WAIT: Workshop on the Advances in Information Technology, Budapest, 2022

Proximity by multiplicity

Péter Salvi and Tamás Várady

Budapest University of Technology and Economics

Abstract

Bézier curves are designed using control points, but—for larger degrees—moving an individual point may have very little effect on the shape of the curve. Proximity curves *introduce a new parameter that controls the pulling force of the control points. Here we propose a very simple method for the proximity control of polynomial curves, and discuss its advantages and shortcomings.*

1. Introduction

Control point based curves are prevalent in computer-aided geometric design. Control points offer a natural way to influence the shape of a curve, but sometimes—and this is especially true in the case of high-degree Bézier curves—the effect of a single control point is so little that local features are hard to introduce. Consequently, there is a need to control how close the curve runs to the control points, i.e., the *proximity* of the curve.

Here we deal exclusively with C^∞-continuous curves. Also, while proximity values can be specified independently for each control point, we will only treat the global case. Local variants can be constructed in a straightforward fashion.

Rational Bézier curves also associate a weight with each control point, so it is instructive to see why it does not solve the proximity problem. Figure 1 shows a rational curve where all internal control points have a weight of 50, while the endpoints have unit weight. The curve closely approximates the first and last control segments, but runs far from the three central control points. The corresponding basis functions are compared to the simple sextic Bézier basis in Figure 2. We can see that the three central basis functions are hardly affected. This is because what counts is the relative magnitude of adjacent weights. (Proximity criteria can be satisfied, in theory, by setting the weights to $w_i = t^{i(n-i)}$, where n is the degree,[1] but these quickly explode numerically, and distort the parameterization.)

We propose a very simple construction: replicate internal control points. A control polygon where each point is doubled (tripled etc.) naturally has a stronger pull on the curve. After a short review of related work in Section 2, we give a more formal treatment of our method in Section 3. This is followed by a discussion of *pros* and *cons* in Section 4, also including some alternative research directions.

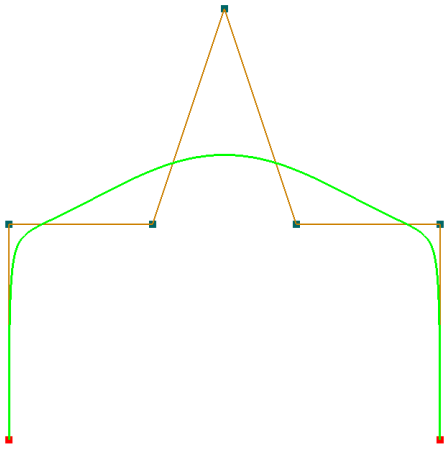
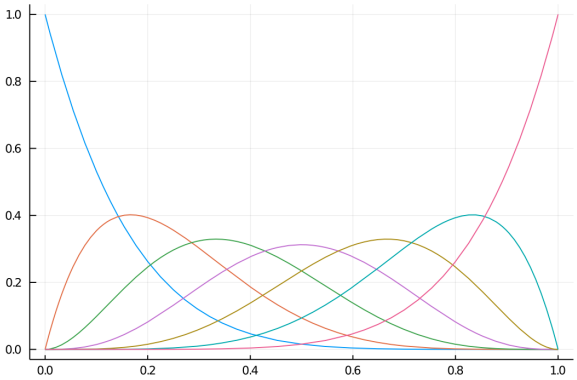


Figure 1: A rational Bézier curve with large internal weights.

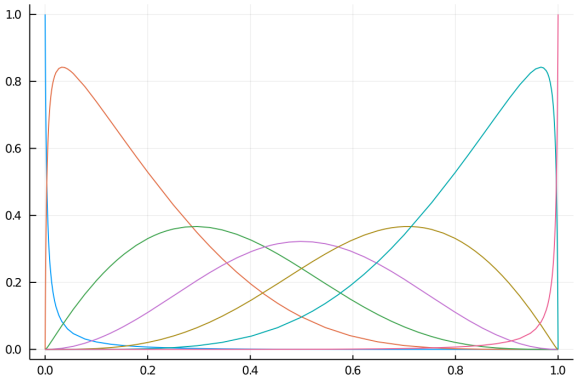
2. Previous work

A recent paper[4] introduces a class of proximity basis functions that can reproduce Bézier and B-spline curves or surfaces. We will show some comparisons to P-Bézier curves in Section 4.1.

The above paper and its predecessor[3] (which coined the term *proximity curve*) has an extensive literature review.



(a) $w = [1,1,1,1,1,1,1]$



(b) $w = [1,50,50,50,50,50,1]$

Figure 2: Comparison of degree-6 rational basis functions.

Here we just note that recent advances include a NURBS-based approach[2], cyclic proximity curves[7] and a proximity variant[6] of κ-curves.[8]

3. μ-Bézier curves

Given a Bézier curve

$$C(u) = \sum_{i=0}^{n} P_i B_i^n(u) \tag{1}$$

of degree n, and a global multiplicity parameter μ, the μ-Bézier proximity curve is a Bézier curve of degree $\hat{n} = \mu \cdot (n-1)+1$, with control points

$$\hat{P}_i = \begin{cases} P_0 & \text{for } i = 0, \\ P_{\lfloor (i+\mu-1)/\mu \rfloor} & \text{for } 0 < i < \hat{n}, \\ P_n & \text{for } i = \hat{n}. \end{cases} \tag{2}$$

Note that this is only a hidden representation; the designer uses the P_i control points to modify the curve.

Conversely, the basis functions associated with *the original* control points are the sums of $\hat{n}$-degree Bernstein polynomials:

$$\hat{B}_i^n(u) = \begin{cases} B_0^{\hat{n}}(u) & \text{for } i = 0, \\ \sum_{j=1+\mu\cdot(i-1)}^{\mu\cdot i} B_j^{\hat{n}}(u) & \text{for } 0 < i < n, \\ B_{\hat{n}}^{\hat{n}}(u) & \text{for } i = n. \end{cases} \tag{3}$$

Consequently the curve can be expressed as

$$C(u) = \sum_{i=0}^{\hat{n}} \hat{P}_i B_i^{\hat{n}}(u) = \sum_{i=0}^{n} P_i \hat{B}_i^n(u). \tag{4}$$

Figure 3 shows quintic basis functions with different μ values.

4. Discussion

In this section we compare μ-Bézier and P-Bézier curves, and explore paths to remedy its shortcomings.

4.1. Comparison with P-Bézier curves

Figure 4 compares the basis functions of μ-Bézier and P-Bézier curves. Here the γ proximity parameter of the P-Bézier construction is chosen such that the resulting basis resembles the μ-Bézier basis in the same plot. It is apparent that while the internal functions are quite similar, those at the end are much larger for μ-Bézier curves. This means that the first and last control segments are approximated much faster than the others.

Another difference is that while the γ parameter is continuous, i.e., it defines a continuum of curves between the Bézier curve and the control polygon, the μ parameter is discrete. While it does approximate the control polygon as μ approaches infinity, the first steps leave out much of the possible design space, see a comparison in Figure 5.

On the positive side, while P-Bézier curves use a rational basis, involving square roots, μ-Bézier curves are polynomial, and as such, CAD-compatible.

4.2. Adding more refinement

A finer range of proximity curves can be generated if we insert two new points near each inner control point, instead of replicating the original ones. An extra $\alpha \in [0.5, 1]$ parameter controls the position of the new points on the segments:

$$\begin{aligned} P_{i,1}^{\text{new}} &= (1-\alpha)P_{i-1} + \alpha P_i, \\ P_{i,2}^{\text{new}} &= (1-\alpha)P_{i+1} + \alpha P_i. \end{aligned} \tag{5}$$

When $\alpha = 1$, this is the same as a μ-Bézier curve with $\mu = 3$. This can also be iterated; Figure 6 shows an example after 3 iterations with different α values.

4.3. Proximity by approximation

The degree of the generated curves is relatively high. An alternative approach that could keep the degree down is to use

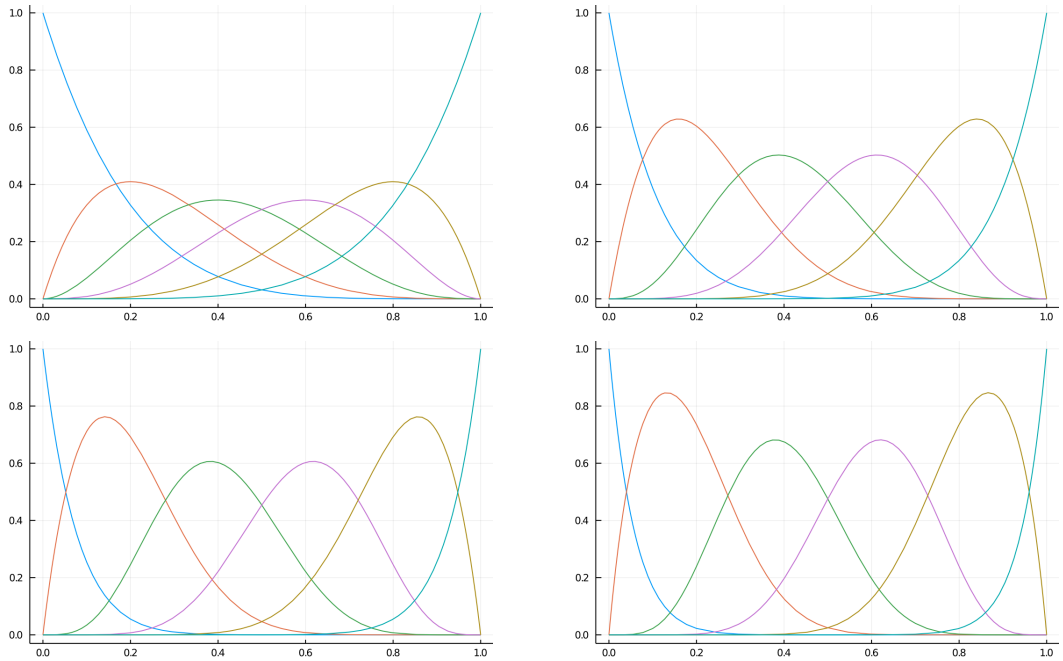


Figure 3: Quintic basis functions with $\mu = 1$, 2 (top row), and $\mu = 3$, 4 (bottom row).

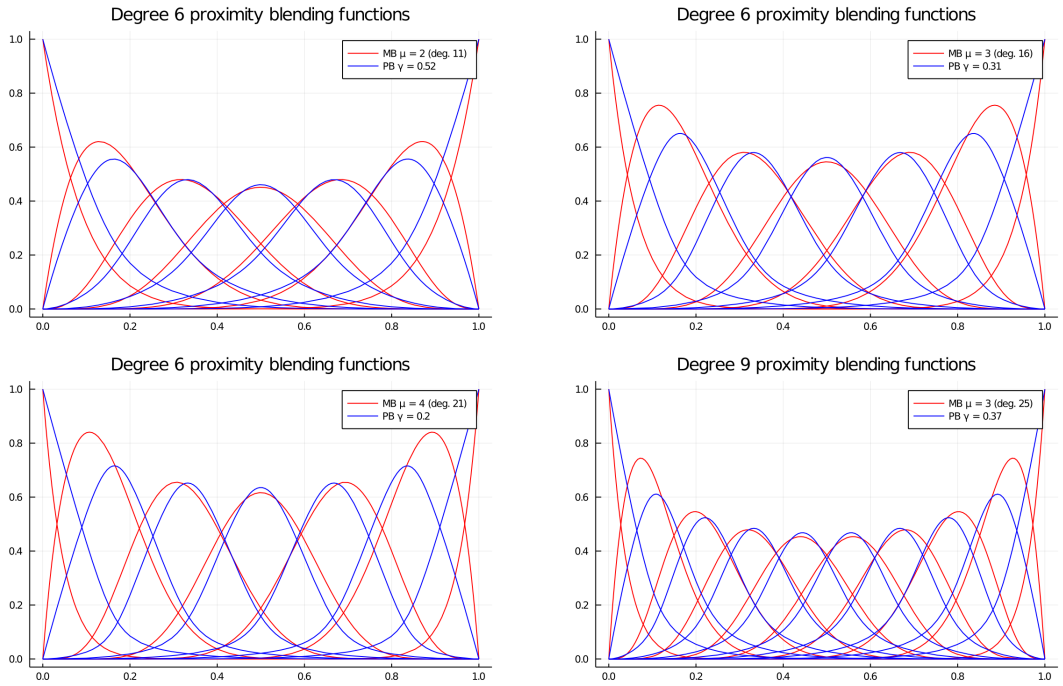


Figure 4: Comparison of μ-Bézier basis functions (MB) to the P-Bézier basis (PB), using similar γ values.

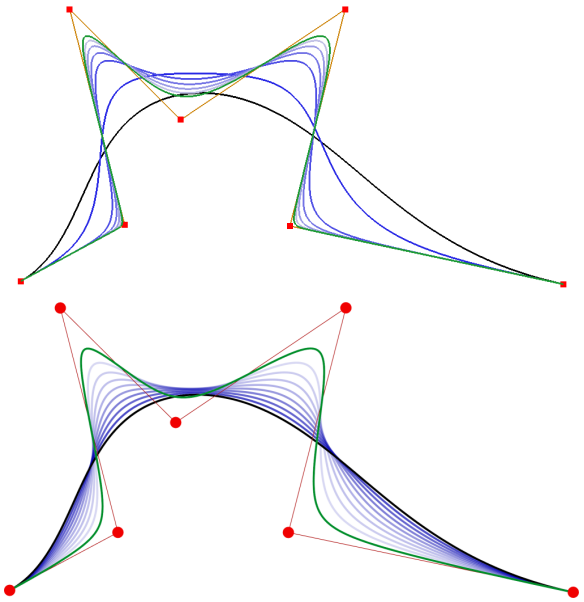


Figure 5: Comparison of μ-Bézier curves (top, $\mu = 1,\ldots,7$) to P-Bézier curves[4] (bottom, image used with permission).

fitting. Take a *target proximity curve* $f : [0,1] \to \mathbb{R}^2$, e.g. a P-Bézier curve, or just the control polygon itself, and try to find an approximation by a Bézier curve of moderately higher degree.

A naïve approximation of sampled points would result in large fluctuations in the control polygon. Adding smoothing terms is a viable approach, but its effects are hard to predict.

We will use a displacement approach. The idea is to elevate the degree of the original curve, step by step, until the desired degree N is reached, and in each step we add displacements to its control points based on the deviation of its footpoints from the corresponding points of f. Let us define

$$C^{(0)}(u) = \sum_{i=0}^{n} P_i^{(0)} B_i^n(u) = C(u). \tag{6}$$

In the first step we perform a degree elevation, obtaining control points $\hat{P}_i^{(0)}$ $(i = 0\ldots n+1)$. Then in each step, we define

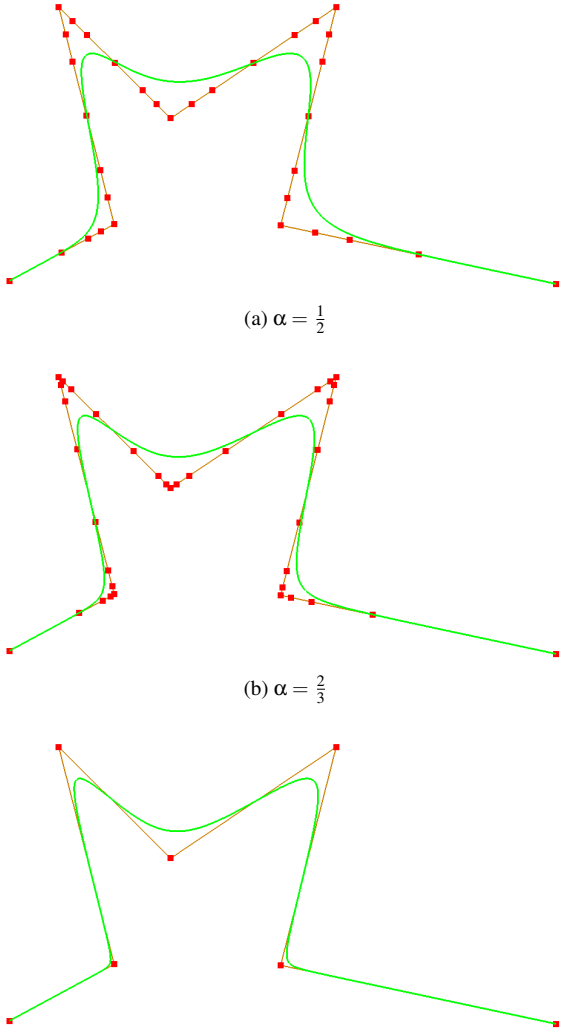
$$C^{(k)}(u) = \sum_{i=0}^{n+k} P_i^{(k)} B_i^{n+k}(u), \tag{7}$$

where

$$P_i^{(k)} = \hat{P}_i^{(k-1)} + \left[f\left(\frac{i}{n+k}\right) - C^{(k-1)}\left(\frac{i}{n+k}\right)\right]. \tag{8}$$

After $N-n$ steps, we arrive at

$$\hat{C}(u) = C^{(N-n)}(u). \tag{9}$$



(a) $\alpha = \frac{1}{2}$

(b) $\alpha = \frac{2}{3}$

(c) $\alpha = 1$ (μ-Bézier curve)

Figure 6: Fine(r) proximity control (3 iterations, degree 36).

4.3.1. More iterations

The procedure outlined above was a sequence of alternating degree elevations and displacements. We could, however, perform several displacement steps before moving on to the next degree. Formally, if the iteration count was J, we define

$$C^{(k)}(u) = C^{(k,J)}(u), \tag{10}$$

where

$$C^{(k,j)}(u) = \sum_{i=0}^{n+k} P_i^{(k,j)} B_i^{n+k}(u), \tag{11}$$

$$P_i^{(k,j)} = \begin{cases} \hat{P}_i^{(k-1)} & \text{for } j = 0, \\ P_i^{(k,j-1)} + f(u_i) - C^{(k-1,j-1)}(u_i) & \text{for } j > 0, \end{cases} \tag{12}$$

and $u_i = i/(n+k)$.

Note that choosing a large J would mean that in each step we do a progressive-iterative approximation (PIA),[5] which is the same as if we interpolated f at the footpoint parameters, resulting in a highly oscillating control polygon, see Figure 7b. On the other hand choosing J value of around 5 can accelerate the approximation, and would allow the same error range with lower degree, see Figure 7c.

A drawback of this method is that it does not preserve the convex hull. We can constrain the displacements so that control points always stay inside, but this blunts the proximity effect, see Figure 8. Here we see a fit first *without*, then *with* the convex hull constraint. For similar results a higher-degree curve is needed, but then the μ-Bézier curve fares just as well, see Figure 8d.

Conclusion

We have proposed a very simple method for creating polynomial proximity curves; limitations include high degree and coarse gradation. Some avenues for the lifting of these problems were also investigated.

Acknowledgements

This project has been supported by the Hungarian Scientific Research Fund (OTKA, No.124727).

References

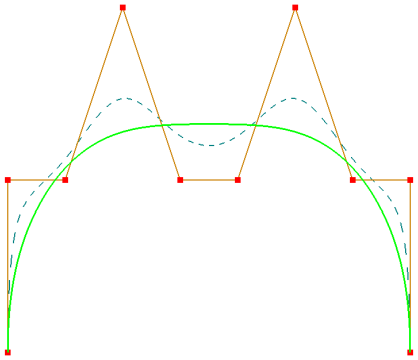
1. Gheorghe Craciun, Luis David García-Puente, and Frank Sottile. Some geometrical aspects of control points for toric patches. In *Mathematical Methods for Curves and Surfaces*, volume 5862 of *Lecture Notes in Computer Science*, pages 111–135. Springer, 2008.

2. Imre Juhász. A NURBS transition between a Bézier curve and its control polygon. *Journal of Computational and Applied Mathematics*, 113626, 2021.

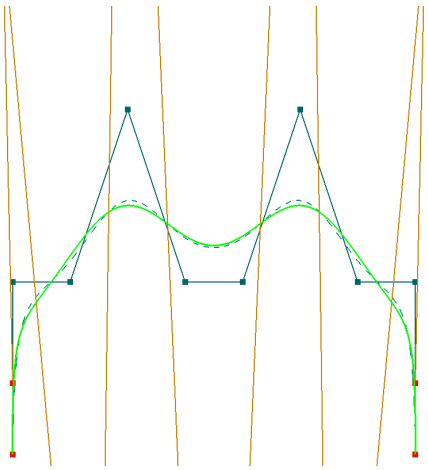
3. István Kovács and Tamás Várady. P-curves and surfaces: Parametric design with global fullness control. *Computer-Aided Design*, 90:113–122, 2017.

4. István Kovács and Tamás Várady. P-Bézier and P-Bspline curves – new representations with proximity control. *Computer Aided Geometric Design*, 62:117–132, 2018.

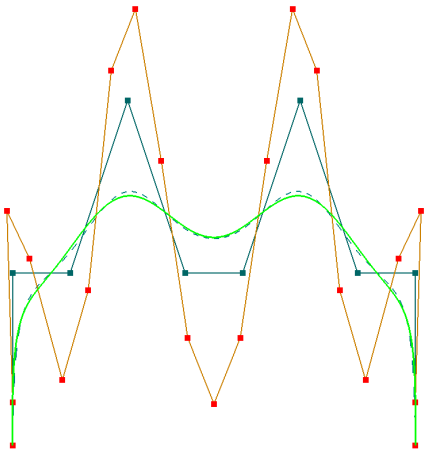
5. Hongwei Lin, Takashi Maekawa, and Chongyang Deng. Survey on geometric iterative methods and their applications. *Computer-Aided Design*, 95:40–51, 2018.



(a) Original degree-9 curve with target

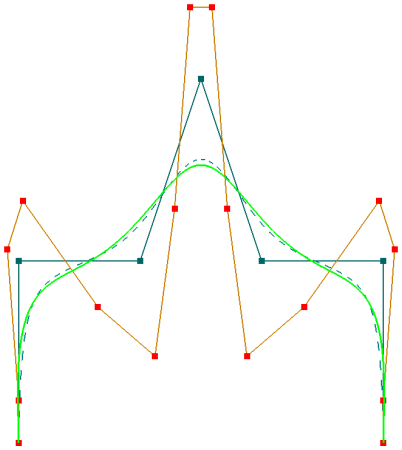


(b) Degree-13 approx. with 50 PIA iterations

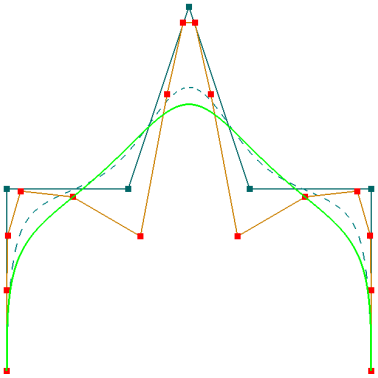


(c) Degree-20 approx. with 5 PIA iterations

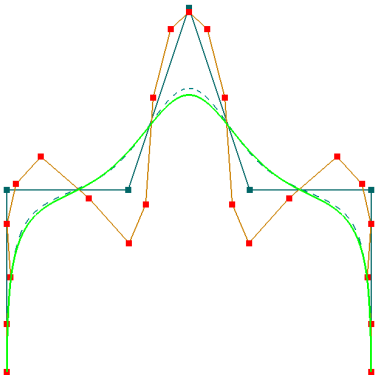
Figure 7: Approximating a $\gamma = 0.4$ P-Bézier curve.



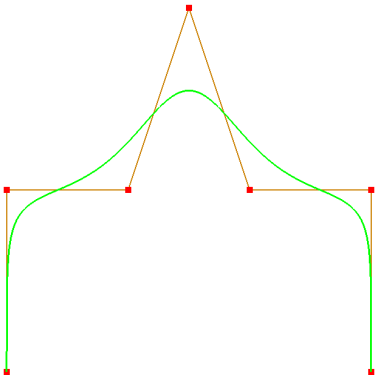
(a) Degree-15 approx. with 5 PIA iterations



(b) As in Figure 8a, but constrained to convex hull



(c) As in Figure 8b, but a degree-22 approximation



(d) A similar μ-Bézier curve with $\mu = 3$ (degree-16)

Figure 8: Approximating a $\gamma = 0.3$ P-Bézier curve.

6. Kenjiro T. Miura, Rudrusamy U. Gobithaasan, Péter Salvi, Dan Wang, Tadatoshi Sekine, Shin Usuki, Junichi Inoguchi, and Kenji Kajiwara. εκ-curves: Controlled local curvature extrema. *The Visual Computer*, (accepted), 2021.
7. Ágoston Róth. Simple and weighted cyclic proximity curves and surfaces. *Computer-Aided Design*, 103043, 2021.
8. Zhipei Yan, Stephen Schiller, Gregg Wilensky, Nathan Carr, and Scott Schaefer. κ-curves: Interpolation at local maximum curvature. *ACM Transactions on Graphics*, 36(4):Article 129, 2017.