REPORT DOCUMENTATION PAGE

Form Approved OMB NO. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggesstions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA, 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any oenalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.
PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE (DD-MM-YYYY)	2. REPORT TYPE	3. DATES COVERED (From - To)
04-10-2017	Final Report	15-Jul-2014 - 14-Jul-2017

4. TITLE AND SUBTITLE

Final Report: Efficient Numeric and Geometric Computations using Heterogeneous Shared Memory Architectures

5a. CONTRACT NUMBER
W911NF-14-1-0437

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
611102

6. AUTHORS

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES AND ADDRESSES

University of North Carolina - Chapel Hill
104 Airport Drive, CB 1350
Suite 2200
Chapel Hill, NC 27599 -1350

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS (ES)

U.S. Army Research Office
P.O. Box 12211
Research Triangle Park, NC 27709-2211

10. SPONSOR/MONITOR'S ACRONYM(S)
ARO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)
64545-CS.16

12. DISTRIBUTION AVAILIBILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

The views, opinions and/or findings contained in this report are those of the author(s) and should not contrued as an official Department of the Army position, policy or decision, unless so designated by other documentation.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	15. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dinesh Manocha
UU	UU	UU	UU		19b. TELEPHONE NUMBER 919-962-1749

RPPR Final Report
as of 03-Nov-2017

Agency Code:

Proposal Number: 64545CS **Agreement Number: W911NF-14-1-0437**
INVESTIGATOR(S):

Name: Dinesh Manocha
Email: dm@cs.unc.edu
Phone Number: 9199621749
Principal: Y

Organization: **University of North Carolina - Chapel Hill**
Address: 104 Airport Drive, CB 1350, Chapel Hill, NC 275991350
Country: USA
DUNS Number: 608195277 EIN: 566001393
Report Date: 14-Oct-2017 Date Received: 04-Oct-2017
Final Report for Period Beginning 15-Jul-2014 and Ending 14-Jul-2017
Title: Efficient Numeric and Geometric Computations using Heterogeneous Shared Memory Architectures
Begin Performance Period: 15-Jul-2014 **End Performance Period:** 14-Jul-2017
Report Term: 0-Other
Submitted By: Dinesh Manocha Email: dm@cs.unc.edu
Phone: (919) 962-1749
Distribution Statement: 1-Approved for public release; distribution is unlimited.

STEM Degrees: **STEM Participants:** 6

Major Goals: The main goals were to develop new set of algorithms for scientific and geometric computing by exploiting the power and performance efficiency of heterogeneous shared memory architectures. These include multi-core CPUs and many-core GPUs. This involves design of new algorithms, development of appropriate software environments, and demonstrating their potential on a few applications. These applications include ray and path tracing for visual and sound rendering and solving the scientific models for fluid simulation. New heterogeneous algorithms will be developed to
exploit CPU and GPU cores in parallel. These include:

1. Development of new methods to improve the performance of Monte Carlo Path Tracing Algorithms for visual and sound rendering. These methods will be mapped to the memory architectures of CPUs and GPUs to obtain good performance and result in good memory performance using cache management. These methods will also be extended to non-linear ray tracing.

2. Develop new solvers for large-scale fluid simulation and fluid control for different applications using the parallelism of commodity processors. Besides solving the Navier-Stoke equations, these methods will be combined with optimization methods to generate appropriate behaviors. We will measure the speedups obtained by novel methods and exploitation of parallelism.

These algorithms will be demonstrated for various applications including outdoor acoustics, sound rendering, smoke and dense fluid simulation.

The PI will also work closely with the researchers at ARL and RDECOM to evaluate the performance on DoD and Army datasets and transition the technologies. he PI has worked closely with many research groups at Army, including Army Research Labs,
RDECOM, PEO STRI, and USACE. The new heterogeneous algorithms for ray tracing can be used for faster line-of-sight computations for
semi-automated forces and virtual reality training (collaboration with RDECOM, Orlando), RF propagation (ARL, Aberdeen Proving Ground),
sound propagation for outdoor environments (ERDC-CRREL Signature Physics Branch, USACE, Hanover). New fluid simulation algorithms can be
used for training environment (RDECOM, Orlando) and blast wave simulation (ARL, Aberdeen Proving Ground).

Accomplishments: The PI and students has developed new methods for path and ray tracing and their

applications to sound simulation. Moreover, the PI and the students developed new methods for fluid simulation and control.

RAY AND PATH TRACING AND APPLICATIONS TO SOUND RENDERING

The problem of generating high-fidelity sound effects is important for many interactive applications, including games, virtual reality, and human-computer interaction systems. It is well-known that realistic sounds can improve a user's sense of presence and immersion. Moreover, recent availability of commodity head-mounted displays and augmented reality devices has renewed interest in sound simulation and rendering.

We addressed the problem of interactive sound propagation in complex indoor and outdoor environments with a large number of sources.
The primary goal is to simulate how sound waves in a virtual scene interact with the environment and are heard by a listener.
This includes modeling of various acoustic effects including reflections, diffraction, Doppler shift, echoes, etc.
Another major issue is simulating reverberation, the persistence of sound that occurs due to the build up of a large number of reflections that then decay over time due to absorption.
%The length of the decay, or reverberation time, is an important criteria in the design of architectural spaces and can vary anywhere from very short (e.g. 100ms}) to long (e.g. > 3 s}).
For example, music halls are designed so that they can generate a reverb that lasts anywhere from 1.2 to 3$seconds, or longer. In a typical room (e.g. conference room), the reverb should be anywhere from 0.2 to 1 seconds.
Outdoor scenes usually have short reverb times (< 500ms}) because most sound escapes the scene after a few reflections.
The kind of reverb in a scene can give important cues related to the nature of the environment, including its size, localization, speech intelligibility, etc.

A key challenge is to generate high-fidelity sound effects that can accurately model real-world experiences at interactive rates, i.e. >10Hz.
Current algorithms for interactive sound propagation in dynamic scenes are based on geometric sound propagation. These methods compute the propagation paths from each source to the listener using ray tracing, beam tracing, or frustum tracing.
The result of sound propagation is a one-dimensional filter called
an impulse response (IR) that specifies the linear transfer function between a source and listener pair.
The final audio at the listener position is generated by convolving the IR with the unprocessed source sound.

The runtime complexity of geometric propagation algorithms is generally dominated by the computation of IRs using ray tracing.
For example, Monte Carlo path tracing methods are frequently used for diffuse reflections and account for a significant fraction of the total time spent in IR computation. The runtime complexity is a linear function of the number of rays being traced for each acoustic effect, and it is important to minimize the number of rays in order to achieve interactive performance.
A recent method exploits temporal coherence of the sound field and reduces the number of rays required to achieve high-quality sound. However, it can require over 100MB per source to store the propagation paths for high-order reflections.

Another significant issue for sound propagation algorithms is to determine the length of the impulse response a priori. The length of the IR specifies how long that sound should travel at the speed of sound. If this parameter is longer than the actual response, ray tracing computation is wasted on sound that is inaudible to a human listener. In highly reverberant environments (e.g. parking garage, cathedral), the IR will be truncated if it does not capture the entire sound decay. The loudness of the sound source is also an important factor, since louder sources tend to have longer audible reverberation.
In practice, the length of the IR needs to be tuned for each acoustic space
and sound source that the listener is likely to encounter.
As a result, choosing the appropriate IR length is a major issue in generation of realistic sound effects at interactive rates.

We develalgorithms for adaptive impulse response modeling that can significantly improve the performance and

reduce the memory requirements for ray-tracing based interactive sound propagation.
The two novel contributions include:

Impulse Response Cache:
Our algorithm caches the IRs from prior frames and performs recursive filtering in order to take advantage of temporal coherence and trace fewer rays during each frame while maintaining high-quality sound. The sound field on the current frame is expressed as a weighted sum of the impulse responses from the current and previous frame. This reduction in the number of rays traced increases the runtime performance (up to 5X) and reduces the memory overhead.

Adaptive Impulse Response Length: We augment our ray-tracing algorithm to dynamically detect the current impulse response length for each sound source based on a psychoacoustic metric and the source's power. A feedback mechanism is used to determine how far to propagate rays on the next frame, thereby reducing the amount of computation needed for less-audible sources and for rooms with short reverberation times.

These techniques have been implemented on a 4-core consumer desktop machine and evaluated using a variety of complex dynamic indoor and outdoor scenes with dozens of sources. We exploited the parallelism and measured its benefits. We demonstrate a 5x savings in computation time as well as almost 2 orders-of-magnitude decrease in memory requirements as compared to previous interactive methods. We have also performed a preliminary user evaluation that suggests our results are comparable to a ground-truth offline simulation with 100x as many rays. In practice, our approach can generate realistic acoustic effects at interactive rates for a wide variety of complex indoor and outdoor scenes with dynamically-varying reverberation.

FLUID SIMULATION

We developed novel algorithm to control the physically-based animation of
smoke. Given a set of keyframe smoke shapes, we compute a dense sequence of control force fields that can drive the smoke shape to match
several keyframes at certain time instances. Our approach formulates this
control problem as a PDE-constrained spacetime optimization. In order to
compute the locally optimal control forces, we alternatively optimize the
velocity fields and density fields using optimizer. In order to
reduce the high complexity of multiple passes of fluid resimulation during
velocity field optimization, we utilize the coherence between consecutive
fluid simulation passes. . Our approach exploits the special structure of the
Navier-Stokes equations discretized on a regular staggered grid,
and solves the optimization problem by finding the stationary point
of the first order optimality (Karush-Kuhn-Tucker) conditions. We have evaluated our approach on several benchmarks. Our
benchmarks vary in terms of the grid resolution, the number of
timesteps, and the control force regularization parameter. We highlight
results with up to 60 timesteps at the resolution of 64 cubed. Without
using force templates, we allow each component of the velocity
field to be controlled. We demonstrate the benefits of our approach by computing accurate solutions on 2D and 3D benchmarks and parallelize the algorithm on CPU and GPU. In practice, we observe up to an order of magnitude improvement over prior optimal control
methods.

Training Opportunities: Most of the funding was used to support the graduate students. In particular, six PhD students were supported, and four of them finished their PhD during the period of the grant.

Results Dissemination: A high number of papers were published based on this grant. This includes more than 12 papers in the leading conferences and journals on sound simulation, computer graphics, robotics, etc. All those papers are available at:

https://wwwx.cs.unc.edu/~geom/papers/all/

They have been uploaded to ARO online database as part of the prior interim reports.

RPPR Final Report
as of 03-Nov-2017

Honors and Awards: Many students supported by this grant received prestigious awards. These include:

1. NC Royster Award to Nicolas Morales at NC ASA meeting in 2016.

2. IEEE VR Dissertation Award to Ravish Mehra at IEEE VR Conference in March 2017.

Protocol Activity Status:

Technology Transfer: UNC has filed a few patent application on research results supported by this grant. A report of all those inventions was submitted to ARO as part of a separate report. Some companies have expressed in licensing these patents.

Some of the earlier work supported by ARO grants was transitioned to Impulsonic Inc, which was acquired by Valve in November 2016.

PARTICIPANTS:

Participant Type: PD/PI
Participant: Dinesh Manocha
Person Months Worked: 1.00 **Funding Support:**
Project Contribution:
International Collaboration:
International Travel:
National Academy Member: N
Other Collaborators:

ARTICLES:

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: ACM Transactions on Graphics
Publication Identifier Type: DOI Publication Identifier: 10.1145/3016963
Volume: 36 Issue: 5 First Page #: 1
Date Submitted: 9/1/17 12:00AM Date Published: 7/1/17 4:00AM
Publication Location:
Article Title: Efficient Solver for Spacetime Control of Smoke
Authors: Zherong Pan, Dinesh Manocha
Keywords: fluid simulation, optimization
Abstract: We present a novel algorithm to control the physically-based animation of smoke. Given a set of keyframe smoke shapes, we compute a dense sequence of control force fields that can drive the smoke shape to match several keyframes at certain time instances. Our approach formulates this control problem as a PDE-constrained spacetime optimization. In order to compute the locally optimal control forces, we alternatively optimize the velocity fields and density fields using an ADMM optimizer. In order to reduce the high complexity of multiple passes of fluid resimulation during velocity field optimization, we utilize the coherence between consecutive fluid simulation passes. We demonstrate the benefits of our approach by computing accurate solutions on 2D and 3D benchmarks. In practice, we observe up to an order of magnitude improvement over prior optimal control methods
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 03-Nov-2017

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: IEEE Transactions on Visualization and Computer Graphics
Publication Identifier Type: DOI Publication Identifier: 10.1109/TVCG.2016.2516990
Volume: 22 Issue: 10 First Page #: 2229
Date Submitted: 9/1/17 12:00AM Date Published: 10/1/16 4:00AM
Publication Location:
Article Title: Compressed Coverage Masks for Path Rendering on Mobile GPUs
Authors: Pavel Krajcevski, Dinesh Manocha
Keywords: path rendering, textures, parallel algorithms, GPUs
Abstract: We present an algorithm to accelerate resolution independent curve rendering on mobile GPUs. Recent trends in graphics hardware have created a plethora of compressed texture formats specific to GPU manufacturers. However, certain implementations of platform independent path rendering require generating grayscale textures on the CPU containing the extent that each pixel is covered by the curve. In this paper, we demonstrate that generating a compressed grayscale texture prior to uploading it to the GPU creates faster rendering times in addition to the memory savings. We implement a real-time compression technique for coverage masks and compare our results against the GPU-based implementation of the highly optimized Skia rendering library. We also analyze the worst case properties of our compression algorithms. We observe up to a 2X speed improvement over the existing GPU-based methods in addition to up to a 9:1 improvement in GPU memory gains. We demonstrate the performance on multiple
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

Publication Type: Journal Article Peer Reviewed: Y **Publication Status:** 1-Published
Journal: ACM Transactions on Graphics
Publication Identifier Type: DOI Publication Identifier: 10.1145/2980179.2982439
Volume: 35 Issue: 6 First Page #: 1
Date Submitted: 9/1/17 12:00AM Date Published: 11/1/16 4:00AM
Publication Location:
Article Title: GST: GPU-decodable Supercompressed Textures
Authors: Pavel Krajcevski, Srihari Pratapa, Dinesh Manocha
Keywords: texture compression, parallel computation
Abstract: Modern GPUs supporting compressed textures allow interactive application developers to save scarce GPU resources such as VRAM and bandwidth. Compressed textures use fixed compression ratios whose lossy representations are significantly poorer quality than traditional image compression formats such as JPEG. We present a new method in the class of supercompressed textures that provides an additional layer of compression to already compressed textures. Our texture representation is designed for endpoint compressed formats such as DXT and PVRTC and decoding on commodity GPUs. We apply our algorithm to commonly used formats by separating their representation into two parts that are processed independently and then entropy encoded. Our method preserves the CPU-GPU bandwidth during the decoding phase and exploits the parallelism of GPUs to provide up to 3X faster decode compared to prior texture supercompression algorithms. Along with the gains in decoding speed, our method maintains both the
Distribution Statement: 1-Approved for public release; distribution is unlimited.
Acknowledged Federal Support: **Y**

CONFERENCE PAPERS:

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: ACM Symposium on Interactive 3D Graphics and Games
Date Received: 07-Sep-2017 Conference Date: 27-Feb-2015 Date Published: 27-Feb-2015
Conference Location: San Francisco, California
Paper Title: Compressed coverage masks for path rendering on mobile GPUs
Authors: Pavel Krajcevski and Dinesh Manocha
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 03-Nov-2017

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2016 IEEE International Conference on Robotics and Automation (ICRA)
Date Received: 07-Sep-2017 Conference Date: 16-May-2016 Date Published: 16-May-2016
Conference Location: Stockholm, Sweden
Paper Title: Real-time reciprocal collision avoidance with elliptical agents
Authors: Andrew Best, Sahil Narang, Dinesh Manocha
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2016 IEEE International Conference on Robotics and Automation (ICRA)
Date Received: 07-Sep-2017 Conference Date: 16-May-2016 Date Published:
Conference Location: Stockholm, Sweden
Paper Title: Proxemic group behaviors using reciprocal multi-agent navigation
Authors: Liang He, Jia Pan, Wenping Wang, Dinesh Manocha
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2016 IEEE International Conference on Robotics and Automation (ICRA)
Date Received: 28-Aug-2016 Conference Date: 16-May-2016 Date Published: 16-May-2016
Conference Location: Stockholm, Sweden
Paper Title: GLMP- realtime pedestrian path prediction using global and local movement patterns
Authors: Aniket Bera, Sujeong Kim, Tanmay Randhavane, Srihari Pratapa and Dinesh Manocha
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2017 IEEE International Conference on Robotics and Automation (ICRA)
Date Received: 01-Sep-2017 Conference Date: 29-May-2017 Date Published:
Conference Location: Singapore, Singapore
Paper Title: Efficient multi-agent global navigation using interpolating bridges
Authors: Liang He, Jia Pan, Dinesh Manocha
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: the 21st ACM SIGGRAPH Symposium
Date Received: 01-Sep-2017 Conference Date: 25-Feb-2017 Date Published:
Conference Location: San Francisco, California
Paper Title: MPTC: Video Rendering for Virtual Screens using Compressed Textures
Authors: Srihari Pratapa, Pavel Krajcevski, Dinesh Manocha
Acknowledged Federal Support: **Y**

Publication Type: Conference Paper or Presentation **Publication Status:** 1-Published
Conference Name: 2017 IEEE International Conference on Robotics and Automation (ICRA)
Date Received: 01-Sep-2017 Conference Date: 29-May-2017 Date Published:
Conference Location: Singapore, Singapore
Paper Title: Efficient probabilistic collision detection for non-convex shapes
Authors: Jaesung Park, Chonhyon Park, Dinesh Manocha
Acknowledged Federal Support: **Y**

RPPR Final Report
as of 03-Nov-2017

Analytic Ray Curve Tracing for Outdoor Sound Propagation

Qi Mo[a,*], Hengchin Yeh[a], Ming Lin[a], Dinesh Manocha[a]

[a]*Department of Computer Science, University of North Carolina at Chapel Hill, NC*

Abstract

Outdoor sound propagation, which propagates sound through inhomogeneous, moving media with complex obstacles, presents challenging scenarios for computational simulation. In this paper, we present a ray-tracing method that uses analytic ray curves as tracing primitives in order to improve the efficiency of outdoor sound propagation in fully general settings. This ray-curve tracer inherits the efficiency and flexibility of rectilinear ray tracers in handling boundary surfaces, and it overcomes the performance limitations imposed by approximating the curved propagation paths in inhomogeneous media with rectilinear rays. Adaptive media traversal, as well as acceleration structures for surfaces intersections, lead to further savings in computation. Our method's speedup over existing ray models, at least an order of magnitude for simple 2D scenarios and up to two orders of magnitude for 3D complex scenes, is demonstrated on outdoor benchmark scenes.

Keywords: Ray Tracing, Sound Propagation, Atmospheric Acoustics, Underwater Acoustics

1. Introduction

Sound propagation in outdoor environments[1, 2, 3] and room acoustics[4] are traditionally investigated separately. Room acoustics studies focus on the high-order interactions that sound waves make with surfaces within an enclosed

*Corresponding Author: Email, qmo@cs.unc.edu; Phone, (319)331-4931 (Qi Mo)

space, and assume a homogeneous medium. On the other hand, outdoor sound propagation (atmospheric and underwater) deals with media that are spatially varying and that move; they prominently differ from indoor models in the curved propagation trajectories that come from refraction. In the outdoor context, obstacles that constitute media boundaries are generally assumed to be sparse, and the complexity of the boundary surfaces is limited in the investigations.

Many outdoor applications can, however, benefit from simultaneous modeling of fully general media and complex boundaries. Examples include large expanses of complex terrain and sprawling urban areas, which often have their own micro-climate. As data on spatially- and temporally- varying outdoor media, such as the atmosphere and the ocean, becomes increasingly available, and geographic information on terrains and man-made structures is recorded in ever richer details, sound propagation models should ideally adapt to utilize the full scale of such data. Existing numerical methods suffer from scalability issues with large domain volume, high frequency, or complex boundaries. Geometric acoustics (GA)[5] methods are efficient at handling boundary surfaces, but with inhomogeneous media the curved trajectories become expensive to compute.

One way to speed up ray models for inhomogeneous media is to subdivide the media into spatial cells, so that within each cell the media assumes simple profile that leads to an analytic curved trajectory. The *cell method* was proposed as exactly such a model; based on a triangular subdivision of two-dimensional media, the cell method has been implemented in some acoustic simulation software (e.g. TRIMAIN[6], RAYWAVE[7]). A later work[8] adapted the idea of cell method to both light and sound propagation by decomposing three-dimensional media profiles into tetrahedral cells. This adaptation utilized mesh connectivity to trace rays across neighboring cells, and embedded boundary surfaces into the mesh so that surface intersections are also solved by the mesh-traversal process.

This paper further develops the cell method idea, and the algorithm improves upon the tetrahedral cell method[8] in three important aspects:

1. **Simple ray formulation** We select the parabolic ray curve as the ray-tracing primitive, which offers the simplest analytic form for trajectory,

intersection, and ray properties (Sec.3).

2. **Implicit cell** We use a mesh-less approach to media traversal, tracing ray curve segments of adaptive sizes based on on-the-fly sampling of the media profile. This implicit-cell approach avoids costly mesh construction, and it supports moving media as well as dynamic media (Sec.4.2).
3. **Acceleration of surface intersection** For the ray curve tracer, we adapt the hierarchical acceleration structures used in rectilinear ray tracers. Further acceleration is achieved by spatial bounding of ray curves based on their geometric properties, which offers higher culling efficiency (Sec.4.3).

Overall this analytic ray curve tracer is designed to be efficient for general media profiles and complex scenes with tens of thousands of surface primitives. Its performance is demonstrated on outdoor benchmarks (Sec.5), where it shows one to two orders of magnitude speedup over previous ray models. It is complementary to a set of numerical and geometric methods and can be extended in multiple ways (as discussed in Sec. 6).

2. Prior Work

Outdoor sound propagation has been studied extensively in underwater and atmospheric acoustics. Methods including the Fast Field Program (FFP)[9], Parabolic Equation (PE)[10, 11], and Normal Modes[1, 3] provide full-wave solutions that depend upon simplifying assumptions about the media and the obstacles in the scene. More general numerical methods, able to handle arbitrary media and obstacles, have also been proposed; the leading one is the Finite Difference Time Domain(FDTD) method, which solves the linearized Euler equation[12, 13]. The main disadvantage of the FDTD is its high computational cost, and its poor scalability with the volume of the simulation domain or with the frequency. Other numerical methods, including Finite Element Method[14, 15], Boundary Element Method[16], Pseudo Spectral Time Domain[17, 18], and Transmission Line Matrix (TLM)[19, 20, 21], face similar challenges in scalability.

Geometric acoustics (GA)[5] methods are widely used in room acoustics[4]. GA methods rely on the high-frequency approximation of wave propagation using rays, which are perpendicular to the wave fronts and can be intersected efficiently with scene surfaces. Various GA techniques have been developed, including image source method[22, 23], ray tracing[24, 25], frustum tracing[26], beam tracing[27, 28], sonal mapping[29, 30, 31], path tracing[32], and radiosity[33]. In these methods, sound speed is generally assumed to be constant, so rectilinear rays are used to trace out straight-line paths. Complex surface interactions, such as specular reflection, Snell's Law refraction, diffuse reflection [25, 32], and Bidirectional Reflectance Distribution Function (BRDF)[31, 34], are modeled.

Ray models have also been applied to inhomogeneous outdoor media by numerical integration of the differential ray equation (see survey [1, 3]). The integration step size is limited by the range within which the media can be assumed to be homogeneous, and intersection with surfaces need to be tested for each segment. With complex media the great number of small ray segments dominate the propagation computation. Higher-order integration, like the fourth-order Runge-Kutta method, have been adopted to improve efficiency, but with those methods the trajectory between each advancement of the ray integration has no explicit form, complicating intersection tests with surfaces (see details in Fig. 4). Several widely used software tools are based on ray models, including HARPO[35], BELLHOP[36], WaveQ3D[37] in underwater acoustics, HARPA[38] in atmospheric acoustics (see [39] for the collection). Although performance of these ray-based tools is orders of magnitude better than wave-based alternatives (reported recently[40, 41], a few seconds vs. thousands of seconds on 2D propagation in layered media), these methods quickly become prohibitively expensive when dealing with general media and 3D complex scenes.

3. Analytic Ray Curve

Ray models simulate wave propagation with rays that are perpendicular to the wavefront. For certain media profiles, the ray trajectories, as well as the

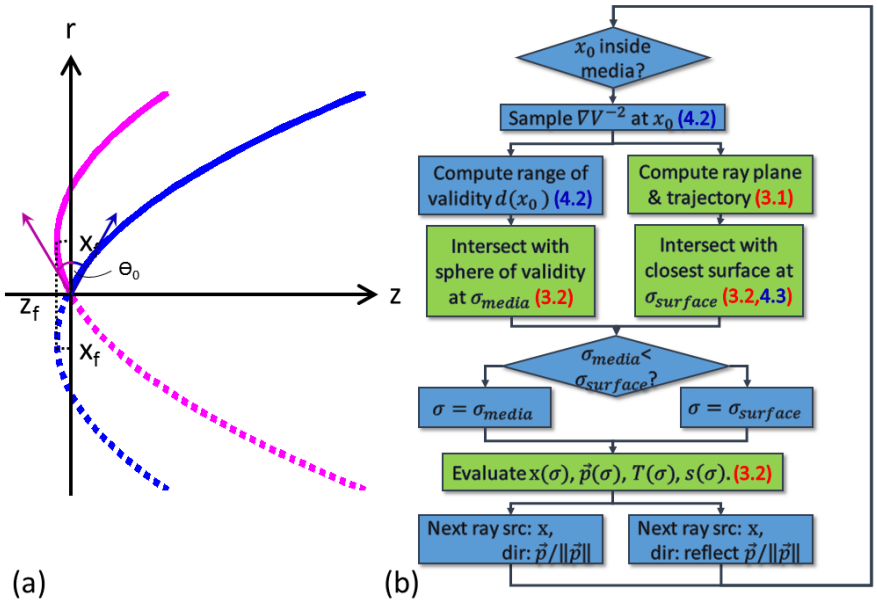


Figure 1: **(a) Parabolic ray curve.** Two rays with different initial directions (marked by arrows) are shown in the *ray plane*, with z-axis being the direction of ∇V^{-2}. (x_f, z_f), which defines the vertex of the parabola, is computed according to Eq. 6. **(b) Ray curve tracing.** The green-colored blocks have analytic solutions with the parabolic ray curve, which offer large savings in ray-tracing costs. For each key step, the relevant sections that give the analytic formulation are marked in red, and the relevant sections that describe algorithm details are marked in blue.

pressure on the ray, have analytic forms that can be evaluated at any point along the ray at constant cost. Our ray tracer utilizes one of these analytic formulations (Fig. 1(a)), chosen for its particularly simple form, and the resulting ray curve lends itself to easy intersections with boundary surfaces. As can be seen from Fig. 1(b), this particular ray curve enables analytic evaluations at a constant per-ray cost in key steps of the ray tracing (green-colored blocks). This ray formulation is a special case of the more general ray theory discussed in depth in Červený's comprehensive work[42].

3.1. Ray formulation

Within a medium with spatially-varying sound speed, we denote $V(\mathbf{x})$ as the sound speed at location $\mathbf{x}$. The *arrival time* or *travel time* field T is defined

as the time that a sound wave takes to travel from its source to a field location, and the spatial derivative of T: $\vec{p} = \nabla T$ is called the *slowness vector*. Within such a medium, the ray trajectories can be solved from the general *Hamiltonian* form of the Eikonal equation, which in Cartesian coordinates reads:

$$\mathcal{H}(x_i, p_i) = n^{-1}\{(p_i p_i)^{n/2} - V^{-n}\}, i = 1, 2, 3, \tag{1}$$

with n as a real number corresponding to different forms of the equation. Eq. 1 can be solved in terms of *characteristics*: 3-D trajectories $x_i = x_i(u)$ parameterized by u, along which 1 is satisfied, as follows:

$$\begin{aligned} &\frac{\mathrm{d}x_i}{\mathrm{d}u} = (p_k p_k)^{n/2-1} p_i, \qquad \frac{\mathrm{d}p_i}{\mathrm{d}u} = \frac{1}{n}\frac{\partial}{\partial x_i}(V^{-n}), \\ &\frac{\mathrm{d}T}{\mathrm{d}u} = (p_k p_k)^{n/2} = V^{-n}. \end{aligned} \tag{2}$$

When the gradient of V^{-n} is constant, 2 can generally be solved analytically. When we assume that the gradient of V^{-2} is constant (i.e. $n = 2$: $V(\mathbf{x})^{-2} = A_0 + \vec{\mathbf{A}} \cdot \mathbf{x}$), we obtain the simplest analytic form for ray trajectory and travel time, in terms of the parameter $u = \sigma$:

$$x_i(\sigma) = x_{i0} + p_{i0}(\sigma - \sigma_0) + \frac{1}{4}A_i(\sigma - \sigma_0)^2, \tag{3}$$

$$p_i(\sigma) = p_{i0} + \frac{1}{2}A_i(\sigma - \sigma_0), \tag{4}$$

$$T(\sigma) = T(\sigma_0) + V_0^{-2}(\sigma - \sigma_0) + \frac{1}{2}A_i p_{i0}(\sigma - \sigma_0)^2 + \frac{1}{12}A_i A_i(\sigma - \sigma_0)^3. \tag{5}$$

σ is related to the travel time T and to arclength s by $\mathrm{d}\sigma = V^2 \mathrm{d}T = V\mathrm{d}s$.

3.2. Ray properties

We take advantage of two key properties of the ray curve given in Sec. 3.1:

1. **Intersection** The ray is a parabolic plane curve and can be intersected analytically with planar surfaces to solve for the intersection point.
2. **Evaluation** For an arbitrary parameter σ along the ray, the corresponding position, tangent direction, and the ray properties contributing to pressure

computation, including the slowness vector, arrival time, and arclength, can be evaluated analytically at constant cost.

It is evident from Eq. 3 that the analytic ray lies in a plane with the normal of $\vec{A} \times \vec{p_0}$. We denote the direction of media gradient $\vec{A}$ as z-axis and $(\vec{A} \times \vec{p_0}) \times \vec{A}$ as r-axis; the r-z plane is then called the *ray plane*. When the origin of the coordinate system is placed at ray origin $\mathbf{x_0}$, the ray trajectory is:

$$z(r) = \frac{\alpha}{4\xi_0^2}(r - r_f)^2 - z_f, \qquad r_f = -\frac{2\xi_0\sqrt{-{\xi_0}^2 + {n_0}^2}}{\alpha}, z_f = \frac{{n_0}^2 - {\xi_0}^2}{\alpha}, \quad (6)$$

where $\alpha = \|\vec{A}\|$ is the magnitude of media gradient, $\xi_0 = V(0)^{-1} cos\theta_0$, $V(0)$ is the sound speed at ray origin, and θ_0 is the angle between the ray direction at origin and the r axis (see Fig. 1). The parabolic plane curve given by Eq. 6 has closed-form solutions for its intersections with planar surfaces.

For an arbitrary point on the ray curve with coordinates (r_p, z_p) in the ray plane, the corresponding parameter σ_p can be solved from Eq. 3 as $\sigma_p = r_p/\xi_0 + \sigma_0$, where σ_0 is the σ at ray origin. Given σ_p, we can calculate the position $\vec{x}(\sigma_p)$, the slowness vector $\vec{p}(\sigma_p)$, and travel time $T(\sigma_p)$ from Eq. 3,4,5, respectively. The tangent direction $\vec{t}$ coincides with the direction of the slowness vector $\vec{p}$ and can be evaluated by $\vec{t}(\sigma_p) = \vec{p}(\sigma_p)/\|\vec{p}(\sigma_p)\|$. The arclength s of the parabolic ray curve can be evaluated at (r_p, z_p) by

$$s(r_p) = \left.\frac{\sqrt{(ar_p + b)^2 + 1}(ar_p + b) + \operatorname{arcsinh}(ar_p + b)}{2a}\right|_0^r, \quad (7)$$

$$\text{where } a = \frac{1}{2}\frac{\alpha}{\xi_0^2}, \quad \text{and } b = \frac{\sqrt{-\xi_0 + V(0)^{-2}}}{\xi_0'}.$$

The arclength s determines the attenuation along the ray path, and it is one of the key variables that we measure for accuracy of the ray tracing (see Sec. 5.1).

Similar to rectilinear rays, the parabolic ray curve comes with efficient operations for finding intersection and for updating the key properties at intersection points, making it amenable to be used as a tracing primitive. The ray curve serves as a building block for a ray tracer that handles general media; it does

so by computing the propagation paths, which consist of consecutive segments of the aforementioned ray curves. The flow chart in Fig. 1(b) illustrates the steps involved in tracing one such segment, with the computational savings highlighted by green-colored blocks. In the next section, we give the details of our ray-tracing algorithm using the parabolic ray formulation.

4. Ray Tracing Algorithm

4.1. Tracing Analytic Segments

Our ray tracer computes the propagation paths in a segment-by-segment fashion, with each segment being a parabolic ray curve (formulated in Sec. 3). There are two criteria for terminating one segment and starting the next one:

1. The media no longer satisfy the assumption of constant ∇V^{-2},
2. The boundary surfaces in the scene are encountered.

For the former criterion, the change in the media gradient ∇V^{-2}, we propose an adaptive segment size that is computed from on-the-fly sampling of the media; this is discussed in detail in Sec. 4.2. For the latter criterion, in which the segment encounters a surface, we use acceleration data structures and methods adapted from rectilinear ray tracer; these are discussed in detail in Sec. 4.3. Both aspects of the ray-tracing algorithm contribute to its efficiency and offer performance improvements over previous methods (see Fig. 2 and Fig. 3).

Based on these criteria for ending segments, we find out how far a particular segment will go in two ways. For the surface-encounter criterion, we compute the closest intersection point; for the media gradient criterion, we compute the point at which the ray curve reaches the edge of the *range of validity*. The closest point between the two cases is taken as the end point of the current segment, and σ, $\mathbf{x}(\sigma)$, $\vec{t}(\sigma)$, $T(\sigma)$, $s(\sigma)$ are evaluated at this point. The next segment starts at $\mathbf{x}(\sigma)$, and its trajectory is computed from the local media gradient $\nabla V^{-2}(\mathbf{x}(\sigma))$. σ, $T(\sigma)$, $s(\sigma)$ are continued over the next segment; the ray direction $\vec{t}(\sigma)$ is also continued, except in the case of the surface encounter.

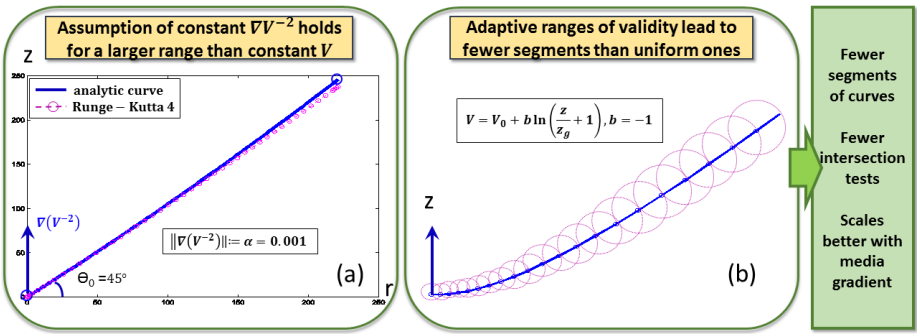


Figure 2: **Efficient media traversal** (Sec. 4.2). **(a)** With constant $\|\nabla V^{-2}\|$, one segment of the analytic curve (blue line) suffices, as compared to the many steps (magenta circles) taken by numerical ray integration. The trajectory traced out by RK-4 visibly diverges from the parabolic curve; this can be improved by further reducing step sizes; **(b)** With a logarithmic sound speed profile, our ray tracer proceeds in segments of adaptive-sized curves, evident from the spheres of validity drawn in magenta.

In the case of the surface encounter, we compute a reflection direction $\vec{t}'$ as the initial direction of the next segment, based on the surface-reflection model.

The ray traversal continues in this segment-by-segment manner until one of three things happens: it reaches outside the media volume; it interacts with surfaces a pre-determined number of times; or the pressure amplitude along the ray drops below a pre-set threshold. To compute the pressure field, the ray tracer finds the set of propagation paths starting from the sound source, the ray segments making up those paths can be queried for any field location.

4.2. Adaptive Media Traversal

Within a smoothly varying medium, the gradient ∇V^{-2} can be assumed to be constant within a certain range of validity. This assumption is valid for naturally occurring media under stable conditions, such as the atmosphere or the ocean. Furthermore, our ray tracer uses the heuristic that the range of validity for a locally constant ∇V^{-2} should be proportional to $\|\nabla V^{-2}\|$, so that the constant gradient remains valid for a relatively small spatial range in areas of great media variation. Such adaptive sizing facilitates accurate traversal of the media without the cost incurred by a uniformly small segment size.

Based on the aforementioned heuristic, we compute the range of validity $d(\mathbf{x})$

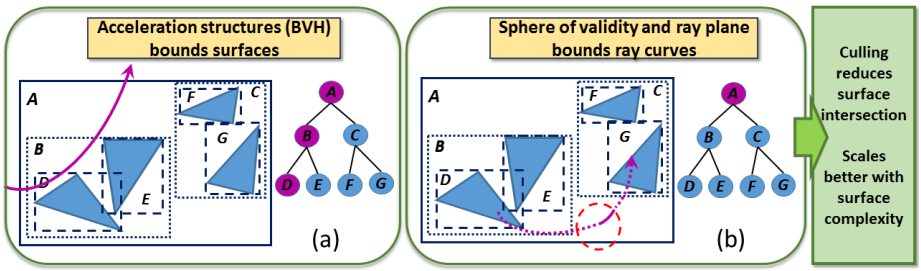


Figure 3: **Efficient surface intersection** (Sec. 4.3). **(a)** A parabolic ray (magenta arrow) is tested against surfaces and their bounding volumes, shown on the left, and the corresponding BVH is shown on the right. Only the magenta nodes are traversed. **(b)** A ray segment is bound by the ray plane and sphere of validity, which provides further culling against scene surfaces. Here only the root node of the BVH is traversed.

by $\delta = \frac{1}{8}\|\nabla V^{-2}\| d^3(\mathbf{x})$, controlled by a global parameter δ. The ray curve given the local $\nabla V^{-2}(\mathbf{x})$ is computed from Eq. 6 and continued until it intersects the *sphere of validity*, centered at the ray origin $\mathbf{x}$ and with a radius $d(\mathbf{x})$. The parabolic curve has a closed-form intersection with this sphere, and the exit point $\mathbf{x}'$ from the sphere is used to sample the media for the next segment.

Given an input media profile $V(\mathbf{x})$, on-the-fly sampling of $\nabla V^{-2}(\mathbf{x})$ can be computed either analytically (if the input profile is available as an analytic function), or by differentiation techniques, such as central differences. Moving media can be approximated by the standard *effective sound speed* approach (adding to or subtracting the wind velocity $w(\mathbf{x})$ from $V(\mathbf{x})$ for upwind or downwind propagation respectively). Particularly for our ray tracer, the on-the-fly media-gradient sampling enables us to handle the vector wind field by adding the wind speed projected onto the ray direction $V(\mathbf{x}) + \vec{w}(\mathbf{x}) \cdot \vec{t}(\mathbf{x})$.

4.3. Handling Boundary Surfaces

For rectilinear ray tracers, the most expensive part of the computation is computing ray intersections with complex surfaces within the scene; many acceleration structures and techniques have been developed to speed up this process. We adopt the hierarchical acceleration structures from rectilinear ray tracers that bound the scene surfaces. The parabolic ray curve as our tracing primitives can be used directly with these structures to cull non-intersecting surfaces.

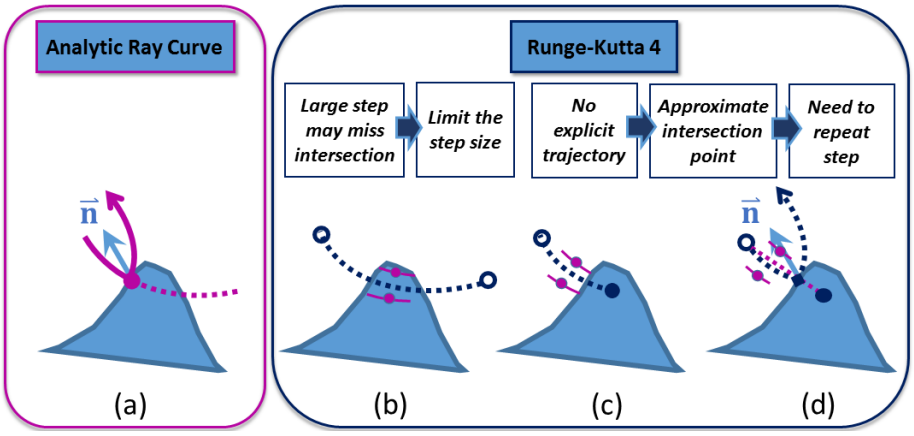


Figure 4: **Ray curve vs. RK4 on surface intersection: (a)** with the ray curve (in magenta), intersection with obstacles (blue block) is efficient and robust; **(b,c,d)** with higher-order numerical methods (blue dotted lines with the 2 mid-points in RK-4 drawn in magenta), challenges include: **(b)** step size is limited, **(c)** intersection point can only be approximated, and **(d)** after the intersection point is approximated (assuming linear trajectory here as shown by the magenta line), step needs to be repeated adjusting for the intersection distance. The poor performance of RK4 on scenes with complex surfaces (Table 3) reflect these challenges.

For scene-surfaces culling, we built a Bounding Volume Hierarchy(BVH)[43], which groups boundary surfaces and bounds them spatially. A BVH is a hierarchical tree structure whose internal nodes are bounding volumes(BVs); for the shape of the nodes, we choose axis-aligned bounding boxes(AABBs). The parabolic ray curve can be intersected with an AABB much faster than it can with all the AABB's enclosed surfaces, and when the ray curve and the AABB are not intersecting, all those surfaces can be culled from further computations. To find the closest intersection with the scene surfaces, the BVH is traversed by each ray curve in a top-down fashion, achieving on average a logarithmic intersection time with regard to the number of surfaces in the scene. The BVH can also be efficiently re-fitted or rebuilt to accommodate dynamic scenes.

Besides spatial culling of the scene surfaces with a BVH, we also perform spatial bounding and culling of ray curves. As we traverse the media (as described in Sec. 4.2), we compute a bounding sphere (the sphere of validity) for each segment as a by-product. The culling efficiency is improved when inter-

secting such a segment with the scene BVH by first intersecting the bounding sphere of the segment with the BVs. And since the parabolic ray is a plane curve, we use the ray plane to reduce the dimension of the intersection test.

At the intersection point with a boundary surface, the interaction can be modeled in a variety of ways: by specular reflection, by Snell's law refraction, or by BRDF-based reflection, depending upon the surface properties. We can then compute the direction of the secondary ray segment accordingly. In addition to modifying the ray direction, a surface impedance model can be used to modify the amplitude and the phase of the pressure along the ray.

5. Results

5.1. Performance Comparison of Ray Models

Ray models are the foundation of many software tools in underwater acoustic simulation[6, 7, 36, 37], and in atmospheric acoustics[38]. The majority of the available tools assume media profiles that vary in at most two dimensions, with the exception of HARPO and HARPA, which computes three-dimensional Hamiltonian ray tracing on clusters. The tools are written in languages with different performance characteristics (e.g. MATLAB, Fortran). We reviewed the relevant data in Sec. 2; in this section we compare our method's performance to that of a numerical ray integration implemented in our code base. Platform-neutral results such as number of ray steps are reported in place of raw running time, in order to highlight the algorithmic difference in comparison.

Existing ray models perform direct numerical integration of the ray equations, using either the Euler method or a higher-order method, such as the fourth order Runga-Kutta (RK4). In these methods, the size of the integration timestep is an important parameter that controls the trade-off between performance and accuracy. Similarly, our ray tracer's balance between performance and accuracy depends on the global parameter δ (Sec. 4.2), which controls step size. As in the numerical-integration methods, smaller step sizes produce more accurate results, at the cost of longer computation time. To highlight this trade-

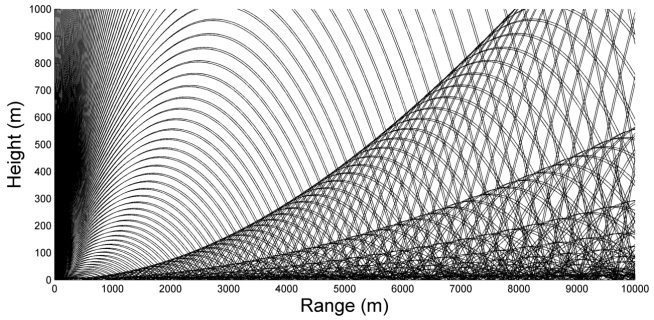


Figure 5: **Downward refracting atmosphere:** computed by the ray curve tracer.

off, we conduct two types of comparisons: same-quality comparisons, for which ray models are tuned to achieve the same level of error, then compared in terms of performance; and same-speed comparisons, in which ray models are run at the same level of performance, then compared in terms of resulting errors.

We measure the accuracy of our ray tracer by the two variables: hit location of the ray at a particular range and the arclength of the entire trajectory. These variables characterize the shape and size of the computed propagation path, which determine the location to which the path contribute and the phase and attenuation along the path for the pressure field computation. To quantify the difference between two sets of results, rt and $\tilde{rt}$, tracing the same set of initial ray directions, we compute the relative error: $E_{rel} = \frac{\|rt-\tilde{rt}\|}{\|rt\|}$.

We select two media profiles as test cases: a downward-refracting atmosphere with a sound-speed gradient of 0.1, and the Munk profile from ocean acoustics. Both profiles have ∇V^{-2} and ∇V in analytic form, which decouple the measurements of ray-tracing accuracy from the accuracy of gradient estimation. We also limit our test to 2D cases with only flat ground as the obstacle, so as to fairly compare our ray tracer's performance to that of the majority of existing ray-based software. The performance improvements for 3D scenes with complex obstacles is expected to be greater, as demonstrated in Section 5.2.

5.1.1. Downward-refracting atmosphere

The first media profile we tested is a linear sound speed profile that leads to a downward-refracting atmosphere: $c(z) = c(0) + 0.1m/s$, where $c(z)$ is the effective sound speed at the height z, and $c(0)$ is the effective sound speed on the ground. We compute the ray-tracing results for the range $0 - 10km$ and the height $1 - 1km$. The ray-tracing results are gathered and compared at the $10km$ mark, a point at which the rays have been reflected up to 25 times. An example of the computed ray trajectories can be seen in Fig. 5, which shows a fan of 201 rays between the elevation angle -90˚and 90˚.

Rel. Error	1e-2	5e-3	1e-3	5e-4	2e-4	
Ray Curves	219	892	2,810	3,972	8,881	(segments/path)
RK4 Steps	467	2,330	4,881	24,588	49,158	(steps/path)

Table 1: **Same-quality comparison**. A fan of rays is traced under a downward-refracting atmosphere to a range of 10km. At each level of relative error in the ray-tracing results, the average number of ray-curve segments per propagation path is compared with the average number of Runga-Kutta 4 steps. The analytic ray curve is able to achieve same level of accuracy with a much smaller tracing cost.

# Ray steps/ RK4 segments	1×10^2	5×10^2	1×10^3	5×10^3	1×10^4
RT rel. error	1.31e-2	2.76e-3	8.04e-4	1.98e-4	4.33e-5
RK4 rel. error	4.35e-2	9.44e-3	4.92e-3	2.14e-3	9.12e-4

Table 2: **Same-speed comparison**. A fan of rays is traced under a downward-refracting atmosphere to a range of 10km. With a comparable number of ray curve segments and Runga-Kutta 4 steps, our ray tracer (RT) is able to achieve lower levels of relative error in ray-tracing results (both hit points and path length) than RK4.

The same-quality and the same-speed comparisons between our ray tracer and the RK4 ray integrator are listed in Table 1 and 2. To achieve the same error level, our ray tracer runs an order of magnitude faster than the numerical integrator. And with the same running time, the numerical integrator produces results with higher levels of error. For this comparison only the computation of ray trajectories are included, the pressure computation is excluded, and the intersection computation is minimum with only a flat ground. The higher number

of ray steps used by the numerical integrator indicate that, if it had to deal with complex intersection tests and additional pressure-related computations at each step, the performance improvements of our ray tracer would be even higher.

We also test the scalability of the ray models with increasing magnitudes of media gradient in Fig. 6 left. The media gradient is increased gradually from 0.04 to 0.4 with 0.04 increments, and the relative error is kept at 2×10^{-4} by tuning the step size for RK 4 and the parameter δ for our ray curve tracer. Our ray tracer is shown to scale better with increased media variations than RK 4.

5.1.2. Munk profile

The second media profile we tested is the Munk profile commonly used in ocean acoustics: $c(z) = 15001.0 + 0.00737[x - 1 + exp(-x)]$, where $c(z)$ is the effective sound speed at the depth z, $z \leq 5000m$. We compute the ray-tracing results for the range $0 - 100km$ and the depth $0 - 5km$. An example of the computed ray trajectories can be seen in Fig. 8, which shows a fan of 21 rays between the elevation angle -13˚and 13˚. The relative errors in the results from the numerical integrator with decreasing step sizes are plotted in Fig. 6; compare this to the same cost-accuracy analysis results for our analytic ray tracer. Again, for this comparison only the computation of ray trajectories and arclength are included, and the pressure computation is excluded.

For this profile we also show the adaptive segment size adopted by our ray tracer by plotting the segment size with depth within the profile (Fig. 7). We trace rays spanning the launch angles between -90˚and 90˚to record the step sizes adopted, and we contrast this with a plot of sound speed of the Munk profile. The Munk profile's local coherence of media properties is taken advantage of by our adaptive ray tracer, to focus computation and improve efficiency. Even with the analytic ray formulation, a uniform step size (instead of an adaptive one) will lead to significant performance degradation.

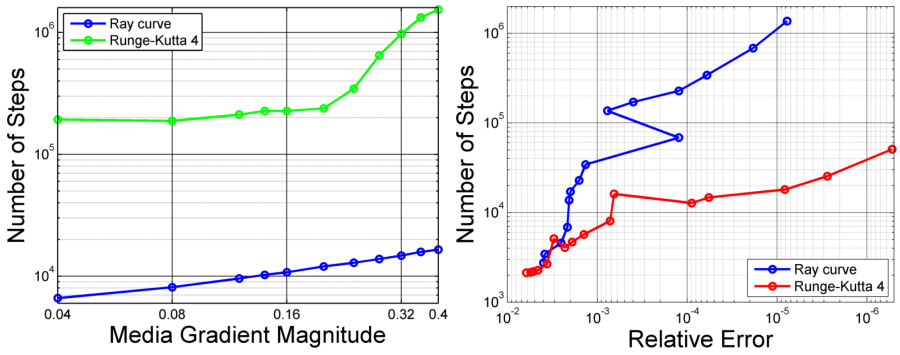


Figure 6: **Left: downward-refracting atmosphere.** As the magnitude of media gradient increases, both RK 4 and ray curve tracer adopt smaller steps to keep the error below 2×10^{-4}; RK 4 shows both higher cost and lower scalability. **Right: Munk profile** The cost-accuracy trade-off of Runga-Kutta 4 and ray curve tracer. No adaptive step/segment size is used for this comparison.

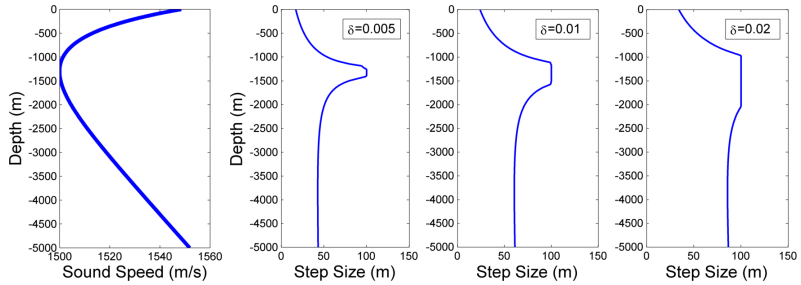


Figure 7: **Adaptive segment size.** The range of validity for each segment of ray curve is controlled by the parameter δ and adapts to local media gradient. **(a)**Munk profile sound speed. **(b,c,d)** Segment sizes plotted with depth for the Munk profile, for $\delta = 0.002, 0.001, 0.0005$, respectively. Without adaptive sizes, the required number of ray segments will be 3 to 10 times higher to achieve comparable accuracy.

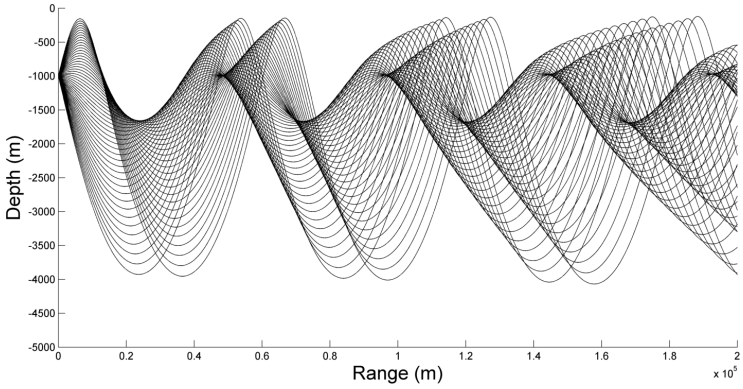


Figure 8: **Munk profile** characteristic ray plot computed by the ray curve tracer.

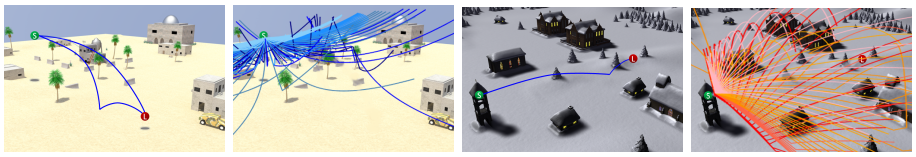


Figure 9: **Outdoor benchmarks** Desert and Christmas village, upward and downward refraction, respectively. For purposes of illustration, the media gradient is exaggerated and the ray path number is kept very small. See Table 3 for scene stats and actual performance numbers when tracing 10k rays up to 3 orders of reflections.

5.2. Outdoor Applications of Analytic Ray Tracer

To show the application of our ray tracer on fully general outdoor scenes, we generate a representative media profile from the stratified-plus-fluctuation model that is widely used in atmospheric acoustics[1]. The atmospheric acoustic index-of-refraction $n = c_0/c$, where c_0 is the reference sound speed, is modeled with a stratified component n_{str} and a fluctuation component n_{flu} so that $n = n_{str}+n_{flu}$. The stratified component follows a logarithmic profile of the altitude z: $n_{str}(z) = c_0/(c_0 + b\ln\left(\frac{z}{z_g}+1\right))$, with parameters n_0, b, and z_g. c_0 is the sound speed at the ground surface, taken as the reference sound speed, and z_g is the roughness length of the ground surface. Typical values for parameter b are 1 m/s for a downward-refracting atmosphere and -1 m/s for an upward-refracting atmosphere. The fluctuation component models the three-dimensional random temperature variation and wind speed turbulence in the atmosphere. The value at position $\mathbf{x}$ can be computed as $n_{flu}(\mathbf{x}) = \sum_i G(\mathbf{k}_i)\cos(\mathbf{k}_i \cdot \mathbf{x} + \varphi_i)$[1], where $\mathbf{k}_i$ is the wave vector describing the *spatial* frequency of the fluctuation, φ_i is a random angle between $[0, 2\pi]$, and $G(\mathbf{k_i})$ is a normalization factor.

The Desert and Christmas benchmarks represent large-volume outdoor acoustic scenes that have complex surface geometry (e.g. terrains and buildings). The surface-primitive count and the volumetric expanse of the two scenes are listed in Table 3. Both scenes are visualized in Fig. 9, and we include illustrative ray paths for both upward-refractive and downward-refractive media conditions.

For each benchmark we trace 10K rays for up to three orders of reflections, and report the overall ray-tracing performance alongside a breakdown of the



Figure 10: **Visualization of ray-surface intersection cost:** brighter colors indicate higher costs. It is shown that the computation concentrates in areas of dense surface geometry and around obstacle silhouettes, illustrating that the spatial culling of ray-surface intersections using BVH is effective. **Left:** Desert **Right:** Christmas village.

Scenes (m^3)	**Surf. count**	**Avg. # segments**	**BV test**	**Surf. test**	**Frame time**	**RK4 time**
Desert ($200 \times 200 \times 50$)	5,000	26.73 per path	31.88 (94%)	2.32 (1.47%)	446 ms	77 s
Christmas ($120 \times 80 \times 50$)	16,000	44.45 per path	34.31 (96.16%)	2.08 (1.22%)	917 ms	128 s

Table 3: **Breakdown of ray-tracing time:** the computation of analytic ray trajectories and ray properties takes less than 0.001% of frame time and thus is omitted here. We report the number of ray-BV and ray-surface intersection tests, and the respective percentage of frame time they take. The frame times of the analytic ray curve tracer are compared to the running time of Runge-Kutta 4 on the same scenes.

running times, as shown in Table 3. We see that the ray intersection computation dominates the ray-tracing cost, and that the majority of the intersections are with Bounding Volumes (BVs). The number of intersections with surface primitives is kept very low, which shows that the spatial culling using BVH is effective. A detailed visualization in Fig. 10 demonstrates this point further by showing that the intersection cost is concentrated in the vicinity of obstacles, and that for the majority of the scenes the ray intersections are resolved with relatively low cost. The analytic ray tracer achieves close to interactive performance for both benchmarks, which is two orders of magnitude faster than RK-4, and it scales well with the complexity in both media and the boundaries.

6. Discussions

Our ray tracer can be extended in ways including but not limited to:

- coupling with flow models and using realistic media profiles as inputs;
- augmenting GA methods to account for inhomogeneous media;
- combining with other methods to form hybrid propagation algorithms.

The analytic ray curve tracer takes as input any general media profile, as long as that profile can be sampled for sound speed and gradient at any spatial location. The input can be an analytic function (e.g. the Monin-Obukhov similarity model[44, 45]) or a set of discrete sample points (e.g. a grid). Numerical methods have been coupled with flow models[46, 47, 48], and the detailed media conditions produced by the flow model as output can then serve as input for propagation simulation. Our ray tracer is also compatible with such a coupling.

GA methods model complex surface interactions[32, 31] such as diffuse reflections and surface BRDFs, assuming homogeneous media. By substituting our curved trajectory computation for the GA assumption of straight-line propagation, we can account for inhomogeneous media while still utilizing GA's surface handling techniques. Such augmented GA solutions can be applied to large indoor spaces such as auditoriums, and to dense outdoor urban areas.

The efficiency of our method makes it a good candidate for forming hybrid schemes with wave-based models. One possibility is to couple the ray curve tracer with Equivalent Source Method(ESM), similar to what Yeh et al.[49] proposed. The ray curve tracer could be focused on high-frequency and long-range propagation while ESM handles the rest. Alternatively, our ray tracer could be used for fast initial evaluations, and costly numerical methods such as FDTD can be applied economically to areas of interest [50, 51, 52].

The analytic ray curve tracer has several limitations. This method is based on the geometric ray model, which is a high-frequency approximation rather than a full wave solution. The efficiency depends on media coherence; if the media is chaotic and lacking spatial coherence, our method's performance will degenerate to about the same level as the numerical ray integration.

7. Conclusion

This paper addresses the challenge of efficient sound propagation in inhomogeneous, moving media and in large outdoor scenes with complex boundary surfaces. Our method adapts the geometric ray models to inhomogeneous media; its key performance improvement upon rectilinear rays is achieved by tracing as primitives analytic ray curves computed from local media gradients. Segments of ray curves are computed by sampling the media gradient on-the-fly, enabling us to account for both inhomogeneous and moving media without needing to precompute explicit cell structures. Acceleration based on BVH is readily adapted to speed up surface intersections of the ray curves, which enables logarithmic scaling with scene complexity and allows for dynamic scenes. The performance of this method is demonstrated in comparison to existing ray models, as well as on scenarios that are prohibitively expensive with existing methods. In addition to being an efficient stand-alone tool, this method also has the potential of complementing other GA and numerical methods.

Acknowledgements

This research was supported in part by ARO Contracts W911NF-12-1-0430, W911NF-13-C-0037,W911NF-14-1-0437, and the National Science Foundation (NSF awards 1320644 and 1305286).

[1] E. M. Salomons, Computational atmospheric acoustics, Kluwer Academic Publishers, 2001.

[2] K. Attenborough, K. M. Li, K. Horoshenkov, Predicting outdoor sound, CRC Press, 2006.

[3] F. B. Jensen, W. A. Kuperman, M. B. Porter, H. Schmidt, Computational ocean acoustics, Springer, 2011.

[4] H. Kuttruff, Room acoustics, CRC Press, 2009.

[5] A. D. Pierce, Acoustics: an introduction to its physical principles and applications, McGraw-Hill, 1981.

[6] B. G. Roberts, Horizontal-gradient acoustical ray-trace program TRIMAIN (1974).

[7] W. H. Watson, R. W. McGirr, RAYWAVE II: A propagation loss model for the analysis of complex ocean environments (1975).

[8] Q. Mo, H. Yeh, D. Manocha, Tracing analytic ray curves for light and sound propagation in non-linear media, arxiv preprint arxiv:1409.2235 (2014).

[9] F. R. DiNapoli, R. L. Deavenport, Theoretical and numerical Green's function field solution in a plane multilayered medium, Journal of Acoustical Society of America 67 (1) (1980) 92–105.

[10] M. K. Myers, G. L. McAninch, Parabolic approximation for sound propagation in the atmosphere, Journal of AIAA 16 (8) (1978) 836–842.

[11] K. E. Gilbert, M. J. White, Application of the parabolic equation to sound propagation in a refracting atmosphere, Journal of Acoustical Society of America 85 (2) (1989) 630–637.

[12] V. E. Ostashev, D. K. Wilson, L. Liu, D. F. Aldridge, N. P. Symons, Equations for finite-difference, time-domain simulation of sound propagation in moving inhomogeneous media and numerical implementation, Journal of Acoustical Society of America 117 (2) (2005) 503–517.

[13] D. K. Wilson, L. Liu, Finite-difference, time-domain simulation of sound propagation in a dynamic atmosphere, Cold Regions Research and Engineering Lab, Hanover NH (No. ERDC/CRREL-TR-04-12.).

[14] T. Nomura, K. Takagi, S. Sato, Finite element simulation of sound propagation concerning meteorological conditions, International Journal for Numerical Methods in Fluids 64 (10-12) (2010) 1296–1318.

[15] C. P. Vendhan, G. C. Diwan, S. K. Bhattacharyya, Finite-element modeling of depth and range dependent acoustic propagation in oceanic waveguides, Journal of Acoustical Society of America 127 (6) (2010) 3319–3326.

[16] C. A. Brebbia, Boundary element methods in acoustics, Computational Mechanics Publications, 1991.

[17] M. Hornikx, R. Waxler, J. Forssén, The extended Fourier pseudospectral time-domain method for atmospheric sound propagation, Journal of Acoustical Society of America 128 (4) (2010) 1632–1646.

[18] M. Hornikx, R. Waxler, The extended Fourier pseudospectral time-domain (pstd) method for fluid media with discontinuous properties, Journal of Acoustical Society of America 18 (04) (2010) 297–319.

[19] J. Hofmann, K. Heutschi, Simulation of outdoor sound propagation with a transmission line matrix method, Applied Acoustics 68 (2) (2007) 158–172.

[20] G. Guillaume, P. Aumond, B. Gauvreau, G. Dutilleux, Application of the transmission line matrix method for outdoor sound propagation modelling-part 1: Model presentation and evaluation, Applied Acoustics 76 (2014) 113–118.

[21] P. Aumond, G. Guillaume, B. Gauvreau, C. Lac, V. Masson, M. Bérengier, Application of the transmission line matrix method for outdoor sound propagation modellingpart 2: Experimental validation using meteorological data derived from the meso-scale model Meso-NH, Applied Acoustics 76 (2014) 107–112.

[22] J. B. Allen, D. A. Berkley, Image method for efficiently simulating small-room acoustics, Journal of Acoustical Society of America 65 (4) (1979) 943–950.

[23] J. Borish, Extension of the image model to arbitrary polyhedra, Journal of Acoustical Society of America 75 (6) (1984) 1827–1836.

[24] M. Vorländer, Simulation of the transient and steadystate sound propagation in rooms using a new combined raytracing/imagesource algorithm, Journal of Acoustical Society of America 86 (1) (1989) 172–178.

[25] M. Taylor, A. Chandak, Q. Mo, C. Lauterbach, C. Schissler, D. Manocha, Guided multiview ray tracing for fast auralization, IEEE Transactions on Visualization and Computer Graphics 18 (11) (2012) 1797–1810.

[26] M. Taylor, A. Chandak, L. Antani, D. Manocha, Resound: interactive sound rendering for dynamic virtual environments, in: Proceedings of 17th ACM international conference on Multimedia, 2009.

[27] T. Funkhouser, I. Carlbom, G. Elko, G. Pingali, M. Sondhi, J. West, A beam tracing approach to acoustic modeling for interactive virtual environments, in: Proceedings of 25th Conference on Computer Graphics and Interactive Techniques, 1998.

[28] A. Chandak, L. Antani, M. Taylor, D. Manocha, FastV: From-point visibility culling on complex models, Computer Graphics Forum 28 (4) (2009) 1237–1246.

[29] M. Bertram, E. Deines, J. Mohring, J. Jegorovs, H. Hagen, Phonon tracing for auralization and visualization of sound, IEEE Visualization.

[30] E. Deines, M. Bertram, J. Mohring, J. Jegorovs, F. Michel, H. Hagen, G. M. Nielson, Comparative visualization for wave-based and geometric acoustics, IEEE Transactions on Visualization and Computer Graphics 12 (5) (2006) 1173–1180.

[31] B. Kapralos, M. Jenkin, E. Milios, Sonel mapping: a probabilistic acoustical modeling method, Building Acoustics 15 (4) (2008) 289–313.

[32] C. Schissler, R. Mehra, D. Manocha, High-order diffraction and diffuse reflections for interactive sound propagation in large environments, ACM Transactions on Graphics 32 (4) (2014) 39.

[33] N. Tsingos, J. D. Gascuel, A general model for the simulation of room acoustics based on hierarchical radiosity, ACM SIGGRAPH sketch.

[34] S. Siltanen, T. Lokki, S. Kiminki, L. Savioja, The room acoustic rendering equation, Journal of Acoustical Society of America 122 (3) (2007) 1624–1635.

[35] R. M. Jones, J. P. Riley, T. M. Georges, HARPO: A versatile three-dimensional hamiltonian ray-tracing program for acoustic waves in an ocean with irregular bottom (1986).

[36] M. B. Porter, The BELLHOP manual and users guide: Preliminary draft (2011).

[37] S. M. Reilly, G. Potty, M. Goodrich, WaveQ3D: Fast and accurate acoustic transmission loss (TL) eigenrays, in littoral environments, for active sonar simulation/stimulation systems (2014).
URL `http://oalib.hlsresearch.com/Rays/USML/usml_frontpage`

[38] R. M. Jones, J. P. Riley, T. M. Georges, HARPA: A versatile three-dimensional hamiltonian ray-tracing program for acoustic waves in the atmosphere above irregular terrain (1986).

[39] U. O. of Naval Research, Ocean acoustics library: Ray models/software (2014).
URL `http://oalib.hlsresearch.com/Rays/index.html`

[40] P. Wang, L. Zhang, V. O. K. Li, A stratified acoustic model accounting for phase shifts for underwater acoustic networks, Sensors 13 (5) (2013) 6183–6203.

[41] B. D. Dushaw, J. A. Colosi, The eigenray acoustic ray tracing code (2013).
URL `http://909ers.apl.washington.edu/twiki/bin/view/Main/EigenRay`

[42] V. Červený, Seismic ray theory, Cambridge University Press, 2005.

[43] V. Havran, Heuristic ray shooting algorithms, Ph.D. thesis, Faculty of Electrical Engineering, Czech Technical University, Prague (2000).

[44] H. A. Panofsky, J. A. Dutton, Atmospheric turbulence. Models and methods for engineering applications, Wiley, 1984.

[45] R. B. Stull, An introduction to boundary layer meteorology, Vol. 13, Springer, 1988.

[46] R. Blumrich, D. Heimann, A linearized Eulerian sound propagation model for studies of complex meteorological effects, Journal of Acoustical Society of America 112 (2) (2002) 446–455.

[47] Z. C. Zheng, W. Li, Numerical stabilities and boundary conditions in time-domain eulerian simulations of acoustic wave propagations with and without background flow, Applied Mathematics and Computation 202 (1) (2008) 146–161.

[48] T. Oshima, M. Imano, Y. Hiraguri, Y. Kamoshida, Linearized euler simulations of sound propagation with wind effects over a reconstructed urban terrain using digital geographic information, Applied Acoustics 74 (12) (2013) 1354–1366.

[49] H. Yeh, R. Mehra, Z. Ren, L. Antani, D. Manocha, M. C. Lin, Wave-ray coupling for interactive sound propagation in large complex scenes, ACM Transactions on Graphics 32 (6) (2013) 165.

[50] T. V. Renterghem, E. M. Salomons, D. Botteldooren, Efficient FDTD-PE model for sound propagation in situations with complex obstacles and wind profiles, Acta Acustica united with acustica 91 (4) (2005) 671–679.

[51] T. V. Renterghem, E. M. Salomons, D. Botteldooren, Parameter study of sound propagation between city canyons with a coupled FDTD-PE model, Applied Acoustics 67 (6) (2006) 487–510.

[52] B. Bergen, B. V. Genechten, T. V. Renterghem, B. Pluymers, D. Vandepitte, D. Botteldooren, W. Desmet, Comparison of numerical prediction techniques for sound propagation in complex outdoor environments, in: Proceedings of 15th International Congress on Sound and Vibration, 2008.

Efficient Solver for Spacetime Control of Smoke

Zherong Pan Dinesh Manocha [1]
Department of Computer Science
the University of North Carolina

We present a novel algorithm to control the physically-based animation of smoke. Given a set of keyframe smoke shapes, we compute a dense sequence of control force fields that can drive the smoke shape to match several keyframes at certain time instances. Our approach formulates this control problem as a PDE-constrained spacetime optimization. In order to compute the locally optimal control forces, we alternatively optimize the velocity fields and density fields using an ADMM optimizer. In order to reduce the high complexity of multiple passes of fluid resimulation during velocity field optimization, we utilize the coherence between consecutive fluid simulation passes. We demonstrate the benefits of our approach by computing accurate solutions on 2D and 3D benchmarks. In practice, we observe up to an order of magnitude improvement over prior optimal control methods.

Additional Key Words and Phrases: Fluid Simulation, Optimal Control

1. INTRODUCTION

Physically-based fluid animations are widely used in computer graphics and related areas. Over the past few years, research in fluid simulation has advanced considerably and it is now possible to generate plausible animations for movies and special effects in a few hours on current desktop systems. In this paper, we mainly deal with the problem of the keyframe-based spacetime control of smoke, a special kind of fluid. Given a set of keyframe smoke shapes, our goal is to compute a dense sequence of control forces such that the smoke can be driven to match these keyframes at certain time instances. This problem is an example of directable animation and arises in different applications, including special effects [Rasmussen et al. 2004] (to model a character made of liquid) or artistic animations [Angelidis et al. 2006] (to change the moving direction of the smoke plume). Some of these control techniques, such as [Nielsen and Bridson 2011], are used in the commercial fluid software.

In practice, the keyframe-based control of fluids is still regarded as a challenging problem. Unlike fluid simulation, which deals with the problem of advancing the current fluid state to the next one by time integrating the Navier-Stokes equations, a fluid controller needs to consider an entire sequence of fluid states that results in a high dimensional space of possible control forces. For example, to control a 3D smoke animation discretized on a uniform grid at resolution 64^3 with 60 timesteps, the dimension of the resulting space of control forces can be as high as 10^8. The problem of computing the appropriate control force sequence in such a high dimensional space can be challenging for any continuous optimization algorithm. Furthermore, the iterative computation of control forces would need many iterations, each of which involves solving a 2D or 3D fluid simulation problem that can take hours on a desktop system.

Fluid control problems have been well studied in computer graphics and animation. At a broad level, prior techniques can be classified into proportional-derivative (PD) controllers and optimal controllers. PD controllers [Fattal and Lischinski 2004; Shi and Yu 2005] guide the fluid body using additional ghost force terms that are designed based on a distance measure between the current fluid shape and the keyframe. On the other hand, optimal controllers [Treuille et al. 2003; McNamara et al. 2004] formulate the problem as a spacetime optimization over the space of possible control forces constrained by the fluid governing equations, i.e., the Navier-Stokes equations. The objective function of this optimization formulation consists of two terms: The first term requires the fluid shape to match the keyframe shape at certain time instances, while the second term requires the control force magnitudes to be as small as possible.

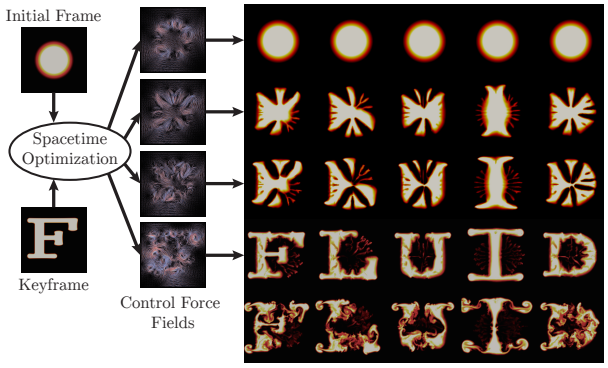


Fig. 1: Given the keyframe, we use spacetime optimization to compute a dense sequence of control force fields, matching a smoke ball to the word "FLUID". We highlight the control force fields. Five such animations are generated, at resolution 128^2 with 40 timesteps. Each of these optimization computations take about half an hour on a desktop PC, and is about 17 times faster than conventional gradient-based optimizer.

Optimal controllers are advantageous over PD controllers in that they are less sensitive to the choice of the parameters and they search for the control forces with the smallest possible magnitude, which usually provides smoother keyframe transitions as well as satisfies the fluid dynamic constraints. Treuille et al. [2003] and McNamara et al. [2004] use a simple gradient-based optimizer to search for control forces. This method is easy to implement, but can be computationally inefficient since each gradient evaluation needs to solve a fluid simulation problem. Such repeated fluid simulations slow down the overall performance. In the original work [Treuille et al. 2003], this issue is alleviated by reducing the dimension of the search space using a set of control force templates. However, this treatment also restricts the amount of fluid-like details in the controlled animations.

We present a new, efficient optimization algorithm for controlling smoke. Our approach exploits the special structure of the Navier-Stokes equations discretized on a regular staggered grid, and solves the optimization problem by finding the stationary point of the first order optimality (Karush-Kuhn-Tucker) conditions [Nocedal and Wright 2006]. However, unlike prior methods [Treuille et al. 2003; McNamara et al. 2004] that only solve for the primal variables, we maintain both the primal and dual variables (i.e., the

Lagrangian multipliers). By maintaining the additional dual variables, we can iteratively update our solution without requiring it to satisfy the Navier-Stokes equations exactly in each iteration. Therefore, we alternatively update the velocity fields and control force fields using the alternating direction method of multiplier (ADMM) [Boyd et al. 2011]. In order to update the control force fields efficiently without fluid resimulation, we present a nonlinear multigrid solver: the full approximation scheme (FAS) [Brandt and Livne 2011]. Using a novel spacetime smoothing operator that takes all the timesteps into consideration, our multigrid can converge within a number of iterations independent of the grid resolution and the number of timesteps.

We have evaluated our approach on several benchmarks. Our benchmarks vary in terms of the grid resolution, the number of timesteps, and the control force regularization parameter. We highlight results with up to 60 timesteps at the resolution of 64^3. Without using force templates, we allow each component of the velocity field to be controlled. In practice, our algorithm can compute a convergent animation in less than 50 iterations, and the overall runtime performance is about an order of magnitude faster than a gradient-based quasi-Newton optimizer [Nocedal and Wright 2006] for similar accuracy. An example of achieved smooth transitions between keyframes is illustrated in Figure 1.

2. RELATED WORK

In this section, we give a brief overview of prior techniques for fluid simulation, multigrid solvers and animation control algorithms.

Fluid simulation has been an active area of research in both computer graphics and computational fluid dynamics. The simulation of fluid is typically solved by a discretized time integration of the Navier-Stokes equations or their equivalent forms. At a broad level, prior fluid simulators can be classified into Lagrangian or Eulerian solvers according to the discretization of the convection operator. In order to model smoke and fire, a purely Eulerian solver [Fedkiw et al. 2001] is the standard technique. In terms of free-surface flow, hybrid Lagrangian-Eulerian representation [Zhu and Bridson 2005] has been widely used in computer graphics. In our work, we confine ourselves to the control of fluids without free-surface, i.e., smoke or fire. We use [Harlow and Welch 1965] as our underlying fluid simulator.

Multigrid solvers are widely used for fluid simulation. Multigrid is a long-standing concept that has been widely used to efficiently solve linear systems discretized from elliptic partial differential equations (see [Brandt and Livne 2011]). This idea has been successfully applied to fluid simulation [Chentanez et al. 2007; Chentanez and Müller 2011; Zhang and Bridson 2014] to find the solenoidal component of the velocity field. In terms of PDE-constrained optimization and control theory, the idea of multigrid acceleration has been extended to the spatial temporal domain. Borzi and Griesse [2005] proposed a semi-coarsening spacetime multigrid to control the time-dependent reaction-diffusion equation. Hinze et al. [2012] used a spacetime multigrid to solve the velocity tracking problem governed by the Navier-Stokes equations. The nonlinear multigrid used in our method is closely related to [Hinze et al. 2012], which also solves a spacetime optimization problem. However, our method is a spatial-only multigrid with the smoothing operator that handles all the timesteps at once. Moreover, unlike [Hinze et al. 2012], which solves a velocity tracking problem, our formulation is a density tracking problem. Therefore, the multigrid is used as a subproblem solver in our ADMM optimization framework.

Fluid control problems tend to be challenging and computationally demanding. Compared to other kinds of animations, e.g., character locomotion [Mordatch et al. 2012], the configuration space of fluid body is of much higher dimension. Prior work in this area can be classified into two categories: PD controllers [Fattal and Lischinski 2004; Shi and Yu 2005] and optimal controllers [Treuille et al. 2003; McNamara et al. 2004]. PD controllers compute the control forces by considering only the configuration of the fluid at the current and next time instance. For example, in [Shi and Yu 2005], a PD controller is used where the control forces are made proportional to the error between the current fluid shape and the target keyframe shape. Similar ideas are used for controlling smoke [Fattal and Lischinski 2004] and liquid [Shi and Yu 2005; Raveendran et al. 2012]. In contrast, optimal controllers search for a sequence of control forces that minimize an objective function. Prior methods [Treuille et al. 2003; McNamara et al. 2004] typically solve spacetime optimization over a high-DOF search space to compute such control forces. Recently, these two methods have been combined [Pan et al. 2013] by first optimizing for the fluid shape at each keyframe and then propagating the changes to the neighboring timesteps. Fluid control can also be achieved by combining or modifying the results of existing fluid simulation data [Raveendran et al. 2014] or guiding fluid using a designed or captured low-resolution animation [Nielsen and Bridson 2011; Nielsen and Christensen 2010; Gregson et al. 2014].

Our algorithm is based on spacetime optimization, similar to [Treuille et al. 2003; McNamara et al. 2004]. In order to solve this optimization problem, we use the ADMM method [Boyd et al. 2011]. This solver has also been previously used in [Gregson et al. 2014] for fluid capturing and guiding. However, our method differs from these previous works in three ways. First, unlike [McNamara et al. 2004], whose method solves for primal variables only, we use a primal-dual formulation. This treatment does not require the Navier-Stokes equations to be satisfied exactly in each iteration of optimization. Moreover, to solve fluid tracking problem, [Gregson et al. 2014] uses ADMM method to address incompressible constraints in one timestep. While we use ADMM method for spacetime optimization taking all timesteps into consideration. Finally, [Gregson et al. 2014] considers the linear solenoidal constraints as hard constraints. In our work, we take into account the entire nonlinear fluid governing equations as the hard constraints, resulting in a nonlinear subproblem that is solved using a multigrid method.

3. FLUID CONTROL

In this section, we formulate the spacetime fluid control problem based on fluid dynamics (Section 3.1) and optimal control theory (Section 3.2). The set of symbols used throughout the paper can be found in Figure 3, and the subscript i is the timestep index. In general, we are dealing with a dynamic system whose configuration space is denoted as s_i at physical time $i\Delta t$. Consecutive configurations s_i and s_{i+1} are related by the partial differential equation denoted as the function f: $s_{i+1} = f(s_i, u_i, \Delta t)$, where u_i is the control input. An optimal controller computes a set of control inputs $\{u_i | i = 0, \cdots, N-1\}$ that minimize the objective function denoted as function $E(s_0, \cdots, s_N)$. The overall optimal control problem is specified using the pair of functions f and E. In the case of smoke control problems, f is a discretization of the Navier-Stokes equations, and E measures the difference between the smoke and keyframe shapes at certain time instances.

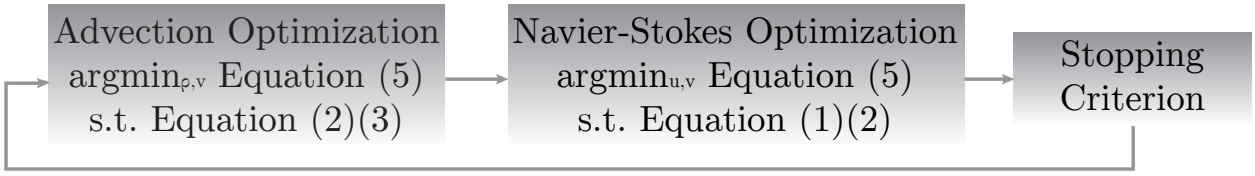


Fig. 2: Our Algorithm Pipeline: We use the ADMM method to decompose the problem into two subproblems: Advection Optimization (AO) which only considers passive advection as hard constraints; and Navier-Stokes Optimization (NSO) which only considers fluid dynamics as hard constraints.

Symbol	Meaning
v_i	velocity field
u_i	ghost force field
ρ_i	density or dye field
p_i	pressure field
s_i	state vector
$\mathbf{Adv}$	self advection operator
$\mathbf{A}$	passive advection operator
Δt	timestep size
N	number of timesteps

(a) Symbols for fluid dynamic system

Symbol	Meaning
v_i^*	slack velocity field
λ_i	augmented Lagrangian multiplier field
μ_i	Lagrangian multiplier for passive advection
γ_i	Lagrangian multiplier for $\nabla \cdot v_i^* = 0$
$\bar{p}_i$	Lagrangian multiplier for $\nabla \cdot v_i = 0$
K	penalty coefficient for constraint $v_i = v_i^*$
r	regularization coefficient for u_i
c_i	indicator of keyframe at timestep i
C_i	metric measure for density field
$\mathbf{R}$	restriction operator of FAS
$\mathbf{P}$	prolongation operator of FAS
$\mathbf{S}$	smoothing operator of FAS
$\mathbf{Q}$	solenoidal projection operator

(b) Symbols for spacetime optimization

Fig. 3: Symbol table.

3.1 Fluid Dynamic System

In our work, we restrict ourselves to the control of incompressible fluids without a free surface. Fluids such as smoke and fire, which are commonly used in movies and animations, fall into this category. We omit viscous terms for brevity. Small viscosity can be handled by a slight modification to f, which does not increase the complexity of our algorithm. Following [Harlow and Welch 1965; Pavlov et al. 2011], we discretize the velocity-vorticity version of the Navier-Stokes equations using finite difference scheme and backward Euler time integrator for advection. Our configuration space $s_i = \left({v_i}^T \, {p_i}^T \, {\rho_i}^T\right)^T$ concatenates the velocity field v_i, the kinetic pressure field p_i, and the density or dye field ρ_i. These scalar and vector fields are discretized on a staggered grid, which has been widely used by previous works such as [Fedkiw et al. 2001]. The transfer function f under such discretization can be represented as:

$$\frac{v_{i+1} - v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] = u_i - \nabla p_{i+1}, \tag{1}$$

$$\nabla \cdot v_{i+1} = 0, \tag{2}$$

$$\rho_{i+1} = \mathbf{A}\left[\rho_i, v_i\right], \tag{3}$$

where the self-advection operator $\mathbf{Adv}\left[\bullet\right]$ is a discretization of the quadratic operator $\nabla \times \bullet \times \bullet$ and we assume constant unit fluid density. The pressure field p_{i+1} is identified with the Lagrangian multiplier of the divergence free constraints $\nabla \cdot v_{i+1} = 0$. Finally, the operator $\mathbf{A}\left[\bullet, \bullet\right]$ is the passive scalar advection operator discretized as: $\rho_{i+1} = \mathbf{e}^{\mathbf{A}(v_i)\Delta t}\rho_i$, where matrix $\mathbf{A}(v_i)$ is the second order upwinding stencil [Leonard 1979]. By approximating the matrix exponential using Taylor series, the advection operator can be defined as:

$$\mathbf{A}\left[\rho_i, v_i\right] = \sum_{k=0}^{\infty} \frac{\Delta t^k}{k!} \mathbf{A}(v_i)^k \rho_i. \tag{4}$$

When k tends to infinity, this upwinding advection operator is unconditionally stable since $\mathbf{A}(v_i)$ is skew-symmetric, so that $\mathbf{e}^{\mathbf{A}(v_i)\Delta t}$ is an orthogonal matrix and $\|\rho_{i+1}\| = \|\rho_i\|$. In practice, we truncate k to a finite value. Specifically, we set k adaptively to be the smallest integer satisfying $\frac{\Delta t^k}{k!}\mathbf{A}(v_i)^k \rho_i < 1e^{-5}$. Although this operator is computationally more expensive than the widely used semi-Lagrangian operator, it generates smoother controlled animations with large timestep size, as shown in Figure 4. This is useful when fewer timesteps are used to reduce the runtime cost.

3.2 Spacetime Optimization

The optimal control of the dynamic system, discussed in Section 3.1, can be formulated as a spacetime optimization over the configuration trajectory $s_0, \cdots, s_N$. Our objective function is similar to the ones proposed in prior works [Treuille et al. 2003; McNamara et al. 2004] that try to match ρ_i to a set of keyframes ρ_i^* while minimizing the magnitude of control forces u_i. The overall optimization problem can be formulated as:

$$\begin{aligned} \underset{u_i}{\mathbf{argmin}} \quad & E = \frac{1}{2}\sum_{i=0}^{N} c_i \|\rho_i - \rho_i^*\|^2 + \frac{r}{2}\sum_{i=0}^{N-1} \|u_i\|^2 \\ \mathbf{s.t.} \quad & s_{i+1} = f(s_i, u_i, \Delta t), \end{aligned} \tag{5}$$

where c_i is 1 if there is a keyframe ρ_i^* at frame i and 0 otherwise. r is the regularization coefficient of the control forces.

Treuille et al. [2003] and McNamara et al. [2004] solve this optimization by eliminating the transfer function f and plugging them into the objective function. Although this reformulation simplifies the problem into an unconstrained optimization, their new objective function takes a much more complex form, which is a long chain of function compositions. To minimize the new objective function, Treuille et al. [2003] and McNamara et al. [2004] use a general-purpose gradient-based optimizer. A typical gradient-based optimizer such as the Quasi-Newton method [Byrd et al. 1995] requires repeated gradient calculation to approximate the Hessian matrix and performs line search to compute the stepsize. Each such gradient calculation requires a fluid resimulation, which becomes the major bottleneck in their algorithm.

3.3 Our Approach

Prior methods require that the solution computed during each iteration should satisfy the Navier-Stokes equations exactly, i.e., is a feasible solution. As a result, each iteration takes considerable running time. In practice, this requirement can be overly conservative because we only need to ensure that the final computed solution at the end of the algorithm is feasible. Thus, we can relax this requirement during the intermediate steps, and only need to ensure that the final solution lies in the feasible domain. This is a well-known idea and has been used by many other numerical optimization algorithms such as the interior point method [Nocedal and Wright 2006].

Based on this observation, we design a new optimization algorithm illustrated in Figure 2. We first notice that our objective function is essentially constrained by two kinds of partial differential equations: the passive advection (Equation 3) governing the time evolution of the density field ρ_i; and the incompressible Navier-Stokes (Equation 1 and Equation 2) governing the time evolution of the velocity field v_i. We introduce a set of slack variables to break these two kinds of constraints into two subproblems: Advection Optimization (AO) is constrained only by Equation 3 and Navier-Stokes Optimization (NSO) is constrained only by Equation 1 and Equation 2. In order to solve the Advection Optimization (Section 4.1), we use a fixed point iteration defined for its KKT conditions. For the Navier-Stokes Optimization (Section 4.2), we update our solution using the full approximation scheme (FAS) to avoid repeated fluid resimulations. This leads to considerable speedup over prior methods, not only because of the fast convergence of our multigrid solver, but also because the multigrid solver allows warm-starting so that we can utilize coherence between consecutive iterations. In contrast, previous methods use fluid resimulations, which always solve Navier-Stokes equations from scratch, and solve them exactly.

4. SPACETIME OPTIMIZATION

In this section, we present our novel algorithm to solve Equation 5. By introducing a series of slack variables v_i^*, we can decompose the overall optimization problem into two subproblems and reformulate Equation 5 as:

$$\begin{aligned} \underset{u_i}{\mathbf{argmin}} \quad & \frac{1}{2}\sum_{i=0}^{N} c_i \|\rho_i - \rho_i^*\|^2 + \frac{r}{2}\sum_{i=0}^{N-1}\|u_i\|^2 + \\ & \lambda_i^T(v_i - v_i^*) + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2 \qquad (6) \\ \mathbf{s.t.} \quad & \frac{v_{i+1}-v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] = u_i - \nabla p_{i+1} \\ & \rho_{i+1} = \mathbf{A}\left[\rho_i, v_i^*\right], \quad \nabla \cdot v_i = 0, \end{aligned}$$

where we added the augmented Lagrangian term $\lambda_i^T(v_i - v_i^*)$ and the penalty term $\frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2$. This kind of optimization can be solved efficiently using the well-known alternating direction method of multipliers (ADMM) [Boyd et al. 2011]. Specifically, in each iteration of our algorithm, we first fix v_i, p_i and solve for v_i^*. This subproblem is denoted as the Advection Optimization (AO) because the PDE constraints are just passive advection of the density field ρ_i. We then fix v_i^* and solve for v_i, p_i. We denote this subproblem as the Navier-Stokes Optimization (NSO), constrained by the incompressible Navier-Stokes equations. The final step is to adjust λ_i according to the constraint violation as: $\lambda_i = \lambda_i + K\beta(v_i - v_i^*)$ where β is a constant parameter.

The idea of introducing slack variables to relax hard constraints has been used in several previous works, e.g., for fluid tracking [Gregson et al. 2014] and projective dynamic simulation [Narain et al. 2016]. The advantage of decomposing the problem up is that we can derive simple and effective algorithms to solve each subproblem. Our algorithm directly solves the first order optimality (KKT) conditions of both problems. To solve the AO subproblem, we introduce a fixed point iteration in Section 4.1, while for the NSO subproblem, which is the bottleneck of the algorithm, we introduce the nonlinear multigrid solver (FAS) in Section 4.2.

4.1 Advection Optimization

The goal of solving the AO subproblem is to find a sequence of velocity fields v_i^* to advect ρ_i so that it matches the keyframes, assuming that these v_i^* are uncorrelated. By dropping terms irrelevant to v_i^* from Equation 6, we get a concise formulation for the AO subproblem:

$$\begin{aligned} \underset{v_i^*}{\mathbf{argmin}} \quad & \frac{1}{2}\sum_{i=0}^{N}(\rho_i - \rho_i^*)^T C_i(\rho_i - \rho_i^*) + \qquad (7) \\ & \frac{K}{2}\sum_{i=0}^{N-1}\|v_i + \lambda_i/K - v_i^*\|^2 \\ \mathbf{s.t.} \quad & \rho_{i+1} = \mathbf{A}\left[\rho_i, v_i^*\right] \quad \nabla \cdot v_i^* = 0, \end{aligned}$$

where we can absorb the augmented Lagrangian term $\lambda_i^T(v_i - v_i^*)$ by setting: $v_i \leftarrow v_i + \lambda_i/K$.

Due to the inherent nonlinearity and ambiguity in the advection operator, an AO solver is prone to falling into local minimum, leading to trivial solutions. We introduce two additional modifications to Equation 7 to avoid these trivial solutions. First, we replace the scalar coefficient c_i with a matrix C_i which could be used to avoid the problem of a zero gradient if the keyframe ρ_i^* is far from the given density field ρ_i. Similar to [Treuille et al. 2003; Fattal and Lischinski 2004], we use the idea of Gaussian Pyramid [Adelson et al. 1984] and take $C_i = c_i\Sigma_k G_k^T G_k$ to be a series of Gaussian filters G_k with receding support. Specifically, G_k has a standard deviation of $\sigma(G_k) = 2\sigma(G_{k-1})$. The Gaussian Pyramid makes our method almost resolution invariant, since any local error in the density field will always lead to a non-zero gradient value at every point in the grid domain. We also introduce additional solenoidal constraints on v_i^*. Note that this term does not alter the optima of Equation 6 since $v_i = v_i^*$ on convergence. However, it prevents the optimizer from creating or removing densities in order to match the keyframe, which is a tempting trivial solution.

We solve this optimization via a fixed point iteration derived from its KKT conditions. To derive this system we introduce Lagrangian multipliers μ_i for each advection equation $\rho_{i+1} = \mathbf{A}\left[\rho_i, v_i^*\right]$ and γ_i for the solenoidal constraints, giving a Lagrangian function:

$$\begin{aligned} \mathcal{L} = \frac{1}{2}\sum_{i=0}^{N}(\rho_i - \rho_i^*)^T C_i(\rho_i - \rho_i^*) + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2 + \\ \sum_{i=0}^{N-1}\mu_i^T(\rho_{i+1} - \mathbf{A}\left[\rho_i, v_i^*\right]) + \gamma_i^T \nabla \cdot v_i^*. \end{aligned}$$

After taking the derivative of the above Lagrangian against ρ_i, v_i^* (primal variables) and μ_i, γ_i (dual variables), respectively, we get the following set of KKT conditions for $0 \le i \le N$:

$$\begin{aligned} \mu_{i-1} &= \frac{\partial \mathbf{A}\left[\rho_i, v_i^*\right]}{\partial \rho_i}^T \mu_i - C_i(\rho_i - \rho_i^*) \\ v_{i-1}^* &= \mathbf{Q}(v_{i-1} + \frac{\partial \mathbf{A}\left[\rho_{i-1}, v_{i-1}^*\right]}{\partial v_{i-1}^*}^T \frac{\mu_{i-1}}{K}) \qquad (8) \\ \rho_{i+1} - \mathbf{A}\left[\rho_i, v_i^*\right] &= 0, \quad \nabla \cdot v_i^* = 0, \end{aligned}$$

where we set $\mu_{-1} = \mu_N = 0$ to unify the index, and we have replaced γ_i with a solenoidal projection operator $\mathbf{Q}$. This actually defines a fixed point iteration where we can first update ρ_i in a forward pass and then update μ_i, v_i in a backward pass. This is closely related to the adjoint method [McNamara et al. 2004], which also

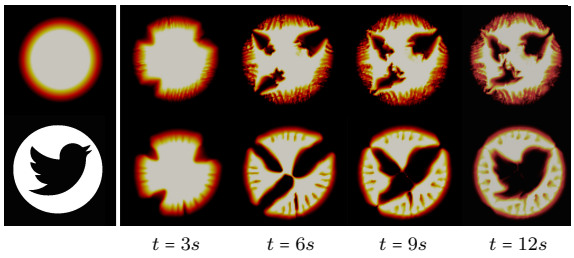


Fig. 4: We tested the fixed point iteration Equation 8 using different advection operator $\mathbf{A}[\bullet,\bullet]$ to deform an initially circle-shaped smoke (top left) into the bird icon (bottom left). The AO subproblem solved using the semi-Lagrangian operator involves lots of popping artifacts (top row). The upwinding operator in Equation 4 does not suffer from such problems (bottom row).

takes a forward-backward form. Unlike [McNamara et al. 2004] which then solves v_i^* using quasi-Newton method, a fixed point iteration is much simpler to implement, and a general-purpose optimizer is not needed. The most costly step in applying Equation 8 is the operator $\mathbf{Q}$ where we use conventional multigrid Poisson solver [Brandt and Livne 2011].

A pseudo-code of our AO solver is given in Algorithm 1. We have introduced two additional strategies to guarantee the convergence of the fixed point iteration. First, we determine the order of Taylor expansion (k in Equation 4) in the forward pass (Line 11) and fix this k value in the backward pass to ensure that the order of expansion is fixed within each gradient estimation. Moreover, we introduce a simple line search strategy (Line 14 to Line 28).

In the above derivation, since we do not exploit any structure in the operator $\mathbf{A}[\rho_i, v_i^*]$, basically any advection operator other than Equation 4, such as semi-Lagrangian, could be used as long as its partial derivatives against ρ_i, v_i^* are available. Empirically, however, Equation 4 generally gives smoother animations especially under large timestep size. This is because the semi-Lagrangian operator can jump across multiple cells when performing backtracking, and the density value changes in these cells are ignored. As a result, the semi-Lagrangian operator suffers from popping artifacts as illustrated in Figure 4, while our operator (Equation 4), being purely grid-based, does not exhibit such problem. Unlike [Treuille et al. 2003], where these popping artifacts can be alleviated by constraining control force fields to a small set of force templates, we allow every velocity component to be controlled. In this case, the use of our new advection operator is recommended.

4.2 Navier-Stokes Optimization

Complementary to Section 4.1, the goal of the Navier-Stokes Optimization is to enforce the correlation between v_i given the sequence of guiding velocity fields v_i^*. The optimization takes the following form:

$$\begin{aligned} \underset{v_i}{\mathbf{argmin}} \quad & \frac{r}{2}\sum_{i=0}^{N-1}\|u_i\|^2 + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^*\|^2 \\ \mathbf{s.t.} \quad & \frac{v_{i+1}-v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] = u_i - \nabla p_{i+1} \\ & \nabla \cdot v_i = 0. \end{aligned}$$

This subproblem is the bottleneck of our algorithm, for which a forward-backward adjoint method similar to Equation 8 requires solving the Navier-Stokes equations exactly in the forward pass. To avoid this costly solve, we update primal as well as dual variables from the previous iteration using a unified algorithm. In the same way as in Section 4.1, we derive the KKT conditions and assemble them into a set of nonlinear equations:

$$f = \begin{pmatrix} \frac{K}{r}(v_i - v_i^*) + \frac{\partial u_i}{\partial v_i}^T u_i + \frac{\partial u_{i-1}}{\partial v_i}^T u_{i-1} + \nabla \bar{p}_i \\ \nabla \cdot v_i \\ \frac{v_{i+1}-v_i}{\Delta t} + \mathbf{Adv}\left[v_{i+1}\right] - u_i + \nabla p_{i+1} \\ \nabla \cdot u_i \end{pmatrix} = 0, \forall i \quad (9)$$

where the partial derivatives are $\frac{\partial u_i}{\partial v_i} = -\frac{I}{\Delta t}$, $\frac{\partial u_{i-1}}{\partial v_i} = \frac{I}{\Delta t} + \frac{\partial \mathbf{Adv}[v_i]}{\partial v_i}$, and the additional variable $\bar{p}_i$ is the Lagrangian multiplier for the solenoidal constraint: $\nabla \cdot v_i = 0$. We refer readers to Appendix A for the derivation of Equation 9. In summary, we have to solve for the primal variables u_i, v_i as well as the dual variables $p_i, \bar{p}_i$. Unlike Equation 8, however, we do not differentiate these

Algorithm 1 The Fixed Point Iteration: This is used to solve the AO subproblem. The algorithm consists of a forward sweep that updates the density fields ρ_i and a backward sweep that updates μ_i and v_i^*. Here, we introduce a line-search parameter α to ensure algorithm convergence, where v_i^{**} stores the tentative solution.

Input: Initial v_i, ρ_0, $\alpha \in (0,1]$, and keyframes ρ_i^*
Output: Fixed point solution v_i^*, μ_i

1: $E \leftarrow \infty$
2: **for** $i = 0, \cdots, N-1$ **do**
3: ▷ Initialization
4: $v_i^* \leftarrow v_i$
5: $v_i^{**} \leftarrow v_i$
6: **end for**
7: **while** not converged **do**
8: ▷ Forward pass
9: **for** $i = 1, \cdots, N$ **do**
10: ▷ Find primal variables ρ
11: Find smallest k such that $\frac{\Delta t^k}{k!}\mathbf{A}(v_{i-1}^{**})^k \rho_{i-1} < 1e^{-5}$
12: $\rho_i \leftarrow \mathbf{A}[\rho_{i-1}, v_{i-1}^{**}]$
13: **end for**
14: ▷ Ensure function value decrease
15: $E^{new} \leftarrow \frac{1}{2}\sum_{i=0}^{N}\|\rho_i - \rho_i^*\|_{C_i}^2 + \frac{K}{2}\sum_{i=0}^{N-1}\|v_i - v_i^{**}\|^2$
16: **if** $E^{new} < E$ **then**
17: $E \leftarrow E^{new}$
18: **for** $i = 1, \cdots, N-1$ **do**
19: $v_i^* \leftarrow v_i^{**}$
20: **end for**
21: increase α
22: **else**
23: **for** $i = 1, \cdots, N-1$ **do**
24: $v_i^{**} \leftarrow v_i^*$
25: **end for**
26: decrease α
27: **goto** *Line* 8
28: **end if**
29: ▷ Backward pass
30: set $\mu_{-1} \leftarrow 0$, $\mu_N \leftarrow 0$
31: **for** $i = N, \cdots, 1$ **do**
32: ▷ Find dual variables μ
33: $\mu_{i-1} \leftarrow \frac{\partial \mathbf{A}[\rho_i, v_i^*])}{\partial \rho_i}^T \mu_i - C_i(\rho_i - \rho_i^*)$
34: ▷ Find primal variables v
35: $v_{i-1}^{**} \leftarrow (1-\alpha)v_{i-1}^* + \alpha\mathbf{Q}(v_{i-1} + \frac{\partial \mathbf{A}[\rho_{i-1}, v_{i-1}^*])}{\partial v_{i-1}^*}^T \frac{\mu_{i-1}}{K})$
36: **end for**
37: **end while**

two sets of variables and solve for them by iteratively bringing the residual f to zero.

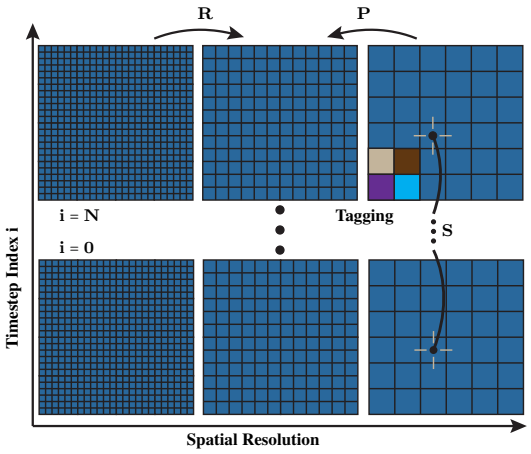


Fig. 5: A 2D illustration of our full approximation scheme (FAS). We use semi-coarsening only in the spatial direction (horizontal), with each finer level doubling the grid resolution. We use trilinear interpolation operators for $\mathbf{P}, \mathbf{R}$ and tridiagonal SCGS smoothing for $\mathbf{S}$, which solves the primal variables v_i, u_i (defined on faces as short white lines) and dual variables $p_i, \bar{p}_i$ (defined in cell centers as black dots) associated with one cell across all the timesteps (vertical) by solving a block tridiagonal system. The solve can be made parallel by the $8-color$ tagging in 3D or $4-color$ tagging in 2D.

To this end, we develop a full approximation scheme (FAS), which is a geometric multigrid algorithm designed for solving a nonlinear system of equations as illustrated in Figure 5. The multigrid solver is a classical tool originally used for solving linear systems induced from elliptical PDEs. Recently, the multigrid solver has also been used for solving nonlinear problems such as PDE-constrained optimization. For example, Hinze et al. [2012] used a spacetime multigrid to track and stabilize the velocity field of incompressible Navier-Stokes flow, and Borzi et al. [2005] used a semi-coarsening multigrid to control a reaction-diffusion flow. The nonlinear multigrid in our formulation can be considered as a combination of these two approaches: a semi-coarsening multigrid to control the incompressible Navier-Stokes flow. Given that we want to solve a density tracking problem, instead of a velocity tracking problem, we use the spatial FAS multigrid as our NSO subproblem solver. We refer the readers to [Brandt and Livne 2011] for a detailed introduction and briefly review the core idea in the following parts.

4.3 Full Approximation Scheme

Since Equation 9 is valid for all the indices i, we concatenate all the timestep-related variables and discard subscripts for convenience. A multigrid solver works on a hierarchy of grids in descending resolutions. In each FAS iteration, it refines the solution $(v, \bar{p}, u, p)$ by reducing the residual $f(v, \bar{p}, u, p)$. Since different components of the residual can be reduced most effectively at different resolutions, the multigrid solver downsamples the residual to the appropriate resolutions and then upsamples and combines their solutions. With properly defined operators introduced in this section, our multigrid algorithm can generally achieve a linear rate of error reduction, which is optimal in the asymptotic sense.

To adopt this idea to solve Equation 9, we introduce a hierarchy of spatial grids $(v^h, \bar{p}^h, u^h, p^h)$, where h is the cell size. We use semi-coarsening in spatial direction only where every coarser level doubles the cell size. We denote the coarser level as $(v^{2h}, \bar{p}^{2h}, u^{2h}, p^{2h})$, and use the simple FAS-VCycle(2,2) iteration to solve the nonlinear system of equations: $f(v, \bar{p}, u, p) = \mathbf{res}$. See Algorithm 2 for details of the NSO solver.

Algorithm 2 FAS $\mathbf{VCycle}(v^h, \bar{p}^h, u^h, p^h, \mathbf{res}^h)$: This is used to solve the NSO subproblem. The algorithm is a standard FAS-VCycle with 2 pre and post smoothing (Line 8, Line 28) and 10 final smoothing (Line 3).

Input: A tentative solution $(v^h, \bar{p}^h, u^h, p^h)$
Output: Refined solution to $f(v^h, \bar{p}^h, u^h, p^h) = \mathbf{res}^h$

1: **if** h is coarsest **then**
2: ▷ Final smoothing for the coarsest level
3: **for** $k = 1, \cdots, 10$ **do**
4: $\mathbf{S}(v^h, \bar{p}^h, u^h, p^h)$
5: **end for**
6: **else**
7: ▷ Pre smoothing
8: **for** $k = 1, 2$ **do**
9: $\mathbf{S}(v^h, \bar{p}^h, u^h, p^h)$
10: **end for**
11: ▷ Down-sampling
12: **for** $t = v, \bar{p}, u, p$ and **do**
13: $t^{2h} \leftarrow \mathbf{R}(t^h)$
14: $t^h \leftarrow t^h - \mathbf{P}(t^{2h})$
15: **end for**
16: ▷ Compute FAS residual by combining:
17: ▷ 1. the solution on coarse resolution
18: ▷ 2. the residual on fine resolution
19: $\mathbf{res}^{2h} \leftarrow f(v^{2h}, \bar{p}^{2h}, u^{2h}, p^{2h})$
20: $\mathbf{res}^{2h} \leftarrow \mathbf{res}^{2h} + \mathbf{R}(\mathbf{res}^h - f(v^h, \bar{p}^h, u^h, p^h))$
21: ▷ VCycle recursion
22: $\mathbf{VCycle}(v^{2h}, \bar{p}^{2h}, u^{2h}, p^{2h}, \mathbf{res}^{2h})$
23: ▷ Up-sampling
24: **for** $t = v, \bar{p}, u, p$ **do**
25: $t^h \leftarrow t^h + \mathbf{P}(t^{2h})$
26: **end for**
27: ▷ Post smoothing
28: **for** $k = 1, 2$ **do**
29: $\mathbf{S}(v^h, \bar{p}^h, u^h, p^h)$
30: **end for**
31: **end if**

The fast convergence of the geometric FAS relies on a proper definition of the three application-dependent operators: $\mathbf{R}, \mathbf{P}$ and $\mathbf{S}$. The restriction operator $\mathbf{R}$ downsamples a fine grid solution to a coarser level for efficient error reduction, and the prolongation operator $\mathbf{P}$ upsamples the coarse grid solution to correct the fine grid solution. We use simple trilinear interpolation for these two operators whether applied on scalar or vector fields. Finally, designing the smoothing operator $\mathbf{S}$ is much more involved. $\mathbf{S}$ should, by itself, be a cheap iterative solver for $f(v, \bar{p}, u, p) = \mathbf{res}$. Compared with previous works such as [Chentanez and Müller 2011] where multigrid is used for solving the pressure field p only, we are faced with two new challenges. First, since we are solving the primal as well as dual variables, which gives a saddle point problem, the Hessian matrix is not positive definite in the spatial domain, so that a

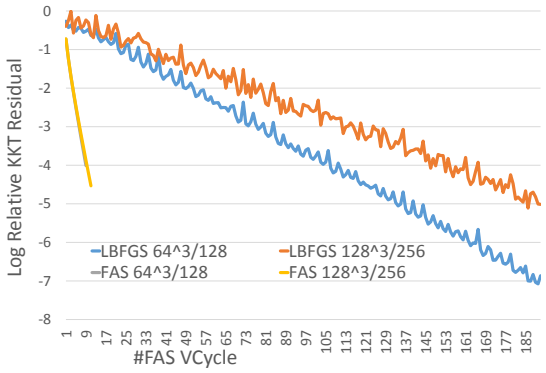


Fig. 6: Convergence history of FAS compared with that of the LBFGS optimizer, running on two grid resolutions and with a different number of timesteps (denoted as n^d/N). FAS achieves a linear rate of error reduction independent of grid resolution and number of timesteps, as the two curves overlap.

Jacobi or Gauss-Seidel (GS) solver does not work. Second, we are not coarsening in the temporal domain, so the temporal correlation must be considered in the smoothing operator.

Our solution is to consider the primal and dual variables at the same time using the Symmetric Coupled Gauss-Seidel (SCGS) smoothing operator [Vanka 1983]. SCGS smoothing is a primal-dual variant of GS. In our case, where all the variables are stored in a staggered grid, SCGS smoothing considers one cell at a time. It solves the primal variables v, u stored on the 6 cell faces as well as the dual variables $p, \bar{p}$ stored in the cell center at the same time by solving a small 14×14 linear problem (10×10 in 2D). Like red-back-GS smoothing, we can parallelize SCGS smoothing using the 8-color tagging (see Figure 5).

The above SCGS solver only considers one timestep at a time. To address the second problem of temporal correlation, we augment the SCGS solver with the temporal domain. We solve the 14 variables associated with a single cell across all the timesteps at once. Although this involves solving a large $14N \times 14N$ linear system for each cell, the left hand side of the linear system is a block tridiagonal matrix so that we can solve the system in $\mathcal{O}(N)$. Indeed, the Jacobian matrix of f takes the following form:

$$\frac{\partial f}{\partial v,\bar{p},u,p} = \begin{pmatrix} \frac{K}{r}I & \nabla & \frac{\partial u_0}{\partial v_0}^T & & \\ \nabla^T & & & & \\ \hline \frac{\partial u_0}{\partial v_0} & & -I & \nabla & \frac{\partial u_0}{\partial v_1} \\ & & \nabla^T & & \\ \hline & & \frac{\partial u_0}{\partial v_1}^T & & \frac{K}{r}I \quad \nabla \\ & & & & \nabla^T \quad \ddots \end{pmatrix}, \tag{10}$$

where the size of each block is 5×5 in 2D and 7×7 in 3D. Due to this linear time solvability, the optimal multigrid performance is still linear in the number of spatial-temporal variables. The average convergence history for our multigrid solver is compared with a conventional LBFGS algorithm [McNamara et al. 2004] in Figure 6. Our algorithm achieves a stable linear rate of error reduction independent of both the grid resolution and the number of timesteps.

4.4 ADMM Outer Loop

Equipped with solvers for the two subproblems, we present our ADMM outer loop in Algorithm 3. We find it very time-consuming for either Equation 8 or a quasi-Newton method solving the AO subproblem to converge to an arbitrarily small residual due to the non-smooth nature of the operator $\mathbf{A}[\bullet, \bullet]$. Both algorithms decrease the objective function in the first few iterations and then wander around the optimal solution. In view of this, we run Equation 8 (Section 4.1) for a fixed number of iterations before moving on to the NSO subproblem (Section 4.2) so that each ADMM iteration has $\mathcal{O}(n^d N)$ complexity and is linear in the number of space-time variables. Finally, our stopping criterion for the NSO subproblem is that the residual $\|f\|_\infty < \epsilon_{FAS}$. Our stopping criterion for the ADMM outer loop is that the maximal visual difference, the largest difference of the density field over all the timesteps, generated by two consecutive ADMM iterations should be smaller than ϵ_{ADMM}.

Algorithm 3 ADMM Outer Loop

Input: Parameters $K, r, \rho_i^*, \epsilon_{STFAS}, \epsilon_{ADMM}$
Output: Optimized velocity fields v_i and density fields ρ_i

for $i = 0, \cdots, N$ **do**
  Set $v_i \leftarrow 0$
  Set $\rho_i^{last} \leftarrow \rho_i$
end for
while true **do**
  ▷ Solve the AO subproblem
  Run Algorithm 1 for a fixed number of iterations
  ▷ Solve the NSO subproblem
  while $\|f(v, \bar{p}, u, p)\|_\infty > \epsilon_{STFAS}$ **do**
    Algorithm 2
  end while
  ▷ Stopping criterion
  if $\max_i \|\rho_i^{last} - \rho_i\|_\infty < \epsilon_{ADMM}$ **then**
    Return v_i, ρ_i
  end if
  for $i = 0, \cdots, N$ **do**
    Set $\rho_i^{last} \leftarrow \rho_i$
    ▷ Update augmented Lagrangian multiplier
    Set $\lambda_i \leftarrow \lambda_i + K\beta(v_i - v_i^*)$
  end for
end while

5. RESULTS AND ANALYSIS

Name	Value
Δt	$0.4 \sim 2.0s$
K	10^3
r	$10^{2\sim4}$
β for updating λ_i	1
#Equation 8	2
ϵ_{FAS}	10^{-5}
ϵ_{ADMM}	$\frac{\rho_{max}}{100}$

Table I. : Parameters.

Parameter Choice: We use the same set of parameters listed in Table I for all experiments, where ρ_{max} is the maximal density magnitude at the initial frame. In our experiments, the ADMM algorithm always converges in fewer than 50 iterations. Further, running only 2 iterations of Equation 8 in each ADMM loop will not deteriorate the performance. In

fact, according to the averaged convergence history of the AO subproblem illustrated in Figure 7, the fixed point iteration Equation 8 usually converges in the first 4 iterations before it wanders around a local minimum. After fine tuning, we found that 2 iterations lead to the best overall performance. In this case, the overhead of solving the AO subproblem is marginal compared with the overhead of solving the NSO subproblem. Finally, unlike fluid simulation, the performance of spacetime optimization does not depend on the timestep size due to our robust advection operator (Equation 4). When we increase the timestep size from $0.4s$ to $2s$ for the examples in Figure 8 and Figure 9, which is extremely large, our algorithm's convergence behavior is about the same. Under this setting, the convergence history of the ADMM outer loop for our first example (Figure 1) is illustrated in Figure 7. The convergence history can be decomposed into two stages. In the first stage, the first term of Equation 5 (keyframe shape matching) dominates, the solver gradually evolves the solution to match the keyframe shape, and the KKT-Residual is not monotonically decreasing. In the second stage, however, the second term (control force regularization) dominates Equation 5 and the KKT-Residual quickly decreases. Since the solutions of consecutive ADMM iterations do not change much, we have also tried to use just a few SCGS smoothing steps, instead of the entire FAS Algorithm 2, to approximately solve the NSO subproblem. In practice, we observed this treatment smoothed out the fluid-like behaviors, when large regularization r is used.

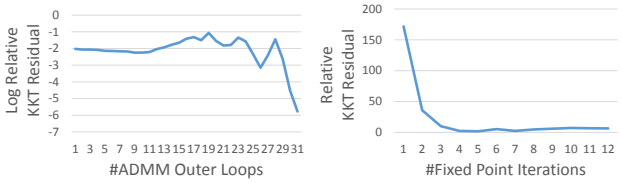


Fig. 7: We profile the convergence history of the example Figure 1. We plot the logarithm of relative KKT residual of the optimized velocity field after each ADMM loop (left); and the absolute residual of the AO subproblem's KKT conditions after each iteration of Equation 8 (right).

Benchmarks: To demonstrate the efficiency and robustness of our algorithm, we used 7 benchmark problems that vary in their grid resolution, number of timesteps, and number of keyframes. The memory overhead and computational overhead are summarized in Table II. All of the results are generated on a desktop PC with an i7-4790 8-core CPU 3.6GHz and 12GB of memory. We use OpenMP for multithread parallelization.

Our first example is five controlled animations matching a circle to the letters "FLUID". Compared with [Treuille et al. 2003], which uses a relatively small set of control force templates to reduce the search space of control forces, we allow control on every velocity component so that the matching to keyframe is almost exact. After the keyframe, we remove the control force, and rich smoke details are generated by pure simulation as illustrated in Figure 1. However, in the controlled phase of Figure 1, this example seems "too much controlled", meaning that most smoke-like behaviors are lost. This effect has also been noticed in [Treuille et al. 2003]. However, unlike their method, in which the number of templates needs to be carefully tuned to recover such behavior, we can simply adjust the regularization r in our system to balance matching exactness and the amount of smoke-like behaviors. In Figure 8, we generated three animations with two keyframes: first two circles and then a bunny, using $r = 10^{2,3,4}$ respectively. These animations are also shown in the video. Our algorithm is robust to a wide range of parameter choices. But more iterations are needed for the multigrid to converge for a larger r as shown in Table II. Finally, since we allow every velocity component to be optimized, the resulting animation exhibits lots of small-scale details as indicated in Figure 9, which is not possible with the small set of force templates used in [Treuille et al. 2003].

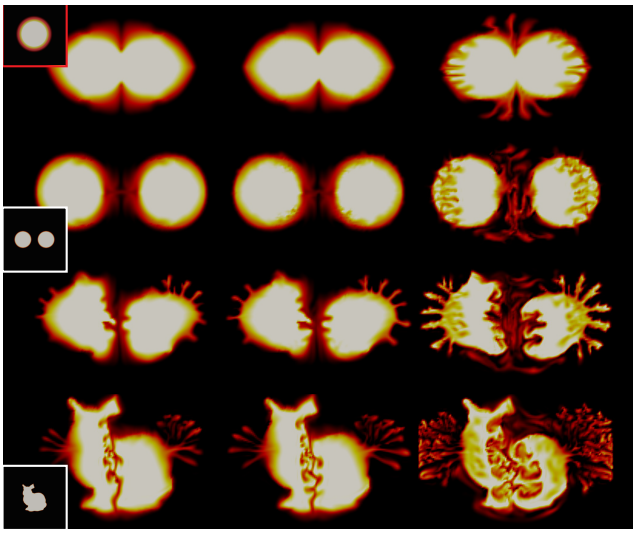


Fig. 8: For this animation, we match the circle (red) first to two smaller circles and then to a bunny (we show frames $20, 40, 60, 80$ from top to bottom). The resolution is $128^2/80$, and we test three different values of ghost force regularization $r = 10^{2,3,4}$ (from left to right). More smoke-like behaviors are generated as we increase r.

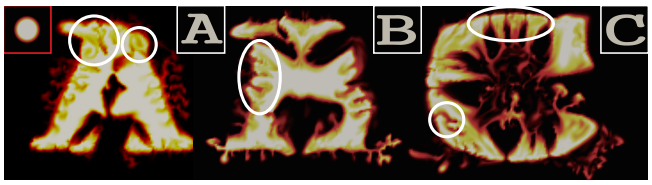


Fig. 9: In this example, we deform a sphere into letter "A", then letter "B" and finally letter "C". For such complex deformation, it is advantageous to allow every velocity component to be optimized. So that a lot of fine-scale details can be generated as illustrated in the white circles.

In addition to these 2D examples, we also tested our algorithm on some 3D benchmarks. Our first example is shown in Figure 10 and runs at a resolution of $64^3/40$. We use two keyframes at frame 20 and 40, and the overall optimization takes about 7 hours. In our second example, shown in Figure 11, we try to track the smoke with a dense sequence of keyframes from the motion capture data of a human performing a punch action. Such an example is considered the most widely used benchmarks for PD-type controllers such as [Shi and Yu 2005]. With such strong and dense guidance, our algorithm converges very quickly, within 5 iterations. Our third example (Figure 13) highlights the effect of regularization coefficient r in 3D. Like our 2D counterpart Figure 8, larger r usually results in more wake flow behind moving smoke bodies. Finally, we evaluated our algorithm on a benchmark with keyframe shapes of varying genera. As illustrated in Figure 12, the initial smoke shape has genus zero, but we use two keyframes, where the smoke shapes

Example(n^d/N)	Boundary	#ADMM	Avg. AO (s)	Avg. NSO (s)	Total(hr)	Memory(Gb)	Total LBFGS(hr)
Letters "FLUID"($128^2/40, r = 10^3$)	Neumann	13	10	60	0.25	0.06	4
Letters "FLUID"($128^2/80, r = 10^3$)	Neumann	17	21	142	0.76	0.12	9
Circle Bunny($128^2/80, r = 10^2$)	Neumann	25	20	130	1.04	0.2	12
Circle Bunny($128^2/80, r = 10^3$)	Neumann	37	20	220	2.46	0.2	15
Circle Bunny($128^2/80, r = 10^4$)	Neumann	43	20	218	2.84	0.2	16
Letters ABC($128^2/60, r = 10^4$)	Neumann	33	16	179	1.78	0.15	14
Sphere Armadillo Bunny($64^3/40, r = 10^3$)	Neumann	17	103	1341	6.81	1.34	N/A
Varying Genus($64^2 \times 32/40, r = 10^3$)	Periodic	20	82	840	5.12	0.67	N/A
Human Mocap($64^2 \times 128/60, r = 10^3$)	Periodic	5	1437	3534	6.9	4.0	N/A
Moving Sphere($64^3/60, r = 10^2$)	Neumann	17	630	1792	11.43	2.2	N/A
Moving Sphere($64^3/60, r = 10^3$)	Neumann	22	630	1978	15.93	2.2	N/A
Letters ABC 3D($64^3/150, r = 10^2$)	Periodic	20	1512	3220	26.28	5.9	N/A

Table II. : Memory and computational overhead for all the benchmarks. From left to right: name of example (resolution parameters); the spatial boundary condition; number of outer ADMM iterations; average time spent on each AO subproblem; average time spent on each NSO subproblem; total time until convergence using our algorithm; memory overhead; total time until convergence using LBFGS. By comparing the three "Circle Bunny" examples, we can observe that the number of ADMM outer loops is roughly linear to $log_{10}(r)$. More ADMM outer loops are needed, if more fluid-like behaviors are desired. From the two examples of the Letters "FLUID" (Line 1 and Line 2), we can observe that the computational cost of each ADMM outer loop (Avg. AO + Avg. NSO) is roughly linear in the number of timesteps. This cost is also governed by the number of keyframes. By comparing Line 2 and Line 4, we can observe that the Circle Bunny example which involves two keyframes requires more computation to solve the NSO subproblem.

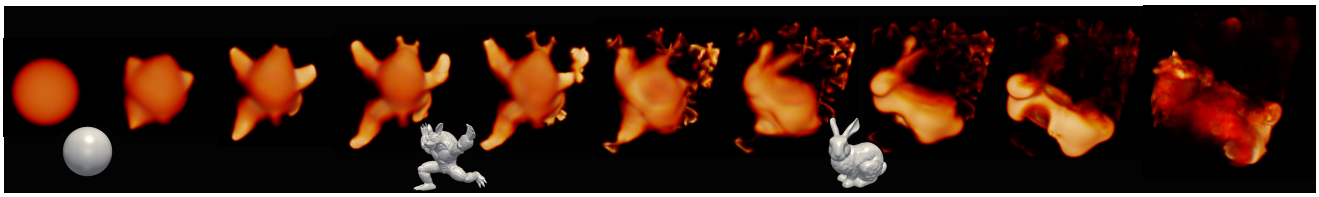


Fig. 10: 3D smoke control example of deforming a sphere first to an armadillo and then to a bunny. This example runs at the resolution of 64^3 with 40 timesteps. The optimization can be accomplished in 7hr.

have genus one and two. Our algorithm can handle such complex cases.

Comparison with LBFGS: We compared our ADMM-based solver with a gradient-based quasi-Newton optimizer in solving the original problem (Equation 5). Specifically, we use LBFGS method [Nocedal and Wright 2006]. Such method approximates the Hessian using a history of gradients calculated by past iterations. We set the history size to be 8, which is typical. We use same stopping criteria for both LBFGS and our method. Under this setting, we compared the performance of LBFGS and the ADMM solver on two of our 2D examples: Figure 1 and Figure 8. For the example of letter matching in Figure 1, LBFGS algorithm takes 4hr and 71 iterations to converge. While for the example of changing regularization in Figure 8, LBFGS algorithm takes 12hr and 152 iterations at $r = 10^2$, 15hr and 170 iterations at $r = 10^3$, and 16hr and 212 iterations at $r = 10^4$. Therefore, our algorithm is approximately an order of magnitude faster than a typical implementation of LBFGS.

The speedup over LBFGS optimizer occurs for two reasons. First, we break the problem up into the AO subproblem and the NSO subproblem, that have sharply different properties. The AO subproblem is nonsmooth while the NSO subproblem is not. In practice, neither our fixed point iteration scheme in Equation 8 nor the LBFGS algorithm can efficiently solve AO to arbitrarily small KKT residual. Without such decomposition, it takes a very long time to solve the overall optimization problem by taking a lot of iterations. The second reason is the use of warm-started FAS solver for the NSO subproblem. Note that LBFGS algorithm not only takes more iterations, but each iteration is also more expensive. This is mainly because of the repeated gradient evaluation in each LBFGS iteration, where each evaluation runs the adjoint method with a cost equivalent to two passes of fluid resimulation.

Comparison with PD Controller: We also compared our method with simple tracker type controllers such as PD controller [Fattal and Lischinski 2004]. To drive the fluid body towards a target keyframe shape using heuristic ghost forces, PD controllers result in much lower overhead in terms of fluid resimulation, as compared to our approach based on optimal controllers. In contrast, optimal controllers provide better flexibility and robust solutions as compared to PD controllers. A PD controller tends to be sensitive to the parameters used for the ghost forces. Moreover, its performance also depends on the use of the compressible control forces (see Figure 14). These compressible forces can potentially eliminate the visually appealing vortical fluid motions. On the other hand, an optimal controller always achieves exact keyframe timing, while such exact timing requires fine-tuning the strength of control forces in a PD controller, as shown in Figure 14. Moreover, with an optimal controller, users can easily balance between the exactness of keyframe matching and the amount of fluid-like behaviors based on a single parameter r (see Figure 8).

Memory Overhead: Since fluid control problems usually have a high memory overhead, we derive here an analytical upper bound of the memory consumption $M(n, d, N)$:

$$M(n,d,N) \sim \left[(n^d) * (1+d) * 2 * 2\right] * \left[1 + \tfrac{1}{2} + \tfrac{1}{4} \cdots\right] * N = 8n^d(1+d)N,$$

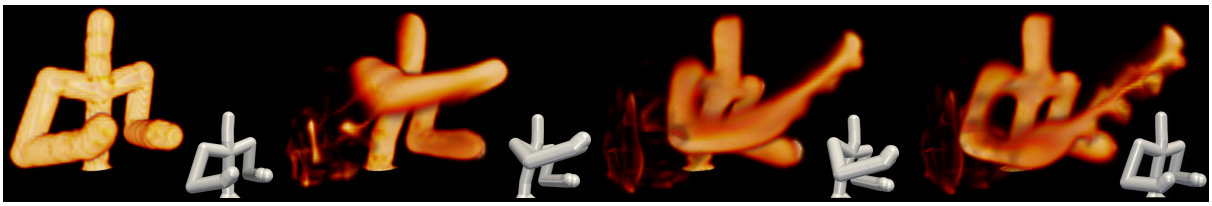


Fig. 11: We generate the famous example of tracking smoke with a dense sequence of keyframes, which comes from human motion capture data. Our algorithm converges and generates rich smoke drags within 5 ADMM iterations.

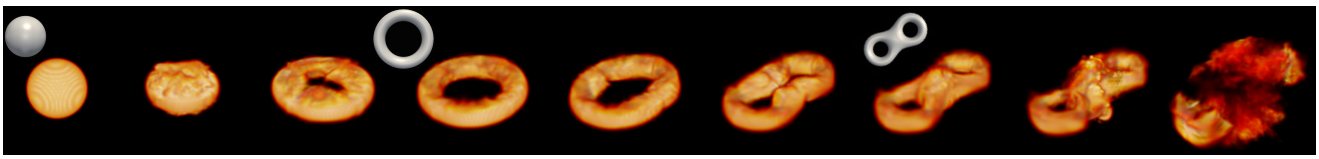


Fig. 12: Example of smoke control where the keyframes have varying genera. The initial frame is a sphere (genus 0). The first keyframe located at frame 20 is a torus (genus 1) and the second keyframe located at frame 40 is the shape eight (genus 2). The resolution is $64^2 \times 32/40$ and the overall optimization takes 5hr with $r = 10^3$.

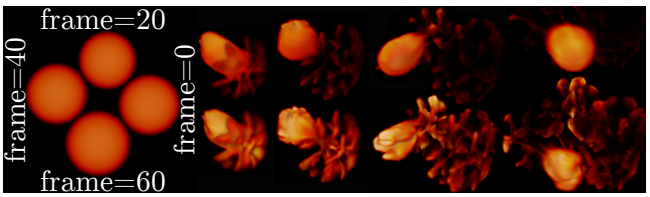


Fig. 13: A moving smoke sphere guided by the 3 keyframes (left). We experimented with $r = 10^3$ (top) and $r = 10^4$ (bottom). Larger regularization results in more wake flow behind moving smoke bodies. The same effect can be observed in Figure 8.

where n is the grid resolution, d is the dimension, and N is the number of timesteps. To derive this bound, note that we can reuse the memory consumed by Algorithm 2 in Algorithm 1, and Algorithm 2 always consumes more memory than Algorithm 1, so that we only consider the memory overhead of Algorithm 2. The first term $n^d * (1 + d)$ is the number of variables needed for storing a pair of pressure and velocity fields. This number is doubled because we need to store $u_i, \bar{p}_i$ in addition to v_i, p_i at each timestep. We double it again because we need additional memory for storing $\mathbf{res}$ in FAS. Finally, the power series is due to the hierarchy of grids. At first observation, this memory overhead is higher than [Treuille et al. 2003; McNamara et al. 2004] since we require additional memory for storing the dual variables at multiple resolutions. However, due to the quasi-Newton method involved in their approach, additional memories are needed to store a set of L gradients to approximate the inverse of the Hessian matrix. L is usually $5 \sim 10$, leading to the following upper bound:

$$M_{LBFGS}(n, d, N) \sim \left[(n^d) * (1 + d)\right] * L * N = Ln^d(1 + d)N.$$

In our benchmarks, the memory overheads of our ADMM and LBFGS solvers are comparable.

Convergence Analysis: Here we analyze the convergence of our approach and discuss some modifications towards improved convergence of Algorithm 3. We have applied some of these modifications for Line 7 and Line 9 of Algorithm 3, which then takes a slightly more complex form.



Fig. 14: We deform a smoke ball into a dragon using our method (top left) and PD controller with different parameter settings. Top Right: Without gathering forces, the keyframe shape is not matched. Bottom Left: The animation appears to be oscillatory without viscous forces, resulting in lots of escaped smokes. Bottom Right: A stable, non-oscillatory animation that matches the keyframe well can be achieve by fine-tuning the three parameters: the strength of guiding forces, the strength of gathering forces, and the strength of viscous forces.

For our AO solver (Line 7 of Algorithm 3), we observe that it can be difficult for Algorithm 1 to converge to an arbitrarily small KKT residual in each loop of Algorithm 3. As illustrated in Algorithm 1, one could use a simple strategy that can guarantee function value decreases by blending a new solution with the previous solution and tuning the blending factor in a way similar to the line search algorithm. This modification has low computational overhead since one does not need to apply the costly solenoidal projection operator

Q again after the blending, as the sum of two solenoidal vector fields is still solenoidal. In our benchmarks, this strategy leads to a convergent algorithm with low overhead, but the error reduction rate after the first few iterations can still be slow.

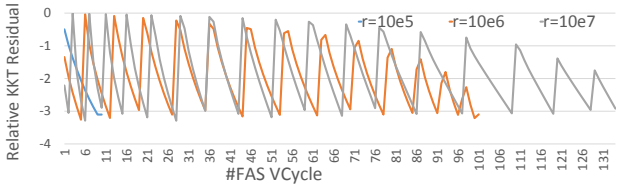


Fig. 15: Convergence history of the NSO solver in the Circle Bunny example. When the regularization coefficient r is extremely large, we have to treat the FAS-Vcycle as a subproblem solver of the LM algorithm and the entire NSO solve requires many more FAS-Vcycles.

The same analysis can also be used for the NSO solver (Line 9 of Algorithm 3). To ensure convergence of Algorithm 2, we could add a perturbation to the penalty coefficient K in the Hessian matrix Equation 10. Note that as $K \to \infty$, $v_i \to v_i^*$. Therefore, this strategy essentially makes Algorithm 2 the subproblem solver for the Levenberg-Marquardt algorithm [Nocedal and Wright 2006], which in turn guarantees convergence. As illustrated in Figure 15, Levenberg-Marquardt modification can be necessary when one uses extremely large regularization r, because we observe that the convergence rate decreases as r increases. In these settings, however, many more FAS-Vcycles are needed to solve the NSO subproblem and the advantage over the LBFGS solver also decreases.

Finally, for the ADMM outer loop (Line 5 of Algorithm 3), state-of-the-art results showing its convergence rely on strong assumptions of its objective function, such as global convexity. Therefore, our current implementation of outer loop is not guaranteed to converge. However, such guarantee can be provided by using a standard Augmented Lagrangian solver, instead of ADMM solver. Specifically, one can run Algorithm 3 without applying Line 18 until the decrease in function value is lower than some threshold. Further exploration of this option is left as future work.

6. CONCLUSION AND LIMITATIONS

In our work, we present a new algorithm for the optimal control of smoke animation. Our algorithm finds the stationary point of the KKT conditions, solving for both primal and dual variables. Our key idea is to refine primal as well as dual variables in a warm-started manner, without requiring them to satisfy the Navier-Stokes equations exactly in each iteration. We tested our approach on several benchmarks and a wide range of parameter choices. The results show that our method can robustly find the locally optimal control forces while achieving an order of magnitude speedup over the gradient-based optimizer, which performs fluid resimulation in each gradient evaluation.

On the downside, our method severely relies on the spatial structure and the staggered grid discretization of the Navier-Stokes equations. This imposes a major restriction to the application of our techniques. Nevertheless, generalizing our idea to other fluid discretization is still possible. For example, our method can be used with a fluid solver discretized on a general tetrahedron mesh such as [Chentanez et al. 2007; Pavlov et al. 2011] since the KKT conditions are invariant under different discretizations, and the three operators to define FAS stay valid. On the other hand, generalizing our method to free-surface flow or to handle internal boundary conditions can be non-trivial. The distance metric C_i in Equation 7 needs to be modified to make it aware of the boundaries, e.g., Euclidean distances should be replaced with Geodesic distances. However, modifying the NSO solver to handle the boundaries can be relatively straightforward. This is because our multigrid formulation is the same as a conventional multigrid formulation in spatial domain, using simple trilinear prolongation and restriction operators. Therefore, existing works on boundary aware multigrid such as [Chentanez and Müller 2011] can also be applied to our space-time formulation.

In addition, unlike [Treuille et al. 2003; McNamara et al. 2004], which use a set of template ghost force bases to reduce the search space, our method allows every velocity component to be optimized. This choice is application dependent. For matching smoke to detailed keyframes with lots of high frequency features, our formulation can be useful. However, using a reduced set of template ghost forces could help to avoid the popping artifacts illustrated in Figure 4, and at the same time it allows more user control over the applied control force patterns. For example, the use of vortex force templates encourages more swirly motions in the controlled animations. Moreover, from Figure 8, we can see many small-scale escaping smoke parcels using a large r. Our controller does not apply control forces on these parcels in order to reduce the magnitude of control forces. If these smoke parcels are undesirable in the final animation, a template-based formulation can be used. Combining the control force templates with our formulation is considered as future work.

In terms of computational overhead, since our optimal controller always solves the spacetime optimization by considering all the timesteps, it is much slower than a simple PD controller which considers one timestep at a time. For example, it took more than 26 hours to generate our longest animation with 150 timesteps illustrated in Figure 16. In order to reduce runtime cost, we can use a larger timestep size to reduce the number of timesteps. Fluid simulation with a large timestep size has been addressed in prior work, such as [Lentine et al. 2012]. We handle the large timestep size by using a novel advection operator (Equation 4) with an adaptive order of Taylor expansion. An alternative solution is to use a conventional advection scheme with an adaptive timestep size determined using the CFL condition. A thorough analysis of alternative advection schemes is a good topic of future work. Also, we can lower the spatial resolution in the control phase and then use smoke upsampling methods such as [Nielsen and Bridson 2011] to generate a high quality animation.

Further accelerations to our method are still possible. For example, one can parallelize our algorithm in a distributed environment. Indeed, multigrid is known as one of the most cluster-friendly algorithms. Moreover, meta-algorithms such as multiple shooting [Bock and Plitt 1984] try to break the spacetime optimization into a series of sub-optimizations that consider only a short animation segment and are thus faster to solve. Finally, the benefits of both optimal and PD controllers can be combined by borrowing the idea of receding horizon control [Mayne and Michalska 1990]. In these controllers, optimal control is applied only to a short window of timesteps starting from the current one, and the window keeps being shifted forward to cover the whole animation.



Fig. 16: The same example as Figure 9 in 3D. We use 150 timesteps with keyframes at timestep 50, 100, 150, respectively. Our algorithm converges after 20 iterations and the total computation time is 26 hours.

APPENDIX

A. KKT SYSTEM OF THE NSO SUBPROBLEM

We derive here the KKT system for the NSO subproblem. Instead of simply introducing the Lagrangian multipliers and following standard techniques as we did for the AO subproblem, we present a derivation based on the analysis of the ghost force u_i. We first eliminate the Navier-Stokes constraints by writing u_i as a function of v_i and v_{i+1}. Next, we plug this function into our objective to obtain:

$$\frac{r}{2}\sum_{i=0}^{N-1}\|u_i(v_i,v_{i+1})\|^2+\frac{K}{2}\sum_{i=0}^{N-1}\|v_i-v_i^*\|^2.$$

Taking the derivative of this objective against v_i and considering the additional solenoidal constraints on v_i, we get the first two equations in f:

$$\begin{aligned}&\frac{K}{r}(v_i-v_i^*)+\frac{\partial u_i}{\partial v_i}^T u_i+\frac{\partial u_{i-1}}{\partial v_i}^T u_{i-1}+\nabla\bar{p}_i=0\\&\nabla\cdot v_i=0,\end{aligned}$$

where $\bar{p}_i$ is the Lagrangian multiplier. Now in order to derive the other two conditions in Equation 9, we need to determine the additional pressure p_i. We assert that p_{i+1} is the Lagrangian multiplier of the solenoidal constraints on u_i. In fact, if u_i is not divergence-free, we can always perform a pressure projection on u_i by minimizing $\|u_i-\nabla p_{i+1}\|^2$ to get a smaller objective function value. As a result, u_i must be divergence-free at the optima with p_{i+1} being the Lagrangian multiplier, and we get the two additional equations of f:

$$\begin{aligned}&\frac{v_{i+1}-v_i}{\Delta t}+\mathbf{Adv}\left[v_{i+1}\right]-u_i+\nabla p_{i+1}=0\\&\nabla\cdot u_i=0.\end{aligned}$$

From these two conditions, we can see that $\frac{\partial u_i}{\partial v_i}=-\mathbf{Q}\frac{I}{\Delta t}$, $\frac{\partial u_{i-1}}{\partial v_i}=\mathbf{Q}(\frac{I}{\Delta t}+\frac{\partial\mathbf{Adv}[v_i]}{\partial v_i})$. Here $\mathbf{Q}$ is the solenoidal projection operator introduced in Equation 8. However, we can drop this $\mathbf{Q}$ because we have $\frac{\partial u_{i-1}}{\partial v_i}^T u_i=(\frac{I}{\Delta t}+\frac{\partial\mathbf{Adv}[v_i]}{\partial v_i})^T\mathbf{Q}^T u_i$ and $\mathbf{Q}^T u_i=\mathbf{Q}u_i=u_i$ by the fact that u_i is already solenoidal.

REFERENCES

Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. 1984. Pyramid methods in image processing. *RCA engineer* 29, 6 (1984), 33–41.

Alexis Angelidis, Fabrice Neyret, Karan Singh, and Derek Nowrouzezahrai. 2006. A Controllable, Fast and Stable Basis for Vortex Based Smoke Simulation. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '06)*. Eurographics Association, 25–32.

Hans Georg Bock and Karl-Josef Plitt. 1984. A multiple shooting algorithm for direct solution of optimal control problems. Proceedings of the IFAC World Congress.

Alfio Borzi and R Griesse. 2005. Experiences with a space–time multigrid method for the optimal control of a chemical turbulence model. *International journal for numerical methods in fluids* 47, 8-9 (2005), 879–885.

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3, 1 (2011), 1–122.

Achi Brandt and Oren E Livne. 2011. *Multigrid techniques: 1984 guide with applications to fluid dynamics*. Vol. 67. SIAM.

Richard H Byrd, Peihuang Lu, Jorge Nocedal, and Ciyou Zhu. 1995. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing* 16, 5 (1995), 1190–1208.

Nuttapong Chentanez, Bryan E Feldman, François Labelle, James F O'Brien, and Jonathan R Shewchuk. 2007. Liquid simulation on lattice-based tetrahedral meshes. In *Proceedings of the 2007 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 219–228.

Nuttapong Chentanez and Matthias Müller. 2011. Real-time eulerian water simulation using a restricted tall cell grid. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 82.

Raanan Fattal and Dani Lischinski. 2004. Target-driven smoke animation. In *ACM Transactions on Graphics (TOG)*, Vol. 23. ACM, 441–448.

Ronald Fedkiw, Jos Stam, and Henrik Wann Jensen. 2001. Visual simulation of smoke. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. ACM, 15–22.

James Gregson, Ivo Ihrke, Nils Thuerey, and Wolfgang Heidrich. 2014. From Capture to Simulation: Connecting Forward and Inverse Problems in Fluids. *ACM Trans. Graph.* 33, 4, Article 139 (July 2014), 11 pages. DOI:http://dx.doi.org/10.1145/2601097.2601147

F. H. Harlow and J. E. Welch. 1965. Numerical calculation of time-dependent viscous incompressible flowof fluid with free surfaces. *Physics of Fluids* 8 (1965), 2182–2188.

Roland Herzog and Karl Kunisch. 2010. Algorithms for PDE-constrained optimization. *GAMM-Mitteilungen* 33, 2 (2010), 163–176.

Michael Hinze, Michael Köster, and Stefan Turek. 2012. A space-time multigrid method for optimal flow control. In *Constrained optimization and optimal control for partial differential equations*. Springer, 147–170.

Michael Lentine, Matthew Cong, Saket Patkar, and Ronald Fedkiw. 2012. Simulating Free Surface Flow with Very Large Time Steps. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '12)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 107–116. http://dl.acm.org/citation.cfm?id=2422356.2422373

Brian P Leonard. 1979. A stable and accurate convective modelling procedure based on quadratic upstream interpolation. *Computer methods in applied mechanics and engineering* 19, 1 (1979), 59–98.

David Q Mayne and Hannah Michalska. 1990. Receding horizon control of nonlinear systems. *Automatic Control, IEEE Transactions on* 35, 7 (1990), 814–824.

Antoine McNamara, Adrien Treuille, Zoran Popović, and Jos Stam. 2004. Fluid control using the adjoint method. In *ACM Transactions On Graphics (TOG)*, Vol. 23. ACM, 449–456.

Igor Mordatch, Emanuel Todorov, and Zoran Popović. 2012. Discovery of complex behaviors through contact-invariant optimization. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 43.

Rahul Narain, Matthew Overby, and George E. Brown. 2016. ADMM ⊇ Projective Dynamics: Fast Simulation of General Constitutive Models. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '16)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 21–28. http://dl.acm.org/citation.cfm?id=2982818.2982822

Michael B Nielsen and Robert Bridson. 2011. Guide shapes for high resolution naturalistic liquid simulation. In *ACM Transactions on Graphics (TOG)*, Vol. 30. ACM, 83.

Michael B Nielsen and Brian B Christensen. 2010. Improved variational guiding of smoke animations. In *Computer Graphics Forum*, Vol. 29. Wiley Online Library, 705–712.

Jorge Nocedal and Stephen Wright. 2006. *Numerical optimization*. Springer Science & Business Media.

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), 184.

Dmitry Pavlov, Patrick Mullen, Yiying Tong, Eva Kanso, Jerrold E Marsden, and Mathieu Desbrun. 2011. Structure-preserving discretization of incompressible fluids. *Physica D: Nonlinear Phenomena* 240, 6 (2011), 443–458.

Nick Rasmussen, Doug Enright, Duc Nguyen, Sebastian Marino, Nigel Sumner, Willi Geiger, Samir Hoon, and Ron Fedkiw. 2004. Directable photorealistic liquids. In *Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 193–202.

Karthik Raveendran, Nils Thuerey, Chris Wojtan, and Greg Turk. 2012. Controlling liquids using meshes. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 255–264.

Karthik Raveendran, Chris Wojtan, Nils Thuerey, and Greg Turk. 2014. Blending liquids. *ACM Transactions on Graphics (TOG)* 33, 4 (2014), 137.

Lin Shi and Yizhou Yu. 2005. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*. ACM, 229–236.

Adrien Treuille, Antoine McNamara, Zoran Popović, and Jos Stam. 2003. Keyframe control of smoke simulations. *ACM Transactions on Graphics (TOG)* 22 (2003), 716–723.

SP Vanka. 1983. *Fully coupled calculation of fluid flows with limited use of computer storage*. Technical Report. Argonne National Lab., IL (USA).

Xinxin Zhang and Robert Bridson. 2014. A PPPM fast summation method for fluids and beyond. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 206.

Yongning Zhu and Robert Bridson. 2005. Animating sand as a fluid. In *ACM Transactions on Graphics (TOG)*, Vol. 24. ACM, 965–972.

Feedback Motion Planning for Liquid Pouring Using Supervised Learning

Zherong Pan[1] and Dinesh Manocha[1]
http://gamma.cs.unc.edu/RLFluid

***Abstract*— We present a novel motion planning algorithm for pouring a liquid body from a source to a target container. Our approach uses a receding-horizon optimization strategy that considers liquid dynamics and various other constraints. To handle liquid dynamics without costly fluid simulations, we use a neural network to infer a set of key liquid-related parameters from the observation of the current liquid configuration. To train the neural network, we generate a dataset of successful pouring examples using stochastic optimization in a problem-specific search space. These parameters are then used in the objective function for trajectory optimization. Our feedback motion planner achieves real-time performance, and we observe a high success rate in our simulated 2D and 3D liquid pouring benchmarks.**

I. INTRODUCTION

Robotic manipulation of non-rigid objects such as fluids, elastic bodies, and strings is a challenging problem that arises in different applications. In this paper, we address the problem of pouring liquids, where the goal is to use a robot to pour a liquid body from a source to a target container. These tasks arise when industrial robots are used for painting, cleaning, or dispensing lubricants. Other applications include the use of service robots for cooking, cleaning, or feeding.

A key issue in such motion planning algorithms is to satisfy the liquid dynamics constraints. The liquid body can have a complex topology and undergo large deformations. The underlying solvers tend to use a large number of particles (tens of thousands) to model their motion. This results in a very high-dimensional configuration space of the liquid body and makes it hard to directly use sampling-based motion planning algorithms. Other techniques based on an optimization-based planner [1], [2] may not work well because the free-surface of a liquid body introduces non-smooth changes.

Prior planning algorithms for fluid manipulation are either based on demonstration and learning methods, or use dynamics constraints. The demonstration-based methods use example trajectories and ignore all physical constraints so that they may not generalize to new scenarios. On the other hand, methods using reinforcement learning [3], [4] can take physics constraints into consideration, but they require a problem specific training dataset for each manipulation task. Other techniques use trajectory optimization, which takes into account a full-featured liquid dynamics model [5], [6], but these techniques have a very high computational overhead. This problem can be alleviated using reduced or simplified dynamics models [7], [8], [9], where many liquid dynamics constraints are ignored, and these models are combined with open-loop planners. However, open-loop methods suffer from simulation bias.

[1] Department of Computer Science, the University of North Carolina at Chapel Hill {zherong,dm}@cs.unc.edu

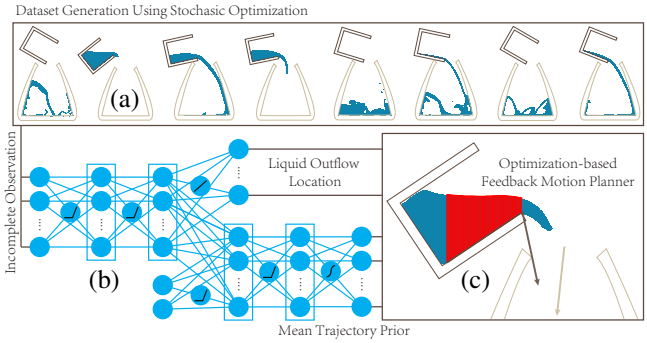


Fig. 1: An illustration of our feedback motion planning framework. From the training dataset found by stochastic optimization (a), we train a neural network that predicts liquid-related parameters: liquid outflow curve and the mean trajectory prior (b). Our online planner then computes the source container trajectory from an incomplete observation (c), e.g., the liquid heightfield in red and the moving speed of the source/target containers (the two arrows).

Main Results: We present a feedback motion planning algorithm to control the fluid flow for liquid pouring that is subject to dynamics constraints. To handle the high dimensionality of fluids, we present a learning-based approach to predict the liquid configurations based on low-dimensional features. This is combined with a receding-horizon trajectory optimization method for online planning. The main novel components of our approach include:

- A supervised learning algorithm that enables the prediction of the state of a high-dimensional liquid body using low-dimensional features. We also describe an efficient strategy to train the neural network.
- A feedback motion planning algorithm that solves a spacetime optimization problem using the receding horizon strategy. The objective function is guided by the neural network, and this significantly increases the efficiency of our planner, as we do not need to perform the costly 3D fluid simulation step at each timestep.

Our neural network is trained using a large amount of successful pouring trajectories. This dataset is automatically generated offline using a large number of random configurations of liquid pouring problems. These configurations are generated by changing the relative positions of two containers, the amount of liquid in the source container, and the speed of the target container movement. As a result, the

trained neural network is robust to environmental changes. We compute the optimal liquid pouring trajectory for each configuration using stochastic optimizations with the help of a fully-featured liquid simulator.

We have evaluated our algorithm on many new and challenging scenarios that are quite different from the training dataset. In practice, our learning-based planner achieves almost the same success rate as was obtained using the groundtruth trajectories on our training dataset. Moreover, we have also tested our planner in new and different environments, including 3D workspaces, liquids with different physical characteristics (e.g., varied the viscosities), and different container shapes. Our results demonstrate that the neural networking can be very robust when used with an optimization-based motion planner, although our training dataset uses a single low-viscous liquid material and a rectangular container shape. Furthermore, the online algorithm is very fast and takes less than 10 milliseconds on a single core of Intel CPU to plan the motion amongst dynamic obstacles.

II. ASSUMPTIONS AND PROBLEM FORMULATION

In this work, we restrict ourselves to a simulated environment. The pouring problem is formulated in Section IV. Our algorithm is composed of three components (see Figure 1).

In the preprocessing stage, we generate a large set of random liquid pouring problems. For each problem, we find a successful pouring trajectory using stochastic optimization (see Section VI). We then extract a low-dimensional feature of the liquid body and train a 4-layered neural network to predict a set of key parameters in pouring (liquid outflow curve and mean trajectory prior). We have not yet applied our framework in a real-life robotic system and currently only assume that the features can be acquired from the sensing data. However, we propose two kinds of possible liquid features to inspire future work on a real-life robotic system. Other features can also be easily used with our method.

During the online stage, the predicted parameters are used in the objective function of a receding-horizon optimization-based motion planner (see Section V). The planner determines the configuration of the source container at the next timestep by minimizing an objective function that encourages successful pouring while considering various other constraints including collision avoidance and trajectory smoothness.

III. RELATED WORKS

Our approach builds on three areas of prior work: motion planning, planning for dynamic objects, and reinforcement learning.

A. GENERAL MOTION PLANNING

A motion planning algorithm searches for a trajectory that satisfies a set of constraints (collision-free, smoothness), which may also be optimal under a given quality measure. Many early motion planners such as [10] and its descendants [11], [12], [13] consider only collision-free constraints. Unlike these methods, which tend to compute a trajectory by sampling in the space of possible trajectories, optimization-based motion planners such as [2], [1], [14] can easily take into account other constraints, such as dynamics, smoothness, etc. Many of these approaches formulate the problem as a spacetime continuous optimization. Such optimization methods have also been used for liquid transfer [6], [9] based on simplified dynamics when limited to static environments.

There is considerable work on feedback motion planning that uses refinement schemes based on feedback control laws. This can be performed using replanning [15], [16], [17] or by formulating the problem as a Markov Decision Process [18]. These ideas have been applied to high-dimensional continuous systems such as humanoid robots [19], [20]. In this work, we present such a feedback motion planning algorithm for liquid transfer.

B. PLANNING FOR DYNAMIC OBJECTS

The extension of conventional motion planning algorithms to the manipulation of non-rigid objects has been addressed in the context of virtual suturing [21], cloth folding [22], and surgical simulation [23]. It can be challenging to deal with non-rigid objects with high-dimensional configuration spaces. This is especially the case with liquid manipulation tasks, where the dimension can be as high as several million (see [6] for a detailed discussion). For certain types of fluids such as smoke and fire, optimization-based motion planning can be adapted to solve the problem by exploiting the special structure of the resulting fluid simulator [24], [25]. However, it is non-trivial to extend these methods to control liquid bodies with non-smooth, rapidly-changing free surfaces. Moreover, prior methods are designed for offline applications and computationally very costly. Previous work [9] reduced the computational cost by using a much simplified liquid model, dependent on just two variables.

C. IMITATION OR REINFORCEMENT LEARNING

Reinforcement and imitation learning have been shown to be effective in terms of controlling high-dimensional dynamic systems, e.g., a humanoid robot [19], [26]. Recently, imitation learning has been used to perform liquid manipulation using example container trajectories from a human demonstrator [27], [28], [29]. However, the learning framework in this work does not take fluid dynamics constraints into account. Moreover, trajectories of liquid body shapes from real-life experiments have to be captured and digitized to construct the dataset, which is challenging in and of itself (see [30], [31]). More recently, reinforcement learning has also been used to learn pouring of granular materials in [4], [3]. Our methods differ from these methods in that we only use supervised learning, but we combine it with trajectory optimization to enhance the robustness of our motion planner.

IV. OVERVIEW

In this section, we first introduce our formulation of the liquid pouring problem and then outline our motion planning

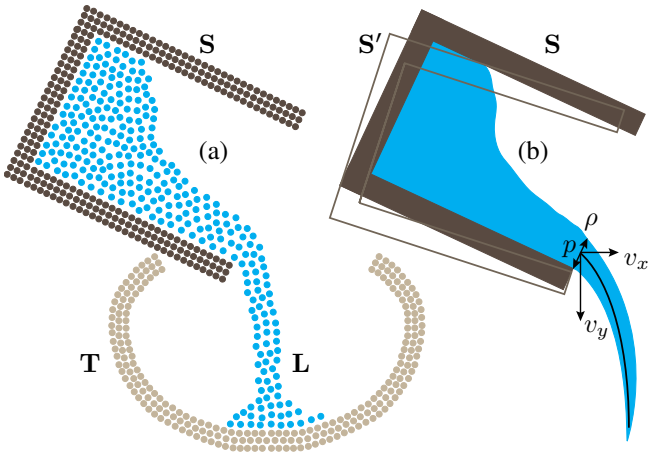


Fig. 2: (a): An illustration of our problem setting. We consider a source container $\mathbf{S}$ in dark gray, a target container $\mathbf{T}$ in light gray, and finally the liquid body $\mathbf{L}$ in blue. We use a particle-based spatial discretization for both the rigid and liquid bodies. (b): The set of learned parameters used for formulating C_o: the mean trajectory prior $\mathbf{S}'$, the parameters of liquid outflow curve p, v_x, v_y, and the outflow flux ρ.

algorithm.

A. PROBLEM FORMULATION

In each problem, we consider a source container denoted by rigid body $\mathbf{S}$, a target container denoted by $\mathbf{T}$, and the liquid body denoted by $\mathbf{L}$. Without ambiguity, we reuse these symbols to denote their spacetime trajectory $\mathbf{S}(t)$, $\mathbf{T}(t)$, and $\mathbf{L}(t)$, where t is the time index. Among these three trajectories, the liquid body trajectory $\mathbf{L}(t)$ is constrained by the Navier-Stokes equation, the governing PDE of the liquid body. Therefore, the configuration of $\mathbf{L}$ evolves as $\mathbf{L}(t + \Delta t) = f(\mathbf{S}(t), \mathbf{T}(t), \mathbf{L}(t))$, taking $\mathbf{S}(t), \mathbf{T}(t)$ as boundary conditions. Here f is a time-integration of the Navier-Stokes equation over timestep size Δt.

In practice, we define a discrete version of the problem using a particle-based spatial discretization scheme, the finite-difference temporal discretization scheme, and the same particle-based fluid simulator as [6]. As a result, all three bodies, $\mathbf{S}, \mathbf{T}$, and $\mathbf{L}$, become a set of particles, as illustrated in Figure 2 (a). Their corresponding trajectories are sampled uniformly in time with timestep size Δt. In this setting, the time-integration function f can be approximately evaluated using a discrete version of the Navier-Stokes equation.

In our problem setting, the target container trajectory $\mathbf{T}(t)$ is given. The goal of our feedback motion planner is then to determine the source container trajectory $\mathbf{S}(t)$, so that the induced liquid body trajectory $\mathbf{L}(t)$ will have the liquid body end up inside the target container $\mathbf{T}$.

V. FEEDBACK MOTION PLANNING

In order to find an entire trajectory $\mathbf{S}(t)$, our motion planner iteratively solves the following spacetime optimization problem:

$$\underset{\mathbf{S}(t+\Delta t),\cdots,\mathbf{S}(t+K\Delta t)}{\mathbf{argmin}} \sum_{1\le k\le K} C(\mathbf{S}(t+k\Delta t)) \tag{1}$$
$$C(\mathbf{S}(t)) \triangleq C_l(\mathbf{S}(t)) + C_r(\mathbf{S}(t)) + C_o(\mathbf{S}(t)),$$

over a horizon of $K\Delta t$ and then only adopts the control $\mathbf{S}(t+\Delta t)$, i.e., using the receding horizon strategy. The objective function C involves three terms: the first term C_l encourages the liquid body to fall inside $\mathbf{T}$, C_r is a regularization term that prefers smooth trajectories, and finally C_o penalizes any collisions between $\mathbf{S}$ and $\mathbf{T}$ or any other obstacles in the environment.

We can use the same C_o as [14], [6] for collision avoidance. Specifically, we define C_o as:

$$C_o(\mathbf{S}(t)) = \|D(\mathbf{S}(t))\|^2 + \|\dot{D}(\mathbf{S}(t))\|^2,$$

where D is the maximal penetration depth between $\mathbf{S}$ and any obstacle. To accelerate collision detection, each rigid object is approximated using a set of spheres. Note that we penalize both the D and $\dot{D}$ to encourage smooth movements when $\mathbf{S}$ is in the vicinity of boundaries. The regularization term $\mathbf{C}_r$ is the Laplace of the trajectory:

$$C_r(\mathbf{S}(t)) = \|\mathbf{S}(t+\Delta t) - 2\mathbf{S}(t) + \mathbf{S}(t-\Delta t)\|^2.$$

In order to define C_o such that it encourages successful liquid pouring, a naive and straightforward formulation would be to reconstruct $\mathbf{L}(t)$ from $\mathbf{S}(t), \mathbf{T}(t)$ using a liquid simulation function f, then to measure the amount of liquid that falls outside the target container and makes C_o proportional to this amount. However, this formulation has two clear drawbacks. First, reconstructing $\mathbf{L}(t)$ using liquid simulation function f is computationally costly, making it impossible for the motion planner to respond in real-time. Second, the function f involves a lot of non-smooth operators, making it difficult for numerical optimizers to find a good local minimum of Equation 1.

Our solution is to base C_o on a set of low-dimensional parameters that can be inferred from an observation of the current configuration of the liquid body $\mathbf{L}(t)$ without resorting to its predicted future configurations. This idea is like the temporal decomposition method [4]. We assume that two kinds of parameters are crucial to the task of pouring liquids. First, in most human pouring examples, the pattern is that the source container $\mathbf{S}$ moves closer to the target container $\mathbf{T}$, simultaneously increasing its turning angle so that liquid can flow out. This common pattern is encoded as mean trajectory prior $\mathbf{S}'(t)$. In addition, after liquid leaves the source container $\mathbf{S}$, the flow can be approximated as a quadratic curve, as shown in [9]. This quadratic curve could be used to guide the motion planner so that the curve is centered around the target container opening, thus avoiding spillage. In a 2D workspace, this curve is characterized by its starting point p on $\mathbf{S}$ and its leaving velocity v_x, v_y, as illustrated in Figure 2 (b). In summary, the required liquid-related parameters are $(\mathbf{S}'(t), p, v_x, v_y)$ and we then define C_l as:

$$\begin{aligned}\mathbf{C}_l(\mathbf{S}(t)) \quad = \quad & \|dist(\mathbf{T}(t), p, v_x, v_y)\|^2 \mathbf{max}(\rho, 0) + \\ & \|\mathbf{S}(t) - \mathbf{S}'(t)\|^2,\end{aligned}$$

where the first term measures the distance between liquid outflow curve and the center of the target container opening,

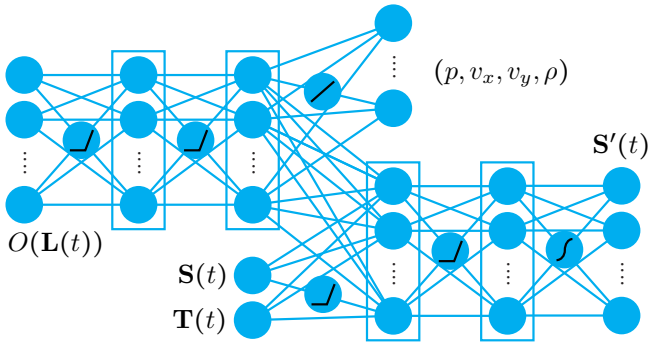


Fig. 3: Our 4-layer neural network structure for parameter estimation. The input is an observation of the current liquid configuration $O(L(t))$ and other rigid body configurations $\mathbf{S}(t), \mathbf{T}(t)$, where we assume the rigid body configurations are fully observable. We use the ReLU activation function for all internal layers and the sigmoid activation function for the output layer of mean trajectory prior $\mathbf{S}'(t)$, and linear function for the output layer of liquid outflow curve parameters. Each of the four hidden layers has 32 hidden units.

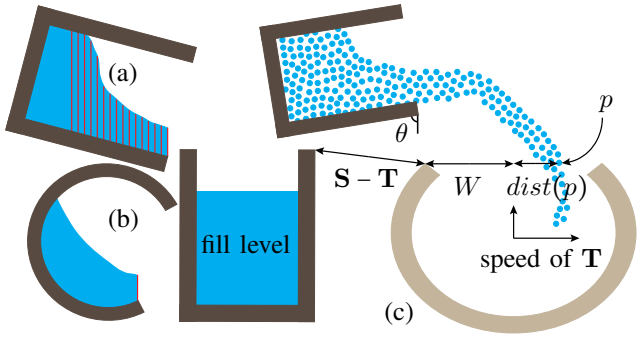


Fig. 4: The two kinds of features we use. (a): The heightfield of the liquid free-surface (red). This feature is used as groundtruth in a simulated environment. (b): The height of liquid at the lip of source container $\mathbf{S}$ (red). This feature makes it easier to apply our method on a real-life robotic system and on different container shapes. (c): The variables that define one instance of our pouring problem: relative position $\mathbf{S}$ – $\mathbf{T}$, speed of $\mathbf{T}$, and fill level. We also illustrate the variables that define our reward function R: the container opening width W and the distance from a particle to a center point $dist(p)$.

while the second term measures the discrepancy between the current source container configuration and mean trajectory prior, leading to a successful pour. Note that we added a weighting $\mathbf{max}(\rho, 0)$, where ρ is the predicted outflow flux. Therefore, we only apply the first term if liquid is flowing out. This ρ is added to the required set of parameters. These parameters $(\mathbf{S}'(t), p, v_x, v_y, \rho)$ are predicted efficiently using supervised learning introduced in Section VI.

VI. SUPERVISED LEARNING

In this section, we present our preprocessing algorithm, which uses supervised learning to predict the liquid-related parameters.

As part of the objective function, the learning algorithm needs to infer the parameters $(\mathbf{S}'(t), p, v_x, v_y, \rho)$ from the observations $O(\mathbf{L}(t))$ of a liquid body and the configurations of the two rigid bodies $\mathbf{S}(t), \mathbf{T}(t)$. The inference is accomplished with a neural network trained from a set of successful liquid pouring trajectories. See Figure 3 for a specification of our neural network.

The problem of inferring mean trajectory prior $\mathbf{S}'(t)$ from the current configuration makes our neural network a control policy representation that can be optimized using reinforcement learning as in [4], [32]. Optimized this way, the policy will predict $\mathbf{S}'(t)$ such that it can be realized directly. Instead, only supervised learning is used in our work, so the learned policy may suffer from simulation-bias issues and have a limited ability to generalize to new environments. However, we can solve this problem by plugging the predicted $\mathbf{S}'(t)$ into Equation 1 to further adjust the predicted result.

The observation function O, or the input feature for the neural network, can be defined in several ways depending on the available sensors in a certain application. In this work, we consider two kinds of observations. In a simulated environment, we use the heightfield of the liquid surface (which can be easily computed from an exact geometrical representation of $\mathbf{L}$, such as a set of particles). However, the dynamics of the liquid body are captured largely by the velocity field instead of its shape. To recover this information, we maintain a short memory of the heightfield over the past 4 frames as input to the neural network so that velocity information can be recovered by finite difference. This is illustrated in Figure 4 (a). Longer-term memory can also be recovered using, e.g., a recurrent neural network or LSTM. However, according to the definition of velocity as the derivative of positions, only the most recent memory is needed and these structures are therefore unnecessary. In a real-life robotic system, acquiring the heightfield may be very difficult. As a result, we also experimented with a simplified feature, which is the height of a liquid surface only at the lip of source container $\mathbf{S}$, as illustrated in Figure 4 (b). A common drawback of the heightfield feature is that we can only represent laminar flow without internal air bubbles in the liquid. However, we can carefully design our training dataset to contain only laminar flow (see Section VII for more details).

A. TRAINING DATA GENERATION

The dataset used to train the neural network is difficult to acquire. This dataset should contain only successful pouring trajectories. Moreover, the neural network should be expressive enough to regenerate these trajectories. Previous works [33], [34] address this problem for some applications by modifying the dataset during training. [33] assumes there is an expert who can provide additional training samples for error recovery on request. However, our approach does not assume the presence of any such expert. A human demonstrator may serve as an expert, but digitizing or capturing the liquid shape trajectory can be very challenging. On the other hand, [34] assumes that the governing equation is differentiable with respect to u. This assumption does not hold due to the non-smooth free-surface changes. Moreover, the computational complexity of [34] makes it infeasible for the high-dimensional configuration space of a liquid body.

Our solution is to use stochastic optimization to automatically search for successful pouring trajectories similar to [5] in 2D workspaces. We introduce several kinds of variations so that the learned neural network can be generalized to different problems. As illustrated in Figure 4 (a), each pouring problem can be specified by three variables:

- The relative position $\mathbf{S} - \mathbf{T}$ in range $[-3, 0] \times [-3, 3](m)$.
- The constant moving speed of $\mathbf{T}$ in range $[-5, 5]^2(m/s)$.
- The liquid fill level of $\mathbf{S}$ in range $[0.3, 0.8]$, where 1 means fully filled.

To quickly find a large number of successful pouring trajectories, we design a problem specific search space and reward function. Our liquid simulator requires very small timestep size ($\Delta t < 0.01s$) to ensure accuracy, leading to a large number of timesteps. We first limit the number of variables by using spline interpolation with 6 control points for source container trajectory $\mathbf{S}(iK\Delta t/5)$, where $0 \leq i \leq 5$. In 2D workspaces, each rigid configuration of $\mathbf{S}$ consists of 2-dimensional translation and orientation $(\mathbf{x_S}, \mathbf{y_S}, \theta_\mathbf{S})$, leading to a 15 dimensional search space (the initial control point is fixed). However, we found that this is still too large of a search space in practice because the optimizer can frequently generate zig-zag trajectories, contrary to our intuitive observation of human pouring behaviour. Therefore, we further restrict the search space by observing that source container $\mathbf{S}$ is always moving closer to $\mathbf{T}$ and its turning angle is always increasing. This gives the following relationship:

$$(\alpha_i \triangleq \frac{|\mathbf{x_S} - \mathbf{x_T}|_i}{|\mathbf{x_S} - \mathbf{x_T}|_{i-1}}, \beta_i \triangleq \frac{|\mathbf{y_S} - \mathbf{y_T}|_i}{|\mathbf{y_S} - \mathbf{y_T}|_{i-1}}, \gamma_i \triangleq \frac{|\theta_{max} - \theta|_i}{|\theta_{max} - \theta|_{i-1}}).$$

We propose to search in the transformed coordinates $(\alpha_i, \beta_i, \gamma_i) \in (0, 1]^3$ using the CMA-ES algorithm [35]. Although this is still a 15 dimensional search space, much fewer random samples are needed in each iteration with such a transformation. Finally, we use the following reward function for CMA-ES optimization, which encourages particles to pass through the center of a target container's opening and penalizes spillage:

$$R = \sum_p R_p \quad R_p = \begin{cases} \frac{W - dist(p)}{W}, & \text{if } dist(p) < W \\ -100, & \text{otherwise} \end{cases}$$

where p loops over all particles, $dist(p)$ is its distance to the center of the opening of $\mathbf{T}$, and W is half the width of the opening, as illustrated in Figure 4 (b).

After we find the optimal $\mathbf{S}(t)$ in our search space, we extract the groundtruth observation $O(\mathbf{L}(t))$ and label $(\mathbf{S}'(t), p, v_x, v_y, \rho)$ for each spline interpolated timestep. To extract the quadratic curve parameters, we use the same greedy quadratic curve fitting method as [6]. Finally, our neural-network outputs the transformed coordinates $(\alpha_i, \beta_i, \gamma_i)$, instead of $(\mathbf{x_S}, \mathbf{y_S}, \theta_\mathbf{S})$, which fit in the range of sigmoid activation function.

VII. ANALYSIS AND RESULTS

In this section, we evaluate the online and offline phases in Figure 1 from different aspects.

Quality of Dataset: All the trajectories in our dataset reside in 2D workspaces. The computational complexity of generating these trajectories is $\mathcal{O}(I \times R \times K \times P)$, where I is the number of iterations needed in each CMA-ES optimization, R is the number of random samples used in each CMA-ES iteration, K is the number of timesteps in each fluid simulation, and finally P is the number of trajectories $\mathbf{S}(t)$ we want to generate. In all the optimizations, we set $I = 200$, $R = 30$, $K = 500$, and $\Delta t = 0.01(sec)$.

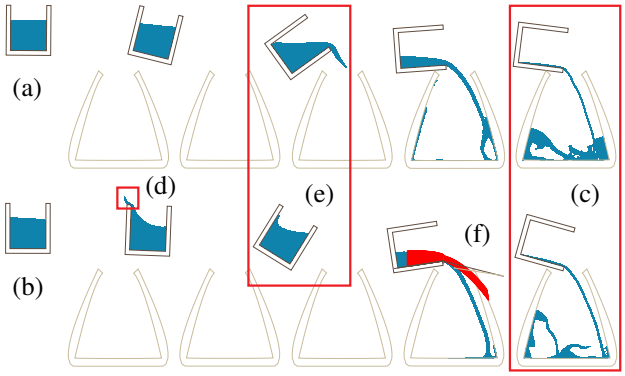


Fig. 6: We illustrate an exemplary trajectory of TRANSFER+FOLLOW (a) and TRANSFER+ZERO (b). On convergence of CMA-ES optimization, the liquid flow is well centered around the opening of $\mathbf{T}$ (c). TRANSFER+ZERO encourages spillage at an early stage of pouring (d), so that $\mathbf{S}$ must move and turn slowly (e). For each timestep in each trajectory, we extract the groundtruth water heightfield and liquid outflow curve as in training dataset (f).

In other words, each trajectory lasts for $K\Delta t = 5(sec)$. The computational complexity of dataset generation is also proportional to the overhead of performing liquid simulation, i.e., evaluating function f, which in turn is proportional to the number of particles. In 2D workspaces, we use approximately 10^5 particles and each evaluation of f takes $2.5(sec)$ while the number of particles in 3D workspaces is 10^7 and each evaluation takes $345(sec)$. Therefore, generating a 3D dataset is orders of magnitude more expensive than generating a 2D dataset.

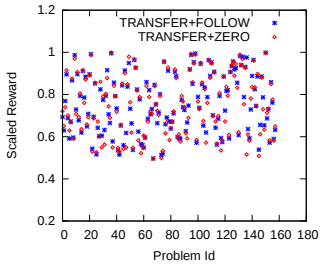


Fig. 5: The scaled reward function $R/\#Particle$ for a set of transfer problems in TRANSFER+FOLLOW (blue) and TRANSFER+ZERO (red). These values are all positive, meaning that very little spillage happens and particles are well centered around $\mathbf{O_T}$.

We generated two datasets named TRANSFER+FOLLOW and TRANSFER+ZERO, each containing $P = 1000$ successful pouring trajectories. We first sample the fill level with interval 0.12 and then select 200 relative positions $\mathbf{S} - \mathbf{T}$ and select a moving speed of $\mathbf{T}$ for each fill level using uniform random sampling in the given range. The generation of TRANSFER+FOLLOW and TRANSFER+ZERO involves a total of 6×10^9 evaluations of f. TRANSFER+FOLLOW and TRANSFER+ZERO differ in their initial liquid configuration. In TRANSFER+FOLLOW, the initial liquid velocity follows that of $\mathbf{S}$, which is typical if the liquid has moved with $\mathbf{S}$ for a certain distance and reached equilibrium. However, in TRANSFER+ZERO, the initial liquid velocity is zero, which is typical if we start the transfer from a stationary scenario. Problems in TRANSFER+ZERO are considered more difficult than those in TRANSFER+FOLLOW, as moving $\mathbf{S}$ too quickly will lead to spillage and thus negative reward. These two datasets can be downloaded at our project page and are illustrated in Figure 6.

Figure 5 shows the distribution of scaled reward function $R/\#Particle$ in TRANSFER+FOLLOW and TRANS-

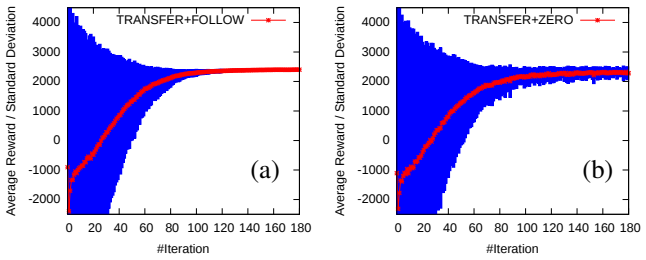


Fig. 7: The average convergence history of the CMA-ES algorithm over 1000 problems. This algorithm converges equally well for TRANSFER+FOLLOW (a) and TRANSFER+ZERO (b).

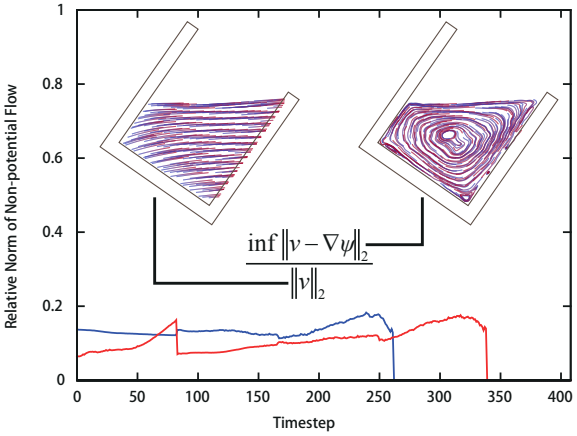


Fig. 8: We picked 100 random trajectories in TRANSFER+FOLLOW (blue) and TRANSFER+ZERO (red) datasets, and visualized the temporal change of the relative strength of the vortical velocity component (the numerator $\mathbf{inf}\|v-\nabla\psi\|_2$). Since this value is always less than 0.2, the flow is very close to a potential flow ($\mathbf{inf}\|v-\nabla\psi\|_2 = 0$). Since problems in the TRANSFER+FOLLOW dataset are less difficult, the pouring is usually completed faster, which is consistent with the early termination of the blue curve.

FER+ZERO. Figure 7 shows the CMA-ES convergence history. These figures show that our stochastic optimization algorithm can efficiently find successful transfer trajectories. Furthermore, we need to verify that the generated trajectories always transfer liquid using laminar flow, rather than turbulent flow, so that a heightfield feature can represent the shape of $\mathbf{L}$. We note that the velocity field of a laminar flow should have no internal vortex. Therefore, we compute the vortical velocity component and plot its strength relative to the original velocity field in Figure 8. This figure clearly shows that the trajectories in both TRANSFER+FOLLOW and TRANSFER+ZERO use only laminar flow.

Accuracy of Neural Network: Our neural network serves two purposes, which are verified by separate experiments.

We first verify the accuracy of the liquid outflow curve predictor by testing it on 100 randomly selected trajectories in TRANSFER+FOLLOW/TRANSFER+ZERO, i.e., we test it using the training dataset. Next, we test it on 100 new trajectories, $\mathcal{P}$, held out from training data. Figure 9 (a) plots the average accuracy against the turning angle α. This plot reveals that our predictor has low accuracy at a small turning angle (up to 0.46% at $40° \pm 10°$). The average relative accuracy throughout the pouring process is 15% using the heightfield feature, Figure 4 (a), and 24% using the height at lip feature, Figure 4 (b). At a small turning angle, using

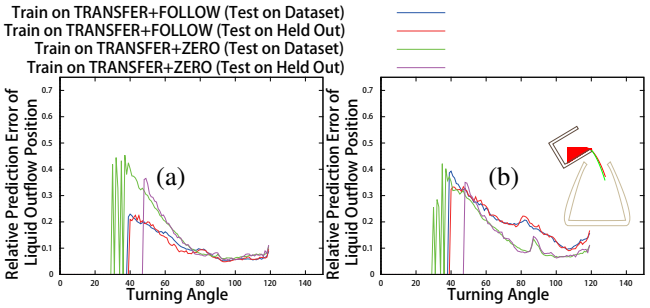


Fig. 9: Relative error of predicted liquid outflow curve parameters ($\|v_x, v_y, p\|^2$) using the heightfield feature (a) and the height at lip feature (b). The error is plotted against the turning angle of $\mathbf{S}$.

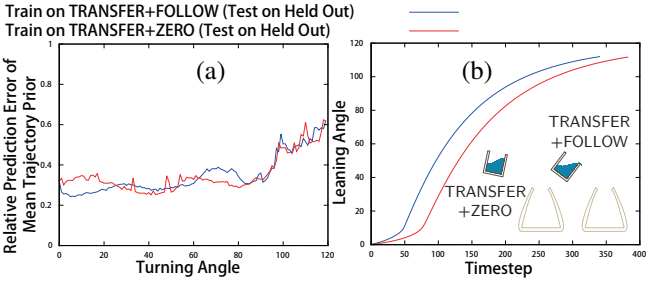


Fig. 10: (a): The relative error of predicted $\mathbf{S}(t)$ plotted against the turning angle. (b): The predicted turning angle of $\mathbf{S}$ against timestep index; the mean trajectory prior predictor trained using TRANSFER+ZERO dataset (red) learns to turn $\mathbf{S}$ slowly to avoid spillage, compared with the predictor trained using TRANSFER+FOLLOW dataset (blue). We also show such difference using one exemplary frame of a testing problem.

the height at lip feature increases the error by 36% for TRANSFER+FOLLOW dataset, while the performance is almost the same for TRANSFER+ZERO dataset. At a larger turning angle, using the height at lip feature only increases the error by 10%, at most.

We further validate the accuracy of the mean trajectory prior predictor. Figure 10 (a) illustrates the relative accuracy of $\mathbf{S}(t)$ compared with the groundtruth. The error is higher than that of the liquid outflow curve predictor. One of the main purposes of our online feedback motion planner is to compensate for this error. In addition, we conduct another experiment to see if our predictor has the ability to avoid spillage. Since we know that TRANSFER+FOLLOW does not model spillage but TRANSFER+ZERO can model spillage, we plot the predicted temporal change of α using TRANSFER+FOLLOW and TRANSFER+ZERO as the training dataset. Our predictor trained using the TRANSFER+ZERO dataset prefers a lower $\dot{\alpha}$ at small α, which implies that our network can learn the ability to avoid spillage.

Performance of Feedback Planner: In the online phase, we solve Equation 1 using a horizon length of $1.25(sec)$, with $K = 25$ and $\Delta t = 0.05(sec)$. This requires querying the neural network 25 times and performing an optimization using the LBFGS optimizer, which is very efficient due to the small size of our neural network and the optimization formulate. We tested our feedback controller on 30 new problems. For each testing problem, we experimented with three different container shapes, as illustrated in Figure 11

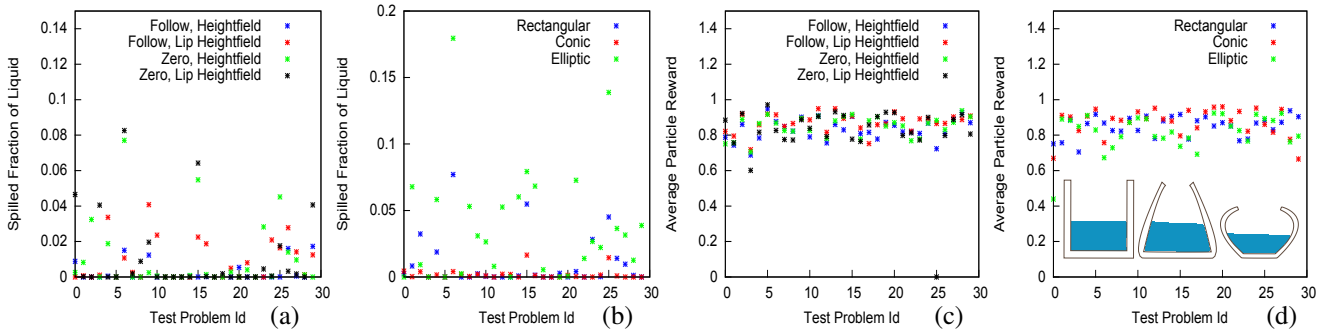


Fig. 11: Performance of the feedback planner. (a): The fraction of spilled particles using rectangular container $\mathbf{S}$, two kinds of datasets and features. (b): The fraction of spilled particles using the height at lip feature and three different container shapes. (c): Average reward of particles that fall into $\mathbf{T}$ in experiment (a). (d): Average reward of particles that fall into $\mathbf{T}$ in experiment (b).

(d). We use the same set of 30 experimental problems, which are not covered in the datasets. If the training is accomplished using TRANSFER+ZERO dataset, then we set the initial velocity to be zero and vice versa. For experiments using new container shapes, we use TRANSFER+ZERO dataset.

The performance of our planner is summarized in Figure 11, where we plot the spilled fraction of liquid and the average reward. Note that the average is taken over particles that fall into $\mathbf{T}$ (i.e., excluding the spilled particles), which is different from Figure 5, where the average includes negative values (spilled particles). We can thus analyze spillage avoidance and the accuracy of liquid flow, respectively. From Figure 11 (a), we can see that the fraction of spilled particles is less than 5% in 28 of 30 problems for both features in Figure 4. We also see that using the lip height feature increases the spillage by 4%, at most, and 0.32% on average. From Figure 11 (b), we find that, although our datasets use only rectangular containers, the spillage fraction is also very small if we test on a conic container, which results in a spillage fraction over 5% for only 2 problems. However, some containers, such as the conic container, encourage spillage and we have seen over 5% spillage in 10 problems. From Figure 11 (c), we can see that the liquid flow is generally well centered around the center opening of $\mathbf{T}$, with a mean reward of $R_p = 0.82$. If we generalize to new container shapes, the mean reward is still over $R_p = 0.8$. However, the variance is larger especially for the conic container, as illustrated in Figure 11 (d).

Generalizing to New Problems: Our motion planner is only experimented on liquid bodies of fixed material parameters, on a 2D workspace, and with two rigid bodies. Some of these limitations can be relaxed.

First, although our dataset considers only a liquid material with low viscosity, applying it to pour more viscous liquids is possible. As illustrated in Figure 12, we experimented the planner on these liquids. As we increase the viscosity, the fraction of spillage decreases and the average reward increases accordingly. This is because viscosity suppresses turbulence flow and makes our quadratic curve assumption more accurate.

Moreover, we can also relax the assumption of 2D workspaces and apply the method to 3D workspaces. As illustrated in Figure 12 (c), this is done by applying the method to only the symmetric cross section. A drawback of this treatment is that fluid may spill from directions other than the 2D plane and our method cannot avoid such spillage.

Finally, a major benefit of the optimization-based formulation, Equation 1, is that we can naturally take other dynamic obstacles into consideration without retraining or regenerating the dataset, as illustrated in Figure 12 (b). However, current obstacle avoidance term only takes rigid bodies into consideration, and collisions between liquid and rigid bodies are not considered. As a result, when the obstacle blocks the pouring, more spillage happens.

VIII. LIMITATIONS AND CONCLUSIONS

In conclusion, we present a feedback motion planner for liquid transfer problems. Our formulation uses an optimization-based receding horizon planner, which is guided by a machine learning model that provides clues for global movements (mean trajectory prior) and local adjustments (liquid outflow curve). Our experiments show that the planning framework can achieve promising online performance and the machine learning model gains important skills such as spillage avoidance and liquid position prediction. However, the current system also introduces a series of limitations, which we discuss below. We refer readers to our extended paper [36] for a thorough discussion.

First, although the spillage fraction in our dataset is almost zero, the spillage fraction can be as high as 5% in our 30 testing problems. We believe better performance can be achieved by using imitation learning [33] instead of supervised learning. Moreover, it is inherently difficult to generalize our method to discrete material types, such as a bunch of rigid bodies or granular materials as in [28], [3]. In addition, we only consider the problem of pouring the entire liquid body from $\mathbf{S}$ to $\mathbf{T}$. Other requirements might arise, e.g., if only some of liquid is needed in $\mathbf{T}$ due to the small volume of $\mathbf{T}$. Finally, two problems need to be addressed for extension to real robotic systems. We must be able to efficiently acquire liquid-related features and we must be able to extend the entire planning pipeline to 3D workspaces.

ACKNOWLEDGEMENT

This research is supported in part by ARO Contract W911NF-14-1-0437, NSF award 1305286.

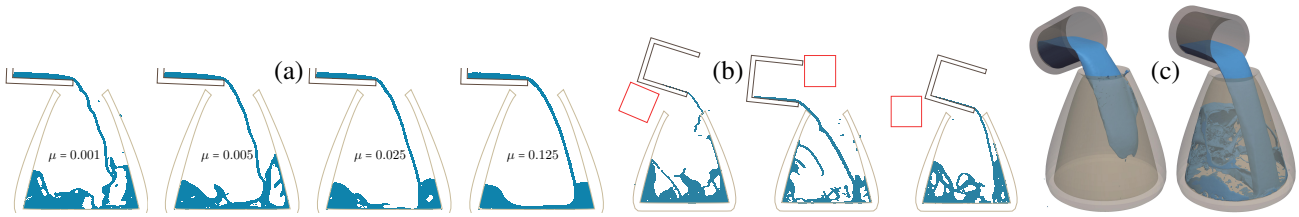


Fig. 12: (a): We tested our feedback controller on new fluid materials with higher dynamic viscosity ($kg/(ms)$). In this case, we achieved even higher online reward. (b): The controller can also account for dynamic obstacles. (c): In a 3D workspace, the controller only sees the 2D cross section.

REFERENCES

[1] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "STOMP: Stochastic trajectory optimization for motion planning," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2011, pp. 4569–4574.

[2] J. Schulman, Y. Duan, J. Ho, A. Lee, I. Awwal, H. Bradlow, J. Pan, S. Patil, K. Goldberg, and P. Abbeel, "Motion planning with sequential convex optimization and convex collision checking," *The International Journal of Robotics Research*, vol. 33, no. 9, pp. 1251–1270, 2014.

[3] A. Yamaguchi and C. G. Atkeson, "Neural networks and differential dynamic programming for reinforcement learning problems," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 5434–5441.

[4] ——, "Differential dynamic programming with temporally decomposed dynamics," in *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*. IEEE, 2015, pp. 696–703.

[5] Y. Kuriyama, K. Yano, and M. Hamaguchi, "Trajectory planning for meal assist robot considering spilling avoidance," in *Control Applications, 2008. CCA 2008. IEEE International Conference on*. IEEE, 2008, pp. 1220–1225.

[6] Z. Pan, C. Park, and D. Manocha, "Robot motion planning for pouring liquids," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2016.

[7] L. Kunze, M. E. Dolha, E. Guzman, and M. Beetz, "Simulation-based temporal projection of everyday robot object manipulation," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 107–114.

[8] M. Tzamtzi and F. Koumboulis, "Robustness of a robot control scheme for liquid transfer," in *Novel Algorithms and Techniques In Telecommunications, Automation and Industrial Electronics*. Springer, 2008, pp. 156–161.

[9] Z. Pan and D. Manocha, "Motion planning for fluid manipulation using simplified dynamics," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2016.

[10] S. M. Lavalle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[11] D. Wilkie, J. P. van den Berg, and D. Manocha, "Generalized velocity obstacles," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009, pp. 5573–5578.

[12] M. Stilman, "Task constrained motion planning in robot joint space," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 3074–3081.

[13] D. Berenson, S. S. Srinivasa, D. Ferguson, A. Collet, and J. J. Kuffner, "Manipulation planning with workspace goal regions," in *Proceedings of IEEE International Conference on Robotics and Automation*, 2009, pp. 618–624.

[14] C. Park, J. Pan, and D. Manocha, "ITOMP: Incremental trajectory optimization for real-time replanning in dynamic environments," in *Proceedings of International Conference on Automated Planning and Scheduling*, 2012.

[15] D. Ferguson, N. Kalra, and A. Stentz, "Replanning with rrts," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 1243–1248.

[16] K. Hauser, "On responsiveness, safety, and completeness in real-time motion planning," *Autonomous Robots*, vol. 32, no. 1, pp. 35–48, 2012.

[17] S. Koenig and M. Likhachev, "Fast replanning for navigation in unknown terrain," *IEEE Transactions on Robotics*, vol. 21, no. 3, pp. 354–363, 2005.

[18] H. Kurniawati, D. Hsu, and W. S. Lee, "Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces," in *In Proc. Robotics: Science and Systems*, 2008.

[19] S. Levine and P. Abbeel, "Learning neural network policies with guided policy search under unknown dynamics," in *Advances in Neural Information Processing Systems*, 2014, pp. 1071–1079.

[20] C. Park, J. Pan, and D. Manocha, "High-dof robots in dynamic environments using incremental trajectory optimization," *International Journal of Humanoid Robotics*, vol. 11, no. 02, p. 1441001, 2014.

[21] J. Schulman, J. Ho, C. Lee, and P. Abbeel, "Learning from demonstrations through the use of non-rigid registration," in *Robotics Research*. Springer, 2016, pp. 339–354.

[22] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen, "Folding deformable objects using predictive simulation and trajectory optimization," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 6000–6006.

[23] N. Chentanez, R. Alterovitz, D. Ritchie, L. Cho, K. K. Hauser, K. Goldberg, J. R. Shewchuk, and J. F. O'Brien, "Interactive simulation of surgical needle insertion and steering," in *Proceedings of ACM SIGGRAPH 2009*, Aug 2009, pp. 88:1–10. [Online]. Available: http://graphics.berkeley.edu/papers/Chentanez-ISN-2009-08/

[24] A. Treuille, A. McNamara, Z. Popović, and J. Stam, "Keyframe control of smoke simulations," in *ACM Transactions on Graphics (TOG)*, vol. 22, no. 3. ACM, 2003, pp. 716–723.

[25] Z. Pan and D. Manocha, "Motion planning for fluid manipulation using simplified dynamics," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*. IEEE, 2016, pp. 4224–4231.

[26] X. B. Peng, G. Berseth, and M. van de Panne, "Terrain-adaptive locomotion skills using deep reinforcement learning," *ACM Transactions on Graphics (Proc. SIGGRAPH 2016)*, vol. 35, no. 5, 2016, to appear.

[27] C. Bowen and R. Alterovitz, "Closed-loop global motion planning for reactive execution of learned tasks," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1754–1760.

[28] A. Yamaguchi, C. G. Atkeson, S. Niekum, and T. Ogasawara, "Learning pouring skills from demonstration and practice," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 908–915.

[29] S. Brandi, O. Kroemer, and J. Peters, "Generalizing pouring actions between objects using warped parameters," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 616–621.

[30] H. Wang, M. Liao, Q. Zhang, R. Yang, and G. Turk, "Physically guided liquid surface modeling from videos," in *ACM Transactions on Graphics (TOG)*, vol. 28, no. 3. ACM, 2009, p. 90.

[31] J. Gregson, M. Krimerman, M. B. Hullin, and W. Heidrich, "Stochastic tomography and its applications in 3d imaging of mixing fluids." *ACM Trans. Graph.*, vol. 31, no. 4, pp. 52–1, 2012.

[32] I. Mordatch, K. Lowrey, G. Andrew, Z. Popovic, and E. V. Todorov, "Interactive control of diverse complex characters with neural networks," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 3132–3140.

[33] S. Ross, G. J. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS-11)*, G. J. Gordon and D. B. Dunson, Eds., vol. 15. Journal of Machine Learning Research - Workshop and Conference Proceedings, 2011, pp. 627–635. [Online]. Available: http://www.jmlr.org/proceedings/papers/v15/ross11a/ross11a.pdf

[34] S. Levine and V. Koltun, "Learning complex neural network policies with trajectory optimization," in *ICML '14: Proceedings of the 31st International Conference on Machine Learning*, 2014.

[35] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[36] Z. Pan and D. Manocha, "Feedback motion planning for liquid transfer using supervised learning," *https://arxiv.org/abs/1609.03433*.