Microfluidics and Nanofluidics (2018) 22:74
https://doi.org/10.1007/s10404-018-2093-x

RESEARCH PAPER

uFlow: software for rational engineering of secondary flows in inertial microfluidic devices

Daniel Stoecklein[1] · Keegan Owsley[1] · Chueh-Yu Wu[1] · Dino Di Carlo[1] · Baskar Ganapathysubramanian[2]

Received: 17 April 2018 / Accepted: 18 June 2018 / Published online: 25 June 2018

Abstract
Approaches to abstract and modularize models of fluid flow in microfluidic devices can enable predictive and rational engineering of microfluidic circuits with rapid designer feedback. The shape of co-flowing streams in the inertial flow regime has become of particular importance for new developments in high throughput microscale manufacturing, biological, and chemical research. In a process known as flow sculpting, the cross-sectional distribution of fluid elements is deformed due to the combined effects of diffusion and transverse advection, which are brought on by interaction with velocity gradients induced by sequences of pillar structures. However, the difficulty in solving the Navier–Stokes equations for complex flow-deforming geometries makes design in this space unintuitive, time-consuming, and costly. To mitigate these issues, we have efficiently embedded flow deformation operations previously relegated to high-performance computing into a free, user-friendly, and cross-platform framework called "uFlow", to bring flow sculpting to the broader community. uFlow computes flow deformation including both advection and diffusion effects from a single pillar in 25 ms on modern consumer hardware, enabling real-time manual design and exploration of microfluidic devices, and fast visualization of 3D particles fabricated via stop flow lithography or optical transient liquid molding. Advanced numerical routines give instant access to a practically infinite set of flow transformations. We showcase uFlow's design models, describe their implementation and usage, and validate the algorithms which allow real-time feedback with confocal imaging and cutting-edge microfluidic particle fabrication.

Keywords Inertial microfluidics · Microparticles · Modeling · 3D visualization · Computer-aided design

1 Introduction

Enabled by electronic design automation software, modularity and abstraction have been critical concepts that fueled a revolution in producing complex electronic circuits to perform information processing tasks. Similar approaches can aid in the design of microfluidic devices, where software can store pre-computed operations of microfluidic components, giving those researchers without access to high-performance computational resources the ability to rapidly explore complex physics. An important aspect of these software packages is the ability to make rapid prediction of the outputs of the entire system when modifications are made to modular components, or to their arrangement within the system. For example, design of monolithic microfluidic devices has been facilitated by hydraulic circuit analysis. However, unlike analogous electronic devices—where one-dimensional electron flow is important—microfluidic engineers may also aim to control the fluid elements within the cross section of a channel for mixing, reaction, or separation applications. Software that abstracts the Navier–Stokes and convection–diffusion equations, focusing only on the input–output transfer functions while tracking the cross-sectional flow structure, could benefit the next generation of microfluidic designers. Only recently has there been effort in this direction for mixing flows and shaping cross-sectional

Electronic supplementary material The online version of this article (https://doi.org/10.1007/s10404-018-2093-x) contains supplementary material, which is available to authorized users.

✉ Baskar Ganapathysubramanian
baskarg@iastate.edu

Dino Di Carlo
dicarlo@ucla.edu

[1] Department of Bioengineering, University of California, Los Angeles, 420 Westwood Plaza, Los Angeles, CA 90095, USA

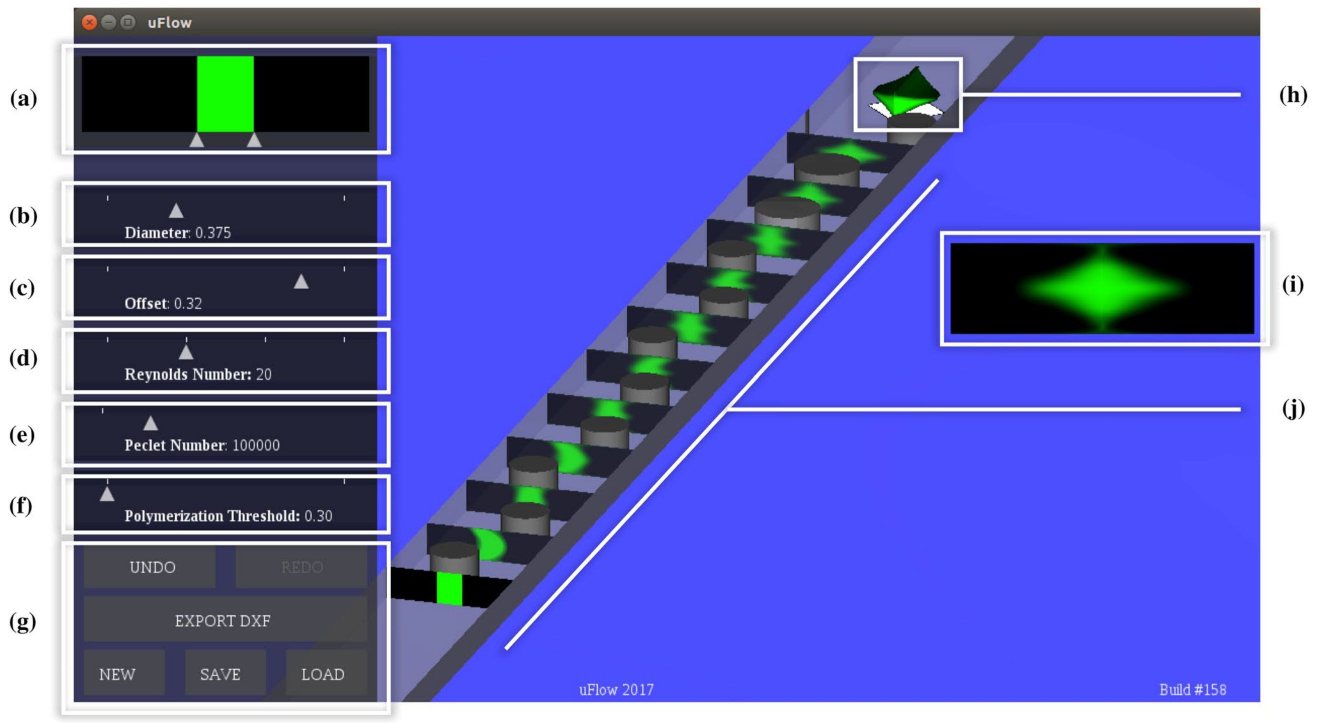
[2] Department of Mechanical Engineering, Iowa State University, 2025 Black Engineering, Ames, IA 50011, USA

flow shape (Mott et al. 2006; Howell et al. 2008; Amini et al. 2013).

Over the last two decades, the microfluidics community has developed multiple approaches for the passive manipulation of a fluid's cross-sectional material composition at the microscale. Liu et al. (2000) used serpentine channels to induce mixing, in which a curving channel was the modular element that led to turning fluid streamlines with finite inertia. Stroock et al. (2002) demonstrated a more straightforward method for mixing by embedding herringbone structures as modular elements in the wall of a microchannel. This approach allowed for mixing from the Stokes flow regime ($Re \ll 1$) to inertial flows ($1 < Re < 100$) with qualitatively similar results. Howell et al. (2004) showed how Dean flow in curved channels can achieve efficient mixing, and Sudarsan and Ugaz (2006) enhanced this method with sudden channel expansions. For more precise control over the fluid, Howell et al. (2008) extended the idea of patterned microchannel surfaces with the use of grooves for hydrodynamic focusing. However, each of these methods typically employs a small set of flow operations that leverage chaotic advective forces, limiting their potential for arbitrary user-designed flow shapes. Amini et al. (2013) introduced the idea of using sequenced microstructures for hierarchically designed fluid flow deformation and cross-sectional shaping in the inertial regime. This technique for flow manipulation, dubbed "flow sculpting", has dramatically enhanced the level of control possible over the form of microfluid streams.

Flow sculpting uses a sequence of bluff body microstructures, each acting as independent operators, to deform the cross-sectional structure of inertially flowing fluid. To maintain independence between the flow deforming operators, there are necessary restrictions on the design of a flow sculpting device. The primary restriction is that the microstructures—cylindrical pillars are commonly used, and assumed here—must be spaced far enough apart in the microchannel to allow for an individual pillar's flow deformation, i.e., advection map (Mott et al. 2006; Stoecklein et al. 2014) (see Fig. 2), to saturate before the fluid arrives at the subsequent pillar. The flow deformation should also have no time-dependent effects. These two features prevent cross-talk between pillars, and enable an elegant computational shortcut for simulating flow sculpting devices: the fluid flow deformation for any structure can be independently simulated ahead of time, and since we are only concerned with the transformation of flow after passing the pillar, advection maps of the cross-sectional fluid flow shape can be sequentially composed for *real-time* simulation of fluid flow deformation. This is especially useful for the inertial flow regime, as accurate computational fluid dynamics (CFD) simulations that include convective terms to account for inertia are computationally expensive to solve, making their use in design costly, and beyond the reach of researchers who are without access to high-performance computing (HPC) resources. On the other hand, rapid use of pre-computed results from complex CFD simulations makes exploratory investigations more feasible, and previously difficult design problems tractable. In 2014, we implemented this approach and showed how one could program the structure of inertial fluid streams within a user-friendly graphical interface (Stoecklein et al. 2014). This framework's fast advection model has been experimentally validated over a wide variety of sculpted fluid flow shapes (Stoecklein et al. 2014), and has enabled researchers to use flow sculpting for novel applications in advanced manufacturing and biological sciences: Nunes et al. (2014) fabricated shaped microfibers with tailored cross sections by sculpting a UV-polymer pre-cursor, and Sollier et al. (2015) designed a solution transfer scheme to shift fluid away from larger particles in flow, thus introducing a high-throughput approach to purify samples, or collect reusable reagents.

We have been working on this framework, dubbed "uFlow" since its debut, and report here several significant advances as part of a high-functionality release of the software, including a diffusion model and 3D polymer particle shape prediction, that we think will have a broad impact on the microfluidics community. A number of features have been incorporated into uFlow to bring the user from simple point-and-click assembly of flow sculpting devices, through post-processing their design with high-resolution images and computer-aided design (CAD) models, to advanced modeling of mass diffusion and particle fabrication via UV polymerization. The inclusion of a transverse mass diffusion model can be crucial for design, as diffusive blurring is problematic with the longer channels that come with more complex flow shapes (increasing fluid time-of-flight), thereby limiting the practically useful number of pillars in a flow sculpting device, and making advection-only models impractical. Mass diffusion can also be leveraged to decouple fluid streams connected by a thin band of fluid, or purposely erode small-scale features of a sculpted shape. Software-aided particle design dovetails with recent developments in particle fabrication, e.g., Paulsen et al. (2015) using stop flow lithography to fabricate shaped millimeter-scale 3D particles, following up with microscale 3D particles (Paulsen et al. 2018), and Wu et al. (2015) building a rapid manufacturing approach for 3D microscale particles. These processes can be used to create functionalized particles for cell capture and diagnostics (Bong et al. 2015; Chen et al. 2016) or accelerated wound healing (Do et al. 2015; Griffin et al. 2015), encoded hydrogel particles (Lee et al. 2014; Appleyard et al. 2011), photonic crystals (Galisteo-López et al. 2011), and self-assembling hydrogels (Gurkan et al. 2012), for example. However, current routes for particle design require full-scale Navier–Stokes/convection–diffusion CFD simulations for whole devices (Paulsen et al. 2015, 2018), whereas uFlow



Diameter: 0.375
Offset: 0.32
Reynolds Number: 20
Peclet Number: 100000
Polymerization Threshold: 0.30

Fig. 1 uFlow graphical user interface (GUI): **a** inlet flow pattern design of the channel cross section indicating different regions of the flow; **b**, **c** pillar diameter and location for the next pillar, or for altering a selected pillar; **d** Reynolds number (flow physics) for the flow sculpting device; **e** Peclét number (diffusion) for the flow sculpting device; **f** polymerization threshold for microparticle rendering; **g** undo/redo, device saving/loading/export; **h** microparticle rendering area; **i** net flow deformation visualization (can be selected to save a high-resolution image); **j** pillar sequence area (for placement/adjustment of pillars). Note that for ease of use, the pillars are shown much closer together than they should be in a fabricated device

gives a comparable result in a matter of seconds, with no CFD expertise required.

Here for the first time we describe the uFlow framework in complete detail, along with the implementation and usage of the aforementioned features. Our goal is to aid other researchers who wish to adopt the framework to address other microfluidic design problems of interest, or add functionality themselves. We anticipate that uFlow will assist the microfluidics community for design, enable rapid visual feedback for fluid mechanics education, and to help accelerate the maturation and adoption of microfluidic technologies.

2 uFlow overview

The three primary physics models available in the uFlow library (dubbed “uFlowlib”) are those of fluid advection, transverse mass diffusion, and microparticle fabrication visualization, all of which are accessible through its graphical user interface (GUI) (see Fig. 1). uFlow also stores several thousand advection maps for instant recall via principal component analysis (PCA), with interpolation giving access to a practically infinite set of flow deformations. uFlowlib leverages the graphical processing unit (GPU) available on most standard laptops/desktops to accelerate the computing to make real-time exploration possible.

We adapt the computation of flow sculpting to GPUs in a platform-agnostic manner using the open graphics library (OpenGL), which is a cross-platform programming interface for GPUs that is typically utilized in 2D/3D rendering for visualization. uFlow itself is written in the Python language, while OpenGL routines are programmed in the Graphics Library Shader Language (GLSL), and sent to the GPU through the Python module “Pyglet”.[1] Most of uFlow’s library makes use of *shaders*, which are small programs that run in parallel on the GPU and can use data stored in *textures* which reside in GPU memory (analogous to arrays in system memory). By aligning flow sculpting simulations with computational routines endemic to the GPU, we have created an extremely fast framework for exploration and design. uFlow is freely available for Windows and Linux platforms,[2] and

[1] https://www.pyglet.org.

[2] www.biomicrofluidics.com/software.php.

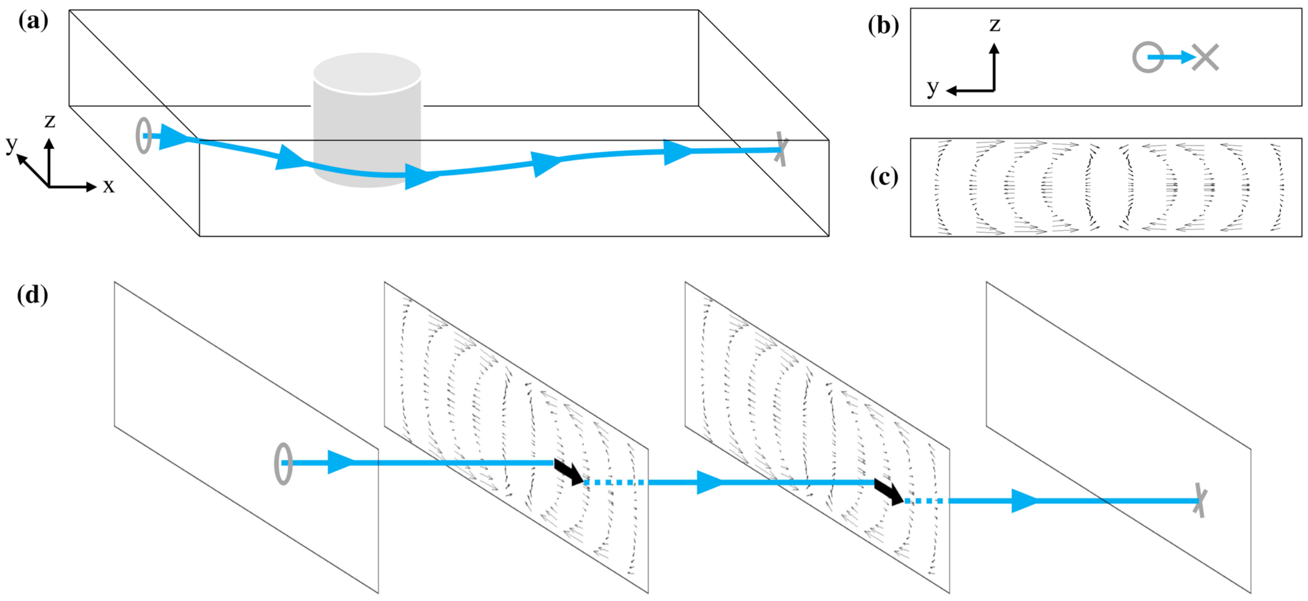


Fig. 2 Illustration of the creation of advection maps and their use in flow sculpting. (**a**) Fluid flow is simulated within a 3D domain for a single pillar and a given fluid flow condition (Reynolds number *Re*), producing a 3D velocity field. Infinitesimal, massless particles are streamtraced through the velocity field, determining each fluid element's (particle) lateral displacement (**b**) in the microchannel cross-section. This is performed for a lattice of particles across the entire cross-sectional area, with their collective displacements informing a single advection map (**c**) for that pillar's geometry and the given flow physics. The advection map can then be used in place of the 3D velocity field for the simulation of a flow sculpting device, where (**d**) transverse fluid displacement is computed for a single fluid element by sampling the advection map at the fluid's current location in the channel cross section. This process can be repeated for an arbitrary number of pillars or other microchannel modules, provided that each 3D simulation has matching boundary geometry and flow physics

its source code is also freely accessible.[3] In the next two sections, we detail physics models and utilities of uFlowlib, along with their implementation and usage in the software.

3 uFlow physics models

3.1 Advection

The most basic operation of uFlow is to simulate the convection-driven net fluid flow deformation induced by a user-designed pillar sequence. Previous work of Amini et al. (2013) and Stoecklein et al. (2014) explored the nature of steady-state inertial flow past cylindrical bluff-bodies (pillars) in microfluidic channels. The overall lateral displacement due to net inertial convection yields a so-called "advection map" for a single pillar and fluid flow condition, *Re* (we define the Reynolds number as $Re = \frac{UD_H}{\nu}$, with fluid kinematic viscosity ν and average downstream velocity U, and microchannel hydraulic diameter D_H). An advection map is a 2D vector field describing the lateral displacement of fluid in the microchannel cross section, acting as a computational shortcut for simulating the full CFD flow deformation for a single pillar. By collapsing the streamtrace through a 3D velocity field to a 2D map, the net flow deformation from a sequence of pillars in a flow sculpting device is well simulated in an extremely rapid manner by simply traversing from map to map (see Fig. 2). In fact, the flow transformation induced by any arbitrary microchannel geometry (e.g., grooves, curved channels, ridged channels) can be represented as an advection map and added to a library of pre-computed deformations, any of which are readily placed into a flow sculpting device simulation. We ensure seamless utility by enforcing a set of requirements on the library of advection maps. First, the channel geometry and flow physics must match at the inlet and outlet of each microfluidic "module". One cannot append an advection map for which $Re = 40$ to a device which uses maps computed for $Re = 20$, nor mix-and-match microchannel height-to-width aspect ratios, without a connecting channel that matches the boundary conditions at the outlet to the preceding module, and at the inlet of the following module. Second, the 3D streamtrace must completely capture the flow-induced deformation within the 3D domain, i.e., transverse flow displacement should saturate before fluid exits the domain. Finally, the flow should not exhibit any time-dependent effects.

uFlow uses custom full-color RGBA 32-bit floating point textures to store advection map data, which describe flow displacement on the y–z plane (see Fig. 2a) for a discretized

[3] www.bitbucket.com/baskargroup/uflow.

grid of $N_Y \times N_Z$ fluid element locations. The red and green channels store $(\mathrm{d}y, \mathrm{d}z)$ displacement data at their respective locations in the microchannel cross section. Because each displacement is computed independently in the 3D streamtrace, displacing fluid in uFlow (now represented as pixels) is an embarrassingly parallel operation, and well suited to execute on the GPU. To simulate flow deformation from a microfluidic module (e.g., a pillar), uFlow uses the process defined in Appendix 1.

Using uFlow through its graphical user interface (GUI) is a simple game-like process (see Fig. 1). Once the program is started, most interaction is through the mouse, which can select a pillar diameter and Reynolds number from the left-pane of uFlow (Fig. 1b, d), and place the pillar within the rendered microchannel itself in a selected location (Fig. 1j). A view of the flow device's outlet cross section is then shown in color in the upper-right portion of the window. More complex deformations can be created by placing additional pillars as desired, although several practical limitations—beyond GPU memory to store deformations—should be considered in creating a flow sculpting device: first, transverse mass diffusion will blur deformed shapes, the effects of which become exacerbated with longer channels associated with longer sequences of pillars (diffusion is discussed in greater detail in the following section). Second, an increased number of obstacles and a lengthened microchannel will require higher pressure at the inlet to drive flow. High pressure in a microfluidic device can be impractical to implement, and introduce flexure (or damage) to the walls of the device itself, especially if it has been fabricated from softer polymers such as polydimethylsiloxane (PDMS) (Sollier et al. 2011). Finally, it is desirable to minimize flow sculpting's footprint in a lab-on-a-chip device. In addition to the above concerns, a smaller device with fewer flow sculpting parts will allow more room for additional microfluidic components (e.g., polymerization, sensors, or visual interrogation), and have fewer points of failure during fabrication.

The flow streams and pillar properties are dynamically adjustable in uFlow with real-time feedback to the computed flow shape. The inlet flow stream widths (Fig. 1a) can be changed dynamically by clicking and dragging on the small triangles at the lower edge of a stream, a menu to change a stream's color can be opened by right-clicking a stream, and new flow streams can be added by double-clicking within the inlet flow pattern design. Because the fluid particles are colored in each rendered frame, adjustments to the inlet flow pattern are immediately reflected in real time in the deformed flow shape. Similarly, as uFlow stores each microfluidic part's contribution to the net flow deformation in memory, changes made to previously placed parts (e.g., resizing a previous pillar's diameter, or changing its location in the channel) will instantly update the deformed flow shape. The 3D view of the flow sculpting device can be rotated (holding in the right-mouse button and moving the mouse) moved along the channel (holding the left-mouse button and dragging the mouse), or zoomed in/out (using the scroll wheel on the mouse). An image of the deformed fluid is shown after each individual pillar, allowing the user to see the change in the fluid flow shape as each pillar sculpts it. This can be informative for iterative design, and help provide insight into the chaotic effects of changes early in the sequence. Notably for applications in material fabrication, different regions of sculpted flow can be functionalized with designed properties by doping inlet flow streams with functional materials, such as biotin or superparamagnetic particles (Wu et al. 2015).

The net deformation can be output as a high-resolution portable network graphics (PNG) format image, and the device design itself can be saved in two different forms: uFlow's custom .dev format, which can be loaded into other instances of uFlow to recreate the design; and a CAD-like .dxf format (see Fig. 1g). The .dxf format, which can be opened by many CAD programs, is intended to aid in device fabrication, and contains a top-down view of the entire flow sculpting device with pillars correctly spaced to avoid cross-talk.

3.2 Diffusion

The effects of mass diffusion have generally been neglected in previous flow sculpting work, as it was assumed that high Peclét numbers ($Pe = \frac{L_c U_c}{D}$, for characteristic length L_c and velocity U_c, and diffusivity D, showing the relative dominance of the rate of advection vs. diffusion in mass transport) would be employed in their use cases. However, the practical effects of diffusion can still play a significant role in flow sculpting even at relatively high Pe. For example, work done by Stoecklein et al. (2014) relied upon diffusion to produce several flow transformations—bridging small gaps to create closed shapes, or dissipating connecting threads to separate streams. Mass diffusion also affects the minimum feature size in sculpted flow shapes, thereby playing a significant role in flow lithography approaches that utilize flow sculpting. Hence, we aimed to include the effects of diffusion in uFlow's toolkit in a manner that was compatible with the modular assembly of microfluidic components.

In principle, the flow physics of a fluid parcel in inertial flow is determined by the advection–diffusion equation. We track the Lagrangian motion (i.e., pure advection) of the fluid parcel using the previously described advection maps. Using a frame of reference that is moving with the particle, we can consider how diffusion occurs in the direction normal to the streamline direction. Diffusion in the cross-sectional y–z plane is modeled using Fick's law:

$$\frac{\partial c}{\partial t} = D\nabla^2 c. \quad (1)$$

We utilize the analytical solution for 2D isotropic diffusion from a point source, computing the concentration $c(t)$ from an initial mass M at point ξ, η:

$$c(\xi, \eta, t) = M\left(\frac{1}{4\pi Dt}\right)\mathrm{e}^{\frac{-(\xi^2+\eta^2)}{4Dt}}, \tag{2}$$

where ξ, η are local coordinates in the y–z plane (with $(\xi = 0, \eta = 0)$ representing the center of the point source (pixel) being diffused) and t is the time of diffusion. Use of this solution is reasonable since we consider pixel-width streamlines.

uFlow operates in a non-dimensional domain to allow scaling to any microfluidic device which exhibits the scaling parameters of *Pe* and *Re*. *Pe* is defined using the microchannel width w for the characteristic length L_c, while the flow Reynolds number *Re* uses the characteristic velocity U_c as $Re = \frac{U_c L_{Dh}}{\nu}$, where L_{Dh} is the microchannel hydraulic diameter. Equation (2) is made non-dimensional for use in uFlow by substituting the Peclét number *Pe* for D, non-dimensional time $t^* = \frac{tU_c}{L_c}$, non-dimensional distances $\xi^* = \frac{\xi}{L_c}$ and $\eta^* = \frac{\eta}{L_c}$, and multiplying each side by $\frac{L_c^2}{M}$:

$$\frac{L_c^2}{M} c(\xi^*, \eta^*, t^*) = \left(\frac{Pe}{4\pi t^*}\right)\mathrm{e}^{\frac{-Pe\left(\xi^{*2}+\eta^{*2}\right)}{4t^*}} \tag{3}$$

Note that each pixel in the cross section will suffer a different amount of diffusion, as the transit time of fluid at various points on the cross section is different. Fluid in the center of the channel will experience low time-of-flight due to higher velocity, and, therefore, low diffusion, while fluid closer to the boundaries have much larger time-of-flight (due to the slower velocities as a result of the no-slip boundary condition), leading to greater diffusion. We, therefore, create an effective diffusivity $D_\tau(y^*, z^*) = D\tau(y^*, z^*)$, where $\tau(y^*, z^*)$ is a scalar field of a fluid particle's non-dimensional time-of-flight $\tau(y^*, z^*) = \frac{L^*}{u^*(y^*, z^*)}$, for non-dimensional channel segment length L^*, fluid velocity $u^* = \frac{u}{U_c}$, and coordinates $(y^* = \frac{y}{L_c}, z^* = \frac{z}{L_c})$ spanning the microchannel cross section. This can be incorporated into a spatially dependent Peclét number, $Pe(y^*, z^*) = \frac{U_c L_c}{D_\tau(y^*, z^*)}$, which can be used in Eq. (3) with a time step $t^* = 1$ to estimate diffusive blur resulting from fluid travel through a single channel segment for a pixel at (y^*, z^*). That is, uFlow diffuses the mass at a point (pixel) at (y^*, z^*) with a gaussian distribution in local pixel-centered coordinates (ξ^*, η^*), with a diffusive length proportional to $\frac{1}{Pe(y^*, z^*)}$.

Although a unique time-of-flight as a function of cross section position and pillar component could be utilized in this scheme, we approximate D_τ by assuming that the bulk of a fluid element's time-of-flight will be spent in the open channel, and calculate $\tau(y^*, z^*)$ from an analytical solution for the velocity field $u^*(y^*, z^*)$ considering laminar flow in a rectangular duct (Spiga and Morino 1994). A non-dimensional time-of-flight is calculated based on a series sum for velocity from Spiga and Morino (1994), and the known inter-pillar spacing (Amini et al. 2013) ($L^* = 10\frac{w}{L_c}$), yielding the scalar field $\tau(y^*, z^*)$. The resulting diffusive blur is applied after each pillar deforms the cross-sectional flow shape, with the diffused result becoming the input for the next pillar. This is trivially incorporated into the uFlow framework.

We consider the diffused mass M as the RGB color profile at each pixel, and embed the previously defined gaussian function within the advection-coloring fragment shader. To diffuse a single point, a fragment shader uses Eq. (3) to sample a 7×7 neighborhood of pixels (the span of which is defined by the diffusive length $\frac{1}{Pe}$), and calculates how each neighbor will affect the single point under consideration. The point is then colored by summing each neighbor's diffused contribution, along with its own diffused color, advected from the inlet streams. This sum is normalized by the number of points sampled, allowing this approach to work near the boundaries where fewer pixels are sampled. The diffusion operation automatically updates with each frame render, so adjustments to the Peclét number in the GUI (see Fig. 1e) will change the visualized diffusive blur in real-time. If a non-diffusive image is desired (advection only), the device's Peclét number can be increased to the point where diffusive effects are no longer present. Practically, we see this for $Pe \geq 10^7$.

Despite some simplification of the physics, this diffusion model replicates experimental behavior well. Fig. 3 shows how the time-of-flight field $\tau(y^*, z^*)$, based on the velocity field for laminar duct flow, diffuses a co-flow of streams in a straight channel in the expected "hourglass" shape (Ismagilov et al. 2000). In Fig. 4, uFlow's diffusion model is compared to confocal images taken from flow sculpting devices with $Pe = 10^5$ [fabricated and imaged using the techniques outlined by Amini et al. (2013)], alongside non-diffusive simulations ($Pe = 10^7$) to demonstrate the difference. Note that near the boundaries—where time-of-flight goes to infinity as the fluid velocity goes to zero—the large effective diffusive length will significantly blur the sculpted fluid, diluting the fluid mass. This is an effect uFlow captures, as shown by comparison to experimental results in Fig. 4. While showing qualitative accuracy, there is further scope for improvement of incorporating the effect of diffusion. Future work will include per-pillar time-of-flight information, as advection maps use only two of the four available channels in an RGBA texture to store a $(\mathrm{d}y, \mathrm{d}z)$ vector field.

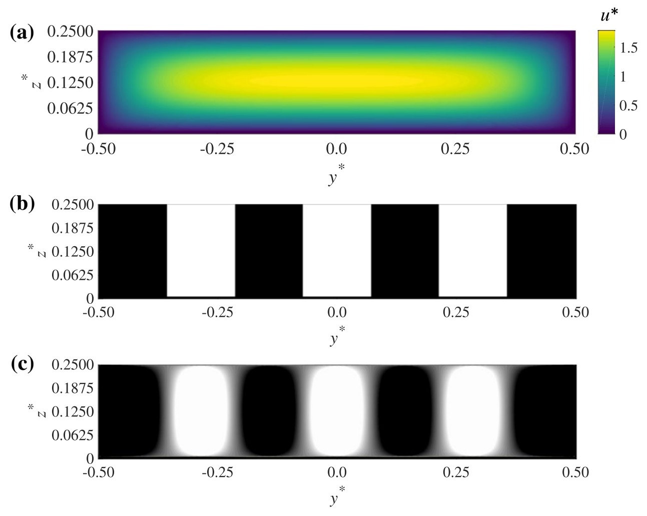


Fig. 3 Diffusion in uFlow. **a** A parabolic velocity profile u^* for a rectangular channel is calculated using Spiga and Morino's analytical solution (Spiga and Morino 1994), which is used with a fixed channel length to calculate a non-dimensional fluid flow time-of-flight τ. The scalar field $\tau(y^*, z^*)$ is then used to create a spatially dependent diffusivity coefficient $D_\tau(y^*, z^*)$, which is in the fundamental solution to Fick's law to blur the fluid flow. This is demonstrated for a non-advected inlet flow pattern (**b**), which is diffused for two equivalent pillar lengths (**c**) with $Pe = 10^5$

3.3 Microparticle fabrication visualization

Flow sculpting has been used with stop flow lithography (Paulsen et al. 2015, 2018) and optical transient liquid molding (Wu et al. 2015) to fabricate shaped millimeter- and micrometer-scale particles. The basic idea is to shape an inlet stream containing a crosslinkable polymer precursor, and polymerize the shaped stream using a UV light source with a defined mask shape once the flow is stopped (see Fig. 5a). This creates a 3D polymer microparticle with a flow-sculpted shape in one orientation, and a mask-defined shape in an orientation normal to that. The cross-sectional flow shape is limited by what is possible through inertial flow sculpting. The mask used to define the UV light—typically a standard photolithography mask—has high precision for smaller features, allowing for essentially any 2D design to form the shape of the microparticle in the UV-light direction. Ultimately, both shapes will be affected by an extended diffusive process during the stopped-flow portion of fabrication (≈ 1s), which should be anticipated during the design of both shapes and minimized or accounted for in software as much as possible for a more faithful recreation of the target 3D geometry (Wu et al. 2015).

uFlow uses an indirect approach for rendering the expected particle shape, without computing the actual particle geometry as an intermediate step. The rendering algorithm uses a *signed distance function* (SDF) representation of the particle in combination with a *ray marching* scheme (Hart 1996), discussed in detail as Supplementary Fig. S1. The result of this algorithm is a fast renderer that produces an image of the predicted particle shape.

To visualize the fabricated microparticle, uFlow needs a sculpted flow shape and a shape for the orthogonal optical mask that selectively exposes the shaped flow stream to UV light. By default, uFlow selects the central fluid stream as

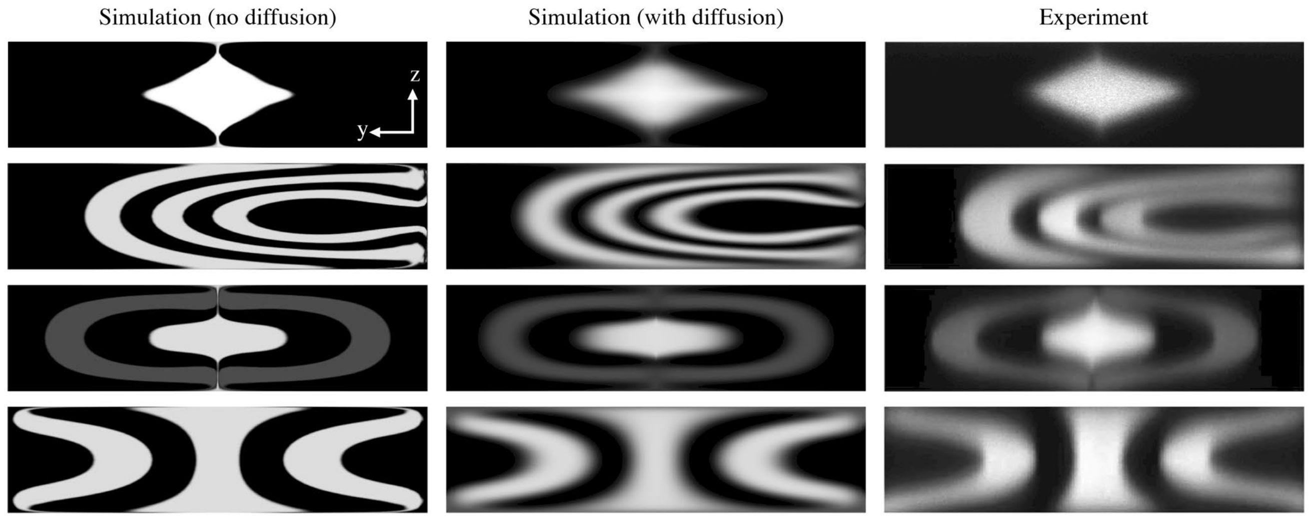


Fig. 4 Examples of uFlow's simulated diffusion compared to confocal images from experiments. Shown are simulations with no effective diffusion ($Pe = 10^7$, left) and moderate diffusion ($Re = 20$, $Pe = 10^5$, middle), with confocal images from flow sculpting devices with matching flow conditions ($Re = 20$, $Pe = 10^5$, right). These results show how uFlow's diffusion model provides more realistic grounding in design. Note the loss of small features in the sculpted flow shapes, e.g., in the second and 4th rows near the right wall. Additionally, the well-defined gaps in the alternating bands of fluid in the third row become quite diffused, which is a loss of a distinct feature that uFlow predicts

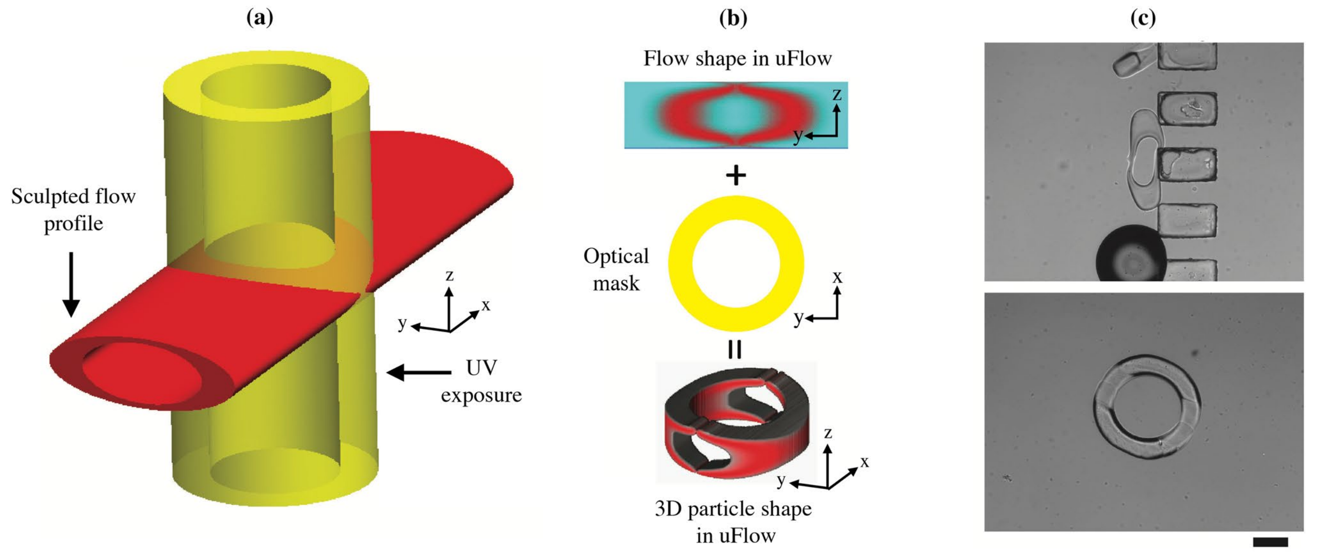


Fig. 5 Microparticle fabrication via optical transient liquid molding (OTLM) and visualization in uFlow. **a** Flow sculpting can be used with OTLM to fabricate tailored 3D microparticles. A pillar sequence shapes a UV-crosslinked polymer precursor into some user-designed shape in the flow direction (red), while a mask is placed over the channel in the post-pillar sequence region (yellow). When the flow stops, a UV light shines on the mask, polymerizing the sculpted flow to have the mask's shape in the UV light orientation, and the flow sculpted shape in the flow direction. **b** 3D particle estimation of a porous particle in uFlow, with a flow shape, optical mask (defined in mask.png), and the resulting 3D particle visualization. **c** OTLM fabrication of the porous particle designed in **b**, shown with its flow direction face (top image) and mask-defined face (bottom). uFlow's prediction and the experiment show good agreement. The scale bar is 100 µm. (Color figure online)

the polymer pre-cursor, while sheath flow surrounding the central stream is the same monomer but without the photoinitiator (making viscosity uniform throughout the channel). Different streams can be chosen to be polymerized by altering a sequence of booleans in the `polymerized_streams` variable within the main script, `uflow.py`. The user-modifiable mask shape is a 256 × 256 black-and-white image,"mask.png", placed in uFlow's top-level directory (a rectangular UV-mask is visible in the floor of the visualized microchannel in Fig. 1h). We demonstrated this method of design using uFlow's flow sculpting operations with a donut-shaped mask to design a 3D particle with two pores, shown in Fig. 5b. Optical transient liquid molding (OTLM) (Wu et al. 2015) was used to fabricate the resulting shape, which is shown Fig. 5c.

While purely advective flow would retain all photoinitiator within the sculpted polymer precursor shape, diffusion—naturally present to some degree in most flows—will bring some amount of photoinitator into neighboring regions. To take the effects of diffusion on the fabricated particle into account, uFlow has a user-adjustable "polymerization threshold" value which determines the UV-curable volume fraction required to achieve solidification (see Fig. 1(f)). This threshold is also affected by the energy in the UV light, which will depend on intensity and exposure time [this is discussed in more detail in Ref. Wu et al. (2015)]. By adjusting both the Peclét number and polymerization threshold, one changes the degree to which the fluid shape expands and neighboring fluids diffuse into the final polymerized particle shape.

4 Advanced compression and interpolation routines

The utility of uFlow is predicated on quick access to advection maps derived from full-scale solutions to the Navier–Stokes equations. However, the intent of its users, whether to create a particular flow shape, direct fluid to a particular region of the microchannel cross section, or some other discipline-specific goal, cannot be fully anticipated when preparing a useful set of advection maps. We, therefore, seek to provide as large of a design space as possible.

uFlow allows access to a continuous set (i.e., infinite set) of pillar diameters and locations by compressing a discrete dataset of pre-computed advection maps using principal component analysis (PCA). PCA uses a set of n high-dimensional observations $\left(O_k\right)_{k=1}^{n}$, each of dimension m, to create a new *basis* of principal components $\left(PC_k\right)_{k=1}^{n}$ in the high-dimensional space which maximally explains the variance among all observations in the dataset. Once the high-dimensional basis has been constructed, each of the dataset's original observations can be recreated as $\hat{O}_i$ from a linear combination of the new basis vectors PC_j (where each vector PC_j is of size m):

$$\hat{O}_i = \sum_{j=1}^{n} \alpha_{i,j} \mathrm{PC}_j, \tag{4}$$

where the coefficient $\alpha_{i,j}$ is an observation-specific low-dimensional mapping, indicating how the observation O_i aligns with the direction of the principal component PC_j. This mapping is generated by simply projecting an observation onto the PCA basis.

PCA orders the principal components by the variance explained in the original dataset, i.e., PC_1 explains the most variance of all principal components, PC_2 the second-most, and so on. Therefore, if the data efficiently map to a linear basis, a small subset of all n available principal components can be used to reconstruct any of the original data to a nominal degree of accuracy. This efficiency is determined by calculating the cumulative proportion of variance explained by the chosen subset of principal components and comparing it to the total variance in the dataset. This produces an effective cumulative "energy content", E, which, for a principal component's variance-explained λ_j is found for a reduced number of n_R principal components by:

$$E(n_R) = \frac{1}{E_T} \sum_{j=1}^{n_R} \lambda_j, \tag{5}$$

where E_T, the total cumulative energy content of the dataset, is found by:

$$E_T = \sum_{j=1}^{n} \lambda_j. \tag{6}$$

A dataset is well represented using the basis formed by n_R principal components as $E(n_R) \to 1$. A tradeoff can then be made between complete reconstruction of the original data (which requires all n principal components, in which case one may as well use the original data), and using a subset of the first $n_R \ll n$ principal components to reconstruct data to an acceptable degree. This is a powerful form of adaptive data compression (Andrews et al. 1967), as one need only store n_R principal components and the original data's projected mapping in the low-dimensional space (m arrays of n_R floating point values).

For uFlow, the high-dimensional dataset being compressed is a collection of 3,224 advection maps, calculated for $Re = \{10, 20, 30, 40\}$ from 31 different normalized pillar diameters of $d/w = [0.2 : 0.02 : 0.8]$ and 26 normalized pillar locations $y/w = [0.0 : 0.02 : 0.5]$ for channel width w. These maps were created using an in-house streamtracing code (there are many freely available visualization software packages that contain effective streamtracing tools, such as Paraview), tracing through 3D velocity fields created on HPC resources using an experimentally validated Finite Element Method framework (Diaz-Montes et al. 2014). Such a large dataset may not be easily stored and transferred across computers, or easily recalled for computation on the GPU, but its fine sampling is well-suited to PCA compression and interpolation. We found that splitting the dataset's dy, dz displacement data into separate datasets made for more efficient PCA compression, likely due to differences in variance between vertical and horizontal velocities being difficult to explain with the same linear basis. In using a reduced number of $n_\mathrm{R} = 300$ principal components (from a possible $n = 3224$), we are able to reconstruct any of the original observations with a cumulative energy content of $E(n_R) = 99.89\%$. Therefore, uFlow need only store the first 300 principal components (as 2D RGBA textures) and each observation's low-dimensional mapping to effectively recreate any of the original 3,224 advection maps, with data compressed to $\approx 9.3\%$ of its original size.

Beyond compressing the original data, PCA enables interpolation within the low-dimensional mapping to create advection maps not contained in the original dataset (given a smoothly varying low dimensional embedding). This is achieved by linearly interpolating a desired pillar's low dimensional mapping from its nearest neighbors within the original dataset, and using the new mapping with Eq. (4) and the dataset's principal components to construct a new advection map.

We tested PCA interpolation by reconstructing non-original advection maps across the $Re = 20$ domain, and comparing them to natively-streamtraced advection maps using the L_2-norm: error $= \|\hat{O} - O\|_2 / \|O\|_2$. We found that over 98% of the reconstructed data had less than 5% relative error, with a mean reconstruction relative error of 1.76%, and a maximum relative error of 10% (see Supplemental Fig. S2). However, most of the error does not contribute to a significant difference in the flow prediction (see Fig. 6), and the worst-case error can be explained by low-magnitude displacements making the relative error larger (see Supplemental Fig. S3). With such low error overall, we are confident that PCA interpolation can be used in uFlow to access a practically infinite set of pillar configurations, provided the number of pillars in a sequence stays within the limits of practical fabrication (i.e., fewer than 30 pillars) to avoid significant error propagation. PCA interpolation in uFlow is currently limited to pillar geometry and location only, as the sampling of Reynolds numbers is likely too coarse for accurate interpolation.

The PCA library is implemented in uFlow as the `OGLPCAFactory` class, which reads from a stored set of 300 principal components and the original dataset's dy, dz

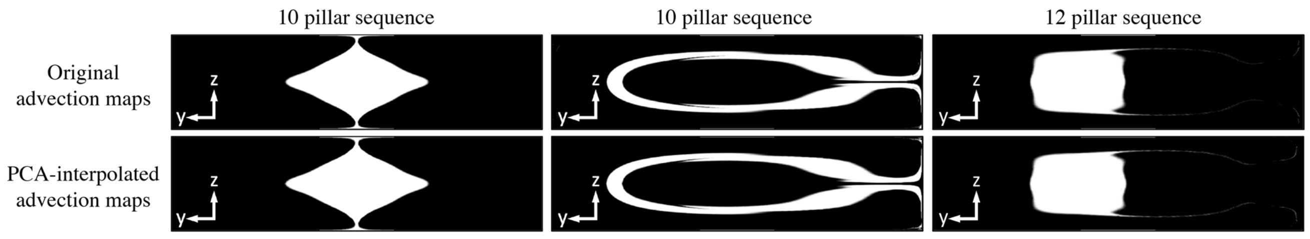


Fig. 6 First row: flow shapes generated with original, natively streamtraced advection maps using 10- to 12-pillar sequences. Second row: flow shapes generated from PCA-interpolated advection maps using the same designs in the first row. For the flow shapes in the second and third columns, none of the pillars were among the observations used to construct the PCA basis (e.g., $d/w = 0.75$, $y/w = 0.125$), and required interpolation within the low-dimensional PCA space

low-dimensional mappings (stored as text files). The linear reconstruction in Eq. (4) is encoded into a set of fragment shaders to reconstruct a requested advection map on-the-fly on the GPU, with no discernible latency compared to recalling a stored advection map. On a personal computer with an AMD Ryzen 1700 CPU at 3.0 GHz, an NVIDIA GTX 1060 GPU, and 16 GB of memory running Ubuntu 16.04, the entire process of recalling an advection map via PCA, advecting and diffusing fluid, and estimating the 3D microparticle, is completed in approximately 25 ms.

5 Conclusions

uFlow gives users with modest computing hardware access to a practically infinite set of high-performance computational fluid flow simulations, capable of real-time design and exploration, and complete with estimations of diffusion effects and visualization of fabricated microparticles using various masks. uFlow also enables users to incorporate new microfluidic components to sculpt flow of their own design, and shows how they can be efficiently stored for quick access on the GPU.

With the addition of diffusion modeling and an expansive and extensible library of pillar configurations, major shortcomings have been ameliorated. The primary modes of uFlow's functionality—flow sculpting and visualization—are accessible through the GUI, while the new advection map storage is found in the publicly available uFlow code. By including microparticle fabrication visualization, uFlow is now reaching into a set of post-flow-sculpted operations and analyses which can have immediate impact on microparticle design and optimization. Our goal is to enable the broader microfluidics community—be they researching diagnostic technology, advanced manufacturing, or other new applications of flow sculpting—to easily integrate

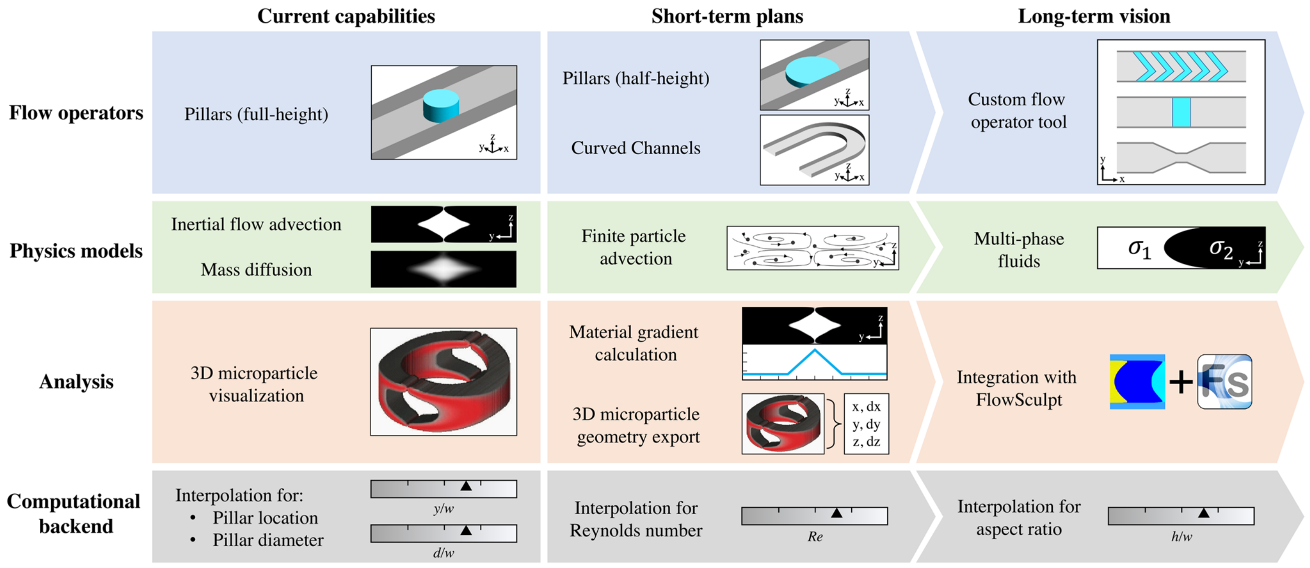


Fig. 7 An overview of the current capabilities of uFlow, with short-term and long-term visions for new features and improvements for uFlow's flow operators, physics models, analysis, and computational backend

complex flow physics into their research and technology designs.

In Fig. 7, we summarize uFlow's current capabilities and outline future plans, with uFlow's features categorized as flow operators, physics models, tools for analysis, and improvements to the computational backend (e.g., the PCA advection map library introduced in this work). Future directions for uFlow's flow operators include curved channels (introducing Dean flow) and top-down asymmetric full-channel flow sculpting using pillars of variable height, herringbone structures, or steps in the channel. Eventually, we envision a toolkit enabling users to easily pipeline their own customized flow operators into uFlow. We also plan to integrate predictions of finite-sized particle advection into uFlow, and in the far future, multi-phase fluid transport via flow sculpting. In terms of analysis and post-processing, calculations of concentration gradients and voxelized particle geometries are logical next steps, and long-term plans are to integrate uFlow with the automated design software FlowSculpt (Stoecklein et al. 2016). Finally, we plan to continue building upon uFlow's computational backend by implementing a continuous flow Reynolds number selection, and introducing new channel aspect ratios.

Acknowledgements This research is supported in part by the National Science Foundation (NSF) through NSF-1306866 and NSF-1149365.

Appendix 1

The following steps describe the advection operation that is programmed into a fragment shader for per-pixel execution on the GPU:

1. A set of fluid "particles" are distributed uniformly across the 2D domain (microchannel cross section) as a set of `GL_POINTS` (an OpenGL primitive, describing a single point in 1D, 2D, or 3D space). The points will collectively describe the fluid as it is transformed via flow sculpting.
2. All fluid particles are colored and rendered as pixels in parallel by a *fragment* shader, coloring each point based on which inlet stream they are located in at the microchannel inlet.
3. After a pillar—or some microfluidic part—is placed in the microchannel, a *vertex* shader concurrently samples the advection map for that part (stored in the GPU as a 32-bit 2D texture) on a uniform grid, computes the displacement (using hardware-accelerated bilinear interpolation), and stores the net displacement in a new 2D texture.
4. This texture—now storing the net flow displacement—is sampled by another vertex shader, transforming all fluid particles to their updated locations in parallel.
5. A fragment shader then colors each pixel based on its initial location at the inlet.

Steps 1 and 2 are part of uFlow's initialization, step 3 only occurs when a new microfluidic part is added to the flow sculpting device, and steps 4–5 are processed for each frame rendered (at 60 frames per second). Currently, uFlow contains a library of pillars spanning the height of their channel with a channel aspect ratio of 4:1, width:height, so each transformation is symmetric about the horizontal channel midline. Therefore, we only compute advection in the top half-channel cross section, and mirror the result about the channel midline.

References

Amini H, Sollier E, Masaeli M, Xie Y, Ganapathysubramanian B, Ha Stone, Di Carlo D (2013) Engineering fluid flow using sequenced microstructures. Nat Commun 4(May):1826

Andrews CA, Davies JM, Schwarz GR (1967) Adaptive data compression. Proc IEEE 55(3):267–277

Appleyard DC, Chapin SC, Srinivas RL, Doyle PS (2011) Bar-coded hydrogel microparticles for protein detection: synthesis, assay and scanning. Nat Protoc 6(11):1761–1774

Bong KW, Kim JJ, Cho H, Lim E, Doyle PS, Irimia D (2015) Synthesis of cell-adhesive anisotropic multifunctional particles by stop flow lithography and streptavidin-biotin interactions. Langmuir 31(48):13,165–13,171

Chen L, An HZ, Haghgooie R, Shank AT, Martel JM, Toner M, Doyle PS (2016) Flexible octopus-shaped hydrogel particles for specific cell capture. Small 12:2001–2008. https://doi.org/10.1002/smll.201600163

Diaz-Montes J, Xie Y, Rodero I, Zola J, Ganapathysubramanian B, Parashar M (2014) Federated computing for the masses—aggregating resources to tackle large-scale engineering problems. Comput Sci Eng 16(4):62–72

Do AV, Khorsand B, Geary SM, Salem AK (2015) 3d printing of scaffolds for tissue regeneration applications. Adv Healthc Mater 4(12):1742–1762

Galisteo-López JF, Ibisate M, Sapienza R, Froufe-Peréz LS, Blanco A, López C (2011) Self-assembled photonic structures. Adv Mater 23(1):30–69

Griffin DR, Weaver WM, Scumpia PO, Di Carlo D, Segura T (2015) Accelerated wound healing by injectable microporous gel scaffolds assembled from annealed building blocks. Nat Mater 14(7):737–744

Gurkan UA, Tasoglu S, Kavaz D, Demirel MC, Demirci U (2012) Emerging technologies for assembly of microscale hydrogels. Adv Healthc Mater 1(2):149–158

Hart JC (1996) Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. Vis Comput 12(10):527–545

Howell PB, Golden JP, Hilliard LR, Erickson JS, Mott DR, Ligler FS (2008) Two simple and rugged designs for creating microfluidic sheath flow. Lab Chip 8(7):1097–1103

Howell PB, Mott DR, Golden JP, Ligler FS (2004) Design and evaluation of a Dean vortex-based micromixer. Lab Chip 4(6):663–669

Ismagilov RF, Stroock AD, Kenis PJa, Whitesides G, Stone HA (2000) Experimental and theoretical scaling laws for transverse diffusive broadening in two-phase laminar flows in microchannels. Appl Phys Lett 76(17):2376

Lee J, Bisso PW, Srinivas RL, Kim JJ, Swiston AJ, Doyle PS (2014) Universal process-inert encoding architecture for polymer microparticles. Nat Mater 13(5):524–529

Liu RH, Ma Stremler, Sharp KV, Olsen MG, Santiago JG, Adrian RJ, Aref H, Beebe DJ (2000) Passive mixing in a three-dimensional serpentine microchannel. J Microelectromech Syst 9(2):190–197

Mott DR, Howell PB Jr, Golden JP, Kaplan CR, Ligler FS, Oran ES (2006) Toolbox for the design of optimized microfluidic components. Lab Chip 6(4):540

Nunes JK, Wu CY, Amini H, Owsley K, Di Carlo D, Stone HA (2014) Fabricating shaped microfibers with inertial microfluidics. Adv Mater 26:3712–3717

Paulsen KS, Di Carlo D, Chung AJ (2015) Optofluidic fabrication for 3D-shaped particles. Nat Commun 6:6976

Paulsen KS, Deng Y, Chung AJ (2018) DIY 3D Microparticle Generation from Next Generation Optofluidic Fabrication. Adv Sci. https://doi.org/10.1002/advs.201800252

Sollier E, Amini H, Go DE, Pa Sandoz, Owsley K, Di Carlo D (2015) Inertial microfluidic programming of microparticle-laden flows for solution transfer around cells and particles. Microfluid Nanofluid 19:53–65

Sollier E, Murray C, Maoddi P, Di Carlo D (2011) Rapid prototyping polymers for microfluidic devices and high pressure injections. Lab Chip 11(22):3752

Spiga M, Morino GL (1994) A symmetric solution for velocity profile in laminar flow through rectangular ducts. Int Commun Heat Mass Transfer 21(4):469–475

Stoecklein D, Davies M, Wubshet N, Le J, Ganapathysubramanian B (2016) Automated design for microfluid flow sculpting: multiresolution approaches, efficient encoding, and GPU implementation. J Fluids Eng 139:1–11

Stoecklein D, Wu CY, Owsley K, Xie Y, Di Carlo D, Ganapathysubramanian B (2014) Micropillar sequence designs for fundamental inertial flow transformations. Lab Chip 14(21):4197–4204

Stroock AD, Dertinger SKW, Ajdari A, Mezic I, Stone HA, Whitesides GM (2002) Chaotic mixer for microchannels. Science (New York, NY) 295:647–651

Sudarsan AP, Ugaz VM (2006) Multivortex micromixing. Proc Nat Acad Sci USA 103(Track II):7228–7233

Wu CY, Owsley K, Di Carlo D (2015) Rapid software-based design and optical transient liquid molding of microparticles. Adv Mater 27(48):7970–7978