

**DISTRIBUTED SPECTRAL GRAPH METHODS
FOR ANALYZING LARGE-SCALE
UNSTRUCTURED BIOMEDICAL DATA**

by

Shannon Patrick Quinn

M.S. Computational Biology, Carnegie Mellon University, 2010

B.S. Computer Science, Georgia Institute of Technology, 2008

Submitted to the Graduate Faculty of
the Department of Computational and Systems Biology in partial
fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2014

UNIVERSITY OF PITTSBURGH
DEPARTMENT OF COMPUTATIONAL AND SYSTEMS BIOLOGY

This dissertation was presented

by

Shannon Patrick Quinn

It was defended on

24 November 2014

and approved by

Dr. Chakra Chennubhotla, Computational and Systems Biology

Dr. Takis Benos, Computational and Systems Biology

Dr. Cecilia Lo, Developmental Biology

Dr. Arvind Ramanathan, Oak Ridge National Laboratory

Dr. Russell Schwartz, Carnegie Mellon University Department of Biology

Dr. Lans Taylor, Drug Discovery Institute

Dissertation Director: Dr. Chakra Chennubhotla, Computational and Systems Biology

DISTRIBUTED SPECTRAL GRAPH METHODS FOR ANALYZING LARGE-SCALE UNSTRUCTURED BIOMEDICAL DATA

Shannon Patrick Quinn, PhD

University of Pittsburgh, 2014

There is an ever-expanding body of biological data, growing in size and complexity, outstripping the capabilities of standard database tools or traditional analysis techniques. Such examples include molecular dynamics simulations, drug-target interactions, gene regulatory networks, and high-throughput imaging. Large-scale acquisition and curation biological data has already yielded results in the form of lower costs for genome sequencing and greater coverage in databases such as GenBank, and is viewed as the future of biocuration. The “big data” philosophy and its associated paradigms and frameworks have the potential to uncover solutions to problems otherwise intractable with more traditional investigative techniques.

Here, we focus on two biological systems whose data form large, undirected graphs. First, we develop a quantitative model of ciliary motion phenotypes, using spectral graph methods for unsupervised latent pattern discovery. Second, we apply similar techniques to identify a mapping between physiochemical structure and odor percept in human olfaction. In both cases, we experienced computational bottlenecks in our statistical machinery, necessitating the creation of a new analysis framework. At the core of this framework is a distributed hierarchical eigensolver, which we compare directly to other popular solvers. We demonstrate its essential role in enabling the discovery of novel ciliary motion phenotypes and in identifying physiochemical-perceptual associations.

TABLE OF CONTENTS

PREFACE	xi
1.0 INTRODUCTION	1
1.1 Petabyte-scale biomedicine	1
1.2 Contributions	5
1.3 Thesis Outline	6
1.4 Publications	8
1.5 Mathematical Notation	9
2.0 SPECTRAL GRAPH ANALYTICS	11
2.1 Introduction	11
2.2 Graph structures and properties	12
2.2.1 Graph affinities and neighborhoods	12
2.2.2 Random walks	16
2.3 Spectral clustering	17
2.4 Methods for finding eigenvalues and eigenvectors	21
2.4.1 Direct solvers	21
2.4.2 Iterative solvers	22
2.4.3 Hierarchical and multigrid solvers	22
2.4.4 Geometric multigrid (GMG) eigensolvers	23
2.4.5 Algebraic or Combinatorial multigrid (AMG, CMG) eigensolvers	27
2.5 Parallel and distributed implementations	29
2.6 Conclusions	30
3.0 LEARNING CILIARY MOTION PHENOTYPES	32

3.1	Introduction	32
3.2	Data acquisition and study design	35
	3.2.1 Subject recruitment and data cohort breakdown	35
	3.2.2 Digital video annotation and preprocessing	39
	3.2.3 Software	41
3.3	Representing ciliary motion as dynamic textures	41
3.4	Derivation of differential invariants	44
	3.4.1 Optical Flow	44
	3.4.2 Differential Invariants	45
	3.4.2.1 Divergence	46
	3.4.2.2 Curl	47
	3.4.2.3 Deformation	47
	3.4.3 Differential feature filters	47
3.5	Computing digital signatures of ciliary motion phenotypes	49
	3.5.1 Autoregressive models	49
	3.5.2 Magnitude and frequency histograms	53
3.6	Classification of digital signatures	56
	3.6.1 Structure of SVM input	56
	3.6.2 Classifier design for CM recognition	57
	3.6.3 Cross-validation and consensus diagnosis	58
	3.6.4 Results of CM classification	59
3.7	Unsupervised discovery of novel motion phenotypes	68
	3.7.1 Automated region selection	69
	3.7.2 Spectral clustering of AR parameters	71
	3.7.3 Large-scale analysis	76
3.8	Conclusions	77
4.0	LEARNING PERCEPTUAL OLFACTORY DIMENSIONS	78
4.1	Introduction	78
4.2	Dravnieks odor profile and physiochemical descriptors	80

4.2.1	Nonnegative matrix factorization to determine ground-truth odor percepts	80
4.2.2	Software	83
4.3	Derivation of a generalized odorant similarity metric	84
4.3.1	Diagonal Metric	87
4.3.2	Full Metric	87
4.3.3	Populating constraint sets \mathcal{S} and \mathcal{D}	88
4.4	Using the similarity metric to improve odorant classification	89
4.4.1	Substituting each odorant \vec{x} with $G^{1/2}\vec{x}$	90
4.4.2	Projecting each odorant \vec{x} using leading eigenvectors of G	90
4.4.3	Baseline methods for comparison to G	90
4.4.4	Classification results	92
4.4.5	Effects of downsampling \mathcal{S} and \mathcal{D} on classification	100
4.5	Comparison to other metrics	103
4.5.1	Euclidean distance	103
4.5.2	Cosine angle	103
4.5.3	Alternative descriptor sets	105
4.5.4	Alternative models of olfaction	105
4.6	Physiochemical signatures uniquely identify odor percepts	106
4.7	Quantitative evidence for 10 perceptual categories	112
4.8	Semi-supervised propagation of odor percepts to unobserved odorants	117
4.8.1	Graph kernels for semi-supervised learning	118
4.8.2	Using PubChem to test large-scale odor-percept mapping	119
4.8.3	Semi-supervised odor percept propagation	120
4.9	Conclusions	122
4.10	Appendix: List of physiochemical properties used	124
5.0	DRAENOR: A DISTRIBUTED SCIENTIFIC COMPUTING FRAME- WORK	128
5.1	Introduction	128
5.2	Distributed Computing	128

5.3	Practical distributed hierarchical eigensolvers	129
5.3.1	Language and architecture	130
5.3.2	Geometric multigrid	131
5.3.3	Algebraic multigrid	134
5.4	Experiments	137
5.4.1	Complexity analysis	137
5.4.1.1	Distributed geometric multigrid	137
5.4.1.2	Distributed algebraic multigrid	138
5.4.2	Image analysis	139
5.5	Discussion	141
6.0	CONCLUSIONS	149
6.1	Future Directions	150
6.1.1	Ciliary motion analysis	150
6.1.2	Perceptual olfactory recognition	151
6.1.3	Large-scale scientific computing	152
	BIBLIOGRAPHY	154

LIST OF TABLES

1	Description and breakdown of datasets	37
2	Classification results.	62
3	AR results.	62
4	Histogram results using CHP dataset.	63
5	Histogram results using CNMC dataset.	64
6	Classification results for the linear SVM.	94
7	Classification results for the nonlinear SVM.	95
8	Top 60 physiochemical features.	97
9	Top 13 physiochemical features.	98
10	Effects of downsampling similarity constraints.	100
11	The properties listed are those that have a z-score of at least 1.0.	110
12	Descriptors with z-scores of at least 2.0.	111
13	Four images used to conduct initial tests of the Draenor GMG.	145
14	Empirical eigensolver runtime.	147

LIST OF FIGURES

1	Era of “Big Data.”	2
2	Graphs.	4
3	Undirected vs Directed graphs.	13
4	Affinity matrices for corresponding 2D data.	15
5	2D random walks.	17
6	K-means vs spectral clustering.	19
7	Graph spectra.	20
8	Hierarchical schematic.	24
9	Pixel neighborhoods and graph structure.	25
10	Properties of ciliary motion.	33
11	CHP and CNMC dataset breakdowns.	38
12	Website proof-of-concept.	40
13	Derivation of elemental components.	43
14	Basic distortion types.	47
15	Atomic flow detectors.	48
16	Autoregressive representation of ciliary motion.	50
17	Pairwise angles between principal components of CM.	52
18	CM histogram representations.	54
19	Pixel selection.	55
20	Classification pipeline.	60
21	CM AR model representations.	66
22	Classification accuracy as a function of ROIs per patient.	67

23	Automated patch selection.	70
24	CM subtypes.	74
25	3D space of CM subtypes.	75
26	Physiochemical properties.	81
27	Dravnieks odor profile database.	82
28	Learning algorithm.	85
29	Outcome of PCA on metric.	91
30	All classification results.	96
31	Classification accuracy using feature selection.	99
32	Effects of downsampling.	101
33	Classification accuracy with principal components.	102
34	Energy vs angle.	104
35	Physiochemical z-scores.	108
36	Properties with significant z-scores.	109
37	3D projections of odorants.	113
38	Submatrix containing CIM and thermodynamics properties.	114
39	Baseline classification accuracy.	116
40	Odor space organization and chemical similarity.	119
41	Decimation process.	145
42	Built-in eigensolver compared to distributed GMG.	146
43	Large-scale GMG performance.	147
44	Spark implementation bottlenecks.	148

PREFACE

ACKNOWLEDGEMENTS

Brace yourselves¹; I have a lot of wonderful people I want to thank and acknowledge.

First and foremost, I want to acknowledge my advisor, Dr. Chakra Chennubhotla. I cannot enumerate all the ways in which he has truly been the best advisor I could have possibly asked for, but I will start by saying he is singularly responsible for keeping me motivated, interested, and excited whenever we hit dead ends, or had a manuscript rejected from the 27th journal we had submitted it to, or when administrative bureaucracy threatened a project that had otherwise unassailable momentum. He respected and encouraged my out-of-the-lab pursuits (training for and running marathons takes a lot of time!), and regularly inquired about when my next race was or how the most recent one had gone. He gave me free rein with bringing my own skill set and interests to whatever problem we were working on, and made himself readily available to be inundated with questions when I inevitably (and *frequently*) hit a roadblock. In my humble opinion, Chakra should skip Associate Professor and go straight to a full Professor, to say nothing of tenure (as in, he should get tenure). His enthusiasm and desire to learn is infectious and I cannot emphasize enough how critical that was to my own personal success.

I want to thank the rest of my committee for their hard work and dedication to guiding me on the path to graduation this past year. I have worked with some of them as fellow collaborators, as a rotation student, as a summer subcontractor, as a student in their class; for some, I had only heard of the ways in which they transformed their respective fields. Each of them brings their own experience and wisdom to my committee, and I cannot thank them

¹<http://i.imgur.com/6vdT0uq.png>

enough, especially since they have to read this document. I hope they will enjoy reading it, and enjoy even more when they have finished reading it.

My fellow classmates of the (incoming?) class of 2010: Aaron, Devin, Jose, Matt, Murat Can, and Salim, who provided an invaluable social backdrop to the entire experience, bred out of commiseration during our core courses in the first two years. They have since become my workout and racquetball buddies, my board gaming partners, and even my groomsmen. I can only hope we keep in touch once we all scatter; they have truly made this experience more than the sum of its parts. L4D2, anyone?

To all the wonderful folks I have met in Pittsburgh through running, in particular: Danielle and Jose, Kim and Scott, Kelly, and Mark. I want to thank Tim and Alys as well; even though I knew them before I got into running, the Ragnar DC relay was really the start of regular interactions. Running has been an integral and absolutely essential part of me for the past four years helping to keep the stress manageable, in no small part because of these wonderful, supportive, and perpetually-hopped-up-on-endorphins folks.

I also want to convey a heartfelt thank-you to Ellen, who I have known since high school and who also ended up in Pittsburgh working for the university. Her support and incredible wisdom and common sense have helped keep both myself and my wife rooted as we have developed our lives in Pittsburgh both inside and outside of work. She has an unflappable work ethic that is contagious, and a selfless devotion to her friends that is humbling. I am (selfishly) grateful that her parents recently moved back to Atlanta; hopefully my wife and I will be able to continue to enjoy her company over the coming years.

Speaking of supportive, I want to thank the members of the dissertation group I have had the pleasure of meeting regularly with for the past year, specifically: Alba, Andrea, Aubrey, Chelsea, Peter, Prerna, and my Panera writing partner emeritus Nathan; our fearless leader Kym; and my individual mentor Kelly, all of whom have contributed immensely to helping me maintain a relatively healthy and functional degree of crazy throughout the dissertation process. The constant outpouring of support and wealth of insight has made all the difference.

I want to thank Tess, another Atlanta transplant who I've known since grade school who also ended up in Pittsburgh. The morning Insanity and weightlifting workouts, early mornings on the track, joint Java programming and thesis writing sessions, and innumer-

able social outings have been enjoyable beyond measure and helped maintain a degree of “normalcy” throughout my graduate years. I take comfort knowing she will make occasional trips south to visit her family, and hopefully our paths will continue to cross.

To my dear Kat, who I have known since high school (although we too went to the same grade school but never knew it at the time): though she has resided across the pond in an older and more civilized part of the world for the past few years, she has managed to make her presence felt through the occasional trip stateside (e.g., weddings!), Skype chats, hangout messages, and even in lending her finely-tuned editing skills to my academic publishing endeavors. She has been one of my best friends for over a decade, and I cannot thank her enough for all her love and support.

I also want to thank faculty who have mentored me in some way beyond my immediate research: Jim Faeder, for his perpetual enthusiasm and readiness to run 15 miles with me at 5am on a departmental retreat; Dan Mosse, for his patience in allowing me to learn by making mistakes in supervising his undergraduates in their capstone projects, and for providing guidance throughout the job search; Rob Hall, for imparting invaluable machine learning and statistics knowledge and also pacing me on tempo runs; Bob Murphy, for getting me started in imaging and taking me on as a master’s student; Arvind Ramanathan, for introducing me to Chakra and bringing me to Oak Ridge for a summer; and to Merrick Furst, for his poignant advice, wisdom, and insights at and since leaving Georgia Tech.

I cannot thank my best man Dan enough for everything he has done over the years we have known each other, and especially over these years I have been in graduate school. He’s provided healthy perspective, come and visit on several occasions, run races with me, and left the most fantastic voicemails. With everything from bouncing algorithm ideas off each other to racing strategies, he has been a constant source of inspiration, calm, and humor. Even though he still (incorrectly) insists his Towers of Hanoi solver was more efficient than mine, I cherish the friendship we share.

To my family: words cannot express how important their constant love and support has been, both over the years of graduate school and for as long as I can remember. My two sisters, while distinct and incredible individuals who are extraordinarily talented in their own rights, are the best versions of myself and the people I strive to emulate (though I would

never admit this to them, of course). My parents are among the most open-minded, loving, and supportive people I have ever known. Among the myriad lessons I have learned from them, the first that comes to mind is how they approach everything and everyone with love first and above all else. I do, however, ultimately blame my dad for being an impeccable role model for becoming a career academic, but at least I will be working a few miles away and can come into his office and complain about it in person. And discuss the details of our latest joint academic project.

And finally, I want to say thank you to my better half, Cathryn. Over the last four years of graduate school, she went from girlfriend to fiancée to wife (and always when I had just gotten used to the previous title change). But even more than that, she has been my life partner. She kept me calm and focused when the stress of graduate school made even the shortest training run unbearable and frustratingly difficult and it felt like the sky was crumbling. She has sacrificed much on my behalf, particularly these last few months as my dissertation has taken front and center, and for that—and so much over the past 8 years—I am eternally grateful. I cannot express enough thanks to convey my gratitude for her edits of my various publications and this very thesis. Though I am sorry to admit I can never remember when I am supposed to use “which” versus “that,” but I suppose that is yet one more dimension of our partnership. Thank you, Cathryn, for your encouragement, your omnipresent love and support, and your unwavering confidence in me. Thank you for being *you*.

DEDICATION

To my wife, my family, and my friends, without whose love and support I would not be here.

1.0 INTRODUCTION

If you think it's simple, then you have misunderstood the problem.

Bjarne Stroustrup

1.1 PETABYTE-SCALE BIOMEDICINE

There is an ever-expanding body of biological data, growing in size and complexity, outstripping the capabilities of standard database tools or traditional analysis techniques. Such examples include long time-scale molecular dynamics (MD) simulations [1], gene regulatory networks [2], biomedical image analysis [3], and specifically video libraries of ciliary motion [4], and odor percept association [5]. It is often raw, or unstructured, and some is already in the public domain¹². Large-scale acquisition and curation of unstructured biological data have already yielded results in the form of lower costs for genome sequencing and greater coverage in databases such as GenBank, and is viewed as the future of biocuration [6, 7]. The “big data” approach and its associated paradigms have the potential to uncover solutions to problems otherwise intractable with more traditional investigative techniques. NIH Director Francis Collins recently cast the analysis of large-scale biomedical data as the “bottleneck” to new discoveries³.

What is responsible for the recent “data deluge?” In a word, *cost*. Prices for commodity hardware have fallen precipitously over the last few decades, and simultaneously the amount of computing power has risen dramatically. This is Moore’s Law [8] in practice: where this

¹<http://www.cellimagelibrary.org/>

²<http://databrary.org>

³<http://videocast.nih.gov/launch.asp?17711>

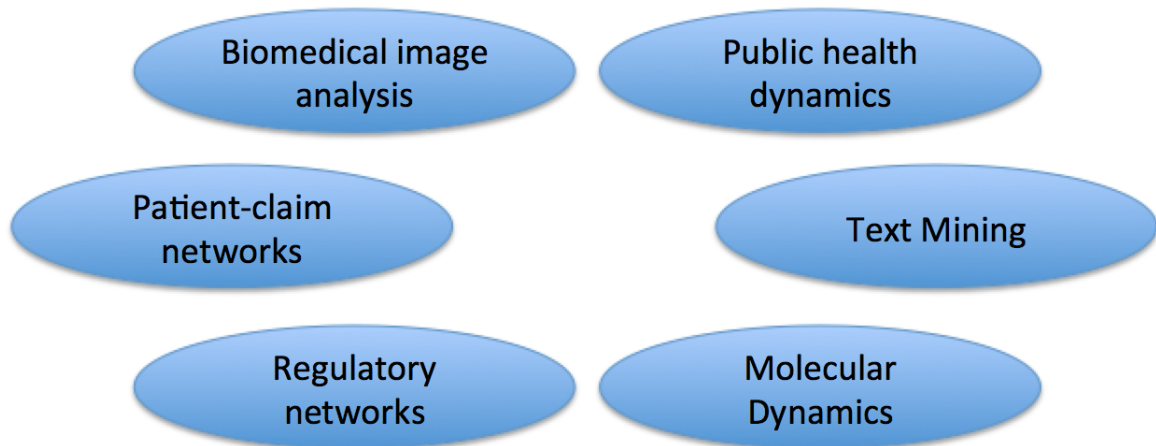


Figure 1: **Era of “Big Data.”** Biomedicine is but one of many areas of scientific and corporate research in which new, scalable data analysis techniques are required to make sense of increasingly large volumes of information.

theory relates to the narrow case of the number of transistors in dense integrated circuits doubling every two years, Moore’s Law echoes across all areas of technology. As storage space and processing power have increased while simultaneously dropping in cost, the bottleneck in data analysis has shifted from curation and identification to downstream analysis [9]. It is now cheaper and more cost-effective than ever to place sensors recording every possible fluctuation in the data, gathering and storing petabytes of raw, unstructured information, and to then perform comprehensive analysis after all data collection is complete.

However, there is an idiom in computer science: most problems can be solved by adding another layer of abstraction, which tends to introduce a new problem. While we now capture more information than ever in unprecedented resolution, the quantity of data has far outstripped the abilities of our traditional analysis toolkits to make sense of it. Instead, we have been driven to devise new, more powerful techniques that discard the traditional notion of a physical computer and instead rely on clusters of machines acting in a cohesive unit. This addresses the immediate problem of more data than one machine can handle, but

simply throwing more hardware at the problem is not necessarily the final panacea; we have to gain deeper insights into the problems we are attempting to solve, and the algorithms we are applying to solve them, in order to find the optimal solution.

In this thesis, we pursue two seemingly-unrelated projects in biomedicine and computer vision. In our first project, we examine the motion of cilia, microscopic hairlike structures that line most of the internal organs in humans. We use a series of techniques to quantitatively decompose the observed motion, and in the process hypothesize that there exist more than two discrete types of motion (healthy and abnormal). But to uncover these latent motion patterns and develop objective quantitative concepts for these motion phenotypes, we represent ciliary motion as a *graph*: an interconnected series of nodes, each representing distinct samples of cilia, connected to other samples by edges weighted according to how similar the motions between them are (Fig. 2). This graph structure appears in our second project as well, in which we investigate computational approaches to defining and predicting perceptual dimensions in olfaction. This problem is particularly interesting, given that olfaction, unlike other sensory modalities, does not have a known well-defined mapping from its stimulus space to the perceptual space. We use terms such as “fruity,” “nutty,” or “gasoline” not in a rigorous context, but to vaguely group odorants into categories with poorly-defined boundaries. However, by defining a metric according to pairwise similarity information, we can generalize the quantitative similarity definitions to unobserved odorants. Before this can happen, however, we have to organize the odorants into a graph structure, with edges weighted according to their similarity. Only then can we use the learned metric to propagate perceptual information across the network.

In both projects, the sheer quantity of data required novel and scalable analysis methods to avoid resorting to significant downsampling. This motivated the design for the third contribution of this thesis: a **distributed hierarchical eigensolver**, or *Draenor*. This framework derives its namesake from the Orcish homeland in *WarCraft*, a landmass that shattered into pieces. The eigensolver invokes the metaphor: not only does it make use of hierarchical, or *multigrid*, techniques for breaking up the complex initial problem into smaller, simpler version of the original problem, but *Draenor* is a *distributed* framework, capable of operating in parallel across many distinct physical and logical instantiations. As we will discuss

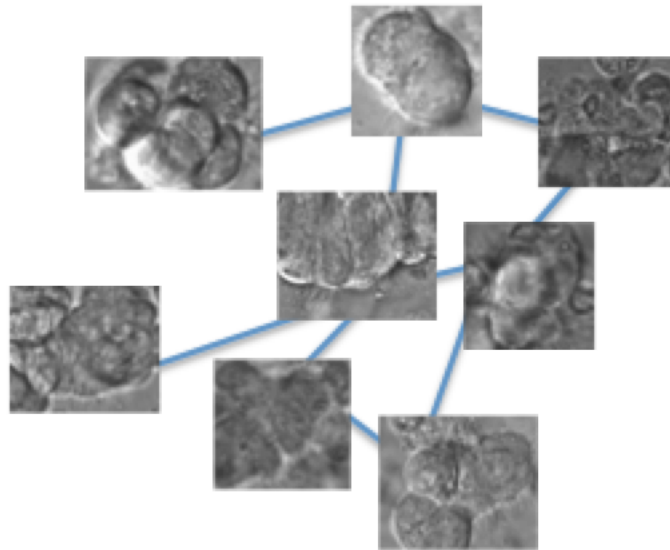


Figure 2: **Graphs.** Network structures are ubiquitous and field-agnostic, appearing in social media, disease outbreaks, and automated object recognition. Shown here is a toy example of a graph we create when searching for novel ciliary motion phenotypes.

in greater detail in the subsequent chapters, graphs naturally lend themselves to analysis by computing the eigenvalues and eigenvectors. This is how Google’s famous PageRank algorithm [10] functions. Websites are interconnected through a series of hyperlinks, which results in a graph structure. By estimating the first eigenvector of this graph, PageRank can produce a robust estimate of the most relevant websites given a particular search.

This addresses the need in biological research for a method to analyze the corpus of data in a way that accounts for its increasing size. We have discussed existing parallel implementations, but these require additional and expensive hardware to scale to larger datasets; supercomputers are an example of highly parallel but expensive resources that are difficult for many researchers to access. By contrast, assembling a cluster of commodity hardware for distributed computation is not only vastly cheaper but can rapidly approach supercomputing performance [11]. To our knowledge, no distributed version of a multigrid hierarchical eigensolver exists.

1.2 CONTRIBUTIONS

This thesis contains the following contributions:

- Introducing a high-throughput computational method for classifying ciliary motion as normal or abnormal with high accuracy. This method is novel in that it is the first quantitative, objective framework for ciliary motion phenotype recognition.
- Revealing novel insights into the low-dimensional manifold of ciliary motion through large-scale spectral analytics. We reveal through unsupervised machine learning techniques and intelligent region selection that multiple motion phenotypes exist beyond strictly normal and abnormal.
- Learning a pairwise odorant comparison metric that achieves unprecedented accuracy in recognizing and predicting perceptual categories. Our similarity metric substantially improves classification of odorants into perceptual categories by reorganizing their relative orientations in physiochemical space.

- Showing the structure of the physiochemical space, and in particular which properties are most informative both in general and for recognizing specific odor percepts. The metric reveals the interplay and complex dependencies between physiochemical properties in determining odor percept.
- Developing *Draenor*, a distributed hierarchical eigensolver, for the purpose of efficiently solving linear equations that arise in large-scale datasets, such as our ciliary motion phenotypes or olfaction recognition case studies. Draenor combines the theoretical $O(n)$ runtime of traditional hierarchical solvers with the scalability of distributed computing. We demonstrate the capabilities of Draenor and compare its speed and accuracy to other distributed eigensolvers.
- Providing these frameworks, and Draenor in particular, as actively maintained open source software for use on commodity hardware.

1.3 THESIS OUTLINE

Before we delve into the specifics of Draenor, we first investigate the history and background of solving linear systems, and in particular, finding eigenvalues and eigenvectors. In Chapter 2, we examine the theoretical underpinnings of linear systems and the foundations of solving for the eigenvalues and eigenvectors. We investigate some of the seminal advances in building efficient solvers and insights into the subspaces spanned by the eigenspectrum. Additionally, we link the logical structure of the network into the technical machinery of finding eigenvalues and eigenvectors, and why this is of particular interest at extremely large scales. Finally, we conclude the background section with a discussion on the benefits and disadvantages of the current state-of-the-art hierarchical, or multigrid, eigensolvers, and why these powerful methods are necessary for efficient analysis techniques in the area of biomedicine.

In Chapter 3, we introduce our first case study in large-scale biomedical data analysis. We describe efforts in biomedical research and in clinical diagnostics to identify ciliopathies, a class of diseases marked by abnormal or otherwise impaired ciliary motion. Cilia are microscopic hairlike structures that line most internal organs including the throat, lungs,

kidneys, and brain. They beat in synchronous waves to clear particulate matter. Their beat pattern phenotypes can be indicative of various ciliopathies, and providing an objective measure for quantifying the observed motion phenotypes is clinically compelling. In this chapter, we discuss our innovative approach to differentiating between normal and abnormal ciliary motion. We further expand upon this notion by pursuing an unsupervised method to discover novel ciliary motion patterns. We conclude by motivating large-scale analysis frameworks, particularly for solving linear systems, as the means for achieving additional breakthroughs.

In Chapter 4, we introduce our second case study: an investigation into the modes of human olfaction. In contrast to other sensory modalities, the basic perceptual dimensions remain unclear. We describe numerous approaches, both psychophysical and computational, that have been proposed to elucidate the primary percepts of olfaction. In particular, we examine our hypothesis that the odor space is occupied by a discrete number of percepts that are intrinsically clustered. Using this information, we devise a pairwise metric based on the physiochemical properties of odorant compounds which incorporates odorant similarity. This novel method provides the quantitative foundation for large-scale computational studies in olfaction; this metric could be utilized within a large-scale semi-supervised framework to provide *in silico* percept predictions for the entirety of the PubChem corpus.

In Chapter 5, we discuss the central contribution of this thesis: *Draenor*, the **DistR**ibuted **hierA**rchical **EigeNsOlveR**. While other highly efficient parallel and distributed eigensolvers exist, to our knowledge this constitutes the first distributed hierarchical multigrid eigensolver of its kind. We discuss its theoretical basis in the context of its technical implementations. We investigate the practical trade-offs with this method, and compare its performance and accuracy against other serial and distributed eigensolvers. We conclude with a discussion on the technical limitations of this method, and its future directions.

In Chapter 6, we conclude with a summary of the novel computational methods and subsequent discoveries enumerated here, the role of *Draenor* in enabling these types of large-scale analytics, and informed speculation regarding the necessity of such methods to the future of biomedical discoveries.

1.4 PUBLICATIONS

A significant portion of the materials of this thesis has either been published in conference proceedings or has been submitted to conferences or journals. Listed below are the relevant publications and the chapters with which they are primarily associated.

Chapter 3

- *Novel use of differential image velocity invariants to categorize ciliary motion defects.* **Shannon Quinn**, Richard Francis, Cecilia Lo, and Chakra Chennubhotla. Published in the proceedings of the Biomedical Sciences and Engineering Conference (BSEC).
- *Dynamic texture analysis for automated identification of abnormal respiratory ciliary motion.* **Shannon Quinn**, Maliha Zahid, John Durkin, Richard Francis, Cecilia Lo, and Chakra Chennubhotla. Submitted to the journal Science Translational Medicine.

Chapter 4

- *Designing a physiochemical descriptor based metric for categorizing odors.* **Shannon Quinn**, Arvind Ramanathan, Jason Castro, and Chakra Chennubhotla. Submitted in parallel to the journal Scientific Reports and the proceedings of the Research in Computational Molecular Biology (RECOMB) conference.

Chapter 5

- *ORBiT: Oak Ridge Bio-surveillance Toolkit for Public Health.* Arvind Ramanathan, L Pullum, T Hobson, **Shannon Quinn**, Chakra Chennubhotla, and S Valkova. Published in the journal BMC Bioinformatics.
- *Discovery of Disease Co-occurrence Patterns from Electronic Healthcare Reimbursement Claims Data.* Arvind Ramanathan, L Pullum, T Hobson, **Shannon Quinn**, Chakra Chennubhotla, and S Valkova. Published in the proceedings for the Big Data Analytic Technology for Bioinformatics and Health Informatics workshop (KDDBHI) at the KDD conference.

- *Oak Ridge Bio-Surveillance Toolkit (ORBiT): Integrating Big-Data Analytics with Visual Analysis for Public Health Dynamics.* Arvind Ramanathan, L Pullum, C Steed, Tara Parker, **Shannon Quinn**, and Chakra Chennubhotla. Published in the proceedings for the IEEE VIS conference.
- *Statistical inference for big-data problems in molecular biophysics.* Arvind Ramanathan, Andrej Savol, Virginia Burger, **Shannon Quinn**, PK Agarwal, and Chakra Chennubhotla. Published in the proceedings for the Parallel and Large-Scale Machine Learning workshop at the NIPS conference.

1.5 MATHEMATICAL NOTATION

Throughout this thesis, we will consistently use notation that should be familiar to discrete mathematicians, computer scientists, and statisticians. We denote a column vector as \vec{x} . The i^{th} element of this vector is denoted by x_i . Similarly, scalar values x are shown without an arrow. Functions (discrete or continuous) \mathbf{f} are denoted with boldface. Matrices M are denoted with capital letters. Unless otherwise stated, the scalar n will be used to refer to the number of data points, and m the number of dimensions per datum. Thus, $M \in R^{n \times m}$ denotes the matrix M which defines a mapping of n points in m -dimensional space. Individual elements of a matrix M are indicated with subscripts M_{ij} , denoting the element in the i^{th} row and the j^{th} column. Calligraphic capital letters \mathcal{S} often denote sets; these can be tensors (sets of matrices) or other arbitrary data. These sets use brackets to denote the elements of the set, such as $\mathcal{S} = \{S_1, S_2, \dots, S_n\}$, where S_i is the i^{th} element of the set \mathcal{S} . In some cases, calligraphic letters will be used to denote matrices that are composed of one or more submatrices. These are not tensors, but rather larger matrix conglomerates of smaller matrices of interest. For example, suppose we define the matrix \mathcal{P} as

$$\mathcal{P} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}$$

In this case, it is implied that submatrices P_{11} , P_{12} , P_{21} , and P_{22} have particular properties

that are relevant to the current topic. These properties will be discussed in depth wherever this notation appears.

Greek letters are used to denote specific parameters of interest. For example, σ is often used as the measure of standard deviation for a parametric distribution. Likewise, λ is commonly used to denote a specific eigenvalue. Explanations will be provided within the specific contexts of their uses throughout this thesis.

When discussing algorithmic complexity, we use the $O(\cdot)$ notation that is traditionally used in theoretical computer science. Reporting the runtime of a procedure as $O(n)$ on a dataset with n elements explicitly states the algorithm complexity is proportionally linear; that is, in the limit as the quantity of data approaches infinity, the runtime converges to linear in the number of elements.

2.0 SPECTRAL GRAPH ANALYTICS

2.1 INTRODUCTION

One of the most fundamental queries when studying graphs is to examine the diffusion of information across them, spatially and temporally. How many connected components are there, and what are the constituent members of each component? Where are the bottlenecks in information diffusion across the network? Are there any hubs in the graph? These are all common questions to ask when analyzing graph structures. The heat equations governing distribution of temperature within an insulated container [12] are one example. There are many other such applications—including clustering, stability of dynamic systems, and Markov chain models [13]—that are relevant to the study of biological systems. The spectral properties of the graph are extremely useful for gaining insight into these behaviors.

At the core of spectral analysis is the process of finding some or all of the eigenvectors and eigenvalues of a matrix that algebraically represents the graph of interest. The fundamental equation defining this relationship is given as

$$L\vec{u} = \lambda\vec{u} \tag{2.1}$$

where \vec{u} and λ are an eigenvector and corresponding eigenvalue respectively of the matrix L . How we define a graph algebraically, how we explicitly compute the eigenvectors and eigenvalues, and how we can use the eigenspectrum of the graph to gain some intuition for its structure and information diffusion, are the topics of the following sections.

2.2 GRAPH STRUCTURES AND PROPERTIES

In this section, we provide an overview of the mathematical formulation of graphs and the methods used to analyze their structure. We consider only the case of *undirected* graphs, whereby each edge connecting two nodes is bidirectional; that is, one can walk from node i to node j and vice versa. Directed graphs are of considerable research interest; Google’s PageRank algorithm is designed specifically for directed graphs, in which there can exist an edge connecting node i to node j , but once at node j one may not necessarily be able to move back to node i (Fig. 3). However, for the purposes of our case studies, we examine only undirected graphs.

2.2.1 Graph affinities and neighborhoods

We start by defining a graph G in terms of its vertices V and the edges E that connect them: $G = (V, E)$. The vertices V correspond to individual data points; these can be pixels in an image or users in a social network. Edges connect vertices that have some degree of similarity; heavier weights on the edges indicate a higher degree of similarity. In terms of a random walk, the heavier edges correspond to hops of higher probability. Formally, we define affinities between nodes \vec{x}_i and \vec{x}_j as some measure of similarity. A common affinity measure is the radial-basis function (RBF):

$$a_{ij} = \exp \left\{ -\gamma (\vec{x}_i - \vec{x}_j)^2 \right\},$$

where a_{ij} is the affinity, or weight, between nodes \vec{x}_i and \vec{x}_j in the graph, and γ is a scaling parameter that dictates how quickly the affinity function falls off as distance between \vec{x}_i and \vec{x}_j increases. All weights a_{ij} are nonnegative; a weight of 0 indicates the absence of an edge between the specified nodes. In undirected graphs, $a_{ij} = a_{ji}$, resulting in a symmetric affinity matrix A . There are many strategies for constructing A , three of which we discuss here.

- **Fully connected:** In this case, we compute all n^2 pairwise affinities, resulting in a dense (fully-connected) graph, where $a_{ij} > 0$ for all pairs of nodes \vec{x}_i and \vec{x}_j .

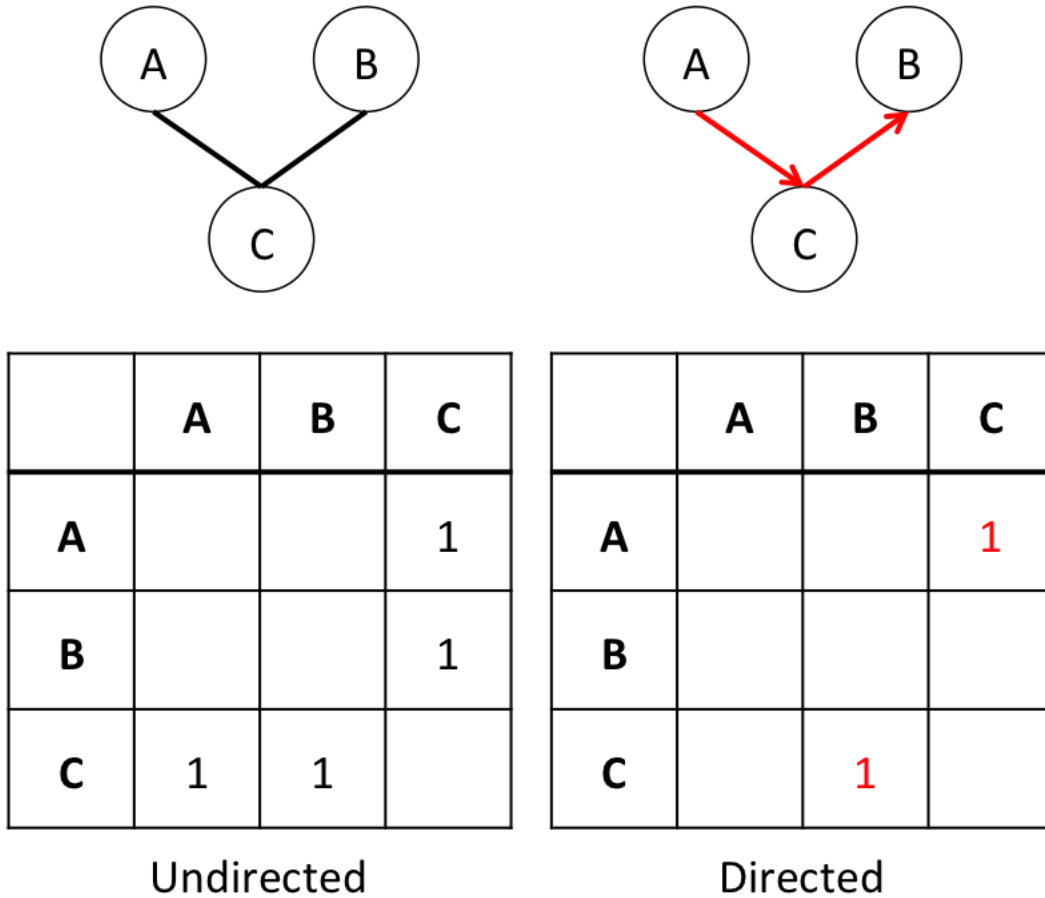


Figure 3: **Undirected vs Directed graphs.** Two examples of a three-node graph, one undirected (left) and the other directed (right). The technical differences between the two are particularly apparent in their connectivity matrices (bottom row), in which the presence of a “1” indicates the existence of an edge between the two nodes. Of interest is the property of symmetry that is inherent to connectivity matrices of undirected graphs.

- **ϵ -neighborhoods:** For each node \vec{x}_i , we compute pairwise affinities for all other nodes \vec{x}_j that are within a distance ϵ from \vec{x}_i . If the pairwise distance is greater than ϵ , the affinity is set to 0. This strategy is employed in Fig. 4.
- **k -nearest neighbors:** Each node \vec{x}_i is connected to its nearest k neighboring nodes. There are circumstances under which node \vec{x}_i will have node \vec{x}_j in its k -nearest neighbors, but the reverse will not be true. We can resolve this one of two ways: an edge can be created between two nodes if *either* contains the other in its k -nearest neighbors, or an edge can be created only if *both* nodes are in each other's k -nearest neighbors. The latter strategy is known as *mutual k -nearest neighbors*.

This touches on the notion of *sparsity*. This is a particularly interesting property, especially as graphs become arbitrarily large; it becomes useful to have only a very small number of edges in the graph relative to the number of vertices. Sparse graphs, which are opposed in concept to dense graphs, can have their sparsity exploited to improve the runtime and accuracy of certain graph analysis techniques.

Image data are a special case. Each pixel is a node in the graph, and we often connect its four or eight neighboring pixels (edges and corners will have only five and three neighboring pixels, respectively).

We define the degree d_i of node $\vec{x}_i \in V$ as

$$d_i = \sum_{j=1}^n a_{ij}$$

From this definition, we form the diagonal *degree matrix* $D = \text{diag} \{d_1, d_2, \dots, d_n\}$. This matrix is important for constructing the *graph Laplacian* L , a critical component of spectral graph analytics [14]. There are many different formulations of graph Laplacians [15], far too many to discuss here in depth. For our purposes, we are concerned with only a form generally referred to as the *normalized graph Laplacian*:

$$L = D^{-1/2}AD^{-1/2}, \tag{2.2}$$

where D is the degree matrix, A is the affinity matrix, and L is the normalized graph Laplacian. This form of the graph Laplacian has several useful properties. As L is symmetric,

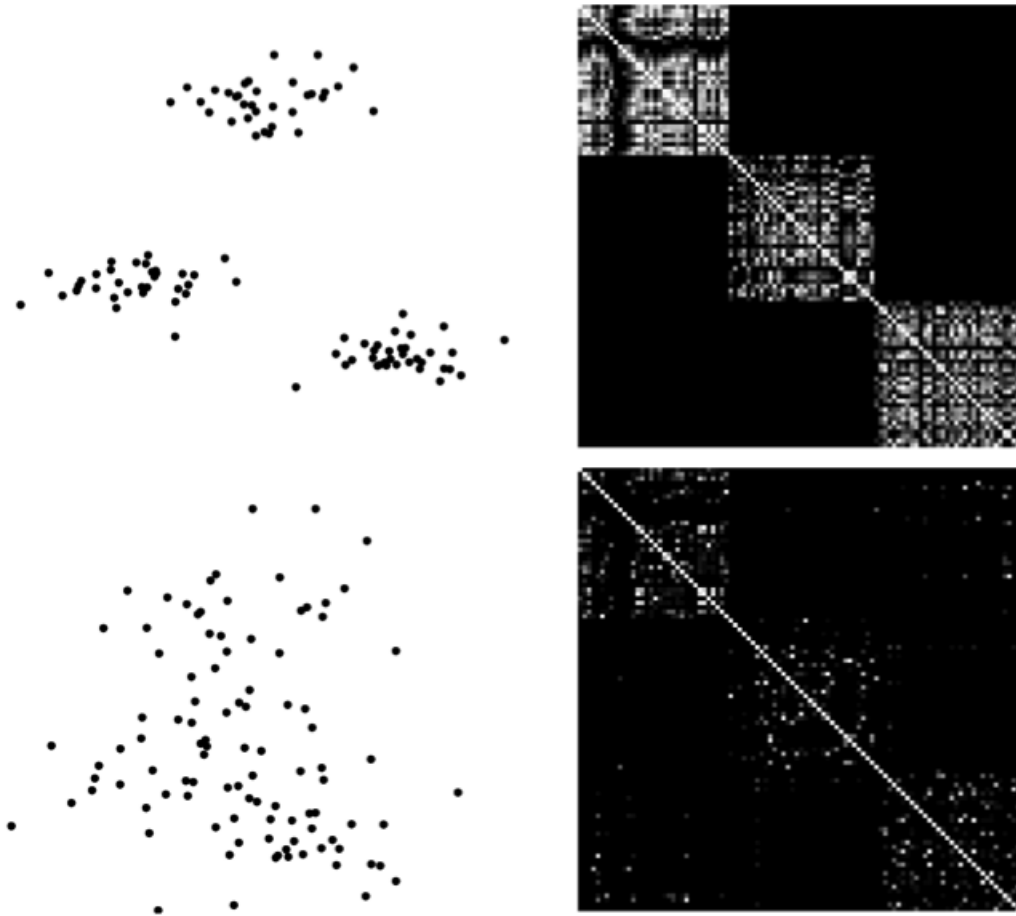


Figure 4: **Affinity matrices for corresponding 2D data.** Tightly coupled clusters of points (top row) result in block-diagonal affinity matrices, relative to more loosely coupled data (bottom row).

its eigenvalues are real and nonnegative. Because of the normalized form, the eigenvalues $\lambda_1, \dots, \lambda_n$ satisfy $0 \leq \lambda_1, \dots, \lambda_n \leq 1$. We will use this form of the normalized graph Laplacian throughout this document.

2.2.2 Random walks

An important analogy we use when discussing information diffusion over graphs and graph segmentation is that of a *random walk*. We gain a great deal of intuition for the underlying dynamics of a graph by considering it as a series of probability distributions. Imagine a particle at some vertex v . What are the other possible vertices to which the particle can move, and what are the relative probabilities of moving to each?

Formally, we can convert the affinity matrix A into a stochastic transition matrix M using the following relationship with the degree matrix D [16]:

$$M = D^{-1}A$$

The rows of the resulting matrix M sum to 1. Therefore, the element M_{ij} represents the probability of the particle moving from vertex v_i to v_j in one step, given the particle is currently at vertex v_i . This defines a Markov chain of probabilities between vertices, which are proportional to the pairwise edge affinities A_{ij} .

To explore this concept of a Markov transition matrix further, we illustrate with an example of a random walk. Suppose the initial probability of a particle being at vertex v_i is p_i^0 , for $i = 1, \dots, n$. At the next time step, the probability of the particle moving to vertex v_j from its starting position at vertex v_i is $M_{ij}p_i^0$. Using matrix notation, we can define the probability of a particle traveling to any one of the vertices $\vec{v} = (v_1, v_2, \dots, v_n)$ from its starting vertex as $\vec{p}^1 = M\vec{p}^0$. We can iterate this process over β steps, giving the relation

$$\vec{p}^\beta = M^\beta \vec{p}^0.$$

We can observe this process for different values of β in the two-dimensional example given in Fig. 5. Each pixel in the two-dimensional plane is a vertex, and each vertex is connected to its four neighbors above, below, and to either side of the pixel. For very large

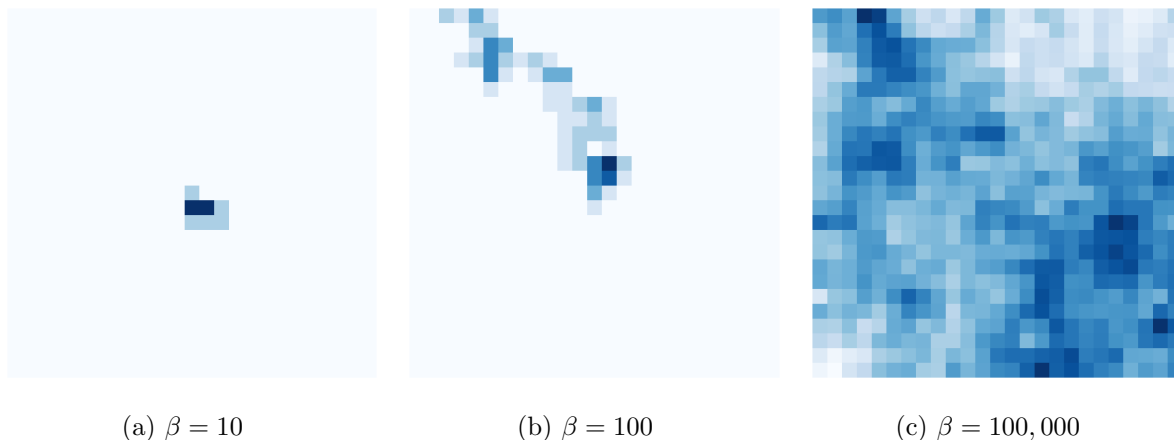


Figure 5: **2D random walks.** Starting in the middle of each panel, we simulated the random walk of a particle in two dimensions over β time steps. The intensity at each pixel indicates the number of times the particle visited that position.

numbers of steps, we see the particle visits the entire graph; however, for smaller numbers, it visits only the nodes for which it has the highest probability of visiting: in this example, the immediate neighbors of a given pixel. This concept of a random walk across the underlying graph to identify bottlenecks in transition probability is central to the application of spectral clustering.

2.3 SPECTRAL CLUSTERING

Spectral methods for clustering rely on finding the eigenvalues and eigenvectors of the underlying graph Laplacian. The normalized graph Laplacian L has several useful properties that make it ideal for spectral analysis. In particular, it is positive semi-definite ($L \succeq 0$), and has n nonnegative real-valued eigenvalues $0 \leq \lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_n$. Relative to other matrices, such as the Markov transition matrix M , it is more stable to small perturbations in the underlying graph, and is computationally more tractable to analyze. Partitioning the

graph k -ways can be done using the eigenvectors and eigenvalues of L [17]. Relating back to the concept of a random walk, the eigenvectors of the Laplacian can identify the bottlenecks in transition probability between vertices, which is crucial to segmenting graphs.

Suppose we have n data points, $\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n$, have computed our pairwise affinity matrix A , and have determined *a priori* the number of clusters k we are interested in computing. Traditional spectral clustering proceeds according to the following steps:

1. Construct the normalized graph Laplacian L according to Eq. 2.2.
2. Build a matrix U of eigenvectors $[\vec{u}_1, \vec{u}_2, \dots, \vec{u}_k]$ associated with the largest k eigenvalues of L .
3. Create a matrix V by normalizing each row of U to be unit length:

$$V_i = \frac{U_i}{\|U_i\|}$$

4. Consider each row of V as a point in \mathcal{R}^k , and cluster them using K-means or some other method.
5. Assign node \vec{x}_i to cluster k if the corresponding row V_i was assigned to cluster k .

Fig. 6 shows the results of spectral clustering as compared to traditional K-means in two different datasets. For data that is largely isotropic (top row), both algorithms perform similarly, correctly identifying the distinct clusterings. However, for data that is anisotropic, such as the concentric circles in the bottom row, K-means cannot derive the desired clustering. Spectral clustering, however, embeds these points in a low-dimensional space where they are easily separable. Fig. 7 provides the intuition behind this mechanism: the data in both the top and bottom rows result in distinct eigenvectors that behave uniquely for each cluster, allowing them to be easily identified. The middle case, where the data are loosely coupled, causes problems for both spectral clustering and traditional K-means.

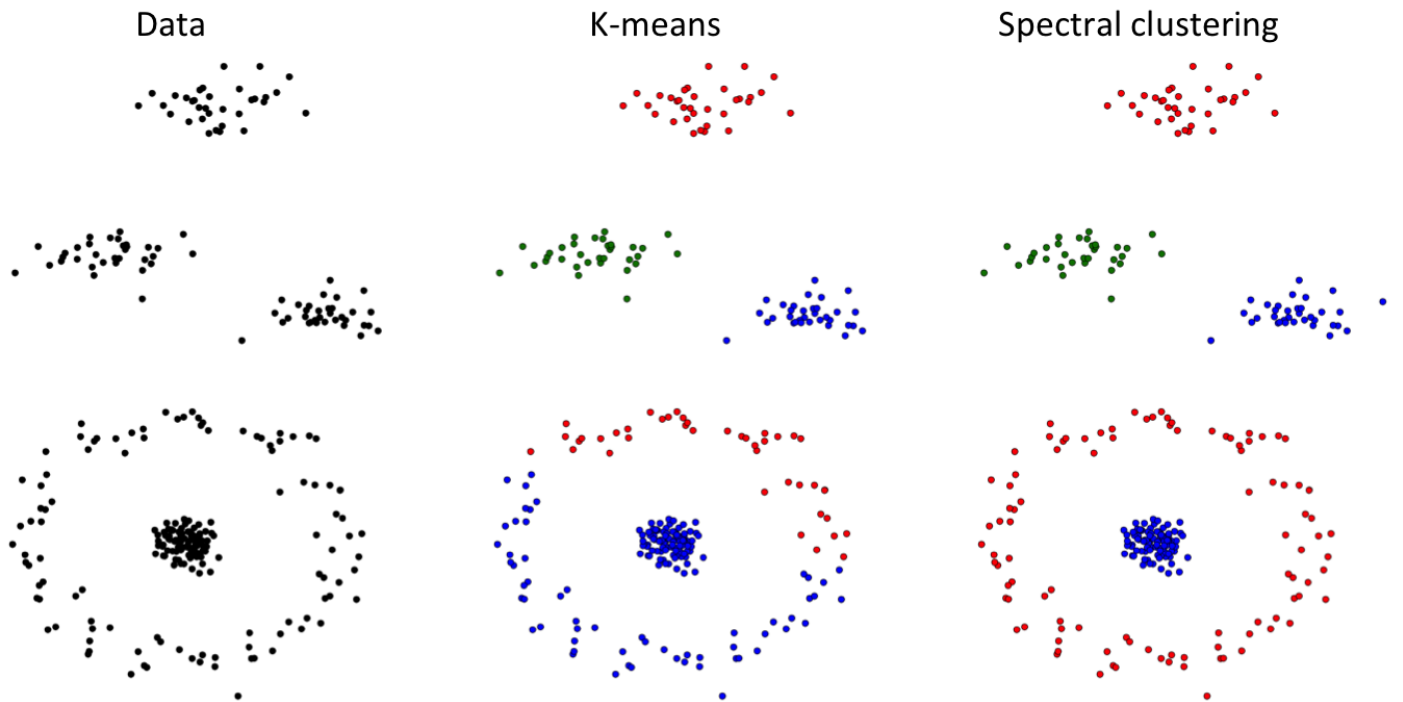


Figure 6: **K-means vs spectral clustering.** Results of clustering using K-means and spectral clustering on two different datasets.

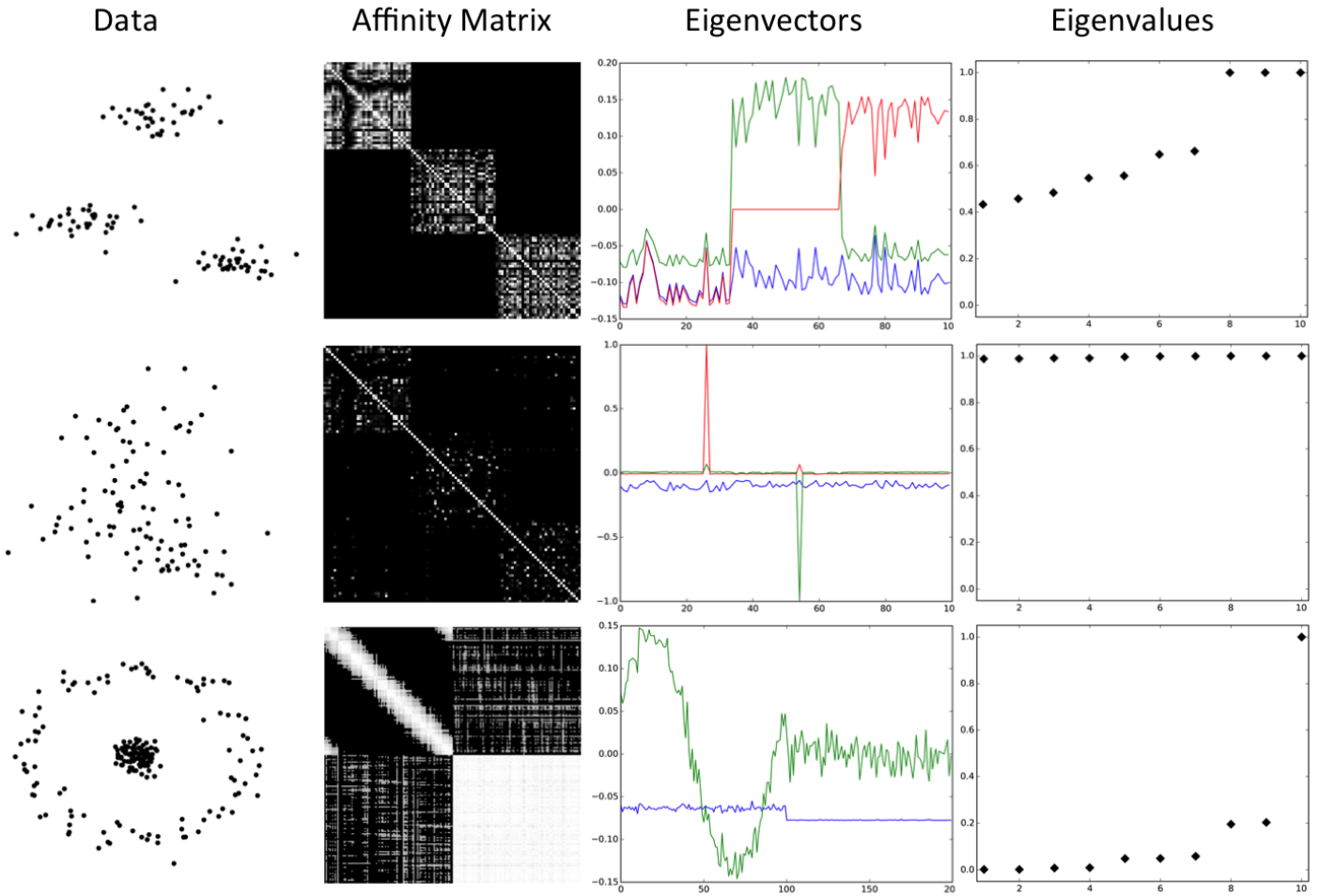


Figure 7: **Graph spectra.** For three different data distributions, we show the resulting affinity matrices, top k eigenvectors, and top 10 eigenvalues.

2.4 METHODS FOR FINDING EIGENVALUES AND EIGENVECTORS

Eigenvectors and eigenvalues arise from solving linear equations of the form $A\vec{x} = \vec{b}$, particularly when modeling dynamics. The prefix *eigen* is adopted from German, translating to “own” or “belonging to.” This is best exemplified using the expression $A\vec{x}$. For nearly all vectors \vec{x} , multiplication with A results in the vector changing its direction. However, in some special cases, \vec{x} are in the same direction as $A\vec{x}$. These special vectors are referred to as *eigenvectors* of A . Their magnitude may differ by a corresponding constant λ , therefore referred to as *eigenvalues* (Eq. 2.1).

There are many different methods for finding the eigenvectors and eigenvalues of matrices, typically optimized for different conditions. This is a broad and active area of research with a long history; we will only provide a brief overview to illustrate the breadth of available solvers, the benefits and drawbacks of each, and to provide a background for the category of solvers we are particularly interested in.

2.4.1 Direct solvers

Direct solvers, such as solving the accompanying characteristic polynomial of the graph Laplacian, can directly factor the eigenvalues [18]. The characteristic polynomial takes the form

$$\det(A - \lambda I) = 0.$$

As each eigenvalue λ is found, its corresponding eigenvector \vec{u} can be found by solving $(A - \lambda I)\vec{u} = 0$ for each λ . When A is $n \times n$, this equation has degree n .

Therein lies the core problem with the characteristic polynomial: as Arthur Cayley found in his seminal 1861 publication [19], polynomials higher than degree 5 cannot be directly solved, and only some quintic polynomials are solvable. Thus, other methods are required to directly solve this equation for when the polynomial has a higher degree. Gaussian elimination is one of the oldest direct methods, running in $O(n^3)$ operations [20]. However, this is only tractable for a small number of eigenvalues.

2.4.2 Iterative solvers

Iterative methods, such as Arnoldi [21] (for non-symmetric systems) and Lanczos [22] (for symmetric systems), repeatedly perform a matrix-vector multiplication (known as a *power iteration*) until convergence to an eigenvector and eigenvalue. The resulting Arnoldi or Lanczos vectors, respectively, form an orthonormal basis spanning the Krylov subspace. This subspace is effectively a collection of $n \times n$ matrices (such as A) that are multiplied by n -length vectors (such as \vec{u}), forming a quantitative basis in n -dimensional space for the results of the iterative methods for finding eigenvalues and eigenvectors. There are many methods that exploit the Krylov subspace for this purpose. They are advantageous in that they provide partial results after a relatively few number of iterations, whereas direct solvers must finish completely before providing any results. Some of these methods include conjugate gradient [23], GMRES [24], and MINRES [25], among many others [26,27]. While they operate much more efficiently than direct solvers, their running time can still be large and difficult to accurately estimate [22].

Most of these iterative methods rely on sparsity, the disparity between the number of edges compared to the number of vertices. Sparse graphs are interesting from a computational perspective, as their graph Laplacians, while still $n \times n$, are also sparse, implying most entries in the matrices are 0. In this document, we will focus exclusively on sparse graphs. Many eigensolvers are specifically tailored for sparse graphs, including direct solvers [28], as significant performance optimizations can be made under these circumstances.

2.4.3 Hierarchical and multigrid solvers

The class of sparse eigensolvers we focus on are *hierarchical* or *multigrid* [29–32]. The basic premise involves the iterative reduction of the initial graph to simpler versions. Successively simpler, or *coarser*, graphs are constructed, creating a graph hierarchy until some coarseness threshold is reached and the graph becomes simple enough to solve directly. This is known as *preconditioning*. A preconditioner is the transformation of a problem into a version more suitable for finding a numerical solution [33–35]. The preconditioner must be carefully chosen so as to ensure the resulting system is easier to solve than the original, and that the solution

to the original system can be easily derived from the solution to the preconditioned system. The resulting eigenvectors for the coarse graph are then combined with the interpolation operators defined at each level of the hierarchy, interpolating the approximated eigenvectors until the original level is reached.

There are two main families of hierarchical methods: geometric multigrid (GMG) and algebraic multigrid (AMG) [36,37]. GMGs have prescribed grid hierarchies; for example, the first hierarchy may consist of selecting every other pixel from an image. For smooth images, or *homogeneous* systems, this will yield good results. However, for *inhomogeneous* systems with discontinuities and large, sudden derivative fluctuations, such a prescribed grid will not capture the underlying structure. AMGs, by contrast, pick interpolation operators automatically based on the current topology of the Laplacian at each level, conferring enormous empirical performance benefits over GMGs in inhomogeneous systems [34,38,39].

A core objective with hierarchical methods is to reduce the condition number of the system. The condition number κ is defined as the ratio of the system’s largest and smallest eigenvalues:

$$\kappa(L) = \frac{\lambda_{\max}}{\lambda_{\min}}.$$

2.4.4 Geometric multigrid (GMG) eigensolvers

When this ratio is large, as in most inhomogeneous systems, this typically implies a wide distribution of both strong and weak edges in the graph, introducing the spatial irregularities that are difficult for GMGs to solve.

Still, we found GMGs to be useful in establishing baseline performance, and particularly for performing image analysis. Geometric methods are known to give linear $O(n)$ runtime for spatially homogenous systems. Graph matrices associated with images have very regular structures; while the distribution of edge weights can vary according to the affinity function used, most associated graph matrices from images share the same highly regular pattern of edges between neighboring pixels (Fig. 9). Another classic example application for GMG solvers is the regular homogenous Poisson equations [29].

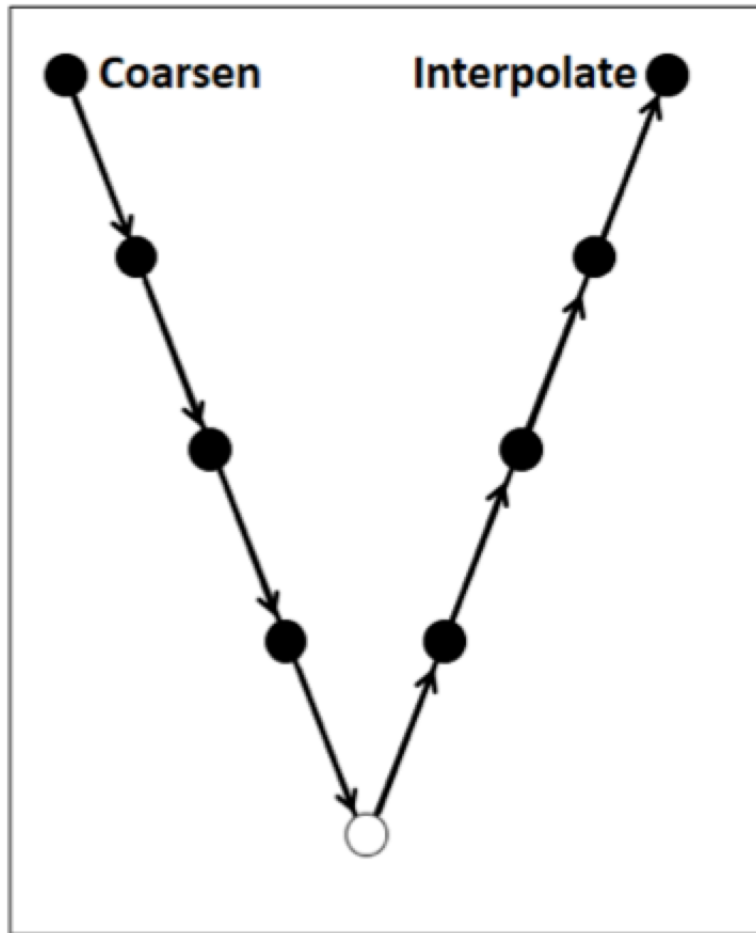


Figure 8: **Hierarchical schematic.** This schematic depicts the broad conceptual process of hierarchical eigensolvers. The original graph is iteratively “coarsened” in such a way that it maintains a similar structure to the original while simultaneously simplifying the problem to be solved. During this process, interpolation operators are generated at each step of the hierarchy which defines the coarsening procedure. This process is repeated until a threshold is reached, at which point it is solved directly. The interpolations operators at each level of the hierarchy are then used to interpolate the results back up to the finest level.

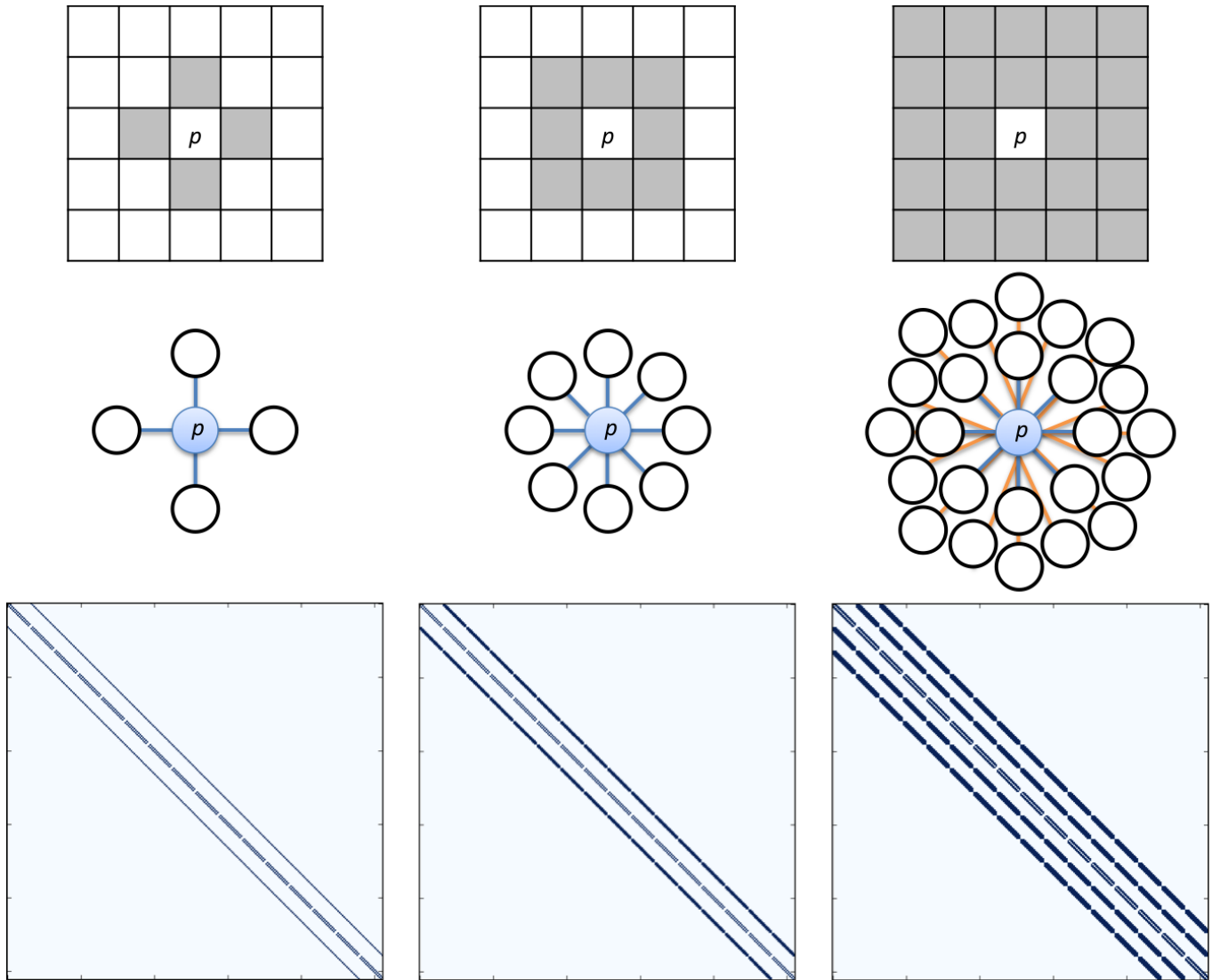


Figure 9: **Pixel neighborhoods and graph structure.** The graphs underlying image data have very regular structure which make them ideal candidates for analysis by geometric multigrid. The top row shows three common strategies for constructing a graph from image data: each pixel (identified p) is connected to its 4, 8, and 24 nearest neighbors (identified as shaded boxes). The resulting subgraph around this pixel is illustrated in the middle row. The bottom row depicts the structure of the final resulting graph matrix given the connectivity strategy. These matrices are extremely sparse, with only 1.47%, 2.84%, and 7.97% respectively of the total number of possible edges between vertices.

Algorithm 1 dncuts

```
1: Input:  $A, D, K$ 
2:  $A_0 \leftarrow A$ 
3: for  $d = [1 : D]$  do
4:    $i_d \leftarrow \text{pixel\_decimate}(A_{d-1})$ 
5:    $B_d \leftarrow A_{d-1}[:, i_d]$ 
6:    $C_d \leftarrow \text{diag}(B_d \vec{1})^{-1} B_d$ 
7:    $A_d \leftarrow C_d^T B_d$ 
8: end for
9:  $X_D \leftarrow \text{eig}(A_D, K)$ 
10: for  $d = [D : -1 : 1]$  do
11:    $X_{d-1} \leftarrow C_d X_d$ 
12: end for
13: return  $\text{whiten}(X_0)$ 
```

We focus on the algorithm as proposed in [40], reproduced in Algorithm 1. Given a graph matrix A (an affinity, similarity, or connectivity matrix representing the underlying graph of the image), a positive integer number of iterations D , and a desired number of eigenvectors K , this method performs a repeated *decimation and squaring* operation, whereby every alternate pixel in the original image is removed entirely and its corresponding entries in the graph matrix are deleted. This is the *decimation* operation. Where this operation by itself would significantly perturb the eigenvectors of the underlying graph, the second operation, *squaring*, involves multiplying the graph matrix by itself as a way of propagating information along the edges of the graph, thereby preserving some of the information lost in the decimation step.

We see the decimation in Step 5 of Algorithm 1, which is renormalized in Step 6 and squared in Step 7 to compute the new, smaller graph matrix for the next iteration. This process repeats D times, at which point the resulting graph matrix A_D is given to a direct eigensolver in Step 9, and the top K eigenvectors are computed and stored as a matrix X_D . Using the interpolation operators C_D, C_{D-1}, \dots, C_1 defined in the first loop, the eigenvectors

are interpolated back up the hierarchy to provide a robust estimation of the top K eigenvectors of the original graph matrix A . Some whitening is performed on the final eigenvectors in Step 13 to reorthormalize them, as interpolation can skew the vectors.

However, GMGs perform poorly when strong spatial inhomogeneities are introduced. For most problems, this alone would remove GMGs from consideration. Consider the examples of ciliary motion analysis in Chapter 3 and olfactory perception recognition in Chapter 4: both problems live in high-dimensional spaces that are very likely nonlinear.

2.4.5 Algebraic or Combinatorial multigrid (AMG, CMG) eigensolvers

Algebraic multigrid methods have a distinct advantage of GMGs in that they iteratively reformulate the problem over a hierarchy of adaptively coarsened grids. These grids are linked throughout the hierarchy by a series of interpolation weights, capturing increasingly coarse connections in the original matrix. Whereas GMGs operate by homogenous coarsening of the original grid, AMGs adaptively select portions of the grid to coarsen as a reaction to the algebraic properties and distributions of the grid complexity. This formulation makes AMGs significantly more robust to spatial inhomogeneities, and much more useful for most practical linear systems.

We use the method proposed in [35], which aims to find a preconditioning matrix Q^{-1} such that the condition number of the preconditioned system, $\kappa(Q^{-1}L)$, is significantly lower than that of $\kappa(L)$. This is achieved through a hierarchical coarsening process, in which the system is iteratively decreased in size until it becomes trivial for a fast, direct computation to solve the system. The coarsening process at each level involves the segmentation of the graph Laplacian into coarse and fine vertices, building a matrix such that

$$\mathcal{L} = \begin{bmatrix} L_{cc} & L_{cf} \\ L_{fc} & L_{ff} \end{bmatrix}, \quad (2.3)$$

where the subscripts indicate whether coarsened vertices are connected to other coarsened vertices (L_{cc}), or fine vertices are connected to coarsened vertices (L_{fc}), and so on (note that $L_{fc} = L_{cf}^T$). Fine edges are disconnected from each other, making L_{ff} a diagonal matrix that

is therefore trivial to solve directly. To eliminate these edges and generate a coarser hierarchy, the method computes the Schur decomposition by creating a transformation matrix \mathcal{T}

$$\mathcal{T} = \begin{bmatrix} I_{cc} & 0 \\ -L_{ff}^{-1}L_{fc} & I_{ff} \end{bmatrix},$$

where I denotes an identity matrix of suitable size. The lower corner of T is the interpolation matrix for this level, specifically the interpolation matrix $P = -L_{ff}^{-1}L_{fc}$. When we apply the transformation matrix T to both sides of L

$$\mathcal{T}^T \mathcal{L} \mathcal{T} = \begin{bmatrix} L_{cc} - L_{cf}L_{ff}^{-1}L_{fc} & 0 \\ 0^T & L_{ff} \end{bmatrix}.$$

L_{ff} is easy to solve directly, both because it is diagonal and orders of magnitude smaller than $L_{cc} - L_{cf}L_{ff}^{-1}L_{fc}$. The upper block forms a Laplacian matrix that is coarser than the original but retains many similar properties. This reveals a recursive hierarchical structure that can be followed by setting $L_{cc} - L_{cf}L_{ff}^{-1}L_{fc}$ as the Laplacian for the next iteration and repeating the procedure. Once the size of the coarse Laplacian dips below a certain threshold, it is solved directly, and the resulting matrix (e.g. eigenvector matrix U) is interpolated back up the hierarchy (Fig. 8). In addition to the interpolation operators P defined at each step in the hierarchy, the method can use block Davidson smoothing [33] to refine the vectors even further.

In order to identify, or color, variables as fine or coarse, a loop iterates over each vertex, examining its neighborhood and both coloring the variable appropriately and locally sparsifying the graph.

At first glance, the method proposed in [35] and summarized in Algorithm 2, *hierarchical sparsify-and-compensate* (or hsc), appears significantly more complex than the GMG we investigated previously. However, there are many similar elements on closer inspection. Both methods rely heavily on sparsifying the graph so as to simplify the problem, but both sparsify the graph in such a way as to respect the underlying structure. In the GMG, alternating pixels are eliminated; here, a more sophisticated and localized heuristic is used. For each vertex v in the underlying graph, the algorithm examines any triangles it may participate in. If v participates in any triangles of connected vertices, the weakest edge is

identified and removed; this is the *sparsification* step. The weight of the removed edge is added to the weights of the remaining two edges of the triangle. This is the *compensation* step.

After sparsifying and compensating the graph, remaining nodes are identified as either coarse or fine in Step 9. Once colored, the variables are reordered as seen in Eq. 2.3, resulting in the block-diagonal representation of the sparsified matrix and providing the next iteration’s graph matrix, as well as means for computing the current level’s interpolation matrix.

2.5 PARALLEL AND DISTRIBUTED IMPLEMENTATIONS

For addressing larger datasets, there is a significant body of work dedicated to creating highly parallel implementations of these algorithms [41–44], allowing for much larger graphs to be analyzed. SLEPc [45] is one such implementation, performing many independent tasks, such as matrix-vector multiplication, in parallel to increase throughput. Nonetheless, there is still a limitation in terms of raw computing resources: a single physical machine can only store a finite and relatively small amount of information in memory. HEigen [46], included in the Pegasus library [47], is a distributed eigensolver on the open source MapReduce framework Apache Hadoop. However, these libraries have not been maintained, and do not implement multigrid methods. Apache Spark [48], one of the newest open source distributed frameworks, implements a distributed iterative method that invokes ARPACK on the head node once the system has sufficiently converged. Its GraphX library implements the SVD++ collaborative filtering algorithm for dimensionality reduction [49]. Apache Mahout [50], another open source framework, uses a stochastic method for performing dimensionality reduction, sacrificing a small amount of accuracy for a significant speedup. GraphLab [51] implements both a variant of SVD++, and a distributed version of the iterative Lanczos algorithm.

Supercomputers provide a highly optimized and parallel environment for such computations. However, most researchers do not have access to such resources, and both the cost

of maintenance and high learning curve for the specialized parallel protocols can make deployment of such an eigensolver prohibitive. Instead, we consider the case of distributed computing, in which a collection of commodity hardware is tied into a high-performance computing cluster [11] using one of several software frameworks.

2.6 CONCLUSIONS

Graphs are a convenient way of structuring interconnected data for the purpose of identifying discrete clusters and measuring the diffusion of information across the network. Critical to this analysis is the process of computing the eigenvectors and eigenvalues of the graph, i.e. studying the *spectrum* of the graph. However, this is a computationally expensive process; direct solvers operate in cubic time to the number of vertices, limiting their utility to very small graphs. We focus on hierarchical methods, algebraic multigrids in particular, for adaptively coarsening a graph until it can be solved trivially and using the computed interpolation operators to find a robust estimate of the final eigenvectors and eigenvalues.

Scalable solvers are of particular interest to us, as the problems we examine in biomedicine approach and, at times, exceed the capacity of even the most robust traditional solvers (e.g. ARPACK). While there are numerous distributed solutions to consider, we found hierarchical solvers to be especially compelling given their flexibility, empirical performance, and theoretical runtime guarantees. Furthermore, to the best of our knowledge, we found no existing distributed implementations of hierarchical eigensolvers. As we shall discuss in the following two chapters, the problem of finding the eigenvectors and eigenvalues of a large biomedical system has proved to be the bottleneck to new insights, motivating the need for a scalable solver. In Chapters 3 and 4, we will investigate the problems that highlighted this need. Chapter 5 will detail *Draenor*, our solution, and how it ties into our two projects.

Algorithm 2 hsc

```
1: Input:  $A, t, K$ 
2:  $A_0 \leftarrow A$ 
3:  $i = 1$ 
4: while  $\text{length}(A_{i-1}) > t$  do
5:   for  $v \in \text{vertices}(A_{i-1})$  do
6:      $t = \text{triangles}(v)$ 
7:      $\text{sparsifyAndCompensate}(t)$ 
8:   end for
9:    $A_{i-1} = \text{colorAndReorder}(A_{i-1})$ 
10:   $A_i = A_{i-1}[\text{cc}]$ 
11:   $i = i + 1$ 
12: end while
13:  $U_i \leftarrow \text{eig}(A_i, K)$ 
14: for  $d = [i : -1 : 1]$  do
15:    $U_{d-1} \leftarrow P_d U_d$ 
16: end for
17: return  $U_0$ 
```

3.0 LEARNING CILIARY MOTION PHENOTYPES

3.1 INTRODUCTION

Cilia are microtubule based hair-like projections of the cell that can be motile or immotile, and in humans are found on nearly every cell of the body. Ciliopathies, or diseases with disruption of nonmotile or motile cilia function, can result in a wide spectrum of disorders. In primary ciliary dyskinesia (PCD), cilia in the airway that normally beat in synchrony to mediate mucus clearance can exhibit dyskinetic motion or become immotile, resulting in severe sinopulmonary disease [52–55]. As motile cilia are also required for left-right patterning [56, 57], PCD patients can exhibit mirror symmetric organ placement as in Kartagener’s syndrome, or randomized left-right organ placement as in heterotaxy [58]. Patients with congenital heart disease (CHD) and heterotaxy exhibit a high prevalence of ciliary motion (CM) defects similar to those seen with PCD [59]. This was associated with increased respiratory complications and poor postsurgical outcomes [59–61]. Similar findings were observed in patients with a variety of other CHD, including transposition of the great arteries (TGA), a CHD that may also arise from left-right patterning defects [62] with an incidence as high as 1 in 200 [63]. Interestingly, respiratory CM defects and airway inflammatory disease have also been reported in patients with Leber congenital amaurosis [64], a ciliopathy involving cone-rod dystrophy in the connecting cilium of the retina. Diagnosing CHD patients with CM abnormalities prior to surgery may provide the clinician with opportunities to institute prophylactic respiratory therapies to prevent these complications. Together these findings suggest motile cilia dysfunction may have broader clinical impact beyond PCD. Therefore, the role and importance of diagnosing CM abnormalities will only grow.

Given the long-term prognosis for patients with airway clearance defects can be improved

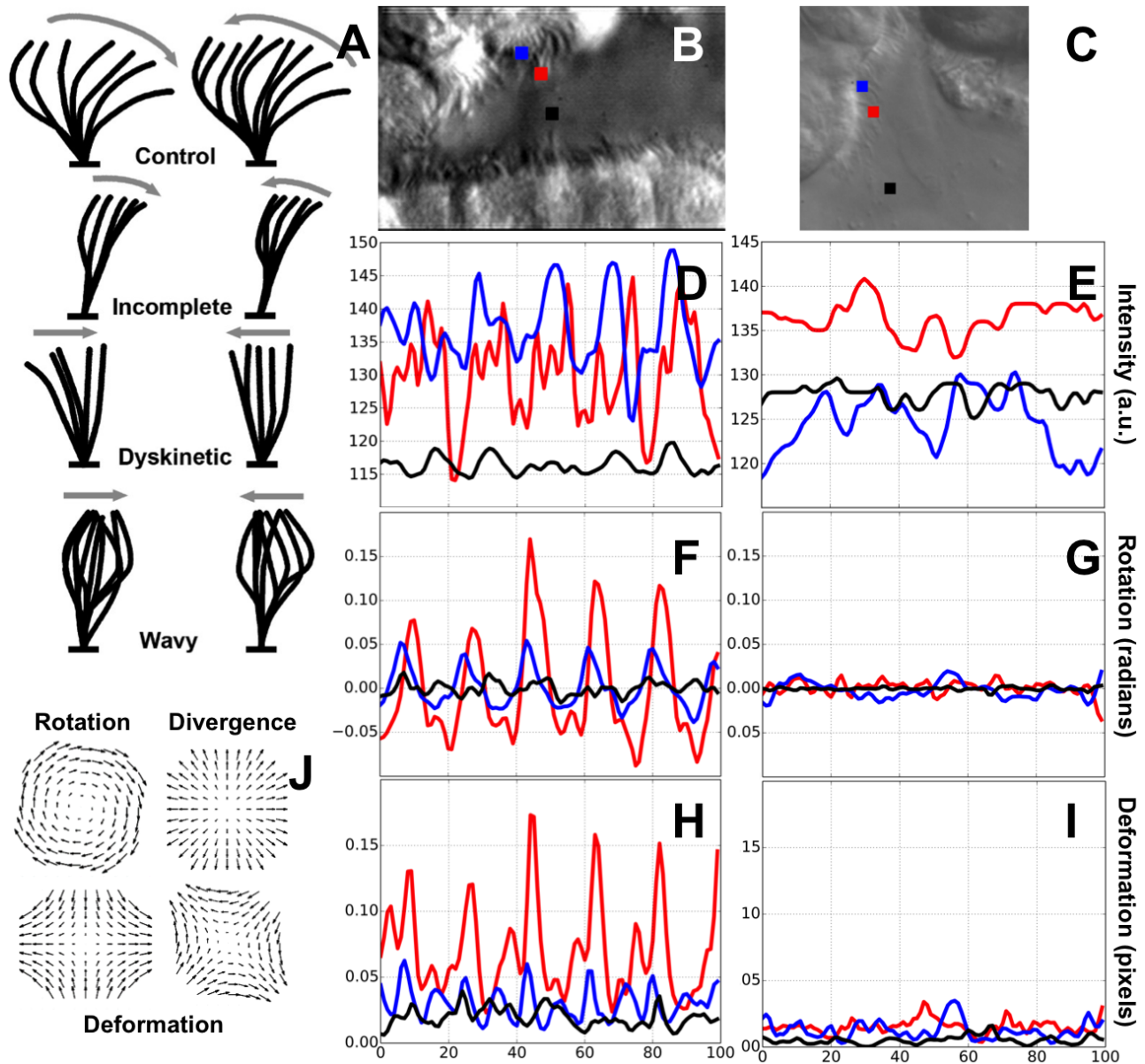


Figure 10: **Properties of ciliary motion.** (A) Schematic (hand drawn) diagrams of ciliary motion subtypes to aid clinical diagnosis. (B, C) Single frame of a video of normal CM (B) and abnormal CM (C) with three pixels identified: blue (proximal), red (distal), and black (background). See Videos S1-S9 for scale bars. (D, E) Time series of *gray-level pixel intensities* over 100 frames at each of the pixel locations in (B) and (C), respectively: (D) shows pixel intensity variations for corresponding locations in (B); (E) corresponds to pixel locations in (C). (F, G) Time series of *rotation* over 100 frames at each of the pixel locations in (B) and (C), respectively. (H, I) Time series of *deformation* amplitude over 100 frames at each of the pixel locations in (B) and (C), respectively. (J) Elemental components of rotation, deformation, and divergence, shown here in a template form. Deformation is a vectorial quantity requiring two templates for measurement.

with early diagnosis and intervention, a robust pipeline for diagnosis of airway CM defects is clinically compelling. One of the current methods for assessing CM entails the use of videomicroscopy for CM analysis of nasal brush biopsies [65]. While ciliary beat frequency (CBF) can be quantitated from these videos [66–68], this has low sensitivity for abnormal CM due to a variety of technical and methodological limitations [54, 69–73]. Even under ideal conditions, CBF does not capture the broad distribution of frequencies present in ciliary biopsies. Normal CM is comprised of a forward power stroke followed by a recovery stroke coordinated across the multi-ciliated airway epithelium. In contrast, abnormal CM may be described as dyskinetic, wavy, asynchronous, and/or with incomplete stroke. Schematics as used by clinicians to diagnose CM are shown in Fig. 10A. CM exhibits a great deal of variability, even within the same region of cilia. Time series pixel intensity variations are shown for three different pixels in ciliary biopsies of healthy motion (Fig. 10B) and dyskinetic motion (Fig. 10C). While the healthy motion (Fig. 10D) demonstrates stronger periodicity than abnormal motion (Fig. 10E) in general, the distal (red) and proximal (blue) regions of the cilia nonetheless exhibit heterogeneity of motion that can be problematic for characterizing motion within a single phenotype. Clinicians often employ visual assessment of ciliary beat pattern to augment CBF measurements; however, this relies on reviewer experience and is therefore highly subjective and error prone [71, 74]. Electron microscopy (EM), considered one of the most reliable methods for PCD diagnosis, cannot identify all PCD patients given some mutations causing PCD do not cause ultrastructural defects; in some instances these have been associated with high CBF and abnormal beat pattern [69]. Furthermore, it is difficult to compare results of the diagnostic ensemble in cross-institutional studies. CM heterogeneity stipulates a larger-scale method for aggregate analysis, and a quantitative method is needed to ensure cross-institutional relevance of the results. The types of motion that human eyes are optimized to detect are not necessarily those that are most clinically relevant when identifying CM phenotypes. Computational methods, however, can be trained to detect the best types of motion for identifying CM defects, and either present them in a quantitative format, or classify the motion phenotypes with greater precision and objectivity to make them suitable for clinical diagnosis.

Our approach is twofold. First, we design an automated CM classification framework

that recapitulates manual expert review, providing a computational blackbox for classifying CM in a high-throughput manner. Given a high-speed digital biopsy video, we decompose the CM into idealized elemental components (Fig. 10J) that form a “digital signature,” a quantitative description of the CM. We used two independent datasets of differing quality (Table 1, Fig. 11) to test our classification framework.

Second, we build on the computational framework and design a fully-automated, unsupervised pipeline for discovering novel, latent CM phenotypes. It is generally believed that CM phenotypes encompass a richer spectrum than strictly normal and abnormal motion, with subcategories of motion existing in both overarching phenotypes (Fig. 10A). However, in the absence of rigorous quantitative methods for measuring CM, there has previously been no method by which these motion subtypes can be identified and defined. Here, we demonstrate the utility of unsupervised spectral analytics on a larger scale to discover latent CM phenotypes and begin the process of building a quantitative CM library.

3.2 DATA ACQUISITION AND STUDY DESIGN

3.2.1 Subject recruitment and data cohort breakdown

All study protocols were approved by the University of Pittsburgh Institutional Review Board.

Nasal epithelial tissue was collected by curettage of the inferior nasal turbinate under direct visualization using an appropriately sized nasal speculum utilizing rhino-Probe (Arlington Scientific, Springville, UT). Nasal brushings and tracheal biopsies have been shown to provide tissue of comparable quality, and shown similar pathology with increased sensitivity over nasal biopsies [75–77]. Three passages were made and the collected tissue was resuspended in L-15 medium (Invitrogen, CA) for immediate videomicroscopy using a Leica inverted microscope with a 100x oil objective and differential interference contrast (DIC) optics. Digital high-speed videos were recorded at a sampling frequency of 200 Hz using a Phantom v4.2 camera. At least 8 videos were obtained per subject. These videos were

used in our study. However, to establish ground truth CM, these samples were reciliated, and these reciliated biopsies were analyzed by a panel of researchers (M.Z., R.F., C.W.L.) blinded to the subject’s clinical diagnosis, nasal nitric oxide values, and reciliation results. This process of establishing ground truth using reciliated samples while performing the computational analysis on original samples eliminates, or otherwise minimizes, the possibility of introducing secondary CM defects as a result of tissue sampling. After reviewing all reciliated videos, a call of normal or abnormal CM was made by consensus. Where differences could not be resolved, the majority vote was accepted.

We performed our analysis on two independent data cohorts. From Children’s Hospital of Pittsburgh (CHP), 49 patients were recruited with TGA. Additionally, 27 healthy subjects were recruited to serve as controls. Informed consent was obtained from adult subjects or parents/guardians of children, with assent obtained from children over seven years of age. The video samples were subjected to rigorous quality control, and data from numerous subjects were discarded on the grounds of spurious camera motion, variable lighting conditions, poor focus, or other recording artifacts. In addition, we recruited 5 PCD patients to serve as abnormal controls. The resulting corpus formed the first data cohort (CHP), consisting of high-quality videos with minimal artifacts and noise, depicting biopsies from 49 individuals (27 healthy controls, 5 PCD controls, and 17 TGA patients).

The second cohort consisted of nasal biopsy videos from 31 subjects from Children’s National Medical Center (CNMC) in the Nakhleh *et al* study [60]. The cohort included 27 subjects who were patients with heterotaxy; 17 had normal CM and 10 had abnormal CM, as evaluated by a blinded panel of investigators in an identical manner to the CHP cohort. Four additional subjects were included as PCD controls. The original dataset was used with only minimal quality control, discarding videos that were uninterpretable to manual evaluation. Thus, the first cohort (CHP) fulfilled the role of establishing the baseline viability of our framework, and the second cohort (CNMC) tested its robustness to noisier data. See Table 1 and Fig. 11 for the breakdown of the cohorts.

Table 1: Description and breakdown of datasets

	Individuals	Videos	ROIs
--	-------------	--------	------

Children’s Hospital of Pittsburgh (CHP)

Healthy Controls	27	76	114
PCD Controls	5	38	96
CHD/TGA with Abnormal CM	17	56	121
Total	49	170	331

Children’s National Medical Center (CNMC)

PCD Controls	4	25	58
Heterotaxy with Normal CM	17	65	139
Heterotaxy with Abnormal CM	10	31	65
Total	31	121	262

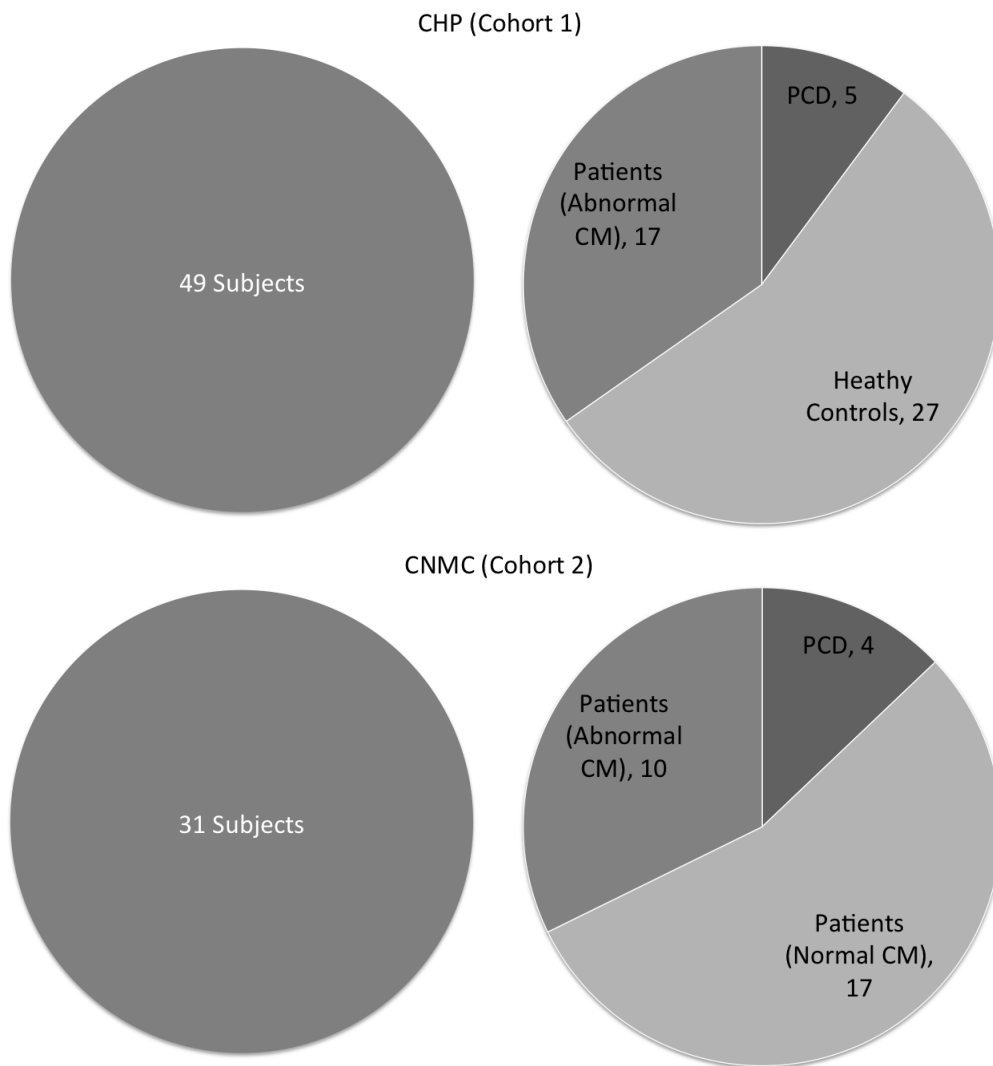


Figure 11: **CHP and CNMC dataset breakdowns.** Relative fractions of the subject demographics in both of our ciliary motion data cohorts.

3.2.2 Digital video annotation and preprocessing

Collaborators uploaded high-speed digital videos of ciliary biopsies at 200 fps in AVI format. After upload, the user was presented with an HTML5 canvas interface through which they could specify regions of interest (ROIs, Fig. 13) by drawing boxes over a still frame of the video (Fig. 12). ROIs were drawn wherever ciliated cells were seen in profile in order to avoid overlapping cells or multiple layers of ciliated cells; multiple cilia beating in different planes could spuriously lead us to believe the motion was asynchronous. Only areas where mucus or cell debris is seen overlying the cilia and interfering with motion are excluded. Each ROI inherited the normal or abnormal label of the patient from which it was derived. For each subject, an average of three to four videos were uploaded, and an average of five to eight ROIs were selected, though the ROI count per patient varied from as few as two to as many as 18 (performance of our framework as a function of ROIs per patient is shown in Fig. 22). All subsequent analysis was performed at the ROI level.

We preprocessed the videos prior to cross-validation in order to filter out noisy pixel data in the ROIs, such as pixels depicting cells or space beyond the cilia. This method discarded pixels whose intensity changes fell below a set threshold. The threshold value was adaptive and specific to each ROI, as the intensities between ROIs varied greatly. For a single ROI, we computed the standard deviation σ_i of the time-varying intensity changes at each pixel p_i and constructed a histogram of these standard deviations. We used the Kolmogorov-Smirnov distance metric [78] to determine whether the histogram more closely resembled a gamma distribution or a Gaussian distribution. In the former case, we used the distribution’s peak, or σ_{peak} , as the pruning threshold value, and discarded all pixels p_i for which $\sigma_i < \sigma_{peak}$. If the distribution was better approximated by a Gaussian, we used the distribution’s mean, or σ_{mean} , and discarded all pixels p_i for which $\sigma_i < \sigma_{mean}$. We performed a connected component analysis on the remaining pixels and discarded all but those in the largest component. Subsequent analysis was performed only on these remaining pixels.

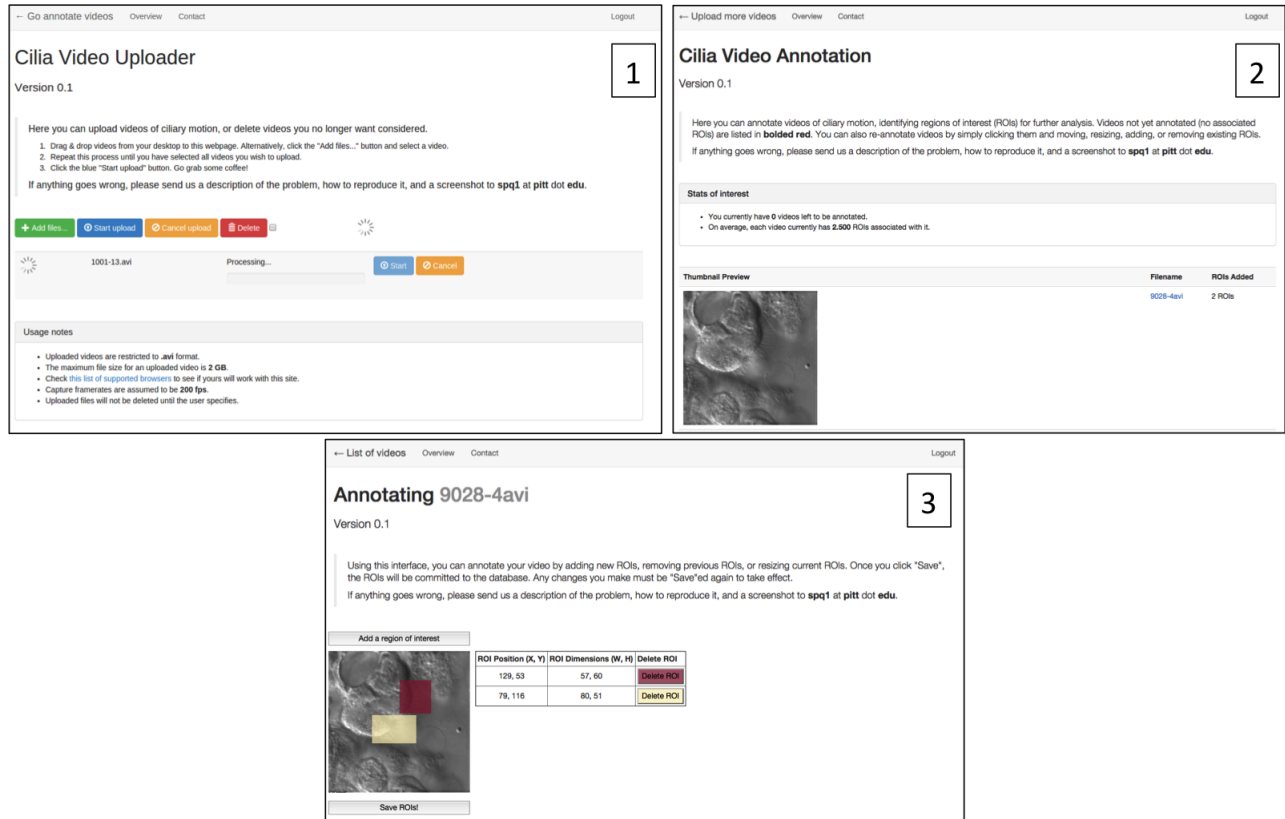


Figure 12: **Website proof-of-concept.** We implemented a barebones proof-of-concept website for uploading, annotating, and analyzing videos of ciliary motion. This was used over the course of this study to enhance remote collaboration. This shows the three manual steps involved in uploading (1) and annotating (2 and 3) videos.

3.2.3 Software

Python 2.7 was used to implement the analysis pipeline. We used the scientific computing packages NumPy and SciPy, and the plotting package Matplotlib. For computing optical flow vectors, we used the pyramidal Lucas-Kanade [79] implementation packaged in OpenCV 2.4 and confirmed its viability using the software package by Black *et al* [80] for Matlab. For video collection and annotation, we used a website built using the open source jQuery-File-Upload application (<https://github.com/blueimp/jQuery-File-Upload>) on an Apache 2.2 server running PHP 5. Annotations were stored in a MySQL database. Video transcoding was performed using ffmpeg. Statistical classification was performed using the Python scikit-learn machine learning library [81], which uses the popular `libsvm` implementation for support vector machines. All of these packages (with the exception of Matlab) are publicly available under open source licenses.

3.3 REPRESENTING CILIARY MOTION AS DYNAMIC TEXTURES

We hypothesize that CM is an instance of dynamic texture (DT) [4, 82]. DTs are characterized by rhythmic motions of particles subjected to stochastic noise [83], and are active areas of research in the fields of computer vision and machine learning. Examples of DTs include many familiar motion patterns such as flickering flames, billowing smoke, rippling water, or grass in the wind. Likewise, cilia beat in rhythmic waves with some stochastic behavior that collectively determines their CM. DT analysis has been shown to be an effective analysis method in other biomedical contexts, such as localizing cardiac tissue in 3D time-lapse heart renderings [84] and to quantify thrombus formations in time-lapse microscopy [85]. DT analysis relies on using linear dynamics systems, such as autoregressive (AR) models [86], to parameterize the components of DT motion.

Our approach is to use differential invariants, referred to here as *elemental components*, rather than pixel intensities, in order to quantitatively represent the CM. These are fundamental quantities of motion that can be difficult to detect by manual inspection, but which

computational methods are extremely well-suited to analyze. They are computed as functions of the pixel intensities, and are computed at each pixel. To determine the elemental components, we first compute the *optical flow* of the CM. Optical flow [80] (Fig. 13) is used to quantify the direction and magnitude of apparent motion observed at each pixel between two successive frames. We do not explicitly delineate or track the cilium when determining CM; rather, we estimate CM using spatial and temporal derivatives of the optical flow [87,88] (see Appendix for full optical flow and elemental components derivations). We derive the elemental components from the optical flow. The specific quantities are instantaneous *rotation* (curl), *divergence* (dilation), and *deformation* (biaxial shear) (Fig. 13), computed at each pixel position in each frame. The efficacy of elemental components for DT analysis has been demonstrated in previous studies [89–91]. We excluded divergence from this analysis, as the quantity did not offer insight into the differences between CM types in this study or in our previous work [4]. This is likely a consequence of divergence as a dilation quantity: the vast majority of our videos depict cilia moving within the two-dimensional plane of the video. With little motion towards or away from the camera, dilation was near-constant for most of the video samples.

Like pixel intensities, elemental components exhibit periodic temporal behavior (Fig. 10F-I) that can be analyzed with similar techniques. Rotation and deformation computed for healthy motion (Fig. 10F,H) show strong periodic behavior and high magnitudes, particularly at the distal point on the cilia. By contrast, the rotation and deformation in dyskinetic cilia (Fig. 10G,I) show little periodic behavior in addition to markedly reduced magnitudes of each. However, unlike 8-bit grayscale pixel intensities, these quantities can be compared directly between video samples; pixel intensities would require a normalization step that would need to be tailored to the specific lighting conditions and microscope settings. Further, and most importantly, these quantities are orientation-invariant: automated CM analysis can be conducted regardless of the orientation of the cilia. That is, the computed elemental components for the same video will be identical irrespective of orientation of the light source or the plated ciliary biopsy relative to the microscope camera. We will revisit this property in the following section.

A critical hurdle in the current CM evaluation process is accounting for and capturing

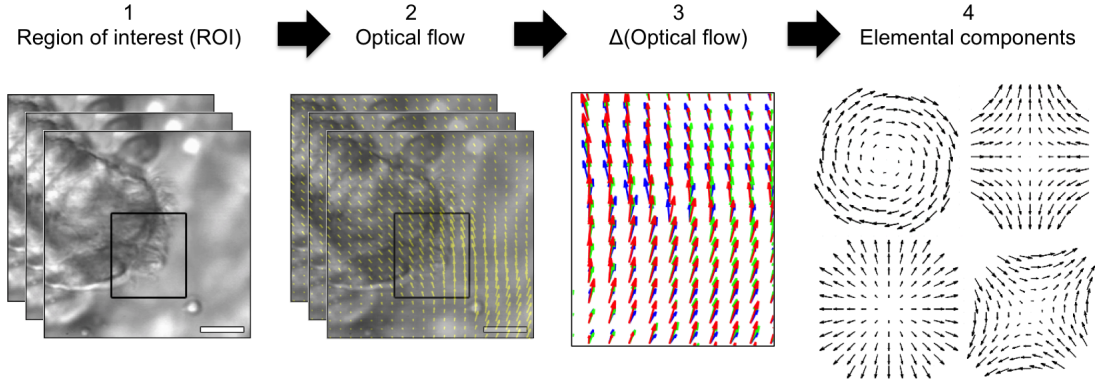


Figure 13: **Derivation of elemental components.** First, the ROIs for each video are selected and programmatically identified. Second, optical flow is computed at each pixel location for every frame. Third, the first-order derivatives of the optical flow are computed (red: optical flow at frame t ; green: optical flow at frame $t + 1$; blue: optical flow at frame $t + 2$). Finally, the derivatives of the optical flow are used to compute the elemental components of rotation, divergence, and deformation.

the significant motion heterogeneity in a robust quantitative way. There is a great deal of noise present in CM video samples, confounding both manual inspection and any naive automated analysis implementations. A single nasal brush biopsy often contains a spectrum of beat frequencies and motile behaviors. Consequently, a single numerical value such as CBF cannot encapsulate an entire motion phenotype such as normal or abnormal. Motion heterogeneity can arise from multiple sources, some an inherent property of the CM, some technical artifacts: overlapping cilia with distinct motions, background particulate obstructing proper view of the cilia, and video capture artifacts such as changes in the plane of focus or translational motion of the sample. Therefore, one challenge is to make the CM analysis framework robust to these sources of heterogeneity while also respecting the heterogeneity that exists within each cilium’s beat patterns and distinct regions of cilia. Briefly, our strategy is to employ higher-order DT statistics to capture the distributions of these elemental quantities, and use them to develop the digital signatures that will be used to differentiate normal from abnormal CM.

3.4 DERIVATION OF DIFFERENTIAL INVARIANTS

In this section, we motivate the use of differential features of optical flow for characterizing ciliary motion dynamics. The discussion below follows the notation from Kooenderink *et al* [87] and Kovesi *et al* [92].

3.4.1 Optical Flow

Optical flow computation follows from invoking the standard *brightness constancy assumption*,

$$I(x, y, t) = I(x + u\delta t, y + v\delta t, t + \delta t), \quad (3.1)$$

which states that image intensity I (or a filtered version of it) at a location (x, y) at time t is preserved locally for small changes $(u\delta t, v\delta t)$ observed in space in a small time interval δt . Here (u, v) are the horizontal and vertical image velocity components of the optical flow vector \vec{f}^T at pixel location (x, y) .

A first-order approximation of the right hand term in the brightness constant equation above gives rise to the gradient constraint:

$$I_x u + I_y v + I_t = 0, \quad (3.2)$$

where the subscripts x , y , and t on image intensity I denote partial derivatives of the image structure at location (x, y) .

The gradient constraint is pooled over a small image neighborhood around pixel (x, y) to form an overdetermined system of linear equations from which the optical flow vector (u, v) is estimated. We use a variation of the classical optical flow algorithm suggested by Black *et al* [80] that incorporates a non-local smoothness term to integrate information over larger neighborhoods.

While optical flow vector $\vec{f} = (u, v)^T$ provides information on the image dynamics, the first-order flow derivatives: (u_x, u_y, v_x, v_y) , can be additionally used to derive a linear (affine) model of optical flow, and provide a statistical means of characterizing ciliary dynamics.

3.4.2 Differential Invariants

Consider two spatially nearby image points $\vec{r}_1 = \vec{r}$ and $\vec{r}_2 = \vec{r} + \delta\vec{r}$ along a cilium. The vector $\delta\vec{r} = \vec{r}_2 - \vec{r}_1$ gives their relative position. We assume that the points move according to their optical flow velocities $\vec{f}_1 = \vec{f} = (u, v)^T$ and $\vec{f}_2 = \vec{f} + \delta\vec{f}$ and after a small time interval δt they are at locations $\vec{r}_1' = \vec{r}_1 + \vec{f}_1\delta t$ and $\vec{r}_2' = \vec{r}_2 + \vec{f}_2\delta t$. It follows that

$$\vec{r}_2' - \vec{r}_1' = (\vec{r}_2 - \vec{r}_1) + (\vec{f}_2 - \vec{f}_1) \delta t, \quad (3.3)$$

$$\delta\vec{r}' = \delta\vec{r} + \delta\vec{f}\delta t. \quad (3.4)$$

Given the spatial nearness of the two points \vec{r}_1 and \vec{r}_2 , we can relate the flow vectors \vec{f}_1 and \vec{f}_2 by Taylor series expansion that uses first-order differentials of optical flow:

$$\vec{f}_2 \approx \vec{f}_1 + \frac{\partial \vec{f}_1}{\partial \vec{r}} \delta\vec{r} + \dots, \quad (3.5)$$

$$\vec{f}_2 \approx \vec{f}_1 + \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} \delta\vec{r} + \dots, \quad (3.6)$$

where (u_x, u_y, v_x, v_y) are elements of the spatial derivative of optical flow, i.e. flow gradient: $\frac{\partial \vec{f}}{\partial \vec{r}}$. As shown by Kooendernik and van Doorn, the flow gradient can be further decomposed into scaling (divergence), shearing (deformation) and rotational (curl) components. These are scalar quantities defined as

$$\text{div } \vec{f} = u_x + v_y \quad (3.7)$$

$$\text{rot } \vec{f} = v_x - u_y \quad (3.8)$$

$$\text{def } \vec{f} \cos(2\mu) = u_x - v_y \quad (3.9)$$

$$\text{def } \vec{f} \sin(2\mu) = u_y + v_x \quad (3.10)$$

where μ is the angle of maximal distortion. The quadruplet of quantities:

$$\text{div } \vec{f}, \text{rot } \vec{f}, \left(\text{def } \vec{f}\right) \cos(2\mu), \left(\text{def } \vec{f}\right) \sin(2\mu)$$

form a linear space and provide an equivalent representation of flow gradient $\frac{\partial \vec{f}}{\partial \vec{r}}$. Observe that the deformation magnitude and orientation can be derived as:

$$\text{def } \vec{f} = \sqrt{(u_x - v_y)^2 + (u_y + v_x)^2}, \quad (3.11)$$

$$2\mu = \arctan\left(\frac{u_y + v_x}{u_x - v_y}\right). \quad (3.12)$$

The quantities $\text{def } \vec{f}$, $\text{div } \vec{f}$, and $\text{rot } \vec{f}$ are differential invariants as they are independent of coordinate system used to measure the flow.

Using these definitions the velocity gradient can be rewritten as:

$$\begin{aligned} \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix} &= \frac{\text{div } \vec{f}}{2} \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} + \frac{\text{curl } \vec{f}}{2} \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix} + \\ &\quad \frac{\text{def } \vec{f}}{2} \begin{pmatrix} \cos(2\mu) & \sin(2\mu) \\ \sin(2\mu) & -\cos(2\mu) \end{pmatrix}. \end{aligned} \quad (3.13)$$

Fig. 14 illustrates the geometric/image distortions with which each of these differential features are associated.

3.4.2.1 Divergence Divergence is image distortion seen geometrically as a local isotropic expansion with speed $\frac{1}{2} \text{div } \vec{f}$ about a focus of expansion (Fig. 14A). We do not expect these distortions to appear in the lateral views of the cilia, but they could be useful in characterizing the ciliary motions captured by a perpendicular view of the cilia. Divergence is invariant to the orientation of the cilia in the image plane.

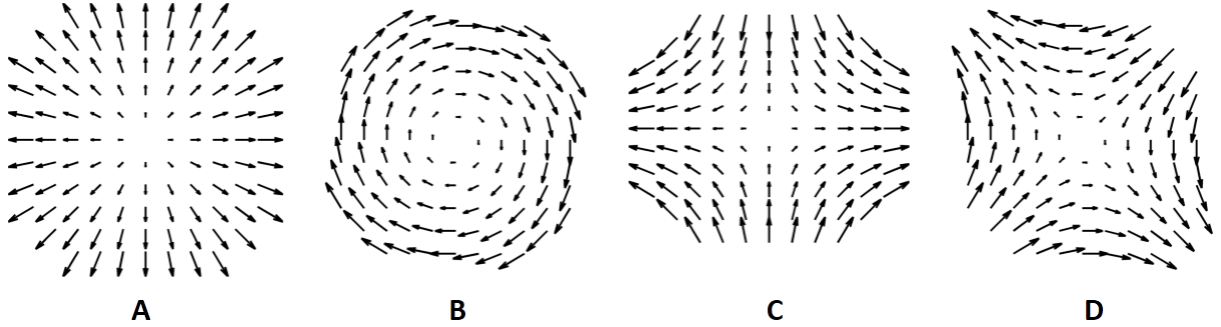


Figure 14: **Basic distortion types.** (A) divergence, (B) curl, and (C, D) deformation. We omit divergence in this study, as the majority of our video data is taken from a top-down, 2D perspective, thereby all but eliminating the utility of a feature that tracks distortion along the z-axis.

3.4.2.2 Curl The most salient features of ciliary motions are the sweeping forward and backward strokes. Curl captures the local rotation of cilia with angular velocity $\frac{1}{2} \text{rot } \vec{f}$. Note this rotation is a component perpendicular to the viewing direction (Fig. 14B). Curl is orthogonal to divergence. Like divergence, curl is invariant to the orientation of the cilia in the image plane.

3.4.2.3 Deformation Deformation measures distortions that affect orientation of a ciliary region while preserving apparent areas (Fig. 14C, D). Two axes, the axis of maximal extension and the axis of maximal contraction, form an orthogonal basis for describing all possible motion field distortions or shearing motions. Since cilia are stuck to the cell wall, it is more appropriate to see their motions as having both a rotational component and a shearing motion (hence a directed shear).

3.4.3 Differential feature filters

Ciliary motion videos have high sampling frequency (200Hz) relative to their natural beats ($\approx 10\text{Hz}$). While it is easy to construct the optical flow derivatives with Gaussian derivatives,

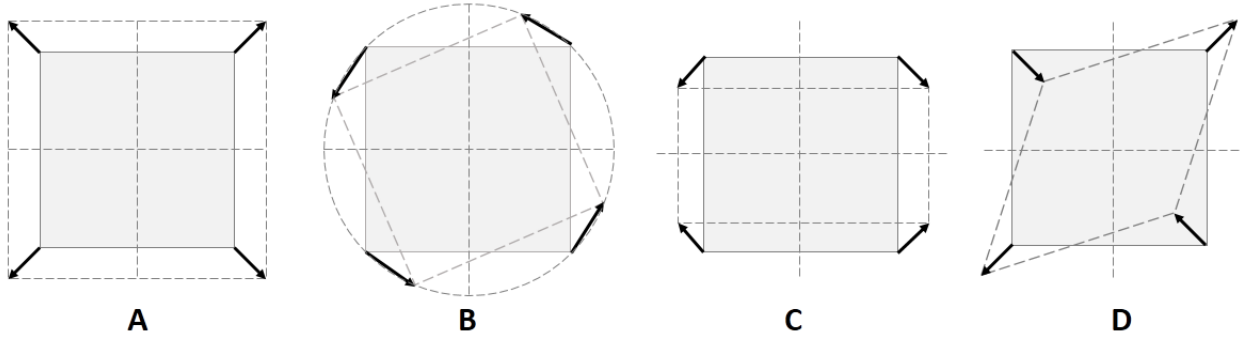


Figure 15: **Atomic flow detectors.** (A) divergence and (B) curl detectors with radius of 6 pixels. (C) and (D) deformation detectors of radius 6 and $\mu = 0^\circ$ and $\mu = 45^\circ$.

it is also instructive to consider how to design a filter mask that can elicit differential flow information. Intuitively, the detector template mask would resemble a miniature vector field exhibiting *atomic* motion types: divergence, curl or deformation. The magnitudes of individual vectors in the motion template will be proportional to distances from the center of the filter and the vector directions will be a function of atomic motion type. Indeed, Eqs. 3.7, 3.8, 3.9, and 3.10 make it obvious how to construct these filters, as shown in Fig. 15.

Fig. 15A shows a divergence filter mask that has vectors that point radially outward. The curl filter has tangential vectors as shown in Fig. 15B. Observe that the curl filter is orthogonal to the divergence filter. For deformation two orthogonal masks are necessary for capturing distortion in all directions. For illustration in Fig. 15C and D, we show two deformation masks with maximal expansion axes aligned to 0 and $\pi/4$ degrees. To reduce corner artifacts, a circular envelope is applied on each of these masks.

3.5 COMPUTING DIGITAL SIGNATURES OF CILIARY MOTION PHENOTYPES

3.5.1 Autoregressive models

Our first method for representing and quantifying CM involves the use of autoregressive (AR) processes. AR models are linear dynamics systems that are useful for representing periodic signals, and are among the state-of-the-art DT analysis methods [82–85]. While linear models can be limited in their ability to capture complex behaviors, the high capture speed of most CM videos (200Hz) virtually guarantees that linear transformations will be more than sufficient to model the motion between successive frames. We use the formulation of AR processes as defined in the Materials and Methods [83, 86],

$$\vec{y}_t = C\vec{x}_t + \vec{u}_t \tag{3.14}$$

$$\vec{x}_t = B_1\vec{x}_{t-1} + B_2\vec{x}_{t-2} + \dots + B_d\vec{x}_{t-d} + \vec{v}_t \tag{3.15}$$

where Eq. 3.14 models the *appearance* of the cilia \vec{y} at a given time t (plus a noise term \vec{u}_t), and Eq. 3.15 represents the *state* \vec{x} of the CM in a low-dimensional subspace defined by an orthogonal basis C at time t , and how the state changes from t to $t + 1$ (plus a noise term \vec{v}_t).

Eq. 3.14 is a decomposition of each frame of a CM video \vec{y}_t into a low-dimensional state vector \vec{x}_t and a white noise term \vec{u}_t , using an orthogonal basis C (Fig. 16A). This basis was derived using Singular Value Decomposition (SVD). The input to the SVD consisted of a raster-scan of the original video; that is, the video was restructured into a matrix where each row corresponded to a single pixel from the video, and each column was a frame (or the value of that pixel in a given frame). Therefore, if the height and width of the video in pixels were given by h and w respectively, and the number of frames as f , the dimensions of the raster-scanned matrix would be $hw \times f$.

A core assumption in DT analysis is that the DT lives in a low-dimensional subspace as defined by the principal components C ; that is, a significant majority of the variance in

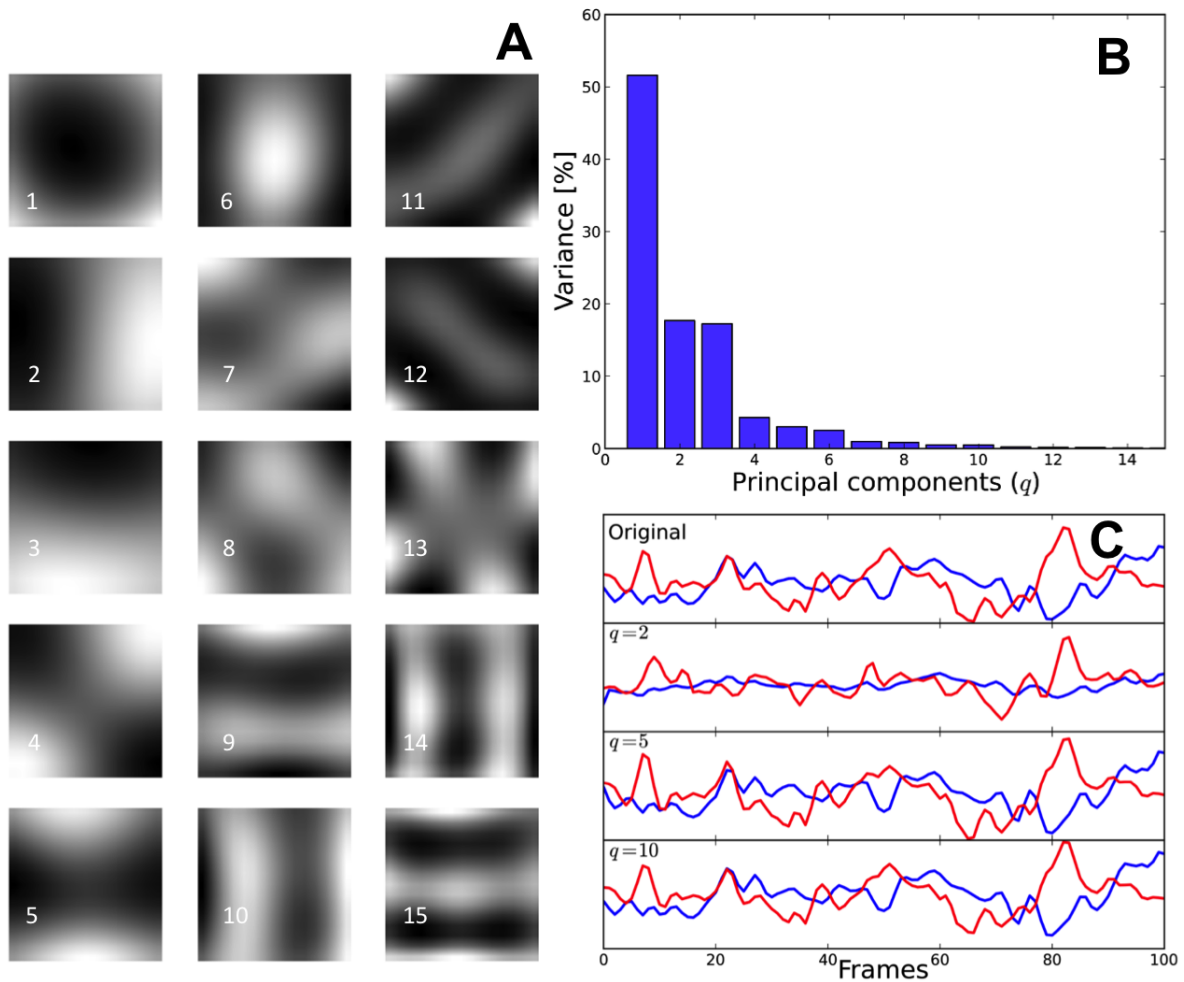


Figure 16: **Autoregressive representation of ciliary motion.** (A) Top 15 principal components of CHP rotation data. The first q are used to compute the AR motion parameters. (B) Relationship between each principal component and the amount of variance each captures from the original rotation signal. (C) CM amplitudes for normal (blue) and abnormal (red) CM as they appear reconstructed using the first principal components ($q = 2$), the first five ($q = 5$), and the first 10 ($q = 10$) as compared to the original rotation signal (top). A small number of principal components can reconstruct the original signal with a high degree of fidelity.

the data can be explained with only a few dimensions (Fig. 16B-C). Once the data \vec{y}_t are projected into this subspace, the motion of the DT \vec{x}_t can be modeled with relatively few parameters by virtue of its low dimensionality, relative to \vec{y}_t . We can think of this motion as a linear process: the position of the cilia in this low-dimensional space at time $t + 1$ is a linear function of its position at time t . Eq. 3.15 reflects this intuition: position \vec{x}_t of the CM is a function of the sum of d of its previous positions $\vec{x}_{t-1}, \vec{x}_{t-2}, \dots, \vec{x}_{t-d}$, each multiplied by corresponding coefficients $\mathcal{B} = \{B_1, B_2, \dots, B_d\}$. The noise terms \vec{u} and \vec{v} are used to represent the residual difference between the observed data and the solutions to the linear equations; often, these are modeled as Gaussian white noise.

When comparing DTs using AR models, each DT is often represented as a combination of its coefficients \mathcal{B} and its subspace C ; explicitly, the DT is represented as $M = (\mathcal{B}, C)$ [93]. However, CM analysis differs in that we hypothesize all CM to live within the *same subspace*; that is, all instances of CM share the same orthogonal basis C and therefore the same principal components. What differentiates CM using this method, we claim, is the pattern of motion in this subspace defined by C . Fig. 17 provides strong evidence for this hypothesis: we averaged pairwise angles between the first 20 principal components from each video of CM. Each pairwise comparison was orthogonal or nearly orthogonal, suggesting they are derived from the same subspace. Therefore, we represent each instance of CM with only the coefficients \mathcal{B} ; these formed the “digital signature” of the CM sample for the AR method.

The orientation-invariance property of the elemental components are critical to the success of AR models. PCA realigns the axes of the data in the directions of maximal variance (Fig. 16A). If we perform PCA on a video of raw pixel intensities, this will result in different principal components depending on the relative orientations of the structures in the video. For example, if a video depicting a profile-view of cilia beating from left to right, the principal components of this video would be different from those of the very same video after rotating it 90 degrees. However, since rotation and deformation are computed from the magnitudes of optical flow derivatives (see Appendix), the relative orientation of structures as defined by the pixel intensities does not matter in the computations, thereby making rotation and deformation orientation-invariant. Consequently, a video can be rotated 90 degrees relative to another, and the rotation and deformation quantities will still capture properties of the

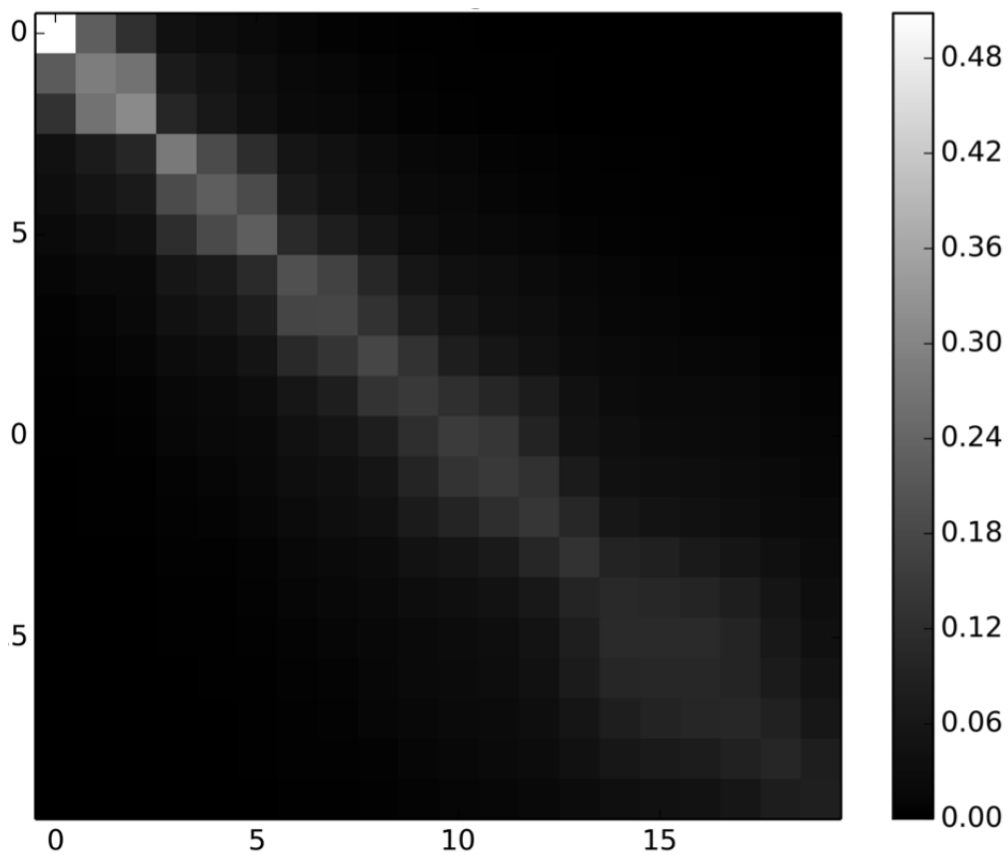


Figure 17: **Pairwise angles between principal components of CM.** Mean-squared average of all pairwise inner products of principal components derived from all image patches of ciliary motion used in this study. The large number of orthogonal (0 inner product) basis vectors provides strong evidence that all instances of CM occupy the same subspace.

CM regardless of the orientation of the cilia. In practical terms, this prevents the introduction of any additional and potentially onerous requirements on the format of the data that can be used in this framework. This underscores one of the main advantages of using orientation-invariant properties of the CM such as rotation and deformation in lieu of pixel intensities for automated DT analysis.

3.5.2 Magnitude and frequency histograms

Our second method for quantifying CM involves computing a series of histograms to represent the distributions of elemental components present in CM samples. Our motivation for this method is interpretability: whereas AR models are complex statistical tools and represent the state-of-the-art for DT analysis, histograms are still powerful tools for statistical analysis that are also extremely valuable for gaining an intuition for the behaviors and characteristics of the data. We aim to provide that intuition using this method.

For each CM sample, we compute four histograms: a *rotation magnitude histogram* (RMH, Fig. 18A), a *deformation magnitude histogram* (DMH, Fig. 18B), a *rotation frequency histogram* (RFH, Fig. 18C), and a *deformation frequency histogram* (DFH, Fig. 18D). The magnitude histograms were built by placing all rotation and deformation values computed at each pixel position (Fig. 10F-I) into respective histograms. The frequency histograms were computed by transforming the time-series rotation and deformation data into the frequency domain using a Fast Fourier Transform. Specifically, we computed a spectrogram [94], or a sliding average of frequency spectra, which resulted in a robust Fourier representation of the original signal. We then computed the dominant frequency present at each pixel position (analogous to computing CBF from pixel intensity variations, Fig. 19), and placed the dominant frequencies from rotation and deformation into respective histograms. These four histograms collectively formed a “digital signature” of the CM sample for the histogram method.

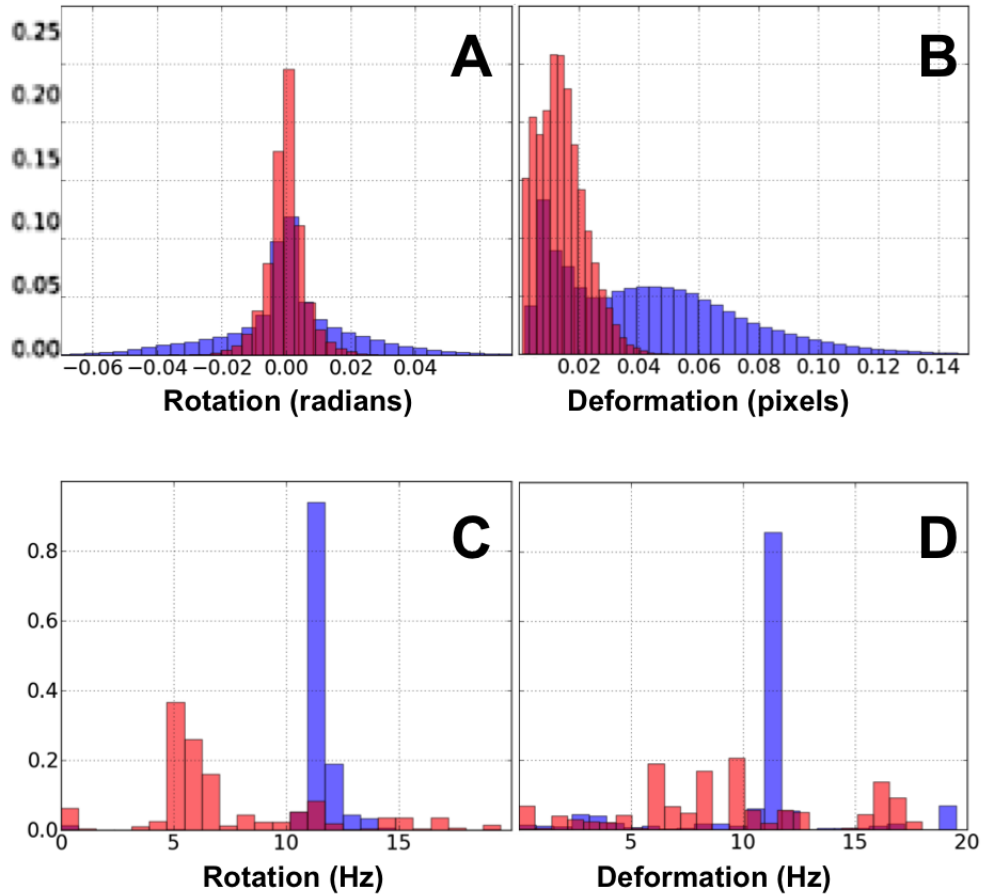


Figure 18: **CM histogram representations.** (A, B) Time domain histograms of ciliary rotation (A) and deformation (B) time-series from normal (blue) and abnormal (red) CM. We project the time-series shown in Fig. 10F-I onto the vertical axis and construct histograms of rotation (B) and deformation (C) magnitudes. (C, D) Frequency domain histograms of ciliary rotation and deformation time-series from normal (blue) and abnormal (red) CM. We use a Fast Fourier Transform (FFT) on the rotation and deformation time-series, compute the dominant frequency at each pixel from the Fourier response, and create histograms of these frequencies for rotation (C) and deformation (D) over all the selected pixels in a ROI.

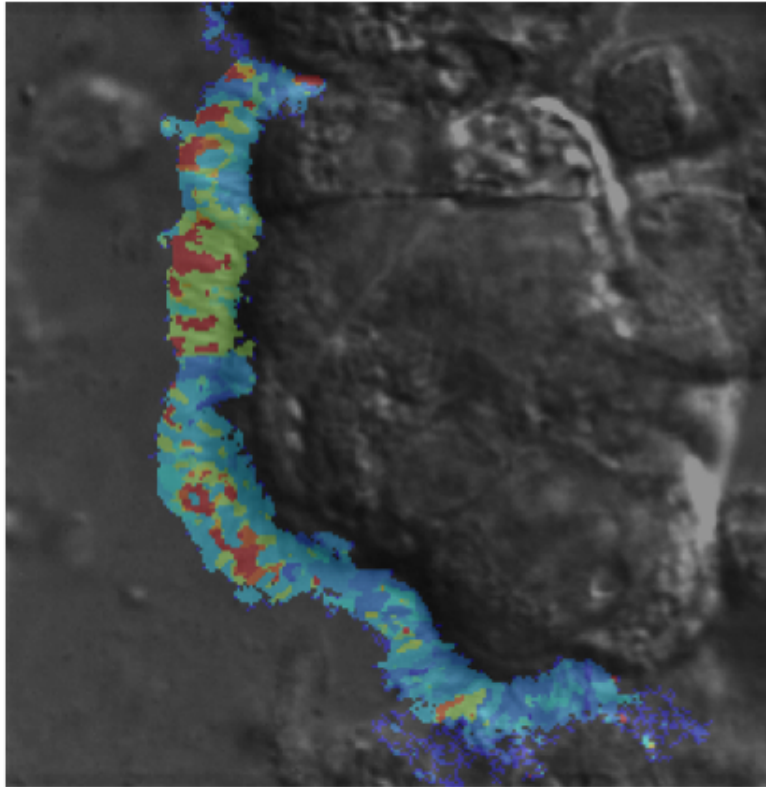


Figure 19: **Pixel selection.** The heatmap overlay indicates the dominant frequencies at each pixel. Light blue indicates low-frequency motion (1-5Hz), where yellow and red indicate higher-frequency motion (5-15Hz). Pixels without color overlays were discarded by the adaptive pruning method.

3.6 CLASSIFICATION OF DIGITAL SIGNATURES

To test our methods, we used two data cohorts (Table 1). The first cohort consisted of videos from 49 individuals (27 healthy controls, 5 PCD controls, and 17 TGA patients) recruited from Children’s Hospital of Pittsburgh (CHP). The second cohort consisted of videos from 31 subjects (27 patients with heterotaxy, 4 PCD controls) recruited from the Children’s National Medical Center (CNMC) reported in the Nakhleh *et al* study [60]. The methods for recruiting subjects and acquiring data are fully elucidated in Materials and Methods, and a visual breakdown of these cohorts (CHP and CNMC) are listed in Table 1 and visualized in Fig. 11. Using these two cohorts, we compared the performance of our framework to the beat pattern calls made by expert reviewers.

We used a Support Vector Machine (SVM) [95], a popular classification algorithm in machine learning, to test our methods. All classifiers operate on the premise of finding a rule, or decision boundary, which most accurately separates data into their correct categories. These boundaries often take the form of lines, or planes, which separate data in Cartesian space. Functionally, each video of CM (or individual regions of CM within a single video) can be considered a point in high-dimensional space; thus, an SVM will attempt to find a plane in that space which most accurately separates the healthy instances from the dyskinetic ones. In this study, these high-dimensional points representing instances of CM take the form of either the AR coefficients, or the four histograms.

3.6.1 Structure of SVM input

For the AR method, we located a pixel nearest the middle of a ROI with a signal at the dominant frequency for the ROI, and expanded a 15×15 box around that pixel, forming a patch. For each frame of the video (truncated at 250 frames), we flattened the pixels in the 15×15 patch into a single 225-length vector (\vec{y}_t in Eq. 3.14). Repeating this process over 250 frames, each patch was contained in a data structure with shape 225×250 . We repeated this process for all ROIs, appending each patch to the end of the previous one. For the CHP dataset with 331 ROIs, this resulted in a $225 \times (331 * 250)$ data structure, or matrix with

dimensions 225×82750 . Performing SVD on this structure yielded the principal components C (Fig. 16A). Having C , we solved for \vec{x} in Eq. 3.14 and subsequently the AR coefficients \mathcal{B} in Eq. 3.15, which we used as the digital signature. The parameter q modulated the dimensionality of the CM subspace C ; therefore, each coefficient B_i was a matrix with dimensions $q \times q$. The parameter d specified the number of AR coefficients $\mathcal{B} = \{B_1, B_2, \dots, B_d\}$. The coefficients B_1, B_2, \dots, B_d were flattened row-wise and concatenated, resulting in a single vector with length q^2d as the digital signature for each ROI. We performed parameter scans over $q \in [2, 20]$, and $d \in [1, 5]$.

For our histogram method, the magnitude histograms were constructed using rotation (RMH) and deformation (DMH) values. The frequency histograms were constructed using the dominant rotation (RFH) and deformation (DFH) frequencies computed at each pixel. These four histograms were combined by comparing them pairwise against the four matching histograms of all other ROIs (Eq. 3.16), forming an $n \times n$ matrix \mathcal{K} , where $n = 331$ for the CHP cohort, and $n = 262$ for the CNMC cohort (Table 1). This matrix, used to initialize the SVM classifier, is specifically referred to as a *kernel matrix*. We found that the values of two parameters had the greatest effect on classification accuracy with the histogram method: the size of Gaussian smoothing of the rotation or deformation time series σ , and the number of bins in the frequency histogram κ . We performed parameter scans over $\sigma \in [0, 8]$ and $\kappa \in [5, 100]$.

3.6.2 Classifier design for CM recognition

We used an instance of the NuSVC SVM in the scikit-learn library [81] with the default, nonlinear radial-basis function (RBF) kernel. We found the RBF kernel significantly outperformed other strategies, such as linear SVMs and ensemble methods including random forests; the performance of linear classifiers was much lower in comparison. SVMs with nonlinear kernels are well-suited for high-dimensional classification problems where data are not plentiful. For our AR strategy, the concatenated coefficients \mathcal{B} constituted the input to the classification algorithm.

For our histogram method, we employed a different strategy. Histograms lend themselves

to direct comparison through the chi-square (χ^2) distance metric. Therefore, rather than concatenate all four histograms into a single vector as with the AR strategy, we instead combined the four histograms from each CM sample into a custom SVM kernel matrix \mathcal{K} [96]. Given a pair of ROIs, $x(i)$ and $x(j)$, we compared the four histograms of each ROI pairwise, computing the χ^2 metric between matching histograms. The metrics were in turn weighted independently using weights α_1 (RFH), α_2 (RMH), β_1 (DFH), and β_2 (DMH), such that $\alpha_1 + \alpha_2 + \beta_1 + \beta_2 = 1$. Multiple weighting schemes were tested to determine if, for example, weighting the χ^2 distance between magnitude histograms more heavily than frequency histograms resulted in an improvement or decline in overall classification accuracy (weights and subsequent classification accuracy shown in Tables 4, 5). The four weighted χ^2 metrics were summed into a final similarity score between ROIs $x(i)$ and $x(j)$:

$$\mathcal{K}_{i,j} = \sum_{w \in \alpha_1, \alpha_2, \beta_1, \beta_2} w \exp(-\mu_w \chi^2(x_w(i), x_w(j))) \quad (3.16)$$

where x_w is a histogram with associated weight $w \in \alpha_1$ (RFH), α_2 (RMH), β_1 (DFH), and β_2 (DMH). Furthermore, μ_w was the average χ^2 distance for histogram type w across all ROIs. This was done for all pairwise combinations of ROIs $x(i)$ and $x(j)$, generating an $n \times n$ kernel matrix \mathcal{K} , where n is the number of ROIs in our data cohort (Table 1). This was used to initialize the SVM for classifying the histograms. Such an initialization was not required for the AR method; the default RBF kernel was used.

3.6.3 Cross-validation and consensus diagnosis

k -fold cross-validation, sometimes referred to as rotation validation, is a verification process for classification algorithms to estimate their performance against unobserved data. The data are split into k blocks, or folds, each containing roughly the same number of ROIs. In the first iteration, the ROIs in the first $k - 1$ folds are used to train the algorithm, meaning that the ROIs in these folds and their “ground truth” labels are provided to the algorithm to learn the quantitative associations. The k^{th} fold is explicitly held out, filling the role of “new” and unobserved data. The k^{th} fold is then used to test the algorithm, whereby the ROIs in that fold are provided to the algorithm without their ground truth labels, and the

algorithm must predict the labels given what it learned in training. The predictions are then compared to the true labels, and a percentage accuracy is computed. The process then moves to the second iteration, whereby the k^{th} fold becomes one of the training folds, and the next fold in line becomes the testing fold. This continues until all folds have been used exactly once as the testing fold.

We treated each ROI as a single datum with its corresponding ground-truth label (0 for healthy, 1 for abnormal). Due to the relatively small size of our data cohorts, we chose to perform 10-fold cross-validation to test our methods, maximizing the size of the training set while also creating more diverse testing subsets.

Since ROIs were treated as single data instances, the algorithm would therefore predict the CM of individual ROIs. However, our goal was to predict CM at the patient level. Furthermore, ROIs from the same patient could potentially receive differing predictions from the classification algorithm. Therefore, to translate the CM prediction for ROIs into a CM prediction for each patient, we performed a *consensus diagnosis*. We first grouped ROI predictions together according to the patients from which they originated; that is, all the predictions for ROIs originating from patient p would be collected. If the majority of the CM predictions on the ROIs for patient p were abnormal, then the patient-level prediction for patient p would also be abnormal (Fig. 20). Consensus diagnosis was performed with each iteration of cross-validation, and the accuracy reported was computed from consensus diagnosis.

3.6.4 Results of CM classification

Both cohorts were classified independently of each other; no data from one cohort was used when classifying CM in another cohort. For the CHP cohort, the histogram method achieved an optimal classification accuracy of 93.8%. Classification performed using AR models achieved an accuracy of 88.6% with rotation and 86.4% with deformation. For the CNMC cohort, we obtained an optimal accuracy of 86.7% with the histogram method. The AR models applied to this dataset yielded an accuracy of 83.3% using rotation and 70.0% with deformation. PCD was the most accurately identified motion abnormality. In the CHP

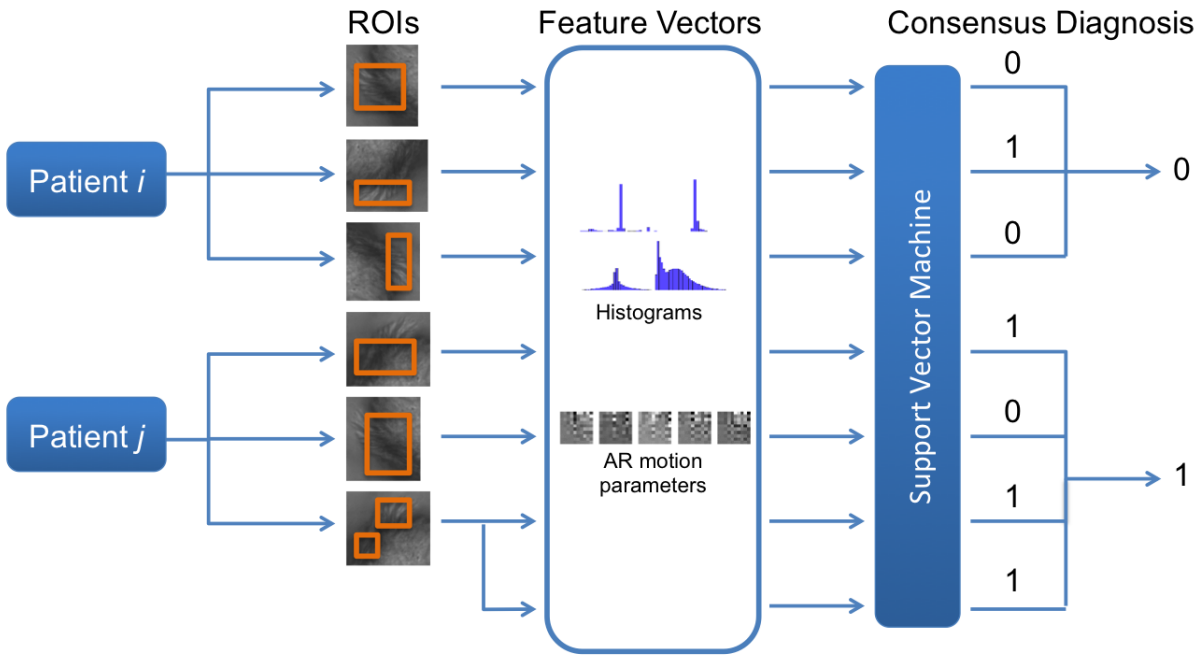


Figure 20: **Classification pipeline.** Patient data, in the form of a handful of ROIs, is classified as normal (0) or abnormal (1) based on the method (histograms or AR). A majority vote, or “consensus diagnosis,” is performed using the ROI classifications for a single patient to predict the CM of the patient. All results are reported as predictions for each patient.

cohort, it was correctly classified as abnormal 93.5% of the time; in the CNMC cohort, it was correctly classified 100% of the time. These results, as well as their sensitivity, specificity, and comparison to baseline CM analysis methods such as CBF, can be found in Table 2.

In all cases, rotation most accurately differentiated normal versus abnormal CM: the AR model using rotation data outperformed the model using deformation (Table 3), and in all the highest-accuracy histogram classifications, rotation (magnitudes in particular) was used (Tables 4,5). Despite the subjectivity in manual identification of ciliary beat pattern, clinical studies consistently describe abnormal motion as having reduced beat amplitude, stiff beat pattern, failure to bend along the length of the ciliary shaft, static cilia, or a flicking or twitching motion [70,71]. Rotation in particular is affected by the stiffness that is often observed in abnormal CM; this specific phenotype could account for the performance difference between rotation and deformation.

Both rotation and deformation were superior features to raw pixel intensities; as the histogram method did not rely on PCA, we could compare the use of raw pixel intensities in this method directly to elemental components (Table 2). The magnitude histograms (Figs. 18A,B) depict a broad distribution of rotation and deformation values for normal motion, which contrasts with the much more narrow distributions of rotation and deformation for abnormal motion. Both rotation and deformation frequency histograms (Figs. 18C,D) depict normal CM as having a clear dominant frequency. This contrasts again with abnormal CM, in which there is noticeable power at multiple frequencies. Collectively, this forms an intuitive picture of normal CM as having a relatively uniform beat frequency but which rotates or deforms with a relatively wide variance, suggesting much greater freedom of movement. This also underscores the importance of properly quantitating motion heterogeneity, as this was an important property for differentiating CM.

For our AR method, the differences in the structure of the AR coefficients further supports the interpretation of the histogram results: the coefficients associated with abnormal CM (Fig. 21A, bottom row) are largely uniform in the lower triangular half. By comparison, the coefficients for normal CM (Fig. 21A, top row) are significantly more complex, particularly in the higher-order coefficients. Intuitively, this equates to the former capturing significantly less complex motion than the latter. This is precisely what we see in Fig. 21B:

Table 2: Classification results.

Method	Dataset	Accuracy	Sensitivity	Specificity
Proposed Methods				
Histogram	CHP	93.88%	0.9524	0.9286
AR (rotation)	CHP	88.64%	0.8000	0.9583
Histogram	CMNC	86.67%	0.9167	0.8333
AR (rotation)	CMNC	83.33%	0.8333	0.8333
Baseline Methods				
Histogram (raw intensities)	CHP	72.73%	0.6316	0.8000
Ciliary beat frequency	CHP	52.27%	0.3500	0.5833

Table 3: AR results.

q	d	Dataset	Accuracy	Sensitivity	Specificity
10	5	CHP (rotation)	88.64% ¹	0.8000	0.9583
10	1	CHP (deformation)	86.36%	0.7619	0.9565
19	1	CNMC (rotation)	83.33% ¹	0.8333	0.8333
1	2	CNMC (deformation)	70.00%	0.5909	1.0000

¹ See Table 2.

Table 4: Histogram results using CHP dataset.

α_1 (RFH)	α_2 (RMH)	β_1 (DFH)	β_2 (DMH)	σ	κ	Accuracy	Sensitivity	Specificity
1.0	0.0	0.0	0.0	2.4	20	72.73%	0.6667	0.7586
0.0	1.0	0.0	0.0	2.4	5	93.18%	0.8500	1.000
0.0	0.0	1.0	0.0	0.0	15	75.00%	0.6364	0.8636
0.0	0.0	0.0	1.0	0.0	5	88.64%	0.8333	0.9231
0.5	0.5	0.0	0.0	7.2	15	93.18%	0.8500	1.0000
0.5	0.0	0.5	0.0	0.0	15	77.27%	0.7059	0.8148
0.5	0.0	0.0	0.5	6.8	15	90.91%	0.8421	0.9600
0.0	0.5	0.5	0.0	0.4	5	90.91%	0.8421	0.9600
0.0	0.5	0.0	0.5	0.8	5	90.91%	0.8421	0.9600
0.0	0.0	0.5	0.5	0.0	20	90.91%	0.8421	0.9600
0.33	0.34	0.33	0.0	0.0	15	93.18%	0.8889	0.9615
0.33	0.34	0.0	0.33	0.8	5	90.91%	0.8421	0.9600
0.33	0.0	0.33	0.34	0.0	15	90.91%	0.8421	0.9600
0.0	0.34	0.33	0.33	0.8	5	90.91%	0.8421	0.9600
0.25	0.25	0.25	0.25	3.6	15	93.88% ¹	0.9524	0.9286

¹ See Table 2.

Table 5: Histogram results using CNMC dataset.

α_1 (RFH)	α_2 (RMH)	β_1 (DFH)	β_2 (DMH)	σ	κ	Accuracy	Sensitivity	Specificity
1.0	0.0	0.0	0.0	0.4	20	73.33%	0.8000	0.7000
0.0	1.0	0.0	0.0	0.0	5	83.33%	0.8462	0.8235
0.0	0.0	1.0	0.0	0.0	10	73.33%	0.7143	0.7500
0.0	0.0	0.0	1.0	0.0	5	83.33%	0.8462	0.8235
0.5	0.5	0.0	0.0	6.4	10	80.00%	0.8333	0.7778
0.5	0.0	0.5	0.0	0.8	10	73.33%	0.7500	0.7222
0.5	0.0	0.0	0.5	1.6	15	76.67%	0.8182	0.7368
0.0	0.5	0.5	0.0	5.2	5	86.67% ¹	0.9167	0.8333
0.0	0.5	0.0	0.5	0.0	5	83.33%	0.8462	0.8235
0.0	0.0	0.5	0.5	0.0	10	83.33%	0.8462	0.8235
0.33	0.34	0.33	0.0	7.2	10	80.00%	0.8333	0.7778
0.33	0.34	0.0	0.33	6.8	5	83.33%	0.8462	0.8235
0.33	0.0	0.33	0.34	0.8	5	76.67%	0.8182	0.7368
0.0	0.34	0.33	0.33	0.8	5	83.33%	0.8462	0.8235
0.25	0.25	0.25	0.25	7.2	15	83.33%	0.8462	0.8235

¹ See Table 2.

using the first three principal components to observe the motion of cilia in three dimensions, we observe a wider range of motion for normal CM (blue) than abnormal CM (red). For visual clarity, the distributions of x , y , and z values in each of the three dimensions of Fig. 21B are plotted as histograms. There is a large amount of overlap in the movement of the CM in the first dimension (Fig. 21C), but even with two or three dimensions (Fig. 21D,E) we observe a noticeable divergence in trajectories separating normal from abnormal: the former has much more freedom of movement than the latter.

The few misclassifications made by our framework, particularly on the CNMC set, could be attributed to poor sample and video quality. Shifts in focal plane and other motion artifacts were particularly problematic, resulting in deleterious effects on optical flow computation. The consensus diagnosis step (Fig. 20) enhanced robustness to noise; while some ROIs could be misclassified, the framework would still predict the CM of the *patient* correctly provided enough ROIs were chosen from videos of sufficient quality to represent that patient. We found that, beyond a minimum number of roughly three ROIs per patient, the overall quality of the ROIs (and, by proxy, the video samples) was much more important than quantity of ROIs. In the CHP cohort (Fig. 22A,B), there is a slight correlation between number of ROIs per patient, and subsequent average classification accuracy for that patient. In the CNMC cohort, however, there is no noticeable correlation (Fig. 22C,D), suggesting that ROI quality is more important than quantity.

One weakness pertains to the optical flow computations. Specifically, any defects in the pixel intensities of the original grayscale videos (e.g., recording artifacts, lack of contrast) will persist in some form through the optical flow and elemental components. Videos with a significant amount of particulate matter and recording artifacts were most consistently misclassified, suggesting that even while the optical flow computations involve smoothing and filtering input, the artifacts still persist when digital signatures are computed, ultimately confusing the framework. Careful and deliberate ROI selection can minimize this issue, but even more effective is the use of high-quality biopsies and videos.

To further elucidate the reasons behind systemic mistakes made by our framework, we examined several videos that were consistently misclassified by our algorithm. We note amongst both data cohorts, the PCD controls were consistently identified as exhibiting abnormal CM

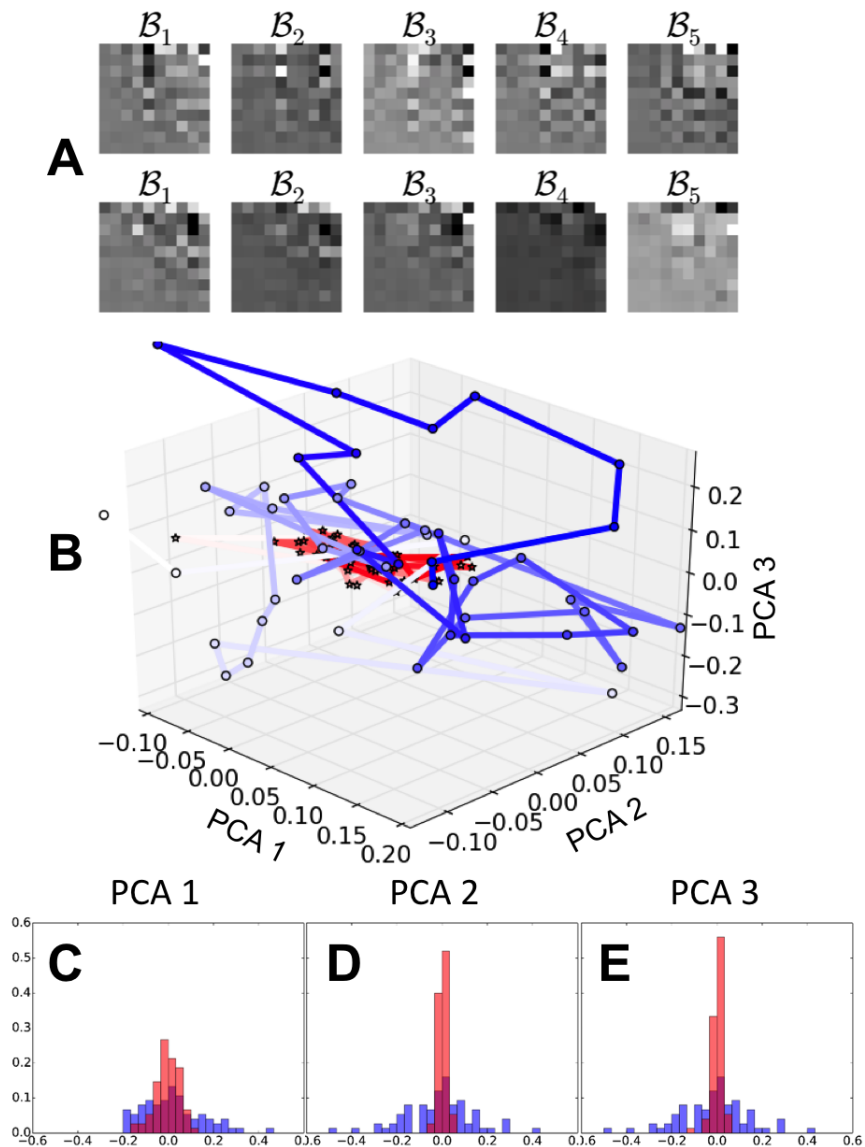


Figure 21: **CM AR model representations.** (A) Coefficients of the AR model for normal (top) and abnormal (bottom) CM, represented as heatmaps. For this system, $d = 5$ as indicated by the number of coefficients for each ROI, and $q = 10$, specifying the square dimensions of each coefficient matrix. (B) CM is visualized in $q = 3$ CM subspace of the AR model for normal (blue) and abnormal (red) CM. This motion is governed by the AR coefficients. (C, D, E) Histograms show the distributions of values taken by normal (blue) and abnormal (red) AR motion in these dimensions, as depicted in (B).

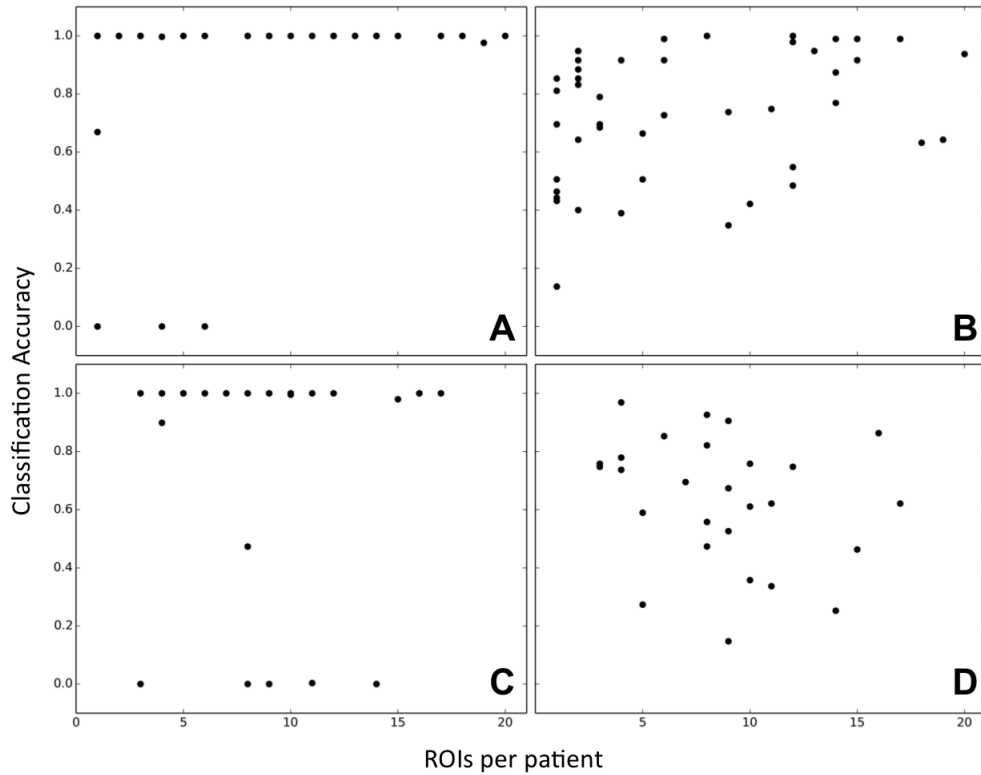


Figure 22: **Classification accuracy as a function of ROIs per patient.** While the overall classification accuracy for each method is specified in Table 2, these plots provide some intuition into how the number of ROIs per individual affected how accurately the CM for that individual was identified. 1.0 indicates the CM of that particular individual was always identified correctly; conversely, 0.0 indicates our framework consistently misclassified the CM of that individual. **(A)** Classification accuracy for each patient in the CHP cohort using the histogram method. **(B)** Classification accuracy for each patient in the CHP cohort using AR rotation models. **(C)** Classification accuracy for each patient in the CNMC cohort using the histogram method. **(D)** Classification accuracy for each patient in the CNMC cohort using AR rotation models.

by both quantitative methods; these were among the CM instances our framework classified with the greatest confidence. The histogram method, especially in the CHP cohort, was extremely confident in all the predictions it made, whether those predictions were correct or not. This is alluded to in Fig. 22, where the accuracy for each patient tends to be either 0% or 100% with few in between. The patients that were exclusively misclassified almost always had associated videos with recording artifacts such as a shaking stage or camera, or changes in the plane of focus. These artifacts introduced a significant amount of spurious motion, which when converted to rotation and deformation took the form of highly complex AR coefficients (Fig. 21A, top), wide magnitude histograms (Fig. 18A-B, blue), and narrow frequency histograms (Fig. 18C-D, blue). These closely mimicked the digital signatures generated by normal CM; this effect is particularly prevalent in the CNMC dataset where the data are noisier, explaining the lower specificity values (Table 2). However, aside from videos with recording artifacts, a small number of patients were consistently misclassified in both directions (healthy as having abnormal CM, and vice versa), and closer inspection revealed that these patients were potentially assessed incorrectly when establishing the ground truth. Our ground truth method relied on the review of multiple experts, in which a majority vote among the experts established the ground truth CM for each patient. In examining the patients that were 1) consistently misclassified, and 2) did *not* have videos with substantial recording artifacts, these were patients for whom the majority vote in establishing ground truth CM was not unanimous. This highlights the primary benefit of this framework: eliminating reviewer subjectivity and uncertainty.

3.7 UNSUPERVISED DISCOVERY OF NOVEL MOTION PHENOTYPES

Our classification framework, while recapitulating expert review to a high degree of accuracy, is fundamentally limited in its expressive power. For instance, while it can identify motion as either normal or abnormal, it cannot answer the question as to whether or not there exist other more subtle ciliary motion phenotypes, or if the ground truth as established by manual review is even correct. Addressing these questions requires a shift to unsupervised

approaches. Our goal is to create a quantitative library of ciliary motion phenotypes, building a graph of these phenotypes linked by their similarities to one another.

3.7.1 Automated region selection

One of the biggest drawbacks in our classification framework was that it required manual selection of ROIs to focus the analysis. While this functioned perfectly well, particularly for establishing a robust baseline, this inherently limited the degree of automation and objectivity in the framework. As part of our efforts to develop a fully automated, high-throughput, and objective framework for evaluating ciliary motion, we first developed a completely automated method for selecting patches for analysis.

Building on our findings from our classification framework, we used *rotation data only* in determining patches of interest, and for analysis using the AR framework mentioned previously. In classification, we found rotation was the most predictive property of CM and the most robust to sources of noise. Additionally, its property of orientation-invariance made it the perfect candidate for use in unsupervised discovery of novel motion patterns.

For each video, we created a two-dimensional map of rotation amplitudes at each pixel location, and ignored all locations whose amplitudes fell below some threshold of interest. The remaining pixel locations showed a significant amount of rotation relative to the rest of the video, theoretically identifying objects in motion. However, we found this was not sufficient for isolating cilia; particulate matter and even background medium subject to the inertia generated by beating cilia tended to have high rotation amplitudes.

To address this, we used texture filters on the regions we had so far identified. For regions that were monotonic or otherwise not textured, the output of the filter was very low. However, in regions with a significant amount of texture, i.e., regions with cilia, the output was very high. This allowed us to differentiate between cilia and background motion with a high degree of accuracy. Using the outputs of the filter to guide our selection process, we identified pixels of interest and expanded 15×15 patches around them, ensuring they did not overlap. This process is shown in Fig. 23.

Below is the full procedure for processing the videos and identifying patches.

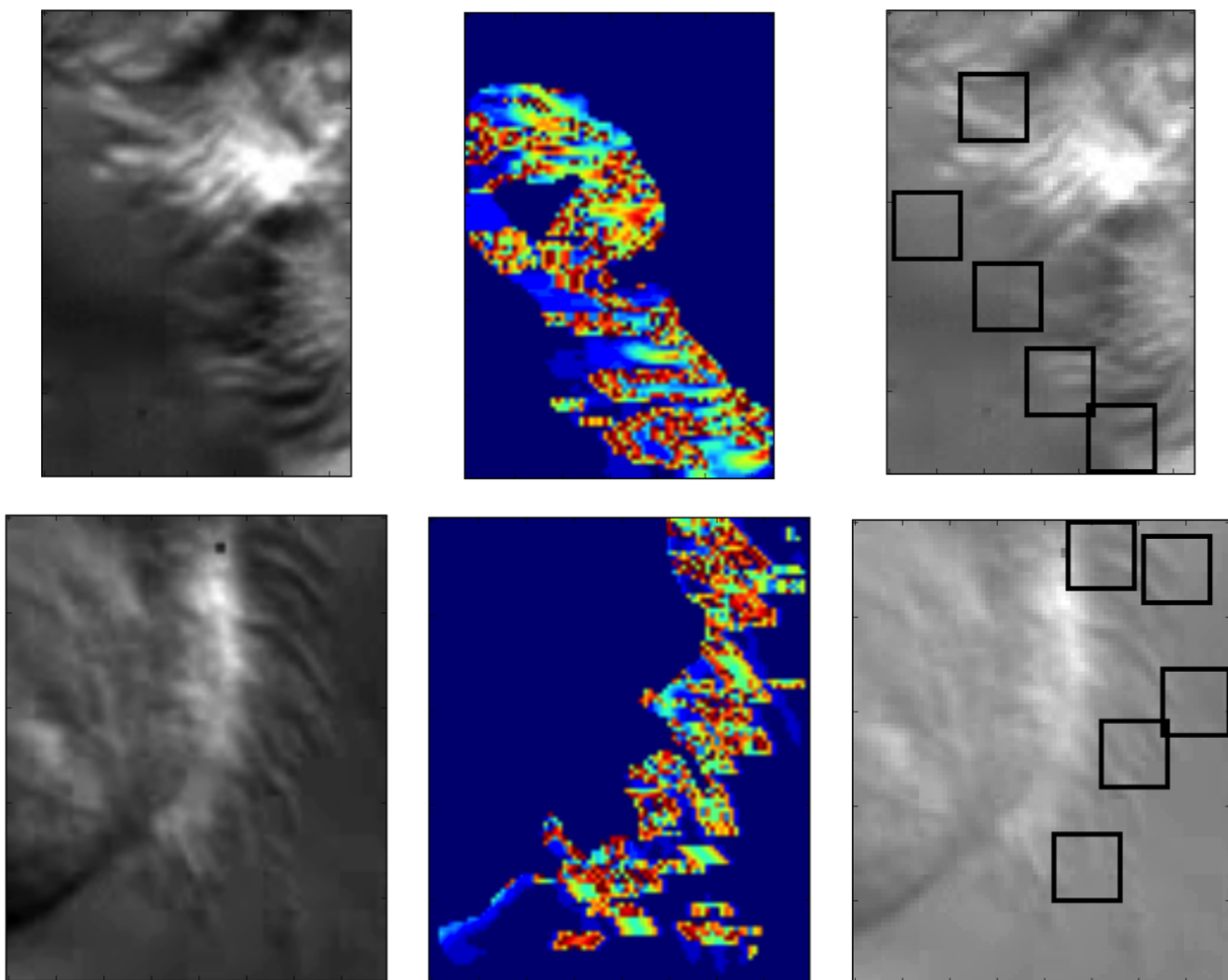


Figure 23: **Automated patch selection.** Starting with the initial video (left column), we identified pixels of interest through a combination of rotation amplitude at each pixel and a texture filter, with the larger values of the output corresponding to regions of cilia (middle column). Using these positions as “seeds,” we expanded boxes around each to identify patches to be used in analysis (right column).

1. Compute rotation using the optical flow of the video (see the previous sections).
2. Raster scan the rotation video, resulting in each row identifying a pixel, and each column corresponding to a frame. Formally, we define the video $V \in \mathcal{R}^{n \times f}$, where n is the number of pixels, and f is the number of frames in the video.
3. Compute the amplitude at each pixel i , computing the difference in its min and max across all frames.

$$\vec{a}_i = \max(V_i) - \min(V_i), \forall i \in [1, n]$$

4. Run a two-dimensional median filter over the amplitudes to smooth out noise (we used a kernel of size 25). Clamp the vector of magnitudes \vec{a} at the 80-percentile, discarding all pixels whose amplitude falls below.
5. Run a texture filter over the remaining pixels to differentiate background from cilia. In our case, we used a Gaussian gradient magnitude filter with $\sigma = 2.5$.
6. Sort the gradient filter values in descending order. Starting from the largest value, expand a patch around the “seed” pixel. Repeat until the desired number of patches have been selected, or no more patches can fit without overlapping.

Once the patches were identified and extracted, we computed their AR parameters using the methods stated previously. We were able to identify more than 3,500 patches across all videos in our two data cohorts. Each patch was 15×15 in height and width, and 250 frames in length. As per the process for determining the low-dimensional AR representation of the CM, we raster-scanned all the patches into 225×250 matrices and stacked them horizontally, resulting in a representative CM structure space with dimensions $225 \times 900, 250$, a dense matrix with over 200 million floating-point elements. While this was not outside the scope of conventional linear solvers, it pushed their limits noticeably and nonetheless proved to be the bottleneck in this project.

3.7.2 Spectral clustering of AR parameters

After computing the AR parameters $\mathcal{B} = \{B_1, B_2, \dots, B_d\}$ for each patch, we built a pairwise affinity matrix A and subsequent normalized graph Laplacian L in accordance with the spectral clustering machinery outlined in Chapter 2. However, rather than using the traditional

RBF kernel for pairwise affinities, we used an approach that is common for discriminating between parameters of linear dynamics systems: Martin distance [93]. It is based on the subspace angles between two systems, and while we have empirical evidence that all CM exists in the same subspace (Fig. 17), we found the Martin distance metric significantly outperformed RBF in terms of identifying potential CM subtypes.

The Martin distance is defined over both the subspace of the system C and the motion parameters B . Specifically, we have

$$\mathcal{B}^T \mathcal{P} \mathcal{B} = -\mathcal{C}^T \mathcal{C}, \quad (3.17)$$

where, for two patches v_i and v_j and some number of subspace dimensions q ,

$$\mathcal{P} = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \in \mathcal{R}^{2q \times 2q},$$

$$\mathcal{B} = \begin{bmatrix} B_{v_i} & 0 \\ 0 & B_{v_j} \end{bmatrix} \in \mathcal{R}^{2q \times 2q},$$

$$\mathcal{C} = \begin{bmatrix} C & C \end{bmatrix} \in \mathcal{R}^{225 \times 2q}.$$

Once we have found \mathcal{P} by solving the above Lyapunov equations (Eq. 3.17), we construct a symmetric matrix of the constituent components of \mathcal{P} and perform an eigendecomposition. Specifically, each eigenvalue λ_i of this matrix is the cosine of the subspace angle θ_i .

$$\cos^2 \theta_k = k^{\text{th}} \text{eigenvalue}(P_{11}^{-1} P_{12} P_{22}^{-1} P_{21})$$

Finally, we can use the eigenvalues of this matrix to compute the Martin distance d_M between patches v_i and v_j :

$$d_M(v_i, v_j)^2 = -\ln \prod_{k=1}^q \cos^2 \theta_k. \quad (3.18)$$

Throughout the literature, when dynamic textures as represented by linear dynamics systems are compared using the Martin distance, \mathcal{C} is often composed of two distinct subspaces, C_1 and C_2 . However, as we mentioned previously, all CM is hypothesized to occupy the same

subspace C , thus \mathcal{C} is therefore composed of the same subspace horizontally stacked. Furthermore, due to time constraints, we could only use this formulation of the Martin distance when systems were of first order, or $d = 1$. Future work will generalize the above discrete Lyapunov equations to work for higher order AR processes.

The resulting pairwise Martin distance matrix for all patches resembled Fig. 24 on the left. After sparsifying this matrix (eliminating all edges with weights under a certain threshold) and performing spectral biclustering [97], we found very good alignment with four distinct clusters of motion phenotypes that provided interesting results when annotated with existing ground-truth information.

We found the four clusters in the right panel of Fig. 24 correlated well with the ground-truth identification from our expert collaborators. Specifically, each patient was assigned a number 1-4, 1 indicating completely normal and 4 indicating completely abnormal. Labels 2 and 3 built some uncertainty into this scale, and this clustering recapitulated those readings. Cluster C correlated very well with patients rated 1, or completely normal; most of the patients identified this way were constituents of our health controls in cohort 1 (Fig. 11). By contrast, Cluster B was almost exclusively PCD patients; all nine PCD controls from both data cohorts were recognized in this cluster, in addition to a few of the heterotaxy patients. Clusters A and D showed combinations of both, with Cluster A showing more normal phenotypes and Cluster D more abnormal phenotypes. Some of the patients we consistently misclassified in our first approach were placed in either Cluster A or D, signifying a somewhat more ambiguous motion phenotype.

This spectrum of motion phenotypes is more fully visualized in Fig. 25. Here, after computing the graph Laplacian L from the affinity matrix of Martin distances, we used the leading eigenvectors of L to embed each patch in a low-dimensional subspace spanned by the principal components of L .

Visually, we observe what we intuited from the biclustering of the affinity matrix: there exists a *spectrum* of ciliary motion phenotypes, ranging from abnormal (Fig. 25, red) to normal (Fig. 25, blue). While we can and have achieved a high level of accuracy in distinguishing normal from abnormal ciliary motion, it is an oversimplification to group all phenotypes into this binary system.

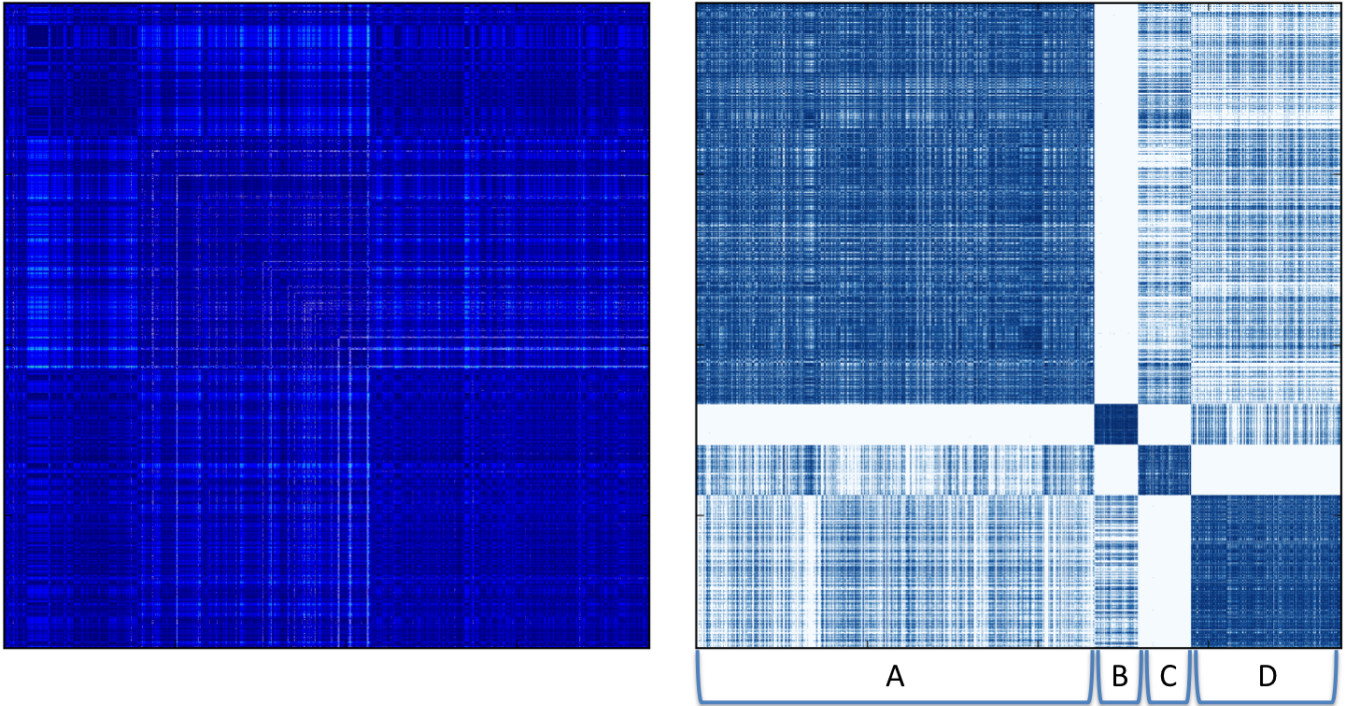


Figure 24: **CM subtypes.** The dense pairwise Martin distance matrix is shown on the left. On the right is the sparsified and biclustered matrix, showing four distinct clusters of CM phenotypes, identified as such with A, B, C, and D.

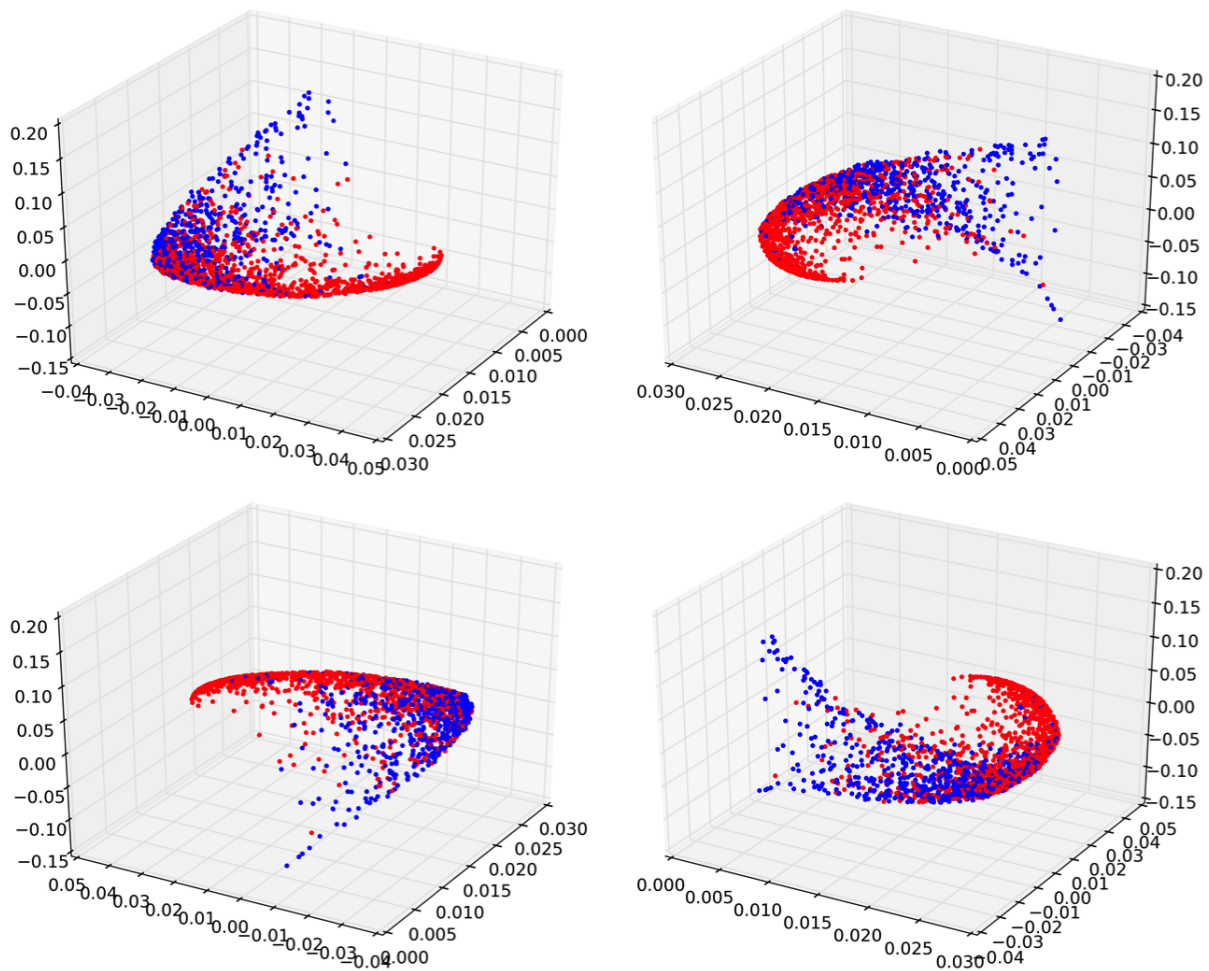


Figure 25: **3D space of CM subtypes.** Each patch is represented by a dot. Red indicates the patch came from a patient whose diagnosis was abnormal; blue denotes patches from healthy patients. Each panel shows the same data, with the viewing angle rotated 90 degrees.

3.7.3 Large-scale analysis

The patch selection strategy enumerated in the previous section is the final step in a fully autonomous, high-throughput analysis framework. Coupled with a web front-end (Fig. 12), clinicians and researchers need only upload video data. This opens the door for very large-scale analysis of ciliary motion phenotypes, in particular for latent pattern discovery.

There are two operations that potentially require new techniques to allow for such large-scale analysis. The first is the PCA step in deriving the ciliary motion subspace component of the AR process (C in Eq. 3.14). This subspace is computed from all available data. The videos we used in these studies were 200 fps, and we typically truncated each video at 250 frames, thereby capturing a little over 1 second of ciliary motion. In our unsupervised pattern discovery, we sampled 3,600 patches. This equates to computing the principal components of a data structure with nearly 1 million columns. While still feasible using traditional methods, this represents only a small sampling of available video data. Even with the 291 videos in both datasets used here (Table 1), sampling 100 patches from each video would result in a dense matrix with over 72 million columns, well beyond the capabilities of traditional linear solvers.

However, we found evidence that a full principal component analysis of all available video data may be unnecessary for deriving a representative subspace (Fig. 17). In lieu of such a computationally expensive step, a sophisticated sampling method could be used to drastically reduce the amount of data needed to compute a ciliary motion subspace while still offering a robust low-dimensional representation of the manifold occupied by ciliary motion patterns. Thus, this particular bottleneck is potentially avoidable.

The second operation that absolutely necessitates new analysis techniques is the identification of latent motion patterns. We demonstrated the use of spectral biclustering (Fig. 24), a technique that examines the spectral embeddings of the AR parameters for each patch to discover clusters with similar motion phenotypes. Determining this spectral embedding an extremely expensive process, and for more than roughly 10,000 patches, would require a distributed implementation. We discuss such an implementation in Chapter 5.

3.8 CONCLUSIONS

The innovative framework described here can be deployed in a clinical setting, helping establish objective standards for the diagnosis of CM defects and facilitating cross-institutional collaborations in multi-center trials through quantitative analyses of ciliary biopsies. Our framework improves on the current methods for ciliary beat pattern analysis by developing quantifiable digital signatures of the CM and replicating expert CM assessment to a high level of accuracy.

The few misclassifications made by our framework are elucidated further with our unsupervised clustering analysis for discovering novel ciliary motion phenotypes. We have compelling quantitative evidence grouping ciliary motion into one of two phenotypes, normal or abnormal, is an oversimplification that ignores a smoother spectrum of phenotypes. In particular, we have painted a picture with no fewer than four distinct motion phenotypes, with the likely possibility of more given additional data.

Future work on this project will include the acquisition of additional data; ideally, we will collect and generate roughly five to 10 times the number of patches extracted in this study, putting the SVD step of generating AR parameters well out of reach of conventional linear solvers. While Martin distance proved extremely useful for comparing AR parameters, it was limited to comparing only one parameter from each patch. We will generalize this process to handle an arbitrary number of parameters per patch, particularly given our finding in the classification framework that multiple parameters provide a higher classification accuracy, therefore more readily recognizing the underlying CM and theoretically providing a more accurate quantitative description.

While the quantity of data used here does not qualify as large enough to necessitate more scalable analysis techniques, some slight modifications were necessary to guarantee a reasonable runtime. Furthermore, while these initial results were compelling, we were left to conclude that more data was required to tease out additional potential ciliary motion phenotypes. We pushed our conventional computational machinery to its limits in conducting the spectral analysis required to uncover these insights; in the next chapter, we discuss a project for which more scalable techniques were no longer a luxury, but a necessity.

4.0 LEARNING PERCEPTUAL OLFACTORY DIMENSIONS

4.1 INTRODUCTION

For most sensory modalities, basic stimulus dimensions are mirrored in the organization and topography of neural circuits, and in turn define important perceptual axes [98,99]. In several well-known cases, simple heuristics can describe this mapping from *stimulus space* to *perceptual space*, with the correspondence between the wavelength and color of light providing the prototypical example [99,100]. In the case of olfaction, the details of this mapping are much less clear, being complicated principally by the high dimensionality of odor stimuli (odorants). Whereas color is effectively a readout of a single, continuous stimulus dimension by a handful of narrowly tuned receptor types [101,102], odor quality is determined by the combinatorial activation of many dozens to hundreds of broadly tuned receptor types [103–107]. Simply put, there are too many ways for molecules to vary for any single physicochemical feature to uniquely determine odor quality [108,109]. Rather, the space of molecules is high-dimensional, discrete, and intermittently occupied, and olfaction must employ a strategy to match.

The goal of this case study was to gain insight into this strategy by establishing a correspondence between the physicochemical space of odorants, and the space of olfactory percepts. More specifically, we sought to develop a metric for odorant comparison that both recapitulates perceptual judgements of pairwise similarity, and provides a basis for accurate odor classification. Several recent studies have reported important successes on the first front, using principal component analysis (PCA) to identify “molecular compactness” as a candidate stimulus dimension [110–113]. Here, we sought to build on this foundation by using machine-learning (ML) based strategies for discovering an odorant metric. The

key potential advantage of ML is that it explicitly folds in perceptual similarity data to guide and constrain the discovery of the metric. In other words, ML seeks the combination of physicochemical features (and their relative couplings) that best explain similarity judgments, whereas correlative methods are constrained by the assumption that olfactory perception latches onto the most salient structure in the world. Arguments against the latter type of model include the well-known observation that compounds deemed similar by the obvious criteria (chain length, functional group, etc.) need not smell similar [106, 107], as well as the fact that judgments of odor similarity are highly species specific, and driven by organisms’ unique ecological needs [114, 115].

Our contributions in this study are threefold. First, using methods from Xing *et al* [116], we derive a novel metric for computing pairwise odorant similarity. This metric is generalizable to the form of similarity information, and can readily accommodate most odor discrimination data: it can be obtained from human psychophysical experiments [117], or as in this study, from computational similarity predictions, such as Euclidean distance [118]. We build the metric by enumerating all combinatorial pairs of odorants, and separating them into nonoverlapping sets of similar and dissimilar odors, defined using results from our previous work in Castro *et al* [5]. The metric reveals a low-dimensional embedding of the odorants in the physiochemical space, providing rules for mapping physiochemical properties to odor percepts.

Second, we use the metric in conjunction with statistical machine learning techniques to implement a classification scheme for scalable and automated percept prediction. By exploiting the structure of the underlying manifold as defined by the similarity metric, we achieve an unprecedented level of accuracy in classifying odorants into one of 10 discrete perceptual categories. We demonstrate how the metric can be used to achieve state-of-the-art odorant categorization with a relatively small number of odorant dimensions.

Third, we exploit the standalone formulation of the metric to investigate semi-supervised methods of predicting the perceptual categories of novel, unobserved odorants. Using the original metric as a kernel, and employing a distributed computational framework, we can computationally predict the perceptual categories of odorants from databases such as PubChem using only their physiochemical descriptors. Furthermore, odorants with perceptual

labels can be used to update our learned metric to provide more accurate predictions.

4.2 DRAVNIIEKS ODOR PROFILE AND PHYSIOCHEMICAL DESCRIPTORS

For this study, we use 141 of the 144 odorants defined in the Dravnieks odor database [119, 120]. We used Molecular Modeling Pro (MMP) to compute the physiochemical properties from [111]; three of the compounds were omitted from this study, as these compounds (either at higher or lower concentrations) cannot be represented in our analysis. Of the list of 126 descriptors used by Koulakov *et al* [111], we chose the 78 properties that could be computed with MMP. These properties included atom counts, molecular mass, size, hydrophobicity, solubility, QSAR properties, dipole moments and charges, connectivity indices, thermodynamics, and properties of polymer and surfactants. The complete list of these 78 properties is included in the Appendix; the relative distributions these properties take are shown in Fig. 26. The heatmap of odorants and their physiochemical properties are shown in Fig. 27.

We normalized the descriptors according to the methods described in [111]. In particular, for properties that took values ≤ 0 , we used the z-score ($z = \frac{x - \mu_x}{\sigma_x}$) [110]. Properties that took values > 0 often had log-normal distributions, precluding the use of z-scores. For these properties, if the standard deviation of the logarithm was ≥ 1 , we used the z-score of the logarithm. For properties with a standard deviation < 1 and for those with some negative values, we used the direct z-score.

4.2.1 Nonnegative matrix factorization to determine ground-truth odor percepts

Our previous work [5] demonstrated a computational method for elucidating a low-dimensional representation of the odor perceptual space by decomposing the Dravnieks odor profile database [119, 120] using nonnegative matrix factorization (NMF) [121–124]. NMF and PCA are similar in that both methods attempt to capture the low-dimensional structure of

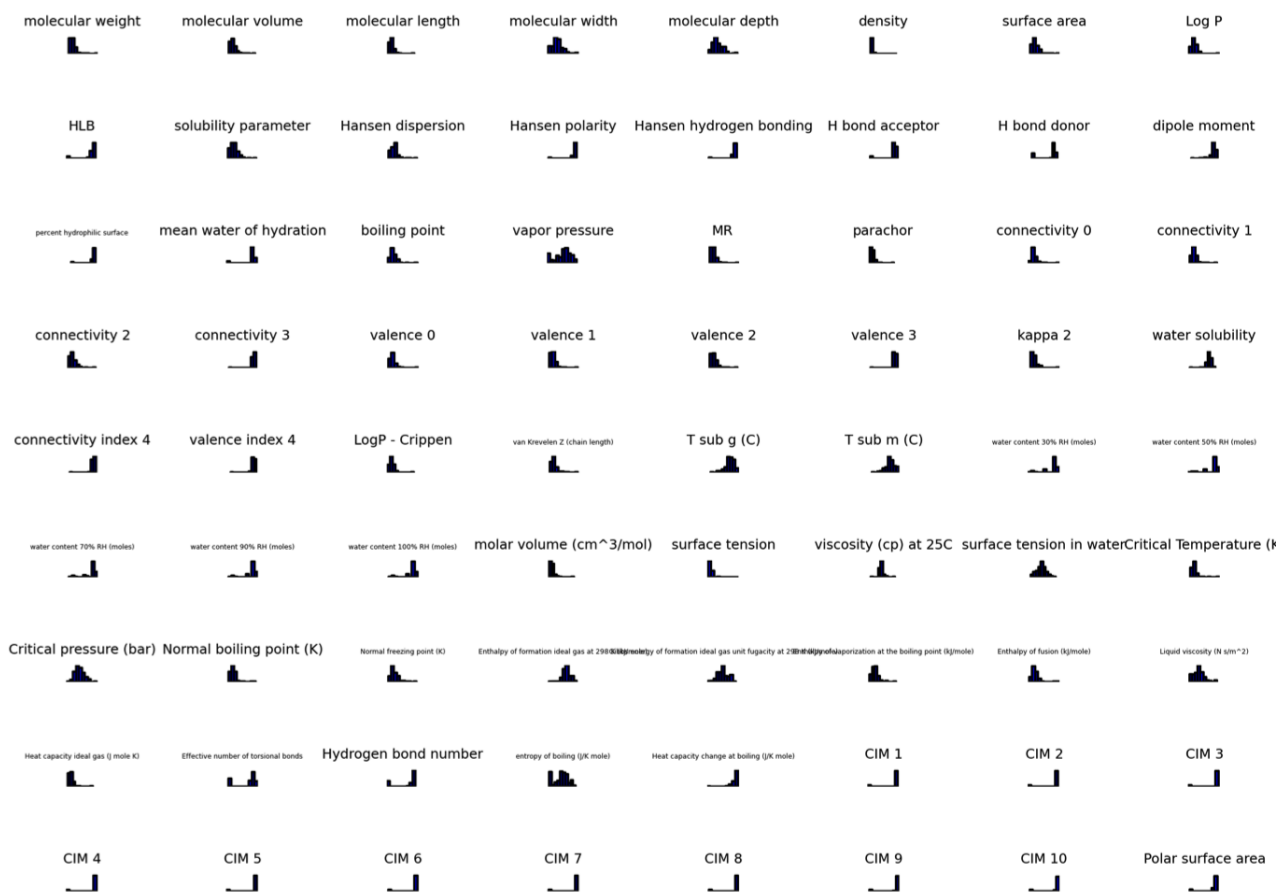


Figure 26: Histograms of the physiochemical properties used across all 141 Dravnieks odorants used this study.

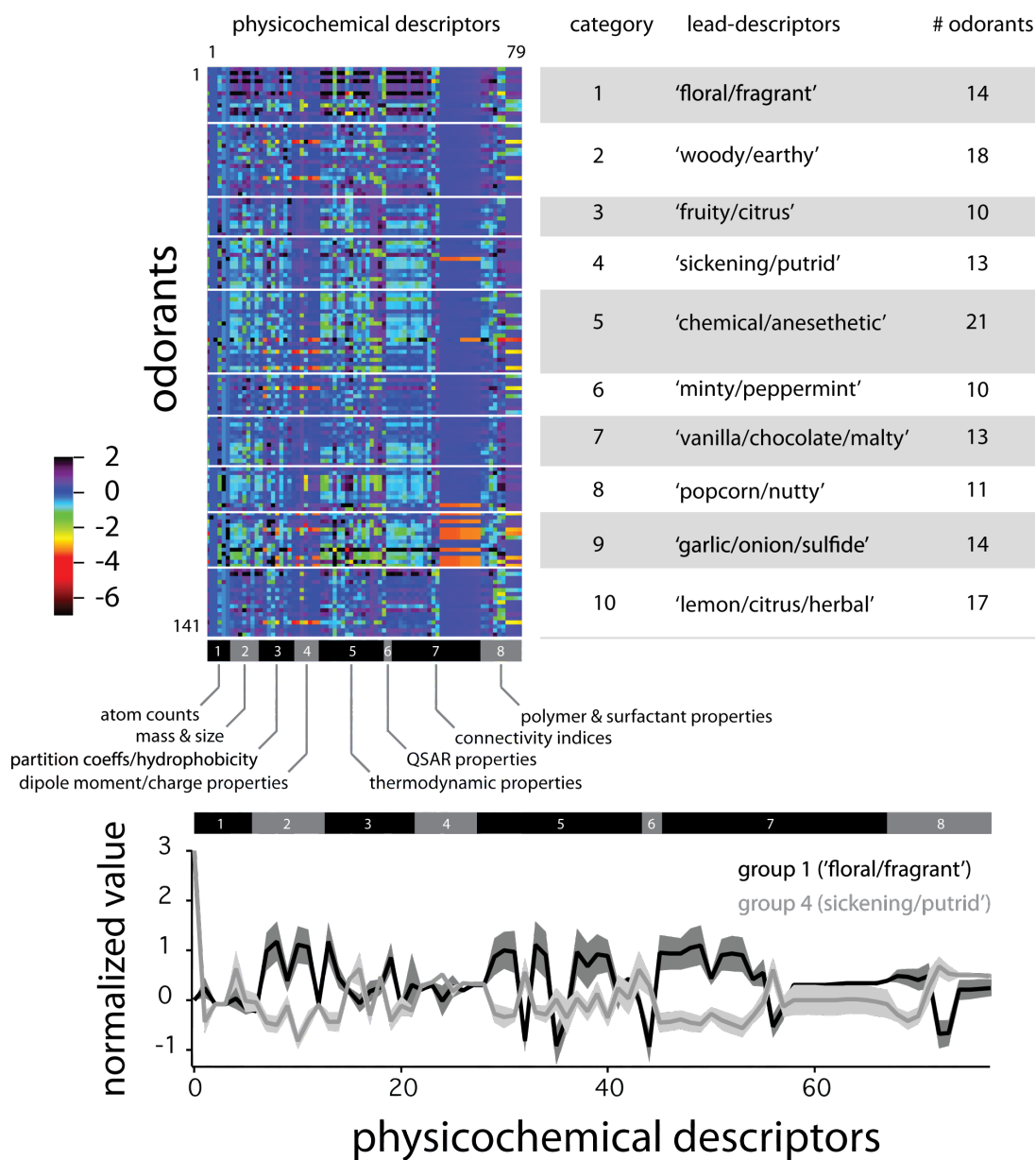


Figure 27: In this study, we use 141 of the 144 odorants in the Dravnieks odor profile database [119]. Rather than using the psychophysical descriptors from that experiment, we use physicochemical properties derived from Koulakov *et al* [111], the general types of which are listed. The 10 perceptual categories are derived from the methods in Castro *et al* [5]. This is the format of the odorants we use here.

data; they differ, however, in the conditions that drive dimensionality reduction. Whereas components obtained from PCA are chosen to maximize variance, those obtained from NMF are constrained to be nonnegative. This constraint has proven especially useful in the analysis of documents and other semantic data where data are intrinsically nonnegative [124,125], a condition that is met by the Dravnieks database. NMF factorizes a matrix subject to the constraints of returning nonnegative and near-orthogonal component vectors of the form

$$D = WH$$

where D is the Dravnieks profile database, W describes patterns in the physiochemical space and H provides pattern affinities of all odorants. Each odorant is assigned the perceptual category for which its coefficient is the largest. Fig. 27 depicts the results of the NMF decomposition and the number of odorants assigned to each category. Three of the highest-ranked perceptual labels within each category are also listed. These categories were shown to be near-orthogonal in the perceptual space, providing strong evidence for the existence of independent odor categories populated by constituent odorants.

4.2.2 Software

Python 2.7 was used to implement the analysis pipeline. We used the scientific computing packages NumPy and SciPy, and the plotting package Matplotlib. Statistical analysis and classification was performed using the Python `scikit-learn` machine learning library [81], which uses the popular `libsvm` implementation for support vector machines. All of these packages are publicly available under open source licenses. The only exception was the derivation of the NMF coefficients, which was performed according to the methods described in our previous work [5].

4.3 DERIVATION OF A GENERALIZED ODORANT SIMILARITY METRIC

Suppose we are given three odorants as described by their physiochemical properties: \vec{x} , \vec{y} , and \vec{z} . We are also given information that \vec{x} and \vec{y} are “similar,” but \vec{x} and \vec{z} are not. Using this information, we want to define a distance metric $d(\vec{x}, \vec{y})$ that encodes this similarity, such that $d(\vec{x}, \vec{y})$ is small, and $d(\vec{x}, \vec{z})$ is large.

Our proposed similarity metric takes the form of a matrix G , formulated according to Xing *et al* [116],

$$d(\vec{x}, \vec{y}) = d_G(\vec{x}, \vec{y}) = \|\vec{x} - \vec{y}\|_G = \sqrt{(\vec{x} - \vec{y})^T G (\vec{x} - \vec{y})}. \quad (4.1)$$

Setting $G = I$, where I is the identity matrix, results in unweighted Euclidean distance. If we define G as a diagonal matrix, this results in a weighted Euclidean distance, where each physiochemical property has its own weight in the distance computation. Deriving a full G matrix allows the metric to incorporate complex interactions between multiple physiochemical properties, most completely representing the physiochemical space.

The process of deriving, or learning, the metric G is an iterative optimization problem. In each iteration, a series of constraints is enforced to guide the subsequent iterations, and to guarantee that the final G is a valid metric (see Materials and Methods). As a result of these constraints, G is symmetric and positive semi-definite, implying its eigenvalues and eigenvectors exist and are real numbers. Fig. 28 depicts the first six iterations in learning the metric G ; its property of symmetry can be observed in each step. We can use the eigenvectors of G to embed the odorants into a low-dimensional space, implementing PCA. However, by performing PCA on the metric G instead of directly on the odorants, the low-dimensional embedding of the odorants incorporates pairwise similarity information. Intuitively, G has the effect of reorganizing the arrangement odorants in this low-dimensional space while maintaining pairwise similarity constraints.

These constraints take the form of two distinct sets of odorant pairs: \mathcal{S} , the set containing all pairs of odorants (\vec{x}, \vec{y}) that are “similar,” and \mathcal{D} , the set containing all pairs of odorants (\vec{x}, \vec{z}) that are “dissimilar” (or, more simply, the pairs of odorants not in \mathcal{S}). In this study, we

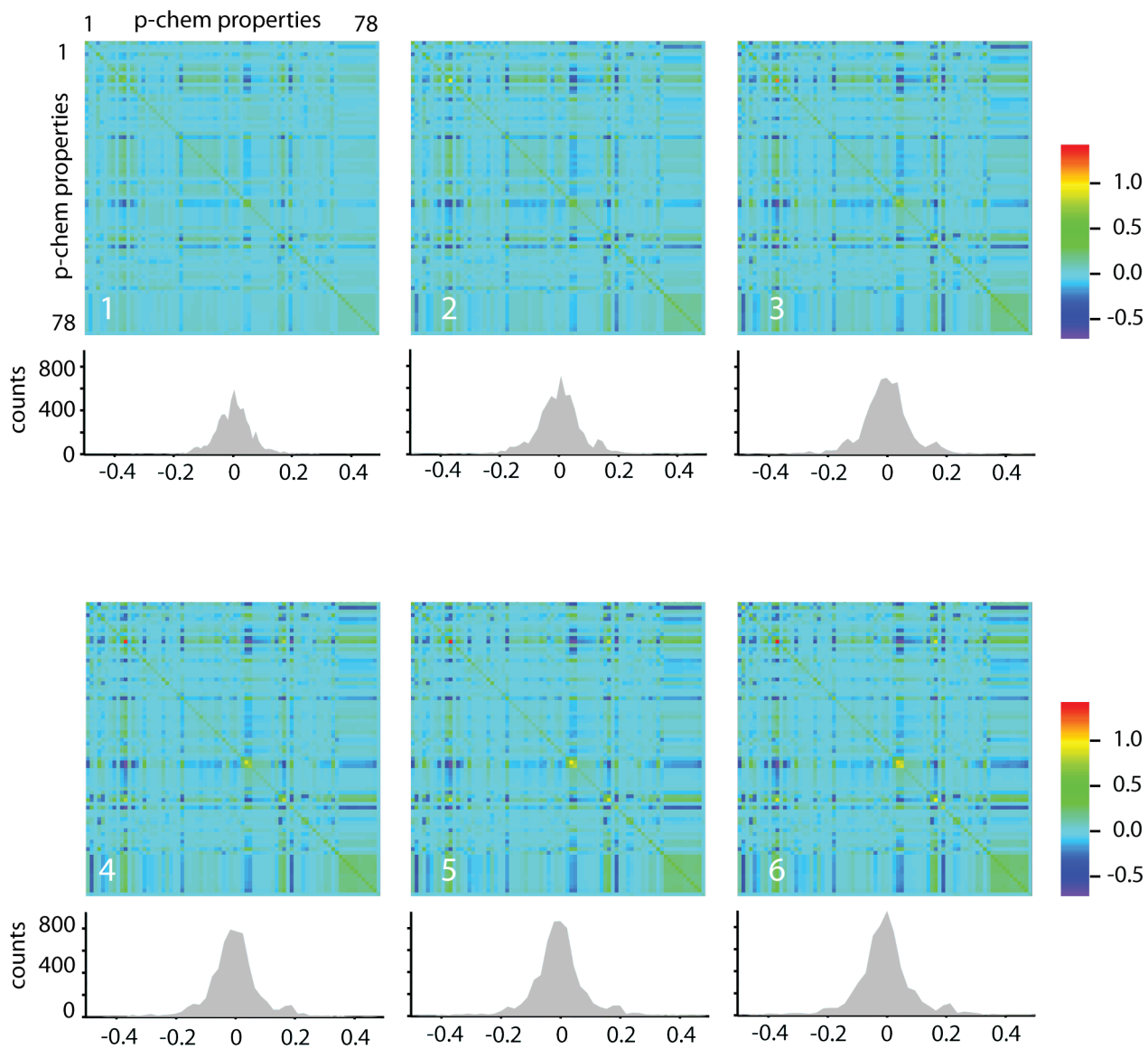


Figure 28: First six iterations of the learning algorithm to derive the similarity metric G . Each entry in G captures quantitative correlation between physiochemical properties; these correlations are plotted as a histogram beneath each heatmap.

claimed a pair of odorants were “similar” if their perceptual categories as defined using the methods in Castro *et al* [5] were identical; if not, the odorants were considered “dissimilar.” Like the Castro *et al* study, we used the odorants from the Dravnieks study; however, in lieu of the psychophysical data used in Dravnieks and Castro *et al*, we used the physiochemical properties from Koulakov *et al* [111] (property types shown in Fig. 27). In populating the sets \mathcal{S} and \mathcal{D} , we based our results on full combinatorial enumerations of all possible odorant pairs.

To learn G , we iterate over the following constraints until they are satisfied:

$$\min_G \sum_{(x,y) \in \mathcal{S}} \|\vec{x} - \vec{y}\|_G^2, \tag{4.2}$$

$$\text{s.t.} \sum_{(x,z) \in \mathcal{D}} \|\vec{x} - \vec{z}\|_G \geq 1, \tag{4.3}$$

$$G \succeq 0. \tag{4.4}$$

In the first step, we minimize the distance $d_G(\vec{x}, \vec{y})$ between pairs of odorants in \mathcal{S} , or those that are similar. Since this can be trivially solved with $G = 0$, we have to enforce another constraint. In the second step, we enforce the constraint that the distances between pairs of odorants in \mathcal{D} , or those that are dissimilar, are ≥ 1 . The choice of the constant 1 is arbitrary and can be replaced by any positive constant a , as long as G is replaced with a^2G . In the second step, one could consider squaring the quantity as in the first step, as it would result in a simple linear constraint. However, this would ultimately cause G to be rank 1, meaning the odorants would always be projected on a line.

Finally, we enforce the property of positive semi-definiteness in G in the last step. This is critical to guarantee that G is a valid metric, satisfying nonnegativity and the triangle inequality (technically, it is a pseudo-metric, in that $d_G(\vec{x}, \vec{y}) = 0$ does not imply $\vec{x} = \vec{y}$). It can be shown that these steps result in a convex optimization problem, guaranteeing that there is a global minimum [116]. We iterate these steps until G converges.

Using the specified formulation, the metric can take two distinct forms. The first is a simpler, diagonal metric, that independently weights each physiochemical property. The second is a full, more complex metric, which is more difficult to derive but can represent

complex linear correlations between physiochemical properties, resulting in a much richer characterization of the physiochemical space. In this study, we use both forms, as the diagonal version is extremely simple to derive, and the full version converges much faster when initialized with the diagonal version (as opposed to a random initialization).

4.3.1 Diagonal Metric

In our programmatic implementation, we first learned a diagonal version of the metric G , where $G = \text{diag}(G_{11}, G_{22}, \dots, G_{nn})$. We defined

$$g(G) = g(G_{11}, G_{22}, \dots, G_{nn}) = g(G_{ii}) = \sum_{(x,y) \in \mathcal{S}} \|\vec{x}_i - \vec{y}_i\|_G^2 - \log \left(\sum_{(x,z) \in \mathcal{D}} \|\vec{x}_i - \vec{z}_i\|_G \right) \quad (4.5)$$

As shown in [116], this is equivalent to the formulation we specified in Eq. 4.2-4.4 up to a multiplication of G by a positive constant. Each term G_{ii} can be solved for in parallel, and Newton-Raphson can be used to find the terms very efficiently. The result of this was our diagonal matrix, G_0 . We did not find any discernible improvement in classification accuracy using the diagonal metric, but we did observe a significant improvement in convergence for the full metric, whose formulation is given next.

4.3.2 Full Metric

The second step of our programmatic implementation used the diagonal matrix G_0 to initialize the following algorithm, which is qualitatively different from the formulation given in Eq. 4.1 as the Newton-Raphson optimization method becomes prohibitively expensive to run over n^2 parameters.

$$\max_G g(G) = \sum_{(x,z) \in \mathcal{D}} \|\vec{x} - \vec{z}\|_G, \quad (4.6)$$

$$\text{s.t. } f(G) = \sum_{(x,y) \in \mathcal{S}} \|\vec{x} - \vec{y}\|_G^2 \leq 1, \quad (4.7)$$

$$G \succeq 0. \quad (4.8)$$

This process consisted of two nested loops. The inner loop contained Eq. 4.7 and 4.8, iteratively enforcing these constraints until G converged and the inner loop finished. Eq. 4.7 involved minimizing a quadratic objective subject to a linear constraint. Eq. 4.8 enforced positive semi-definiteness by diagonalizing $G = U\Lambda U^T$, where $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ are the eigenvalues of G , and U are the column eigenvectors. We replaced G with $G' = U\Lambda'U^T$, where $\Lambda' = \text{diag}(\max\{0, \lambda_1\}, \dots, \max\{0, \lambda_n\})$, guaranteeing the resulting matrix is positive semi-definite.

Once G converged and the inner loop broke, we performed a gradient ascent step according to Eq. 4.6. We performed the update $G := G + \alpha(\nabla g(G))$ (we set $\alpha = 0.0001$, but included a momentum term to further increase convergence speed). Since $g(G)$ is a function of the set of dissimilar odorant pairs \mathcal{D} , we wish to maximize this function (thereby making the resulting distance between odorant pairs in \mathcal{D} as large as possible), hence we take a step up the gradient $\nabla g(G)$. If G had not converged after this gradient step, the inner loop began again.

If G had converged, the algorithm ended. Convergence was defined as a Frobenius norm $\|\bullet\|_F$ below some threshold (we used $\|G - G'\|_F \leq 1$). However, we found that after this condition was reached, we needed to perform one final application of the Eq. 4.8 constraint, as the gradient step appeared to break the positive semi-definiteness of G .

4.3.3 Populating constraint sets \mathcal{S} and \mathcal{D}

Our approach to defining the pairs of similar odorants \mathcal{S} and the pairs of dissimilar odorants \mathcal{D} made use of the 10-category classification in [5]. For each of the 141 Dravnieks odorants used in this study, a pair of odorants \vec{x} and \vec{y} were *similar* if they were assigned to the same perceptual category, and *dissimilar* if they were assigned to different perceptual categories. Referencing the data breakdown in Fig. 27, we enumerated all combinatorial pairings to compute the metric G . To that end, set \mathcal{S} contained 978 pairs of odorants, and set \mathcal{D} contained 17,772 pairs of odorants.

4.4 USING THE SIMILARITY METRIC TO IMPROVE ODORANT CLASSIFICATION

The formulation for defining G in Eq. 4.1 lends itself to the task of *supervised classification*, an area of machine learning concerned with learning a rule that can be used to predict the class, or category, of unobserved data. The rule often takes the form of a line or plane that splits data in such a way as to have the largest number of odorants from a single perceptual category on the same side of the plane. Thus, new odorants are categorized, or classified, based on which side of the plane they fall.

To perform classification, we used a family of machine learning algorithms called Support Vector Machines (SVM) [95, 126], which are particularly effective when the data are sparse and high-dimensional, as with odorant similarity information. We used both linear and nonlinear formulations of the SVM, and detail the results of both where applicable. To tune the classifier, we used k -fold cross-validation. It is a verification process for classification algorithms to estimate their performance against new data. The odorants were split into k groups, or folds, each containing roughly the same number of odorants. In this study, we set $k = 5$; therefore, with 141 odorants, each fold contained 28 odorants, with one fold containing 29. In the first iteration, the first four folds are used to train the algorithm, meaning that the odorants in those folds and their assigned perceptual categories are provided to the algorithm to learn the quantitative associations. The fifth fold is explicitly held out, filling the role of unobserved data. The fifth fold is then used to test the algorithm, whereby the odorants in that fold are provided to the algorithm *without* their assigned perceptual categories, and the algorithm must predict the categories given what it learned in training. The predictions are then compared to the actual percepts, and a percentage accuracy is computed. The process then moves to the second iteration, whereby the fifth fold becomes one of the four training folds, and the next fold in line becomes the testing fold. This continues until all five folds have been used exactly once as the testing fold.

We took myriad approaches to maximizing classification accuracy, exploiting the structure revealed by the similarity metric in numerous ways. These are detailed in the following sections. In addition to these methods, we also implemented other pairwise similarity met-

rics and classification schemes used in olfaction research to compare the performance of our metric.

4.4.1 Substituting each odorant \vec{x} with $G^{1/2}\vec{x}$

Our first approach was to replace each odorant \vec{x} with $G^{1/2}\vec{x}$, enforcing similarity constraints. This has the effect of rearranging each odorant to be closer to those they are “similar” to, and farther away from those they are “dissimilar” from.

4.4.2 Projecting each odorant \vec{x} using leading eigenvectors of G

Our next approach was to use the leading eigenvectors of G as principal components, embedding the odorants in a low-dimensional space spanned by these eigenvectors. Thus, by diagonalizing $G = U\Sigma U^T$, where U are the eigenvectors and Σ is the diagonal matrix containing the eigenvalues, we replace each odorant \vec{x} with $\hat{\Sigma}^{-1/2}\hat{U}^T\vec{x}$, where $\hat{\Sigma}^{-1/2}$ are the leading τ eigenvalues of G , and \hat{U}^T are the corresponding leading τ eigenvectors. In this way, we can vary τ to determine the optimal number of principal components to use.

4.4.3 Baseline methods for comparison to G

Our final approach involved a handful of baseline methods from previous studies to compare against our metric. First, we computed the principal components of the odorants directly using Singular Value Decomposition (SVD) and embedded the odorants in the low-dimensional space spanned by these principal components [110–112] in the same way as our method used the eigenvectors of G ; these methods both attempt to maximize the variance captured along each principal axis (Fig. 29a-b). Resulting embeddings of the odorants in a 3D space using the first three principal components are shown in Fig. 29c-d, and the histograms of pairwise Euclidean distances between odorants in these spaces are shown in Fig. 29e.

In our second baseline method, we used feature selection methods to identify the subset of physiochemical properties that maximized classification performance [126–129]. We employed two feature selection strategies for identifying the subset of physiochemical properties

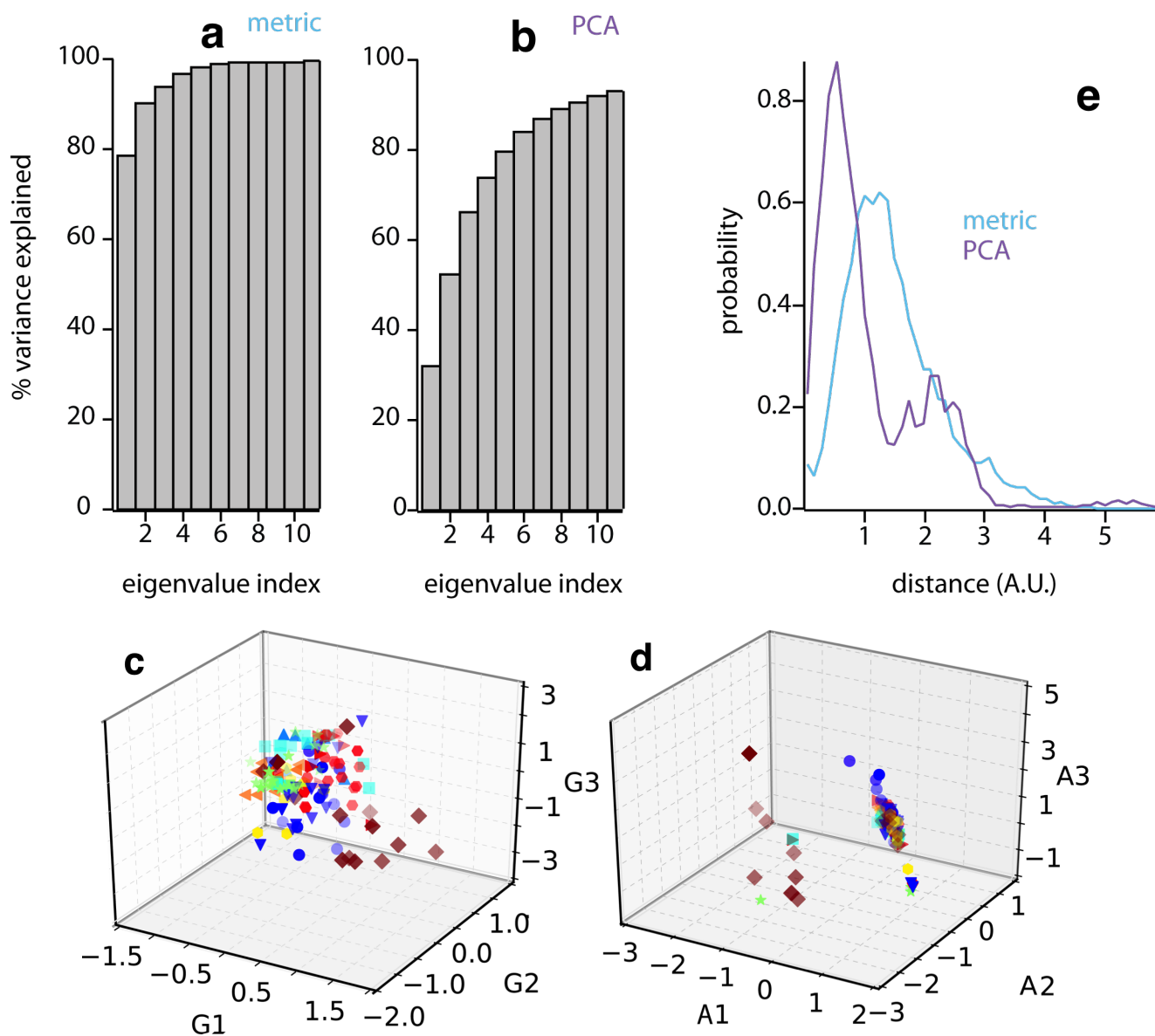


Figure 29: Outcome of PCA on metric G versus directly on odorants. **(a, b)** Cumulative sums of the variances explained by each subsequent principal component of the metric G and of the odorants, respectively. **(c, d)** 3D projections of the 141 odorants using the first three principal components of the metric G and of the odorants, respectively. Each sprite type indicates an odorant of a specific perceptual category as defined in Fig. 27. **(e)** Distribution of pairwise Euclidean distances between odorants in Fig. 29c (black) and Fig. 29d (gray).

best suited for classification, based on the classification method we used. The first strategy, used when performing classification with a linear SVM, was recursive feature elimination cross-validation (RFECV) [126]. The formulation of most linear classifiers involve assigning weights specific features, ranking their relative importances in correctly predicting the category of new input. In this case, RFECV conducts a series of recursive cross-validations in which physiochemical properties are iteratively eliminated from classification based on weights, until only the optimal properties remain. However, RFECV does not work for nonlinear classifiers, as per-feature weights cannot be computed. This was the case with our nonlinear SVM. Instead, we used an iterative method called sequential feature selection (SFS) [127]. This method has been used in previous olfaction studies [128, 129]. In this process, subsets of properties were created and tested against the classifier; the properties which attained the highest accuracy or improved accuracy the most were retained. These features were then included by default in subsequent iterations. This process continued until none of the remaining unselected properties improved classification performance.

In our third method, we performed odorant classification with all available physiochemical properties as outlined in Fig. 27. No PCA embeddings or feature selection methods were used.

In our final baseline method, we developed a null model, which entailed random “scrambling” of the physiochemical properties for each odorant [5].

4.4.4 Classification results

All results using the linear SVM are shown in Table 6, and the results for the nonlinear SVM are in Table 7. For methods that involved parameter scans, only the optimal result and number of associated dimensions used to obtain the result is shown. Our first approach, replacing each odorant \vec{x} with $G^{1/2}\vec{x}$, outperformed all others with an average accuracy of 52.48% ($\pm 2.3\%$), surpassing a five-fold increase over random chance. All other methods still exceeded a four-fold improvement over random chance. Fig. 30a shows the full parameter scan over τ , the number of principal components, of G (black) and the odorants directly (gray). The rows of subsequent subpanel pairs in Fig. 30 depict a confusion matrix on

the left, and per-odor percept accuracy on the right, for our $G^{1/2}\vec{x}$ approach (top row), principal components embedding of G (middle row), and principal components embedding using the odorants directly (bottom row). The confusion matrix is a convenient way to represent the predictions made by a classifier: for a given odorant \vec{x} with *true* perceptual category $y_t \in [1, \dots, 10]$ as indicated by the row number, the count for the corresponding column is incremented based on the *predicted* category y_p . Therefore, zeros everywhere except the diagonal would indicate a perfect classifier, where $y_t = y_p$ for all odorants \vec{x} . In this way, the confusion matrix is a visual representation of where and how the classifier made mistakes. The per-percept plots in the right column show how effective each technique was for recognizing odorants in specific perceptual categories.

More generally, with this metric we require significantly fewer dimensions than previous studies to characterize the principal components of the odorant space. The principal components of our metric G explain a significantly larger percentage of the variance in the odorant data than the principal components of the odorants directly (Fig. 29a-b). Our classifier achieves optimal performance with only six principal components of G , as compared to 40 with the principal components of the odorants. Furthermore, while the first three to four principal components of the odorants explain 80% of the variance (confirming what previous studies have shown), the first principal component of G alone accounts for 80% of the variance, and the first three to four account for over 95%.

The efficacy of the feature selection techniques—RFECV for the linear SVM, and SFS for the nonlinear SVM—are also shown. RFECV achieved a maximum accuracy of 44.09% ($\pm 2.71\%$) using a subset of 60 physiochemical descriptors (Table 6). The second technique, sequential feature selection (SFS) [127], achieved a maximum accuracy of 47.55% ($\pm 2.64\%$) using a subset of 13 physiochemical descriptors (Table 7). It should be noted that this is the only instance in which the nonlinear SVM outperformed the linear SVM. Of particular interest is the fact that, using feature selection, the nonlinear SVM required substantially fewer physiochemical descriptors (13 out of 78) to attain its maximum accuracy than did the linear SVM (60 out of 78). Tables 8 and 9 detail out the physiochemical properties and their respective categories that resulted in the optimal linear and nonlinear SVM performances, respectively, using these feature selection techniques.

$f(\vec{x})$	Accuracy	Dimensions
$G^{1/2}\vec{x}$	52.48% ($\pm 2.3\%$)	78
$\hat{\Sigma}^{-1/2}\hat{U}^T\vec{x}$	49.55% ($\pm 1.95\%$)	6
$\hat{S}^{-1/2}\hat{V}^T\vec{x}$	45.77% ($\pm 2.36\%$)	40
$\hat{\vec{x}}$	44.09% ($\pm 2.71\%$)	60
\vec{x}	42.14% ($\pm 2.48\%$)	78
X_{row}^*	12.1% ($\pm 2.2\%$)	78
X_{col}^*	9.5% ($\pm 1.9\%$)	78
X_{both}^*	8.9% ($\pm 2.1\%$)	78

Table 6: Classification results for the linear SVM. Rows indicate how each odorant \vec{x} was represented in the classification scheme, and the subsequent results. **1st row**: Replace each odorant with $G^{1/2}\vec{x}$, where G is the similarity metric. **2nd row**: Replace each odorant with $\hat{\Sigma}^{-1/2}\hat{U}^T\vec{x}$, where $\hat{\Sigma}^{-1/2}\hat{U}^T$ are the leading eigenvectors and eigenvalues of G . **3rd row**: Replace each odorant with $\hat{S}^{-1/2}\hat{V}^T\vec{x}$, where $\hat{S}^{-1/2}\hat{V}^T$ are the leading singular values and principal components of the odorants. **4th row**: Replace each odorant with a smaller number of physiochemical descriptors $\hat{\vec{x}}$, as found using feature selection (RFECV). **5th row**: Use all available physiochemical properties \vec{x} . **6th-8th rows**: Variations of a null model X^* , depicting the results of scrambling the odorants X by row, column, and both, respectively.

$f(\vec{x})$	Accuracy	Dimensions
$G^{1/2}\vec{x}$	42.40% ($\pm 2.79\%$)	78
$\hat{\Sigma}^{-1/2}\hat{U}^T\vec{x}$	46.02% ($\pm 2.39\%$)	5
$\hat{S}^{-1/2}\hat{V}^T\vec{x}$	41.06% ($\pm 2.75\%$)	30
$\hat{\vec{x}}$	47.55% ($\pm 2.64\%$)	13
\vec{x}	41.16% ($\pm 2.54\%$)	78
X^*_{row}	17.2% ($\pm 1.8\%$)	78
X^*_{col}	13.9% ($\pm 2.2\%$)	78
X^*_{both}	15.5% ($\pm 1.8\%$)	78

Table 7: Classification results for the *nonlinear* SVM. Each row indicates how each odorant \vec{x} was represented in the classification scheme, and the subsequent results. **1st row**: Replace each odorant with $G^{1/2}\vec{x}$, where G is the metric. **2nd row**: Replace each odorant with $\hat{\Sigma}^{-1/2}\hat{U}^T\vec{x}$, where $\hat{\Sigma}^{-1/2}\hat{U}^T$ are the leading eigenvectors and eigenvalues of the metric. **3rd row**: Replace each odorant with $\hat{S}^{-1/2}\hat{V}^T\vec{x}$, where $\hat{S}^{-1/2}\hat{V}^T$ are the leading singular values and vectors of the odorant-descriptor matrix X . **4th row**: Replace each odorant with a smaller number of physiochemical descriptors $\hat{\vec{x}}$, as found using feature selection (SFS). **5th row**: Use the full 78-element descriptor vector \vec{x} . **6th-8th rows**: Variations of a null model X^* , depicting the results of scrambling the odorant-descriptor matrix X by row, column, and both, respectively.

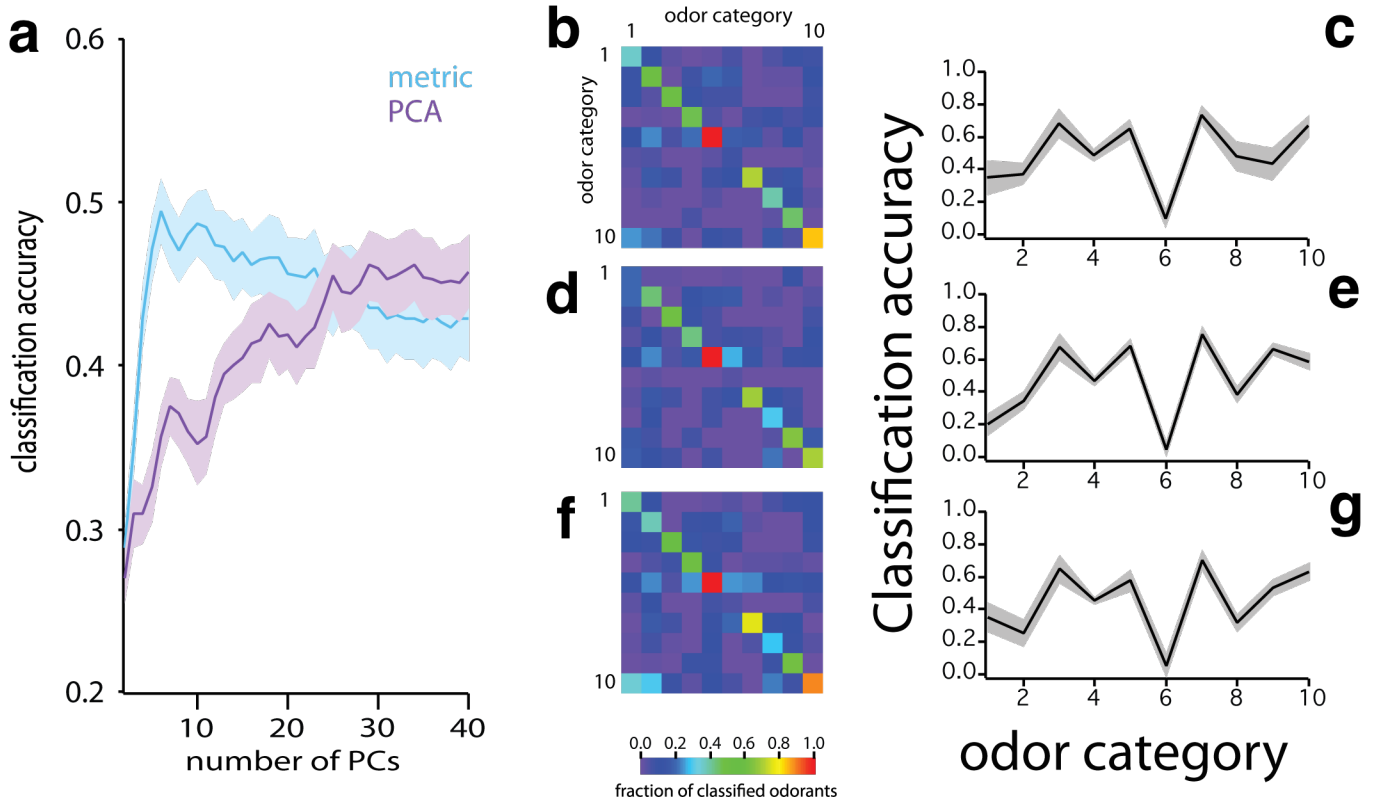


Figure 30: Classification results. (a) Classification accuracy as a function of τ (x-axis), the number of principal components used in G (black line) and the odorants (gray line). (b, c) Confusion matrix and average per-category classification accuracy, respectively, after replacing each odorant \vec{x} with $G\vec{x}$. (d, e) Confusion matrix and average per-category classification accuracy, respectively, using the optimal number of eigenvectors of G (6, from Fig. 30a). (f, g) Confusion matrix and average per-category classification accuracy using the optimal number of principal components of the odorants (33, from Fig. 30a).

Top Linear SVM Features	Feature Set
surface_tension_in_water	Polymer and surfactant properties
valence_index_4	Connectivity indices
viscosity_(cp)_at_25C	Polymer and surfactant properties
surface_area	Mass and size
Critical_pressure_(bar)	Thermodynamics
Normal_freezing_point_(K)	Thermodynamics
boiling_point	Thermodynamics
Enthalpy_of_fusion_(kJmole)	Thermodynamics
Heat_capacity_change_at_boiling_(JK_mole)	Thermodynamics
molecular_weight	Mass and size
C	Atom count
T_sub_g_(C)	Polymer and surfactant properties
O	Atom count
molecular_volume	Polymer and surfactant properties
kappa_2	Connectivity indices
Log_P	Partition coefficients, hydrophobicity, and solubility
vapor_pressure	Thermodynamics
dipole_moment	Dipole moment and other charge properties
valence_2	Connectivity indices
molecular_depth	Mass and size
Hydrogen_bond_number	Dipole moment and other charge properties
N	Atom count
water_content_100%_RH_(moles)	Polymer and surfactant properties
connectivity_index_4	Connectivity indices
van_Krevelen_Z_(chain_length)	Polymer and surfactant properties
H_bond_donor	Dipole moment and other charge properties
water_content_90%_RH_(moles)	Polymer and surfactant properties
Hansen_polarity	Polymer and surfactant properties
molecular_length	Mass and size
Gibbs_energy_of_formation_ideal_gas_at_298_K_(kJmole)	Thermodynamics
Liquid_viscosity_(N_sm ²)	Thermodynamics
HLB	Partition coefficients, hydrophobicity, and solubility
CIM_10	Connectivity indices
entropy_of_boiling_(JK_mole)	Thermodynamics
I	Atom count
T_sub_m_(C)	Polymer and surfactant properties
Normal_boiling_point_(K)	Thermodynamics
LogP_-_Crippen	Partition coefficients, hydrophobicity, and solubility
water_solubility	Partition coefficients, hydrophobicity, and solubility
solubility_parameter	Polymer and surfactant properties
connectivity_3	Connectivity indices
surface_tension	Polymer and surfactant properties
Critical_Temperature_(K)	Thermodynamics
mean_water_of_hydration	Partition coefficients, hydrophobicity, and solubility
Hansen_dispersion	Polymer and surfactant properties
Enthalpy_of_vaporization_at_the_boiling_point_(kJmole)	Thermodynamics
molecular_width	Mass and size
H_bond_acceptor	Dipole moment and other charge properties
valence_3	Connectivity indices
Effective_number_of_torsional_bonds	Thermodynamics
Enthalpy_of_formation_ideal_gas_at_298_K_(kJmole)	Thermodynamics
CIM_4	Connectivity indices
percent_hydrophilic_surface	Polymer and surfactant properties
water_content_70%_RH_(moles)	Polymer and surfactant properties
connectivity_2	Connectivity indices
density	Mass and size
connectivity_1	Connectivity indices
Heat_capacity_ideal_gas_(J_mole_K)	Thermodynamics
Hansen_hydrogen_bonding	Polymer and surfactant properties
CIM_9	Connectivity indices

Table 8: List of the top 60 physiochemical features, ranked by weight from RFECV [126], when using Linear SVM for classification, as⁹⁷ well as the feature sets they are included in from Koulakov *et al* [111].

Top Nonlinear SVM Features	Feature Set
CIM_6	Connectivity indices
Gibbs_energy_of_formation_ideal_gas_at_298_K_(kJmole)	Thermodynamics
Log_P	Partition coefficients, hydrophobicity, and solubility
kappa_2	Connectivity indices
solubility_parameter	Polymer and surfactant properties
T_sub_g_(C)	Polymer and surfactant properties
Liquid_viscosity_(N_sm ²)	Thermodynamics
surface_area	Mass and size
surface_tension	Polymer and surfactant properties
dipole_moment	Dipole moment and other charge properties
Cl	Atom count
Hydrogen_bond_number	Dipole moment and other charge properties
Normal_freezing_point_(K)	Thermodynamics

Table 9: List of the top 13 physiochemical features, ranked by SFS [118], for nonlinear SVM classification, as well as the feature sets they are included in by Koulakov *et al* [111].

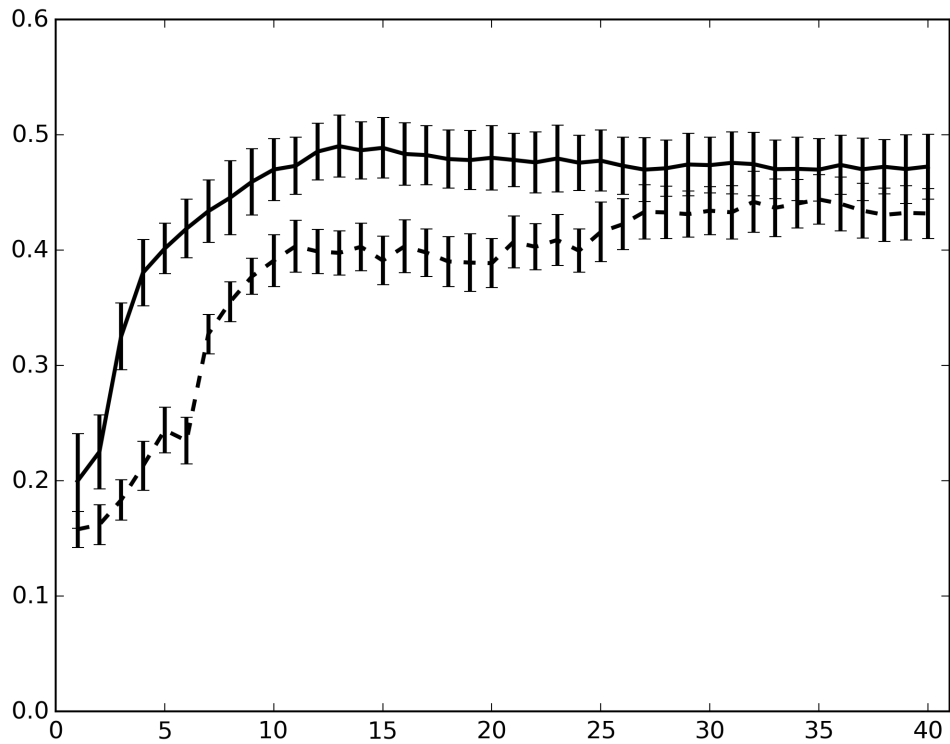


Figure 31: Classification accuracy using feature selection techniques for the linear (dotted) and nonlinear (solid) SVMs, as a function of the number of descriptors used. Descriptors were added according to selection criteria (RFECV for linear, SFS for nonlinear).

Downsampled	Accuracy
25%	51.68% ($\pm 1.92\%$)
50%	51.50% ($\pm 2.08\%$)
75%	46.31% ($\pm 2.07\%$)

Table 10: Effects of downsampling the similarity constraints when constructing the metric G on subsequent classification using $f(\vec{x}) = G^{1/2}\vec{x}$. The percentage indicates the proportion of pairs that are discarded at random when constructing \mathcal{S} and \mathcal{D} .

4.4.5 Effects of downsampling \mathcal{S} and \mathcal{D} on classification

We also experimented with downsampling the constraint sets \mathcal{S} and \mathcal{D} to observe the effect of discarding pairwise information on the integrity of the metric G and the subsequent classification accuracy. We performed three experiments, wherein 25%, 50%, and 75% of the pairs in each set were randomly discarded, resulting in set sizes ($|\mathcal{S}|, |\mathcal{D}|$) of roughly (733, 13329), (489, 8886), and (244, 4443), respectively. These represent averages, as our procedure for discarding data was random. For each pair of odorants under consideration for either \mathcal{S} or \mathcal{D} , we drew a random number from a uniform $[0, 1]$ distribution. If that number did not exceed our discard threshold (0.25, 0.5, and 0.75, respectively), the current pair was discarded. Otherwise, the pair was included. We attempted discard rates over 90%, however in these cases the metric failed to converge due to lack of sufficient data to test the constraints.

The results of our downsampling experiments on the pairwise similarity information in the constraint sets \mathcal{S} and \mathcal{D} strongly suggest the metric is robust to discarding a significant amount of similarity information. Fig. 32 shows that the metric retains its ability to explain much of the variance in the data with only a few dimensions. Retaining between 50% and

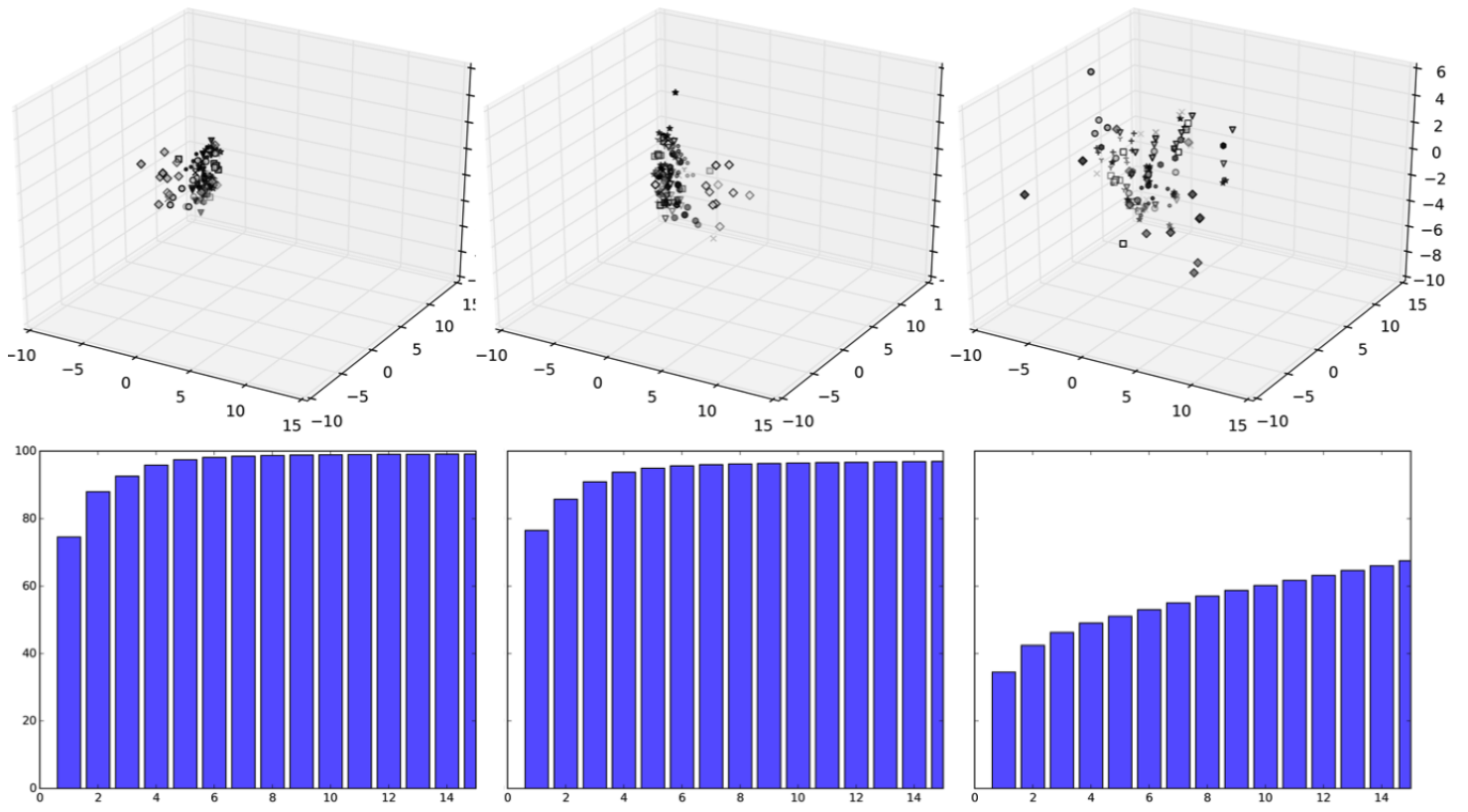


Figure 32: Effects of downsampling the sets \mathcal{S} and \mathcal{D} by 25% (left column), 50% (middle column), and 75% (right column), as reflected in the 3D distribution of centroids (top row) and variance contributed by each eigenvector (bottom row).

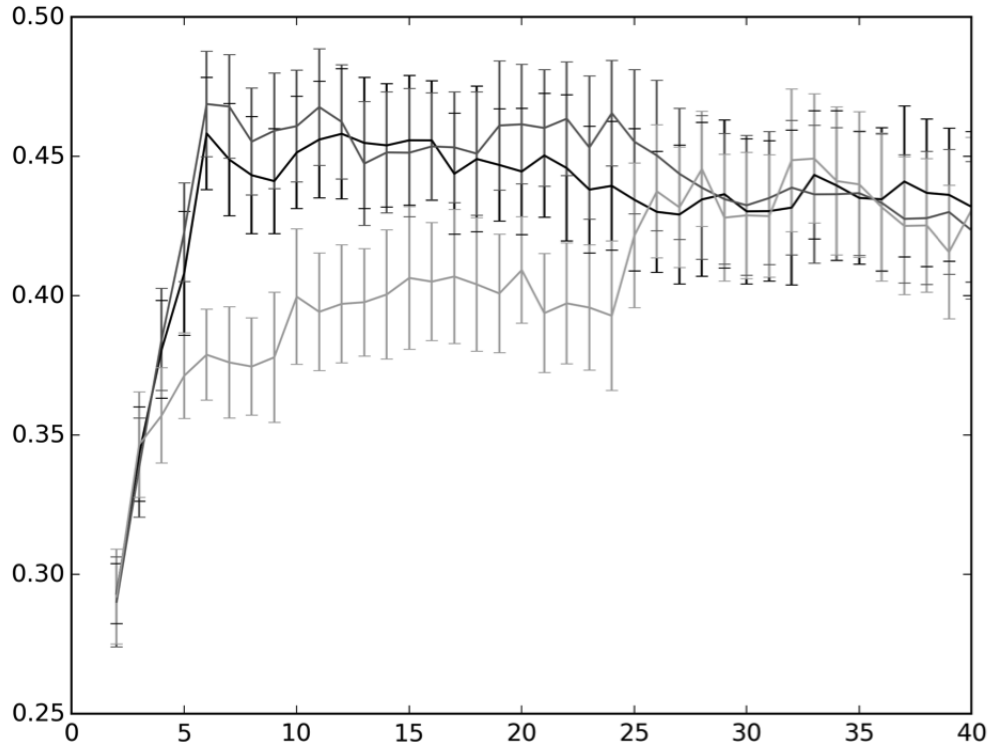


Figure 33: Classification accuracy as a function of number of principal components of the metric G , after downsampling the constraint sets to create G by 25% (black), 50% (dark gray), and 75% (light gray).

75% of all possible pairings of odorants provided a robust similarity metric that generalized well. As shown in Table 10, there is not a noticeable drop in classification performance until only 25% of the original quantity of similarity information remains in \mathcal{S} and \mathcal{D} . This behavior is recapitulated in Fig. 33, wherein only the first few principal components of the metric are required to attain optimal classification accuracy, even after having discarded half of the pairwise similarity information.

4.5 COMPARISON TO OTHER METRICS

4.5.1 Euclidean distance

One closely related distance measure is pairwise Euclidean distances between odorants. For two odorants \vec{x} and \vec{y} , the Euclidean distance is $d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T(\vec{x} - \vec{y})}$, which is equivalent to setting our metric $G = I$, where I is the identity matrix. Computing unweighted pairwise Euclidean distances between odorants in the Dravnieks odor profile database yielded the distribution of values in Fig. 29e (gray). The multimodal nature of the histogram makes it difficult to identify a similarity threshold. Furthermore, it is not clear that odorants, as described by physiochemical properties, exist in a space where small Euclidean distance correlates with perceptual similarity [117].

4.5.2 Cosine angle

Snitz and Yablonka *et al* [117] found a strong correlation between odor percept and angles between odorants as described by their physiochemical properties. Given a pair of odorants \vec{x} and \vec{y} , one can compute the angle $\theta_{\vec{x}, \vec{y}}$ between these vectors $\theta_{\vec{x}, \vec{y}} = \arccos\left(\frac{\vec{x} \cdot \vec{y}}{|\vec{x}| |\vec{y}|}\right)$. We computed pairwise angles between physiochemical properties and plotted those angles against a quantity ϵ derived from our metric. We define the “energy” between a pair of odorants in \mathcal{S} or \mathcal{D} to be $\epsilon = (\vec{x} - \vec{y})^T G (\vec{x} - \vec{y})$. For the pairs in \mathcal{S} , $r = 0.503$ ($p < 0.0001$), and for the pairs in \mathcal{D} , $r = 0.474$ ($p < 0.0001$), strongly suggesting a correlation (Fig. 34).

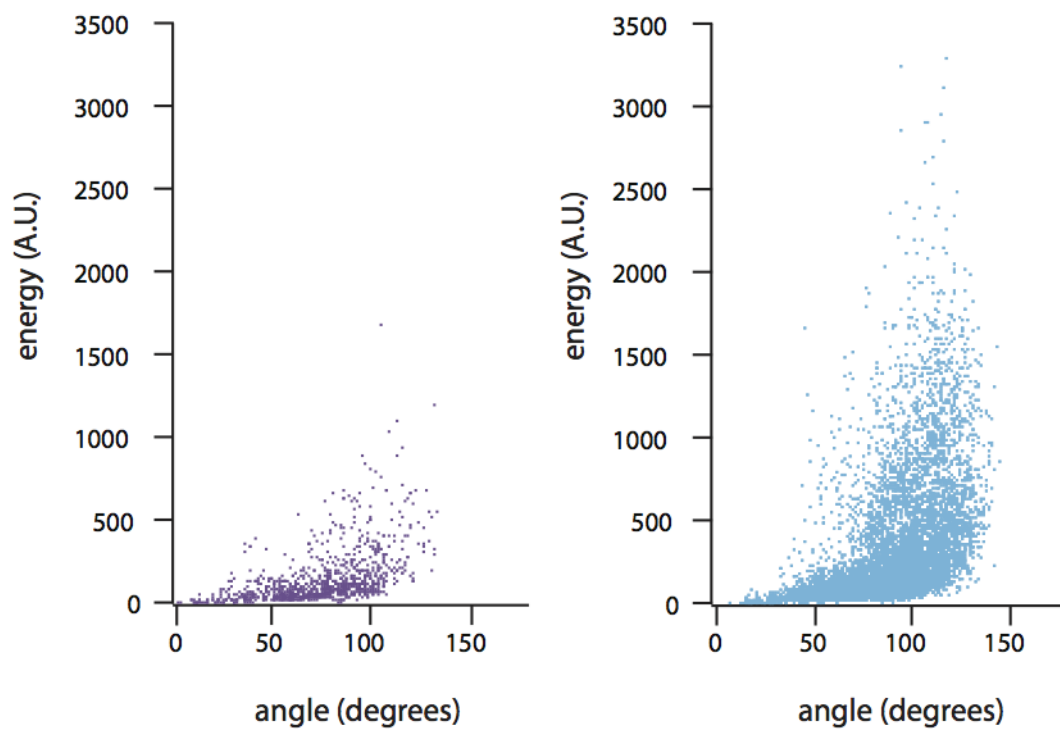


Figure 34: Energy ϵ (y -axis) is plotted against angle θ (x -axis) for odorant pairs in \mathcal{S} (left) and \mathcal{D} (right). The correlation coefficients are $r = 0.503$ (left) and $r = 0.474$ (right). In both cases, $p < 0.0001$.

4.5.3 Alternative descriptor sets

Finally, we also took into account the metric proposed by Haddad *et al* [118]. While the study by Castro *et al* [5] concluded that 10 perceptual categories existed, the same method could be used to categorize odorants into two percepts, theoretically aligning with the “pleasant” and “unpleasant” axes of odor perception. We ultimately converted the number of perceptual categories to a parameter $c \in [2, 25]$ to observe the classification accuracy as it compared to random chance, and in the binary case, to the metric proposed by Haddad *et al*. Our binary classification averaged just over 70%, comparable to the $r = 0.69$ obtained by Haddad *et al*.

4.5.4 Alternative models of olfaction

A recent study [130] posited that humans can discriminate well over one trillion odor percepts using a sphere-packing approach to identifying percepts. However, serious concerns have been raised over the implicit assumptions made in this study; in particular, that the sphere packing relies on an implicit nearest-neighbor approach that gives rise to a fundamentally flawed conclusion [131]. The analysis method appears not to adequately consider the domain to which it is applied, and thus the intrinsic failures of the method misleads the authors. In particular, the authors attempt to quantify the threshold at which two distinct odorants become imperceptibly similar to human olfaction, and use this “distance” to reduce the original question to a hypersphere-packing problem. Once the authors determine the distance d at which 50% of the pairs of odorants are indiscriminable to human subjects, the authors use combinatorics to arrange the maximum number of hyperspheres (representing odorants) in the original high-dimensional olfactory space while constraining the diameter of the hyperspheres to be less than d . The number of spheres, then, is the number used in the paper: more than one trillion.

A simple counterexample is provided in [131]. Using the methods proposed in [130], an example bacterium with only three true percepts is seen to discriminate well over 100 million. This suggests the model proposed in the original paper vastly overestimates the number of discriminable percepts, and that the discrimination data could be explained by a much smaller number of percepts. Given the sphere-packing method, the implicit assumption

made is that all the spheres within the diameter d must be assigned different percepts, when the only requirement of the method is that *neighboring* spheres have different percepts. In the trivial example of packing circles on a two-dimensional plane, the diameter d could be made infinitely large, and yet only three unique percepts are needed such that no two neighboring circles are assigned the same percept. Several other *a priori* assumptions about the olfactory space are made, such as that it is at least 128-dimensional. To satisfactorily develop this method, one needs to find the largest set of stimuli to olfaction such that every percept can be discriminated from every other percept, not just from its nearest neighbors. To avoid brute forcing $O(n^2)$ psychophysical comparisons between every pair of n odorants, a low-dimensional representation of the olfactory space is required. This is the approach we take.

Our method more closely aligns with another recent study [132] in which the authors employ a multivariate regression to correlate physiochemical descriptors of odorants with one of the 146 different odor descriptors from the Dravnieks odor profile database. The resulting framework is a straightforward classification scheme, where each odorant is represented by an 18-dimensional vector of physiochemical descriptors, and the output is one of 146 possible percepts. While the methods are sound and the work in odor stenography are particularly compelling, the method did not explore the low-dimensional embedding of the perceptual space. The authors used a regularization parameter to maintain sparsity in the statistical computations, but no further elucidation of low-dimensional spaces was performed. In this way, our work fulfills one of Markus Meister’s closing statements in [131]: “...knowing that there are > 1 million distinct color percepts is a minor advance. Similarly, finding a low-dimensional basis set for odors would be truly profound.”

4.6 PHYSIOCHEMICAL SIGNATURES UNIQUELY IDENTIFY ODOR PERCEPTS

A particularly exciting application of the metric was deriving “physiochemical signatures,” or small subsets of physiochemical properties, that most affect overall classification accuracy,

and influence each discrete odor percept the most.

For a pair of odorants \vec{x} and \vec{y} , let $\epsilon = (\vec{x} - \vec{y})^T G (\vec{x} - \vec{y})$, where ϵ is defined as the “energy.” We iterated over each physiochemical property x_i , setting that descriptor to 0 for all odorants and recomputing the energies ϵ_0 for all pairs of odorants. We converted these energies to z-scores, where $z = \frac{|\epsilon - \epsilon_0|}{\epsilon}$. The resulting z-scores provided a measure of which physiochemical descriptors most perturbed the similarity constraints encoded in G , and therefore which had the heaviest influence on general classification performance and in forming each distinct perceptual category (Fig. 35). The physiochemical properties whose z-scores exceeded 2.0 for each perceptual category are listed in Table 12; those that exceeded 1.0 for general classification (as very few exceeded 2.0) are listed in Table 11. The physiochemical properties that have z-scores of more than 2.0 for each perceptual category are shown in Fig. 36 with representative odorants from that category.

We observed that each percept emphasized a different combination of physiochemical properties (Fig. 35); the unique combinations and associated weightings of these properties drives the accuracy of classification. This reinforces the findings of previous studies that suggest while odor percepts inhabit a low-dimensional space, this space cannot be reduced to only one or two indicator dimensions. Fig. 37 visualizes the odorants in three dimensions to provide an intuition. The three dimensions used to visualize the odorants were those with the highest z-scores for that percept, in theory representing the three physiochemical properties that most accurately characterize that specific percept. In comparing the 3D projections of Fig. 37 with the percept-specific classification accuracies in Fig. 30c, we note a qualitative correlation: the categories on which we perform better (e.g. 3, 5, 7, and 10) appear to be distributed more evenly across the three dimensions. Conversely, the categories in which we performed worse (e.g. 1, 6, 8, and 9) do not appear to be spread as uniformly, in fact resembling the direct PCA projections of the odorants in Fig. 29d.

Although the physiochemical signatures indicate that the thermodynamic aspects dominate each discrete odor percept, we note that molecular topology and structural features such as connectivity indices and atom counts (such as number of Carbon, Nitrogen, Oxygen, and Sulphur atoms) contribute significantly in improving the accuracy of the classifier. Within the individual percepts we can observe that molecular volume, depth, and width

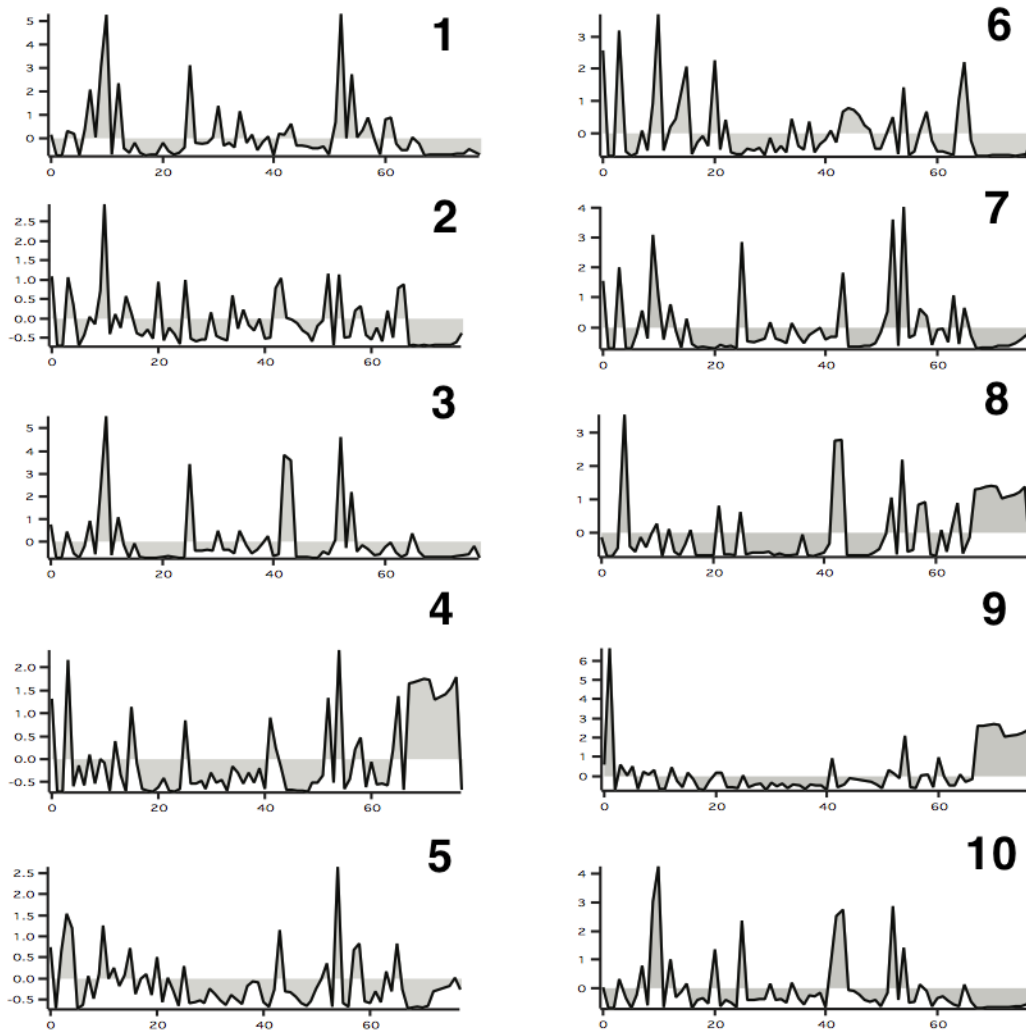


Figure 35: Z-scores of each physiochemical property for all perceptual odor categories. This effectively forms a physiochemical signature of each individual percept, highlighting distinct features that are more heavily represented in certain odor percepts versus others.

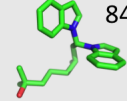
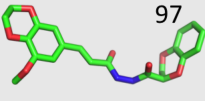
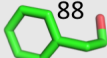
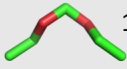
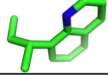
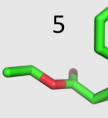
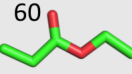
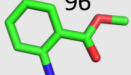
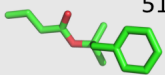
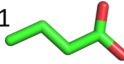

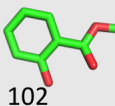
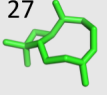



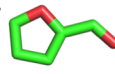
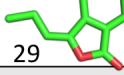
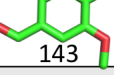

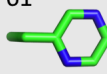
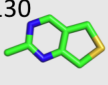
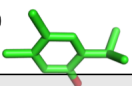
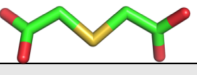
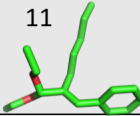
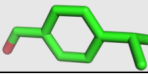
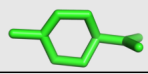
Odor Category	Chemical Descriptors	Odorant Examples
1	Molecular_volume Molecular_width Molecular_depth Surface_area Vapor_pressure Surface_tension_in_water	 84  97  88  115
2	Molecular_depth	 24
3	Molecular_depth Molecular_width Vapor pressure T_sub_g(C) T_sub_m(C) Surface_tension_in_water Critical_pressure_(bar)	 5  60  96  51
4	O Critical_pressure_(bar)	 21
5	Critical_pressure_(bar)	 85
6	C, O Molecular_depth Solubility_parameter H_bond_donor Entropy_of_boiling	 102  27  25
7	O Molecular_width Vapor_pressure Surface_tension_in_water Critical_pressure	 1  142  67  29  143
8	N T_sub_g T_sub_m Critical_pressure	 53  61  130
9	Cl Critical_pressure CIM_{1, 2, 4, 5, 6, 7, 8, 9, 10}	 30  131
10	Molecular_width Molecular_depth Vapor_pressure T_sub_g, T_sub_m, surface_tension_in_water	 11  90  40

Figure 36: Each perceptual category, the physiochemical descriptors associated with it for which their z-scores exceeded 2.0, and example odorant molecules, indicated by their indices from the Dravnieks database. The overall distributions of these physiochemical properties among all odorants and how these compare to the distributions within each perceptual category are shown in Table 12.

Descriptors
molecular_width
molecular_depth
vapor_pressure
T_sub_g_(C)
T_sub_m_(C)
surface_tension_in_water
Critical_pressure_(bar)
CIM.1
CIM.2
CIM.3
CIM.4
CIM.5
CIM.6
CIM.7
CIM.8
CIM.9
CIM.10

Table 11: The properties listed are those that have a z-score of at least 1.0.

Perceptual Category	Descriptors	Overall Distribution	Category Distribution
1	molecular_volume	102.62 (\pm 39.57)	149.34 (\pm 54.53)
	molecular_width	7.39 (\pm 1.37)	8.91 (\pm 1.77)
	molecular_depth	5.53 (\pm 1.23)	6.84 (\pm 1.51)
	surface_area	13.55 (\pm 4.94)	19.35 (\pm 6.63)
	vapor_pressure	3.78 (\pm 13.57)	8.81 (\pm 31.73)
	surface_tension_in_water	26.42 (\pm 0.80)	27.22 (\pm 1.02)
2	molecular_depth	5.53 (\pm 1.23)	5.76 (\pm 1.27)
3	molecular_width	7.39 (\pm 1.37)	7.26 (\pm 1.22)
	molecular_depth	5.53 (\pm 1.23)	5.91 (\pm 1.20)
	vapor_pressure	3.78 (\pm 13.57)	1.57 (\pm 3.33)
	T_sub_g_(C)	-29.69 (\pm 55.78)	-42.44 (\pm 41.35)
	T_sub_m_(C)	178.21 (\pm 108.21)	165.71 (\pm 86.81)
	surface_tension_in_water	26.42 (\pm 0.80)	26.33 (\pm 0.71)
	Critical_pressure_(bar)	31.26 (\pm 10.09)	29.49 (\pm 5.57)
4	O	0.12 (\pm 0.09)	0.18 (\pm 0.11)
	Critical_pressure_(bar)	31.26 (\pm 10.09)	36.95 (\pm 9.06)
5	Critical_pressure_(bar)	31.26 (\pm 10.09)	35.00 (\pm 6.82)
6	C	0.84 (\pm 0.11)	0.88 (\pm 0.07)
	O	0.12 (\pm 0.09)	0.12 (\pm 0.08)
	molecular_depth	5.53 (\pm 1.23)	6.18 (\pm 0.98)
	solubility_parameter	20.26 (\pm 3.05)	19.16 (\pm 2.56)
	H_bond_donor	0.24 (\pm 0.18)	0.16 (\pm 0.15)
	entropy_of_boiling_(J/K_mole)	96.28 (\pm 6.11)	95.28 (\pm 4.94)
7	O	0.12 (\pm 0.09)	0.20 (\pm 0.07)
	molecular_width	7.39 (\pm 1.37)	7.39 (\pm 1.00)
	vapor_pressure	3.78 (\pm 13.57)	0.51 (\pm 1.43)
	surface_tension_in_water	26.42 (\pm 0.80)	26.37 (\pm 0.49)
	Critical_pressure_(bar)	31.26 (\pm 10.09)	36.87 (\pm 6.21)
8	N	0.02 (\pm 0.05)	0.11 (\pm 0.11)
	T_sub_g_(C)	-29.69 (\pm 55.78)	-12.24 (\pm 60.36)
	T_sub_m_(C)	178.21 (\pm 108.21)	242.70 (\pm 124.32)
	Critical_pressure_(bar)	31.26 (\pm 10.09)	35.30 (\pm 6.88)
9	Cl	0.0006 (\pm 0.007)	0.006 (\pm 0.02)
	Critical_pressure_(bar)	31.26 (\pm 10.09)	41.92 (\pm 15.79)
	CIM_1	2.64 (\pm 0.78)	1.02 (\pm 1.37)
	CIM_2	2.49 (\pm 0.73)	0.95 (\pm 1.28)
	CIM_3	2.32 (\pm 0.70)	0.85 (\pm 1.15)
	CIM_4	2.18 (\pm 0.66)	0.79 (\pm 1.07)
	CIM_5	2.04 (\pm 0.64)	0.75 (\pm 1.01)
	CIM_6	1.89 (\pm 0.62)	0.68 (\pm 0.92)
	CIM_7	1.78 (\pm 0.60)	0.60 (\pm 0.83)
	CIM_8	1.63 (\pm 0.59)	0.53 (\pm 0.74)
	CIM_9	1.48 (\pm 0.63)	0.44 (\pm 0.64)
CIM_10	1.31 (\pm 0.63)	0.37 (\pm 0.54)	
10	molecular_width	7.39 (\pm 1.37)	7.83 (\pm 1.57)
	molecular_depth	5.53 (\pm 1.23)	6.40 (\pm 0.85)
	vapor_pressure	3.78 (\pm 13.57)	0.21 (\pm 0.72)
	T_sub_g_(C)	-29.69 (\pm 55.78)	-77.80 (\pm 46.99)
	T_sub_m_(C)	178.21 (\pm 108.21)	79.09 (\pm 86.34)
	surface_tension_in_water	26.42 (\pm 0.80)	26.90 (\pm 0.54)

Table 12: For each perceptual category, the descriptors listed are those that have a z-score of at least 2.0 (Fig. 35), indicating descriptors that play a significant role in quantitatively defining that category (Fig. 36). We list each descriptor, its overall distribution of values across all odorants used in this study, and its distribution of values in odorants within the specific perceptual category. In many cases, we find the overall distribution lies far outside the category-specific distribution.

(which are purely molecular topology descriptors) dominate in four out of 10 categories; similarly, thermodynamic features such as vapor pressure, surface tension, and critical pressure dominate in five out of 10 categories. Only in category 9 we observe that the chemical intuitive molecular (CIM) indices play a role in quantitatively defining that class, perhaps as a consequence of the linear connectivity between atoms represented in this class.

To further quantitate this association between classification accuracy and specific physiochemical property subsets, we used the same scheme as our best performing classifier (Table 6, top row) and re-learned our metric G leaving out atom count information. Resulting odorant classification accuracy fell to 48.47% ($\pm 2.37\%$). We repeated the process, excluding only thermodynamics properties. We observed a considerable drop in accuracy, to 41.93% ($\pm 2.31\%$). We note that, when performing classification using the metric that omitted thermodynamics properties, a substantial shift occurred in some of the perceptual categories’ physiochemical signatures. While per-category accuracy dropped across all percepts relative to the original metric (Fig. 30c), perceptual categories 3 and 9 in particular fell precipitously to near-random performance. We found that the z-scores of the CIM indices had dropped considerably, suggesting a correlation between thermodynamics and CIM properties that the metric captured (Fig. 38). We would also like to point out that in our study we did not incorporate any specific inputs regarding the diverse odor receptors that play a role in recognizing specific classes of odors. As previous studies have shown, the incorporation of both receptor diversity (odorants are chemically recognized by more than one class of odor receptors) and specificity can play an important role in giving rise to a particular odor percept.

4.7 QUANTITATIVE EVIDENCE FOR 10 PERCEPTUAL CATEGORIES

While the choice of 10 perceptual categories was not arbitrary, the same NMF method could be used to reclassify the Dravnieks odorants into different numbers of categories. The method for doing so remained the same: decomposition of the Dravnieks data using NMF yields a certain number of nonnegative basis vectors. The number of basis vectors indicates

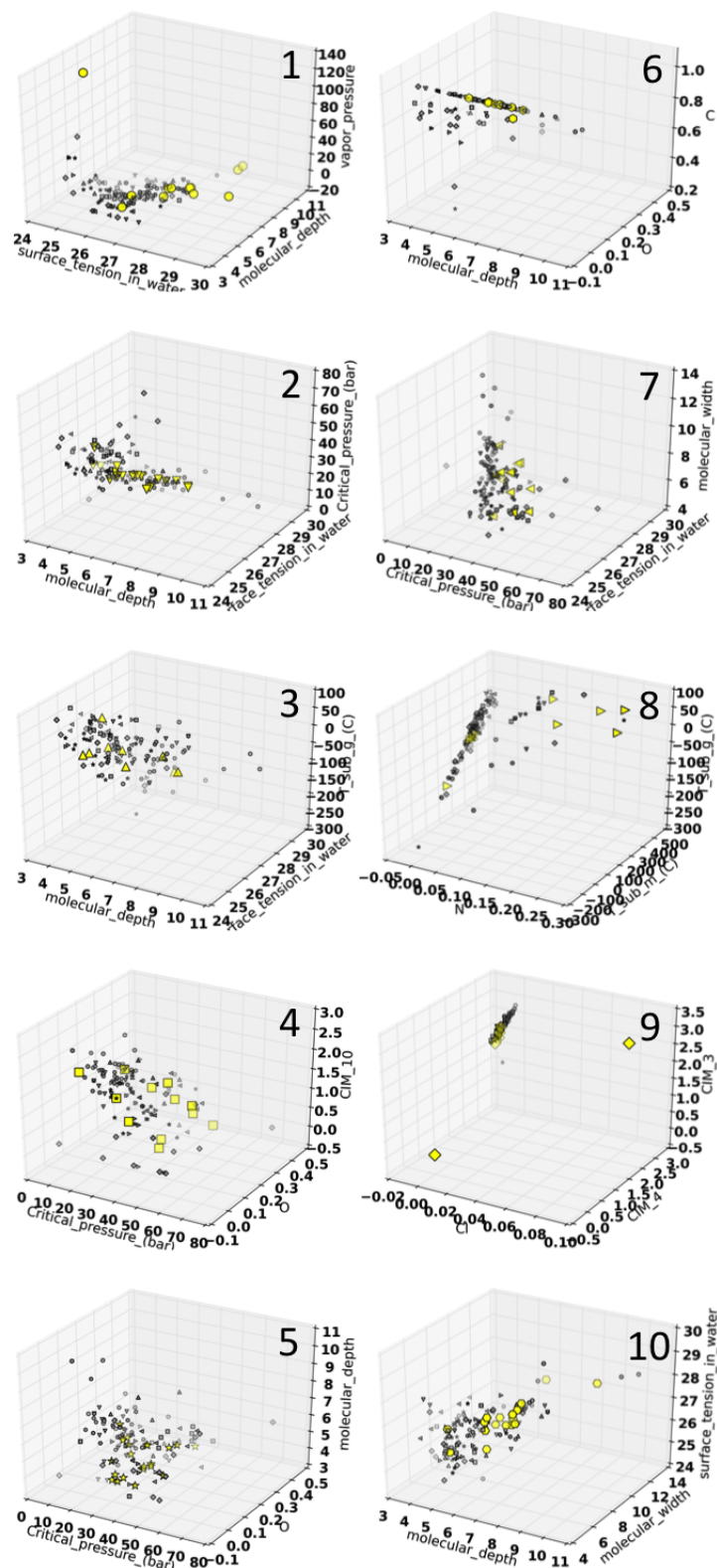


Figure 37: 3D projections of all 141 odorants, using the three physiochemical properties for each discrete odor percept that had the highest energy z-scores for that percept. The odorants that belong to the specified percept (numbered in upper right corner of each panel) are highlighted as yellow markers.

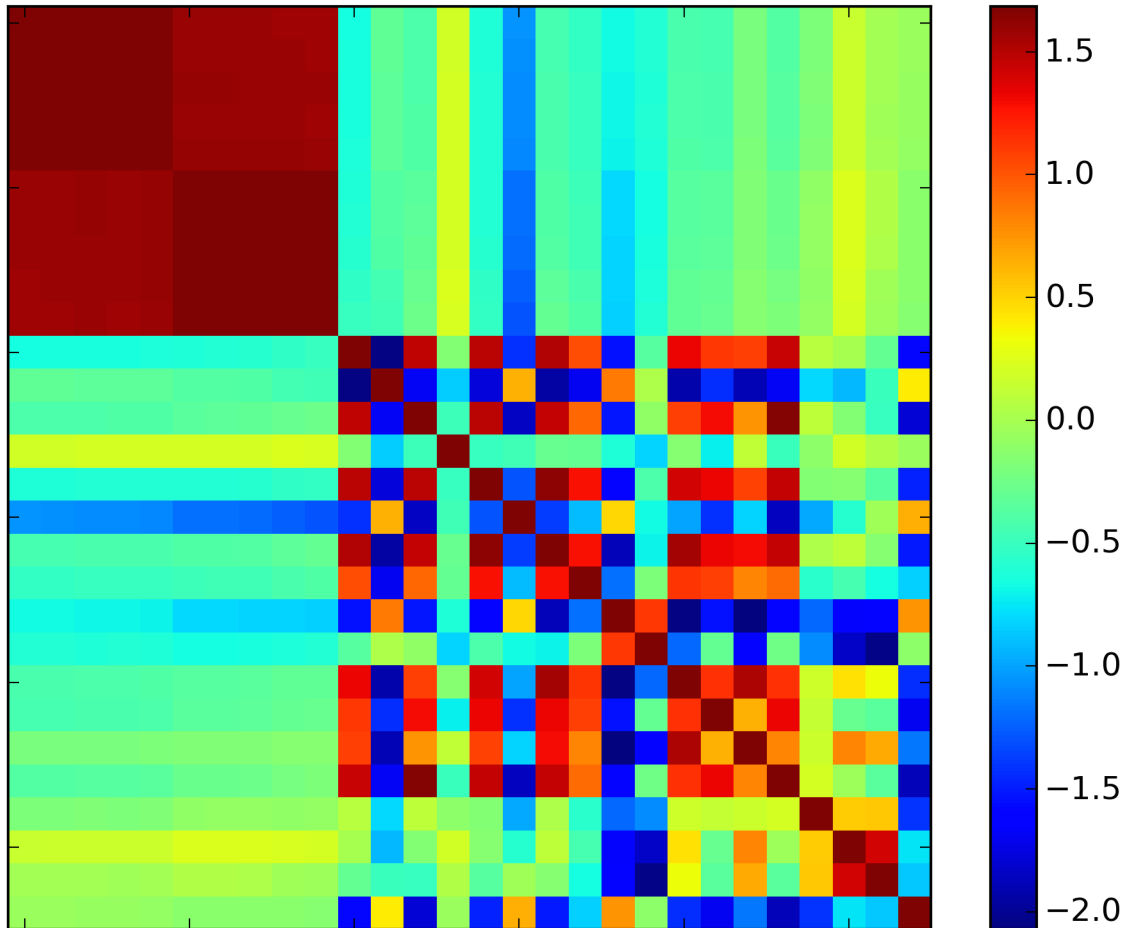


Figure 38: Submatrix of G containing only the rows and columns corresponding to CIM (left-hand block) and thermodynamics properties (right-hand block). The off-diagonal blocks (lower left, upper right) are nonzero, indicating crosstalk between the thermodynamics and CIM properties, providing quantitative support for the observed drop in CIM efficacy when thermodynamics properties were eliminated entirely.

the number of perceptual categories, and the largest element in the respective odorants' descriptor set indicates the category to which it belongs. The motivation for applying this method was twofold: on one hand, we wished to compare our correlation results to studies which identified only two perceptual categories of odorants (pleasant and unpleasant); on the other, since we were using a different feature set (physiochemical properties) from our previous work (psychophysical odor percepts), we wished to observe classification accuracy as a function of the number of perceptual categories in this new physiochemical space.

To this end, we performed NMF and scanned over the number of discrete perceptual categories from 2 to 25, reassigning odorants for each number of categories. For each assignment of perceptual categories, we performed the same supervised classification method using linear and nonlinear SVMs as our baseline method. In this way, we could compare the baseline classification accuracy for each number of categories to the expected random accuracy (e.g., 50% for two categories, 33% for three categories).

The full results of the parameter scan over c are shown in Fig. 39. These results provide additional quantitative evidence to support the findings of our previous work: that there exist more than two perceptual odor categories [5]. In comparing our metric to other pairwise similarity metrics in the literature, we made use of the method in our previous work to perform a parameter scan over a range of possible odor percepts. In performing baseline odorant classification, after having assigned each odorant to one of [2, 25] possible perceptual categories, we found a discernible peak in the classification accuracy around 10-15 perceptual categories (Fig. 39) as compared to random chance. This is significant for two reasons. First, it aligns with the conclusions of our previous work. Second, this alignment appears despite not using the psychophysical descriptors from the Dravnieks study to quantitatively describe the odorants and perform classification. Rather, we used the Koulakov physiochemical properties, which are completely distinct from our previous work.

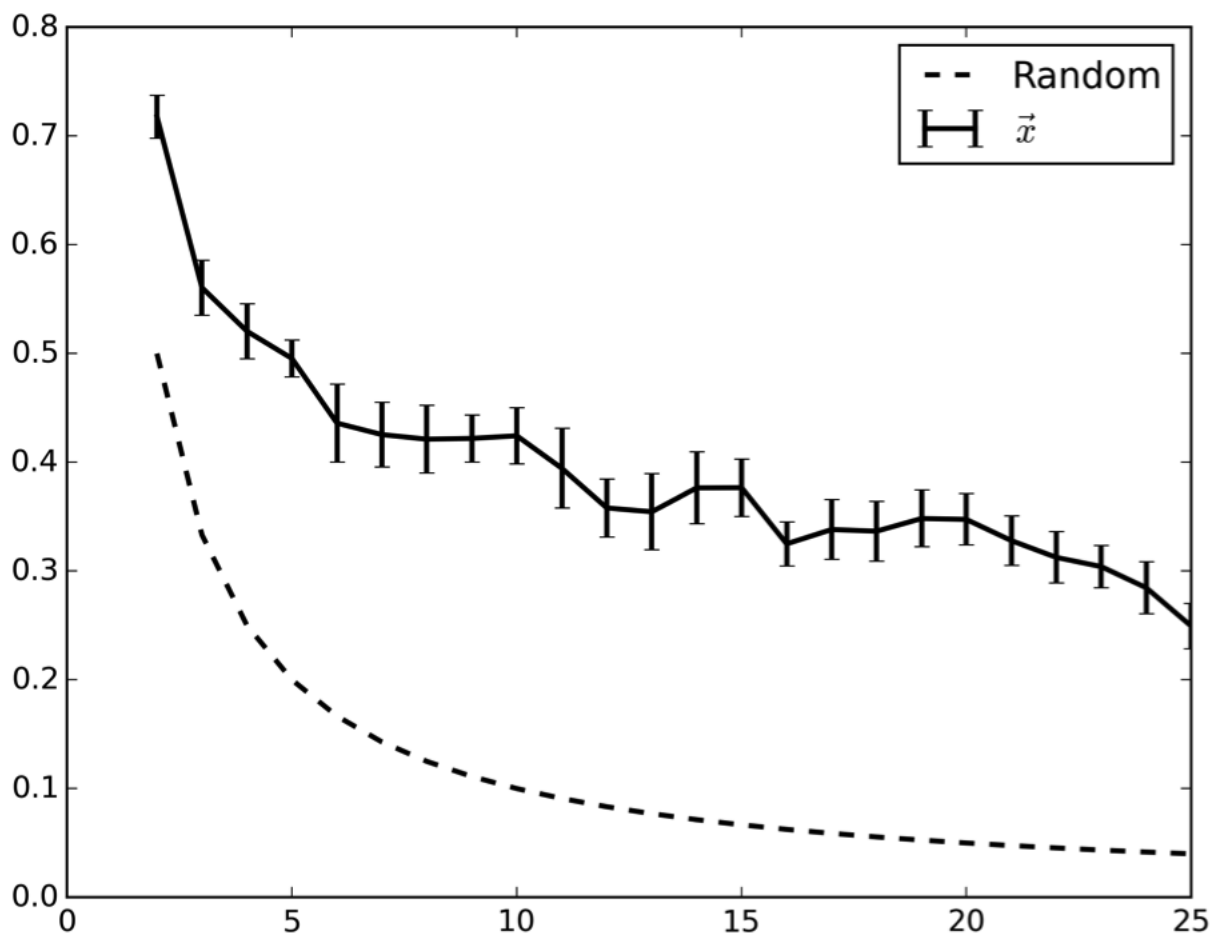


Figure 39: Baseline classification accuracy with all 78 physiochemical properties (solid) compared to random classification accuracy (dotted) against the specified number of perceptual categories, as found using the methods in Castro *et al* [5].

4.8 SEMI-SUPERVISED PROPAGATION OF ODOR PERCEPTS TO UNOBSERVED ODORANTS

We have demonstrated the efficacy of incorporating pairwise similarity and dissimilarity information into a formal classification framework for the purpose of automated odorant categorization. By encoding this information in an analytical framework, we reveal a physiochemical space that can be accurately characterized using very few dimensions. The resulting similarity metric is an inductive model, capable of predicting the perceptual category of any unobserved odorant as described by its physiochemical properties.

The single most important limitation of this method is the relatively small quantity of data used to train and test our metric. Our data (Fig. 27) was a matrix with dimensions 141 odorants \times 78 physiochemical properties. While we have employed statistical techniques to provide results that are as robust to the choice of odorants as possible, this study would benefit significantly from a larger odorant database. However, one of the many advantages of the proposed method is that the trained metric can be used in-place to classify arbitrary odorants; it need not be recomputed from scratch when a new query odorant is introduced.

We employ a semi-supervised learning (SSL) approach to discover potential structure in odor space. SSL refers to a suite of statistical techniques that perform clustering and categorization on data that are partially labeled [133]. Formally, this problem can be addressed by constructing a graph kernel [134] that defines a distance between individual data points, incorporating label information from the few data points that have labels. Acquiring labeled instances, such as associating descriptors to chemicals, can often be difficult, expensive, error-prone, and time consuming. However, unlabeled data may be relatively easy to collect. SSL bridges this gap between labeled and unlabeled datasets. One recent notable success of SSL was work by Fergus *et al* which showed object recognition on an internet-sized image database [135]. In our case, the challenge is to construct a graph kernel on the structural and physiochemical odor space that respects label information (“floral,” “fruity,” etc.) and allows for quantification of molecular characteristics that define the odor space. The kernel learns how to categorize the labels, and this knowledge can be propagated across the space of all odor compounds lacking known percepts. Critically, the approach is agnostic about what

features are the most important criteria for classification, making no *a priori* assumptions about which chemical features should define perceptual similarity.

Traditionally, structure-percept mappings in olfaction have been sought through experiments using (necessarily) small and idiosyncratic subsets of odor space. While this has had some notable successes, in particular that of the Dravnieks experiment [120], we favor an alternative, machine-learning approach that directly grapples with the massive dimensionality of olfactory stimuli. We aim to densely characterize odor space by applying SSL to a publicly available database (PubChem) of $\approx 3 \times 10^7$ chemical compounds, with each compound itself being a multi-dimensional object described by dozens to hundreds of molecular and structural descriptors. Thus, the overarching goal of this project is to develop a SSL framework to parse this high-dimensional chemical space into odor-quality specific domains that recapitulate those observed in human perception.

4.8.1 Graph kernels for semi-supervised learning

Our previous work shows that the perceptual space of odors is well-organized by 10 near-orthogonal dimensions that apply categorically. In the subset of odorspace we studied, groups of odorants are defined by their “membership” in a given one of these dimensions (to the exclusion of others), suggesting that a many-to-few mapping may organize olfaction. Testing this idea on a larger and more representative set of odorspace requires a robust and well-defined metric for quantifying odorant similarity.

We construct a marginalized graph kernel [136] using the metric we derived previously. The metric will serve as the basis for further SSL procedures. The metric will take as arguments two graphs (Fig. 40) (two molecules), and return a distance. In other words, the kernel will provide a metric on odorspace, measuring how “far apart” two distinct odor compounds are. By virtue of its construction (described below), label information about perceptual qualities will be incorporated into this metric.

We combine a kernel based on molecular graph properties, like the marginalized graph kernel described above, with a radial-basis function kernel, which incorporates the difference in the vectorial physiochemical descriptors of the two input odorants. A linear combination

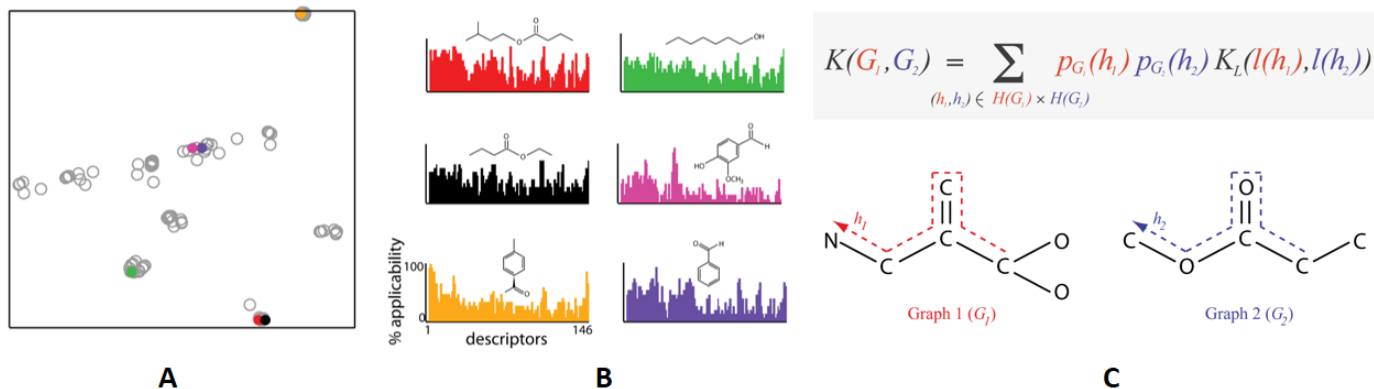


Figure 40: Odor space organization and chemical similarity. **(A)** Two-dimensional embedding of 144 odors expressed in the ten-dimensional perceptual space. **(B)** Odor profiles for chemical indicated in panel A, with structures for each chemical shown above the profile. **(C)** Marginalized graph kernel [136] to measure chemical similarity.

of two kernels is also a kernel [134]. The physiochemical descriptors, such as those generated by E-Dragon [137], comprise a list of 1,986 properties summarizing 20 broad categories of descriptors, including topological, constitutional, and connectivity measures. The weighting on the different kernels can be optimized [138]. By using different types of kernels defined either on the same input or heterogeneous input, one can take different aspects of the molecular information into account.

4.8.2 Using PubChem to test large-scale odor-percept mapping

To make general claims about descriptors that apply to all of odor space (i.e., millions of chemicals), it is necessary to have a standardized library of descriptors, and to describe possible hierarchical relationships among descriptors that are spontaneously applied to odors; for example, the descriptors “jasmine,” “rose,” and “sweet” might all be applied to the same chemical compound, necessitating a statistical consideration of odor classification that disambiguates odor descriptors, and allows hierarchical nesting or mixing of descriptive odor terms. The current vocabulary of odor descriptors has two broad categories: a binary delineation such as “pleasant” or “unpleasant,” and a relation that reflects similarity to

other odors. For example, “lime” evokes similarity to “citrus.” Koulakov *et al* [111] show that the semantic space has much higher dimensionality than olfactory space and that low-dimensionality found in the odor space of Dravniek’s database is not caused by the limited vocabulary used to construct the odor profiles. Following Koulakov *et al*, we will pursue a “bag-of-words” approach [139], in which text searches will be performed over the web using keywords from a list of predefined odor descriptors. All the words that appear within a contextual window of a given size (e.g., 25) around the odor descriptor in the retrieved text are parsed to remove verb/action words and retain non-redundant nouns and adjectives. Together, we expect the standardized list of descriptors to contain $\approx 1,000$ words, each of which is numerically encoded for SSL propagation on PubChem.

We start by constructing a network of all the chemicals from PubChem, some of which are labeled with perceptual descriptors. Specifically, we are given a relatively small labeled set of odorant-descriptor pairs, and a much larger unlabeled set of odorants. In order to use the unlabeled data, we will form a network where the vertices are the odorants and the edges are weighted by the graph kernel K defined in the previous section. While the network will have to be sparse in order to be used effectively within our hierarchical eigensolver, the exact level of sparsity is a heuristic that can be tuned.

4.8.3 Semi-supervised odor percept propagation

Graph-based SSL estimates a label function \mathbf{f} on the graph that satisfies two properties: for a labeled odorant \vec{x}_i with perceptual descriptor y_i the prediction of $\mathbf{f}(\vec{x}_i)$ is close to y_i , and that \mathbf{f} should be smooth on the whole graph. This can be posed as a regularization problem and solved in a linear algebra framework [135]. Let D be a diagonal matrix whose diagonal elements are given by $D_{ii} = \sum_k K_{ij}$. To measure the smoothness of label function \mathbf{f} , we will use the graph Laplacian: $L = D - K$ and compute $\mathbf{f}^T L \mathbf{f} = \frac{1}{2} \sum_{ij} K_{ij} (\mathbf{f}_i - \mathbf{f}_j)^2$

To estimate the label function we will minimize a combination of smoothness and an error terms: $C(\mathbf{f}) = \mathbf{f}^T L \mathbf{f} + \sum_{i=1}^l \gamma (\mathbf{f}_i - y_i)^2 = \mathbf{f}^T L \mathbf{f} + (\mathbf{f}_i - y_i)^T \Delta (\mathbf{f}_i - y_i)$, where Δ is a diagonal matrix with elements $\Delta_{ii} = \gamma$ for labeled odorants and 0 for unlabeled odorants. The solution that minimizes this equation is given by $(L + \Delta) \mathbf{f} = \Delta \mathbf{f}$. This linear algebra

solution is in closed form but requires solving a system of $n \times n$ linear equations. For large n , such the size of PubChem, this poses a problem of scalability that our distributed hierarchical eigensolver is well-suited to address.

Specifically, we will impose the constraint that the label function \mathbf{f} has the form $\mathbf{f} = U\alpha$, where U is a $n \times k$ matrix whose columns are the k eigenvectors of the graph Laplacian L with smallest eigenvalues. The error function simplifies to $C(\alpha) = \alpha^T \Sigma \alpha + (U\alpha - y)^T \Delta (U\alpha - y)$, where Σ is a diagonal matrix of generalized eigenvalues of the graph Laplacian. The solution vector α follows from solving the $k \times k$ system of linear equations $(\Sigma + U^T \Delta U) \alpha = U^T \Delta \mathbf{f}$.

Intuitively, we return to the random walk concept of identifying discrete clusters of similar information within a graph structure. While we have only a small quantity of odorants with experimentally-verified perceptual information and very large quantity with no such perceptual annotations, we do have the results of our similarity method which present strong evidence for physiochemical descriptors as a method for predicting perceptual similarity. Consequently, we can conceptualize our semi-supervised framework as a random walk across a graph of odorants [140], linked by their physiochemical similarity into a graph structure. In this graph, unlabeled odorants that have strong connections to labeled odorants, corresponding to a high transition probability in our random walk, will intuitively have a similar or identical perceptual descriptor. This shares a great deal of theoretical similarity with the PageRank algorithm powering Google search. In Lin *et al*, this takes the form

$$\vec{r} = (1 - d)\vec{u} + dM\vec{r},$$

where M is the Markov transition matrix for the graph, \vec{u} is a sparse vector whose nonzero entries correspond to labeled odorants, d is a constant damping factor, and \vec{r} is the ranking vector. Known as *MultRankWalk*, or MRW, this algorithm conducts random walks starting at the labeled, or “seed,” odorants in \vec{u} and constructs candidate perceptual labels for the unlabeled odorants it visits. The damping constant d modulates how frequently this process restarts, effectively building a probability distribution of perceptual labels over odorants as a consequence of the random walks. This is closely related to the methods proposed by Macskassey *et al* and Zhu *et al* [141, 142], which consider the graph as a *harmonic field*; that is, each unlabeled odorant in the graph is a harmonic (or linearly weighted) average of its

neighboring labeled odorants.

Yet another candidate algorithm is the Modified Adsorption algorithm, or MAD [143], which converts the process of propagating perceptual labels to a convex optimization problem, similar to the one we used previously in this chapter. By casting this as an optimization problem, MAD has several potent advantages, chief among them being 1) it can be solved explicitly; put another way, the conditions under which it will converge are known, and 2) by providing an analytical form, the framework is generalizable to a number of possible requirements.

Each of these approaches not only constitute current state-of-the-art in semi-supervised label propagation, but are also easily parallelized. Some, such as MAD, already have a distributed implementation¹. A particularly noteworthy aspect of these methods is that none explicitly compute the eigenvectors and eigenvalues of the underlying graph; rather, they are inferred through iterative methods (or, in the case of MAD, solved through a separate objective function). We are interested to determine if these methods outperform the explicit computation of eigenvectors and eigenvalues using an efficient, distributed eigensolver.

4.9 CONCLUSIONS

We have presented a machine learning framework that, using pairwise similarity information derived either from psychophysical experiments or computational distance metrics, can learn a metric that is capable of generalizing to unobserved odorants. This is the first step towards automating the process of predicting odor percept from its physiochemical properties. The next step is to use the learned metric to define a graph kernel, which takes as input two odorants and returns some measure of their similarity. Using this information and semi-supervised learning techniques, we can construct a graph of odorants based in their similarity according to the graph kernel. We can then propagate the perceptual categories of known odorants across the graph, using the weights to measure similarity and therefore optimal perceptual predictions. For the vast majority of odorants in the PubChem database that

¹Junto for semi-supervised learning: <https://github.com/parthatalukdar/junto>

have no perceptual annotations, this will provide millions of perceptual predictions and, therefore, potential psychophysical experiments to perform.

While this proposed methodology currently suffers due to the lack of an available method for downloading the PubChem odorants with their associated physiochemical properties, the technical framework is available, and serves to motivate the next chapter. The PubChem database is comprised of millions of odorant compounds, far too many for a semi-supervised approach, which relies on computing eigenvectors and eigenvalues of the underlying graph, to be applied naively. For the technical framework devised here to function on such a large graph, we need a scalable method of analyzing it.

4.10 APPENDIX: LIST OF PHYSIOCHEMICAL PROPERTIES USED

Atom Counts

- Carbon
- Oxygen
- Nitrogen
- Sulphur
- Iodine
- Chlorine

We include counts of heavy atoms only for this study (excluding hydrogens) mainly because we are not aware apriori the potential protonated states of the different compounds. The properties listed below are calculated by Molecular Modeling Pro (ChemSW, Fairfield, CA, USA).

Mass, size

- Molecular weight
- Van der Waals volume (calculated with geometry)
- Molar volume (van Krevelen type method)
- Surface area (calculated with geometry)
- Length, width, depth (current, maximum and minimum calculated by geometry)
- Density (proprietary method for small molecules)
- Mass Percent

Partition coefficients, hydrophobicity and solubility

- Log water octanol partition coefficient (4 methods, Fragment addition generally following the methods of Hansch and Leo, atom based generally following Ghose and Crippen, charge and atom based, and Q Log P after N. Bodor and P. Buchwald, J. Phys. Chem. B, 1997, 101: 3404-3412)
- HLB (hydrophilic lipophilic balance, proprietary method)
- Hydrophilic surface area (proprietary method)
- Percent hydrophilic surface area (proprietary method)

- Polar surface area (J. Med. Chem. 43: 3714-3717)
- Hydration number
- Water solubility (after Klopman *et al.* J. Chem. Inf. Comput. Sci. 32:474 and S. Yalkowsky, J. Pharm Sci., 70:971)
- Olive oil gas partition coefficient (after Klopman *et al.* J. Med. Chem. 43: 3714-3717)

Note that we have excluded concentrations of compounds as part of our physiochemical features.

Properties used in QSAR

- Sterimol properties (L1, B1, B2, B3, B4, B5 and 3 more)
- Hammett Sigma (sigma para, meta, sigma induction (SIND), sigma star)(proprietary method)
- MR (molar refractivity after Ghose and Crippen)

Dipole moment and other charge related properties

- Dipole moment (Modified methods based on Del Re method: G. Del Re, J. Chem. Soc. 4031 (1958); D. Poland and H.A. Scheraga, Biochemistry 6: 3791 (1967); Coefficients modified in MAP 4.0 to take into account pi contributions ; PEOE method: J. Gasteiger and M. Marsili, Tetrahedron 36:3219 (1980); MPEOE (DQP) method: K.T. No, J.A. Grant and H.A. Scheraga, J. Phys. Chem. 94:4732 (1990) and K.T. No, J.A. Grant, M.S. Jhou and H.A. Scheraga, J. Phys. Chem. 94: 4740 (1990); J.M. Park, K.T. No, M.S. Jhou and H.A. Scheraga, J. Comp. Chem. 14:1482 (1993). Semi-empirical Quantum Mechanics methods in CNDO and MOPAC are alternative methods used by MMP to calculate dipole moment.
- Partial charge (many methods - see Dipole moment)
- HOMO/LUMO (via CNDO or MOPAC)
- Hydrogen bond acceptor and donor from charge calculations

Connectivity indices

- Randic, Hall, Kier type connectivity indices 0-4
- Randic, Hall, Kier type valence indices 0-4

- Kier type Kappa shape index 2
- Wiener index
- Chemically Intuitive Molecular Index (F. Burden, *Quant. Struct.-Act.Relat.* 16:309-314 (1997))

Thermodynamics

- Critical temperature, pressure and volume (after Joback and Reid)
- Normal boiling and freezing point (after Joback and Reid)
- Enthalpy of formation, ideal gas at 298 K (after Joback and Reid)
- Gibbs energy of formation, ideal gas, unit fugacity at 298 K
- Enthalpy of vaporization at the boiling point (after Joback and Reid)
- Enthalpy of vaporization at the boiling point (after Joback and Reid)
- Enthalpy of fusion (after Joback and Reid)
- Liquid viscosity (after Joback and Reid)
- Heat capacity, ideal gas (after Joback and Reid)
- Effective number of torsional bonds (τ) (after S. Yalkowsky *et al.*)
- Hydrogen Bond Number (after S. Yalkowsky *et al.*)
- Entropy of boiling (after S. Yalkowsky *et al.*)
- Effective number of torsional bonds (τ) (after S. Yalkowsky *et al.*)
- Heat capacity change on boiling (after S. Yalkowsky *et al.*)
- Vapor pressure (after S. Yalkowsky *et al.*)
- Vapor pressure (after *The Handbook of Chemical Property Estimation Methods*)
- Boiling point (after *The Handbook of Chemical Property Estimation Methods*)
- Parachor (after *The Handbook of Chemical Property Estimation Methods*)

Polymer and Surfactant Properties

- Solubility parameter
- 3-D solubility parameters (dispersion, polarity and hydrogen bonding)
- Water content of polymers at different relative humidities
- Melt transition temperature
- Glass transition temperature

- Chain length (van Krevelen Z)
- Surface tension of liquids
- Surface tension in water
- Molecular weight, molar volume, van der Waals volume, surface area (listed above)
- HLB, hydrophilic surface area, % hydrophilic surface area (listed above)

5.0 DRAENOR: A DISTRIBUTED SCIENTIFIC COMPUTING FRAMEWORK

5.1 INTRODUCTION

To meet the needs of biomedical community going forward, we propose a distributed scientific computing framework with a hierarchical eigensolver at its core. It is derived from multigrid methods, designed specifically for use in distributed environments on large, sparse graphs. We focus on two methods, the GMG proposed by Arbeláez *et al* [40] and the AMG proposed by Krishnan *et al* [35]. These methods both represent cutting-edge efficient hierarchical methods for different scenarios. We have already observed in the previous two chapters that there is a need in biomedical analysis for large-scale spectral analytics. However, this need extends to the broader scientific community. In particular, a focus for future applications of this work is to form the basis of the analytical engine behind the Oak Ridge Biosurveillance Toolkit, or ORBiT [144]. Given the expansive potential applications for a highly scalable distributed eigensolver, we opted to pursue both the GMG and AMG approaches to optimize the frameworks as much as possible for a given category of graph structure, while also making them as generalizable as possible to potential inputs.

5.2 DISTRIBUTED COMPUTING

Distributed computing is a branch of computer science involved in performing computing tasks across many networked physical and virtual machines [11]. Of the many protocols through which such a cluster of machines can communicate, MapReduce [145] is a par-

ticularly popular method. A well-supported, free, and open-source implementation of the MapReduce protocol is the Apache Hadoop project [146]. Hadoop provides a baseline of generic distributed tools and data formats that can be extended for use in specific cases. Many sub-projects, such as Mahout [50] and Giraph [147, 148], build on the Hadoop framework’s scalability to facilitate the efficient implementation of new algorithms. Mahout, for example, provides a general machine learning library that operates at scale, complete with tools for classification, clustering, and recommendation; our group has contributed spectral clustering algorithms to this project. Mahout includes a suite for dimensionality reduction, and its distributed eigensolver, based on the Lanczos iterative algorithm, will be one basis for comparison of our approach. Giraph, while not explicitly a machine learning library, streamlines graph analytics on Hadoop and competes with other graph-based distributed frameworks such as GraphLab [149]. For the purposes of implementing our eigensolver, we will focus primarily on Mahout, Giraph, and GraphLab.

A relative newcomer to the field of distributed computing frameworks is Apache Spark [150, 151], a project born from the Berkeley computer science department. While Spark supports many of the same primitives as Hadoop (map, reduce, etc), it has a couple of critical differences. First and foremost, it primarily operates in main memory rather than reading and writing from main disk, as with Hadoop HDFS. This alone makes Spark orders of magnitude faster for operations that will fit within the main memory of a Spark cluster. Second, Spark uses lazy evaluation. This enables the Spark preprocessor to optimize the command pipeline from the user under the hood during runtime, picking the optimal route according to hardware availability and data accessibility across the cluster. Most typical distributed applications run several orders of magnitude faster on Spark clusters as equivalent Hadoop clusters.

5.3 PRACTICAL DISTRIBUTED HIERARCHICAL EIGENSOLVERS

Because of their theoretical $O(n)$ complexity guarantees, and their excellent empirical performance [35, 38, 40, 152], we elected to implement hierarchical methods. These come primarily

in two families: geometric multigrid (GMG) and algebraic multigrid (AMG). We introduced these in Chapter 2. Here, we discuss how specifically they can be implemented efficiently in a distributed architecture. As we mentioned before, multigrid eigensolvers focus primarily on building successively simpler, or coarser, versions of the original graph until a threshold is reached, at which point the graph is analyzed directly. Both of the methods we examine here rely on this same premise but take a different approach to deciding how to coarsen the graph.

In both cases, we used the Apache Spark framework, specifically the Python wrappers, to implement Draenor.

5.3.1 Language and architecture

We implemented the hierarchical eigensolvers in the Apache PySpark framework [151]. Spark is a relative newcomer to the scene of open source distributed frameworks; however, it has already gathered an impressive following in both research and industry. It operates similarly to Hadoop in terms of distributed architecture; however, it employs Resilient Distributed Datasets (RDDs) [150] to back its data structures in memory, as opposed to disk as with Hadoop’s HDFS. This makes Spark orders of magnitude faster for basic operations that will fit in memory across all the nodes of the network, and more resilient to node failure: results that are lost can be recomputed on the fly.

Spark is particularly advantageous in that its language is optimized internally at runtime. It employs lazy evaluation, such that no operations are performed until absolutely necessary, allowing Spark to find the optimal pathway to completing the requested pipeline in terms of data availability and intercommunication. As cluster and distributed computing becomes more ubiquitous, network latency has become the dominating bottleneck. Spark aims to combat this through intelligent under-the-hood optimizations that minimize data transfer. Lazy evaluation assists in this endeavor by delaying the execution of operators until absolutely necessary.

Furthermore, we found Spark very amenable to rapid prototyping and development. Python has been our language of choice in our other projects, and while the Spark Python

API lags behind its core Scala API, it is robust enough to accomplish some of our core goals. Additionally, having the NumPy, SciPy, and scikit-learn libraries available within a distributed environment is particularly appealing.

5.3.2 Geometric multigrid

As discussed in detail in Chapter 2, GMGs operate most effectively on graphs with regular, predictable structures. Images, as seen in their subsequent graph matrices (Fig. 9), exhibit such regularities. To take advantage of the structural regularities seen in graphs derived from images, we examined one method which used a “decimation” technique for rapidly shrinking the affinity matrix associated with the image without losing precision [40].

Our distributed implementation deviates somewhat from the the method as originally proposed in [40] (which is reproduced here as Algorithm 1). As with most distributed implementations, some assumptions must be made and data structures tailored specifically to these assumptions. We have listed these assumptions and differences here, and provided references to the specific steps in Algorithm 1 where we deviate from the original procedure.

- We implement a single, hybrid distributed matrix which contains the *current* graph matrix A_d , and all the interpolator matrices C_1, C_2, \dots, C_d . The matrix is distributed across the cluster in such a way that each row potentially lives on a different worker.
- Subsequent to the previous point, we assume that a single row of the graph matrix A and interpolator matrix C will fit in main memory. These rows are represented as sparse vectors to further minimize memory usage.
- We assume the indices as specified at Step 4 will fit in main memory. As these indices are a list of integers that can be no larger than $\frac{n}{2}$, this is trivially held in main memory for most modern systems. Therefore, these are broadcasted to the entire cluster for easy and immediate access by the workers. These are also precomputed before starting the main iterations.
- We assume that the coarsest version of the graph matrix and the resulting eigenvectors as computed in Step 9 will fit in main memory. We use a sparse eigensolver (ARPACK) and sparse matrix representation of the data at this level, but it is no longer distributed

across a cluster. Once the eigenvectors are computed, they are parallelized across the cluster, one per worker.

- While we use sparse vector representations of the graph matrix, it is important to emphasize that the coarsened graph will be much less sparse than the original version. We assume the sparse representation will still be more efficient given the size of the data, but this is an assumption that can be tested more thoroughly.
- We changed loop conditional: rather than perform a set number of iterations D , we instead allow the user to set a threshold t on the number of variables remaining before the loop terminates. Therefore, the resulting graph matrix A in Step 9 will have dimensions no greater than $t \times t$, and represented using sparse matrix structures to minimize overhead.

Every operation in this algorithm is extremely efficient. Particularly on the Apache PySpark architecture, array indexing and slicing is $O(1)$ time. Furthermore, the row sum and renormalization steps can also be done very efficiently given that each worker has access to a single row of the matrix. The most expensive operations in this routine is the matrix-matrix multiplication. This can be done in a single map-reduce pass; in many distributed frameworks, for performing a distributed multiplication of two distributed matrices AB , it is implemented as follows:

1. Perform an implicit transpose on A (we use “implicit” to mean that this operation is transparent to the end user). If, however, A is being explicitly transposed, as in $A^T B$, we can omit this first step, greatly improving the overall efficiency (many distributed frameworks have an explicit method call which takes this optimization into account).
2. Conduct an inner join on the rows of A and B , joining each row vector by the row index.
3. In the map phase, loop over the nonzero values in \vec{a} . For each nonzero value a_i , compute $a_i * \vec{b}$ and emit a key/value pair of the form $[i, a_i * \vec{b}]$.
4. In the reduce phase, given a key i , sum the list of the partial multiplications together to form row i of the final matrix AB .

These procedures and assumptions result in the following modified distributed implementation of the original method proposed in [40], which we show as Algorithm 3.

Algorithm 3 Draenor: GMG

```
1: Input:  $A, t, K$ 
2:  $A_0 \leftarrow A$ 
3:  $i = 1$ 
4: while  $\text{length}(A) > t$  do
5:   broadcast( $i$ )
6:    $A_i = A_{i-1}.\text{map}().\text{reduce}()$ 
7:    $i = i + 1$ 
8: end while
9:  $X_i \leftarrow \text{eig}(A_i, K)$ 
10: for  $d = [i : -1 : 1]$  do
11:    $X_{d-1} \leftarrow C_d X_d$ 
12: end for
13: return  $\text{whiten}(X_0)$ 
```

Using the hybrid distributed data structure we described earlier, we could eliminate the join step of the matrix-matrix multiplication, instead performing the decimation of the graph matrix A (Fig. 41) and computing the interpolator matrix C row-by-row within the same worker. Step 5 sends the current level of the hierarchy to every worker on the cluster; the indices marked for decimation were precomputed. At Step 6, each mapper decimates the row of the graph matrix they operate on, and from this row computes the corresponding row of the interpolator C^T . Recall that distributed matrix-matrix multiplication relies on an implicit transpose of the left-hand operand; since Algorithm 1 specifically calls for computing $C^T B$, we can eliminate the implicit transpose altogether. Each mapper will therefore have the i^{th} row of A and the i^{th} column of the interpolator C . These can be multiplied together as described previously, with key/value pairs emitted to the reduce step. In this step, the partial vectors \vec{a} and \vec{c} for the graph matrix and interpolator are respectively summed, resulting in the next level’s graph matrix A and interpolator C . We store these simultaneously in our hybrid data structure.

Once the pixel threshold was reached, we ceased decimation and performed a “collect,”

pulling the disparate elements of the distributed graph matrix together into a coherent structure that resided entirely on a single node. In this way, we could directly invoke a built-in eigensolver to efficiently solve the linear system. Once more in contrast to the original method, we took advantage of the fact that the graph matrix A , its associated Markov transition matrix M , and its graph Laplacian L all share the same eigenspace. To this end, we computed the normalized graph Laplacian $L = D^{-1/2}AD^{-1/2}$, where D is the diagonal degree matrix of A .

After finding the eigenvectors of L , we began the interpolation process, first distributing the eigenvectors to separate workers. The interpolation step, while logically complex, was extremely efficient. Each interpolator C_d was retained across the cluster from the decimation operation; therefore, this was a simple process of multiplying each eigenvector by the corresponding interpolator for the current level.

5.3.3 Algebraic multigrid

For any application outside image analysis, a multigrid method capable of adapting to the algebraic structure of the underlying graph was required. To this end, we implemented a distributed version of the AMG as proposed in [35]. As discussed in Chapter 2, AMGs are particularly well-suited for problems whose graphs depict large spatial or structural inhomogeneities and irregularities, in stark contrast to the graph matrices that result from images.

As a consequence of the increased algorithmic complexity inherent to AMGs, implementing an effective and efficient control flow over a distributed architecture was extremely challenging and still represents an open problem. Several of the core operations required a combination of distributed operators, resulting in a much lengthier average runtime for the basic operations of the eigensolver in comparison to the distributed GMG.

Like the GMG, we make explicit the assumptions and deviations from the original method.

- We implemented a `DistributedRowMatrix` native object, with all the primitives of a typical matrix (e.g., add, subtract, multiply, transpose). Similar to the distributed GMG,

a single row of the matrix is assumed to fit fully in memory on a single worker. Rows are indexed by positive integers, and the rows themselves are represented as sparse vectors for added efficiency.

- Unlike the GMG, the data structure is not hybrid. We have explicit `DistributedRowMatrix` instantiations for the current graph matrix, in addition to the computed interpolators.
- The triangle-finding sparsification step in this algorithm (Step 6 in Algorithm 2) is fully distributed; however, the full collection of triangle instances is assumed to fit in its entirety in local memory. This is a crucial assumption we will revisit in the discussion.
- We assume that the coarsest version of the graph matrix and the resulting eigenvectors as computed in Step 9 will fit in main memory. We use a sparse eigensolver (ARPACK) and sparse matrix representation of the data at this level, but it is no longer distributed across a cluster. Once the eigenvectors are computed, they are parallelized across the cluster, one per worker.
- While we use sparse vector representations of the graph matrix, it is important to emphasize that the coarsened graph will be much less sparse than the original version. We assume the sparse representation will still be more efficient given the size of the data, but this is an assumption that can be tested more thoroughly.
- We assume the two lists of indices (used to reorder the variables, as well as to indicate their colors) will both simultaneously fit in main memory, and are therefore broadcasted to the entire cluster.

Triangle-finding was particularly straightforward to parallelize. By implementing a series of message-passing between rows of a distributed matrix, we could rapidly identify all the triangles and their edge weights within a very large graph. Once the graph has been sparsified and colored, nodes are shifted around to more easily identify the coarsened portion of the graph. In particular, after identifying the coarse c and fine f nodes in the graph, the Laplacian L can be represented as a combination of submatrices (Eq. 2.3).

This reordering operation is also extremely efficient in a distributed setting. Rows can be swapped by simply changing their integer keys; no actual data copying or writing is performed. Furthermore, column swapping is very efficient when the data are represented as sparse vectors.

The final operation, computing the current level’s interpolator matrix and the next level’s graph matrix, is by far the most expensive in terms of moving data across the network in a distributed architecture. This step requires a significant amount of random access to very specific matrix elements in order to compute the two structures needed to continue building out the hierarchy. Each of these operations entail at least one full map-reduce pass, often in conjunction with other distributed operators such as inner joins and filters. This step is a major technical bottleneck in our distributed AMG, and we discuss this limitation further at the conclusion of this chapter.

These deviations from the original algorithm result in a distributed implementation that closely follows the original structure of Algorithm 2; however, Steps 9, 10, and 15 are performed over a cluster of machines. Steps 6 and 7 are interesting cases warranting their own description.

Rather than perform Step 6 for each vertex $v \in A$, we moved this step outside the vertex loop, finding *all* the triangles in the current graph matrix A , each indexed by constituent vertex in the triangles. In this way, we could proceed to loop over the vertices and use the vertex index to the map, instantly obtaining all the triangles in which the current vertex participated.

We also developed a caching mechanism for the sparsification and compensation steps. Rather than push each sparsification update out as soon as the evaluation occurred, we created a list of operations to perform, also indexed by vertex so as to avoid the potential issue of compensating an edge that had already been sparsified for a previous vertex. In this way, we could make a local list of all sparsification and compensation events, and only once all vertices had been iterated over, make one single update push out to the graph matrix (this is known as *lazy evaluation*, in contrast to *eager evaluation*; Spark implements the former in the execution of its distributed operations, also as a way to optimize the analytical pipeline as much as possible).

5.4 EXPERIMENTS

We will cover the empirical performance of the distributed algebraic multigrid in the Discussion section; suffice to say, results were poor enough to warrant the need for further work on streamlining the implementation to better take advantage of the distributed architecture and eliminate bottlenecks. Thus, the entirety of our experiments centered around the empirical performance of the distributed geometric solver and its comparison to our ground-truth method, the sparse SciPy eigensolver as backed by ARPACK.

5.4.1 Complexity analysis

Conducting a thorough complexity analysis with distributed algorithms is potentially very tricky; network traffic and data locality are the two biggest drivers in the runtime of distributed programs. The network is by far the slowest component among the memory, disk, and processing units (CPU and GPU); if the program is not optimized in such a way as to exploit data locality, a great deal of time is spent shuffling the data over the network, significantly slowing the program. While this is technically an aspect of empirical performance, it is a crucial aspect of large-scale analysis that cannot be ignored when considering the complexity of distributed programs. Network speed is many orders of magnitude slower than CPU speed; for sufficiently small n , $O(n)$ theoretical runtime with each of the n instances residing on n different nodes will yield a significantly slower runtime than $O(n)$ theoretical runtime with $\frac{n}{2}$ on two different nodes. Nevertheless, complexity analysis is a useful way of proving that, all else being equal, the algorithm as implemented runs as expected.

5.4.1.1 Distributed geometric multigrid The GMG implementation of Draenor as described in Algorithm 3 has a roughly linear runtime. Again, excluding empirical network performance, the broadcast method in Step 5 is $O(1)$. The map in Step 6 is $O(n)$, whereby each row of A is independently accessed and operated on. Between the map and reduce steps, a network shuffle occurs, where the output of the former is sorted. This is done efficiently in $O(n \log n)$ time. The reduction is keyed; therefore, two vectors will show up in the same node

only if they have the same key (in this case, the row number). As Spark optimizes this to be a rolling process, as opposed to a grouped list of all values with the same key as in Hadoop, determining the algorithmic complexity of this step is difficult as it is nearly exclusively based on the operations of the underlying architecture. Within the reduction itself, nothing more complex than the addition of two vectors occurs; given their sparse construction, this is effectively $O(1)$. Each of these operations is invoked at each level of the hierarchy, resulting in successively smaller definitions of n which falls as an exponential function. Therefore, at each level, $O(n)$ implies at least an order of magnitude speedup as coarser representations are derived, an effect we see in empirical experiments.

Step 9 is the direct eigensolver, relying on ARPACK to perform an iterative Arnoldi decomposition on sparse data structures. The interpolation in Step 11 is slightly more expensive in theoretical terms than the decimation process; where previously we could use a hybrid data structure to enforce data locality between the interpolator and the graph matrix, we cannot maintain this same constraint here and therefore must conduct a full distributed matrix-matrix multiplication. Additionally, we must perform an inner join by row index on the current eigenvectors and the interpolation matrix. However, we can take advantage of the fact that the number of eigenvectors k is significantly smaller than n , thereby resulting in fewer intermediate values than the previous operation in Step 6. This operates the same way otherwise, an $O(n)$ operation on each node, before invoking a similar reduction as before in which rows with the same key are summed.

5.4.1.2 Distributed algebraic multigrid The AMG implementation of Draenor has a roughly quadratic runtime, which is unfortunately infeasible for large-scale computing (see Discussion). The chief bottleneck is the coarsening process, in which the interpolation operators for the current level and graph matrix for the next level are computed from disparate elements of the current graph matrix.

Similar to the GMG, the distributed AMG loops until a threshold of remaining coarse variables is reached. Unlike the GMG, this convergence occurs at a rate that is less than logarithmic; fewer than half the variables are eliminated at each level. Coupled with the larger overall runtime, this further implicates this implementation for additional work. As

mentioned in the previous section, triangle-finding is a single operation performed in parallel only once for each level, an operation in $O(n)$. Each of these triangles is then iterated over, and a list of sparsification and compensation updates is built and cached until all such updates have been compiled for all vertices. These updates are then pushed out to the cluster, another $O(n)$ operation.

The variable coloring process involves examining the neighborhoods of all remaining unmarked variables (often an empty set), all fine variables connected to other fine variables, and all coarse variables connected to other coarse variables. These three operations require a filter and map pass, two $O(n)$ operations per coloring process for a total of $O(6n)$. Empirically, Spark performs some under-the-hood optimizations to decrease this runtime in practice; theoretically we can say this is ultimately $O(n)$.

Reordering the vertices based on their coloring is also an $O(n)$ operation, and straightforward to implement in a distributed setting with a single map-reduce pass. The final step, coarsening the graph matrix for the next level and computing the interpolator for the current level, is the most expensive. This requires extracting blocks from distributed data structures and performing three full matrix-matrix operations, pushing the empirical runtime close to $O(n^2)$. Every effort is made to optimize these processes—for instance, eliminating the implicit left-operand transpose by using L_{fc} in place of L_{cf}^T —however, one of the matrix-matrix multiplications cannot utilize this optimization. While not explicitly $O(n^2)$, each of these operations requires full map-reduce passes, equating to significant quantities of network traffic and subsequent sorting ($O(n \log n)$ for each submatrix extraction and matrix-matrix multiplication) prior to the next operation.

5.4.2 Image analysis

In general, we found the distributed GMG to reproduce eigenvectors of extremely high fidelity relative to the ground truth. While we observed significant deviations in absolute values of the eigenvectors, they still fulfilled the quantitative definition of eigenvectors of the graph matrix. In particular, the `whiten` operation in the final step was crucial to reorthonormalizing the vectors and retaining them as a true basis for the graph.

We conducted a series of empirical performance tests using two different images of varying sizes. These images are described in Table 13. When using images, it is useful to consider them in terms of the number of pixels, as this effectively determines the full dimensionality of the linear equation the eigensolver analyzes. In this view, the smallest problem we attempted had over 22,000 variables, resulting in a graph matrix of size $22,000 \times 22,000$, well beyond the limits of dense eigensolvers. However, because of the extremely sparse nature of image graph matrices, these dimensions are still within the feasibility of sparse eigensolvers, such as those in ARPACK. We used the image in Fig. 41 to establish a ground truth performance comparison.

The results of the comparison performance for the two eigensolvers are shown in Fig. 42. There are clear absolute differences in the resulting eigenvectors for the image; however, for additional analytical purposes, such as image segmentation (Fig. 42, bottom row), the eigenvectors computed from the distributed GMG are as effective, perhaps even more effective, than those computed from the sparse eigensolver. The ground-truth eigenvectors contain significantly more detail, which in image segmentation could perhaps be somewhat of a hindrance, making the eigenvectors more sensitive to noise and artifacts. The decimation step effectively functions as a smoothing method in this case.

Following these successful initial tests, we next tested the scalability of our distributed GMG on larger images. We chose the cat image from Table 13 of three different sizes, equating to problems with 300k, 700k, and 1.3 million variables, respectively. Even for extremely large images, the decimation strategy of removing alternating pixels ensured a rapid coarsening that rarely required storing more than three to five interpolator matrices. In all cases, the eigenvectors were successfully computed and a resulting image segmentation rendered (Fig. 43).

It should be noted that these images constituted systems of equations whose dimensions far exceeded the capabilities of the sparse SciPy eigensolver, therefore eliminating the possibility of a direct performance comparison. The empirical runtimes for the two eigensolvers on the images of varying size are shown in Table 14. With the exception of the largest image, the distributed GMG was tested in a small two-machine Spark cluster with a combined total of 28GB of main memory, 16 cores, and a gigabit intra-departmental ethernet connection.

The largest image used only a single machine; the rationale for this is described in detail in the following section.

5.5 DISCUSSION

At a high level, it is clear that additional engineering is required on both distributed eigensolvers, particularly the AMG. The AMG, as currently implemented, does not fully utilize the distributed architecture, spending significant portions of compute time and resources to shuffling data around the cluster. Indeed, the very theoretical strength of algebraic methods—examining the underlying structure of the graph to optimize the sparsification decisions and preserve the coarse-level graph manifolds—are at the core of the technical pitfalls in a distributed setting, where data locality becomes the biggest challenge and random direct access to elements of a matrix beyond a narrow window (in our case, single row vectors) are prohibitively expensive. To this end, a great deal more work is required to create a competitive distributed AMG eigensolver.

To expound further, a great deal of random access is required in the coloring phase of the algorithm—that is, identifying which variables are coarse (to be recursed over in the next iteration) and which are fine (sparsified and ignored in future iterations). This process relies exclusively on examining the surrounding neighborhoods of these variables, each of which requires two full map-reduce passes over the data. These updates cannot be fully parallelized because of their interdependence: changing the status of a single node will affect the neighborhood of surrounding nodes and must be accounted for in any additional computation. This is same problem encountered with triangle-finding: while identifying and collecting all the triangles is trivially parallelizable and done extremely efficiently, sparsifying and compensating the triangles cannot be easily parallelized, as a single edge can participate in multiple triangles, one of which it may be the weakest (subject to sparsification) while in others it may not be (subject to compensation). This was the rationale behind the assumption that the list of triangles could fit in memory; the PySpark architecture did not provide an efficient means for sparsifying triangles in parallel while enforcing effective “locks”

on edges that participated in multiple triangles.

This comes down to a potential issue of architecture philosophy. PySpark is built as a wrapper around the Scala Spark API, and while extremely efficient in terms of processing data through the standard map-reduce pipeline, the more advanced GraphX API is not yet available through PySpark; it is limited exclusively to Scala. GraphX is a distributed API that is *vertex-centric*. This is similar to Apache Giraph and GraphLab, in which the data primitive is the vertex, and changes are propagated in an efficient way through the edges of the graph. This is the explicit edge-locking needed for edges that participate in multiple triangles and for coloring variables based on their surrounding neighborhood. Therefore, we are considering the myriad problems with the AMG technical, engineering obstacles. Spark is still a very young architecture undergoing rapid development; further time and resources are required to fully exploit its strengths and also compare those of other frameworks.

The distributed GMG showed promise. By virtue of its spatially homogenous sparsification strategy, a much lesser degree of random access was required in the underlying graph. Furthermore, the data structure we used was optimized such that the row indices of the current graph matrix and interpolation matrix for the current level aligned, enforcing data locality and minimizing the quantity of data being sent over the network. This effect was particularly pronounced in the interpolation process, where even for the largest image, the eigenvector interpolation was done on the order of seconds (Fig. 44).

However, the stage breakdown in Fig. 44 provide detailed insight into the processing bottlenecks encountered in the distributed GMG. In both cases, a distinct bottleneck is observed at Stage 3, which coincides with the first decimation-and-squaring step at the finest level of the hierarchy, when the dimensionality of the problem is the largest. In the small image (Fig. 44, top), Stage 3's 17 minutes comprises 72.8% of the total runtime; in the medium image (Fig. 44, bottom), Stage 3 runs for a full hour and comprises 83.4% of the total runtime. The decimation strategy ensures the dimensionality of the problem will fall extremely fast, requiring only a handful of such steps, but overcoming the first is clearly the biggest hurdle and results in a bottleneck that keeps the remainder of the algorithm idle for long periods of time.

The precise cause of this bottleneck is difficult to identify; as we mentioned in the com-

plexity analysis, this stage of the algorithm comprises only linear operations over sparse data structures. However, the most likely culprit is not the algorithm itself so much as the intervening shuffling of the underlying framework: matrix-matrix multiplication results in an explosion of intermediate values that must be sorted in between analysis phases and sent to the correct nodes on the cluster for further processing. In particular, in distributed matrix-matrix multiplication, a full sparse vector is emitted for *each* nonzero value of *each* row. Even if the number of nonzero values p is significantly smaller than the total number of variables n , the network will receive pn intermediate values from the `map` phase, resulting in a sort time of $pn \log pn$ and subsequent network traffic proportional to pn . There is a difference of several orders of magnitude between linearly iterating over p nonzero values of a sparse vector in memory, and sorting pn values over a network when n is extremely large.

As we alluded to in the previous section, the empirical tests on the largest image were performed on a single-machine “cluster.” We initially tested the algorithm on a very large virtual cluster from the Amazon Web Services (AWS) EC2 platform. The cluster consisted of a single master with 10 workers, each of the `m3.xlarge` variety, optimized specifically for in-memory computations. However, we found that Stage 3 never completed within an acceptable period of time; we ran the algorithm for 12 hours before terminating it. Given that a single machine was capable of executing the algorithm on the same dataset whereas an extremely large EC2 cluster was not, this points to network latency as the primary bottleneck.

Perhaps with the exception of the second point, however, the primary issues were technical, not theoretical. The empirical performance observed under certain conditions point to hierarchical solvers as a potential boon in distributed computing. Draenor is novel in that, while not of cutting-edge performance, is the first hierarchical solver to our knowledge to be implemented on top of an open-source general-purpose distributed computing framework. With additional investigation, maturity of the Scala and Python APIs in Spark, and additional development time, Draenor could very likely operate on par with existing solvers. Already we have observed that the analytical performance of the resulting eigenvectors from the distributed GMG are robust and just as useful for purposes such as segmentation as their ground-truth counterparts (Figs. 42, 43).

This addresses the need in biological research for a method to analyze the corpus of data in a way that accounts for its increasing size. We have not discussed existing parallel implementations, as these require additional and expensive hardware to scale to larger datasets; supercomputers are an example of highly parallel but expensive resources that are difficult for many researchers to access. By contrast, assembling a cluster of commodity hardware for distributed computation is not only vastly cheaper but can rapidly approach supercomputing performance [11]. The Draenor framework is the first step in that direction: making scientific computing readily available at a large scale.

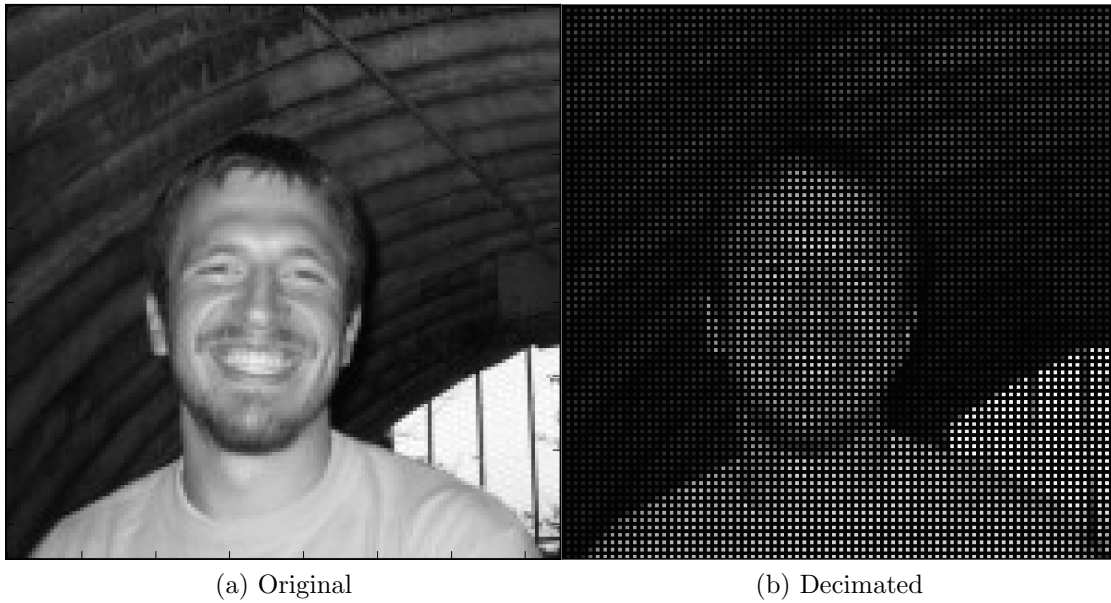


Figure 41: **Decimation process.** This visualizes the decimation process of the GMG sparsification strategy, providing some intuition for how the underlying graph matrix is simplified while still minimizing the loss of information: in the same way as one’s visual process “fills in” the blanks to a certain extent, the subsequent squaring procedure propagates this information and “fills in” the missing pixels.

Image	Dimensions	Pixels
Person	150×150	22,500
Cat (small)	540×720	324,000
Cat (medium)	768×1024	786,432
Cat (large)	1024×1365	1,397,760

Table 13: Four images used to conduct initial tests of the Draenor GMG.

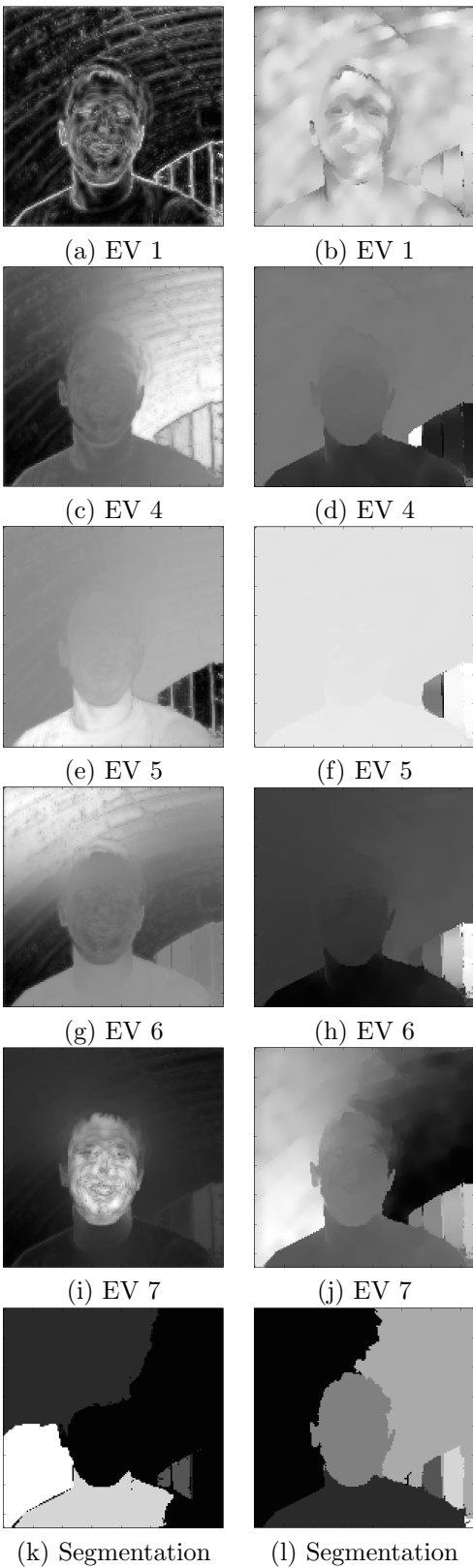


Figure 42: **Built-in eigensolver compared to distributed GMG.** Eigenvectors of the associated graph Laplacian of the image in Fig. 41, computed from SciPy's built-in sparse eigensolver (ARPACK, left column) and the Draenor distributed GMG (right column). The image segmentation results from using the depicted eigenvectors are shown in the bottom row.

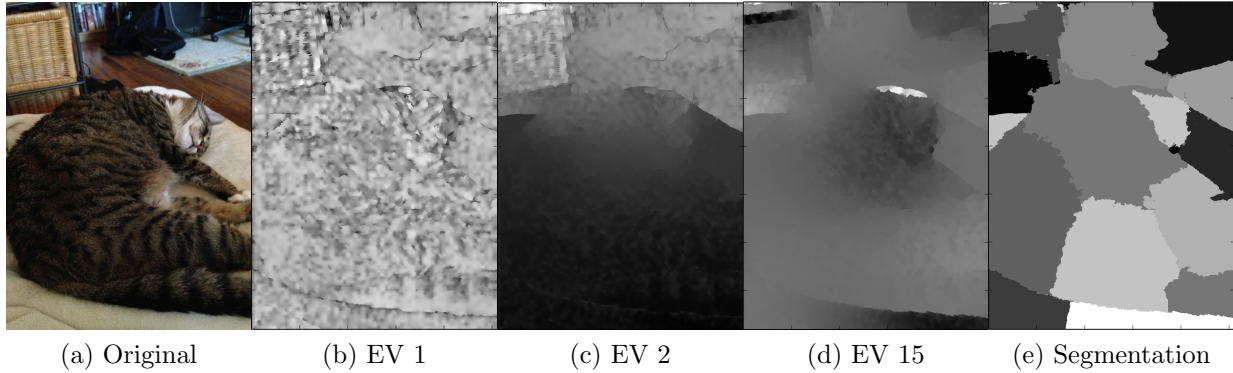


Figure 43: **Large-scale GMG performance.** From the original image, we computed the top 32 eigenvectors across a cluster using the adjusted distributed GMG method. This yielded an extremely satisfactory image segmentation.

Image	SciPy Sparse	Draenor GMG
Person	14s	67s ($t = 1024$); 57s ($t = 8196$)
Cat (small)	n/a	1,322s (22min) ($t = 8196$)
Cat (medium)	n/a	4,327s (72.1min) ($t = 8196$)
Cat (large)	n/a	18,326s (5hrs, 6min) ($t = 8196$)

Table 14: **Empirical eigensolver runtime.** For four different images, we tested the efficacy of the Draenor GMG and compared it to the built-in sparse SciPy eigensolver, insofar as such dimensionality was feasible for a single machine.

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total
6	collectAsMap		2014/11/20 12:51:05	3 s	48/48
20	RDD at PythonRDD.scala:261	+details	2014/11/20 12:50:48	17 s	16/16
19	RDD at PythonRDD.scala:261	+details	2014/11/20 12:50:46	3 s	96/96
14	RDD at PythonRDD.scala:261	+details	2014/11/20 12:50:41	5 s	16/16
13	RDD at PythonRDD.scala:261	+details	2014/11/20 12:50:39	2 s	96/96
12	RDD at PythonRDD.scala:261	+details	2014/11/20 12:50:37	2 s	16/16
7	RDD at PythonRDD.scala:261	+details	2014/11/20 12:50:35	2 s	96/96
1	collect		2014/11/20 12:50:23	9 s	48/48
5	RDD at PythonRDD.scala:261	+details	2014/11/20 12:49:27	56 s	48/48
4	RDD at PythonRDD.scala:261	+details	2014/11/20 12:46:03	3.4 min	48/48
3	RDD at PythonRDD.scala:261	+details	2014/11/20 12:29:07	17 min	48/48
2	RDD at PythonRDD.scala:261	+details	2014/11/20 12:27:55	1.2 min	16/16
0	collect		2014/11/20 12:27:44	8 s	16/16

Stage Id	Description		Submitted	Duration	Tasks: Succeeded/Total
7	collectAsMap		2014/11/20 15:41:58	4 s	72/72
25	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:31	27 s	24/24
24	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:27	4 s	144/144
18	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:19	8 s	24/24
17	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:16	4 s	144/144
16	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:13	3 s	24/24
15	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:10	3 s	144/144
14	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:08	1 s	24/24
8	RDD at PythonRDD.scala:261	+details	2014/11/20 15:41:05	3 s	144/144
1	collect		2014/11/20 15:40:59	5 s	72/72
6	RDD at PythonRDD.scala:261	+details	2014/11/20 15:40:36	23 s	72/72
5	RDD at PythonRDD.scala:261	+details	2014/11/20 15:39:10	1.4 min	72/72
4	RDD at PythonRDD.scala:261	+details	2014/11/20 15:32:05	7.1 min	72/72
3	RDD at PythonRDD.scala:261	+details	2014/11/20 14:32:01	1.0 h	72/72
2	RDD at PythonRDD.scala:261	+details	2014/11/20 14:30:10	1.8 min	24/24
0	collect		2014/11/20 14:29:53	12 s	24/24

Figure 44: **Spark bottlenecks.** Spark provides a breakdown of the various stages of the distributed computation, with the first stages at the bottom and the final stages at the top. The top box are the timeline statistics for the small cat image; on the bottom, the medium-sized cat image (large breakdown not available). It is evident the bottleneck exists at the very first level of the decimation process (Stage ID 3 in both images).

6.0 CONCLUSIONS

In this thesis, we have provided two concrete examples of biomedical research where large-scale scientific computing can play a significant role in advancing the field. First, we examined different phenotypes of ciliary motion. Ciliary beat pattern is of particular interest in diagnosing abnormal respiratory conditions, making an objective method for quantifying ciliary behavior clinically compelling. We developed a novel automated classification framework capable of recapitulating the current state-of-the-art for abnormal phenotype identification. This led to exploring the generally held notion that cilia express a spectrum of beat patterns; in this case, however, we sought to quantify this hypothesis using unsupervised clustering techniques. Our classification framework revealed the efficacy of representing ciliary motion using the autoregressive parameters of the dynamic texture, and in particular of deriving these parameters from rotation data. However, the significant variability in the data necessitated the capture and analysis large amounts of information. Spectral graph techniques were an obvious choice for analyzing anisotropic data in high-dimensional space; however, the bottleneck of the eigensolver lent credence to the development of more scalable analytics.

Second, we explored automated enumeration of the perceptual olfactory space from physiochemical properties of odorants. Unlike most other sensory modalities, it is somewhat unclear what the major perceptual dimensions are in olfaction; indeed, it is also unclear how neural stimuli are interpreted to initiate perception of an odor. While an all-encompassing model of olfaction remains elusive, we demonstrated the utility of pairwise similarity information, derived either from psychophysical experiments or computational methods, for learning a generic metric that can be embedded within a perceptual prediction framework to enhance its performance. We found that the low-dimensional embedding of the odorants in this similarity space was much more informative than the equivalent embedding using only

psychophysics or physiochemical information. This laid the groundwork for a large-scale predictive framework, using the small amount of perceptual information we have learned from this study to establish similarity with and propagate odor percepts to otherwise unlabeled odorants.

In both cases, the latter in particular, scalable analysis techniques were needed. Finding the eigenvectors and eigenvalues of a dynamic system can yield critical information about its evolution or information propagation; however, this is typically an expensive computation to perform. Iterative and hierarchical methods abound for closely approximating the solution with speed and accuracy tradeoffs. We focused on hierarchical methods due to their theoretical runtime and performance guarantees. Specifically, we implemented the first-ever distributed hierarchical eigensolver, embedded within a large-scale scientific computing platform called Draenor. We found that while overall performance was not yet comparable to other state-of-the-art distributed iterative eigensolvers such as Lanczos or stochastic SVD, hierarchical solvers have a great deal of promise both in theory and practice.

6.1 FUTURE DIRECTIONS

There are many interesting directions to take this work, both technical and theoretical.

6.1.1 Ciliary motion analysis

We demonstrated the expressive power of autoregressive coefficients for identifying different motion phenotypes. An intrinsic property of AR models is that they are generative, capable of synthesizing novel sequences that adhere to certain statistical properties. This would serve as an ideal framework for creating training videos for clinicians or synthetic data for researchers that are from a particular motion phenotype or a combination of several.

The web front-end can be more fully developed into a complete application, packaging the framework in a blackbox for clinicians across the world to access and analyze their digital videos without any specialized knowledge in computer vision or machine learning.

This would open up many opportunities for collaboration and the acquisition of additional data, helping tune our models further.

We have only begun to scratch the surface of the potential myriad latent motion phenotypes. While the manifold appears as a smooth spectrum with few discernible transition points from one motion phenotype to the next, additional data will likely tease out these boundaries. Furthermore, incorporating higher-order AR systems into a generalized Martin distance framework will allow for the metric to account for significantly more complex motion.

6.1.2 Perceptual olfactory recognition

The pairwise metric we developed is significant for several reasons. First, it is novel in its generalizability to new odorants. Previous pairwise metrics were largely specific to the data from which they were derived. In this case, unobserved odorants as described by their physiochemical properties can be examined and a perceptual category predicted. Second, it is novel in its generalizability to similarity information. While the metric we proposed was technically built from computational pairwise similarity, the original similarity information from which the perceptual categorization was derived was from psychophysical experiments. In this way, both computational methods and manual psychophysical experiments can yield similarity information to be incorporated into the metric.

Third, and most significantly, this pairwise metric provides valuable similarity information for predicting the perceptual category of unobserved odorants. We derived the machinery necessary for a semi-supervised framework, in which we use a graph kernel to propagate the perceptual labels of known odorants to those whose percepts are unknown. This process has no theoretical bound; however, acquiring the physiochemical properties of a large enough corpus, e.g., PubChem, has proven difficult. This would be the first area of future work to pursue.

6.1.3 Large-scale scientific computing

There are many open-source large-scale scientific computing frameworks, with more constantly being released. However, we found the theoretical guarantees and empirical performance of hierarchical eigensolvers to be of particular interest for the purpose of further streamlining large-scale graph analytics. Given the novelty of a distributed multigrid eigensolver, we implemented geometric and algebraic multigrid methods within the Apache Spark framework. Empirical performance was not particularly impressive compared to other more mature distributed eigensolvers; however, this was but a first pass, with much room for improvement.

The distributed geometric solver achieved impressive performance for homogeneous problems such as image segmentation. However, the algebraic solver hit a bottleneck in the sparsification process. Specifically, identifying triangles in a graph and breaking them up poses challenging theoretical and practical concerns. Even in sparsely connected graphs, triangles can occur in very dense clusters, making iterative identification all but infeasible. This process is easily parallelized, however triangles in networks can and often do share edges, precluding eager updates of the underlying graph. There are both theoretical and practical solutions for this problem: triangles could be sampled according to a coarse distribution of edges that could be computed cheaply, or a graph-based distributed framework could be used that specifically allows for concurrent updates between connected nodes.

Along those lines, there are myriad frameworks whose advantages and drawbacks must be considered for implementation of Draenor. While Apache Spark represents one of the fastest-growing large-scale scientific computing frameworks available, other distributed platforms warrant testing. Apache Giraph, Apache GraphX, and GraphLab each use vertex-based computing, propagating changes across a graph structure rather than a matrix abstraction. While this can make certain operations such as solving linear equations more difficult in general, certain graph-specific applications such as triangle-finding can be made substantially simpler. Furthermore, we used only the Python API for Apache Spark, which lags significantly behind the Scala API in terms of functionality. Adopting the latter may yield significant performance gains with little changes to the core framework.

More generally, these techniques and frameworks will only become more relevant and necessary to further biomedical breakthroughs. The trends are very clear: storage and processing will continue to become cheaper and more abundant, incentivizing the casting of extremely broad nets to capture as much data as possible for downstream analysis. Consequently, scalable methods for analysis will be absolutely essential. Here we have demonstrated two relevant biomedical use cases, both of which would greatly benefit from enhanced data curation and analysis methods.

BIBLIOGRAPHY

- [1] A. Ramanathan, A. Savol, V. Burger, S. Quinn, P. K. Agarwal, and C. Chennubhotla, “Statistical inference for big data problems in molecular biophysics,” Oak Ridge National Laboratory (ORNL); Center for Computational Sciences, Tech. Rep., 2012.
- [2] V. Emilsson, G. Thorleifsson, B. Zhang, A. S. Leonardson, F. Zink, J. Zhu, S. Carlson, A. Helgason, G. B. Walters, S. Gunnarsdottir *et al.*, “Genetics of gene expression and its effect on disease,” *Nature*, vol. 452, no. 7186, pp. 423–428, 2008.
- [3] S. Mavandadi, S. Dimitrov, S. Feng, F. Yu, U. Sikora, O. Yaglidere, S. Padmanabhan, K. Nielsen, and A. Ozcan, “Distributed medical image analysis and diagnosis through crowd-sourced games: a malaria case study,” *PLoS One*, vol. 7, no. 5, p. e37245, 2012.
- [4] S. Quinn, R. Francis, C. Lo, and C. Chennubhotla, “Novel use of differential image velocity invariants to categorize ciliary motion defects,” in *Biomedical Sciences and Engineering Conference (BSEC)*. IEEE, 2011, pp. 1–4.
- [5] J. B. Castro, A. Ramanathan, and C. S. Chennubhotla, “Categorical dimensions of human odor descriptor space revealed by non-negative matrix factorization,” *PloS one*, vol. 8, no. 9, p. e73289, 2013.
- [6] D. Howe, M. Costanzo, P. Fey, T. Gojobori, L. Hannick, W. Hide, D. P. Hill, R. Kania, M. Schaeffer, S. St Pierre *et al.*, “Big data: The future of biocuration,” *Nature*, vol. 455, no. 7209, pp. 47–50, 2008.
- [7] E. Schadt, M. Linderman, J. Sorenson, L. Lee, and G. Nolan, “Computational solutions to large-scale data management and analysis,” *Nature Reviews Genetics*, vol. 11, no. 9, pp. 647–657, 2010.
- [8] R. R. Schaller, “Moore’s law: past, present and future,” *Spectrum, IEEE*, vol. 34, no. 6, pp. 52–59, 1997.
- [9] A. Sboner, X. J. Mu, D. Greenbaum, R. K. Auerbach, and M. B. Gerstein, “The real cost of sequencing: higher than you think!” *Genome biology*, vol. 12, no. 8, p. 125, 2011.

- [10] L. Page, S. Brin, R. Motwani, and T. Winograd, “The pagerank citation ranking: Bringing order to the web.” Stanford InfoLab, Tech. Rep., 1999.
- [11] J. Napper and P. Bientinesi, “Can cloud computing reach the top500?” in *Proceedings of the combined workshops on UnConventional high performance computing workshop plus memory access workshop*. ACM, 2009, pp. 17–20.
- [12] P. Arbenz, D. Kressner, and D.-M. E. Zürich, “Lecture notes on solving large scale eigenvalue problems,” *D-MATH, EHT Zurich*, 2012.
- [13] Y. Saad, *Numerical methods for large eigenvalue problems*. SIAM, 1992, vol. 158.
- [14] F. R. Chung, *Spectral graph theory*. American Mathematical Soc., 1997, vol. 92.
- [15] U. Von Luxburg, “A tutorial on spectral clustering,” *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.
- [16] M. Meila and J. Shi, “A random walks view of spectral segmentation,” in *8th International Conference on AI and Statistics*. Citeseer, 2001.
- [17] A. Y. Ng, M. I. Jordan, Y. Weiss *et al.*, “On spectral clustering: Analysis and an algorithm,” *Advances in neural information processing systems*, vol. 2, pp. 849–856, 2002.
- [18] G. Strang, “Introduction to linear algebra,” *Cambridge Publication*, 2003.
- [19] A. Cayley, “On a new auxiliary equation in the theory of equations of the fifth order,” *Philosophical Transactions of the Royal Society of London*, pp. 263–276, 1861.
- [20] J. Butler, “Improving coarsening and interpolation for algebraic multigrid,” Ph.D. dissertation, University of Waterloo, 2006.
- [21] W. E. Arnoldi, “The principle of minimized iterations in the solution of the matrix eigenvalue problem,” *Quart. Appl. Math.*, vol. 9, no. 1, pp. 17–29, 1951.
- [22] J. Kuczynski and H. Wozniakowski, “Estimating the largest eigenvalue by the power and lanczos algorithms with a random start,” *SIAM journal on matrix analysis and applications*, vol. 13, no. 4, pp. 1094–1122, 1992.
- [23] A. V. Knyazev, “Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method,” *SIAM journal on scientific computing*, vol. 23, no. 2, pp. 517–541, 2001.
- [24] S. A. Kharchenko *et al.*, “Eigenvalue translation based preconditioners for the gmres (k) method,” *Numerical linear algebra with applications*, vol. 2, no. 1, pp. 51–77, 1995.
- [25] F. A. Dul, “Minres and minerr are better than symmlq in eigenpair computations,” *SIAM Journal on Scientific Computing*, vol. 19, no. 6, pp. 1767–1782, 1998.

- [26] C. C. Paige, B. N. Parlett, and H. A. van der Vorst, “Approximate solutions and eigenvalue bounds from krylov subspaces,” *Numerical linear algebra with applications*, vol. 2, no. 2, pp. 115–133, 1995.
- [27] W. Chen and B. Poirier, “Parallel implementation of efficient preconditioned linear solver for grid-based applications in chemical physics. ii: Qmr linear solver,” *Journal of Computational Physics*, vol. 219, no. 1, pp. 198–209, 2006.
- [28] T. A. Davis, *Direct methods for sparse linear systems*. SIAM, 2006.
- [29] U. Trottenberg, C. W. Oosterlee, and A. Schuller, *Multigrid*. Access Online via Elsevier, 2000.
- [30] C. Chevalier and I. Safro, “Comparison of coarsening schemes for multilevel graph partitioning,” in *Learning and Intelligent Optimization*. Springer, 2009, pp. 191–205.
- [31] H. d. Sterck, V. E. Henson, G. Sanders *et al.*, “Multilevel aggregation methods for small-world graphs with application to random-walk ranking,” *Computing and Informatics*, vol. 30, no. 2, pp. 225–246, 2012.
- [32] K. Miller, “Algebraic multigrid for markov chains and tensor decomposition,” Ph.D. dissertation, University of Waterloo, 2012.
- [33] P. Arbenz, U. L. Hetmaniuk, R. B. Lehoucq, and R. S. Tuminaro, “A comparison of eigensolvers for large-scale 3d modal analysis using amg-preconditioned iterative methods,” *International Journal for Numerical Methods in Engineering*, vol. 64, no. 2, pp. 204–236, 2005.
- [34] D. Krishnan and R. Szeliski, “Multigrid and multilevel preconditioners for computational photography,” in *ACM Transactions on Graphics (TOG)*, vol. 30, no. 6. ACM, 2011, p. 177.
- [35] D. Krishnan, R. Fattal, and R. Szeliski, “Efficient preconditioning of laplacian matrices for computer graphics,” *ACM Transactions on Graphics (TOG)*, vol. 32, no. 4, p. 142, 2013.
- [36] R. Alcouffe, A. Brandt, J. Dendy, Jr, and J. Painter, “The multi-grid method for the diffusion equation with strongly discontinuous coefficients,” *SIAM Journal on Scientific and Statistical Computing*, vol. 2, no. 4, pp. 430–454, 1981.
- [37] A. Brandt, “Algebraic multigrid theory: The symmetric case,” *Applied mathematics and computation*, vol. 19, no. 1, pp. 23–56, 1986.
- [38] M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, and J. Ruge, “Adaptive algebraic multigrid,” *SIAM Journal on Scientific Computing*, vol. 27, no. 4, pp. 1261–1286, 2006.

- [39] A. Brandt and D. Ron, “Multigrid solvers and multilevel optimization strategies,” in *Multilevel optimization in VLSICAD*. Springer, 2003, pp. 1–69.
- [40] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, “Multiscale combinatorial grouping,” in *CVPR*, 2014.
- [41] I. Koutis and G. L. Miller, “A linear work, $O(n^{1/6})$ time, parallel algorithm for solving planar laplacians,” in *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1002–1011.
- [42] G. E. Blelloch, A. Gupta, I. Koutis, G. L. Miller, R. Peng, and K. Tangwongsan, “Near linear-work parallel sdd solvers, low-diameter decomposition, and low-stretch subgraphs,” in *Proceedings of the 23rd ACM symposium on Parallelism in algorithms and architectures*. ACM, 2011, pp. 13–22.
- [43] E. Romero and J. E. Roman, “Computing subdominant unstable modes of turbulent plasma with a parallel jacobi–davidson eigensolver,” *Concurrency and Computation: Practice and Experience*, vol. 23, no. 17, pp. 2179–2191, 2011.
- [44] V. Hernández, J. E. Román, and A. Tomás, “Parallel arnoldi eigensolvers with enhanced scalability via global communications rearrangement,” *Parallel Computing*, vol. 33, no. 7, pp. 521–540, 2007.
- [45] V. Hernandez, J. E. Roman, and V. Vidal, “Slepc: A scalable and flexible toolkit for the solution of eigenvalue problems,” *ACM Transactions on Mathematical Software (TOMS)*, vol. 31, no. 3, pp. 351–362, 2005.
- [46] U. Kang, B. Meeder, and C. Faloutsos, “Spectral analysis for billion-scale graphs: Discoveries and implementation,” in *Advances in Knowledge Discovery and Data Mining*. Springer, 2011, pp. 13–25.
- [47] U. Kang, C. E. Tsourakakis, and C. Faloutsos, “Pegasus: A peta-scale graph mining system implementation and observations,” in *Data Mining, 2009. ICDM’09. Ninth IEEE International Conference on*. IEEE, 2009, pp. 229–238.
- [48] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: cluster computing with working sets,” in *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing*, 2010, pp. 10–10.
- [49] Y. Koren, “Factorization meets the neighborhood: a multifaceted collaborative filtering model,” in *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2008, pp. 426–434.
- [50] S. Owen, R. Anil, T. Dunning, and E. Friedman, *Mahout in action*. Manning, 2011.

- [51] Y. Low, D. Bickson, J. Gonzalez, C. Guestrin, A. Kyrola, and J. M. Hellerstein, “Distributed graphlab: a framework for machine learning and data mining in the cloud,” *Proceedings of the VLDB Endowment*, vol. 5, no. 8, pp. 716–727, 2012.
- [52] M. Chilvers, M. McKean, A. Rutman, B. Myint, M. Silverman, and C. O’Callaghan, “The effects of coronavirus on human nasal ciliated respiratory epithelium,” *European Respiratory Journal*, vol. 18, no. 6, pp. 965–970, 2001.
- [53] B. Thomas, A. Rutman, R. A. Hirst, P. Haldar, A. J. Wardlaw, J. Bankart, C. E. Brightling, and C. O’Callaghan, “Ciliary dysfunction and ultrastructural abnormalities are features of severe asthma,” *Journal of Allergy and Clinical Immunology*, vol. 126, no. 4, pp. 722–729, 2010.
- [54] C. O’Callaghan, M. Chilvers, C. Hogg, A. Bush, and J. Lucas, “Diagnosing primary ciliary dyskinesia,” *Thorax*, vol. 62, no. 8, pp. 656–657, 2007.
- [55] M. W. Leigh, J. E. Pittman, J. L. Carson, T. W. Ferkol, S. D. Dell, S. D. Davis, M. R. Knowles, and M. A. Zariwala, “Clinical and genetic aspects of primary ciliary dyskinesia/kartagener syndrome,” *Genetics in Medicine*, vol. 11, no. 7, pp. 473–487, 2009.
- [56] J. McGrath, S. Somlo, S. Makova, X. Tian, and M. Brueckner, “Two populations of node monocilia initiate left-right asymmetry in the mouse,” *Cell*, vol. 114, no. 1, pp. 61–73, 2003.
- [57] A. Becker-Heck, I. E. Zohn, N. Okabe, A. Pollock, K. B. Lenhart, J. Sullivan-Brown, J. McSheene, N. T. Loges, H. Olbrich, K. Haeffner *et al.*, “The coiled-coil domain containing protein ccdc40 is essential for motile cilia function and left-right axis formation,” *Nature Genetics*, vol. 43, no. 1, pp. 79–84, 2010.
- [58] A. S. Aylsworth, “Clinical aspects of defects in the determination of laterality,” *American Journal of Medical Genetics*, vol. 101, no. 4, pp. 345–355, 2001.
- [59] M. Swisher, R. Jonas, X. Tian, E. S. Lee, C. W. Lo, and L. Leatherbury, “Increased postoperative and respiratory complications in patients with congenital heart disease associated with heterotaxy,” *The Journal of Thoracic and Cardiovascular Surgery*, vol. 141, no. 3, pp. 637–644, 2011.
- [60] N. Nakhleh, R. Francis, R. A. Giese, X. Tian, Y. Li, M. A. Zariwala, H. Yagi, O. Khalifa, S. Kureshi, B. Chatterjee *et al.*, “High prevalence of respiratory ciliary dysfunction in congenital heart disease patients with heterotaxy,” *Circulation*, vol. 125, no. 18, pp. 2232–2242, 2012.
- [61] B. Harden, X. Tian, R. Giese, N. Nakhleh, S. Kureshi, R. Francis, S. Hanumanthaiah, Y. Li, M. Swisher, K. Kuehl, I. Sami, K. Olivier, R. Jonas, C. W. Lo, and L. Leatherbury, “Increased postoperative respiratory complications in heterotaxy congenital heart disease patients with respiratory ciliary dysfunction,” *The Journal of*

Thoracic and Cardiovascular Surgery, vol. Available online 22 July 2013, 2013. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0022522313006697>

- [62] M. Zahid, O. Khalifa, W. Devine, C. Yau, R. Francis, D. M. Lee, K. Tobita, P. Wearden, L. Leatherbury, S. Webber, and C. W. Lo, “Airway ciliary dysfunction in patients with transposition of the great arteries,” in *Circulation*, vol. 126, 2012, p. A15746.
- [63] A. S. Garrod, M. Zahid, X. Tian, R. J. Francis, O. Khalifa, W. Devine, G. C. Gabriel, L. Leatherbury, and C. W. Lo, “Airway ciliary dysfunction and sinopulmonary symptoms in congenital heart disease patients,” *Annals of the American Thoracic Society*, no. ja, 2014.
- [64] J. F. Papon, I. Perrault, A. Coste, B. Louis, X. Gerard, S. Hanein, L. Fares-Taie, S. Gerber, S. Defoort-Dhellemmes, A. M. Vojtek *et al.*, “Abnormal respiratory cilia in non-syndromic leber congenital amaurosis with cep290 mutations,” *Journal of medical genetics*, vol. 47, no. 12, pp. 829–834, 2010.
- [65] S. Dimova, F. Maes, M. E. Brewster, M. Jorissen, M. Noppe, and P. Augustijns, “High-speed digital imaging method for ciliary beat frequency measurement,” *Journal of Pharmacy and Pharmacology*, vol. 57, no. 4, pp. 521–526, 2005.
- [66] M. A. Olm, J. E. Kögler, M. Macchione, A. Shoemark, P. H. Saldiva, and J. C. Rodrigues, “Primary ciliary dyskinesia: evaluation using cilia beat frequency assessment via spectral analysis of digital microscopy images,” *Journal of Applied Physiology*, vol. 111, no. 1, pp. 295–302, 2011.
- [67] G. Mantovani, M. Pifferi, and G. Vozzi, “Automated software for analysis of ciliary beat frequency and metachronal wave orientation in primary ciliary dyskinesia,” *European Archives of Oto-Rhino-Laryngology*, vol. 267, no. 6, pp. 897–902, 2010.
- [68] C. O’Callaghan, K. Sikand, M. Chilvers *et al.*, “Analysis of ependymal ciliary beat pattern and beat frequency using high speed imaging: comparison with the photomultiplier and photodiode methods,” *Cilia*, vol. 1, no. 1, p. 8, 2012.
- [69] W. A. Stannard, M. A. Chilvers, A. R. Rutman, C. D. Williams, and C. O’Callaghan, “Diagnostic testing of patients suspected of primary ciliary dyskinesia,” *American Journal of Respiratory and Critical Care Medicine*, vol. 181, no. 4, pp. 307–314, 2010.
- [70] B. Thomas, A. Rutman, and C. O’Callaghan, “Disrupted ciliated epithelium shows slower ciliary beat frequency and increased dyskinesia,” *European Respiratory Journal*, vol. 34, no. 2, pp. 401–404, 2009.
- [71] C. M. Smith, R. A. Hirst, M. J. Bankart, D. W. Jones, A. J. Easton, P. W. Andrew, and C. O’Callaghan, “Cooling of cilia allows functional analysis of the beat pattern for diagnostic testing of temperature and ciliary function,” *CHEST Journal*, vol. 140, no. 1, pp. 186–190, 2011.

- [72] C. Clary-Meinesz, J. Cosson, P. Huitorel, and B. Blaive, “Temperature effect on the ciliary beat frequency of human nasal and tracheal ciliated cells,” *Biology of the Cell*, vol. 76, no. 3, pp. 335–338, 1992.
- [73] M. Salathe, “Regulation of mammalian ciliary beating,” *Annu. Rev. Physiol.*, vol. 69, pp. 401–422, 2007.
- [74] J. Raidt, J. Wallmeier, R. Hjeij, J. G. Onnebrink, P. Pennekamp, N. T. Loges, H. Olbrich, K. Häffner, G. W. Dougherty, H. Omran *et al.*, “Ciliary beat pattern and frequency in genetic variants of primary ciliary dyskinesia,” *European Respiratory Journal*, pp. erj00 520–2014, 2014.
- [75] J. MacCormick, I. Robb, T. Kovesi, and B. Carpenter, “Optimal biopsy techniques in the diagnosis of primary ciliary dyskinesia,” *Journal of otolaryngology*, vol. 31, no. 1, pp. 13–17, 2002.
- [76] J. Papon, A. Coste, F. Roudot-Thoraval, M. Boucherat, G. Roger, A. Tamalet, A. Vojtek, S. Amselem, and E. Escudier, “A 20-year experience of electron microscopy in the diagnosis of primary ciliary dyskinesia,” *European Respiratory Journal*, vol. 35, no. 5, pp. 1057–1063, 2010.
- [77] T. P. Plesec, A. Ruiz, J. T. McMahon, and R. A. Prayson, “Ultrastructural abnormalities of respiratory cilia: a 25-year experience,” *Archives of pathology & laboratory medicine*, vol. 132, no. 11, pp. 1786–1791, 2008.
- [78] F. J. Massey Jr, “The kolmogorov-smirnov test for goodness of fit,” *Journal of the American Statistical Association*, vol. 46, no. 253, pp. 68–78, 1951.
- [79] J.-Y. Bouguet, “Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm,” *Intel Corporation*, vol. 5, 2001.
- [80] D. Sun, S. Roth, and M. J. Black, “Secrets of optical flow estimation and their principles,” in *Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2010)*, 2010, pp. 2432–2439.
- [81] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [82] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto, “Dynamic texture recognition,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001)*, vol. 2, 2001, pp. II–58.
- [83] G. Doretto, A. Chiuso, Y. N. Wu, and S. Soatto, “Dynamic textures,” *International Journal of Computer Vision*, vol. 51, no. 2, pp. 91–109, 2003.

- [84] J. Huang, X. Huang, D. Metaxas, and L. Axel, “Dynamic texture based heart localization and segmentation in 4-d cardiac images,” in *Biomedical Imaging: From Nano to Macro, 2007. ISBI 2007. 4th IEEE International Symposium on.* IEEE, 2007, pp. 852–855.
- [85] N. Brieu, N. Navab, J. Serbanovic-Canic, W. H. Ouwehand, D. L. Stemple, A. Cvejc, and M. Groher, “Image-based characterization of thrombus formation in time-lapse dic microscopy,” *Medical image analysis*, vol. 16, no. 4, pp. 915–931, 2012.
- [86] M. Hyndman, A. D. Jepson, and D. J. Fleet, “Higher-order autoregressive models for dynamic textures.” in *British Machine Vision Conference (BMVC) 2007*, 2007, pp. 1–10.
- [87] S. F. Te Pas, A. M. Kappers, and J. J. Koenderink, “Detection of first-order structure in optic flow fields,” *Vision Research*, vol. 36, no. 2, pp. 259–270, 1996.
- [88] B. Brown, “Invariant properties of the motion parallax field due to the movement of rigid bodies relative to an observer,” *Optica acta*, vol. 22, pp. 773–791, 1975.
- [89] G. Zhao and M. Pietikainen, “Dynamic texture recognition using local binary patterns with an application to facial expressions,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 915–928, 2007.
- [90] Z. Lu, W. Xie, J. Pei, and J. Huang, “Dynamic texture recognition by spatio-temporal multiresolution histograms,” in *Seventh IEEE Workshop on Application of Computer Vision (WACV/MOTIONS’05)*, vol. 2, 2005, pp. 241–246.
- [91] J. Chen, G. Zhao, M. Salo, E. Rahtu, and M. Pietikainen, “Automatic dynamic texture segmentation using local descriptors and optical flow,” *IEEE Transactions on Image Processing*, vol. 22, pp. 326–339, 2013.
- [92] S. C. Fu and P. Kovesi, “Robust extraction of optic flow differentials for surface reconstruction,” in *Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on.* IEEE, 2010, pp. 468–473.
- [93] A. Ravichandran, R. Chaudhry, and R. Vidal, “View-invariant dynamic texture recognition using a bag of dynamical systems,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009, pp. 1651–1657.
- [94] A. V. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-time Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1999.
- [95] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [96] M.-E. Nilsback and A. Zisserman, “Automated flower classification over a large number of classes,” in *Sixth Indian Conference on Computer Vision, Graphics & Image Processing (ICVGIP’08)*, 2008, pp. 722–729.

- [97] Y. Kluger, R. Basri, J. Chang, and M. Gerstein, “Spectral biclustering of microarray data: coclustering genes and conditions,” *Genome Res*, vol. 13, no. 4, pp. 703–716, 2003.
- [98] A. Arzi and N. Sobel, “Olfactory perception as a compass for olfactory neural maps,” *Trends Cogn. Sci. (Regul. Ed.)*, vol. 15, no. 11, pp. 537–545, Nov 2011.
- [99] R. B. Lotto and D. Purves, “A rationale for the structure of color space,” *Trends Neurosci.*, vol. 25, no. 2, pp. 84–88, Feb 2002.
- [100] P. Lennie and M. D’Zmura, “Mechanisms of color vision,” *Crit Rev Neurobiol*, vol. 3, no. 4, pp. 333–400, 1988.
- [101] H. Lapid, S. Shushan, A. Plotkin, H. Voet, Y. Roth, T. Hummel, E. Schneidman, and N. Sobel, “Neural activity at the human olfactory epithelium reflects olfactory perception,” *Nat. Neurosci.*, vol. 14, no. 11, pp. 1455–1461, Nov 2011.
- [102] R. Haddad, T. Weiss, R. Khan, B. Nadler, N. Mandairon, M. Bensafi, E. Schneidman, and N. Sobel, “Global features of neural activity in the olfactory system form a parallel code that predicts olfactory behavior and perception,” *J. Neurosci.*, vol. 30, no. 27, pp. 9017–9026, Jul 2010.
- [103] H. Henning, *Der Geruch*. Leipzig: JA Barth, 1916.
- [104] J. E. Amoore, “Evidence for the chemical olfactory code in man,” *Ann. N. Y. Acad. Sci.*, vol. 237, pp. 137–143, Sep 1974.
- [105] —, “Specific anosmia: a clue to the olfactory code,” *Nature*, vol. 214, no. 5093, pp. 1095–1098, Jun 1967.
- [106] A. G. Khan, M. Thattai, and U. S. Bhalla, “Odor representations in the rat olfactory bulb change smoothly with morphing stimuli,” *Neuron*, vol. 57, no. 4, pp. 571–585, Feb 2008.
- [107] J. A. Gottfried and K. N. Wu, “Perceptual and neural pliability of odor objects,” *Ann. N. Y. Acad. Sci.*, vol. 1170, pp. 324–332, Jul 2009.
- [108] R. C. Araneda, A. D. Kini, and S. Firestein, “The molecular receptive range of an odorant receptor,” *Nature neuroscience*, vol. 3, no. 12, pp. 1248–1255, 2000.
- [109] C. Sell, “On the unpredictability of odor,” *Angewandte Chemie International Edition*, vol. 45, no. 38, pp. 6254–6261, 2006.
- [110] R. M. Khan, C. H. Luk, A. Flinker, A. Aggarwal, H. Lapid, R. Haddad, and N. Sobel, “Predicting odor pleasantness from odorant structure: pleasantness as a reflection of the physical world,” *J. Neurosci.*, vol. 27, no. 37, pp. 10 015–10 023, Sep 2007.

- [111] A. A. Koulakov, B. E. Kolterman, A. G. Enikolopov, and D. Rinberg, “In search of the structure of human olfactory space,” *Front Syst Neurosci*, vol. 5, p. 65, 2011.
- [112] L. Secundo, K. Snitz, and N. Sobel, “The perceptual logic of smell,” *Current opinion in neurobiology*, vol. 25, pp. 107–115, 2014.
- [113] Y. Yeshurun and N. Sobel, “An odor is not worth a thousand words: from multidimensional odors to unidimensional odor objects,” *Annual review of psychology*, vol. 61, pp. 219–241, 2010.
- [114] R. S. Herz, “The effect of verbal context on olfactory perception.” *Journal of Experimental Psychology: General*, vol. 132, no. 4, p. 595, 2003.
- [115] W. Li, E. Luxenberg, T. Parrish, and J. A. Gottfried, “Learning to smell the roses: experience-dependent neural plasticity in human piriform and orbitofrontal cortices,” *Neuron*, vol. 52, no. 6, pp. 1097–1108, 2006.
- [116] E. P. Xing, M. I. Jordan, S. Russell, and A. Y. Ng, “Distance metric learning with application to clustering with side-information,” in *Advances in neural information processing systems*, 2002, pp. 505–512.
- [117] K. Snitz, A. Yablonka, T. Weiss, I. Frumin, R. M. Khan, and N. Sobel, “Predicting odor perceptual similarity from odor structure,” *PLoS computational biology*, vol. 9, no. 9, p. e1003184, 2013.
- [118] R. Haddad, R. Khan, Y. K. Takahashi, K. Mori, D. Harel, and N. Sobel, “A metric for odorant comparison,” *Nature methods*, vol. 5, no. 5, pp. 425–429, 2008.
- [119] A. Dravnieks, “Odor quality: semantically generated multidimensional profiles are stable,” *Science*, vol. 218, no. 4574, pp. 799–801, Nov 1982.
- [120] A. Dravnieks, *Atlas of Odor Character Profiles*. Philadelphia: ASTM Data Series ed. ASTM Committee E-18 on Sensory Evaluation of Materials and Products. Section E-18.04.12 on Odor Profiling, 1985.
- [121] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetrics*, vol. 5, pp. 111–126, 1994.
- [122] P. Paatero, “Least squares formulation of robust non-negative factor analysis,” *Chemo-metrics Int. Lab. Sys.*, vol. 37, pp. 23–35, 1997.
- [123] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, Oct 1999.
- [124] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons, “Algorithms and Applications for Approximate Nonnegative Matrix Factorization,” *Computational Statistics & Data Analysis*, vol. 52, no. 1, pp. 155–173, 2007.

- [125] W. Xu, X. Liu, and Y. Gong, “Document clustering based on non-negative matrix factorization,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, ser. SIGIR ’03, 2003, pp. 267–273.
- [126] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, “Gene selection for cancer classification using support vector machines,” *Machine learning*, vol. 46, no. 1-3, pp. 389–422, 2002.
- [127] A. W. Whitney, “A direct method of nonparametric measurement selection,” *Computers, IEEE Transactions on*, vol. 100, no. 9, pp. 1100–1103, 1971.
- [128] P. Kain, S. M. Boyle, S. K. Tharadra, T. Guda, C. Pham, A. Dahanukar, and A. Ray, “Odour receptors and neurons for deet and new insect repellents,” *Nature*, 2013.
- [129] S. M. Boyle, S. McInally, and A. Ray, “Expanding the olfactory code by in silico decoding of odor-receptor chemical space,” *eLife*, vol. 2, 2013.
- [130] C. Bushdid, M. Magnasco, L. Vosshall, and A. Keller, “Humans can discriminate more than 1 trillion olfactory stimuli,” *Science*, vol. 343, no. 6177, pp. 1370–1372, 2014.
- [131] M. Meister, “Can humans really discriminate 1 trillion odors?” *arXiv preprint arXiv:1411.0165*, 2014.
- [132] K. R. Varshney and L. R. Varshney, “Olfactory signals and systems,” *arXiv preprint arXiv:1410.4865*, 2014.
- [133] X. Zhu and A. B. Goldberg, “Introduction to semi-supervised learning,” *Synthesis lectures on artificial intelligence and machine learning*, vol. 3, no. 1, pp. 1–130, 2009.
- [134] T. Zhang, “An introduction to support vector machines and other kernel-based learning methods,” *AI Magazine*, vol. 22, no. 2, p. 103, 2001.
- [135] R. Fergus, Y. Weiss, and A. Torralba, “Semi-supervised learning in gigantic image collections,” in *Advances in neural information processing systems*, 2009, pp. 522–530.
- [136] H. Kashima, K. Tsuda, and A. Inokuchi, “Marginalized kernels between labeled graphs,” in *ICML*, vol. 3, 2003, pp. 321–328.
- [137] I. V. Tetko, J. Gasteiger, R. Todeschini, A. Mauri, D. Livingstone, P. Ertl, V. A. Palyulin, E. V. Radchenko, N. S. Zefirov, A. S. Makarenko *et al.*, “Virtual computational chemistry laboratory—design and description,” *Journal of computer-aided molecular design*, vol. 19, no. 6, pp. 453–463, 2005.
- [138] C. Rasmussen and C. Williams, “Gaussian processes for machine learning, ser. adaptive computation and machine learning,” *MIT Press*, vol. 10, pp. 15–20, 2006.

- [139] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.
- [140] F. Lin and W. W. Cohen, “Semi-supervised classification of network data using very few labels,” in *Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on*. IEEE, 2010, pp. 192–199.
- [141] X. Zhu, Z. Ghahramani, and J. Lafferty, “Semi-supervised learning using gaussian fields and harmonic functions,” in *ICML*, vol. 3, 2003, pp. 912–919.
- [142] S. A. Macskassy and F. Provost, “Classification in networked data: A toolkit and a univariate case study,” *The Journal of Machine Learning Research*, vol. 8, pp. 935–983, 2007.
- [143] P. P. Talukdar and K. Crammer, “New regularized algorithms for transductive learning,” in *Machine Learning and Knowledge Discovery in Databases*. Springer, 2009, pp. 442–457.
- [144] A. Ramanathan, L. L. Pullum, C. A. Steed, S. P. Quinn, C. S. Chennubhotla, and T. Parker, “Integrating heterogeneous healthcare datasets and visual analytics for disease bio-surveillance and dynamics,” in *IEEE Workshop on Interactive Visual Text Analytics (Atlanta, GA, 2013)*.
- [145] J. Dean and S. Ghemawat, “Mapreduce: simplified data processing on large clusters,” *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.
- [146] T. White, *Hadoop: the definitive guide*. O’Reilly, 2012.
- [147] L. G. Valiant, “A bridging model for parallel computation,” *Communications of the ACM*, vol. 33, no. 8, pp. 103–111, 1990.
- [148] T. Kajdanowicz, W. Indyk, P. Kazienko, and J. Kukul, “Comparison of the efficiency of mapreduce and bulk synchronous parallel approaches to large network processing,” in *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*. IEEE, 2012, pp. 218–225.
- [149] J. E. Gonzalez, Y. Low, H. Gu, D. Bickson, and C. Guestrin, “Powergraph: Distributed graph-parallel computation on natural graphs,” in *Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2012, pp. 17–30.
- [150] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, “Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing,” in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. USENIX Association, 2012, pp. 2–2.

- [151] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker, and I. Stoica, “Discretized streams: A fault-tolerant model for scalable stream processing,” DTIC Document, Tech. Rep., 2012.
- [152] I. Koutis, G. L. Miller, and R. Peng, “A fast solver for a class of linear systems,” *Communications of the ACM*, vol. 55, no. 10, pp. 99–107, 2012.