COMPUTATIONAL APPROACHES TOWARDS PATTERN RECOGNITION AND MATCHING FOR SCIENTIFIC IMAGES

A Dissertation in
Information Sciences and Technology
by
Xinye Zheng

© 2020 Xinye Zheng

Submitted in Partial Fulfillment
of the Requirements
for the Degree of

Doctor of Philosophy

August 2020
The dissertation of Xinye Zheng was reviewed and approved by the following:

James Z. Wang  
Professor of Information Sciences and Technology  
Dissertation Co-Advisor, Co-Chair of Committee

Jia Li  
Professor of Statistics  
Dissertation Co-Advisor, Co-Chair of Committee

Saeed Abdullah  
Assistant Professor of Information Sciences and Technology

Anna Squicciarini  
Associate Professor of Information Sciences and Technology

Chris E. Forest  
Professor of Meteorology and Atmospheric Science

Mary Beth Rosson  
Professor of Information Sciences and Technology  
Graduate Program Director
Abstract

Pattern recognition and pattern matching are two important tasks in scientific image analysis. Researchers use patterns on image data to observe the phenomena that cannot be captured by human eyes, to model the activities of certain objects, and to evaluate the mathematical and physical assumptions. Tasks of pattern recognition and pattern matching on scientific images are complicated because they require domain knowledge to interpret the patterns. Compared with standard photography, scientific images usually depict a fuzzy state which is not self-explainable in most cases. In addition, the increasing size of image data in this big data era compounds the workload for researchers in processing and analyzing these data. This thesis is motivated to construct faster and more accurate computational models for scientific images using machine learning and optimization techniques.

Instead of seeking a unified processing tool for all scientific images, this thesis proposes computational approaches according to the actual requirements and challenges in two real-world problems. In the first part, this thesis employs pattern recognition to detect severe weather events early. While numerical weather prediction is widely used in the weather forecast, we still rely on people to interpret high-level visual clues from satellite and radar visualizations. A machine-learning-based approach is proposed to detect any “comma-shaped cloud”, which is a specific cloud distribution pattern strongly associated with storm formation. To train the model, we extracted shape and motion-sensitive features from delicately selected regions. The validated utility and accuracy of our method suggest a high potential for assisting meteorologists in weather forecasting.

In the second part, this thesis proposes problem-specific pattern matching frameworks on scientific image processing applications under the settings of optimal transport (OT). OT is a widely used optimization tool that can solve the minimal transportation cost between two probability distributions. The classical OT formula, however, asserts strict mass transportation constraints, which is not feasible for matching the complicated patterns in scientific images. Additionally, because the standard OT setup optimizes minimal spatial distance, it may not work on the special data structures of scientific images. Scientific images can match according to other features, including shape, scale, color, and contours. To overcome the limitations of standard OT, we use the geometry of data structure to regulate OT and design computational models according to the requirement of applications. For example, for the task to match cells on microscopy, a bipartite graph is proposed to regularize OT and combine cell shape and location information in the optimization function. For another task to match two color distributions in image editing, Gaussian mixture models (GMM) are used to capture the color semantics better. Instead
of forcing the mapping between two discrete color histograms, this thesis proposes a parametric framework to remove the artifact. This thesis concludes by discussing some future directions in developing more intelligent pattern recognition and matching systems for interpreting scientific images.
# Table of Contents

List of Figures viii

List of Tables xiii

Acknowledgments xiv

Chapter 1

Introduction 1

1.1 Evolution of Scientific Image Processing 1

1.2 Pattern Recognition and Pattern Matching Methods 3

1.3 Organization of the Thesis 5

Chapter 2

Pattern Recognition Application: Severe Weather Events Detection on Weather Images 7

2.1 Problem Introduction 7

2.2 Proposed Machine Learning Framework 11

2.2.1 Data Collection and Ground Truth 12

2.2.2 High-Cloud Segmentation 17

2.2.3 Correlation with Motion Prior 19

2.2.4 Region Proposals 20

2.2.5 Construction of Weak Classifiers 23

2.2.6 AdaBoost Detector 25

2.3 Experiments 26

2.3.1 Ablation Study 26

2.3.2 Detection Result 27

2.3.3 Storm Detection Ability 29

2.3.4 Case Studies 31

2.4 Summary and Discussion 33

Chapter 3

Pattern Matching with Optimal Transport: Background 35

3.1 Mathematical Setup of OT 37

3.2 Computational OT Models 39
List of Figures


2.1 An example of the satellite image with the comma-shaped cloud in the north hemisphere. This image is taken at 03:15:19 Coordinated Universal Time (UTC) on Dec 15, 2011 from the fourth channel of the GOES-N weather satellite.

2.2 *Left:* The pipeline of the comma-shaped cloud detection process, including high-cloud segmentation, region proposals, correlation with motion prior, constructions of weak classifiers, and the AdaBoost detector. *Right:* The detailed selection process for region proposals.

2.3 Proportions and geographical distributions of different severe weather events in the year 2011 and 2012. *Left:* Proportions of different categories of selected storm types. *Right:* State-wise geographical distributions of land-based storms.

2.4 IoU-Recall curve in the Region Proposal steps. The blue dot on the blue curve is our final IoU choice, with the corresponding recall of 0.91.

2.5 Cropped satellite images. (a) The original data. (b) Segmented high clouds with a single threshold. (c) Segmented high clouds with GMM. (d) Cross-correlation in [4]. (e) Correlation with motion prior.
2.6 Top: The correlation probability distribution of all sliding windows. Middle: Some segmented image examples. The last example image is the average image of the manually labeled regions in the training set. The correlation score $\gamma$ is defined in Equation (2.5), and the diagram is the normalized probability distribution of $\gamma$ of the training set. Bottom: Average comma-shaped clouds in different categories. 22

2.7 The accuracy distribution of weak classifiers with Segmented HOG feature and Motion Histogram feature. 24

2.8 Evaluation curves of our comma-shaped clouds detection method. Left: Missing rate curve with Detections. Right: Some reference cutoff values on the curve. 28

2.9 Comparison of the baseline methods. Left: Part of the Recall-Precision curve of the two baseline storm detection methods and our method. Right: The maximum recall rate they can reach. Here Intensity = Intensity Threshold Detection and Spatial-Intensity = Spatial-Intensity Threshold Detection. 30

2.10 (a-c) Three detection cases. Green frames: our detection windows; Blue frames: our labeled windows; Red dots: storms. Some images have blank in left-bottom because it is out of the satellite range. All the times are in UTC. 32

3.1 Schematic OT illustration, where the red points represent the source $X_1$ and blue points represent the target $X_2$. Left: probability distributions at dimension $d = 1$ and $d = 2$. Right: transportation plan $T$ represented by black dots, where the size of dots represents the mass in transportation. Image credit to [5]. 37

3.2 Impact of the entropy regularization term $\epsilon$ to the optimization function. Here $T^*$ is the OT solved from Equation (3.3) and $T^*_\epsilon$ is the optimal transportation plan solved from Equation (3.4). Image credit to [5]. 40

3.3 Example of irregularity in the classical OT formula. Image credit to [6]. 42

4.1 An example of cell alignment in two consecutive PSC microscopic images. Although cell B is closer to cell A, cell A actually moved and became cell C which is more similar to cell A in shape. 46
4.2 Point registration evolution with the increase of weight parameter $\lambda$. Left: Two pairs of point sets aligned by OT. For the two rectangles in the first row, the gravity center of the source shape has the same position in the image plane as the midpoint on the left side border of the target shape. In the second row, the gravity centers of the source shape and the target shape are located at the same position. Right: Visualization of point registration with the increase of $\lambda$.

4.3 The proposed cell tracking framework based on pixel-level point registration via OT. The top panel illustrates the four steps of cell tracking: cell segmentation, cell alignment by the RBOT algorithm, division detection, refined cell segmentation, and the determination of the entire tracking sequence for each cell. The bottom panel shows the four steps of the RBOT algorithm on two consecutive image frames: initialization, updating edge weights in each cell cluster, updating cell clusters, and computing the total number of cell clusters. The cells and their transportation weights are coded in the bipartite graph with each vertex corresponding to one cell. The RBOT algorithm stops when the number of cell clusters remains unchanged.

4.4 Visualization of cell clusters generated by RBOT. These cell clusters are binary cell masks of two consecutive frames of dataset PSC-Passage7 in [7]. Cells in the same color are from the same frame. Every subfigure corresponds to the result obtained up to a certain step. In each subfigure, the bipartite graph formed from the cell-level transportation plan is shown on the right, while the cell clusters are described below the figure. Top-Left: Two consecutive frames with segmented cells. Top-Right: The initialization of the cell-level transportation plan yielded from multiscale OT. Bottom-Left: The cell-level transportation plan after one round of recursion in RBOT. Bottom-Right: The cell-level transportation plan after two rounds of recursion in RBOT.

4.5 Pixel motion vectors at cell division. To illustrate the motion vectors, their directions are coded by the color hue, and their magnitudes are coded by saturation. A more saturated color means a higher magnitude. The motion vector is the difference between the coordinates of the registered pixels in the target and source images. Top-Left: Cells in division with pixels moving in opposite horizontal directions. Bottom-Left: Cells in division with pixels moving in opposite vertical directions. The motion vectors are computed based on the pixel-level transportation plan yielded from WGWD at $\lambda = 0.90$. Right: The color palette to code the motion vectors in [8].
4.6 Comparison of tracking results on dataset PhC-PSC-01 at time $T = 192 - 196$. From top to bottom, Row 1: microscopy frames. The red bounding box at $T = 193$ shows a cell division. Row 2: Cell tracking results with VT. Row 3: Cell tracking results with CT. Row 4: Cell tracking results with SCOTT at $\lambda = 0.1$. Row 5: Ground-truth labeling. One random color marks one cell in the time series.

4.7 Failure cases of cell tracking by SCOTT. Left: three cells are detected as two cells. Middle: two cells are separated with inaccurate boundary. Right: Part of a cell is incorrectly labeled as a portion of another cell nearby.

5.1 A simple example of Definitions 4 and 5. $\mathcal{M}_1$ is composed of purple, blue and green Gaussian components, and $\mathcal{M}_2$ is composed of yellow, orange and red. Left: affine map in Definition 4. Right: posterior probability map in Definition 5.

5.2 A simple example of Definitions 6 and 7. $\mathcal{M}_1$ is composed of blue and green Gaussian components, and $\mathcal{M}_2$ is composed of orange and red. Point-wise weights on the affine map $f_{1,1}$ are listed in the figures. Left: Elementary Point Transfer in Definition 6. Right: GRT in Definition 7.

5.3 Process of implementing GRT to color transfer tasks.

5.4 Comparison of our approaches using mean-shift estimation of GMM parameters (annotated as GRT+MS) and using BIC+EM to estimate GMM parameters (annotated as GRT+BIC) with other color transfer methods in [9–13]. 1st row: Original images with color histograms. 2nd-8th row, 1st-2nd column: Original images with color histograms after color transfer to the target. 2nd-8th row, 3rd-4th column: Target images with color histograms after color transfer to the originals.

5.5 Color transfer examples with GRT+MS (GRT+BIC has the similar results). Left: original and target images. Right: transferred images.

5.6 Point registration with different initialization methods. Left: source shapes and target shapes. Right: visualization of point registration by WD, GWD with WD initialization (WD-GWD), GWD with random alignment initialization (Random-GWD), and GWD with uniform alignment initialization (Uniform-GWD).
Visualization of cell clusters generated by RBOT algorithm for a dataset in Cell Tracking Challenge [14]. Instead of the original images, cartoons of segmented cells are shown. Blue colored cells are from the source image and red colored cells are from the target. Each row from left to right: cells labeled by ground truth and their corresponding alignment, initialization with multiscale OT, after one round of recursion, and after two rounds of recursion. Cell category from top to bottom: Rat mesenchymal stem cells on a flat polyacrylamide substrate, Pancreatic Stem Cells on a Polystyrene substrate, HeLa cells stably expressing H2b-GFP, and HeLa cells on a flat glass.
# List of Tables

2.1 Avg. Accuracy of weak classifiers for the segmented HOG and motion histogram features in different parameter settings ........................................ 24

2.2 Accuracy of stacked generalization methods on the cross-validation set ... 25

2.3 Accuracy of the AdaBoost detector on the cross-validation set with different parameters ......................................................................................... 25

2.4 Accuracy of the AdaBoost classifier on the cross-validation set with different features. Here HOG with high-cloud segmentation = Segmented HOG feature; Motion Hist. = Motion Histogram Feature ........................................ 27

4.1 Tracking accuracy for cell migration based on every two consecutive frames in datasets DIC-HeLa, PhC-U373, and PSC-Passage7. ............................... 63

4.2 Tracking accuracy for different cell activities based on every two consecutive frames in dataset PhC-PSC. ................................................................. 64

.1 Comparison of computation time and tracking accuracy at different scales of image downsizing. For every image sequence, three image sizes are experimented with, the largest size is that of the original data. All the experiments are conducted under the same setup as specified in the main paper ................................................................. 90
Acknowledgments

I would like to first express my heartfelt gratitude for all the support I get from my advisors, Professor James Z. Wang and Professor Jia Li. They are excellent mentors. They devoted unconditional help to formulating my research motivations, conducting experiments, and formatting my research work into the final thesis. During the journey of my Ph.D. study, their encouragement developed my optimism and persistence to research. In addition, their rigorous attitude towards science has asserted a profound influence on my future study and work.

I would like to thank my committee members, Professor Saeed Abdullah, Professor Anna Squicciarini, and Professor Chris E. Forest for providing additional insights and support for my thesis. My collaborators, Mr. Stephen Wistar and Mr. Michael Steinberg from AccuWeather Inc., and Professor Fuqing Zhang and Professor Yunji Zhang from the Department of Meteorology and Atmospheric Science at Penn State University inspired me with helpful discussions and informative guidance from a meteorology perspective. My gratitude extends to Professor Jose A. Piedra Fernández. He offered generous help when I visited Universidad de Almería, Spain in the summer of 2016. Also, I am grateful for the support and mentorship from Hongjie Wang, Dongzhe Yang, and Bob Hung when I worked at PNC Bank, Mathworks Inc., and Google LLC, respectively.

I am fortunate to have a close collaboration with my group members, Jianbo Ye, Yukun Chen, Baris Kandemir, Farshid Farhat, Yu Luo, Mohammad M. Kamani, Zhuomin Zhang, Chenyan Wu, Dolzodmaa Davaasuren, and Tongan Cai. They gave me invaluable partnership and commitment to completing our collaborative projects.

Last but not least, I would like to express my sincere gratitude to my family and friends for their continuous support and love for me. In particular, I would like to thank my parents and my best friend Haibo Zhang. I could not finish my studies without their help.

Part of this dissertation is based upon work supported by the National Science Foundation under Grant Nos. 1027854, 1521092, and 1548562. The primary computational infrastructures used were supported by the NSF under Grant Nos. ACI-0821527 (CyberStar) and ACI-1053575 (XSEDE), Amazon AWS Cloud Credits for Research Program, and the NVIDIA Corporation’s GPU Grant Program. Any opinions, findings, and conclusions or recommendations expressed in this dissertation do not necessarily reflect the views of the funding agencies.
Chapter 1 | Introduction

In today’s information age, the success of computational modeling strongly corresponds to the advancements in scientific images. On the one hand, the intensive application of artificial intelligence (AI) in modeling relies heavily on the image characteristics. For example, the training of deep learning methods uses massive image datasets. On the other hand, scientific images are the most definitive examples and output of the computational models. They aid researchers in reexamining the design, architecture, and engineering of their methodology. Furthermore, scientific image processing is the integration of computational design and scientific rationale. In this thesis, I will examine current approaches to scientific image processing and provide several computer-aided methods to address problem-specific scientific image processing problems.

1.1 Evolution of Scientific Image Processing

The term “scientific”, according to the definition in [15], refers to both the purpose of the image and the manner of generating those images. First, scientific images are not artistic. Instead, they primarily originate from the fields of science and technology. Second, these images go through special processing procedures in delicate sensors or instruments before the visualization. Examples of scientific images include radar reflectivity, infrared satellite visualization, microscope observation, and MRI scanning. Their fundamental roles in large-scale research areas encompass meteorology, microscopy, genomics, bacteriology, and medical science.

The history of scientific image processing can be traced back to the 18th century when researchers used hand drawing to record observations of nature. Both botany and zoology extensively used image illustrations to study scientific objects, which is referred
to as biological illustration in some literature [16,17]. An example of biological illustration is shown in the left of Figure 1.1. Starting in the 1970s, scientists began to use the first generation of computational software to assist scientific image recording and analysis [18]. With the emergence of image processing software like ImageJ, Adobe Photoshop, and CAD, computers continue to play an increasingly crucial role in analyzing scientific images.

![Figure 1.1: Evolution of scientific images. Left: Illustration of different fish from the book [1] published in 1754. Middle: First television picture from space taken by TIROS I satellite [2] in 1960. Right: The bridge between two gold nano atoms which are 2.3 angstroms apart taken from TEAM 0.5 microscope [3].](image)

The development of electronic sensors makes scientific images an essential resource for research. When the first satellite TIROS I launched to space, it could only capture low-resolution images at 1.5 miles in one band. As shown in the middle of Figure 1.1, the salt-and-pepper noise was significantly visible on satellite data. In comparison, the latest GOES-17 weather satellite maintains a higher resolution at 0.6 miles with 16 bands. Also, the frequency to capture one frame from space also improved from 90 minutes to 15 minutes. Electronic sensors are also critical for microscopic images. The right frame of Figure 1.1 illustrates the bridge between two gold nano atoms taken from the TEAM 0.5 microscope, which is the most powerful microscope with a resolution of half-angstrom (half a ten-billionth of a meter). More specifically, the higher resolution of scientific images makes their data more accurate, and a higher frequency also makes the data size much more abundant. The improved quality and increasing number of such images have created a high demand for automatic image processing methods for these data amongst scientists.

The ability of the scientific image processing to bridge the gap between theoretical assumptions and reality invisible to the human eye is its most appealing feature. Additionally, screens, scans, and investigations help register abstract scientific theory onto
the visible signals on the images. As a result, researchers can validate their assumptions by checking the signal patterns.

1.2 Pattern Recognition and Pattern Matching Methods

Pattern recognition and matching are two main methods for pattern identification. Pattern recognition refers to the automatic discovery of regularities in data through computational methods [19]. Generally, modern pattern recognition requires researchers to label datasets or have high-level expertise on the targeting representations. The task is to apply the information gained from theoretical understanding or statistical information extracted from patterns to locate and classify target patterns on the scientific images. Pattern recognition is widely used for scientific model verification. Recently, machine learning and statistical models are two main categories of methods applying to this task. However, there are two prerequisites for applying machine learning: (1) researchers should have sufficient knowledge to classify the pattern. (2) the data size should be enough for analyzing the variations between different samples.

Pattern matching refers to the registration of the target pattern onto a particular template or a given sample, classified into template matching (local pattern) [20] or image registration (global pattern) [21]. While pattern recognition mainly focuses on detecting an object, pattern matching does so to compare and evaluate the difference between an image with the given template or to integrate two observations of the same object. Pattern matching has two main usages. The first usage is to help identify certain regularities in data, which is similar to pattern recognition. When researchers do not have a large labeled data, or they cannot classify certain pattern based on current knowledge, the similarity between the sample and the known template is an evidence for pattern identification. The second usage is the registration between multiple images. Images taken from different angles or different sensors will have to register to one high-dimensional model. The registration process has to rely on pattern matching techniques.

There are two main differences between pattern identification on scientific images and standard photography. The first is the accuracy of signal interpretation from sensors to the visualization. Some scientific images are translations of models that provide visual access to the studying apparatus [22], which raises concerns regarding the accuracy of such interpretation on these images [23]. The visualization, color, spatiality, and intensity of scientific images significantly differ from those in standard photography. The former follows the representations designed by the sensors or algorithms to process
these signals. Usually, these scientific images are gray-scaled, and the adjusted colors accommodate the visible range, which is comfortable from perception perspective. While some researchers express concern that this artificial data characteristic will lead to the so-called “performing knowledge” [24], which is a manually designed result, the majority of researchers believe that the artificiality of scientific images enables a better assessment of the methodology and makes the results more interpretable [25].

The second difference between scientific images and standard photography is the form of noise on the images. There are various reasons for this noise, including errors from the electronic instrument and the internal noise from the fussy state of the chaotic system. For example, a German physician named Franz Josef Gießibl and his team published their analysis of the internal structure of the atom using the observation from a microscope image [26,27]. Later on, there was a debate raised by Basel claiming that the observed cloud-form structure could either be the noise in the microscopy or the underlying structure of atom. This long-lasting debate on this observation added some uncertainty to the scientific findings. With the development of sensor technology, researchers now use quantitative analysis to evaluate the effect of instrument errors [28,29]. However, merely removing the noise that originated from the complicated fuzzy system may affect the target object on the image. Such noise is prevalent in numerous scientific observations, such as in long-term weather change or atom movements under the microscope. In some literature, this concept was named as the “deterministic chaos” in the system [30].

Additionally, scientific images are not generally self-explanatory [31], meaning non-professionals are unable to identify patterns on their own. For example, radiologists diagnose disease using medical images; weather forecasters produce forecasting results on radar reflectivity, and biologists analyze bacteria on a microscope observation. Additionally, experts need to consult other non-image data sources to produce the final assessment. This interpretative requirement remains a challenge when designing such automatic computational algorithms.

The problems associated with pattern recognition are complex, and the variance across the scientific images is significant. Consequently, it is not feasible to design an applicable algorithm for all these images. The application of general image processing methods in the prepossessing stage includes resolution restoring [32,33], illustration correction [34,35], and denoising [36,37]. On top of that, specially designed pattern recognition and matching algorithms are applied on different scientific images. In the context of weather images, the Gaussian process modeled phases of different scatters on radar images [38], where each component represented different geometric structures on
the earth. In comparison, watershed transformation is more widely used on microscopy to model different cell components [39].

Recently, researchers have begun to use AI to perform an increasingly vital role in computational methods. One of the well-known machine learning methods is U-Net [40], a widely used cell-segmentation structure utilizing a convolutional neural network (CNN). At the ISBI 2015 [41], U-Net won the Cell Tracking Challenge in the two most challenging transmitted light microscopy categories known as the Phase contrast and DIC microscopy, by a large margin. Medical image registration extensively uses Generative Adversarial Networks (GANs) to provide additional regularization to the model or to help transfer the multi-model to the monomodel [42,43]. While classification models trained on deep neural networks show much effectiveness on some tasks, the interpretability of models and errors remain far from perfect [44]. Additionally, the construction of some scientific image processing methods on small datasets with problem-specific requirements makes it difficult to apply deep learning methods [45].

Current pattern recognition and matching algorithms on scientific images still face some challenges. While we are unable to cover all of them in detail, this thesis introduces two important issues and offers solutions on specific scientific images. The first difficulty is the lack of high-level pattern interpretation. Classical scientific image classification procedure includes the following steps: image preprocessing, segmentation, feature extraction, and classification [46]. However, features directly extracted from the images are mostly low-level features, including color, texture, and edges. While these features are intuitive for the construction, their performance is not as reliable as the deep-learning-based methods are [44]. Deep learning methods can extract high-level features via the convolutional structures but are hard to interpret them from a scientific perspective. The second challenge is that the measurement of pattern matching algorithms is not unified.

1.3 Organization of the Thesis

This dissertation is organized as follows. In Chapter 2, we introduce a pattern recognition algorithm on satellite images to improve severe weather forecasting. To be specific, this data-driven method targets early visual clues on satellite images. Beginning in Chapter 3, we will introduce the basics of OT, a powerful optimization tool that is widely used in pattern matching on images. OT aims to solve the distribution matching problem within its mathematical setup. However, the strict mass transportation constraint of OT sometimes makes the mapping unfeasible in the real-life application. To handle
this artifact, we propose the problem-specific OT frameworks for two pattern matching applications. One is the cell-tracking algorithm on microscope observations in Chapter 4, another is the GMM-based distribution transfer on color histograms in Chapter 5. We will conclude and list some future directions for scientific image processing in the final Chapter 6.
Chapter 2  
Pattern Recognition Application:  
Severe Weather Events Detection on Weather Images

Pattern recognition is a key technique widely used in severe weather forecasting. Meteorologists have summarized some early visual patterns according to rich experience, including comma-shaped cloud [47], bow echo [48], velocity couplet [49], the Debris ball [50] and so on. These early features are important evidence for weather forecasters. In this chapter, we use the comma-shaped cloud as an example to introduce how we construct an automatic severe weather detection system with the help of computer vision and machine learning techniques.

2.1 Problem Introduction

Severe weather events such as thunderstorms cause significant losses in property and lives. Many countries and regions suffer from storms regularly, leading to a global issue. For example, severe storms kill over 20 people per year in the U.S. [51]. The U.S. government has invested more than 0.5 billion dollars [52] on research to detect and forecast storms, and it has invested billions for modern weather satellite equipment with high-definition cameras.

The fast pace of developing computing power and increasingly higher definition satellite images necessitate a re-examination of conventional efforts regarding storm forecast, such as bare eye interpretation of satellite images [53]. Bare eye image interpretation by experts requires domain knowledge of cloud involvements and, for a variety of reasons, may result in omissions or delays of extreme weather forecasting. Moreover, the enhancements from the latest satellites which deliver images in real-time at a very high resolution demand tight processing speed. These challenges encourage us to explore how applying modern learning schema on forecasting storms can aid meteorologists in interpreting visual clues of storms from satellite images.

Satellite images with the cyclone formation in the mid-latitude area show clear visual patterns, known as the comma-shaped cloud pattern [47]. This typical cloud distribution pattern is strongly associated with mid-latitude cyclonic storm systems. Figure 2.1 shows an example of comma-shaped clouds in the Northern Hemisphere, where the cloud shield has the appearance of a comma. Its “tail” is formed with the warm conveyor belt extending to the east, and “head” within the range of the cold conveyor belt. The dry-tongue jet forms a cloudless zone between the comma head and the comma tail. The comma-shaped clouds also appear in the Southern Hemisphere, but they form in the opposite direction (i.e., an upside-down comma shape). This cloud pattern gets its name because the stream is oriented from the dry upper troposphere and has not achieved saturation before ascending over the low-pressure center. The comma-shaped cloud feature is strongly associated with many types of extratropical cyclones, including hail, thunderstorm, high winds, blizzards, and low-pressure systems. Consequently, we can observe severe events like ice, rain, snow, and thunderstorms around this visual feature [54].

To capture the comma-shaped cloud pattern accurately, meteorologists have to read different weather data and many satellite images simultaneously, leading to inaccurate or untimely detection of suspected visual signals. Such manual procedures prevent meteorologists from leveraging all available weather data, which increasingly are visual in form and have high resolution. Negligence in the manual interpretation of weather data can lead to serious consequences. Automating this process, through creating intelligent computer-aided tools, can potentially benefit the analysis of historical data and make meteorologists’ forecasting efforts less intensive and more timely. This philosophy is persuasive in the computer vision and multimedia community, where images in modern image retrieval and annotation systems are indexed by not only metadata, such as author and timestamp, but also semantic annotations and contextual relatedness based on the
While all comma-shaped clouds resemble the shape of a comma mark to some extent, the appearance and size of one such cloud can be very different from those of another. This makes conventional object detection or pattern matching techniques developed in computer vision inappropriate because they often assume a well-defined object shape (e.g. a face) or pattern (e.g. the skin texture of a zebra).

Cloud is the main object in satellite images. Hence, researchers developed multiple image processing techniques to capture cloud patches better. These methods can be categorized into cloud segmentation and motion estimation. Cloud segmentation is an important method for detecting storm cells. Lakshmanan et al. [57] proposed a hierarchical cloud-texture segmentation method for satellite image. Later, they improved the method by applying the watershed transform to the segmentation and using pixel intensity thresholding to identify storms [58]. However, brightness temperature in a single satellite image is easily affected by lighting conditions, geographical location, and satellite imager quality, which is not adequately considered in the thresholding-based methods. Therefore, we consider these spatial and temporal factors and segment the high cloud part based on the Gaussian Mixture Model (GMM).

Cloud motion estimation is also an essential method for storm detection, and a common approach estimates cloud movements through cross-correlation over adjacent images. Some earlier work [4] and [59] applied the cross-correlation method to derive
the motion vectors from cloud textures, which was later extended to multi-channel satellite images in [60]. The cross-correlation method could partly characterize the airflow dynamics of the atmosphere and provide meaningful speed and direction information on large areas [61]. After being introduced in the radar reflectivity images, the method was applied in the automatic cloud-tracking systems using satellite images. A later work [62] implemented the cross-correlation in predicting and tracking the Mesoscale Convective Systems (MCS, a type of storm). Their motion vectors were computed by aggregating nearby pixels at two consecutive frames; thus, they are subject to spatially smoothed effects and miss fine-grained details. Inspired by the ideas of motion interpretation, we define a novel correlation aiming to recognize cloud motion patterns in a longer period. The combination of motion and shape features demonstrates high classification accuracy on our manually labeled dataset.

Researchers have applied pattern recognition techniques to interpret storm formulation and movement extensively. Before the satellite data reached a high resolution, earlier works constructed storm formulation models based on 2D radar reflectivity images in the 1970s. The primary techniques can be categorized into cross correlation [63] and centroid tracking [64] methods. According to the analysis, cross-correlation based methods are more capable of accurate storm speed estimation, while centroid-tracking-based methods are better at tracking isolated storm cells.

Taking advantage of these two ideas, Dixon and Wiener developed the renowned centroid-based storm nowcasting algorithm, named Thunderstorm Identification, Tracking, Analysis and Nowcasting (TITAN) [65]. This method consists of two steps: identifying the isolated storm cells and forecasting possible centroid locations. Compared with former methods, TITAN can model and track some storm merging and splitting cases. This method, however, can have large errors if the cloud shape changes quickly [66]. Some later works attempted to model the storm identification process mathematically. For instance, [67] and [68] used statistical features of the radar reflection to classify regions into storm or storm-less classes.

Recently, Kamani et al. proposed a severe storm detection method by matching the skeleton feature of bow echoes (i.e., visual radar patterns associated with storms) on radar images in [69], with an improvement presented in [70]. The idea is inspiring, but radar reflectivity images have some weaknesses in extreme weather precipitation [71]. First, the distribution of radar stations in the contiguous United States (CONUS) is uneven. The quality of ground-based radar reflectivity data is affected by the distance to the closest radar station to some extent. Second, detections of marine events are
limited because there are no ground stations in the oceans to collect reflectivity signals. Finally, severe weather conditions would affect the accuracy of radar. Since our focus is on severe weather event detection, radar information may not provide enough timeliness and accuracy for detection purposes.

Compared with the weather radar, multi-channel geosynchronous satellites have larger spatial coverages and thus are capable of providing more global information to the meteorologists. Take the infrared spectral channel in the satellite imager as an example: the brightness of a pixel reflects the temperature and the height of the cloud top position [72], which in turn provides the physical condition of the cloud patch at a given time. To find more information about the storm, researchers have applied many pattern recognition methods to satellite data interpretation, like combining multiple channels of image information from the weather satellite [60] and combining images from multiple satellites [73]. Image analysis methods, including cloud patch segmentation and background extraction [57] [74], cyclone identification [75] [76], cloud motion estimation [77], and vortex extraction [78] [79], have also been incorporated in severe weather forecasting from satellite data. However, these approaches lack an attention mechanism that can focus on areas most likely to have major destructive weather conditions. Most of these methods do not consider high-level visual patterns (i.e. larger patterns spatially) to describe the severe weather condition. Instead, they represent extreme weather phenomena by relatively low-level image features.

2.2 Proposed Machine Learning Framework

We propose a machine-learning and pattern-recognition-based approach to detect comma-shaped clouds from satellite images. The comma-shaped cloud patterns, which have been manually searched and indexed by meteorologists, can be automatically detected by computerized systems using our proposed approach. We leverage the large satellite image dataset in the historical archive to train the model and demonstrate the effectiveness of our method in a manually annotated comma-shaped cloud dataset. Moreover, we demonstrate how this method can help meteorologists to forecast storms using the strong connection of comma-shaped cloud and storm formation.

The key visual cues that human experts use in distinguishing comma-shaped clouds are shape and motion. During the formulation of a cyclone, the “head” of the comma-shaped cloud, which is the northwest part of the cloud shield, has a strong rotation feature. The dense cloud patch forms the shape of a comma, which distinguishes the
cloud patch from other clouds. To emulate meteorologists, we propose two novel features that consider both shape and motion of the cloud patches, namely, *Segmented HOG* and *Motion Correlation Histogram*, respectively. We detail our proposals in Section 2.2.2 and Section 2.2.3.

Our work makes two main contributions. First, we propose novel shape and motion features of the cloud using computer vision techniques. These features enable computers to recognize the comma-shaped cloud from satellite images. Second, we develop an automatic scheme to detect the comma-shaped cloud on the satellite images. Because the comma-shaped cloud is a visual indicator of severe weather events, our scheme can help meteorologists forecast such events.

Figure 2.2 shows our proposed comma-based cloud detection pipeline framework. We first segment the cloud from the background in Section 2.2.2, and then we extract shape and motion features of clouds in Section 2.2.3. Well-designed region proposals in Section 2.2.4 shrink the searching range on satellite images. The features on our extracted region proposals are fed into weak classifiers in Section 2.2.5 and then we ensemble these weak classifiers as our comma-shaped cloud detector in Section 2.2.6. We now detail the technical setups in the following.

### 2.2.1 Data Collection and Ground Truth

We first introduce the dataset structure in this problem to motivate further feature selection. Our dataset consists of the GOES-M weather satellite images for the year 2008 and the GOES-N weather satellite images for the years 2011 and 2012. We select these three years because the U.S. experienced more severe thunderstorm activities than it does in a typical year. GOES-M and GOES-N weather satellites are in the geosynchronous orbit of Earth and provide continuous monitoring for intensive data analysis. Among the five channels of the satellite imager, we adopt the fourth channel, because it is infrared among the wavelength range of (10.20 - 11.20 µm), and thus can capture objects of meteorological interest including clouds and sea surface [80]. The channel is at the resolution of 2.5 miles and the satellite takes pictures of the northern hemisphere at the 15th minute and the 45th minute of each hour. We use these satellite frames of CONUS at 20°-50° N, 60°-120° W. Each satellite image has 1,024×2,048 pixels, and a gray-scale intensity that positively correlates with the infrared temperature. After the raw data are converted into the image data in accordance with the information in [81], each image pixel represents a specific geospatial location.
Figure 2.2: Left: The pipeline of the comma-shaped cloud detection process, including high-cloud segmentation, region proposals, correlation with motion prior, constructions of weak classifiers, and the AdaBoost detector. Right: The detailed selection process for region proposals.
The labeled data of this dataset consist of two parts, (1) comma-shaped clouds identified with the help of meteorologists from AccuWeather Inc., and (2) an archive of storm events for these three years in the U.S. [82].

To create the first part of the annotated data, we manually label comma-shaped clouds by using tight squared bounding boxes around each such cloud. If a comma-shaped cloud moves out of the range, we ensure that the head and tail of the comma are in the middle part of the square. The labeled comma-shaped clouds have different visual appearances, and their coverage varies from a width of 70 miles to 1,000 miles. Automatic detection of them is thus nontrivial. The labeled dataset includes a total of 10,892 comma-shaped cloud frames in 9,321 images for the three years 2008, 2011, and 2012. Most of them follow the earlier description of comma-shaped clouds, with the visible rotating head part, main body heading from southwest to northeast, and the dark dry slot area between them.

The second part of the labeled data consists of storm observations with detailed information, including time, location, range, and type. Each storm is represented by its latitude and longitude in the record. We ignore the range differences between storms because the range is relatively small (< 5 miles) compared with our bounding boxes (70 ~ 1000 miles). Every event recorded in the database had enough severity to cause death, injuries, damage, and disruption to commerce. The total estimated damage from storm events for the years 2011 and 2012 surpassed two billion dollars [83]. From the database, we chose eight types of severe weather records that are known to correlate strongly with the comma-shaped clouds and happen most frequently among all types of

Tornadoes are included in the Thunderstorm Wind type.
events. The distribution of these eight types of severe weather events is shown in the left part of Fig. 2.3. Among those eight types of severe weather events, thunderstorm winds, hail, and heavy rain happen most frequently (∼ 93% of the total events). The state-wise geographical distributions of some types of storm events are in the right half of Fig. 2.3. Because marine-based events do not have associated state information, we only visualize the geographical distributions for land-based storm events. With the exception of heavy rains, these severe weather events happen more frequently in East CONUS.

In our experiments, we include only storms that lasted for more than 30 minutes because they overlapped with at least one satellite image in the dataset. Consequently, we have 5,412 severe storm records for the years 2011 and 2012 in the CONUS area for testing purposes, and their existences vary from 30 minutes to 28 days.

To partition the data into positive and negative samples, we use the widely-used “sliding windows” in [84] as the first-step detection. Sliding windows with an image pyramid help us capture the comma-shaped clouds at various scales and locations. Because most comma-shaped clouds are in the high sky, we run our sliding windows on the segmented cloud images. We set 21 dense $L \times L$ sliding windows, where $L \in \{128, 128 \cdot 8^{1/20}, \ldots, 128 \cdot 8^{19/20}, 1024\}$. For each sliding window size $L$, the movement pace of the sliding window is $\lfloor L/8 \rfloor$, where $\lfloor \cdot \rfloor$ is the floor function. Under that setting, each satellite image has more than $10^4$ sliding windows, which is enough to cover the comma-shaped clouds in different scales.

Before we apply machine learning techniques, it is important to define whether a given bounding box is positive or negative. Here we use the Intersection over Union metric (IoU) [85] to define the positive and negative samples, which is also a common criterion in object detection. We set bounding boxes with IoU greater than a value to be the positive examples, and those with IoU = 0 to be the negative samples.

A suitable IoU threshold should strike a balance between high recall and high accuracy of the selected comma-shaped clouds. Several factors prevent us from achieving a perfect recall rate. First, we only choose limited sizes of sliding windows with limited strides. Second, some of the satellite images are (partially) corrupted and unsuitable for a data-driven approach. Third, some cloud patches are in a lower altitude, hence they are removed in the high-cloud segmentation process in Section 2.2.2. Fourth, we design simple classifiers to filter out most sliding windows without comma-shaped clouds (see Section 2.2.4). Though we can get high efficiency by region proposals, the method inadvertently filters a small portion of true comma-shaped clouds. We show the IoU-recall curves in Fig. 2.4 for analyzing the effect of these factors to the recall rate. We
provide our choice of IoU=0.50 as the blue dot in the plot and explain the reasons below.

Among the three curves in Fig. 2.4, the green curve, marked as the Optimal Recall, indicates the theoretical highest recall rate we can obtain with IoU changes. Because we have strong requirements for the sizes and locations of sliding windows in our algorithm, but do not apply those restrictions to human labelers, labeled regions and sliding windows cannot have a 100% overlap due to human perception variations. Thus, we use the maximum IoU between each labeled region and all sliding windows as the highest theoretical IoU of this algorithm. The red curve, marked as Recall before Region Proposals, indicates the true recall we can get which considers missing images, image corruption, and high-cloud segmentation errors. Within our dataset, there are 11.26% (5,926) of satellite images that are missing from the NOAA satellite image dataset, 0.36% (188) that have no recognized clouds, and 3.33% (1,751) that have abnormally low contrast. Though low contrast level or dark images can be adjusted by histogram equalization, the pixel brightness values do not completely follow the GMMs estimated in the background extraction step. Some high clouds are mistakenly removed with the background. In that experimental setting, this curve is the highest recall we can get before region proposals. The blue curve, marked as Recall after Region Proposals, indicates the true recall we can get after region proposals, where the detailed process to design region proposals is in the following Section 2.2.4.

The positive training samples consist of sliding windows whose IoU with labeled regions are higher than a carefully chosen threshold in order to guarantee both a reasonably high recall and a high accuracy. As a convention in object detection tasks, we expect IoU threshold $\geq 0.50$ to ensure visual similarity with manually labeled comma-shaped clouds,
and a reasonably high recall rate (≥ 90%) in total for enough training samples. Finally, the IoU threshold is set to be 0.50 for our task. The recall rate is 92.26% before region proposals and 90.66% after region proposals.

After establishing these boundaries, we partition the dataset into three parts: training set, cross-validation set, and testing set. We use the data of the first 250 days of the year 2008 as the training set, the last 116 days of that year as the cross-validation set, and data from the years 2011 and 2012 as the testing set. The separation of the training set is due to the unusually large number of severe storms in 2008. The storm distribution ratio in the training, cross-validation, and testing sets are roughly 50% : 15% : 35%. There are strong data dependencies between consecutive images. Splitting our data by time rather than randomly breaks this type of dependencies and more realistically emulates the scenarios within our system. This data partitioning scheme is also valid for the region proposals described in Section 2.2.4.

2.2.2 High-Cloud Segmentation

We first segment the high cloud part from the noisy original satellite data. Raw satellite images contain all the objects that can be seen from the geosynchronous orbit, including land, seas, and clouds. Among all the visible objects in satellite images, we focus on the dense middle and top clouds, which we refer to as “high cloud” in the following. The high cloud is important because the comma-shaped phase is most evident in this part, according to [47].

The prior work [86] implemented the single-threshold segmentation method to separate clouds from the background. This method is based on the fact that the high cloud looks brighter than other parts of the infrared satellite images [72]. We evaluate this method and show the result in the second column of Fig. 2.5. Although this method can segment most high clouds from the background, it misses some cloud boundaries. Because Earth has periodic temperature changes and ground-level temperature variations, and the variations are affected by many factors including terrains, elevation, and latitudes, a single threshold cannot adapt to all these cases.

The imperfection of the prior segmentation method motivates us to explore a data-driven approach. The overall idea of the new segmentation scheme is described as follows: To be aware of spatiotemporal changes of the satellite images, we divide the image pixels into tiles, and then model the samples of each unit using a GMM. Afterward, we identify the existence of a component that most likely corresponds to the variations of high cloud-sky brightness.
We build independent GMMs for each hour and each spatial region to address the challenges of periodic sunlight changes and terrain effects. As sunlight changes in a cycle of one day, we group satellite images by hours and estimate GMMs for each hour separately. Furthermore, since land conditions also affect light conditions, we divide each satellite image into non-overlapping windows according to their different geolocations. Suppose all the pixels are indexed by their time stamp \( t \) and spatial location \( x \), we divide each day into 24 hours, and divide each image into non-overlapping windows. Each window is a square of \( 32 \times 32 \). Thus, for each hour \( h \) and each window \( L \), we form a group of pixels \( G_{h,L} = \{ I(t, x) : t \in T_h, x \in X_L \} = \{ I_{h,L}(t, x) \} \), with brightness \( I(t, x) \in [0, 255] \). Each pixel group \( G_{h,L} \) has about 150,000 samples. We model each group by a GMM with the number of components of that group \( K_{h,L} \) to be 2 or 3, i.e.,

\[
I_{h,L}(t, x) \sim \sum_{i=1}^{K_{h,L}} \phi_{h,L}^{(i)} \mathcal{N}(\mu_{h,L}^{(i)}, \Sigma_{h,L}^{(i)}),
\]

where

\[
K_{h,L} = \arg \min_{i=2,3; (t, x) \in T_h \times X_L} \{ \text{AIC}(K_{h,L} = i | t, x) \}.
\]

Here \( \psi_{h,L} = \{ \phi_{h,L}^{(i)}, \mu_{h,L}^{(i)}, \Sigma_{h,L}^{(i)} \}_{i=1}^{K_{h,L}} \) are GMM parameters satisfying \( \mu_{h,L}^{(i)} > \mu_{h,L}^{(j)} \) for \( \forall i > j \), which are estimated by the k-means++ method [87]. AIC(\cdot) is the Akaike information criterion function of \( K_{h,L} \). We can interpret the GMM component number \( K = 2 \) as the GMM peaks fit high-sky clouds and land, while \( K = 3 \) as the GMM peaks fit high-sky clouds, low-sky clouds, and the land. So for each GMM \( \psi_{h,L} \), the component with the largest mean is the one modeling high cloud-sky. We then compute the normalized density of the point \((t, x)\) over \( \psi_{h,L} \). We annotate this normalized density as \( \{ p_{h,L}^{(i)}(t, x) \}_{i=1}^{K_{h,L}} \) and define the intensity value after segmentation to be \( \tilde{I}(t, x) := \)

\[
\begin{cases} 
I_{h,L}(t, x) \cdot p_{h,L}^{(1)}(t, x) & \text{if } I_{h,L}(t, x) \cdot p_{h,L}^{(1)}(t, x) \geq \sigma \\
0 & \text{otherwise}
\end{cases}
\]

(2.1)

where \( \sigma \) is chosen empirically between 100 and 130, with low impact to the features extracted. In our experiment, we choose 120 for convenience.

We then apply a min-max filter between neighboring GMMs in spatiotemporal space. Based on the assumption that cloud movement is smooth in spatiotemporal space, GMM parameters \( \psi_{h,L} \) should be a continuous function over \( h \) and \( L \). For most pixel groups
which we have examined, we observe that our segmented cloud changes smoothly. But in case the GMM component number changes, \( \mu^{(1)}_{h,L} \) would also change in both \( h \) and \( L \), resulting in significant changes to the segmented cloud. To smooth the cloud boundary part, we post-process a min-max filter to update \( \mu^{(1)}_{h,L} \), which is given by

\[
\mu^{(1)\text{new}}_{h,L} := \max \left\{ \mu^{(1)}_{h,L}, \min_{h' \in \mathcal{N}_h} \left\{ \mu^{(1)}_{h',L'} \right\} \right\},
\]

where \( \mathcal{N}_h = [h - 1, h + 1] \) and \( \mathcal{M}_L = \{ l : | l - L| \leq 1 \} \). The min-max filter leverages the smoothness of GMMs within spatiotemporal neighbors. After applying Equation (2.2), we update normalized densities and receive more smooth results with Equation (2.1). Example high-cloud segmentation results are shown in the third column of Fig. 2.5. At the end of this step, high clouds are separated with detailed local information, while the shallow clouds and the land are removed.

### 2.2.3 Correlation with Motion Prior

Another evident feature of the comma-shaped clouds is motion. In the cyclonic system, the jet stream has a strong trend to rotate around the low center, which makes up the head part of the comma in the satellite image [64]. We design a visual feature to extract this cloud motion information, namely \textit{Motion Correlation} in this section. The key idea is that the \textit{same} cloud at two \textit{different} spatiotemporal points should have a strong positive correlation in appearance, based on a reasonable assumption that clouds move at a nearly-uniform speed within a small spatial and time span. Thus, the cloud movement direction can be inferred from the direction of the maximum correlation. This assumption was first applied to compute cross-correlation in [4].

We therefore define the motion correlation of location \( x \) on the time interval of \((t - T, t]\) to be:

\[
M(t, x) = \text{corr}_{t_0 \in (t - T, t]}(I(t - t_0, x), I(t, x + h)),
\]

where \( \text{corr}(\cdot, \cdot) \) denotes the Pearson correlation coefficient, and \( h \) is the cloud displacement distance in time interval \( T \). This motion correlation can be viewed as an improved cross-correlation in [4], which we mentioned in Section 2.1. The cross-correlation can be written in the following form:

\[
M_0(t, x) = \text{corr}_{\|x_1 - x\| \leq h_0}(I(t - T_0, x), I(t, x_1 + h)),
\]
where $T_0$ is the time span between two successive satellite images.

We can conclude that our motion correlation is *temporally smoothed* and the cross-correlation is *spatially smoothed* by comparing Equation (2.3) and (2.4). The cross-correlation feature focuses on the differences in only two images, and then it takes the average on a spatial range. On the other hand, our correlation feature, with motion prior, interprets movement accumulation during the entire time span. We further re-normalize both $M(\cdot, \cdot)$ and $M_0(\cdot, \cdot)$ to $[0, 255]$ and visualize these two motion descriptors in the fourth and fifth columns of Fig. 2.5, where we fix $h$ to be 10 pixels, $T$ to be 5 hours, and $h_0$ to be 128 pixels. The cross-correlation feature (fourth column of Fig. 2.5) is noncontinuous across the area boundary. In image time series, the cross-correlation feature expresses less consistent positive/negative correlation in one neighborhood than our motion correlation does. Compared with the cross-correlation feature, our motion correlation feature (fifth column of Fig. 2.5) shows more consistent texture with the cloud motion direction.

### 2.2.4 Region Proposals

In this stage, we design simple classifiers to filter out a majority of negative sliding windows. This method was also applied in the sliding window selection of [88]. Because only a very small proportion of sliding windows generated contain comma-shaped clouds, we can save computation in subsequent training and testing processes by reducing the number of sliding windows.

We apply three weak classifiers to decrease the number of sliding windows. The first classifier removes candidate sliding windows if their average pixel intensity is out of the range of $[50, 200]$. Comma-shaped clouds have typical shape characteristics that the cloud body part consists of dense clouds, but the dry tongue part is cloudless. Hence, the average intensity of a well-cropped patch should be within a reasonable range, neither too bright nor too dark. Finally, this classifier removes most cloudless bounding boxes while keeping over 98% of the positive samples.

The second classifier uses a linear margin to separate positive examples from negative ones. We train this linear classifier on all the positive sliding windows with an equal number of randomly chosen negative examples, and then validate on the cross-validation set. All the sliding windows are resized to $256 \times 256$ pixels and vectorized before feeding into training, and the response variable is positive (1) or negative (0). As a result, the classifier has an accuracy of over 95% on the training set and over 80% on the cross-validation set. To ensure a high recall of our detectors, we output the probability
of each sliding window and then set a low threshold value. Sliding windows that output probability less than this threshold value are filtered out. The threshold ensures that no positive samples are filtered out. We randomly change the train-test split for ten rounds and set the final threshold to be 0.2.

Finally, we compute the pixel-wise correlation $\gamma$ of each sliding window $I$ with the average comma-shaped cloud $I_0$. This correlation captures the similarity to a comma shape. $\gamma$ is computed as:

$$\gamma = \frac{I \cdot I_0}{\|I\|_{L_2} \cdot \|I_0\|_{L_2}}.$$

(2.5)

Because there are no visual differences between different categories of storms (as shown in the last row of the table in Fig. 2.6), $I_0$ is the average labeled comma-shaped
Figure 2.6: Top: The correlation probability distribution of all sliding windows. Middle: Some segmented image examples. The last example image is the average image of the manually labeled regions in the training set. The correlation score $\gamma$ is defined in Equation (2.5), and the diagram is the normalized probability distribution of $\gamma$ of the training set. Bottom: Average comma-shaped clouds in different categories.
clouds in the training set. The computation process of $I_0$ consists of the following steps. First, we take all the labeled comma-shaped clouds bounding boxes in the training set and resize them to $256 \times 256$. Next, we segment the high-cloud part from each image using the method in Section 2.2.2. Finally, we take the average of the high-cloud parts. The resulting $I_0$ is marked as Avg. in the middle row of the table in Fig. 2.6. To be consistent in dimensions, every sliding window $I$ is also resized to $256 \times 256$ in Equation (2.5).

The higher correlation $\gamma$ indicates that a cloud patch has the appearance of comma-shaped clouds. Based on this observation, a simple classifier is designed to select sliding windows whose $\gamma$ is higher than a certain threshold. Fig. 2.6 serves as a reference to choose a customized threshold of $\gamma$. The distribution and some example images of $\gamma$ is listed in the table of Fig. 2.6. In the training and cross-validation sets, less than 1% of positive examples have a $\gamma$ value lower than 0.15. So, we use $\gamma \geq 0.15$ as the final filter choice to eliminate sliding window candidates.

The region proposal process only permits about $10^3$ bounding boxes per image, which is only one-tenth of the initial number of bounding boxes. As shown in Fig. 2.4, the region proposals process does not significantly affect the recall rate, but it can save much time for the later training process.

### 2.2.5 Construction of Weak Classifiers

We design two sets of descriptive features to distinguish comma-shaped clouds. The first is the histogram of oriented gradients (HOG) [88] feature based on segmented high clouds. For each of the region proposals, we compute the HOG feature of the bounding box. Because we compute the HOG feature on the segmented high clouds, we refer to it as *Segmented HOG* feature in the following paragraphs. The second is the histogram feature of each image crop based on the motion prior image, where the texture of the image reflects the motion information of cloud patches. We fine-tune the parameters and show the accuracy of the cross-validation set in Table 2.1. We use the Segmented HOG setting #4 and Motion Histogram Setting #5 as the final parameter setting in our experiments because it has a better performance on the cross-validation set. The feature dimension is 324 for Segmented HOG and 27 for Motion Histogram.

Since severe weather events have a low frequency of occurrence, positive examples only take up a very small proportion ($\sim 1\%$) in the whole training set. To utilize negative samples fully in the training set, we construct 100 linear classifiers. Each of these classifiers is trained on the whole positive training set and an equal number of negative
Table 2.1: Avg. Accuracy of weak classifiers for the segmented HOG and motion histogram features in different parameter settings

<table>
<thead>
<tr>
<th>Seg. HOG Settings</th>
<th>Orientation Directions</th>
<th>Pixels/Cell</th>
<th>Cells/Block</th>
<th>(%)Avg. Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>9</td>
<td>64 x 64</td>
<td>1 x 1</td>
<td>69.88 ± 1.15</td>
</tr>
<tr>
<td>#2</td>
<td>18</td>
<td>64 x 64</td>
<td>1 x 1</td>
<td>70.65 ± 1.25</td>
</tr>
<tr>
<td>#3</td>
<td>9</td>
<td>128 x 128</td>
<td>1 x 1</td>
<td>61.21 ± 0.62</td>
</tr>
<tr>
<td>#4</td>
<td>9</td>
<td>64 x 64</td>
<td>2 x 2</td>
<td>73.18 ± 0.98</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Motion Hist. Settings</th>
<th>Pixels to the West h*</th>
<th>Time Span in hours T*</th>
<th>Hist. Bins</th>
<th>(%)Avg. Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>10</td>
<td>2</td>
<td>18</td>
<td>58.84 ± 0.59</td>
</tr>
<tr>
<td>#2</td>
<td>5</td>
<td>2</td>
<td>18</td>
<td>52.97 ± 0.20</td>
</tr>
<tr>
<td>#3</td>
<td>10</td>
<td>2</td>
<td>9</td>
<td>58.83 ± 0.20</td>
</tr>
<tr>
<td>#4</td>
<td>10</td>
<td>2</td>
<td>27</td>
<td>61.05 ± 0.63</td>
</tr>
<tr>
<td>#5</td>
<td>10</td>
<td>5</td>
<td>27</td>
<td>63.25 ± 0.67</td>
</tr>
</tbody>
</table>

* h and T have the same meaning as annotated in Equation (2.3).

Figure 2.7: The accuracy distribution of weak classifiers with Segmented HOG feature and Motion Histogram feature.

samples. We split and randomly select these 100 batches according to their time stamp so that their time stamps are not overlapped with other batches. We train each logistic regression model on the segmented HOG features and the motion histogram feature of the training set. Finally, we get 200 linear classifiers. We evaluate the accuracy of the trained linear classifiers based on a subset of testing examples whose positive/negative ratio is also 1-to-1. The average accuracy of the segmented HOG feature is 73.18% and that of the motion histogram features is 63.25%. The accuracy distribution of these two types of weak classifiers is shown in Figure 2.7. From the statistics and the figure, we know the
Table 2.2: Accuracy of different stacked generalization methods on the cross-validation set

<table>
<thead>
<tr>
<th>Method</th>
<th>(%) Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>LR</td>
<td>85.10 ± 0.20</td>
</tr>
<tr>
<td>Bagging</td>
<td>81.98 ± 0.46</td>
</tr>
<tr>
<td>RF</td>
<td>82.34 ± 0.40</td>
</tr>
<tr>
<td>ERF</td>
<td>82.45 ± 0.34</td>
</tr>
<tr>
<td>GBM</td>
<td>85.77 ± 0.25</td>
</tr>
<tr>
<td>AdaBoost</td>
<td>86.47 ± 0.25</td>
</tr>
</tbody>
</table>

*LR: Logistic Regression; RF: Random Forest; ERF: Extremely Random Forest; GBM: Gradient Boosting Machine with deviance loss.

Table 2.3: Accuracy of the AdaBoost detector on the cross-validation set with different parameters

<table>
<thead>
<tr>
<th>Tree layer</th>
<th>Leaf Nodes</th>
<th>(%) Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>40</td>
<td>86.47 ± 0.25</td>
</tr>
<tr>
<td>2</td>
<td>40</td>
<td>86.27 ± 0.24</td>
</tr>
<tr>
<td></td>
<td>60</td>
<td>84.97 ± 0.25</td>
</tr>
</tbody>
</table>

Segmented HOG feature has higher average accuracy than the Motion Histogram feature, and a larger variation in the accuracy distribution. As shown in Figure 2.7, about 90% of the classifiers on the motion histogram feature have an accuracy between 63% and 65%, while those on the segmented HOG feature distribute in a wider range from 53% to 80%.

2.2.6 AdaBoost Detector

We apply the stacked generalization on the probability output of our weak classifiers [89]. For each proposed region, we use the probability output of the 200 weak classifiers as the input, and get one probability $p$ as the output. We define the proposed region as positive for $p \geq p_0$ or negative in other cases, where $p_0 \in (0, 1)$ is our given cutoff value.

We adopt AdaBoost [90] as the method for stacked generalization because it achieves the highest accuracy on the balanced cross-validation set, as shown in Table 2.2. All these classifiers are constructed on the training set and fine-tuned on the cross-validation set. Table 2.3 shows the accuracy of the AdaBoost classifier with different parameters. For each set of parameters, we provide a 95% confidence interval computed on 100 random seeds in both Table 2.2 and 2.3. The classification accuracy reaches the maximum at 86.47% with 40 leaf nodes and one layer. The AdaBoost classifier running on region proposals is our proposed comma-shaped cloud detector.

We then run the AdaBoost detector on the testing set and then compute the ratio of
the labeled comma-shaped clouds that our method can detect. For each image, we choose the detection regions that have the largest probability scores of having comma-shaped clouds (abbreviated as probability in this paragraph), and we ensure every two detection regions in one image have an IoU less than 0.30 — a technique called non-maximum suppression (NMS) in object detection [91]. If one output region has IoU larger than 0.30 with another output region, we remove the one with a lower probability from the AdaBoost detector. Finally, the detector outputs a set of sliding windows, with each region indicating one possible comma-shaped cloud.

In our experiment, the training set for ensembling is a combination of all 68,708 positive examples and a re-sampled subset of negative examples sized ten times larger than the size of positive examples (i.e., 687,080). We carry out experiments with the Python 2.7 implementation on a server with the Intel® Xeon X5550 2.67GHz CPUs. We apply our algorithm on every satellite image in parallel. If the cutoff threshold is set to be 0.50, the detection process for one image, from high-cloud segmentation to AdaBoost detector, costs about 40.59 seconds per image. Within that time, the high-cloud segmentation takes 4.69 seconds, region proposals take 14.28 seconds, and the AdaBoost detector takes 21.62 seconds. We only get two satellite images per hour now, and these three processes finish in sequential order. If higher speed is needed, implementation in C/C++ is expected to be substantially faster.

2.3 Experiments

In this section, we present the evaluation results for our detection method. First, we present an ablation study. Second, we show that our method can effectively detect both comma-shaped clouds and severe thunderstorms. Finally, we compare our method with two other satellite-based storm detection schemes and show that our method outperforms both.

2.3.1 Ablation Study

To examine how much each step contributes to the model, we carried out an ablation study and show the results in Table 2.4. We enumerate all the combinations in terms of high-cloud segmentation and features. The first column indicates whether the region proposals are on the original satellite images or on the segmented ones. The second column separates HOG feature, Motion Histogram feature, and their combinations.
Table 2.4: Accuracy of the AdaBoost classifier on the cross-validation set with different features. Here HOG with high-cloud segmentation = Segmented HOG feature; Motion Hist. = Motion Histogram Feature.

The last column shows the accuracy of the cross-validation set with a 95% confidence interval. If we do not use high-cloud segmentation, the combination of HOG and Motion Histogram features outperforms each of them. If we use high-cloud segmentation, the combination of these two features also performs better than each of them, and it also outperforms the combination of features without high-cloud segmentation. In conclusion, the effectiveness of our detection scheme is due to both high-cloud segmentation process and weak classifiers built on shape and motion features.

### 2.3.2 Detection Result

The evaluation in Figure 2.8 shows our model can detect up to 99.39% of the labeled comma-shaped clouds and up to 79.41% of storms of the year 2011 and 2012. Here we define the term “detect comma-shaped clouds” as: If our method outputs a bounding box having IoU $\geq 0.50$ with the labeled region, we consider such bounding box detects comma-shaped clouds; otherwise not. We also define “detect a storm” as: If any storm in the NOAA storm database is captured within one of our output bounding boxes, we consider we detect this storm.

The comma-shaped clouds detector outputs the probability $p$ of each bounding box from AdaBoost classifier. If $p \geq p_0$, this bounding box consists of comma-shaped clouds. We recommend $p_0$ to be set in $[0.50, 0.52]$, and we provide three reference probabilities $p_0 = 0.50, 0.51$ and 0.52. The number of detections per image as well as the missing rate of comma-shaped clouds and storms corresponding to each $p_0$ are available in the right part of Figure 2.8. For a user who desires a high recall rate, e.g. a meteorologist, we recommend setting $p_0 = 0.50$. The recall rate of the comma-shaped clouds is 99% and
the recall rate of storms is 64% under that setting. Our detection method outputs an average of 7.76 bounding boxes per satellite image. Other environmental data, like wind speed and pressure, are needed to be incorporated into the system to filter the bounding boxes. For a user who desires accurate detections, we recommend setting $p_0 = 0.52$. The recall rate of the comma-shaped clouds is 80%, and our detector outputs an average of 1.09 bounding boxes per satellite image. The recall rate is reasonable, and the user will not get many incorrectly reported comma-shaped clouds.

The setting $p_0 \in [0.50, 0.52]$ could give us the best performance for several reasons. When $p_0$ goes under the value 0.50, the missing rate of the comma-shaped clouds almost remains the same value ($\sim 1\%$), and we need to check more than 8 bounding boxes per image to find these missing comma-shaped clouds. It consumes too much human effort. When $p_0$ goes over the value 0.52, the missing rate of comma-shaped clouds goes over 20%, and the missing rate of storms goes over 77%. Since missing a storm could cause severe loss, $p_0 > 0.52$ cannot provide us with a recall rate that is high enough for the storm detection purpose.

Though our comma-shaped clouds detector can effectively cover most labeled comma-shaped clouds, it still misses at least 20% storms in the record. Among different types of storms, severe weather events on the ocean have a higher probability to be detected in the algorithm than other types of severe weather events. At the point of the largest recall, our method detects approximately 85% severe weather events on the ocean versus 75% on the land. Our detector misses such events because severe weather does not always

---

*Here severe weather events on the ocean includes marine thunderstorm wind, marine high wind, and marine strong wind.*
happen near the low center of the comma-shaped clouds. According to [47] and [92], the exact cold front and the warm front streamline cannot be accurately measured from satellite images. Hence, comma-shaped clouds are simply an indicator of storms, and further investigation in their geological relationships is necessary to improve our method.

### 2.3.3 Storm Detection Ability

We compare the storm identification ability of our algorithm with other baseline methods that use satellite images. The first baseline method comes from [93] and [94], and the second baseline improves the first in [57]. We call them *Intensity Threshold Detection* and *Spatial-Intensity Threshold Detection* hereafter.

The Intensity Threshold Detection uses a fixed reflectivity level of radar or infrared satellite data to identify a storm. A continuous region with a reflectivity level larger than a certain threshold $I_0$ is defined as a storm-attack area. Spatial-Intensity Threshold Detection improves it by changing the cost function to be a weighted sum:

$$ E = \sum_{i=1}^{n} \lambda d_m(x_i) + (1 - \lambda) d_c(x_i), $$

where $X = \{x_i\}_{i=1}^{n}$ is the point set representing a cloud patch, $d_m$ is the spatial distance within the cluster, and $d_c$ is the difference between the pixel brightness $I(x_i)$ and the average brightness of the cloud $X$.

We make some necessary changes to the baselines to make two methods comparable. First, we explore different $I_0$ value, because we use the different channels and satellites with the baselines. In addition, the light distribution of images is changed through histogram equalization in the preprocessing stage, so we cannot simply adopt $I_0$ used in the baselines. Second, we change the irregular detected regions to the square bounding boxes and use the same criteria to define positive and negative detections. We adopt the idea in [58] and view these pixel distributions as 2D GMM. We use Gaussian means and the larger eigenvalue of Gaussian covariance matrix to approximate a bounding box center and a bounding box size, respectively. The number of Gaussian components and other GMM parameters are estimated by mean Silhouette Coefficient [95] and the k-means++ method.

The partial recall-precision curve in Figure 2.9 shows that our method outperforms both Intensity Threshold Detection and Spatial-Intensity Threshold Detection when the recall is less than 0.40. We provide only a partial recall-precision curve because
of the limited range of $I_0$ under the limited time and computation resources. In our experiment, we change parameters $I_0$ in Intensity Threshold Detection from 210 to 230. When $I_0$ goes over 230, very few pixels would be selected so this method cannot ensure a high recall rate. When $I_0$ goes under 210, many pixels representing low clouds are also included in the computation, which slows down the computations ($\sim$ 5 minutes per image). Consequently, we do not explore those values. As for Spatial-Intensity Threshold Detection, $I_0$ is fixed at the value of 225, and $\lambda$ is the weight between 0 and 1. When $\lambda$ changes from 0 to 1, the recall first goes up and then moves down, while the precision changes very little. The curve representing Spatial-Intensity Threshold Detection reaches the highest recall at 43.66% when $\lambda$ approaches 0.7.

Compared with the two baselines that detect storm events directly, our proposed method has the following strengths: (1) Our method can reach a maximum recall of 79.41%, almost twice as those of the baseline methods. Due to computational speed issues, we could not increase the recall rate of the two baseline methods to be higher than 45%, which limits their use in practical storm detections. For our method, however, we can reach a high recall rate without heavy computational cost. (2) Our method outperforms these two baseline methods in the precision rate of Figure 2.9. Compared with these two methods that mostly rely on pixel-wise intensity, our method comprehensively combines the shape and motion information of clouds in the system, leading to a better performance in storm detection.

None of the three curves in Figure 2.9 have a high precision of detecting storm
events, because this task is difficult especially without the help of other environmental data. In addition, our method aims to detect comma-shaped clouds, rather than to forecast storm locations directly. Sometimes severe storms are reported later than the appearance of comma-shaped clouds. Such cases are not counted in the precision rate of Figure 2.9. In those cases, our method can provide useful and timely information or alerts to meteorologists who can make the determination.

Figure 2.9 also points out the importance of exploring the spatial relationship between comma-shaped clouds and storm observations, as the Spatial-Intensity Threshold Detection method slightly outperforms our method when the recall rate is higher than 0.40. According to the trend of the green curve, adding spatial information to the detection system can improve the performance to some extent. We will consider combining spatial information into our detection framework in the future.

### 2.3.4 Case Studies

We present three case studies (a-c) in Figure 2.10 to show the effectiveness and some imperfections of our proposed detection algorithm. The green bounding boxes are our detection outputs, the blue bounding boxes are comma-shaped clouds identified by meteorologists, and the red dots indicate severe weather events in the database [82]. The detection threshold is set to 0.52 to ensure the precision of each output-bounding box. The descriptions of these storms are summarized from [83].

In the first case (row 1), strong wind, hail, and thunderstorm wind developed in the central and northeast part of Colorado, west of Nebraska and east of Wyoming, on late June 6, 2012. The green bounding box in the top-left corner of Fig. 2.10 - (a1) indicated this region. Then, a dense cloud patch moved in the eastern direction and covered eastern Wyoming, western South Dakota, and western Nebraska on early June 7, 2012. At that time, these states reported property damages to different degrees. Later on June 7, 2012, the cloud patch became thinner as it moved northward to Montana and North Dakota, as shown in (a3). Our method had a good tracking of that cloud patch all the time, even though the cloud shape did not look like a typical comma. In comparison, human eyes did not recognize it as a typical comma shape because the cloud lost the head part. Another region detected to have a comma-shaped cloud in (a1) was around North Texas and Oklahoma. At that time, hail and thunderstorm winds were reported, but the comma shape in the cloud began to disappear. Another comma-shaped cloud began to form in the Gulf of Mexico, as seen in the center part of (a1). At that time, the comma shape was too vague to be discovered by either our computer detector or by human eyes.
Figure 2.10: (a-c) Three detection cases. Green frames: our detection windows; Blue frames: our labeled windows; Red dots: storms. Some images have blank in left-bottom because it is out of the satellite range. All the times are in UTC.

As time passed \((a_2)\), the comma-shaped cloud appeared, and it was detected by both our detector and human eyes. The clouds gathered as some severe events in North Florida in \((a_2)\). According to the record, a person was injured, and Florida experienced severe property damage at that time. Later that day, the large comma-shaped cloud split into two parts. The cloud patch in the west had an incomplete shape, which is difficult for human eyes to discover, as shown in \((a_3)\). However, our method successfully detected this change. In addition, our method detected all the recorded severe weather events. This example indicates our method is able to detect incomplete or atypical comma-shaped clouds, including when one comma-shaped cloud splits into two parts.

In the second case (row 2), a comma-shaped cloud appeared in the sky over Oklahoma, Kansas, and Missouri on Feb 24, 2011, when these areas were attacked by winter weather, flooding, and thunderstorm winds. Our method detected the comma-shaped cloud half an hour earlier than human eyes were able to capture it, as shown in \((b_1)\). Soon \((b_2)\), a clear comma-shaped cloud formed in the middle of the image, which was detected by both our method and human eyes. Red dots in \((b_2)\) show the location of some severe weather events that happened in Tennessee and Kentucky at that time. Since the cloud
patch was large, it was difficult to include the whole cloud patch in one bounding box. In that case, human eyes could correctly figure out the middle part of the wide cloud to label the comma-shaped cloud. In comparison, our detector used two bounding boxes to cover the cloud patch, as shown in (b₂) and (b₃). Because there was only one comma-shaped cloud, our method outputs a false negative in that case.

In the third case (row 3), there were two comma-shaped cloud patches from late Jan 2, 2011 to the early next day, located in the left and the right part of the image, respectively. Our method detected the comma-shaped cloud in south California one hour (i.e., two continuous satellite images) later than the human eye detected it. Importantly, however, after the region is detected, our method detected the right comma-shaped cloud over the North Atlantic Ocean one hour earlier than human eyes did. As indicated in the left part of (c₂) and (c₃), our output is highly overlapped with the labeled regions. Our method was able to recognize the comma-shaped cloud when the cloud just began to form in (c₂). In the beginning, human eyes cannot recognize its shape, but our method was able to capture that vague shape and motion information to make a correct detection.

To summarize these studies, our method can capture most human-labeled comma-shaped clouds. Moreover, our method can detect some comma-shaped clouds even before they are fully formed, and our detections are sometimes earlier than human eye recognition. These properties indicate that using our method to complement human detections in practical weather forecasting may be beneficial. On the other hand, our detection scheme has a weakness as indicated in case (b). It has difficulty outputting the correct position of spatially wide comma-shaped clouds.

### 2.4 Summary and Discussion

We propose a new computational framework to extract the shape-aware cloud movements that relate to storms. Our algorithm automatically selects the areas of interest at suitable scales and then tracks the evolving process of these selected areas. Compared with the human annotator’s performance, the computational algorithm provides an objective (yet agnostic) standard for defining the comma shape. The system can assist meteorologists in their daily case-by-case forecasting tasks.

Shape and motion are two visual clues frequently used by meteorologists in interpreting comma-shaped clouds. Our framework includes both the shape and the motion features based on the cloud segmentation map and correlation with the motion-prior map. Our experiments also validate the usage of these two visual features in detecting comma-
shaped clouds. Further, considering the high variability of cloud appearance in satellite images affected by seasonal, geographical and temporal factors, we take a learning-based approach to enhance the robustness, which may also benefit from additional data.

Finally, the detection algorithm provides us a top-down approach to explore how severe weather events happen. Our future work will integrate this framework with the use of other data sources and models to improve the reliability and timeliness of storm forecasting.
Chapter 3  
Pattern Matching with Optimal Transport: Background

Instead of recognizing certain known patterns on scientific images, some applications analyze changes of scientific image time series, or a combination of multiple images towards the same object. Examples include the diagnosis of disease on radiology after some treatment, estimating the speed of a tornado from radar videos, and observing bacteria activity from microscopy. Pattern matching is the right technique to study such changes in time series and to further build the pairwise relationship between two images.

The analysis in [96] categorizes pattern matching algorithms into two types: geometry methods and iconic methods. Geometric methods extract a subset of landmarks, such as key points on two objects, and maximize the pairwise similarity between these two point sets. Methods of key points selection include using blob detectors [97], Harris detector [98], and SIFT detector [99]. On the other hand, iconic methods integrate the dissimilarity measurement into the optimization function and set up the matching by minimizing the total dissimilarity function. Examples of dissimilarity measurement include square of total difference [100], cross correlation [101], Kullback-Leibler divergence [102], and OT [103,104]. Because landmark selection is infeasible on some fussy scientific images like satellite observations, we only discuss iconic methods in this thesis.

Histogram is a widely adopted form of image representation. Multiple image statistics such as gray-scale level, color, texture, and edges are extracted and represented in the form of histograms to describe features of image data. In this context, pattern matching is re-formulated to a new problem of comparing two sets of histogram descriptors. While researchers assert many efforts on designing more effective image descriptors, the metric between two statistical distributions in the forms of histograms is also a valuable topic
for exploration.

OT is an attractive framework to handle this problem because it provides a unified method for computing and comparing statistical distributions. Compared with some other metrics in information theory, such as Euclidean distance, Kullback-Leibler divergence, Wasserstein distance (a distance in OT, which will be detailed in the following section) takes the spatial distribution of histogram into consideration in pattern matching [105]. Additionally, by solving OT we can also get the transportation plan between two histograms, which is the pairwise matching between two distributions.

OT has gained increasing popularity in pattern matching applications. After first introduced in image comparison in [106], OT is widely used in image retrieval as the comparison criterion between image descriptors. The distance computed in OT was also named as *Earth Mover’s distance* in the literature [107]. Because OT encodes geometry information in histogram matching, it is especially helpful for image retrieval on small datasets [108]. This fact is consistent with our eye perception where both intensity and spatial information contribute to pattern recognition. Besides, OT was also successfully applied to various image processing applications, including image segmentation [109], image restoration [110], shape abstraction [104], and texture mixing [111].

In the following chapters, we will introduce the mathematical setup of OT and apply OT on some applications of scientific images. OT is a powerful and widely used optimization framework aiming to minimize the transportation cost between two statistical distributions. A byproduct with OT computation is the transportation plan between these two distributions. If we represent each pixel on the image with the associating feature vector and assign proper transportation cost between each pair of feature vectors, the transportation plan could be then treated as the matching plan between the two images. This is the underlying rationale of applying OT on pattern matching problems. In this chapter, we first introduce the classical formulation of the problem, then we come to some variations of the original conditions and their solutions. Finally, we introduce some challenges when we apply OT to the real-life pattern matching problems on scientific images.
3.1 Mathematical Setup of OT

OT problem originates from an engineering problem. The problem was abstracted by Monge in [112]. In discrete situation, the problem can be represented in the following. Suppose a worker is asked to transport a pile of sand from multiple source stations to multiple target stations. The locations of these stations and the amount of sand in each station are known, and the effort to transport a unit of sand for a unit distance is constant. The worker wants to minimize the total effort to move those sands. In this case, such effort refers to the sum of (distance between every source to target station) $\times$ (amount of sand transport between them). Later, the deterministic property of transportation was relaxed by Kantorovich [113]. He did not restrict one unit of sand in the source to transport to a deterministic unit of sand in the target, but rather he allowed a probabilistic transportation plan. This problem setting is also called the Monge-Kantorovich problem in the literature [114]. Mathematically, the solution of the minimal effort is attractive, because it defines the distance between two distributions, and the transportation plan naturally provides us a matching plan between the two distributions, which we will heavily rely on in the following applications. We refer to the transportation plan associated with minimal effort as optimal transportation plan in the following.

![Figure 3.1: Schematic OT illustration, where the red points represent the source $X_1$ and blue points represent the target $X_2$. Left: probability distributions at dimension $d = 1$ and $d = 2$. Right: transportation plan $T$ represented by black dots, where the size of dots represents the mass in transportation. Image credit to [5].](image)

Because we are more interested in using OT to solve pattern matching problem,
we only cover the discrete probability distributions in this thesis. Denote the source distribution by \( X_1 \) and target distribution by \( X_2 \). \( X_i \) is specified by a total of \( n_i \) support points \( x_{i,j} \in \mathbb{R}^d \) for \( i = 1, 2 \), where \( \mathbb{R}^d \) is a \( d \)-dimensional Euclidean space. For example, \( d = 1 \) indicates the sand locations are represented by 1D coordinate on a line, and \( d = 2 \) indicates the locations are specified by their 2D coordinates, as shown in the left part of Figure 3.1. In addition, each support point is assigned a mass associated with that point, which we can intuitively treat as the amount of sands to transit in the aforementioned toy example. Let the transit mass assigned to \( x_{i,j} \) be \( w_{i,j} \). Then the distributions are represented as \( X_i = \{ (x_{i,1}, w_{i,1}), (x_{i,2}, w_{i,2}), \ldots, (x_{i,n_i}, w_{i,n_i}) \} \) for \( i = 1, 2 \).

Let \( a_i = (w_{i,1}, \ldots, w_{i,n_i}) \in \mathbb{R}^{n_i}_+ \) be the weight simplex for \( i = 1, 2 \). Quantitatively, the total mass before the transportation in the source should be equivalent to the mass after the transportation in the target, i.e., \( \| a_1 \|_{L_1} = \| a_2 \|_{L_1} = 1 \). Denote the Euclidean distance matrix between the support points in \( X_1 \) and \( X_2 \) by \( M = (m_{j,j'}) \in \mathbb{R}^{n_1 \times n_2}_+ \), where \( m_{j,j'} \) is the distance between \( x_{1,j} \) and \( x_{2,j'} \). A transportation plan, or a point registration plan, specifies how the mass of simplex \( a_1 \) are redistributed to simplex \( a_2 \). The transportation plan can be represented by a matrix \( T = (t_{j,j'}) \in \mathbb{R}^{n_1 \times n_2} \), where element \( t_{j,j'} \) is the weight assigned from \( x_{1,j} \) to \( x_{2,j'} \). Denote \( 1_n \) as a \( n \)-dimensional vector with all the elements equal to one, and \( 1_{n_1 \times n_2} \) as a \((n_1 \times n_2)\)-dimensional matrix with all the elements equal to one. The set of transportation plans from \( a_1 \) to \( a_2 \) is defined as:

\[
U(a_1, a_2) \overset{\text{def}}{=} \{ T \in \mathbb{R}^{n_1 \times n_2}_+ | T 1_{n_2} = a_1, T' 1_{n_1} = a_2 \}.
\]

Intuitively, we can view the weight \( t_{j,j'} \) as the mass transported from source point \( x_{1,j} \) to target point \( x_{2,j'} \), and \( m_{j,j'} \) is the cost to transport a unit mass between two points, as illustrated by black dots in the right part of Figure 3.1. The minimized distance is named \( p \)-Wasserstein distance [115] in probability theory.

**Definition 1 (p-Wasserstein distance)** Consider two weight simplex \( a_1 \) and \( a_2 \) on Euclidean distance \( \mathbb{R}^d_+ \) and the cost matrix \( M \in \mathbb{R}^{d \times d}_+ \), \( WD_p(a_1, a_2) \) — the \( p \)-Wasserstein distance \( WD_p(a_1, a_2) \) raise to the power \( p \) — is defined by the minimized transportation cost

\[
WD_p(M, a_1, a_2) \overset{\text{def}}{=} \min_{T \in U(a_1, a_2)} \langle T, M \rangle ,
\]

38
where the optimal transportation plan

\[ T_{WD}^* (M, a_1, a_2) = \arg \min_{T \in U(a_1, a_2)} \langle T, M \rangle, \quad (3.3) \]

that reaches the minimum in Equation (3.2) is the optimal transportation plan.

OT not only solves the Wasserstein distance, but also provides the transportation plan \( T \in U(a_1, a_2) \) defined on the two probability distributions, which can be naturally treated as a “matching” between two datasets.

### 3.2 Computational OT Models

At the beginning, OT problem was proposed to solve by linear programming [116]. Assuming \( n = \max \{n_1, n_2\} \), the current best achievable computational complexity to solve OT is \( \tilde{O} \left( n^{5/2} \right) \) in theory [117] and \( \tilde{O} \left( n^3 \right) \) in practice [118]. High computational complexity has been a major concern of OT since that time [119–122], especially for high-definition images. More computational methods have been proposed to increase the computational efficiency. In addition, because spatial information is not sufficient to describe statistical distributions, other forms of transportation cost are proposed in the OT model. In this section, we introduce some variations of the original OT formula.

#### 3.2.1 Sinkhorn-Knopp Algorithm

Sinkhorn-Knopp Algorithm was proposed by Cuturi [120] to numerically approximate the solution of OT problem by a smooth function. The idea is to use the entropy term to regularize the total transportation cost, i.e.,

\[ T_{WD, \varepsilon} (M, a_1, a_2) \overset{\text{def}}{=} \arg \min_{T \in U(a_1, a_2)} \langle T, M \rangle - \varepsilon h(T). \quad (3.4) \]

Here \( \varepsilon \) is a small number to regularize the data and \( h(T) \) is the entropy function

\[ h(T) = - \sum_{j, j'} t_{j, j'} \log t_{j, j'}. \quad (3.5) \]

After adding the regularization term, optimization function in Equation (3.4) becomes \( \varepsilon \)-convex and smooth, as is shown in Figure 3.2. It is also proved that \( T_{WD, \varepsilon} \rightarrow T_{WD}^* \) as
\( \epsilon \to 0 \), and the differences can be ignored when \( \epsilon < 1e-2 \) according to the experiments on MNIST dataset [120].

![Figure 3.2: Impact of the entropy regularization term \( \epsilon \) to the optimization function. Here \( T^* \) is the OT solved from Equation (3.3) and \( T^*_\epsilon \) is the optimal transportation plan solved from Equation (3.4). Image credit to [5].](image)

The best achievable complexity of Sinkhorn algorithm is \( \tilde{O} \left( \frac{n^2}{\epsilon^3} \right) \) [119], according to the analysis in [121]. In pattern matching applications where the size of support points is large and entropy regularization factor \( \epsilon \) is fixed, Sinkhorn iterations are preferable because it achieves near-linear performance to the size of support points.

### 3.2.2 Gromov-Wasserstein Distance

For pattern matching applications, an important limitation of OT is that standard OT formula is not invariant to affine transforms. Some modified setup of the standard formula has been proposed in handling affine transforms [123,124], but they still require the source and target space can be transferred by some measurement, and the optimal transportation plan computed under this manner relies on the initial registration of two points sets.

A more general way to overcome this limitation is proposed by Mémoli [125,126], where he proposed to use the Gromov-Hausdorff distance [127] as a measurement of shape comparison between two metric spaces. This idea was further extended and formulated by Peyré et al. [122], where he used the similarity matrix \( C_i \in \mathbb{R}^{n_i \times n_i} \) to represent the intrinsic geometry dissimilarity of the distribution \( X_i \), and formulated the definition and computational approach to compute pattern matching with Gromov-Wasserstein distance. The Gromov-Wasserstein distance between two metric spaces \( (C_1, a_1) \) and \( (C_2, a_2) \) is defined as:
Definition 2 (Gromov-Wasserstein distance) The Gromov Wasserstein distance between two datasets with the metric-measure spaces \((C_1, a_1)\) and \((C_2, a_2)\) is defined as

\[
\text{GWD} (C_1, C_2, a_1, a_2) \overset{\text{def}}{=} \min_{T \in U (a_1, a_2)} \langle T, L (C_1, C_2) \otimes T \rangle ,
\]

where \(C_i\) is the similarity matrix of support points and \(a_i\) is the associated mass. \(L (a, b) = \frac{1}{2} |a - b|^2\) is the quadratic distance, and \(L \otimes T \overset{\text{def}}{=} \left( \sum_{k,l} L_{i,k,i} T_{i,j} \right)_{i,j}\) be the tensor-matrix multiplication.

\[
L (C_1, C_2) \overset{\text{def}}{=} \left( L \left( (C_1)_{i,k} , (C_2)_{j,l} \right) \right)_{i,j,k,l}
\]

is a 4-way tensor.

The transportation plan that reaches the minimum in Equation (3.6) is the optimal transportation plan under Gromov-Wasserstein distance.

\[
\mathcal{T}_{\text{GWD}}^* (C_1, C_2, a_1, a_2) = \arg \min_{T \in U (a_1, a_2)} \langle T, L (C_1, C_2) \rangle ,
\]

that reaches the minimum in Equation (3.2) is the optimal transportation plan.

An example of the similarity matrix \(C_i\) is the pairwise distance matrix, which is robust in coordination translation, scaling, and rotation.

The optimization function in Equation 3.6 is non-convex. Thus, computing the global minimum of the function is not feasible. Similar to the Sinkhorn-Knopp Algorithm, GWD can also be approximated by its entropy regularization form. The point registration \(\mathcal{T}_{\text{GWD},\varepsilon}\) deduced from KL projection iterations in [122] is defined as:

\[
\mathcal{T}_{\text{GWD},\varepsilon} (C_1, C_2, a_1, a_2) \overset{\text{def}}{=} \arg \min_{T \in U (a_1, a_2)} \langle T, L (C_1, C_2) \otimes T \rangle - \varepsilon h (T) .
\]

3.2.3 OT on Gaussians

In general, there is no closed-form formula of 2-Wasserstein distance between two general probability densities. But Givens and Shortt [128] provide the closed-form formula when the data follow Gaussian distributions.

Definition 3 (2-Wasserstein distance on Gaussians) Suppose \(X_i \sim \phi_i (\mu_i, \Sigma_i)\) follows Gaussian distributions, where \(\mu_i\) being the mean and \(\Sigma_i\) being the covariance matrix
for $i = 1, 2$, then 2-Wasserstein distance is defined as

$$
WD_2^2(\phi_1, \phi_2) = \|\mu_1 - \mu_2\|^2 + tr \left[ \Sigma_1 + \Sigma_2 - 2 \left( \Sigma_1^{\frac{1}{2}} \Sigma_2 \Sigma_1^{\frac{1}{2}} \right)^{\frac{1}{2}} \right].
$$

(3.10)

3.3 Limitations of Standard OT in Pattern Matching

As mentioned at the beginning of this chapter, the OT framework provides a pointwise pattern matching plan between two scientific images by solving the minimal distance between them. Images are thus represented by the set of feature vectors, for example, pixel coordinates, color distributions, and output from a 2D filter. The distance function is then defined based on the selected feature vectors.

There are two major limitations in this setting. First is the irregularity of the mapping between complicated densities. Classical OT setup requires the mass transported between two densities to be equivalent. But this setup is not feasible when we match two complicated scientific images. An example of this issue in [6] is shown in Figure 3.3.

We assume that both source and target distributions have two components, component $X_{1,1} - X_{2,1}$ and $X_{1,2} - X_{2,2}$ are matched according to spatial locations. But the matched pairs of components do not have equivalent masses, thus the extra masses in component $X_{1,1}$ are transported to component $X_{2,2}$. This small mass transportation is usually long distant and does not follow the component-wise information, thus we call this *irregularity in the transportation plan*.

![Figure 3.3: Example of irregularity in the classical OT formula. Image credit to [6].](image)

Figure 3.3: Example of irregularity in the classical OT formula. Image credit to [6].
Many efforts have been made to overcome this irregularity in classical OT. They can be categorized into two types: (1) imposing regularization onto the optimization function [129], and (2) relaxing the mass transportation conditions [6]. These two methods could avoid such artifacts only when the mass variation is small, and parameters of the regularization also require to be predetermined. In addition, the semantics of the whole image provide global information to the human eye. Such global information is not encoded in the regularization. This fact inspired us to redesign the OT framework specifically for pattern matching tasks on images.

Besides the irregularity issue, we are also motivated to re-examine the optimization function. Due to the large diversity in multiple scientific image processing, the structure of data is various: color images, videos, network structure (including protein organization, social network illustration, etc), and multi-dimensional images. If the support points are represented by features other than their spatial coordinates, the cost function should change to the dissimilarity of features accordingly. For example in the color transfer problem between two images, two datasets are represented by both the pixel-domain coordination and the color-domain values. The optimization formula and the cost functions would have to change because Euclidean distance on RGB color space cannot reflect their perceptual dissimilarity.

Another challenge for this problem is the heavy computational cost. In pattern matching problems where the datasets feature vectors of all the pixels, the computational time can become extensive, especially for high definition images. Solving OT efficiently is another objective we want to achieve.
Chapter 4  
Pattern Matching Application 1:  
Cell Tracking on Microscope Observations

In this chapter, we introduce a novel OT-based pattern matching algorithm on microscopy data. To be specific, our proposed method can track individual cells throughout the video by matching patterns across two consecutive frames. To overcome the limitations of pattern matching mentioned in Section 3.3, we combine the Sinkhorn approximation, unbalanced OT, and multiple iterations in local regions in the algorithm. We encode both shape and location of the cells in the optimization function and design a novel distance, namely weighted Gromov-Wasserstein distance, to better describe the cells. The details of this method follow.

4.1 Problem Introduction

In this section, we introduce how to use OT to design better algorithms for cell tracking, the task of associating biological cells in consecutive image frames from time-lapse image sequences. Cell tracking is an important task in Microbiology, as it provides better understanding of cell activities, such as division (mitosis), migration, and death (apoptosis).

There are two major categories of approaches to cell tracking, namely, tracking based

on cell detection [130–133] and tracking by evolving models [134–137]. The approach of tracking based on cell detection contains two steps: (1) cells are detected in each image frame and (2) the detected cells are aligned across the image frames. For tracking by evolving models, cell detection and tracking are tackled simultaneously based on models with parameters evolving with time. One main challenge in cell tracking for the existing approaches is aligning cells in consecutive frames. A cell is specified either by its centroid location in the image plane or by an expanded feature vector containing its location and other features, e.g., intensity and volume. A common idea is to match cells from two frames to their nearest neighbors in the feature space. This approach requires that cells are well separated in at least one dimension of the feature space [138].

Managing cell alignment between consecutive frames is a challenging task in cell tracking. Most existing methods align the cells according to an optimization criterion based on distances between cells in two image frames. A cell is typically described by a feature vector which is used to determine the cell-level distance. Such features include center coordinates in the image plane [7,139], histogram of pixel intensities [140], and contour-based features [141]. Various techniques have been developed to optimize the alignment, e.g., multi-assignment [132], integer programming [130,140,142], minimal cost flow [143], dynamic programming [133], OT [144], graphical models [131,145], and minimization of contour changes [134,135].

Cell shape is an important cue to distinguish multiple cells when they merge into one blob in the image [146]. Some contour-based features of the cell, such as the concave degree [147] and the parametric active contour models [148,149], are proposed to assist cell detection. However, contour-based methods rely heavily on the accuracy of cell detection [142] as well as prior knowledge about the cell shape [149]. Instead of using boundaries solely, we use the full set of pixels included in a cell to better capture shape characteristics.

Mitosis (cell division) is an important process during which the genetic material of a parental cell is equally distributed between its descendants through nuclear division. We can compute valuable features for a population after identifying mitosis, e.g., the proliferation rate. Capturing mitosis timely and accurately is challenging because two daughter cells may adhere to each other for a stretch of time [146]. Various methods have been developed for detecting mitosis. For example, [150] computes multiple cell features for each blob region; [151] uses Haar-like filters to extract features of the cell; and [147] measures the extent of the concavity of the boundary (roughly, how severe a convex contour is dented). These methods are usually designed for specific types of cells.
and are not generalizable. We attempt to develop a general framework for broad cell tracking applications.

Although the cells are described by the group of pixels in the detection stage, during the cell alignment phase, their specification is simplified into a certain parametric form, e.g., its centroid location [139, 152, 153], or its Gaussian distribution fitted onto pixel coordinates [142, 145]. However, these simplified specifications may lose considerable information about the shapes of the cells. Figure 4.1 shows an example from Pancreatic Stem Cells (PSC) [7]. The PSCs have rather irregular shapes, which cannot be captured by the aforementioned descriptors. Cell A evolves into cell C, as indicated by similar shapes, despite the fact that cell B is closer to cell A spatially.

Figure 4.1: An example of cell alignment in two consecutive PSC microscopic images. Although cell B is closer to cell A, cell A actually moved and became cell C which is more similar to cell A in shape.

To address this challenge, we propose a new approach called SCOTT (Shape-Location Combined Tracking with Optimal Transport). SCOTT exploits cell detection, but in the alignment phase, pixel-level tracking is performed instead of cell-level tracking. The detected cells impose constraints on the alignment of pixels, and conversely, the result of pixel alignment is used to adjust the detected cells. The motivation for pixel-level tracking is to better preserve the shape of the cells during alignment because the cells will be specified in their original forms, that is, a group of pixels. However, in order to be sensitive to shape, the choice of the optimization criterion for pixel alignment matters. This issue has been addressed in the literature of Optimal Transport (OT) [105, 122].

We specify a cell by a set of 2-D pixel coordinates. The problem of pixel-level tracking is transformed into point-set registration between two consecutive frames, which is a widely studied problem in OT. A commonly used criterion for registering two point sets is to minimize the total transportation cost, often called the Wasserstein Distance (WD) [115], which is a linear cost in terms of the matching weights to be optimized. Point set registration based on WD has been used to solve object tracking [144, 154].
However, if the goal is to best preserve the shape of the point set after the transport, then a quadratic cost should be used, which results in the Gromov-Wasserstein Distance (GWD) [122]. GWD aims at keeping the fidelity of the similarity matrix between points in one set instead of the absolute locations of the points themselves. As a result, GWD tends to keep the geometric relationship between the points in one set. See [122] for the detailed mathematical justification.

In SCOTT, because we need to consider both spatial location similarity and shape similarity between cells, we employ a new transportation cost which is a weighted linear combination of the WD cost and the GWD cost, referred to as weighted Gromov-Wasserstein distance (WGWD). In addition, our problem has another layer of complexity because the pixels are grouped into multiple cells in one image. When the pixels of one cell in an image are mapped to pixels in another image, it is only sensible if the mapped pixels also form a cell. The original WD or GWD problems have no such concern. We have developed a bipartite graph partitioning (BGP) strategy to ensure that the pixels are transported consistently with respect to their memberships to the cells. In addition, BGP adopts a two-stage process, in which OT based on WD is used in the first stage and the computationally more intensive OT based on WGWD is evoked in the second stage only for small groups of cells that are difficult to track.

By establishing a pixel-to-pixel correspondence in microscopy image frames, SCOTT performs better in revealing complex situations, such as division of cells or the appearance of multiple cells merging into one when they are adjacent to or overlapping with each other. For brevity, we use the term “merger” to mean that multiple cells blend into one blob in the image. This technical “artifact” in imaging can last over a widely varying length of time. Merging is not a biological process, while cell division is. As a result, when cells merge, to precisely track them, their individual identities should be kept. Pixel-level registration enables us to detect the case of merging as well as division.

In summary, the major contribution of our proposed method is a new OT-based framework for cell tracking. We have proposed WGWD to combine location and shape information. In addition, we have developed a new algorithm that integrates BGP with OT so that pixel-level registration can be solved more accurately and efficiently for high-resolution microscopic images. Based on the pixel-level information, we can better track cells.
4.2 Weighted Gromov-Wasserstein Distance

As introduced in the Section 4.1, we reexamine this problem in pixel level and use pixel coordinates to represent cells. We denote pixels of the cells specified by their coordinates in the 2D image plane of the former frame by $X_1 = x_{1,i} \in \mathbb{R}^2$, and those of the latter frame by $X_2 = x_{2,i} \in \mathbb{R}^2$. Each pixel has the same weight and the cost matrix is the pairwise Euclidean distance matrix. In our case, we define a 1-Wasserstein distance problem, which is the special case of p-Wasserstein distance introduced in Section 3.1. Besides, we use pairwise distance matrix $C_i \in \mathbb{R}_{+}^{n_i \times n_i}$, $i = 1, 2$ to represent shapes of the data $X_i$, because the $C_i$ is more robust in coordination translation, scaling, and rotation. Then we can define the Gromov Wasserstein distance introduced in Section 3.2.2 under the cell tracking concept.

For cell tracking, similarities in both location and shape are valuable cues to establish the correspondence between cells in multiple image frames. As explained in the previous section, WD minimizes a weighted sum of location differences, while GWD aims at retaining the geometric characteristics of the point set and the geometry are captured in a rich manner by a similarity matrix between the points. We are thus motivated to combine the WD and GWD distances and propose the weighted Gromov-Wasserstein distance (WGWD). Let $\lambda \in [0, 1]$ be the weight to balance the influence of shape versus location, WGWD is defined as:

$$
\text{WGWD} (C_1, C_2, M, a_1, a_2, \lambda) \defeq \min_{T \in U(a_1, a_2)} \langle T, \lambda L (C_1, C_2) \otimes T + (1 - \lambda) M \rangle.
$$

(4.1)

Here WD and GWD can be viewed as special cases of WGWD at $\lambda = 0$ and $\lambda = 1$ respectively.

An example of how the point registration changes with the weight parameter $\lambda$ is shown in Figure 4.2, where we aim to register points in the source to those in the target in the 2-D coordinate. We use the gradually changed color in the source shape to represent the registered locations in the target. As shown, points are registered according to the minimized total spatial distance when $\lambda = 0$, and then the points are registered according to the most geometric similarity when $\lambda = 1$. When $\lambda$ changes from 0 to 1, the shape is becoming increasingly important compared with the location.

Similar with the process to compute WD and GWD, WGWD can be approximated
Figure 4.2: Point registration evolution with the increase of weight parameter $\lambda$. Left: Two pairs of point sets aligned by OT. For the two rectangles in the first row, the gravity center of the source shape has the same position in the image plane as the midpoint on the left side border of the target shape. In the second row, the gravity centers of the source shape and the target shape are located at the same position. Right: Visualization of point registration with the increase of $\lambda$ by its entropy regularization, i.e.,

$$T_{\text{WGWD}, \varepsilon} (C_1, C_2, M, a_1, a_2, \lambda) \overset{\text{def}}{=} \arg \min_{T \in U(a_1, a_2)} \langle T, \lambda L (C_1, C_2) \otimes T + (1 - \lambda) M \rangle - \varepsilon h (T) .$$

(4.2)

We use the projected gradient descent in the Kullback–Leibler (KL) metric space similar to equation (8) in [122] to solve the problem. The iterations are given by:

$$T \leftarrow \text{proj}^{\text{KL}}_{U(a_1, a_2)} \left( T \odot e^{-\tau \nabla K} \right) ,$$

(4.3)

where $a \odot b = (a_i b_i)$ is the element-wise product, $K = \langle T, \lambda L (C_1, C_2) \otimes T + (1 - \lambda) M \rangle - \varepsilon h (T)$, and the KL projector on any matrix $N$ is defined as:

$$\text{proj}^{\text{KL}}_{U(a_1, a_2)} (N) \overset{\text{def}}{=} \arg \min_{T' \in U(a_1, a_2)} \text{KL} (T' \| N) ,$$

(4.4)

According to [155], the KL projection gives the solution to the entropy-regularized Wasserstein Distance $T_{\text{WD}, \varepsilon}$, i.e.,

$$\text{proj}^{\text{KL}}_{U(a_1, a_2)} (N) = T_{\text{WD}, \varepsilon} \left( -\varepsilon \log N, a_1, a_2 \right) .$$

(4.5)

Take the gradient descent on $T$, we have:

$$\nabla K = \nabla \langle T, \lambda L (C_1, C_2) \otimes T + (1 - \lambda) M \rangle - \nabla \varepsilon h (T)$$

(4.6a)
\[
L \left( C_1, C_2 \right) \otimes T + (1 - \lambda) M + \varepsilon \log(T). \quad (4.6b)
\]

Let \( \tau \varepsilon = 1 \) and substitute equation (4.5) and equation (4.6b) into equation (4.3). We get the final iteration:

\[
T \leftarrow T_{WD,\varepsilon} \left( \lambda L \left( C_1, C_2 \right) \otimes T + (1 - \lambda) M, a_1, a_2 \right), \quad (4.7)
\]

which solves the entropy-regularized WGWD in equation (4.2).

### 4.3 SCOTT Cell Tracking System

Figure 4.3: The proposed cell tracking framework based on pixel-level point registration via OT. The top panel illustrates the four steps of cell tracking: cell segmentation, cell alignment by the RBOT algorithm, division detection, refined cell segmentation, and the determination of the entire tracking sequence for each cell. The bottom panel shows the four steps of the RBOT algorithm on two consecutive image frames: initialization, updating edge weights in each cell cluster, updating cell clusters, and computing the total number of cell clusters. The cells and their transportation weights are coded in the bipartite graph with each vertex corresponding to one cell. The RBOT algorithm stops when the number of cell clusters remains unchanged.

Ideally we can view cell tracking as a pixel matching problem formulated by OT.
with hard matching (each pixel is mapped entirely to another pixel) under the coupling constraint that all pixels in one cell are mapped to pixels in another cell. To have feasible solutions, a cell is allowed to drop or expand its pixels in a new image frame. This constrained OT problem is NP-hard [156]. We solve an approximation by combining the following ideas:

- The basic OT problem allows soft matching between pixels and does not assume an equal number of pixels in the cells of the two image frames. In order to obtain hard matching between pixels, that is, each pixel matching entirely with another pixel, virtual pixels/points are added so that the mapping between virtual and actual pixels indicates changes in cell mass.

- To speed up the basic OT, we use multiscale OT as an approximation so that the complexity of matching is reduced to approximately $\tilde{O}(n)$, where $n$ is the number of cell pixels in one image frame and $n$ is large.

- The basic OT does not ensure that all the pixels in one cell match with pixels in another cell. In order to achieve the cell-wise hard matching, that is, to satisfy the cell-wise coupling constraint, we define an approximate cell-wise transportation plan $\tilde{T}(\mathcal{Y})$. First, we compute a cell-wise transportation plan $T(\mathcal{Y})$ by aggregating the pixel-wise transportation plan $T(\mathcal{X})$ (see Section 4.3.1), which is in general not hard matching. Secondly, via thresholding, we obtain hard-matching cell-wise transportation plan $\tilde{T}(\mathcal{Y})$ from $T(\mathcal{Y})$ (see Section 4.3.2).

A schematic diagram for the process of cell tracking by SCOTT is shown in Figure 4.3. SCOTT has the following major steps. First, pixels belonging to cells are segmented in each image frame. These pixels are then divided into connected components, which are treated as segmented cells, but only initially. Precise identification of cells depends on the feedback from later steps. For instance, a connected component may be partitioned into multiple cells. To clarify the terminology, we will call a connected component extracted by initial segmentation a “segment” and a cell identified by SCOTT a “detected cell”. Second, we align cell pixels by OT and apply bipartite graph partitioning (BGP) to find correspondence between groups of clusters. The cascade of OT and BGP is applied recursively until the cell clusters cannot be further divided. Due to intricate scenarios such as cell division or cell merging, some of the cell clusters are subject to another round of OT-based analysis. We refer to this algorithm as Recursive BGP with OT (RBOT). Finally, based on the detected divisions and pixel-level point registration, we refine cell segmentation and obtain the tracked sequence of every cell.
4.3.1 Cell-level Transportation

Although we are motivated to solve the pixel-level registration, as discussed in Section 4.1, we must convert pixel-level registration to cell-level alignment so as to ultimately track cells. Ideally, pixels of the same cell in one image frame are mapped to pixels which also belong to one cell. We call this requirement cell-wise consistency, which is not guaranteed by pixel-level transportation. Frequently, a small portion of one cell is mismatched with part of another cell. Sometimes, the mismatched part comes from a cell located quite far away. This phenomenon seems to be counter-intuitive given that OT attempts to minimize the total transportation cost. This mismatching is a potential consequence of the marginal constraints of OT forcing every pixel to be matched with some pixel in another frame. Therefore, if the size of a cell varies slightly in two frames, the extra pixels in one cell are matched with distant pixels in other cells because there is no nearby pixel left for matching. As a result, cell alignment cannot be easily achieved by mapping the transportation plan from pixel-level to cell-level. We explain our approach to overcoming this difficulty in the next section. Nevertheless, mapping transportation plan across the two levels is a basic technical component in our system, which will be described next.

We adopt the notations introduced in Section 3. For image $i = 1, 2$, let the set of all the cell pixels be $X_i$. Let $n_i = |X_i|$, i.e., the cardinality of set $X_i$. Suppose there are $N_i$ cells in $X_i$. Denote cell $k$, $k = 1, ..., N_i$, by $Y_{i,k}$, which is the set of pixels contained in cell $k$. Thus, $X_i = \bigcup_{k=1}^{N_i} Y_{i,k}$. Let the set of indices $I_{i,k} = \{j | x_{i,j} \in Y_{i,k}\}$. Equivalently, $Y_{i,k} = \{x_{i,j} | j \in I_{i,k}\}$. Denote the pixel-level transportation plan by $T(X) = \left( t^{(X)}_{j,j'} \right) \in \mathbb{R}^{n_1 \times n_2}$ and the cell-level transportation plan by $T(Y) = \left( t^{(Y)}_{k,k'} \right) \in \mathbb{R}^{N_1 \times N_2}$. The intuition is that the matching weight between two cells is the sum of all the matching weights between pixels contained in the two cells. Specifically, $T(Y)$ is defined by

$$t^{(Y)}_{k,k'} \overset{\text{def}}{=} \sum_{j \in I_{0,k}, j' \in I_{1,k'}} t^{(X)}_{j,j'}.$$  \hfill (4.8)

For the convenience of describing the RBOT algorithm, we represent the cell-level alignment by a graph, denoted $G$, in which every node corresponds to one cell $Y_{i,k}$, $i = 0, 1$, $k = 1, ..., N_i$, and the edge weight between $Y_{0,k}$ and $Y_{1,k'}$ is $t^{(Y)}_{k,k'}$. Graph $G$ is bipartite since an edge only exists between cells in different frames.
4.3.2 Cell Alignment by RBOT

To achieve cell-wise consistency in the pixel-level transportation plan, our strategy is to gradually shrink the region of pixels involved in OT, guided by the pixel-level transportation itself. Specifically, we start from the transportation of all the cell pixels. Once we obtain the bipartite graph $G$ that encodes the alignment between cells, we perform subgraph extraction by setting edge weights that are below a threshold to zeros. Each subgraph contains a group of cells in one image frame and another group of cells in the other frame. We call the group of cells in each frame a cell cluster. The two cell clusters in every subgraph are treated as “aligned” and the decision is final. In contrast, the cell-level transportation plan $T^{(Y)}$ is only an intermediate result that leads to the extraction of these subgraphs. Since we need the alignment at the granularity level of cells, each pair of matched cell clusters are subject to further alignment for the cells they contain. We thus apply to the pixels contained in each subgraph the same steps of aligning pixels, assigning edge weights in the bipartite graph for cells, and partitioning the graph to obtain even smaller subgraphs. During this recursive process, the pixel-level transportation plan is solved by OT based on WD. However, after the recursive graph partitioning is completed, for certain difficult types of cell clusters, such as those caused by cell division, OT based on WGWD is applied to each cell cluster. Exploiting OT based on both WD and WGWD allows us to balance complexity and accuracy.

We call the above algorithmic design the bipartite graph partitioning strategy. This method, inspired by Zha et al. [157], aims to cluster data represented by a bipartite graph. By this recursive top-down approach, we shrink the region of pixels involved in transportation step by step. This strategy effectively boosts the consistency in the transportation plan. The approach can also avoid the forced matching of cells that are far apart and improve computational efficiency since OT is conducted on smaller and smaller sets of pixels. We now elaborate on the main components in the RBOT algorithm, namely, Initialization, Recursion, and Interpretation of Graph Partitioning Results. After introducing these components, we will analyze the complexity of the algorithm. At the end of this section, a toy example is provided in Figure 4.4 to illustrate the various aspects of RBOT.

Initialization. We initialize graph $G$ by a bipartite graph where the edge weights are derived from the pixel-level transportation plan with equation (4.8), preserving only the edges with weights greater than a predetermined threshold. Because the scale to compute the pixel-level transportation plan is very large, we adopt the multiscale OT approximation proposed by Gerber and Maggioni [158] as the initialized pixel-level
transportation. The idea of this algorithm is to use a hierarchical approach so that point transportation is restricted within a neighborhood. We denote the transportation plan yield from the multiscale OT approximation method by $T_0^{(X)}$. In addition to fast computation, $T_0^{(X)}$ solved by the multiscale OT restricts the transportation within a neighborhood of a pixel, and hence yields a simpler graph $G$ for the cells. Compared with a fully-connected bipartite graph $G$ with $O(N_1 N_2)$ edges, the graph $G$ constructed from $T_0^{(X)}$ only has $O(N_1 + N_2)$ weight-bearing edges, which is a substantial reduction.

**Recursion.** In RBOT, cells are clustered recursively by extracting disconnected subgraphs with more edges in $G$ removed via thresholding. Initially, $G$ is the only existing subgraph. Each round of recursion aims at dividing every existing subgraph into even smaller subgraphs. In one round, sweeping through every existing subgraph, the following steps are applied:

- **Step 1:** Compute pixel-level transportation plan $T^{(X)}$ for pixels in the subgraph.

- **Step 2:** Compute edge weights in the graph, i.e., the cell-level transportation plan $T^{(Y)}$, from $T^{(X)}$ by equation (4.8).

- **Step 3:** Update cell-wise transportation plan to $\tilde{T}^{(Y)}$ by removing edges with weights below a predetermined threshold $t_0$. Identify the newly-formed disconnected subgraphs. Here the element of the matrix is computed by $\tilde{t}_{ij}^{(Y)} = \left(t_{ij}^{(Y)} > t_0 ? t_{ij}^{(Y)} : 0\right)$ and $t_0$ is the minimal proportion of divided cytoplasm.

The recursion terminates when no subgraph can be further partitioned.

One technical issue in OT as applied to the pixels in any subgraph is that the numbers of cell pixels in the two images are in general different. If we assign the same weight to each pixel in every image, then the total weights for the two sets are not equal. In this case, the problem of point set transportation is called *Unbalanced OT* or *Kantorovich Distance* (KD) problem [104, 159]. On the contrary, if the total weights are equal, we have *Balanced OT*. Note that when the two sets contain different numbers of points, we can construct a Balanced OT problem by maintaining the same total weight while allocating different weights to individual points. With this setup, however, we no longer obtain one-to-one mapping between the points. A point in the smaller set is split and transported to multiple points in the other set. We adopt the method by Gramfort et al. [104] to solve unbalanced OT. The main idea is to add virtual points to the smaller set and define a distance between any actual point and a virtual one.

For example, for a subgraph with $n_1$ and $n_2$ pixels in each image frame, assume without loss of generality $n_1 < n_2$. We add $n_2 - n_1$ virtual points to the source. We need
to extend the distance matrix $M \in \mathbb{R}_{+}^{n_{1} \times n_{2}}$ to $\bar{M} \in \mathbb{R}_{+}^{n_{2} \times n_{2}}$. Let $\Delta \in \mathbb{R}_{+}^{(n_{2} - n_{1}) \times n_{2}}$ be the distance matrix between the virtual points and the actual ones. In order to properly define $\Delta$ to satisfy the requirement of KD, we first solve the Balanced OT problem for the subgraph and obtain the transportation plan $T(X) = (t_{i_{j}}^{(X)})$. Let $\mathcal{I} = \{(j, j') | t_{j_{j}'}^{(X)} > 0\}$. We set $\Delta = \max_{(j, j') \in \mathcal{I}} (m_{j,j'}) \cdot 1_{(n_{2}-n_{1}) \times n_{2}}$ to satisfy the definition of KD (Definition 1 of [104]). Then $\bar{M} = \begin{bmatrix} M \\ \Delta \end{bmatrix}$. Let $\bar{a}_{1} = \frac{1}{n_{2}} 1_{n_{2}}$. Then we have

$$W \left( \bar{M}, \bar{a}_{1}, a_{2} \right) = W \left( \begin{bmatrix} M \\ \Delta \end{bmatrix}, \frac{1}{n_{2}} 1_{n_{2}}, \frac{1}{n_{2}} 1_{n_{2}} \right). \quad (4.9)$$

We then apply Sinkhorn iterations to equation (4.9) to solve the transportation plan $T_{W,\epsilon} \left( \bar{M}, \bar{a}_{1}, a_{2} \right) \in \mathbb{R}_{+}^{n_{2} \times n_{2}}$. Since the last $n_{2} - n_{1}$ rows specify transportation between virtual points and actual points, we only output the first $n_{1}$ rows of $T_{W,\epsilon}$, which is denoted by $T_{W,\epsilon} \left[ 1 : n_{1} \right]$.

**Interpretation of Graph Partitioning Results.** We categorize the cell clusters according to the number of segmented cells in the source and target images, and then code these categories by the format of $\#\text{Source-to-}\#\text{Target}$. We describe the meaning of each category for the sake of obtaining the tracking sequence of each cell. The term “multiple” here means “two or more”.

- **One-to-One** can be the result of two cases. The first is the simplest case when a cell moved without touching other cells. The second is the difficult case when a cell has divided but the new cells do not appear separated in the new frame.

- **One-to-Multiple** is the case when one cell divides into multiple distinct cells.

- **Zero-to-One** is the case when a new cell appears in the observation range. All the pixels of this cell are registered to virtual pixels in the source image.

- **One-to-Zero** is the reverse case of **Zero-to-One** when an existing cell moves out of the observation range. All the pixels of this cell are registered to virtual pixels in the target image.

- **Multiple-to-One** is the case when multiple cells are merged into one segment. In this case, we need to align the pixels with the corresponding cells in the source frame.
• **Multiple-to-Multiple** is the case when the cells are difficult to align. When this case occurs, we iteratively remove the edge with the smallest weight and re-normalize the other edge weights until the graph can be separated into two subgraphs. Applying this operation recursively, **Multiple-to-Multiple** will eventually be reduced to one of the previous cases.

If the cell cluster is of type **One-to-One** or **Multiple-to-One**, we apply OT based on WGWD to the pixels contained in this cluster. For **One-to-One**, this extra point registration is needed to detect cell division, which will be elaborated on in the next subsection. For **Multiple-to-One**, OT based on WGWD provides more accurate registration by combining shape and location information. The complete RBOT algorithm is presented formally in Algorithm 1.

**Algorithm 1** Recursive Bipartite Graph Partitioning with OT (RBOT)

**Input:** two sets of cell pixels $X_1, X_2$, the cells $Y_1, Y_2$.

**Output:** point-level registration $T^{(X)}$ and cell-level registration $T^{(Y)}$.

1. construct a bipartite graph $G$. Compute distance matrix $M$ and similarity matrix $C_1, C_2$.
2. initialize edge weight with multiscale OT approximation $T^{(X)} \leftarrow T_0^{(X)}$ and equation (4.8).
3. repeat
4.    add virtual points and extend the boundary condition $(M, a_1, a_2)$ to $(\tilde{M}, \tilde{a}_1, \tilde{a}_2)$.
5.    $T^{(X)} \leftarrow T_{\text{WD}, \epsilon} \left( \tilde{M}, \tilde{a}_1, \tilde{a}_2 \right) [1 : n_1]$
6.    compute $T^{(Y)}$ with equation (4.8).
7.    compute $\tilde{T}^{(Y)}$ by thresholding $T^{(Y)}$. Remove the corresponding edges in $G$.
8.    partition $G$ into disconnected subgraphs $G = \bigcup_i G_i$.
9.    For each subgraph $G_i$, let $T \leftarrow G_i$ and apply RBOT on each $G$.
10. until $G$ cannot be partitioned.
11. For subgraphs with cell merging and division, $T^{(X)} \leftarrow T_{\text{WGWD}, \epsilon} (C_1, C_2, M, a_1, a_2, \lambda)$.
12. update the matrix $T^{(X)}$ and $T^{(Y)}$ from every subgraph.
13. return $T^{(X)}, T^{(Y)}$.

**Complexity.** High computational complexity is a major concern of OT [119–122]. Assume there are $N$ cells in each frame and $c$ pixels in each cell. Suppose RBOT terminate within constant number recursions (5 in our experiments). Then the time complexity of the whole process is $\tilde{O} (N^3 c^2)$. The computation comprises three parts. First, the multiscale OT approximation has complexity of $\tilde{O} (N^2 c^3)$ [158]. Second, Sinkhorn method with virtual points has complexity $\tilde{O} (N^2 c^2)$ when cells are not fully separate and $\tilde{O} (Nc^2)$ when cells are fully separate [120]. Third, if cell division is detected, the complexity of
computing WGWD within cells that are eventually divided is $\tilde{O}(Nc^3)$ [122]. We note that if WGWD is applied to all the pixels in the frame, the complexity is $\tilde{O}(N^3c^3)$, much higher than that of our algorithm.

![Cells in Two Frames Labeled by A-H](image1)

![Initialization with Multiscale OT](image2)

![After One Round of Recursion](image3)

![After Two Rounds of Recursion](image4)

Figure 4.4: Visualization of cell clusters generated by RBOT. These cell clusters are binary cell masks of two consecutive frames of dataset PSC-Passage7 in [7]. Cells in the same color are from the same frame. Every subfigure corresponds to the result obtained up to a certain step. In each subfigure, the bipartite graph formed from the cell-level transportation plan is shown on the right, while the cell clusters are described below the figure. Top-Left: Two consecutive frames with segmented cells. Top-Right: The initialization of the cell-level transportation plan yielded from multiscale OT. Bottom-Left: The cell-level transportation plan after one round of recursion in RBOT. Bottom-Right: The cell-level transportation plan after two rounds of recursion in RBOT.

**Example.** Figure 4.4 shows an example of cell alignment by the RBOT algorithm. In each subfigure, the picture on the left shows the segmented cells and the cell-level transportation plan, while to the right of the picture, the bipartite graph for cell alignment is shown. The cell clusters extracted so far are indicated by the text below the picture. We use the blue letters to list the cells in the source, the red letters for cells in the target,
and the green lines to show the cell-level transportation plan. In this simple example, we can easily align the cells according to their spatial closeness. We use letters A-H to mark each pair of correctly aligned cell clusters, as shown in the top-left figure. Initially, we compute the pixel-level alignment with multiscale OT approximation and compute the cell-level OT alignment with equation (4.8). Without imposing cell-wise consistency (see Section 4.3.1), we obtain long-distance mapping between some cells, e.g., F-G in the upper-left figure. This kind of error in mapping is caused by the marginal constraints of traditional OT, while our RBOT algorithm is designed to address this issue. After one recursion, as shown in the upper-right figure, many edges have been removed based on the pixel-level registration computed by multi-scale OT and equation (4.8). Cells are now clustered into three disconnected subgraphs: the cell pairs A to E, the cell pairs F and G, and the cell pair H, as shown in the top-right figure. In two figures of the bottom, after more recursions have been completed, the cells are grouped into smaller clusters. Finally, as shown in the bottom-right figure, every disconnected cell cluster now contains only a pair of cells from the source and the target, and hence cannot be further partitioned. The RBOT algorithm terminates and produces the cell-level transportation plan, as shown in the bottom-right figure. We provide more cell alignment examples computed with RBOT algorithm, which can be found in Section 2 of the supplementary material.

4.3.3 Cell Division Detection

In cell biology, cell division usually refers to two processes: mitosis and cytokinesis. At the mitosis stage, the chromosomes of a single cell replicate and separate into two new nuclei. At the cytokinesis stage which follows mitosis, the cytoplasm of a single cell separates into two parts. In some literature [160], cell division is considered to be marked by the end of mitosis or the beginning of cytokinesis, which is also the notion of cell division adopted in the Cell Tracking Challenges where we obtained the data. For comparison with existing work, we also take the beginning of cytokinesis as the biological definition of cell division in this paper.

SCOTT can detect cell division by finding the cell division trend from the pixel-wise alignment in WGWD. First, we assume that all the cells are alive and there is no cell fusion phenomenon where multiple cells appear to merge into one component in the image. In addition, SCOTT is only effective under the following conditions: (1) The cell did not have a contraction in shape during division, as often happens with DNA-stained
Figure 4.5: Pixel motion vectors at cell division. To illustrate the motion vectors, their directions are coded by the color hue, and their magnitudes are coded by saturation. A more saturated color means a higher magnitude. The motion vector is the difference between the coordinates of the registered pixels in the target and source images. *Top-Left:* Cells in division with pixels moving in opposite horizontal directions. *Bottom-Left:* Cells in division with pixels moving in opposite vertical directions. The motion vectors are computed based on the pixel-level transportation plan yielded from WGWD at $\lambda = 0.90$. *Right:* The color palette to code the motion vectors in [8].

cells. (2) The chromosome and cytoplasm of the two daughter cells move in opposite directions during mitosis. (3) The sizes of the separated cytoplasm of the two daughter cells are almost equal; and the separated cells do not have cell walls.

Detecting cell division from 2D images is challenging because cell division is a continuous process with a time span varying between different types of cells or even individual cells. Sometimes cytokinesis finishes so quickly that we can observe one single cell at time $t$ dividing into two disconnected cells at time $t + 1$. In this case, RBOT can capture the division by recognizing that a cell cluster belongs to the **One-to-Two** category. In another case, cytokinesis may take a long stretch of time, and the cytoplasm of the two daughter cells appears to be a single segment over many frames. It is then difficult to quickly detect the division without adequate prior knowledge about the shape of the cells.

We now explain how to detect cell division by computing WGWD. Based on the pixel-level transportation plan, we propose a novel approach to detect cell division before the spatial separation of the daughter cells is shown in the image. Importantly, our approach is not based on cell-level geometric features such as contours, which often vary with the types of cells and are not necessarily straightforward to quantify. As a result, our approach is general and easy to apply. Our basic idea is to compute the motion of individual pixels based on their transportation plan and then examine the uniformity of the motion vectors. When a cell divides, the pixels are likely to move in significantly
different directions. On the contrary, pixels in the same cell normally move in similar
directions. As shown in Figure 4.5, pixels in one cell exhibit a strong pattern of moving
in two opposite directions at the moment of division. We found that this pattern of pixel
movement is more evident when the transportation is solved by WGWD than that by
WD or GWD. This observation reflects that both location and shape information are
valuable for establishing the correspondence between pixels.

Let $x_{1,k}$ be any pixel in the source image, and denote its registered pixel in the
target by $\pi(x_{1,k})$. Then the motion vector of $x_{1,k}$ is defined by $\pi(x_{1,k}) - x_{1,k}$, which is
represented by its polar coordinates $(r_k, \theta_k)$: $r_k e^{i\theta_k} = \pi(x_{1,k}) - x_{1,k}$. Since a cell usually
divides into exactly two new cells [161], we are motivated to estimate two main movement
directions $\hat{\theta}_1$ and $\hat{\theta}_2$ by fitting a two-component Gaussian mixture model on all the angles
$\theta_k$. Denote the density of the $i$th, $i = 1, 2$, Gaussian component by $\hat{\varphi}_i = \varphi(\hat{\theta}_i, \hat{\sigma}_i^2)$, where
$\hat{\theta}_i$ is the mean and $\hat{\sigma}_i^2$ is the variance. Using the typical mixture model-based clustering,
the angles $\theta_k$’s are clustered into two groups. This result determines the grouping of the
pixels $x_{1,k}$. Denote the two pixel groups by $Z_1^{(i)}$, $i = 1, 2$.

For pixels $x_{1,k}$ with non-zero $r_k$ (motion vector magnitude), we compute the proportion
$\hat{q}_i$ of each pixel group $Z_1^{(i)}$, $i = 1, 2$. We have $\hat{q}_1 + \hat{q}_2 = 1$. There three conditions to
satisfy in order to declare cell division:

1. The difference between the two main movement angles, denoted $\Delta \theta$, is larger than
   a threshold. Note that
   \[ \Delta \theta \overset{\text{def}}{=} \min \left( |\hat{\theta}_1 - \hat{\theta}_2|, 2\pi - |\hat{\theta}_1 - \hat{\theta}_2| \right). \tag{4.10} \]

2. The standard deviation of the two fitted Gaussian components $\hat{\sigma}_i$ is less than a
   threshold.

3. The proportions $\hat{q}_i$ of $Z_1^{(i)}$, $i = 1, 2$ are both larger than a threshold. $i = 1, 2$ are
   smaller than a threshold.

Conditions (1) and (2) ensure that there are two sufficiently distinct moving directions
relative to the variation within each group. Condition (3) ensures that there are a
sufficient number of pixels moving in each direction. As shown by Table 4.2 (see the
column $Div.$), our method captures cell division more accurately when the two daughter
cells have not appeared as separate segments in the image.
4.4 Experiments

We have implemented the whole pipeline shown in Figure 4.3 in Python 3, a part of the codes are provided by [158, 162], specifically, the implementation of multiscale OT and the WD/GWD. We conducted the experiments on Amazon Web Service instance EC2-C5.12Xlarge, which has 48 virtual Intel Xeon Scalable Processors CPUs with 96GiB memory. In the current implementation, approximately 30% of the computation is parallelized, mainly in the computation of OT. It takes 10-800 seconds for our algorithm to process a single frame, and the exact speed depends on the image size and the average number of pixels in one cell. Our code can be improved in computation efficiency by a more effective implementation, e.g., in C++ or by further parallelization. The processing time can also be reduced substantially by reducing the resolution of the images (that is, to shrink the image size), but this reduced resolution also compromises accuracy. The processing time of each dataset and the effect of reducing image size can be found in the supplementary materials. In our experiments, we set the minimal proportion of cell division \( t_0 = 0.1 \). In cell division detection, we set the cell division criteria to be \( \Delta \theta \geq 0.60, \hat{\sigma}_i \leq 0.12, \) and \( \hat{q}_i \geq 0.30 \) for \( i = 0, 1 \). The thresholds we used for \( \Delta \theta, \hat{\sigma}_i, \) and \( \hat{q}_i \) were chosen by grid search. We applied SCOTT using different combinations of threshold values to the first ten frames in the sequences and evaluated the results using the ground truth. The values yielding the best accuracy are used in later experiments. For new datasets, we recommend searching within these ranges: \( \Delta \theta_{\text{min}} \in [30^\circ, 120^\circ], \hat{\sigma}_{\text{max}} \in [0.1, 0.5], \) and \( \hat{q}_{\text{min}} \in [0.2, 0.4] \).

4.4.1 Test Sequences

We use seven sequences with annotations from four open-source datasets to test our algorithm. They are:

**DIC-HeLa** consists of two videos of HeLa cells on a flat glass (denote as DIC-HeLa-01 and DIC-HeLa-02). The initial segmentation yields 15 segments on average over 84 image frames of size 512 × 512. The dataset has a high cell density and a low signal-to-noise ratio of 0.74 [41].

**PhC-U373** consists of two videos of Glioblastoma-astrocytoma U373 cells on a polyacrylamide substrate (denote as PhC-U373-01 and PhC-U373-02). The initial segmentation yields about 8 segments on average over 115 image frames of size 520 × 696.

The code for our experiments can be found in https://github.com/RachelZheng/scott.
PhC-PSC consists of two time series observations of PSC on a Polystyrene substrate (denote as PhC-PSC-01 and PhC-PSC-02). Only the 150th - 250th frames are used in the experiment because they have ground-truth labeling. The initial segmentation yields about 300 segments on average over the 101 image frames of size 576 × 720. The dataset also has a high cell density and a low signal-to-noise ratio. All three aforementioned datasets are collected from Cell Tracking Challenges [14].

PSC-Passage7 is one image time series of PSC observations collected by [7]. They are proliferating PSC observations in different shapes during a time-lapse experiment with oblique illumination. The data set contains images taken every ten minutes with a total of 399 frames of size 1038 × 1376 and the human-labeled ground-truth is available.

4.4.2 Baselines

SCOTT is compared with the following two state-of-the-art automatic cell tracking methods.

Validation with Tracking (VT) [7]. The code is publicly available online. We only use the tracking function of the codes. We use the default parameters minPathLength=30 and minOverlap=0.70 for PhC-PSC and PSC-Passage7, and minPathLength=10 and minOverlap=0.50 for DIC-HeLa and PhC-U373 to get the best results.

Conservation Tracking (CT) [145]. We ran Ilastik V1.3.2 [163] that implements the method by Schiegg et al. [145]. This method applies the probabilistic graphical model to track the cells, which can effectively handle the over-segmentation and under-segmentation of the cells. Since CT algorithm requires the initial segmentation, we manually labeled the cells in the first frame. In addition, to train the cell division detectors, we manually labeled the first five cell division cases in each image time series.

Because segmenting cells from the background is not the focus of this paper, we adopt the cell segmentation methods proposed by Sixta et al. [164] for datasets DIC-HeLa, PhC-U373, PhC-PSC, and Rapoport et al. [7] for dataset PSC-Passage7. Cell segmentation is performed independently on every image frame. Each connected component of the cell pixels is treated as one segmented cell. For multiple cells that overlap in the image, they will be marked as one cell. Starting from this initial cell segmentation, we then apply the SCOTT tracking system.
4.4.3 Evaluation Measures

We use precision, recall and F-measure (the harmonic mean of precision and recall) to evaluate each algorithm’s ability to detect cell migration, division and merger. The cell activities labeled in the ground-truth are called correct activities, and the cell activities output by the tracking algorithm are called captured activities. Then the precision and recall are defined as:

\[
\text{precision} = \frac{\#(\text{correct activities} \cap \text{captured activities})}{\#(\text{correct activities})},
\]

\[
\text{recall} = \frac{\#(\text{correct activities} \cap \text{captured activities})}{\#(\text{captured activities})}.
\]

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Move</th>
<th>MOTA</th>
<th>TRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIC-HeLa-01</td>
<td>Prec.</td>
<td>Rec.</td>
<td>F-M.</td>
</tr>
<tr>
<td>VT</td>
<td>0.70</td>
<td>0.11</td>
<td>0.20</td>
</tr>
<tr>
<td>CT</td>
<td>0.79</td>
<td>0.24</td>
<td>0.37</td>
</tr>
<tr>
<td>Ours(NoDiv)</td>
<td>0.91</td>
<td>0.48</td>
<td>0.63</td>
</tr>
<tr>
<td>DIC-HeLa-02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>0.44</td>
<td>0.06</td>
<td>0.11</td>
</tr>
<tr>
<td>CT</td>
<td>0.68</td>
<td>0.26</td>
<td>0.38</td>
</tr>
<tr>
<td>Ours(NoDiv)</td>
<td>0.75</td>
<td>0.77</td>
<td>0.76</td>
</tr>
<tr>
<td>PhC-U373-01</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>0.79</td>
<td>0.80</td>
<td>0.79</td>
</tr>
<tr>
<td>CT</td>
<td>0.80</td>
<td>0.78</td>
<td>0.79</td>
</tr>
<tr>
<td>Ours(NoDiv)</td>
<td>0.93</td>
<td>0.83</td>
<td>0.88</td>
</tr>
<tr>
<td>PhC-U373-02</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>0.66</td>
<td>0.70</td>
<td>0.68</td>
</tr>
<tr>
<td>CT</td>
<td>0.77</td>
<td>0.75</td>
<td>0.76</td>
</tr>
<tr>
<td>Ours(NoDiv)</td>
<td>0.87</td>
<td>0.85</td>
<td>0.86</td>
</tr>
<tr>
<td>PSC-Passage7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>0.98</td>
<td>0.92</td>
<td>0.95</td>
</tr>
<tr>
<td>CT</td>
<td>0.97</td>
<td>0.88</td>
<td>0.92</td>
</tr>
<tr>
<td>Ours(NoDiv)</td>
<td>0.98</td>
<td>0.88</td>
<td>0.93</td>
</tr>
</tbody>
</table>

Table 4.1: Tracking accuracy for cell migration based on every two consecutive frames in datasets DIC-HeLa, PhC-U373, and PSC-Passage7.

The activities are identified based on every two consecutive image frames on initial cell segments. This evaluation method is also adopted in the literature [142, 145]. Here the term “merger” means multiple cells are in one connected component in the segmentation map, one of the under-segmentation cases discussed by Schiegg et al. [145]. Because
VT does not attempt to separate merged cells, the ground-truth labels in dataset PSC-Passage7 do not separate merged cells. We thus cannot evaluate the detection of this type of activity. In Table 4.2, the measures for the “merger” case are marked N/A. The cell is correctly-tracked if a value larger than 0.5 is obtained for the ratio between the intersection and the union of the detected cell and the ground-truth cell.

In addition to precision, recall, and F-measurement, we also evaluate the results by the multiple object tracking accuracy (MOTA) [165] and the tracking accuracy (TRA) based on Acyclic Oriented Graph Matching (AOGM) measure [14]. MOTA is used for multiple objects tracking challenges [166]. To compute MOTA, we first use the nearest neighbor rule to map a detected object to a labeled object in the ground truth. The center pixel location of an object is taken as the position of the whole object. MOTA combines the ratios of objects that are undetected, incorrectly detected, and mismatched. TRA is a scaled version of AOGM. The idea of AOGM is to represent the tracking result as a graph, in which every node represents a detected cell and every edge represents tracking of the same cell in two consecutive frames. The misalignment, including undetected cell division, incorrectly aligned cells, and errors in cell segmentation are assigned with different error weights and penalized in the final accuracy scores. For some datasets like PhC-PSC, whose ground truth are labeled as dots rather than the whole segment, we annotate cell segments with the nearest annotation as the ground truth. All the aforementioned accuracy measures are within the range [0,1] and a higher value of any measurement indicates better performance.

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Move</th>
<th>Merger</th>
<th>Div.</th>
<th>MOTA</th>
<th>TRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhC-PSC-01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>0.85</td>
<td>0.95</td>
<td>0.90</td>
<td>N/A</td>
<td>0.23</td>
</tr>
<tr>
<td>CT</td>
<td>0.82</td>
<td>0.90</td>
<td>0.85</td>
<td>0.78</td>
<td>0.89</td>
</tr>
<tr>
<td>Ours(λ = 0)</td>
<td>0.84</td>
<td>0.94</td>
<td>0.89</td>
<td>0.79</td>
<td>0.99</td>
</tr>
<tr>
<td>Ours(λ = 0.1)</td>
<td>0.84</td>
<td>0.95</td>
<td>0.89</td>
<td>0.80</td>
<td>0.99</td>
</tr>
<tr>
<td>Ours(λ = 0.3)</td>
<td>0.84</td>
<td>0.95</td>
<td>0.89</td>
<td>0.81</td>
<td>0.99</td>
</tr>
<tr>
<td>Ours(λ = 0.5)</td>
<td>0.84</td>
<td>0.94</td>
<td>0.89</td>
<td>0.80</td>
<td>0.95</td>
</tr>
<tr>
<td>Ours(λ = 1)</td>
<td>0.83</td>
<td>0.90</td>
<td>0.87</td>
<td>0.78</td>
<td>0.90</td>
</tr>
<tr>
<td>PhC-PSC-02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VT</td>
<td>0.87</td>
<td>0.96</td>
<td>0.92</td>
<td>N/A</td>
<td>0.18</td>
</tr>
<tr>
<td>CT</td>
<td>0.86</td>
<td>0.99</td>
<td>0.92</td>
<td>0.65</td>
<td>0.80</td>
</tr>
<tr>
<td>Ours(λ = 0)</td>
<td>0.88</td>
<td>0.93</td>
<td>0.90</td>
<td>0.85</td>
<td>0.99</td>
</tr>
<tr>
<td>Ours(λ = 0.1)</td>
<td>0.89</td>
<td>0.93</td>
<td>0.91</td>
<td>0.85</td>
<td>0.99</td>
</tr>
<tr>
<td>Ours(λ = 0.3)</td>
<td>0.88</td>
<td>0.93</td>
<td>0.90</td>
<td>0.85</td>
<td>0.99</td>
</tr>
<tr>
<td>Ours(λ = 0.5)</td>
<td>0.88</td>
<td>0.92</td>
<td>0.89</td>
<td>0.84</td>
<td>0.97</td>
</tr>
<tr>
<td>Ours(λ = 1)</td>
<td>0.88</td>
<td>0.90</td>
<td>0.89</td>
<td>0.83</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 4.2: Tracking accuracy for different cell activities based on every two consecutive frames in dataset PhC-PSC.
Figure 4.6: Comparison of tracking results on dataset PhC-PSC-01 at time $T = 192 - 196$. From top to bottom, Row 1: microscopy frames. The red bounding box at $T = 193$ shows a cell division. Row 2: Cell tracking results with VT. Row 3: Cell tracking results with CT. Row 4: Cell tracking results with SCOTT at $\lambda = 0.1$. Row 5: Ground-truth labeling. One random color marks one cell in the time series.

### 4.4.4 Comparison with Baselines

Datasets DIC-HeLa and PhC-U373 do not follow our assumptions about cell division and have a negligible number of cell merger cases. In dataset PSC-Passage7, cells that merge and appear as one segment are not labeled as separate cells in the manually generated ground truth. We thus apply SCOTT without division detection to these three datasets. The result is provided in Table 4.1. For dataset PhC-PSC, we apply SCOTT with cell division detection, and the result is provided in Table 4.2.

Table 4.1 shows that SCOTT performs the best for aligning cells that only have the
migration activity. Compared with the two baseline methods, for DIC-HeLa, SCOTT achieves higher values for MOTA and TRA, frame-wise tracking recall, and precision. Cells in DIC-HeLa vary substantially in size and have a high density, while the two baseline methods assume that the cell sizes are similar and cells separate with enough in-between space. As a result, SCOTT outperforms the baseline methods. For PhC-U373 and PSC-Passage7, SCOTT is only slightly better than the two baseline methods.

Table 4.2 shows that SCOTT performs better in detecting cell division and solving cell merger cases. On the other hand, SCOTT achieves comparable accuracy in cell movement. To be more specific, in terms of the F-measurement for cell division, the accuracy of SCOTT is about 20% higher than VT, and about 5% higher than CT. SCOTT significantly increases the recall of cell division and cell merger, which is the reason for the increase of the F-measurement.

In terms of MOTA and TRA, SCOTT achieves higher accuracy than the baseline methods, especially for the more complicated dataset PhC-PSC. When parameter $\lambda$ varies from 0 to 1, MOTA and TRA obtained by SCOTT first increases and then decreases. This observation indicates that the WGWD distance has advantages over WD by taking into account shape information. However, the improvement demonstrated by PhC-PSC is moderate. Specifically, when $0.1 < \lambda < 0.3$, the best MOTA and TRA are obtained, but only slightly better than that at $\lambda = 0$. When $\lambda > 0.5$, SCOTT over detects cell divisions, resulting in a drop in MOTA and TRA. The overall gain from shape information is limited because the datasets we have experimented with do not contain many cell divisions. Whether we can obtain more impressive results from WGWD in comparison with WD on other datasets is an interesting question for future investigation.

Figure 4.6 shows an example demonstrating that our SCOTT method works better to detect cell division and merging. Here one random color marks one cell. As we can see, our method separates merged cells well, in a way consistent with both the ground truth and our visual perception. In addition, as shown in the red bounding box at $T = 193$, Row 1, one cell divides into two which are still merged. Only our method detects the division right away. In comparison, the two baseline methods fail to recognize the emergence of the two cells until they are completely separated.

Figure 4.7 shows a few failure cases of SCOTT. The first row contains three original images. Cells that are not accurately identified in the image below are highlighted by a red box. The second row shows the corresponding cell tracking results. For the image in the left column, SCOTT detects two cells for a group of three cells; for the image in the middle column, SCOTT does not separate two cells with the correct boundary;
for the image in the right column, part of a cell is incorrectly labeled as a portion of another cell located right above it. It is possible to improve the algorithm by exploiting the characteristics of cell boundaries in more sophisticated ways.

4.5 Summary and Discussion

In summary, we present a novel cell tracking system called SCOTT, which is based on pixel-level registration via OT. Experiments show that SCOTT performs better in the detection of cell division and cell merging than two other cell tracking algorithms can. In future work, we will extend our approach to 3D+t microscopy images. Applying machine learning to determine “cell division or not” could yield a more accurate classification than manually selecting the thresholds. When more labeled data is available, this can be an interesting direction to explore. Next, we will discuss some limitations of the method and future work in the following.

4.5.1 Effect of Initial Segmentation

Tracking results by our system rely heavily on the initial cell segment. To measure the tracking accuracy, AOGM contains terms corresponding directly to the number of cells that are not detected or that of non-existing cells that are falsely detected. Furthermore, segmentation can affect the calculation of changes in cell mass, a quantity used for detecting cell division. Consequently, we need to start with sufficiently accurate initial cell segmentation, which we assume is achievable by an existing method from the literature.
To evaluate the quality of segmentation, we use reference sequences (sequences with cells and the tracking of them manually labeled) as the gold standard and compute a level of agreement between the reference and the segmentation. The segmentation method we used can capture on average $\sim 95\%$ of cells in the reference sequences with which we experimented with.

4.5.2 Extension to 3D Space

The point alignment method can be easily adapted to 3D if we replace 2D coordinates by 3D coordinates. In practice, however, 3D images pose a few additional challenges. First, the increased amount of pixels in 3D images (that is, a stack of 2D images) require more computation for our algorithm. Second, to fully take advantage of the 3D information, the segmentation algorithm will become more complex. Third, as more cell activities are observed in the 3D space, we need to consider movement and division along the direction orthogonal to the 2D image plane.
Chapter 5  
Pattern Matching Application 2:  
Image Recoloring on Gaussian Mixture Models

In this chapter, we aim to solve the irregularity problem of discrete OT from the statistical perspective. We propose a new parametric formula of OT under the setting of GMM, namely Gaussian-regularized Transport (GRT). The novel GRT formula takes advantage of both the ability to describe different distributions in GMM structure and the flexibility of discrete OT solutions.

The proposed GRT algorithm exhibits excellent abilities to overcome the artifact in the color transfer application. Color transfer is an essential task in image color editing. It refers to transferring the pixel colors in one image (original) to match the colors in another (target). It is desired that the transferred colors are similar to those in the target, while artifacts that make the image look unnatural need to be avoided. This technique is widely used in scientific image processing and can conclude the contrast adjustment on grayscale images and image recoloring on colored images. In the following of this chapter, we first introduce GRT from the mathematical perspective and then detail the process to apply this method to the color transfer problem.
5.1 Problem Introduction

Color transfer, a long-standing problem, aims at imposing the color palette of a source image on a target image while preserving other characteristics of the target. This technique has a wide range of applications including color tone mapping for paintings, adjustment of illumination conditions for photos, and video restoration [167–169]. Most color transfer methods seek for a transportation plan of colors to best match the color distributions of the two images, and OT has been used to accomplish the goal [170,171].

OT is an optimization problem with the objective to minimize the transportation cost from a source distribution to a target, a byproduct of which is the optimal transportation plan between two distributions. OT has been widely used in image color modification tasks in the past decade [6,9,11,13]. OT is appealing for color transfer for multiple reasons. First, OT computes a true metric (distance) between color distributions, and it has been argued that the distance agrees well with human visual perception [107]. Second, the transportation plan between continuous probability distributions computed by OT is proved to be monotonic [172], a desirable property for human visual perception [171,173,174].

However, instead of continuous distributions, OT for discretized color distributions (e.g., histograms) is used in practice. Hence, monotonicity is not guaranteed. Moreover, directly applying OT for color transfer often generates artifacts in transferred images, for instance, errors due to JPEG compression becoming much more noticeable, salt-and-pepper noise, and loss of details [175,176]. Similar artifacts also tend to appear for other methods that aim at matching the transferred color distribution with the target, not necessarily OT-based. The reason is that OT-based transfers are subject to marginal constraints determined by the original and target color distributions, which can be too rigid. For example, if one color in the target image has a high proportion, multiple drastically different colors in the source may be forced to transfer to that color, resulting in artifacts (see detailed discussion in [12]). Remedies have been proposed, e.g., relaxation with mass constraints [12] and keeping local color consistency within superpixels [13]. However, it is difficult to determine the degree of relaxation in OT, and the revised methods cannot adequately address the drawbacks of OT.

We propose a new optimization framework, named the Gaussian-component Regularized Transport (GRT). Specifically, we model the color distribution of each image by a Gaussian Mixture Model (GMM). The OT between two Gaussian components is given by an affine transform [128]. The affine transforms between all the pairs of
components across two images form a basis. Instead of directly taking the OT between a pair of Gaussian components as the final transform, we define the transform between the pair as a linear combination of the basis. This approach provides flexibility for the component-wise transform while keeping component-wise coherence. This formulation requires the solution of a four-way tensor, which is obtained by optimization. The transform of a single point in the source to a point in the target is determined by an equation that depends on the component-wise transforms and the posterior probabilities of the point belonging to any component.

5.2 Gaussian-regularized Transport (GRT)

In this section, we introduce GRT, which is a framework solving the point-wise matching between two datasets $X_1 \in \mathbb{R}^{N_1 \times D}$ (referred to as “source”) and $X_2 \in \mathbb{R}^{N_2 \times D}$ (referred to as “target”). The detailed mathematical setup can be found in the previous Section 3.1. Compared with the original setup that mappings are unconstrained, GRT aims to constraint the transportation plan within the Gaussian form.

The two datasets modeled under GMM assumptions are denoted by $\mathcal{M}_k$. Assume $\mathcal{M}_k$ has $n_k$ components with prior probabilities $\pi_{k,i}$, $i = 1, ..., n_k$. Then its density is $\sum_{i=1}^{n_k} \pi_{k,i} \phi_{k,i}(x)$. Here, $\phi_{k,i}(x)$ is the Gaussian density $\mathcal{N}(\mu_{k,i}, \Sigma_{k,i})$ with $\mu_{k,i}$ being the mean and $\Sigma_{k,i}$ the covariance matrix. We want to solve the point-wise transport function $F : X_1 \rightarrow X_2$ under this setup.

5.2.1 GRT formula

Before we come to the formula of GRT, we first introduce some critical components of it in the following definitions.

**Definition 4 (Affine Map Between Two Gaussians)** Given two Gaussian distributions $\phi_{1,i}$ and $\phi_{2,j}$ where $\phi_{k,l}$ is $\mathcal{N}(\mu_{k,l}, \Sigma_{k,l})$ and a sample point $x \sim \phi_{1,i}$. Let $U \Lambda^2 U^T$ be the spectral decomposition of $\Sigma_{2,j}$. Let $B = U \Lambda$ and $A = B \left( B^T \Sigma_{1,i} B \right)^{-\frac{1}{2}} B^T$. The affine map $f_{i,j}(x) = A (x - \mu_{1,i}) + \mu_{2,j}$ satisfies $f_{i,j}(x) \sim \phi_{2,j}$.

The definition can be directly deduced from Definition 3. Under our assumption, the set of affine maps $\{f_{i,j}\}$ transfers $\mathcal{M}_1$ to $\mathcal{M}_2$. We call the set of affine maps $\mathcal{T} \triangleq \{f_{i,j}\}$ the transport basis, where $i = 1, n_1$, $j = 1, n_2$. 

71
Definition 5 (Posterior Probability for Transportation) The posterior probability of \( x \) belonging to a Gaussian component \( \phi_{k,i} \) is:

\[
p_{k,i}(x) \propto \pi_{k,i} \exp \left[ -\frac{1}{2} (x - \mu_{k,i})^T \Sigma_{k,i}^{-1} (x - \mu_{k,i}) \right],
\]

with \( \sum_{i=1}^{n_k} p_{k,i}(x) = 1. \)

Recall that the point-wise transport function is \( F \). We define a level of dual-component association for a point \( x \) with a pair of Gaussian components \( (\phi_{1,i}, \phi_{2,j}) \) by \( g_{i,j}(x; F) \):

\[
g_{i,j}(x; F) \triangleq p_{1,i}(x) p_{2,j}(F(x)).
\]

Figure 5.1: A simple example of Definitions 4 and 5. \( \mathcal{M}_1 \) is composed of purple, blue and green Gaussian components, and \( \mathcal{M}_2 \) is composed of yellow, orange and red. Left: affine map in Definition 4. Right: posterior probability map in Definition 5.

Figure 5.1 provides an intuitive example of an affine map and posterior probability map. The affine map on the left builds up the pair-wise mapping between Gaussian components, which is considered to encode the global distribution information. In comparison, the probability map on the right computes the posterior probability according to each data point, which can be considered to encode the individual data point information.

To motivate our formulation, let us first consider a relatively simple and intuitive approach. Suppose \( \tilde{S} \triangleq (\tilde{S}_{i,j}) \in \mathbb{R}_{+}^{n_1 \times n_2} \) contains the matching weights between any pair of components in \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \). For example, if the two GMMs have the same number of components which are matched by a permutation, then \( \tilde{S} \) is a permutation matrix containing precisely one 1 in each row (or column) and the rest are zeros. Given the transport basis \( \mathcal{T} \) for all the pairs of components, we can define the point-wise transport \( F \) as follows.
Definition 6 (Elementary Point Transport Formula) Given a point \( x \in \Omega \) and two GMMs \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \), the transport function \( \mathcal{F} \) is given by:

\[
\mathcal{F}(x; \tilde{S}) = \sum_{i,j} \tilde{S}_{i,j} g_{i,j}(x; \mathcal{F}) f_{i,j}(x). \tag{5.3}
\]

Heuristically, \( \tilde{S}_{i,j} \) is the weight assigned to the component pair \((\phi_{1,i}, \phi_{2,j})\), while \( g_{i,j}(x; \mathcal{F}) \) is the level of association between the transported pair of points \((x; \mathcal{F})\) and the pair of components \((\phi_{1,i}, \phi_{2,j})\). As a result, \( \tilde{S}_{i,j} g_{i,j}(x; \mathcal{F}) \) is taken as the point-wise combined weight for the transport function \( f_{i,j}(x) \). Finally, we simply define the point-wise transport function as a weighted sum of the transport basis \( \{f_{i,j}\} \).

Following the terminology in [172,173], transfer \( \mathcal{F} \) is monotonic if inner product \( \langle x_1 - x_2, \mathcal{F}(x_1) - \mathcal{F}(x_2) \rangle \geq 0 \) holds for any two points \( x_1 \) and \( x_2 \). It is straightforward to prove that \( f_{i,j} \) is a monotonic function. Assuming that \( g_{i,j} \) is smooth, thus roughly constant within a small neighbourhood, the transport function in Equation (5.3) is locally monotonic.

One important limitation of the elementary formulation is that each pair of Gaussian components is treated separately from the other pairs in terms of their transport plan. This formulation is restrictive for achieving a good match between the two global color distributions. In addition, the transport direction \( f_{i,j} \) is not necessarily bounded up with the posterior \( g_{i,j} \). Thus we are motivated to add information of GMM structure into the transport formula, and also give more freedom to the weight parameters. Our strategy is to treat \( \mathcal{T} \) as a basis and allow the actual transport between two Gaussian components to be a linear combination of the basis. The linear weights are solved by an optimization formula. These linear weights arrange into a four-way tensor \( S \triangleq (S_{i,j,i',j'}) \in \mathbb{R}^{n_1 \times n_2 \times n_1 \times n_2} \) because each component pair \((i,j)\) has a set of weights over all the pairs \((i',j')\). We call the transportation formula determined by Equation (5.4) Gaussian-component Regularized Transport (GRT).

Definition 7 (GRT) Given a point \( x \in \Omega \) and two GMMs \( \mathcal{M}_1 \) and \( \mathcal{M}_2 \), the transport function \( \mathcal{F} \) is given by:

\[
\mathcal{F}(x; S) = \sum_{i',j'} \left( \sum_{i,j} S_{i,j,i',j'} g_{i,j}(x; \mathcal{F}) \right) f_{i',j'}(x), \tag{5.4}
\]
We illustrate the differences between Elementary Point Transfer and GRT using an example in Fig. 5.2. We expand the point-wise weight of \( f_{1,1} \) at the bottom of the figures and use colors to indicate the Gaussian component associated with the weight items. In Elementary Point Transfer (left subfigure), the weight only changes with the posterior probability on Gaussian pairs \( \{\phi_{1,1}, \phi_{2,1}\} \). But in GRT (right subfigure), the weight varies with posterior probabilities on every two pairs of Gaussian components in \( \{\phi_{1,i}, \phi_{2,j}, \phi_{1,2}, \phi_{2,2}\} \), providing more freedom in the transport.

\[5.2.2 \text{ Iterative Computational Process to Solve GRT} \]

There are two technical difficulties in computing the GRT transport in Equation (5.4). Firstly, the tensor \( \mathbf{S} \) is unknown and must be optimized. Second, \( \mathcal{F}(\mathbf{x}; \mathbf{S}) \) appears on both sides of the equation and must be numerically solved. To tackle the first difficulty, we use an iterative scheme inspired by the Iterative Closest Point algorithm (ICP) [177]. We estimate \( \mathbf{S} \) given an initialization of \( \mathcal{F}(\mathbf{x}; \mathbf{S}) \), where \( \mathbf{S} \) satisfies \( \mathbf{S} = \arg \min_{\mathbf{S}} \text{dist} (\mathcal{F}; \mathbf{S}) \), where \( \text{dist} (\mathcal{F}; \mathbf{S}) \) is the transportation cost defined below.

**Definition 8 (Transportation Cost)** Given the transport function \( \mathcal{F} \), we can find the closest point in the target set to the transferred point. Similarly, for each point in the target set, we can find the closest transferred point in the original. We define two sets of
indices as follows:

\[
I_1 \triangleq \left\{ (i,j) \mid i = 1, N_1, j = \arg \min_{j_0=1}^{N_2} \| \mathcal{F}(x_{1,i}; \cdot) - x_{2,j_0} \|_2 \right\}, \tag{5.5}
\]

\[
I_2 \triangleq \left\{ (i',j') \mid j' = 1, N_2, i' = \arg \min_{i_0=1}^{N_1} \| \mathcal{F}(x_{1,i_0}; \cdot) - x_{2,j'} \|_2 \right\}.
\]

The transportation cost (also called total distance) is defined as follows, which captures the overall disparity between the transferred points and the target points:

\[
\text{dist} (\mathcal{F}; S) \triangleq \sum_{(i,j) \in I_1} \| \mathcal{F}(x_{1,i}; S) - x_{2,j} \|_2^2 + \sum_{(i',j') \in I_2} \| \mathcal{F}(x_{1,i'}; S) - x_{2,j'} \|_2^2. \tag{5.6}
\]

Notice that the transportation cost over set \( I_1 \) (first term) ensures that every transferred point from the original registers to the closest point in the target, and conversely, the cost over \( I_2 \) (second term) ensures that every point in the target finds the closest point in the transferred point cloud. It is crucial to include the second term in the cost so as to avoid ill-posed transfer, e.g., transferring all the points to a single point in the target.

We apply fixed point iterations to solve \( \mathcal{F}(x; \cdot) \) in Equation (5.4). Given the point registration function at time \( (t - 1) \) denoted as \( \mathcal{F}^{(t-1)} \) and the tensor \( S^{(t-1)} \), we update \( \mathcal{F}^{(t)} \) and \( S^{(t)} \) by Equations (5.8) and (5.9), respectively.

\[
\mathcal{F}_{s}^{(t)} (x; S^{(t-1)}) = \sum_{i',j'} \left( \sum_{i,j} S_{i,j,i',j'}^{(t-1)} g_{i,j} (x; \mathcal{F}^{(t-1)}) \right) f_{i',j'} (x), \tag{5.7}
\]

\[
\mathcal{F}^{(t)} (x; S^{(t-1)}) = \frac{1}{2} \left[ \mathcal{F}^{(t-1)} (x; S^{(t-1)}) + \mathcal{F}_{s}^{(t)} (x; S^{(t-1)}) \right]. \tag{5.8}
\]

\[
S^{(t)} = \arg \min_{S \geq 0} \text{dist} (\mathcal{F}^{(t)}; S). \tag{5.9}
\]
5.3 Experiments

Figure 5.3: Process of implementing GRT to color transfer tasks.

We now introduce our implementation of GRT on the color transfer problem. Before applying GRT, we first represent all pixels by their coordinates in the $l\alpha\beta$ color space because it better approaches human perception [178]. We use two methods to estimate the GMM parameters of both the original and the target: (1) the Mean-Shift method [179] with bandwidth 7, denoted by $GRT+MS$, and (2) the EM algorithm with BIC to determine the number of Gaussian components, denoted by $GRT+BIC$. Then, we apply GRT in the previous section to get the point-wise transfer. Finally, we use the indices set $I_1$ generated with GRT in Equation (5.5) to update the pixels with their transported color. The whole process is schemed in Figure 5.3 and detailed in Algorithm 2.

We conducted the color transfer experiment on Matlab R2019b, where approximated nearest neighbors (ANN) search [181] and quadratic programming in Mosek [182] are adopted for code efficiency. The codes are available in the appendix. The execution time is 2 to 15 minutes on one Intel Xeon E5-8860 v3 CPUs with 16 cores at 2.2-3.2 GHz and 200 GB RAM, where up to 20 Gaussian components can be estimated in one GMM, GRT converges within 20 iterations in most cases. The processing time is higher with more Gaussian components and higher image resolution, and there is room for improving the computational efficiency by using C++ or speeding up the quadratic programming in Equation (5.9).

We compare our method with the other widely adopted color transfer methods including linear color transfer [11,178], statistical inference changes [9,10], and OT-based method [6] with L1 and L2 loss. For memory and speed constraints, we only compared
Algorithm 2 Color Transfer with GRT

Input: Original and target images.
Output: Original image transferred in target palette.

1. Represent color points of original and target images as $X_1$ and $X_2$ in $l\alpha\beta$ space.
2. Estimate two GMMs $X_1 \sim M_1$ and $X_2 \sim M_2$.
3. Initialize the pixel-wise transportation $F_0$ with Approximate Nearest Neighbours algorithm [180].
4. Initialize transfer parameter matrix $S_0$ with Equation (5.9).
5. while $(t < 2 || \text{dist}_t - \text{dist}_{t-1} > \delta)$ do
6.   Update transfer $F_{t+1}(\cdot; S_t)$ with Equation (5.8). Update the index pairs $I_1$ and $I_2$ in Equation (5.5). Update tensor parameters $S_{t+1}$ with Equation (5.9).
7. end while
8. Assign pixel colors of the original image to the target palette with $I_1$
9. return transferred image.

with the algorithms that can run on a computer, which requires memory less than 200 GB and without a heavy parameter tuning process. After transferring color distribution from the original to the target, we swap the original and the target and do the transfer reversely. The results are shown in Fig. 5.4, where the color distribution is placed to the right side of the original image. As we can see, this color transfer task is challenging because the color distributions in the original and target are significantly different. Compared with other methods, ours better preserve local details. The color distribution after transportation still resembles the original one, which ensures meaningful semantic representations in transferred images.
Figure 5.4: Comparison of our approaches using mean-shift estimation of GMM parameters (annotated as \textit{GRT+MS}) and using BIC+EM to estimate GMM parameters (annotated as \textit{GRT+BIC}) with other color transfer methods in [9–13]. 1st row: Original images with color histograms. 2nd-8th row, 1st-2nd column: Original images with color histograms after color transfer to the target. 2nd-8th row, 3rd-4th column: Target images with color histograms after color transfer to the originals.
Here in Figure 5.5, we show more results of the color transfer.

Figure 5.5: Color transfer examples with GRT+MS (GRT+BIC has the similar results). *Left:* original and target images. *Right:* transferred images.
5.4 Summary and Discussion

We proposed a novel optimization framework GRT for matching two sets of points and demonstrated that GRT reduces the artifacts that often occur from discrete OT with strict marginal constraints. Under our framework, the transportation of colors is locally affine and monotonic. Furthermore, component-wise consistency is imposed on transportation. Because pixels from the same semantic region tend to locate closely in the color space and thus belong to the same component of the GMM, they tend to be transferred by similar functions. As a result, a semantic region is likely to retain color coherence.
Chapter 6  
Conclusions and Future Work

To conclude, this thesis provides some novel pattern recognition and pattern matching methods on scientific image processing. By analyzing and harnessing a large quantity of carefully collected scientific image data, using state-of-the-art machine learning and statistical modeling methods, researchers will possess a deeper understanding of natural philosophy and scientific patterns.

6.1 Summary of Contributions

The contributions of this thesis are shown in multiple computational modelings. In the first application, I designed a machine learning method to capture patterns of severe weather events from satellite images. Our model learned the critical shape and motion features of the comma-shaped cloud to identify the early visual clues of severe weather events, which is of great importance to the decision making to the public commercial and security.

In the second application, I exploited the usage of OT in pattern matching. Our pattern matching framework could overcome the strict boundary constraint of standard OT. By combining spatial loss and shape-similarity loss in the cost function, our method is able to combine these two perspectives of information in cell matching. As a result, our cell tracking algorithm could increase the cell tracking accuracy on complicated microscopy videos.

Furthermore, I am interested in constraining OT with the global structure of distributions. To eliminate the artifact in image recoloring tasks, I use GMM to approximate the source and target distributions and design the matching framework under the GMM form of structure. By such a design, our new framework could preserve the perceptual
consistency between pixels that locate closely in the color transfer.

6.2 Future Work

Pattern recognition and matching on scientific images are highly challenging because of the complicated nature of these images and the large variety of tasks. Currently, we are still far from revealing all the secrets in scientific images. With the development of sensor technology and machine learning, there will be increasing availability to better interpret scientific images.

One of the potential future directions is the deep-learning-based scientific image understandings. Because manual labeling on scientific images requires domain knowledge, retrieving large labeled datasets was challenging and tedious. Recently, large manually labeled scientific image datasets have appeared through the help of experts, including pulmonary landmark point pairs [183], EMPIRE10 challenge [184], and Medical Image segmentation [185]. These data make it possible to design the problem-specific deep learning models.

Another potential trend of scientific image analysis is the personalized home-use models on cheaper sensors, especially for medical usage. For the patients that need to monitor their body periodically, computational models on those medical instruments help self-identifying disease and provide affordable medical solutions.

Finally, I list some problem-specific future directions to the problems that I have discussed in this thesis.

6.2.1 Severe Weather Event Forecasting with Deep Neural Network (DNN)

The proposed severe weather forecasting frameworks face two main issues: (1) lack of high-quality labels. The accuracy of machine learning algorithms highly relies on the quality of data. In this example, the identification of the comma-shaped cloud requires domain knowledge in meteorology. However, it is infeasible to use substantial labor of experts to manually label the comma-shaped cloud for a long time, say, ten years, and (2) the complicated relationship between visual features on weather images and severe weather events. Although the comma-shaped cloud is an important visual feature of storms, their spatial-temporal relationship is still an unrevealed mystery. Severe weather events are observed around the head and the tail of the comma-shaped cloud,
but sometimes they appear in cloudless regions. Also, severe weather events can happen long before or long after the comma-shaped events. Even if meteorologists capture a comma-shaped cloud timely and accurately, they have to rely on other information to locate the event. In addition, there are other human identified visual features, such as a bow echo, velocity couplet, etc. Only by understanding the underlying relationship between visual features and severe weather events can we define a reliable severe weather forecasting system using weather images.

Instead of targeting particular human identified visual features, our future work will directly focus on severe weather events forecasting. To be specific, we take the data-driven approach and let DNN learn the patterns of weather from massive data. An advantage of directly targeting severe weather events is that there are detailed records of storms in the NOAA database, including event types, time, locations, duration and affected ranges. These complete labels enable the training tasks to be more scalable.

DNN possesses a powerful ability to learn object features and image structures from massive data. Take the convolutional neural network (CNN) as an example, its network structure mimics the neuron of human brains of information sharing, and its convolutional filters capture the geographical features in images. They have shown good performance in challenging object recognition tasks such as ImageNet. Some recent work also explored using DNN frameworks in weather forecasting and retrieved impressive results. Therefore, I believe DNN-assist severe weather forecasting is a promising direction in the future.

6.2.2 Simultaneous Cell Tracking over the Whole Video

While our proposed method only matches cell patterns on consecutive pairs of consecutive frames, some researchers consider the entire video as a whole and do the tracking task on all the frames. Compared with methods that track cells based on pairs of frames, the methods that make use of the whole video simultaneously can achieve higher accuracy at higher computational and memory costs because a broader spatial-temporal context is exploited. However, the latter methods usually require carefully crafted features to characterize a cell. Thus, whole video methods are impractical for general datasets.

In contrast, there are two main limitations if we only consider pairs of image frames. First is the difficulty in observing long-lasting activities. If activities like cell division take longer than the span of two frames, it is hard to detect them promptly. Second is the sensitivity to random segmentation errors. In cell segmentation, changes in lighting
conditions may cause false or missing segmentation of a cell. Such errors can be easily noticed if we observe the whole video, but they are harder to spot if we inspect only two frames. One remedy to overcome these limitations, to some extent, is by post-processing.

In a more principled approach that performs pixel mapping by optimization over the entire video sequence, we can adopt the so-called multi-marginal OT problem [190], an extended OT problem in our future work. In particular, we want to align the pixels in all the frames simultaneously. Index the image frames by $s \in [0, S]$. Using the same math notations of [190], let the point set at time $s$ be $X_s$ with the weight simplex $a_s$. The set of couplings with the boundary constraints $\{a_s\}_{s=1}^S$ is given by:

$$U(\{a_s\}) \overset{\text{def}}{=} \left\{ T \in \mathbb{R}_{+}^{n_1 \times \cdots \times n_S} | \forall s, \forall i_s, \sum_{l \neq s} \sum_{i_l=1}^{n_l} t_{i_1, \cdots, i_S} = (a_s)_{i_s} \right\}. \quad (6.1)$$

Here $(a_s)_{i_s}$ is the $i_s$-th element of the simplex $a_s$. If tensor matrix $M \in \mathbb{R}^{n_1 \times \cdots \times n_S}$ represents the transportation cost between multiple point sets $\{X_s\}$, the optimization problem can be represented as $\min_{T \in U(\{a_s\})} \langle T, M \rangle$, which is the dot product of tensor $T$ and $M$.

The solution space of multi-marginal OT is $O(n_1 n_2 \cdots n_S)$, exponential in the number of frames. Currently, the problem is computationally intractable. Our method to compute a series of OT problems between every two frames is an approximation to the global optimum in the whole video. With the development of the numerical approximations in multi-marginal OT, our proposed method can be running on the whole video to get more accurate tracking results.

### 6.2.3 Color Palette-based Image Recoloring with GRT

One of the limitations in our proposed GRT image recoloring framework is that, it assumes we know color distributions of both source and target images, which is referred to as Example-based image recoloring in the related literature [173]. However, in real-life color editing tasks, it is not available for users to provide a concrete example in the desired color hue. Most commercial software like Photoshop or iPhoto allows users to manipulate the color histogram directly, namely Palette-based image recoloring. However, these tools require users to have sufficient knowledge to interpret RGB color histograms, while non-expert users only have the intuition such as “the red color should be less saturated”. In this case, an algorithm that implements human-guided palette-based image recoloring is needed.

I modify the requirement of GRT according to the new setup. The following mathemat-
ical annotations have a consistent meaning with those in Chapter 5. In the aforementioned example-based image recoloring setup, all parameters of target GMM $\mathcal{M}_2$ are predetermined by the target image. In the new palette-based image recoloring setup, only means of color components $\mu_{2,i}$, $i = 1, N_2$ are predetermined, which is the color palette required by the user. The covariance matrix $\mu_{2,i}$ and the prior probability $\pi_{2,i}$ of each color are to be determined by the algorithm. To avoid unreasonable transportation and maintain the semantics of the recolored image, I manually set up the following two requirements for GRT:

- The target proportion of each component is lower-bounded, i.e., $\pi_{2,i} \geq v_0$.
- The target proportion distributions have the minimized total transportation cost from the source.

The first requirement is for avoiding color elimination, and the second requirement is for maintaining the texture and features of the original image. Since covariance matrix $\mu_{2,i}$ has a limited effect in large sample size, I fix the covariance matrix and optimizing prior probability $\pi_{2,i}$ in the following linear programming problem:

$$\text{dist} (\mathcal{M}_1, \mathcal{M}_2) = \sum_{i=1}^{N_1} \sum_{j=1}^{N_2} w_{i,j} d_{i,j}^2,$$

(6.2)

with the boundary conditions:

$$\sum_j w_{i,j} = \pi_{1,i}, \forall i = 1, N_1$$
$$\sum_i w_{i,j} \geq v_0, \forall j = 1, N_2$$

where $w_{i,j}$ is the mass transit between $\phi_{1,i}$ and $\phi_{2,j}$ and $d_{i,j}$ is the W-2 Wasserstein distance between two Gaussians discussed in Chapter 3.2.3. The prior probability is then computed by $\pi_{2,j} = \sum_i w_{i,j}$. By this design, we can solve the palette-based image recoloring task with the design of GRT.

Besides the scenarios mentioned above, the realm of computer-assist image recoloring is far from explored. AI-assisted automatic image recoloring framework will be a future trend in the movie and art industries. I will continue developing image recoloring algorithms that produce more natural results in human perception.
Appendix
Supplementary Material for Cell Tracking on Microscopy Observations

1 Effect of Different Initialization Methods on GWD

WGWD is a non-convex problem when weight $\lambda = 1$ (i.e., GWD) [122]. Because the problem is convex when $\lambda = 0$ [120], we only discuss the effect of initialization on GWD. The optimization algorithm to compute GWD will converge to a local minimum, but the achieved value of the objective function depends on initialization. In cell tracking, for most cases, we do not observe changes in point alignment with different initialization because the actual shape change between two frames is negligible and most cells have quite regular shapes.

We experimented on more special shapes from the dataset in [122] so that the difference is more visible. For a pair of shapes, we compute point alignment from all the pixels of the source shape to those of the target shape. Two shapes are resized to the same scale and each pixel on the shape has the same weight. To visualize the registration map, pixels in the source are artificially assigned with varying colors, and their corresponding matching pixels in the target are given the same color.

We adopt three initialization methods for the transportation matrix $T$: point registration computed from WD, random alignment, and uniform alignment. We use entropy regularization $\epsilon = 0.001$ and the square loss in the experiment. The results are displayed in Figure 1. The first and second columns show the source and the target shapes, the third column shows point registration computed from WD, and the fourth to the
sixth columns show point registration computed from GWD with the three initialization methods.

<table>
<thead>
<tr>
<th>Source</th>
<th>Target</th>
<th>WD</th>
<th>WD-GWD</th>
<th>Random-GWD</th>
<th>Uniform-GWD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 1: Point registration with different initialization methods. Left: source shapes and target shapes. Right: visualization of point registration by WD, GWD with WD initialization (WD-GWD), GWD with random alignment initialization (Random-GWD), and GWD with uniform alignment initialization (Uniform-GWD).

As shown in Figure 1, the transportation matrix computed from WD (the column marked as WD) has obvious artifacts, demonstrating that GWD achieves better matching by enforcing shape consistency. If the source and the target shapes do not flip, e.g., the shapes in the first to the fourth row of the figure, GWD with initialization by WD (the column marked as WD-GWD) has yielded high-quality matching. However, if
the shapes are flipped, e.g., the shapes in the fifth to the eighth row, GWD with WD initialization has failed to discover the flip. The results by GWD with random alignment initialization (the column marked as Random-GWD) seem to be the worst among the three initialization methods. The best results are consistently achieved by GWD with uniform alignment initialization (the column marked as Uniform-GWD). In fact, uniform alignment initialization is computationally more efficient than WD initialization. As a result, we adopt uniform alignment as the initialization method for computing WGWD.

2 Visualization of RBOT Algorithm

In Figure 2 we provide more examples of cell clusters generated by RBOT algorithm (see Section 4.3.2). For the meaning of the colors in the figures, please refer to Section 4.3.2 (see the part Example). If we show the cells in the source and target images at their real positions, the location changes of the aligned cells are usually so small that they will overlap tremendously, making them impossible to discern in one image. To avoid this difficulty in visualization, we artificially shift cells in the target image by 20 pixels in the south-east direction.

3 Computation Time and Downsize Effects

Compared with cell-wise tracking algorithms, our OT-based algorithm is more time consuming since it solves the more fine-grained point alignment. The computational efficiency can be improved by downsizing the images, but doing so may reduce the tracking accuracy. We show results in Table 1. The tracking accuracy as measured by TRA is listed for images downsized to 0.5× and 0.25× of the original size (here size is the length or width, not the area).

According to Table 1, the TRA tracking accuracy decreases when the images are downsized. The reduction in accuracy is small if cells in the downsized images are still large enough for recognition. For example, for datasets DIC-HeLa and PhC-U373, downsizing to 0.5× affects only marginally the final accuracy (< 2%). But for images with small and dense cells, the loss in accuracy can be significant. For example, in the dataset PhC-PSC, one cell only has ~ 30 pixels in the downsized image (0.5×), which is hard to see by the eye. The tracking accuracy for this dataset drops by 7% or as much
Figure 2: Visualization of cell clusters generated by RBOT algorithm for a dataset in Cell Tracking Challenge [14]. Instead of the original images, cartoons of segmented cells are shown. Blue colored cells are from the source image and red colored cells are from the target. Each row from left to right: cells labeled by ground truth and their corresponding alignment, initialization with multiscale OT, after one round of recursion, and after two rounds of recursion. Cell category from top to bottom: Rat mesenchymal stem cells on a flat polyacrylamide substrate, Pancreatic Stem Cells on a Polystyrene substrate, HeLa cells stably expressing H2b-GFP, and HeLa cells on a flat glass.

as 50%+ when the downsizing ratio is 0.5 or 0.25.

In summary, although downsizing can be useful for speeding up the algorithm, there is a trade-off with accuracy. The practice is not recommended if the cell size is small, e.g., below 300 pixels on average.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Cells/Frame</th>
<th>Size</th>
<th>#Pixels/Cell</th>
<th>Time/Frame(s)</th>
<th>TRA</th>
</tr>
</thead>
<tbody>
<tr>
<td>PhC-PSC-01</td>
<td>340</td>
<td>576 × 720</td>
<td>109</td>
<td>49</td>
<td>0.83</td>
</tr>
<tr>
<td></td>
<td></td>
<td>388 × 360</td>
<td>27</td>
<td>14</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>194 × 180</td>
<td>7</td>
<td>11</td>
<td>0.26</td>
</tr>
<tr>
<td>PhC-PSC-02</td>
<td>278</td>
<td>576 × 720</td>
<td>117</td>
<td>33</td>
<td>0.84</td>
</tr>
<tr>
<td></td>
<td></td>
<td>388 × 360</td>
<td>29</td>
<td>14</td>
<td>0.74</td>
</tr>
<tr>
<td></td>
<td></td>
<td>194 × 180</td>
<td>8</td>
<td>11</td>
<td>0.26</td>
</tr>
<tr>
<td>DIC-HeLa-01</td>
<td>13</td>
<td>512 × 512</td>
<td>10098</td>
<td>1283</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256 × 256</td>
<td>2524</td>
<td>225</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128 × 128</td>
<td>631</td>
<td>119</td>
<td>0.89</td>
</tr>
<tr>
<td>DIC-HeLa-02</td>
<td>12</td>
<td>512 × 512</td>
<td>11901</td>
<td>1422</td>
<td>0.78</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256 × 256</td>
<td>2975</td>
<td>250</td>
<td>0.77</td>
</tr>
<tr>
<td></td>
<td></td>
<td>128 × 128</td>
<td>743</td>
<td>128</td>
<td>0.76</td>
</tr>
<tr>
<td>PhC-U373-01</td>
<td>7</td>
<td>520 × 696</td>
<td>4243</td>
<td>828</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>260 × 348</td>
<td>1061</td>
<td>124</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>130 × 174</td>
<td>266</td>
<td>57</td>
<td>0.89</td>
</tr>
<tr>
<td>PhC-U373-02</td>
<td>6</td>
<td>520 × 696</td>
<td>3269</td>
<td>754</td>
<td>0.91</td>
</tr>
<tr>
<td></td>
<td></td>
<td>260 × 348</td>
<td>817</td>
<td>116</td>
<td>0.87</td>
</tr>
<tr>
<td></td>
<td></td>
<td>130 × 174</td>
<td>205</td>
<td>48</td>
<td>0.71</td>
</tr>
<tr>
<td>PSC-Passage7</td>
<td>189</td>
<td>1038 × 1376</td>
<td>293</td>
<td>83</td>
<td>0.96</td>
</tr>
<tr>
<td></td>
<td></td>
<td>519 × 688</td>
<td>73</td>
<td>14</td>
<td>0.90</td>
</tr>
<tr>
<td></td>
<td></td>
<td>260 × 344</td>
<td>18</td>
<td>10</td>
<td>0.82</td>
</tr>
</tbody>
</table>

Table 1: Comparison of computation time and tracking accuracy at different scales of image downsizing. For every image sequence, three image sizes are experimented with, the largest size is that of the original data. All the experiments are conducted under the same setup as specified in the main paper.
Bibliography


Vita
Xinye Zheng

Xinye Zheng received her B.S. degree of statistics in 2015 from the University of Science and Technology of China. As a Ph.D. student at College of Information Sciences and Technology from the Penn State University, she worked under supervision of Professor James Wang and Professor Jia Li. Her research interests lie in optimization problems, weather image processing, and machine learning applications. She was an exchange student at University of Almeria in summer 2016. During her Ph.D. study, she worked as a research intern for PNC Bank in summer 2017. She also worked as a software development intern for Mathworks Inc. and Google LLC. in the summer of 2018 and 2019, respectively. After graduation in July 2020, she will join Facebook Inc. as a research scientist.