Eurographics Symposium on Geometry Processing 2020
Q. Huang and A. Jacobson
(Guest Editors)

Volume 39 (2020), Number 5

Supplemental materials for *Integer-Grid Sketch Simplification and Vectorization*

Tibor Stanko[1], Mikhail Bessmeltsev[2], David Bommes[3] and Adrien Bousseau[1]

[1] Université Côte d'Azur, Inria
[2] Université de Montréal
[3] University of Bern

Contents

This file contains supplemental materials for *Integer-Grid Sketch Simplification and Vectorization*. Paper, source code and supplemental materials are available at https://repo-sam.inria.fr/d3/grid-vectorization/.

Note: The results shown for Liu et al. [LRS18] come from their paper. We ran the method by Bessmeltsev and Solomon [BS19] with default parameters and the methods by Favreau et al. [FLB16], Simo-Serra et al. [SII18], and Xu et al. [XXM*19] with a set of parameters that we adjusted to perform best overall.

References

[BS19] Bessmeltsev, M. and Solomon, J. "Vectorization of Line Drawings via Polyvector Fields". *ACM Transactions on Graphics* 38.1 (Jan. 2019), 9:1–9:12.

[FLB16] Favreau, J.-D., Lafarge, F., and Bousseau, A. "Fidelity vs. Simplicity: A Global Approach to Line Drawing Vectorization". *ACM Transactions on Graphics (Proc. SIGGRAPH)* 35.4 (July 2016), 120:1–120:10.

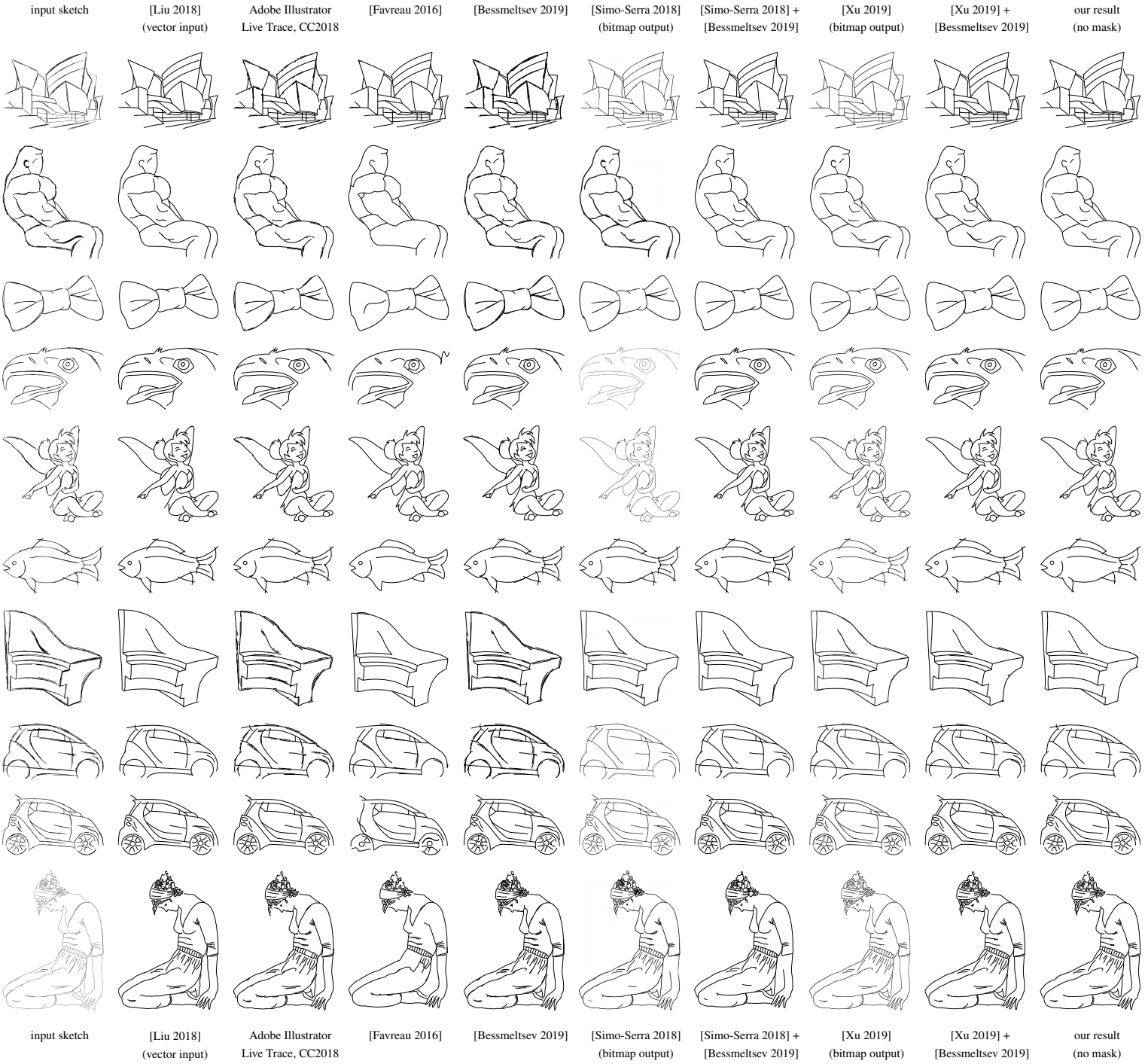
[LRS18] Liu, C., Rosales, E., and Sheffer, A. "StrokeAggregator: Consolidating Raw Sketches into Artist-intended Curve Drawings". *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37.4 (July 2018), 97:1–97:15.

[NHS*13] Noris, G., Hornung, A., Sumner, R., Simmons, M., and Gross, M. "Topology-driven Vectorization of Clean Line Drawings". *ACM Transactions on Graphics* 32.1 (Feb. 2013), 4:1–4:11.

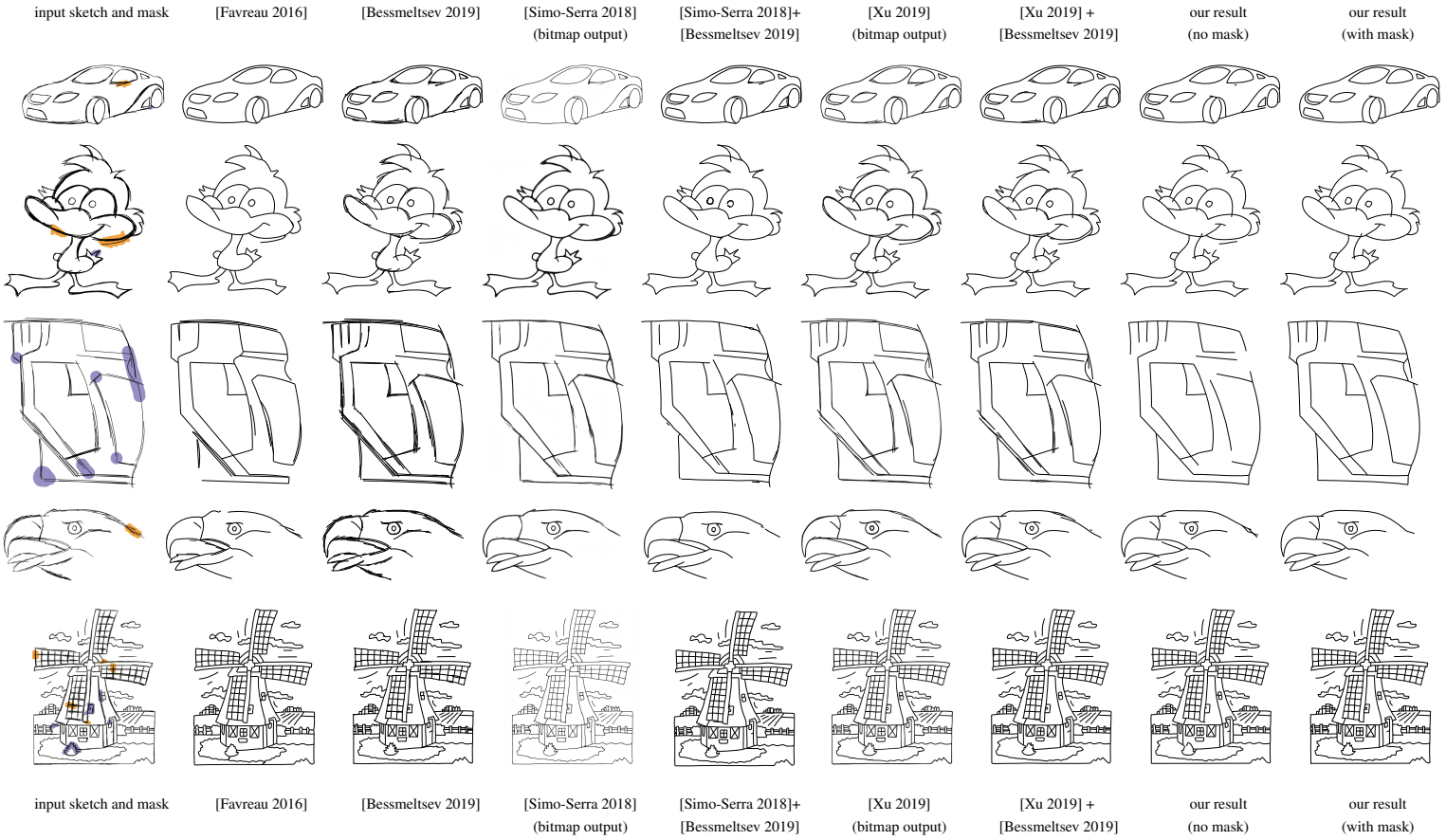
[SII18] Simo-Serra, E., Iizuka, S., and Ishikawa, H. "Real-Time Data-Driven Interactive Rough Sketch Inking". *ACM Transactions on Graphics (Proc. SIGGRAPH)* 37.4 (July 2018), 98:1–98:14.

[XXM*19] Xu, X., Xie, M., Miao, P., Qu, W., Xiao, W., Zhang, H., Liu, X., and Wong, T.-T. "Perceptual-aware Sketch Simplification Based on Integrated VGG Layers". *IEEE Transactions on Visualization and Computer Graphics* (2019).

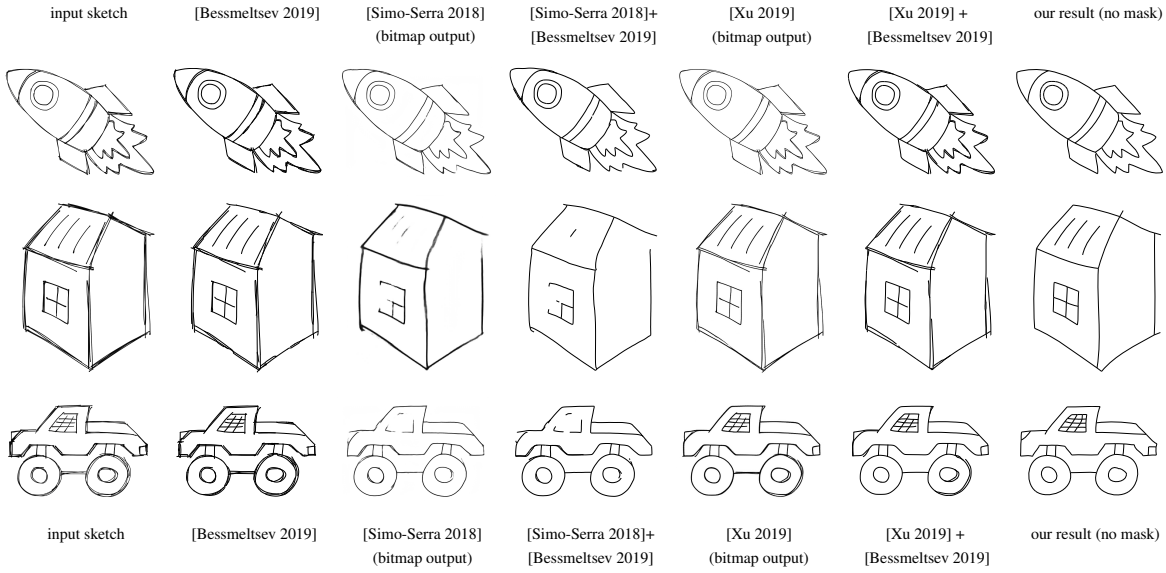
1. Comparison to previous methods (without masks)



2. Comparison to previous methods (with masks)



3. Comparison to bitmap filtering methods [SII18; XXM*19] (rough input)



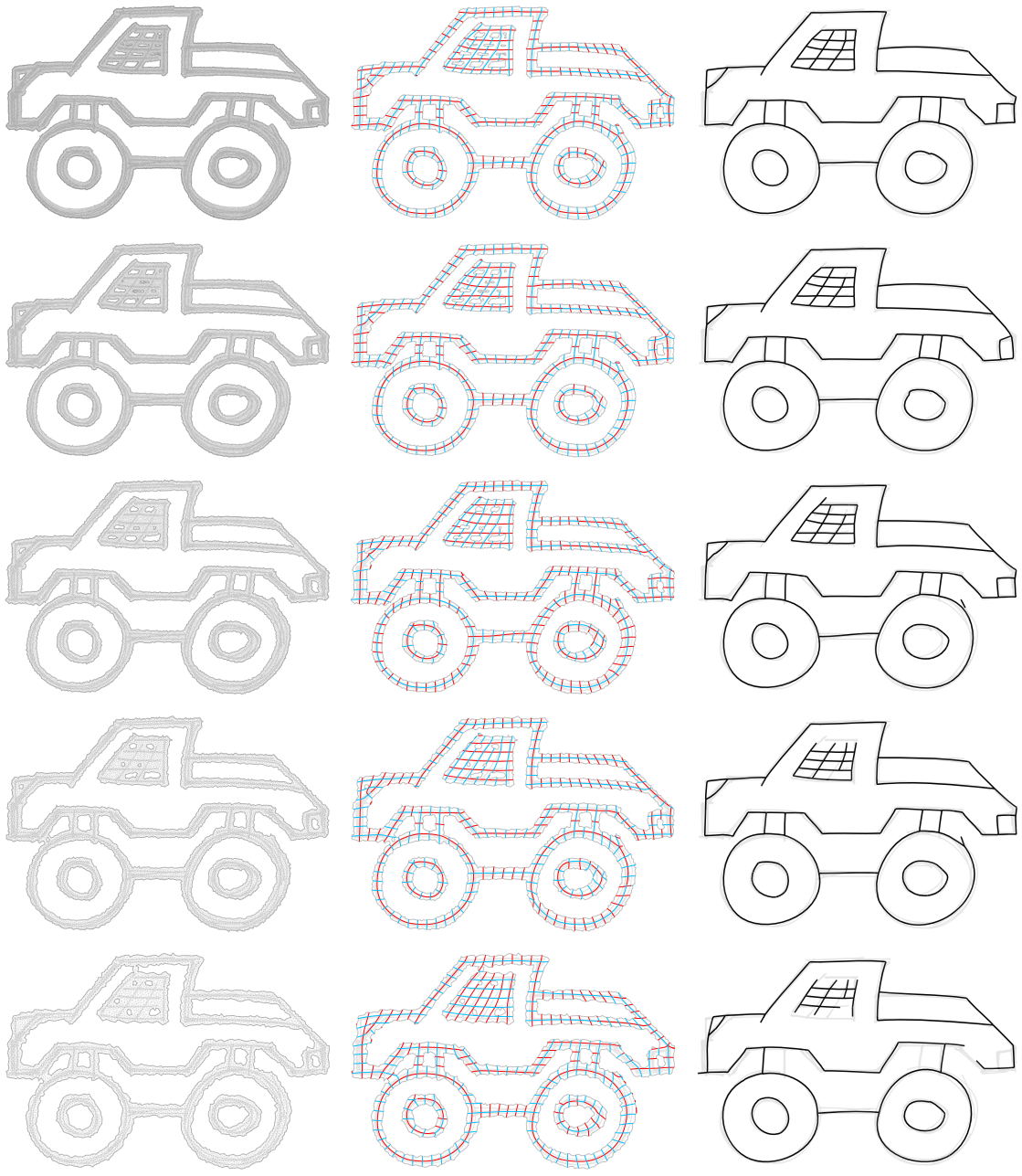
4. Comparison to vectorization methods [NHS*13; BS19] (clean input)

input sketch | [Noris 2013] | [Bessmeltsev 2019] | our result (no mask)

input sketch | [Noris 2013] | [Bessmeltsev 2019] | our result (no mask)

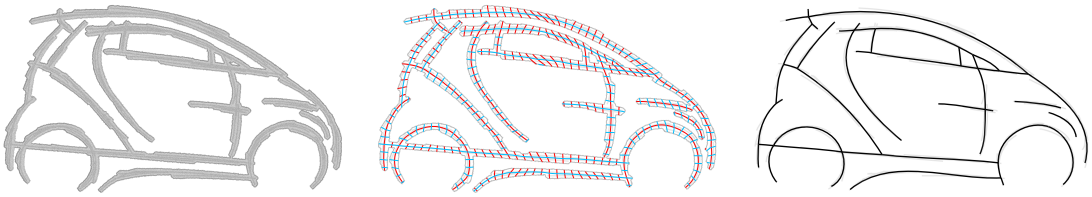
5. Parameter study: mesh resolution

When computing the triangulation, we set a bound on the size of triangles by requiring their circumdiameter to be smaller than $x\bar{\omega}$. Here, $\bar{\omega}$ is the average stroke width, and x is set to 0.25, yielding the expression $\bar{\omega}/4$ in the paper. The following figure shows results of our method for five different values of x: 0.25 (finest mesh, default value in the paper), 0.5, 1.0, 2.0, 4.0 (coarsest mesh). As expected, the quality of the parametrization generally decreases with decreasing mesh resolution. Note that our setting of $x = 0.25$ might be too conservative for some inputs – for instance, the SMALL CAR in the third example was vectorized equally well, using $x = 1.0$, but 5x faster (10 seconds versus 51 seconds).

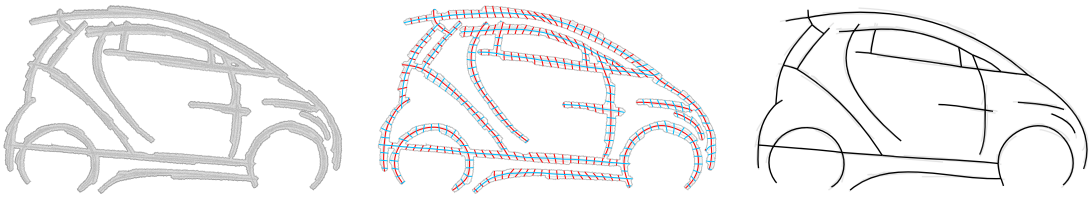



$x = 0.25$ (our preset)
vertices: 76293
triangles: 131443
integer vars: 195
total time: 71 sec.
$x = 0.5$
vertices: 40910
triangles: 67269
integer vars: 308
total time: 39 sec.
$x = 1.0$
vertices: 22494
triangles: 34795
integer vars: 1115
total time: 42 sec.
$x = 2.0$
vertices: 12539
triangles: 18042
integer vars: 2838
total time: 45 sec.
$x = 4.0$
vertices: 7221
triangles: 9501
integer vars: 2171
total time: 20 sec.
triangle mesh
parametrization
output curves

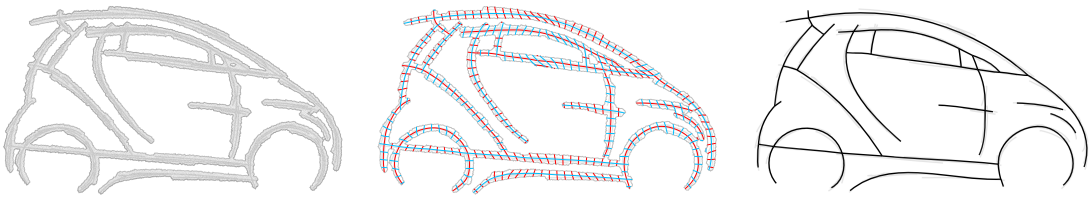
$x = 0.25$ (our preset)
vertices: 76446
triangles: 144197
integer vars: 71
total time: 51 sec.



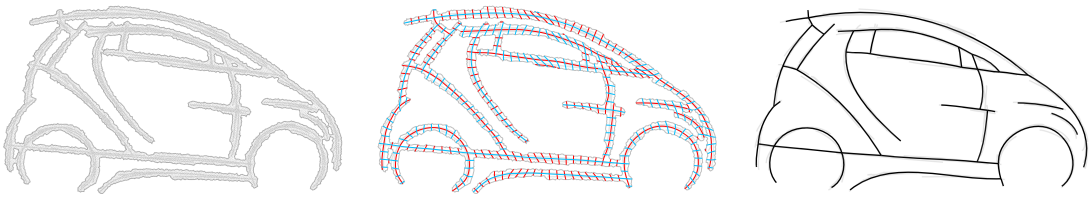
$x = 0.5$
vertices: 39357
triangles: 72674
integer vars: 66
total time: 19 sec.



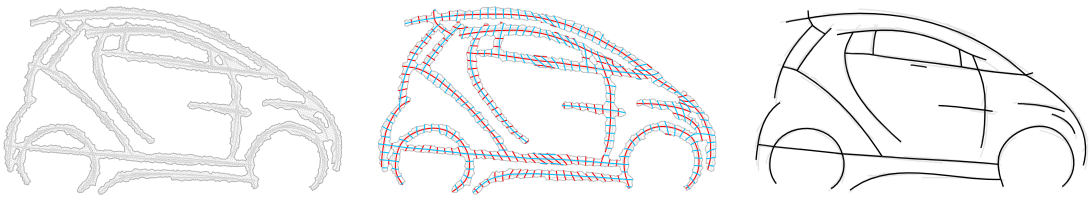
$x = 1.0$
vertices: 20449
triangles: 36653
integer vars: 83
total time: 10 sec.



$x = 2.0$
vertices: 10829
triangles: 18747
integer vars: 150
total time: 6 sec.



$x = 4.0$
vertices: 5772
triangles: 9560
integer vars: 200
total time: 4 sec.



triangle mesh | parametrization | output curves

6. Parameter study: threshold for orientation labeling

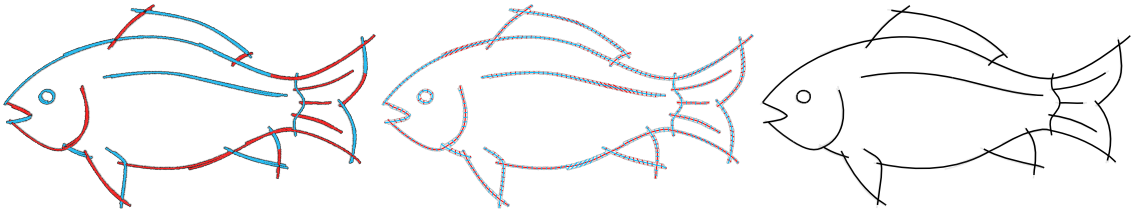
In order to determine which frame field direction is the tangent direction in a given triangle, we trace two streamlines starting from the triangle's barycenter. We then count the number of stroke pixels k_0, k_1 encountered along each of the streamlines, and assign the stroke triangle to the direction that covers more stroke pixels. We keep the triangle unlabeled if the ratio $k_0/(k_0+k_1) \in [0, 0.5]$ of number of stroke pixels $k_0 \leq k_1$ covered by the two streamlines is above a threshold z.

The following figures shows results of our method for five different values of the threshold z: 0.48 (default value in the paper), 0.36, 0.24, 0.12, 0.06. Left column shows the computed labels: red and blue represent the two frame field directions ($k_0/(k_0+k_1) \leq z$), while black means that the triangle is unlabeled ($k_0/(k_0+k_1) > z$). For each value of z, we also show the number of integer variables, and the total running time of our method.

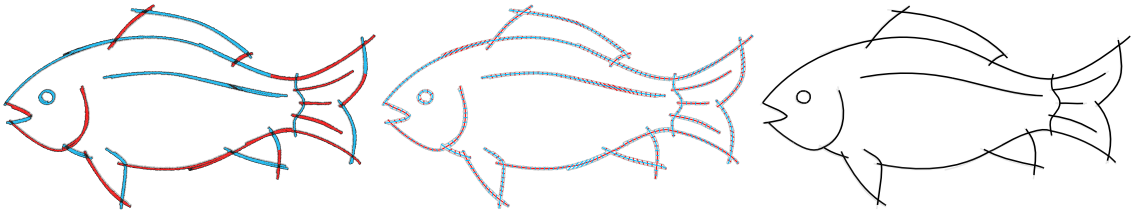
As the threshold decreases, more triangles are left unlabeled (black dots). This leads to a sparser set of snapping constraints, and, ultimately, parametrization isolines that do not follow the input curves. Note that we also use the labels to identify tangent isolines during extraction, which explains why some curves are missing in the output for lower values of z.

The decreasing number of integer variables for increasing values of the threshold z is due to our integer elimination heuristic. Despite having more snapping constraints (more triangles are labeled), bigger values of z lead to fewer and larger labeled regions – therefore, fewer snapping integers.

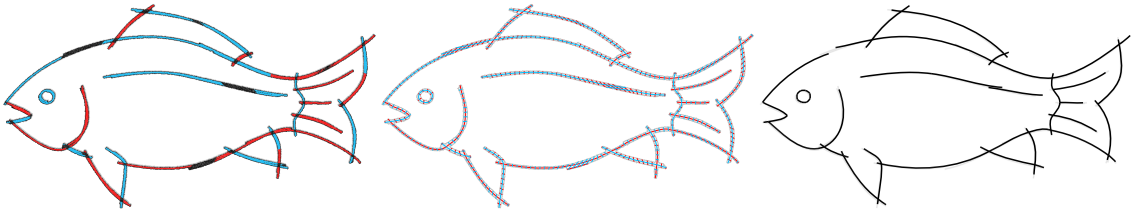
$x = 0.48$ (our preset)
integer vars: 64
total time: 21 sec.



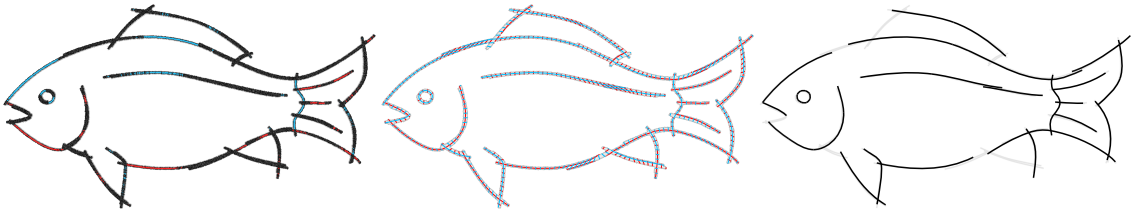
$x = 0.36$
integer vars: 80
total time: 19 sec.



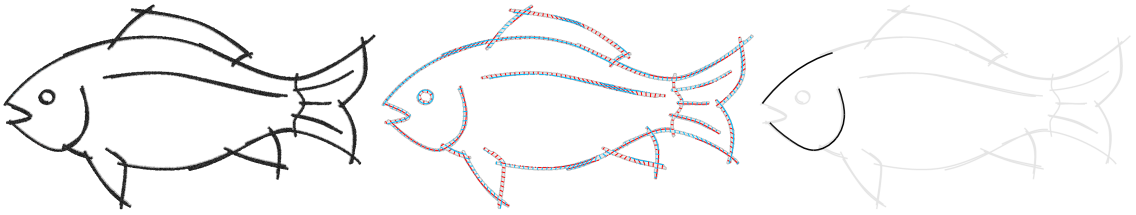
$x = 0.24$
integer vars: 85
total time: 19 sec.



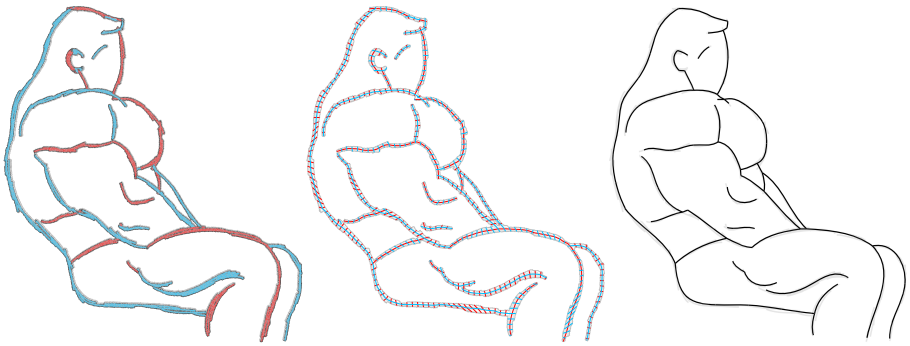
$x = 0.12$
integer vars: 127
total time: 17 sec.



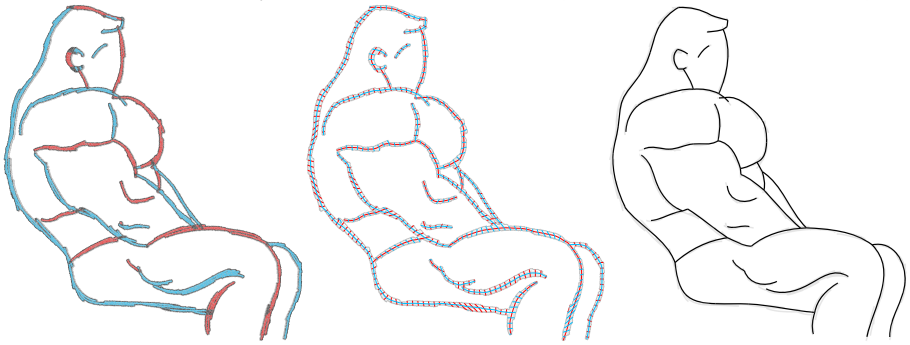
$x = 0.06$
integer vars: 14
total time: 11 sec.



$x = 0.48$ (our preset)
integer vars: 97
total time: 48 sec.

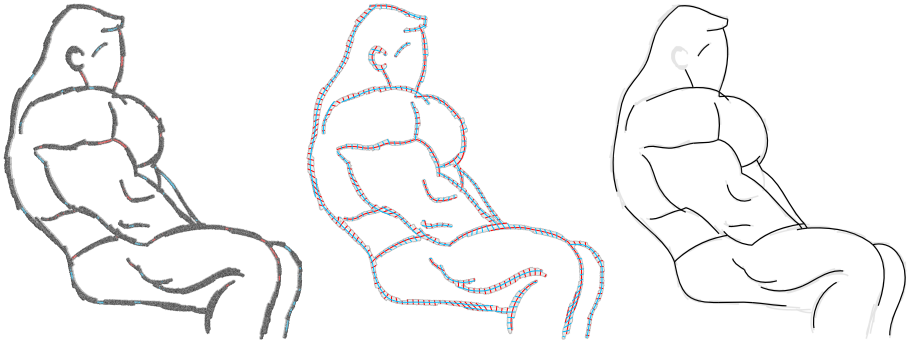


$x = 0.36$
integer vars: 108
total time: 46 sec.

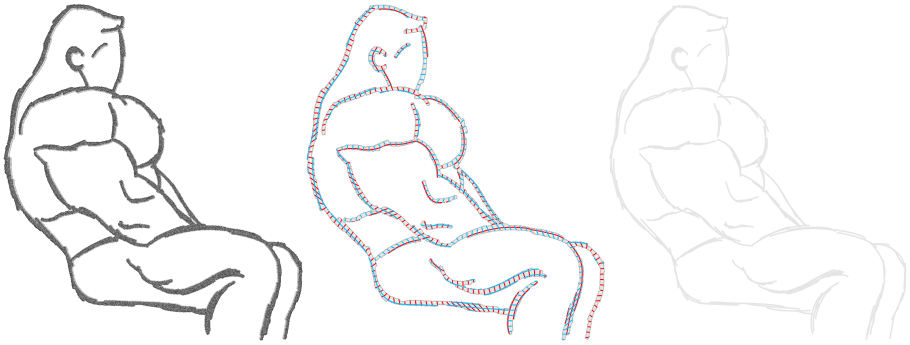


$x = 0.24$
integer vars: 124
total time: 31 sec.

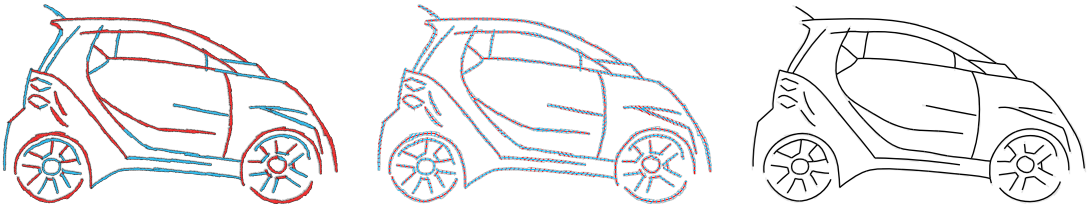
$x = 0.12$
integer vars: 138
total time: 26 sec.



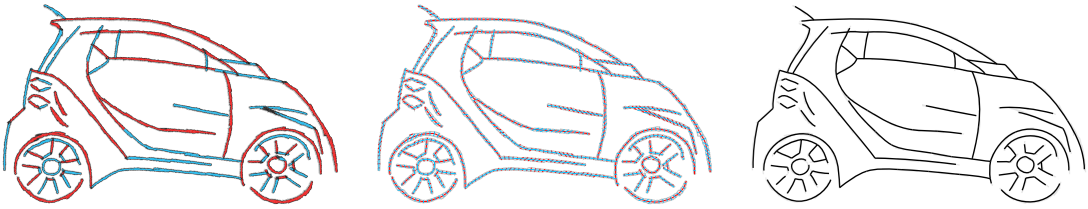
$x = 0.06$
integer vars: 34
total time: 19 sec.



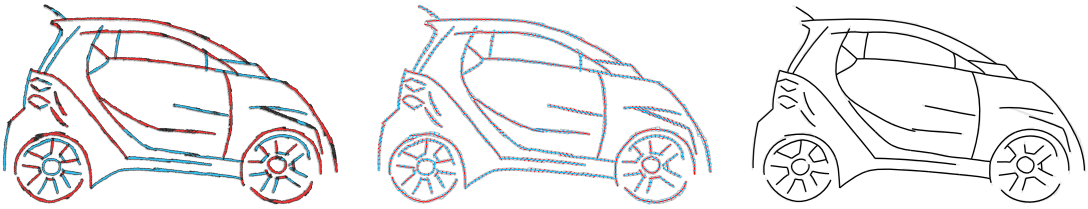
$x = 0.48$ (our preset)
integer vars: 208
total time: 41 sec.



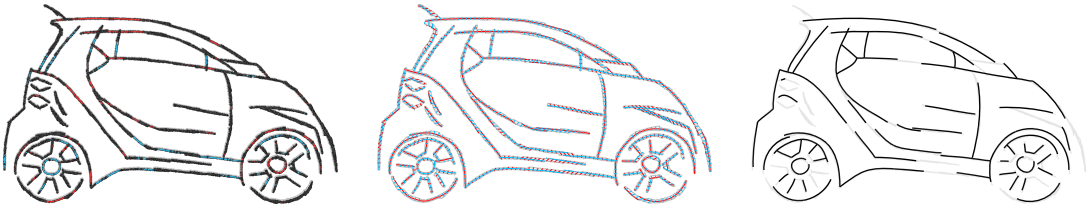
$x = 0.36$
integer vars: 193
total time: 40 sec.



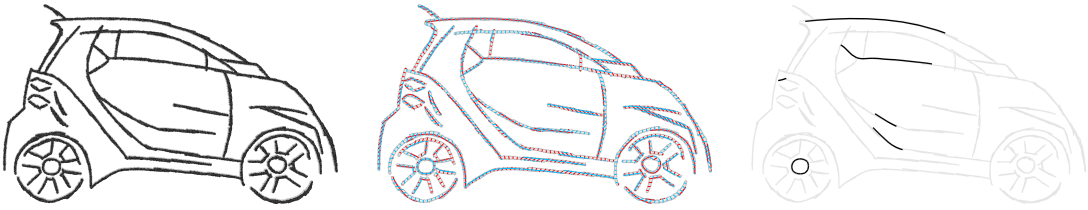
$x = 0.24$
integer vars: 211
total time: 41 sec.



$x = 0.12$
integer vars: 308
total time: 41 sec.



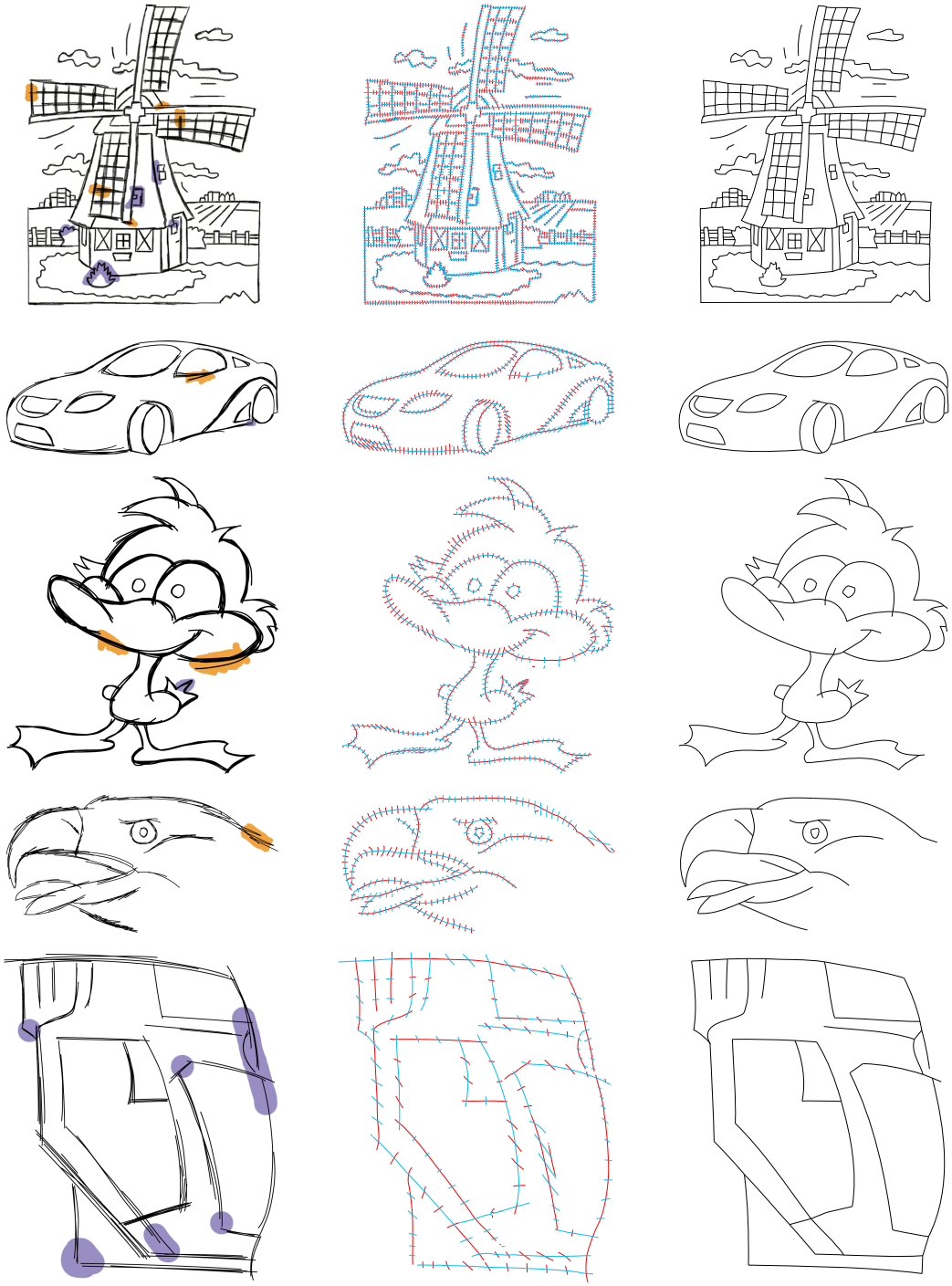
$x = 0.06$
integer vars: 49
total time: 21 sec.



7. Results with parametrizations (with masks)

Please zoom-in on the parametrization to better see the details at fine scales.

input sketch and mask | parametrization | output curves (with mask)

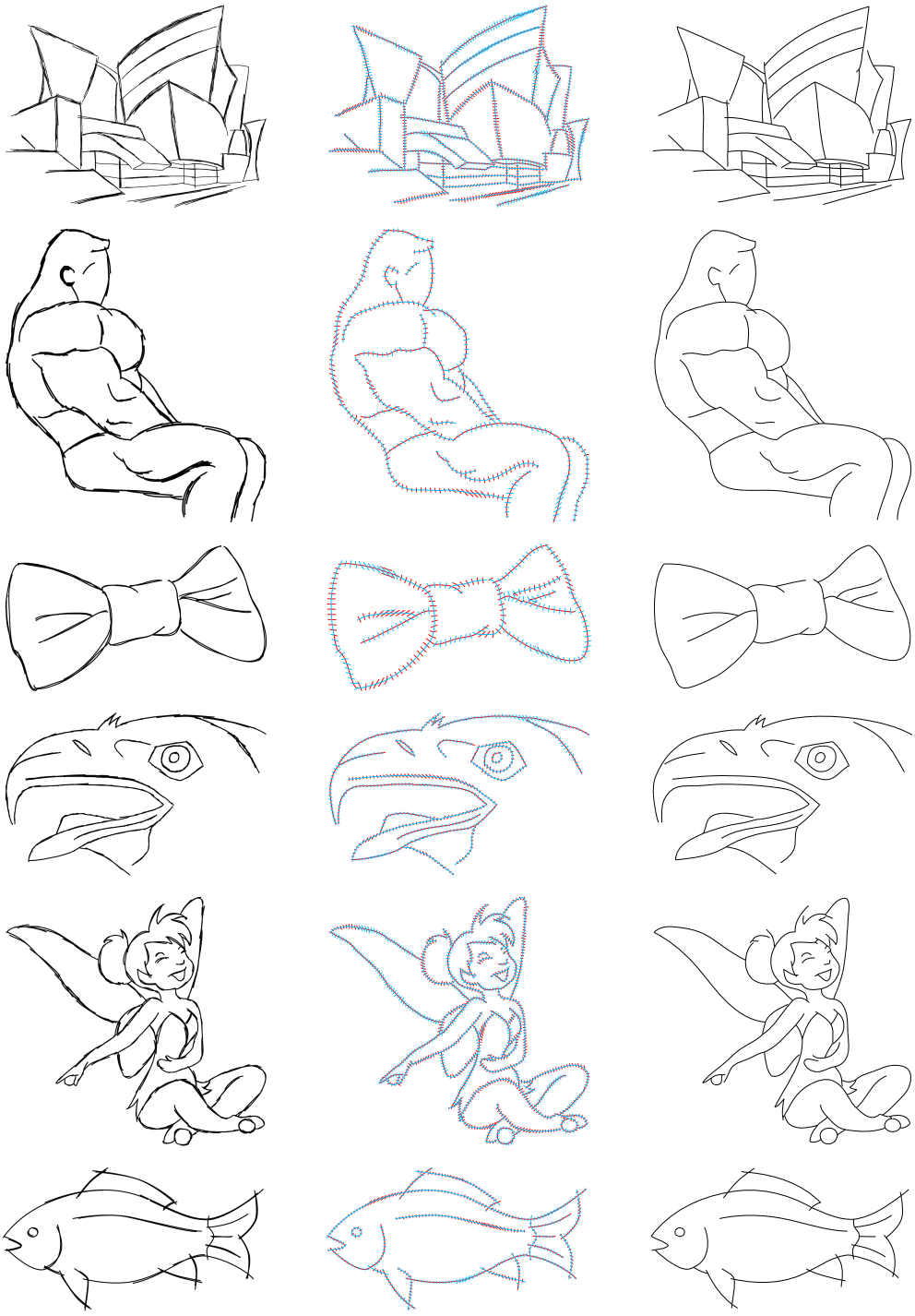


input sketch and mask | parametrization | output curves (with mask)

8. Results with parametrizations (without masks)

Please zoom-in on the parametrization to better see the details at fine scales.

input sketch | parametrization | output curves (no mask)

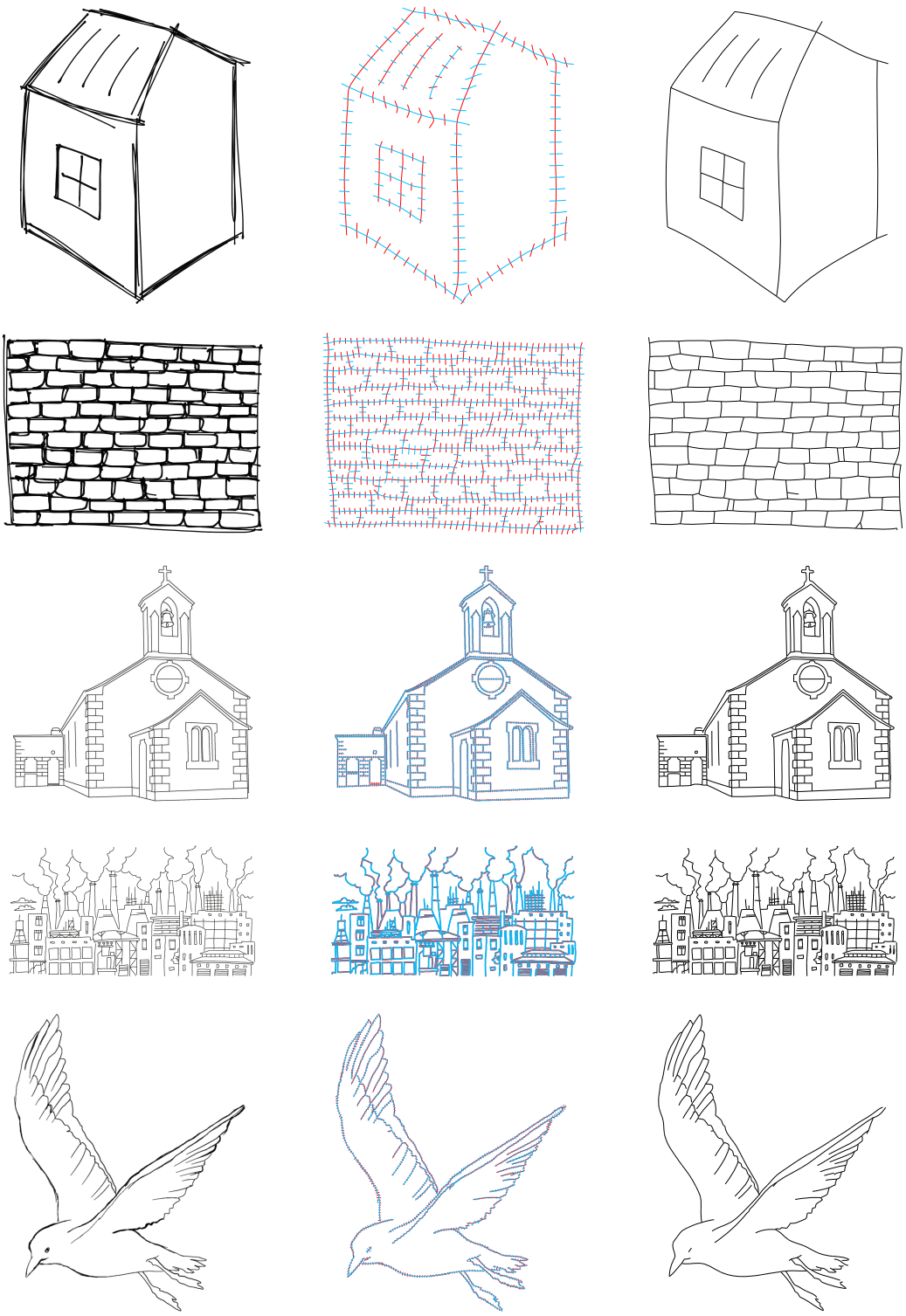


input sketch | parametrization | output curves (no mask)

input sketch | parametrization | output curves (no mask)

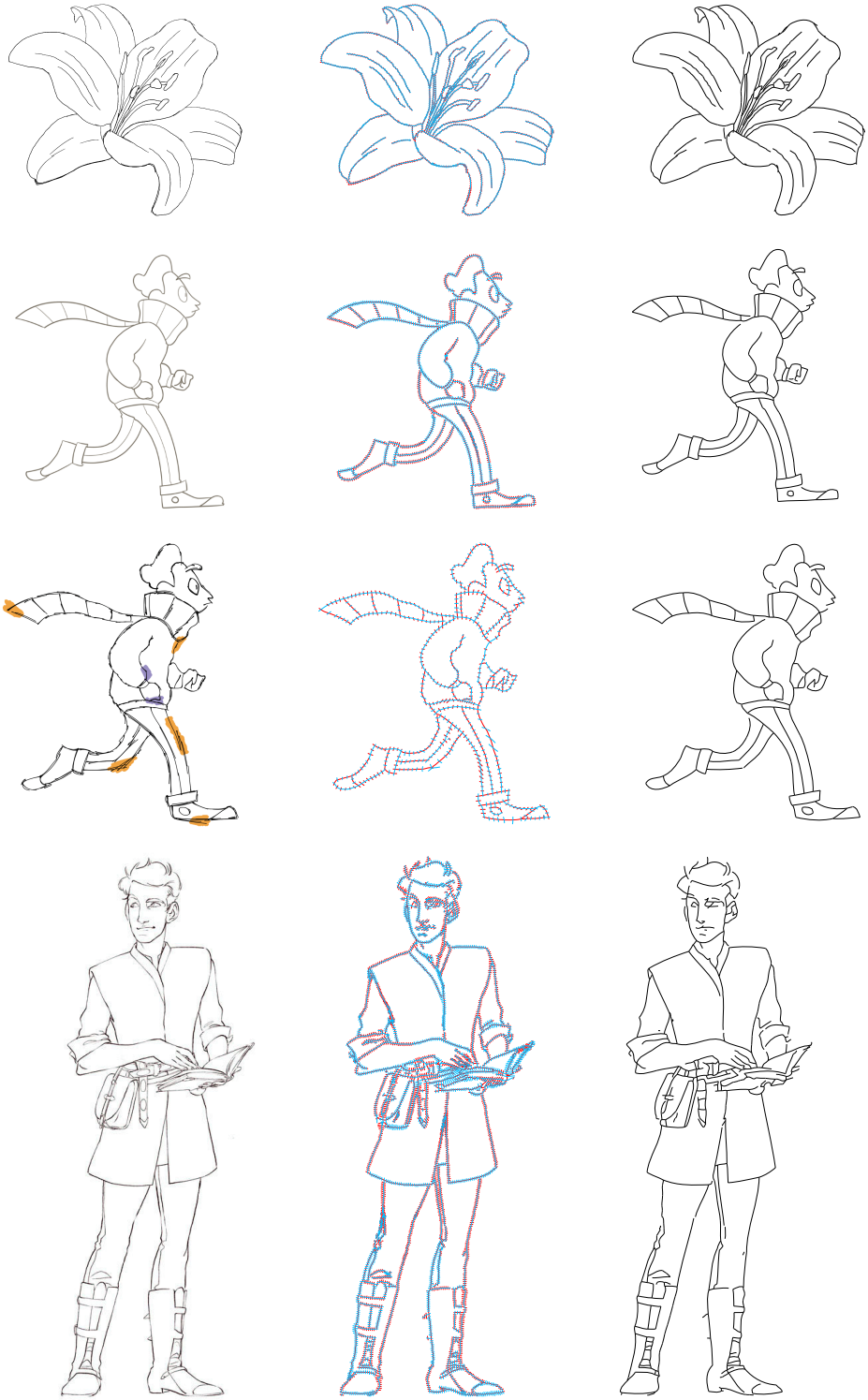
input sketch | parametrization | output curves (no mask)

input sketch | parametrization | output curves (no mask)

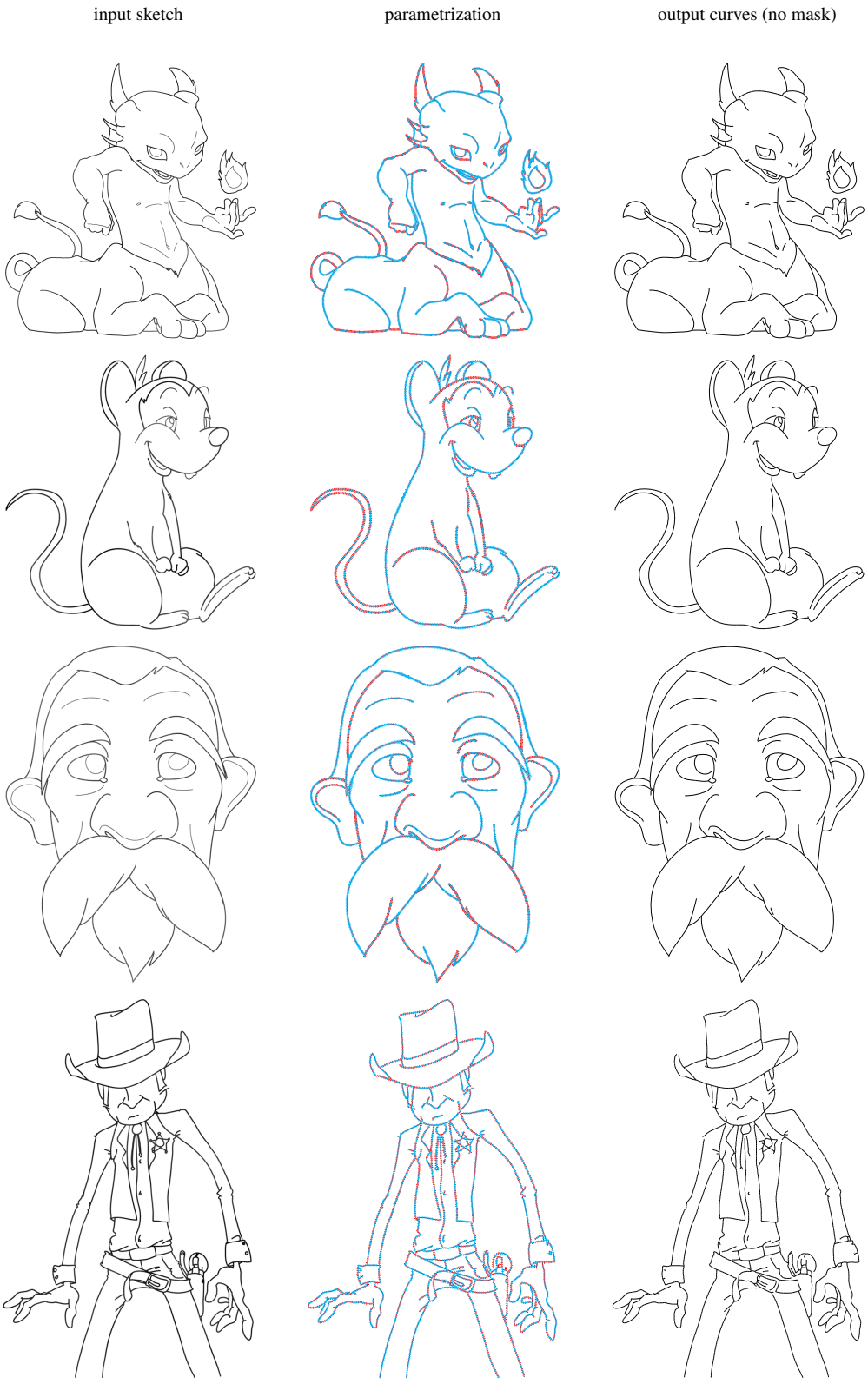


input sketch | parametrization | output curves (no mask)

input sketch parametrization output curves (no mask)



input sketch parametrization output curves (no mask)


input sketch
parametrization
output curves (no mask)
input sketch
parametrization
output curves (no mask)