Fast Splat Morphing: Supplementary Material

Marcel Padilla
marcel.padilla@inf.ethz.ch
ETH Zürich
Zürich, Switzerland

Abstract

This document accompanies the paper *Fast Splat Morphing*. It holds the padding construction, the co-bisection's ablations and timing methodology, the accuracy ladder with its certified bound, the reference solvers, the per-pair tables, the frame dependence and failure search, the lightness weight, correspondence diagnostics, the pair screen, further galleries, and a map from each result to the script that produces it. Section, figure, table and equation numbers without an S refer to the paper.

Keywords

3D Gaussian splatting, optimal transport, morphing

S1 Figure notes

In Figure 2, both generated (image-to-3d) objects are downsampled to 2×10^4 Gaussians so that the exact solve is feasible, turned to a common heading, and count-matched, so the instance carries no padding. In the teaser, the lion is turned about its up axis to face the bicycle, and the three rows use six distinct objects. Every strip figure is rendered headlessly from the demo with the named assignment, and the renderer reports the active mapping, so a strip cannot silently fall back to another assignment.

S2 Count-mismatch padding

Figure S1 shows why the padding of Section 3.1 exists. Padding belongs to the problem, not to a method: it is applied once, before any assignment, so every solver receives the identical instance, and only the cloud with fewer splats is padded. It is not a formality. The padded side is 1 to 97% clones (median 70%), and the repeated rows change the difficulty by different amounts for different solvers (Section 4.1).

Two clone-placement policies are implemented, measured on a ring-to-cross toy where the target keeps half its splats. The default pads with clones of randomly chosen splats and then matches everything, so a fading splat can be sent anywhere in the cloud. Matching the real splats first and cloning each surplus at its nearest real splat of the other side keeps the fade on the surface, cutting the mean travel of the clone pairs from 0.40 to 0.24 and the overall mean from 0.39 to 0.31. The nearest-host search costs 2 to 18× the assignment itself, so it is not the default; it is exposed in the demo.

S3 The co-bisection in detail

The shared axis. Choosing each cloud's own widest axis instead of the shared one pairs geometrically unrelated halves whenever extents are nearly tied; measured on matched shells, mean travel is roughly 1 against 0.02.

Where splats go. The recursion separates a splat from its own neighbors when they fall on opposite sides of a cut, but source and target of a pair descend the same branch together. Splats can be matched far apart either by global displacement, since even an all-straight descent carries the source's spatial ranking onto the target's, or by a crossing decision high in the tree.

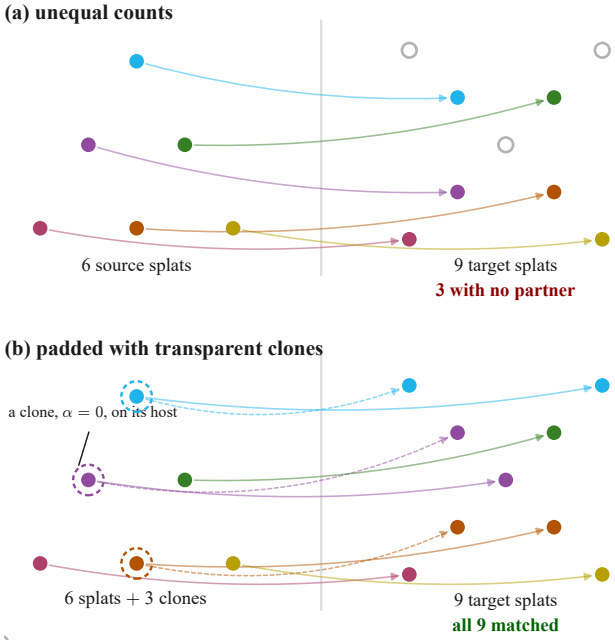

Figure S1: Why the padding exists, on six splats into nine. *(a)* The optimal partial matching leaves three target splats with no partner, which would have to appear out of nothing. *(b)* Three zero-opacity clones, dashed rings on the hosts they copy, make it a bijection: every target splat is used once and both endpoints still render exactly. Padding changes the problem, so two of the six real pairs move.

The 3D toy. On a 3D torus-to-sphere pair (Figure S2) the same thing happens at a 13.6% gap: mean travel is 0.35 against the optimum's 0.33, and the excess sits in a tail, 37 splats past 0.74 against the optimum's 15.

Two ablations of the node rule. On parked prototypes, the simple rule is near the best of its family. Replacing the binary node with a 2×2 co-partition whose 4×4 block assignment is solved exactly changes mean travel by 0.2% on cat to bee at $N = 4 \times 10^5$. A grid-accelerated 2-opt polish at the same size fires up to 6×10^4 profitable swaps and moves mean travel by about 0.15%. Better coarse decisions and local repairs both saturate.

Table S1: Every unordered pair of the eight captures at full render density, FSM at its default $\omega = 1/4$. N is the paired count (the larger side after culling; the smaller is clone-padded, Section 3.1). Times are the fastest of nine builds of the shipped single-threaded JavaScript, each in its own process; the $\omega = 0$ cells of the same grid are quoted as ranges in Sections 3.3 and 4.3.

pair	N	time	travel	ΔE_{76}
cat to textile ball	1.19M	1.9 s	0.520	38.0
cat to bee	1.19M	1.9 s	0.214	42.6
cat to BMX	1.19M	1.9 s	0.420	40.5
cat to fountain	1.19M	2.0 s	0.296	32.4
cat to plant	1.19M	2.0 s	0.468	34.7
cat to shell	1.19M	1.9 s	0.299	29.6
cat to tarot box	1.19M	1.9 s	0.540	45.1
bee to textile ball	950k	1.3 s	0.505	30.1
bee to BMX	950k	1.4 s	0.445	40.7
bee to fountain	950k	1.4 s	0.322	32.0
bee to plant	950k	1.4 s	0.389	40.0
bee to shell	950k	1.4 s	0.364	33.3
bee to tarot box	950k	1.4 s	0.511	34.4
textile ball to BMX	382k	400 ms	0.522	29.9
textile ball to fountain	382k	386 ms	0.542	20.9
textile ball to plant	382k	393 ms	0.493	33.8
textile ball to shell	382k	383 ms	0.453	26.7
textile ball to tarot box	382k	380 ms	0.508	30.2
shell to BMX	187k	184 ms	0.517	28.8
shell to fountain	187k	184 ms	0.443	23.8
shell to plant	187k	181 ms	0.516	28.2
shell to tarot box	187k	178 ms	0.558	36.2
tarot box to BMX	114k	110 ms	0.432	45.7
tarot box to fountain	114k	108 ms	0.473	32.3
tarot box to plant	114k	104 ms	0.299	36.7
plant to BMX	113k	106 ms	0.384	38.8
plant to fountain	113k	106 ms	0.448	31.5
BMX to fountain	90k	93 ms	0.553	33.0

Timings. Repeat runs on this machine move by up to 18%, which bounds any claim about the exponent. Across the measured decades $N \log N$ and N differ by a factor of two in total, an extra log-log slope of 0.10, so the timings cannot distinguish them; the guide under our curve in Figure 5 is therefore the complexity class of the construction, not a fit. The exact curve of that figure has no exponent to read either: it is the envelope of the fastest *feasible* exact solve at each count, and the 93× jump in its last step is the network simplex running out of memory and the ladder falling back to a much slower solver.

S4 Timing harnesses

Table 1's two time columns come from one `numpy` harness for the fast methods, at $N = 4000$ per side on the cat-to-bee instance and at full render density, so a difference between two of those rows is a difference between assignments, not between languages. That harness is not what an artist waits for: the same co-bisection ships as single-threaded JavaScript, whose median build over these twenty-eight pairs is 0.40 s against the port's 29 s, and 2.0 s on the slowest of them. The two agree to 0.03% in transport cost. One harness equalizes the language but not the shape of the work: a rank pairing is one vectorized sort at C speed, while a recursion pays interpreter cost at each of its $2N$ nodes. This accounts for most of the distance between Morton's 0.27 s and our 29 s there,

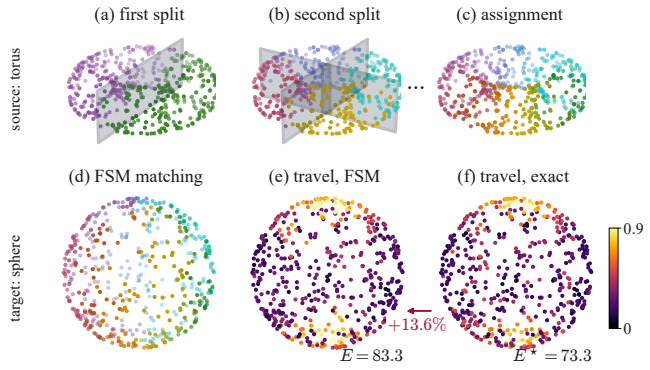

Figure S2: The recursion of Section 3.2 in 3D, a torus mapped to a sphere (512 splats each, $\omega = 0$; the toy carries no color). *Top, the source:* the first split into two blocks, the second into four, and the limit of that sequence, the assignment. A block keeps the color its points already carry, so a split resolves a color into two neighboring ones; the wheel is Figure 3's. *Bottom, the target:* the sphere wearing each splat's matched source color, then the travel every splat pays under FSM and under the exact optimum, on one shared scale, each panel carrying its plan's energy (1).

and for why the compiled build of the same algorithm is 74 times faster than the interpreted one. The reference solvers each need the device they were written for and are compared only with each other. Table S1 lists the default assignment's build time, travel and recolor at full render density for every pair; the $\omega = 0$ cells of the same grid are quoted as ranges in Sections 3.3 and 4.3.

S5 The accuracy ladder

Computing an optimum to measure against caps the experiment at $N = 4 \times 10^4$; that row cost 8745 seconds and 17.1 GB, still 30 times smaller than the captures the paper works at. One harness runs every method on the *same* augmented point sets of the cat-to-bee pair, each timed in the best implementation available to it; the host has 34 GB of memory. Unlike Table 1, Table S2 asks what each method costs at its best. Repeat runs move by up to 18%, so every time here is a single run and no claim turns on a factor under two.

Which exact solver runs is a question of constants: on identical matrices the four solvers we tried return the same optimum but differ by 12.3× in time and 3× in memory, so the ladder uses whichever still fits at each count (Section S6). Memory is what ends the experiment: the cost matrix alone is 51 GB at 8×10^4. Every memory figure here is a measured peak for the whole solve, not the $8N^2$ bytes of the matrix inside it.

Both FSM and refinement improve with N and converge (ours 12.3, 11.1, 8.9, 8.0% against refinement's 17.1, 17.7, 14.3, 9.9% from $N = 10^3$ to 2×10^4). Which sits closer is pair-dependent, itself a symptom of the near-degeneracy: on BMX to tarot at $N = 4000$ refinement is the more accurate, 6.3% against our 13.7% on the balanced subsample this ladder uses, while the compute difference is the same.

Table S2: What each assignment costs and what it buys, as the splat count grows (cat-to-bee, $\omega = 1/4$). Each cell gives compute time, and under it the extra transport energy (1) over the row's reference and the speed ratio to it. The reference is the feasible solver with the lowest energy at that count, whatever it costs, and its cell carries the absolute energy E. Read with Section 4.1.

N	reference	exact	HiRef	entropic OT	FSM (ω=0)	FSM
2.0k	exact	937 ms	48.1 s	173 ms	2 ms	2 ms
		reference E=136.8	+17.7% 51× slower	+5.0% 5.4×	+70.0% 496×	+11.1% 508×
4.0k	exact	5.1 s	89.9 s	347 ms	5 ms	4 ms
		reference E=286.9	+14.3% 18× slower	+4.5% 15×	+65.4% 1.0×10^3	+8.9% 1.2×10^3
20k	exact	94.4 s	12 min	16.8 s	14 ms	16 ms
		reference E=1359.5	+9.9% 8× slower	+3.1% 5.6×	+70.5% 6.7×10^3	+8.0% 5.9×10^3
40k	exact	146 min	21 min	2 min	28 ms	28 ms
		reference E=2694.8	+8.3% 7.0×	+2.6% 79×	+72.6% 3.1×10^5	+7.8% 3.2×10^5
80k	entropic OT	*not feasible*	48 min	23 min	61 ms	59 ms
			+5.6% 2× slower	**reference** E=5458.6	+69.6% 2.3×10^4	+5.6% 2.4×10^4
200k	*ours only*	*not feasible*	*not run*	*not run*	152 ms	156 ms
					+60.1% 1.0×	*best measured* E=14453.6
800k	*ours only*	*not feasible*	*not run*	*not run*	827 ms	863 ms
					+60.6% 1.0×	*best measured* E=57779.3

The reference, once the optimum is gone. Table S2's reference is, at each count, the feasible solver with the lowest energy: the exact optimum through 4×10^4 and entropic OT at 8×10^4. A feasible plan bounds the optimum from *above*, so scoring against it understates every error; at $N = 4000$ it would report FSM at 4.2% when the true gap is 8.9%. From 2×10^5 on nothing else was measured, so the table marks those rows *ours only*; *not feasible* is a wall, and *not run* a solve we chose not to pay for. At those sizes entropic OT's cost lies in the rounding, not the sweep. Its soft plan has to become a bijection, and the free block left after each source claims its argmax is an assignment problem of its own, 23% of splats at 4×10^4 and 27% at 8×10^4. Solving that block exactly keeps the reference accurate and dominates its time, 91 of the 110 s at 4×10^4 and 1335 of the 1408 s at 8×10^4. Closing it greedily is nearly free but 20 points worse, 22.7% above the optimum at 4×10^4 where the exact repair is at 2.6%. We round exactly at every count and charge the repair to its time. As a *bijective* method, entropic OT is therefore five to nineteen times slower than its own transport solve.

A certificate, where no reference can reach. Past $N = 4 \times 10^4$ no reference in the table reveals how far the winner is from the true optimum, but the dual bounds it. Any potentials with $f_i + g_j \le c(i, j)$ satisfy $\sum_i f_i + \sum_j g_j \le E(\sigma^\star)$, since $E(\sigma) = \sum_i c(i, \sigma(i)) \ge \sum_i \big(f_i + g_{\sigma(i)}\big)$ for every permutation. A few streaming passes of $g_j \leftarrow \min_i (c(i, j) - f_i)$, alternated with the same step in f, project Sinkhorn's near-feasible potentials onto exact feasibility, so the bound is *certified* at any count the sweep can reach. The bound is tight wherever the exact optimum is also computable: LB sits within 1% of it at $N = 2 \times 10^3$, 4×10^3 and 2×10^4 alike. Against it, FSM is at most 8.8, 8.9 and 8.6% above the optimum at $N = 4 \times 10^4$, 8×10^4 and 2×10^5, flat across the range, while entropic OT is at most 3.5 and 3.1% at the first two. The bound at 4×10^4 was recorded before the exact solve at that size existed, which makes it an out-of-sample test of the certificate. The solve later returned $E(\sigma^\star) = 2694.8$, putting FSM at 7.8% and entropic OT at 2.6%, inside the certified 8.8 and 3.5% and loose by about the one point of slack the bound shows wherever both are computable.

S6 The two reference solvers

Four exact solvers on the same matrices return the identical optimum, so the choice is a constant-factor question whose answer changes with N. A network simplex beats Jonker-Volgenant by 5.4× at $N = 8000$ and 12.3× at 2×10^4, where the same solve falls from 19 minutes to 94 seconds. It holds, however, about four copies of the cost matrix against Jonker-Volgenant's one and a third, both measured. The ladder therefore uses whichever still fits: Jonker-Volgenant below $N \approx 3000$, where its smaller constant wins, the simplex at 4×10^3 and 2×10^4 (there at 13.3 GB), then Jonker-Volgenant again at 4×10^4, which needs 17.1 GB where the simplex would need 54 and pays 8745 s for the row. At 8×10^4 neither can start.

Hierarchical refinement's constant has a similar explanation. It performs one iterative low-rank solve per node of its tree with a fixed iteration count, so a node costs about the same whether it holds ten points or ten thousand. At $N = 10^3$ its schedule leaves eight 125-point problems, and the sub-solves together hold essentially the entire wall clock, which is why its timings look wrong at small sizes (23 s where the exact solve costs 0.16 s). Batching would lower the constant without changing the growth rate, and the authors of [1] call for faster low-rank subproblem solvers themselves. Our comparison uses one solver configuration, timed on a GPU while ours runs on the CPU: faster hardware would narrow the time difference without changing its cost, and a higher-rank schedule could lower its cost at further time expense.

S7 The full corpus

Table S3 is the per-pair accuracy the paper quotes medians of, Table S1 the build time, travel and recolor of the default assignment at full render density for every pair, and Table S4 the one count at which every method was instrumented for peak memory. Each

pair appears once because the assignment is direction-symmetric, exactly when N is a power of two and to under 0.03% in mean cost otherwise. In Table S3 the bold median marks *our* method, not the column's winner: the entropic reference is nearer the optimum on all twenty-eight pairs, at a compute cost that table does not show. The corpus measurements use one subsample draw per pair (seed 7).

S8 Frame dependence

Figure S3 is the measurement behind the first paragraph of Section 5. Rotating *both* clouds by one rotation leaves the ground cost, the optimal permutation and its energy invariant, so it isolates the heuristic's frame dependence with no confound. Ten of the pairs are each re-solved in 16 random shared orientations.

S9 Where the method breaks

Twenty-eight pairs are twenty-eight data points, all chosen because we wanted to morph them, so we also searched for the failure on purpose. We scored FSM against the exact optimum on all 28 unordered pairs of eight 2D shapes at $N = 2048$, then swept the relative orientation of the worst pair and the best one (Figure S4). It shows two effects, and only the second is a failure.

At a common orientation the geometry-only gap runs from 0.7% (ring to bar) to 28.5% (star to cross), a factor of 39. The top of that range is not an adversarial worst case: only 4 of the 28 pairs exceed our capture corpus, 14 of them sit *below* its minimum, and the median 2D pair is at 5.9% against the corpus's 12.5%. The shape search mostly varies the denominator: across the same 28 pairs ρ (8) stays between 98.3 and 100.0%, a spread of 1.7 points against the gap's factor of 39. A thin shape has a cheap optimum, not a hard one.

Relative rotation is the failure. Turning one cloud against the other takes ring to bar from 0.7% to 34.4% at 45°, a factor of 47, and star to cross from 28.5% to 66.7% at 20°. Unlike the shape search, this also moves ρ, from 99.7 to 87.3%: the plan itself got worse, not merely its denominator cheaper. A median cut along a coordinate axis follows a bar that lies along one and cannot follow the same bar at 45°. The effect is not an artifact of the rotated cloud's extent: re-running the sweep with each rotated cloud rescaled to its own bounding box leaves both peaks standing and slightly higher, 66.7 → 69.7% and 34.4 → 43.4%. Exactly balanced splits and thin limbs compound: thin-limbed shapes reach 28.5% above the optimum with no rotation at all.

S10 The lightness weight

Figure S5 is the sweep summarized in Section 3.3; mean travel accelerates past $\omega = 1$. On a toy of independent two-splat blocks whose targets hold the source's positions with the brightness swapped, each block can either stay (no travel, full recolor) or swap (no recolor, travel equal to the block's width), and as ω grows the blocks flip narrowest first, cheapest brightness fix taken first. Figures S5 and 4 are that trade measured on real pairs.

S11 Correspondence diagnostics

On one pair (cat to tarot box at $N = 4 \times 10^5$), the *random* pairing is instant but moves splats a mean 0.804 and a worst pair 1.75, most of the object diameter. The co-bisection cuts mean travel to 0.539 and worst-pair travel to 0.91, and the lightness coordinate then drops recolor from ΔE 58.5 to 45.2 for a 0.2% travel premium, the only one of the three that lowers recolor at all. Figure S6 shows the same comparison through two correspondence diagnostics of our demo. The travel row makes the coarse-split structure of Section 3.2 legible: what brightness the co-bisection leaves sits in axis-aligned blocks, the seam geometry named in Section 5, where random scatters it everywhere. The NOCS row is the coherence the paper reports as S: random transfers color at random, and both FSM columns transfer smooth regions. The coherence the lightness term gives up is too small to see at this scale; the two FSM columns differ by two levels of 255. Both diagnostics are live in the demo.

S12 Which pairs morph well

The exact minimum-cost bijection between two clouds is affordable at a stable subsample ($n = 1500$ per cloud, about a second), so we solved it for every pair of the 71 objects of Section 4.3 and read the result as a property of the two inputs, independent of any method (Figure S7). Objects of the same kind dominate the head of the ordering, with the same-kind share among the k closest pairs starting near 60% and decaying onto a base rate of 6%. Figure S8 draws five pairs from that head, and their middles stay legible; one solve answers the question before any assignment is built.

Figure S9 repeats the layout of Figure S11 on five more same-kind pairs, ten objects that appear nowhere else. The two galleries differ in how their rows were chosen as well as in kind, and two of Figure S11's five rows are same-kind too, so this is a second look, not a controlled comparison. The travel panels show where the two differ: a rug into a cat spends its budget everywhere at once, while a jackdaw into a kiwi pays in the front body and the neck and leaves the beak and the rear cool. With something in common, the correspondence is local and so is the cost.

S13 Further galleries

Figure S10 spans six of our eight captures under the same assignment and trajectory as the paper, with no pair chosen for being favorable. The last row is the extreme count mismatch of the corpus, cat to fountain, where 97 of every 100 lion-side splats are zero-opacity clones. The mid-morph frames spread further than the same-kind rows: with no shared structure between two captures, minimum travel spreads one object through the other. Across these galleries the coherent trajectory compacts mid-morph silhouettes and softens the co-bisection's seams, though it cannot repair a scrambled assignment, consistent with prior reports that regularization is not optional for splat interpolation [2, 4]. Figure S11 morphs onto and between generated objects.

S14 Why the gap needs ρ

Two 2D pairs are why we report ρ (8) beside the gap. FSM sits 20.1% above the optimum on a heart-to-star pair against 5.5% on the ring-to-cross at the same N and in the same frame, nearly four times the gap, and yet the two assignments are visually indistinguishable and ρ puts them within half a point of each other, 98.7% against 99.2%. The pair is not harder; its *optimum* is cheaper, because a filled star sits almost entirely inside a filled heart, so $E(\sigma^\star)$ is 40.4

Table S3: The optimality gap on every pair of the corpus: 28 exact solves at $N = 4000$ splats per side, seed 7, $\omega = 1/4$, each pair defining its own reference. *clones* is the share of the paired cloud that is zero-alpha padding at full render density (Section 3.1). *Left:* the gap (7) for FSM at both weights, the entropic reference, and the three fast heuristics of Section 4.2; *random* is omitted, at 126 to 656%. *Middle: drift* between where FSM and the optimum send the same splat, in units of the optimum's mean travel, and the excess mean travel. *Right:* the neighborhood distortion S (6). Bold marks the median for *our* method, not the winner of its column.

		optimality gap (%)						FSM vs the optimum		distortion S	
pair	clones (%)	FSM	ω=0	entropic	greedy	sliced	Morton	drift	travel	FSM	exact
cat to bee	20	9.2	65.7	2.3	232	457	552	0.44	+3.6	4.09	3.38
cat to ball	68	8.4	17.0	0.2	91.8	150	123	0.58	+5.7	10.96	9.80
cat to shell	84	9.1	26.6	0.2	344	412	338	0.44	+3.5	5.14	4.41
cat to tarot	90	5.3	12.5	0.0	58.3	77.8	70.4	0.51	+2.7	11.17	10.75
cat to plant	91	9.4	16.4	0.0	50.2	96.3	90.3	0.61	+3.5	10.57	9.43
cat to BMX	92	15.1	23.4	0.0	62.7	101	64.6	0.60	+8.4	6.53	5.80
cat to fountain	97	16.3	30.3	0.0	203	332	292	0.59	+4.7	6.17	5.20
bee to ball	60	12.3	34.1	0.3	70.6	188	176	0.56	+5.0	9.12	8.17
bee to shell	80	8.9	33.7	0.1	210	268	210	0.47	+4.7	7.22	5.90
bee to tarot	88	7.5	26.0	0.0	70.2	109	84.9	0.41	+1.7	8.41	8.55
bee to plant	88	13.7	33.4	0.1	57.7	148	98.1	0.63	+4.6	8.05	9.59
bee to BMX	91	13.3	28.0	0.1	73.4	96.2	70.3	0.60	+8.2	7.16	6.09
bee to fountain	96	13.2	42.3	0.1	97.0	257	272	0.48	+3.9	6.43	5.60
ball to shell	51	17.2	34.1	1.1	101	289	204	0.47	+9.7	2.71	2.27
ball to tarot	70	6.8	24.3	0.5	77.8	262	207	0.27	+1.9	2.65	2.28
ball to plant	70	12.8	25.7	0.3	54.6	228	200	0.40	+5.1	2.89	2.58
ball to BMX	77	12.9	22.0	0.1	31.5	153	80.7	0.38	+6.4	2.20	2.07
ball to fountain	90	6.7	16.0	0.2	32.7	176	182	0.23	+1.8	2.36	2.06
shell to tarot	39	6.0	16.3	0.4	76.2	129	94.1	0.38	+1.6	4.35	4.54
shell to plant	40	10.7	20.0	0.3	45.6	137	111	0.43	+3.8	4.36	4.50
shell to BMX	52	16.0	25.1	0.2	37.8	113	74.5	0.49	+9.0	3.53	3.34
shell to fountain	79	13.9	27.0	0.4	62.0	260	208	0.48	+7.2	3.72	3.63
tarot to plant	1	17.4	54.9	4.6	159	542	238	0.58	+8.8	4.77	3.86
tarot to BMX	22	13.7	27.8	1.2	44.8	198	127	0.46	+6.9	3.84	2.82
tarot to fountain	65	5.2	20.7	0.2	80.4	192	89.3	0.26	+1.1	3.82	3.23
plant to BMX	21	12.2	26.3	0.7	62.1	225	129	0.46	+5.6	4.77	3.96
plant to fountain	65	16.8	29.4	0.4	105	183	152	0.56	+8.5	5.28	5.17
BMX to fountain	55	21.5	29.8	0.2	67.8	102	78.3	0.69	+13.0	8.03	8.72
median	70	**12.5**	26.5	0.2	70.4	185	128	0.48	+4.9	4.96	4.52

Table S4: Accuracy against ground truth, and what ground truth costs. Error is the optimality gap (7) in the objective (1) at $\omega = 1/4$, which every solver here minimizes, so the exact solve is optimal for it by construction. Memory is the measured peak of the whole solve on whichever device it runs; the two FSM rows carry an analytic footprint instead, since a process meter cannot resolve 0.2 MB. Each method is timed in its own best implementation, ours in the browser engine that ships it, so bold marks the lowest error only: time and memory are not ranked.

cat to bee, $N = 20 \times 10^3$ splats per side, 59× below capture density; a true optimum reaches 40×10^3 (Table S2)

			peak	grows as	
method	error	time	mem.	time	mem.
FSM (ω=1/4)	+8.0%	16 ms	0.2 MB	$N \log N$	N
FSM (ω=0)	+70.5%	14 ms	0.2 MB	$N \log N$	N
Sinkhorn, GPU	**+3.1%**	16.8 s	2.0 GB	N^2	N
HiRef [1], GPU	+9.9%	12 min	34 MB	$N \log N$	N
exact, ground truth	0, by def.	94.4 s	13.3 GB	N^3	N^2

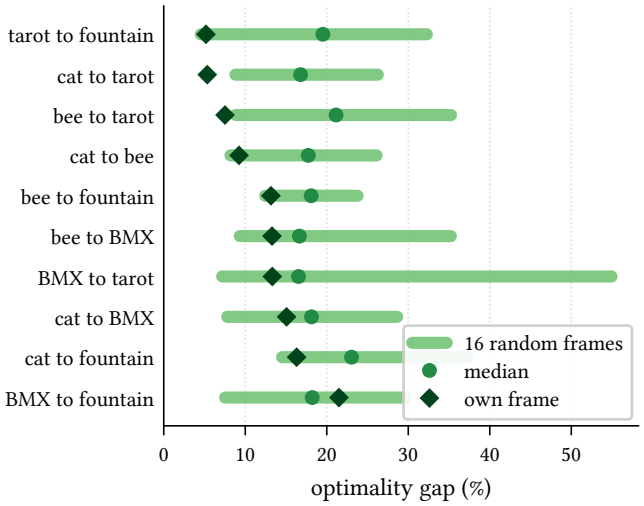

Figure S3: Every accuracy number in the paper is measured in one frame, and the frame matters. Ten capture pairs are each re-solved in 16 random shared orientations at $N = 4000$: the bar is the range of the optimality gap over them, the circle their median, and the diamond the gap in the frame the captures arrived in. On nine of the ten pairs the capture's own frame is better than the median random one, and the widest pair spans 7 to 55%. This is a *common* rotation of both clouds, which leaves the optimum invariant and changes only the axis-aligned partition; Figure S4 measures a rotation of one cloud against the other, which is worse.

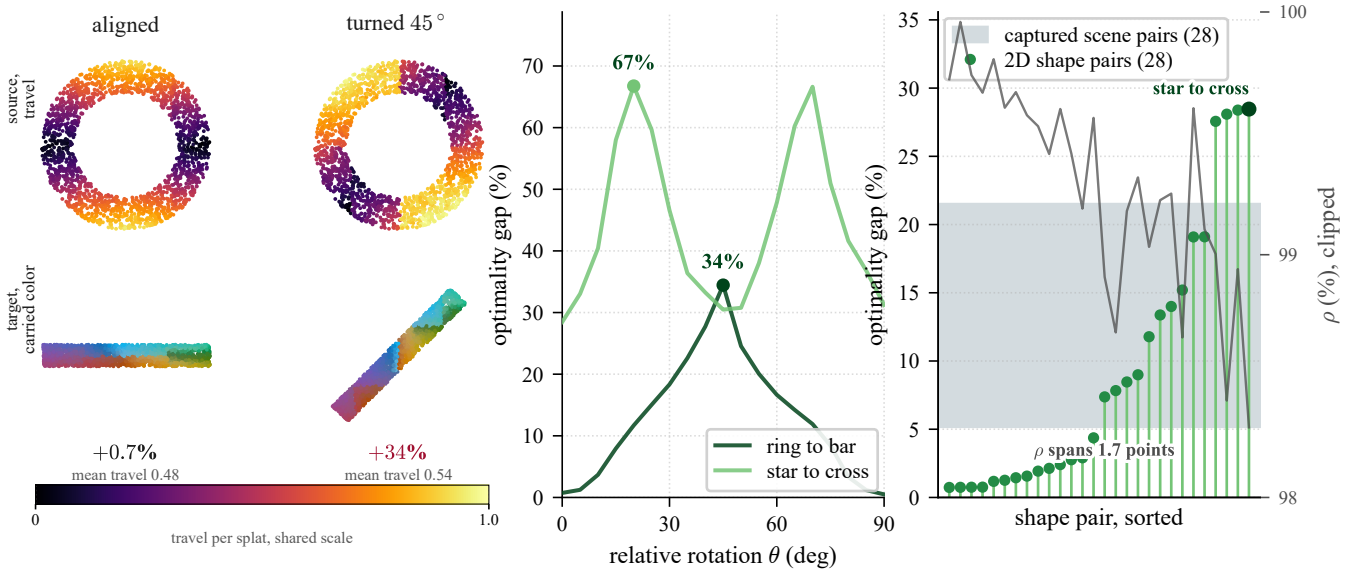

Figure S4: Failure cases found by search ($N = 2048$ per cloud). *Left:* the worst configuration, a star into a cross rotated by 20°, under FSM and the exact optimum; sources are shaded by each splat's travel, targets carry their matched source's position color. *Middle:* the relative-orientation sweep. *Right:* all 28 shape pairs, sorted, against the band of the captured pairs.

against 117.7, and a similar absolute sloppiness divided by a smaller number reads as a larger percentage. The excess is also diffuse, so no single error is visible: here FSM is worse than the optimum on 56% of splats and the worst 5% of those carry only 34% of the total excess.

S15 Reproduction

Everything in the paper and this supplement is regenerated from the public repository (https://github.com/marcelpadilla/fast-splat-morphing), except the reference solver of [1], which the harness

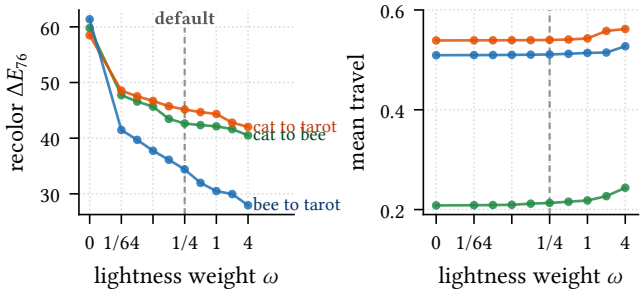

Figure S5: Sweeping the lightness weight ω on three pairs at $N = 4 \times 10^5$ (ω on a log axis, 0 leftmost): recolor on the left, mean travel on the right, the dashed line at the default.

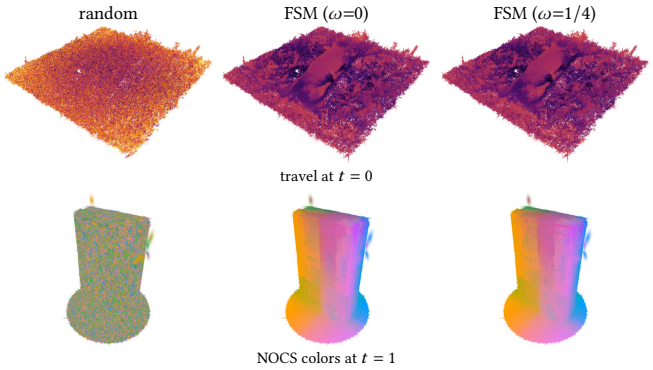

Figure S6: Three assignments through two correspondence diagnostics (cat to tarot box, $N = 1.2{\times}10^5$ for legibility). *Top:* the source at $t = 0$, each splat shaded by the travel it carries on a shared scale, so bright is far. *Bottom:* the target at $t = 1$ wearing NOCS position colors [5] carried from its matched source splats, so a coherent assignment transfers smooth regions and a scrambled one noise.

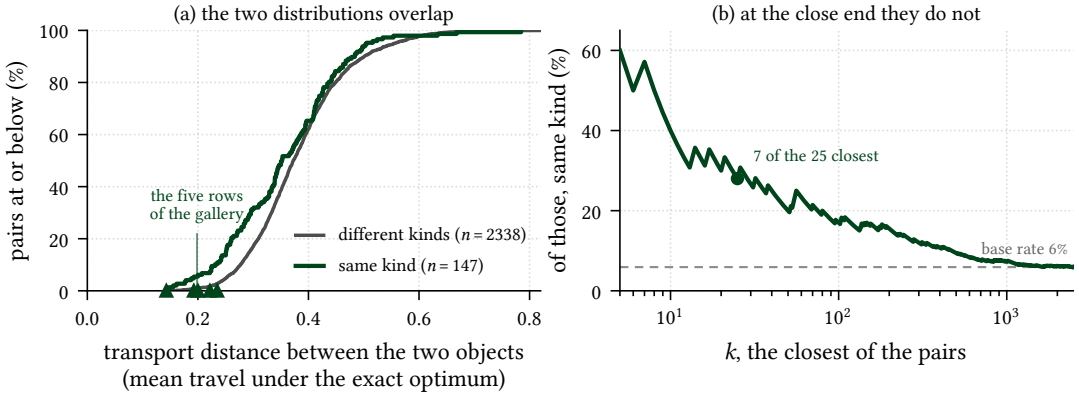

Figure S7: How far apart two objects are before any method touches them: the exact minimum-cost bijection between every pair of the 71 objects of Section 4.3, in the frame the demo morphs in, $n = 1500$ per cloud and 2485 pairs. *(a)* Being the same kind of thing is a weak predictor on average: the two distributions overlap through most of their range. *(b)* It is a strong predictor at the close end, the end that matters: among the k closest of those pairs the same-kind share starts near 60% and decays onto the base rate of 6%; at $k = 25$ it is 7 pairs. The five rows of Figure S8 are marked in *(a)* and are all inside that head.

calls but does not redistribute. figures/make_all.py rebuilds every figure and table, and code/capture.py re-renders every strip headlessly from the demo. The scripts behind individual results:

- code/gap_allpairs.py: the per-pair exact solves and gaps (Table S3); code/random_floor.py: the stability of $E(\sigma_0)$ across random draws.
- code/bench_hier.mjs: the shipped JavaScript's build times, travel and recolor (Table S1), including the cat-to-tarot comparison (`compare`) and the render-density distortion (`traj`); code/parity_hier.py: its agreement with the `numpy` port.
- code/run_exact.py and code/bench_lap.py: the exact solves and the solver comparison; code/reround_entropic.py: the

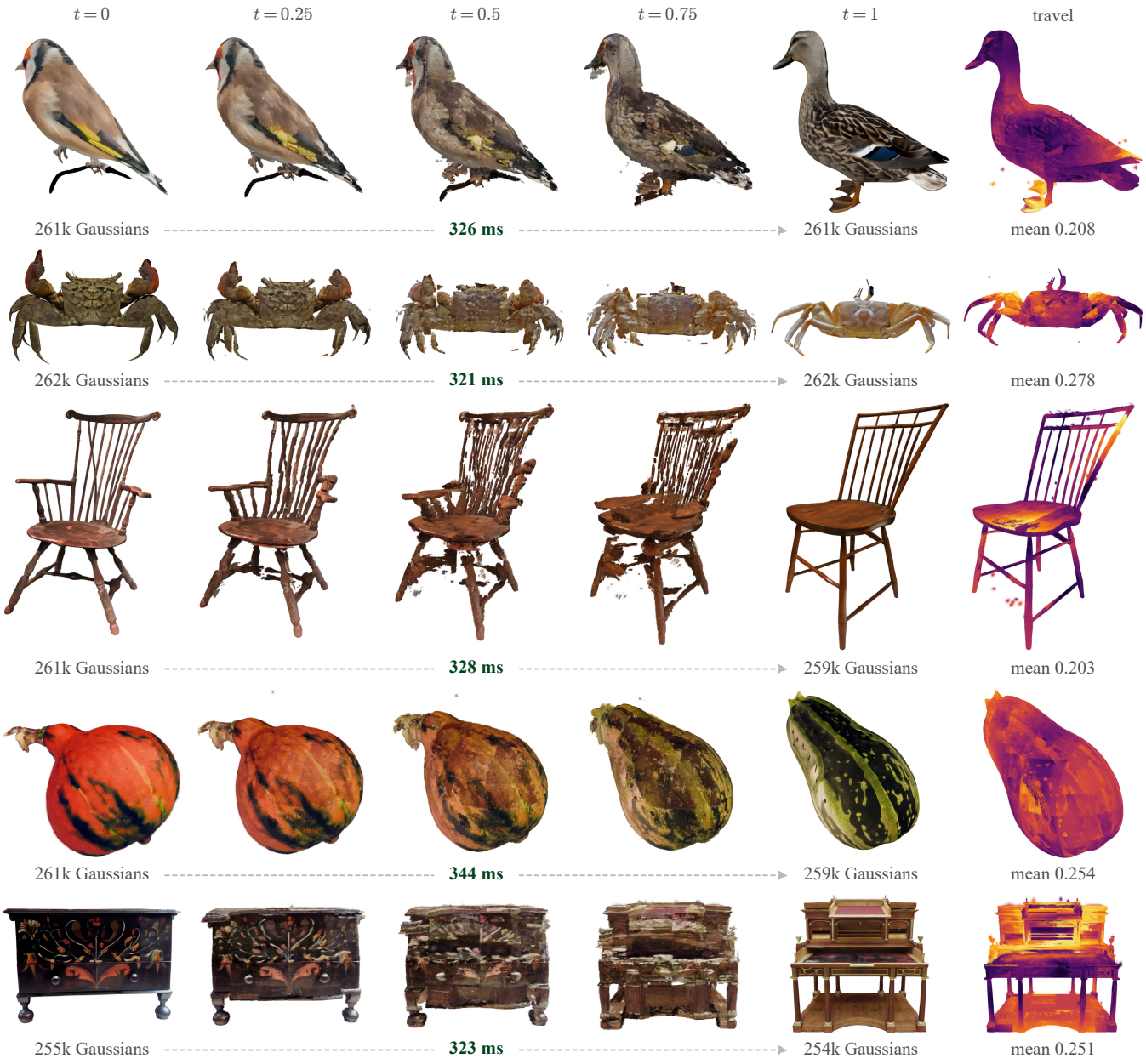

Figure S8: Morphs between objects of the *same* kind at $t = 0, 0.25, \ldots, 1$, default FSM assignment and motion-coherent trajectories, each object capped at 4×10^5 splats. *Rows:* goldfinch to mallard; crab to ghost crab; Windsor armchair to Windsor side chair; red squash to striped gourd; painted chest to rolltop desk. Every mid-morph frame is still recognizably a bird, a crab, a chair, a gourd or a cabinet. All five pairs sit at the low end of Figure S7, where the assignment has real correspondence to find instead of a global rearrangement to pay for. All ten objects are image-to-3d generations from our public dataset [3], outside the measured corpus. The last panel is the target at $t = 1$ shaded by each splat's travel, on one scale across rows, running to 0.65, with the row's mean below it.

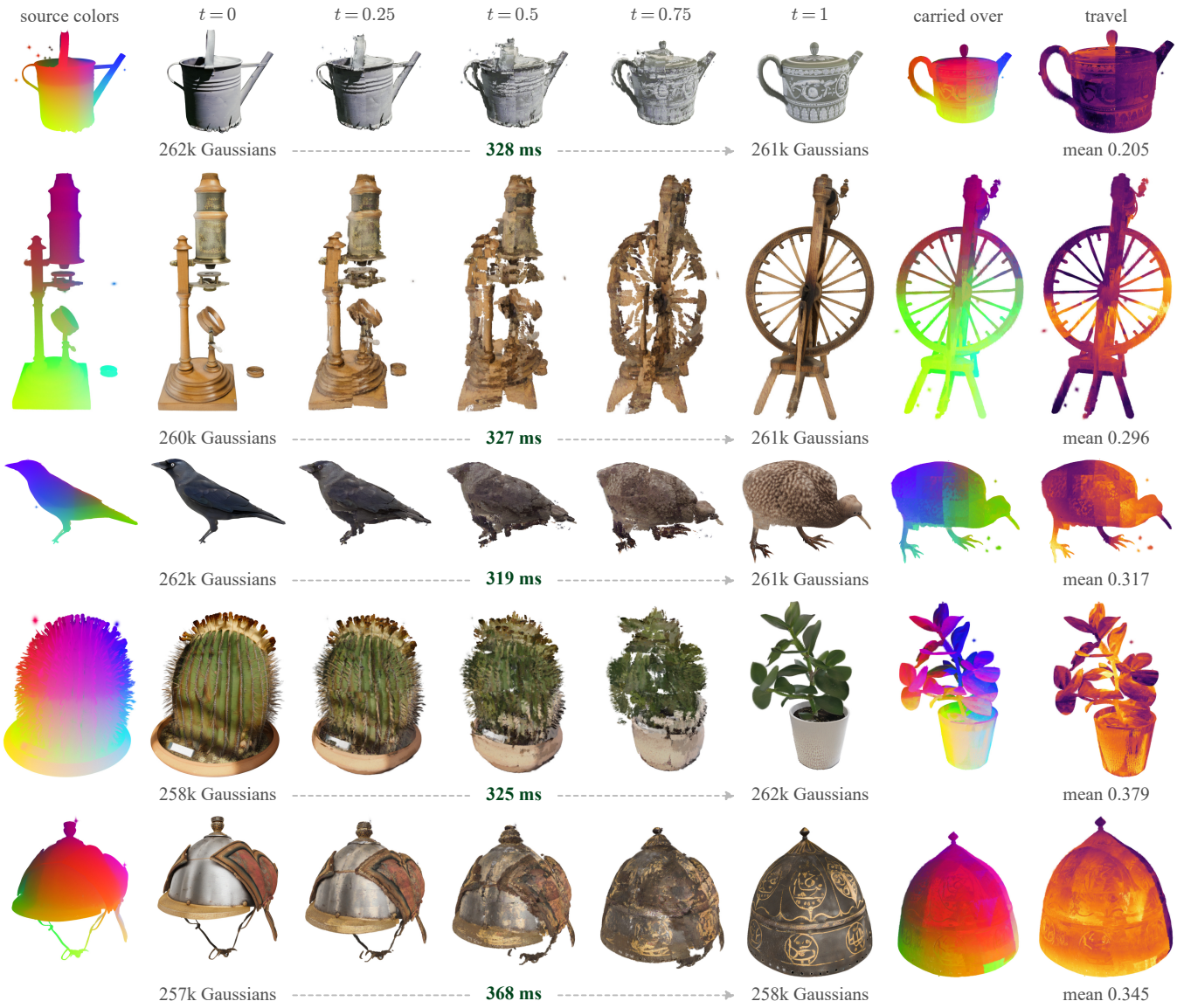

Figure S9: Five more same-kind pairs, ten objects that appear nowhere else in this paper, at $t = 0, 0.25, \ldots, 1$ under the default assignment and motion-coherent trajectories, each capped at 4×10^5 splats. *Rows:* a zinc watering can to a Wedgwood teapot; a brass microscope to a spinning wheel; a jackdaw to a kiwi; a barrel cactus to a jade plant; a cavalry helmet to a gilded one. Chosen as Figure S8 was, by the exact transport distance of Figure S7: for each kind, the closest pair among objects no other figure spends (0.243 to 0.341, against a cross-kind median of 0.372). The outer panels carry NOCS position colors [5] as in Figure S11, and the last panel is the target at $t = 1$ shaded by each splat's travel, on one scale across rows running to 0.77.

Figure S10: Cross-scene morphs at $t = 0, 0.25, \ldots, 1$, default FSM assignment and motion-coherent trajectories, each capture capped at 4×10^5 splats. *Rows:* bee to shell; potted plant to tarot box; and cat to fountain lion, the extreme count mismatch of our set. No capture appears twice. The last panel is the target at $t = 1$ shaded by each splat's travel, on one scale across rows, running to 0.76, with the row's mean below it.

rounding of entropic plans; code/dual_bound.py: the certified lower bound; code/diag_impl.py: refinement's per-node cost.

- code/rot_robust.py: frame dependence (Figure S3); code/worstcase.py: the failure search (Figure S4); figures/plot_lumlift.py: the lightness toy. The pair screen of Figure S7 ships as its precomputed distances, since it reads objects outside the repository.
- code/audit_numbers.py: checks every number printed in the paper and this supplement against the stored measurements.

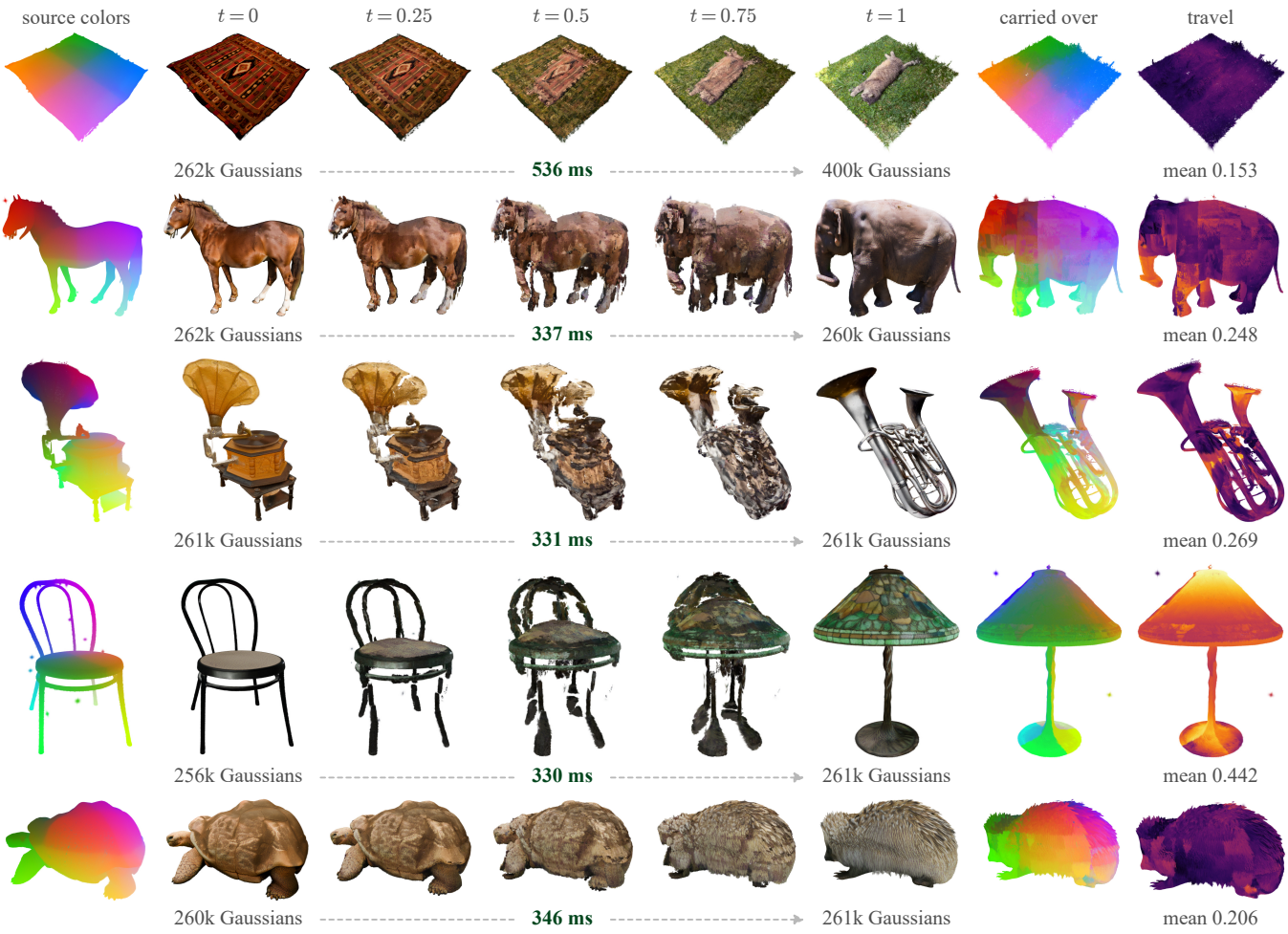

Figure S11: Morphs onto and between *generated* objects at $t = 0, 0.25, \ldots, 1$, default FSM assignment and motion-coherent trajectories, each object capped at 4×10^5 splats. *Rows:* kilim rug to a cat on a grass patch; horse to elephant; gramophone to euphonium; bentwood chair to Tiffany lamp; tortoise to hedgehog. The outer panels carry NOCS position colors [5]: the left paints the source by where each splat sits, the right shows the target wearing the colors its matched source splats carried into it. Every object except the cat is an image-to-3d generation from our public dataset [3], with an invented unseen hemisphere, outside the measured corpus; each pair was brought to a common heading by rotation about the shared up axis. The four generated-to-generated rows are count-matched at 2^{18} splats and carry no clone padding. The last panel is the target at $t = 1$ shaded by each splat's travel, on one scale across rows, running to 0.85, with the row's mean below it.

References

[1] Peter Halmos, Julian Gold, Xinhao Liu, and Benjamin J. Raphael. 2025. Hierarchical Refinement: Optimal Transport to Infinity and Beyond. In *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025 (Proceedings of Machine Learning Research, Vol. 267)*, Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (Eds.). PMLR / OpenReview.net. https://proceedings.mlr.press/v267/halmos25a.html

[2] Mengtian Li, Yunshu Bai, Yimin Chu, Yijun Shen, Zhongmei Li, Weifeng Ge, Zhifeng Xie, and Chaofeng Chen. 2025. GaussianMorphing: Mesh-Guided 3D Gaussians for Semantic-Aware Object Morphing. *CoRR* abs/2510.02034 (2025). arXiv:2510.02034 doi:10.48550/ARXIV.2510.02034

[3] Marcel Padilla. 2026. splats: a small collection of Gaussian splat objects. https://github.com/marcelpadilla/splats. Snapshot data-2026-08-11, 109 objects.

[4] Chih-Jung Tsai, Cheng Sun, and Hwann-Tzong Chen. 2022. Multiview Regenerative Morphing with Dual Flows. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XVI (Lecture Notes in Computer Science, Vol. 13676)*, Shai Avidan, Gabriel J. Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner (Eds.). Springer, 492–509. doi:10.1007/978-3-031-19787-1_28

[5] He Wang, Srinath Sridhar, Jingwei Huang, Julien Valentin, Shuran Song, and Leonidas J. Guibas. 2019. Normalized Object Coordinate Space for Category-Level 6D Object Pose and Size Estimation. arXiv:1901.02970 [cs.CV] https://arxiv.org/abs/1901.02970