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

Leonardo Sacht[1] and Alec Jacobson[2,3]

[1]Universidade Federal de Santa Catarina, Brazil
[2]University of Toronto, Canada [3]Adobe 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 dA, 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 $\frac{u_{max}-l}{dA} < 10^{-k}$, $k = 1, \ldots, 8$.

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× faster than the method of Kang et al. [KKYK18] and 2.2× faster than the method of Zheng et al. [ZSL*22] for the Dental Crown pair, 2.4× faster than [KKYK18] and 3.5× faster than [ZSL*22] for the Monster pair, 2.5× faster than [KKYK18] and 6.4× faster than [ZSL*22] for the Bust pair, and 7.2× faster than [KKYK18] and 9.8× 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

[KKYK18] KANG Y., KYUNG M.-H., YOON S.-H., KIM M.-S.: Fast and robust Hausdorff distance computation from triangle mesh to quad mesh in near-zero cases. *Comput. Aided Geom. Des. 62*, C (may 2018), 91–103. URL: https://doi.org/10.1016/j.cagd.2018.03.017, doi:10.1016/j.cagd.2018.03.017. 1, 3

[ZSL*22] ZHENG Y., SUN H., LIU X., BAO H., HUANG J.: Economic

Order	**(1)**	**(2)**	**(2,1)**	**(1,2)**	**(4,1)**	**(4)**	**(4,1,2)**	(4,2,1)	(4,2)	(1,4)
Shared wins (%)	**9 (0.12%)**	**75 (1.01%)**	**93 (1.25%)**	**112 (1.51%)**	**410 (5.52%)**	**413 (5.56%)**	**575 (7.74%)**	578 (7.78%)	585 (7.88%)	633 (8.52%)
(1,4,2)	(2,1,4)	(2,4,1)	(2,4)	(1,2,4)	**(3)**	**(3,2,4,1)**	(3,4,1,2)	(3,4,2,1)	(3,2,4)	(3,2,1,4)
871 (11.7%)	1,049 (14.1%)	1,051 (14.2%)	1,065 (14.3%)	1,111 (15.0%)	**3,471 (46.7%)**	**3,494 (47.0%)**	3,504 (47.2%)	3,518 (47.4%)	3,529 (47.5%)	3,532 (47.6%)
(3,4,2)	(1,3,4,2)	(1,3)	(1,3,2,4)	(3,1,2,4)	(1,3,4)	(3,1)	(3,1,4,2)	(3,4,1)	(3,4)	(3,1,4)
3,536 (47.6%)	3,541 (47.7%)	3,543 (47.1%)	3,547 (47.8%)	3,562 (48.0%)	3,568 (48.0%)	3,578 (48.2%)	3,582 (48.2%)	3,590 (48.3%)	3,592 (48.4%)	3,653 (49.2%)
(1,3,2)	(3,2)	(3,1,2)	(3,2,1)	(4,1,3,2)	(4,1,2,3)	(4,2,1,3)	(4,1,3)	(4,3,1,2)	(4,3)	(4,3,2,1)
3,752 (50.5%)	3,826 (51.5%)	3,830 (51.6%)	3,840 (51.7%)	4,236 (57.0%)	4,372 (58.9%)	4,406 (59.3%)	4,407 (59.3%)	4,408 (59.4%)	4,426 (59.6%)	4,431 (59.7%)
(4,3,2)	(4,2,3,1)	(4,2,3)	(4,3,1)	(1,4,3)	(1,4,3,2)	(1,4,2,3)	(2,4,1,3)	(2,4,3)	(2,1,4,3)	(2,4,3,1)
4,446 (59.9%)	4,532 (61.0%)	4,575 (61.6%)	4,634 (62.4%)	4,660 (62.7%)	4,700 (63.3%)	4,878 (65.8%)	4,904 (66.0%)	4,940 (66.5%)	5,013 (67.5%)	5,041 (67.9%)
(2,3)	**(1,2,4,3)**	**(2,3,1)**	**(2,1,3)**	**(1,2,3)**	**(2,3,4,1)**	**(2,3,1,4)**	**(2,3,4)**	**(2,1,3,4)**	**(1,2,3,4)**	*Order*
5,101 (68.7%)	**5,121 (69.0%)**	**5,163 (69.5%)**	**5,166 (69.6%)**	**5,212 (70.2%)**	**5,297 (71.3%)**	**5,326 (71.7%)**	**5,341 (71.9%)**	**5,342 (71.9%)**	**5,472 (73.7%)**	*Shared wins (%)*

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.*

upper bound estimation in Hausdorff distance computation for triangle meshes. *Computer Graphics Forum* (2022). doi:10.1111/cgf.14395. 1, 3

	D. Crown	Prep (ms) 143		Max 54916		D. Crown	Prep (ms) 163		Max 55176		D. Crown	Prep (ms) **99**		Max 54783	
10^{-k}	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size
10^{-1}	0.0011498	0.0272993	25	0	19593	0.0011498	0.0250696	79	0	19589	0.0011498	0.0223865	**17**	0	19593
10^{-2}	0.0017492	0.0117476	31	757	21864	0.0017492	0.0117391	88	783	21938	0.0017492	0.0117454	**21**	624	21465
10^{-3}	0.0021256	0.0031256	229	36512	41567	0.0021256	0.0031256	428	37030	43225	0.0021256	0.0031256	**196**	35778	41678
10^{-4}	0.0021981	0.0022706	398	70550	7782	0.0022149	0.0023149	715	70435	10335	0.0021981	0.0022706	**303**	69909	7782
10^{-5}	0.0022149	0.0022217	411	72935	5404	0.0022214	0.0022278	753	74876	5922	0.0022149	0.0022217	**311**	72320	5376
10^{-6}	0.0022214	0.0022217	411	72941	5408	0.0022214	0.0022224	756	75174	5625	0.0022214	0.0022217	**311**	72324	5380
10^{-7}	0.0022216	0.0022217	411	72946	5409	0.0022217	0.0022218	757	75209	5611	0.0022216	0.0022217	**311**	72329	5385
10^{-8}	0.0022217	0.0022217	411	72955	5412	0.0022217	0.0022217	757	75225	5606	0.0022217	0.0022217	**311**	72330	5387

	Monster	Prep (ms) 61		Max 20		Monster	Prep (ms) 111		Max 4190		Monster	Prep (ms) **42**		Max 15	
10^{-k}	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size
10^{-1}	0.0101105	0.0297328	52	0	16	0.0101105	0.0250135	53	0	4188	0.0101105	0.0246585	**5**	0	13
10^{-2}	0.0101105	0.0194350	52	3	20	0.0101105	0.0188189	53	3	4190	0.0101105	0.0189429	**5**	1	15
10^{-3}	0.0101105	0.0110357	52	13	10	0.0101105	0.0108320	53	12	4181	0.0101105	0.0106876	**5**	10	6
10^{-4}						0.0101105	0.0101876	53	17	4176					
10^{-6}	0.0101105	0.0101114	52	17	6										
10^{-8}	0.0101105	0.0101105	52	18	5	0.0101105	0.0101105	53	18	4175	0.0101105	0.0101105	**5**	11	5

	Bust	Prep (ms) 309		Max 2161		Bust	Prep (ms) 797		Max 7471		Bust	Prep (ms) **214**		Max 315	
10^{-k}	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size
10^{-2}	0.0018266	0.0074853	381	0	2130	0.0018266	0.0075448	995	0	7436	0.0018266	0.0070458	**66**	0	285
10^{-3}	0.0018266	0.0028241	381	37	2140	0.0018266	0.0028001	995	36	7456	0.0018266	0.0028241	**66**	30	298
10^{-4}	0.0018266	0.0019265	388	826	1352	0.0018266	0.0019264	999	406	7087	0.0018266	0.0019259	**66**	91	237
10^{-5}	0.0018266	0.0018365	399	1974	204	0.0018266	0.0018363	1003	813	6682	0.0018266	0.0018364	**67**	271	57
10^{-6}	0.0018266	0.0018276	400	2150	28	0.0018266	0.0018272	1004	866	6629	0.0018266	0.0018276	**67**	319	9
10^{-7}	0.0018266	0.0018266	401	2171	7	0.0018266	0.0018266	1004	874	6621					
10^{-8}	0.0018266	0.0018266	401	2172	6	0.0018266	0.0018266	1004	875	6620	0.0018266	0.0018266	**67**	322	6

	Ramesses	Prep (ms) 163		Max 5		Ramesses	Prep (ms) 1824		Max 2649		Ramesses	Prep (ms) **111**		Max 21	
10^{-k}	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size	l/dA	$u_{\max}/dA$	Time	Iter	Size
10^{-3}						0.0254824	0.0256320	441	0	2644					
10^{-4}	0.0254824	0.0255508	1507	0	5	0.0255078	0.0255508	441	2	2646	0.0254824	0.0255418	**120**	0	5
10^{-5}						0.0255416	0.0255508	441	7	2646	0.0255325	0.0255418	**121**	3	11
10^{-6}	0.0255416	0.0255426	1508	7	5	0.0255416	0.0255426	441	8	2647	0.0255416	0.0255418	**121**	4	13
10^{-7}	0.0255417	0.0255418	1509	20	5	0.0255417	0.0255418	442	22	2649	0.0255417	0.0255418	**121**	10	15
10^{-8}	0.0255418	0.0255418	1509	27	5	0.0255418	0.0255418	442	27	2649	0.0255418	0.0255418	**121**	14	21
	Method of Kang et al. [KKYK18]					Method of Zheng et al. [ZSL*22]					Our method				

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: PompeiuHausdorff$(V_A, F_A, V_B, F_B, \varepsilon, m) \to l, u$

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
- ε — tolerance
- m — factor to define maximum number of triangles

Outputs:
- l — lower bound
- u — upper bound

```
begin
    d_A ← BoundingBoxDiagonal(V_A)
    T_B ← AABB(V_B, F_B)
    [D_A, I_A, C_A] ← Distance(V_A, V_B, F_B, T_B)
    l ← Max(D_A)
    U ← UpperBounds(V_A, F_A, V_B, F_B, D_A, I_A, C_A, l)
    u ← Max(U)
    Q ← PriorityQueue(⟨double,int⟩, less)
    foreach k ∈ {1, ..., n_A} do
        if U(k) ⩾ l then
            Q.Emplace(U(k), k)
    m_f ← m · n_A
    c_f ← n_A
    while (u − l)/d_A > ε do
        f ← Q.top().second
        Q.pop()
        [W_A, G_A] ← Subdivide(V_A, F_A, f)
        V_A ← Append(V_A, W_A)
        F_A ← Append(F_A, G_A)
        [E_A, J_A, B_A] ← Distance(W_A, V_B, F_B, T_B)
        D_A ← Append(D_A, E_A)
        I_A ← Append(I_A, J_A),
        C_A ← Append(C_A, B_A)
        l ← Max(Max(E_A), l)
        U_new ← UpperBounds(V_A, G_A, V_B, F_B, D_A, I_A, C_A, l)
        u ← Max(Max(U_new), Q.top().first)
        foreach k ∈ {1, 2, 3, 4} do
            if U_new(k) ⩾ l then
                Q.Emplace(U_new(k), c_f + k)
        c_f ← c_f + 4
        if c_f > m_f then
            error("exceeded maximum number of faces")
```

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

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 — $m_A \times 3$ matrix with closest points
- l — running lower bound

Output:
- U — n_A-long vector of upper bounds

```
begin
    foreach i ∈ {1, ..., n_A} do
        done ← false
        if I_A(F_A(i,1)) = I_A(F_A(i,2)) = I_A(F_A(i,3)) then
            U(i) ← Max(D_A(F_A(i,1)), D_A(F_A(i,2)),
                        D_A(F_A(i,3)))
            done ← true
        if done = false then
            u_1 ← FirstBound(V_A, F_A, i, D_A)
            U(i) ← u_1
            if U(i) < l then
                done ← true
        if done = false then
            u_2 ← SecondBound(V_A, F_A, i, D_A)
            U(i) ← Min(u_2, U(i))
            if U(i) < l then
                done ← true
        if done = false then
            u_3 ← ThirdBound(V_A, F_A, i, V_B, F_B, D_A, I_A)
            U(i) ← Min(u_3, U(i))
            if U(i) < l then
                done ← true
        if done = false then
            u_4 ← FourthBound(V_A, F_A, i, C_A)
            U(i) ← Min(u_4, U(i))
```

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

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 ← ∞
    E ← EdgeLengths(V_A, F_A, i)
    foreach k ∈ {1, 2, 3} do
        d ← D_A(F_A(i,k))
        u_1 ← Min(u_1, d + Max(E(k+1), E(k+2)))
```

Algorithm 4: SecondBound$(V_A, F_A, i, D_A) \rightarrow u_2$

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)
- $s \leftarrow \frac{E(1)+E(2)+E(3)}{2}$
- $a \leftarrow \sqrt{s\cdot(s-E(1))\cdot(s-E(2))\cdot(s-E(3))}$
- $r \leftarrow \frac{E(1)\cdot E(2)\cdot E(3)}{4\cdot a}$
- $t \leftarrow \frac{a}{s}$
- $u_2 \leftarrow \mathrm{Max}(D_A(F_A(i,1)), D_A(F_A(i,2)), D_A(F_A(i,3)))$
- **if** $s - t > 2\cdot r$ **then**
  - $u_2 \leftarrow u_2 + r$
- **else**
  - $u_2 \leftarrow u_2 + \frac{\mathrm{Max}(E(1),E(2),E(3))}{2}$

Algorithm 5: ThirdBound$(V_A, F_A, i, V_B, F_B, D_A, I_A) \rightarrow u_3$

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
- 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

Output:
- u_3 — third upper bound for the i-th triangle

begin
- $case \leftarrow$ DecideCase$(I_A(i,1), I_A(i,2), I_A(i,3))$
- **if** $case = 1$ **then**
  - $[s_1, s_2, V_1, V_2, V_3, d_1, d_2, d_3] \leftarrow$ TsVsDs(I_A, V_A, D_A, i)
  - $B_1 \leftarrow$ EdgeBisectorIntersect(s_1, s_2, V_1, V_2)
  - **if** $B_1 = \emptyset$ **then** $B_1 \leftarrow \frac{V_1+V_2}{2}$;
  - $B_2 \leftarrow$ EdgeBisectorIntersect(s_1, s_2, V_1, V_3)
  - **if** $B_2 = \emptyset$ **then** $B_2 \leftarrow \frac{V_1+V_3}{2}$;
  - $h_1 \leftarrow$ ExactPompeiuHausdorff$((V_1,d_1), B_1, B_2, s_1)$
  - $h_2 \leftarrow$ ExactPompeiuHausdorff$(B_1, (V_2,d_2), (V_3,d_3), B_2, s_2)$
  - $u_3 \leftarrow \mathrm{Max}(h_1, h_2)$
- **else**
  - $V_1 \leftarrow V_A(i,1)$, $V_2 \leftarrow V_A(i,2)$, $V_3 \leftarrow V_A(i,3)$
  - $d_1 \leftarrow D_A(i,1)$, $d_2 \leftarrow D_A(i,2)$, $d_3 \leftarrow D_A(i,3)$
  - $s_1 \leftarrow I_A(i,1)$, $s_2 \leftarrow I_A(i,2)$, $s_3 \leftarrow I_A(i,3)$
  - $M_1 \leftarrow \frac{V_1+V_2}{2}$, $M_2 \leftarrow \frac{V_2+V_3}{2}$, $M_3 \leftarrow \frac{V_3+V_1}{2}$
  - $B \leftarrow \frac{V_1+V_2+V_3}{3}$
  - $h_1 \leftarrow$ ExactPompeiuHausdorff$((V_1,d_1), M_1, B, M_3, s_1)$
  - $h_2 \leftarrow$ ExactPompeiuHausdorff$((V_2,d_2), M_2, B, M_1, s_2)$
  - $h_3 \leftarrow$ ExactPompeiuHausdorff$((V_3,d_3), M_3, B, M_2, s_3)$
  - $u_3 \leftarrow \mathrm{Max}(h_1, h_2, h_3)$
  - $h_1 \leftarrow$ ExactPompeiuHausdorff$((V_1,d_1), V_2, V_3, s_1)$
  - $h_2 \leftarrow$ ExactPompeiuHausdorff$(V_1, (V_2,d_2), V_3, s_2)$
  - $h_3 \leftarrow$ ExactPompeiuHausdorff$(V_1, V_2, (V_3,d_3), s_3)$
  - $u_3 \leftarrow \mathrm{Min}(u_3, \mathrm{Min}(h_1, h_2, h_3))$

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

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**
  - $Q_2 \leftarrow C_A(F_A(i,e+a), :)$
  - $d_1 \leftarrow \mathrm{Min}(\|V_1 - Q_1\|, \|V_1 - Q_2\|)$
  - $d_2 \leftarrow \mathrm{Min}(\|V_2 - Q_1\|, \|V_2 - Q_2\|)$
  - $d_3 \leftarrow \mathrm{Min}(\|V_3 - Q_1\|, \|V_3 - Q_2\|)$
  - $u_p \leftarrow \mathrm{Max}(d_1, d_2, d_3)$
  - **foreach** $b \in \{1,2,3\}$ **do**
    - $P \leftarrow$ EdgeBisectorIntersect$(Q_1, Q_2, V_{b+1}, V_{b+2})$
    - **if** $P \neq \emptyset$ **then**
      - $u_p \leftarrow \mathrm{Max}(\mathrm{Max}(\|P - Q_1\|, \|P - Q_2\|), u_p)$
  - $u_4 \leftarrow \mathrm{Min}(u_p, u_4)$