigl’s AABB.

Each row contains lower and upper bounds (relative to $d_A$, the length of the diagonal of the bounding box of $A$), time (in ms), number of iterations (an iteration consists of popping a triangle, subdividing it into four triangles, computing bounds, and enqueuing subtriangles that are not discarded) and the number of triangles in the queue at the moment where the methods reached $\frac{\text{num} + \text{discarded}}{d_A} < 10^{-4}$, $k = 1, \ldots, 8$.

We first observe how similar the table on the left (obtained by us running the code by Kang et al. [KKYK18]) is to Tables 2 and 3 in their paper. The central table [ZSL∗22] is new since the authors of this (more recent) paper did not compare their method to the method of Kang et al. [KKYK18].

Highlighted in bold is the fastest time among the three methods to conclude each stage of the computation (preparation or bound update): our method is the fastest in all cases. In terms of total time (preparation time in the top row, plus the time to reach the tolerance $\varepsilon = 10^{-8}$ in the last row), our method is $1.4\times$ faster than the method of Kang et al. [KKYK18] and $2.2\times$ faster than the method of Zheng et al. [ZSL∗22] for the Dental Crown pair, $2.4\times$ faster than [KKYK18] and $3.5\times$ faster than [ZSL∗22] for the Monster pair, $2.5\times$ faster than [KKYK18] and $6.4\times$ faster than [ZSL∗22] for the Bust pair, and $7.2\times$ faster than [KKYK18] and $9.8\times$ faster than [ZSL∗22] for the Ramesses pair.

Columns Iter in Table 2 also show the number of subdivided triangles since each iteration of branch and bound subdivides one triangle. Our method is the one with the smallest number of subdivisions.

References

Table 1: Number and percentage of shared wins for each possible bound ordering on a benchmark with 7,427 mesh pairs. Highlighted in bold are the data used in Figure 11 of the paper.

<table>
<thead>
<tr>
<th>Order</th>
<th>(1)</th>
<th>(2)</th>
<th>(2.1)</th>
<th>(1.2)</th>
<th>(4.1)</th>
<th>(4)</th>
<th>(4, 1.2)</th>
<th>(4, 2.1)</th>
<th>(4, 2)</th>
<th>(1, 4)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shared wins (%)</td>
<td>(9)</td>
<td>(75)</td>
<td>(93)</td>
<td>(112)</td>
<td>(410)</td>
<td>(413)</td>
<td>(575)</td>
<td>(578)</td>
<td>(585)</td>
<td>(633)</td>
</tr>
<tr>
<td>(1, 4, 2)</td>
<td>(1.4, 2)</td>
<td>(2, 1.4)</td>
<td>(2.4, 1)</td>
<td>(2.4)</td>
<td>(1.2, 4)</td>
<td>(3)</td>
<td>(3.2, 4, 1)</td>
<td>(3.4, 1, 2)</td>
<td>(3.4, 2, 1)</td>
<td>(3, 2, 1, 4)</td>
</tr>
<tr>
<td>871</td>
<td>(11.7%)</td>
<td>(1.049)</td>
<td>(1.051)</td>
<td>(1.065)</td>
<td>(1.111)</td>
<td>(3.471)</td>
<td>(3.494)</td>
<td>(3.504)</td>
<td>(3.518)</td>
<td>(3.529)</td>
</tr>
<tr>
<td>(3, 4, 2)</td>
<td>(1.3, 4, 2)</td>
<td>(1.3)</td>
<td>(1.3, 2, 4)</td>
<td>(3.1, 2, 4)</td>
<td>(1.3, 4)</td>
<td>(3.1)</td>
<td>(3.1, 4, 2)</td>
<td>(3.4, 1)</td>
<td>(3.4)</td>
<td>(3, 1, 4)</td>
</tr>
<tr>
<td>3.536</td>
<td>(47.6%)</td>
<td>(3.541)</td>
<td>(3.543)</td>
<td>(3.547)</td>
<td>(3.562)</td>
<td>(3.568)</td>
<td>(3.578)</td>
<td>(3.582)</td>
<td>(3.590)</td>
<td>(3.592)</td>
</tr>
<tr>
<td>(1, 3, 2)</td>
<td>(3.2)</td>
<td>(3, 1, 2)</td>
<td>(3.2, 1)</td>
<td>(4.1, 3, 2)</td>
<td>(4, 1, 2, 3)</td>
<td>(4, 1, 3)</td>
<td>(4, 1, 3, 2)</td>
<td>(4, 3, 1, 2)</td>
<td>(4.3, 2, 1)</td>
<td></td>
</tr>
<tr>
<td>(4, 3, 2)</td>
<td>(4.2, 3, 1)</td>
<td>(4.2, 3)</td>
<td>(4.3, 1)</td>
<td>(1.4, 3)</td>
<td>(1.4, 3, 2)</td>
<td>(1.4, 2, 3)</td>
<td>(2, 4, 1, 3)</td>
<td>(2, 4, 3)</td>
<td>(2, 1, 4, 3)</td>
<td>(2, 4, 3, 1)</td>
</tr>
<tr>
<td>(2, 3)</td>
<td>(1, 2, 4, 3)</td>
<td>(2.3, 1)</td>
<td>(2.1, 3)</td>
<td>(1.2, 3)</td>
<td>(2.3, 4, 1)</td>
<td>(2.3, 1, 4)</td>
<td>(2, 3, 4)</td>
<td>(2, 1, 3, 4)</td>
<td>(1.2, 3, 4)</td>
<td></td>
</tr>
<tr>
<td>5.101</td>
<td>(68.7%)</td>
<td>(5.121)</td>
<td>(5.163)</td>
<td>(5.166)</td>
<td>(5.212)</td>
<td>(5.297)</td>
<td>(5.326)</td>
<td>(5.341)</td>
<td>(5.342)</td>
<td>(5.472)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Prep (ms)</th>
<th>Max</th>
<th>Time</th>
<th>Iter</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>D. Crown</td>
<td>143</td>
<td>Max 54916</td>
<td>1/4α</td>
<td>( u_{\max}/dA )</td>
<td>Time</td>
</tr>
<tr>
<td>10^{-1}</td>
<td>0.0011498</td>
<td>0.00272993</td>
<td>25</td>
<td>0</td>
<td>19593</td>
</tr>
<tr>
<td>10^{-2}</td>
<td>0.0017492</td>
<td>0.0117476</td>
<td>61</td>
<td>757</td>
<td>21864</td>
</tr>
<tr>
<td>10^{-3}</td>
<td>0.0021981</td>
<td>0.0031256</td>
<td>229</td>
<td>36512</td>
<td>41567</td>
</tr>
<tr>
<td>10^{-4}</td>
<td>0.0022149</td>
<td>0.0022706</td>
<td>398</td>
<td>70550</td>
<td>7782</td>
</tr>
<tr>
<td>10^{-5}</td>
<td>0.0022214</td>
<td>0.0022217</td>
<td>411</td>
<td>72935</td>
<td>5040</td>
</tr>
<tr>
<td>10^{-6}</td>
<td>0.0022217</td>
<td>0.0022217</td>
<td>411</td>
<td>72941</td>
<td>5408</td>
</tr>
<tr>
<td>10^{-7}</td>
<td>0.0022216</td>
<td>0.0022217</td>
<td>411</td>
<td>72946</td>
<td>5409</td>
</tr>
<tr>
<td>10^{-8}</td>
<td>0.0022217</td>
<td>0.0022217</td>
<td>411</td>
<td>72955</td>
<td>5412</td>
</tr>
</tbody>
</table>

| Table 2: Statistics obtained running the three methods on the benchmark proposed by Kang et al. [KKYK18]. Our method is the fastest in all stages of computation. |
Algorithm 1: Pompeiu-Hausdorff distance

**Inputs:**
- $V_A$: $m_A \times 3$ matrix with vertices from mesh $A$
- $F_A$: $n_A \times 3$ matrix with triangles from mesh $A$
- $V_B$: $m_B \times 3$ matrix with vertices from mesh $B$
- $F_B$: $n_B \times 3$ matrix with triangles from mesh $B$
- $\varepsilon$: tolerance
- $m$: factor to define maximum number of triangles

**Outputs:**
- $l$: lower bound
- $u$: upper bound

**begin**
- $d_A \leftarrow$ BoundingBoxDiagonal($V_A$)
- $T_B \leftarrow$ AABB($V_B, F_B$)
- $[D_A, I_A, C_A] \leftarrow$ Distance($V_A, V_B, F_B, T_B$)
- $l \leftarrow$ Max($D_A$)
- $U \leftarrow$ UpperBounds($V_A, F_A, V_B, F_B, D_A, I_A, C_A, I_A, C_A, I_A, C_A, I_A, C_A$)
- $u \leftarrow$ Max($U$)
- $Q \leftarrow$ PriorityQueue((double,int), less)

**forall** $k \in \{1, \ldots, n_A\}$ **do**
- if $U(k) \geq l$ then
  - $Q$.Emplace($U(k), k$)
- $m_f \leftarrow m \cdot n_A$
- $c_f \leftarrow n_A$

**while** $\frac{\max U(k)}{m} > \varepsilon$ **do**
- $f \leftarrow Q$.top().second
- $Q$.pop()
- $[W_A, G_A] \leftarrow$ Subdivide($V_A, F_A, f$)
- $V_A \leftarrow$ Append($V_A, W_A$)
- $F_A \leftarrow$ Append($F_A, G_A$)
- $[E_A, J_A, B_A] \leftarrow$ Distance($W_A, V_B, F_B, T_B$)
- $D_A \leftarrow$ Append($D_A, E_A$)
- $I_A \leftarrow$ Append($I_A, J_A$)
- $C_A \leftarrow$ Append($C_A, B_A$)
- $l \leftarrow$ Max($\max(E_A, I_A)$)
- $U_{\text{new}} \leftarrow$ UpperBounds($V_A, G_A, V_B, F_B, D_A, I_A, C_A, I_A, C_A, I_A, C_A$)
- $u \leftarrow$ Max($\max(U_{\text{new}}), Q$.top()$.first$)
- **forall** $k \in \{1, 2, 3, 4\}$ **do**
  - if $U_{\text{new}}(k) \geq l$ then
    - $Q$.Emplace($U_{\text{new}}(k), c_f + k$)
  - $c_f \leftarrow c_f + 4$
  - if $c_f > m_f$ then
    - error(‘exceeded maximum number of faces’)

Algorithm 2: UpperBounds($V_A, F_A, V_B, F_B, D_A, I_A, C_A, I_A, C_A, I_A, C_A$) $\rightarrow$ $U$

**Inputs:**
- $V_A$: $m_A \times 3$ matrix with vertices from mesh $A$
- $F_A$: $n_A \times 3$ matrix with triangles from mesh $A$
- $V_B$: $m_B \times 3$ matrix with vertices from mesh $B$
- $F_B$: $n_B \times 3$ matrix with triangles from mesh $B$
- $D_A$: $m_A$-long vector of vertex distances
- $I_A$: $m_A$-long vector of closest triangles
- $C_A$: $m_A$-long vector with closest points
- $l$: running lower bound

**Output:**
- $U$: $n_A$-long vector of upper bounds

**begin**
- **foreach** $i \in \{1, \ldots, n_A\}$ **do**
  - $u_i \leftarrow$ FirstBound($V_A, F_A, i, D_A$)
  - $U(i) \leftarrow u_i$
  - if $U(i) < l$ then
    - $\text{done} \leftarrow \text{true}$
  - if $\text{done} = \text{false}$ then
    - $u_i \leftarrow$ SecondBound($V_A, F_A, i, D_A$)
    - $U(i) \leftarrow \text{Min}(u_i, U(i))$
    - if $U(i) < l$ then
      - $\text{done} \leftarrow \text{true}$
  - if $\text{done} = \text{false}$ then
    - $u_i \leftarrow$ ThirdBound($V_A, F_A, i, V_B, F_B, D_A, I_A$)
    - $U(i) \leftarrow \text{Min}(u_i, U(i))$
    - if $U(i) < l$ then
      - $\text{done} \leftarrow \text{true}$
  - if $\text{done} = \text{false}$ then
    - $u_i \leftarrow$ FourthBound($V_A, F_A, i, C_A$)
    - $U(i) \leftarrow \text{Min}(u_i, U(i))$

Algorithm 3: FirstBound($V_A, F_A, i, D_A$) $\rightarrow u_1$

**Inputs:**
- $V_A$: $m_A \times 3$ matrix with vertices from mesh $A$
- $F_A$: $n_A \times 3$ matrix with triangles from mesh $A$
- $i$: triangle index
- $D_A$: $m_A$-long vector of vertex distances

**Output:**
- $u_1$: first upper bound for the $i$-th triangle

**begin**
- $u_1 \leftarrow \infty$
- $E \leftarrow$ EdgeLengths($V_A, F_A, i$)
- **foreach** $k \in \{1, 2, 3\}$ **do**
  - $d \leftarrow D_A(F_A(i, k))$
  - $u_1 \leftarrow \text{Min}(u_1, d + \max(E(k + 1), E(k + 2)))$
Algorithm 4: SecondBound($V_A, F_A, i, D_A$) → $u_2$

Inputs:
- $V_A$ $m_A \times 3$ matrix with vertices from mesh $A$
- $F_A$ $n_A \times 3$ matrix with triangles from mesh $A$
- $i$ triangle index
- $D_A$ $m_A$-long vector of vertex distances

Output:
- $u_2$ second upper bound for the $i$-th triangle

begin

$E \leftarrow$ EdgeLengths($V_A, F_A, i$)
$s \leftarrow \frac{E(1) + E(2) + E(3)}{2}$
$a \leftarrow \sqrt{s \cdot (s - E(1)) \cdot (s - E(2)) \cdot (s - E(3))}$
$r \leftarrow \frac{a}{2}$

$u_2 \leftarrow \max(D_A(F_A(i, 1)), D_A(F_A(i, 2)), D_A(F_A(i, 3)))$
if $s - 2 > 2 \cdot r$ then
$u_2 \leftarrow u_2 + r$
else
$u_2 \leftarrow \frac{\max(E(1), E(2), E(3))}{2}$

end

Algorithm 6: FourthBound($V_A, F_A, i, C_A$) → $u_4$

Inputs:
- $V_A$ $m_A \times 3$ matrix with vertices from mesh $A$
- $F_A$ $n_A \times 3$ matrix with triangles from mesh $A$
- $i$ triangle index
- $C_A$ $m_A \times 3$ matrix with closest points

Output:
- $u_4$ fourth upper bound for the $i$-th triangle

begin

$u_4 \leftarrow \infty$
$V_1 \leftarrow V_A(i, 1), V_2 \leftarrow V_A(i, 2), V_3 \leftarrow V_A(i, 3)$
$e \leftarrow$ ShortestEdge($V_1, V_2, V_3$)
$Q_1 \leftarrow C_A(F_A(i, e), :)$

foreach $a \in \{1, 2\}$ do
$Q_2 \leftarrow C_A(F_A(i, e + a), :)$
$d_1 \leftarrow \min(||V_1 - Q_1||, ||V_1 - Q_2||)$
$d_2 \leftarrow \min(||V_2 - Q_1||, ||V_2 - Q_2||)$
$d_3 \leftarrow \min(||V_3 - Q_1||, ||V_3 - Q_2||)$
$u_p \leftarrow \max(d_1, d_2, d_3)$

for each $b \in \{1, 2, 3\}$ do
$P \leftarrow$ EdgeBisectorIntersect($Q_1, Q_2, V_{b+1}, V_{b+2}$)
if $P \neq \emptyset$ then
$u_p \leftarrow \max(||P - Q_1||, ||P - Q_2||, u_p)$

$u_4 \leftarrow \min(u_4, u_p)$

end