Differentiable Curl-Noise: Boundary-Respecting Procedural Incompressible Flows Without Discontinuities

XINWEN DING, University of Waterloo, Canada
CHRISTOPHER BATTY, University of Waterloo, Canada

We present Differentiable Curl-Noise, a C^1 procedural method to animate strictly incompressible fluid flows in two dimensions. While both the original Curl-Noise method of Bridson et al. [2007] and a recent modification by Chang et al. [2022] have been used to design incompressible flow fields, they often suffer from non-smoothness in their handling of obstacles, owing in part to properties of the underlying Euclidean distance function or closest point function. We therefore propose a differentiable scheme that modulates the background potential in a manner that respects arbitrary solid simple polygonal objects placed at any location, without introducing discontinuities. We demonstrate that our new method yields improved flow fields in a set of two dimensional examples, including when obstacles are in close proximity or possess concavities.

CCS Concepts: • **Computing methodologies → Procedural animation**; **Physical simulation**.

Additional Key Words and Phrases: procedural animation, incompressible flow, fluids, differentiability, vector field design

ACM Reference Format:
Xinwen Ding and Christopher Batty. 2023. Differentiable Curl-Noise: Boundary-Respecting Procedural Incompressible Flows Without Discontinuities. *Proc. ACM Comput. Graph. Interact. Tech.* 6, 1 (May 2023), 16 pages. https://doi.org/10.1145/3585511

1 INTRODUCTION

The use of physics-based simulation of fluid dynamics has been an active research topic in computer animation for many years now, with a wide range of industrial applications in visual effects, computer games, and animated movies. However, despite progress towards developing effective simulation control techniques, directability remains a challenge; moreover, the resulting discretized flows are not necessarily strictly (pointwise) incompressible and may not precisely respect obstacles. A longstanding alternative to simulation in computer animation is the use of procedural techniques, which sidestep the need for solving partial differential equations and instead place the onus on the user to express their desired result more directly. A popular example for fluids is the *Curl-Noise* approach of Bridson et al. [2007] which offers a fast and simple technique for generating a range of pointwise incompressible velocity fields that exhibit laminar flow or noisy turbulence, while also respecting obstacles. Thus, rather than setting up boundary conditions and hoping for a desirable fluid simulation result, users can sculpt their flows explicitly and analytically, leading to an entirely different workflow. Moreover, Curl-Noise ideas have recently been *combined* with simulation to achieve the best of both worlds; improvements to Curl-Noise may therefore also benefit traditional fluid simulation workflows.

Authors' addresses: Xinwen Ding, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada; Christopher Batty, University of Waterloo, 200 University Avenue West, Waterloo, Ontario, Canada.

2577-6193/2023/5-ART $15.00
https://doi.org/10.1145/3585511

Curl-Noise is based on the observation that a vector field constructed as the curl of some other vector field is divergence-free by construction. Therefore, if we define our velocity field using $\mathbf{v} = \nabla \times \psi$, then $\nabla \cdot \mathbf{v} = 0$, since basic vector calculus ensures that the divergence of curl is always zero: $\nabla \cdot (\nabla \times \psi) = 0$. The vector field ψ is called a *vector potential.* For the present paper, our focus is on the two-dimensional case in which the vector potential ψ reduces to a scalar *stream function*, by setting the x and y components of the vector ψ to zero; the curl operation then simplifies such that the velocity field $\mathbf{v}$ is just a 90 degree rotation of $\nabla\psi$, that is, $\mathbf{u} = (\nabla\psi)^{\perp}$. (For simplicity we will assume ψ is a scalar henceforth.) Curl-Noise begins with a user-defined ψ that dictates the ambient flow field, with streamlines (i.e., particle trajectories) lying on isocontours of ψ. One subsequently modulates ψ in regions near obstacles to enforce that the fluid should not flow into or out of obstacles. For static obstacles, this amounts to requiring that ψ be constant along each disjoint boundary since the tangential derivative of ψ dictates the normal component of velocity; to put it another way, the boundary must be an isocontour. Curl-Noise therefore introduces a multiplicative ramping procedure that forces ψ to a constant as the point being evaluated approaches the boundary. More recently, Chang et al. [2022] observed that the multiplicative ramping strategy of the original Curl-Noise method can lead to undesired no-slip behavior and proposed an additive alternative that better preserves free-slip effects. However, in both methods the construction of the fluid velocity field $\mathbf{v}$ fundamentally relies on taking partial derivatives of a modulated ψ field, yet neither boundary treatment ensures that ψ is everywhere continuous and differentiable. The unfortunate result is that the computed fluid velocity field can often exhibit kinks or outright discontinuities, thereby destroying the illusion of a natural fluid flow.

In this paper, we propose a *differentiable* approach to generate ψ fields for procedural fluid animation in two dimensions. We do so by carefully investigating the sources of discontinuities in both the original Curl-Noise and the variation of Chang et al. and introducing solutions that avoid these pitfalls. Differentiability then ensures that we can generate mathematically and physically plausible procedural animations of fluids flowing past arbitrary static simple polygonal obstacles, with velocity fields $\mathbf{v}$ that are at least continuous.

2 RELATED WORK

Two related methods have dominated the procedural generation of analytically incompressible flow fields in computer animation. The first approach constructs the velocity as the cross product of the gradients of two scalar fields, as proposed by DeWolf [2006]. The same basic approach was also used for incompressible shape modeling by von Funck et al. [2006]. The second approach constructs the velocity as the curl of a vector potential leading to the Curl-Noise approach of Bridson et al. [2007]. The two approaches lead to distinct constructions of noise fields, laminar flows, obstacle enforcement, and other features. Curl-Noise was subsequently used as a sub-component of techniques that model the evolution of fine-scale turbulence within simulated flows [Kim et al. 2008; Narain et al. 2008; Schechter and Bridson 2008]. The same concept has also been used for simulation editing [Pan et al. 2013] and to improve boundary-handling in simulated flow fields [Chang et al. 2022].

3 LIMITATIONS OF PREVIOUS WORK

Assuming that the initial background potential ψ given by the user is smooth, then all sources of non-differentiability lie in how the field is modulated near obstacles. To clearly demonstrate the drawbacks of existing methods, let us first outline consistent notations and definitions. Let $\mathcal{D}$ be a bounded domain of interest and let $\boldsymbol{x} \in \mathcal{D}$ be a query point in the domain. Assume there are n static obstacles defined in $\mathcal{D}$, denoted as $O_1, O_2, \cdots, O_n$. Furthermore, we define $d(\boldsymbol{x}) : \mathcal{D} \rightarrow \mathbb{R}$ to be the minimum signed Euclidean distance from the query point $\boldsymbol{x}$ to any obstacle in $\mathcal{D}$ and let

$d_i(\boldsymbol{x}) : \mathcal{D} \longrightarrow \mathbb{R}$ be the minimal signed Euclidean distance from the query point $\boldsymbol{x}$ to the i^{th} obstacle, O_i (where we have adopted the convention that negative distance values lie on the interiors of obstacles). Therefore, we can write $d(\boldsymbol{x})$ as:

$$d(\boldsymbol{x}) = \min\{d_1(\boldsymbol{x}), d_2(\boldsymbol{x}), \cdots, d_n(\boldsymbol{x})\}, \tag{1}$$

and we can write $d_i(\boldsymbol{x})$ as:

$$d_i(\boldsymbol{x}) = \min\{d_1^*(\boldsymbol{x}), d_2^*(\boldsymbol{x}), \cdots, d_n^*(\boldsymbol{x})\}, \tag{2}$$

where $d_n^*(\boldsymbol{x})$ is the Euclidean distance from $\boldsymbol{x}$ to the n^{th} edge in O_i.

Now, we are ready to write down the existing modulation methods and state their defects. Bridson et al. [2007] modulated ψ by multiplying against a smooth ramp function $ramp(\cdot)$ to zero:

$$\psi' = \alpha(\boldsymbol{x})\psi, \quad \alpha(\boldsymbol{x}) = ramp\left(\frac{d(\boldsymbol{x})}{d_0}\right), \quad ramp(r) = \begin{cases} 1 & \text{if } r \geq 1 \\ \frac{15}{8}r - \frac{10}{8}r^3 + \frac{3}{8}r^5 & \text{if } -1 < r < 1 \\ -1 & \text{if } r \leq -1 \end{cases}, \tag{3}$$

where the constant $d_0 > 0$ is the maximum influence radius of modulation. However, this construction has two limitations. First, the $\min\{\cdot\}$ function, appearing in $d(\boldsymbol{x})$, $i \in [n]$ and $d_i(\boldsymbol{x})$, is C^0. Second, as pointed out by Chang et al. [2022], the modulation in (3) can lead to spurious flow near the boundary of the obstacle and thus visual artifacts. Figure 1(a)(b) show two examples exhibiting the two aforementioned problems. Chang et al. [2022] suggested a slightly different modulation function to partially resolve the issue by setting each boundary's potential to be a given constant ψ_g,

$$\psi'(\boldsymbol{x}) = \alpha(\boldsymbol{x})\psi(\boldsymbol{x}) + (1 - \alpha(\boldsymbol{x}))\psi_g(\boldsymbol{x}), \tag{4}$$

where the target boundary value $\psi_g \neq 0$ in general. In practice, a natural choice for the value of ψ_g is the unmodified background potential ψ evaluated at the geometric center of the obstacle closest to $\boldsymbol{x}$. Figure 1(c) shows that Eq.4 removed the spurious tangential flow near objects, but introduced another source of discontinuity.

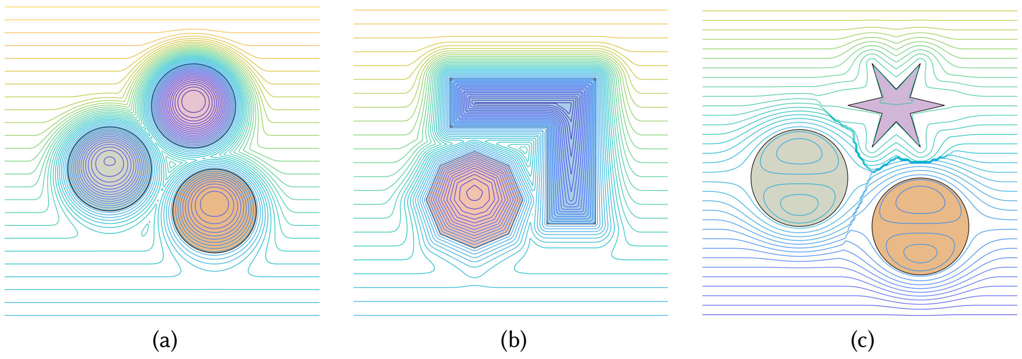


Fig. 1. (a) and (b) show the isocontour plot of ψ' under the modulation function by Bridson et al. [2007]: The C^0 property of the min{·} function in $d(\boldsymbol{x})$ is apparent at the equidistant curves between objects. Moreover, all obstacles in the plots are surrounded by spurious tangential flows, regardless of the smoothness of their boundaries, since all boundaries are ramped to the same constant zero value. (c). Ramping to distinct values per obstacle improves the situation, but unfortunately it also escalates the previously C^0 points on the equidistant curves to become fully discontinuous. This discontinuity exists between objects with both smooth and C^0 boundaries.

In the same paper , Chang et al. [2022] further pointed out that either multiplicative strategy (Eq. 3 or Eq. 4) undesirably impacts the normal derivatives of ψ, and correspondingly damages the tangential (free-slip) velocities. They proposed instead an *additive* modulation approach,

$$\psi'(\boldsymbol{x}) = \psi(\boldsymbol{x}) + (\psi_g(\boldsymbol{x}) - \psi(cp(\boldsymbol{x})))(1 - \alpha(\boldsymbol{x})), \tag{5}$$

where $cp(\boldsymbol{x})$ is the closest boundary point to $\boldsymbol{x}$.
Unfortunately, the C^0 property of $d(\boldsymbol{x})$ and $d_i(\boldsymbol{x}), i \in [n]$ is inherited by Eq. 5. Furthermore, the constant ψ_g, together with a constant maximum influence radius, d_0, can exacerbate the first problem in Eq. 3 by making ψ' discontinuous on the equidistant curve as follows. Let O_i and O_j be the two closest obstacles to the query point $\boldsymbol{x} \in \mathcal{D}$. Consider the case where the minimum distance between O_i and O_j is less than the constant $2d_0$. A discontinuity occurs on the equidistant curve of the two obstacles. If $d(\boldsymbol{x}) < d_0$, the second terms in Eq. 4, and Eq. 5, namely $(1 - \alpha(\boldsymbol{x}))\psi_g(\boldsymbol{x})$ and $(\psi_g(\boldsymbol{x}) - \psi(cp(\boldsymbol{x})))(1 - \alpha(\boldsymbol{x}))$, respectively, are nonzero. Approaching the equidistant curve in different directions (O_i side and O_j side), discontinuity arises since the obstacle that decides $\psi_g(\boldsymbol{x})$ switches instantaneously from O_i to O_j or from O_j to O_i. Obviously, the locations of distinct obstacles are not continuous. Figure 2(a)(b) give some concrete examples of the drawbacks of a constant d_0. To make things worse, even for a single obstacle, the closest point function itself, $cp(\boldsymbol{x})$,

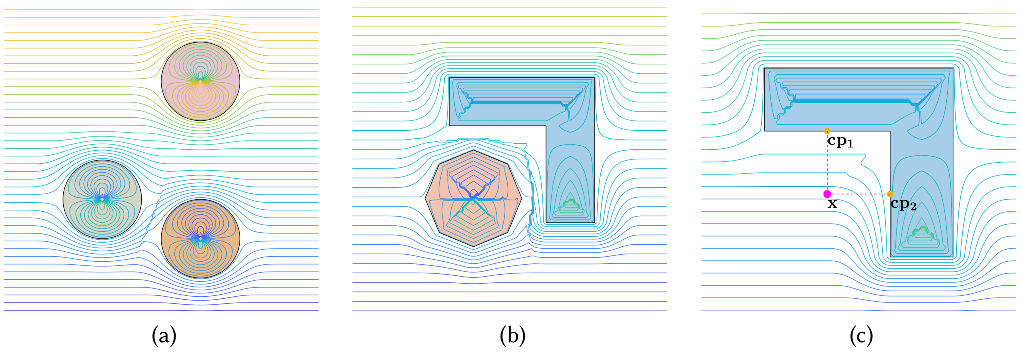


Fig. 2. Potential isocontours under the *additive* modulation function by Chang et al. [2022]: (a) shows that the discontinuity caused by $\psi_g(\boldsymbol{x})$ and a constant d_0 happens on the equidistant curve between two circles below with smooth boundary. Also, the spurious tangential flow is eliminated. (b) For polygonal objects, discontinuity also appears on the equidistant curve between objects that are close to each other. (c) Given a L-shaped object and a query point $\boldsymbol{x}$, cp_1 and cp_2 are both valid closest point on the boundary of the object to $\boldsymbol{x}$.

also contributes to the discontinuity of ψ', especially if the closest obstacle to $\boldsymbol{x}$ is non-convex. By the nature of a non-convex obstacle, any point on the exterior medial axis has more than one closest point on the boundary of the obstacle. It follows that $cp(\boldsymbol{x})$ is not unique. Figure 2(c) shows $cp(\boldsymbol{x})$ can have more than one reasonable output for a nonconvex object.

4 THE METHOD

4.1 LogSumExp Minimal Distance

Fortunately, there exists a solution for the issue of the hard-minimum-based Euclidean distance function $d_i(\boldsymbol{x})$ being only C^0. In their discussion of possible future work, Chang et al. [2022] suggested the possibility of replacing Curl-Noise's Euclidean distance measure with the smoothed distance function proposed by Madan and Levin [2022], which builds on the LogSumExp (LSE) approximation of the maximum function. We adopt their terminologies but slightly modify the notations in their Equation (3), by letting $\hat{d}_i(\boldsymbol{x})$ be their LSE distance function defined as

$$\hat{d}_i(\boldsymbol{x}) = -\frac{1}{a}\log\left(\sum_{f_n \in F_i} w_n(\boldsymbol{x})\exp(-a d_n^*(\boldsymbol{x}))\right), \tag{6}$$

where F_i is the set of primitives composing the i^{th} object O_i, $w_n(\boldsymbol{x})$ is a polynomial, d_n^* is the Euclidean distance from $\boldsymbol{x}$ to the simplex f_n in O_i, and a is a large positive constant that simply controls how tightly this "smoothmin" mimics the true minimum function; see [Madan and Levin 2022] for an illustration (we used $a = 200$ for all our results). This construction is at least C^1. With this replacement, the "distance" to the closest object is now $d(\boldsymbol{x}) = \min\{\hat{d}_1(\boldsymbol{x}), \hat{d}_2(\boldsymbol{x}), \cdots, \hat{d}_n(\boldsymbol{x})\}$. The LSE distance is designed to be conservative [Madan and Levin 2022], so for an object O, the zero isocontour of the LSE distance strictly encloses the object and induces a slightly modified boundary. Later, we will refer to it as the LSE boundary and denote it as ∂LSE_O.

To completely eliminate non-differentiability in $d(\boldsymbol{x})$, it remains to employ a differentiable approximation to $\min\{\cdot\}$ as follows:

$$d(\boldsymbol{x}) = \min\{\hat{d}_1(\boldsymbol{x}), \hat{d}_2(\boldsymbol{x}), \cdots, \hat{d}_n(\boldsymbol{x})\} \approx \overline{d}^1(\boldsymbol{x}) := \frac{\sum_{i=1}^n \hat{d}_i(\boldsymbol{x})\exp(-b\hat{d}_i(\boldsymbol{x}))}{\sum_{i=1}^n \exp(-b\hat{d}_i(\boldsymbol{x}))}, \tag{7}$$

where b is another large positive constant controlling the smooth maximum's accuracy (we used $b = 200$ for all our results) and $\hat{d}_k(\boldsymbol{x})$ is defined by Eq. 6, $\forall k \in [n]$. Since $\hat{d}_k(\boldsymbol{x})$, $\forall k \in [n]$ and $\overline{d}^1(\boldsymbol{x})$ are both at least C^1, then $d(\boldsymbol{x})$ in Eq. 7 is also at least C^1. We will use this new notation of $\overline{d}^1(\boldsymbol{x})$ throughout the rest of this paper. Figures 5(a) and 5(d) make a comparison between a C^0 distance field and its C^1 alternative using our method.

4.2 Self-Adaptive Influence Radius

After adopting the smoothed distance measure, we next tackle the discontinuity caused by the constant maximum influence radius d_0 around obstacles. One obvious option would be to simply set the influence radius to a very small value, such that the influence region of one object never interferes with that of another. However, the default region of influence should be a user-chosen parameter, dependent on the scene they are attempting to model, and setting a tiny influence radius would largely destroy the visual benefits of smooth ramping. Instead, we present a differentiable, self-adaptive function to modify the influence radius in difficult scenarios. In this function, the influence radius is co-determined by the query point position and the objects' locations. As the influence radius is now a function rather than a constant, we change our notation to denote the influence radius as $d_0(\boldsymbol{x})$.

All of the proofs we provide in this section either use Eq. 4 to modulate ψ or, alternatively, use Eq. 5 with the added assumption that $cp(\boldsymbol{x})$ is at least C^1. (We will present our differentiable construction of $cp(\cdot)$ later in Section 4.3.) Furthermore, we assume the user-defined potential ψ is also at least C^1 differentiable.

Given a query point $\boldsymbol{x}$, we first define the sorted list of LSE distances from $\boldsymbol{x}$ to the set of objects as $\hat{d}_{s_1}(\boldsymbol{x}), \hat{d}_{s_2}(\boldsymbol{x}), \cdots, \hat{d}_{s_n}(\boldsymbol{x})$, such that $\hat{d}_{s_1}(\boldsymbol{x}) \leq \hat{d}_{s_2}(\boldsymbol{x}) \leq \cdots \leq \hat{d}_{s_n}(\boldsymbol{x})$ and $s_i \in [n]$. That is, we sort the LSE distances in increasing order and reindex them using s_i. In Section 1 of our supplementary material, we prove that setting the influence radius to be $d_0(\boldsymbol{x}) = \min\{\hat{d}_{s_2}(\boldsymbol{x}), d_0\}$, where d_0 is the previous constant maximum influence radius and $\hat{d}_{s_2}(\boldsymbol{x})$ is the LSE of the *second closest* object, suffices to smooth out $\psi'(\boldsymbol{x})$ from being discontinuous to C^0 (but not C^1). The intuition for this construction is that if the second closest object is more than d_0 away from the query point, we can ignore that object; if it is closer, then we use the distance to that object as the maximum influence radius for the closest object (Figure 3). Since the identity of the closest object changes when the query point is equidistant from the two closest objects, we have $d_0(\boldsymbol{x}) = \hat{d}_{s_2}(\boldsymbol{x}) = \hat{d}_{s_1}(\boldsymbol{x})$ (when both objects are nearby); thus the function will be continuous through this transition.

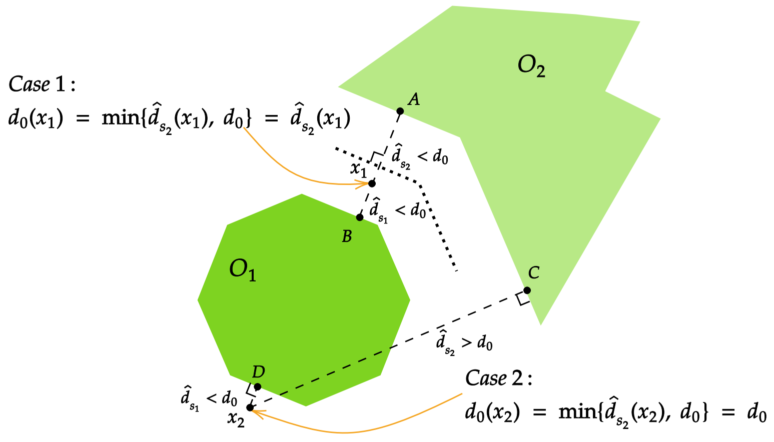


Fig. 3. An illustration of the motivation for the self-adaptive $d_0(\boldsymbol{x})$ with the hard-min function. For query point $\boldsymbol{x_1}$, since $\hat{d}_{s_1}(\boldsymbol{x_1}), \hat{d}_{s_2}(\boldsymbol{x_1}) < d_0$, then we have $d_0(\boldsymbol{x}) = \hat{d}_{s_2}(\boldsymbol{x_1})$. For query point $\boldsymbol{x_2}$, since $\hat{d}_{s_1}(\boldsymbol{x_2}) < d_0$, but , $\hat{d}_{s_2}(\boldsymbol{x_2}) > d_0$, then we have $d_0(\boldsymbol{x}) = d_0$.

If one looks closely at the proof (Section 1 of the supplementary material), it becomes clear that the $\min\{\cdot\}$ function in $d_0(\boldsymbol{x})$ and the C^0 function $\hat{d}_{s_2}(\boldsymbol{x})$ are two possible causes preventing $\psi'(\boldsymbol{x})$ from being C^1. Motivated by this fact, the guiding principle in designing our ultimate $d_0(\boldsymbol{x})$ is that this function should be a differentiable approximation of $\hat{d}_{s_2}(\boldsymbol{x})$. We therefore write the problem as follows: given a query point $\boldsymbol{x}$ and n LSE distances, $\hat{d}_1(\boldsymbol{x}), \hat{d}_2(\boldsymbol{x}), \cdots, \hat{d}_n(\boldsymbol{x})$, we want to approximate $\hat{d}_{s_2}(\boldsymbol{x})$ by some function that is at least C^1.

Now, we present our differentiable approximator. First, we use an LSE-based function to approximate the min-2-sum from below, which is defined as the minimum element in the set S given by

$$S = \{\hat{d}_j(\boldsymbol{x}) + \hat{d}_k(\boldsymbol{x}) : j \neq k, \quad j,k \in [n]\}, \quad \text{where } |S| = \binom{n}{2}. \tag{8}$$

Adapting a known max-2-sum formula [Todd 2018] to a min-2-sum, the approximation formula is

$$\begin{aligned}\min S &= \hat{d}_{s_1}(\boldsymbol{x}) + \hat{d}_{s_2}(\boldsymbol{x}) \\ &\approx \overline{d}^2 = \frac{n-2}{c}\log\left(\sum_{i=1}^{n}\exp(-c\hat{d}_i(\boldsymbol{x}))\right) - \frac{1}{c}\sum_{i=1}^{n}\log\left(\sum_{j=1, j\neq i}^{n}\exp(-c\hat{d}_j(\boldsymbol{x}))\right)\end{aligned} \tag{9}$$

where c is another large positive constant (we used $c = 200$ for all our results). We use the notation $\overline{d}^2$ for the min-2-sum with the constant c for all results in the paper. Next, we approximate $\hat{d}_{s_2}(\boldsymbol{x})$ by subtracting the result of Eq. 7 from Eq. 9, which gives $\hat{d}_{s_2}(\boldsymbol{x}) \approx \tilde{d}^2(\boldsymbol{x}) := \overline{d}^2(\boldsymbol{x}) - \overline{d}^1(\boldsymbol{x})$. In other words, we smoothly approximate the second smallest value as the difference of smooth approximations of the min-2-sum and the minimum.

Finally, to complete the whole process of smoothing $d_0(\boldsymbol{x}) = \min\{\hat{d}_{s_2}(\boldsymbol{x}), d_0\}$, we use the LSE function as an underestimation of $\min\{\cdot\}$, since we want d_0 to be the maximum influence radius.

We summarize the whole process in Algorithm 1 and reveal the benefit of this design by comparing Figures 5(b) and 5(e).

Algorithm 1 Continuously Differentiable $d_0(\boldsymbol{x})$

Input: Query point $\boldsymbol{x}$, LSE distances from $\boldsymbol{x}$ to all n objects $\hat{d}_1(\boldsymbol{x}), \hat{d}_2(\boldsymbol{x}), \cdots, \hat{d}_n(\boldsymbol{x})$, large positive constants a, b, c, and the maximum influence radius d_0.
Output: $d_0(\boldsymbol{x})$.

```
1: min_dist = $\frac{\sum_{i=1}^{n} \hat{d}_i(\boldsymbol{x}) \exp(-b\hat{d}_i(\boldsymbol{x}))}{\sum_{i=1}^{n} \exp(-b\hat{d}_i(\boldsymbol{x}))}$    ▷ Eq. 7.
2: min_two_sum = $\frac{n-2}{c} \log\left(\sum_{i=1}^{n} \exp(-c\hat{d}_i(\boldsymbol{x}))\right) - \frac{1}{c}\sum_{i=1}^{n} \log\left(\sum_{j=1, j\neq i}^{n} \exp(-c\hat{d}_j(\boldsymbol{x}))\right)$    ▷ Eq. 9.
3: $\tilde{d}^2(\boldsymbol{x})$ = min_two_sum - min_dist
4: $d_0(\boldsymbol{x})$ = $-\frac{1}{a}\log(\exp(-a\tilde{d}^2(\boldsymbol{x})) + \exp(-ad_0))$    ▷ $d_0(\boldsymbol{x}) \approx \min\{\hat{d}_{s_2}(\boldsymbol{x}), d_0\}$
```

4.2.1 Discussion.

Our resulting construction of the adaptive influence radius $d_0(\boldsymbol{x})$ is at least C^1, essentially because all its constituent components are differentiable. However, we must further explain why this deliberately constructed combination of overestimation and underestimation in turn guarantees that the final potential $\psi'(\boldsymbol{x})$ is differentiable, whether $\psi'(\boldsymbol{x})$ is defined by Eq. 4 or Eq. 5. In section 1 of the supplementary material, *Observation 5* notes that $\psi_g(\boldsymbol{x})$ (the targeted constant surface ψ value on the closest object) is a Heaviside step function that is discontinuous on the equidistant curve between the closest and the second closest object to $\boldsymbol{x}$. Therefore, $\psi_g(\boldsymbol{x})$ is not differentiable whenever $\hat{d}_{s_1}(\boldsymbol{x}) = \hat{d}_{s_2}(\boldsymbol{x})$. In particular, the first order partial derivatives of $\psi_g(\boldsymbol{x})$ are linear combinations of Dirac delta functions, whose values are almost everywhere zero except query points precisely on the equidistant curves. Considering the differentiability of the general form of potential modulation given by Eq. 3 in Section 1 of our supplementary material, the only hope to hide the discontinuity of $\frac{\partial \psi_g(\boldsymbol{x})}{\partial x}$ and $\frac{\partial \psi_g(\boldsymbol{x})}{\partial y}$ is to let $1 - \alpha(\boldsymbol{x})$ be zero when $\hat{d}_{s_1}(\boldsymbol{x}) = \hat{d}_{s_2}(\boldsymbol{x})$, since $0 \cdot \delta(\boldsymbol{x}) = 0$. This means we need $\frac{d(\boldsymbol{x})}{d_0(\boldsymbol{x})} \geq 1$ when $\hat{d}_{s_1}(\boldsymbol{x}) = \hat{d}_{s_2}(\boldsymbol{x})$.

In our case, since $\overline{d}^1$ is an overestimation of $\hat{d}_{s_1}$ and $\overline{d}^2$ is an underestimation of $\hat{d}_{s_1} + \hat{d}_{s_2}$, then no matter which of $\hat{d}_{s_2}$ or d_0 is smaller, $\frac{d(\boldsymbol{x})}{d_0(\boldsymbol{x})}$ will always be an overestimation. In particular, when $\hat{d}_{s_1}(\boldsymbol{x}) = \hat{d}_{s_2}(\boldsymbol{x})$, we will always have $\frac{d(\boldsymbol{x})}{d_0(\boldsymbol{x})} \geq 1$ and thus

$$\alpha(\boldsymbol{x}) = ramp\left(\underbrace{\frac{d(\boldsymbol{x})}{d_0(\boldsymbol{x})}}_{\geq 1}\right) = 1,$$

which ultimately gives $1 - \alpha(\boldsymbol{x}) = 0$ and $\nabla\alpha(\boldsymbol{x}) = 0$. Hence, this construction conceals the discontinuity in $\psi'(\boldsymbol{x})$ and $\nabla\psi'(\boldsymbol{x})$ caused by $\psi_g(\boldsymbol{x})$. We leave the detailed proof of $\psi'(\boldsymbol{x})$ being a C^1 function to the supplementary material (Section 2). At this point, Eq. 4 has become differentiable. However, Eq. 5 is still not differentiable due to the closest point function.

4.3 Replacing the Closest Point Function

Let us complete our task by removing the discontinuities caused by the closest boundary point function, $cp(\cdot)$. Instead of mapping the query point $\boldsymbol{x}$ to the closest boundary point on its closest object, we generalize this idea by replacing $cp(\cdot)$ with a continuously differentiable function mapping $\boldsymbol{x}$ to some point on the boundary of its closest object. In addition to at least first-order differentiability, an ideal replacement would map query points to the boundary in a manner that yields relatively uniformly distributed points on the boundary. In other words, given a set of query points evenly scattered on some isocontour of the LSE distance function, the perfect alternative should have their images evenly spaced along the boundary.

However, even if we find such a map, the image itself is generally a C^0 function, since polygonal boundary geometry is not differentiable at its vertices. To circumvent this issue, we use the zero isocontour of the LSE distance function as a proxy for the polygonal boundary of the closest object to $\boldsymbol{x}$. The implementation details of discretizing the LSE boundary are described in Appendix B. Therefore, Eq. 5 can be rewritten as:

$$\psi'(\boldsymbol{x}) = \psi(\boldsymbol{x}) + (\psi_g(\boldsymbol{x}) - \psi(\boldsymbol{m}(\boldsymbol{x})))(1 - \alpha(\boldsymbol{x})), \tag{10}$$

where $\boldsymbol{m}(\boldsymbol{x}) : \mathbb{R}^n \longrightarrow \partial LSE_{O_i} \in \mathbb{R}^2$ is the C^1 function that we will use to replace $cp(\cdot)$, given that O_i is the closest object to $\boldsymbol{x}$. Below, we propose a possible C^1 mapping for the 2D setting we consider.

To tackle this problem, we will interpret the Euclidean coordinates of our query point $\boldsymbol{x} = (x, y)$ as a complex number $z = x + iy \in \mathbb{C}$ and incorporate the idea of the Schwarz-Christoffel (SC) exterior mapping to construct $\boldsymbol{m}(\boldsymbol{x}) = \boldsymbol{m}(z)$. Being almost everywhere biholomorphic (Appendix A) in our finite and bounded domain of interest $\mathcal{D}$, the SC exterior mapping, defined by $f : \mathbb{D} \longrightarrow \mathbb{V} \subseteq \mathbb{C}$, maps the interior of the complex unit disk, $\mathbb{D} = \{\zeta \in \mathbb{C} : |\zeta| < 1\}$, to the exterior of the polygon, $\mathbb{V}$. Then, we can write the inverse of the SC exterior mapping as $f^{-1} : \mathbb{V} \longrightarrow \mathbb{D}$. Following the conventional definition, we name $\mathbb{D}$ as the canonical domain and $\mathbb{V}$ as the physical domain.

Now, we propose the construction of a three-step mapping $\boldsymbol{m}(z)$. With the complex coordinate z, we first compute $w = f^{-1}(z)$, such that $|w| < 1$ is a point in $\mathbb{D}$. Next, we scale w and thereby project w to a point w' on a complex circle within an epsilon distance of the boundary of the unit disk, defined as $w' = (1 - \epsilon)\frac{w}{|w|} \in \mathbb{C}$. Finally, we calculate the SC exterior mapping, $f(w')$, mapping w' in the canonical domain back to a point on the boundary of the polygon in the physical domain.

In our implementation, given a query point $\boldsymbol{x} \in \mathbb{R}^2$ with object O_i being its closest object, we first discretize the LSE boundary of O_i in order to create a polygonal input for the inverse of the SC exterior map. We name the new polygon P_i with all vertices on the LSE boundary of O_i as the proxy polygon of O_i. For the SC exterior mapping, we use the `extermap` function provided by the MATLAB SC Toolbox ([Driscoll 1996]). To reduce time spent on computing the `extermap` and its inverse, the proxy polygon should contain as few vertices as possible. We achieve this by a simple curvature-dependent remeshing procedure.

We illustrate our overall mapping procedure in Figure 4 and summarize the process in Algorithm 2. Finally we make a remark on the differentiability of this construction of $\boldsymbol{m}(\boldsymbol{x})$. As we only care

Algorithm 2 Schwarz-Christoffel Mapping for $\boldsymbol{m}(\boldsymbol{x})$

Input: A query point $\boldsymbol{x} = (x, y)$, an object O, a constant $\epsilon > 0$.
Output: $\boldsymbol{m}(\boldsymbol{x})$.

```
1: $z = x + iy$                                    ▹ Convert a 2D coordinate to a complex number
2: $P$ = Discretize_LSE_boundary($O$)
3: $w$ = Inverse_SC_exterior_map($P, z$)           ▹ $w$ is in the Canonical Domain (unit disk).
4: $w' = (1-\epsilon)\frac{w}{|w|}$                ▹ Scaling $w'$ to be $\epsilon$ close to the boundary of the unit disk.
5: $\boldsymbol{m}(\boldsymbol{x})$ = SC_exterior_map($P, w'$)   ▹ Back to the Physical Domain
```

about the differentiability of $\boldsymbol{m}(\boldsymbol{x})$ in $\mathbb{R}^2$, we do not expect $\boldsymbol{m}(\boldsymbol{x})$ to be analytic in $\mathbb{C}$, while we do

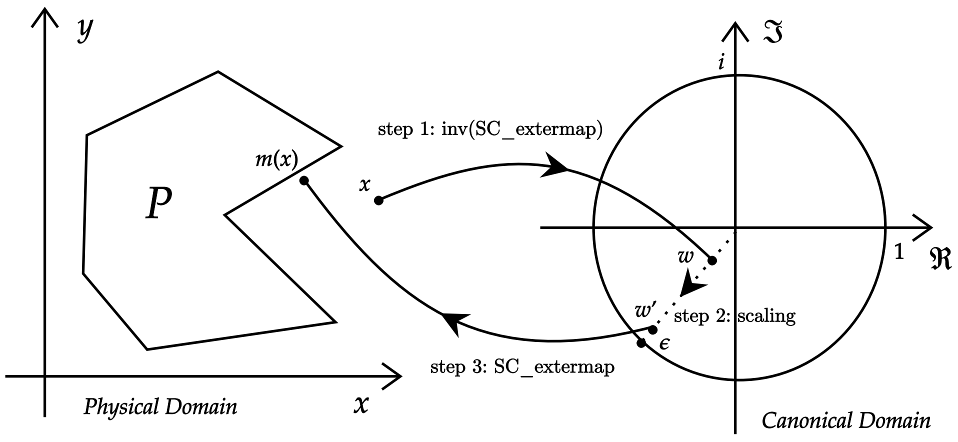


Fig. 4. An illustration for Algorithm 2.

require the real and complex components of $\boldsymbol{m}(\boldsymbol{x})$, corresponding to the x- and y-coordinates in $\mathbb{R}^2$, respectively, to both be differentiable in $\mathbb{R}^2$. This suffices to prove $\boldsymbol{m}(\boldsymbol{x})$ is differentiable in $\mathbb{R}^2$, as all components of $\boldsymbol{m}(\boldsymbol{x})$ are differentiable in $\mathbb{R}^2$. In particular, although scaling w in the second step of $\boldsymbol{m}(\boldsymbol{x})$ is not analytic in $\mathbb{C}$, both of its real and complex components are differentiable. A detailed proof of the componentwise differentiability of $\boldsymbol{m}(\boldsymbol{x})$ is provided in Appendix C.

5 RESULTS

We combined the LogSumExp Minimal Distance (Section 4.1), self-adaptive influence radius (Section 4.2), and our alternative to $cp(\boldsymbol{x})$ (Section 4.3) all together to construct our proposed potential modulation function ψ' that is almost everywhere C^1. We compare the three different modulated potentials ψ' designed by Bridson et al. [2007], Chang et al. [2022], and ourselves to show that our ψ' is indeed C^1, and demonstrate the benefits of such a C^1 potential field. Figure 6 shows tests on a collection of complex obstacles, where the advantages of our approach are apparent.

Flowing Particles. While the isocontour visualizations indicate the flow field structure and direction, they do not indicate the speeds. To do so, we flow passive particles through various example fields $\mathbf{v}$, as shown in our supplementary video. In our implementation, we applied a simple Euler's method to update the positions of particles. The results show that the discontinuities present in the previous constructions prevent particles from flowing naturally and smoothly through the flow field, while ours offers more attractive results.

With small time steps (e.g., $dt = 2 \times 10^{-4}$) most particles will adhere to the same isocontour as the one they started on, throughout the animation. For larger time steps some isocontour drift becomes apparent, but in a typical application scenario this drift causes no visual artifacts. However, if desired, we can force every particle to strictly follow its initial isocontour line with larger time steps (e.g., $dt = 2 \times 10^{-3}$) via an isocontour error correction procedure after each step. Since our ψ' is now C^1, such a correction can be achieved by adapting a variant of Newton's method by Chopp [2001], which requires only $\nabla\psi'$: we store the potential value of each particle once it is seeded and project the updated location of the particles to the closest isocontour that shares the same initial potential. This technique conveniently compensates for drift that can arise during particle time integration, especially for large time steps, albeit at the cost of the extra evaluations of ψ' and $\nabla\psi'$ required for the Newton's method. Our video also includes a comparison with and without isocontour error correction. Later (in Section 6), we provide more details about the extra

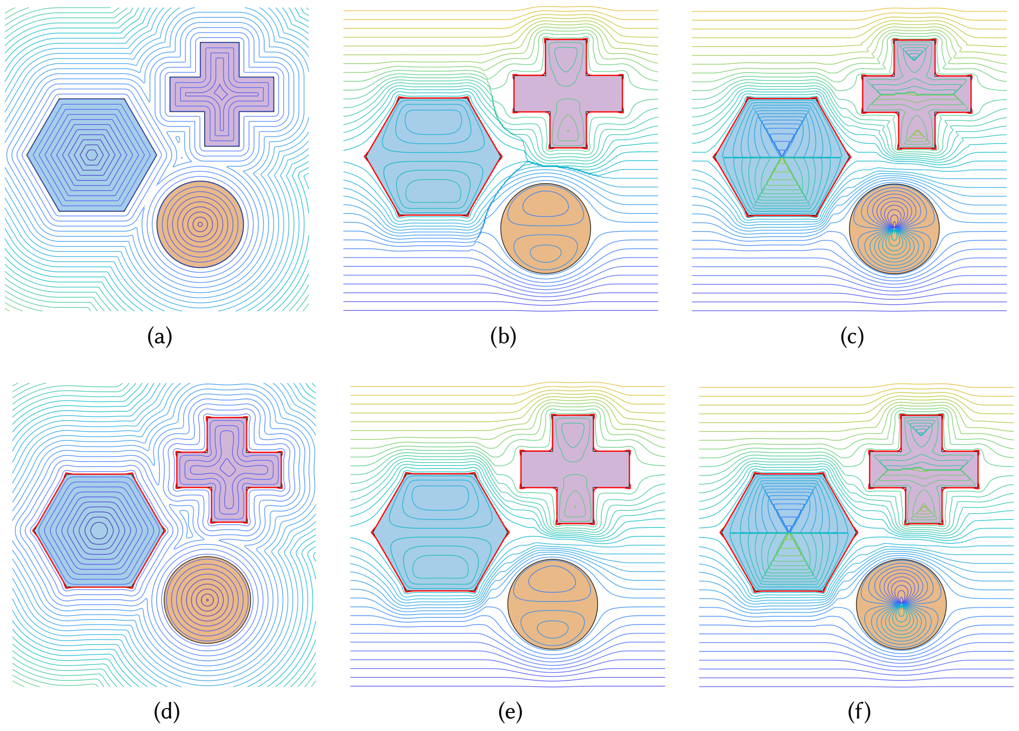


Fig. 5. (Scenario 1) The zero isocontours for several polygonal objects in an LSE distance field are shown in red. (a) The contour plot of a C^0 distance field using the $\min\{\cdot\}$ function. (b) The contour plot of multiplicative ψ' with $d_0(\boldsymbol{x}) \equiv 0.1$. There are discontinuities between the objects. (c) The contour plot of *additive* ψ' with self-adaptive $d_0(\boldsymbol{x})$. Discontinuities are caused by $cp(\boldsymbol{x})$ at the nonconvex corners of the plus-shape. (d) The contour plot of a C^1 distance field using Eq. 6 and Eq. 7. (e) The contour plot of multiplicative ψ' with our self-adaptive influence radius, where discontinuities are avoided. (f) The contour plot of *additive* ψ' with self-adaptive $d_0(\boldsymbol{x})$ and SC exterior map. Discontinuities at the nonconvex corners are removed.

computation time caused by isocontour error correction (referred to briefly as correction). The decision of whether or not to use the isocontour error correction will be application-dependent.

6 PERFORMANCE

While our focus was on achieving smoothness of the resulting vector fields, performance is nevertheless important to consider. Here we discuss the time performance of our prototype implementation. Cost scales in proportion to the number of particles and we consider only the particles remaining inside our domain of interest, $\mathcal{D}$, at any point in time. Therefore, in general, the time consumption for one frame of our animations will decrease as particles gradually exit the domain. We coded our method in MATLAB and recorded time information using MATLAB's built-in functions `tic` and `toc`. All of the following time data are collected on a machine with an Intel(R) Xeon(R) Silver 4316 CPU @ 2.30GHz CPU processor.

We evaluate the performance of our method from two different aspects. First, we compare the amount of time spent to generate each frame (of animation) between the original Curl-Noise by Bridson et al. [2007], the multiplicative and additive designs by Chang et al. [2022], and our

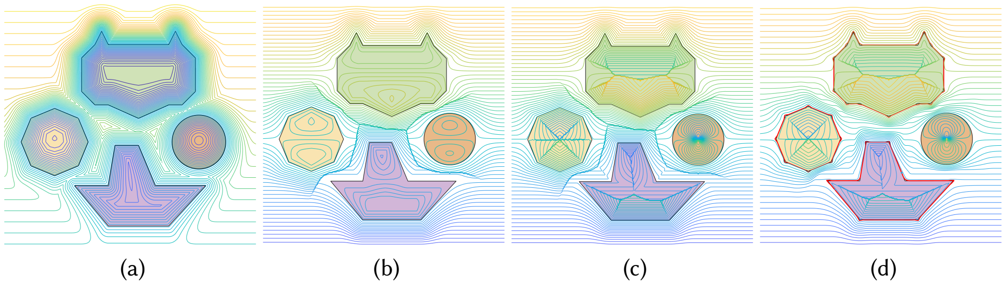


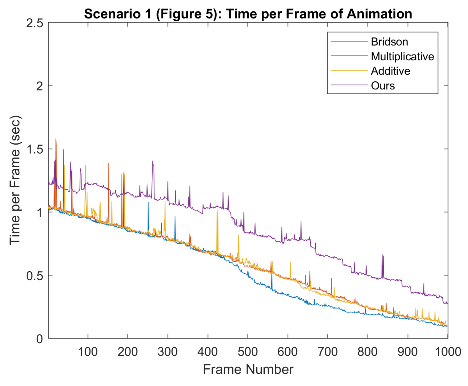
Fig. 6. (Scenario 2) Comparison of potential ψ', modulated by Bridson et al. [2007], Chang et al. [2022] and our differentiable approach. (a) ψ' modulated by the basic multiplicative ramping strategy (Eq. 3). (b) ψ' modulated by the corrected multiplicative ramping strategy (Eq. 4). (c) ψ' modulated by the *additive* ramping strategy (Eq. 5). (d) ψ' modulated according to our method. The LSE (proxy) boundaries of the polygonal objects are shown in red.

Table 1. Comparison of Time per Frame Under Different Methods.
Time format: mean ± std ∈ [min, max]. Time Unit: seconds

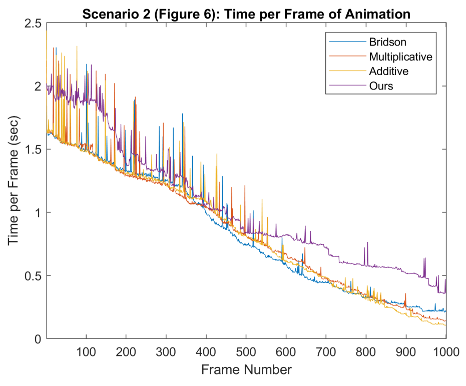
Methods	Scenario 1 (Figure 5)	Scenario 2 (Figure 6)
Bridson et al. [2007]	$0.66 \pm 0.39 \in [0.12, 1.87]$	$1.05 \pm 0.60 \in [0.26, 2.77]$
Chang et al. [2022] (Multiplicative)	$0.72 \pm 0.36 \in [0.12, 1.98]$	$1.07 \pm 0.60 \in [0.17, 2.88]$
Chang et al. [2022] (Additive)	$0.72 \pm 0.37 \in [0.12, 1.87]$	$1.06 \pm 0.62 \in [0.13, 2.89]$
Ours (without correction)	$0.86 \pm 0.30 \in [0.28, 1.43]$	$1.07 \pm 0.48 \in [0.38, 2.20]$
Ours (with correction)	$3.38 \pm 1.69 \in [0.65, 8.45]$	$3.91 \pm 2.43 \in [0.83, 10.96]$

method. The comparison is made in the context of Figure 5 (Scenario 1) and Figure 6 (Scenario 2); the animations can be viewed in our supplementary material. Based on an animation with 1000 frames, we report the numerical time results in Table 1 in the form *mean* ± *standard deviation* ∈ *[min, max]* with seconds as the unit of time. The decreasing trend of time per frame can be clearly visualized in Figure 7. It can be seen that our method is slightly more expensive than its predecessors when error correction is disabled; this is the typical setting we use. If error correction is desired, the computational cost jumps significantly, so it should only be used if one strictly requires that particles stay on their initial streamlines.

Next, we consider how time per frame changes as the size of our problems increases. As the computational cost for the Schwarz-Christoffel (SC) exterior mapping is the dominating component, we focus on the factors that affect the SC exterior mapping. Two facets that vary the problem size are the number of polygons and the number of vertices. We consider scenarios with 1, 2, 4, 8, or 16 objects defined in the domain $\mathcal{D}$, chosen to be regular polygons with 4, 6, 8, 12, or 16 vertices. Figure 8 demonstrates the exact locations of the objects and Table 2 records the time per frame for each case. Roughly, cost increases proportionally to the number of objects and to the number of vertices.

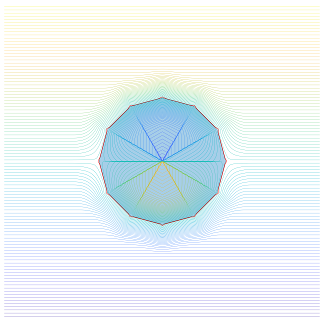


(a)

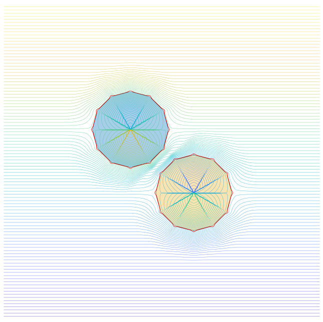


(b)

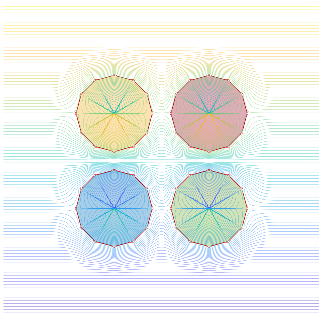
Fig. 7. A visualization of the overall decreasing trend of time per frame in the context of Figure 5 (Scenario 1) and Figure 6 (Scenario 2). Our method is typically more expensive, but not by a large margin.



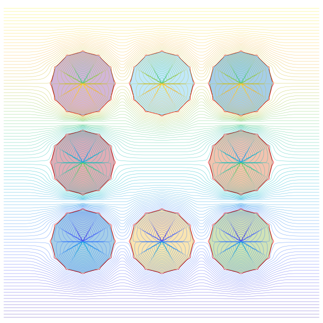
(a)



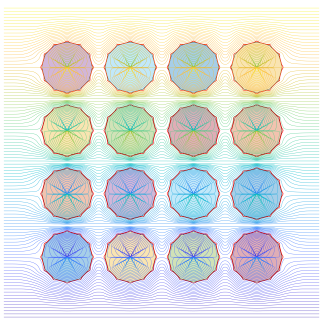
(b)



(c)



(d)



(e)

Fig. 8. Locations of the regular polygons used for our scaling evaluation. The five plots above use dodecagons for illustration; our benchmark process also tests squares, hexagons, octagons, decagons, and hexadecagons.

Table 2. Run Time Performance as Problem Scales Up.
Time format: mean ± std range in [min, max]. Time Unit: seconds.

# of objects \ # of vertices	4	6	8	12	16
1	0.28 ± 0.09 range in $[0.13, 0.60]$	0.34 ± 0.14 range in $[0.12, 0.75]$	0.80 ± 0.33 range in $[0.27, 1.76]$	0.76 ± 0.28 range in $[0.31, 1.53]$	1.71 ± 0.67 range in $[0.58, 3.09]$
2	0.33 ± 0.11 range in $[0.16, 0.68]$	0.42 ± 0.14 range in $[0.17, 0.90]$	1.13 ± 0.42 range in $[0.40, 2.39]$	1.87 ± 0.65 range in $[0.69, 3.14]$	1.99 ± 0.76 range in $[0.74, 3.63]$.
4	0.37 ± 0.12 range in $[0.18, 0.79]$	0.49 ± 0.17 range in $[0.23, 1.05]$	1.26 ± 0.52 range in $[0.48, 2.85]$	2.11 ± 0.91 range in $[0.73, 4.62]$	2.37 ± 1.10 range in $[0.69, 4.90]$.
8	0.63 ± 0.18 range in $[0.32, 1.01]$	1.06 ± 0.41 range in $[0.38, 1.86]$	1.76 ± 0.67 range in $[0.58, 3.24]$	2.01 ± 0.73 range in $[0.75, 4.09]$	2.91 ± 1.12 range in $[1.00, 5.37]$.
16	1.09 ± 0.28 range in $[0.61, 2.08]$	1.88 ± 0.65 range in $[0.76, 3.66]$	2.60 ± 0.86 range in $[1.19, 4.80]$	3.46 ± 1.14 range in $[1.66, 6.17]$	3.96 ± 1.30 range in $[1.95, 7.16]$.

7 CONCLUSIONS AND FUTURE WORK

We have presented an extension of the Curl-Noise method for procedural flow design in two dimensions to avoid obvious artifacts due to non-differentiability in the presence of obstacles. Our results demonstrate that the method is effective in producing smooth flows for a range of scenes with polygonal obstacles in mutual proximity. Although we have only explored static obstacles, we believe our approach could be naturally extended to moving obstacles.

One drawback of our method is its additional complexity. While the smoothed distance function and modified influence radius are relatively inexpensive, the existing Schwarz-Christoffel mapping procedure (`extermap`) is somewhat expensive to evaluate for polygons with many vertices. We noticed that the SC Toolbox we are currently using is unable to accurately calculate the SC exterior map for a polygon with hundreds of vertices in a reasonable amount of time. Further research into an accurate, robust, scalable, and optimized numerical algorithm (and its implementation) to construct and evaluate the Schwarz-Christoffel exterior map would therefore be valuable.

Another important challenge is extending to three dimensions. While the smooth approximations to the $\min\{\cdot\}$ function and the self-adaptive d_0 can naturally be generalized to design 3D fluid flows, we do not yet have a satisfying C^1 alternative to $cp(\boldsymbol{x})$ in 3D. It is nontrivial to extend the idea of the Schwarz-Christoffel exterior mapping from 2D to 3D since there is no complex counterpart for 3D Euclidean coordinate systems. One possibility we have preliminarily explored is to approach the boundary of the objects from a query point using a Ray Marching algorithm, obeying the following update rule:

$$\boldsymbol{x} := \boldsymbol{x} - d(\boldsymbol{x})\nabla d(\boldsymbol{x}). \tag{11}$$

To maintain the differentiability of our method, this expression requires $\nabla d(\boldsymbol{x})$ to be a C^1 function. It follows that $d(\boldsymbol{x})$, as well as $\hat{d}_i(\boldsymbol{x}), i \in [n]$, should be at least C^2. Currently, the LSE-based distance constructed by Madan and Levin [2022] only guarantees the continuity of the first order derivative of $\hat{d}_i(\boldsymbol{x}), i \in [n]$. Although one can achieve second order differentiability by slightly modifying the *weight function* of Madan and Levin [2022] such that $\frac{\partial^2 w_\ell}{\partial \phi_\ell^2} = 0, \forall \ell \in F_i, \forall i \in [n]$, where F_i is set of primitives composing the object O_i (more details are provided in Section 3 of our supplementary material), $\nabla d(\boldsymbol{x})$ usually experiences sudden changes near non-convex corners of objects. Unfortunately, we have found that this sudden change in the gradient results in strong visual artifacts, even in 2D scenarios, rendering this approach impractical.

Finally, we would like to pursue further optimizations for our method and its implementation. Since each particle's calculation is independent, a parallelized implementation would naturally provide a convenient speedup, e.g., on GPU.

ACKNOWLEDGMENTS

The authors thank Jumyung (JC) Chang for early discussions of the behavior of curl-noise schemes near objects in close proximity. This research was supported in part by the Natural Sciences and Engineering Council of Canada (NSERC) Discovery Grant program (RGPIN-2021-02524).

REFERENCES

Robert Bridson, Jim Houriham, and Marcus Nordenstam. 2007. Curl-Noise for Procedural Fluid Flow. *ACM Trans. Graph.* 26, 3 (Jul 2007), 46–es. https://doi.org/10.1145/1276377.1276435

Jumyung Chang, Ruben Partono, Vinicius C. Azevedo, and Christopher Batty. 2022. Curl-Flow: Boundary-Respecting Pointwise Incompressible Velocity Interpolation for Grid-Based Fluids. *ACM Trans. Graph.* 41, 6 (2022).

David Chopp. 2001. Some Improvements of the Fast Marching Method. *Siam Journal on Scientific Computing* 23 (06 2001). https://doi.org/10.1137/S106482750037617X

Ivan DeWolf. 2006. Divergence-free noise. *Martian Labs* (2006).

Tobin A. Driscoll. 1996. Algorithm 756: A MATLAB Toolbox for Schwarz-Christoffel Mapping. *ACMMathSoft* 22, 2 (Jun 1996), 168–186. https://doi.org/10.1145/229473.229475

Peter Henrici. 1974. *Applied and computational complex analysis: Vol.: 1.: Power series, integration, conformal mapping, location of Zeros.* John Wiley and Sons, New York, NY.

Theodore Kim, Nils Thürey, Doug James, and Markus Gross. 2008. Wavelet turbulence for fluid simulation. *ACM Transactions on Graphics (TOG)* 27, 3 (2008), 1–6.

Abhishek Madan and David I. W. Levin. 2022. Fast evaluation of smooth distance constraints on co-dimensional geometry. *ACM Transactions on Graphics* 41, 4 (Jul 2022), 1–17. https://doi.org/10.1145/3528223.3530093

Rahul Narain, Jason Sewall, Mark Carlson, and Ming C Lin. 2008. Fast animation of turbulence using energy transport and procedural synthesis. *ACM Transactions on Graphics (TOG)* 27, 5 (2008), 1–8.

Z. Nehari. 1952. *Conformal Mapping.* McGraw-Hill, New York, NY. 189–196 pages.

Zherong Pan, Jin Huang, Yiying Tong, Changxi Zheng, and Hujun Bao. 2013. Interactive localized liquid motion editing. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–10.

Hagit Schechter and Robert Bridson. 2008. Evolving sub-grid turbulence for smoke animation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics symposium on Computer animation.* 1–7.

Michael J. Todd. 2018. On Max-k-Sums. *Math. Program.* 171, 1–2 (Sep 2018), 489–517. https://doi.org/10.1007/s10107-017-1201-0

Wolfram Von Funck, Holger Theisel, and Hans-Peter Seidel. 2006. Vector field based shape deformations. *ACM Transactions on Graphics (ToG)* 25, 3 (2006), 1118–1125.

A SC EXTERIOR MAP IS BIHOLOMORPHIC ALMOST EVERYWHERE

Here we argue that the SC exterior mapping is almost everywhere biholomorphic in a finite domain of interest, $\mathcal{D}$. The standard SC mapping, which maps the upper-half plane onto the interior of a polygon, is the biholomorphic function whose existence is assured [Nehari 1952] by the Riemann Mapping Theorem. As mentioned by [Driscoll 1996], the standard SC mapping is analytic in $\mathbb{C}^+ \setminus \{z_1, z_2, \cdots, z_n\}$, where the z_i are pre-vertices, and it can be continuously extend to $\mathbb{C}^+$. Composing the standard SC mapping with a Möbius transformation, we can map the interior of a unit disk to the interior of a polygon, which is called the SC disk mapping. Notice that a Möbius transformation is invertible and its inverse is another Möbius transformation. Therefore, except for the non-differentiability on the polygon vertices, only two other points on the boundary of the polygon will be non-differentiable due to the Möbius transformation and its inverse. Therefore, the SC disk mapping is still almost everywhere (in the mathematical sense) biholomorphic.

In our case, as the query points are located in the exterior of the polygon, we wish to map the interior of a unit disk to the exterior of a polygon, known as the SC exterior map. This can be done by simply reversing the order of the polygonal vertices when we construct the SC disk map. The explicit formula of this map is [Henrici 1974]:

$$f(z) = A + B \int_0^z s^{-2} \prod_{k=1}^{n} (s - z_k)^{\beta_k} ds, \tag{12}$$

where z_k are the pre-vertices on the complex unit circle and $-\beta_k \pi$ defines the exterior angles of the polygon. Apart from the sources of non-differentiability mentioned above on the boundary of the polygon, the SC exterior mapping introduces another singular point in the interior of the unit disk, which is the pre-image of ∞. In the SC Toolbox, the singular point is selected to be the origin of the unit disk. Therefore, as our work considers a finite and bounded domain $\mathcal{D}$, we will never encounter the singular point in the process. Because there are only a finite number of non-differentiable points on the boundary of the polygon, we conclude that the SC exterior mapping is almost everywhere biholomorphic.

B DISCRETIZING THE LOGSUMEXP (LSE) BOUNDARY

We describe the implementation details of how we explicitly discretize the smoother LSE boundaries for polygonal objects; this new mesh is necessary for constructing the SC exterior map. Given an object O given as a polygon, we first finely subsample all edges of the object in counterclockwise order and denote the sample points as $p_1, p_2, \cdots p_M$. Then, we use the variant of Newton's method proposed by Chopp [2001] to move these points onto the LSE boundary of O, creating a new explicit sampling of it; notably, this Newton search method requires only gradients. As pointed out by Madan and Levin [2022], their LSE distance is everywhere C^1 except for points on the original polygonal boundaries of objects. Therefore, in order to obtain meaningful gradient information within the LSE distance field to kick off the Newton's method, we shift the sampled points to be ϵ away from the polygonal boundary along the outward tangential direction of the edges where the sample points are initially created. The black dots in Figure 9(a) are one such set of initial guesses for Newton's method. Hence, applying Newton's method, we obtained our first discretization of the LSE boundary of the given object, such as the example provided in Figure 9(b).

This initial discretization yields an LSE boundary polygon with thousands of vertices, where each vertex corresponds to a sample point $p_k, k \in [M]$. However, it is computationally expensive to find the SC exterior map and its inverse with respect to polygons with so many vertices, even if both the SC exterior map and its inverse can be precomputed. Therefore, we adopt an extremely simple scheme to reduce the total number of vertices. Let $\tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_M$ be the points on the LSE boundary and let $\tilde{P}$ be the polygon defined by $\tilde{p}_1, \tilde{p}_2, \cdots, \tilde{p}_M$. We consider all consecutive pairs of edges of $\tilde{P}$, and if their three points are within a small angle threshold of being collinear, then we remove the middle vertex of the three, resulting in the final polygon P. In Figure 9(c), we demonstrate an example of this procedure for a star-shaped object. In general, alternative curvature-dependent remeshing strategies could naturally be substituted provided they still effectively sample the LSE boundary.

In our implementation, we then precompute the SC exterior map (and its inverse) with respect to the final (static) polygon P, reusing it across frames.

C DIFFERENTIABILITY OF ALGORITHM 2 IN $\mathbb{R}^2$

Claim: Our construction of $\boldsymbol{m}(\boldsymbol{x}) : \mathbb{R}^2 \to \partial LSE_{O_i}$ in Section 4.3 is differentiable in both its real and imaginary components.

Proof. As described in Section 4.3, $\boldsymbol{m}(\boldsymbol{x})$ with respect to the proxy polygon consists of three steps. We will prove the statement by showing the real and imaginary components are differentiable in each step.

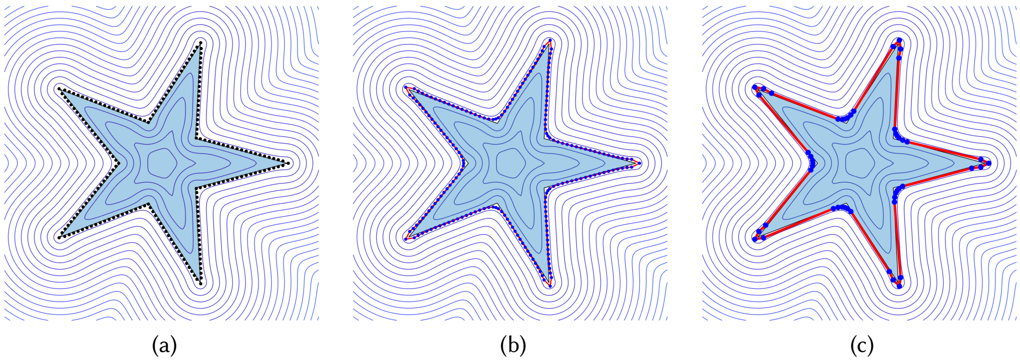


Fig. 9. An example of discretizing the LSE boundary of a given object following the process described above. Isocontours represent the LSE distance field of the star. (a) Black dots are the initial guesses for the Newton's method, equally spaced on the edges of the star, before being pushed away to be ϵ-close to the boundary of the object. (b) Blue dots are points on the zero isocontour found by the Newton's method. The red line segments connecting the blue dots represents the new polygon $\tilde{P}$. (c) Most of the blue dots in (b) are removed, while the remaining ones characterize the LSE boundary of the star. The red segments connecting the remaining blue dots define the final proxy polygon P.

Step 1: The inverse of SC exterior map. In Appendix A, we showed that the SC exterior map is a biholomorphism almost everywhere, except at a finite number of points on the boundary of the polygon in the physical domain and the unit disk in the canonical domain. Then the inverse of the SC exterior map is also almost everywhere complex differentiable. Therefore, both of its real and imaginary components must be differentiable in $\mathbb{R}^2$, since they must satisfy the Cauchy-Riemann Equation.
Step 2: Scaling. After obtaining $w = f^{-1}(z)$ after *Step 1*, we scale w to be ϵ-close to the boundary of the unit disk by setting $w' = (1-\epsilon)\frac{w}{|w|}$. Let $w = a + bi$; we want to show the scaling operation is componentwise differentiable by first writing w' in its component form:

$$w' = (1-\epsilon)\frac{w}{|w|} = \frac{(1-\epsilon)a}{\sqrt{a^2+b^2}} + i\frac{(1-\epsilon)b}{\sqrt{a^2+b^2}} \tag{13}$$

It follows that $\mathfrak{Re}(w') = \frac{(1-\epsilon)a}{\sqrt{a^2+b^2}}$ and $\mathfrak{Im}(w') = \frac{(1-\epsilon)b}{\sqrt{a^2+b^2}}$ are both differentiable everywhere except the origin. However, as discussed in Appendix A, our finite and bounded domain of interest never touches the origin in the canonical domain. Therefore, under our construction, the real and imaginary parts of the second step are differentiable.
Step 3: SC exterior map. The reason is almost the same as that of *Step 1*, since the SC exterior map is almost everywhere holomorphic.