UNIVERSITY OF CALIFORNIA
RIVERSIDE

Microfluidic Tools for Precise Temperature Measurement and Chemical Analysis

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Bioengineering

by

Brittney Aquaila McKenzie

June 2019

The Dissertation of Brittney Aquaila McKenzie is approved:

Committee Chairperson

University of California, Riverside

Acknowledgments

I am grateful to Dr. Grover for his guidance and support, and to my colleagues from the Grover Lab for their help and support.

For my parents Frank and Iona McKenzie

and my sisters Maya and LaToya McKenzie

with love and gratitude

ABSTRACT OF THE DISSERTATION

Microfluidic Tools for Precise Temperature Measurement and Chemical Analysis

by

Brittney Aquaila McKenzie

Doctor of Philosophy, Graduate Program in Bioengineering
University of California, Riverside, June 2019
Dr. William H. Grover, Chairperson

Tools for identifying substances are important in many different fields and have a wide range of applications. Conventional analytical tools and instruments such as spectroscopy and chromatography systems are exceptional in their ability to measure and analyze samples. However, the cost, size, and complexity of these instruments limit their use in important applications in resource-limited settings. While "frugal science"— science solutions that prioritize both cost and function— has made strides in expanding access to health-care and research, there is still an unmet need for more low-cost and accessible research and analytical tools. In this work, I develop simple, low-cost, and effective analytical tools on microfluidic platforms that utilize changes in the physical properties of substances to examine and analyze samples. I introduce microfluidic tools and methods to accurately measure temperature and identify substances, or adulteration of substances, in small volumes. I accomplish this work through three projects. In the first project, a *Microfluidic Thermometer*, I demonstrate a simple and low-cost technique to accurately measure temperatures in small volumes using a 3D-printed microfluidic chip. With this method, the temperature of a sample can be measured with about a quarter of a degree Celsius uncertainty. For the second project, *Chronoprints*, I demonstrate a simple and low-cost method for identifying a sample based on visualizing how the sample changes in response to a perturbation over space and time. With this technique, authentic foodstuffs are distinguished from adulterated foodstuffs, adulterated medication is identified, and occasionally-confused pharmaceutical ingredients are easily distinguished. Finally for the third project, *Enhanced Chronoprints*, I demonstrate a simple and low-cost method for identifying a sample based on visualizing changes in the interaction of a falling metal bead with the sample. The cost, simplicity, versatility, and accuracy of these techniques make them valuable analytical tools in a variety of different fields and settings.

Contents

List of Figures **xi**

1 Introduction **1**

1.1 Motivation and Significance . . . 1

1.2 Approach . . . 2

1.3 Project Overview . . . 4

2 A microfluidic thermometer: Precise temperature measurements in microliter- and nanoliter-scale volumes **7**

2.1 Introduction . . . 7

2.2 Materials and Methods . . . 12

2.2.1 Designing and fabricating microfluidic thermometer chips . . . 12

2.2.2 Preparing liquids with known freezing points . . . 13

2.2.3 Establishing a temperature gradient across the measurement region on the thermometer chip . . . 13

2.2.4 Using the microfluidic thermometer chip . . . 14

2.2.5 Modeling the microfluidic thermometer chip . . . 16

2.3 Results and Discussion . . . 17

2.3.1 Modeling isotherm and thermal isocline shape in the thermometer chip 17

2.3.2 Measuring isotherm and thermal isocline shape in the thermometer chip 20

2.3.3 Measuring temperature of an "unknown" solution using the thermometer chip . . . 25

2.3.4 Using the microfluidic thermometer at temperatures well above 0 °C . 25

2.4 Conclusions . . . 26

2.5 Acknowledgments . . . 28

2.6 Supporting Information for "A microfluidic thermometer: Precise temperature measurements in microliter- and nanoliter-scale volumes" . . . 28

2.6.1 Design of the microfluidic thermometer chip in the standard .STL format used by most 3D printers. . . . 28

2.6.2 `microfluidic_thermometer.py`: A custom Python program that automates the data analysis required when using the microfluidic thermometer chip. Used to generate Figures 2.1D and E, 2.4, and 2.5B. . . 29
2.6.3 `microfluidic_thermometer.py`: Step-by-step description of using `microfluidic_thermometer.py` to perform the analyses in Fig 2.4. . 55

3 Chronoprints: Identifying samples by visualizing how they change over space and time 78
3.1 Introduction . . . 78
3.2 Experimental Section . . . 84
3.2.1 Fabricating microfluidic thermometer chips . . . 84
3.2.2 Preparing samples . . . 84
3.2.3 Obtaining chronoprints . . . 85
3.2.4 Comparing chronoprints . . . 86
3.2.4.1 Comparing chronoprints using feature tracing . . . 86
3.2.4.2 Comparing chronoprints using image differences . . . 87
3.2.4.3 Comparing chronoprints using image hashing . . . 89
3.3 Results and discussion . . . 93
3.3.1 Identifying food fraud . . . 93
3.3.2 Detecting counterfeit medicine . . . 97
3.3.3 Identifying toxic pharmaceutical ingredients . . . 102
3.4 Conclusion . . . 103
3.5 Acknowledgements . . . 107
3.6 Supporting Information for "Chronoprints: Identifying substances by visualizing how they change over time" . . . 108
3.6.1 Thermometer_chip.dxf: Design of the microfluidic thermometer chip in DXF format, used when milling the chips used in this work on a CNC mill. . . . 108
3.6.2 Supporting information text and supplementary Figures 3.10–3.22: data from replicate chronoprint experiments, and results from different analysis methods for the experiments shown in the main text. . . . 110
3.6.2.1 Optimal lighting . . . 110
3.6.2.2 Optimal experiment duration . . . 110
3.6.2.3 Thermometer chip edge effects . . . 113
3.6.2.4 Replicate of experiment in main text Figure 3.5C . . . 113
3.6.2.5 Comparisons of chronoprint image similarity algorithms . . . 113
3.6.3 `Chronoprintgen.m`: MATLAB software for converting videos into chronoprints, used to generate all of the chronoprints shown in this work. . . 116
3.6.4 `OilChronoprintAnalysis.m`: MATLAB software for analyzing chronoprints using the *feature tracing* method, used to generate Figures 3.2 and 3.5 and supplementary Figures 3.10–3.12. . . . 130

3.6.5 `chronoprint.py` and `workup.py`: Python software for analyzing chronoprints using the *image hashing* and *image differences* methods, used to generate Figures 3.3, 3.4, and 3.6–3.8, and supplementary Figures 3.13 – 3.22. 135

4 Enhanced chronoprints: Identifying samples by visualizing changes in particle interactions 156

4.1 Introduction 156

4.2 Materials and Methods 160

4.2.1 Fabricating microfluidic thermometer chips 160

4.2.2 Preparing samples 161

4.2.3 Obtaining enhanced chronoprints 161

4.2.4 Comparing enhanced chronoprints 162

4.3 Results and Discussion 163

4.3.1 Distinguishing colorless solutions 163

4.3.2 Identifying toxic pharmaceutical ingredients 165

4.4 Conclusion 167

4.5 Acknowledgements 169

4.6 Supplementary Information for Enhanced chronoprints: Identifying samples by visualizing changes in particle interactions with sample viscosity 170

4.6.1 Design of the microfluidic thermometer chip used to engrave the chips used in this work on a CNC mill. 170

4.6.2 Replicates of experiments in Figures 4.3 and 4.4 of the main text . . . 170

4.6.3 `ChronoprintAnalysis.m`: MATLAB software for analyzing enhanced chronoprints, used to generate Figures 4.2C – 4.4 180

5 Conclusions 183

5.1 Prospective pharmaceutical applications for the demonstrated techniques . . . 183

Bibliography 185

List of Figures

1.1 Chemical analysis instruments such as the liquid chromatography - Mass Spectrometer (Thermo Scientific Exactive LC-MS) are complex, take up a lot of space, are expensive, and often require additional equipment to operate. 2

1.2 Low-cost tools improve access to healthcare and research capabilities. 3

1.3 A general phase diagram for a substance. In this work I focus on the freezing and melting properties of substances and leverage the phase changes of these samples (between the solid and liquid phases) to develop techniques for measuring temperature and identifying substances. 5

2.1 Design and operation of the 3D-printed microfluidic thermometer chip 10

2.2 Finite element analysis computer simulations of the 3D-printed microfluidic thermometer chip with all five channels filled with water 19

2.3 Finite element analysis computer simulations of the temperature profile across the five channels in the measurement region inside the microfluidic thermometer chip, with the channels filled with water (left), mineral oil (center), and toluene (right). 20

2.4 (A) To characterize isotherm shape, all five channels of the microfluidic thermometer chip were filled with water and a stable temperature gradient was formed along the channels using the setup in Figure 2.1. (B) To characterize the thermal isocline shape, each of the five channels were filled with a different solution with precisely-known freezing/melting points (deionized water and 2%, 4%, 6%, and 8% *m/m* NaCl solutions in channels *E*, *D*, *C*, *B*, and *A*, respectively). (C) To measure an unknown freezing/melting point, four of the thermometer chip channels were filled with solutions with known freezing points (water in channels *A* and *D*, and 8% (*m/m*) NaCl in channels *B* and *E*) and the remaining channel *C* was filled with a NaCl solution with an "unknown" freezing point. 22

2.5 **(A)** Using a glass microfluidic chip as a microfluidic thermometer. Solid-liquid interfaces in channels containing only 400 nL of water mark the location of the $T = 0$ °C isotherm. **(B)** Using coconut oil (solidifying/melting point = 24.1 °C) in a 3D-pri crofluidic thermometer chip. The interfaces between the solid oil (white) and liquid oil (transparent) are easily identified and form a stable linear isotherm in the chip. Including an additional material with a different freezing/melting point in the thermometer chip would enable measurement of temperatures well above zero Celsius. 24
2.6 A drawing of the of microfluidic thermometer chip design used to 3D print the device. 29

3.1 Producing a "chronological fingerprint" or *chronoprint* capturing how six samples (in this example, authentic and adulterated samples of an over-the-counter liquid cold medicine) respond to a perturbation over space and time (in this case, a rapidly-changing temperature gradient). 81
3.2 Overview of the *feature tracing* method of comparing chronoprints (in this case, obtained from two different food oils). 88
3.3 Overview of the *image differences* method of comparing chronoprints (in this case, obtained from authentic and diluted samples of liquid cold medicine). . 90
3.4 Overview of the *image hashing* method of comparing chronoprints (in this case, obtained from authentic and diluted samples of liquid cold medicine). . 92
3.5 Identifying authentic and adulterated food oils using chronoprints. 95
3.6 Detecting authentic liquid cold medicine using chronoprints. 99
3.7 Detecting adulterated liquid cold medicine using chronoprints. 101
3.8 Distinguishing toxic and nontoxic pharmaceutical ingredients using chronoprints. 104
3.9 A drawing of the thermometer chip design used to fabricate chips in this work.109
3.10 Curves resulting from using the *feature tracing* analysis of chronoprints from from 100% olive oil (channels 1 and 4), 100% peanut oil (channels 2 and 5), and a 1:1 mixture of olive and peanut oils (channels 3 and 6). Excessive lighting during video recording saturated some pixels and adversely affected the quality of these results. 111
3.11 Curves resulting from using the *feature tracing* analysis of chronoprints from from a 1:1 mixture of olive and peanut oils (channels 1 and 4), 100% peanut oil (channels 2 and 5), and 100% olive oil (channels 3 and 6). 112
3.12 Curves resulting from using the *feature tracing* analysis of chronoprints from from 100% peanut oil (channels 1 and 4), a 1:1 mixture of olive and peanut oils (channels 2 and 5), and 100% olive oil (channels 3 and 6). 114
3.13 Comparisons of different chronoprint similarity algorithms for six cold medicine samples from one bottle (the experiment shown in Figure 3.6A of the main text). 117
3.14 omparisons of different chronoprint similarity algorithms for six cold medicine samples from six different manufacturing lots (the experiment shown in Figure 3.6B of the main text). 118

3.15 Replicate of comparing six cold medicine samples from one bottle (replicate of the experiment shown in Figure 3.6A of the main text. 119
3.16 Replicate of comparing six cold medicine samples from six different manufacturing lots (replicate of the experiment shown in Figure 3.6B of the main text). 120
3.17 Comparisons of different chronoprint similarity algorithms for 50%, 75%, and 100% (v/v) dilutions of cold medicine in water (the experiment shown in Figure 3.7A of the main text). 121
3.18 Replicate of the analysis of 50%, 75%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7A of the main text). 122
3.19 Replicate of the analysis of 50%, 75%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7A of the main text). 123
3.20 Comparisons of different chronoprint similarity algorithms for 90%, 95%, and 100% (v/v) dilutions of cold medicine in water (the experiment shown in Figure 3.7B of the main text). 124
3.21 Replicate of the analysis of 90%, 95%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7B of the main text). 125
3.22 Replicate of the analysis of 90%, 95%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7B of the main text). 126

4.1 Producing an "*enhanced chronoprint*" capturing how six samples (in this example, 50% and 60% (v/v) glycerol in water dilutions) with metal beads respond to a perturbation over space and time (in this case, a temperature gradient and gravitational force). 158
4.2 Overview of comparing chronoprints (in this case, obtained from two different glycerol solutions). 164
4.3 Distinguishing colorless glycerol dilutions using enhanced chronoprints. 166
4.4 Distinguishing toxic and nontoxic pharmaceutical ingredients using enhanced chronoprints. 168
4.5 A drawing of the thermometer chip design used to fabricate chips in this work. 171
4.6 Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable. 172
4.7 Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable. 173
4.8 Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable. 174
4.9 Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable. 175
4.10 Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable. 176

4.11 Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable. 177
4.12 Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable. 178
4.13 Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable. 179

Chapter 1

Introduction

1.1 Motivation and Significance

Technology for measuring the physical properties of samples and Identifying substances by their properties are important in many different fields and have a wide range of applications. Conventional analytical tools and instrumentals such as spectroscopy and chromatography systems are exceptional in their ability to measure and analyze samples. However, the initial cost and operating costs, along with the size and complexity of these instruments limit their use in important applications in resource-limited settings (Figrure 1.1). While "frugal science"— science solutions that prioritize both cost and function— has made strides in expanding access to healthcare and scientific research, with technologies such as the "Foldscope" (fold-able microscope with over 60,000 distributed to over 135 countries (Figure 1.2A) [1] and the "Mbira Sensor" (modified music instrument that measures and identifies substances, Figure 1.2B [2]), along with the MECs (Multifluidic Evolutionary Components, Figure 1.2C) [3] "building blocks" to create instruments and the quick rise in the use of microcontrollers and open-source 3D-printing to create lab equipment [4, 5, 6] (Figure 1.2D), there is still an unmet need for more low-cost and accessible research and analytical tools. In my work I develop simple, low-cost, and effective analytical tools that utilize changes in intrinsic physical properties for the analysis of samples on microfluidic platforms.

Figure 1.1: Chemical analysis instruments such as the liquid chromatography - Mass Spectrometer (Thermo Scientific Exactive LC-MS) are complex, take up a lot of space, are expensive, and often require additional equipment to operate.

1.2 Approach

Intrinsic physical properties are inherent qualities of matter and they only change when the matter itself has changed. Since they are independent of the amount of material or substance present, examining measurements of these properties for a substance or material

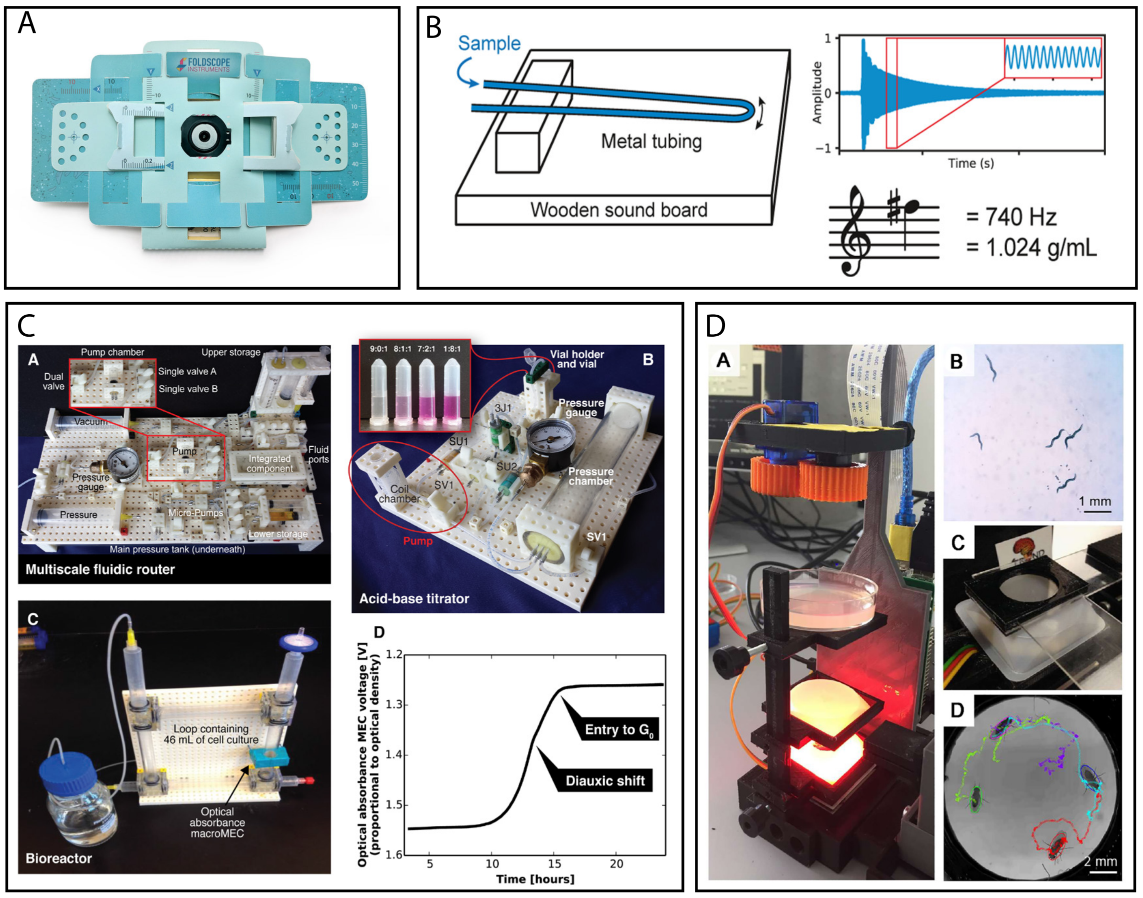


Figure 1.2: Low-cost tools improve access to healthcare and research capabilities. **(A)** The Foldscope paper microscope has been used all over the world to study the environment, plant biology and pathology, as well as human an animal biology and pathology [1]. **(B)** The Mbira Sensor identifies a substance by detecting the sound frequency emitted by the instrument when a sample flows through the device— which correlates to the density of the sample— and has shown to distinguish pharmaceutical ingredients and detect adulterated medicines [2]. **(C)** The MECs "building blocks" are designed to integrate together to build custom instruments quickly and perform a wide array of different tasks [3]. **(D)** The 3D printed microscope cost under 100 euros, was designed for open-source use, and is made of 3D-printed components and a microcontroller system with a high-definition camera; it has been used to study neurogenetics [6].

while the sample undergoes a change can provide a wealth of information about the sample. Since all matter have intrinsic physical properties, a wide range of substances and materials can be studied and variety of intrinsic property measurements (e.g. density, boiling point, freezing/melting point, etc.) can be explored. In this work I focus on the freezing and melting properties of substances and develop techniques for measuring temperature and identifying substances in microfluidic chips based on these properties; I leverage the phase changes of the samples between the solid and liquid states, Figure 1.3. I chose to develop these methods in a microfluidic configuration because microfluidic devices are capable of rapid and high throughput processing of samples, they're scalable, typically have a small footprint, and they consume minimal sample and generate minimal waste. The versatility of microfluidic devices make them valuable tools in a variety of settings and fields.

1.3 Project Overview

In this work I introduce microfluidic tools and methods to accurately measure temperature and identify substances, or adulteration of substances, in small volumes. I accomplish this through three projects: a *Microfluidic Thermometer, Chronoprints,* and *Enhanced Chronoprints.*

A microfluidic thermometer: Precise temperature measurements in microliter and nanoliter-scale volumes— In this project I develop a simple, and low-cost method to precisely measure the temperature of samples within a microfluidic chip based on their freezing and melting points. Measuring the temperature of a sample is a fundamental need in many biological and chemical processes. By providing a low-cost and simple way to

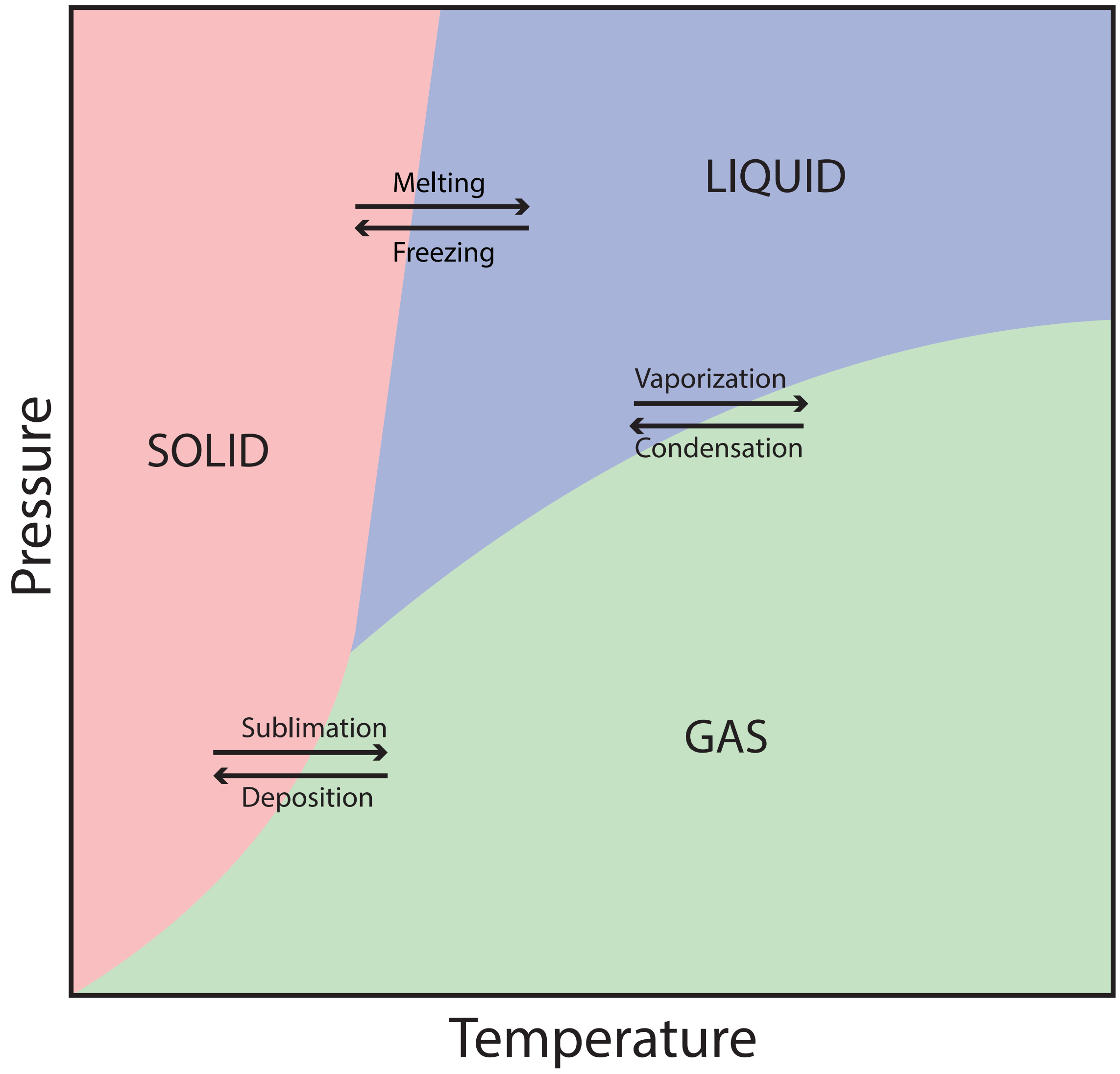


Figure 1.3: A general phase diagram for a substance. In this work I focus on the freezing and melting properties of substances and leverage the phase changes of these samples (between the solid and liquid phases) to develop techniques for measuring temperature and identifying substances.

accurately measure temperatures in small volumes, this technique can have applications in a wide variety of research and educational laboratories.

Chronoprints: Identifying samples by visualizing how they change over space and time— In this project I build upon the microfluidic thermometer work and take it in a new direction to identify adulterated food products and medicine. The modern tools of chemistry excel at identifying a sample, but the cost, size, complexity, and power consumption of these instruments often preclude their use in resource-limited settings. In this work, I demonstrate a simple and low-cost method for identifying a sample based on visualizing how the sample changes in response to a perturbation over space and time. The simplicity and versatility of this technique should make them valuable analytical tools in a variety of different fields.

Enhanced chronoprints: Identifying samples by visualizing changes in particle interactions— The previous chronoprint method has some limitations with the types of samples that can be analyzed and compared, and in this project I develop a method to address those limitations. I build upon the original chonoprint method, and demonstrate a simple and low-cost method for identifying a sample based on visualizing how the interaction of a metal bead with the sample changes in response to a perturbation over space and time. This broadens the range of sample types that can be analyzed with the chronoprint technique and enables the distinction between different substances based on how their material properties change in response to a perturbation.

This work concludes with prospects for applications in the development of pharmaceutical drug and ingredients.

Chapter 2

A microfluidic thermometer: Precise temperature measurements in microliter- and nanoliter-scale volumes

2.1 Introduction

The ability to accurately measure temperatures is a crucial need in many biological and chemical processes [7, 8, 9, 10]. For milliliter-scale volumes, conventional thermometers and sensors like thermocouples and thermistors are adequate for measuring the temperature of a substance. However, these techniques are less suitable for measuring the temperature of microliter- or nanoliter-scale volumes (which are commonly encountered with cells, microorganisms, precious samples, and samples inside microfluidic chips). Infrared (IR) thermometers can measure the temperature of a surface[11, 12, 13], but their sensitivity to a material's emissivity and large sensing area make IR thermometers less suitable for measuring the temperature of the fluid inside a microfluidic chip [14, 15]. Similarly, thermocouples affixed to the surface of a microfluidic chip can measure the surface temperature, but there can be significant temperature differences between the surface of a chip and the fluid inside the chip[16]. Resistance temperature detectors (RTDs) can be fabricated inside microfluidic channels [17, 18, 19, 20, 21], but RTDs complicate the chip fabrication process and can be physically or chemically incompatible with on-chip fluids. Finally, temperature-sensitive fluorophores, magnetic nanoparticles, and nanodiamond probes can be added to a fluid to measure its temperature [22, 23, 24, 25], but these methods require lasers or magnetic fields to activate the probes and may not be chemically or biologically compatible with all samples. In summary, there is an unmet need for simple, broadly-applicable, and label-free techniques for measuring temperatures in small fluid volumes.

In this work we present a "microfluidic thermometer", a simple microfluidic chip that can measure the temperature of microliter- and nanoliter-scale volumes of fluid with an uncertainty of a quarter of a degree Celsius. The microfluidic thermometer shown in Figure 2.1 takes advantage of the fact that when two phases of a substance (for example, liquid water and ice) are both present at the same location, the temperature of that location at equilibrium is precisely known (in this example, 0 °C). By adding an array of channels to a microfluidic chip, filling those channels with materials with known freezing/melting points, establishing a stable and linear temperature gradient perpendicular to the channels within a measurement region on the chip, and locating the solid-liquid interfaces in the channels, one can visualize the temperature gradient inside the chip and predict the temperature of a sample at any arbitrary point in the measurement region. Solid-liquid interfaces have previously been used for adsorption of proteins [26, 27, 28], surfactants [29, 30], and polymers[31] and the formation of lipid bilayers [32], metals and alloys [33, 34], and free radicals [35], but to our knowledge, no previous study has used the locations of multiple solid-liquid interfaces as a tool to measure temperature.

In this proof-of-concept, we created a prototype 3D-printed microfluidic thermometer chip and a custom software tool that can measure the temperature of a 50 microliter sample. We used this chip to measure the “unknown” freezing point of a sodium chloride solution. We also created a 3D heat transfer model of the thermometer chip to visualize the temperature gradient and isotherms inside the device. Finally, we also demonstrated the feasibility of this technique in even smaller (nanoliter-scale) volumes using a glass microfluidic chip. The design of the thermometer chip in standard .STL format (section 2.6.1) and source code for our software tool (sections 2.6.2 and 2.6.3) are available for download, meaning that anyone with access to a suitable 3D printer can replicate our technique and use it to analyze their own samples.

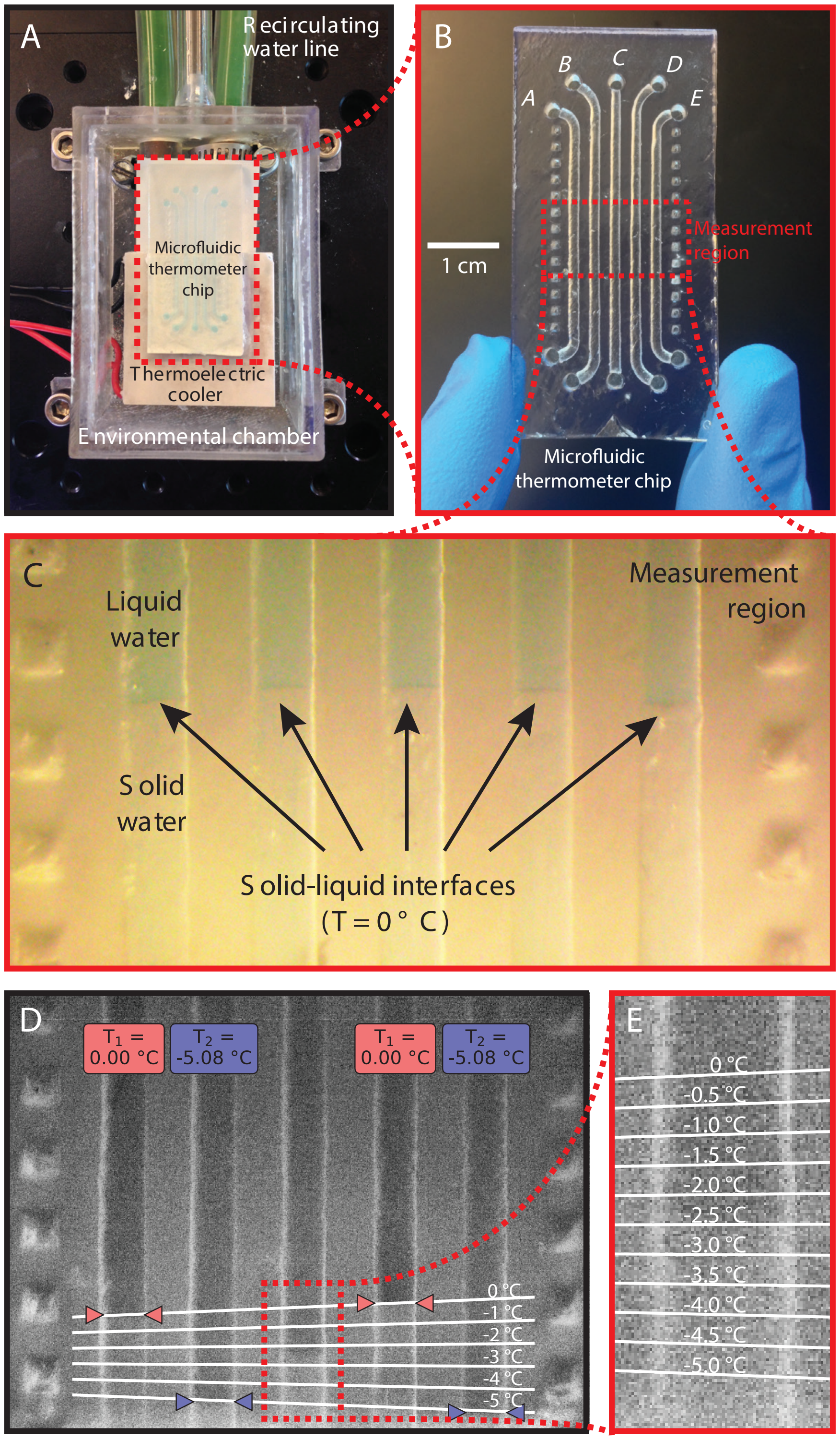

A
Recirculating water line
Microfluidic thermometer chip
Thermoelectric cooler
Environmental chamber
B
A
B
C
D
E
1 cm
Measurement region
Microfluidic thermometer chip
C
Liquid water
Measurement region
Solid water
Solid-liquid interfaces (T = 0 ° C)
D
T₁ = 0.00 °C
T₂ = -5.08 °C
T₁ = 0.00 °C
T₂ = -5.08 °C
0 °C
-1 °C
-2 °C
-3 °C
-4 °C
-5 °C
E
0 °C
-0.5 °C
-1.0 °C
-1.5 °C
-2.0 °C
-2.5 °C
-3.0 °C
-3.5 °C
-4.0 °C
-4.5 °C
-5.0 °C

Figure 2.1

Figure 2.1: Design and operation of the 3D-printed microfluidic thermometer chip. **(A)** The chip is placed halfway on a thermoelectric cooler to establish a temperature gradient in the measurement region inside the chip, and the chip is located inside a 3D-printed environmental chamber to eliminate condensation. **(B)** The thermometer chip includes five channels (*A*–*E*) for containing samples and standards with known freezing/melting points, and a measurement region in which the temperature gradient is roughly linear. **(C)** In a microscope image of the measurement region while the channels are filled with water, the solid-liquid interfaces are visible and define an isotherm (T = 0 °C) inside the chip. **(D)** During use, the user locates solid-liquid interfaces in four channels containing fluids with known freezing/melting points (0 °C and -5.08 °C in this example), and our software uses these locations to calculate the linear temperature gradient along the channels in this region. **(E)** Using this gradient, the temperature of the contents of the middle channel can be determined at any point within the measurement region with an uncertainty of about a quarter of a degree Celsius.

2.2 Materials and Methods

2.2.1 Designing and fabricating microfluidic thermometer chips

The microfluidic thermometer chip shown in Figure 2.1 was designed using SolidWorks (Dassault Systèmes, Vélizy-Villacoublay, France). The chip is 50 mm long, 25 mm wide, and contains five parallel channels with curved entries and exits to provide adequate space for fluid inlets and outlets. Each channel is 1 mm wide, 1 mm deep, and 30 mm long (along the straight portions) with 1.5 mm space between each channel. The chip design was exported as an .STL file (section 2.6.1) and printed using a stereolithography 3D printer (Form 1+, Formlabs, Cambridge, MA) with clear resin (GPCL02; Formlabs). After printing, unpolymerized resin was rinsed away by immersing the chip in isopropanol for 5 minutes, then the device was left to dry overnight.

Additionally, to confirm that our technique can be used with even smaller volumes, a glass microfluidic thermometer chip was fabricated that contains just 400 nL of each fluid. This chip was designed using AutoCAD (Autodesk, San Rafael, CA) and a photomask containing the design was printed using an overhead projector transparency and a conventional inkjet printer. The photomask transparency was placed in contact with a chromium- and photoresist-coated glass photomask blank (Telic, Valencia, CA) and irradiated using ultraviolet light. The exposed regions of photoresist were removed using a developer, and the exposed regions of chromium were removed using a chrome etchant. The now-exposed regions of glass were etched to the desired channel depth using 49% hydrofluoric acid. The remaining photoresist and chromium regions were then removed using acetone and chrome etch, respectively. Fluid inlet/outlet holes were drilled in the glass wafer using diamondtipped drill bits, and the wafer was bonded to a second (blank) glass wafer using thermal fusion bonding (668 °C for 8 hours).

2.2.2 Preparing liquids with known freezing points

As long as a substance has an identifiable interface between its solid and liquid phases and a precisely-known freezing/melting temperature, the substance could in principle be used in a microfluidic thermometer. In this work we used pure water (freezing point 0 °C), sodium chloride (NaCl) solutions with precisely-known freezing points down to −5.08 °C [36], and pure coconut oil with a precisely-known solidification temperature of 24.1 °C [37]. To enhance the visibility of the solid-liquid interface of the sodium chloride solutions, we added a small amount of blue food coloring to each solution (final concentration 0.01% food coloring by mass). Since coconut oil is transparent when liquid but opaque and white when solid, the solid-liquid interface in coconut oil is easily identified and no food coloring or other additives were needed to visualize this interface. Each channel of the 3D-printed microfluidic thermometer chip received 50 μL of liquid, and each channel of the glass microfluidic thermometer chip received 400 nL of liquid.

2.2.3 Establishing a temperature gradient across the measurement region on the thermometer chip

A stable temperature gradient was formed across the measurement region on the microfluidic thermometer chip by placing part of the chip on a thermoelectric cooler; the rest of the chip was suspended in air (Figure 2.1A). The thermoelectric cooler (TEC1-12706, Hebei I.T. Co., Shanghai, China) is connected to a recirculating water line that removes excess heat from the backside of the cooler. To suppress water condensation on the chip (which makes the contents of the chip difficult to visualize), the chip and cooler were placed inside a 3D-printed enclosure that was gently purged with dry nitrogen at 10 °C. A glass lid on the enclosure allows for visualization of the thermometer chip. In this manner, we created a stable temperature gradient across the thermometer chip that spans from -20 °C (at the end of the chip nearest the cooler) to 10 °C (at the suspended end of the chip). For experiments requiring temperature gradients near room temperature (for example, using coconut oil that solidifies at 24.1 °C), we simply reversed the polarity of the thermoelectric cooler to make it function as a heater.

2.2.4 Using the microfluidic thermometer chip

In a typical experiment, the five channels on the thermometer chip (labeled A–E in Figure 2.1B) are filled with three different materials. Channels A and D contain a material with a known freezing point T_1, channels B and E contain a second material with a known freezing point T_2, and channel C contains a material whose temperature is to be measured. While the thermometer chip allows for the measurement of the temperature at any location within channel C, in the following analysis we are interested in measuring the temperature at the location where the solid and liquid phases of the material in channel C touch—that is, the unknown freezing point T_{unk} of the material in channel C. The chip is then placed on the cooler assembly as shown in Figure 2.1A and given 20 minutes to reach thermal equilibrium. An inspection microscope (SM-4TZ-144A, AmScope, Irvine, CA) is used to acquire an image of solid-liquid interfaces inside the five channels on the thermometer chip.

The image of the microfluidic thermometer chip is then opened in a custom Python program (sections 2.6.2 and 2.6.3). The user specifies the known freezing points T_1 and T_2 of the materials in channels A/D and B/E, respectively. The program then instructs the user to click on the locations of the solid-liquid interfaces in all five channels. This provides the program with the (x, y) coordinates of these interfaces in units of pixels:

- (x_A, y_A): location of solid-liquid interface in channel A at temperature T_1
- (x_D, y_D): location of solid-liquid interface in channel D at temperature T_1
- (x_B, y_B): location of solid-liquid interface in channel B at temperature T_2
- (x_E, y_E): location of solid-liquid interface in channel E at temperature T_2
- (x_C, y_C): location of solid-liquid interface in channel C at temperature T_{unk}

The program then calculates the slope m_1 and y-intercept b_1 of the line between the solid-liquid interfaces of channel A and channel D:

$$m_1 = \frac{y_D - y_A}{x_D - x_A} \qquad b_1 = y_A - m_1 x_A \tag{2.1}$$

and the slope m_2 and y-intercept b_2 of the line between the solid-liquid interfaces of channel B and channel E:

$$m_2 = \frac{y_E - y_B}{x_E - x_B} \qquad b_2 = y_B - m_2 x_B \tag{2.2}$$

These two lines represent isotherms on the thermometer chip; the first line marks a region of the chip at known temperature T_1, and the second line marks a region of the chip at known temperature T_2. The program then calculates where these isotherms intersect channel C—in other words, what location $(x_{C(T_1)}, y_{C(T_1)})$ in channel C is at temperature T_1:

$$x_{C(T_1)} = x_C \qquad y_{C(T_1)} = m_1 x_{C(T_1)} + b_1 \tag{2.3}$$

and what location $(x_{C(T_2)}, y_{C(T_2)})$ in channel C is at temperature T_2:

$$x_{C(T_2)} = x_C \qquad y_{C(T_2)} = m_2 x_{C(T_2)} + b_2 \tag{2.4}$$

The program then plots temperature vs. y-coordinate for channel C and calculates the slope m_3 and y-intercept b_3 of this line.

$$m_3 = \frac{T_2 - T_1}{y_{C(T_2)} - y_{C(T_1)}} \qquad b_3 = T_1 - m_3 y_{C(T_1)} \tag{2.5}$$

The equation of this line can be used to calculate the temperature of the contents of channel C at any point along the channel. By solving this equation using the y-coordinate y_C of the solid-liquid interface in channel C, we can determine the unknown freezing point T_{unk} of the solution in channel C:

$$T_{unk} = m_3 y_C + b_3 \tag{2.6}$$

The Python code available for download (sections 2.6.2 and 2.6.3) automates this process and was used to create Figure 2.1D and E as well as the figures in the *Results and Discussion* section.

2.2.5 Modeling the microfluidic thermometer chip

To further characterize the shape of the thermal gradient inside the thermometer chip, a 3D model of the chip was created using finite element analysis (COMSOL Multiphysics, Burlington, MA). The model replicates the geometry and temperature of the chip as well as the chip's orientation partly on the thermoelectric cooler. The "heat transfer physics" module with convective heat flux and a stationary solver was used to model heat transfer between the microfluidic thermometer chip and its channel contents, the cooler, and the surrounding ambient air. Heat transfer coefficients for natural convection normally range from 5 to 50 W m^{-2} K^{-1} [38]; however, in the microfluidic thermometer, heat transfer by conduction via the thermoelectric cooler dominates and convective losses are minimal, so we used a slightly lower estimate of the heat transfer coefficient (1 W m^{-2} K^{-1}).

2.3 Results and Discussion

2.3.1 Modeling isotherm and thermal isocline shape in the thermometer chip

Our method for analyzing the data from the microfluidic thermometer chip makes two assumptions about the shape of the temperature gradient in the measurement region of the chip:

- First, we assume that if we draw a line between two solid-liquid interfaces that are at the same known temperature in the measurement region on the chip, then all points on that line are also at the same known temperature (the line is an *isotherm*).
- Second, we assume that if we draw a line between two points at different known temperatures within the measurement region on the chip, then there is a linear gradient of temperatures along that line (the line is a *thermal isocline*).

We tested the validity of each of these assumptions using both computer simulations and experimental measurements.

Figure 2.2 shows the simulated behavior of the water-filled 3D-printed thermometer chip obtained using COMSOL Multiphysics. In the measurement region on the chip (dotted rectangles in Figures 2.2A and B), the isotherms are straight (Figure 2.2C) and the temperature across the five channels varies by no more than 0.13 °C in the channel region (Figure 2.2D). These results support our assumption that the isotherms perpendicular to the channels are linear within the measurement region of the chip. Additionally, the temperature gradient is roughly linear along the channel length in the roughly 6-mm-long measurement region (gray area in Figure 2.2E); this supports our assumption that temperature is a linear function of distance along the channel within the measurement region. Outside of the measurement region on the chip, the temperature profile along the channels is no longer linear but much more complex (see region outside the gray area in Figure 2.2E). Consequently, solid-liquid interfaces outside the measurement region cannot be used to infer temperatures inside the measurement region.

To determine if the contents of the thermometer chip channels affect the thermal behavior of the chip in the measurement region, we repeated the analysis in Figure 2.2 with the channels filled with water, mineral oil, and toluene. These substances are commonly used in microfluidics and have different thermal properties like thermal conductivity and heat capacity. Figure 2.3 shows that the different contents of the microfluidic channels had minimal impact on the temperature distribution across the channels in the measurement region.

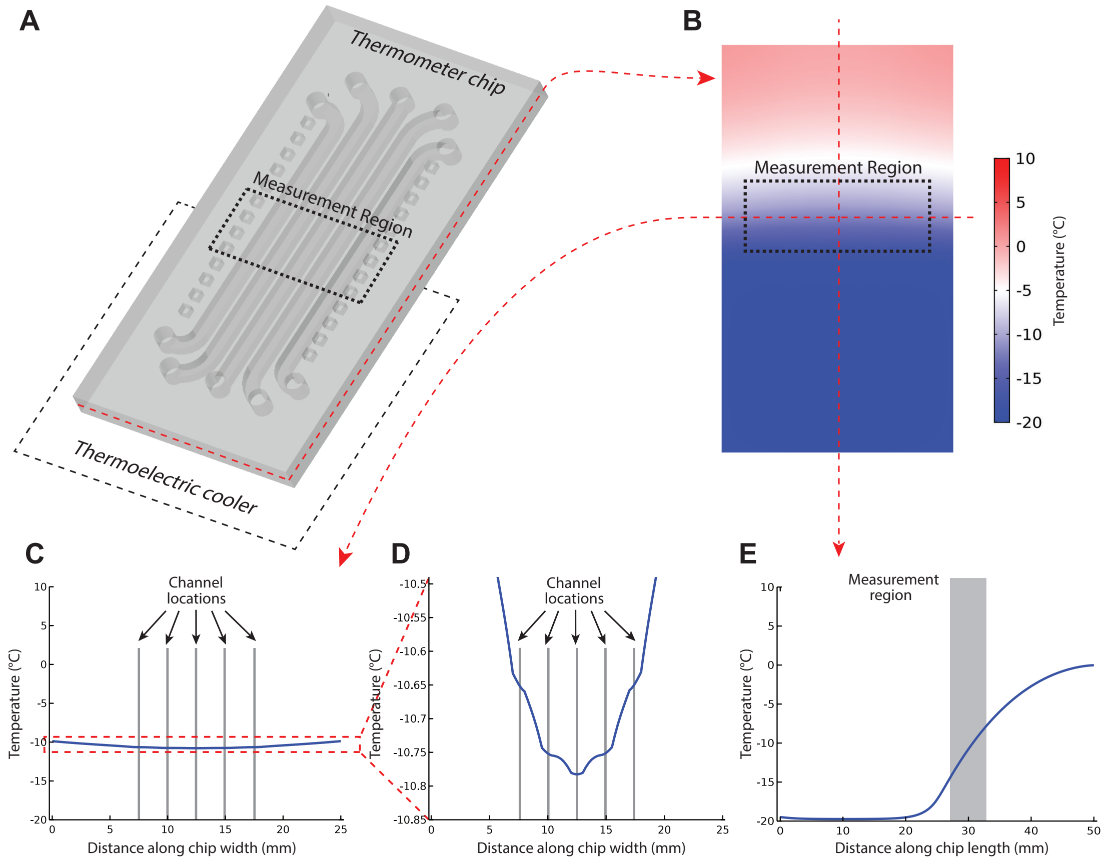


Figure 2.2: Finite element analysis computer simulations of the 3D-printed microfluidic thermometer chip with all five channels filled with water. **(A)** The simulated chip is oriented partway on a thermoelectric cooler ($T = -20°\mathrm{C}$) and the rest of the chip is suspended in air ($T = 10$ °C). By slicing through the middle of the chip, the temperature gradient in the channel plane is visible **(B)**. Plotting the temperature profile in this plane across the five channels in the region where freezing measurements are obtained **(C)** results in a nearly-flat line in this region; closer inspection **(D)** shows a variation of only 0.13 °C across the five channels, supporting our assumption that isotherms are nearly linear in the measurement region. Plotting the temperature profile in the channel plane along the middle channel **(E)** results in a fairly complicated temperature profile ranging from a constant -20 °C above the cooler to 0 °C at the opposite end of the chip. However, in the region of the chip where freezing measurements are observed (corresponding to the shaded region on the plot), the predicted temperature profile is nearly linear; this supports our assumption about the shape of the temperature gradient along the channels in the measurement region.

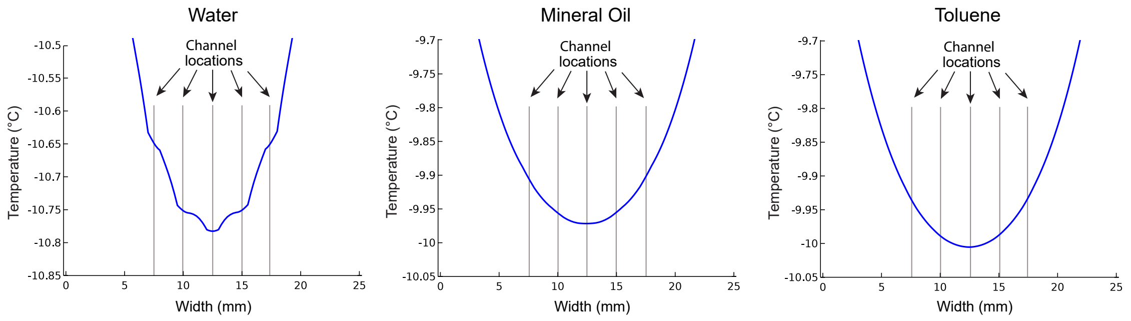


Figure 2.3: Finite element analysis computer simulations of the temperature profile across the five channels in the measurement region inside the microfluidic thermometer chip, with the channels filled with water (left), mineral oil (center), and toluene (right). Despite the different thermal properties of these fluids (with water being the most thermally conductive), the predicted temperatures differ by less than 0.13 °C across the 10 mm wide measurement region. This supports our assumption that isotherms in the measurement region of the thermometer chip are essentially linear, even when filling the channels with materials other than aqueous solutions.

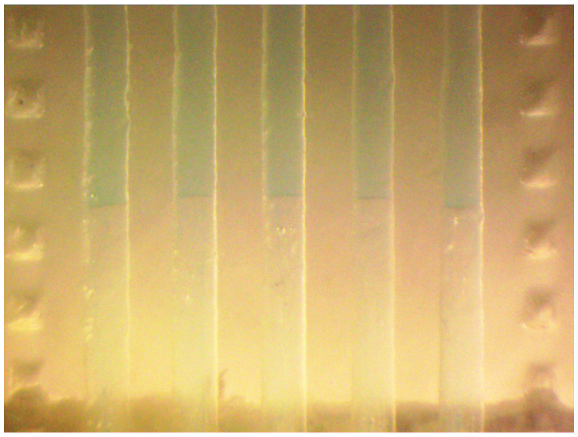
2.3.2 Measuring isotherm and thermal isocline shape in the thermometer chip

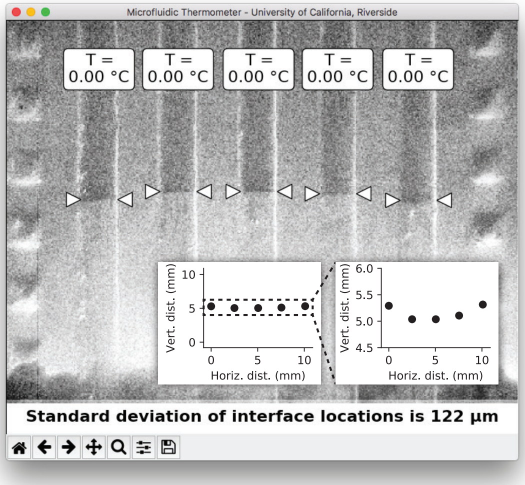
To experimentally verify our assumption that isotherms in the thermometer chip are linear, we filled all five channels in the thermometer chip with the same fluid (deionized water; freezing point = 0 °C) and established a temperature gradient along the length of the channels using the setup shown in Figure 2.1. A micrograph of the solid-liquid interfaces in the measurement region of the thermometer chip is shown in Figure 2.4A. After importing this image into our custom software (sections 2.6.2 and 2.6.3) and clicking on the locations of solid-liquid interfaces in each channel, we found that the solid-liquid interfaces formed a reasonably straight isotherm; the standard deviation of the interface locations in the vertical direction was only 122 μm. Plotting the locations of the interfaces (inset plots in Figure 2.4A) shows that the vertical locations of the five interfaces differ by less than 500 μm (across all five channels; a horizontal distance of 10 mm). We repeated this experiment for a second solution, an 8% (m/m) NaCl solution with a known freezing point of -5.08 °C. The resulting interfaces were again visible and linear (standard deviation of vertical interface location = 130 μm; data not shown). These results further support our assumption that isotherms are linear in the measurement region of the chip.

We also experimentally verified the linearity of isotherms in microfluidic chips with much smaller channel volumes. Figure 2.5A shows a closeup of two solid-liquid interfaces in a borosilicate glass chip containing 400 nL of water with a small amount of food coloring in each channel. The solid-liquid interfaces are clearly visible, confirming that microfluidic thermometer chips can be fabricated and used with nanoliter-scale volumes.

To experimentally verify our assumption that the temperature gradient along the chip is linear within the measurement region (and therefore a line along a channel in this region is a thermal isocline), we filled each of the five channels with different fluids with precisely-known freezing/melting points (8%, 6%, 4%, and 2% m/m NaCl solutions and deionized water in channels A, B, C, D, and E, respectively) and established a temperature gradient along the chip. The micrograph of the measurement region in Figure 2.4B shows that the solid-liquid interfaces appear in different locations along the channels; the material with the lowest freezing point (8% NaCl; channel A) has an interface near the bottom of the image, and the material with the highest freezing point (deionized water; channel E) has an interface near the middle of the image. After importing this image into our software and clicking on the location of each solid-liquid interface, the software generates a plot of material

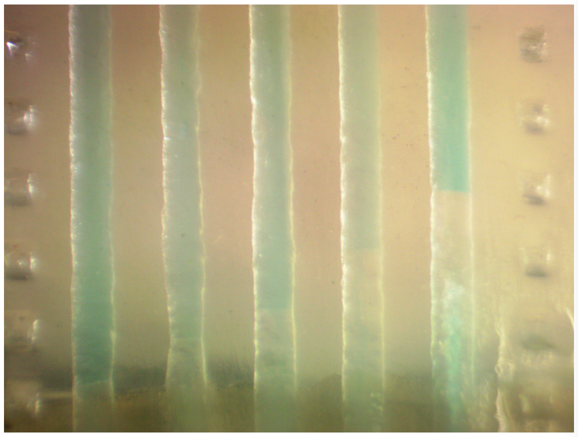
A: Characterize isotherm using single freezing point

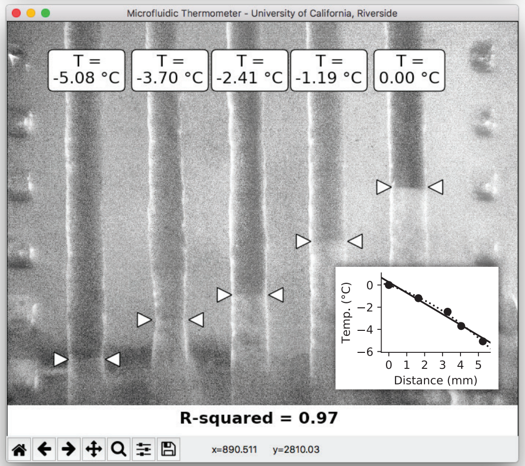




Standard deviation of interface locations is 122 µm

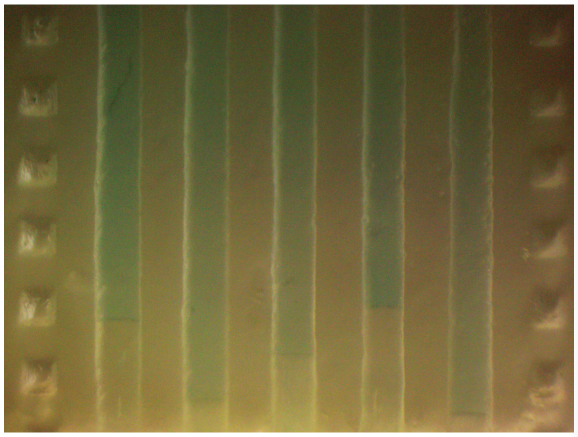
B: Characterize thermal isocline using five freezing points

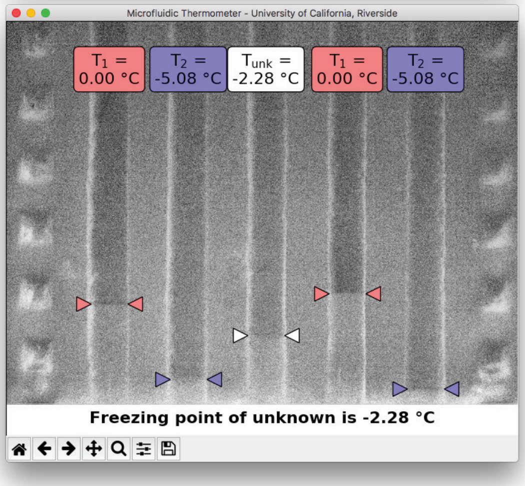




R-squared = 0.97

C: Measure unknown freezing point

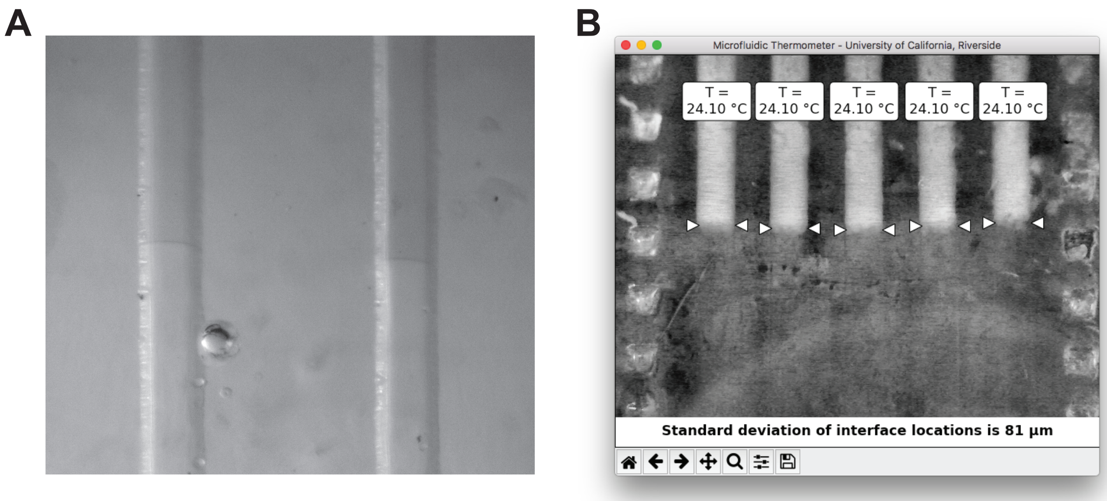




Freezing point of unknown is -2.28 °C

Figure 2.4

Figure 2.4: **(A)** To characterize isotherm shape, all five channels of the microfluidic thermometer chip were filled with water and a stable temperature gradient was formed along the channels using the setup in Figure 2.1. The solid-liquid interfaces that form are roughly linear. After clicking on the locations of each interface, the software draws an isotherm (0 °C) on the image and uses the parameters of this line to estimate the uncertainty of our measurement. The inset shows that the vertical locations of the solid-liquid interfaces differ by less than 500 μm across the entire 10 mm width of the measurement region; this supports our assumption that isotherms are linear in the measurement region. **(B)** To characterize the thermal isocline shape, each of the five channels were filled with a different solution with precisely-known freezing/melting points (deionized water and 2%, 4%, 6%, and 8% m/m NaCl solutions in channels E, D, C, B, and A, respectively). After clicking on the locations of each interface, the software plots the temperature at each interface vs. the vertical locations of the interfaces. A linear fit (solid line; $R^2 = 0.97$ and a maximum difference of only 0.48 °C between predicted and actual temperatures) confirms our assumption that the temperature gradients are roughly linear in the measurement region. For even higher precision, a second-order polynomial (dotted line) can be used with a maximum difference of only 0.22 °C between predicted and actual temperatures). **(C)** To measure an unknown freezing/melting point, four of the thermometer chip channels were filled with solutions with known freezing points (water in channels A and D, and 8% (m/m) NaCl in channels B and E) and the remaining channel C was filled with a NaCl solution with an "unknown" freezing point. After clicking on the locations of each solid-liquid interface, the software draws isotherms between the known temperatures (0 °C at the interfaces in channels A and D, and -5.08 °C at the interfaces in channels B and E), then calculates the linear temperature gradient between the two isotherms in channel C and uses this gradient to determine the temperature at the solid-liquid interface in channel C (-2.28 ± 0.26 °C). This agrees well with the known literature value for the freezing point of this solution, -2.41 °C.


A
B
Microfluidic Thermometer - University of California, Riverside
T = 24.10 °C
T = 24.10 °C
T = 24.10 °C
T = 24.10 °C
T = 24.10 °C

Standard deviation of interface locations is 81 µm

Figure 2.5: **(A)** Using a glass microfluidic chip as a microfluidic thermometer. Solid-liquid interfaces in channels containing only 400 nL of water mark the location of the $T = 0$ °C isotherm. **(B)** Using coconut oil (solidifying/melting point = 24.1 °C) in a 3D-pri crofluidic thermometer chip. The interfaces between the solid oil (white) and liquid oil (transparent) are easily identified and form a stable linear isotherm in the chip. Including an additional material with a different freezing/melting point in the thermometer chip would enable measurement of temperatures well above zero Celsius.

freezing/melting point vs. vertical location of the solid-liquid interface of the material (inset of Figure 2.4B). The plot is linear (solid line; $R^2 = 0.97$) with a maximum difference of 0.48 °C between the measured locations of the solid-liquid interfaces and those predicted by a linear regression fit of the measured locations. These results support our assumption that the temperature gradient is linear along the length of the chip within the measurement region. If even higher precision is needed for an application, the solid-liquid interface locations can be fitted to a second-order polynomial (dotted line in Figure 2.4B inset) which decreases the maximum difference between actual and predicted interface locations to just 0.22 °C. However, we used the linear temperature gradient assumption in this work.

2.3.3 Measuring temperature of an "unknown" solution using the thermometer chip

By utilizing the solid-liquid interface positions of two solutions with known freezing points, we were able to use the microfluidic thermometer chip to measure the temperatures throughout a sample of an "unknown" solution inside the measurement region. Specifically, we used the chip to determine the freezing point of a sodium chloride solution in the chip. Figure 2.4C shows a photograph of the thermometer chip filled with deionized water in channels A and D (freezing point $T_1 = 0$ °C), 8.0% (m/m) NaCl solution in channels B and E (freezing point $T_2 = -5.08$ °C), and a solution with a simulated unknown freezing point in channel C (4.0% (m/m) NaCl solution). After loading the image into our custom software (sections 2.6.2 and 2.6.3) and clicking on the locations of each solid-liquid interface, the software uses Equations 1–6 to determine that the freezing point T_3 of the "unknown" 4% NaCl solution is -2.28 ± 0.26 °C (details in section 2.6.3). This value is only 0.13 °C higher than the known literature value for the freezing point of a 4% (m/m) NaCl solution (-2.41 °C [36]).

2.3.4 Using the microfluidic thermometer at temperatures well above 0 °C

The experiments above use water and aqueous sodium chloride solutions to measure temperatures at or below 0 °C. However, there are obviously many applications for temperature measurements over a wide range of temperatures, not just below zero Celsius. To demonstrate that the microfluidic thermometer chip is also capable of measuring temperatures well above 0 °C, we filled all five channels with coconut oil (which has a precise solidifying/melting point at 24.1 °C [37]) and created a stable temperature gradient along the channels as described above. Since the freezing/melting point of coconut oil is slightly above ambient temperature, we heated (not cooled) one end of the microfluidic thermometer chip by reversing the polarity on the thermoelectric cooler and did not need dry nitrogen to prevent condensation.

Figure 2.5B shows the results from using our custom software (sections 2.6.2 and 2.6.3) to analyze the linearity of the isotherm in the microfluidic thermometer chip while filled with coconut oil. The interfaces between the solid oil (white) and liquid oil (transparent) are very easy to locate, and the standard deviation of the locations of these interfaces along the channels, 81 μm, is actually lower than that observed using water in Figure 2.4A. This shows that the solid-liquid interfaces in coconut oil form stable linear isotherms in the thermometer chip. By also using *e.g.* a second oil with a slightly different freezing/melting point, one can use the microfluidic thermometer chip to measure temperatures well above zero Celsius.

2.4 Conclusions

In this work, we demonstrated a simple technique for making precise measurements of the temperatures of microliter- and nanoliter-scale volumes. This technique requires minimal equipment and no probes, labels, or other modifications to the sample being measured. We used this technique to measure the freezing point of a simulated unknown solution. By using materials with different known freezing/melting points, our technique can be tailored for measurements at many different temperatures . Additionally, by using 3D printing to fabricate our thermometer chip, any researcher can download the design of the chip (section 2.6.1) and fabricate and use the chip. As the glass microfluidic thermometer chip in Figure 2.5 shows, this method is not limited to 3D-printed microfluidic devices and is suitable for use in any fabrication method that provides optical access to the channel contents. Finally, since the freezing point of a substance is an intrinsic property of that substance, our technique could be used as a simple way to identify a substance (or rule out other substances) by accurately measuring its freezing point.

In its current form, the thermometer chip is limited to making temperature measurements within the measurement region (the area of the chip where the temperature gradient is linear; dotted boxes in Figure 2.2A and B and gray region in Figure 2.2E). This roughly 6-mm-long region contains a linear temperature gradient that spans about 5 degrees Celsius. The *location* of this temperature range on the temperature scale can be set at will by loading the microfluidic thermometer channels with materials with different freezing/melting points (for example, using aqueous sodium chloride solutions to measure temperatures around 0 °C as in Figures 2.4 and 2.5A, or using coconut oil to measure temperatures around 24 °C as in Figure 2.5B). However, we cannot use sodium chloride solutions *and* coconut oil *simultaneously* in the thermometer chip because the freezing/melting points of these substances differ by over 24 °C—that means that at least one of the substances' solid-liquid interfaces would lie far outside the measurement region. If one of the solid-liquid interfaces forms outside of the region where the temperature gradient is linear, we would not be able to accurately predict the temperatures *between* the different solid-liquid interfaces. Therefore, our technique is more suited for precisely measuring temperatures in a narrow range using similar materials, not measuring temperatures in a wide range using dissimilar materials.

2.5 Acknowledgments

This work was supported by the National Science Foundation's Instrument Development for Biological Research Program under award DBI-1353974. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript. The text of this dissertation, in part or in full, is a reprint of the material as it appears in "A Microuidic Thermometer: Precise Temperature Measurements in Microliter- and Nanoliter-Scale Volumes". *PLoS One*, 12(12):e0189430, 2017. The co-author William H. Grover listed in that publication directed and supervised the research which forms the basis for this dissertation.

2.6 Supporting Information for "A microfluidic thermometer: Precise temperature measurements in microliter- and nanoliter-scale volumes"

2.6.1 Design of the microfluidic thermometer chip in the standard .STL format used by most 3D printers.

The design of the thermometer chip in standard .STL format is shown below and can be downloaded at

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0189430.

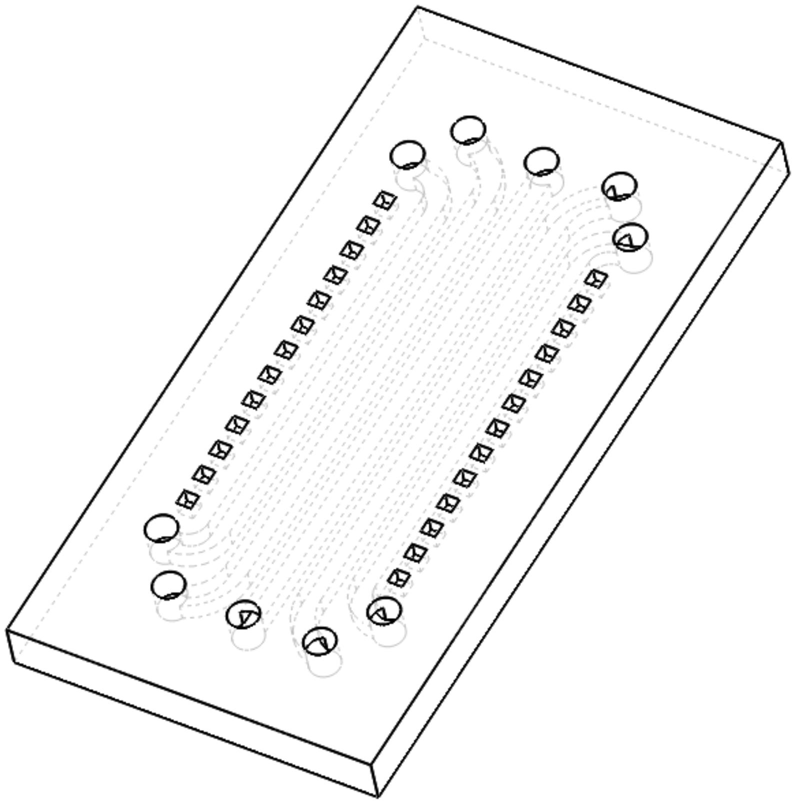


Figure 2.6: A drawing of the of microfluidic thermometer chip design used to 3D print the device.

2.6.2 `microfluidic_thermometer.py`: A custom Python program that automates the data analysis required when using the microfluidic thermometer chip. Used to generate Figures 2.1D and E, 2.4, and 2.5B.

The source code for the software tool is shown below and can be downloaded at https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0189430.

```python
"""
microfluidic_thermometer.py
Software for analyzing images of the microfluidic
    thermometer, as described in
"A microfluidic thermometer: Precise temperature
    measurements in microliter-
and nanoliter-scale volumes" by Brittney A. McKenzie and
    William H. Grover.
The latest version of this software is available at http://
    groverlab.org.
"""

import matplotlib.image
import matplotlib.pyplot
import scipy.stats
import numpy
import os.path
import matplotlib.widgets
import math
from skimage import exposure


```

```python
def setup_fig(filename):
    global fig, xmax, ymax, t, message, log
    filename = os.path.join(os.path.dirname(__file__), filename)
    img = matplotlib.image.imread(filename)
    xmax = img.shape[1]
    ymax = math.floor(img.shape[0] * 1.08)  # adds room for messages
    aspect_ratio = float(ymax) / float(xmax)
    fig = matplotlib.pyplot.figure(figsize=(7, 7 * aspect_ratio))
    fig.canvas.set_window_title(
        'Microfluidic␣Thermometer␣-␣University␣of␣California,␣Riverside')
    ax = fig.add_subplot(111)
    ax.xaxis.set_visible(False)
    ax.yaxis.set_visible(False)
    ax.imshow(exposure.equalize_adapthist(img[:, :, 0]), cmap="gray")  # green
    # ax.imshow(img[:,:,:])   # all channels
    ax.axis([0, xmax, ymax, 0])
    message = matplotlib.pyplot.text(xmax / 2, ymax * 0.96,
        "dummy",
                                      horizontalalignment='
                                         center',
                                      verticalalignment='
                                         center',
                                      color="black",
                                      backgroundcolor="white
                                         ",
                                      size="16",
                                      fontweight='bold',
                                      bbox=dict(boxstyle="
                                         square",
                                                fc="white",
                                                ec="white",
                                                lw=0))


def mark(x, y, s, color="white", unk=False):
    global xmax, ymax
    matplotlib.pyplot.plot(x - xmax * 0.05, y, "k>",
                                 markerfacecolor=color,
                            markeredgecolor="black",
                            markeredgewidth=1,
                            markersize=15)
    matplotlib.pyplot.plot(x + xmax * 0.05, y, "k<",
                            markerfacecolor=color,
                            markeredgecolor="black",
                            markeredgewidth=1,
                            markersize=15)
    if not unk:
        ytemp = ymax * .15
    else:
        ytemp = ymax * .24
    matplotlib.pyplot.text(x, ytemp, s,
                            horizontalalignment='center',
                            color="black",
                            backgroundcolor=color,
                            size="16",
                            bbox=dict(boxstyle="round",
                                        fc=color, ec="k", lw
                                            =1))


```

```
def edge_mark(x, y):
    matplotlib.pyplot.plot(x, y, "k^", markersize=15)


def standard_error(actuals, predicteds):
    sum = 0.0
    for a, p in zip(actuals, predicteds):
        sum += (a - p)**2
    return math.sqrt(sum / len(actuals))


x1 = x2 = x3 = x4 = x5 = y1 = y2 = y3 = y4 = y5 = 0
m1 = m2 = b1 = b2 = 0
known_distance = 0
log = u""
add_isotherms = False  # set this True to generate Fig 1D


# Measure freezing point of unknown using two samples of
    each of two known
# freezing points
def mode1(event):
    global ix, iy, x1, x2, x3, x4, x5, y1, y2, y3, y4, y5,
       m1, m2, b1, b2
    global click, message, log, add_isotherms
    ix, iy = event.xdata, event.ydata
    if click == 0:
        x1, y1 = ix, iy
        mark(x1, y1,
             "$\mathregular{T_1}$␣=\n%.2f" % T1 + u'␣\N{
                DEGREE␣SIGN}C',
             "#ff7777")
        message.set_text("Click␣second␣interface␣with␣
           freezing␣point␣" +
                         str(T1) + u'␣\N{DEGREE␣SIGN}C')
    if click == 1:
        x2, y2 = ix, iy
        mark(x2, y2,
             "$\mathregular{T_1}$␣=\n%.2f" % T1 + u'␣\N{
                DEGREE␣SIGN}C',
             "#ff7777")
        m1 = (y2 - y1) / (x2 - x1)
        b1 = y1 - m1 * x1
        message.set_text("Click␣first␣interface␣with␣
           freezing␣point␣" +
                         str(T2) + u'␣\N{DEGREE␣SIGN}C')
    if click == 2:
        x3, y3 = ix, iy
        mark(x3, y3,
             "$\mathregular{T_2}$␣=\n%.2f" % T2 + u'␣\N{
                DEGREE␣SIGN}C',
             "#7777ff")
        message.set_text("Click␣second␣interface␣with␣
           freezing␣point␣" +
                         str(T2) + u'␣\N{DEGREE␣SIGN}C')
    if click == 3:
        x4, y4 = ix, iy
        mark(x4, y4,
             "$\mathregular{T_2}$␣=\n%.2f" % T2 + u'␣\N{
                DEGREE␣SIGN}C',
             "#7777ff")
        m2 = (y4 - y3) / (x4 - x3)
        b2 = y3 - m2 * x3
        message.set_text("Click␣interface␣with␣unknown␣
           freezing␣point")
    if click >= 4:
        x5, y5 = ix, iy
        # calculate Y location of T1 in channel 3:
        T1y = m1 * x5 + b1
        # calculate Y location of T2 in channel 3:
        T2y = m2 * x5 + b2
        # equation of vertical temperature profile in channel 3:
        m = (T2 - T1) / (T2y - T1y)
        b = T1 - m * T1y
        log += "using␣scale␣" + str(mm_per_pixel) + "␣mm/pixel\n"
        log += "slope␣␣=␣" + str(m) + "␣degrees␣C␣per␣pixel\n"
        log += "slope␣=␣" + str(m / mm_per_pixel) + "␣degrees␣C␣per␣mm\n"
        try:
            log += "using␣stdev␣" + str(stdev) + "␣mm\n"
            log += "standard␣deviation␣of␣temperature␣=␣"
            log += str(stdev * m / mm_per_pixel) + "degrees␣C\n"
        except NameError:
            log += "Note:␣␣stdev␣wasn't␣defined"
        log += "y-int␣=␣" + str(b) + "␣degrees␣C\n"
        # predict temperature in channel 3:
        T = m * y5 + b
        if not add_isotherms:
            mark(x5, y5,
                "$\mathregular{T_{unk}}$␣=\n%.2f" % T + u'␣\N{DEGREE␣SIGN}C',
                "white")
        log += u"Freezing␣point␣of␣unknown␣is␣%.2f␣\N{DEGREE␣SIGN}C\n\n" % T
        message.set_text(u"Freezing␣point␣of␣unknown␣is␣%.2f␣\N{DEGREE␣SIGN}C" % T)
        if add_isotherms:
            message.set_text(u"")
            isotherms = [0.0, -0.5, -1.0, -1.5, -2.0, -2.5, -3.0, -3.5, -4.0, -4.5, -5.0]
            for i in isotherms:
                matplotlib.pyplot.plot([x5], [(i-b)/m], "kx")
        fig.canvas.mpl_disconnect(mode1)
    fig.canvas.draw()
    click += 1


# Calibrate using five samples of one known freezing point
def mode2(event):
    global ix, iy, x1, x2, x3, x4, x5, y1, y2, y3, y4, y5,
       m1, m2, b1, b2
    global stdev, message, log, click
    ix, iy = event.xdata, event.ydata
    if click == 0:
        x1, y1 = ix, iy
        mark(x1, y1, "$\mathregular{T}$␣=\n%.2f" % T1 + u'␣
           \N{DEGREE␣SIGN}C')
        message.set_text("Click␣second␣interface␣with␣
           freezing␣point␣" +
                                 str(T1) + u'\N{DEGREE␣SIGN}C')
    if click == 1:
        x2, y2 = ix, iy
        mark(x2, y2, "$\mathregular{T}$␣=\n%.2f" % T1 + u'␣
           \N{DEGREE␣SIGN}C')
        message.set_text("Click␣third␣interface␣with␣
           freezing␣point␣" +
                         str(T1) + u'\N{DEGREE␣SIGN}C')
    if click == 2:
        x3, y3 = ix, iy
        mark(x3, y3, "$\mathregular{T}$␣=\n%.2f" % T1 + u'␣
           \N{DEGREE␣SIGN}C')
        message.set_text("Click␣fourth␣interface␣with␣
           freezing␣point␣" +
                         str(T1) + u'\N{DEGREE␣SIGN}C')
    if click == 3:
        x4, y4 = ix, iy
        mark(x4, y4, "$\mathregular{T}$␣=\n%.2f" % T1 + u'␣
           \N{DEGREE␣SIGN}C')
        message.set_text("Click␣fifth␣interface␣with␣
           freezing␣point␣" +
                         str(T1) + u'\N{DEGREE␣SIGN}C')
    if click >= 4:
        x5, y5 = ix, iy
        mark(x5, y5, "$\mathregular{T}$␣=\n%.2f" % T1 + u'␣
           \N{DEGREE␣SIGN}C')

        fig2 = matplotlib.pyplot.figure(figsize=(2, 1.5))
        ax2 = fig2.add_subplot(111)
        ax2.spines['top'].set_visible(False)
        ax2.spines['right'].set_visible(False)
        x_locations = numpy.array([x1, x2, x3, x4, x5])
        y_locations = numpy.array([y1, y2, y3, y4, y5])
        x_smallest = min(x_locations)
        y_smallest = min(y_locations)
        # shift relative pixel locations to zero
        x_locations = x_locations - x_smallest
        # shift relative pixel locations to zero
        y_locations = y_locations - y_smallest + numpy.mean
           (x_locations)
        x_locations = x_locations * mm_per_pixel  # convert
            to millimeters
        y_locations = y_locations * mm_per_pixel  # convert
            to millimeters
        ax2.set_xlabel("Horiz.␣dist.␣(mm)")
        ax2.set_ylabel("Vert.␣dist.␣(mm)")
        ax2.plot(x_locations, y_locations, "ko")

        x_fullrange = max(x_locations) - min(x_locations)

        # fit the five points to a line:
```

```
        m, b, r_value, p_value, std_err = scipy.stats.
            linregress(
             [x1, x2, x3, x4, x5], [y1, y2, y3, y4, y5])
        log += "m:␣" + str(m) + "\n"
        log += "b:␣" + str(b) + "\n"
        log += "r_value:␣" + str(r_value) + "\n"
        log += "p_value:␣" + str(p_value) + "\n"
        log += "std_err:␣" + str(std_err) + "\n"
        log += "Standard␣deviation␣in␣pixels:␣"
        log += str(numpy.std([y1, y2, y3, y4, y5])) + "␣
            pixels\n"
        log += "using␣scale␣" + str(mm_per_pixel) + "␣mm/
            pixel\n"
        stdev = numpy.std([y1, y2, y3, y4, y5]) *
            mm_per_pixel
        log += "Standard␣deviation␣in␣mm:␣" + str(stdev) +
            "␣mm\n\n"
        message.set_text(
             u"Standard␣deviation␣of␣interface␣locations␣is␣
                 %.0f␣\N{GREEK␣SMALL␣LETTER␣MU}m" %
             (stdev * 1000))

        fig2.subplots_adjust(left=0.28, right=0.95, top=0.95, bottom=0.3)

        # full range:
        ax2.set_xlim(-0.08 * x_fullrange, 1.08 * x_fullrange)
        ax2.set_ylim(-0.08 * x_fullrange, 1.08 * x_fullrange)
        fig2.savefig("fig_4a_1.pdf")

        # closeup range:
        ax2.set_xlim(-0.08 * x_fullrange, 1.08 * x_fullrange)
        ax2.set_ylim(4.5, 6)
        fig2.savefig("fig_4a_2.pdf")

        matplotlib.pyplot.close(fig2)  # this keeps Matplotlib from hanging

        fig.canvas.mpl_disconnect(mode2)
    fig.canvas.draw()
    click += 1
```

```


# Calibrate temperature profile using five different known freezing points
def mode4(event):
    global ix, iy, x1, x2, x3, x4, x5, y1, y2, y3, y4, y5, m1, m2, b1, b2
    global stdev, message, log, click
    ix, iy = event.xdata, event.ydata
    if click == 0:
        x1, y1 = ix, iy
        mark(x1, y1, "$\mathregular{T}$␣=\n%.2f" % T1 + u'␣\N{DEGREE␣SIGN}C')
        message.set_text("Click␣interface␣with␣freezing␣point␣" +
                                str(T2) + u'\N{DEGREE␣SIGN}C')
    if click == 1:
        x2, y2 = ix, iy
        mark(x2, y2, "$\mathregular{T}$␣=\n%.2f" % T2 + u'␣\N{DEGREE␣SIGN}C')
        message.set_text("Click␣interface␣with␣freezing␣point␣" +
                          str(T3) + u'\N{DEGREE␣SIGN}C')
    if click == 2:
        x3, y3 = ix, iy
        mark(x3, y3, "$\mathregular{T}$␣=\n%.2f" % T3 + u'␣
           \N{DEGREE␣SIGN}C')
        message.set_text("Click␣interface␣with␣freezing␣
           point␣" +
                          str(T4) + u'\N{DEGREE␣SIGN}C')
    if click == 3:
        x4, y4 = ix, iy
        mark(x4, y4, "$\mathregular{T}$␣=\n%.2f" % T4 + u'␣
           \N{DEGREE␣SIGN}C')
        message.set_text("Click␣interface␣with␣freezing␣
           point␣" +
                          str(T5) + u'\N{DEGREE␣SIGN}C')
    if click >= 4:
        x5, y5 = ix, iy
        mark(x5, y5, "$\mathregular{T}$␣=\n%.2f" % T5 + u'␣
           \N{DEGREE␣SIGN}C')

        fig2 = matplotlib.pyplot.figure(figsize=(2, 1.5))
        ax2 = fig2.add_subplot(111)
        ax2.spines['top'].set_visible(False)
        ax2.spines['right'].set_visible(False)
        y_locations = numpy.array([y1, y2, y3, y4, y5])
        smallest = min(y_locations)
        y_locations = y_locations - smallest  # shift pixel locations
        y_locations = y_locations * mm_per_pixel  # convert to millimeters
        ax2.set_xlabel("Distance␣(mm)")
        ax2.set_ylabel(u"Temp.␣(\N{DEGREE␣SIGN}C)")
        ax2.set_xticks([0, 1, 2, 3, 4, 5])
        ax2.set_yticks([0, -2, -4, -6])
        ax2.plot(y_locations, [T1, T2, T3, T4, T5], "ko")

        fullrange = max(y_locations) - min(y_locations)

        # first try linear (1st order polynomial)
        m, b, r_value, p_value, std_err = scipy.stats.linregress(
            y_locations, [T1, T2, T3, T4, T5])
        ax2.plot([-0.1 * fullrange, 1.1 * fullrange],
                [m * -0.1 * fullrange + b, m * 1.1 *
                   fullrange + b], "-k")
        log += "m:␣" + str(m) + "\n"
        log += "b:␣" + str(b) + "\n"
        log += "r-squared:␣" + str(r_value * r_value) + "\n
           "
        predictions = m * y_locations + b
        differences = abs([T1, T2, T3, T4, T5] -
           predictions)
        log += "T1␣actual␣" + str(T1) + "␣-␣predicted␣" + \
            str(predictions[0]) + "␣=␣" + str(differences
               [0]) + "\n"
        log += "T2␣actual␣" + str(T2) + "␣-␣predicted␣" + \
            str(predictions[1]) + "␣=␣" + str(differences
               [1]) + "\n"
        log += "T3␣actual␣" + str(T3) + "␣-␣predicted␣" + \
            str(predictions[2]) + "␣=␣" + str(differences
               [2]) + "\n"
        log += "T4␣actual␣" + str(T4) + "␣-␣predicted␣" + \
            str(predictions[3]) + "␣=␣" + str(differences
               [3]) + "\n"
        log += "T5␣actual␣" + str(T5) + "␣-␣predicted␣" + \
            str(predictions[4]) + "␣=␣" + str(differences[4]) + "\n"

        # second try quadratic (2nd order polynomial)
        A, B, C = numpy.polyfit(y_locations, [T1, T2, T3, T4, T5], 2)
        xs = numpy.arange(-0.1 * fullrange, 1.1 * fullrange, 0.1)
        ax2.plot(xs, A * xs * xs + B * xs + C, ":k")
        log += "A:␣" + str(A) + "\n"
        log += "B:␣" + str(B) + "\n"
        log += "C:␣" + str(C) + "\n"
        predictions = A * y_locations * y_locations + B * y_locations + C
        differences = abs([T1, T2, T3, T4, T5] - predictions)
        log += "T1␣actual␣" + str(T1) + "␣-␣predicted␣" + \
            str(predictions[0]) + "␣=␣" + str(differences[0]) + "\n"
        log += "T2␣actual␣" + str(T2) + "␣-␣predicted␣" + \
            str(predictions[1]) + "␣=␣" + str(differences[1]) + "\n"
        log += "T3 actual " + str(T3) + " - predicted " + \
            str(predictions[2]) + " = " + str(differences[2]) + "\n"
        log += "T4 actual " + str(T4) + " - predicted " + \
            str(predictions[3]) + " = " + str(differences[3]) + "\n"
        log += "T5 actual " + str(T5) + " - predicted " + \
            str(predictions[4]) + " = " + str(differences[4]) + "\n"

        ax2.set_xlim(-0.08 * fullrange, 1.08 * fullrange)
        fig2.subplots_adjust(left=0.28, right=0.95, top=0.95, bottom=0.3)
        fig2.savefig("fig_4b.pdf")
        matplotlib.pyplot.close(fig2)  # this keeps Matplotlib from hanging
        message.set_text(u"R-squared = %.2f " % (r_value * r_value))

        matplotlib.pyplot.close(fig2)  # this keeps Matplotlib from hanging
        fig.canvas.mpl_disconnect(mode4)
```

```
    fig.canvas.draw()
    click += 1


# get image scale
def mode3(event):
    global ix, iy, x1, x2, x3, x4, x5, y1, y2, y3, y4, y5, m1, m2, b1, b2
    global scale, mm_per_pixel, known_distance, message, log, click
    ix, iy = event.xdata, event.ydata
    if click == 0:
        x1, y1 = ix, iy
        edge_mark(x1, y1)
        message.set_text("Click␣the␣right␣edge␣of␣an␣adjacent␣channel")
    if click == 1:
        x2, y2 = ix, iy
        edge_mark(x2, y2)
        scale = x2 - x1
        log += "Scale:␣" + str(known_distance) + "␣mm␣=␣"
        log += str(scale) + "␣pixels\n"
        mm_per_pixel = known_distance / scale
        log += "Scale:␣" + str(mm_per_pixel) + "␣mm/pixel\n\n"
        message.set_text(
            u"Image␣scale␣is␣%.0f␣\N{GREEK␣SMALL␣LETTER␣MU}m/pixel" %
            (mm_per_pixel * 1000))
        fig.canvas.mpl_disconnect(mode3)
    fig.canvas.draw()
    click += 1


def measure_unknown(filename, freezing_point_1, freezing_point_2):
    setup_fig(filename)
    global T1, T2, click, message, log
    log += "measure_unknown:␣" + filename + "\n"
    click = 0
    fig.canvas.mpl_connect('button_press_event', mode1)
    T1 = freezing_point_1
    T2 = freezing_point_2
    message.set_text("Click␣first␣interface␣with␣freezing␣point␣" + str(T1) +
                     u'\N{DEGREE␣SIGN}C')
    matplotlib.pyplot.subplots_adjust(left=0, right=1, top=1, bottom=0)
    matplotlib.pyplot.show()


def calibrate_one_concentration(filename, freezing_point_1):
    setup_fig(filename)
    global T1, click, message, log
    log += "calibrate:␣" + filename + "\n"
    T1 = freezing_point_1
    click = 0
    fig.canvas.mpl_connect('button_press_event', mode2)
    message.set_text("Click␣first␣interface␣with␣freezing␣point␣" + str(T1) +
                     u'\N{DEGREE␣SIGN}C')
    matplotlib.pyplot.subplots_adjust(left=0, right=1, top=1, bottom=0)
    matplotlib.pyplot.show()
    matplotlib.pyplot.close(fig)


def calibrate_five_concentrations(filename,
                                  freezing_point_1,
                                  freezing_point_2,
                                  freezing_point_3,
                                  freezing_point_4,
                                  freezing_point_5):
    setup_fig(filename)
    global T1, T2, T3, T4, T5, click, message, log
    log += "linearity:␣" + filename + "\n"
    click = 0
    fig.canvas.mpl_connect('button_press_event', mode4)
    T1 = freezing_point_1
    T2 = freezing_point_2
    T3 = freezing_point_3
    T4 = freezing_point_4
    T5 = freezing_point_5
    message.set_text("Click␣interface␣with␣freezing␣point␣"
        + str(T1) +
                    u'\N{DEGREE␣SIGN}C')
    matplotlib.pyplot.subplots_adjust(left=0, right=1, top
        =1, bottom=0)
    matplotlib.pyplot.show()
    matplotlib.pyplot.close(fig)


def get_scale(filename, distance):
    setup_fig(filename)
    global known_distance, click, message, log
    log += "set_scale:␣" + filename + "\n"
    known_distance = distance
    click = 0
    fig.canvas.mpl_connect('button_press_event', mode3)
    matplotlib.pyplot.subplots_adjust(left=0, right=1, top
        =1, bottom=0)
    message.set_text("Click␣the␣right␣edge␣of␣one␣channel")
    matplotlib.pyplot.show()
    matplotlib.pyplot.close(fig)


def report(filename):
    global log
    f = open(filename, "w")
    f.write(log.encode("utf8"))
    f.close()
    log = u""
```

2.6.3 `microfluidic_thermometer.py`: Step-by-step description of using `microfluidic_thermometer.py` to perform the analyses in Fig 2.4.

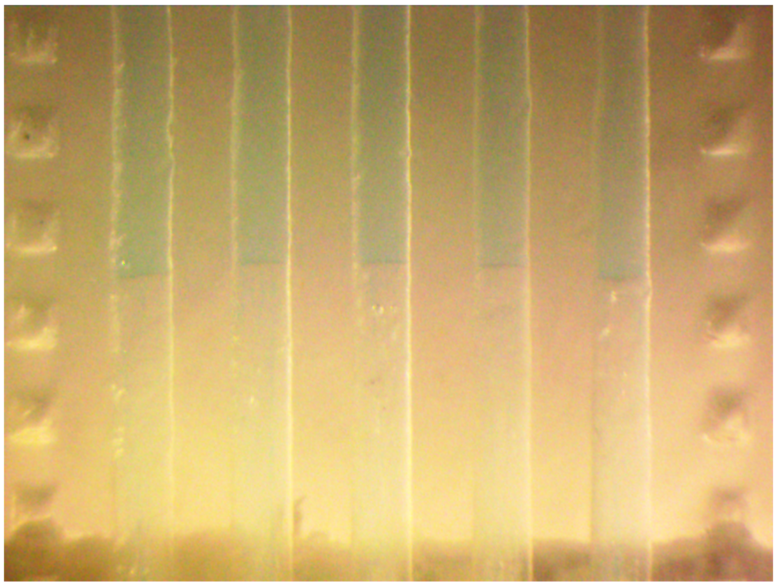
A custom Python program, `microfluidic_thermometer.py`, was written to analyze images of the microfluidic thermometer chip. The current version of the software is available as online Supplementary Information, and the latest version of the software is available for download from `http://groverlab.org`. Sample images of the microfluidic thermometer chip are included for use with the software. To use the software to measure the unknown freezing point of a solution, the user creates a short Python script describing the images to be analyzed. For example, to create Figure 4 in the main text, the following script was used:

```
import microfluidic_thermometer as mt

filename = "00000.png"
mt.get_scale(filename, distance=2.5)
mt.calibrate_one_concentration(filename,
                               freezing_point_1=0)

```

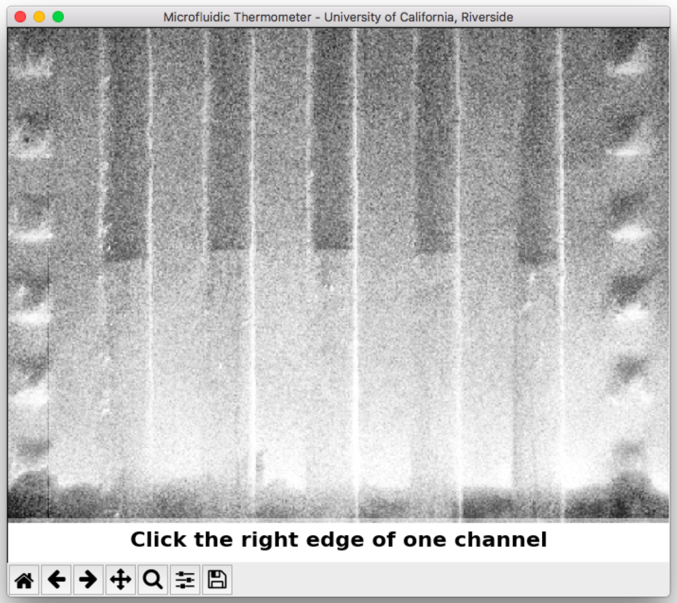
```
 8 filename = "86420.png"
 9 mt.get_scale(filename, distance=2.5)
10 mt.calibrate_five_concentrations(filename,
11                                  freezing_point_1=-5.08,
12                                  freezing_point_2=-3.70,
13                                  freezing_point_3=-2.41,
14                                  freezing_point_4=-1.19,
15                                  freezing_point_5=0.0)
16
17 filename = "08408.png"
18 mt.get_scale(filename, distance=2.5)
19 mt.measure_unknown(filename,
20                    freezing_point_1=0,
21                    freezing_point_2=-5.08)
22
23 mt.report("report.txt")
```

Line 1 imports `microfluidic_thermometer.py` (which should be in the same directory as this script).

Line 3 specifies the name of the first image file used in this analysis. In this case, the file `00000.png` is a photograph of the microfluidic thermometer chip with all five channels filled with water:

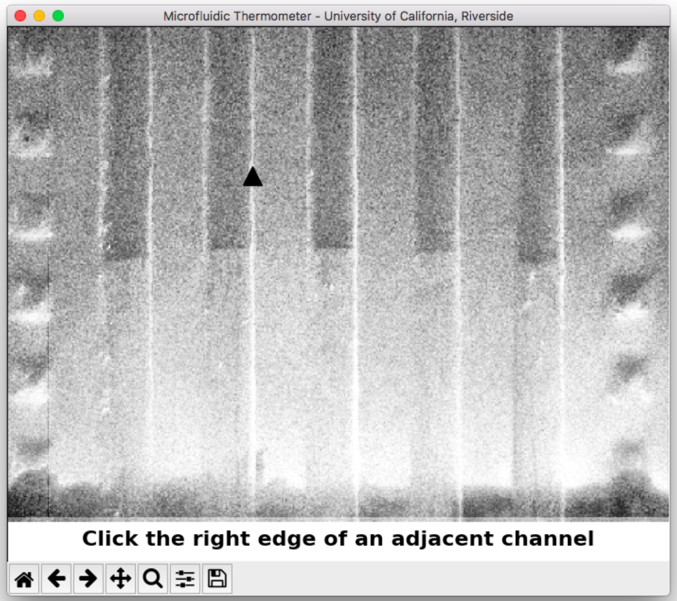


Line 4 calls the `get_scale` function, which is used to determine the scale of the image. This is necessary if subsequent images were acquired at a different magnification or pixel resolution than the current image. The argument `distance = 2.5` tells the code the actual length of a known feature on the chip; in this case the distance between two channel edges in the microfluidic thermometer is 2.5 millimeters. When this script is executed, Line 4 makes a window appear:


Microfluidic Thermometer - University of California, Riverside

Click the right edge of one channel

The window instructs the user to "click on the right edge of one channel." After the edge is clicked, a triangular marker is placed on the edge and the user is instructed to "click the right edge of an adjacent channel:


Microfluidic Thermometer - University of California, Riverside

Click the right edge of an adjacent channel

After the second edge is clicked, another triangular marker is placed on the edge, and the scale of the image is reported to the user ("Image scale is 19 μm/pixel") and is saved for future use:

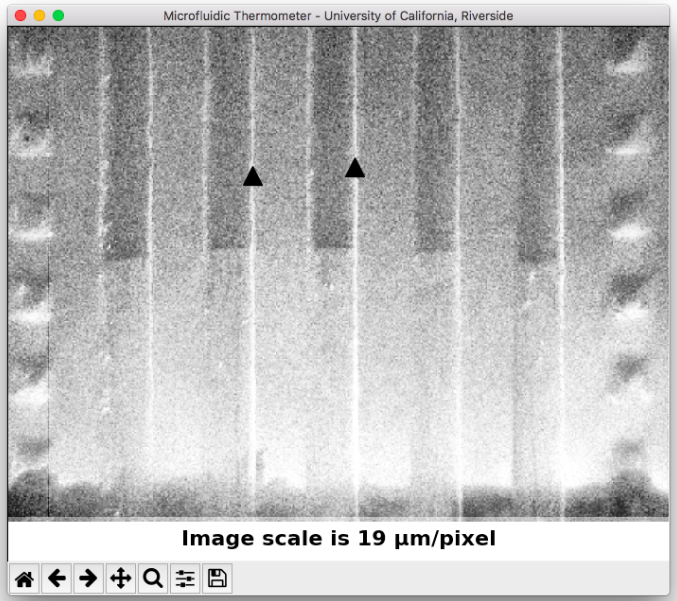
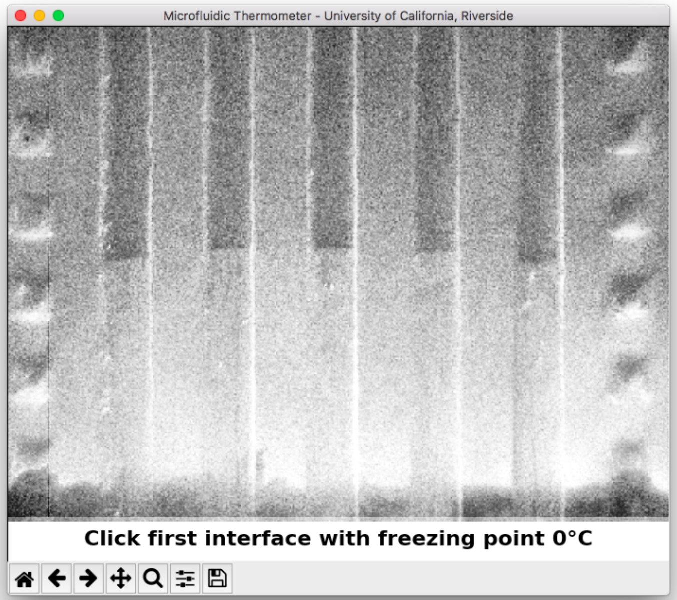

Microfluidic Thermometer - University of California, Riverside

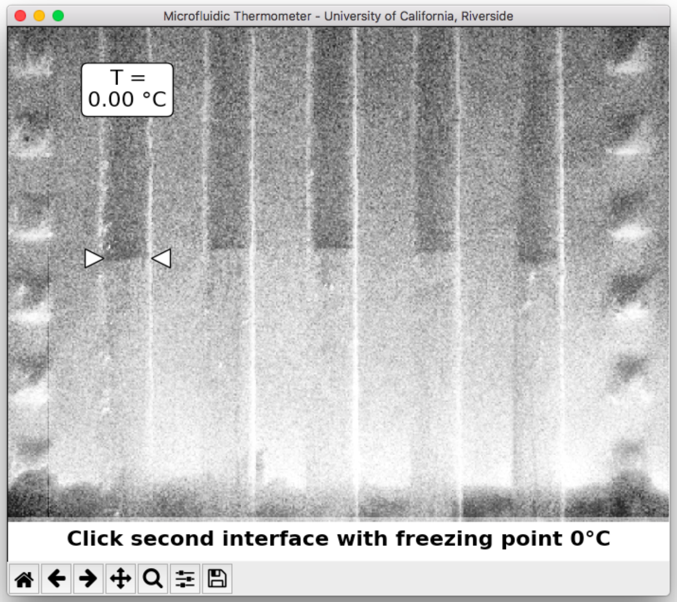
Image scale is 19 µm/pixel

Line 5 calls the `calibrate_one_concentration` function, which is used to analyze the uniformity of an isotherm across the channels in the thermometer chip. The argument `freezing_point_1 = 0.0` tells the code the known freezing point of the solution in all five channels of the chip; in this case the channels are filled with water so the freezing point is 0°C. The window instructs the user to "click the first interface with freezing point 0°C":


Microfluidic Thermometer - University of California, Riverside

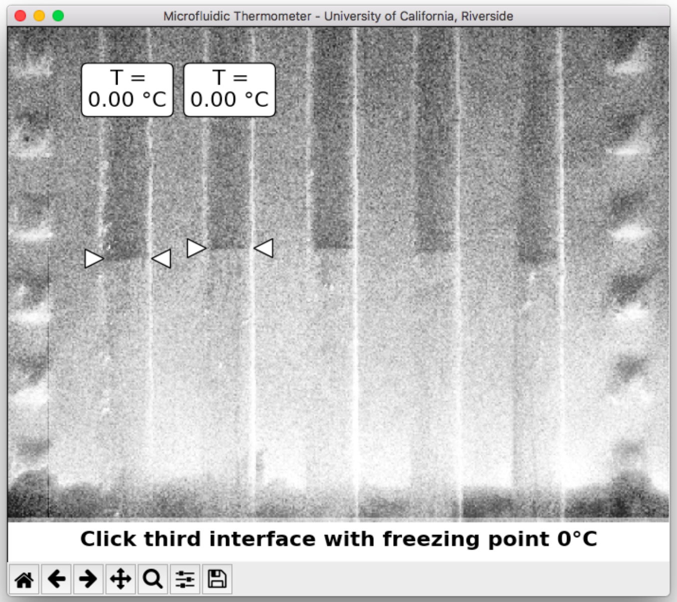
Click first interface with freezing point 0°C

After the user clicks on the solid-liquid interface in the first channel, triangular markers and a label are added to the plot to indicate the known temperature at the interface, and the user is instructed to "click second interface with freezing point 0°C":


Microfluidic Thermometer - University of California, Riverside
T =
0.00 °C

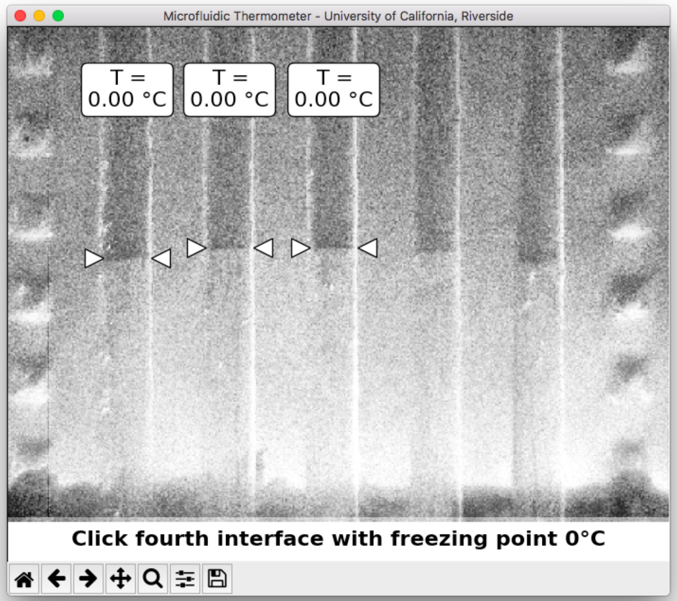
Click second interface with freezing point 0°C

This process repeats, instructing the user to click the third solid-liquid interface:


Microfluidic Thermometer - University of California, Riverside
T =
0.00 °C
T =
0.00 °C

Click third interface with freezing point 0°C

the fourth interface:


Microfluidic Thermometer - University of California, Riverside

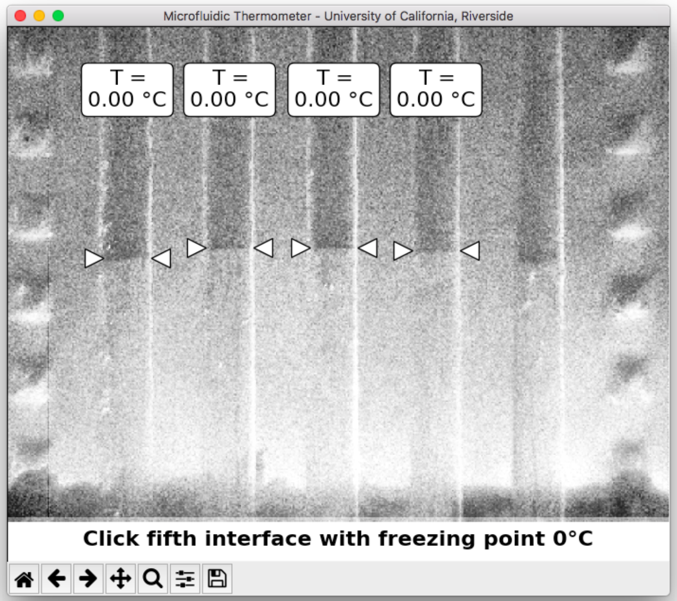
T = 0.00 °C

T = 0.00 °C

T = 0.00 °C

Click fourth interface with freezing point 0°C

and the fifth interface:


Microfluidic Thermometer - University of California, Riverside

T = 0.00 °C

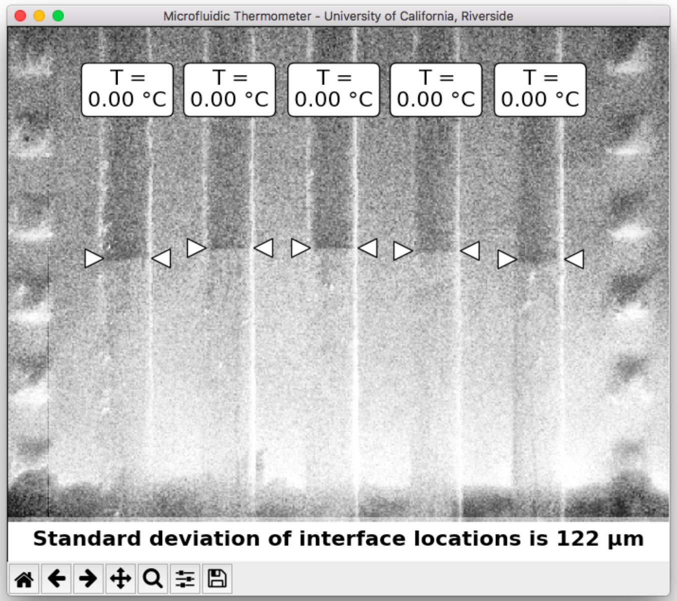
T = 0.00 °C

T = 0.00 °C

T = 0.00 °C

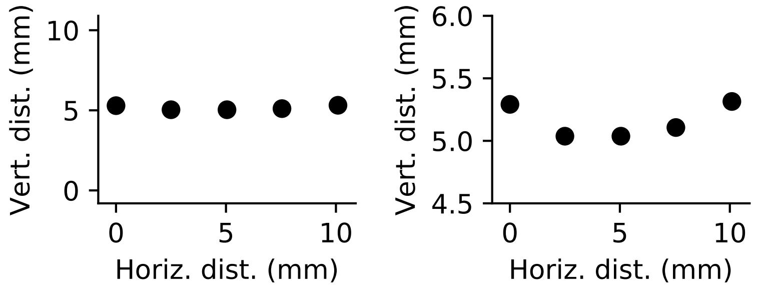
Click fifth interface with freezing point 0°C

The variation of the interface locations in the vertical dimension is reported to the user ("Standard deviation of the interface locations is 122 μm") and is saved for future use:


Microfluidic Thermometer - University of California, Riverside
T = 0.00 °C
T = 0.00 °C
T = 0.00 °C
T = 0.00 °C
T = 0.00 °C

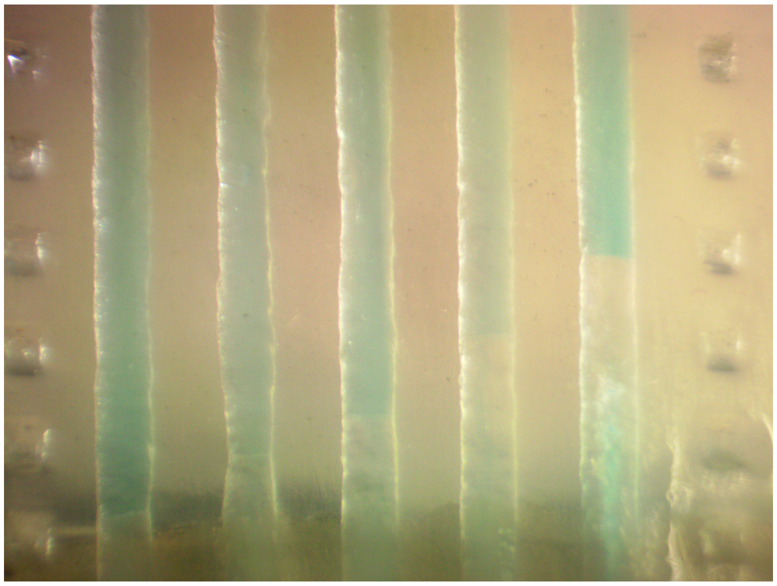
Standard deviation of interface locations is 122 µm

Finally, the software automatically saves two plots (normal and closeup) of the vertical location of each solid-liquid interface vs. the horizontal location of the interface:

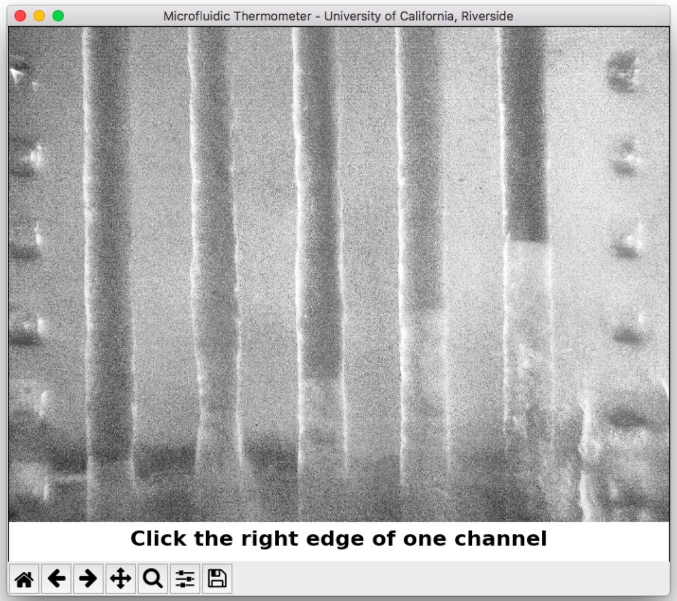


These plots confirm that isotherms are roughly linear in the measurement region of the thermometer chip, with a variation in vertical interface location of less than 500 μm over a 10 mm wide region.

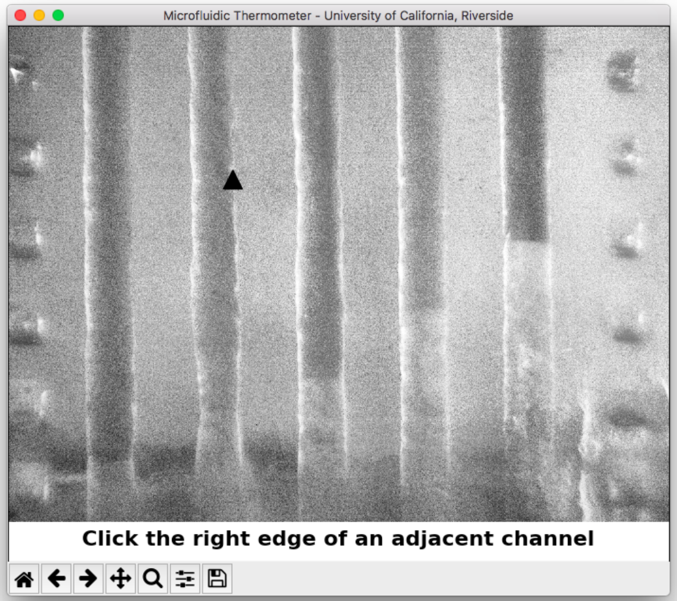
Line 8 specifies a new image file to be used. File `86420.png` is a photograph of the microfluidic thermometer with an 8% *w*/*w* solution of sodium chloride in channel A, a 6% solution in channel B, a 4% solution in channel C, a 2% solution in channel D, and a 0% solution (water) in channel E:



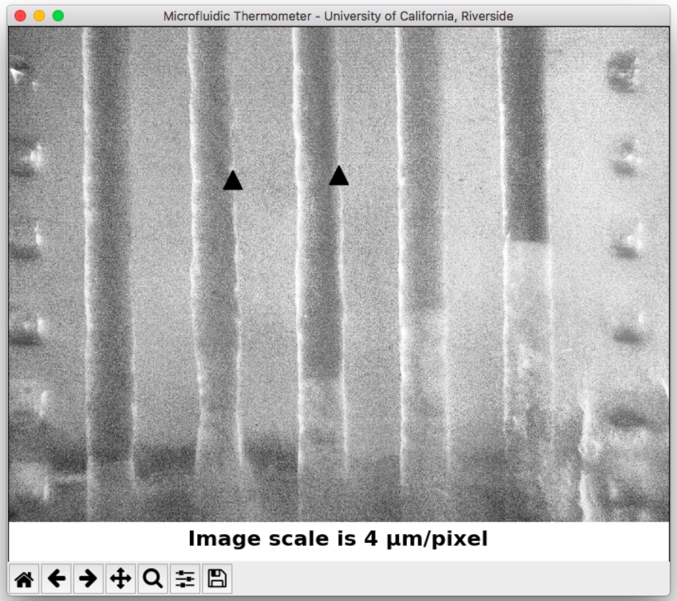
Line 9 again calls the `get_scale` function to determine the scale of the new image. The user clicks on two channel edges, and the scale is reported to the user and saved for future use:


Microfluidic Thermometer - University of California, Riverside

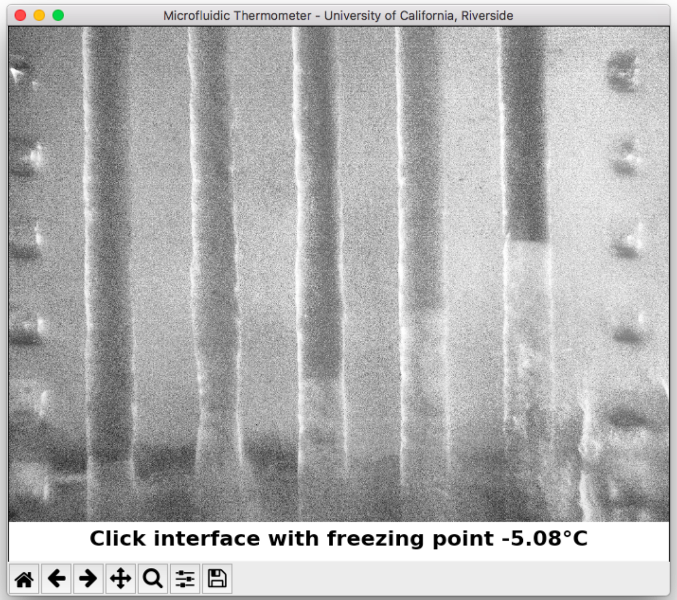
Click the right edge of one channel


Microfluidic Thermometer - University of California, Riverside

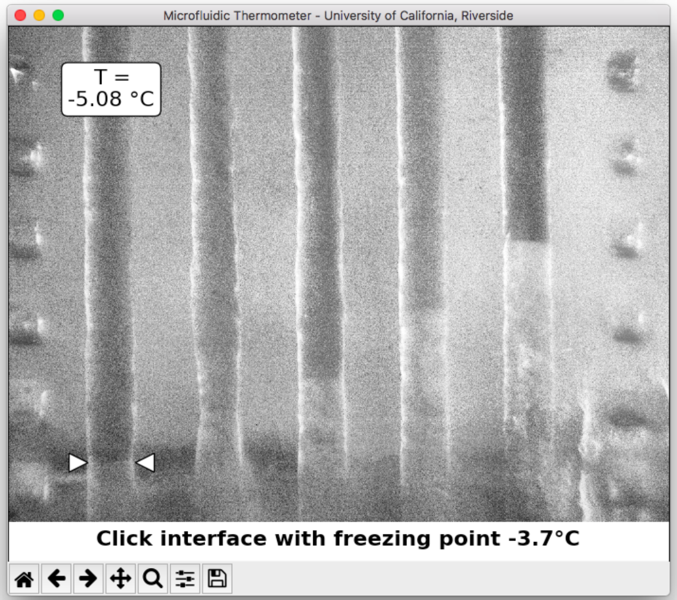
Click the right edge of an adjacent channel


Microfluidic Thermometer - University of California, Riverside
Image scale is 4 µm/pixel

Line 10 calls the `calibrate_five_concentrations` function, which is used to analyze the shape of the temperature gradient along the channels in the thermometer chip. The arguments `freezing_point_1 = -5.08`, `freezing_point_2 = -3.70`, `freezing_point_3 = -2.41`, `freezing_point_4 = -1.19`, and `freezing_point_5 = 0.0` specify the freezing points of the five sodium chloride solutions in channels A through E. The window then instructs the user to click on each of the five solid-liquid interfaces in turn:

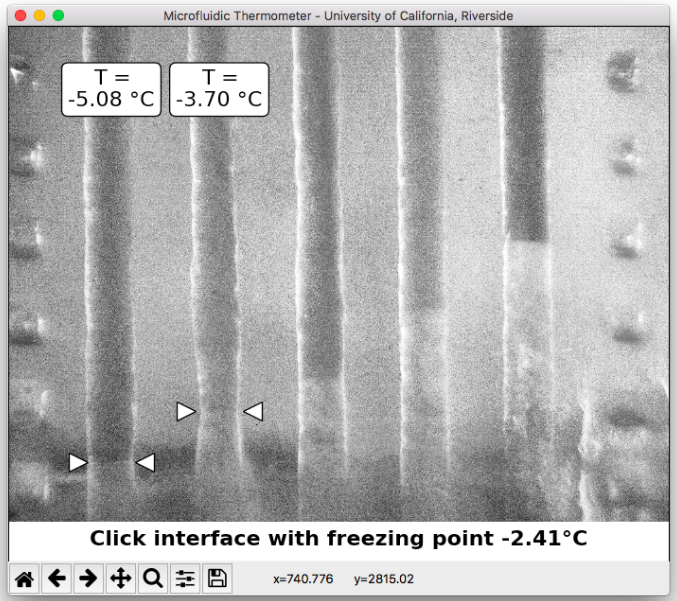

Microfluidic Thermometer - University of California, Riverside

Click interface with freezing point -5.08°C

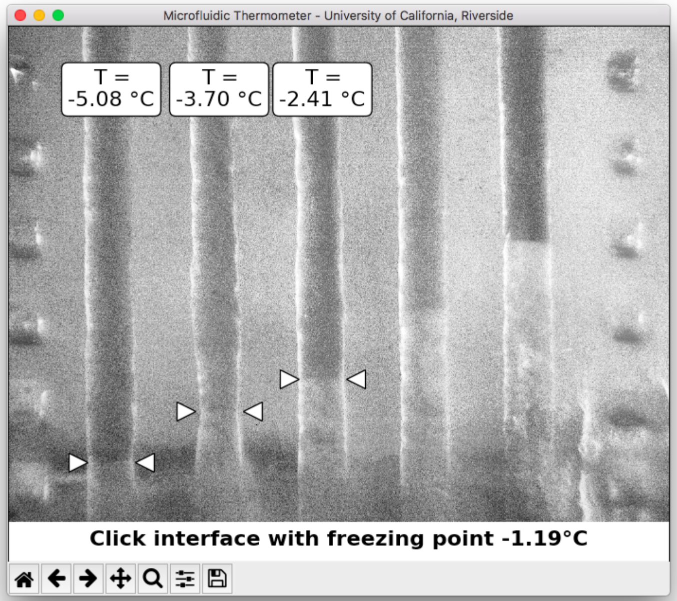

Microfluidic Thermometer - University of California, Riverside

T = -5.08 °C

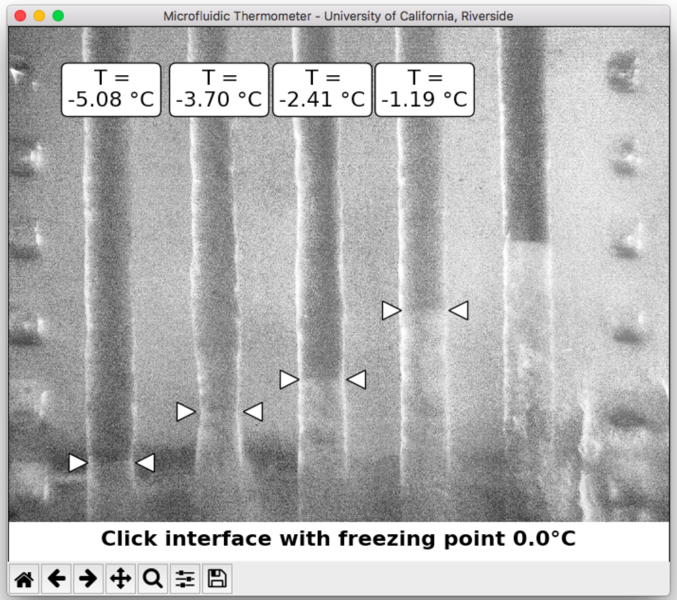
Click interface with freezing point -3.7°C


Microfluidic Thermometer - University of California, Riverside
T = -5.08 °C
T = -3.70 °C
x=740.776 y=2815.02

Click interface with freezing point -2.41°C

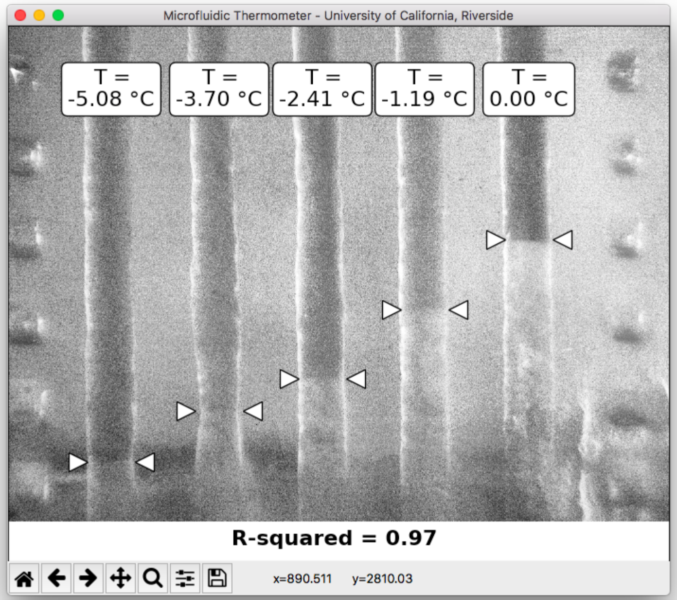

Microfluidic Thermometer - University of California, Riverside
T = -5.08 °C
T = -3.70 °C
T = -2.41 °C

Click interface with freezing point -1.19°C


Microfluidic Thermometer - University of California, Riverside

Click interface with freezing point 0.0°C

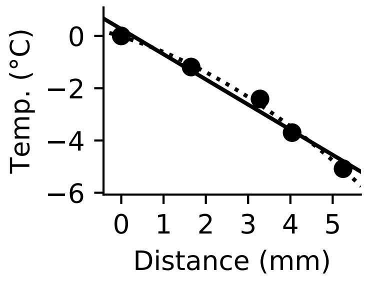
The R^2 value of the linear fit of a plot of interface location vs. temperature is reported to the user (“R-squared = 0.97”):


Microfluidic Thermometer - University of California, Riverside

R-squared = 0.97

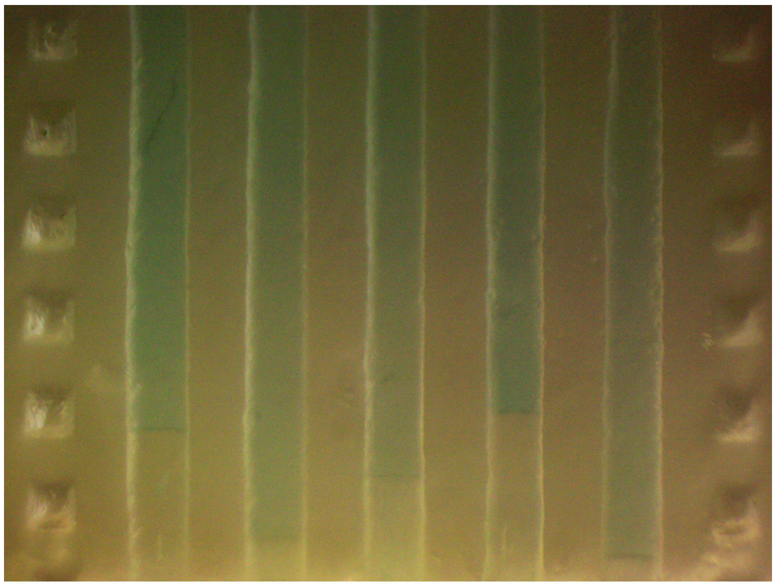
x=890.511 y=2810.03

Finally, the software automatically saves a plot of solid-liquid interface temperature vs. the location of that interface in the vertical direction on the chip:

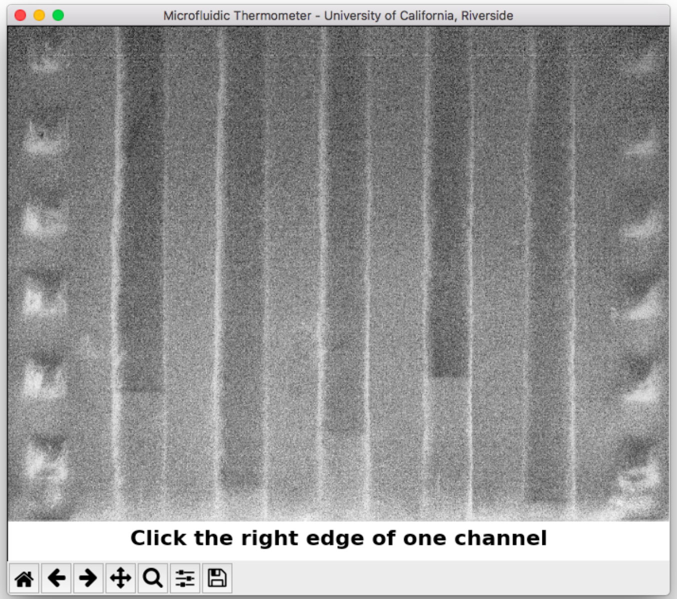


The plot includes two least-squares regression fits, one linear (the solid line, which deviates from the measured temperature values by less than 0.48 °C) and one second-order polynomial (the dotted line, which deviates from the measured temperature values by less than 0.22 °C). This plot confirms that the temperature gradient is roughly linear in the measurement region of the thermometer chip, and other functions (like the second-order polynomial) can be used if additional accuracy is needed).

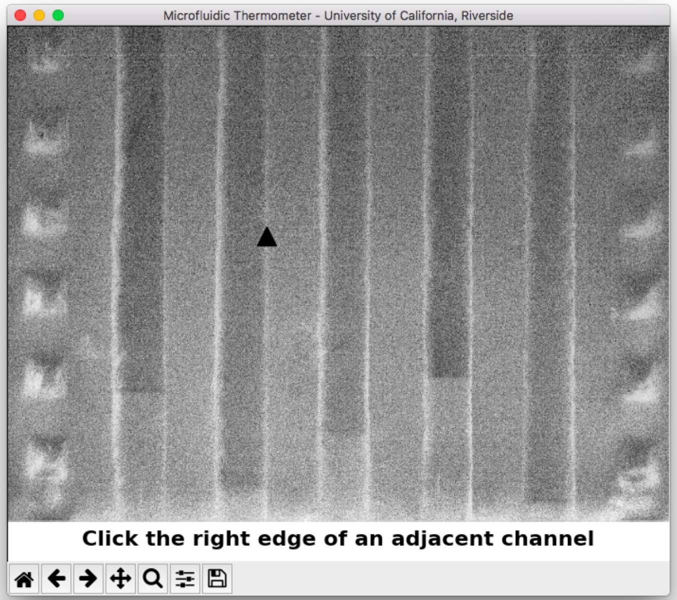
Line 17 specifies a new image file to be used in the rest of the analysis. In this case, the file `08408.png` is a photograph of the microfluidic thermometer chip with water in channels A and D, an 8% *w/w* solution of sodium chloride in channels B and E, and an solution with “unknown” freezing point in channel C:



Line 18 again calls the `get_scale` function to determine the scale of the new image. The user clicks on two channel edges, and the scale is reported to the user and saved for future use:


Microfluidic Thermometer - University of California, Riverside

Click the right edge of one channel


Microfluidic Thermometer - University of California, Riverside

Click the right edge of an adjacent channel

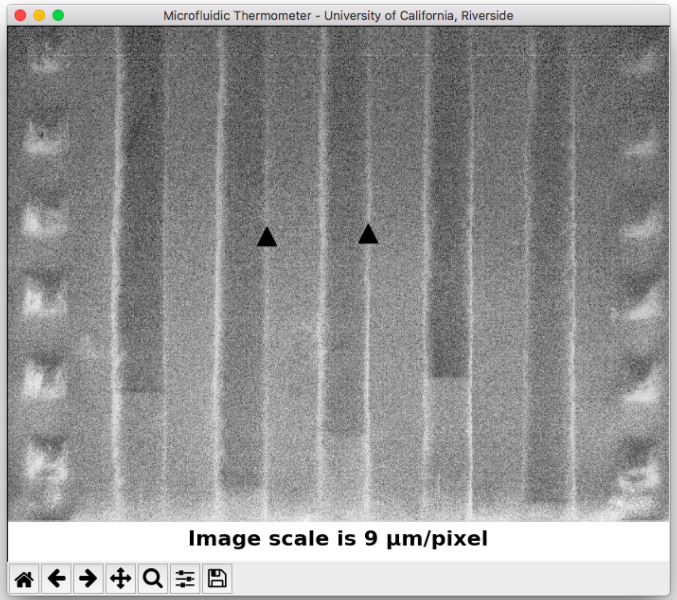
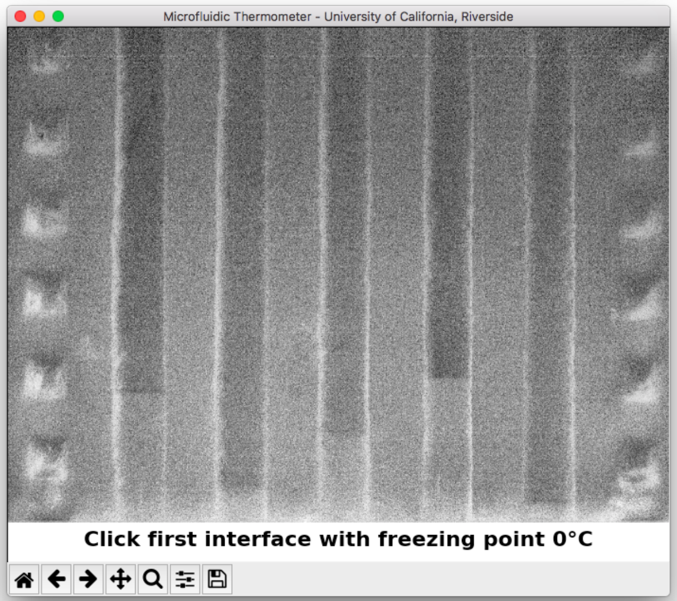

Microfluidic Thermometer - University of California, Riverside

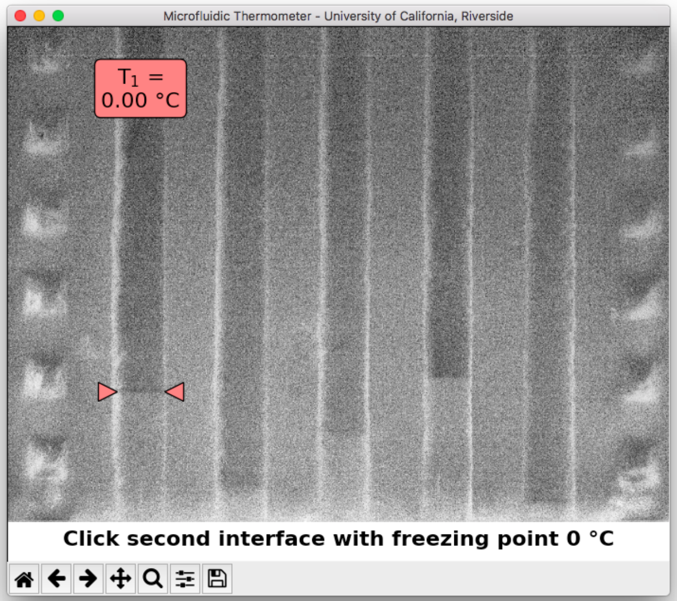
Image scale is 9 µm/pixel

Line 19 calls the `measure_unknown` function, which is used to measure the freezing point of a solution when the other four channels are filled with two solutions with known freezing points. The arguments `freezing_point_1 = 0.0` and `freezing_point_2 = -5.08` tell the code the known freezing point of the solution in channels A and D is 0°C and the known freezing point of the solution in channels B and E is −5.08°C. The window instructs the user to click the locations of each solid-liquid interface in the chip, first the interfaces at freezing point 0°C:



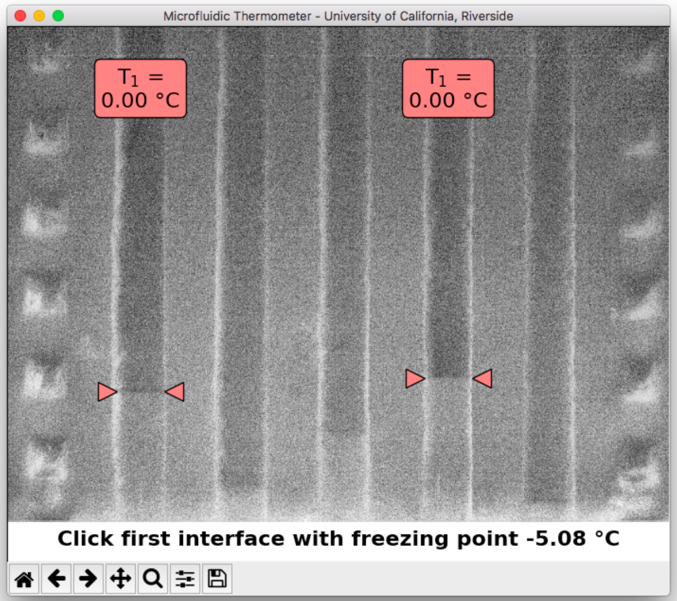
Microfluidic Thermometer - University of California, Riverside

Click first interface with freezing point 0°C

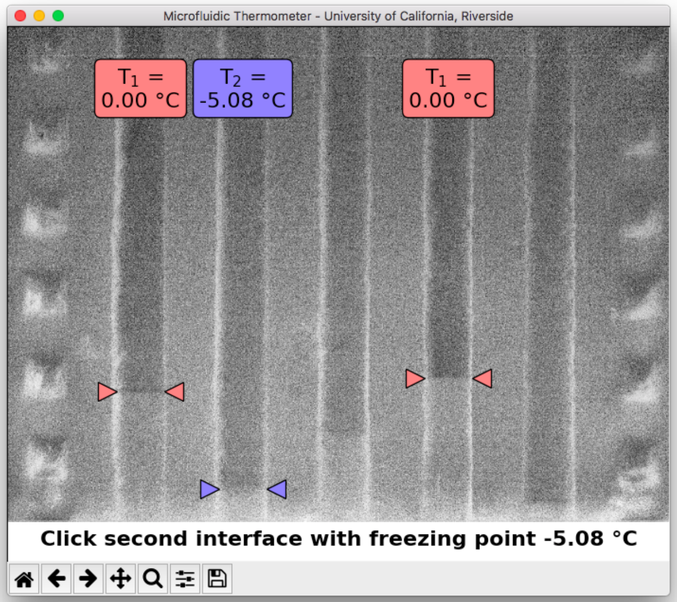

Microfluidic Thermometer - University of California, Riverside
T_1 =
0.00 °C

Click second interface with freezing point 0 °C

then the interfaces at freezing point $-5.08°C$:

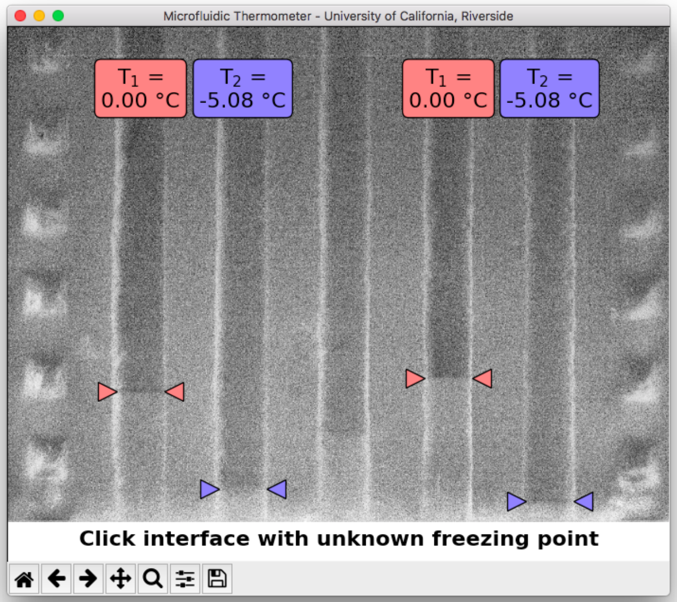

Microfluidic Thermometer - University of California, Riverside
T_1 =
0.00 °C
T_1 =
0.00 °C

Click first interface with freezing point -5.08 °C


Microfluidic Thermometer - University of California, Riverside
T_1 = 0.00 °C
T_2 = -5.08 °C
T_1 = 0.00 °C

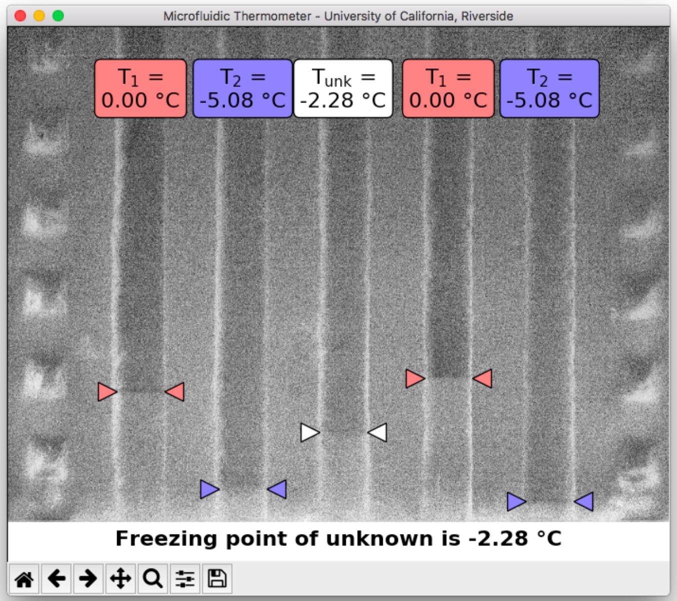
Click second interface with freezing point -5.08 °C

and finally the interface in the channel containing the “unknown” freezing point solution:


Microfluidic Thermometer - University of California, Riverside
T_1 = 0.00 °C
T_2 = -5.08 °C
T_1 = 0.00 °C
T_2 = -5.08 °C

Click interface with unknown freezing point

Finally, the freezing point of the "unknown" solution is reported to the user ("freezing point of unknown is −2.28°C"):



Freezing point of unknown is -2.28 °C

Line 23 uses the `report` function to save a summary of the analysis of these three images in a text file with the specified name.

Chapter 3

Chronoprints: Identifying samples by visualizing how they change over space and time

3.1 Introduction

Techniques for identifying a substance (or the components in a mixture) have many applications across a wide range of different fields. Modern tools of analytical chemistry like gas chromatography-mass spectrometry (GC-MS) are unparalleled in their ability to identify a substance or mixture. However, the size, cost, and complexity of these instruments limit their use in important applications in resource-limited settings. For example, around 10% of all medications in low- and middle-income countries are actually counterfeit and may be worthless (or even dangerous) to patients [39, 40], and while tools like GC-MS could easily detect these adulterated medicines, these tools are not readily available in the poorest parts of the world.

Different substances usually have different physical properties. In some cases, by measuring a physical property of a sample and comparing it to a known value for a pure substance, one can chemically identify the sample. We recently demonstrated two simple and low-cost techniques for measuring two intrinsic physical properties of samples, freezing/melting point [41] and density [2], and we successfully used those physical measurements to identify some samples. But many natural products, medicines, and other complex mixtures may not have a known freezing point or density. To identify or distinguish samples like these, simple measurements of their physical properties may not be enough.

In this work, we show that the way a sample's physical properties change over space and time can be used to chemically identify the sample. Under static and homogenous conditions, a sample's properties usually remain unchanged, so our method relies on inducing a change in the substance by perturbing it in some way. This perturbation could take many different forms, and in this work we used a rapidly-changing temperature gradient to perturb our samples. Different samples react to this perturbation in different ways (for example, in a temperature gradient, different samples might freeze, or thaw, or separate into their components, or change in other ways). Additionally, these changes can occur at different *locations* in different samples (if the perturbation is applied across the sample as a gradient of some sort) and at different *times* in different samples (if the perturbation is changing over time). The resulting multidimensional dataset of how a sample changes over space and time in response to a perturbation can serve as a “fingerprint” to identify the sample.

If a perturbation is always applied to a sample in exactly the same manner, then a sample’s resulting “fingerprint” should be consistent and could in theory be stored in a database and used to identify the same sample in the future. However, in practice, generating highly reproducible perturbations would likely require complex and costly hardware that would disqualify our technique from use in resource-limited settings. In this work, instead of trying to make the perturbation reproducible across experiments, we made sure that *each sample in a given experiment receives exactly the same perturbation.* In other words, we cannot necessarily compare sample fingerprints across multiple experiments, but we *can* compare fingerprints across multiple samples within the same experiment. We accomplished this using our “microfluidic thermometer chip,” [41] a simple microfluidic chip that holds several different microliter-scale samples in close proximity to each other. The chip (shown in Figures 3.1A and B) holds several liquid samples in long parallel microfluidic channels. When we apply a perturbation (like a dynamic temperature gradient) along these channels, each sample receives the same perturbation at the same point in space and time. If two samples in the same experiment display similar changes over space and time in response to the perturbation, then this suggests that the samples may be the same. But if two samples in the same run display significantly different changes over space and time, then this proves that the samples are different.

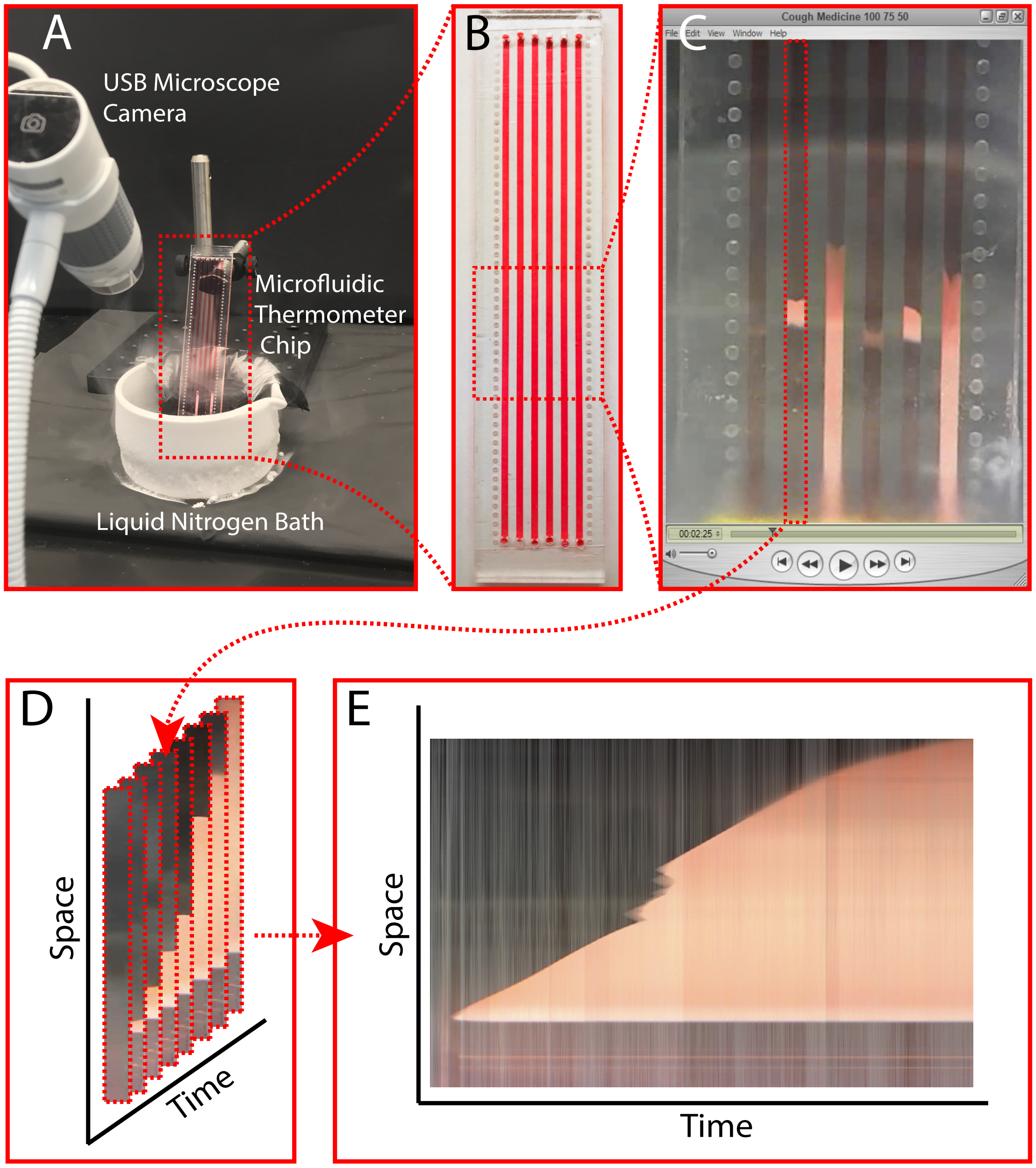

A
USB Microscope Camera
Microfluidic Thermometer Chip
Liquid Nitrogen Bath
B
C
Cough Medicine 100 75 50
File Edit View Window Help
00:02:25
D
Space
Time
E
Space
Time

Figure 3.1

Figure 3.1: Producing a "chronological fingerprint" or *chronoprint* capturing how six samples (in this example, authentic and adulterated samples of an over-the-counter liquid cold medicine) respond to a perturbation over space and time (in this case, a rapidly-changing temperature gradient). **(A)** A microfluidic thermometer chip containing the samples is partially immersed in liquid nitrogen to establish a rapidly-changing temperature gradient along the chip. **(B)** The chip contains six samples (red) loaded in microfluidic channels that run parallel to the dynamic temperature gradient. **(C)** An inexpensive USB microscope records a video of the physical changes in the samples as they react to the dynamic temperature gradient. **(D)** For each sample, our custom MATLAB code (available as *Supporting Information)* extracts an image of the entire channel from each frame of the video. **(E)** By reducing each channel image to a single column of pixels, then placing these columns side-by-side, we create a bitmap image (the sample's chronoprint) that captures how the sample changes over space (the y-axis) and time (the x-axis). Finally, by comparing the chronoprints of all six samples in the chip, we can determine whether the samples are either likely the same or definitely different.

The data resulting from this process—the change in each sample at each point in space and time as the sample is perturbed—are multidimensional and challenging to analyze, but we developed a simple method for comparing different samples. In this work, we record a movie of the thermometer chip in action, then convert that movie into bitmap images (one per sample) that capture the way each sample changes over space and time. We call these images "chronoprints," a portmanteau of *chronological* (the image captures a sample's changes over time) and *fingerprint* (the image serves to identify a substance within a given experiment). Figure 3.1 summarizes the process of obtaining a chronoprint from a complex sample (in this case, six different samples of an over-the-counter cold medicine exposed to a perturbation consisting of a dynamic temperature gradient). Since a chronoprint is fundamentally just a bitmap image, chronoprints can be compared using image similarity analysis algorithms from computer science. In this work we demonstrate three different approaches to quantifying chronoprint similarity: *feature tracing* (which reduces each chronoprint to a curve and is suitable for simpler chronoprints), *image differences* (which calculates the sum of the pixel-by-pixel differences between two chronoprints), and *image hashing* (which converts each chronoprint to a 64-bit representation called a "hash").

To demonstrate the versatility of chronoprints, we used them here to distinguish between authentic and adulterated foodstuffs, identify adulterated or counterfeit medication, and distinguish between toxic and nontoxic pharmaceutical ingredients. But these are just a few of the different samples that could be analyzed using chronoprints—in principle any sample that responds to a perturbation by changing its appearance could be analyzed using chronoprints.

3.2 Experimental Section

3.2.1 Fabricating microfluidic thermometer chips

Microfluidic fluidic thermometer chips[41] were designed in Adobe Illustrator (Adobe Systems Inc., San Jose, CA). Each thermometer chip is 125 mm long, 25 mm wide, and contains six parallel microfluidic channels. Each channel is 1.5 mm wide, 0.5 mm deep, and 115 mm long, with 2.5 mm diameter input/output reservoirs at each end, 1.5 mm space between channels, and markers spaced every 1 mm along the sides of the chip for length measurements. The chip design was exported as a DXF file (available as online *Supporting Information*) and engraved into 3 mm thick poly(methyl methacrylate) pieces (Professional Plastics Inc., Fullerton, CA) using a computer-controlled hobbyist-grade milling machine (Bantam Tools, Berkeley, CA). The open channels were enclosed by applying PCR tape (Bio-Rad Laboratories, Hercules, CA) to the chip.

3.2.2 Preparing samples

Several different types of liquid samples were analyzed in this work. To apply our technique to the problem of counterfeit food products, we obtained chronoprints from samples of two pure food oils, extra virgin olive oil (Wal-Mart Stores Inc., Bentonville, AR) and unrefined peanut oil (Spectrum Organic Products, Petaluma, CA), as well as a 1:1 (v/v) mixture of the two oils that served as a adulterated oil sample. To explore the ability of our technique to determine the authenticity of medications, we obtained chronoprints from six different lots of NyQuil Severe Cold and Flu medicine (a liquid medication containing acetaminophen, phenylephrine, doxylamine succinate, dextromethorphan, and glycerol). The drugs had expiration dates spanning a four-month period from July to October 2019. Additionally, to simulate the detection of an adulterated (diluted or watered down) medicine, we prepared and obtained chronoprints from 50%, 75%, 90%, and 95% (v/v) dilutions of NyQuil Severe Cold and Flu medicine in water. Finally, to show that our technique can distinguish two occasionally-confused chemicals in pharmaceutical manufacturing, we obtained chronoprints from samples of diethylene glycol (a transparent and sweet-tasting but poisonous liquid) and glycerol (a similar but nonpoisonous liquid) from Sigma-Aldrich (St. Louis, MO). About 75 μL of each sample was loaded into the thermometer chip for each experiment.

3.2.3 Obtaining chronoprints

Once a thermometer chip was filled with samples to analyze (Figure 3.1B), one end of the chip was partially submerged in a liquid nitrogen bath while recording a video of the chip contents using an inexpensive USB microscope (Figure 3.1A; Monoprice, Rancho Cucamonga, CA). This created a dynamic temperature gradient that quickly cooled the lower regions of the thermometer chip, then slowly cooled the rest of the chip over the next few seconds. All six sample channels in the chip were exposed to the same changing temperature gradient. After about 80 seconds for the oil samples and 160 seconds for the cold medicine samples, no further changes were observed and the video recording was ended (Figure 3.1C). A custom MATLAB script (*Supporting Information*) was then used to convert each video into six chronoprints (one per sample). For each sample, the script extracts an image of the entire microfluidic channel from each frame of the video (Figure 3.1D). The script then averages each row of pixels in each channel image to convert it to a single column of pixels. By then placing all of these columns of pixels side-by-side, the script creates a bitmap image that is the sample's chronoprint, with space (distance along the channel) in the vertical dimension and time in the horizontal dimension (Figure 3.1E). This process is then repeated for each sample in the experiment, and the resulting chronoprints are ready for comparison and similarity analysis.

3.2.4 Comparing chronoprints

Since chronoprints are just bitmap images, they can be compared using a variety of different techniques, including image similarity algorithms developed by computer scientists.[42] In this work, we used three different techniques to compare chronoprints: *feature tracing*, *image differences*, and *image hashing*. In this manuscript, we show results from only one comparison technique for each experiment, but results from using the other techniques to analyze these experiments' data (plus several additional experiments' data not shown in the main text) are provided in *Supporting Information*.

3.2.4.1 Comparing chronoprints using feature tracing

For simpler chronoprints with just one or two dominant features, one can simply trace the boundary between these features and convert each chronoprint to a curve; these curves can then be compared to each other to quantify the similarity of the samples. An example of this *feature tracing* approach for chronoprint comparison is shown in Figure 3.2. In this process, a custom MATLAB program (*Supporting Information*) first enhances contrast by taking the pixel values of the first frame (the first column of pixels in the chronoprint), halving these values, and subtracting the result from each remaining column of pixels in the chronoprint. This process helps remove background noise that affects each frame of the movie (and therefore each column of the chronoprint) approximately equally. The program then converts each chronoprint from color to monochrome (Figure 3.2A), then the program compares the value of each pixel to a constant threshold provided by the user; pixels with values below that threshold are colored solid black, and pixels with values above that threshold are colored solid white (Figure 3.2B). The program then traces the boundary between the black and white pixels and converts this trace into a curve (Figure 3.2C). Rarely, a column of pixels is encountered where the program fails to find the interface between the black and white pixels; in these cases the program reuses the last successful interface location from the previous column of pixels. Finally, the curve is smoothed slightly using a Savitzky-Golay filter [43, 44] (3rd order polynial; 31 point full window width). If two curves are similar, this suggests that the two samples analyzed may be the same; but if two curves are significantly different, this is proof that the samples are chemically different. The degree of similarity between two samples is quantified by summing the squared differences between the y-axis values of the curves (the distances along the channel) at each point along the curves.

3.2.4.2 Comparing chronoprints using image differences

While the *feature tracing* method described above works well for simpler chronoprints, more complex chronoprints cannot be easily reduced to simple curves for comparison. For these chronoprints, we compare the bitmap images directly. The *image differences* method for chronoprint comparison calculates the sum of the pixel-by-pixel differences be-

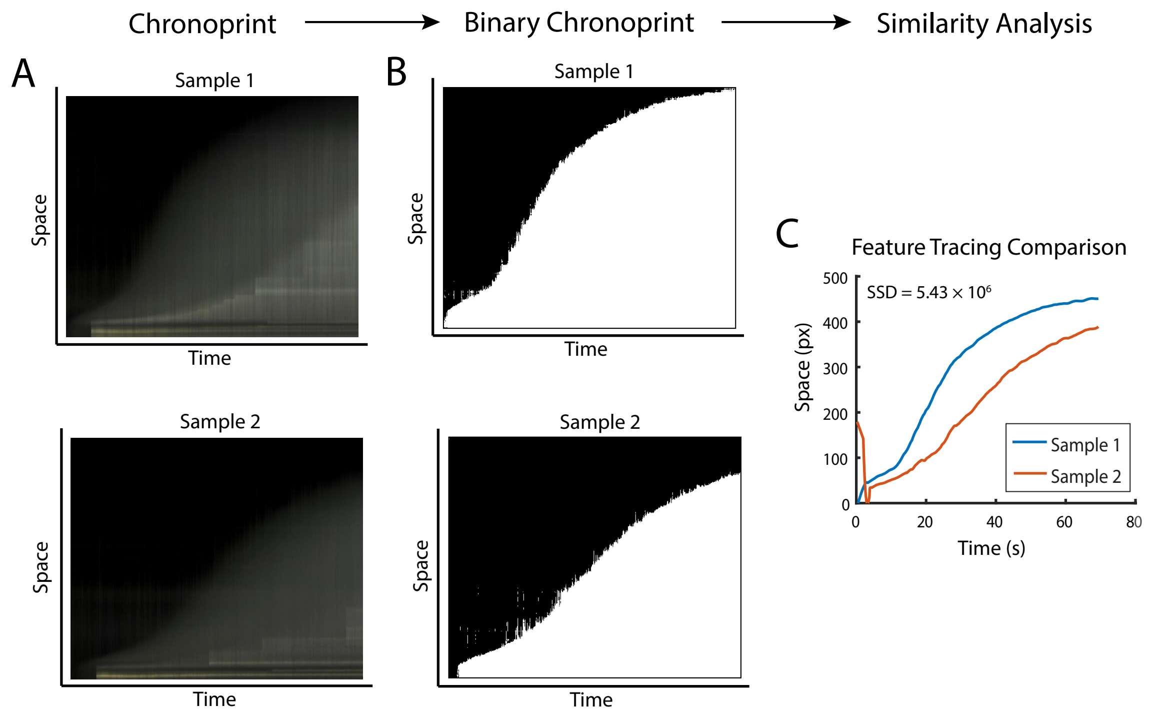


Figure 3.2: Overview of the *feature tracing* method of comparing chronoprints (in this case, obtained from two different food oils). Monochrome chronoprints of each sample **(A)** are converted to binary chronoprints **(B)** by comparing each pixel value to a constant threshold value; pixels above the threshold are colored white, and pixels below the threshold are colored black. Our code (available as online *Supporting Information*) then traces the boundary between white and black pixels on each binary chronoprint, and the resulting traces are smoothed slightly and plotted together to compare the two chronoprints **(C)**. Traces that are significantly different (like these) confirm that the two samples are chemically different. The sum of squared differences (SSD) between the y-axis values of the curves at each point along the curves (5.43×10^6 in this case) serves to quantify the degree of similarity between the two samples of food oils.

tween reduced-resolution versions of two chronoprints. An example of using the *image differences* process is shown in Figure 3.3. In this process, each chronoprint (Figure 3.3A) is first converted from color to monochrome, then the spatial resolution of each chronoprint is downsampled to 8 by 8 pixels (Figure 3.3B). Each of the chronoprint's 64 pixels now has a value between 0 (black) and $2^8 - 1 = 255$ (white). To compare two chronoprints, the absolute value of the difference between the pixel values at each pixel location is calculated, and the sum of these values represents the *image difference* score for the two images (Figure 3.3C). An *image difference* score of 0 indicates that the two chronoprints are exactly identical. The highest possible *image difference* score, $(2^8 - 1) \times 64 = 16{,}320$, corresponds to comparing an all-white chronoprint with an all-black chronoprint. In practice, we found that a threshold of about 1500 separated most sample pairs that are identical (*image difference* score < 1500) from sample pairs that were different (*image difference* score > 1500).

3.2.4.3 Comparing chronoprints using image hashing

The third chronoprint comparison method we used, *image hashing*, is shown in Figure 3.4. This method converts each chronoprint to a reduced-size binary representation (a "hash") that can then be compared to other chronoprints' hashes. The process starts by using the 8 by 8 pixel monochrome version of the chronoprints created in the *image differences* method above. Then, each pixel is converted to either solid white or solid black depending on whether its value lies above or below a threshold (Figure 3.4B). In this study, we explored four different values for this threshold:

- **Local Mean:** the average pixel value in each chronoprint was used as the threshold.

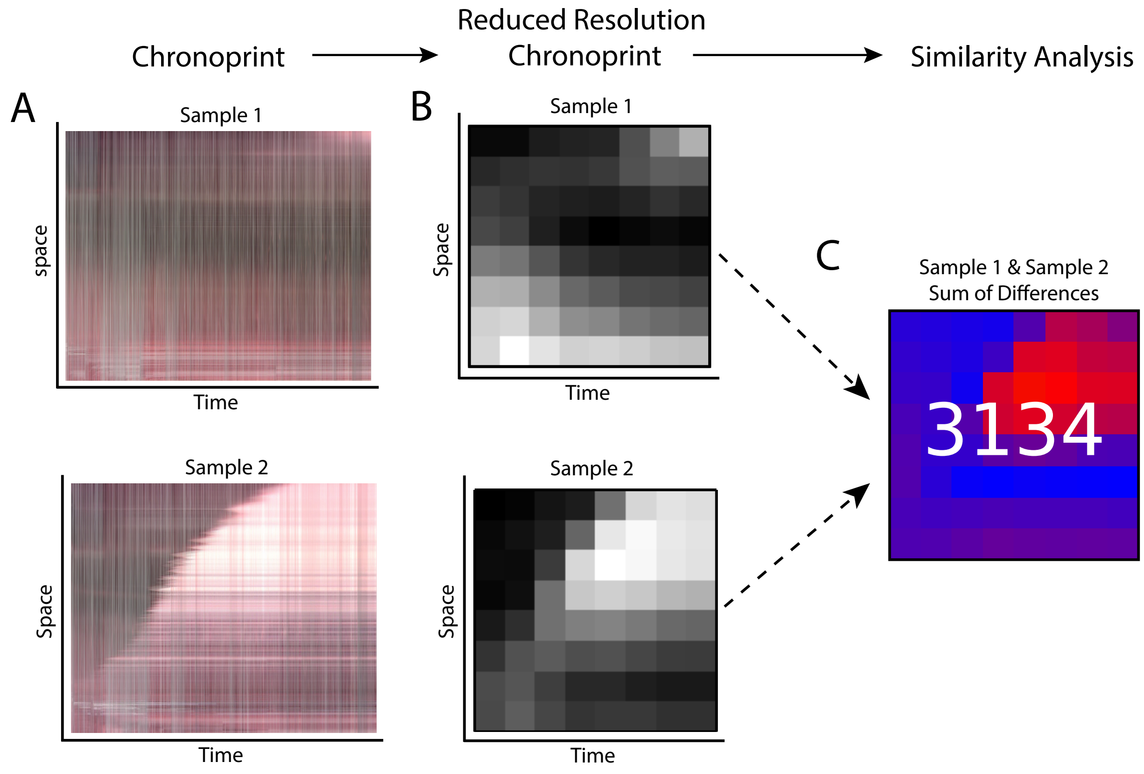


Figure 3.3: Overview of the *image differences* method of comparing chronoprints (in this case, obtained from authentic and diluted samples of liquid cold medicine). Chronoprints of each sample **(A)** are converted to reduced-resolution (8 $\times$ 8 pixel) monochrome chronoprints **(B)**. The monochrome chronoprints are then compared by calculating the difference between the pixel values at each location; the resulting image **(C)** shows which regions of the chronoprints are similar (blue) and which are different (red). The sum of these pixel difference values (3134 in this example) quantifies the similarity of these chronoprints on a scale from 0 (completely identical) to 16,320 (completely different). In practice, we found that a threshold of about 1500 generally separates the *image differences* scores of identical substances from different substances, so the *image differences* score of 3134 in this example is significantly greater than 1500 and confirms that these two samples of cold medicine are chemically different.

- **Local Median:** the median pixel value in each chronoprint was used as the threshold.
- **Global Mean:** the average pixel value across all six chronoprints in an experiment was used as the threshold.
- **Global Median:** the median pixel value across all six chronoprints in an experiment was used as the threshold.

Once a chronoprint is converted to an 8 by 8 binary image, it has effectively been reduced to a 64-bit "hash" of the original chronoprint. To calculate the similarity between two image hashes, our software interprets white pixels as binary "1" or TRUE and black pixels as "0" or FALSE, then calculates the exclusive OR (also called XOR) of each pixel pair between the images. If two pixels in the same location in two image hashes are the same (that is, they are both white or both black), then the result of the XOR of the pixel values is always 0 (that is, 0 XOR 0 = 0 and 1 XOR 1 = 0). However, if the two pixels are different (if one is black and the other is white), then the result of the XOR of the pixel values is always 1 (that is, 1 XOR 0 = 1 and 0 XOR 1 = 1). By then adding up the sum of all 64 pixel-wise XOR operations, we obtain the two chronoprints' *image hashing* similarity score (Figure 3.4C). This value ranges from 0 (for two chronoprints with identical image hashes) to 64 (for chronoprints with exactly opposite image hashes).

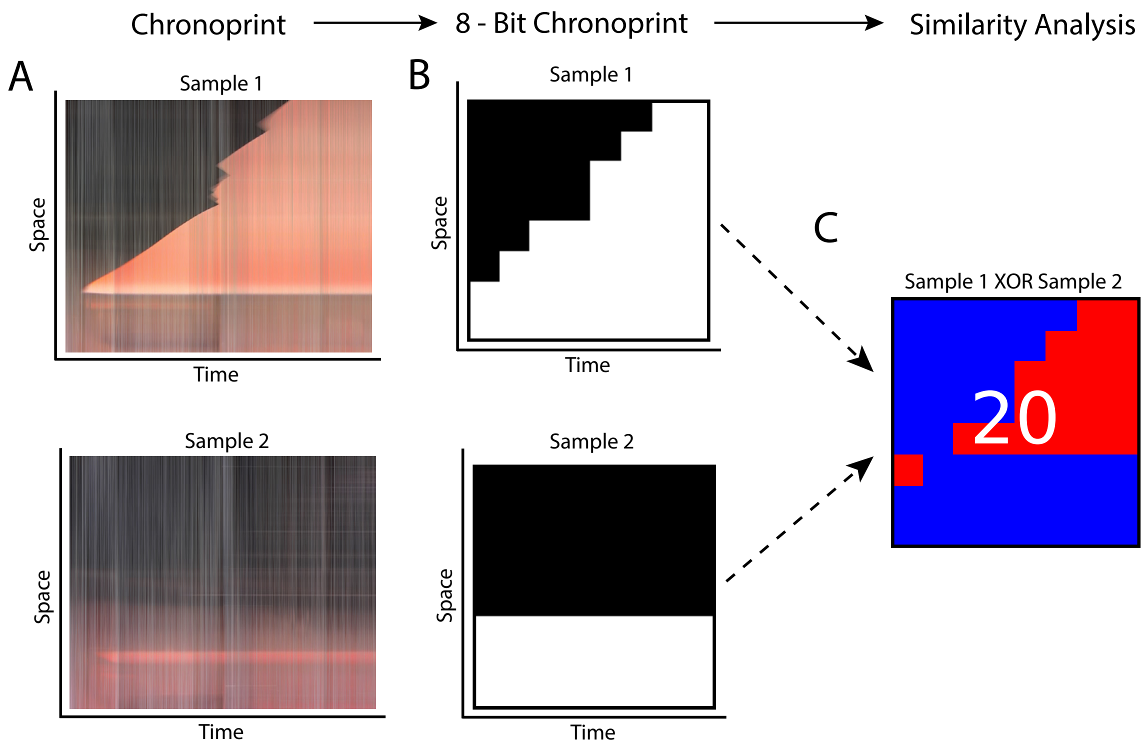


Figure 3.4: Overview of the *image hashing* method of comparing chronoprints (in this case, obtained from authentic and diluted samples of liquid cold medicine). Chronoprints of each sample **(A)** are converted to reduced-resolution (8 × 8 pixel) binary chronoprints **(B)** by comparing each pixel value to a constant threshold. The binary chronoprints are then compared by computing the exclusive OR (XOR) of the pixels at each location in the binary chronoprints, interpreting black = binary "0" or FALSE and white = binary "1" or TRUE. The resulting XOR image **(C)** is shown with blue pixels wherever the chronoprints are similar and red pixels wherever the chronoprints are different. The number of red pixels in the XOR image, the *image hashing* score (20 in this example), quantifies the degree of similarity of these chronoprints on a scale from 0 (completely identical) to 64 (completely different). In practice, we found that XOR images with more than about 10 red pixels corresponded to chronoprint pairs from different samples, so the *image hashing* similarity score of 20 in this example confirms that these two samples of cold medicine are chemically different.

3.3 Results and discussion

3.3.1 Identifying food fraud

The intentional tampering, substitution, or dilution of food or ingredients, also know as food fraud, is a widespread problem that costs consumers and the food industry \$10 billion to \$15 billion a year worldwide. [45] In some cases, food ingredients are substituted or diluted with potentially dangerous or toxic alternates, thereby producing a serious public health concern. For example, in 2008, 22 food companies in China used the toxic compound melamine, commonly used to produce plastic resins, in infant formula to artificially inflate the apparent protein content of their products. This resulted in six infant deaths and nearly 300,000 illnesses [45, 46, 47, 48]. In response to the significant economic and health impact of food fraud, the Grocery Manufacturing Association and the United States Congressional Research Service recommend testing food products during and after their production and suggest that authenticating ingredients is the best way to detect adulteration [45, 47].

Olive oil was found to be one of the most commonly adulterated food products worldwide between the years 1980 and 2010, [49] and the University of California, Davis' Olive Center reported in 2010 that 69% of imported olive oils and 10% of California olive oils labeled “extra virgin” did not meet the legal standard.[50] In some cases, “extra virgin” olive oil is diluted with other less expensive oils such as sunflower seed and peanut oils, which pose serious health risks to individuals who are allergic to these foodstuffs. [47, 48]

To determine if chronoprints can be used to identify adulterated food oils, we used our technique to analyze various samples of pure oils and oil mixtures. Since these samples resulted in relatively simple chronoprints, we used the *feature tracing* comparison technique to convert each chronoprint into a curve and quantify sample similarity. We began by loading a thermometer chip with six identical samples of 100% extra virgin olive oil before partially submerging the chip in liquid nitrogen, recording a video of the chip as it cools, and converting the video into six chronoprints. The results from performing *feature tracing* analysis on each chronoprint are shown in Figure 3.5A. Since all six samples were identical, we expected the resulting curves to be very similar, and this is indeed the case. The maximum sum-of-squared-differences between the curves, 4.22×10^5, is relatively low and indicates that the samples are likely identical. In another experiment, we analyzed six identical samples of a different oil (100% unrefined peanut oil). As expected, the resulting curves are again nearly identical, with a maximum sum-of-squared-difference of 1.73×10^5. These and other results support our claim that all six samples in the microfluidic thermometer receive the same perturbation, and if the samples are identical, then the resulting chronoprints will be very similar (with sum-of-squared differences between their *feature tracing* curves less than about 1×10^6).

To determine whether different samples produce different chronoprints in the same experiment, we loaded two samples each of three different oils into a thermometer chip. Channels 1 and 4 contained pure peanut oil, channels 3 and 6 contained pure olive oil, and channels 2 and 5 contained a 1:1 (v/v) mixture of olive and peanut oils. We then obtained chronoprints for each sample and analyzed them using the *feature tracing* method; the resulting curves are shown in Figure 3.5C. Within each oil type, each pair of samples resulted in very similar curves: the sum-of-squared differences was only 5.98×10^4 for the two olive oil samples, 2.55×10^5 for the two peanut oil samples, and 1.04×10^5 for the two

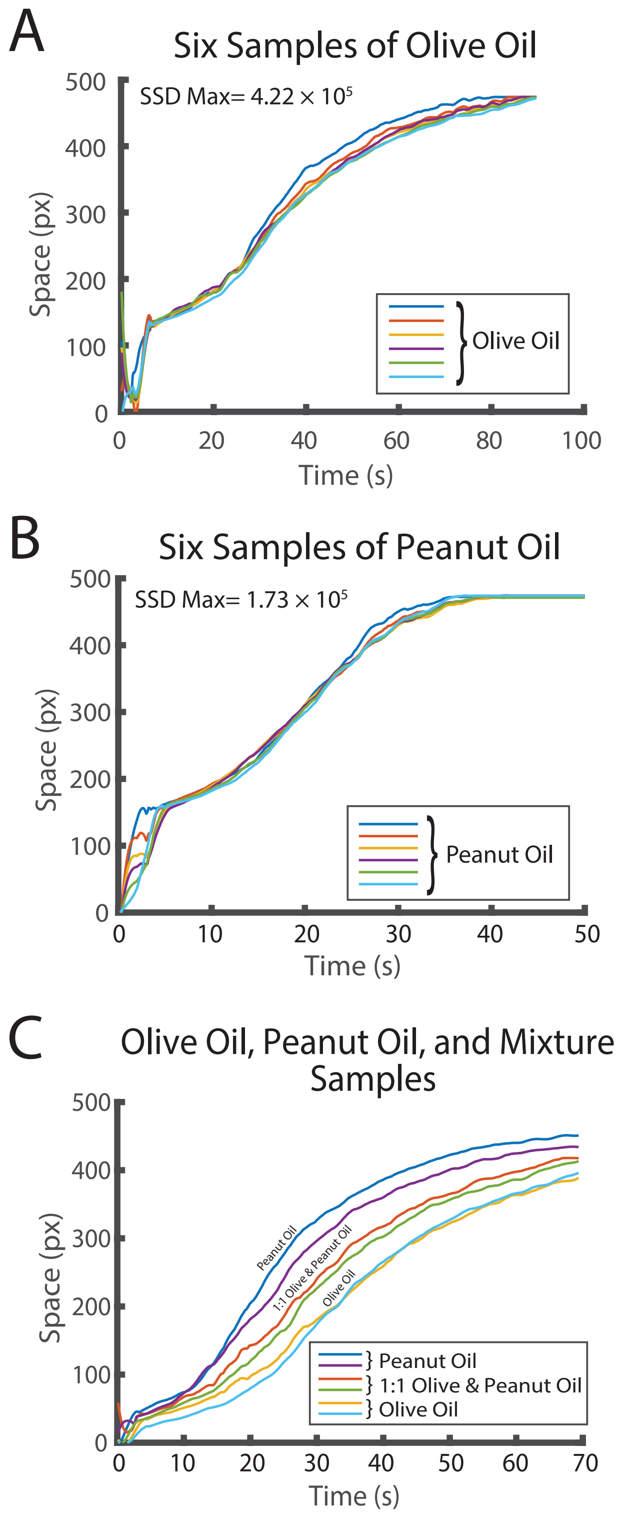

A
Six Samples of Olive Oil
SSD Max= 4.22 × 10^5
Space (px)
Time (s)
Olive Oil
B
Six Samples of Peanut Oil
SSD Max= 1.73 × 10^5
Space (px)
Time (s)
Peanut Oil
C
Olive Oil, Peanut Oil, and Mixture Samples
Peanut Oil
1:1 Olive & Peanut Oil
Olive Oil
Space (px)
Time (s)
} Peanut Oil
} 1:1 Olive & Peanut Oil
} Olive Oil

Figure 3.5

Figure 3.5: Identifying authentic and adulterated food oils using chronoprints. Each plot compares chronoprints from six food oil samples, converted to curves using the *feature tracing* method. **(A)** Chronoprint curves from six identical samples of olive oil are nearly identical and differ by a sum-of-squared-differences (SSD) that is 4.22×10^5 or less; this is less than the experimentally observed threshold of 1×10^6 and confirms that the oil samples are identical. **(B)** Chronoprint curves from six identical samples of peanut oil are similarly identical. **(C)** Chronoprint curves from two samples each of three different oils (olive oil, peanut oil, and a 1:1 mixture of olive and peanut oil) are similar within each oil type but significantly different between the different oil types. The maximum sum-of-squared-differences between two different oil types (5.77×10^6 difference between the olive oil and peanut oil samples) is greater than the threshold of 1×10^6 and confirms that these oils are different.

olive/peanut mixture samples. However, the different oil types had very different curves: the maximum sum-of-squared-differences was 5.77×10^6 for the olive oil and peanut oil samples. In this and other experiments, we found that chronoprints with *feature tracing* scores greater than about 1×10^6 indicated that the oils were different, and chronoprints with scores less than 1×10^6 indicated that the oils were the same. Additional chronoprints and analysis of experiments with olive oil, peanut oil, and a 1:1 mixture of the two oil samples are provided in *Supporting Information.*

3.3.2 Detecting counterfeit medicine

The United Nations estimates that around 10% of all medicines in low- and middle-income countries are counterfeit; consumers waste billions of dollars on these fake drugs every year. [39, 40] Simple and inexpensive tools for identifying adulterated drugs can protect consumers from these threats. For example, recent paper-based tests have been developed that can confirm the authenticity of samples of certain drugs.[51, 52, 53] However, there remains an unmet need for simple and low-cost techniques that can be applied to a wide range of different types of drugs.

To test the use of chronoprints for distinguishing authentic and adulterated medicine samples, we used our technique to analyze samples of over-the-counter cold medicine. These samples resulted in fairly complex chronoprints, so we used *image differences* and *image hashing* to compare these chronoprints. We first filled a microfluidic thermometer chip with six samples of cold medicine from the same bottle and obtained a chronoprint for each sample. Since these drug samples were identical, we expected that the resulting chronoprints would be very similar. Our experimental results (Figure 3.6A) confirm this expectation:

using *image hashing* with a global mean pixel value as the threshold, all six chronoprints' hashes differ by only 5 or fewer bits out of 64. This small difference in the chronoprints' image hashes confirms that the cold medicine samples are identical. We then filled the chip with six samples of cold medicine from six *different* medicine manufacturer's lots and obtained a chronoprint for each sample. Since these medicine samples are all the same brand, we expected that the resulting chronoprints would also be very similar. Our experimental results (Figure 3.6B) confirm this expectation: using *image hashing* with a local mean pixel value as the threshold, all six chronoprints' hashes differ by only 4 or fewer bits out of 64. This small difference in the chronoprints' image hashes confirms that these cold medicine samples are also identical, despite being manufactured at different times over a 4-month period. Additional chronoprint experiments, as well as the different analysis methods for the experiments shown in Figure 3.6, are provided in *Supporting Information.*

If a sample of medicine is adulterated by diluting it with water, the sample's chronoprint might change, and this could be the basis of a test to detect adulterated medicines. To test this idea, we filled a microfluidic thermometer chip with two samples each of 50%, 75% and 100% (v/v) dilutions of cold medicine in water. The resulting chronoprints were again analyzed using *image hashing* with a global mean pixel value as the threshold. The results (Figure 3.7A) show that within each dilution, the two samples' chronoprints are identical or nearly so: the two samples of 100% medicine have *identical* image hashes, as do the two samples of 75% medicine, and the two samples of 50% medicine differ by only 4 bits. However, between the different dilutions, the samples' chronoprints were very different: the 100% and 75% dilutions differed by 24 bits, the 75% and 50% dilutions differed by 23 and 27

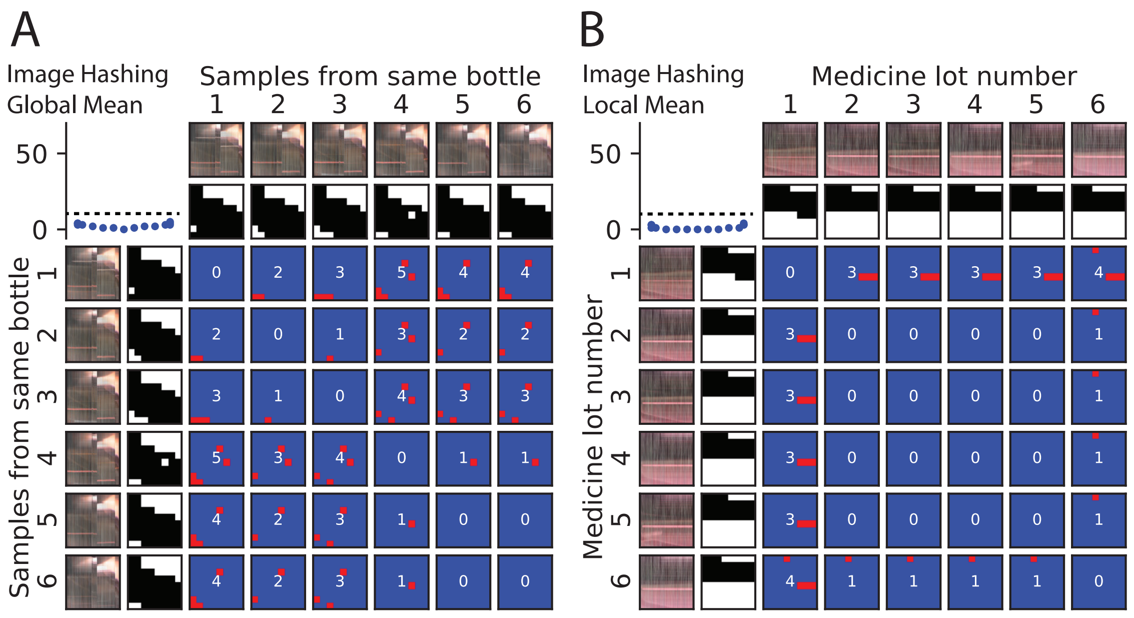
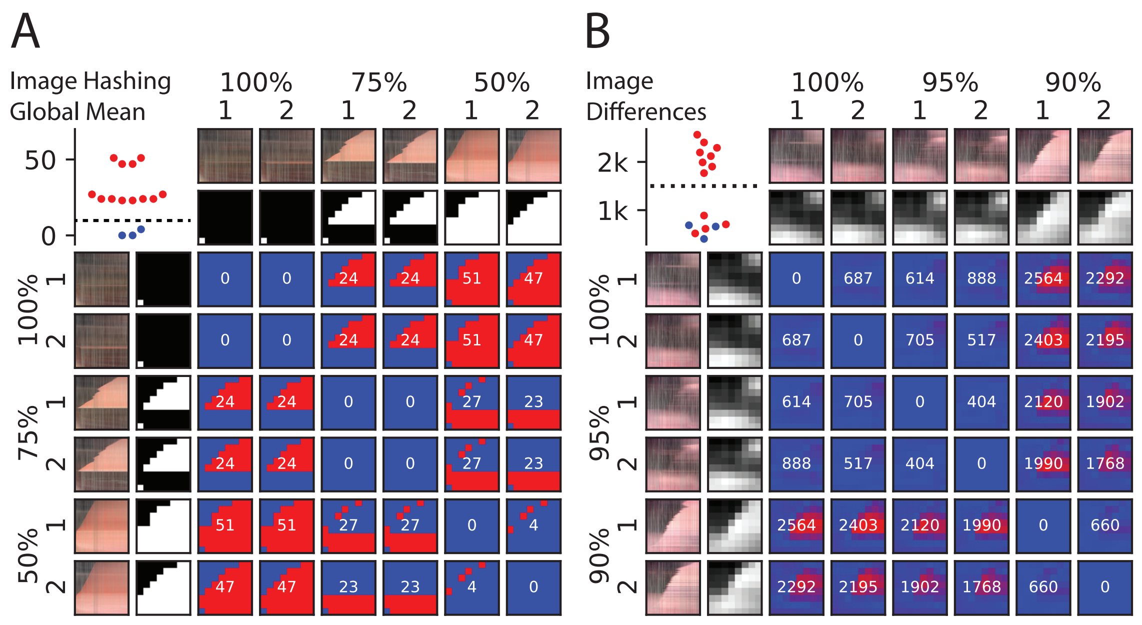


Figure 3.6: Detecting authentic liquid cold medicine using chronoprints. Each set of images shows six chronoprints along with all pairwise comparisons of the six samples in each experiment, plus a small summary plot of difference scores (blue points = known identical samples). **(A)** Chronoprints from six identical samples of cold medicine from the same bottle, compared using the *image hashing* method with the global mean pixel value used as the threshold. The resulting image hashes never differ by more than 5 bits; this is well below the 10-bit experimentally-observed threshold between identical and different samples (dotted line in summary plot) and confirms that all six medicine samples are identical. **(B)** Chronoprints from six samples of cold medicine from six different manufacturer's lot numbers, compared using the *image hashing* method with the local mean pixel value used as the threshold. The resulting image hashes never differ by more than 4 bits; this again confirms that the medicine samples are identical (despite having manufacture dates spanning a four-month period).

bits, and the 100% and 50% dilutions differed by 47 and 51 bits. In this and other studies, we found that chronoprint image hashes that differed by more than about 10 bits indicated that the medicines were different (and potentially adulterated), and hashes that differed by less that 10 bits indicated that the medicines were the same. Additional chronoprint experiments for the 50%, 75%, and 100% (v/v) dilutions of cough medicine in water, as well as the different analysis methods for the experiment shown in Figure 3.7A, are provided in *Supporting Information*.

To explore the sensitivity of the chronoprint technique, we repeated the cold medicine analysis in Figure 3.7A with a smaller difference between the different dilutions: 90%, 95% and 100% (v/v). For this experiment, we found that the *image differences* comparison method provided the clearest results (Figure 3.7B). As expected, within each dilution, the two samples' chronoprints are very similar: the two samples of 100% medicine have *image difference* scores of only 687, the two samples of 95% medicine have scores of 404, and the two samples of 90% medicine have scores of 660. These scores are all less than the $\sim$1500 threshold that we observed separates *image differences* scores of identical (< 1500) and different (> 1500) samples. And also as expected, two different dilutions' chronoprints were very different: the 90% medicine had *image differences* scores from 2195 to 2564 when compared to the 100% medicine and 1768 to 2120 when compared to the 95% medicine. However, the 95% and 100% medicines had indistinguishable chronoprints—their *image differences* scores ranged from 517 to 888, which are below the ~ 1500 threshold and therefore erroneously identified as identical. In summary, the results in Figure 3.7 show that our chronoprint method can identify samples of this cold medicine that have been diluted by as



	100% 1	100% 2	75% 1	75% 2	50% 1	50% 2
100% 1	0	0	24	24	51	47
100% 2	0	0	24	24	51	47
75% 1	24	24	0	0	27	23
75% 2	24	24	0	0	27	23
50% 1	51	51	27	27	0	4
50% 2	47	47	23	23	4	0

	100% 1	100% 2	95% 1	95% 2	90% 1	90% 2
100% 1	0	687	614	888	2564	2292
100% 2	687	0	705	517	2403	2195
95% 1	614	705	0	404	2120	1902
95% 2	888	517	404	0	1990	1768
90% 1	2564	2403	2120	1990	0	660
90% 2	2292	2195	1902	1768	660	0

Figure 3.7: Detecting adulterated liquid cold medicine using chronoprints. **(A)** Chronoprints from two samples each of three different dilutions of cold medicine in water (50%, 75%, and 100%) again compared using the *image hashing* method. The resulting image hashes successfully confirm that the two samples of each dilution are identical (difference scores of 0, 0, and 4 bits; all < 10) and all samples of different dilutions are different (difference scores from 23 to 51 bits; all > 10). **(B)** Chronoprints from two samples each of three additional dilutions of cold medicine (90%, 95%, and 100%) compared using the *image differences* method. The resulting difference images successfully distinguished the 90% samples from the 95% and 100%, with difference scores from 1768 to 2564 (all > 1500, the experimentally-observed threshold between identical and different samples marked with the dotted line). However, the images failed to distinguish the 100% and 95% samples, with difference scores from 517 to 888 (all < 1500 and therefore erroneously identified as identical; red points below the dotted line). Thus, chronoprints are capable of identifying samples of this cold medicine that have been diluted by as little as 10%.

little as 10%. Additional chronoprint experiments for the 90%, 95%, and 100% (v/v) dilutions of cough medicine in water, as well as the different analysis methods for the experiment shown in Figure 3.7B, are provided in *Supporting Information.*

3.3.3 Identifying toxic pharmaceutical ingredients

In 1937, a chemist at the S.E. Massengill Company in Bristol, Tennessee, unwittingly substituted a toxic substance, diethylene glycol, for nontoxic glycerol in a liquid formulation of the early antibiotic sulfanilamide. The resulting medicine, called "Elixir Sulfanilamide," fatally poisoned over 100 persons [54, 55], and the toxicity of diethylene glycol became common knowledge among pharmaceutical companies. But remarkably, poisonings due to diethylene glycol in medicines remain tragically common today, with a mass poisoning occurring somewhere in the world on average every two years since 1985.[56] Many of these poisonings occur in resource-limited settings where pharmaceutical companies may not have the resources needed to confirm the identity (and safety) of their manufacturing stocks. The problem of distinguishing diethylene glycol from glycerol is compounded by the fact that they both have very similar properties: they are both transparent, viscous, sweet-tasting liquids, with similar densities, freezing/melting points, and other properties. Consequently, our initial attempts to distinguish diethylene glycol and glycerol by their melting/freezing points alone (using our microfluidic thermometer[41]) were unsuccessful.

To determine whether our chronoprint technique could distinguish toxic diethylene glycol from nontoxic glycerol, we filled a microfluidic thermometer chip with three samples each of both substances, partially immersed the chip in liquid nitrogen, and obtained chronoprints from the video recording of the chip. The chronoprints were then analyzed using the *image hashing* technique with the global mean pixel value used as the threshold. The results, shown in Figure 3.8, confirm that all the glycerol chronoprint hashes are very similar (never differing by more than 8 bits), as are all the diethylene glycol chronoprint hashes (never differing by more than 10 bits). However, the glycerol chronoprint hashes are significantly different from the diethylene glycol chronoprint hashes (differing by at least 46 bits). These results confirm that our chronoprint method can easily distinguish between toxic diethylene glycol and nontoxic glycerol.

3.4 Conclusion

In this work, we introduced *chronoprints*, an image-based method for identifying or distinguishing substances. Chronoprints are simple and inexpensive to obtain; we used a plastic microfluidic chip, a USB camera, and some liquid nitrogen to obtain ours, but other approaches could be used. In principle, any sample that changes in appearance in response to a perturbation could be analyzed using chronoprints. While we used chronoprints to analyze only liquid samples in this work, the technique is not limited to liquids. For example, solid samples such as pills could be dissolved in constant volumes of water, loaded into the thermometer chip, and analyzed for their authenticity using chronoprints. Even gas mixtures could be analyzed using chronoprints if their components formed condensation or other visible markings on the channel walls of the microfluidic thermometer.

In this proof-of-concept, we used a rapidly-changing temperature gradient as the perturbation; this gradient induces phase changes, separations, and other changes within the samples that make for useful chronoprints. However, some samples may be indistinguishable

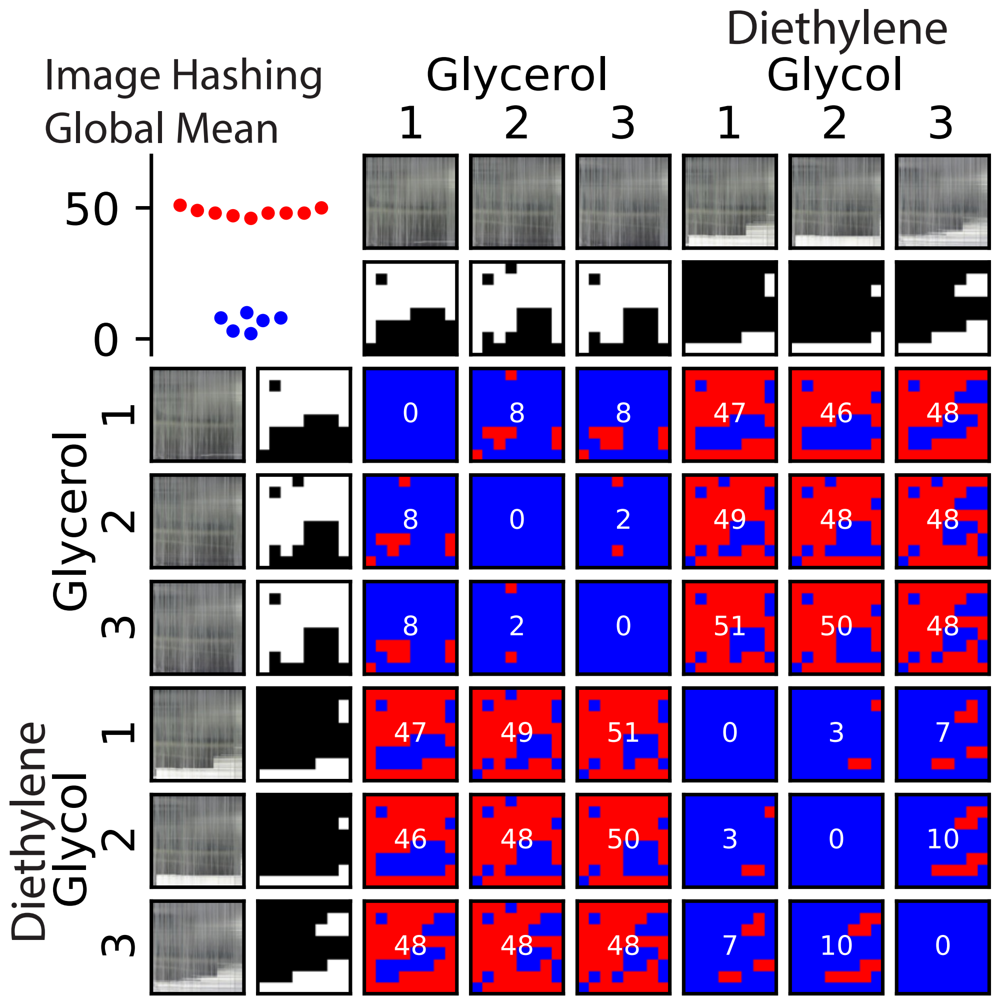


Figure 3.8: Distinguishing toxic and nontoxic pharmaceutical ingredients using chronoprints. Chronoprints of three samples of toxic diethylene glycol and three samples of nontoxic glycerol were analyzed using the *image hashing* technique with the global mean pixel value as the threshold. The three glycerol chronoprint hashes were nearly identical (differing by 8 or fewer bits), as were the three diethylene glycol hashes (differing by only 3 to 10 bits). However, all of the glycerol chronoprint hashes were significantly different from all of the diethylene glycol hashes (differing by 46 to 51 bits out of a maximum of 64). These results confirm that these substances can be easily distinguished by their chronoprints.

using chronoprints based on dynamic temperature gradients. For example, we could not distinguish the 95% and 100% dilutions of cold medicine using dynamic temperature gradients in Figure 3.7B. For samples like these, other kinds of perturbations could be used to generate unique chronoprints. For example, if the samples differ in their boiling points, then a high-temperature dynamic temperature gradient might generate different chronoprints for the different samples. For heterogenous samples containing suspended solids, a changing gravitational acceleration provided by *e.g.* a centrifuge may yield unique chronoprints. In general, any perturbation that affects the appearance of different samples in different ways could be the basis for a chronoprint.

Our method does not require that the temperature gradient is reproducible from run to run. Rather, we use multiple parallel microfluidic channels to ensure that each sample experiences the same temperature gradient within a run (thereby enabling comparisons between the samples within that run). This greatly simplifies our technique by eliminating the need for hardware like temperature sensors or controllers. However, if it *were* possible to compare chronoprints from different runs, it would enable us to construct a database of chronoprints for various substances; this would enable our technique to identify a substance without having a known sample of the substance for comparison. One simple way to accomplish this would be to load one or more reference materials into the thermometer chip. For example, materials like temperature-sensitive liquid crystals (whose appearances change in known and reproducible ways as their temperature changes) could serve as internal standards. By scaling a liquid crystal's chronoprint to make it match a reference chronoprint for the liquid crystal, then applying the same scaling to the chronoprints of other samples in the same run, one might obtain "standardized chronoprints" that could be saved to a database and compared between different runs.

One practical limitation to using our chronoprint technique in resource-limited settings is the availability of liquid nitrogen for creating the dynamic temperature gradient. While liquid nitrogen is relatively inexpensive, it requires production and transport infrastructure that may not exist in some locations. For those without access to liquid nitrogen, dry ice may be available from *e.g.* carbonated beverage production facilities that are found in many towns. We found that dynamic temperature gradients suitable for obtaining chronoprints can be obtained using dry ice in acetone, although one must first confirm that one's microfluidic thermometer chip material is compatible with the solvent. We also previously used inexpensive Peltier (thermoelectric) coolers to generate temperature gradients,[41] and these coolers could also be used to create chronoprints. Finally, a compressor and freezer coil from an ordinary refrigerator could likely be repurposed to provide the temperature gradient necessary for obtaining chronoprints.

Another practical limitation to our technique concerns the training and other resources required to perform it, but we note that many of the experiments in this work were performed by undergraduate researchers (J. Robles-Najar and E. Duong) who required only minimal training to become proficient at the chronoprint technique. The microfluidic thermometer chip we used required a hobbyist-grade milling machine for fabrication, but we provide the computer design file for this chip in *Supporting Information*, and we have also demonstrated that consumer-grade 3D printers can be used to fabricate similar chips.[41] Additionally, the MATLAB- and Python-based analysis code we also provide as *Supporting*

Information should simplify the process of generating chronoprints from video recordings of thermometer chips.

Finally, since chronoprints are fundamentally just bitmap images on a computer, we can leverage the enormous variety of image analysis and comparison techniques that have been developed by computer scientists. The image similarity measurements that we used in this proof-of-concept demonstration generally worked well, but they could not algorithmically distinguish the 95% and 100% dilutions of cold medicine in Figure 3.7B. However, by simply looking at the raw chronoprints in Figure 3.7B, one can nonetheless see slight differences between the two concentrations' chronoprints. Image comparison algorithms that can recognize these differences will enable our chronoprint technique to correctly identify and discriminate an even wider variety of samples.

3.5 Acknowledgements

This work was supported by the National Science Foundation Division of Biological Infrastructure under award 1353974; the National Science Foundation Division of Computer and Communication Foundations under awards 1351115 and 1536026; the National Science Foundation Division of Industrial Innovation and Partnership under award 1640757; and the National Science Foundation Division of Civil, Mechanical, and Manufacturing Innovation under award 1740052. The authors also gratefully acknowledge the anonymous reviewer who suggested alternative methods for generating dynamic temperature gradients in resource-limited settings.

The text of this dissertation, in part or in full, is a reprint of the material as it appears in "Chronoprints: Identifying Samples by Visualizing How They Change over Space and Time." *ACS Central Science*, 5(4):589598, April 2019. The co-author William H. Grover listed in that publication directed and supervised the research which forms the basis for this dissertation. The co-author Philip Brisk supported the work with computer science resources and provided research equipment for the fabrication of the microfluidic device described in that publication. The co-authors Jessica Robles-Najar and Eric Duong assisted with the experiments described in that publication.

3.6 Supporting Information for "Chronoprints: Identifying substances by visualizing how they change over time"

3.6.1 Thermometer_chip.dxf: Design of the microfluidic thermometer chip in DXF format, used when milling the chips used in this work on a CNC mill.

The design of the thermometer chip in standard .DXF format is shown below and can be downloaded at https://pubs.acs.org/doi/10.1021/acscentsci.8b00860.

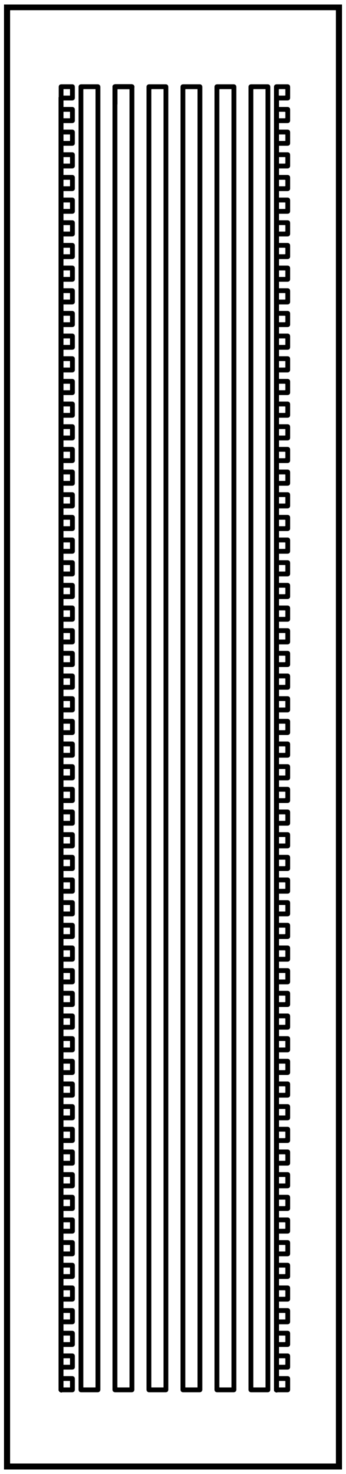


Figure 3.9: A drawing of the thermometer chip design used to fabricate chips in this work.

3.6.2 Supporting information text and supplementary Figures 3.10–3.22: data from replicate chronoprint experiments, and results from different analysis methods for the experiments shown in the main text.

3.6.2.1 Optimal lighting

Our initial experiments with food oils revealed the importance of proper lighting in obtaining quality chronoprints. Overly-lit experiments caused saturated pixels in the resulting videos and poor contrast in the resulting chronoprints, which in turn affected how well our *feature tracing* MATLAB script could convert the chronoprint into curves; see supplementary Figure 3.10 for an example. In later experiments we reduced the lighting on the chip to obtain results such as those shown in the main text.

3.6.2.2 Optimal experiment duration

We also explored the optimal duration for oil chronoprint experiments. Supplementary Figure 3.11 shows the results of using the *feature tracing* analysis of a chronoprint obtained from a 25-second-long video of the oil samples reacting to a dynamic temperature gradient. While the resulting curves are roughly grouped by their oil type, the separation between the oil types is not very pronounced. We attributed this to insufficient experiment time and ran subsequent experiments for longer durations, which resulted in greater separation between the different oil curves.

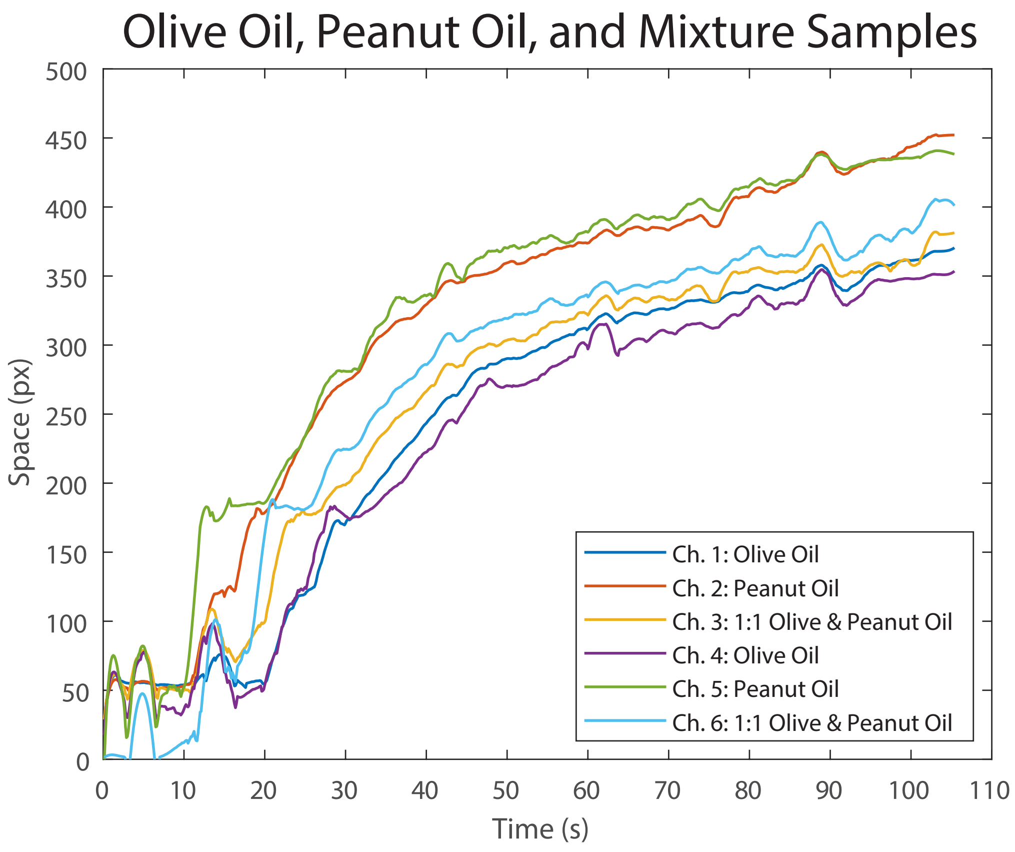


Figure 3.10: Curves resulting from using the *feature tracing* analysis of chronoprints from from 100% olive oil (channels 1 and 4), 100% peanut oil (channels 2 and 5), and a 1:1 mixture of olive and peanut oils (channels 3 and 6). Excessive lighting during video recording saturated some pixels and adversely affected the quality of these results.

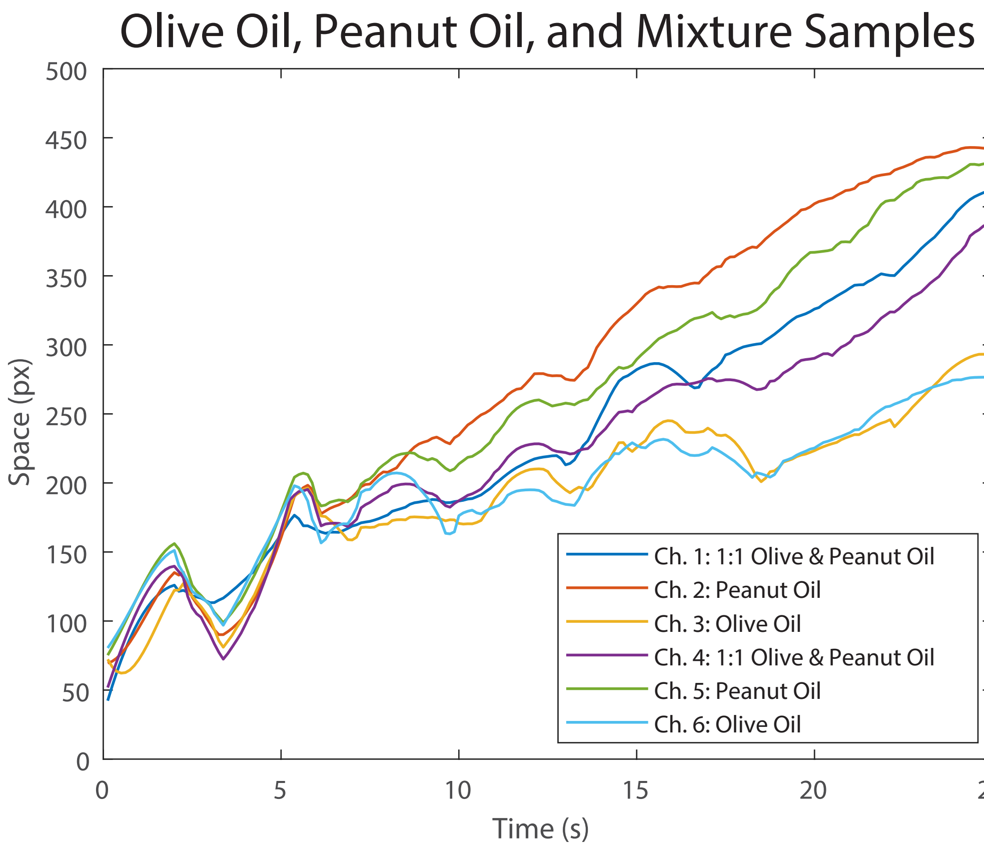


Figure 3.11: Curves resulting from using the *feature tracing* analysis of chronoprints from from a 1:1 mixture of olive and peanut oils (channels 1 and 4), 100% peanut oil (channels 2 and 5), and 100% olive oil (channels 3 and 6). The separation between the oil types is not very pronounced. We attribute this to insufficient experiment duration time. The curves from samples in channels 1 and 2 are slightly higher than the curve from the same samples in channels 4 and 5. We attribute this error to poor centering of the channels on this microfluidic thermometer chip, which exposed channels 1 and 2 to a faster-dropping temperature gradient than the other channels.

3.6.2.3 Thermometer chip edge effects

Occasionally, small variations in our chip milling process can yield thermometer chips with channels that are not perfectly centered on the chip. When this happens, the samples in the channels may experience slightly different changing temperature gradients during an experiment; the sample in the channel closer to one edge of the chip is more exposed to the liquid nitrogen bath and cools faster than the other channels. This results in a shifted curve for the sample, as shown for the oil samples in channels 1 and 2 of Figure 3.11 and the sample in channel 1 of Figure 3.12. Taking care to fabricate symmetric thermometer chips minimizes this source of error.

3.6.2.4 Replicate of experiment in main text Figure 3.5C

We performed several replicates of the oil experiment shown in Figure 3.5C of the main text and included one here (Figure 3.12). In both experiments, chronoprints successfully distinguish these different oil types.

3.6.2.5 Comparisons of chronoprint image similarity algorithms

As explained in the main text, we used five different image similarity algorithms to quantify the similarity of pairs of chronoprints. Four of these algorithms used *image hashing*, which converts each chronoprint to a reduced-resolution, 64-bit binary "hash" of the original chronoprint. These algorithms differed in which pixel value we used as a threshold for converting the chronoprints into binary representations:

Local Mean: the average pixel value in each chronoprint was used as the threshold.

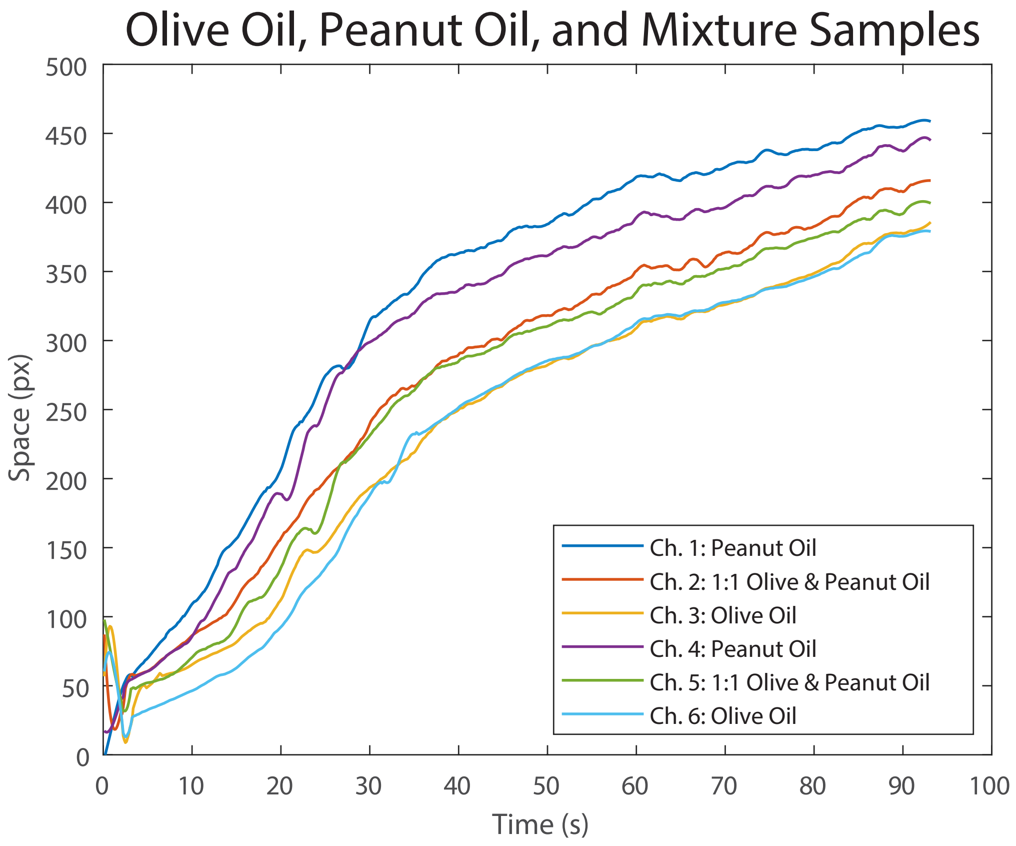


Figure 3.12: Curves resulting from using the *feature tracing* analysis of chronoprints from from 100% peanut oil (channels 1 and 4), a 1:1 mixture of olive and peanut oils (channels 2 and 5), and 100% olive oil (channels 3 and 6). This is a replicate of the experiment shown in Figure 3.5C of the main text. Among the samples of the same oil types, the chronoprint traces had relatively small differences: the maximum sum-of-squared-differences of 8.47 $\times 10^4$, 3.61 $\times 10^5$, and 1.35 $\times 10^5$ between the two 100% olive oil samples, the two 100% peanut oil samples, and the two 1:1 olive and peanut oil samples, respectively. However, between the different oil types, the chronoprint traces had greater differences: the maximum sum-of-squared-differences were 7.34 $\times 10^6$ between the 100% olive oil and 100% peanut oil samples.

Local Median: the median pixel value in each chronoprint was used as the threshold.

Global Mean: the average pixel value across all six chronoprints in an experiment was used as the threshold.

Global Median: the median pixel value across all six chronoprints in an experiment was used as the threshold.

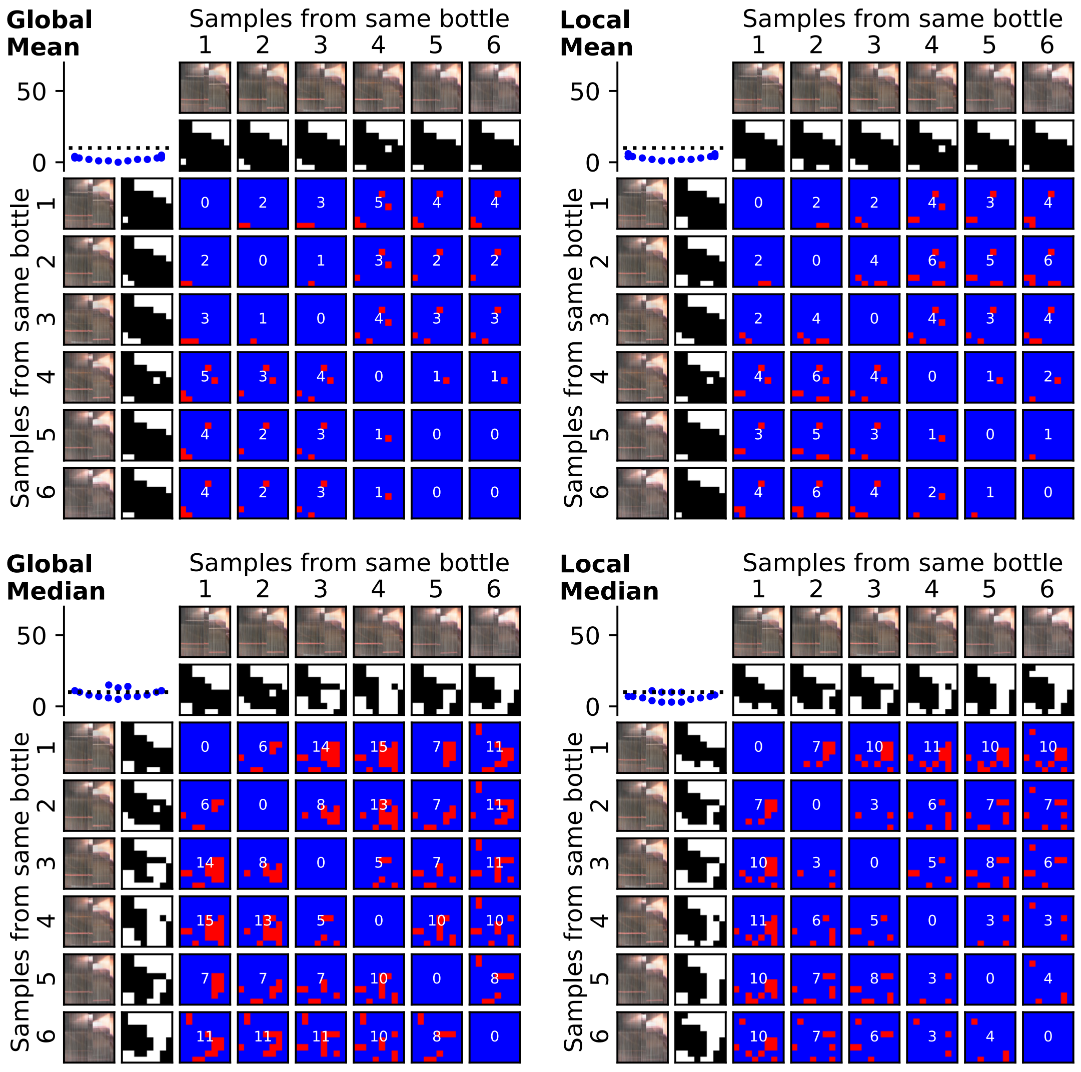
The fifth algorithm, *image differences*, converted each chronoprint to a reduced-resolution grayscale version, then calculated the sum of the pixel-by-pixel differences between two chronoprints to assess their similarity.

In the main text, we presented results from only a single image similarity algorithm for each cold medicine experiment. Here, we present results from all five algorithms for each cold medicine experiment in the main text, as well as analysis of replicates of some of the experiments in the main text. Figure 3.13 compares all five chronoprint similarity algorithms for the experiment in Figure 3.6A of the main text (comparing six identical cold medicine samples taken from the same bottle). Figure 3.14 compares all five algorithms for the experiment in Figure 3.6B of the main paper (comparing six samples of cold medicine from six different manufacturer's lots). Figure 3.15 provides a replicate of the experiment in Figure 3.6A of the main text (comparing six identical cold medicine samples taken from the same bottle); in this case the thawing of the samples was also recorded and included in the chronoprints. Figure 3.16 provides a replicate of the experiment in Figure 3.6B of the main paper (comparing six samples of cold medicine from six different manufacturer's lots). Figure 3.17 compares all five different chronoprint similarity algorithms for the experiment in Figure 3.7A of the main text (comparing 50%, 75%, and 100% v/v dilutions of cough medicine in water). Figures 3.18 and 3.19 provide replicates of the experiment in Figure 3.7A of the main text (comparing 50%, 75%, and 100% v/v dilutions of cough medicine in water). Figure 3.20 compares all five chronoprint similarity algorithms for the experiment in Figure 3.7B of the main paper (90%, 95%, and 100% v/v dilutions of cold medicine in water). Finally, Figures 3.21 and 3.22 provide replicates of the experiment in Figure 3.7B of the main paper (90%, 95%, and 100% v/v dilutions of cold medicine in water).

3.6.3 `Chronoprintgen.m`: MATLAB software for converting videos into chronoprints, used to generate all of the chronoprints shown in this work.

The source code for the software used to generate all of the chronoprints in this work is shown below and can be downloaded at https://pubs.acs.org/doi/10.1021/acscentsci.8b00860.

```
%ChronoprintGen.m
%
%This code accompanies the paper "Chronoprints: Identifying
    samples by
```



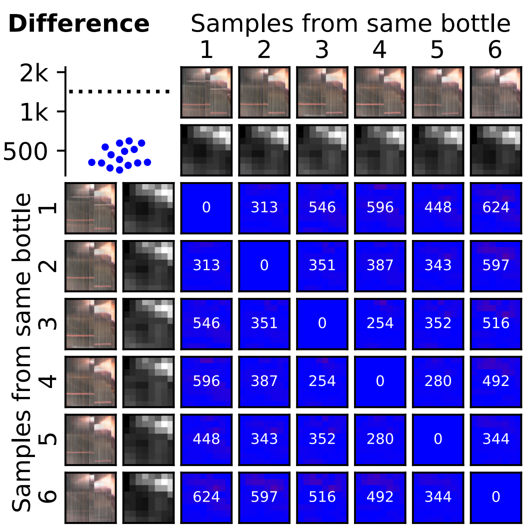


Figure 3.13: Comparisons of different chronoprint similarity algorithms for six cold medicine samples from one bottle (the experiment shown in Figure 3.6A of the main text).

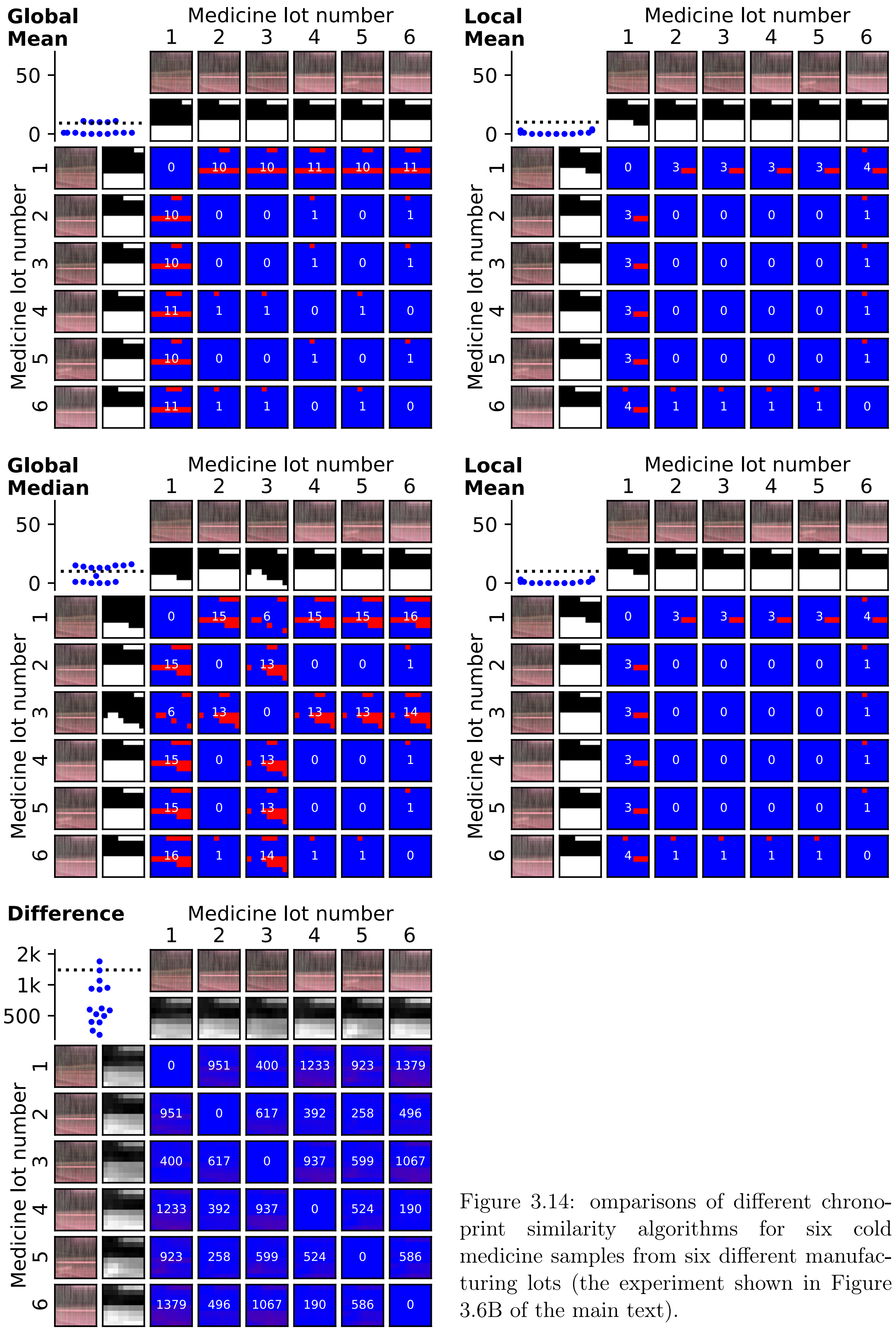


Figure 3.14: omparisons of different chronoprint similarity algorithms for six cold medicine samples from six different manufacturing lots (the experiment shown in Figure 3.6B of the main text).

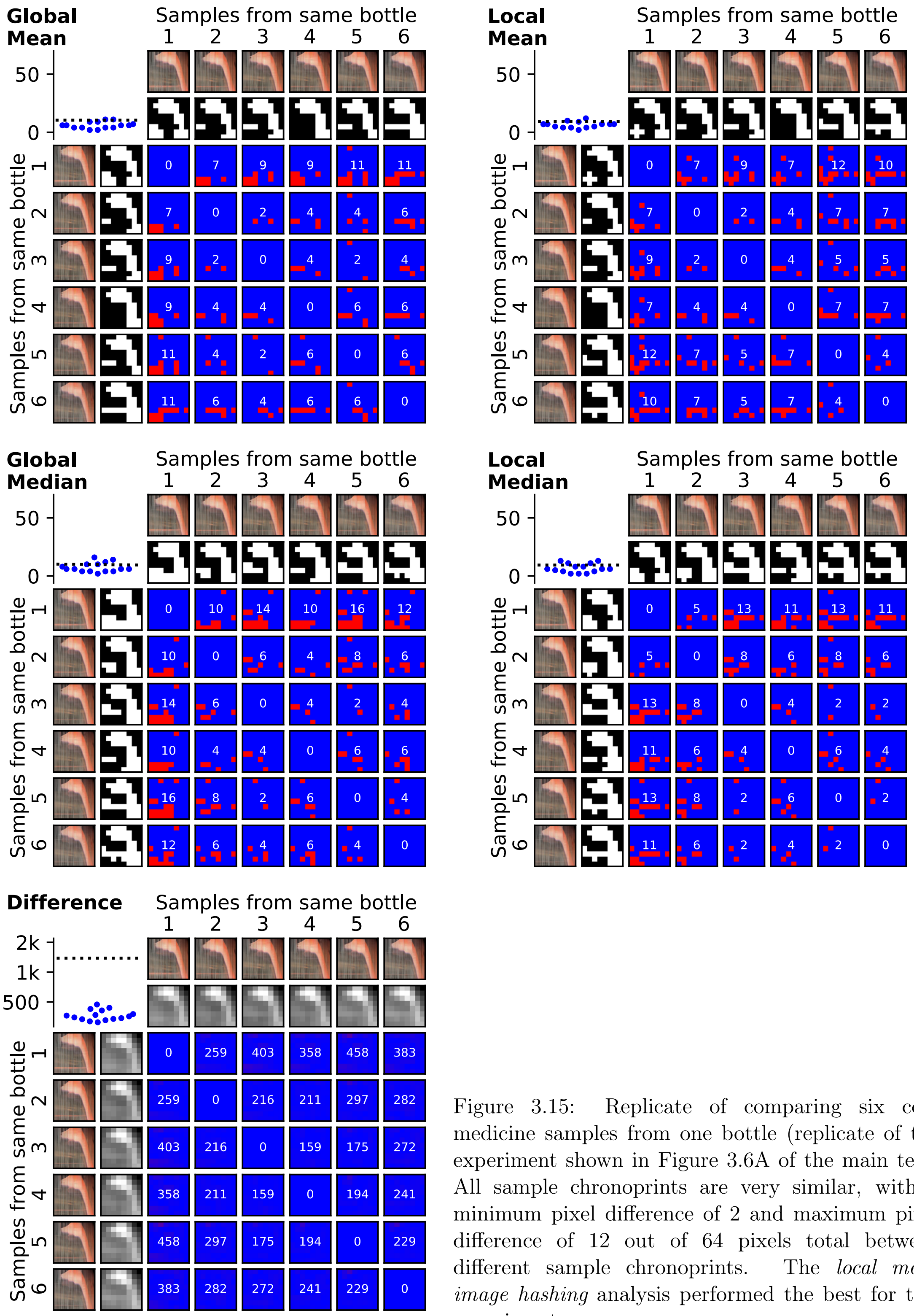


Figure 3.15: Replicate of comparing six cold medicine samples from one bottle (replicate of the experiment shown in Figure 3.6A of the main text. All sample chronoprints are very similar, with a minimum pixel difference of 2 and maximum pixel difference of 12 out of 64 pixels total between different sample chronoprints. The *local mean image hashing* analysis performed the best for this experiment.

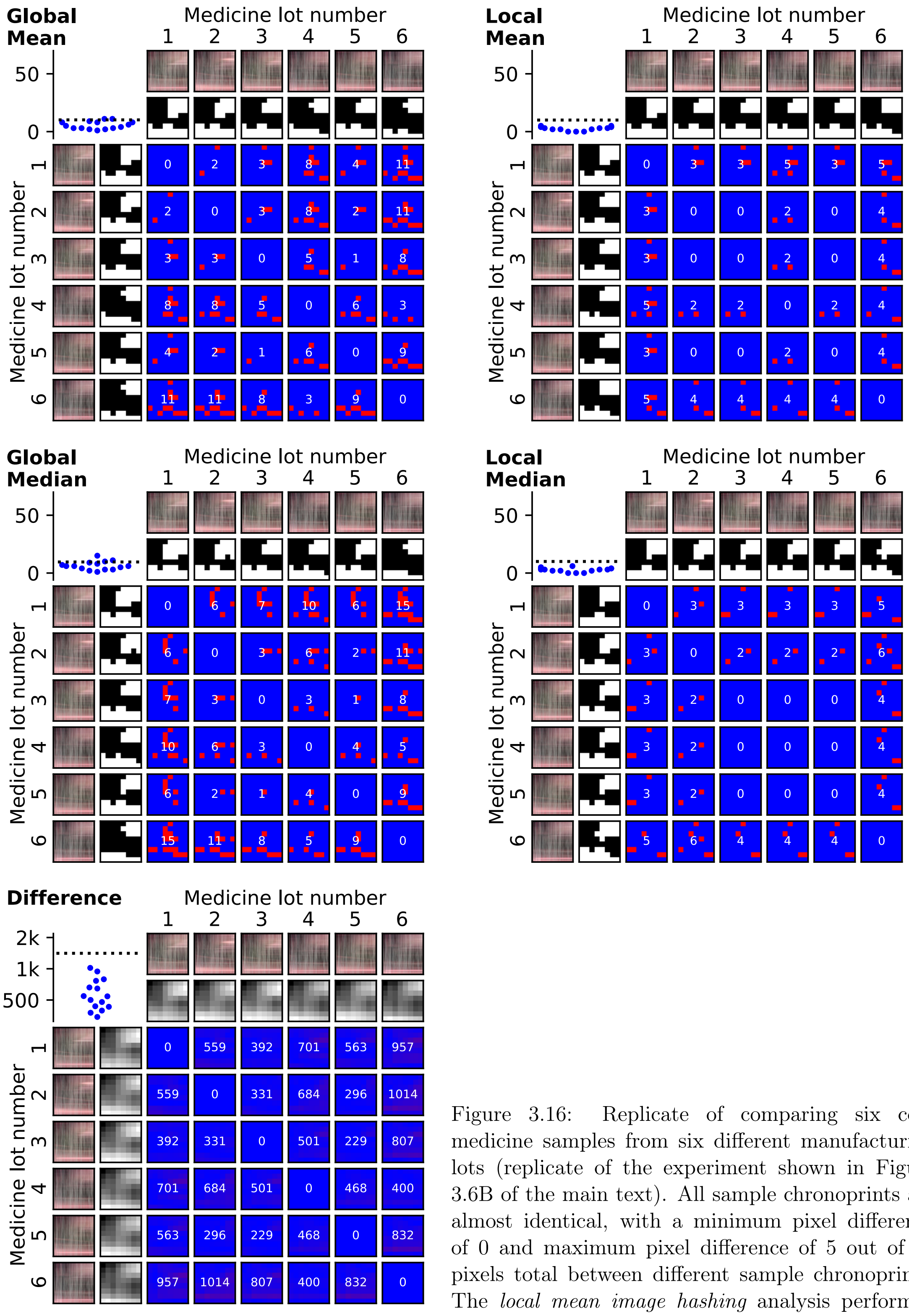


Figure 3.16: Replicate of comparing six cold medicine samples from six different manufacturing lots (replicate of the experiment shown in Figure 3.6B of the main text). All sample chronoprints are almost identical, with a minimum pixel difference of 0 and maximum pixel difference of 5 out of 64 pixels total between different sample chronoprints. The *local mean image hashing* analysis performed the best for this experiment.

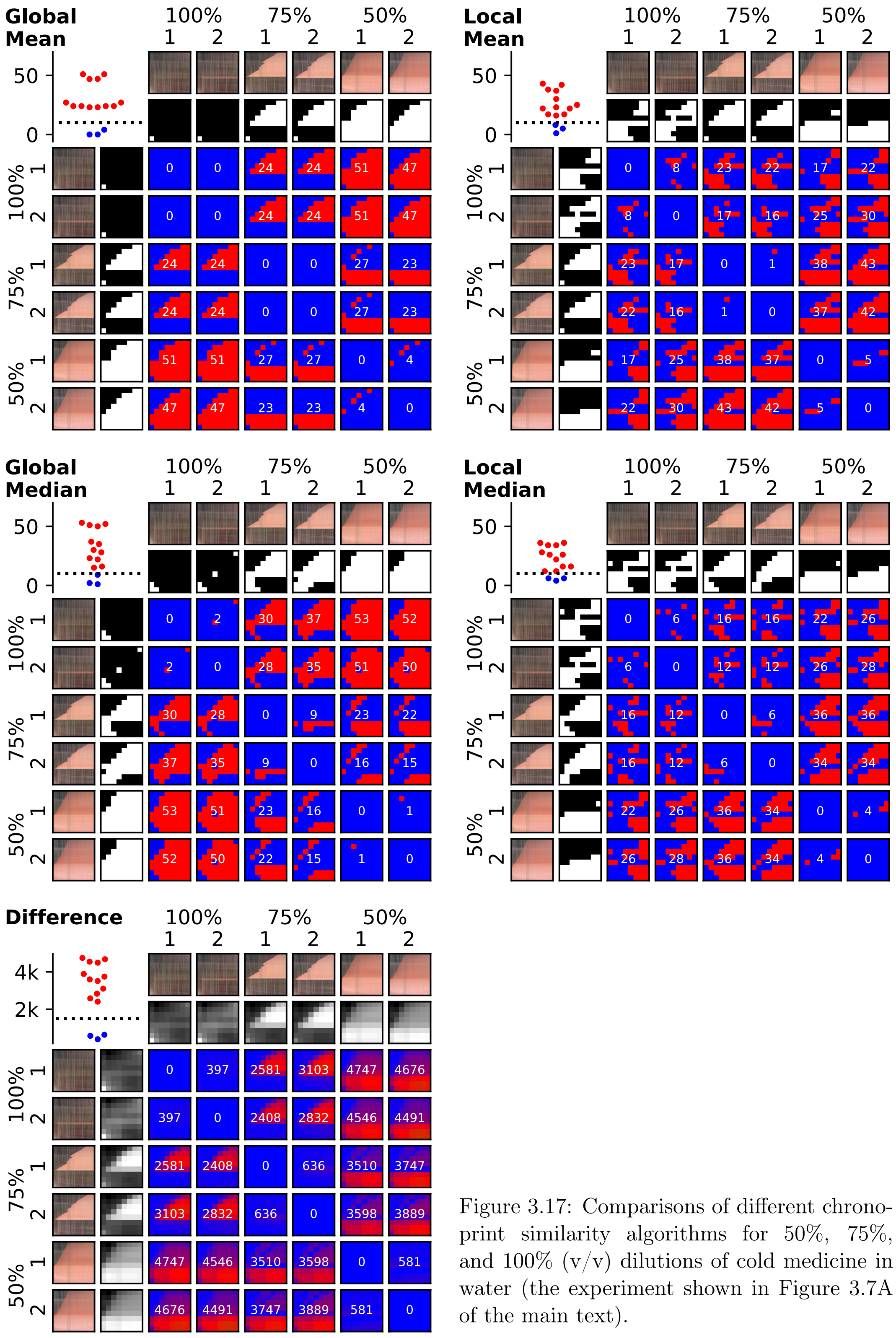
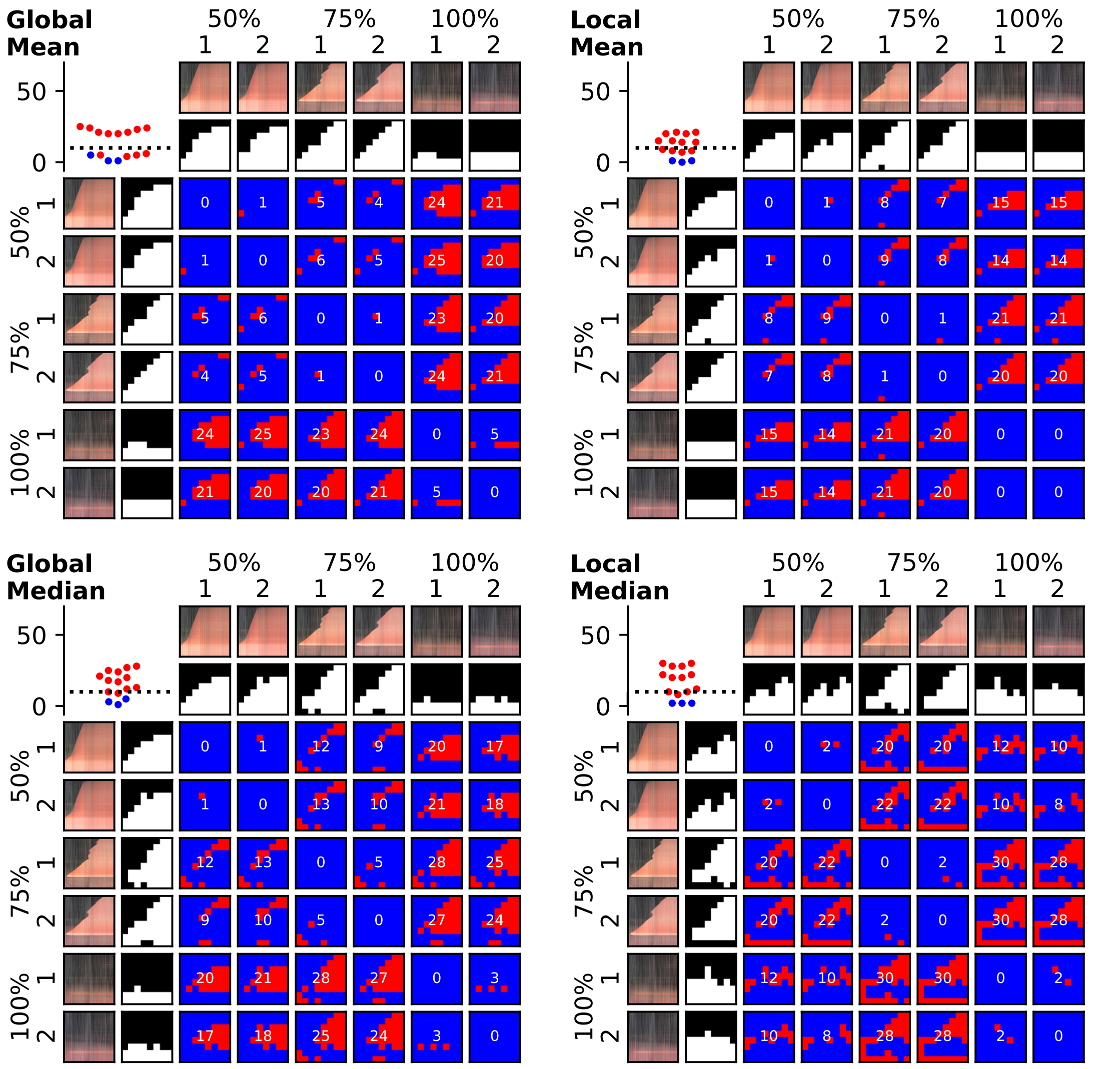


Figure 3.17: Comparisons of different chronoprint similarity algorithms for 50%, 75%, and 100% (v/v) dilutions of cold medicine in water (the experiment shown in Figure 3.7A of the main text).



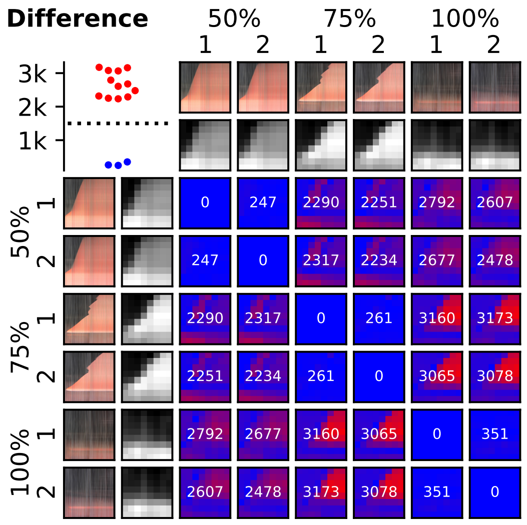
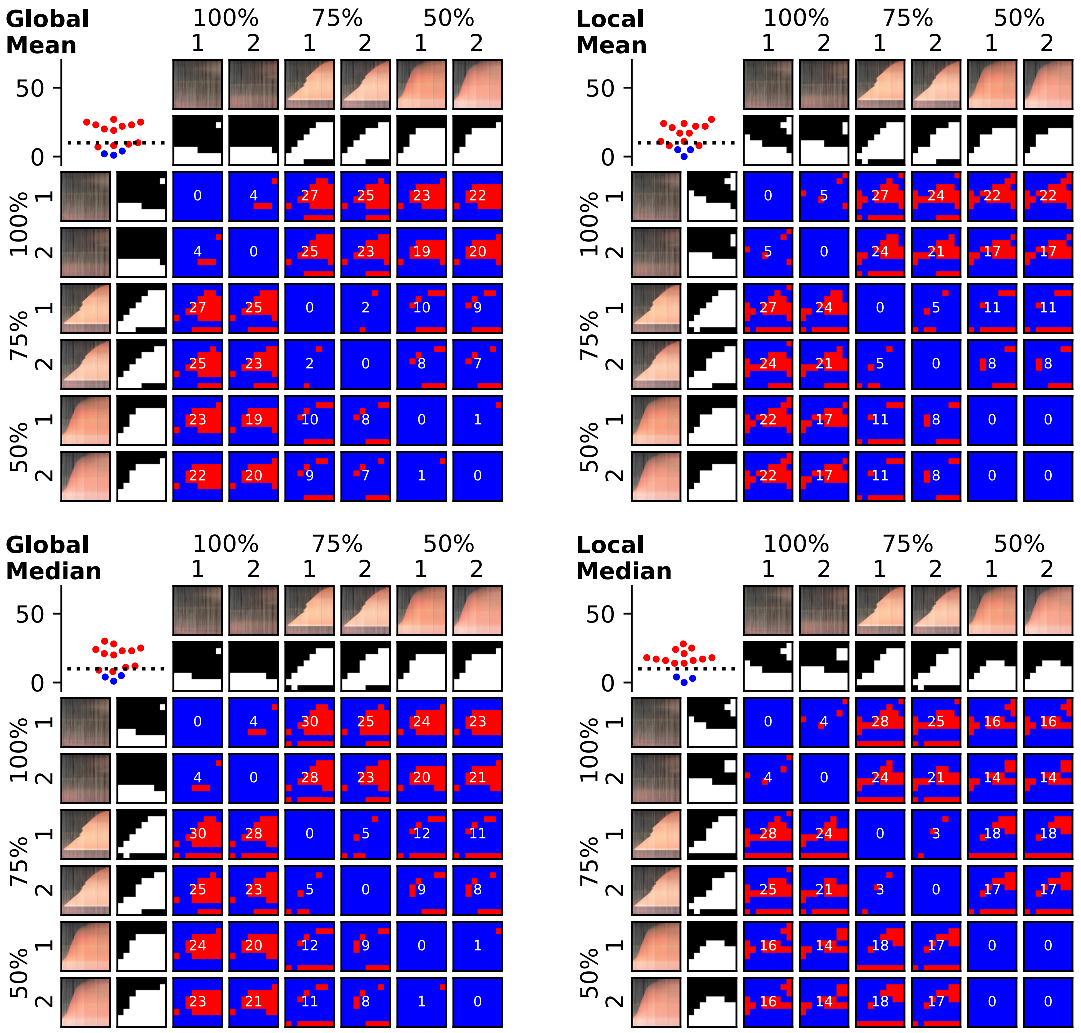


Figure 3.18: Replicate of the analysis of 50%, 75%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7A of the main text). All three dilutions types are distinguishable. The *global median image hashing* analysis performed best for this experiment, with a maximum pixel difference of 28 of 64 pixels between the 75% and 100% sample chronoprints.



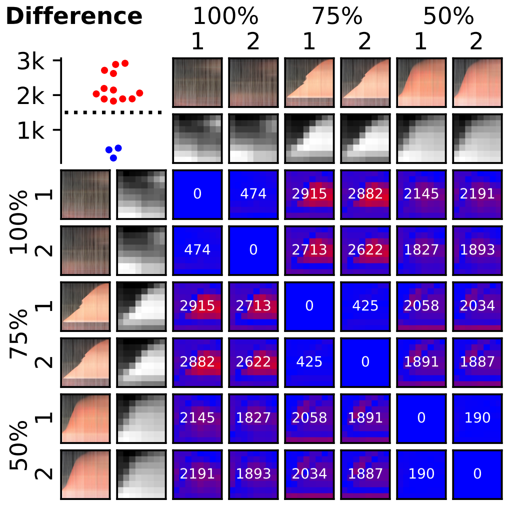


Figure 3.19: Replicate of the analysis of 50%, 75%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7A of the main text). All three sample types are distinguishable. The *local median image hashing* analysis performed the best for this experiment, with a maximum pixel difference of 28 out of 64 pixels between the 75% and 100% sample chronoprints.

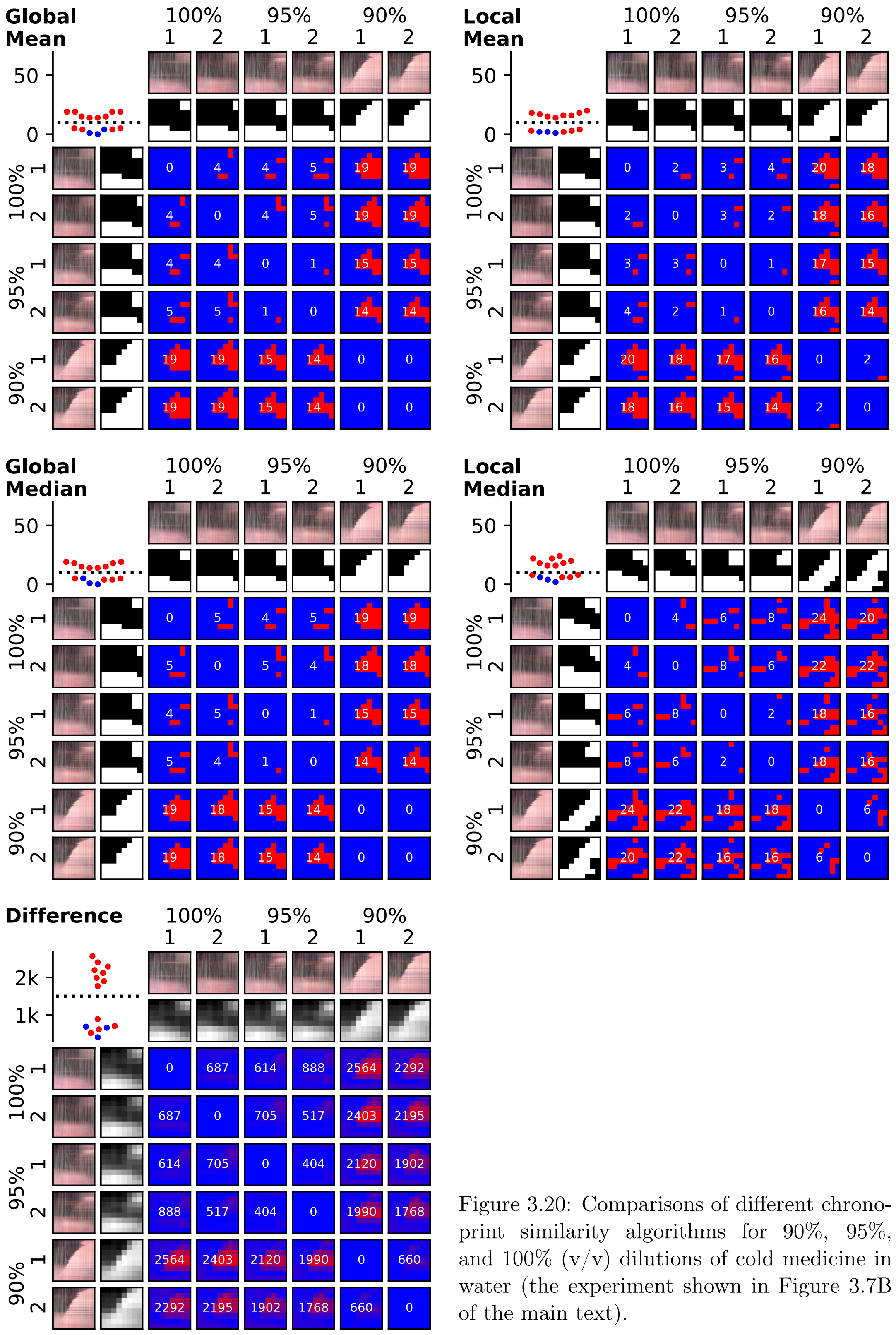
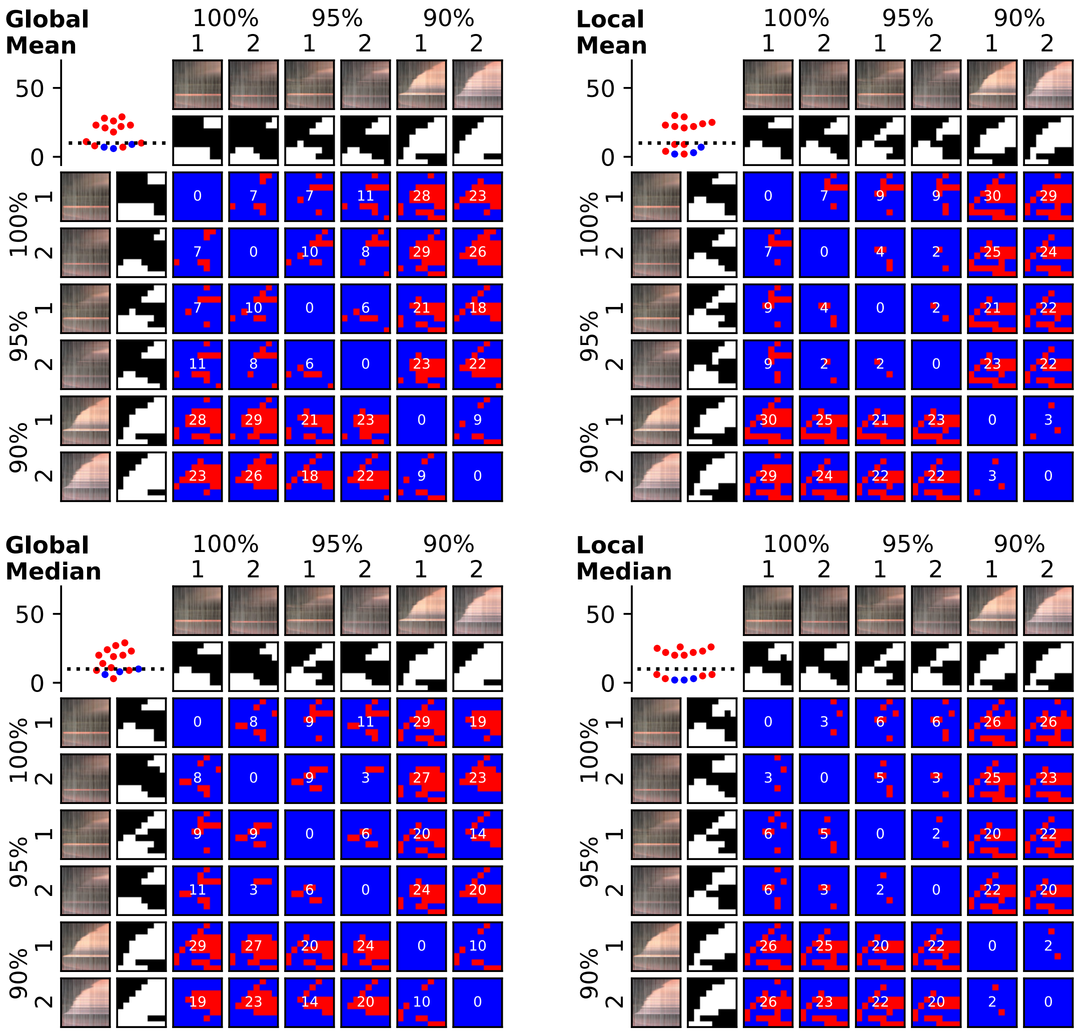


Figure 3.20: Comparisons of different chronoprint similarity algorithms for 90%, 95%, and 100% (v/v) dilutions of cold medicine in water (the experiment shown in Figure 3.7B of the main text).



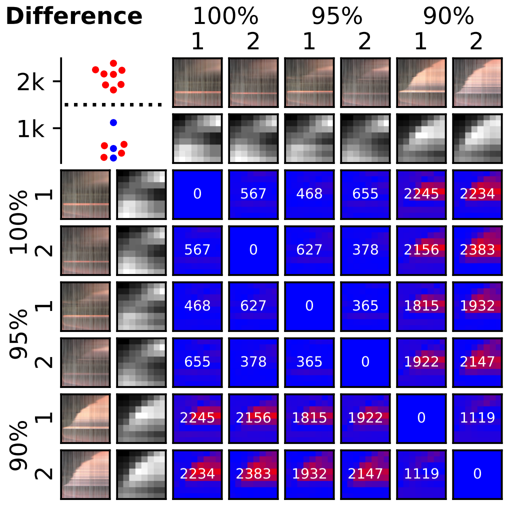


Figure 3.21: Replicate of the analysis of 90%, 95%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7B of the main text). The 90% cold medicine samples are distinguishable from the others, but the 95% and 100% samples were not distinguishable. The *local median image hashing* analysis performed the best for this experiment, with a maximum pixel difference of 26 out of 64 pixels between the 90% and 100% sample chronoprints.

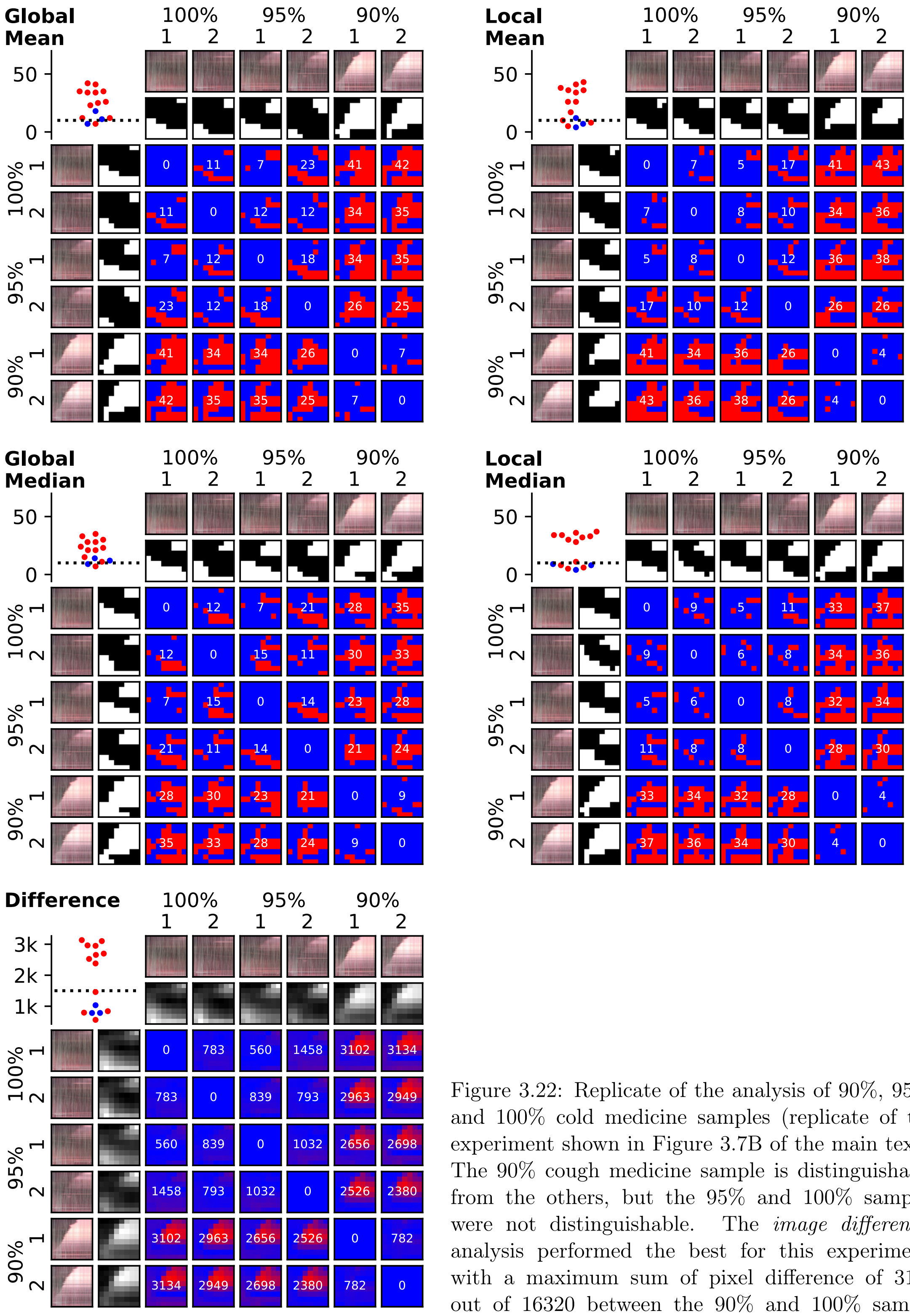


Figure 3.22: Replicate of the analysis of 90%, 95%, and 100% cold medicine samples (replicate of the experiment shown in Figure 3.7B of the main text). The 90% cough medicine sample is distinguishable from the others, but the 95% and 100% samples were not distinguishable. The *image differences* analysis performed the best for this experiment, with a maximum sum of pixel difference of 3134 out of 16320 between the 90% and 100% sample chronoprints.

```
%visualizing how they change over space and time" by
    Brittney A. McKenzie,
%Jessica Robles-Najar, Eric Duong, Philip Brisk, and
    William H. Grover.
%
%Updated versions of this code are available on groverlab.
    org

%Generate Chronoprints
%Access directory folder with video frame images
srcfiles = dir('C:\Users\User\Folder␣with␣video␣frames\*.
    png');

%Naturally sort file names in numerical order
files = natsortfiles({srcfiles.name});

%Number of rows of pixels in channel
numrow = 443;
for framenum = 1 : length(files)%For every frame in your
    directory of frames

    %Concatinates filename into a string
    filename = strcat('C:\Users\User\Folder␣with␣video␣
       frames\',char(files(framenum)));

    %Read image
    frame = imread(filename);

    %Define channel regions by pixel columns
    ch{1} = frame(:,81:98,:);
    ch{2} = frame(:,124:141,:);
    ch{3} = frame(:,167:184,:);
    ch{4} = frame(:,211:228,:);
    ch{5} = frame(:,255:272,:);
    ch{6} = frame(:,299:316,:);

    for row = 1:numrow %For every row of pixels in the
       chanel

        for channelnum = 1:6 %for all channels

            %Separate image into RGB components and compute
               average RGB pixel
            %values for each row of pixels in the channel image for each
            %frame
            mean_r = mean(ch{channelnum}(row,:,1),'native');
            mean_g = mean(ch{channelnum}(row,:,2),'native');
            mean_b = mean(ch{channelnum}(row,:,3),'native');

            %Matrix stores the average pixel RGB values for each row of pixels in the
            %channel image for each frame
            slice{channelnum}(row,framenum,:) = cat(3, mean_r,mean_g,mean_b);
        end
    end
end

%Show all chronoprint images and save them to a file
for num =1:6
```

```
    figure,reslice{num} = imshow(slice{num}); %Show all
        chronoprint images

    %Save chronoprint image to file
    imwrite(slice{num}, sprintf('Chronoprint%01d.png',num))
        ;
end
```

3.6.4 `OilChronoprintAnalysis.m`: MATLAB software for analyzing chronoprints using the *feature tracing* method, used to generate Figures 3.2 and 3.5 and supplementary Figures 3.10–3.12.

The source code for the software used to analyze chronoprints with the *feature tracing* method is shown below and can be downloaded at https://pubs.acs.org/doi/10.1021/acscentsci.8b00860.

```
%OilChronoprintAnalysis.m
%
%This code accompanies the paper "Chronoprints: Identifying
    samples by
%visualizing how they change over space and time" by
   Brittney A. McKenzie,
%Jessica Robles-Najar, Eric Duong, Philip Brisk, and
   William H. Grover.
```

```
%
%Updated versions of this code are available on groverlab.
    org

clc; clear all; close all;

for num = 1:2 %number of samples to analyze

    %define number of image pixel rows and columns
    rownum = 457;
    colnum = 555;
    rownum_1 = rownum-1;

    %Reads chronoprint images
    Reslice1_{num} = imread(sprintf('Chronoprint%01d.png',
        num));

    %subtract background to improve contrast and reduce
        noise
    Reslice2_{num} = Reslice1_{num}(:,:,:) - Reslice1_{num
        }(:,1,:)/2; %remove noise with half of first frame

```

```
%Convert image to binary
thresholdvalue = 0.039;
Reslice3_{num} = im2bw(Reslice2_{num}, thresholdvalue);

%New matrix to store isolated curve values
Reslice4_{num} = ones(rownum_1,colnum);

%Isolate curve values and store values in matrix
for row = 1:rownum_1
    for col = 2:colnum
        if Reslice3_{num}(row,col)== Reslice3_{num}(row
           +1,col) && Reslice3_{num}(row,col)==
           Reslice3_{num}(row+1,col-1)
            Reslice4_{num}(row,col)= 1;
        else
            Reslice4_{num}(row,col)= 0;
        end
    end
end

%New vector to store Y values (space along channel in
   pixels) of curve
```

```
rawdata{num} = zeros(1,colnum);

%Record raw data Y values (pixel distance along channel
    ) of curve
for rows = 1:rownum_1
    for cols = 1:colnum
        if Reslice4_{num}(rows,cols)== 0
            rawdata{num}(cols)= rownum-rows;
        end
    end
end

%Time vectors with adjusted time scale to account for
    frame rate
framerate = 8; %frames/sec
t{num} = (1:length(rawdata{num}))/framerate; %adjusted
    time vector

%Remove zero placeholder values from data and replace
    with last
%successful trace value
rawdata{3}(2)= 0; rawdata{4}(2)= 0; rawdata{6}(2)= 0;
```

```
    newdata{num} = interp1(1:nnz(rawdata{num}), rawdata{num
       }(rawdata{num} ~= 0), cumsum(rawdata{num} ~= 0), '
       NearestNeighbor');
    newdata{num}(isnan(newdata{num})) = rawdata{num}(isnan(
       newdata{num})); %Keeps starting zero

    %Apply Savitzky-Golay filter
    order = 3; %polynomial order
    framelen = 31; %frame length
    Y{num} = sgolayfilt(newdata{num},order,framelen);
end

%Compute sum of squared differences
ssd = sum((Y{1} - Y{2}).^2);

%Plot data
figure
plot(t{1},Y{1},t{2},Y{2});
title('Oil␣Samples');
ylabel('Space␣(px)');
xlabel('Time␣(s)');
legend({'Sample␣1','Sample␣2'});
```

```
axis([0 75 0 500]);
```

3.6.5 `chronoprint.py` and `workup.py`: Python software for analyzing chronoprints using the *image hashing* and *image differences* methods, used to generate Figures 3.3, 3.4, and 3.6–3.8, and supplementary Figures 3.13 – 3.22.

The source code forthe software used to analyze chronoprints with the *image hashing* and *image differences* methods is shown below and can be downloaded at https://pubs.acs.org/doi/10.1021/acscentsci.8b00860.

The **chronoprint.py** code computes the *image differences* and *image hashing* methods for each data set.

```
# chronoprint.py
#
# This code accompanies the paper "Chronoprints:
    Identifying samples by
# visualizing how they change over space and time" by
    Brittney A. McKenzie,
# Jessica Robles-Najar, Eric Duong, Philip Brisk, and
    William H. Grover.
#
# Updated versions of this code are available on groverlab.
    org
```

```

import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import matplotlib.ticker as ticker
import scipy.misc
import os
import seaborn as sns
import pandas as pd


class chronoprint():
    def __init__(self, file, substance, description=""):
        self.file = file
        self.substance = substance
        self.description = description
        self.rgb = mpimg.imread(self.file)
        self.r = self.rgb[:,:,0]
        self.g = self.rgb[:,:,1]
        self.b = self.rgb[:,:,2]
        self.mono = np.dot(self.rgb[...,:3], [0.299, 0.587,
            0.114])
        self.resolution = 8
        self.reduced = scipy.misc.imresize(self.mono, (self
           .resolution, self.resolution))
        self.reduced_mean = np.mean(self.reduced)
        self.reduced_median = np.median(self.reduced)
    def view_rgb(self):
        plt.imshow(self.rgb)
        plt.title(self.file)
        plt.show()
    def view_r(self):
        plt.imshow(self.r, cmap="gray")
        plt.title(self.file + ":␣RED")
        plt.show()
    def view_g(self):
        plt.imshow(self.g, cmap="gray")
        plt.title(self.file + ":␣GREEN")
        plt.show()
    def view_b(self):
        plt.imshow(self.b, cmap="gray")
        plt.title(self.file + ":␣BLUE")
        plt.show()
    def view_mono(self):
        plt.imshow(self.mono, cmap="gray")
        plt.title(self.file + ":␣MONO")
        plt.show()
    def hash(self):
        print(self.mono)


class label():
    def __init__(self, line, location, width, text):
        self.line = line
        self.location = location
        self.width = width
        self.text = text

        if location == -2:
            self.x1, self.y1 = 0.06, 0.91
            self.x2, self.y2 = 0.09, 0.94
        if location == -1:
            self.x1, self.y1 = 0.17, 0.91
            self.x2, self.y2 = 0.09, 0.83
        if location == 0:
            self.x1, self.y1 = 0.28, 0.91
            self.x2, self.y2 = 0.09, 0.72
        if location == 1:
            self.x1, self.y1 = 0.39, 0.91
            self.x2, self.y2 = 0.09, 0.61
        if location == 2:
            self.x1, self.y1 = 0.50, 0.91
            self.x2, self.y2 = 0.09, 0.50
        if location == 3:
            self.x1, self.y1 = 0.61, 0.91
            self.x2, self.y2 = 0.09, 0.39
        if location == 4:
            self.x1, self.y1 = 0.72, 0.91
            self.x2, self.y2 = 0.09, 0.28
        if location == 5:
            self.x1, self.y1 = 0.83, 0.91
            self.x2, self.y2 = 0.09, 0.17
        if location == 6:
            self.x1, self.y1 = 0.94, 0.91
            self.x2, self.y2 = 0.09, 0.06

        if self.width == 2:
            self.x1 = self.x1 + 0.055
            self.y2 = self.y2 - 0.055
        if self.width == 3:
            self.x1 = self.x1 + 0.11
            self.y2 = self.y2 - 0.11
        if self.width == 6:
            self.x1 = self.x1 + 0.22 + 0.055
            self.y2 = self.y2 - 0.22 - 0.055
        if self.line == 1:
            self.y1 = self.y1 + 0.05
            self.x2 = self.x2 - 0.05


class experiment():
    def __init__(self, chronoprints, description="default")
        :
        print("initializing")
        self.chronoprints = chronoprints
        self.reduced_combo = np.hstack((c.reduced for c in
            self.chronoprints))
        self.global_mean = np.mean(self.reduced_combo)
        self.global_median = np.median(self.reduced_combo)
        self.description = description
        self.labels = []
        self.heading = "Temporary"
    def print_stats(self):
        print("Experiment␣min␣=", self.min)
        print("Experiment␣max␣=", self.max)
        for c in self.chronoprints:
            print(c.file, np.min(c.mono), np.max(c.mono))
    def view_rgb(self):
        for i, c in enumerate(self.chronoprints):
            plt.subplot(len(self.chronoprints), 1, i+1)
            plt.imshow(c.rgb)
        plt.show()
    def view_mono(self):
        for i, c in enumerate(self.chronoprints):
            plt.subplot(len(self.chronoprints), 1, i+1)
            plt.imshow(c.mono, cmap="gray", vmin=self.min,
               vmax=self.max)
        plt.show()
    def view_r(self):
        for i, c in enumerate(self.chronoprints):
            plt.subplot(len(self.chronoprints), 1, i+1)
            plt.imshow(c.r, cmap="gray", vmin=self.min, vmax=self.max)
        plt.show()
    def view_g(self):
        for i, c in enumerate(self.chronoprints):
            plt.subplot(len(self.chronoprints), 1, i+1)
            plt.imshow(c.g, cmap="gray", vmin=self.min, vmax=self.max)
        plt.show()
    def view_b(self):
        for i, c in enumerate(self.chronoprints):
            plt.subplot(len(self.chronoprints), 1, i+1)
            plt.imshow(c.b, cmap="gray", vmin=self.min, vmax=self.max)
        plt.show()


    def hash(self, algorithm="None"):
        results = np.zeros((6, 6))
        resolution = 8
        individual_means = True
        fig, ax = plt.subplots(figsize=(3, 3))
        # ADD THE TEXT LABELS
        for label in self.labels:
            fig.text(label.x1, label.y1, label.text, ha="
               center", va="center")
            fig.text(label.x2, label.y2, label.text, ha="
               center", va="center", rotation='vertical')
        fig.text(0.01, 0.98, algorithm.replace("␣", "\n"),
           weight="bold", ha="left", va="top")

        # plot rgb index images
        for i, c in enumerate(self.chronoprints):
            # plot left index images
            plt.subplot2grid((9, 9), (i+3, 1))
            plt.imshow(c.rgb, extent=[0,100,0,1], aspect
               =100)
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])
            # plot top index images
            plt.subplot2grid((9, 9), (1, i+3))
            plt.imshow(c.rgb, extent=[0,100,0,1], aspect
               =100)
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])

        # plot binary index images
        for i, c in enumerate(self.chronoprints):
            if algorithm=="Local␣Mean":
                t1 = c.reduced > c.reduced_mean
            elif algorithm=="Local␣Median":
                t1 = c.reduced > c.reduced_median
            elif algorithm=="Global␣Mean":
                t1 = c.reduced > self.global_mean
            elif algorithm=="Global␣Median":
                t1 = c.reduced > self.global_median
            else:
                raise ValueError('Unsupported␣algorithm')
            # plot left binary index images
            plt.subplot2grid((9, 9), (i+3, 2))
            plt.imshow(t1, cmap="gray")
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])
            # plot top binary index images
            plt.subplot2grid((9, 9), (2, i+3))
            plt.imshow(t1, cmap="gray")
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])

        df = pd.DataFrame({"delta" : [], "relationship" :
            []})

        # plot hash images
        for i, c1 in enumerate(self.chronoprints):
            for j, c2 in enumerate(self.chronoprints):
                if algorithm=="Local␣Mean":
                    t1 = c1.reduced > c1.reduced_mean
                    t2 = c2.reduced > c2.reduced_mean
                elif algorithm=="Local␣Median":
                    t1 = c1.reduced > c1.reduced_median
                    t2 = c2.reduced > c2.reduced_median
                elif algorithm=="Global␣Mean":
                    t1 = c1.reduced > self.global_mean
                    t2 = c2.reduced > self.global_mean
                elif algorithm=="Global␣Median":
                    t1 = c1.reduced > self.global_median
                    t2 = c2.reduced > self.global_median
                else:
                    raise ValueError('Unsupported␣algorithm
                       ')

                # plot ahash images
                plt.subplot2grid((9, 9), (i+3, j+3))
                plt.imshow(np.logical_xor(t1, t2), cmap = "
                   brg",
                           vmin=0, vmax=2)
                plt.gca().set_xticks([])
                plt.gca().set_yticks([])
                results[i][j] = np.sum(np.logical_xor(t1,
                   t2))

                plt.text(3.5, 3.5, np.sum(np.logical_xor(t1
                   , t2)),
                         ha="center", va="center", fontsize
                            =6, color="white")
                if i==j:  # this happens on the diagonal
                    pass
                elif i > j:  # this rules out repeats!
                    pass
                elif c1.substance == c2.substance:
                    df = df.append(pd.DataFrame({"delta" : [np.sum(np.logical_xor(t1, t2))], "relationship" : ["same"]}))
                else:
                    df = df.append(pd.DataFrame({"delta" : [np.sum(np.logical_xor(t1, t2))], "relationship" : ["diff"]}))

        plt.subplot2grid((9, 9), (1, 1), rowspan=2, colspan=2)
        plt.ylim((-6,70))  # this uses the full range of possible values
        sns.swarmplot(y=df["delta"], hue=df["relationship"], x=[""]*len(df), palette=["b", "r"], size=3)
        plt.gca().legend_.remove()
        plt.gca().set_xticks([])
        plt.gca().yaxis.label.set_visible(False)
        plt.gca().spines['top'].set_visible(False)
        plt.gca().spines['right'].set_visible(False)
        plt.gca().spines['bottom'].set_visible(False)
```

```
        plt.subplots_adjust(left=0.01, bottom=0.01, right=0.99, top=0.99,
                            wspace=0.15, hspace=0.15)
        plt.savefig(self.description + "␣" + algorithm.upper() + ".pdf", dpi=600)


    def diff(self):
        differences = np.zeros((len(self.chronoprints), len(self.chronoprints)))
        results = np.zeros((len(self.chronoprints), len(self.chronoprints)))
        resolution = 8
        fig, ax = plt.subplots(figsize=(3, 3))

        # ADD THE TEXT LABELS
        for label in self.labels:
            fig.text(label.x1, label.y1, label.text, ha="center", va="center")
            fig.text(label.x2, label.y2, label.text, ha="center", va="center", rotation='vertical')
        fig.text(0.01, 0.96, "Difference", weight="bold",
           ha="left", va="center")

        # plot rgb index images
        for i, c in enumerate(self.chronoprints):
            # plot left index images
            plt.subplot2grid((9, 9), (i+3, 1))
            plt.imshow(c.rgb, extent=[0,100,0,1], aspect
               =100)
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])
            # plot top index images
            plt.subplot2grid((9, 9), (1, i+3))
            plt.imshow(c.rgb, extent=[0,100,0,1], aspect
               =100)
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])

        # plot mono index images
        for i, c in enumerate(self.chronoprints):
            # t1 = np.float64(c.reduced)
            t1 = c.reduced
            # plot left mono index images
            plt.subplot2grid((9, 9), (i+3, 2))
            plt.imshow(t1, cmap="gray")
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])
            # plot top mono index images
            plt.subplot2grid((9, 9), (2, i+3))
            plt.imshow(t1, cmap="gray")
            plt.gca().set_xticks([])
            plt.gca().set_yticks([])

        df = pd.DataFrame({"delta" : [], "relationship" :
            []})

        # plot diff images
        for i, c1 in enumerate(self.chronoprints):
            for j, c2 in enumerate(self.chronoprints):
                t1 = c1.reduced
                t2 = c2.reduced

                difference = np.where(t1>t2, t1-t2, t2-t1)
                    # uint8, 0-255
                plt.subplot2grid((9, 9), (i+3, j+3))
                plt.imshow(difference, cmap = "brg",
                           vmin=0, vmax=255)  # was
                              difference_max
                plt.gca().set_xticks([])
                plt.gca().set_yticks([])


                differences[i][j] = np.sum(difference)
                plt.text(3.5, 3.5, str(np.sum(difference)),
                    ha="center", va="center", fontsize=6,
                   color="white")
                if i==j:  # this happens on the diagonal
                    pass
                elif i > j:  # this rules out repeats
                    pass
                elif c1.substance == c2.substance:
                    df = df.append(pd.DataFrame({"delta" :
                       [np.sum(difference)], "relationship"
                        : ["same"]}))
                else:
```

```
            df = df.append(pd.DataFrame({"delta" : [np.sum(difference)], "relationship" : ["diff"]}))
        results[i][j] = np.sum(np.square(np.float64(difference)))


plt.subplot2grid((9, 9), (1, 1), rowspan=2, colspan=2)
plt.plot([-1,1],[1500,1500], "k:")
sns.swarmplot(y=df["delta"], hue=df["relationship"], x=[""]*len(df), palette=["b", "r"], size=3)
plt.gca().legend_.remove()
plt.gca().set_xticks([])
plt.gca().yaxis.label.set_visible(False)
plt.gca().spines[’top’].set_visible(False)
plt.gca().spines[’right’].set_visible(False)
plt.gca().spines[’bottom’].set_visible(False)

mkfunc = lambda x, pos: ’%1.0fM’ % (x * 1e-6) if x >= 1e6 else ’%1.0fk’ % (x * 1e-3) if x >= 1e3 else ’%1.0f’ % x
```

```
        mkformatter = ticker.FuncFormatter(mkfunc)
        plt.gca().yaxis.set_major_formatter(mkformatter)

        plt.subplots_adjust(left=0.01, bottom=0.01, right
            =0.99, top=0.99,
                                    wspace=0.15, hspace=0.15)

        plt.savefig(self.description + "␣DIFF.pdf", dpi
            =600)
```

The **workup.py** code displays the *image differences* and *image hashing* computation results.

```
# workup.py
#
# This code accompanies the paper "Chronoprints:
    Identifying samples by
# visualizing how they change over space and time" by
    Brittney A. McKenzie,
# Jessica Robles-Najar, Eric Duong, Philip Brisk, and
    William H. Grover.
#
# Updated versions of this code are available on groverlab.
    org
```

```

import chronoprint as cp

def workup(experiment):
    experiment.hash("Local␣Mean")
    experiment.hash("Local␣Median")
    experiment.hash("Global␣Mean")
    experiment.hash("Global␣Median")
    experiment.diff()

chronoprints = []
for f, s in zip(("1.png", "4.png", "2.png", "5.png", "3.png
   ", "6.png",),
                ("100", "100", "75", "75", "50", "50")):
    chronoprints.append(cp.chronoprint(f, s))

e = cp.experiment(chronoprints, "Cold␣medicine")
e.labels.append(cp.label(line = 1, location = 1, width = 2,
    text = "100%"))
e.labels.append(cp.label(line = 1, location = 3, width = 2,
    text = "75%"))
```

```
e.labels.append(cp.label(line = 1, location = 5, width = 2,
    text = "50%"))
e.labels.append(cp.label(line = 2, location = 1, width = 1,
    text = "1"))
e.labels.append(cp.label(line = 2, location = 2, width = 1,
    text = "2"))
e.labels.append(cp.label(line = 2, location = 3, width = 1,
    text = "1"))
e.labels.append(cp.label(line = 2, location = 4, width = 1,
    text = "2"))
e.labels.append(cp.label(line = 2, location = 5, width = 1,
    text = "1"))
e.labels.append(cp.label(line = 2, location = 6, width = 1,
    text = "2"))
workup(e)
```

Chapter 4

Enhanced chronoprints: Identifying samples by visualizing changes in particle interactions

4.1 Introduction

The chronoprint technique described in Chapter 3 can identify substances by visualizing how they change in response to a perturbation over space and time [57]. The chronoprint technique can identify adulterated foodstuffs and medicines. Although chronoprinting can be used to identify a wide range of samples since all substances have physical properties, the technique is limited to substances that produce a visual change in appearance in response to a perturbation. To overcome this limitation I use beads along with a thermal gradient and gravitational perturbation to discriminate substances in chronological fingerprints. This is an enhancement to the original chronoprint technique, thus I call the resulting images “enhanced chronoprints.”

Essentially any perturbation source that induces a change within a substance can be used to produce chronoprints. In the previous demonstration, a dynamic temperature gradient perturbation was applied to the samples by partially submerging a microfluidic chip containing samples in a liquid nitrogen bath [57]. In this work a chronological fingerprint of a metal bead falling through a substance (using Earth’s applied gravitational acceleration, 9.8 m/s^2 or 1 g) with a temperature gradient is produced and compared (Figure 4.1). Since physical properties vary by substance, the time-dependent trajectory of a bead moving through the substance may also vary. The velocity of a bead accelerated through a sample is a function of the bead’s properties (its size and density), the sample’s properties (its viscosity and density), and the force applied to the bead (1 g if using Earth’s gravitational acceleration, or more if using e.g. centrifugal acceleration). For spherical beads, this relationship is described by Stokes’ Law for the terminal velocity of a sphere falling in a fluid:

$$v = \frac{2}{9}\,\frac{(\rho_p - \rho_f)}{\mu} g r^2 \tag{4.1}$$

where ρ_p is the density of the bead, ρ_f is the density of the fluid, μ is the dynamic viscosity of the fluid, g is the gravitational acceleration, and r is the radius of the bead.

In the enhanced chronoprint technique the size and density of beads are likely constant over the course of an experiment, but the viscosity and density of the samples may vary from one sample to another; this variance will result in different chronoprints for the

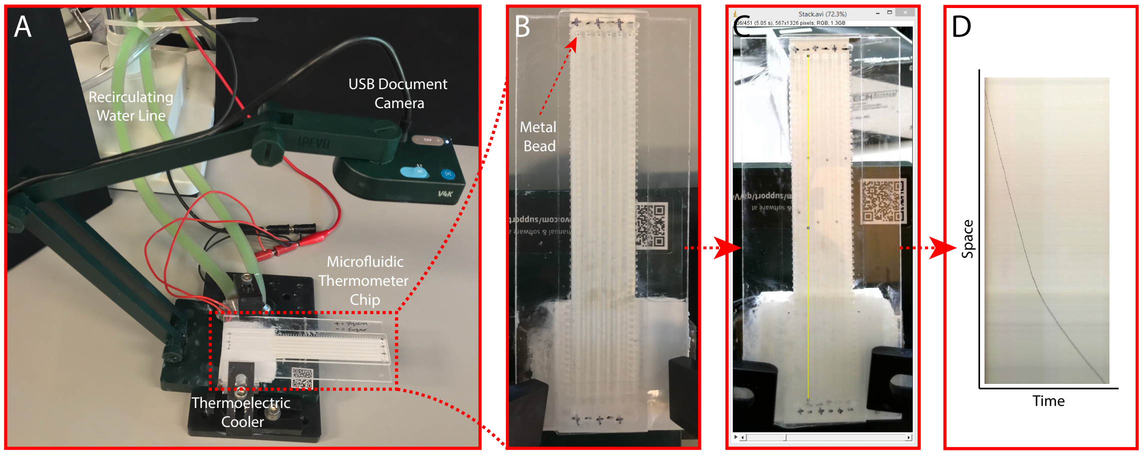


Figure 4.1: Producing an "*enhanced chronoprint*" capturing how six samples (in this example, three 50% and three 60% (v/v) glycerol in water dilutions) with metal beads respond to a perturbation over space and time (in this case, a temperature gradient and gravitational force). **(A)** A microfluidic thermometer chip containing the samples and beads is partially suspended off of a thermoelectric cooler to establish a temperature gradient along the chip. The chip gets tilted at an angle to apply a gravitational force. **(B)** The chip contains six samples (transparent), each with a metal bead (black circle), loaded in microfluidic channels that run parallel to the temperature gradient. **(C)** An inexpensive USB document camera records a video of the change in the falling bead trajectory as the physical properties of the sample changes in response to the temperature gradient. For each sample, a single column of pixels—that intersects the bead trajectory— in each channel image for each frame of the video is extracted. **(D)** By placing these pixel columns side-by-side, we create a bitmap image (the sample's enhanced chronoprint) that captures how the falling bead, and thus the sample, changes over space (the y-axis) and time (the x-axis). Finally, by comparing the enhanced chronoprints of all six samples in the chip, we can determine whether the samples are either likely the same or definitely different.

different samples and enable the user to chemically distinguish the samples. The substance's unique behavior would be captured in the chronorprint digital image and can be used as the basis for distinguishing between different samples. In the experiments a thermal perturbation is activated with thermoelectric coolers to produce a temperature gradient along the chip, and then a gravitational force (Earth's gravitational acceleration 1 g in this case) is applied by tilting the chip at an angle. A video recording of the experiment is converted to a digital image (chronoprint) that shows how the falling bead trajectory, and thus the sample, changes over space and time. During this perturbation a substance's viscosity and density are a function of its temperature, and a particle traveling through the substance during the perturbation would experience a velocity change as it passes through the substance. In practical terms, this means that a bead dropped through a sample in a channel will speed up and slow down, or even stop or reverse at different points within the sample due to the changes in substance's material properties. In addition, the thermal gradients within the sample are functions of the substances' thermal conductivity, Reynolds number (the ratio of inertial forces to viscous forces within the fluid), and Prandtl number (ratio of kinematic viscosity to thermal diffusivity), since the geometry and thermal conductivity of the fluidic chip is the same, the thermal gradient within the substance may differ for chemically different substances resulting in varying particle behavior. Overall, the combination of thermal and gravitational perturbations results in a complex path followed by the particle in the sample's digital image, a path that is dictated not only by the substance's viscosity, density and material properties, but also how those properties change as a function of temperature. Therefore, in the (already unlikely) event that two different substances have the same viscosity and density at one temperature, then chronoprints can still distinguish the two substances based on how their viscosities and densities change at different temperatures. Such a characterization adds significant discrimination power to the chronoprint technique and broadens the range of sample types that can be analyzed with the chronoprint technique.

To demonstrate a proof-of-concept of the enhanced chronoprint technique, I compare enhanced chronoprints of different glycerol dilutions in water, as well as enhanced chonronprints of the occasionally-confused pharmaceutical ingredients glycerol and diethylene glycol (whose accidental or intentional substitution has led to hundreds of deaths).

4.2 Materials and Methods

4.2.1 Fabricating microfluidic thermometer chips

The microfluidic thermometer chips [41] were designed using Adobe Illustrator (Adobe Systems Inc., San Jose, CA). Each chip is 125 mm long, 25 mm wide, and contains six parallel microfluidic channels. Each channel is 1.5 mm wide, 1 mm deep, and 115 mm long, with 1.5 mm diameter input/output reservoirs at each end, 1.5 mm space between channels, and markers spaced every 1 mm along the sides of the chip for length measurements. A DXF file of the chip design was exported and engraved into 3 mm thick poly(methyl methacrylate) pieces (Professional Plastics Inc., Fullerton, CA) using a computer-controlled hobbyist-grade milling machine (Bantam Tools, Berkeley, CA). The open channels were enclosed by fusing on another piece of 3 mm thick poly(methyl methacrylate) with ethanol and heat [58].

4.2.2 Preparing samples

Glycerol solutions along with diethylene glycol soltuions were analyzed in this work. 50% and 60% (v/v) glycerol dilutions in water were prepared and compared. To show that the enhanced technique can still distinguish two occasionally-confused chemicals in pharmaceutical manufacturing, we obtained chronoprints from samples of diethylene glycol (a transparent and sweet-tasting but poisonous liquid) and glycerol (a similar but nonpoisonous liquid) from Sigma-Aldrich (St. Louis, MO). About 75 μL of each sample was loaded into the thermometer chip along with a single 1/32" diameter stainless steel metal bead (McMaster Carr Supply Co, Santa Fe Springs, CA) for each experiment. The chip outlets were taped to prevent the channel contents from emptying out during the experiments.

4.2.3 Obtaining enhanced chronoprints

After filling a chip with samples and the metal beads to analyze, the chip was placed horizontally at a 30° angle to allow all of the beads to consistently fall down one edge of the chip channels when tilted. Two thirds of the chip was then suspended off of a thermoelectric cooler (TEC1-12706, Hebei I.T. Co., Shanghai, China) cooled to -20 °C, which applies a freezing perturbation and creates a temperature gradient along the channels. The thermoelectric cooler is connected to a recirculating water line that removes excess heat from the backside of the cooler. The chip was then tilted vertically from a 0° angle to a 70° angle to apply a gravitational perturbation, and a video recording of the chip contents was captured using an inexpensive USB document camera (V4K Ultra HD, Ipevo Inc., Sunnyvale, CA). All six sample channels in the chip were exposed to the same temperature gradient and the same gravitational force. After all beads finished falling through the 50% and 60% glycerol dilutions in water, the recording was stopped. In some runs, not all of the beads successfully traveled through the samples. In cases where beads were stuck at the inlets or may have became stuck along the way, due to being trapped by tiny air bubbles, chronoprints were not obtained for those samples. An example of a chronoprint with a bead that briefly became stuck after falling is shown in the *Supplementary Information.* For the glycerol and diethylene glycol sample comparisions, the video recording was stopped after all beads in the diethylene glycol were finished falling. The image analysis software, ImageJ (NIH Research Services Branch), was used along with the "reslice" tool within the program to convert the videos to six enhanced chronoprints (one per sample). For each sample, the reslice tool takes a single column of pixels—that intersects the trajectory of the bead— in each channel image for each frame of the video and then places all of these columns of pixels side-by-side to create a bitmap image that is the sample's chronoprint, with space (distance along the channel) in the vertical dimension and time in the horizontal dimension. This process is then repeated for each sample in the experiment, and the resulting chronoprints are ready for comparison and similarity analysis.

4.2.4 Comparing enhanced chronoprints

For enhanced chronoprints with just one or two dominant features, one can simply trace the boundary between these features and convert each chronoprint to a curve; these curves can then be compared to each other to quantify the similarity of the samples. An example of this approach for chronoprint comparison is shown in Figure 4.2, and is a simplified version of the *feature tracing* method used in the previous chronoprint work.[57] In this process, a custom MATLAB program (*Supporting Information*) first enhances transforms the image so that space (distance along channel) is in the vertical direction and time is in the horizontal direction of the image. The program then converts each chronoprint (Figure 4.2A) from color to monochrome (Figure 4.2B), then the program compares the minimum pixel value in each row of pixels in the image to a constant threshold provided by the user; pixels with values below that threshold are recorded and plotted (bead pixels), and pixels with values above that threshold (chip background pixels) are removed (Figure 4.2C). If two curves are similar, this suggests that the two samples analyzed may be the same; but if two curves are significantly different, this is proof that the samples are chemically different. The degree of similarity between two samples is quantified by summing the squared differences between the y-axis values of the curves (the distances along the channel) at each point along the curves.

4.3 Results and Discussion

4.3.1 Distinguishing colorless solutions

As a proof-of-concept, we used*Enhanced Chronoprints* to discriminate two colorless solutions (that might not show a visible change with the previous chronoprint technique). To do this, I obtained enhanced chronoprints for 50% and 60% (v/v) glycerol solutions in water and analyzed them. Glycerol is a nontoxic ingredient commonly used in food products, cosmetics, and medicines for many of its properties including, but not limited to, its sweet taste, its viscosity, and humectancy. The trajectory of the bead in each sample is dictated

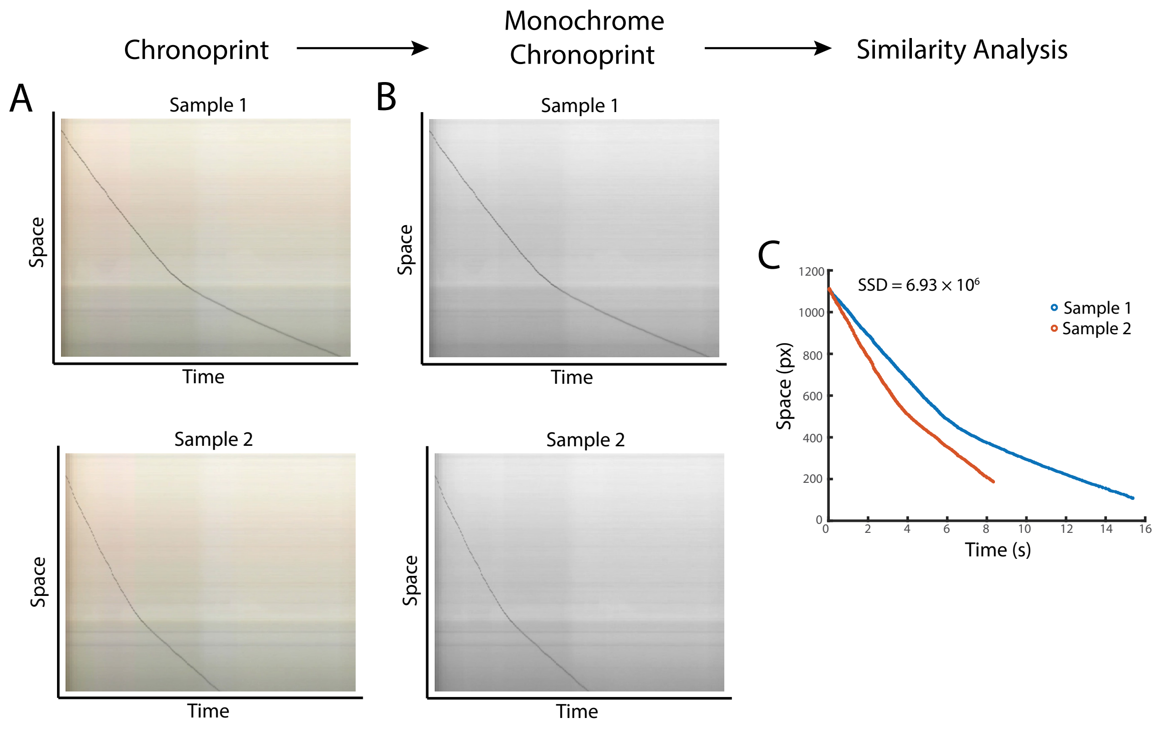


Figure 4.2: Overview of comparing chronoprints (in this case, obtained from two different glycerol solutions). Color chronoprints of each sample **(A)** are converted to monochrome chronoprints **(B)** by comparing each minimum pixel value in each row of pixels to a constant threshold value; pixels above the threshold (chip background pixels) are removed, and pixels below the threshold (bead pixels) are recorded. The code (*Supporting Information*) then plots each curve to compare the two chronoprints **(C)**. Curves that are significantly different (like these) confirm that the two samples are chemically different. The sum of squared differences (SSD) between the y-axis values of the curves at each point along the curves (6.93×10^6 in this case) serves to quantify the degree of similarity between the two glycerol solution samples

by the substance's viscosity, density and material properties, and these properties change as a function of temperature. Therefore samples with differing composition should display unique bead trajectories in the enhanced chronoprints. In Figure 4.3, the experimental results for the 50% and 60% (v/v) glycerol dilutions in water confirm this expectation, and the changes in the colorless samples can be visualized through the bead trajectory curves from the enhanced chronoprints.

Within each glycerol sample dilution, each pair of samples resulted in very similar curves: the sum-of-squared differences was only 8.72×10^4 for the two 50% glycerol samples, 9.68×10^3 for the two 60% glycerol dilution samples. However, the different dilutions had very different curves: the maximum sum-of-squared-differences was 7.41×10^6 for the 50% and 60% glycerol dilution samples. Like in my previous chronoprint work, we found that chronoprints with sum-of-squared-differences greater than about 1×10^6 indicated that the substances were different, and chronoprints with sum-of-squared-differences less than 1×10^6 indicated that the substances were the same. Additional chronoprints and analysis of experiments with glycerol dilution samples are provided in *Supporting Information.*

4.3.2 Identifying toxic pharmaceutical ingredients

In 1937, the S.E. Massengill Company in Bristol, Tennessee, unintentionally substituted the toxic substance, diethylene glycol, for nontoxic glycerol in a liquid formulation of the early antibiotic sulfanilamide. Over 100 people that had used the resulting medicine, called "Elixir Sulfanilamide," were fatally poisoned [54, 55], and the toxicity of diethylene glycol then became commonly known among pharmaceutical companies. Remarkably, how-

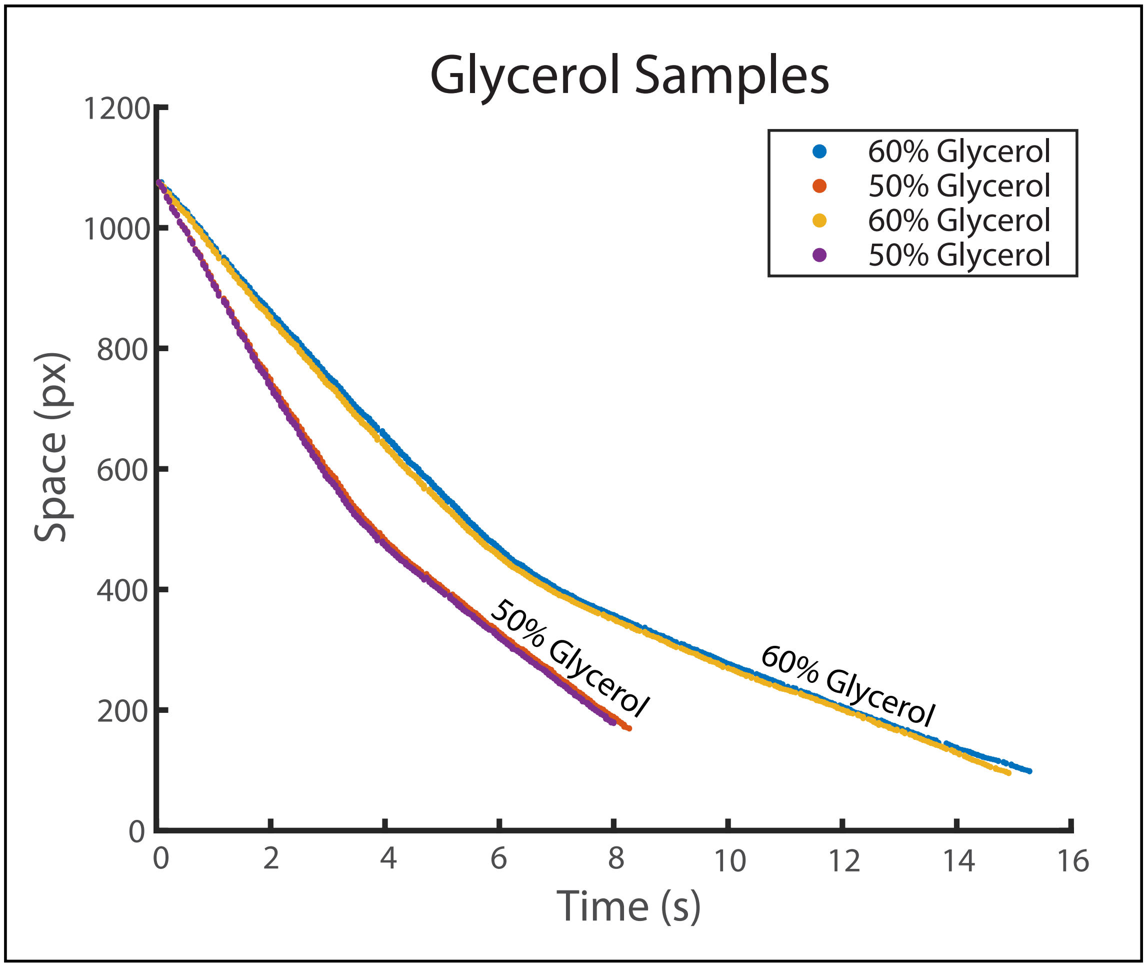


Figure 4.3: Distinguishing colorless glycerol dilutions using enhanced chronoprints. This plot compares chronoprints from four glycerol dilutions in water samples, converted to curves using the method described in section 4.2.4. Chronoprint curves from two samples each of glycerol dilutions (50% and 60% (v/v) in water) are similar within each dilution but significantly different between the different dilutions. The maximum sum-of-squared-differences between two different dilutions (7.41×10^6 difference between the 50% and 60% glycerol dilutions samples) is greater than the threshold of 1×10^6 and confirms that these samples are different.

ever, poisonings due to the use of diethylene glycol in medicines remain tragically recurrent today, with mass poisoning occurring every two years on average somehwere in the world since 1985.[56] A lot of these incidents occur in resource-limited settings where pharmaceutical companies may not have the tools needed to confirm the identity (and safety) of their manufacturing stock ingredients.

To determine whether the enhanced chronoprint technique could distinguish toxic diethylene glycol from nontoxic glycerol, we filled a microfluidic chip with three samples of each substance, partially suspended the chip off of a thermoelectric cooler that is cooled to -20 °C, and obtained chronoprints from the video recording of the chip. The chronoprints were then analyzed using the comparison technique described in section 4.2.4. The results, shown in Figure 4.4, confirm that all the glycerol enhanced chronoprint curves are very similar (never differing by more than a sum-of-squared-differences of 1.12×10^3), as are all the diethylene glycol enhanced chronoprint curves (never differing by more than a maximum of 4.92×10^5). However, the glycerol enhanced chronoprint curves are significantly different from the diethylene glycol enhanced chronoprint curves (differing by at least a sum-of-squared-differences of 1.29×10^6). These results confirm that the enhanced chronoprint method can easily distinguish between the two colorless, toxic diethylene glycol and nontoxic glycerol.

4.4 Conclusion

In this work, I introduced *enhanced chronoprints*, an enhancement to the original chronoprint method [57], an image-based technique for identifying or distinguishing

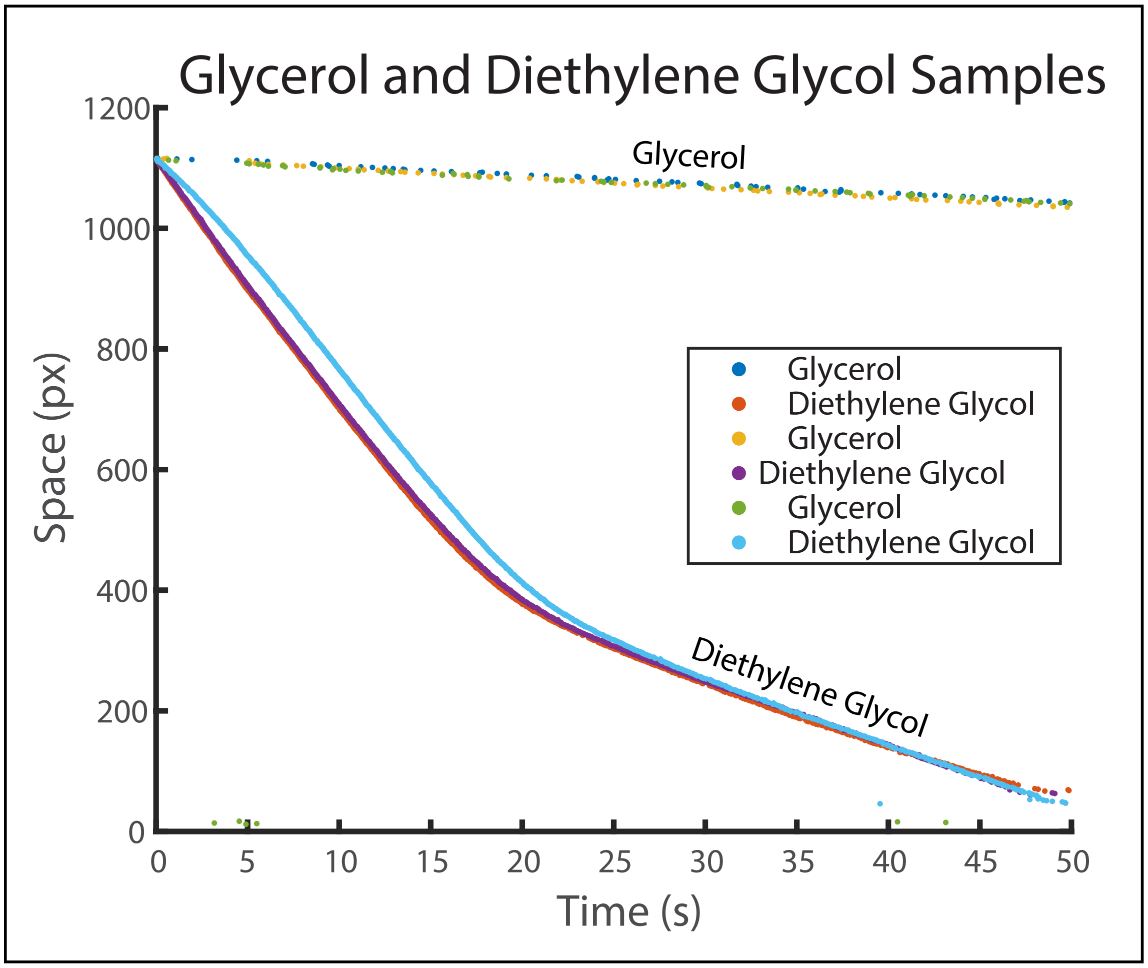


Figure 4.4: Distinguishing toxic and nontoxic pharmaceutical ingredients using enhanced chronoprints. The enhanced chronoprints of three samples of toxic diethylene glycol and three samples of nontoxic glycerol were analyzed using the method described in section 4.2.4. The three glycerol enhanced chronoprint curves were nearly identical (differing by a maximum sum-of-squared-differences of 1.12×10^3), as were the three diethylene glycol curves (differing by a maximum of 4.92×10^5). However, all of the glycerol chronoprint curves were significantly different from all of the diethylene glycol curves (differing by at least 1.29×10^6 sum-of-squared-differences). These results confirm that these substances can be easily distinguished by their enhanced chronoprints.

substances. Much like the original chronoprints, enhanced chronoprints are simple and inexpensive to obtain; we used a plastic microfluidic chip, a USB document camera, and a thermoelectric cooler to obtain ours, and other approaches could be used. Although we analyze only liquid samples in this work using enhanced chronoprints, this method is not limited to liquids. Solid samples such as pills, for example, could be dissolved in constant volumes of water, loaded into the thermometer chip with a bead, and analyzed for their authenticity using enhanced chronoprints.

With the original chronoprint technique, generally any sample that changes in appearance in response to a perturbation could be analyzed using chronoprints. The enhanced chronoprint method builds upon this method to broaden the types of samples that could be analyzed with the technique, to include samples that may not produce a visible change in response to a perturbation. By adding beads to transparent samples, and producing chronoprints of the beads falling through the different substances along a temperature gradient, I demonstrated that changes in the temperature-dependent material properties (viscosity and density) of a substances can be captured in the chronoprints, and can be used to identify and discriminate an even wider variety of samples with enhanced chronoprints; adding significant discrimination power to the chronoprint technique.

4.5 Acknowledgements

This work was supported by the National Science Foundation Division of Biological Infrastructure under award 1353974; the National Science Foundation Division of Computer and Communication Foundations under awards 1351115 and 1536026; the National Science

Foundation Division of Industrial Innovation and Partnership under award 1640757; and the National Science Foundation Division of Civil, Mechanical, and Manufacturing Innovation under award 1740052. Undergraduate student researchers Gabriel Colmenares and Luis Vidal Jacobo assisted with the experiments described in this work.

4.6 Supplementary Information for Enhanced chronoprints: Identifying samples by visualizing changes in particle interactions with sample viscosity

4.6.1 Design of the microfluidic thermometer chip used to engrave the chips used in this work on a CNC mill.

The design of the thermometer chip used in this work is shown in Figure 4.5.

4.6.2 Replicates of experiments in Figures 4.3 and 4.4 of the main text

Figures 4.6 through 4.9 provide replicates of the experiment in Figure 4.3 of the main text (comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water). Figures 4.10 through 4.13 provide replicates of the experiment in Figure 4.4 of the main text (comparing glycerol samples with diethylene glycol dilutions).

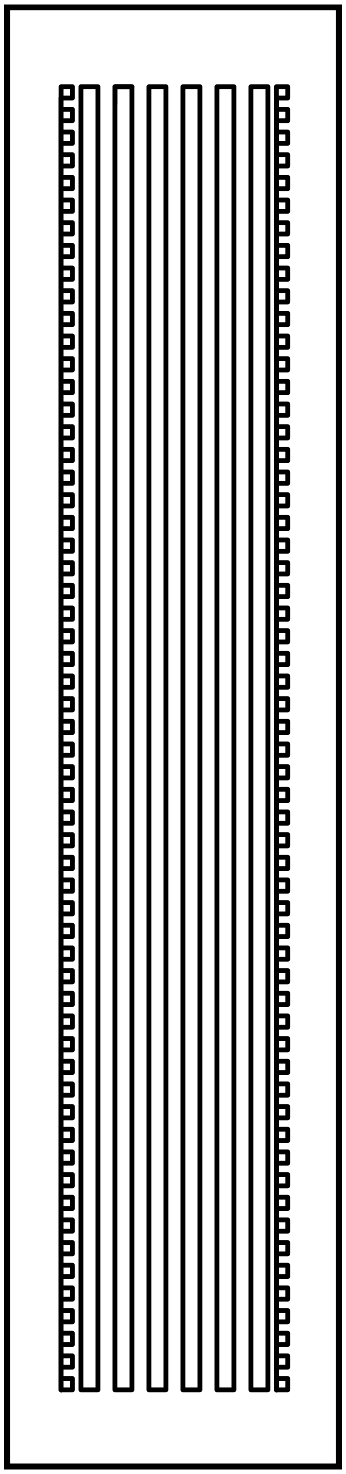


Figure 4.5: A drawing of the thermometer chip design used to fabricate chips in this work.

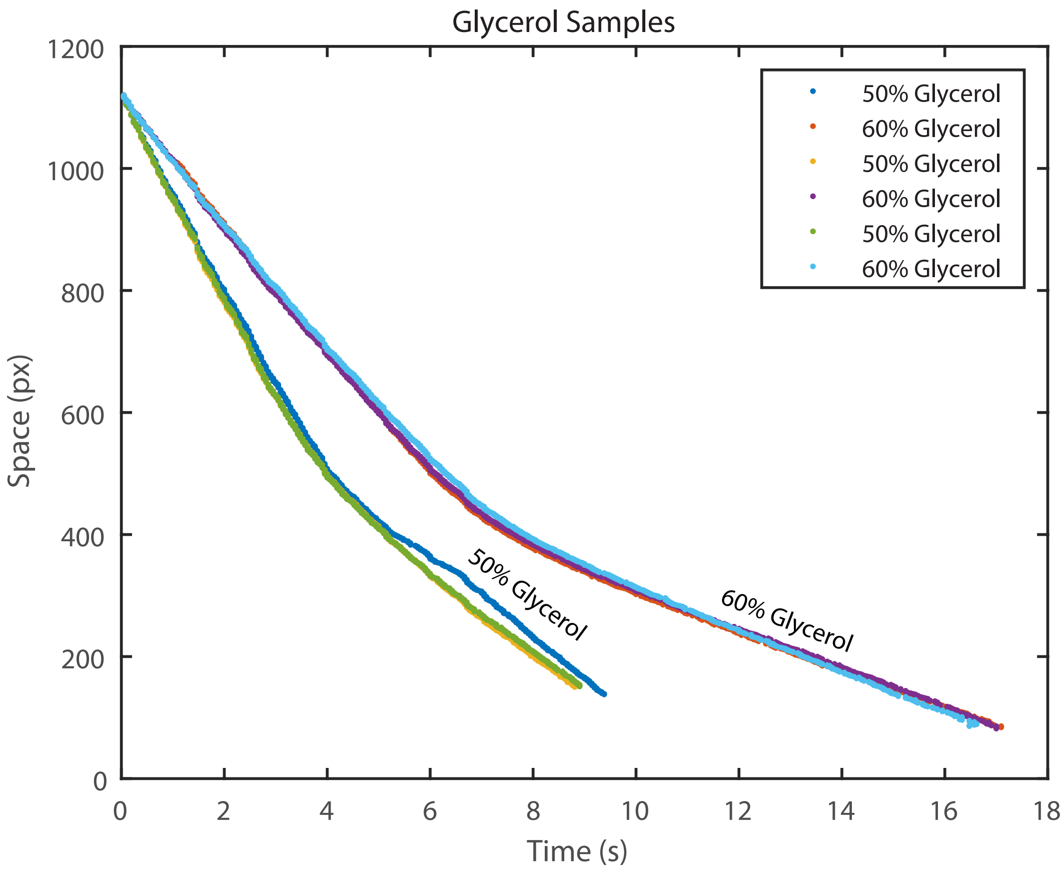


Figure 4.6: Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable. The bead falling through the first 50% glycerol dilution (dark blue curve) very briefly became stuck along the channel wall when falling, and this brief stop was refleced in its chronoprint and its resulting curve. The presence of a tiny air bubble can cause a falling bead to get stuck.

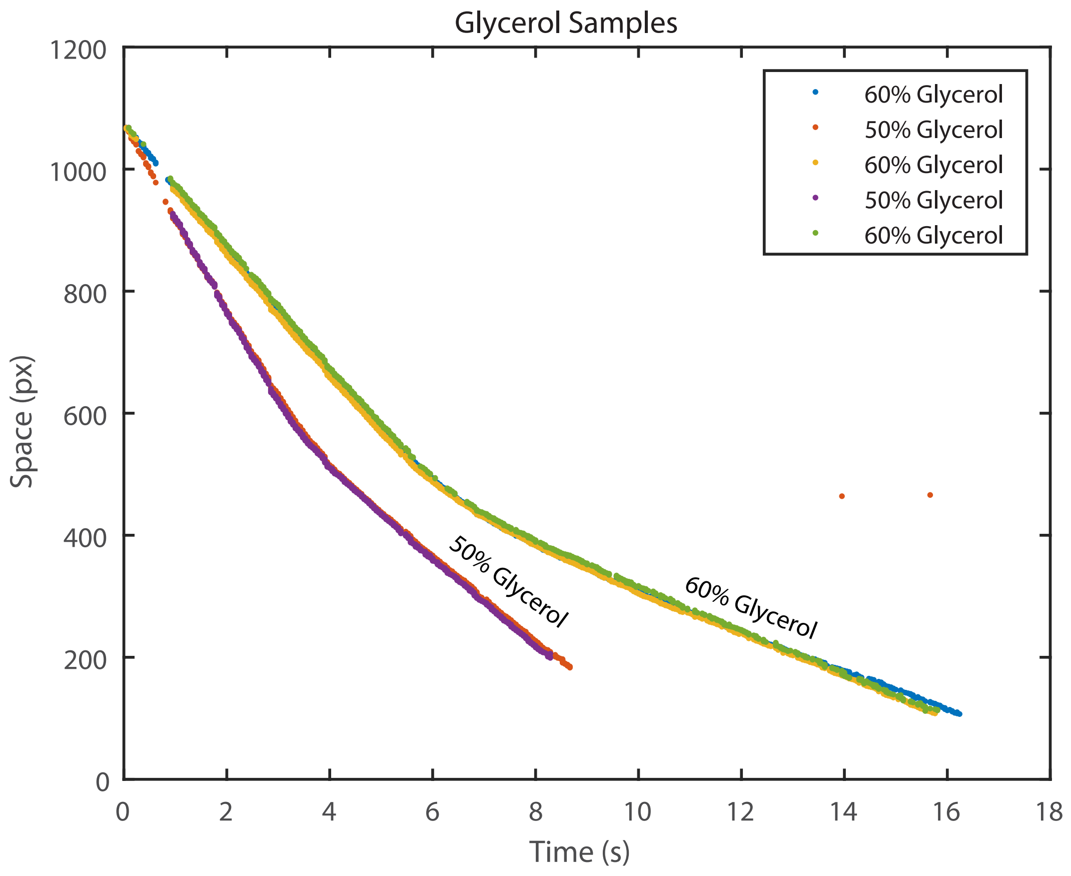


Figure 4.7: Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable.

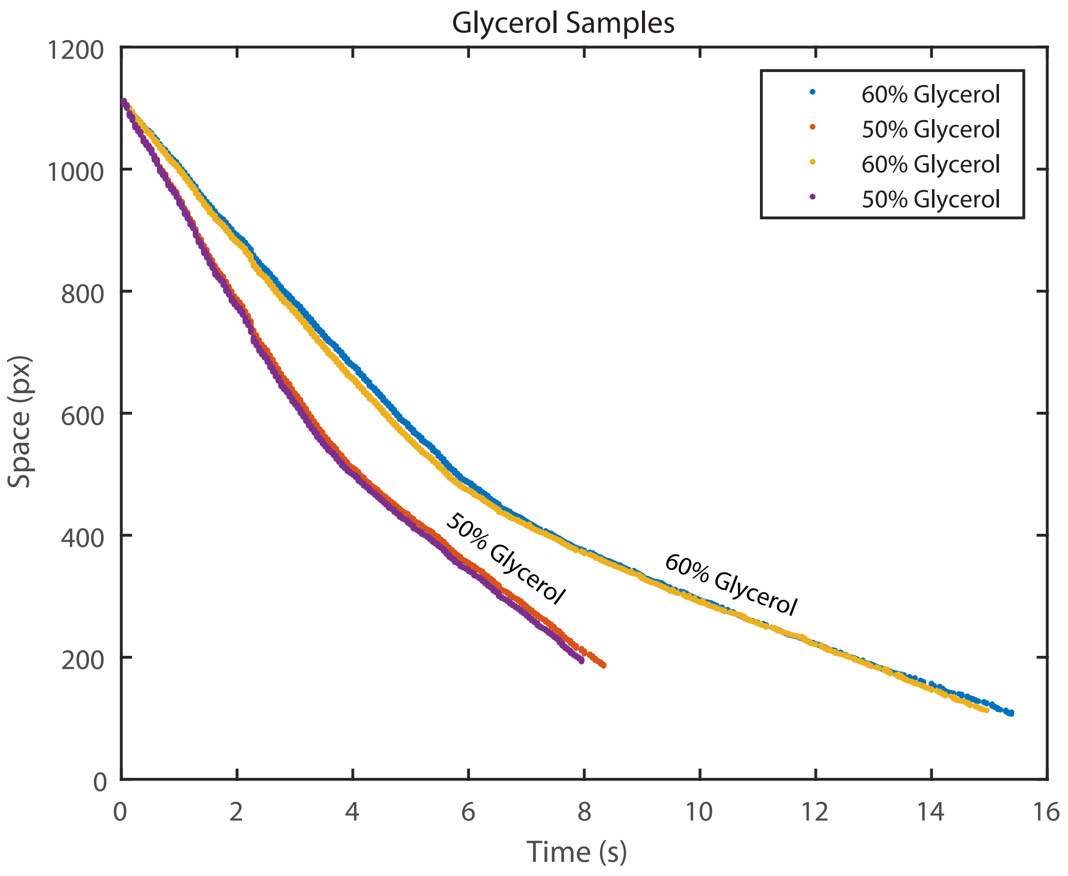


Figure 4.8: Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable.

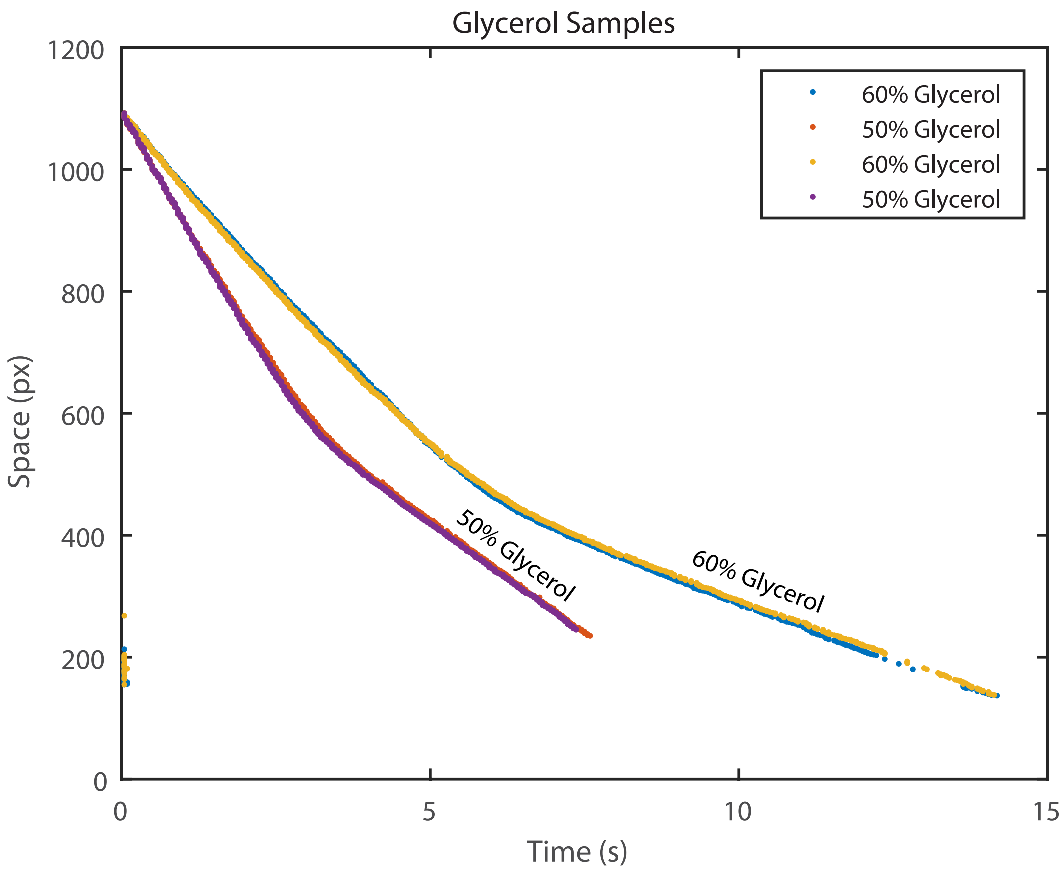


Figure 4.9: Replicate comparing 50% (v/v) glycerol dilutions in water with 60% (v/v) glycerol dilutions in water (replicate of the experiment shown in Figure 4.3 of the main text). All dilutions types are distinguishable.

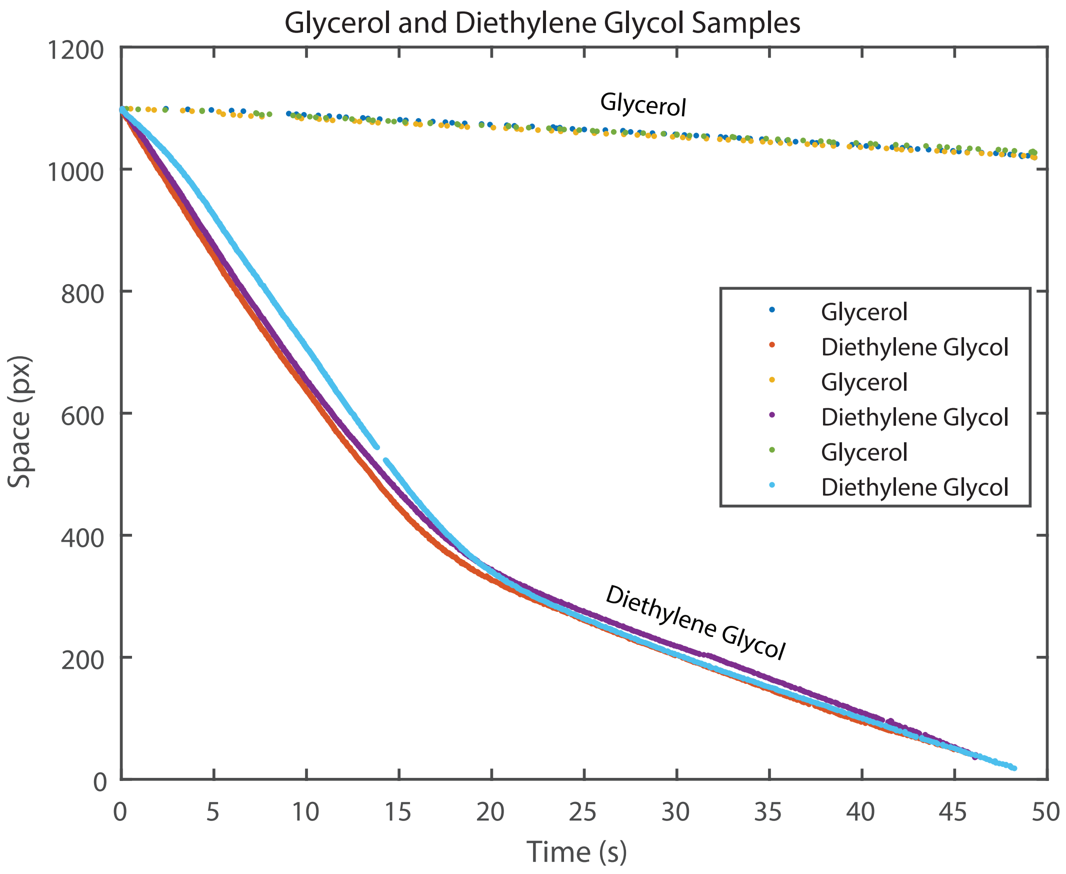


Figure 4.10: Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable.

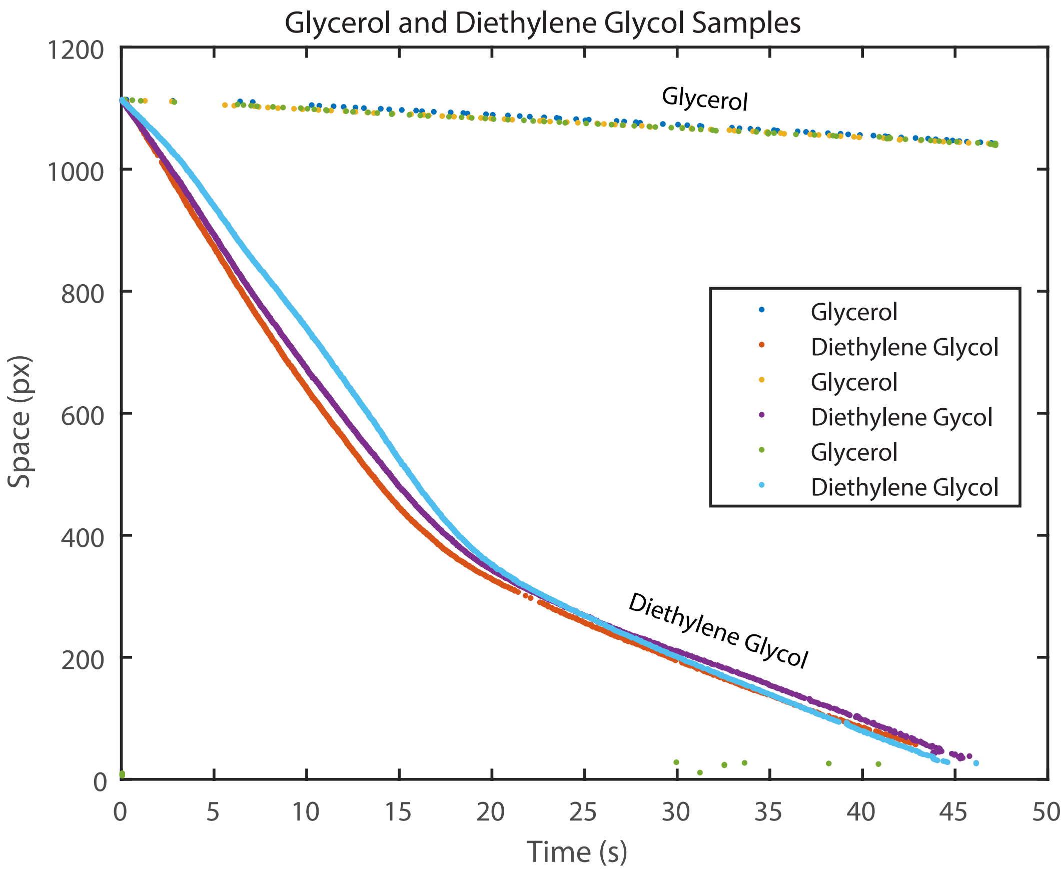


Figure 4.11: Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable.

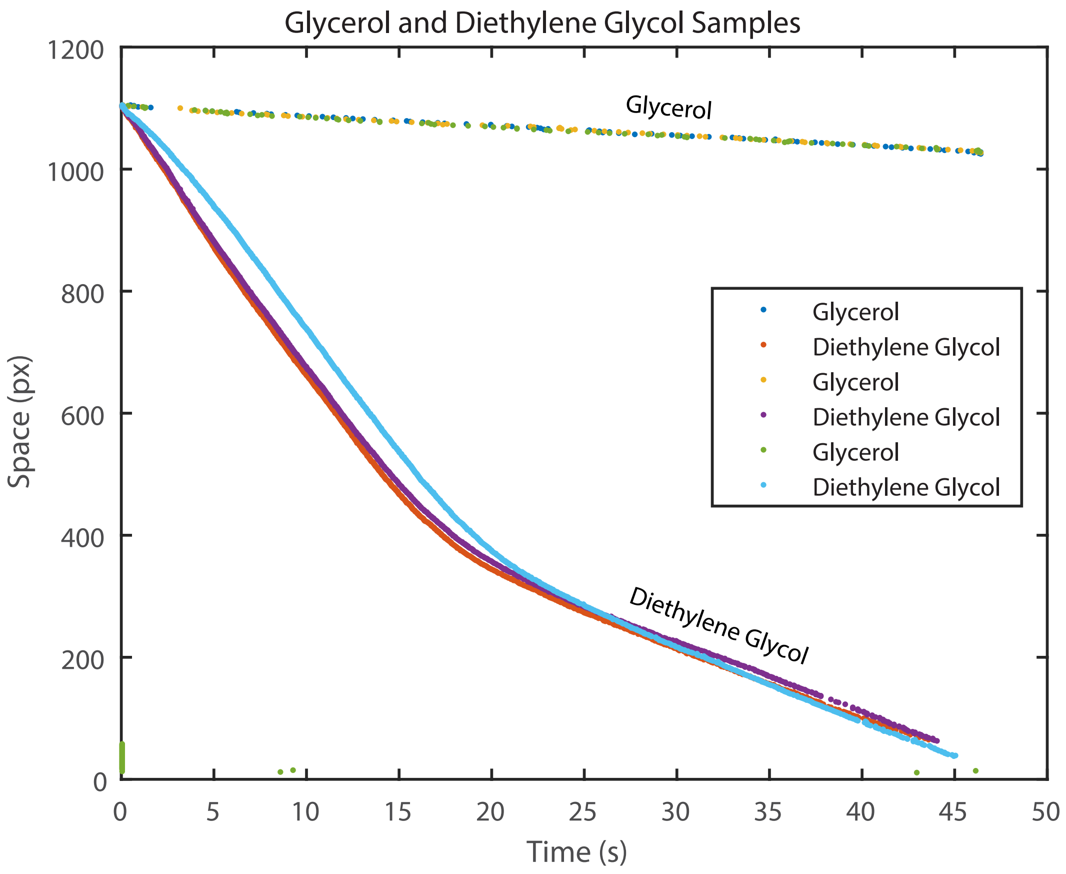


Figure 4.12: Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable.

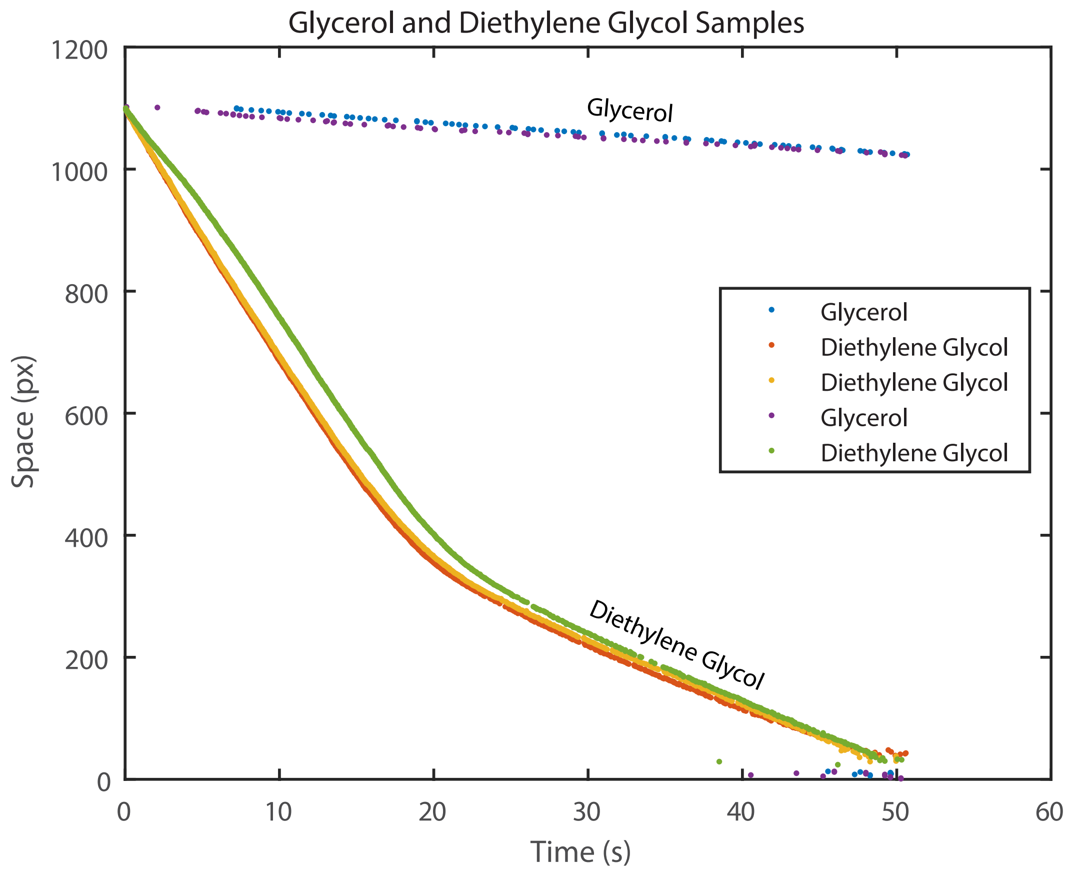


Figure 4.13: Replicate comparing glycerol samples with diethylene glycol samples (replicate of the experiment shown in Figure 4.4 of the main text). The two sample types are distinguishable.

4.6.3 `ChronoprintAnalysis.m`: MATLAB software for analyzing enhanced chronoprints, used to generate Figures 4.2C – 4.4 and Figures 4.6 – 4.13

The source code for the software used to analyze chronoprints is shown below.

```
% ChronoprintAnalysis.m
%
% This code converts each chrononoprint to curves and
    computes the sum of
% squared differences between the curves

clc; clear all; close all;

for num = 1:2 % number of samples to analyze

    % Reads chronoprint images
    Image1_{num} = imread(sprintf('reslice%01d.png',num));

    % Transform image for space to be on y-axis and time to
        be on x-axis
    Image2_{num} = imrotate(flipdim(Image1_{num},1),-90);

    % Convert image to monochrome
    Image3_{num} = rgb2gray(Image2_{num});
    Image3_{num}(:,1:23) = []; %remove shadow near inlets

    % Find minimum pixel values in each row and record the
        value and
    % indices
    [minValues{num}, t{num}] = min(Image3_{num},[],2);

    % Adjust time from number of frames to seconds and
        remove pixels where
    % the minimum pixel values are greater that the
        threshold value
    framerate = 21; %set frame rate
    threshold = 180; % set threshold pixel value

    t{num}= t{num}/framerate; %adjust time from frames to
        seconds
    t{num}(minValues{num}> threshold) = NaN; %remove pixels
         above threshold


    % Define y indicies and shift plots to the same
       starting location
    y{num} = length(t{num})-[1:length(t{num})];
    dy{num} = find(~isnan(t{num}),1);
    y{num} = y{num} + dy{num}; % shift plots to same
       starting location

end

%Compute sum of squared differences
diff = y{1} - y{2}; ssd = sum(diff(:).^2);

figure()
plot(t{1},y{1},'.',t{2},y{2},'.');
xlabel('Time␣(s)');
ylabel('Space␣(px)');
legend('Sample␣1','Sample␣2');
```

Chapter 5

Conclusions

5.1 Prospective pharmaceutical applications for the demonstrated techniques

In my projects I developed methods and tools to measure physical properties of subtsances, to identify authentic and adulterated medicines, as well as to distinguish pharmaceutical ingredients. Many of these techniques can be further explored for applications in the development of new pharmaceutical drugs and medicine products. The distribution coefficient, also known at the partition coefficient, is a critical measurement in chemical and pharmaceutical research. It is used to determine pharmacokinetic properties of drugs, and is commonly measured by shaking two immiscible solvents (an inorganic phase and an organic phase) with the compound of interest in a flask, and then determining the ratio of the concentrations of compound in the two phases at equilibrium. This "shake-flask method" is essentially a measure of the difference of compound solubility in the two phases. The shake-flask method often requires large volumes of solvent, as well as large amounts of the samples of interest, which can makes the method very costly and sometimes infeasible when using precious samples. In addition, the time for these systems to reach equilibrium so that the distribution coefficient can be measured takes hours and in some cases can even take days, making it a time-consuming process as well.

The ability to measure distribution coefficients within a microfluidic chip could drastically reduce the amount of sample and time required for this method; this is where the microfluidic thermometer, as well as the two aforementioned chronoprints methods can be of value. For example, several samples, each containing a very small amount of different variations of a pharmaceutical compound mixed with two immiscible fluids, could be loaded into the channels of the microfluidic thermometer chip, and the the method to produce solid-liquid interfaces could be used. Once the system reaches equilibrium, one could then extract the liquid portion and measure the amount of solute in that phase to determine the distribution coefficient. This could be repeated for all samples on the chip to identify compounds with the desired pharmokinetic properties. The chronoprint and enhanced chronoprint techniques could be used in a similar fashion, but instead to monitor changes in a pharmaceutical substance in response to a perturbation (e.g. pH gradient, temperature gradient, etc.); this could also help identify substances with the desired pharmaceutical properties. With their versatility, the microfluidic thermometer and chronoprint techniques have the potential to provide low-cost and efficient ways to study and develop pharmaceutical drugs and ingredients.

Bibliography

[1] James S. Cybulski, James Clements, and Manu Prakash. Foldscope: Origami-Based Paper Microscope. *PLoS ONE*, 9(6):e98781, June 2014.

[2] Heran C. Bhakta, Vamsi K. Choday, and William H. Grover. Musical Instruments as Sensors. *ACS Omega*, 3(9):11026–11032, 2018.

[3] Douglas A. Hill, Lindsey E. Anderson, Casey J. Hill, Afshin Mostaghim, Victor G. J. Rodgers, and William H. Grover. MECs: "Building Blocks" for Creating Biological and Chemical Instruments. *PLOS ONE*, 11(7):e0158706, July 2016.

[4] J. M. Pearce. Building Research Equipment with Free, Open-Source Hardware. *Science*, 337(6100):1303–1304, September 2012.

[5] C. Zhang, B. Wijnen, and J. M. Pearce. Open-Source 3-D Platform for Low-Cost Scientific Instrument Ecosystem. *Journal of Laboratory Automation*, 21(4):517–525, August 2016.

[6] Andre Maia Chagas, Lucia L. Prieto-Godino, Aristides B. Arrenberg, and Tom Baden. The 100 euro lab: A 3d-printable open-source platform for fluorescence microscopy, optogenetics, and accurate temperature control during behaviour of zebrafish, Drosophila, and Caenorhabditis elegans. *PLOS Biology*, 15(7):e2002702, July 2017.

[7] D. M. Heap, M. G. Herrmann, and C. T. Wittwer. PCR Amplification Using Electrolytic Resistance for Heating and Temperature Monitoring. *BioTechniques*, 29(5):1006–1012, November 2000. bibtex*[pmid=11084862].

[8] D.J. Sadler, R. Changrani, P. Roberts, Chia-Fu Chou, and F. Zenhausern. Thermal Management of BioMEMS: Temperature Control for Ceramic-Based PCR and DNA Detection Devices. *IEEE Transactions on Components and Packaging Technologies*, 26(2):309–316, June 2003.

[9] S. Qin, Y. Geng, D. E. Discher, and S. Yang. Temperature-Controlled Assembly and Release from Polymer Vesicles of Poly(Ethylene Oxide)-Block- Poly(N-Isopropylacrylamide). *Advanced Materials*, 18(21):2905–2909, November 2006.

[10] Ali Abou Hassan, Olivier Sandre, Valérie Cabuil, and Patrick Tabeling. Synthesis of Iron Oxide Nanoparticles in a Microfluidic Device: Preliminary Results in a Coaxial Flow Millichannel. *Chemical Communications*, (15):1783, 2008.

[11] T. Shinozaki, R. Deane, and F. M. Perkins. Infrared Tympanic Thermometer: Evaluation of a New Clinical Thermometer. *Critical Care Medicine*, 16(2):148–150, February 1988. bibtex*[pmid=3342626].

[12] Hasan Kocoglu, Sitki Goksu, Memet Isik, Zekeriya Akturk, and Yildirim A Bayazit. Infrared Tympanic Thermometer Can Accurately Measure the Body Temperature in Children in an Emergency Room Setting. *International Journal of Pediatric Otorhinolaryngology*, 65(1):39–43, August 2002.

[13] G. Edge and M. Morgan. The Genius Infrared Tympanic Thermometer.: An Evaluation for Clinical Use. *Anaesthesia*, 48(7):604–607, July 1993.

[14] Claudia Cianciulli and Hermann Wätzig. Infrared-Based Temperature Measurements in Capillary Electrophoresis. *ELECTROPHORESIS*, 32(12):1530–1536, June 2011.

[15] T. Matsukawa, M. Ozaki, T. Nishiyama, M. Imamura, and T. Kumazawa. Comparison of Infrared Thermometer with Thermocouple for Monitoring Skin Temperature. *Critical Care Medicine*, 28(2):532–536, February 2000. bibtex*[pmid=10708196].

[16] Claudiu A. Stan, Grégory F. Schneider, Sergey S. Shevkoplyas, Michinao Hashimoto, Mihai Ibanescu, Benjamin J. Wiley, and George M. Whitesides. A Microfluidic Apparatus for the Study of Ice Nucleation in Supercooled Water Drops. *Lab on a Chip*, 9(16):2293, 2009.

[17] Eric T. Lagally, Charles A. Emrich, and Richard A. Mathies. Fully Integrated PCR-Capillary Electrophoresis Microsystem for DNA Analysis. *Lab on a Chip*, 1(2):102, 2001.

[18] Runtao Zhong, Xiaoyan Pan, Lei Jiang, Zhongpeng Dai, Jianhua Qin, and Bingcheng Lin. Simply and Reliably Integrating Micro Heaters/Sensors in a Monolithic PCR-CE Microfluidic Genetic Analysis System. *ELECTROPHORESIS*, 30(8):1297–1305, April 2009.

[19] Alex I.K. Lao, Thomas M.H. Lee, I-Ming Hsing, and Nancy Y. Ip. Precise Temperature Control of Microfluidic Chamber for Gas and Liquid Phase Reactions. *Sensors and Actuators A: Physical*, 84(1-2):11–17, August 2000.

[20] Jinbo Wu, Wenbin Cao, Weijia Wen, Donald Choy Chang, and Ping Sheng. Polydimethylsiloxane Microfluidic Chip with Integrated Microheater and Thermal Sensor. *Biomicrofluidics*, 3(1):012005, 2009.

[21] David Erickson and Dongqing Li. Integrated Microfluidic Devices. *Analytica Chimica Acta*, 507(1):11–26, April 2004.

[22] Kohki Okabe, Noriko Inada, Chie Gota, Yoshie Harada, Takashi Funatsu, and Seiichi Uchiyama. Intracellular Temperature Mapping with a Fluorescent Polymeric Thermometer and Fluorescence Lifetime Imaging Microscopy. *Nature Communications*, 3:705, February 2012.

[23] Elizabeth Navarro Cerón, Dirk H. Ortgies, Blanca del Rosal, Fuqiang Ren, Antonio Benayas, Fiorenzo Vetrone, Dongling Ma, Francisco Sanz-Rodríguez, José García Solé, Daniel Jaque, and Emma Martín Rodríguez. Hybrid Nanostructures for High-Sensitivity Luminescence Nanothermometry in the Second Biological Window. *Advanced Materials*, 27(32):4781–4787, August 2015.

[24] G. Kucsko, P. C. Maurer, N. Y. Yao, M. Kubo, H. J. Noh, P. K. Lo, H. Park, and M. D. Lukin. Nanometre-Scale Thermometry in a Living Cell. *Nature*, 500(7460):54–58, July 2013.

[25] Rafael Piñol, Carlos D. S. Brites, Rodney Bustamante, Abelardo Martínez, Nuno J. O. Silva, José L. Murillo, Rafael Cases, Julian Carrey, Carlos Estepa, Cecilia Sosa, Fernando Palacio, Luís D. Carlos, and Angel Millán. Joining Time-Resolved Thermometry and Magnetic-Induced Heating in a Single Nanoparticle Unveils Intriguing Thermal Properties. *ACS Nano*, 9(3):3134–3142, March 2015.

[26] F Macritchie. The Adsorption of Proteins at the Solid/Liquid Interface. *Journal of Colloid and Interface Science*, 38(2):484–488, February 1972.

[27] Willem Norde. Adsorption of Proteins from Solution at the Solid-Liquid Interface. *Advances in Colloid and Interface Science*, 25:267–340, 1986.

[28] X. N. Xu. Long-Range Electrostatic Trapping of Single-Protein Molecules at a Liquid-Solid Interface. *Science*, 281(5383):1650–1653, September 1998.

[29] P. Somasundaran, Thomas W. Healy, and D. W. Fuerstenau. Surfactant Adsorption at the Solid—Liquid Interface—Dependence of Mechanism on Chain Length. *The Journal of Physical Chemistry*, 68(12):3562–3566, December 1964.

[30] Srinivas Manne and Hermann E. Gaub. Molecular Organization of Surfactants at Solid-Liquid Interfaces. *Science*, 270(5241):1480–1482, 1995.

[31] P. Somasundaran and S. Krishnakumar. Adsorption of Surfactants and Polymers at the Solid-Liquid Interface. *Colloids and Surfaces A: Physicochemical and Engineering Aspects*, 123-124:491–513, May 1997.

[32] T. Charitat, E. Bellet-Amalric, G. Fragneto, and F. Graner. Adsorbed and Free Lipid Bilayers at the Solid-Liquid Interface. *The European Physical Journal B*, 8(4):583–593, April 1999.

[33] S.R. Coriell and R.F. Sekerka. Lateral Solute Segregation during Undirectional Solidification of a Binary Alloy with a Curved Solid—Liquid Interface. *Journal of Crystal Growth*, 46(4):479–482, April 1979.

[34] M. J. Williamson, R. M. Tromp, P. M. Vereecken, R. Hull, and F. M. Ross. Dynamic Microscopy of Nanoscale Cluster Growth at the Solid–Liquid Interface. *Nature Materials*, 2(8):532–536, August 2003.

[35] Bice Fubini, Laura Mollo, and Elio Giamello. Free Radical Generation at the Solid/Liquid Interface in Iron Containing Minerals. *Free Radical Research*, 23(6):593–614, January 1995.

[36] William M. Haynes. *CRC Handbook of Chemistry and Physics, 97th Edition.* 2016.

[37] F. D Gunstone. *Vegetable Oils in Food Technology: Composition, Properties and Uses.* Wiley-Blackwell, Hoboken, 2nd ed edition, 2011.

[38] James R. Welty, Charles E. Wicks, Robert E. Wilson, and Gregory L. Rorrer. Fundamentals of Momentum, Heat, and Mass Transfer. page 208. John Wiley & Sons, Inc, 5th edition, 2008.

[39] Erwin A. Blackstone, Joseph P. Fuhr, and Steve Pociask. The Health and Economic Effects of Counterfeit Drugs. *American Health & Drug Benefits*, 7(4):216–224, June 2014. bibtex*[pmcid=PMC4105729;pmid=25126373].

[40] World Health Organization. WHO Global Surveillance and Monitoring System for Substandard and Falsified Medical Products. Technical report, World Health Organization, November 2017.

[41] Brittney A McKenzie and William H Grover. A Microfluidic Thermometer: Precise Temperature Measurements in Microliter- and Nanoliter-Scale Volumes. *PLoS One*, 12(12):e0189430, 2017. bibtex*[pmid=29284028].

[42] Rafael Gonzalez. *Digital Image Processing.* Prentice Hall, Upper Saddle River, N.J, 2008.

[43] Abraham Savitzky and Marcel JE Golay. Smoothing and Differentiation of Data by Simplified Least Squares Procedures. *Analytical Chemistry*, 36(8):1627–1639, 1964.

[44] Jean Steinier, Yves Termonia, and Jules Deltour. Smoothing and Differentiation of Data by Simplified Least Square Procedure. *Analytical Chemistry*, 44(11):1906–1909, 1972.

[45] Grocery Manufacturing Association and A.T. Kearney. Consumer Product Fraud: Deterrence and Detection. Technical report, Grocery Manufacturing Association, 2010.

[46] Karen Everstine, John Spink, and Shaun Kennedy. Economically Motivated Adulteration (EMA) of Food: Common Characteristics of EMA Incidents. *Journal of Food Protection*, 76(4):723–735, April 2013.

[47] Renée Johnson. Food Fraud and "Economically Motivated Adulteration" of Food and Food Ingredients. Technical report, Congressional Research Service, Library of Congress, January 2014.

[48] Eunyoung Hong, Sang Yoo Lee, Jae Yun Jeong, Jung Min Park, Byung Hee Kim, Kisung Kwon, and Hyang Sook Chun. Modern Analytical Methods for the Detection of Food Fraud and Adulteration by Food Category: Adulterated Food Categories and Their Analytical Methods. *Journal of the Science of Food and Agriculture*, 97(12):3877–3896, September 2017.

[49] Jeffrey C. Moore, John Spink, and Markus Lipp. Development and Application of a Database of Food Ingredient Fraud and Economically Motivated Adulteration from 1980 to 2010. *Journal of Food Science*, 77(4):R118–R126, April 2012.

[50] EN Frankel, RJ Mailer, CF Shoemaker, SC Wang, and JD Flynn. Tests Indicate That Imported "Extra Virgin" Olive Oil Often Fails International and USDA Standards. Technical report, Robert Mondavi Institute for Wine and Food Science; University of California, Davis Olive Center, July 2010.

[51] Abigail A. Weaver, Hannah Reiser, Toni Barstis, Michael Benvenuti, Debarati Ghosh, Michael Hunckler, Brittney Joy, Leah Koenig, Kellie Raddell, and Marya Lieberman. Paper Analytical Devices for Fast Field Screening of Beta Lactam Antibiotics and Antituberculosis Pharmaceuticals. *Analytical Chemistry*, 85(13):6453–6460, July 2013.

[52] Myra T. Koesdjojo, Yuanyuan Wu, Anukul Boonloed, Elizabeth M. Dunfield, and Vincent T. Remcho. Low-Cost, High-Speed Identification of Counterfeit Antimalarial Drugs on Paper. *Talanta*, 130:122–127, December 2014.

[53] Katherine E. Boehle, Cody S. Carrell, Joseph Caraway, and Charles S. Henry. Paper-Based Enzyme Competition Assay for Detecting Falsified Beta-Lactam Antibiotics. *ACS Sensors*, 3(7):1299–1307, July 2018.

[54] EMK Geiling and Paul R Cannon. Pathologic Effects of Elixir of Sulfanilamide (Diethylene Glycol) Poisoning: A Clinical and Experimental Correlation. *J. Am. Med. Assoc.*, 111(10):919–926, 1938.

[55] Barbara J. Martin. *Elixir: The American Tragedy of a Deadly Drug.* Barkerry Press, Lancaster, PA, 2014.

[56] Leo J Schep, Robin J Slaughter, Wayne A Temple, and D Michael G Beasley. Diethylene Glycol Poisoning. *Clin Toxicol*, 47(6):525–35, July 2009. bibtex*{'journal-full':'Clinical toxicology (Philadelphia, Pa.)',mesh:'Acidosis; Animals; Antidotes; Drug Contamination; Environmental Exposure; Ethanol; Ethylene Glycols; Humans; Kidney Diseases; Liver; Neurotoxicity Syndromes; Poisoning; Pyrazoles; Rats',pmid:'19586352',pst:'ppublish'}.

[57] Brittney A. McKenzie, Jessica Robles-Najar, Eric Duong, Philip Brisk, and William H. Grover. Chronoprints: Identifying Samples by Visualizing How They Change over Space and Time. *ACS Central Science*, 5(4):589–598, April 2019.

[58] Antonio Liga, Jonathan A. S. Morton, and Maïwenn Kersaudy-Kerhoas. Safe and cost-effective rapid-prototyping of multilayer PMMA microfluidic devices. *Microfluidics and Nanofluidics*, 20(12), December 2016.