UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Deep Learning for Flow Feature Detection

Permalink

https://escholarship.org/uc/item/5j34x838

Author

de Silva, Akila Udakara

Publication Date

2023

Copyright Information

This work is made available under the terms of a Creative Commons Attribution-NonCommercial-NoDerivatives License, available at https://creativecommons.org/licenses/by-nc-nd/4.0/

Peer reviewed|Thesis/dissertation

eScholarship.org Powered by the California Digital Library
University of California

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

# DEEP LEARNING FOR FLOW FEATURE DETECTION

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

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

**Akila de Silva**

December 2023

The Dissertation of Akila de Silva
is approved:

Professor Alex Pang, Chair

Professor James Davis

Dr. Gregory Dusek

Peter Biehl
Vice Provost and Dean of Graduate Studies

Copyright © by

Akila de Silva

2023

# Table of Contents

# List of Figures

# List of Tables

# Abstract

Deep Learning for Flow Feature Detection

by

Akila de Silva

The visualization community uses deep learning in two main ways: to explain the inner workings of deep learning models and to incorporate deep learning into visualization pipelines. In this dissertation, we focus on the latter, with a specific emphasis on feature detection within flow visualization pipelines. This dissertation underscores the efficacy of deep learning for feature detection, particularly in visualizing complex flow phenomena like rip currents and vortices. First, we explore the use of conventional flow visualization methods, such as vector clustering and timelines, to visualize rip currents. Then, we investigate the use of the appearance of the flow field for rip current detection. Subsequently, we propose a hybrid feature detection method that combines conventional flow analysis with deep learning to find rip currents by learning their behavior from short sequences of pathlines. Finally, we introduce a multimodal deep learning approach to find vortex boundaries that learn from the shape and other physical properties of pathlines or streamlines.

To My Parents

# Acknowledgments

I am profoundly grateful to my dissertation reading committee: Professor Alex Pang, Professor James Davis, and Dr. Gregory Dusek, whose invaluable advice, unwavering support, and insightful guidance shaped the trajectory of this work. Additionally, I express my appreciation to the anonymous reviewers whose constructive feedback significantly strengthened the papers forming the foundation of this dissertation.

This dissertation would not be possible without the financial support I received from NOAA and SECOORA. Additionally, the support from the Computer Science and Engineering Department and the Science Internship Program has been instrumental in realizing this work.

I am truly appreciative of the collaborations and feedback extended by exceptional colleagues. Among them, I wish to express my deep appreciation for Issei Mori, Fahim Hasan Khan, Jiahao Luo, Mona Zhao, Donald Stewart, Nicholas Tee, Omkar Ghanakar, Vanshika Vats, Marzia Nizam, Raja Kumar, Minghao Liu, Eric Sandoval Ruezga, Tara Natarajan, Boglárka Ecsedi, Nick Jiang and Max Wang. Their insights and camaraderie have made a remarkable impact on this dissertation.

Finally, I would like to thank my family, especially my parents, for their endless love, support, and encouragement during these five years.

The text of this dissertation includes reprints of the following previously published material:

1. Mori, I., **de Silva, A.**, Dusek, G., Davis, J., & Pang, A. *Flow-Based Rip Current Detection and Visualization. IEEE Access* https://doi.org/10.1109/ACCESS.2022.3140340

2. **de Silva, A.**, Mori, I., Dusek, G., Davis, J., & Pang, A. *Automated rip current detection with region based convolutional neural networks. Coastal Engineering* https://doi.org/10.1016/j.coastaleng.2021.103859

3. **de Silva, A.**, Zhao, M., Stewart, D., Khan, F., Dusek, G., Davis, J., & Pang, A. *RipViz: Finding Rip Currents by Learning Pathline Behavior. IEEE Transactions on Visualization and Computer Graphics.* https://doi.org/10.1109/TVCG.2023.3243834

The co-author listed in this publication directed and supervised the research which forms the basis for the dissertation.

# Chapter 1

# Introduction

In the modern world data has become abundant due to easily accessible, low-cost acquisition devices like web cameras and mobile phones [4]. Effective visualization of these data and processes can reveal trends, relationships, outliers, and hidden insights that might not be apparent when examining raw data alone, making it a valuable tool for both researchers and the general public.[5].

In order to create effective visualizations, the visualization community has been engaging with deep learning methods in two main ways. The first is to explain the inner works of a deep learning model and to gain insight into the decision making process of these often considered black box models. The second is introduce machine learning into a visualization pipelines. This dissertation focuses on the latter.

In this dissertation, we introduce deep learning into flow visualization pipelines.

In flow visualization pipelines, an essential component is feature detection. Feature detection involves identifying and extracting specific patterns or characteristics in the flow field that are interesting for analysis. This dissertation focuses on using deep learning to find interesting flow features. We focus our work on two applications: rip currents, a natural flow pattern found in nearshore oceans and vortices.

In this dissertation, I accomplish the following,

1. A study of traditional flow visualization methods for visualizing rip currents.
2. A deep learning method that uses appearance to find rip currents.
3. A deep Learning method that uses sequence of pathlines to visualize rip currents
4. A deep learning method that uses flowlines to visualize vortices.

The rest of this dissertation is organized as follows. In chapter 02 , I will discuss how rip current features can be found by using its appearance. In chapter 03, I will discuss how transitional flow visualization methods such as timelines can be used for rip current detection. In chapter 04, I will discuss how we can use machine learning to learn from pathlines to find rip currents. In Chapter 05, I will discuss how streamlines and pathlines can be used to find vortices. In Chapter 06, I will conclude my dissertation.

# Chapter 2

# Conventional Flow Analysis for Rip Current Detection.

## 2.1 Introduction

Rip currents are dangerous and can be deadly. The majority of the population does not know how to identify them. Detecting rip currents in webcam video can inform users of potential hazards in near real-time and be utilized to support ongoing efforts at rip current prediction. Previously proposed methods detect some types of rip currents but not others. While image processing and machine learning methods perform well on specific types of rip currents, they are not applicable in all scenarios. This paper proposes and investigates a more general approach based on flow analysis and adapting flow visualization methods to detect rip currents.

Beaches around populated urban areas attract many beach goers and other recreational users. However, the beach can pose a grave danger to the unwary public in the form of rip currents [6]. People who unknowingly enter a rip current may be carried out to the sea if the current is strong enough. Globally there are thousands of drownings each year due to rip currents [7][8]. It is estimated that around 82% of rescues on the beaches in the United States are due to rip currents [9]. Year-to-year statistics collected by the U.S. National Oceanic and Atmospheric Administration (NOAA) [10] indicate that there has been no significant decline in the number of drowning fatalities due to rip currents [11] despite the proliferation of signage, videos and other public safety messaging warning of the potential dangers rip currents pose.

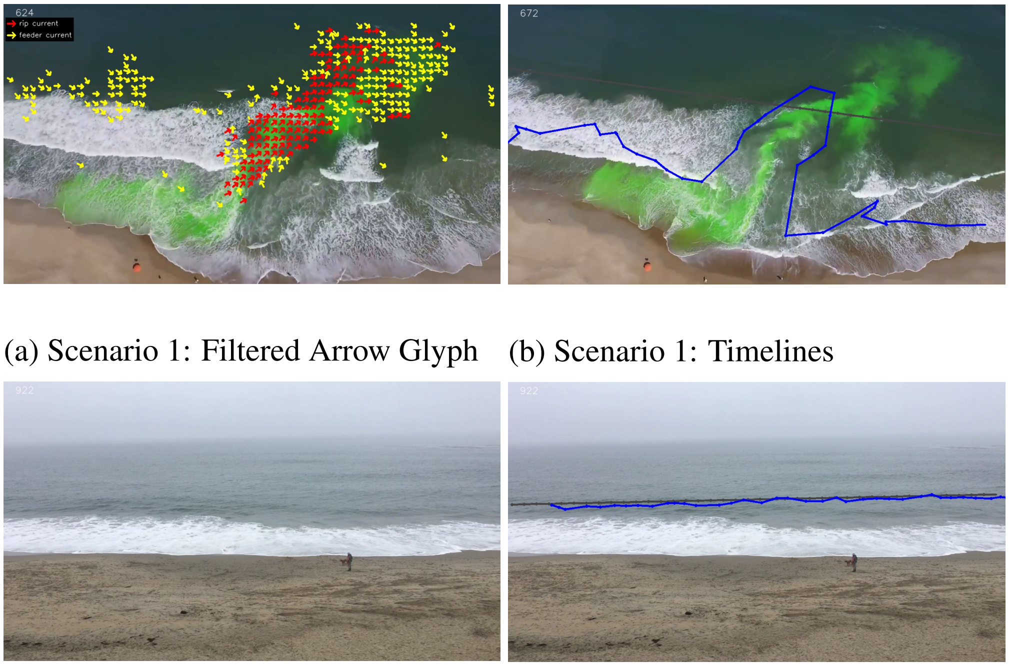
Even people with some knowledge about beach safety can have difficulty in properly identifying rip currents. In a 2008 study, researchers found that almost 80% of surveyed Australian beach goers were aware of common rip safety advice such as "swim parallel to the beach." However, only 40% could identify a rip current when shown a picture of one, even though 80% thought they could [12]. In fact, even professional lifeguards cannot always accurately identify the presence of rip currents [9]. The reality is that rip currents are often not readily or easily identifiable to the average beach goer [13]. Furthermore, the most vulnerable ones are the occasional beach goers and weak swimmers [14]. While there are options for in-situ measurements to identify rip currents, they are generally expensive and require extensive setup. In this sense, there is currently no robust location-independent means of rip current identification.

This paper investigates the potential for using flow visualization methods as a means for identifying rip currents from short video clips. It has the potential for identifying rip currents based on the behavior of water movement rather than simply the appearance of the water state. The analysis and visualization pipeline starts with video pre-processing, including image stabilization and applying optical flow computation to obtain a time-dependent flow field. We investigate multiple flow visualization methods, including color maps, pathlines, arrow glyphs, and timelines, to detect and visualize the presence of rip currents. Examples of two of these methods on scenes with and without rip currents are shown in Figure 2.1. After describing the methods, we evaluate them on various data sets and compare them against other existing methods as well as human-annotated data to showcase their performance in challenging cases.

Contributions of this chapter are:

- Our investigation of flow visualization methods for detecting rip currents found that standard visualization methods performed poorly when applied directly. However, with appropriate modifications driven by wave behavior, flow visualization methods can detect rip currents in scenarios where existing state-of-the-art rip detection methods fail.

- We performed comparisons between the proposed flow based methods against current best practices (classification based on Timex images) and other machine learning based methods. Our flow based methods also improved human labels that relied on Timex images alone.



(a) Scenario 1: Filtered Arrow Glyph (b) Scenario 1: Timelines

(c) Scenario 2: Filtered Arrow Glyph (d) Scenario 2: Timelines

Figure 2.1: **Examples of two recommended visualization methods for detecting rip currents:** The filtered arrow glyph method highlights rip current and potential feeder current with arrows, and the timelines method deforms in the presence of a rip current. In Scenario 1, the original video contains green dye as a visual aid to the rip current present in the middle of the screen. The dye does not impact resulting visualizations. In (a) the Filtered Arrow Glyph highlights the body of the rip in red and the potential feeder current in yellow. The timelines in (b) clearly shows the presence of the seaward flow caused by the rip. On the other hand, Scenario 2 shows our visualizations in the case there is no rip current. In (c), no arrows are shown, indicating that there is no rip current. In (d), the timeline is relatively straight, indicating that the flow filed is mostly uniform, hence the absence of a rip current.

## 2.2 Background and Related Work

*Rip currents:* Rip currents are a well-studied ocean phenomenon [15][16][14]. Many factors contribute to the formation of rip currents, such as bathymetry, wave characteristics, and natural and man-made structures along the beach. As a result, there are different types of rip currents, including bathymetry-controlled rips, structural rips, circulating rips, and others [17]. Rips may either be transient, potentially moving along the beach and lasting only seconds to minutes, or persistent, holding a near-constant position for hours or days at a time. Rips that are frequently found at the same location are usually indicative of a structural feature such as jetties or piers, rocky outcrops, reefs, or persistent sandbars which lead to variations in breaking wave heights alongshore. Regions of larger breaking waves lead to higher water levels, which then flow alongshore to regions of smaller breaking waves and lower water levels and then offshore as a rip current. On sandy beaches, regions of smaller breaking waves are often characterized by deeper channels, which are often indications of rip current presence. Rip currents may pulse, gaining strength when there is a wave set and weakening in between sets. In terms of appearance, some rips may be identified by water discoloration as beach sediment is carried by the rip past the surf zone (region where waves are breaking). In other cases, rips may be identified by a darker region of water that is flanked on either side by breaking waves. The movement of foam or other debris on the water surface can also provide clues of rip current locations. Rip currents are quite varied, dynamic, and pose a challenge to detect robustly.

*Rip Forecasting:* Models that incorporate the dominant factors contributing to rip formation have been proposed to predict future rip current occurrence e.g. [18][19]. In fact, NOAA recently announced an operational rip current forecast for the United States coastline [20]. Note that these models differ from efforts in this paper and other methods described in this section in that the latter focus on detection rather than forecasting. The forecast models will benefit from advances in rip detection to validate and improve rip current predictions.

*Lifeguards:* Lifeguards rely on visual cues and experience to identify rips, which requires training and familiarity with the locale. However, most drownings occur on beaches without trained personnel [21][22]. Posted signs can provide some information regarding what to do if caught in a rip current, but there is evidence that many people do not find existing signs helpful in identifying rip currents [23].

*In-situ measurements:* In-situ measurements such as acoustic doppler current profilers (ADCP), wave sensors, and acoustic velocimeters provide water column flow information [24][25][26][27][28]. However, these are expensive and challenging to deploy in the surf zone, and they only provide data for one location at a time. Fluorescein dye is commonly released into the ocean and the dispersion observed [29][30][31][32]. Floating drifters with embedded GPS units have also been used to measure currents [33][17][34]. However, these methods require some idea of where a rip might exist in the first place.

They are also a research endeavor and not designed for use by the general public. In addition, they are impractical for detecting flash rips that are more transient in nature.

*Time-exposure images:* Experts at the National Oceanic and Atmospheric Administration (NOAA) use images and video to gather statistics about rip currents [35]. These data support the validation of a rip current forecast model to alert people to potential hazards [18]. One method that has promise for visually detecting rip currents is the use of "time exposure" or Timex images [36][37][38][39]. These are obtained by simply averaging frames of a video clip, usually over a period of 10 minute intervals. This approach works well for rips that are characterized by a darker region of water flanked by breaking waves since places with consistent breaking waves will appear blurred white, while the location of a rip will appear darker. However, its main weakness is that it can only identify rips with these visual characteristics. Furthermore, because the time-averaging window is over many wave periods, it may lead to an incorrect classification e.g. for non-stationary rip currents. While Timex images are most commonly viewed by human experts, Maryan et al. [40] trained a machine learning model to determine whether a Timex image contains a rip channel or not. They reported a detection rate of 85% for various beach locations. Nelko also used time-averaged images and noted that prediction schemes developed at one beach location might not be directly applicable to another [41]. Nonetheless, the main weakness of any methods which rely on Timex images is their limitations on the type of rips they can identify – limited to bathymetry-controlled rips.

*Segmentation:* Another method for detecting rip currents is to find discoloration due to sediment transport. Liu and Wu [28] reported that imagery captured from a stationary webcam can be segmented based on hue. Possible rips are identified if the sediment plume extends beyond some distance from the shoreline. Together with environmental monitoring equipment for wind speed, wind direction, wave height, and wave period, they have an automated system that issues alerts of flash rip dangers to beach goers. However, this method is specific to only certain locales with sediment plumes, and as with Timex images, limited to rips that exhibit sediment plumes.

*Neural Networks:* De Silva et al. [42] trained a neural network model to identify rip currents from a sequence of images. They reported a detection rate of 98.4% for their test set consisting of videos with and without rip currents. However, their model can only detect rips with consistent breaking waves since they use single-frame, RGB-based images of rip currents characterized by a gap in breaking waves to train their model. Like the approaches based on Timex, their implementation is currently limited to bathymetry-controlled rips until training data are collected for other types of rips and their model retrained.

*Optical flow:* Dense optical flow [43][44] can be used to detect rip currents in a video. This method is attractive since optical flow fields can be directly compared against ground truth flow fields obtained from in-situ measurements [45]. Philip and Pang [1] identified rip currents by looking at the distribution of detected flow velocities which might include motions due to wave action, possible rip, and motion in the background scenery. They assumed the rip current to be in a *single* seaward direction, with rip regions above a certain velocity threshold and region size. Unfortunately, rip currents do not necessarily flow in a straight line, leading to false-negative results. Also, rip currents often have feeder currents that travel alongshore, bringing water that feeds the rip current's main body. The work reported in this paper builds upon [1] to detect non-linear rip currents and feeder currents, and also include comparisons with alternative detection methods.

More recently, Anderson et al. [46] introduced WAMFlow (wave-average movies) where they pre-filter the video source prior to obtaining the optical flow field. The pre-filtering aims to remove the dominant signal due to incident waves while leaving the signal due to foam or water turbidity features that might indicate the presence of rip currents and non-stationary circulation cells. In contrast, the work presented here applies the filtering on the optical flow field derived from raw video, while achieving real-time processing for the proposed pipeline. Furthermore, the focus of this paper is on adapting flow visualization methods to detect and visualize the presence of rip currents.

## 2.3 Data and Pre-processing

### 2.3.1 Data Sets

The data set used for our investigation is composed of 27 video clips collected from the web, photographed by the authors, or obtained from collaborators. The collection includes cases with rip currents, no rip currents, and possible weak rip currents. Those that contain rip currents include different types of rips: rips with curvature, sediment plume, a dark channel between breaking waves, and structural rips. The data set also contains cases with swash (water movement on the shallow part of the beach after a wave has broken all the way to the high water mark where a wave can run up the beach) and reflection waves (or backwash off a rocky shoreline). We exclude some video from consideration because they are not suitable for flow-based analyses e.g. unstable/shaky video, contains camera pans/rotations, poor video quality e.g. blurry, duration is too short for time averaging. The videos ranged in size from 320x400 to 1920×1080, and ranged in length from 11 seconds to 60 seconds, recorded at 30 fps. In this paper we downsampled large videos to 720-pixels height and the corresponding width to keep their aspect ratio.

### 2.3.2 Data Requirements

In an ideal scenario, the video is taken from a stabilized mount, at as high an elevation as possible, with some beach in the foreground, on a sunny day, with sufficient duration and resolution. Webcams around the world, especially surf cams where there is a higher chance of finding rip currents, offer a rich potential for videos. However, most of them are not configured for rip current detection.

Flow analysis requires sufficient pixels imaging the rip current. With the settings of the optical flow method we used, the minimal width of rip current that the estimated flow field can correctly represent was roughly 80 pixels. For most of the videos obtained from a webcam that is sufficiently close to the beach, we found that image resolution of at least 640x480 is required. When the relevant section of the ocean uses only a small part of the frame, increased resolution is needed. The increased resolution does increase the computational requirements. Therefore, videos that are larger than 1080x720 are either scaled down to a height of 720 while maintaining the aspect ratio, or cropped to the region of interest.

The camera should be steady or fixed and not contain panning, rotations, zooms since frame to frame correspondence is needed to obtain the optical flow field. Small camera vibrations or drifts can be compensated by video stabilization, but excessive shaking will also make the video unsuitable.

Sufficient duration is needed to observe a rip current, usually at least one minute or at least three wave periods. The justification is that rip speed is related to breaking wave height, which in turn varies with infra-gravity wave motion, commonly called wave "sets". A wave set typically varies on the order of minutes, and we find that around one minute of video is sufficient. However, a longer video would be preferable especially for longer period waves.

### 2.3.3 Video Pre-processing

Video must be stabilized prior to extracting the optical flow field. Videos sources from webcams can omit this step, while videos from drone or hand-held cameras can benefit from this step. Optical flow estimates motion based on differences of local neighborhood pixel values in consecutive video frames, and camera motion will produce a confounding flow field. Many video stabilization methods exist [47] [48] [49] [50], and robust automatic stabilization is possible. In this work, we simply use the Warp Stabilizer feature of Adobe After Effects. The Warp Stabilizer tracks selected static objects in the scene (e.g. a pier, rocks) and stabilizes the video using them as fiducials. We supplement with manual adjustment as needed, typically when the video contains insufficient rigid landmarks.

## 2.4 Flow-based analysis

In this paper, flow-based analysis refers to the analysis and visualization of a vector field derived from optical flow of video clips, with the goal of finding rip currents. We investigate Color Maps and Pathlines as standard baseline flow visualization methods and find them lacking. We then investigate the enhanced methods of Filtered Arrow Glyphs and Timelines and find that they perform well.

### 2.4.1 Optical Flow Map

Optical flow map refers to the velocity of pixels in an image derived from motion of neighboring pixels in consecutive frames from a video. Several optical flow algorithms and surveys exist [51] [52] [53] [54], and produce a flow field describing where each pixel moves from one frame to the next. Our implementation uses the OpenCV library [55] for computation. Dense optical flow is calculated at every pixel using Farneback's method [51], and used for visualizations methods that require it, such as pseudo-coloring and flow difference. To save computational cost, sparse optical flow is calculated using Lucas-Kanade [52], and used as input to the remainder of the visualization methods.

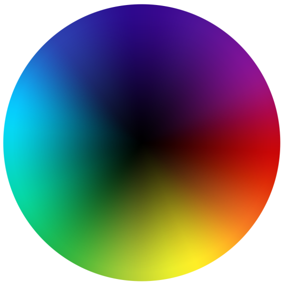
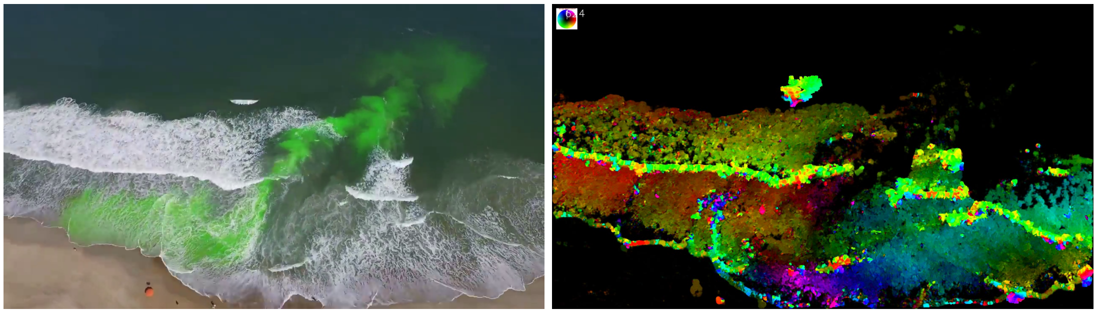


Figure 2.2: **The Color Wheel Used to Map Flow Vector to Color:** The flow directions are mapped to hues, while the relative magnitudes are mapped to value (darker indicating smaller magnitudes). For example, red indicates a flow towards the right.



(a) Original Frame

(b) Single Frame Color Map

Figure 2.3: **Single frame color map is not usable:** The original frame with a rip current is shown in (a), and the corresponding single frame color map is shown in (b). Notice that the rip current traced using the green dye in (a) corresponds to the dark spot in the center of the color map in (b). Usually, rip currents are much slower than incoming waves, and its relatively weak signals are not easily observed in a single frame color map. This baseline visualization method is difficult to interpret and thus not directly usable.

### 2.4.2 Baseline Visualization Methods

#### 2.4.2.1 Color Map

The most basic method of visualizing a scalar field is pseudo-coloring or color mapping. In the case of a vector field such as the vector field obtained from the optical flow across frames in a video, direction and magnitudes are mapped separately to different color properties. Specifically, we use an HSV color model where flow direction is mapped to hue (what is normally referred to as color), while flow magnitude is mapped to value (brighter or darker shade of the color). Saturation is set to one. Figure 2.2 shows the color wheel that we use to map the flow direction and magnitude to hue (color) and value (darker towards the center of the color wheel) respectively. For example, brighter red indicates a strong flow towards the right, while a darker red indicates a weaker flow towards the right.

Color mapping vector fields involve first converting vector information from Cartesian to polar coordinates to obtain angles and magnitudes from the 2D vector components. The magnitudes are normalized so that they range from 0..1. Angles are then mapped to hue while normalized magnitudes are mapped linearly to value. In Figure 2.3, the green fluorescent dye on the left image indicates the location of the rip current. On the right, is the corresponding color mapped image. The bright green regions on the right indicate a strong flow direction towards the bottom left of the image. The darker regions of the right image correspond to regions with weak to no movement.

Notice that these regions correspond to the beach and areas past the surf zone, as well as the rip itself. Since the video has been stabilized, the darker color where the rip is located indicates that the speed of the rip is much slower relative to the rest of the surf zone. While this visualization allowed us to see velocity information, it was confusing to general viewers and is not usable by itself for alerting viewers to the presence of rip currents.

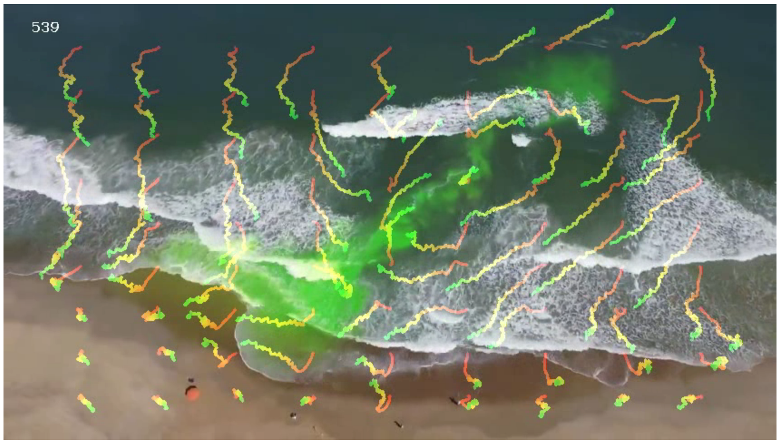


Figure 2.4: **Pathline Visualization of Rip Currents:** A visualization of the flow field obtained by optical flow is shown using the baseline method of pathlines. While they somewhat capture the signal of rip currents, the visualizations are not very clear.

#### 2.4.2.2 Pathlines

Pathlines record the trajectory of a massless particle in a time-varying vector field [56]. The optical flow field derived from video analysis is a time-varying vector field $\vec{v}(\vec{p},t)$ where the velocity $\vec{v}$ is known for each location $\vec{p}$ at time $t$. The pathline is obtained by integrating $\frac{d\vec{p}}{dt} = \vec{v}(\vec{p},t)$ We use an explicit fixed step 4th order Runge-Kutta integration. If a seed point is placed in the vicinity of a rip current, we expect it to be drawn into the rip, and leave a trace of its path from the seed point towards the rip in the process. On the other hand, those seeded outside a rip zone would not be affected and would likely just be washed ashore by the incoming surf. Aside from the trajectory itself, a pathline can be colored to show some other properties such as: (a) age of the particle, which is useful to see if the trajectory is progressing seaward or not; (b) length of trajectory, which is useful to gauge relative speeds; and (c) distance from starting point, which is useful to see if a particle takes a circuitous/cyclic path or a more direct route.

To test this method, we seeded a regularly spaced 9 x 9 sampling grid. Figure 2.4 shows the 81 pathlines. Pathlines are colored by age, starting with red and getting cooler over time. Unlike Figure 2.3, this figure shows the cumulative effects of the flow field on the 81 seed particles, rather than an instantaneous snapshot of where the 81 particles are located. Here, we see that the trajectories are erratic (even after video stabilization). Nevertheless, one can see that pathlines are indeed headed out in the rip, and even further beyond the outer boundary of the green tracer dye. For this method to be a rip detector, one would have to filter out the pathlines that are not in the rip. However, we cannot simply remove pathlines that are seaward, assuming one knows which direction is seaward in any given video source, because of the erratic trajectories but more importantly rips may have a circulating pattern. Because of these reasons, coupled with the clutter evident even with just a few pathines, we do not investigate this further as a detection method.

### 2.4.3 Modified visualization methods

Straightforward application of visualization methods such as the two mentioned in the previous section cannot detect rip currents. Here, we describe how domain knowledge coupled with modifications to visualization methods are necessary to arrive at effective detection and visualization of rips.

#### 2.4.3.1 Filtered Arrow Glyphs

This is an extension of an arrow plot where an arrow is present only when a potential rip current is detected. Feeder currents that can bring material into the main rip are also included. An example of this method is shown in Figure 2.1 (a). Because glyphs can quickly clutter the display, we need to balance the amount of information versus the amount of clutter. Here, each frame is represented by a grid of velocities with 20-pixel spacing to ensure sufficient representation of the flow field. The idea is to alert the user to areas that might potentially be dangerous while leaving the rest of the imagery untouched.

The optical flow map detects movements across frames. Movements of clouds as well as people or pets walking on the beach are also detected. Such movements are irrelevant to rip detection so we apply a mask to ignore sky and beach regions while passing the ocean region for further processing. Because of the nature of wave motion where the predominantly observed motion is that of the incoming waves, we found that there was insufficient signal to detect rips when analyzing the flow field on individual frames. To remedy, we construct a time-averaged vector field over three wave periods. In conditions where the sea state can induce rip currents this ranges from 15-45 seconds and covers short period wind chops to longer period swells. The motivation for time-averaging is that while water may get pushed in with each wave, they also recede back to the ocean. However, the region where a rip current is situated often times have less breaking waves and hence the outward flow from a rip is more persistent and easier to detect from the time-averaged flow field.

Figure 2.5 shows the time-averaged flow fields for the duration of 1 and 3 wave periods. The scene contains a rip current that flows from the left and to the upper right. The corresponding color maps, described earlier, of the time-averaged flow field highlights this rip current in purple, indicating its direction. In this figure, the primary focus is to highlight flow direction and not on flow magnitude. Hence, we temporarily set all the magnitudes to 1.

Notice that the color map of the one wave period averaging has another purple region in the right bottom corner of the surf zone. These false positives can be caused by transient receding water in swash zones. However, with the three wave period averaging, the false positive disappears.

Simply displaying all directional arrows produces a cluttered and confusing image. It is necessary to filter the data so that only the relevant information is shown i.e. the regions of a rip current. To achieve this, once the time-averaged vector field is constructed at each arrow position, all arrows are grouped into six bins, each representing a range of 60 degrees. The bin with the highest frequency represents the predominant flow direction, which we assume to be that of the incoming waves. The opposite direction is then assumed to be the rip direction. Vectors in this bin are represented by red arrows, while vectors in the neighboring bins on either side are represented by yellow arrows. The yellow arrows can potentially show feeder currents. They are also useful when the rip current direction extends beyond the bin that represents the rip direction e.g. for rips with high curvatures. In short, filtered arrow glyphs involve both masking out regions of a frame that is not part of the body of water, and only displaying arrow glyphs of time averaged vectors that are in the rip and feeder directions.

While this assumption of reverse flow in a rip current is simplistic, cases where the rip is quite obvious can be highlighted using this technique as in Figure 2.1 (a). Notice that the region with rip current is obviously marked, in contrast to Figure 2.1 (c) showing an ocean scene with no rip current, and no annotation.

A caveat of this method is that the direction of some rips are not necessarily opposite

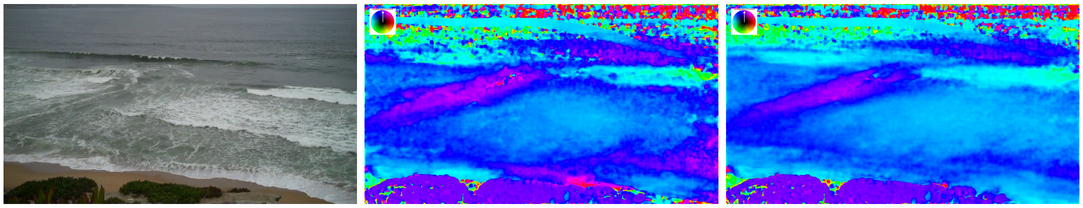
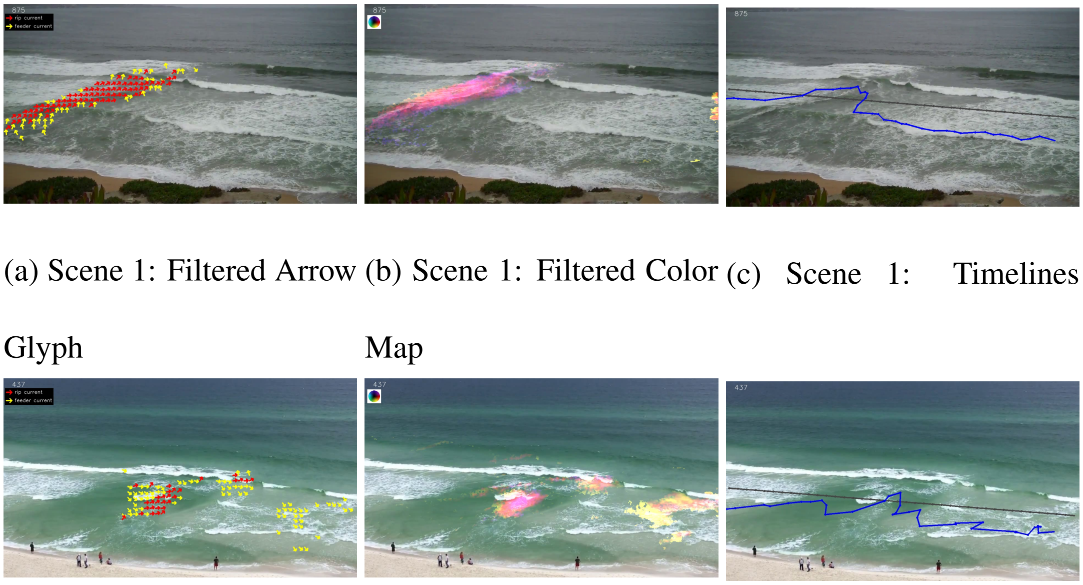

(a) Original frame (b) 1 wave period (c) 3 wave periods

Figure 2.5: **Averaged Flow Field Comparison:** Comparison of the averaged flow field color map for the duration of 1 and 3 wave periods. Original frame is shown in (a). Images in (b) and (c) are the color map of the obtained flow field, where direction is mapped to hue. In (b) and (c), the purple regions on the left correspond to rip current, indicating that there is a seaward directional flow there. Notice that the false positive in the right bottom corner disappears after averaging for 3 wave period.

the incoming wave direction e.g. when waves arrive at an oblique angle to the shoreline, the rip may be as little as 90 degrees from the predominant wave direction. To be able to visualize such cases and any potential feeder current, we experimentally chose 6 bins. With 8 bins, feeder currents tend to be ignored more often, and with 4 bins, it produces much more false positives.

The filtered color map, such as the ones shown in Figure 2.6(b) and (e), is generated in a similar fashion as the filtered arrow glyph. Rip direction and feeder current directions are first determined using the grid of velocities with 20-pixel spacing and the binning strategy. The optical flow for all pixels is computed, time-averaged, and mapped to color as described in section 2.4.2.1. Parts of the color map image is then masked out if they are not in the rip or feeder current directions. Finally, this image is blended with the corresponding frame of the video. Note that regions with lower flow magnitude (low value) will contribute correspondingly less color to the blended image.



(a) Scene 1: Filtered Arrow Glyph (b) Scene 1: Filtered Color Map (c) Scene 1: Timelines

(d) Scene 2: Filtered Arrow Glyph (e) Scene 2: Filtered Color Map (f) Scene 2: Timelines

Figure 2.6: **Evaluations of The Methods:** These evaluation videos show a prominent rip which can be easily seen on the left side of the frame in Scene 1 and on the center of the frame in Scene 2. In both scenes, all visualization methods clearly show where rip current is located. In (a), the arrows covers the body of the rip. In (b), the rip is highlighted in pink, indicating its direction. There is another region to the right where the flow direction is close to that of the rip direction and is highlighted in yellow. However, we can safely rule this out as it's not connected to the main rip and is mainly alongshore. The rip is also highlighted by the single timeline in (c). The gray line indicates the initial position of the timeline. In (d) and (e), the rip current is highlighted in the center of the frame. The region on the right side of the frame with lower right directional flow indicates this rip may circles back to the shore. (f) shows that the rip is highlighted by the single timeline.

#### 2.4.3.2 Timelines

Timelines is another flow visualization technique for analyzing time-varying flow fields. An example of this method is shown in Figure 2.1 (b) In the context of rip current detection, it represents a chain of virtual buoys tethered together by a massless and stretchable rope. When placed in the surf zone close to and along the shoreline, we would expect virtual buoys in the rip to be pulled out to sea, and those nearby will be pulled by feeder currents towards the main rip and eventually out to sea. The rest of the virtual buoys will likely be washed ashore. We introduce a new timeline in the surf zone every 30 seconds in order to track any new currents that may lead to the identification of pulsing rip currents. Virtual buoys along the timeline are initially spaced out at regular intervals. Over time, the relative spacing between adjacent buoys provide additional information if there is a large velocity gradient. Figure 2.1 (b) illustrates a case where a rip is detected. The timeline is clearly deformed and extends out along the rip channel. Contrast this with Figure 2.1 (d) for which no rip current is present and therefore the timeline is relatively undeformed. A grey line indicates the initial placement of the timeline in both cases. When waves are pronounced and the incident angle of the waves is perpendicular to the shoreline, the optimal placement of the timeline is parallel to the beach and in the middle of the surf zone because the rip current direction will likely have a direct seaward heading. However, when waves are weak or the incident angle of the waves is oblique to the shoreline, the rip current direction may not be directly seaward but also at an oblique angle. In such cases, we place an additional timeline perpendicular to the shoreline to better see rips that may form at an angle from the shoreline.

The process of calculating and generating the timeline is similar to that of pathline tracing. Each virtual buoy on a timeline is treated as a seed point of a pathline and its trajectory is calculated using RK4 integration. However, rather than tracing the evolution of each seed point independently, a timeline is drawn by connecting the points from all the virtual buoys, in the same order, to form a curve. As time progresses, the timeline is thus animated.

How fast a timeline moves depend on the local flow velocity. A fast moving wave could potentially push a timeline all the way to the shoreline. Such large displacements are not conducive to detecting rip currents. Simply reducing the integration step size will just increase computation cost without addressing the underlying problem. Just as we saw in Figure 2.5, averaging the flow field for at least three wave periods is crucial. Therefore, the timelines need to move slower than the actual speed of the waves to allow extraction of rip currents. For this reason, we multiply the optical flow field in each frame by an adjustment factor $\alpha$. The $\alpha$ is calculated as $\alpha = d/(\delta \cdot f)$. $d$ is the pixel-wise distance between the initial placement of the timeline and the shoreline, $\delta$ is the pixel-wise velocity of the incoming waves, and $f$ is the total number of frames. For the videos that are longer than three minutes, we cap $f$ to the number of frames that corresponds to three minutes in order to correctly capture transient rips. With the adjustment factor $\alpha$, the timeline reaches the shoreline at the end of the specified number of frames. These slower moving timelines provide stable results.

## 2.5 Results

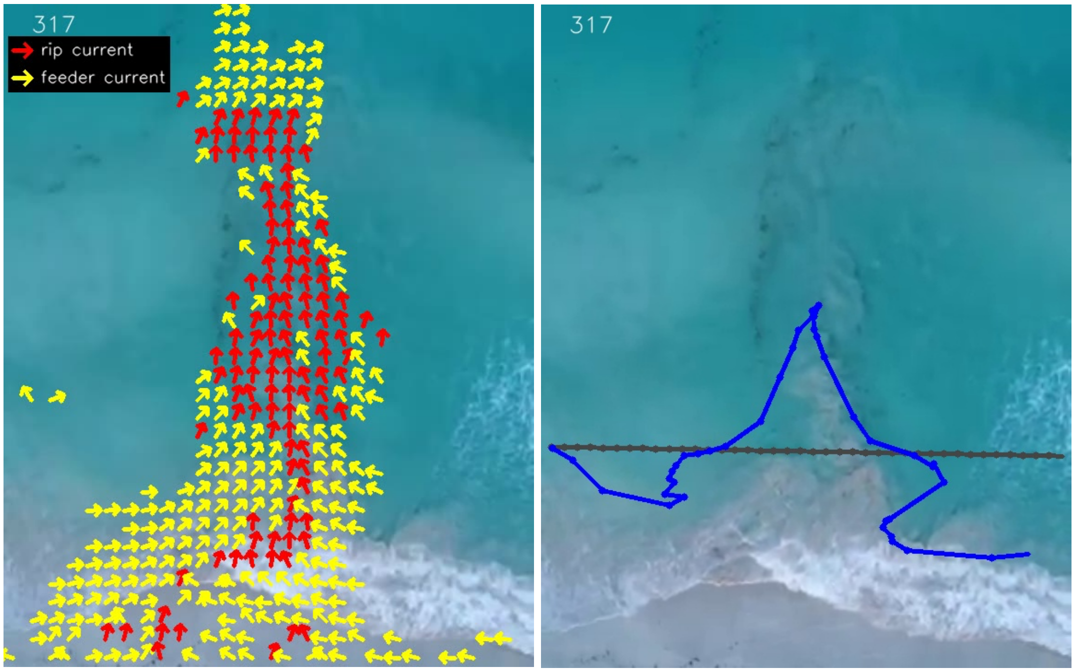
In this section, we first evaluate the two modified flow visualization/detection methods, and then compare their performance against other methods from literature.

### 2.5.1 Evaluation of Proposed Methods

We tested the two modified methods on different data set including visually obvious cases containing strong rip currents, a rip with sediment plume, and less visually apparent cases with weaker rip currents. We then applied the both methods on human labelled data set to confirm or challenge previously annotated results by rip current experts.

*Strong rip currents:* Figure 2.6 shows two examples of strong rip currents of the type that prior methods can also detect. The filtered arrow glyph visualizations in (a) and (d) are consistent in showing regions of flow against the predominant incoming wave direction. The arrows in the neighboring bins are also displayed in yellow, showing how water may feed in and out of the rip.

As noted in section 2.4.2, applying color maps on the flow field on a per frame basis does not help in detecting rips. We modified that method in a similar fashion as the filtered arrow glyphs, by time-averaging prior to color mapping, then filtering out non-interesting regions. This modification is illustrated in Figure 2.6(b) and (e). Rip currents can also be seen in one quick glance. It does require one to look at the color



(a) Filtered Arrow Glyph

(b) Timelines

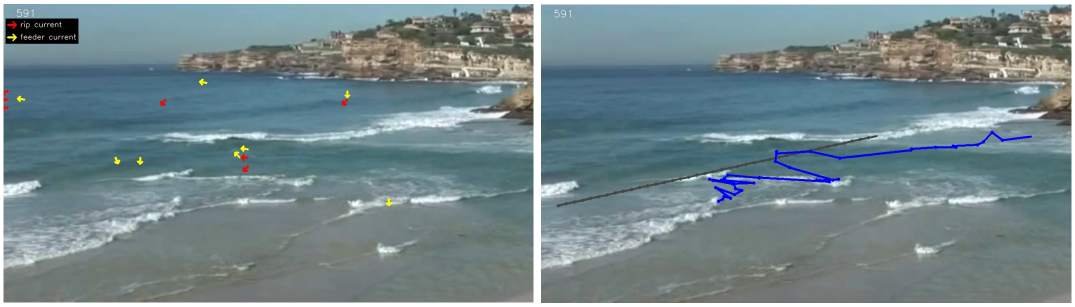
Figure 2.7: **Visualizing Sediment Plumes:** This example showcases a rip with no breaking waves but which has a sediment plume. The existing machine learning methods fail in this type of data sets where there is no breaking wave features that they used for training their models. However, the optical flow methods perform well on this data set. The filtered arrow glyphs show there is a strong and obvious rip in the center. The feeder current on the both sides of the rip is also highlighted, providing additional information of the rip. The deformation of the timeline coincides well with the rip channel.

wheel to confirm the flow direction, whereas this step is not necessary with the filtered arrow glyph. Hence, we omit this method from further consideration.

The timeline visualizations in (c) and (f) also succeeded at showing the rip based on their protrusion or deformation. The astute reader may notice that the timeline protrusion in (f), as well as the position of the arrow glyphs and colored regions in (d) and (e) are on the right region of the darker channel in the surf zone, this indicates the region within the rip zone with strongest velocities.

*Sediment Plumes:* Figure 2.7 shows an example of a rip where the predominant visual signature is an obvious sediment plume. The filtered arrow glyph and the filtered color map clearly show the seaward flow caused by the strong rip. Furthermore, feeder currents from both sides of the rip are highlighted when this method is applied. Swimmers in a feeder current may eventually end up in a rip and swept away. Therefore, it is crucial to visualize these regions as well.

*Weak rip currents:* Figure 2.8 shows a harder case. Even though the visual signature indicates an obvious rip where one sees the darker channel between breaking waves, the velocities are quite low. Hence, Figure 2.8(a) and Figure 2.8(b) do not provide much indication of a rip current. The slight differences in velocities at the rip channel and the other incoming wave regions are however enough to deform the timeline, indicating the presence of the rip at the center of the frame. The timeline method is more sensitive because it can accumulate the small deformations from weak velocities.



(a) Filtered Arrow Glyph (b) Timelines

Figure 2.8: **Timeline Example:** This example showcases the performance of the timeline method when the flow in the rip is weak. Due to the lack of wave textures in the rip region as well as the quality of the video, the optical flow method detected very low velocity in the leftward flowing rip. We can only see a a few arrow glyphs since most of them were filtered out. However, in (b), the ability of timelines to capture cumulative movement effects shows shows a deformation. While most of the timeline has washed in past the gray line, the deformation of the timeline in the darker region indicates a rip, albeit pretty weak and not exceeding much past the original gray line.

Figure 2.9 showcases another difficult case with no obvious surf zone and just shore break. In this scenario, the placement of an additional timeline perpendicular to the shoreline is helpful in visualizing the signal of a rip current that was present. Here, the deformation of the blue timeline indicates a stronger longshore component compared to the green timeline showing seaward component.

### 2.5.2 Comparison with Human Labelled Data Set

Rip current researchers at NOAA have approximately a year's worth of labelled Timex images (from 10 minute video segments) collected from the 40th Street Miami Beach webcam. This particular webcam is part of a larger network of webcams (SECOORA) for coastal monitoring which includes beach erosion and rip current forecasting

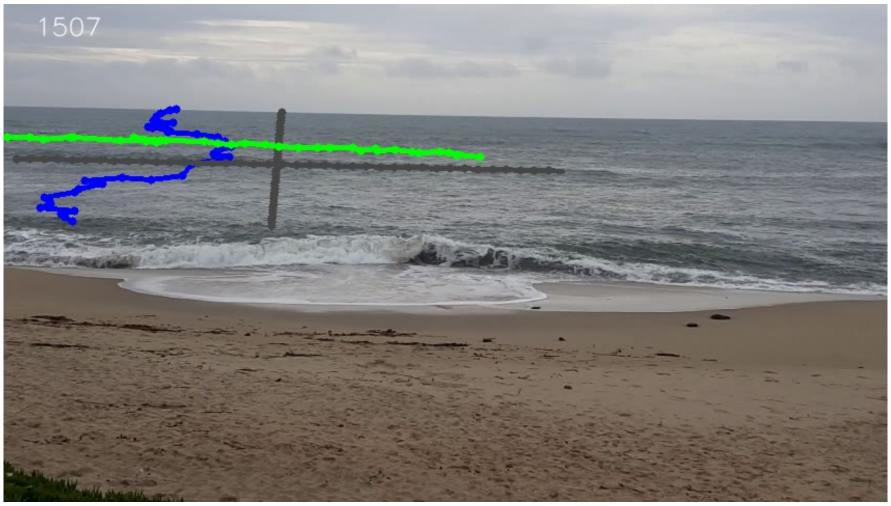
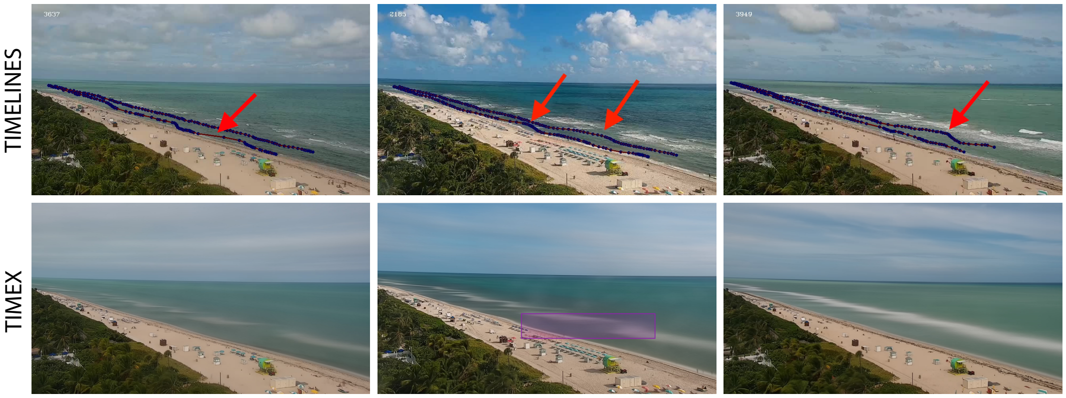


Figure 2.9: **Timeline Example:** This example shows a rip that is not detectable using filtered arrow glyphs nor filtered color maps (and therefore omitted). We do see an seaward movement of the green timeline that is placed parallel to shore, but do not see any deformation to indicate a particular region that may be moving out faster than the rest. Placing another timeline, colored blue, perpendicular to shore, we can see a more pronounced deformation indicating a strong longshore current.

[57].

Each Timex image represents a 10 minute video segment, and were labelled by human experts as `definite yes`, `maybe yes`, `maybe no`, and `definite no` with regards to the presence of rip currents. Figure 2.10 shows typical views from this webcam. We note a few positive and negative aspects about this particular data set. There is minimal camera movement since the camera is on a fixed mount. Camera placement is pretty high but has a wide view. While the wide field of view is good for monitoring coastal erosion, it is less than ideal for monitoring rip currents. About 70% of the frame contains non-interesting parts i.e. sky, vegetation, sandy beach. Furthermore, due to the



(a) Original: Maybe yes (b) Original: Definite yes (c) Original: Definite No

Figure 2.10: **A comparison of *timeline* visualization with *Timex* images commonly used for human labelling.** The top row shows timelines. A new timeline was released in the surf zone every 10 seconds and tracked for the next 20 seconds. In general, timelines were pushed onto the shore by incoming waves except for regions of potential rips. The bottom row shows Timex. In example (a), the Timex was originally labelled as `Maybe yes` to contain rip currents but after consulting our visualization and the complete video, the label was changed to `Maybe no`. Note that this frame of the video did show some changes in spacing of the points along the first timeline. It indicated velocity changes running parallel to the beach or a longshore current. This behavior would not have been detected using the Timex image since it was inside the region of breaking waves and not visually apparent. In example (b), the label was changed from `Definite yes` to `Maybe yes`. There were slight bumps noticeable on both timelines but not considered significant. These slight bumps indicated possibly some weak rip currents. In example (c), the label was revised dramatically from `Definite no` to `Definite yes`. The second timeline showed a fairly significant bump showing where the rip was located. Examples (a) and (b) both had the Timex characteristic darker region in between lighter regions of breaking waves. In (c) the Timex darker region was not present due to stronger surf conditions during that period, which was why the initial human label relying only on the Timex gave incorrect labels.

severe perspective distortion, only the portion of the water closest to the camera have sufficient spatial resolution (distance/pixel) to obtain a meaningful optical flow field. The camera aside, the Miami beach itself is a long sandy beach with a shallow gradual sloping bottom which does not lead to large breaking waves and is geographically protected from large swells by the Bahamas. Hence, the rip currents that do form here are typically weak.

We applied the two modified techniques (filtered arrow glyphs and timelines) on the 10 minute video clips for a random sample of labelled Timex data set. We quickly dismissed the utility of the filtered arrow glyphs and filtered color maps for this data set because of the distant camera set up which made the velocities very low. Coupled with the relatively slow wave propagation, this meant little changes in neighboring pixel values and therefore decreased signal to noise. However, because the displacement of timelines captures the *cumulative effects* of the flow field, even those from weaker rips, we were able to confirm the human labels on most of the cases that we looked at. We did find instances, such as those shown in Figure 2.10, where the timelines suggest a different label than the human labels. We presented these to the experts who originally labelled the data set and asked them to view the video associated with the Timex images to make a more careful determination. In these instances, the experts changed their labels after reviewing the video. Note that the label of `maybe yes` seem to be used for cases where the rip is considered weak or the rip signal is not conclusive.

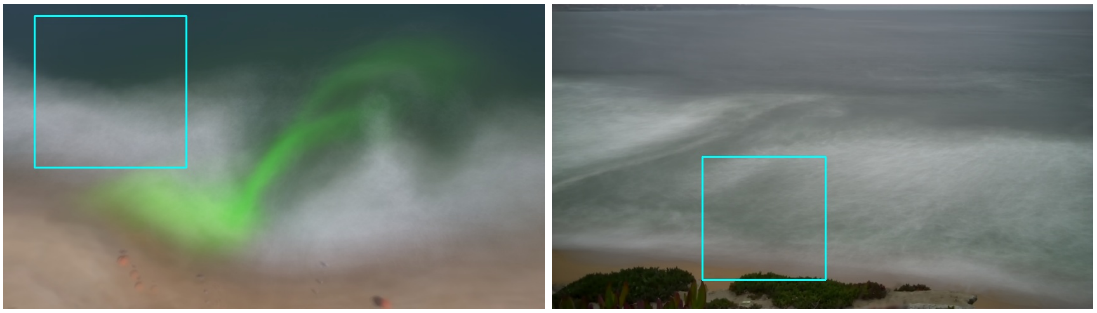
While rip detection using Timex images is less time consuming than watching video clips, and the signature pattern of rip currents in Timex images are easier to discern than looking at raw video by humans, solely relying on them can lead to incorrect classifications. Our small study demonstrates the utility of timelines for more accurate labelling compared to Timex images alone. The human experts indicated that the visualizations were valuable and contained cues which did not exist in Timex visualizations. For study locations with fixed camera mounts, accompanying Timex images with our visualization output can help improve labelling accuracy. This is an important process in collecting good training data for building automated classification models using machine learning methods such as those reported in [40][42].

### 2.5.3 Comparison Across Methods

We compared how the different published methods detected rip currents in the video clips shown in Figures 1, 6, 7, 8, 9, and 10. The following video clips contain rip currents: Figures 1(row 1), 6, 7, 8, 9, 10(column b), and 10(column c). The following video clips do not contain rips: Figures 1(row 2) and 10(column a). Table 2.1 summarizes the results. A *yes* entry means the method correctly detected the rip if present (true positive), or correctly marked the video as not containing a rip (true negative). A *no* entry means the method gave a false positive or false negative result.

#### 2.5.3.1 Comparison with Machine Learning Methods

Currently, there are two rip detection methods that utilize machine learning methods. Maryan et al. [40] employ a Viola-Jones framework [58] to train their model to detect rips from Timex images (see Timex column in Table 2.1), while de Silva et al. [42] used a modified deep learning technique Faster RCNN [59] with an accumulation



(a) Rip detection using Timex of Figure 1(row 1).

(b) Rip detection using Timex of Figure 6(row 1).

Figure 2.11: **Rip Detection Using Timex Figures:** Two examples where rip detection using Timex images failed even when the visual signature of the rips are very strong.

buffer to aggregate frames across time to improve prediction. That model was trained to detect rips from images and video clips (see the Faster RCNN column in Table 2.1). Rip detection that relies on Timex images are inherently limited to bathymetry controlled rips where the visual signature is a darker channel in between breaking waves. The model described in [42] is also limited to bathymetry controlled rips. However, that limitation is not inherent to the deep learning technique they used but rather the training data they used to train their model. Given these considerations, we would expect both detectors to do well in detecting bathymetry controlled rips and not well with other types of rips.

The Timex and Faster RCNN columns in Table 2.1 show how the machine learning methods fare in classifying and detecting rip currents . Our expectation is that Timex should perform well on all but Figures 7 (sediment plume) and 9 (no visual signature).

Looking at the actual tests, we see that Timex performed poorly. Figure 2.11 shows two rips with strong visual signatures where the method based on Timex images failed. When we reran the data but fed individual frames to the model that learned to classify Timex images, the results improved slightly. There were 69% of the frames from Figure 6(1) that were correctly labelled. Hence, we marked Figure 6(1) as a *yes* in the Timex column.

#### 2.5.3.2 Comparison with Previous Flow Based Method

The flow-based method described in [1] also analyzed the optical flow field derived from the video. However, it did not account for wave pulsing and used simple statistics to guess the seaward rip direction. While this may work in the typical bathymetry controlled rips, the assumption that rip direction is always straight also makes this method less flexible. Furthermore, the visualization of the detected rip is a simple reddish region to warn presence of rip and does not impart any velocity information. Figure 12 shows the output of this method on the data set shown in Figure 6(2). The highlighted regions corresponds well with the red arrows in Figure 6(d) and the pink regions in Figure 6(e). Somewhat surprising, it failed on both Figure 6(1) and Figure 8. Upon reviewing those two videos, the speed of the rip is very slow. So, even while there are strong visual cues in both cases, the per frame velocities at the rip regions were low and appeared spurious and therefore did not pass the threshold set by the method. This

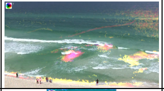
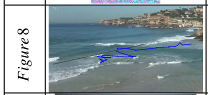
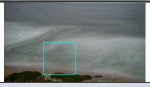
|  | Video Clip | Timex[40] | Faster RCNN[42] | Optical Flow[1] | Image Processing[28] | Filtered Color Map [Ours] | Filtered Arrow Glyph [Ours] | Timeline [Ours] |
|---|---|---|---|---|---|---|---|---|
| *Figure* 1(1) | | *no* | *yes* | *yes* | *no* | *yes* | *yes* | *yes* |
| *Figure* 1(2) | | *no* | *yes* | *yes* | *no* | *yes* | *yes* | *yes* |
| *Figure* 6(1) | | *yes* | *yes* | *no* | *no* | *yes* | *yes* | *yes* |
| *Figure* 6(2) |  | *no* | *yes* | *yes* | *no* | *yes* | *yes* | *yes* |
| *Figure* 7 | | *no* | *no* | *yes* | *yes* | *yes* | *yes* | *yes* |
| *Figure* 8 |  | *no* | *yes* | *no* | *no* | *no* | *no* | *yes* |
| *Figure* 9 |  | *no* | *no* | *no* | *no* | *no* | *no* | *yes* |
| *Figure* 10 | | *no* | *no* | *no* | *no* | *no* | *no* | *yes* |
|  | *Ratio* | 1/8 | 5/8 | 4/8 | 1/8 | 5/8 | 5/8 | 8/8 |

Table 2.1: **Method Comparison:** Comparison of Filtered Color Maps, Filtered Arrow Glyphs, and Timeline methods against other published methods that analyze images and/or videos. A *yes* indicates a (mostly) correct detection. The last row indicates the ratio of # correct to total. Notice that none of the existing methods is able to provide correct detections in all of the videos. In contrast Timelines provides correct detections in all eight videos.

method also failed in Figures 9 and 10. As mentioned earlier, the velocities in Figure 9 are also very low and per frame velocity analyses fail. In Figure 10, because the camera is so far away, each pixel covers a much larger area and therefore less sensitive to velocity changes.

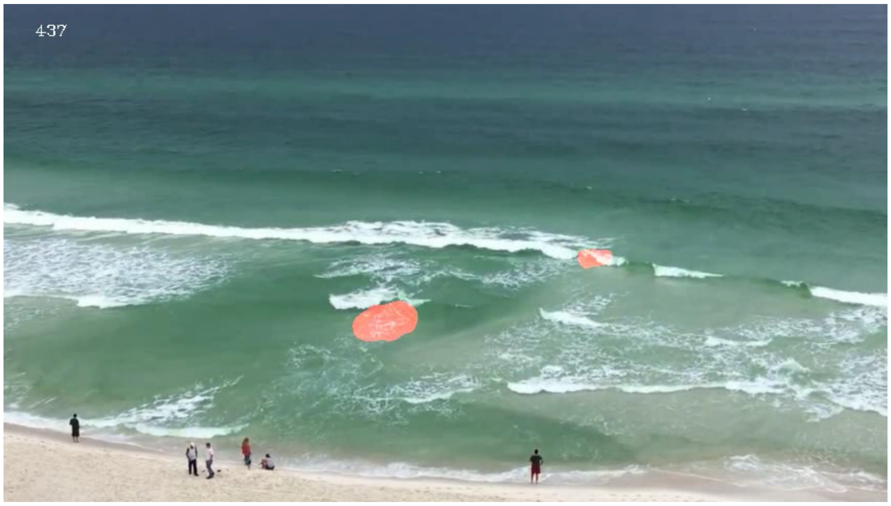


Figure 2.12: **Optical flow method failed:** The same frame as Figure 6(2) but using the method from [1].

#### 2.5.3.3 Comparison with Image Processing Method

The image processing method for rip detection described in [28] detects transient rips in video feeds from a fixed webcam. Rips at this location are characterized by discoloration in the water that extends some distance from shore. Based on the camera orientation, the authors rectified the frames and set a threshold line some distance from shore where any discoloration beyond that line is considered a rip. Given that this method is designed to detect rips with sediment plumes, we expect that it should do well with Figure 7, assuming that a threshold line has been set up as well. Indeed, this is what we observe with this approach in Table 2.1. However, this method is specialize to detect only this type of rip and does not detect other types of rips.

We provide Table 2.1 summarizing the performance of our methods and published methods on the 8 rip current videos shown in this paper. We find that the published methods are each limited to a specific class of rip currents and none of the existing methods are able to correctly label all videos. In contrast, all of the flow based visualization methods work at least as well as the best alternate method, and Timelines in particular is able to correctly identify the presence of rip currents in all 8 of the videos.

## 2.6 Implementation Notes

We implemented the methods above with C++ along with OpenCV library. We used Alienware m3 R15 with RTX 2070 Super for computation, and it consumes roughly 1GB of memory. We used GPU acceleration mainly for optical flow computation, and with a dedicated GPU, our method ran in real-time with 30 FPS on 1080x720 resolution.

## 2.7 Summary and Conclusion

We investigated rip currents detection using optical flow analysis on video clips. This is complicated because rip currents are amorphous without well defined boundaries and ephemeral without well defined temporal bounds. The problem is further complicated by the presence of a dominant quasi-periodic signal from wave motion. We found that the straight-forward application of flow visualization methods did not yield good results. The main contributions of this paper are the modifications to standard flow visualization methods in order to equip them with rip current detection capability without which the detection is difficult or impossible. The modifications significantly improved upon earlier flow based method both in terms of ability to detect subtle rips and clarity in visualization. Our study also shows that our proposed flow based visualization results in improved human judgements versus the existing dominant method of viewing Timex images.

Flow-based approach is a valuable tool to have in our arsenal of rip detection methods. It is best suited for situations where a stable platform for a camera is available e.g. surf cams. In such settings, site specific customizations e.g. placement of timelines can be employed to make the approach fully automated. The work presented here has shown sufficient success that it will be deployed on the SECOORA network of webcams.

The flow visualization techniques such as timelines, show great promise in visualizing rip currents. One of the things that we noticed in rip current videos is the appearance of the rip current is very unique, compared to the other parts of the ocean. In Chapter 03, we look at a deep learning method that can find rip currents by using its appearance as a method of learning.

# Chapter 3

# Using Machine Learning to Detect Rip Currents by Appearance

## 3.1 Introduction

Rip currents are the most significant safety risk to swimmers along the coastlines of oceans, seas, and large lakes. [60, 6, 61]. The majority of beach goers do not know how to identify rip currents, and there is no robust and reliable location-independent method to identify them. Globally there are thousands of drownings each year due to rip currents [7, 8]. A 20 year study by the US Lifesaving Association reports that 81.9% of the 37,000 beach rescues each year are due to rip currents [60]. Lifeguards are often trained to identify rip currents. However the majority of drownings occur on beaches without trained personnel [21, 22]. Posted signs can provide a warning, but there is evidence that most people do not find existing signs helpful in actually identifying rip currents [23]. There has been no decline in the number of associated drowning fatalities, despite warning signs and educational material.

Rip currents are a well-studied ocean phenomenon [15, 16, 14]. They are defined as strong and narrow channels of fast-moving water that flow towards the sea from beaches. When waves break, they form a “setup” or an increase in mean water level. This setup can vary along a shoreline depending on the amount or height of breaking waves. Rip currents form as water tends to flow alongshore from regions of high setup (larger waves) to regions of lower setup (smaller waves) where currents converge to form a seaward flowing rip. Furthermore, macrovortices, induced by alongshore uneven wave breaking, may also be a contributing factor for rip current formation and evolution. [62, 63, 64, 65]. The speed of seaward rips can be quite strong reaching 2 m/s, faster than an Olympic swimmer. There are multiple factors that determine the location and strength of rips, such as bathymetry, wave height and direction, tide, and beach shape. Rip currents may either be transient or persistent in space and time. Rips that are frequently found at the same location are usually indicative of a fairly stable bathymetric feature such as a sand bar or reef, or a hard structure such as rocky outcrop, jetty or pier. These bathymetric features results in variations in wave breaking and setup leading to channelized rip current flow. Transient or flash rips are independent of bathymetry and may move up or down the beach, and may appear or disappear. Transient rips are best understood with respect to vorticity due to short-crested wave breaking and the subsequent eddy coalescence [66, 67].

The Southeast Coastal Ocean Observing Regional Association (SECOORA) in partnership with the National Oceanic and Atmospheric Administration (NOAA) maintains a network of coastal web cameras for different applications such as montoring wave runup, human use of natural resources, and spotting rip currents [35]. These data are supporting the validation of a rip current forecast model to alert people to potential hazards [18]. The most commonly used method to visualize rip currents from video is time averaging, summarizing a video as a single image [37]. However these time averages when manually assessed can be misinterpreted. Furthermore, they are not readily available nor interpretable by the average beachgoers, and the process of averaging removes available information.

In recent years the coastal engineering community has successfully used deep neural networks to solve many problems. Classification problems such as classifying wave breaking in infrared imagery [68], beach scene and other landscape classification [69], automated plankton image classification [70] and ocean front recognition [71] were formulated as deep learning problems using convolutional neural networks. Furthermore, some regression problems such as optical wave gauging [72], tracking remotely sensed waves [73], typhoon forecasting [74] were also solved using deep neural networks. In addition, generative adversarial networks, a type of deep neural networks, were used to improve the quality of downscaling of ocean remote sensing data [75].

Object detection with deep neural networks is well studied in the computer vision community. However most benchmarks and research focus on detecting physical objects with boundaries between what is and is not an object [76, 77, 78]. Rip currents are ephemeral "objects" which are not observable in every frame, and amorphous without clearly defined boundaries even when observable. It is not clear whether existing methods are applicable. Figure 3.1 provides a set of examples, illustrating the difficulty of the problem. In some of the images rip currents are clearly visible while in some it is difficult for a layperson to recognize the presence of a rip current.

Our work is aimed at introducing this problem to the coastal engineering community, and showing that object detection methods *are* applicable. We gathered training data of rip currents and labelled those with bounding boxes indicating the location of the rip current with a co-author who is a rip current researcher at NOAA. We use Faster R-CNN [79] with a custom temporal aggregation stage that allowed us to achieve detection accuracy that is higher than both humans and other methods of rip current detection previously reported in the literature.

The remainder of the paper is organized as follows. All the related work is summarized in Section 3.2. We discuss how the data was collected in Section 3.3. Our method is discussed in 3.4. Results are discussed in 3.5. Limitations and discussion are

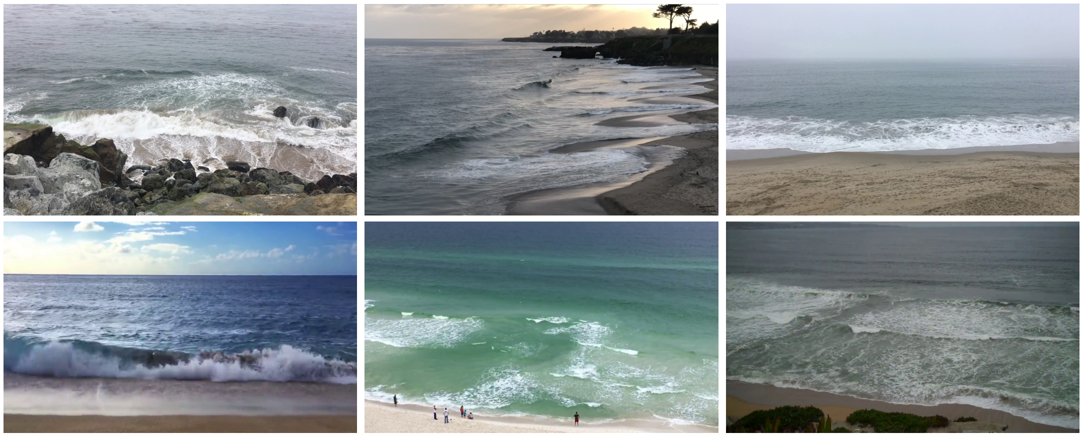


Figure 3.1: **Beach Scenes:** A collection of beach scenes, some of which contain rip currents. Unfortunately these rip current "objects" do not have clear shape, and most people find them hard to identify.

in Section 3.6. In Section 3.7 we conclude our paper. And in Appendix 3.8 we provide the link to the supplementary materials.

## 3.2 Related Work

Rip currents can be observed using both in situ and remote sensing methods. Among the in situ methods, wave sensors, acoustic velocimeters, or current profilers can be deployed at specific locations [25, 26, 27, 80]. Floating drifters with embedded GPS units have also been used to measure currents [33, 61, 34]. These methods are costly, time consuming, require technical expertise and are generally only applicable to highly localized instances in time and space. These limitations severely hinder the applicability of such approaches to both public warnings and model validation. In comparison, remote sensing technologies such as satellite, air-borne, and ground-based imaging as well as radar imaging provide more coverage and less expensive (e.g. web cameras) alternatives [81, 82]. A hybrid approach of adding fluorescein dye into the water and observing its dispersion using aerial video provides dramatic visualization of rip currents [29, 30, 31, 32].

Time averaged images are a routine method for analyzing video in oceanic research, with 10 minutes being a common integration period [36, 37, 38, 83]. This method is popular because averages often make identification of rip channels easier for the human eye. While these images are usually intended for human interpretation, Maryan et. al. apply shallow machine learning to recognize rip channels in time averaged images [3]. Nelko also used time averaged images and noted that prediction schemes developed at one beach location may not be directly applicable to another without some modifications [41]. Haller introduced wave averaging to enhance the detection of rip currents in microwave data; an approach which could also be applied to imagery data [81].

Dense optical flow [43, 44] has been used to detect rip currents in video [84]. This method is attractive since optical flow fields can be directly compared against ground truth flow fields obtained from in situ measurements [45]. Unfortunately these methods are sensitive to camera perturbation, and have difficulty in areas lacking textural information.

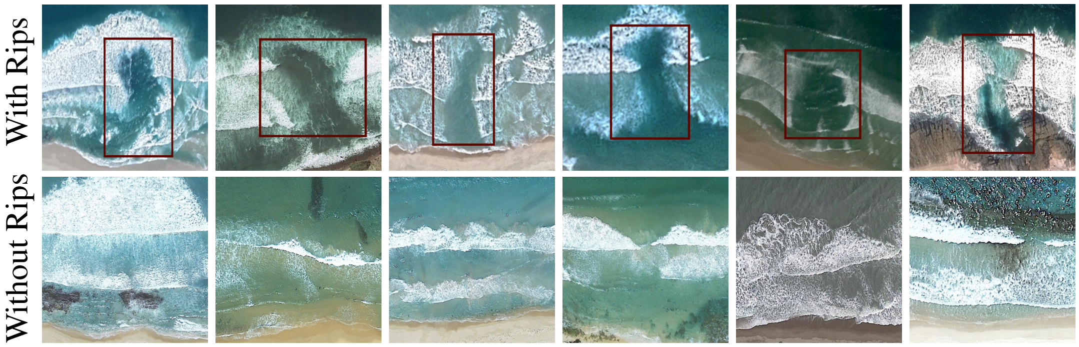


Figure 3.2: **Training Data:** Examples drawn from the 2440 images we collected and labeled to build a training data set. Ground truth bounding boxes are shown in red.

Certain kinds of rip currents are characterized by visible sediment plumes. These can be segmented based on changes in coloration. For example, Liu et al., use thresholding in HSV color space to detect rip currents [80]. Unfortunately, not all rip currents contain sediment plumes, and thus this method is not applicable to our data sets.

Object detection in images is well studied in the computer vision literature [85, 86, 87, 88, 89]. These methods have been extended to detect objects in videos [90, 91, 92]. However, with the exception of Maryan et al. [3] which performs detection based on time averaged images, and Liu et al. [80] which performs detection based on color segmentation, they have not been applied to crisp images or videos for the purpose of rip current detection.

## 3.3 Data sets

### 3.3.1 Training Data

Since rip currents are a new problem domain for computer vision, we did not find any existing public databases of rip current images. Therefore, we assembled a training data set of rip current images and non-rip current images from scratch. Our primary source for the database was Google Earth$^{TM}$, which allowed us to extract high-resolution aerial images of rip currents and non-rip currents. In total, the database contains 1740 images of rip currents and 700 images of similar beach scenes without rip currents. The images range in size from 1086 × 916 pixels to 234 × 234 pixels. We annotated ground truth in the rip current images with axis-aligned bounding boxes: where the $x$ axis and $y$ axis of the bounding box is aligned with the $x$ axis and $y$ axis of the image. Some examples of the training data set are shown in Figure 3.2. Note that this data set contains unambiguous easy examples. We used this training data for training models described in Section 3.4.

### 3.3.2 Test Data

We also collected a data set of 23 video clips consisting of 18042 frames in total. Of those, there are a total of 9053 frames with and 8989 frames without rip currents. Image size varies from 1280 × 720 pixels to 1080 × 920 pixels. We used the bounding boxes labelled on each image as the ground truth. Figure 3.1 contains both positive and negative example frames, as well as a few frames from the training set that might be mistaken as containing a rip current. Note that this set contains more difficult cases.

The frames of this video data set were used for testing. Note that the static images in the training set were taken from high elevation while the videos used to test the model were taken from a lower perspective. Even so, the trained model performed well on the test frames from the video collection.

## 3.4 Method

### 3.4.1 Deep Learning

Deep learning methods have recently been used in many computer vision tasks. These methods have taken over many traditional shallow machine learning methods such as support vector machines, random forests, etc. [93, 94, 95, 96, 97, 98]. However, deep learning models require careful tuning of hyperparameters, which determine the deep learning model architecture. Multiple sensitivity analysis experiments are often needed to optimize hyperparameters for a particular model and task. Also, deep learning models are costly to train, usually taking days and many dedicated computation resources such as GPUs (Graphis Processing Units) or TPUs (Tensor Processing Units).

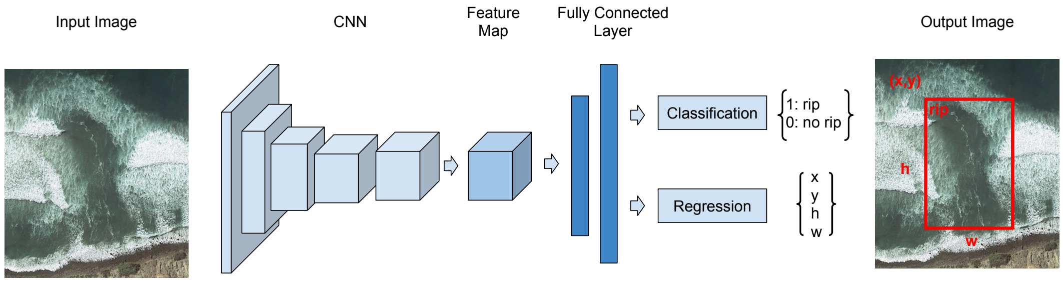


Figure 3.3: **Architecture of Faster R-CNN:** Classification and Regression branch. CNN represents the convolutional neural network.

Unlike deep learning methods, traditional machine learning methods require a fair amount of human feature engineering by a domain expert, such as the features created by Maryan et al. [3] for the rip current detection task. Features engineered by humans may not be the most optimal. In deep learning, the algorithm learns the most optimal features through gradient decent that are necessary for the given task.

Even with a high computation cost of training, deep learning models learn a large number of parameters compared to traditional machine learning models, resulting in a higher accuracy in complex vision problems. Deep learning methods such as region based convolutional neural networks have out-performed traditional machine learning methods in many vision tasks, such as object detection [99, 3, 100]

### 3.4.2 Static Image Detector: Faster R-CNN

Region-based convolutional neural networks have achieved great success in object detection problems. These object detection models usually consist of separate classification and localization networks with a shared feature extraction network. In the computer vision community, Faster R-CNN is generally considered as the most accurate object detector [101]. Many critical applications such as fire detectors [102] and pedestrian detectors [103], successfully use Faster R-CNN. Also, many medical applications such as detecting cervical spinal cord injury and disc degeneration detection [104], breast cancer detection [105], malaria cell detection [106] use Faster R-CNN as their underlying object detector. Therefore, we choose Faster R-CNN as our single image detector.

The first is the deep convolutional neural network that proposes regions. The second is the Fast R-CNN detector [107]. Faster R-CNN follows the traditional object detection pipeline. It first generates region proposals, and then categorizes each proposal as either rip current or background. Secondly, the classified bounding boxes are further refined. Essentially, the model learns a mapping from the generated regions to the actual ground truth with a regression network. The model then uses this mapping "function" during testing to refine the generated region. These refined bounding boxes can be anywhere in a frame as features are translation invariant [79]. If there is more than one bounding box detected in a frame, we only keep track of the largest one and ignore any additional boxes.

As shown in Figure 3.3 the convolutional neural network (CNN) consists of five hierarchical blocks. Each block consists of convolutional layers followed by a max pooling layer. The convolutional layers generate features by applying filters to the input. After which the output is fed into a max pooling layer where the feature map is down-sampled by only keeping the maximum values within a region. These two steps are repeated in the five blocks of the CNN, resulting in the final feature map.

Multiple regions from the input image are then projected onto the feature map. These regions are generated as a rectangle with predefined proportions and positions. These regions have a predefined area of $64^2$ pixels, $128^2$ pixels and $256^2$ pixels. Each region is generated as an rectangle with three different aspect ratios for length and width: $1:1$ , $1:2$ , $2:1$, resulting in nine different regions. These regions are then positioned on a regular grid by generating the nine regions centering round each grid point. The regions that fall beyond the boundary of the image are ignored. The associated features for these regions are then fed into the fully connected layers where the decisions are made. The classification branch predicts if the region is a rip current or not. If the classification branch classifies the region as a rip current then the regression branch further refines the position and the size of the detected rip current.

We trained the Faster R-CNN model with the training data discussed in section 3.3.1. Before training, each image was augmented by rotating $90^\circ$ degrees clockwise and counter clockwise, producing a training data set three times the size of the original training data set. All the training data was resized to $300 \times 300$ pixels before training to save computation time. We used bi-cubic interpolation to resize the images.

### 3.4.3 Frame Aggregation

Static object detection models only consider the information in the frame currently being processed. However, rip currents are natural ocean phenomena, with shape and texture change depending on many external factors such as weather, wind speed, wave field characteristics, water flow speed, floating debris, and dirt sediments. The exact boundaries of a rip current are not well defined. This is different than objects with well-defined edges such as pedestrians or vehicles. Applying detection algorithms to objects with amorphous boundaries such as rip currents produces bounding boxes with variable sizes and locations in adjacent video frames. In Figure 3.4 we illustrate this variability by drawing all correctly detected bounding boxes from one video sequence onto a single frame.

This variability affects overall accuracy, and would not instill confidence in the results if these bounding boxes were presented to a user as a video overlay. Thus we investigate temporal smoothing and aggregation to improve the results.

We find the overlapping regions of the detected bounding boxes by using an accumulation buffer with the same size as the input frame and initialized as a zero matrix. We consider a temporal window of $N$ frames to build the accumulation buffer. In the first $N-1$ frames, the accumulation buffer is incremented by 1 for each region within a detection bounding box. Starting with frame $N$, the area covered by a detection bounding box is incremented by 1 and capped at a maximum of $N$. Regions not covered by a

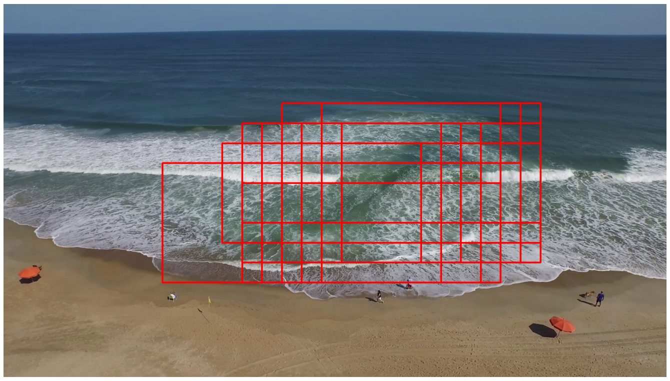


Figure 3.4: **Bouding boxes:** The bounding boxes marking the boundaries of detected rip currents from individual frames of a video segment are superimposed onto the most recent frame. Note how the bounding boxes from individual frames may move and/or change shape

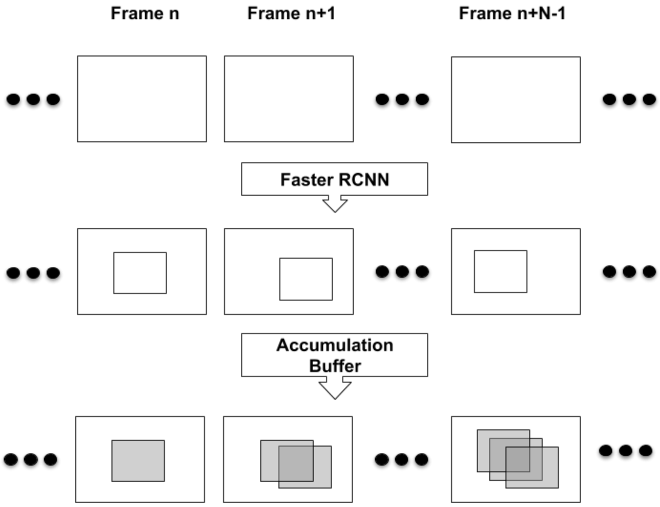


Figure 3.5: **Frame aggregation** for a window of length $N$. First row shows the input frame sequence. Second row shows the detections from Faster R-CNN. Third row shows the accumulation buffer.

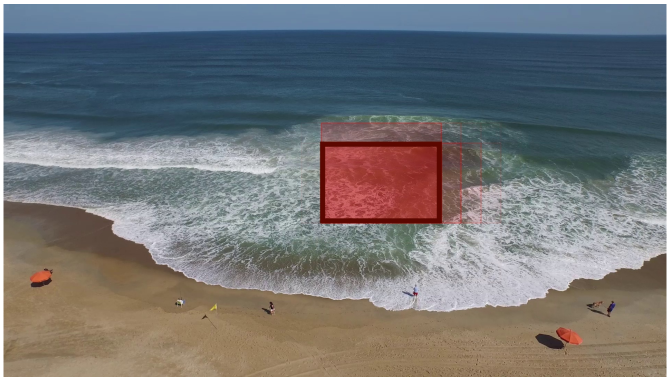


Figure 3.6: **Accumulation Buffer:** A visualization of the resulting accumulation buffer values. Regions with a higher value are shown in more opaque red. The resulting bounding box around thresholded values is shown in solid dark red. The full video can be seen in the supplementary material in appendix 3.8

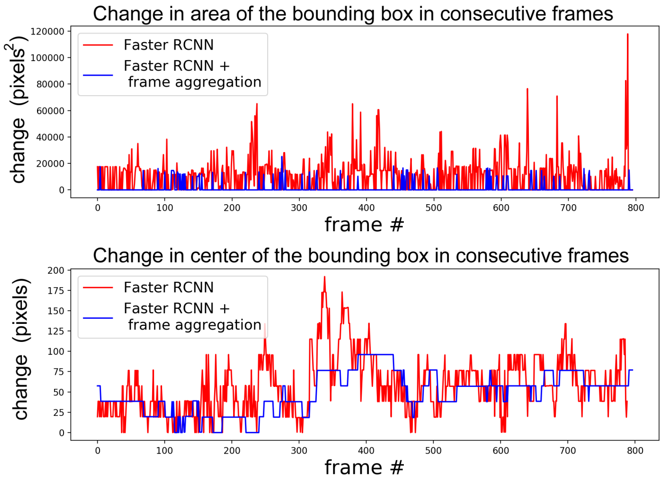


Figure 3.7: **Frame aggregation stabilizes the video:** Plots showing the differences in area and center of bounding boxes in consecutive frames. Without frame aggregation (in red) there are much higher differences in bounding box sizes and positions than there is after frame aggregation (in blue).

detection bounding box are decremented by 1, but retains a minimum of 0 in the accumulation buffer. In effect, the accumulation buffer keeps track of the bounding boxes using a sliding window of $N$ frames. The process of building the accumulation buffer is illustrated in Figure 3.5, where areas with higher values are displayed as darker regions in the accumulation buffer. For purposes of identifying a single bounding box over the collection of bounding boxes across $N$ frames, we consider only the regions of the accumulation buffer where the value is at least $T$, and draw the tightest possible axis-aligned bounding box around this region (see Figure 3.6). This is the aggregated detection. In our implementation we use N=60 and T=30.

Before frame aggregation, large variations in bounding box size occur in almost all consecutive frames, shown in Figure 3.7 top. After frame aggregation, the average size change is much smaller, with most frames having zero change in size from the prior frame. The variation in position of the bounding boxes is similarly reduced by frame aggregation, as seen in Figure 3.7 bottom. This improved temporal coherence provides a smoother and more consistent portrayal of the rip current location when shown as an overlay on the video.

When analyzing video clips, our method analyzes individual frames and places a bounding box around detected rip currents. Since analysis is on a per frame basis, the bounding boxes may move from frame to frame. In addition, it's possible that a rip current may not be detected in a particular frame. However, with our proposed frame

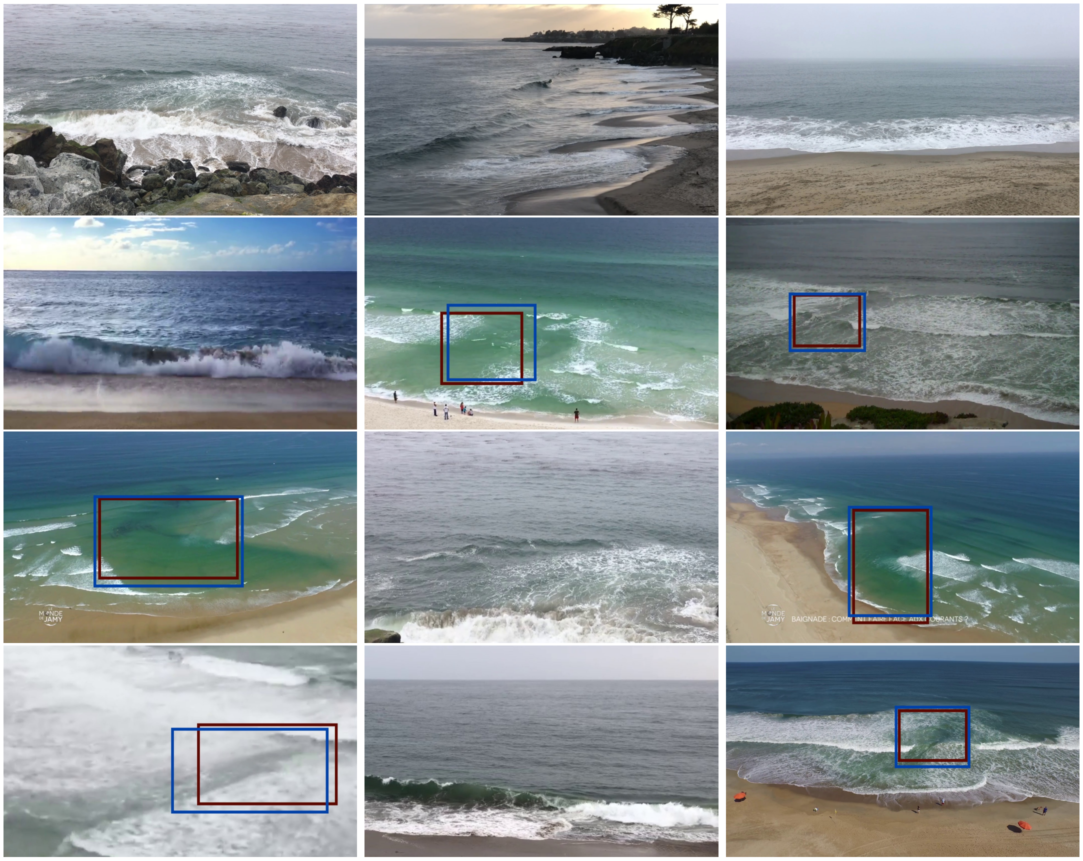


Figure 3.8: **Detections:** Rip current detections on some of the frames from the test data set. Red bounding boxes show the correctly detected rip currents. Blue bounding boxes show the ground truth. Frames without bounding boxes do not contain any rip currents.

aggregation strategy, instances when a few frames where the rip is not detected (false negative), or when there's a sudden change in the location or size of the bounding box (false positive) are both handled gracefully.

| . | Human | Philip [84] | Maryan [3] | Maryan | F-RCNN | F-RCNN+FA |
|---|---|---|---|---|---|---|
| | | | | [modified] | [ours] | [ours] |
| *rip*_01.*mp*4 | 0.976 | 0.895 | 0.358 | 0.460 | 0.966 | 1.000 |
| *rip*_02.*mp*4 | 0.700 | 0.098 | 0.698 | 0.135 | 0.776 | 0.860 |
| *rip*_03.*mp*4 | 0.231 | 0.347 | 0.194 | 0.540 | 0.831 | 0.950 |
| *rip*_04.*mp*4 | 0.757 | 0.800 | 0.487 | 0.780 | 0.939 | 0.970 |
| *rip*_05.*mp*4 | 0.883 | 0.280 | 0.736 | 0.450 | 0.834 | 0.957 |
| *rip*_06.*mp*4 | 0.881 | 0.000 | 0.167 | 0.470 | 0.753 | 0.890 |
| *rip*_08.*mp*4 | 0.492 | 0.063 | 0.328 | 0.730 | 0.860 | 0.850 |
| *rip*_11.*mp*4 | 0.824 | 0.000 | 0.563 | 0.940 | 0.930 | 0.951 |
| *rip*_12.*mp*4 | 1.000 | 0.000 | 0.734 | 1.000 | 1.000 | 1.000 |
| *rip*_15.*mp*4 | 0.967 | 0.137 | 0.315 | 0.390 | 0.760 | 0.870 |
| *rip*_16.*mp*4 | 0.614 | 0.073 | 0.468 | 0.640 | 0.820 | 0.920 |
| *rip*_17.*mp*4 | 1.000 | 0.321 | 0.064 | 0.750 | 0.980 | 1.000 |
| *rip*_18.*mp*4 | 0.563 | 0.218 | 0.250 | 0.240 | 0.790 | 0.890 |
| *rip*_21.*mp*4 | 0.901 | 0.543 | 0.486 | 0.180 | 0.940 | 1.000 |
| *rip*_22.*mp*4 | 0.583 | 0.000 | 0.479 | 0.395 | 0.880 | 0.974 |
| *no*_*rip*_01.*mp*4 | 0.986 | 0.169 | 0.000 | 0.972 | 0.813 | 1.000 |
| *no*_*rip*_02.*mp*4 | 1.000 | 0.789 | 0.000 | 0.985 | 0.807 | 1.000 |
| *no*_*rip*_03.*mp*4 | 0.919 | 0.000 | 0.000 | 0.981 | 0.984 | 1.000 |
| *no*_*rip*_04.*mp*4 | 0.952 | 0.000 | 0.000 | 0.974 | 0.835 | 1.000 |
| *no*_*rip*_05.*mp*4 | 0.903 | 0.000 | 0.000 | 0.986 | 0.833 | 1.000 |
| *no*_*rip*_06.*mp*4 | 1.000 | 0.246 | 0.000 | 0.986 | 0.875 | 1.000 |
| *no*_*rip*_07.*mp*4 | 0.983 | 0.525 | 0.000 | 0.982 | 0.875 | 1.000 |
| *no*_*rip*_11.*mp*4 | 0.988 | 0.198 | 0.000 | 0.964 | 0.924 | 1.000 |
| *average accuracy* | 0.760 | 0.307 | 0.210 | 0.729 | 0.884 | **0.984** |

Table 3.1: **Accuracy for each video in the test set.** Column 3: Maryan[3] is trained on their training data. Column 4: Maryan [modified] is trained on our training data. The F-RCNN method has higher overall accuracy than humans or any of the prior methods tested. Frame aggregation does contribute to improvement in accuracy.

## 3.5 Results

We compared the accuracy of our method with human observers as well as two prior methods. We also compared our own method with and without temporal aggregation.

**Comparison metric.** All methods were tested using the video data set. Frames were labeled as correctly classified if the detected bounding boxes have an Intersection over Union (IoU) [108] score versus ground truth above 0.3. IoU is calculated as $area\_of\_intersection/area\_of\_union$ of the ground truth and the detected bounding boxes. Accuracy of the video was computed as $correct\_labels/total\_frames$, and Table 3.1 provides the results for all methods. Since rip currents do not have a well-defined boundary the scale of ground truth bounding boxes has some uncertainty. Therefore, the predicted bounding box does not need to closely match with the ground truth bounding box for a detection to be marked as correct. Because of this we choose a lower IoU threshold than detecting objects with well defined boundaries such as cars or tanks [79]. We visually verified that even with a lower IoU threshold the detected bounding box shows the location of the rip current. Also, we used this IoU threshold across all methods in Table 3.1.

**Humans.** One of the primary reasons for an automated method of rip current detection is that most people are not good at identifying rip currents [23]. To measure human accuracy of identifying rip currents, annotators were asked to draw bounding boxes around places they believe to have rip currents. We sampled every tenth frame from our video test set and randomized presentation order across all positive and negative examples. Human annotators were not carefully trained, instead they were provided three positive and three negative examples, roughly the amount of information which might fit on a sign at the beach. Annotators were acquired using Mechanical Turk, an online market where jobs are posted for workers [109, 110], with basic screening for reliable workers, and paid $0.10 per image.

Although human performance was relatively poor with only 76% of frames labeled correctly, it was higher than we expected based on past studies [23]. We hypothesize that the sample images showing both locations of rip currents and absence of rip currents are more effective than warning signs alone.

**Time averaged images.** Maryan et al. [3] perform detection on time averaged images using a boosted cascade of simple Haar like features [89]. We used Maryan's time averaged data for training since our training data set consists of only static frames. Testing was performed by first computing time averaged images on each video in our test data set. This method did not perform well. In order to determine if the cause was the images available for training or the model itself, we repeated the experiment with new data. We replaced the relatively small number of low resolution time-averaged images from [3] with the static images from our training data set. Testing this time was against single frames in our test data . This modification is called Maryan[modified] in Table 3.1. When using our training data the model accuracy improved considerably, leading us to conclude that appropriate training data is critical to good results. Furthermore, it suggests further investigation on whether using crisp images, rather than time averaged images, to train a model might produce more accurate results.

In our test images/videos the beach is always located at the bottom half of the image/video. However, the training data used by [3] was cropped from images where the beach is located in the top half of the image. Therefore, we were concerned that the difference of orientation between the training data and the test data contributed to the low accuracy of the model. However, when we retrained the model with vertically flipped training data we did not see any significant difference in accuracy on our test images/videos. We hypothesize that the reason for this is that the cropped training images contain insignificant amount of beach pixels.

**Optical flow.** Philip et al. [84] compute optical flow on video sequences and make the simplifying assumption that rip currents can be identified by regions with the second most predominant flow direction, after that of the primary incoming wave direction, and that they flow in a single seaward direction. This results in regions of actual rip currents, but also picks up swash regions where water is washed up the beach and back out to sea with the passing of each wave. This method was introduced with the primary intention of providing visualizations to users, rather than automated detection. To allow comparison, we modify the method to return a bounding box around the largest detected region, ignoring smaller regions which are less likely to be correct. This method performed poorly on our test data. We noticed that in videos where there is not enough textural information on the rip current, the optical flow field generated was weak, leading to either missed detections or detection in other regions of the video with stronger texture.

**Frame aggregation.** We implemented frame aggregation as a post process to Faster R-CNN initially to temporally stabilize detections, driven by a need for user interpretable visualization of rip current location.

In order to understand whether temporal smoothing also increased accuracy we analyzed our implementation both with and without frame aggregation. We found that temporal aggregation leads to higher accuracy than using Faster R-CNN alone. Example detection results are shown in Figure 3.8. Numerical comparison of humans, prior methods, and our model are provided in Table 3.1. Faster R-CNN with frame aggregation had the highest accuracy in nearly all cases, and the highest overall (last column of Table 3.1). For visual comparisons we have added all the results in the supplementary materials at appendix 3.8.

## 3.6 Discussion and Future Work

As with all machine learning models, our implementation can fail when used with images that do not resemble the training data set. Our data sets included primarily rip currents characterized by a gap in breaking waves, the most common visual indicator

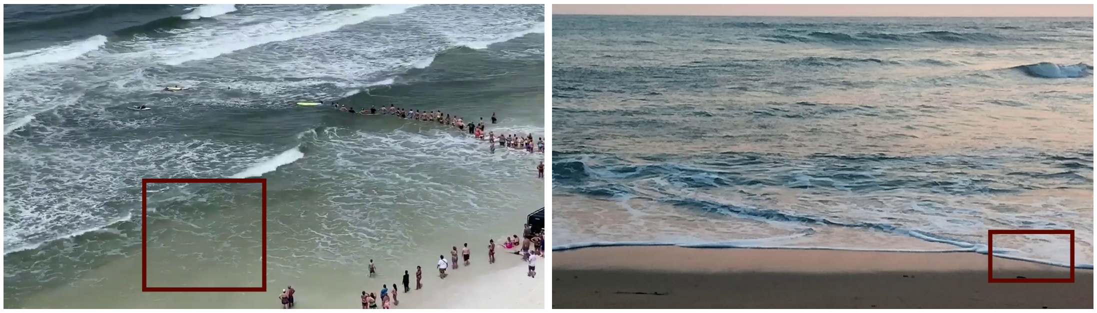


Figure 3.9: **Example failure cases.** The false positives on the beach scene (right) are not easily explainable. The false positive on the left scene happens only on spurious frames, which is then corrected by frame aggregation.

for bathymetry controlled rip currents. Thus we would expect to miss rip currents with other visual indicators like sediment plumes. We also expect to fail when presented with new imagery, and occasionally for no apparent reason at all, as seen in Figure 3.9.

**Sensitivity Analysis.** Sensitivity analysis experiments were conducted to determine the optimal values for parameters $N$ and $T$ used in frame aggregation (Section 3.4.3), training data and number of training iterations. Figure 3.10 shows the average accuracies of rip current detection from our test data using different values of $N$ and $T$. The smallest $N(=60)$ and $T(=30)$ with the highest accuracy was chosen as the optimal $N$ and $T$. By choosing the smallest $N$ and $T$ we are able to save on computation time as well.

Further experiments were conducted to determine the optimal amount of data needed to train the model. The model was trained with 25%, 50% and 100% of the training data. At each stage, the accuracy on the test data was recorded. Training the model

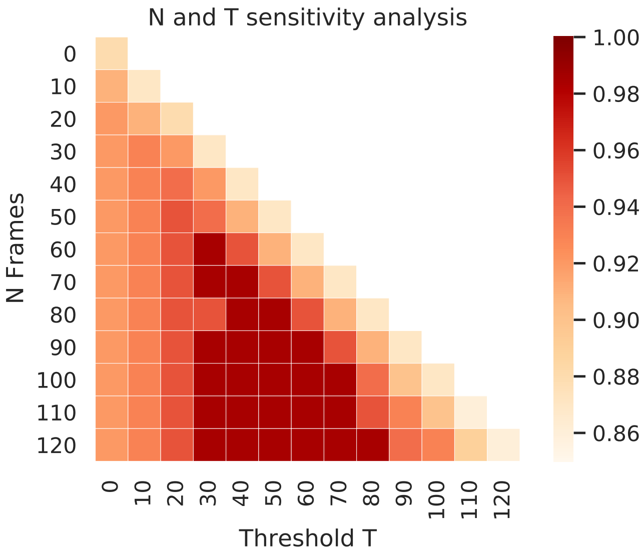


Figure 3.10: **Sensitivity Analysis:** Heatmap of accuracy values for parameters N frames and threshold T.

with the full training data set lead to the highest accuracy as shown in Figure 3.11. Experiments were also conducted on when to stop training our model. The accuracy of the model on the test data was recorded at different training epochs. The training was stopped at epoch 60 when the accuracy plateaued as shown in Figure 3.12.

We did not conduct a sensitivity analysis of the hyperparameters of the network. We relied on the sensitivity analysis conducted by the authors of Faster R-CNN [79] to determine the network's optimal hyperparameters.

With frame aggregation the accuracy for test data without rip currents was high compared to the accuracy for test data with rip currents. Since frame aggregation takes into account the prior *N* detections, we can easily filter out spurious detections, leading

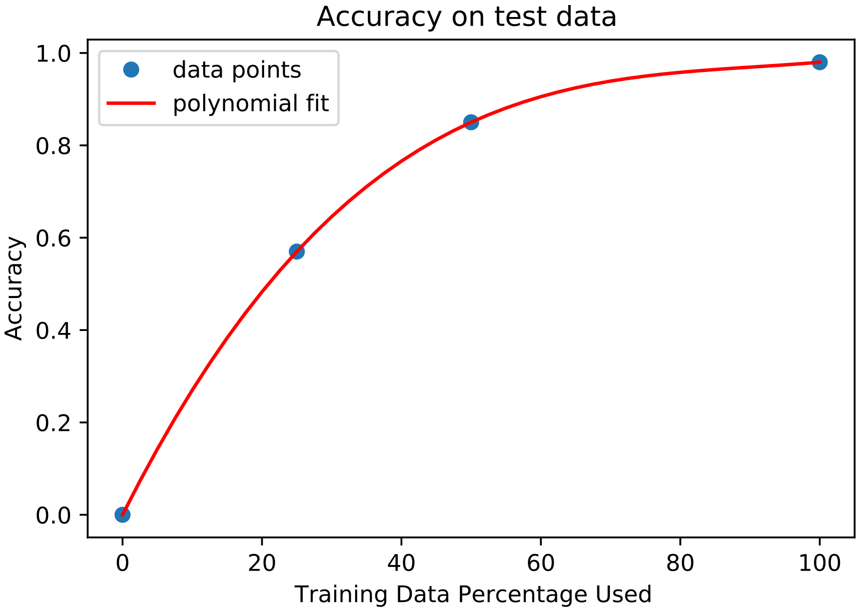


Figure 3.11: **More training data leads to a highly accurate model:** Accuracy for models trained on different percentages of training data.

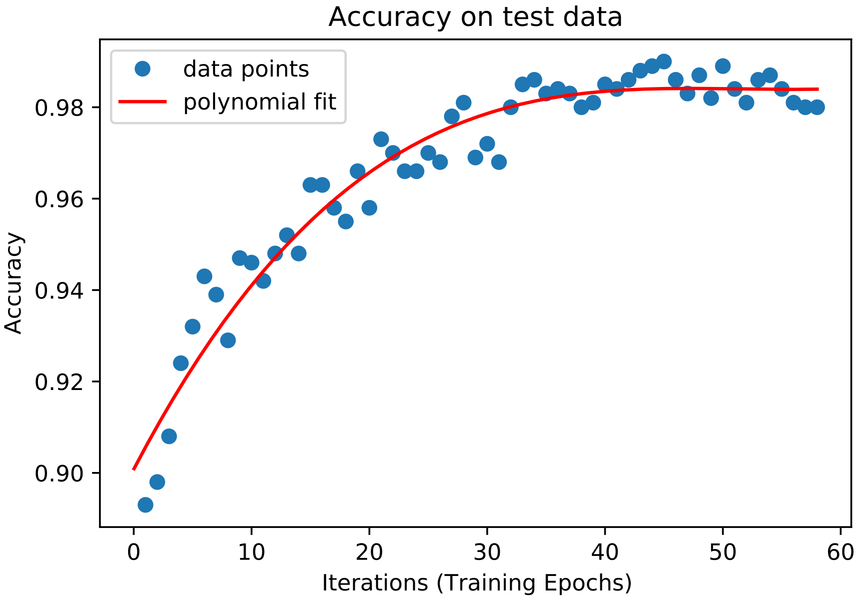


Figure 3.12: **Optimal training epochs:** Accuracy for testing data on each training epoch

to a higher accuracy for test data without rip currents.

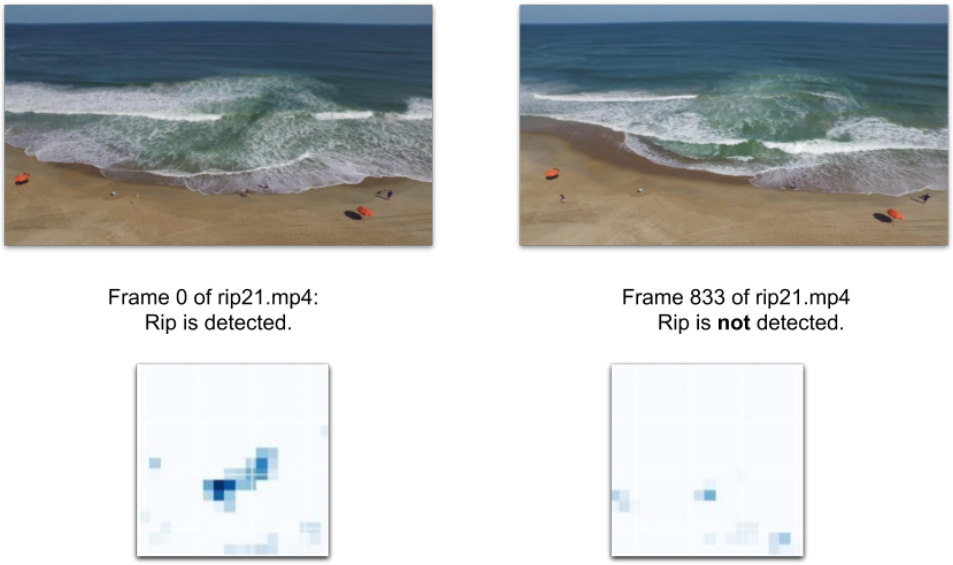
We noticed that for one video, $rip\_08.mp4$, frame aggregation did not improve the accuracy. For $rip\_08.mp4$, the placement of bounding boxes identifying the rip current from individual frames by F-RCNN is more spread out. This leads to smaller over lapping regions of bounding boxes. When used with frame aggregation, the resulting bounding box is smaller than the bounding box without frame aggregation. This produces a more conservative estimate of the size of of the rip but similar "confidence" of its location using the same IOU threshold.

The frame aggregation method discussed in Section 3.4.3 works well when the rip current is fairly stationary relative to the camera. Specifically, with $N = 60$ frames and a frame rate of 30 frames per second, we are assuming that the rip current is in the same location over a 2 second interval of the video. This assumption is generally true for video captured from stationary cameras such as CCTV or web cameras. If the video were captured using mobile devices such as smartphones or drones where the rip current moves a significant amount within the frame, additional information such as the camera's accelerometer and GPS can possibly be incorporated in the processing. This is a subject for future work.

We are aware that there is an array of single image object detectors in computer vision literature that we did not train our model on. Due to the time (usually 3-6 days) and GPU resources required to train the model, it is infeasible to train our model using many different types of object detection approaches. Therefore we relied on prior work on similar critical applications to determine what our single image object detector should be. We have cited those prior applications in section 3.4.2.

It is challenging to explain why a particular deep learning system produces a certain result. A whole academic field has been formed around explainability in deep learning and AI systems. For instance, why the model fails to detect a rip current even though there is a visible rip current to the human observer in the frame?. We attempt to explain the variability in the model's accuracy values by visualizing the input image in the feature space similar to the works of [111] and [112]. As an example, we choose frames 0 and 833 of $rip\_21.mp4$. For both frames, there is a visible rip current observable to the human expert. For frame 0, the model detects a rip current (true positive), but for frame 833, it fails to detect a rip current (false negative). By observing the input images in the feature space as shown in Figure 3.13, we can see that for frame 0, the features resemble a rip current, and for frame 833, the features do not resemble a rip current. We attribute the true positive detection for frame 0 to the strong signal generated by the features. Similarly, we can attribute the false negative of frame 833 to the weak signal generated by its features.

We also noticed that Maryan et al. [3] did not perform well in non-rip current cases as shown in Maryan[3] column in Table 3.1. After further study we realized that the test set they used contained almost entirely of images with rip currents. We think that



Frame 0 of rip21.mp4:
Rip is detected.

Frame 833 of rip21.mp4
Rip is **not** detected.

Normalized Feature map :
channel 453
Features resembling a rip
current can be seen

Normalized Feature map :
channel 453
Features resembling a rip
current is **not** seen.

Figure 3.13: **Explainable rip detector:** Visualization of frame 0 and frame 833 of $rip\_21.mp4$ in feature space

the model proposed by Maryan was not extensively tested on non-rip current images, which accounted for its poor performance for such conditions.

We did not do transfer learning to train our model. Transfer learning is a machine learning technique where the parameters and weights learned in one problem are used as the initial values for training another model on a different problem [113]. This strategy can significantly reduce training time and is useful when the two problems belong to similar domains e.g. detecting cars vs detecting tanks. We were not able to exploit this strategy in training our models as we could not find a deep learning model that was trained on a domain somewhat similar to detecting rip currents. Note that the method proposed by Maryan et al. [3] is not a deep learning method and hence not applicable here.

We found it difficult to compare our method with prior work and verify that our model performs well in all conditions previously researched, due to a lack of public data sets on which to verify our results. In order to ensure that future work has a baseline from which to compare, our data sets with thousands of labeled frames are available as part of the supplementary material. Nevertheless our data sets are still limited. The accuracy numbers presented in this paper are correct on this limited data, but almost certainly overstate probable outcomes in real world deployment. We expect that future work will need to collect more examples including less common rip current visual presentation, a greater variety of scales, and a wider array of beach distractors.

Lastly, our work can benefit from a re-examination of the IoU metric employed by our algorithm. Certainly IoU, true positive rate, mean average precision (mAP), and the like are common in computer vision research, but these are usually used in the context of detection based on appearance rather than on behavior. It would be interesting to study how deep learning methods can be trained to recognize rip current behavior using a metric that incorporates a temporal dimension.

## 3.7 Conclusion

In this chapter we presented a machine learning approach for identifying rip currents automatically. We use Faster R-CNN and a custom temporal aggregation stage to make detections from still images or videos with higher measured accuracy than both humans and other methods of rip current detection previously reported in the literature.

Although, using apperance shows great promise in finding rip currents, some rip currents are not governed by apperance. In such instances our method fails to detect rip currents. Therefore, in Chapter 4, we explore the use of pathlines in an machine learning method to find rip currents.

## 3.8 Appendix

In order to encourage progress in this domain, both training and test data sets will be made available to the public. Supplementary material including the results to this article can be found online at `https://sites.google.com/view/ripcurrentdetection/home`

# Chapter 4

# RipViz: Finding Rip Currents by Learning Pathline Behavior

## 4.1 Introduction

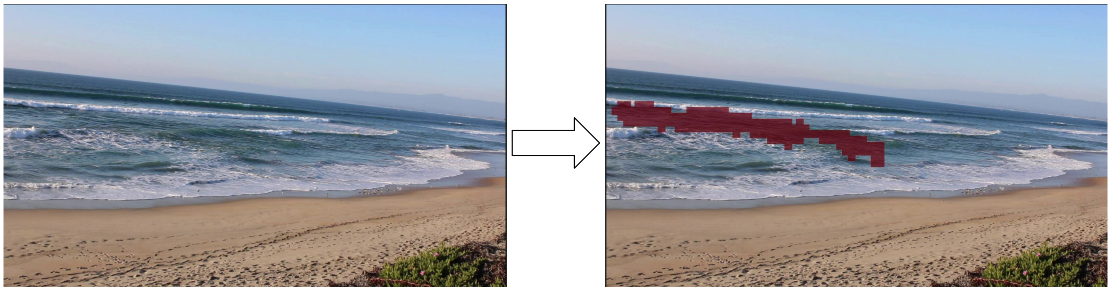


Figure 4.1: **Rip currents are deadly but remain invisible to many:** We propose a feature detection method, RipViz, to make these invisible rip currents visible. RipViz highlights locations of rip currents as a red region. This region is determined by finding seed points that produce pathline sequences deviating from normal ocean flow. Non-experts and experts alike can use RipViz to visualize rip currents.

Rip currents are powerful, narrow channels of fast-moving water flowing towards the sea from the nearshore [15, 61, 114, 115]. The speed of seaward rips can be very strong, reaching two meters per second, faster than an Olympic swimmer. They are a dangerous beach hazard that most people do not recognize. As a result, there are thousands of drownings each year due to rip currents globally [116, 117]. The goal of this work is to improve public safety by helping the beachgoers *see* the rip currents when they are present.

The mechanism for rip currents is an increase in the mean water level, referred to as setup, which occurs when waves break against the shore. This setup can vary along a shoreline depending on the amount of water or height of breaking waves. Rip currents form as water tends to flow from regions of high setup (larger waves) to regions of lower setup (smaller waves), where currents converge to form a seaward flowing rip.

Detecting rip currents with machine learning (ML) is challenging because there are different types of rips, each with a different appearance. The three major factors that lead to different types of rips are the shape of the shoreline, the bathymetry, and hydrodynamic factors (e.g., wave height and direction, tides). The combination of these lead to rip currents with different visual signatures. Rips may also either be transient or persistent in space and time. Detecting rip currents pose unique challenges compared to detecting other objects such as cars, or people, etc. Rips are amorphous without a well defined shape or boundary, and are ephemeral without well defined temporal bounds.

Using an object detector, such as those included in a recent survey [118], would require a substantial training dataset for each type of rip current. To date, there are only two publicly available training datasets for rip detection: individually labeled frames [119] and time averaged images [3], both of the same type of rip current. There are no existing training dataset for other types of rip.

By nature, rips always eventually flow seaward regardless of their visual appearance. Therefore, we propose a novel hybrid approach that incorporates flow analysis with ML for rip detection, rather than relying on image colors. We also propose encoding rips as locations with anomalous flow behavior, transforming the detection task into differentiating normal from anomalous behavior. This greatly simplifies the collection of training data since only a single dataset is needed.

There are two existing approaches to detecting rip currents from stationary videos. The first approach uses the appearance of rip currents to detect them. This includes human experts analyzing time-averaged (Timex) video or running automated object detectors [119, 3, 120, 121] to detect rip currents. The second approach relies on flow analysis, where the flow behavior of ocean waves is analyzed to detect rip currents. Direction-based clustering of flow vectors [122, 123] and timelines [123] placed parallel to the beach are used in this approach. While able to detect weaker rip currents or those where the appearance is not obvious, timelines require user input to specify their initial placement.

In this paper, we introduce RipViz, a fully automated, hybrid of deep learning and flow analysis, feature detection method to find rip currents, as shown in Figure 4.1. We first obtain the time varying flow field from a stationary video using optical flow. We use pathline sequences to capture the flow behavior. One could simply seed pathlines at every point and trace each of them for the duration of the video. However, due to the quasi-periodic nature of wave activity, this simple approach produces too much clutter to be of much use as shown in Figure 4.2. Furthermore, pathlines integrated over the full length of the video accumulate more error, especially in noisy real world datasets. Instead, we generate *sequence of shorter pathlines* for each seed point. By staggering the initiation of pathlines, our expectation is that pathline sequences outside the rip zone will behave differently. For example, pathlines seeded in the surf zone where the waves are breaking will have large variations in their trajectories. Pathlines seeded further out to sea, on the beach, and sky would be fairly still. Pathlines seeded within the rip zone will have less variations in their trajectories yet will be different from the stationary ones.

In RipViz, we frame detecting rip currents as a flow anomaly detection problem. An LSTM autoencoder with a custom weighted loss function is used to learn the spatiotemporal features of pathline sequences for normal ocean flow (i.e. not rip currents). The trained LSTM autoencoder can predict anomalous pathline sequences (i.e. rip currents) during test time. The origination points of anomalous pathlines are identified and

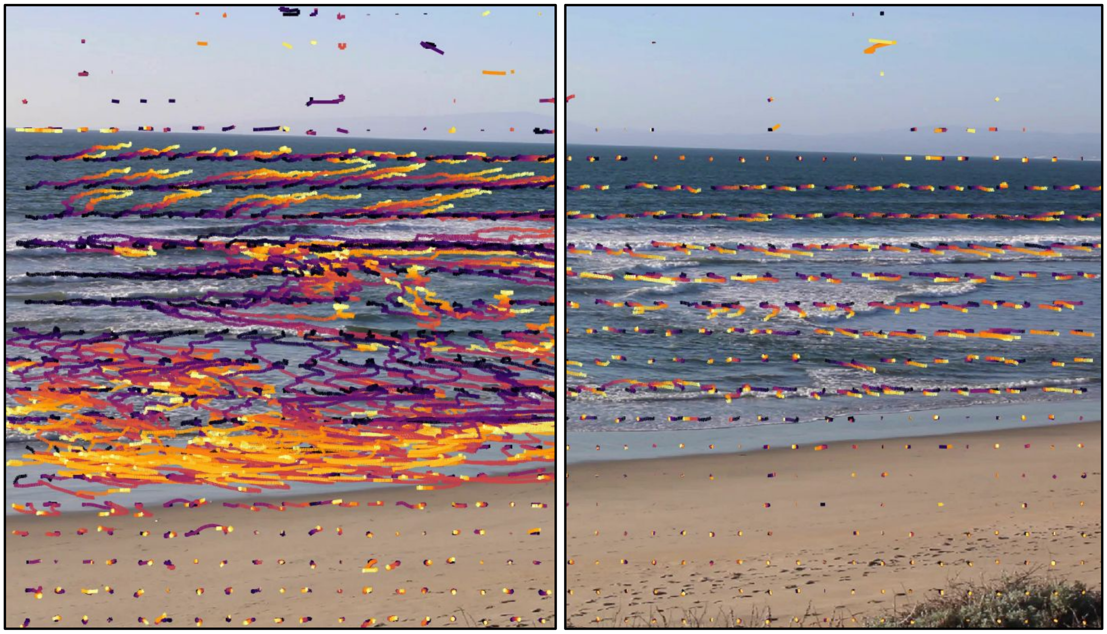


Figure 4.2: **Long pathlines get cluttered making it difficult for ML to learn:** Left pane shows longer pathlines integrated over the entire length of the video. Shorter pathlines, integrated over 900 time steps are shown in the right pane. Longer pathlines are much more cluttered and noisy, making it difficult for ML algorithms to learn its behavior. Notice the relatively large amount of noisy pathlines in the sky and the beach of the left pane. The pathlines are colored by age with yellow representing most recent.

highlighted as a means of visualizing the rip zone. Our target users are general public who are not familiar with rip current dynamics nor visual analytic systems. Hence, our design for the visualization output is to make it as simple and unambiguous as possible. The main contribution of this chapter is:

- A hybrid feature detection method that combines machine learning and flow analysis techniques to automatically find and visualize dangerous rip currents.

In order to realize this, the following innovations are necessary:

- For flow fields with a quasi-periodic behavior, such as ocean flow, working with a *sequence of shorter pathlines* is better than a single long pathline.

- A weighted binary cross entropy, unlike the unweighted binary cross entropy used in previous works, is more effective when learning from sparsely distributed pathlines generated from ocean flow.

## 4.2 Related Work

**Streamline Selection:** Streamline selection and seed placement are well studied in flow visualization. Sane et al. [124] provide a survey of the body of work over the last two decades. These works share overlapping goals with research on streamline clustering [125], and flow simplification [126]. Each aims to produce an uncluttered presentation of the flow field while still capturing the essential flow features and behaviors. For example, Marchesin et al. [127] proposed dynamically selecting a set of streamlines that leads to intelligible and uncluttered streamline selection. Ma et al. [128] used an importance-driven approach to view-dependent streamline selection that guarantees coherent streamline update when the view changes gradually. Yu et al. [129] proposed hierarchical streamline bundles by producing streamlines near critical points without enforcing dense seeding throughout the volume. They grouped the streamlines to form a hierarchy from which they extracted streamline bundles at different levels of detail. Tao et al.[130] proposed two interrelated channels between candidate streamlines and sample viewpoints. They selected streamlines by taking into account their contribution to all sample viewpoints. However, the methods mentioned above use handcrafted features to define the feature representation of streamlines. Handcrafting features to represent pathlines in complex unsteady flow fields, such as ocean flow fields, is a challenging task. In contrast, the method presented in this paper learns complex feature representations without the need for handcrafted features.

**Deep Learning for Flow Visualization:** In recent years, the visualization community has worked with ML in two ways: visualization to understand the ML model, and use of ML in visualization tasks. On the latter, particularly for flow visualization tasks, Berenjkoub et al.[131] used U-net, a deep learning neural network, to identify vortex boundaries. Kim and Günther [132] used a neural network to extract a steady reference frame from an unsteady vector field. Han et al. [133] used an autoencoder-based deep learning model, FlowNet, to learn feature representations of streamlines in 3D steady flow fields which are then used to cluster the streamlines. They used longer streamlines that were integrated through the entire extent of the data. Furthermore, they used one streamline per seed point. The work presented in this paper uses a sequence of pathlines per seed point to learn the flow behavior from noisy unsteady 2D flow fields. This work uses an LSTM autoencoder to detect anomalous pathline sequences from noisy unsteady 2D flow fields.

**Rip Current Detection:** Traditional rip current detection generally involves in-situ instrumentation such as GPS-equipped drifters and current meters [134, 115]. Remote sensing of rip currents is also possible with aerial imaging of the spread of fluorescein dye in the rip zone captured by drones, and the use of marine radar [135, 134]. These require expensive equipment or a team of observers which makes them impractical for rip current monitoring or detection purposes. However, with simple optical video capture such as from surf webcams, it is possible to detect certain rip currents using a time-space (timestack) display or a simulated long time exposure images (timex) display [136]. Timex images reveal rip locations as darker regions in the image that correspond to deeper channels in the bathymetry where water may flow seaward. In contrast, incoming water associated with the breaking waves in the surf zone appear much brighter in timex images. This type of rip is referred to as bathymetry controlled rips and are characterized by a quiet region in the rip channel that is flanked by breaking waves on either side. Interpreting timex images require some domain expertise. Maryan et al. [3] used a Viola-Jones framework to train their model on timex images, and indicate the location of the rips via bounding boxes. Likewise, Rashid et al. [120, 121] also used timex images but utilized a modified version of the Tiny-Yolo V3 architecture. Similarly, Ellis and McGill [137] used timex images in conjunction with environmental information such as tides, wave height, and period to cluster offshore movements to rip currents. However, their method produced false positives in situations where non-water objects such as surfers, paddle boarders, etc., are also moving offshore. Rather than

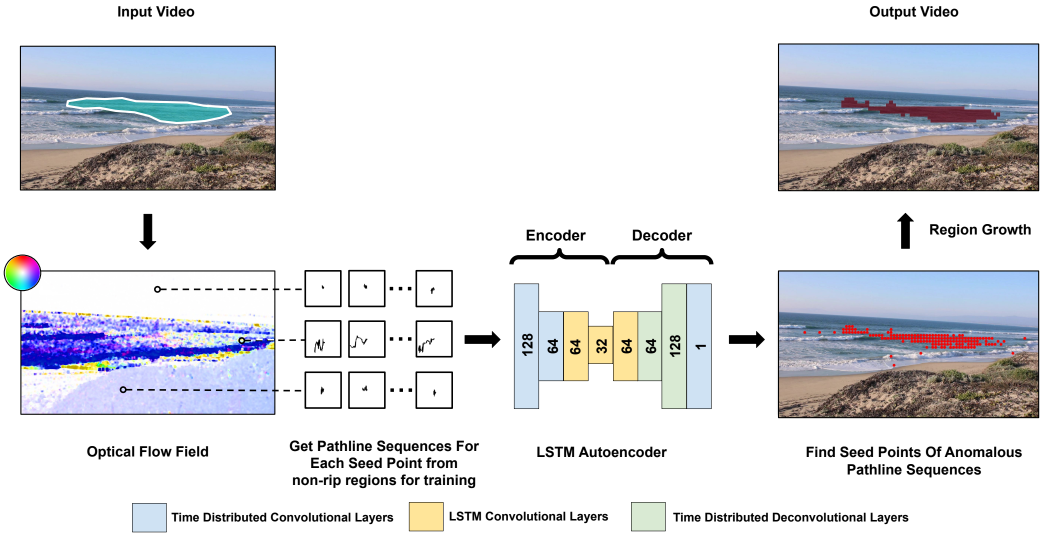

Input Video
Output Video
Region Growth
Encoder
Decoder
128
64
64
32
64
64
128
1
Optical Flow Field
Get Pathline Sequences For Each Seed Point from non-rip regions for training
LSTM Autoencoder
Find Seed Points Of Anomalous Pathline Sequences
Time Distributed Convolutional Layers
LSTM Convolutional Layers
Time Distributed Deconvolutional Layers

Figure 4.3: **RipViz pipeline:** First, an optical flow field was generated from the input video (magnitude and direction of velocities are mapped to value and hue, respectively.) Then pathline sequences were generated for each seed point in non-rip regions (only three seed points are shown) where a new pathline is seeded at every frame. Finally, an LSTM autoencoder was trained with these pathline sequences generated from non-rip regions. For rip detection, the trained LSTM autoencoder is used to detect anomalous pathline sequences by regularly seeding the video frame. We then applied a region growing algorithm to the anomalous points to find the anomalous region, while filtering out singleton seeds.

working with timex images, de Silva et al. [119] trained a Faster R-CNN model with a accumulation buffer to detect bathymetry rips using individual frames of the video. All of these methods detect bathymetry rips based on the appearance of the sea state. However, in instances where appearance is different or weak, these methods fail to detect rip currents.

An alternative approach is to detect rips based on the observed behavior. Philip and Pang [122] obtained an unsteady flow field from the video and hypothesized that the rip current is directly opposite the dominant flow in the vector field due to the incoming wave motion. They grouped vectors based on the direction and magnitude to visualize the rip current. Mori et al.[123] used a similar approach to group vectors and improved the visualization of the rip current by mapping direction to color and magnitude to hue. In the same work, Mori et al.[123] also used timelines to visualize rip currents. They placed timelines parallel to the beach and observed its shape as it gets dragged by the rip current. The work presented in this paper also uses optical video of the sea state to detect rip currents. The underlying approach is also based on the flow behavior, and hence not constrained to detecting bathymetry rips. However, the presented methodology in this paper is the first to propose combining ML and flow analysis to detect different types of rip currents. Because detection is based on treating flow behavior in rip currents as anomalous, the task of collecting and labeling training data for an ML model is unified and simplified. In contrast, a standard ML rip detection model would require a training data set for each type of rip current – a costly and time consuming process.

**Autoencoders:** Autoencoders are a type of neural network that can learn object representations without any supervision [138, 139]. Autoencoders are trained with a loss function that compares the input and reconstructed output by using a reconstruction error. Long short term memory (LSTM) autoencoders are a specialized type of autoencoder that can learn from sequential data. These types of autoencoders are equipped with LSTM layers that can learn how data is temporally related. Furthermore, autoencoders are also used as anomaly detectors[140]. For anomaly detection, autoencoders are first trained with *normal* data. When presented with *anomalous* data, the trained autoencoder will produce a high reconstruction error. We exploit this property to detect anomalous flow behavior.

The flow visualization community has used autoencoders to find feature representations of objects that then can be used for clustering. In their work, Han et al. [133] proposed to learn feature representation of streamlines and stream surfaces of 3D steady flow fields by using an autoencoder with a binary cross-entropy function. They then further reduced the dimensionality of the features and used those to generate clusters. However, their method does not learn how pathlines are temporally related. Additionally, their unweighted binary cross entropy loss function does not account for sparsely distributed pathlines. In our work, we use an LSTM autoencoder with a custom weighted binary cross-entropy loss function, to learn spatiotemporal behavior of sparsely distributed pathline sequences from ocean scenes.

## 4.3 RipViz

Identifying rip currents in complex and chaotic ocean flow is challenging. We first reconstructed a 2D unsteady flow field using optical flow from ocean videos. Our method captured the ocean flow characteristics by using short pathline sequences which are then represented as stacks of binary images. We used an LSTM autoencoder to learn spatiotemporal features of these pathline sequences. We trained our model with pathline sequences of normal ocean flow using a custom weighted binary cross-entropy function, suited for learning pathline sequences of ocean flow. We then used this trained LSTM autoencoder to identify pathline sequences of abnormal ocean flow or rip currents. We visualized the rip currents by projecting the seed points of these abnormal pathline sequences back onto the video frame and growing a transparent red region around these points.

### 4.3.1 Flow Field Reconstruction

An unsteady 2D flow field is obtained from the stationary video using optical flow. Many optical flow algorithms use the relative motion of neighboring pixels between consecutive frames in the video to calculate the local flow. We use Lucas-Kanade [52] sparse optical flow function in the OpenCV library [55] to trace pathlines. We verified each integration step by comparing the forward and backward integration of the flow field.

### 4.3.2 Sequence of Pathlines

Each pathline is represented by a 1D vector, $\mathbf{p} = \{x_1, y_1, \cdots, x_n, y_n\}$, where $(x_i, y_i)$ is a point in the coordinate system of the video frame and $n$ is the length of the pathline. For the LSTM autoencoder to learn about the pathlines, we transformed these pathlines into their own 2D pathline-centric coordinate system. To do this, each pathline is represented by an $L \times L$ binary image $\mathbf{I}$, where each point in $\mathbf{p}$ is translated to center the seed point in the binary image. For longer pathlines that extend beyond $L \times L$, we increased the size of the binary image to accommodate these longer pathlines. We then resized all of these larger binary images down to $L \times L$. The reason why all pathlines were not simply centered then resized to $L \times L$ is that we need to differentiate pathlines that barely moved from their initial seed position versus those that actually travelled beyond $L \times L$. Also note that by centering, the 2D representation of the pathlines are now location agnostic. This combination allows us to compare pathlines from different parts of the video to identify those with similar behavior. As noted earlier, tracing pathlines over the entire length of a video of quasi-periodic motion derived from noisy optical flow calculations result in unusable cluttered flow representations. Instead, we reseeded pathlines at the same seed point over regular time intervals to generate pathline series of length $S$. We represented each pathline sequence as a stack of binary images of size $S \times L \times L$ associated with each seed point.

### 4.3.3 LSTM Autoencoder

We used an LSTM autoencoder to learn from the pathline sequences. The LSTM autoencoder consisted of two components, an encoder, and a decoder, as shown in Figure 4.3. The encoder learns the spatiotemporal features of the pathline sequence.

The decoder reconstructs the pathline sequence by using the learned spatiotemporal feature representation.

We specify the input layer of the LSTM autoencoder to expect pathline sequences of shape $S \times L \times L$. The first and second layers of the encoder are time-distributed 2D convolutional layers. These layers consist of 128 and 64 convolutional filters, respectively. These layers process each pathline of the sequence separately and learn the spatial features of each pathline. The third and fourth layers are 2D convolutional LSTM layers. Each layer consists of 64 and 32 convolutional filters, respectively. These convolutional LSTM layers process the sequence of pathlines together and learn the temporal features of the pathline sequences. The first layer of the decoder is a convolutional LSTM layer with 64 convolutional filters. The second and third layers of the decoder are time-distributed deconvolutional layers with 64, and 128 respectively. The last layer of the LSTM autoencoder is a time-distributed convolutional layer with 1 convolutional filter. Each layer, except for the input and the output layers, is followed by a normalization layer [141]. The input volume has no padding; therefore, we set the stride of all layers to 1. The autoencoder outputs a sequence of pathlines of shape $S \times L \times L$. For the hidden layers, we use the ReLU activation [142]. For the output layer, we use the sigmoid activation.

In comparison to recent autoencoder based stream line selection methods, our method not only learns the spatial features of each pathline but also learns how each pathline is temporally related to other pathlines in the same pathline sequence. Learning these spatio-temporal features allows our method to better learn about the quasi-periodic nature of the chaotic ocean flow.

### 4.3.4 Weighted Loss Function

We trained the LSTM autoencoder by minimizing the difference between each training sample against the corresponding model prediction. The loss function used to compute this difference is a weighted binary cross-entropy loss function. Since each training sample is a pathline sequence represented as a stack of binary images of size $S \times L \times L$, we treat each sample as a binary volume where $p_i$ is 1 when the pathline crosses that voxel and 0 otherwise. $\hat{p}_i$ is the corresponding value from the predicted volume. The loss is calculated over all voxels (i.e., $N = S \times L \times L$ ).

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[w_1 \cdot p_i \log \hat{p}_i + w_0 \cdot (1-p_i)\log(1-\hat{p}_i)] \tag{4.1}$$

$w_1$ is weight assigned to voxels when $p_i = 1$. $w_0$ is weight assigned to voxels when $p_i = 0$. In contrast, previous autoencoder based neural network architectures used in flow visualization tasks used an unweighted version of the same loss function (i.e., $w_1 = w_0 = 1$ ). We found that a weighted binary cross-entropy function is better suited for our application domain for reasons discussed in Section 4.4.2.

### 4.3.5 Detecting and Visualizing Rip Currents

During inference/test time, we calculated the reconstruction error for each pathline sequence by using the loss function defined earlier. If the error is larger than threshold $T$, we labeled those pathline sequences as anomalous (i.e., rip currents). Otherwise, we labeled the sequences as belonging to normal flow. We discuss how $T$ was selected in Section 4.4.2. Once the anomalous pathline sequences were found, the corresponding seed points were connected by using a region growth algorithm to generate a region. Isolated seeds without neighbors are discarded. To visualize the rip zone, We projected this region the seed points of anomalous pathline sequences back onto the frames in the video as shown in Figure 4.1.

### 4.3.6 Network Training

We implemented our neural network in Tensorflow by using the Keras application programming interface (API)[143]. We trained the network using an NVIDIA Tesla A100 graphical processing unit (GPU). In the training process, we initialized parameters in all layers of the neural network using a normal distribution $\mathcal{N}(\mu, \sigma^2)$, where mean $\mu = 0$ and variance $\sigma^2 = 0.01$. We applied the Adam optimizer [38] to update the parameters with a learning rate of $10^{-6}$. We used the minibatch size of 10 and trained our model with 100 epochs.

## 4.4 Results and Discussion

We describe our dataset, provide an analysis of the components and parameters used in RipViz, and present comparisons with other methods.

### 4.4.1 Dataset

Our dataset consists of 55 stationary videos. Each video is 1.5 to 4 minutes long and of size $1920 \times 1080$ pixels. We collected the data from Salinas, Marina, and Davenport beaches in California in winter of 2022. We used either a tripod-mounted Canon EOS Rebel T7 DSLR camera or a Samsung Android phone to acquire the data. We chose 4 videos to generate 16000 non-rip pathline sequences for training our model by using the seeding strategy discussed in Section 4.4.2. Our validation dataset consists of 12000 pathlines generated from three videos. The remaining videos were used for testing. Note that our training data does not include examples of each of the many types of rip currents, only examples of normal ocean flow. This greatly simplifies collection of training data. Each video was labeled under the guidance of the rip current expert. Pathlines seeded in the rip current area were labeled as “rip” and others as “non-rip”.

The framing of these videos, which includes the distance of camera to the water and the focal length or zoom factor of the lens, is meant to be representative of surf webcams (e.g. surfline.com) that might be useful for rip current detection. Parameters such as integration length $n$ and binary image size $L$ are based on such framing. Sensitivity of

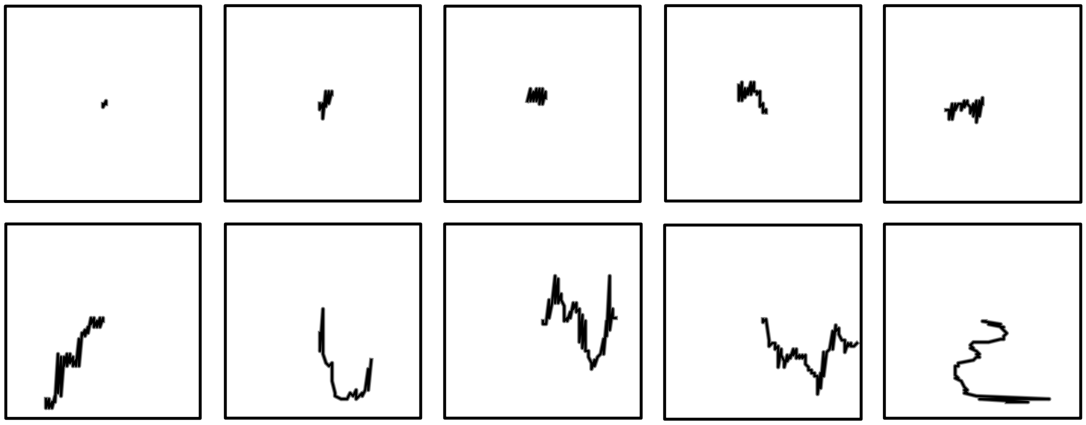


Figure 4.4: **Sparsely distributed pathlines in binary image:** This figure shows a collection of pathlines from our data. Notice the imbalance in 0s (white) and 1s (black) in the binary images. The weighted loss function, as described in Equation 4.1 allows the autoencoder to learn with imbalanced data.

these parameters on different framings are discussed in Section 4.4.2.

Note that we assume a stable video source and hence did not perform any video stabilization. Wind can cause slight movements of the camera, but also movements of grass, clouds, etc. Also, video processing seldom work with raw video but rather on compressed video. Deriving the optical flow field on compressed video may also introduce some motion artifacts especially in highly compressed regions such as the sky or empty sandy beaches. RipViz handles these type of motion artifacts better than existing methods for our dataset.

### 4.4.2 Analysis of RipViz components

The RipViz method contains two important changes from prior methods: the use of a weighted loss function, and the use of a sequences of pathlines rather than a single pathline. Without these changes we found the ML fails to learn ocean flow. In addition, the method has parameters like detection threshold, $T$, and binary image size $L$ which are likely dependent on our specific application domain. In this section we analyze each of these factors, showing that our changes are necessary, and providing the method by which we determined parameters.

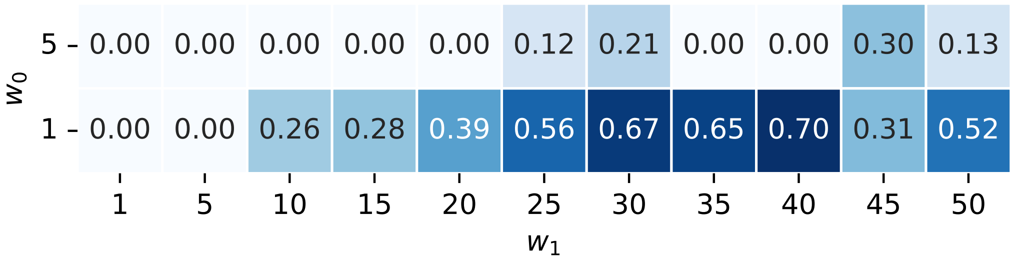
**Threshold $T$:** Threshold $T$ is used to filter anomalous pathlines (i.e, rip currents) based on the reconstruction error as discussed in Section 4.3.5. In order to find the optimal threshold $T$ we calculated $F_1$ score at varying threshold values in a subset of our data. $F_1$ score is defined as,

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = \frac{2}{\frac{FNs+TPs}{TPs} + \frac{FPs+TPs}{TPs}} \tag{4.2}$$

If a detected pathline falls within the expert annotated boundary of the rip current, then it's counted as a TPs (true positive). Otherwise, it's considered an FP (false positive). Suppose a pathline originating within the rip current is not detected, then it's an FN (false negative). The range of the $F_1$ score is between 0 and 1. If most pathlines fall within the rip current boundary, the score will be closer to 1, otherwise closer to 0. We found that threshold, $T = 1.0$ produced the highest $F_1$ score , and was used as $T$ for the experiments in this paper.

**Weights of the loss function $w_0$ and $w_1$:** We reconstructed the flow field from videos using optical flow. As discussed in Section 4.3.2, all pathlines are represented in a binary image **I**. We observed that some pathlines are short and do not extend far from the seed point, while other pathlines are longer and travel farther from the seed point, as shown in Figure 4.4. We needed the model to learn the distribution of these shorter and longer pathlines because it allowed the model to learn the *normal* flow behavior in an ocean scene. However, this leads to an imbalance in the number of voxels with 1s and 0s, especially for shorter pathlines. For the deep learning model to learn effectively about these small features/pathlines, we needed to increase the weight of the loss function when the target probability ($p_i$) is 1. The loss function, as described in Section 4.3.4, penalizes the model more for making mistakes when predicting pixels with the target probability 1 in the training process. Using the weighted binary cross-entropy function made the model better learn the distribution of shorter from longer pathlines. In contrast, in FlowNet [133], the streamlines were well spread out across their input volumes, making the use of a weighted loss function unnecessary.

To find the optimal $w_0$ and $w_1$ for our application domain, we randomly choose a small subset of pathline sequences, some short and some long, to train models at different combinations of $w_0$ and $w_1$. We trained each model for 100 epochs, using the same training conditions specified in Section 4.3.6. For each trained model, we



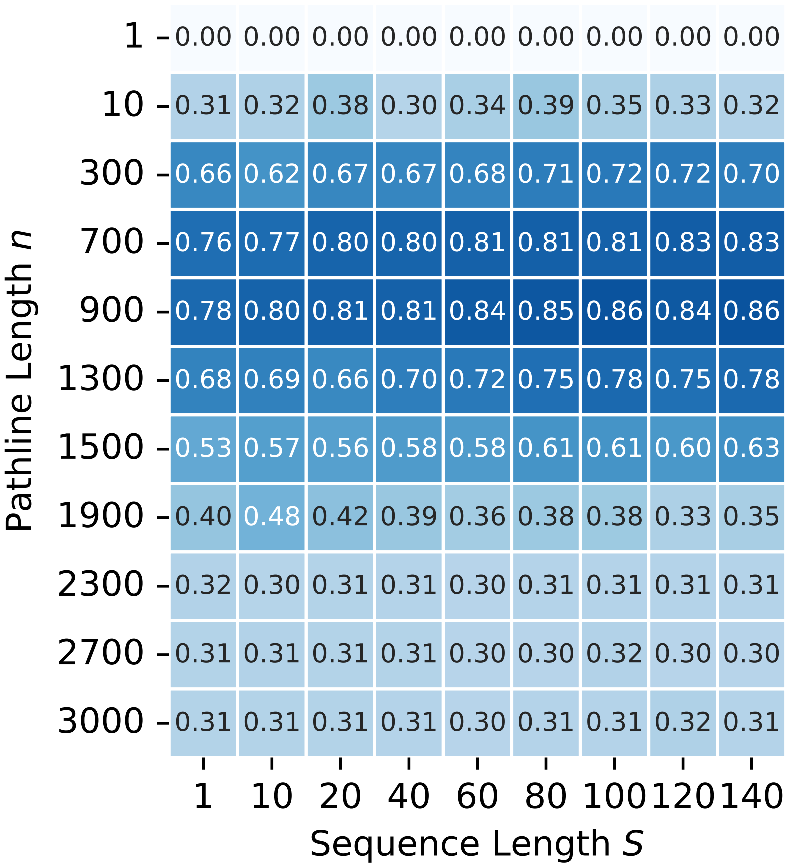
| $w_0$ \ $w_1$ | 1 | 5 | 10 | 15 | 20 | 25 | 30 | 35 | 40 | 45 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 5 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.12 | 0.21 | 0.00 | 0.00 | 0.30 | 0.13 |
| 1 | 0.00 | 0.00 | 0.26 | 0.28 | 0.39 | 0.56 | 0.67 | 0.65 | 0.70 | 0.31 | 0.52 |

Figure 4.5: $F_1$ **Scores for different combinations of $w_0$ and $w_1$:** Notice that the method does not converge when $w_0$ and $w_1$ are balanced. The highest $F_1$ score of 0.70 is achieved when $w_0$ is set to 1 and $w_1$ is set to 40.

observed how many voxels were correctly recreated. If the predicted probability for a voxel with target probability of 1 is greater than 0.7, we marked that voxel as TP; otherwise, it is considered to be FN. If the model predicted pathline voxels in areas with no pathline, we marked that as FP. We calculated the $F_1$ score for each model with different weight combinations using equation 4.2. We observed that non-uniform weighting is necessary, with the highest $F_1$ score generated when $w_0 = 1$ and $w_1 = 40$ as shown in Figure 4.5. Note that using unweighted cross-entropy, as in previous work, is the equivalent of setting $w_0 = w_1 = 1$. In this condition we found that the model completely fails to learn ocean flow.

In some instances where the model did not converge, TPs were 0, which resulted in an undefined $F_1$ score. In such instances, we followed the default reporting convention of Scikit-learn python library [144] and reported those $F_1s$ as 0.00.

**Pathline Length $n$ and Sequence Length $S$:** In RipViz, we use pathline sequences to learn flow behavior. In order to find the optimal pathline length, $n$, and sequence



| | 1 | 10 | 20 | 40 | 60 | 80 | 100 | 120 | 140 |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 |
| 10 | 0.31 | 0.32 | 0.38 | 0.30 | 0.34 | 0.39 | 0.35 | 0.33 | 0.32 |
| 300 | 0.66 | 0.62 | 0.67 | 0.67 | 0.68 | 0.71 | 0.72 | 0.72 | 0.70 |
| 700 | 0.76 | 0.77 | 0.80 | 0.80 | 0.81 | 0.81 | 0.81 | 0.83 | 0.83 |
| 900 | 0.78 | 0.80 | 0.81 | 0.81 | 0.84 | 0.85 | 0.86 | 0.84 | 0.86 |
| 1300 | 0.68 | 0.69 | 0.66 | 0.70 | 0.72 | 0.75 | 0.78 | 0.75 | 0.78 |
| 1500 | 0.53 | 0.57 | 0.56 | 0.58 | 0.58 | 0.61 | 0.61 | 0.60 | 0.63 |
| 1900 | 0.40 | 0.48 | 0.42 | 0.39 | 0.36 | 0.38 | 0.38 | 0.33 | 0.35 |
| 2300 | 0.32 | 0.30 | 0.31 | 0.31 | 0.30 | 0.31 | 0.31 | 0.31 | 0.31 |
| 2700 | 0.31 | 0.31 | 0.31 | 0.31 | 0.30 | 0.30 | 0.32 | 0.30 | 0.30 |
| 3000 | 0.31 | 0.31 | 0.31 | 0.31 | 0.30 | 0.31 | 0.31 | 0.32 | 0.31 |

Figure 4.6: **Pathline length (*n*) vs sequence length (*S*)**: $F_1$ scores for different combinations of $n$ nd $S$. The first column represents traditional pathlines, while subsequent columns are for sequence of pathlines. Notice that increasing the pathline length resulted in higher accuracy up to an optimal length; beyond this, the accuracy decreased. Also notice that using a sequence of pathlines over a single pathline per seed point produced even more accurate results. We found that $n = 900$ and $S = 100$ were optimal for our data set.

length, $S$, for our application domain, we trained multiple models while changing $n$ and $S$, and observed the $F_1$ score as shown in Figure 4.6. We noticed two patterns from this experiment. First, increasing the pathline length resulted in a higher accuracy up to an optimal length; beyond this, the accuracy decreased. Second, using a sequence of pathlines instead of a single pathline per seed point produces more accurate results. We

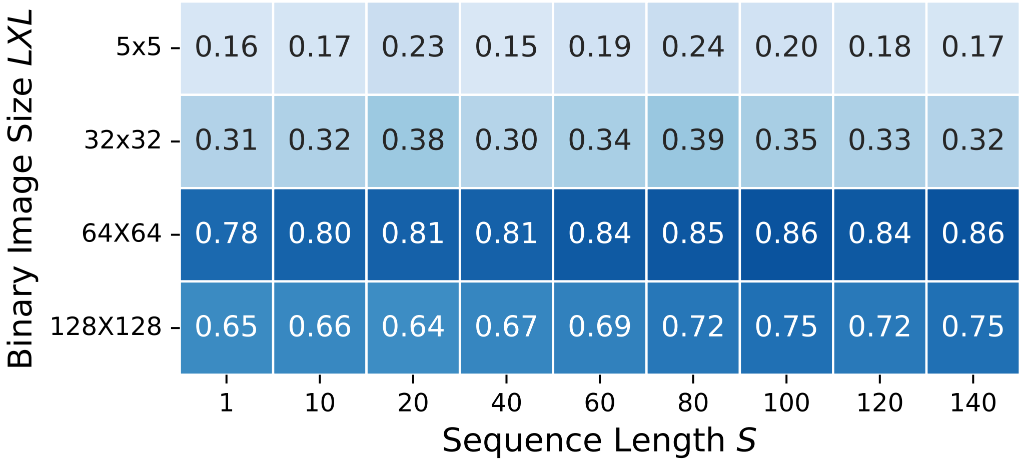
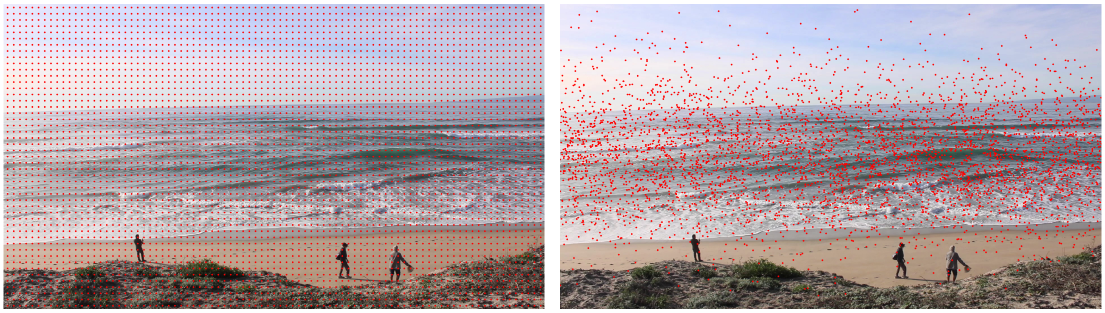


Figure 4.7: **Binary image size ($L \times L$):** Notice that a binary image size of $64 \times 64$ produced the highest $F_1$ score with least amount of computation.

found that $n = 900$ and $S = 100$ were optimal for our data set.

**Size of binary image $I = L \times L$:** We represented each pathline in a binary image as discussed in 4.3.2. We trained several models with our training dataset where binary images were of different sizes as shown in Figure 4.7. We compared each model by calculating the $F_1$ score, similar to how $F_1$ score was calculated for tuning Threshold $T$. Our experiments indicated that small binary images ($L \times L = 5 \times 5$) cannot sufficiently capture the variations of pathline movements, resulting in a lower $F_1$ score. We also found that larger binary images ($L \times L = 128 \times 128$) also resulted in a slightly lower $F_1$ score. Furthermore, larger binary images resulted in a high memory usage and a longer training time. We found that $L \times L = 64 \times 64$ is sufficient at capturing the variations of pathlines resulting in a high $F_1$ score. In our experiments we used $L \times L = 64 \times 64$ as the size of the binary image $I$.

**Seeding strategy:** Various seeding strategies attempt to optimize different goals such as aesthetics, clutter reduction, capturing flow features, etc [124]. In this paper, the primary consideration for pathline seeding take into account where one can have the highest return in informational value regarding the water movement. Because most of our training data and random beach images that one can find on the web are taken in landscape mode, seeds are distributed uniformly along the horizontal axis. Conversely, the water body is usually in the middle, with the sky on top and the beach on the bottom halves of the framing. We therefore use a Gaussian distribution of seeds along the vertical axis. This is a form of importance sampling [145]. During the training process, we use importance seeding to maximize pathlines that track water movement, and reduce irrelevant information such as the beach or cloud movement in the sky. During the testing process, we use regular sampling since there is no guarantee that the test data will also be in landscape mode.



(a) Regular grid seeding

(b) Importance seeding

Figure 4.8: **Seeding strategies:** We used regular grid to seed pathlines when testing. For training we used importance seeding.

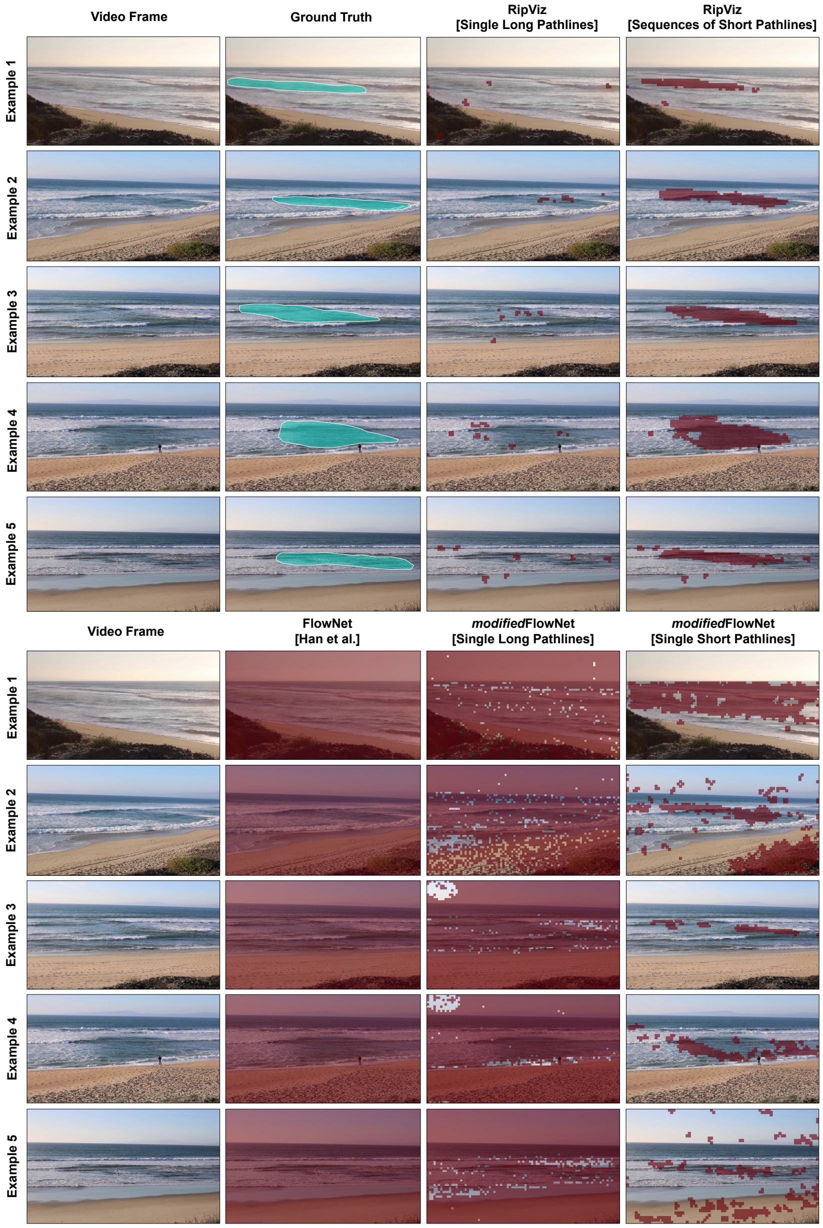

Video Frame
Ground Truth
RipViz [Single Long Pathlines]
RipViz [Sequences of Short Pathlines]
Example 1
Example 2
Example 3
Example 4
Example 5
Video Frame
FlowNet [Han et al.]
*modified*FlowNet [Single Long Pathlines]
*modified*FlowNet [Single Short Pathlines]
Example 1
Example 2
Example 3
Example 4
Example 5

Figure 4.9: **Comparison with FlowNet and different pathline parameters:** The *Video Frame* column shows a representative frame of the video clip while the *Ground Truth* column shows the expert's ground truth estimate. Notice that FlowNet without modifications (*FlowNet* Column) incorrectly clusters all seed points into a single group. Both *modified* Flownet and RipViz when trained with longer pathlines were not able to detect the rip current. When *modified* FlowNet was trained on single shorter pathlines ($n = 900$), we were able to find a cluster representing the rip current but not as definitive. RipViz trained with shorter pathline sequences was able to detect the rip currents more precisely. Best viewed in color.

### 4.4.3 Comparison with FlowNet using Single Long and Sequences of Short Pathlines

We compared RipViz with FlowNet [133], a deep learning based streamline clustering method. In FlowNet, 3*D* streamline features were learned by an autoencoder. These 3D streamlines were traced across the full extent of the data. Additionally, they traced one streamline per seed point. The trained autoencoder was used to generate 1D representations of 3D streamlines. The dimensionality of these 1*D* representations is further reduced by t-distributed stochastic neighbor embedding (t-SNE) [146]. The resulting low dimensional vectors are then clustered by using DBSCAN [147]. An interactive user interface was used to find the desired clustering by changing the maximum distance between two feature descriptors (*eps*) and the minimum number of samples in each cluster (*ms*).The autoencoder in FlowNet is trained with a unweighted binary cross-entropy function (i.e., $w_0 = w_1 = 1$).

In order to compare with FlowNet, we transformed our long 2*D* pathlines to 3*D* by adding a time axis as the $z$ axis [148]. However, since our pathlines are sparsely distributed in the 3*D* binary volume, we found that the FlowNet model with the unweighted binary cross entropy did not learn to differentiate the pathlines. Resulting in a single cluster for all the pathlines. We show the clustering result in Figure 4.9. Additionally, the training time was relatively long. A proper comparison requires modification of FlowNet to work with our application domain. The modifications are discussed in Appendix A.

We trained the modified FlowNet model with long pathlines, integrated across the entirety of the video, and with single short pathlines ($n = 900$). The output of the *modified* FlowNet is shown in Figure 4.9. The first column shows a representative frame of each video. We see that without modifications, FlowNet lumps all the pathlines into one cluster. Both the modified FlowNet and RipViz, when fed with single pathlines that run the entirety of the video, also fail to identify the rips. When fed with shorter pathlines ($n = 900$), modified FlowNet start to show signs of rip currents but includes significant FPs as well. Only, when sequence of pathlines as used in RipViz do we see the rip currents clearly.

In order to obtain a quantitative comparison between *modified* FlowNet with shorter pathlines and RipViz, we calculated their $F_1$ scores. If a seed point is flagged as as a rip by either method is within the expert annotated rip current boundary then we mark it as TP, otherwise we mark it as FP. If seed points within the rip are not selected then we mark those points FN. We found that the $F_1$ score for *modified* FlowNet was 0.32 compared to the $F_1$ score of 0.85 for RipViz as shown in Table 4.2. We hypothesize the low $F_1$ score for the *modified* FlowNet was due to its high FP rate.

Furthermore, we found tuning the two hyperparamters *eps* and *ms* for the clustering step was crucial in finding the appropriate clusters. We noticed having some domain knowledge was helpful for the user when exploring these two hyperparameters to find the ideal clustering. In contrast, RipViz is automated and does not require any user input during run time.

Both *modified* FlowNet and RipViz are based on autoencoders. However, RipViz learns how each pathline of the same seed point was related in time by learning spatiotemporal features of pathline sequences. This allows RipViz to learn ocean flow behavior more effectively. On the other hand, *modified* FlowNet does not learn how pathlines are related in time. Therefore, we hypothesize that using pathline sequences is better for learning quasi-periodic behavior, such as in ocean flow.

### 4.4.4 Comparison with existing rip detection methods

We compared RipViz with existing rip current detection methods as shown in Figure 4.10. The first and second columns of Figure 4.10 show a representative frame of the video and the expert drawn ground truth. The remaining columns show the output of the object detector [119], RipViz, filtered arrow glyphs [123], filtered color maps [123], and timelines [123], respectively.

#### 4.4.4.1 Comparison with behavior based methods

**Timelines:** Mori et al. [123] proposed to use timelines to detect and visualize rip currents. They placed timelines parallel to the beach and traced the points on the timeline using optical flow to update its position. They observed the shape of the timeline as it gets dragged by the rip current. Timelines get deformed and extend to the rip channel above its initial position, as shown in Figure 4.10.

However, the initial placement of the timeline is a major contributing factor for timelines to be successful. All the timelines indicated in Figure 4.10 are placed optimally. To compare the timeline method with RipViz, we calculated the $F_1$ score. If the timeline was able to visualize the rip current in a video, then we counted it as TP, otherwise it is counted as FN. We found that properly placed timelines can visualize rip currents in most cases. However, the user has to specify the good initial placement for each video. In contrast, RipViz is automated, and no user input was needed during run time, as shown in Table 4.2.

**Direction based clustering:** Philip and Pang [122] hypothesized that the rip current is directly opposite the dominant flow in the vector field, which corresponds to the incoming wave direction. Places where the vectors were opposing the majority flow are potential rip zones. Areas with sufficiently large clusters of such vectors and large enough magnitude were highlighted as rip zones.

Mori et al. [123] used a similar approach and mapped direction to color and magnitude to hue for better visualization. As shown Filtered Color Map column of Figure 4.10, the method can detect the location of the rip current. However, this method also highlighted the swash zone, the shallow part of the beach, as evident in examples 1-3. We found that while the rip current was highlighted in most cases, the number of false positive pixels tend to be high. In order to compare with RipViz, we calculated the $F_1$ score. As shown in Table 4.2 filtered color map resulted in a $F_1$ score of 0.28, due to its high false positive rate.

Mori et al. [123] also proposed arrow glyphs to visualize rip currents. The output of this method is shown in Arrow Glyph column of Figure 4.10. We noticed that the arrow glyph method is more susceptible to noise in the flow field compared to other methods and RipViz. This is evident by the arrows projected on the sky and the beach although there is a little movement on those parts of the video. Although arrow glyph method highlighted the rip current in the majority of the videos, there were also a large number of false positive detections. Similarly we calculated the $F_1$ score. As indicated in Table 4.2, filtered arrow glyph had a $F_1$ score of 0.16 due is relatively high false positive rate.

#### 4.4.4.2 Comparison with appearance based methods

**Object Detectors:** de Silva et al. [119] used object detectors to detect rip currents. They trained a deep learning based object detector, Faster R-CNN [79], with images of rip currents. The images they used had a clear visual signature for bathymetry controlled rip currents with a darker region between breaking waves. Then they used the model to predict the location of rip currents, by overlaying a bounding box on the rip currents on videos. The output of the method is shown in the *Object Detector* column of Figure 4.10.

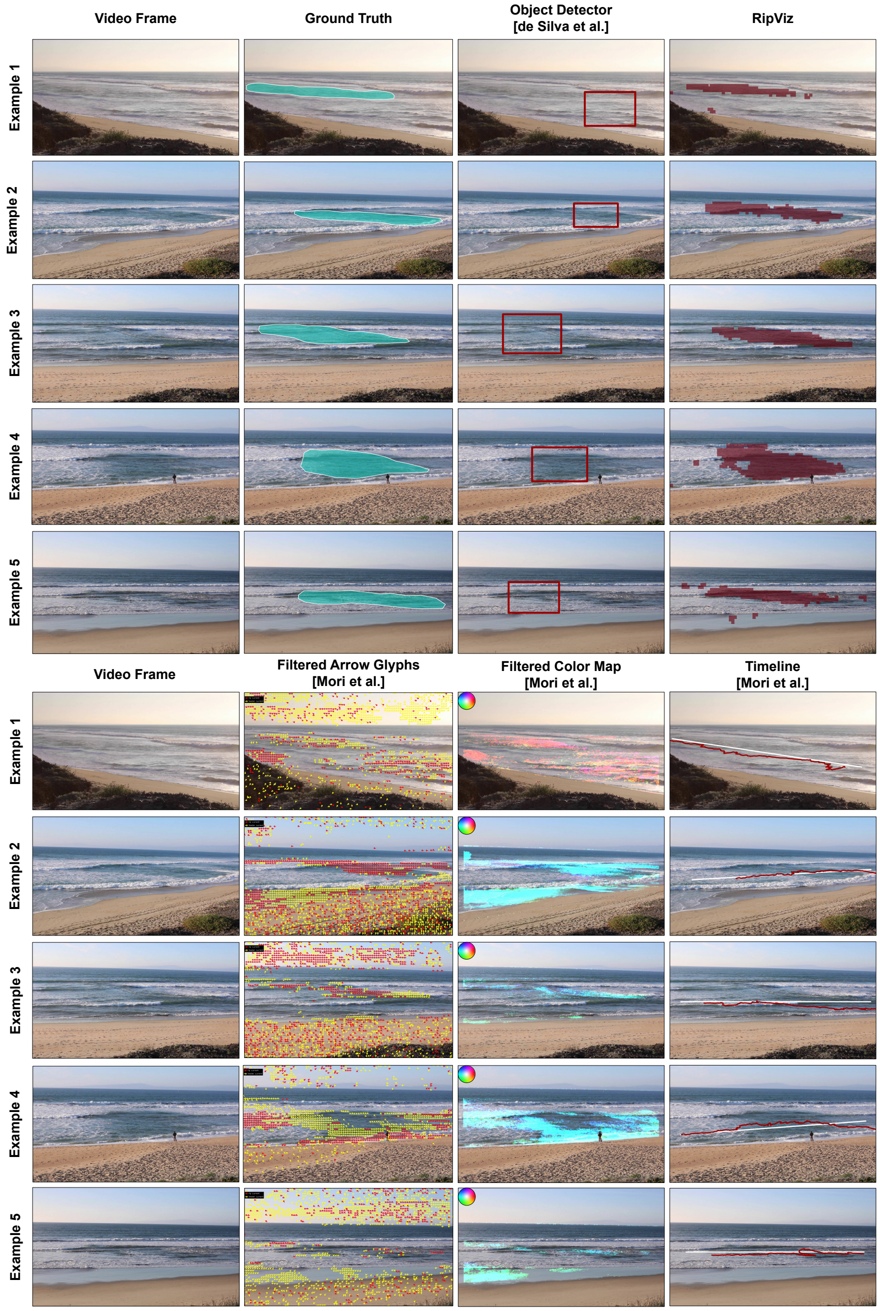


Figure 4.10: **Comparison with prior rip current detection methods:** The *Video Frame* and *Ground Truth* columns show a representative frame of the video clip and the expert ground truth estimate respectively. The remaining columns show object detector, timeline, colormap, arrow glyph, and RipViz, respectively. Note that the timeline method requires user input to specify its initial placement, and that Filtered Arrow Glyphs did not filter out erroneous glyphs from the sky, beach, and non-rip area of the water. Best viewed in color.

We noticed that the object detector can detect parts of rip currents as shown in examples 2-5 in Figure 4.10. However, rip currents have different appearances [61]. As evident in example 1, it is possible for an objector detector to miss the actual rip current location. In contrast, RipViz is trained on pathlines that capture the *behavior* rather than appearance of rip currents. Hence, it was able to detect the rip current in all the examples.

Furthermore, we noticed that bounding boxes only detects the general area of the rip current. Sometimes the bounding boxes can cover parts of the beach or miss parts of the rip especially with long elongated rips.

We also noticed that the information such as the curvature of the rip current cannot be gleaned from bounding boxes alone.

In order to compare with RipViz, we calculated the $F_1$ score. We counted how many pixels were covered by the bounding box. Out of those, rip pixels are counted as TP, the remaining pixels were counted as FP. If the object detector does not predict a bounding box, then we counted the pixels within the boundary as FN. As shown in Table 4.2, the bounding boxes resulted in a $F_1$ score of 0.43 compared to 0.85 for RipViz. We attribute this lower $F_1$ score to the large number of false positives and false negatives generated respectively by the non-rip areas of the bounding box and by part of the rip current not being covered by the bounding box.

Additionally, we tested RipViz on other types of rip currents where currently there

| Method | Automated? | $F_1$ Score |
|---|---|---|
| Object Detector [119] | ***yes*** | 0.43 |
| Timelines [123] | *no* | 0.72 |
| Filtered color map [123] | ***yes*** | 0.28 |
| Filtered arrow glyph [123] | ***yes*** | 0.16 |
| FlowNet[133] | *no* | 0.16 |
| *modified* FlowNet [single long pathlines] | *no* | 0.19 |
| *modified* FlowNet [single short pathlines] | *no* | 0.32 |
| RipViz [single long pathlines] | ***yes*** | 0.08 |
| RipViz [sequence of short pathlines] | ***yes*** | **0.85** |

Table 4.1: **Results summary:** Notice that RipViz has the highest $F_1$ score, and does not rely on user input at run time.

are no published object detector available. In Figure 4.11, we show a few examples of such rip current types. Experts categorize the rip currents in the first and second row as sediment and structural rips. However, as shown in Figure 4.11 RipViz could detect these rip currents regardless of their appearance. More importantly, no additional training data were needed for sediment and structural rips. Examples 3 and 4 illustrate two rip currents where object detectors failed due to a lack of expected visual features. In these two examples, RipViz detected the rip currents because it uses *behavior* rather than appearance to detect rip currents.

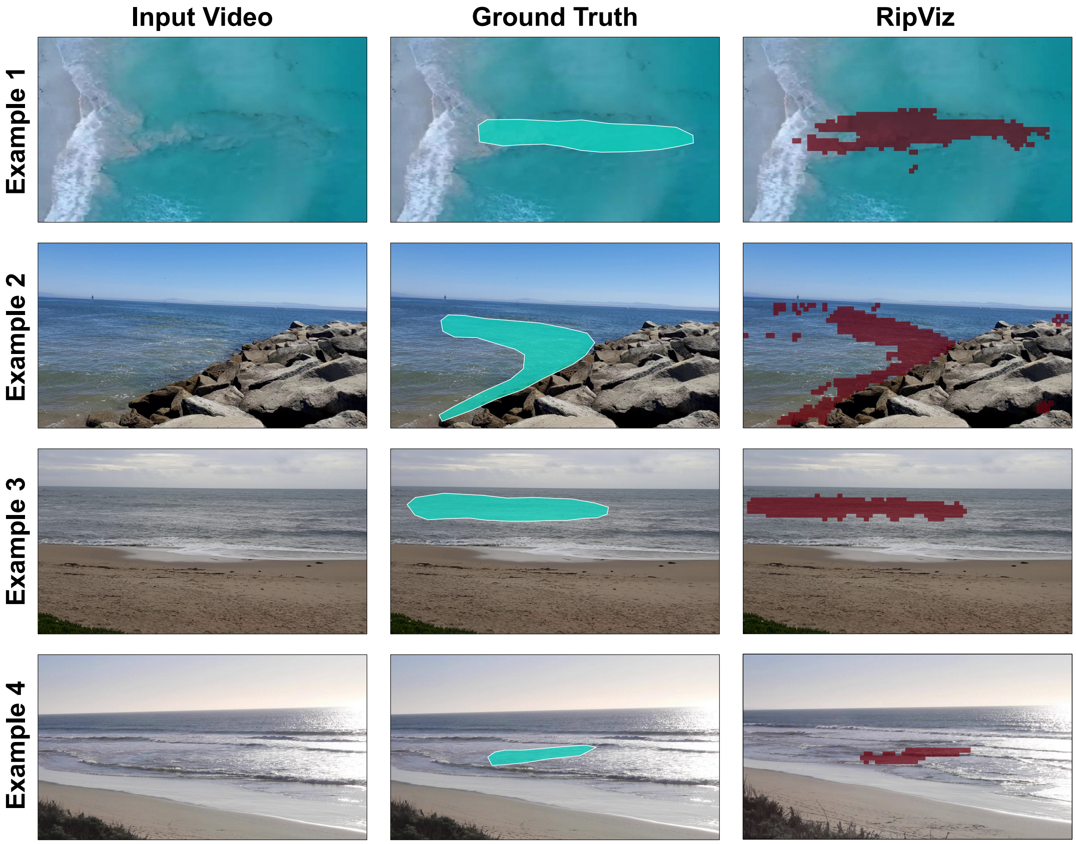


Figure 4.11: **RipViz on novel rip current types:** Examples 1 and 2 are sediment rip and structural rip, respectively. Note that they have very different visual characteristics compared to bathymetry rips. No published object detectors exist for these two types of rips. However, notice that RipViz could detect rip currents regardless of their varying appearance. Examples 3 and 4 illustrate two rip currents where object detectors failed due to a lack of expected visual features. In these two examples, RipViz can still detect the rip current because it uses *behavior* rather than appearance to detect rip currents. Best viewed in color.

### 4.4.5 Discoveries made by Experts using RipViz

Rip current researchers use their expert knowledge to identify the boundary of rip currents by observing the appearance of visual features such as gaps in breaking waves or sediment plumes. However, sometimes parts of the rip current lack these visual features, making it challenging for the expert to identify the entire extent of the rip current.

In contrast to appearance-based identification, RipViz uses the behavior of rip currents, not appearance, to detect rip currents. Experts can use RipViz as a visualization tool to determine the entire boundary of the rip current when visual features are lacking. In Figure 4.12, we demonstrate a few use cases where the rip current expert's original boundary estimate was updated after examining the visualization provided by RipViz. The updated boundaries now include feeder currents near shore and what appears to be a circulating pattern farther offshore in Discovery 1 and Discovery 3, as well as a weaker neighboring rip that merged with the dominant rip in Discovery 2. Incidentally, Discovery 1 and 3 both show circulating rips where the guidance from beach signage to swim parallel to shore may not always work [115]. We offer the complete table of discoveries made on our test data in the supplementary materials.

### 4.4.6 User Study

We conducted a user study to better understand how non-experts can use RipViz as a tool to become more aware of rip currents. We grouped 400 non-experts into two groups; one group was trained only with beach warning signs of rip currents, the other group was trained with RipViz. We then showed each participant a randomly picked video of a rip current and asked them if there was a rip current present. In the group

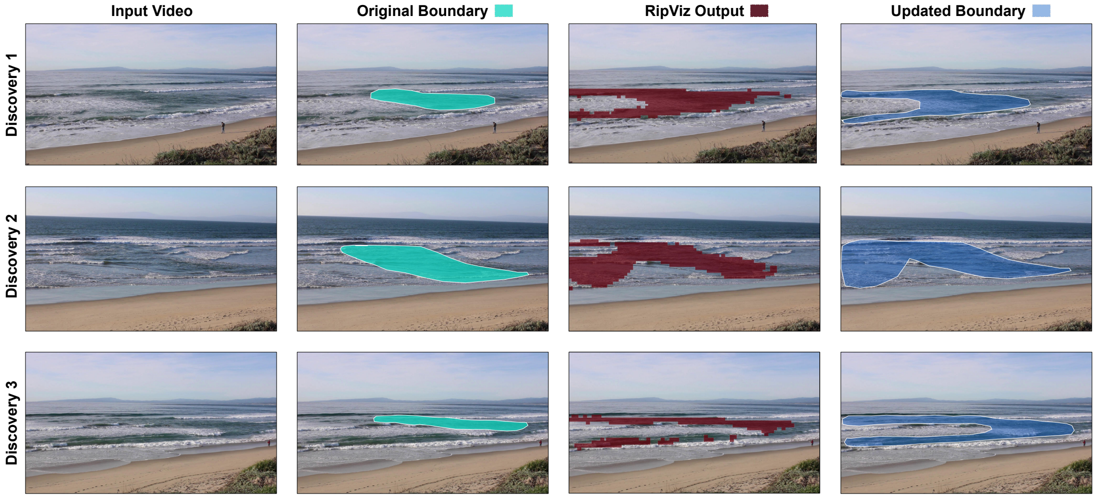


Figure 4.12: **Discoveries made by Experts using RipViz:** Experts estimated the rip current boundary (*Original Boundary* Column) by observing only the input video (*Input Video* Column). After examining the output of RipViz (*RipViz Output* column), the experts updated their original rip current boundary estimate to include parts of the rip current that are not readily observable in the input video due to a lack of distinct visual features (*Updated Boundary* column). In the supplementary materials, we included a complete table of discoveries made on our test data.Best viewed in color.

trained with beach signs, 24% failed to recognize the existence of the rip current in the video. In the group trained with RipViz, only 14% were unable to recognize the presence of the rip current in the video.

We performed a second experiment to better understand if the participants could locate the rip current within the video. Both groups were given multiple choices of rip current boundary estimates and were asked to pick the correct one. In the group trained with beach signs, only 34% could choose the correct boundary. In the group trained with RipViz, 78% could select the correct boundary. The non-experts were acquired using Mechanical Turk [109, 110], with basic screening for reliable workers, and paid

$0.10-$0.20 per task.

### 4.4.7 Sensitivity to video framing

Some of the parameters are dependent on the camera framing which would include distance of camera to the water and focal length or zoom factor of the lens. Alternatively, we can think about the width of the beach that is visible in the frame. We base our framing by examining a number of existing surf webcams (e.g. webcoos.com) and planned webcam installations. We experimented on how sensitive our parameters are to changes in camera framing. In the Figure 4.13, we show effects of varying the framing while keeping our current set of parameters. Recall that during testing, we trace the same number of seed points that are distributed in a regular grid. Based on the results shown in Figure 4.13, we believe the parameters are still valid within 20-30% change in camera framing. We found the $F_1$ score to be $0.82 \pm 0.3$ for all the examples, without much deviation.

## 4.5 Summary and Remarks

As Ben Schneiderman succinctly captured in his quote: “The purpose of visualization is insight, not pictures”, our goal is to make apparent what may not be visible to the untrained eyes. The focus of this work is to first find the feature of interest (rip) and then present them in a simple, easy to understand manner by highlighting their location

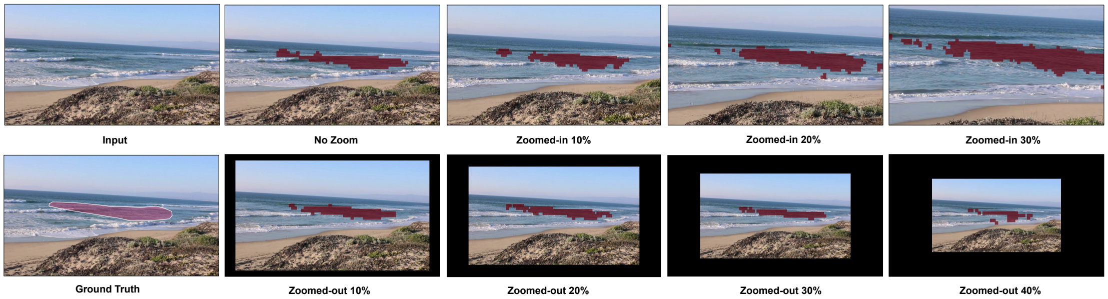


Figure 4.13: **RipViz is less sensitive to different video framing:** We changed the scale of the rip current feature by zooming in and out from the original video, while keeping the aspect ratio of the video unchanged. Notice that for instances where the rip current feature is larger or smaller than the training data (within reasonable bounds), RipViz was still able to detect the rip current feature.

directly on the video. This capabality is encapsulated in RipViz, a hybrid feature detection method that combined ML and flow analysis to extract rip currents from stationary videos. We used shorter pathline sequences to capture the flow behavior in a noisy quasi-periodic flow field. Then we used an LSTM autoencoder to learn the behavior of normal ocean pathline sequences. The trained model allowed us to label pathline sequences belonging to rip currents as anomalous by comparing the reconstruction error. By framing the rip detection problem as an anomalous flow detection problem, the onerous task of finding and labeling training datasets for each type of rip current is also greatly reduced.

The Visualization community has used deep learning methods to learn flow behavior from pathlines. In particular, existing literature uses autoencoders to learn from flow data. The authors use single long pathlines/streamlines per seed point as input to their deep learning models. However, straight-forward learning based on traditional pathlines did not work for our application due to the quasi-periodic flow fields such as those found near the surf zone. Therefore, we adapted and innovated on the existing deep learning methods to use a sequence of pathlines per seed point instead

The main contribution of this chapter is:

- A hybrid feature detection method that combines machine learning and flow analysis techniques to automatically find and visualize dangerous rip currents.

In order to realize this, the following innovations are necessary:

- For flow fields with a quasi-periodic behavior, such as ocean flow, working with a *sequence* of shorter pathlines is better than a single long pathline.
- A weighted binary cross entropy, unlike the unweighted binary cross entropy used in previous works, is more effective when learning from sparsely distributed pathlines generated from ocean scenes.

A key assumption about the stationary videos is that the camera is sufficiently close to the water in order for the optical flow algorithm to pick up measurable velocities. For similar reasons, we assume that the camera is pointed mostly seaward and not parallel to the beach. Pointing the camera down a long stretch of beach will create an optical flow that is not representative especially for points farther away from the camera. Rectifying the frames prior to optical flow calculations may extend the usable range a bit farther but does not justify the extra computational cost. For these reasons, RipViz works best when the camera is close to the water and pointed mostly seaward.

Our training data were videos from mostly uncrowded beaches. We did study seed points along the path of a jogger. Such seed points were not marked as anomalous. This is likely due to the fact that the subsequent pathlines from the sequence marked the initial seed point as mostly normal. We believe that seed points that are placed on surfers or birds or other objects in the scene will likely be marked as normal as well. We plan to study this further with more testing data.

*Extensions and Future Works:* Some of the contributions listed above are not specific to rip current detection. We plan to investigate how pathline sequences of unsteady flow fields coupled with the weighted cross-entropy loss function can be used to distinguish between laminar and non-laminar flows, and possibly train a model to detect vortices as anomalous behavior.

We surmise that analysis using pathline sequences may also benefit certain classes of flow data aside from those in this paper. In fact, we are exploring that avenue and plan to report on the results in a separate paper. In short, the particular needs of rip current detection led us to develop the approach presented in this paper, and which also provides another potential tool for the visualization community to study certain classes of flow fields.

One of the obvious questions, we had about using pathlines is, can we use pathlines to find other types of flow patterns. In Chapter 05, we explore the use of pathlines and streamlines to find vortices. What are the other types of flow patterns we can find using flowlines. We explore this by looking at vortices in the next chapter.

# Appendix A: Modified FlowNet

Han et al. [133] proposed, FlowNet, a deep learning based method to cluster and select streamlines. They first transformed each streamline into a 1D feature vector of length 1024. Then they used the t-distributed stochastic neighbor embedding (t-SNE) [146] to reduce the dimensionality of the 1D vector from 1024 to 2. The resulting vectors are then clustered using DBSCAN[147], a density-based spatial clustering method for applications with noise. The authors provided a user interface to vary the clustering by changing the maximum distance between two feature descriptors (*eps*) and the minimum number of samples in each cluster (*ms*) in the DBSCAN algorithm.

We started with the FlowNet architecture for this work, but modified it according to the requirements of our application. Their primary focus was to cluster streamlines from 3D steady flow fields. Therefore, the authors designed the deep neural network architecture to take in a 3D streamline as input. However, in our application domain, the pathlines are 2D. In order to adapt FlowNet to our application domain, we changed the architecture of FlowNet to take in a 2D pathlines as input. Likewise, we changed the original neural network architecture by replacing 3D convolutional layers and 3D batch normalization layers, respectively, with 2D convolutional layers and 2D batch normalization layers to accommodate 2D pathlines. We also updated the input size and output size of the fully connected layers accordingly. We kept the same number of layers, same activation functions, and the same number of convolutional filters as defined in the FlowNet paper. We refer to the new neural network architecture as *modified* FlowNet.

We also found that the unweighted binary cross-entropy function used in the FlowNet paper was insufficient to learn the variations of pathlines of the ocean flow for reasons discussed in Section 4.4.2. Therefore, we used the weighted binary cross-entropy loss function defined in Section 4.3.4 when training the *modified* FlowNet. We trained the *modified* FlowNet using the same approach as discussed in the FlowNet paper[133].

We used the trained *modified* FlowNet to find 1D representations of the pathlines by extracting the latent feature descriptor from the autoencoder. We reduced the dimensionality of the 1D vector by using t-SNE to a vector with a length of 3. Then we clustered the resulting vectors using DBSCAN, which requires the user to set the maximum distance between two feature descriptors (*eps*) and the minimum number of samples in each cluster (*ms*) before clustering. The cluster with the largest overlap with the ground truth was selected.

# Appendix B: Making the Case for Shorter Pathlines in Noisy Datasets.

We estimated the flow field from videos using optical flow. We assumed a small error/noise associated with the optical flow estimate. Numerical integration, even well designed higher order methods, are subject to accumulation of error. This problem is aggravated when one is working with noisy datasets. For the case of pathline integration, we hypothesized that shorter sequence of pathlines will be more accurate in capturing the flow behavior in noisy flow fields than a single longer pathline.

We used two synthetic datasets to indirectly verify our hypothesis: the *2D unsteady Double Gyre* dataset [149] and *2D Unsteady Four Rotating Centers* dataset [150].

We added noise at 5%, 10%, and 20% levels to each vector component of each dataset. We compared the similarity of pathlines originating from the same seed point. We observed that shorter pathlines were more similar to those without any noise. To quantify our observations, we randomly seeded 200 pathlines in each dataset and traced those pathlines. We calculated mean squared error (MSE) for pathlines at varying lengths as shown in Figure 4.14. We observed that longer pathlines have higher MSE compared to shorter pathlines. Therefore, we anticipate that shorter pathlines will capture a more accurate representation of the flow behavior in reconstructed noisy flow fields such as ours.

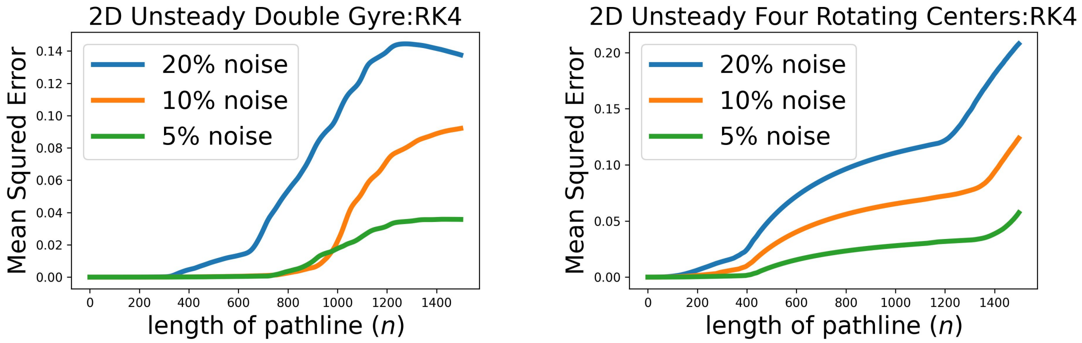


Figure 4.14: **Comparison of pathlines with and without noise:** Mean Squared Error (MSE) of pathlines calculated at varying lengths, averaged over 200 randomly seeded pathlines. Notice that for both datasets longer pathlines have higher MSE compared to shorter pathlines.

Additionally, we visualized how pathline sequences accumulate errors due to noise over longer integration times as shown in Figure 4.15. Here, we show two pathline sequences, one seeded in the sky and the other on the beach. We expected these two pathline sequences to be very short and remain closer to the seed point due to the lack of motion around those seed points. However, as we integrate beyond 900 time steps, we notice that the pathlines seem to travel far from the seed point even though we don't see any visible motion on the video. We attribute this to the accumulation of noise/error when integrating over a more extended time. The noise can come from optical flow estimation or video compression loss.

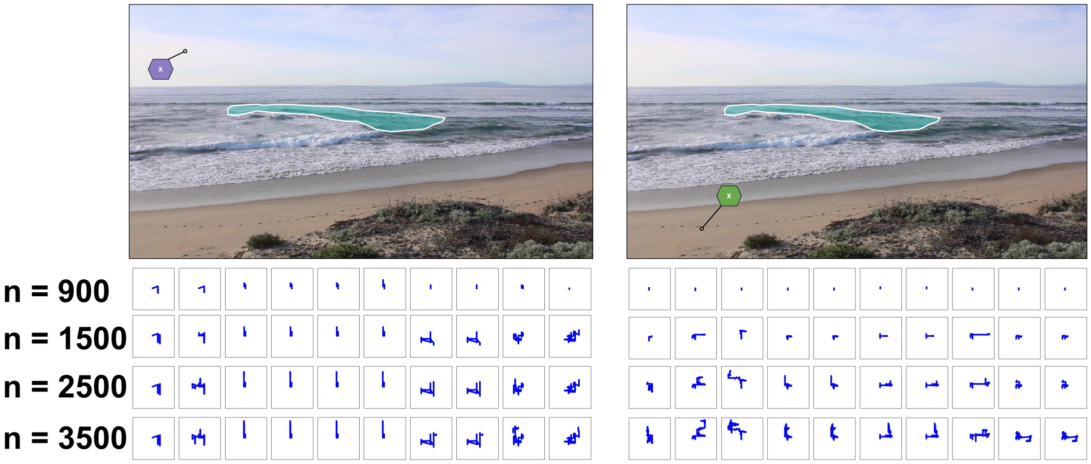


Figure 4.15: **Visualization of noise in our dataset:** Here we visualize two pathline sequences, one seeded in the sky (left panel) and the other on the beach (right panel). Each sequence consists of 10 pathlines. We show the pathlines sequence at four integration time steps(n). We expected these pathlines to be very short and remain closer to the seed point. However, as the pathlines were integrated over a longer period ($n$), they traveled far from the seed point, contradicting our expectation. We attribute this to the accumulation of noise/error when integrating over a longer period in a noisy real world dataset.

# Appendix C: Evaluation based on additional Metrics.

In addition to using the $F_1$ score, we expanded our evaluation of RipViz to include other types of metrics. The metrics we choose are the Jaccard Index($IoU$), The Hausdorff Distance ($HD$) and Frobenius Norm ($FN$). Jaccard index compares how much overlap the detection has with the ground truth. Values that are closer to one in the Jaccard index are considered better. $HD$ and $FN$ computes a distance measure between the ground truth and the detection. Lower values in these distance measures are considered better. We noticed that RipViz performed better in all the metrics. As shown in Table 4.2, RipViz scored highest value in the Jaccard index, highest value in $F_1$ score and the lowest values in $HD$ and $FN$. Since the correctness of the Timeline method is determined by a human expert, additional metrics are not applicable.

| *Method* | *automated?* | $FN(e5)$ | $HD$ | $IoU$ | $F_1 score$ |
|---|---|---|---|---|---|
| Object Detector [119] | ***yes*** | 1.0 | 29 | 0.29 | 0.43 |
| Timelines [123] | *no* | – | – | – | 0.72 |
| Filtered color map [123] | ***yes*** | 1.0 | 26 | 0.15 | 0.28 |
| Filtered arrow glyph [123] | ***yes*** | 2.5 | 34 | 0.09 | 0.16 |
| FlowNet[133] | *no* | 3.5 | 43 | 0.09 | 0.16 |
| *modified* FlowNet [longer pathlines] | *no* | 3.2 | 37 | 0.09 | 0.19 |
| *modified* FlowNet [shorter pathlines] | *no* | 2.8 | 28 | 0.20 | 0.32 |
| RipViz [longer Single Pathlines] | ***yes*** | 3.4 | 41 | 0.03 | 0.08 |
| RipViz | ***yes*** | **0.4** | **17** | **0.72** | **0.85** |

Table 4.2: **Additional Metrics:** Note that RipViz has the highest $F_1$ score, highest Jaccard Index ($IoU$), lowest Hausdorff Distance ($HD$), lowest Frobenius Norm ($FN$) and does not rely on user input at run time.

# Appendix D: Additional Visualizations of Pathline Sequences.

In Figure 4.16, we visualized how pathline sequences seeded in the rip current differ from pathline sequences from other parts of the flow field. The figure shows the expert annotated boundary of the rip current in blue. Also, each pathline sequence is labeled with the corresponding seed point and starts from the center of the binary image.

Notice that pathline sequences in the rip current region generally show a net outward water transport. In comparison, pathline sequences seeded in non-rip parts of the ocean show a general inward movement, but also includes some outward movement. Pathline sequences seeded in the beach or the sky are either still or can be longer. Also, notice that while individual pathlines from different regions may demonstrate similar behavior, pathline sequences are generally distinct.

## 4.6 Detailed view of the LSTM autoencoder.

In Figure 4.17, we provide a detailed view of the layers of the LSTM autoencoder used in this paper.

## 4.7 Example Output Videos

We link some example videos here: https://sites.google.com/view/ripviz/home

## 4.8 Complete List of Discoveries by domain expert

Rip current researchers use their expert knowledge to identify the boundary of rip currents by observing the appearance of visual features such as gaps in breaking waves or sediment plumes. However, sometimes parts of the rip current lack these visual

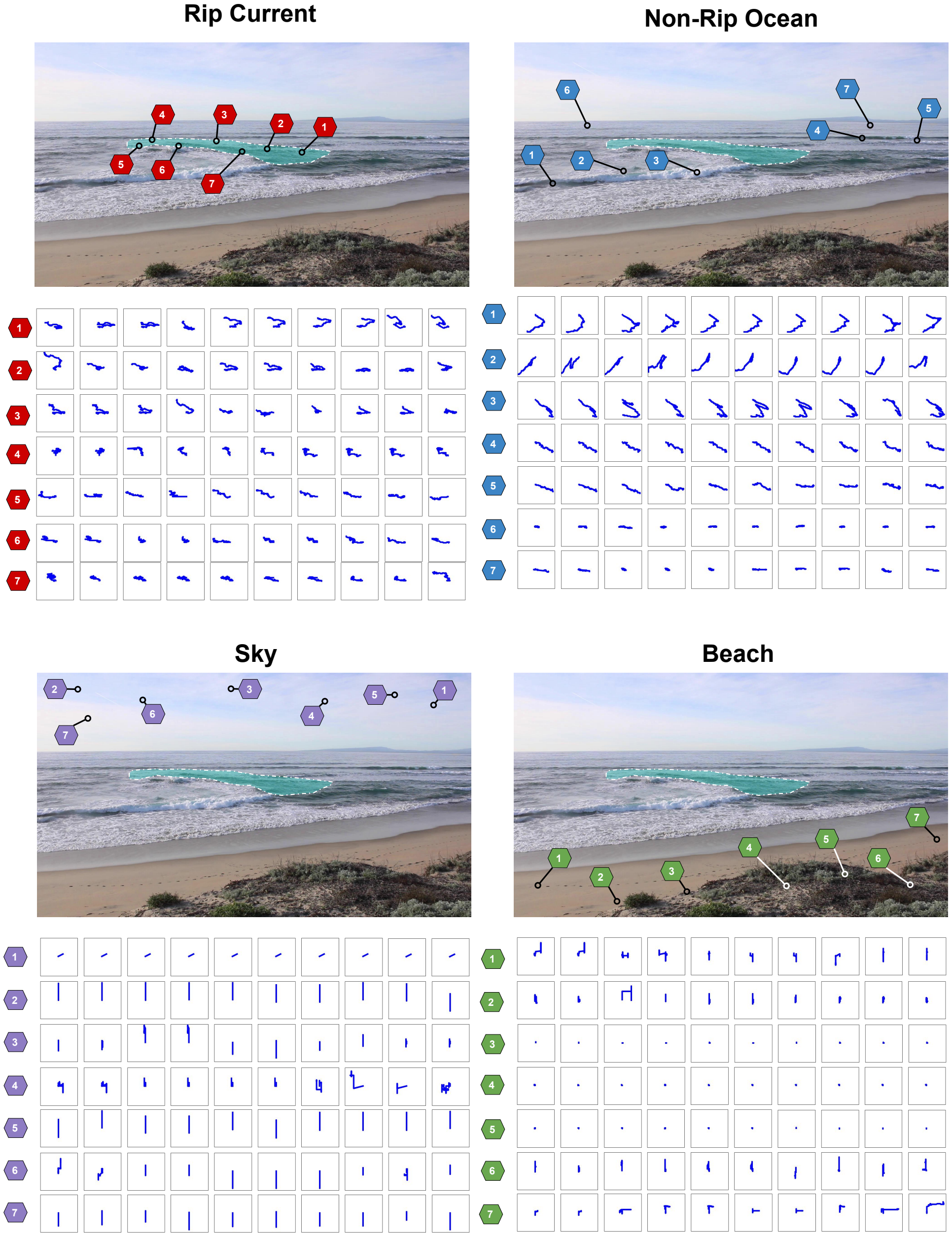


Figure 4.16: **Visualization Pathline Sequences:** We show a sequence of 10 pathlines per seed point. Seed points are numbered and the location is shown in the figure. All pathlines start from the center of the binary image. Notice how the pathline sequences seeded inside the rip current are distinct from pathline sequences generated from other parts (sky, beach and non-rip ocean) of the flow field. Also, note that the individual pathlines may look similar, but the *sequences* are distinct.

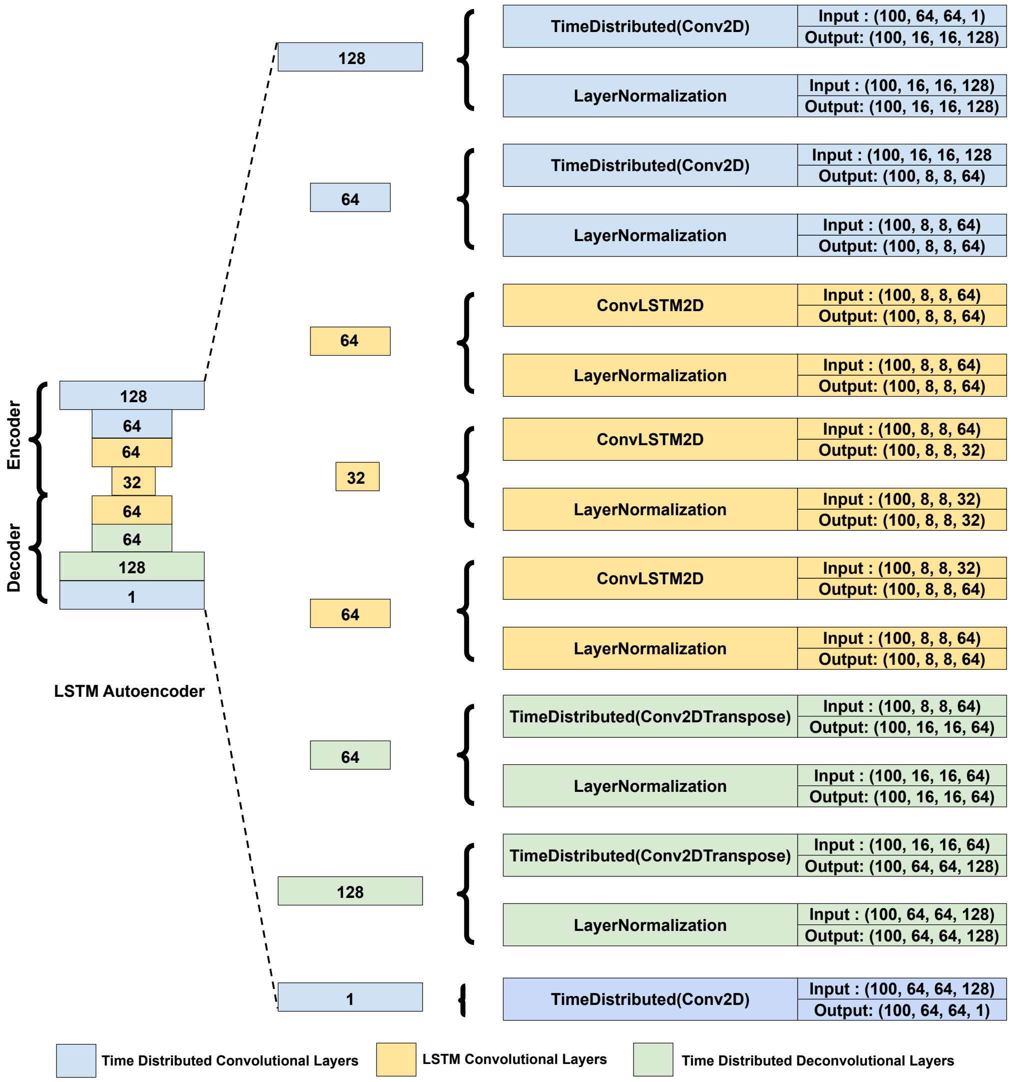


Figure 4.17: **Expanded view of the layers in the LSTM autoencoder:** In this figure we show a more detailed expanded view of the inner layers of the LSTM autoencoder. For each layer, we show the size of the input and output for one pathline sequence.

features, making it challenging for the expert to identify the entire extent of the rip current.

In contrast to appearance-based identification, RipViz uses the behavior of rip currents, not appearance, to detect rip currents. Experts can use RipViz as a visualization tool to determine the entire boundary of the rip current when visual features are lacking. In Figure 4.18, we demonstrate a use case where the rip current expert's original boundary estimate was updated after examining the visualization provided by RipViz. The updated boundaries now include feeder currents near shore and what appears to be a circulating pattern farther offshore in Discovery 1 ,3, 4, 5, 6, 7, 9, and 10, as well as a weaker neighboring rip that merged with the dominant rip in Discovery 2 and Discovery 8.

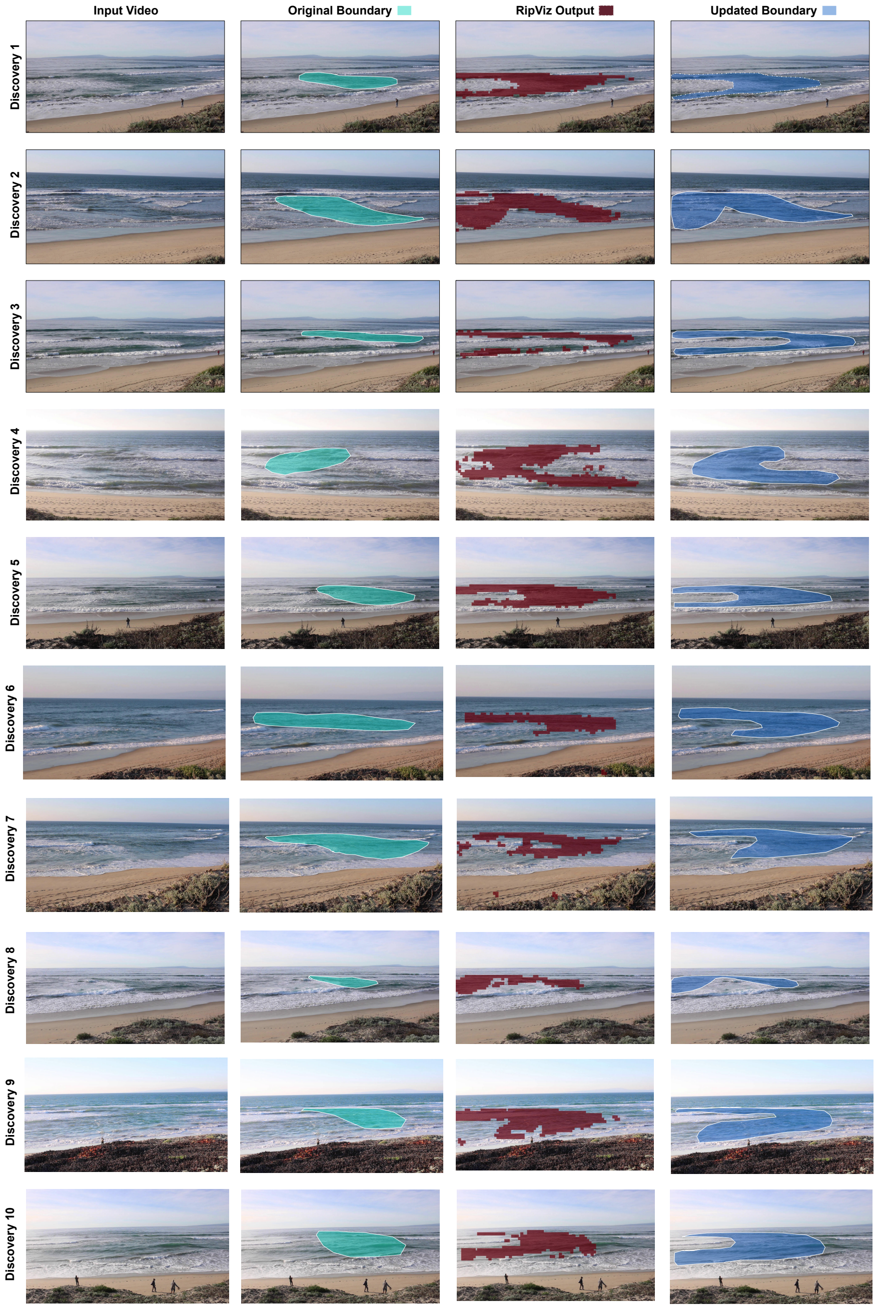


Figure 4.18: **Discoveries made by the domain expert:** Experts estimated the rip current boundary (*Original Boundary* Column) by observing only the input video (*Input Video* Column). After examining the output of RipViz (*RipViz Output* column), the experts updated their original rip current boundary estimate to include parts of the rip current that are not readily observable in the input video due to a lack of distinct visual features (*Updated Boundary* column).

# Chapter 5

# Finding Vortex Boundary by Learning from Streamlines.

## 5.1 Introduction

Vortices are extensively studied in numerous scientific disciplines to gain insight into the behavior of fluid flows. In aerodynamics, researchers focus on studying vortices that form in the wake of an aircraft, aiming to mitigate the creation of vortices with long lifetimes; persistent vortices can potentially impede commercial aviation's operational capacity [151, 152, 153]. Oceanographers, on the other hand, study mesoscale eddies modeled as vortices, to understand the transportation of nutrients and heat in ocean currents [154, 155, 156]. Additionally, astrophysicists examine the vortical structure in black holes to understand the stability of extremal black holes via topological explanations [157, 158, 159]. Vortices are also studied in high temperature superconductors, to better understand the dissipation of free-current [160, 161, 162]. While the study of vortices transcends numerous scientific disciplines, visualization researchers place particular emphasis on visualizing the vortex boundary to gain insight into fluid flow behavior.

While there are many definitions of a vortex, it is generally agreed that "vortices are regions of high vorticity" with "multitude of material particles rotating around a common center" [163]. Analyzing vortex boundaries is essential for gaining insights into fluid flow behavior, including phenomena such as flow separation, turbulence, and vortex formation and dissipation [164]. Observing a distorted or irregular shape along the vortex boundary can provide valuable clues about the potential occurrence of vortex breakdown [131]. While vortex boundary extraction has been studied by visualization researchers for many years [165], its precise mathematical definition may vary depending on the specific context [166]. In such cases where a formal definition is elusive, deep learning techniques hold great promise in identifying and capturing relevant features.

Supervised deep learning based methods, which rely on labeled datasets, have been used to extract the vortex boundary [131, 167, 168, 169, 170]. These approaches primarily rely on velocity fields represented by their $U$ and $V$ components (velocity com-

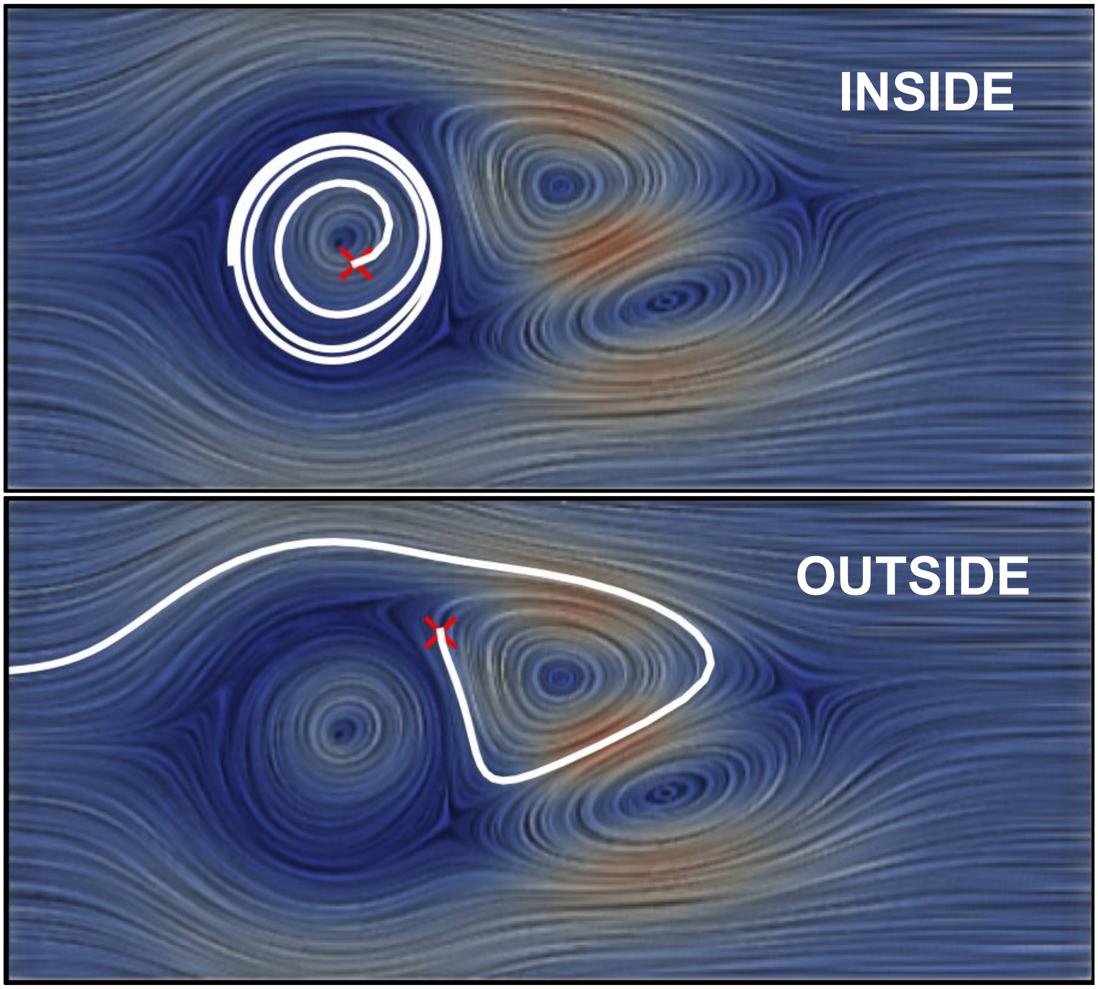


Figure 5.1: **Flowlines (pathlines and streamlines) Behave Differently Inside and Outside of a Vortex:** In this paper, we exploit this behavior difference and physical properties of the flow field along the flowline to find vortex boundaries. (Two streamlines from [2] are shown with a red cross indicating the seed point.)

ponents along x and y axes, respectively) to learn about the vortex boundary. However, we contend that learning solely from the velocity components is insufficient in accurately capturing the vortex boundary. This limitation arises from the fact that the velocity field, represented as *u* and *v* components fails to effectively capture the non-local behavior of the flow field. To address this issue, we propose an alternative approach in this paper, where we utilize flowlines (streamlines or pathlines) instead of velocity fields to learn about the vortex boundary. By incorporating flowlines into the learning process, we aim to enhance the model's ability to capture the rotational behavior or the *swirliness* of the flow field, thereby improving the accuracy of vortex boundary extraction.

The main contribution of this chapter is,

- A novel deep learning methodology utilizing flowlines to learn and identify vortex boundaries.

## 5.2 Related Work

### 5.2.1 Deep Learning for Flow Visualization

The visualization community has been actively engaged with deep learning in two primary ways: visualization to understand the inner workings of a deep learning models, and use of deep learning in visualization tasks [171, 172, 173]. In the context of the latter, particularly for flow visualization tasks, specific flow features for visualization have been identified with the aid of deep learning. Some researchers utilized deep learning to find rip currents, a flow pattern found in the near shore ocean [119, 123, 174]. Kim and Günther [132] used a neural network to extract a steady reference frame from an unsteady vector field. In [175], deep neural network based particle tracing method to explore time-varying vector fields represented by Lagrangian flow maps. Numerous studies have also leveraged deep learning to identify vortex boundaries from velocity fields represented as velocity components [131, 167, 168, 169, 170]. In this paper, we present an novel deep learning approach that learns to identify vortex boundaries by utilizing information from flowlines (streamlines and pathlines), as opposed to relying solely on velocity components.

### 5.2.2 Threshold-based Methods for Vortex Boundary Detection

Threshold-based vortex detection methods can be categorized into two types: local and global methods. Local methods involve the computation of a local quantity at each point within a flow field, resulting in a scalar field. Subsequently, these resulting scalar fields undergo thresholding to identify the contours corresponding to vortices. Most notable local methods are $Q$ criterion [176], $\Omega$ criterion [177], $\lambda_2$ criterion [178], and $\Delta$ criterion [179]. However, these methods may fail to detect obvious vortices while erroneously detecting non-vortical structures. Sadarjoen et al. [180] argue that vortices are a regional phenomenon, and local methods such as the above criteria are ill-equipped to identify them.

In contrast, global methods such as instantaneous vorticity deviation (IVD) [181] and the winding angle method [180] use global flow information to find vortex boundaries. IVD is defined as the absolute value of the difference between the vorticity at a point in the flow field and the spatially averaged vorticity of the global flow field. There are two main post-processing methods that visualization researchers use to find vortex boundaries using IVD. The first method involves applying a user-defined threshold to the IVD field to detect vortex contours [131]. However, in certain scenarios, such as when vortices are dissipating, varying thresholds might need to be set for each individual vortex. On the other hand, other researchers [168, 167, 169] adopt a different approach by identifying isocontours around vortex cores. Vortex cores are located by identifying local maxima in the IVD field; isocontours satisfying pre-defined arc length and convexity criteria are selected as the vortex boundaries. For this method, the thresholds for local maxima, convexity, and arc length are set by users during runtime. However, the substantial user input required at runtime for these IVD-based approaches makes them less appealing for analyzing large datasets.

The winding angle method, introduced in [180], is centered around the identification of streamlines that exhibit rotational behavior around a critical point. Researchers initiate this approach by sparsely seeding streamlines across the flow field. For each streamline, they calculate the sum of signed angles between adjacent line segments. By applying a threshold to this computed sum, streamlines associated with a vortex can be discerned and visually presented to denote vortex locations. It is essential to emphasize that the determination of the threshold value is left to the user and relies on the specific dataset under analysis. Furthermore, it is worth noting that the winding angle method is designed to reveal the locations of vortices by showcasing the streamlines, rather than explicitly outlining the vortex boundary.

### 5.2.3 Deep Learning for Vortex Boundary Extraction

In recent years, supervised machine learning methods, dependent on labeled datasets, have been increasingly applied to detect vortex boundaries. Originally developed within the computer vision community to identify specific pixel patterns within images, these methods have been adapted for finding vortices within flowfields. One notable network architecture frequently employed is U-net [182], originally designed for medical image segmentation but adapted to extract vortex boundaries through segmentation of the flow field [131, 168]. Likewise, various convolutional neural network (CNN) variants have also been tailored to identify vortices in flowfields [167, 169, 131]. Furthermore, ResNet has been repurposed to detect vortices within flowfields as well [131].

### 5.2.4 Deep Learning Methods with Flowlines

The flow visualization community has proposed deep learning methods that learn from flowlines. Flowlines describe the trajectory of a massless particle of fluid. In steady state flows (or a snapshot of a time varying flow) they're referred to as streamlines and otherwise referred to as pathlines. Han et al. [133] used an autoencoder-based deep learning model, FlowNet, to learn feature representations of streamlines in 3D steady flow fields which are then used to cluster the streamlines. Recently, de Silva et al. [174], proposed a deep learning LSTM autoencoder methods, that learns about rip currents, a naturally occurring flow pattern in the near shore ocean, by using short sequences of pathlines. In both these works flowlines were encapsulated in binary volumes (for 3D flowlines) or binary images (for 2D flowlines). In this work we use flowlines to find vortex boundaries.

## 5.3 Method

VortexViz identifies points that are within a vortex boundary by combining flowline patterns and information collected along the flowlines. This is encapsulated in the pipeline shown in Figure 5.2. It consists of several steps. Section 5.3.1 discusses how we generate flowlines. Section 5.3.2 discusses how we represent flowlines. Section 5.3.3 discusses our deep learning model.

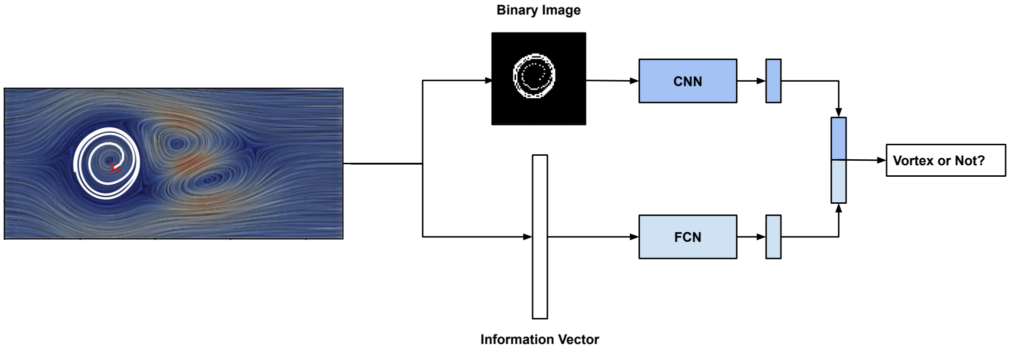


Figure 5.2: **VortexViz Pipeline**: Each flowline is represented as a binary image and an information vector. The binary image is processed by a convolutional Neural Network (CNN) and the information vector is processed by a Fully Connected Neural Network (FCN). The intermediate output layers of both these networks are merged. The merged layer is then used to predict if the seed point that originated the flowline is classified as inside a vortex or not.

### 5.3.1 Flowline Generation

We generate flowlines from higher-order methods like the widely used fourth-order Runge-Kutta integrator $RK4$ [183]. If a flowline travels beyond the domain of the flow field then we stop the numerical integration.

### 5.3.2 Flowline Representation

Each flowline is a collection of ordered points as shown in equation 5.1 where $(x_i, y_i)$ is a point in the coordinate system of the flow field and $n$ is the maximum number of integration steps of the flowline, while $(x_1, y_1)$ is the seed point.

$$flowline = \Big\{(x_1, y_1), \cdots, (x_n, y_n)\Big\} \tag{5.1}$$

In order for the deep learning model to learn about flowlines, we need to encode

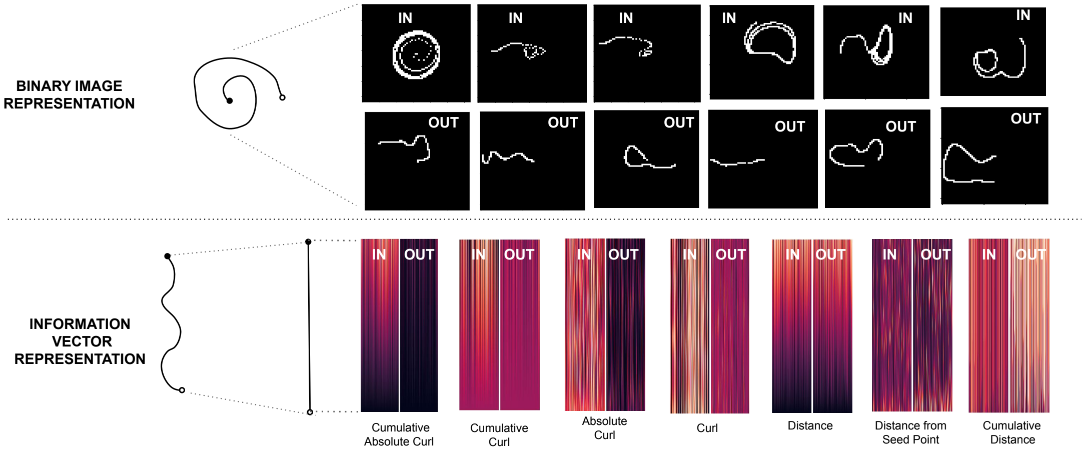


Figure 5.3: **Visualizing Binary Images and Information Vectors:** The top half of the figure shows binary image representations of flowlines. Notice that binary images exhibit notable disparities for flowlines inside (IN) and outside (OUT) of a vortex. The lower half showcases information vector representations of flowlines. These information vectors, presented as stacked values in heatmap form, illustrate differences among vectors of equal length. The information vectors initiate from the bottom of the heatmap, aligned with the hollow circle indicating the seed point of the example flowline. Notice that each type of information vector is also different for flowlines inside (IN) and outside (OUT) the vortex. These visual differences lead us to believe these representations can be used to detect vortices.

the *swirliness* of flowlines. To do this, we represent each flowline as a binary image, as discussed in Section 5.3.2.1, and as an information vector, as discussed in Section 5.3.2.2.

#### 5.3.2.1 Binary Image

In the binary image, the seed point of each flowline is placed at the center, and the trajectory rescaled to this flowline-centric coordinate system. To do this, each flowline is represented by an $L \times L$ binary image $\mathbf{I}$, where each point in equation 5.1 is translated to center the seed point in the binary image. For longer flowlines that extend beyond

$L \times L$, we increased the size of the binary image to accommodate these longer flowlines. We then resized all of these larger binary images down to $L \times L$. The reason why all flowlines were not simply centered then resized to $L \times L$ is that we need to differentiate flowlines that barely moved from their initial seed position versus those that actually travelled beyond $L \times L$. Each point of the flowline is marked as 1 in the binary image, while others are marked as 0. Some examples of binary images are shown in Figure 5.3.

#### 5.3.2.2 Information Vector

In addition to using binary images, we hypothesize that physical quantities of the flow field can also be used to capture the *swirliness* of the flow field. In particular we observed that information such as curl and distance can be used to differentiate flowlines originating within a vortex from others. We denote these vectors as *information vectors* and stored as a $1D$ vector of length $n$, where $n$ is the maximum number of integration steps of the flowline.

We conducted experiments using four different information vectors derived from curl, which gauges the rotational tendency of particles at specific points of the flow field. These are *curl*, *absolute curl* and *cumulative curl* and *cumulative absolute curl* respectively. The *curl* information vector consists of the curl calculated at each point along the flowline, while *absolute curl* contains the absolute value of the curl at each point of the flowline. Additionally, *cumulative curl* contains the cumulative value of the curl at each point of the flowline and *cumulative absolute curl* contains cumulative value of the absolute curl at each point of the flowline. Mathematical expressions for information vectors are shown in the supplementary materials.

In addition to information vectors based on curl, we also explored the use of information vectors based on distance. We noticed that the flowlines seeded in vortices stay a longer time within the domain while the other flowlines exit the domain relatively quickly. Based on this observation, we encoded two types of information vectors, namely *distance*, where distance between consecutive points of the flowline is used, and *cumulative distance*, cumulative distance travelled upto a point of the flowline is used. Additionally we observed that the due to the swirliness of flowlines seeded in vortices, that the Euclidean distance between the seed point and any other point in the flowline was subjected to a maximum value. Based on this observation, we used *distance from seed point* as another information vector. All mathematical expressions for information vectors are shown in the supplementary materials.

In order to visualize how these information vectors are different for flowlines seeded in and out of vortices, we visualize the values of stacks of information vectors of the same length as a heatmap as shown in Figure 5.3. Notice that these information vectors inside and outside vortices are visually different, leading us to believe they maybe useful in detecting vortices.

### 5.3.3 Deep Learning Model

The proposed deep learning model learns about each flowline by using two modalities: binary image and information vector as shown in Figure 5.2. One branch of the deep learning model learns from the binary image and the other branch learns from the information vector. The binary image is processed through a convolutional neural network (CNN). While the information vector is processed through a fully connected neural network (FCN). The CNNs are used to generate features from $2D$ data, while FCNs are used to generate features from $1D$ data. Then the resulting feature vectors are combined. Finally for each seed point we make the decision if that seed point is in a vortex or not. The loss function used in the method is binary cross entropy. We trained the neural network model using TensorFlow Keras API [143]. The code and the trained models will be provided in the supplementary materials once the paper is accepted for publication.

### 5.3.4 Data Sets

We used five unsteady *2D* datasets, namely *2D Unsteady DoubleGyre* [184], *2D Unsteady CylinderFlow* [2], *2D Unsteady Cylinder Flow with von Karman Vortex Street* [185, 186], *2D Unsteady Beads Problem* [187, 188], and *2D Unsteady Cylinder Flow Around Corners* [189, 190]. Training data was generated by partitioning *2D Unsteady DoubleGyre*, *2D Unsteady CylinderFlow* and *2D Unsteady Cylinder Flow with von*

*Karman Vortex Street* into training and test datasets. *2D Unsteady Beads Problem*, and *2D Unsteady Cylinder Flow Around Corners* were exclusively used for testing.

While acknowledging the absence of an absolute “true” definition for a vortex boundary, numerous prior studies [168, 131, 169] have commonly utilized the results obtained from IVD (Instantaneous Vorticity Deviation) as a benchmark for comparison. Following the same convention, our study also relied on IVD to establish ground truth for *2D* datasets, particularly in cases where the IVD output distinctly delineated contours representing vortex boundaries. Notably, within datasets such as *2D Unsteady DoubleGyre*, *2D Unsteady CylinderFlow*, *2D Unsteady Cylinder Flow with von Karman Vortex Street*, we identified unambiguous contours through IVD.

However, for the remaining datasets, namely *2D Unsteady Beads Problem* and *2D Unsteady Cylinder Flow Around Corners*, we observed that the output derived from IVD did not align with the expected vortex structure as one can perceive from the LIC images. Consequently, we lacked a ground truth for these particular datasets. We will provide all data along with exact partition information in the supplementary materials [once the paper is accepted for publication].

### 5.3.5 Comparison Metric

We use $F_1$ *score* as the comparison matric as shown in equation 5.2. If a flowline seeded within the vortex boundary is detected as a vortex, then it is counted as a true positive

(TPs). Otherwise, it is considered a false positive (FPs). Suppose a flowline originating within the vortex boundary is not detected, then it is a false negative (FNs). The range of the $F_1$ score is between 0 and 1. If most flowlines within the vortex boundary are detected as a vortex, the score will be closer to 1, otherwise closer to 0.

$$F_1 = \frac{2}{\frac{1}{recall} + \frac{1}{precision}} = \frac{2}{\frac{FNs+TPs}{TPs} + \frac{FPs+TPs}{TPs}} \tag{5.2}$$

## 5.4 Results and Discussion

### 5.4.1 Comparison with Existing Methods

We compare our method against threshold-based methods and deep learning methods that learn from velocity components. Rows 2-5 of Figure 5.4 shows the results of threshold -based methods and Rows 6-9 of Figure 5.4 shows the output of the deep learning methods on 5 different datasets. Row 10 shows the output of our method and last row shows the ground truth.

#### 5.4.1.1 Comparison with threshold-based methods

First, we compared our method to threshold-based methods such as *IVD* [181], *Q criterion* [176], $\Omega$ *criterion* [177] and $\Delta$ *criterion* [179]. In particular for datasets *2D Unsteady DoubleGyre*, *2D Unsteady CylinderFlow* and *2D Unsteady Cylinder Flow*

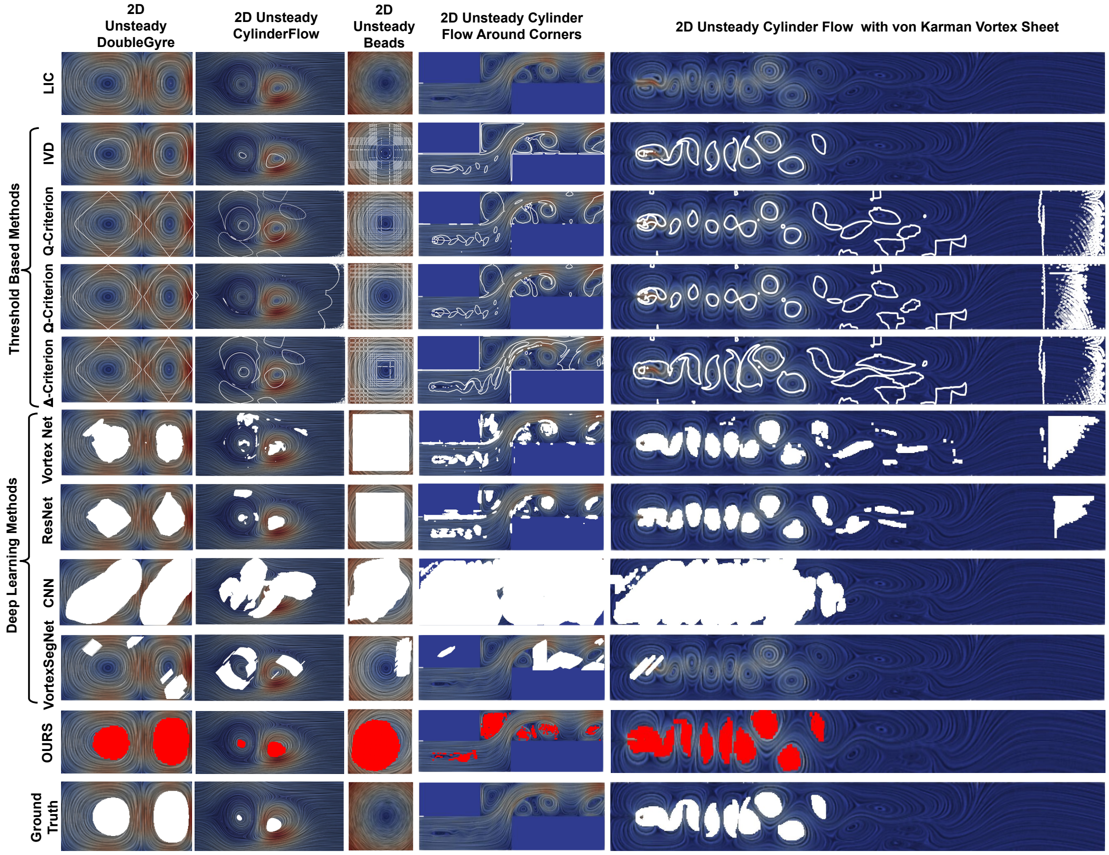


Figure 5.4: **Qualitative Comparison with Other Methods.** The first row displays the LIC image. Rows 2-5 exhibit the output of IVD, Q, Ω, and Δ criteria. Rows 6-9 showcase the output of deep learning methods that learn from velocity components. The second-to-last row presents the output of our method, while the last row depicts the ground truth. Notice that our method can visualize vortices even where other methods fail.

Table 5.1: **Quantitative Comparison with Other Methods.** We use three datasets where IVD can produce an unambiguous ground truth for quantitative comparisons in this table. Notice that our method has a higher $F_1$ score compared to other methods. For the remaining two datasets IVD did not produce reliable ground truth contours that represent vortices. Therefore we did not include those in this table; however for a qualitative comparison on all datasets please refer to Figure 5.4.

| Method | 2D Unsteady DoubleGyre [184] | 2D Unsteady CylinderFlow [2] | 2D Unsteady Cylinder Flow with von Karman Vortex Street [185, 186] |
|---|---|---|---|
| Q criterion [176] | 0.698 | 0.107 | 0.311 |
| Ω criterion [177] | 0.707 | 0.124 | 0.363 |
| Δ criterion [179] | 0.657 | 0.029 | 0.445 |
| Vortex Net [167] | 0.818 | 0.332 | 0.635 |
| ResNet [131] | 0.831 | 0.550 | 0.646 |
| CNN [131] | 0.567 | 0.090 | 0.360 |
| Vortex Seg Net [169] | 0.173 | 0.008 | 0.196 |
| Ours | **0.972** | **0.797** | **0.946** |

*with von Karman Vortex Street* we were able to use *IVD* at dataset specific thresholds that capture the locations of rotational behavior seen in the *LIC* images.. However, for *Q,*Ω*,*Δ *criteria*, there was no single threshold that captured the rotational patterns without introducing erroneous contours as well.

Moreover, we found that threshold-based methods could not find contours that represent the vortex boundary for some datasets such as *2D Unsteady Beads Problem.* In this particular dataset, the U component (velocity component along the x-axis) is constant across each row; therefore, the partial derivative of the U components along the y-axis, $\partial u/\partial y$, is also constant for each row while $\partial u/\partial x$ is 0. Likewise, the V component (velocity component along the y-axis) is constant across each column, and therefore the partial derivative of the V component along the x-axis, $\partial v/\partial x$, is also constant for each column, while $\partial v/\partial y$ is 0. Therefore, mathematical expressions that rely on partial derivatives of velocity components, such as curl, will also produce constant values along rows and columns. Consequently, threshold-based methods, which are derived from curl, produce linear (horizontal and vertical) features around vortex cores instead of the usual circular contours. In comparison, since our method employs flowlines, we can find the vortex.

Additionally, we found that for *2D Unsteady Cylinder Flow Around Corners*, IVD and Δ criterion would highlight the boundary of the obstacle corners as vortices. In contrast, our method does not highlight the corners as a vortex. Visual comparison of the output for each threshold based method is shown in Figure 5.4 and numerical comparison is given in Table 5.1.

#### 5.4.1.2 Comparison with deep learning based methods

Next, we compared our method to existing deep learning approaches that use velocity components such as *VortexNet* [167], *ResNet* [131], *CNN* [131] and *VortexSegNet* [169]. In general, we found that these methods would predict vortices even in places of the flow field where there are no vortices present, as shown in Figure 5.4. These models learn about vortex formations primarily from velocity fields represented by their velocity components U and V. We hypothesize that learning from velocity components is ineffective in learning the vortex boundary. In velocity components, the defining features—regions of distinctive values—are often distributed away from the vortex core. Understanding the interpretation of vortices by existing deep learning models trained on velocity components is crucial in comprehending the limitations of these methods.

Since most existing deep learning techniques use convolutional layers, we can employ explainable methodologies like Gradient-weighted Class Activation Mapping (Grad-cam) [191] to gain insights into what convolutional layers perceive as vortices. Grad-cam identifies parts of the input matrix that most impact the decision of the neural network. In Figure 5.5, we showcase the application of Grad-cam, initially designed for image analysis, to the context of velocity fields. The top row demonstrates how

CNNs trained to identify object classes like *goldfish* and *bear* from images learn distinctive features. These features are highlighted by a heatmap, where warmer colors indicating places in the image that contribute more to the decision of the convolutional neural network. Notice that these features correspond directly to the appearances of the goldfish and bear in the images. However, in the second row of Figure 5.5, we show one counterclockwise votex and one clockwise vortex and what a CNN perceives as a vortex using Grad-cam. Notice that the warmer parts of the heatmap is not aligned with the vortex core. The convolutional operation in the neural network, originally designed to detect edges in images, may not effectively capture the swirling nature of vortices when applied to flow fields. It tends to focus on the prevalent values of the individual components of the flow field rather than capturing the swirliness of vortices. In contrast, our approach differs from this since the model is not learning from the velocity components directly. Instead our model is learning from the pattern and information collected along flowlines. This representation captures the swirliness found in vortices better than velocity components.

### 5.4.2 Performance Assessment on Noisy Data

In our evaluation, we tested the robustness of our method by subjecting the input data to different noise levels, specifically at rates of 1%, 5%, and 10%, by introducing Gaussian noise into the data sets. The primary aim was to gauge and compare the robustness of

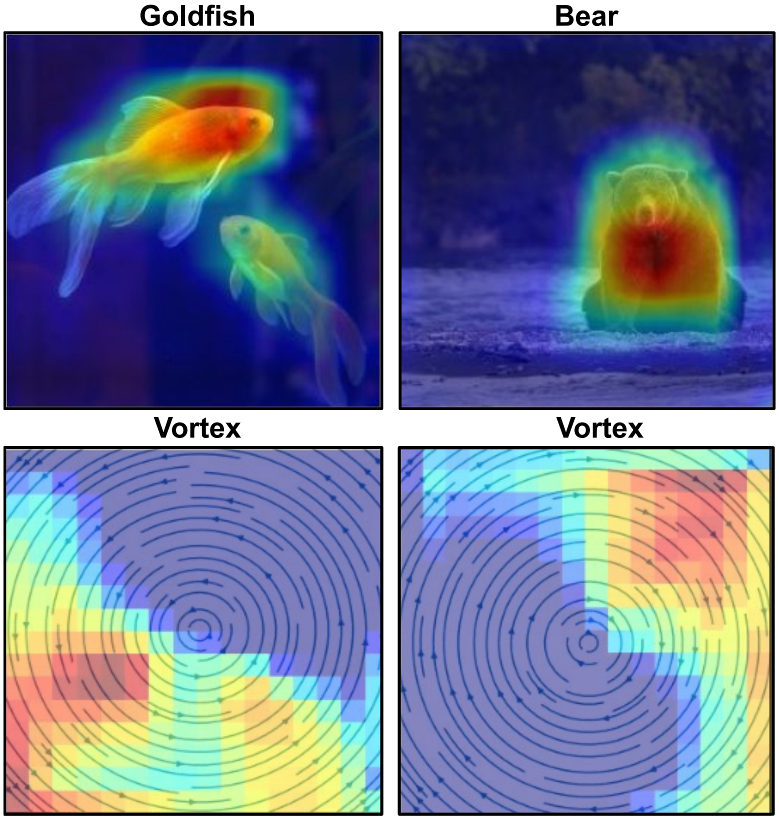


Figure 5.5: **Understanding what Deep Learning Methods Learn from Velocity Components using Grad-cam:** In the first row, we display images of a goldfish and a bear. The warmer sections within the overlaid heatmap depict the specific features learned by a CNN. Notably, these features correspond directly to the appearances of the goldfish and bear in the images. In the second row, we illustrate a counterclockwise and a clockwise vortex. The warmer regions within the overlaid heatmap reveal the features interpreted by a CNN as indicative of a vortex. It's worth observing that these highlighted features do not align with the vortex core. This observation leads us to hypothesize that what the CNN learns does not necessarily relate to the vortex itself.

our method with other methods across varying noise levels.

The outcomes, as depicted in Figure 5.7, on the double gyre dataset, highlight the robustness of our method. Even amidst noisy conditions, our approach preserved the circular shape of the vortex. Notably, our observations revealed the sensitivity of threshold-based techniques, which produced deteriorating vortex shapes under a mere 1% of noise. Conversely, deep learning methods demonstrated robustness, maintaining the vortex's shape with minimal degradation at 1% noise. However, their $F_1$ scores notably declined when noise level is above 5%, marking a rapid deterioration in predicting a circular vortex shape as shown in Figure 5.6.

We extended our evaluation to a real-world scenario by utilizing a dataset reconstructed from dense optical flow extracted from a sequence of satellite imagery, specifically selecting the satellite video capturing Hurricane Dorian [192]. Within this context, noise sources included video compression artifacts and errors in deriving the velocity field using optical flow.

None of the methods, including ours, was optimized for a hurricane dataset. However, we discovered that our method can visualize the general area of the hurricane vortex, further underscoring its robustness under noisy conditions, as shown in Figure 5.8

While the performance of our method on this dataset was surprising, we recognize the need for further optimization to enhance our method's precision in detecting the

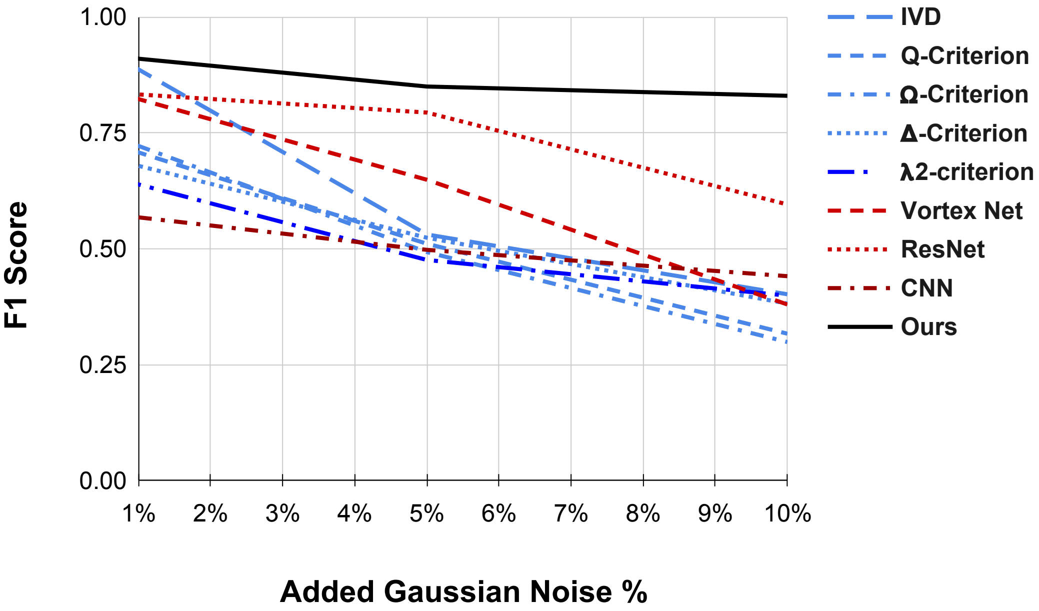


Figure 5.6: **Quantitative Comparison of Performance on Noisy Data:** When introducing gaussian noise, we expect a decline in performance across all methods. However, notice that our method (shown in black) exhibits the least decrease in $F_1$ score compared to velocity component based deep learning methods (shown in red) and threshold-based methods (shown in blue).

hurricane vortex. This outcome serves as a promising starting point, prompting us to refine our approach for more accurate extraction of vortex boundaries in real-world scenarios.

## 5.4.3 Sensitivity Analysis of VortexViz Components

### 5.4.3.1 Pairing of Information Vectors with Binary Images

We conducted experiments to determine the optimal pairing of information vectors and binary images, outlined in Table 5.2. It became evident that relying solely on the shape

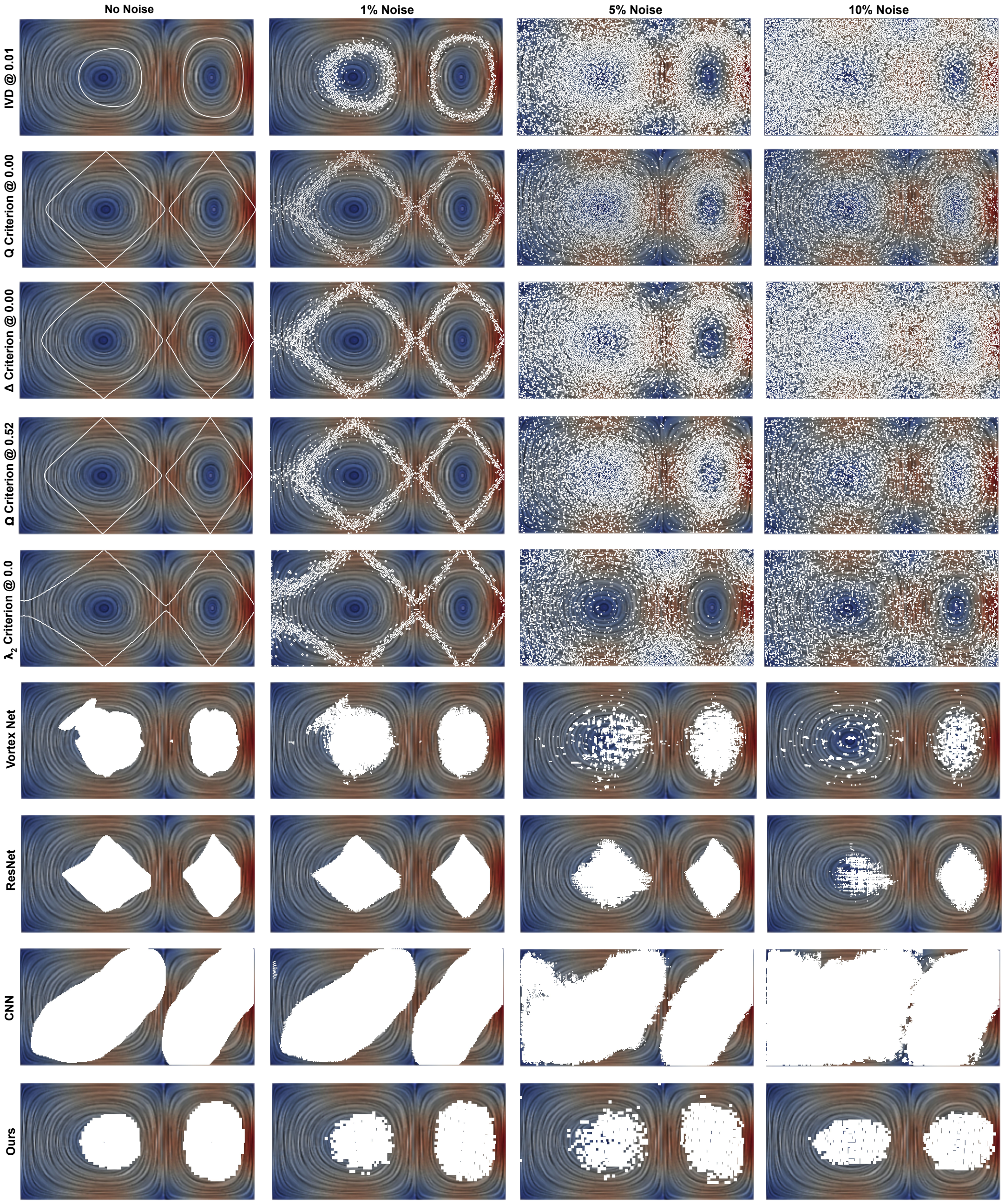


Figure 5.7: **Robust with Noisy Data:** In Rows 1-5, the contours based on IVD, Q, Δ, Ω and $\lambda_2$ criteria are displayed. It is noticeable that as noise levels rise, the discernibility of vortices within these contours diminishes. Similarly, ResNet, VortexNet and CNN (Rows 6-8, respectively) exhibit reduced capability in detecting vortices as noise levels increase. In contrast, our method, depicted in the last row, can maintain the shape of the vortex even amid increased noise levels.

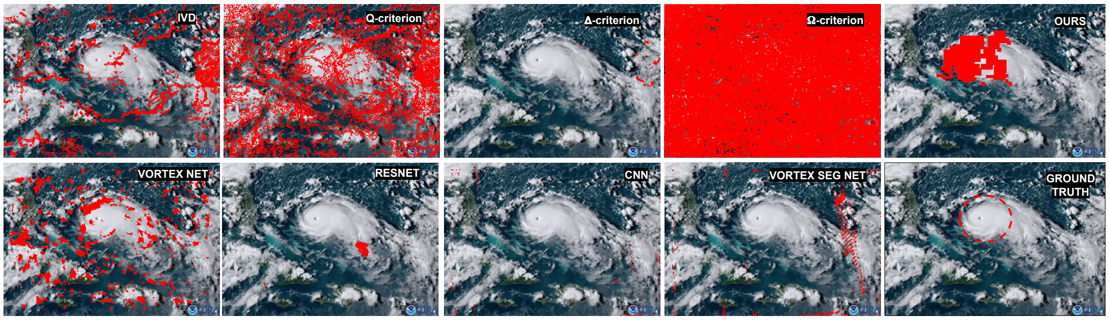


Figure 5.8: **Performance Showcase on Real-World Data (Hurricane Dorian, 2019)** Despite inherent noise in the reconstructed velocity field derived from video via optical flow, our method stands out in indicating the approximate location of the vortex amidst the noise. While all methods struggle to identify the vortical structure of the hurricane, our approach excels in delineating the general region of the vortex, even under noisy conditions.

of flowlines encapsulated within binary images did not yield satisfactory outcomes. While using flowline shape within binary volumes [133] and binary images [174] has proven successful in various flow visualization tasks, for precise vortex boundary detection, depending solely on binary images, proved inadequate.

Moreover, our experiments involved pairing the binary image with different information vectors. The initial set of information vectors is derived from flowline point distances. Surprisingly, all distance-based information vectors—namely, *Distance*, *Cumulative Distance*, and *Distance from seed point*—both in isolation and when paired with a binary image, failed to perform as well as other parings.

The second set of information vectors were derived from curl such as *Curl*, *Absolute Curl*, *Cumulative Curl* and *Cumulative Absolute Curl*. We noticed that these information vectors had superior performance when used on their own and when paired with a binary image. We attribute this performance superiority to the fact that curl along the flowline can capture the rotational behavior of a vortex. Remarkably, apart from *Curl* alone, these vectors maintained robustness even under noisy conditions. Notably, the combination of the binary image with *Cumulative Absolute Curl* emerged as the most effective under these circumstances. This paring was used for the results presented in this paper. Our approach quantifies the general consensus regarding vortices as *concentrated regions of high vorticity* [163].

#### 5.4.3.2 Choosing Between Pathlines and Streamlines

We conducted experiments to find the optimal type of flowline for our method. Flowlines track the trajectory of a massless fluid particle, termed as streamlines in steady-state flows or in snapshots of time-varying flows, and as pathlines in other cases. In our investigation detailed in Table 5.3, we explored both pathlines and streamlines with our approach. While streamlines performed consistently well, the effectiveness of pathlines varied depending on the dataset.

We found both streamlines and pathlines effective when the vortex cores remain fairly stationary such as in the *2D Unsteady DoubleGyre* , *2D Unsteady CylinderFlow* or *2D Unsteady Beads Problem*. However, in instances where vortices exhibit translational movement over time, as observed in the *2D Unsteady Cylinder Flow with von Karman Vortex Street* or *2D Unsteady Cylinder Flow Around Corners*, streamlines exclusively demonstrated effectiveness, whereas pathlines did not yield favorable results.

Table 5.2: **Sensitivity Analysis of Combinations of Information Vectors and Binary Images.** Notice that *binary image + cumulative absolute curl* has the hightest $F_1$ score especially in noisy data.

| Method | 0% noise | 10% noise |
|---|---|---|
| Binary Image *Only* | 0.814 | 0.726 |
| Distance *Only* | 0.034 | 0.004 |
| Cumulative Distance *Only* | 0.063 | 0.004 |
| Distance from Seed Point *Only* | 0.660 | 0.263 |
| Curl *Only* | 0.947 | 0.254 |
| Absolute Curl *Only* | 0.952 | 0.737 |
| Cumulative Curl *Only* | 0.961 | 0.721 |
| Cumulative Absolute Curl *Only* | **0.976** | 0.749 |
| Binary Image + Distance | 0.801 | 0.420 |
| Binary Image + Distance from seed point | 0.822 | 0.510 |
| Binary Image + Cumulative Distance | 0.799 | 0.718 |
| Binary Image + Curl | 0.962 | 0.686 |
| Binary Image + Absolute Curl | 0.954 | 0.711 |
| Binary Image + Cumulative Curl | 0.969 | 0.731 |
| Binary Image + Cumulative Absolute Curl | **0.972** | **0.781** |

Table 5.3: **Pathlines or Streamlines:** While streamlines performed consistently higher $F_1$ score, the effectiveness of pathlines varied depending on the dataset.

| Dataset | Streamlines | Pathlines |
|---|---|---|
| 2D Unsteady DoubleGyre [184] | **0.972** | **0.972** |
| 2D Unsteady CylinderFlow [2] | **0.797** | **0.795** |
| 2D Unsteady Cylinder Flow with von Karman Vortex Street [185, 186] | **0.946** | 0.391 |

Our findings suggest that the suitability of flowline type depends upon the dynamic behavior of vortices, with streamlines exhibiting more consistent performance across various scenarios compared to pathlines.

#### 5.4.3.3 Optimal Flowline Length and Binary Image Size

Our experiments focused on determining our method's ideal flowline length and binary image size. Figure 5.9 illustrates a notable trend: longer flowlines showed a decline in accuracy. We attribute this decrease to flowlines exiting the domain before completing integration over an extended interval. Additionally, our observations indicated that increasing binary image sizes did not notably improve the $F_1$ score. For the results presented in this work, we used binary images of size $16 \times 16$ and a flowline of length 200.

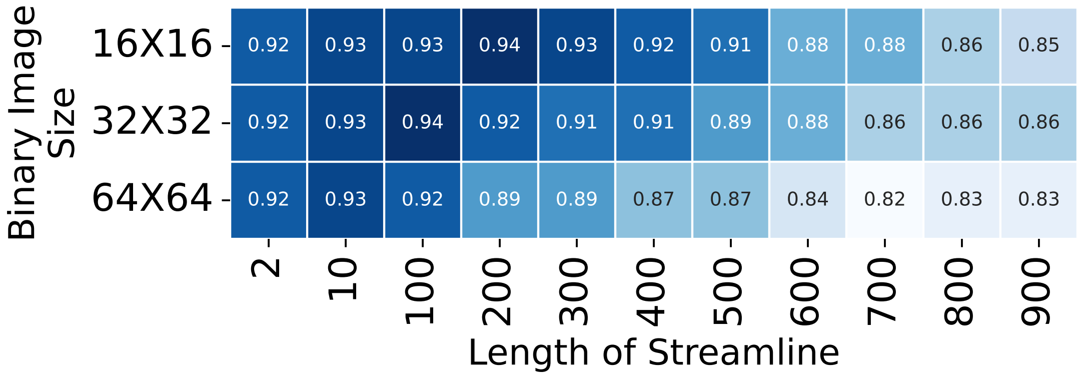


Figure 5.9: **Exploring the Impact of Varying Flowline Lengths and Binary Image Sizes on Binary Image + Cumulative Absolute Curl Pairing**. Notice that longer flowlines tend to perform poorly and scaling up binary image sizes did not significantly enhance the $F_1$ score.

#### 5.4.3.4 Comparison of Numerical Integration Methods for Flowline Generation

Traditionally, the flow visualization community favors higher order numerical integration methods for their superior accuracy [193]. However, in our experiments, we examined both higher order (specifically, fourth order Runge Kutta) and lower order (such as first order Euler) integration methods to explore their impact on deep learning's understanding from flowlines.

Higher-order methods like the widely used fourth order Runge Kutta integrator employ more function evaluations per step to better capture the local behavior of integral curves. Conversely, lower order methods like Euler integration are more straightforward but less accurate in capturing intricate flow behaviors.

Despite the community preference for higher order integration methods in flow visualization, our findings revealed that utilizing lower order integration methods for our

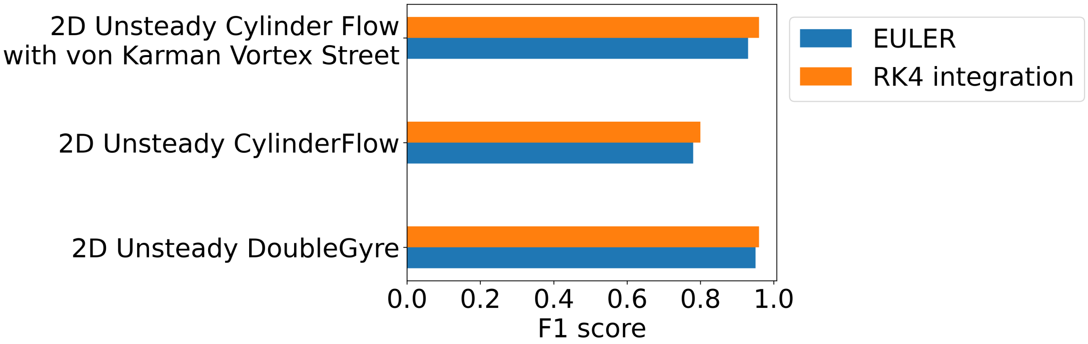


Figure 5.10: **Higher Order or Lower Order Numerical Integration Methods:** Notice that there is no significant improvement in using higher order numerical integration methods

machine learning model did not notably compromise its performance. This unexpected observation suggests that while higher order methods excel in flow visualization tasks, lower order methods can still effectively contribute to machine learning-based analysis of flowlines without significantly impairing model performance. We followed the norm of using RK4 but note that one can use Euler if limited computational resource is a hindrance.

## 5.5 Conclusion

The visualization community has been using deep learning methods directly on velocity components to find vortices. In this paper, we present a novel deep learning approach that learns from flowlines to find vortex boundary. The main contribution of this paper is a deep learning methodology utilizing flowlines to learn and identify vortex boundaries.

Hyperparameters of VortexViz were found based on the datasets tested in this paper. It is possible that further adjustments may be needed if a new dataset is markedly different from those used in this paper.

For each comparison paper, we used our best judgement to infer and reproduce the deep learning models associated with each proposed method. Our inference is based on the technical information provided by each paper. For most of the comparison papers the complete code, trained model weights, test and training data are not publicly available. Therefore, we do not claim that we have perfectly captured the authors' intentions. In order to make it easier for other researchers to improve upon our findings, the code for VortexViz will be available in the supplementary materials after the paper is accepted for publication.

## 5.6 Supplementary Material

### 5.6.1 Mathematical Expressions of Information Vectors

The first set of information vectors are derived from curl at each point of the flowline. For each point in the flowline, we calculated the curl as shown in equation 5.3 where $u$ represents the velocity component along the $x$ axis and $v$ represents the velocity component along the $y$ axis. *curl* measures the tendency at a given point for particles to rotate.

$$curl = \left\{ \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\bigg|_{\substack{x=x_1\\y=y_1}}, \cdots, \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y}\bigg|_{\substack{x=x_n\\y=y_n}} \right\} \tag{5.3}$$

From *curl*, we derive additional information vectors. As shown in equation 5.4 we calculate the absolute value of curl at each point of the flowline. *absolute curl* allows us to encode rotation at each point of the flowline without encoding direction.

$$absolute\ curl = \Big\{ |curl[1]|, \cdots, |curl[n]| \Big\} \tag{5.4}$$

In addition to *curl* and *absolute curl*, where physical quantities are calculated at each point of the flowline, we also calculate the cumulative quantities along the flowline. As shown in 5.5 and 5.6 we calculate the *cumulative curl* and *cummulative absolute curl* respectively. We hypothesize that flowlines originating from within vortices have higher *cumulative curl* and *cumulative absolute curl* than flowlines originating from laminar flow regions.

$$\begin{array}{l} cumulative \\ \\ curl \end{array} = \left\{ \sum_{i=1}^{1} curl[i], \cdots, \sum_{i=1}^{n} curl[i] \right\} \tag{5.5}$$

$$\begin{array}{l} cumulative \\ \\ absolute \\ \\ curl \end{array} = \left\{ \sum_{i=1}^{1} |\ curl[i]\ |, \cdots, \sum_{i=1}^{n} |\ curl[i]\ | \right\} \tag{5.6}$$

In addition to the vectors derived from *curl*, we also derived information vectors from the Euclidean distance between successive points of the flowline as shown in equation 5.7. We hypothesize that the distance traveled by a particle traced from inside a vortex would be different from a particle outside the vortex.

$$\begin{aligned} distance = \Big\{ & 0, \|(x_1,y_1),(x_2,y_2)\|, \cdots, \\ & \|(x_{n-1},y_{n-1}),(x_n,y_n)\| \Big\} \end{aligned} \tag{5.7}$$

We also calculated the Euclidean distance between each point of the flowline from the seed point $(x_1,y_1)$ as shown in equation 5.8. We supposed that the distance from the seed point would not exceed some maximum value for particles traced in the vortex.

$$\begin{aligned} \begin{matrix} distance\ from \\ seed\ point \end{matrix} = \Big\{ & 0, \|(x_1,y_1),(x_2,y_2)\|, \cdots, \\ & \|(x_1,y_1),(x_n,y_n)\| \Big\} \end{aligned} \tag{5.8}$$

With equation 5.7, we calculate the *cumulative distance* up to each point in the flowline as shown in equation 5.9

$$\begin{aligned} \begin{matrix} cumulative \\ distance \end{matrix} = \{ & \sum_{i=1}^{1} distance[i], \cdots, \\ & \sum_{i=1}^{n} distance[i] \} \end{aligned} \tag{5.9}$$

### 5.6.2 Training and Testing Data

We will provide a link to download training data, test data and code after publication

### 5.6.3 More Grad-cam Images

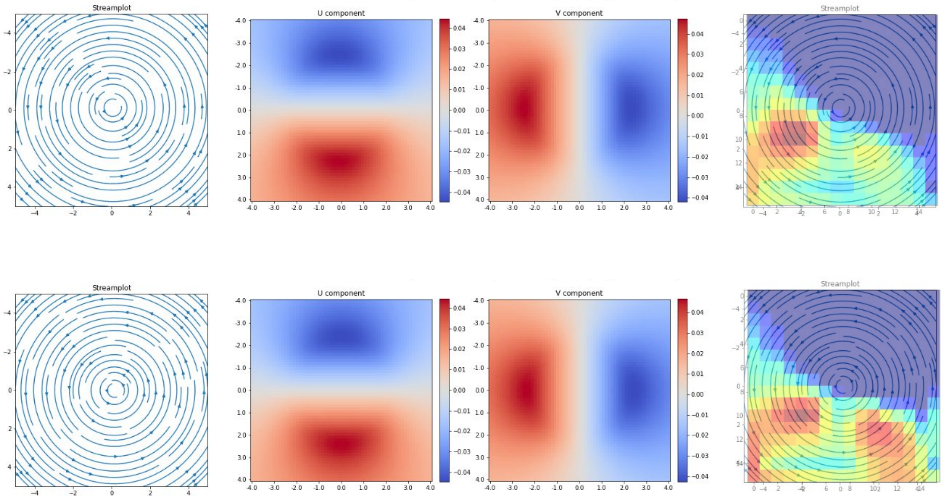


Figure 5.11: **Explaining what a CNN learns using Grad-cam:** We illustrate two vortices. The warmer regions within the overlaid heatmap reveal the features interpreted by a CNN as indicative of a vortex. It's worth observing that these highlighted features do not align with the vortex core. This observation leads us to hypothesize that what the CNN learns does not necessarily relate to the vortex itself but learns where higher values (red) are concentrated in the velocity components (two center columns).

# Chapter 6

# Conclusion

## 6.1 Summary

In this dissertation, we explored the integration of deep learning within flow visualization pipelines with a specific emphasis on feature detection. We demonstrated the use of deep learning for feature detection, specifically within the context of visualizing rip currents and vortices.

First, we explored using conventional flow visualization methods, such as vector clustering and timelines, to visualize rip currents. We found that by clustering the vector field derived from near-shore ocean videos based on direction, we could effectively visualize rip currents. Additionally, we used arrow glyphs and color maps to enhance the visibility of the rip current. Furthermore, We found that carefully placing timelines along the beach and observing its deformation can be used to visualize rip currents.

Second, we noticed a unique appearance corresponding to the rip current. In particular, for bathymetry-controlled rip currents, we found that there is a visible gap in the breaking waves, signifying the location of the rip current. Therefore, we explored using deep learning methods that can detect this pattern from near-shore ocean videos. Assuming stationary videos, we further refined our detection by using an accumulation buffer to filter out false positives and fill in false negatives.

Third, we explored the use of deep learning to learn about the behavior of rip currents without relying on appearance. One of the limitations of relying solely on appearance is the large amount of data that is needed to capture the varying appearances of rip currents. Additionally, some rip currents do not have a distinguishable appearance, making appearance-based methods even more challenging. Nevertheless, our observations revealed a consistent behavioral pattern in rip currents, characterized by their outward motion into the sea across various types of rips. In this work, we introduce a novel approach to learning about rip current behavior by using pathline sequences.

Finally, we delved into extending the use of flowlines (streamlines and pathlines) to find vortices. In previous work, when using flowlines in deep learning methods, flowlines were encapsulated in binary images before using in deep learning methods. However, we found that using the binary image alone is not sufficient for finding vortices. In this current work, we introduce a novel approach, adopting a multimodal framework, where, in addition to the binary image representation, we incorporate additional information extracted along the flowline trajectory, such as cumulative curl, to find vortices.

In conclusion, this dissertation explored approaches to utilize deep learning in flow visualization, showcasing its potential in advancing our understanding and visualization of complex fluid dynamics. It underscored the importance of combining conventional and novel methods to tackle the challenges posed in visualizing complex flow patterns.

## 6.2 Future Work

The work discussed in this dissertation has several promising avenues for future research and expansion. Two such directions stand out.

The first opportunity lies in the extension of the application of flowlines, including pathlines and streamlines, to identify and characterize additional complex flow patterns such as saddles and separatrices. In existing work, the application of deep learning to find other types of flow patterns is limited. This expansion could provide invaluable insights into the intricate dynamics of fluid systems and further expand our understanding of flow behavior.

Secondly, addressing the analysis of uncertainty associated with deep learning models used in flow visualization pipelines is another essential area for future exploration. In particular exploring the quantification and visualization of uncertainty would make flow visualization robust and reliable, especially in critical applications like environmental and ocean sciences.

These potential research directions not only contribute to the advancement of flow visualization and deep learning but also underscore the robustness and reliability of flow visualization pipelines.

# Bibliography

[1] S. Philip and A. Pang. Detecting and visualizing rip current using optical flow. In *Proceedings of the Eurographics*, EuroVis '16, pages 19–23, Goslar Germany, Germany, 2016. Eurographics Association.

[2] C. Jung, T. Tél, and E. Ziemniak. Application of scattering chaos to particle transport in a hydrodynamical flow. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 3(4):555–568, 1993.

[3] Corey Maryan, Md Tamjidul Hoque, Christopher Michael, Elias Ioup, and Mahdi Abdelguerfi. Machine learning applications in detecting rip channels from images. *Applied Soft Computing*, 78:84–93, 2019.

[4] Jarke J Van Wijk. The value of visualization. In *VIS 05. IEEE Visualization, 2005.*, pages 79–86. IEEE, 2005.

[5] Charles D Hansen and Chris R Johnson. *Visualization handbook*. Elsevier, 2011.

[6] Barbara Brighton, Shauna Sherker, Rob Brander, M. Thompson, and A. Bradstreet. Rip current related drowning deaths and rescues in Australia 2004-2011. *Natural Hazards and Earth System Sciences*, 13:1069–1075, 04 2013.

[7] A. H. da F. Klein, G. G. Santana, F. L. Diehl, and J. T. de Menezes. Analysis of hazards associated with sea bathing: Results of five years work in oceanic beaches of Santa Catarina state, Southern Brazil. *Journal of Coastal Research*, pages 107–116, 2003.

[8] James B. Lushine. A study of rip current drownings and related weather factors. *National Weather Digest*, pages 13–19, 1991.

[9] B. C. Brewster, R. E. Gould, and R. W. Brander. Estimations of rip current rescues and drowning in the United States. *Natural Hazards and Earth System Sciences*, 19(2):389–397, 2019.

[10] National Oceanic and Atmospheric Administration. Safety national program surf zone fatalities in nws areas of forecast responsibility, 2020.

[11] Stephen Leatherman John Fletemeyer. Rip currents and beach safety education. *Journal of Coastal Research*, 2010(261):1–3, 2010.

[12] Robert W. Brander and Jamie H. MacMahan. Future challenges for rip current research and outreach. In Stephen Leatherman and John Fletemeyer, editors, *Rip Currents: Beach Safety, Physical Oceanography, and Wave Modeling*, pages 1–30. CRC Press, 2011.

[13] C. Brannstrom, S. Trimble, A. Santos, H.L. Brown, and C. Houser. Perception of the rip current hazard on Galveston Island and North Padre Island, Texas, USA. *Natural Hazards*, 72(2):1123–1138, 2014.

[14] C. Houser, S. Trimble, R. Brander, B. C. Brewster, G. Dusek, D. Jones, and J. Kuhn. Public perceptions of a rip current hazard education program: Break the Grip of the Rip! *Natural Hazards and Earth System Sciences*, 17(7):1003–1024, 2017.

[15] Anthony J Bowen. Rip currents: Theoretical investigations. *Journal of Geophysical Research*, 74(23):5467–5478, 1969.

[16] R. A. Holman, G. Symonds, E. B. Thornton, and R. Ranasinghe. Rip spacing and persistence on an embayed beach. *Journal of Geophysical Research-Oceans*, 111(C1), 2006.

[17] B. Castelle, T. Scott, R. W. Brander, and R. J. McCarroll. Rip current types, circulation and hazard. *Earth-Science Reviews*, 163:1–21, 2016.

[18] G. Dusek and H. Seim. A probabilistic rip current forecast model. *Journal of Coastal Research*, 29(4):909–925, 2013.

[19] L. Sembiring. Rip current prediction system for swimmer safety: Towards operational forecasting using a process based model and nearshore bathymetry from video. In *Rip Current Prediction System for Swimmer Safety: Towards Operational Forecasting Using a Process Based Model and Nearshore Bathymetry from Video*, 2016.

[20] NOAA Launches First National Rip Current Forecast Model.

[21] Surf Life Saving Australia. National coastal safety report. *https://issuu.com/surflifesavingaustralia/docs/ncsr2019*, 2019.

[22] Christine M Branche. Lifeguard effectiveness: A report of the working group. *Centers for Disease Control and Prevention, National Center for Injury Prevention and Control*, 2001.

[23] Christian Brannstrom, Heather Brown, Chris Houser, Sarah Trimble, and Anna Lavoie. "You can't see them from sitting here": Evaluating beach user understanding of a rip current warning sign. *Applied Geography*, 56:61–70, 01 2015.

[24] Stephen B Leatherman and Stephen P Leatherman. Techniques for detecting and measuring rip currents. *International Journal of Earth Science and Geophysics*, 3:1–5, 9 2017.

[25] S. Elgar, B. Raubenheimer, and R. T. Guza. Current meter performance in the surf zone. *Journal of Atmospheric and Oceanic Technology*, 18(10):1735–1746, 2001. n/a.

[26] Kris Inch. Surf zone hydrodynamics: Measuring waves and currents. *Geomorphological Techniques*, Chap. 3, Sec 2.3, 2014.

[27] D. Johnson and C. Pattiaratchi. Transient rip currents and nearshore circulation on a swell-dominated beach. *Journal of Geophysical Research: Oceans*, 109(C2), 2004.

[28] Yuli Liu and Chin H. Wu. Lifeguarding operational camera kiosk system (LOCKS) for flash rip warning: Development and application. *Coastal Engineering*, 152:103537, 2019.

[29] Robert W Brander, Danielle Drozdzewski, and Dale Dominey-Howes. "Dye in the Water": A visual approach to communicating the rip current hazard. *Science Communication*, 36(6):802–810, 2014.

[30] David Clark, Falk Feddersen, and R. Guza. Cross-shore surfzone tracer dispersion in an alongshore current. *Journal of Geophysical Research (Oceans)*, 115, 10 2010.

[31] David B Clark, Luc Lenain, Falk Feddersen, Emmanuel Boss, and RT Guza. Aerial imaging of fluorescent dye in the near shore. *Journal of Atmospheric and Oceanic Technology*, 31(6):1410–1421, 2014.

[32] D. W. Pritchard and J. H. Carpenter. Measurements of turbulent diffusion in estuarine and inshore waters. *International Association of Scientific Hydrology Bulletin*, 5(4):37–50, 1960.

[33] Bruno Castelle, Rafael Almar, Matthieu Dorel, Jean-Pierre Lefebvre, Nadia Senechal, Edward J. Anthony, Raoul Laibi, RÃ©my Chuchla, and Yves du Penhoat. Rip currents and circulation on a high-energy low-tide-terraced beach (Grand Popo, Benin, West Africa). *Journal of Coastal Research*, 70(sp1):633 – 638, 2014.

[34] WE Schmidt, BT Woodward, KS Millikan, RT Guza, B Raubenheimer, and Steve Elgar. A GPS-tracked surf zone drifter. *Journal of Atmospheric and Oceanic Technology*, 20(7):1069–1075, 2003.

[35] Gregory Dusek, Debra Hernandez, Mark Willis, Jenna A Brown, Joseph W Long, Dwayne E Porter, and Tiffany C Vance. WebCAT: Piloting the development of a web camera coastal observing network for diverse applications. *Frontiers in Marine Science*, 6:353, 2019.

[36] K Todd Holland, Robert A Holman, Thomas C Lippmann, John Stanley, and Nathaniel Plant. Practical use of video imagery in nearshore oceanographic field studies. *IEEE Journal of Oceanic Engineering*, 22(1):81–92, 1997.

[37] Rob A. Holman and J. Stanley. The history and technical capabilities of Argus. *Coastal Engineering*, 54(6-7):477–491, 2007.

[38] Tom C. Lippmann and Rob A. Holman. Quantification of sand bar morphology: A video technique based on wave dissipation. *Journal of Geophysical Research: Oceans*, 94(C1):995–1011, 1989.

[39] S. Pitman, S. L. Gallop, I. D. Haigh, S. Mahmoodi, G. Masselink, and R. Ranasinghe. Synthetic imagery for the automated detection of rip currents. *Journal of Coastal Research*, pages 912–916, 2016.

[40] C. Maryan, M. T. Hoque, C. Michael, E. Ioup, and M. Abdelguerfi. Machine learning applications in detecting rip channels from images. *Applied Soft Computing*, 78:84–93, 2019.

[41] V. Nelko and R.A. Dalrymple. Rip current prediction in Ocean City, Maryland. In *Rip Currents: Beach Safety, Physical Oceanography and Wave Modeling*, pages 45–58. CRC Press, 2011.

[42] Akila de Silva, Issei Mori, Gregory Dusek, James Davis, and Alex Pang. Automated rip current detection with region based convolutional neural networks. *Coastal Engineering*, page 103859, 2021.

[43] J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, Feb 1994.

[44] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17(1-3):185–203, 1981.

[45] Pierre Dérian and Rafael Almar. Wavelet-based optical flow estimation of instant surface currents from shore-based and UAV videos. *IEEE Transactions on Geoscience and Remote Sensing*, 55(10):5790–5797, 2017.

[46] Dylan Anderson, A. Spicer Bak, Katherine L. Brodie, Nicholas Cohn, Rob A. Holman, and John Stanley. Quantifying optically derived two-dimensional wave-averaged currents in the surf zone. *Remote Sensing*, 13(4), 2021.

[47] Liu Feng, Michael Gleicher, Jue Wang, Hailin Jin, and Aseem Agarwala. Subspace video stabilization. *ACM Transactions on Graphics (TOG)*, 30(1):1–10, 2011.

[48] Matsushita Yasuyuki, Eyal Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *IEEE Transactions on pattern analysis and Machine Intelligence*, 28(7):1150–1163, 2006.

[49] Zitova Barbara and Jan Flusse. Image registration methods: a survey. *Image and vision computing*, 21(11):977–1000, 2003.

[50] Lisa Gottesfeld Brown. A survey of image registration techniques. *ACM computing surveys (CSUR)*, 24(4):325–376, 1992.

[51] G Farneback. Two-frame motion estimation based on polynomial expansion. *Scandinavian Conference on Image Analysis*, 13:363–370, 2003.

[52] B. Lucas and T Kanade. An iterative image registration technique with an application to stereo vision. *International Joint Conference on Artificial Intelligence*, pages 674–679, 1981.

[53] Baker Simon and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *International journal of computer vision*, 56(3):221–255, 2004.

[54] Beauchemin Steven S. and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[55] Bradski Gary and Adrian Kaehle. Opencv. *Dr. Dobb's journal of software tools*, 3, 2000.

[56] D. A. Lane. Visualization of time-dependent flow fields. In *Proceedings Visualization '93*, pages 32–38, Oct 1993.

[57] Southeast Coastal Ocean Observing Regional Association SECOORA. Webcat – live cameras and historic feeds, 2020.

[58] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR*, 1(511-518):3, 2001.

[59] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 91–99. Advances in neural information processing systems, 2015.

[60] B. Chris Brewster, Richard E. Gould, and Robert W. Brander. Estimations of rip current rescues and drowning in the United States. *Natural Hazards and Earth System Sciences*, 19(2):389–397, 2019.

[61] B Castelle, T Scott, RW Brander, and RJ McCarroll. Rip current types, circulation and hazard. *Earth-Science Reviews*, 163:1–21, 2016.

[62] M. Brocchini, A. Kennedy, L. Soldini, and A. Mancinelli. Topographically controlled, breaking-wave-induced macrovortices. Part 1. Widely separated breakwaters. *Journal of Fluid Mechanics*, 507:289–307, May 2004.

[63] A. B. Kennedy, M. Brocchini, L. Soldini, and E. Gutierrez. Topographically controlled, breaking-wave-induced macrovortices. Part 2. Changing geometries. *Journal of Fluid Mechanics*, 559:57, July 2006.

[64] M. Postacchini, M. Brocchini, and L. Soldini. Vorticity generation due to cross-sea. *Journal of Fluid Mechanics*, 744:286–309, April 2014.

[65] A. Piattella, M. Brocchini, and A. Mancinelli. Topographically controlled, breaking-wave-induced macrovortices. Part 3. The mixing features. *Journal of Fluid Mechanics*, 559:81, July 2006.

[66] Nirnimesh Kumar and Falk Feddersen. The effect of stokes drift and transient rip currents on the inner shelf. part i: No stratification. *Journal of Physical Oceanography*, 47(1):227–241, 2017.

[67] Nirnimesh Kumar and Falk Feddersen. A new offshore transport mechanism for shoreline-released tracer induced by transient rip currents and stratification. *Geophysical Research Letters*, 44(6):2843–2851, 2017.

[68] Daniel Buscombe and Roxanne J. Carini. A Data-Driven Approach to Classifying Wave Breaking in Infrared Imagery. *Remote Sensing*, 11(7):859, January 2019.

[69] Daniel Buscombe and Andrew C. Ritchie. Landscape Classification with Deep Neural Networks. *Geosciences*, 8(7):244, July 2018.

[70] Jessica Y. Luo, Jean-Olivier Irisson, Benjamin Graham, Cedric Guigand, Amin Sarafraz, Christopher Mader, and Robert K. Cowen. Automated plankton image analysis using convolutional neural networks. *Limnology and Oceanography: Methods*, 16(12):814–827, 2018.

[71] Estanislau Lima, Xin Sun, Junyu Dong, Hui Wang, Yuting Yang, and Lipeng Liu. Learning and Transferring Convolutional Neural Network Knowledge to Ocean Front Recognition. *IEEE Geoscience and Remote Sensing Letters*, 14(3):354–358, March 2017.

[72] Daniel Buscombe, Roxanne J. Carini, Shawn R. Harrison, C. Chris Chickadel, and Jonathan A. Warrick. Optical wave gauging using deep neural networks. *Coastal Engineering*, 155:103593, January 2020.

[73] C. E. Stringari, D. L. Harris, and H. E. Power. A novel machine learning algorithm for tracking remotely sensed waves in the surf zone. *Coastal Engineering*, 147:149–158, May 2019.

[74] Guo-Qing Jiang, Jing Xu, and Jun Wei. A deep learning algorithm of neural network for the parameterization of typhoon-ocean feedback in typhoon forecast models. *Geophysical Research Letters*, 45(8):3706–3716, 2018.

[75] Aurelien Ducournau and Ronan Fablet. Deep learning for ocean remote sensing: an application of convolutional neural networks for super-resolution on satellite-derived SST data. In *2016 9th IAPR Workshop on Pattern Recogniton in Remote Sensing (PRRS)*, pages 1–6, December 2016. ISSN: null.

[76] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. IEEE, 2009.

[77] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[78] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[79] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[80] Yuli Liu and Chin H Wu. Lifeguarding operational camera kiosk system (LOCKS) for flash rip warning: Development and application. *Coastal Engineering*, 152:103537, 2019.

[81] Merrick C Haller, David Honegger, and Patricio A Catalan. Rip current observations via marine radar. *Journal of waterway, port, coastal, and ocean engineering*, 140(2):115–124, 2014.

[82] Guy A Meadows, Amanda Grimm, Colin N Brooks, and Robert A Shuchman. Remote sensing-based detection and monitoring of dangerous nearshore currents. In *Remote sensing-based detection and monitoring of dangerous nearshore currents*. IAGLR 58th Annual Conference on Great Lakes Research, 2015.

[83] Sebastian Pitman, Shari L Gallop, Ivan D Haigh, Sasan Mahmoodi, Gerd Masselink, and Roshanka Ranasinghe. Synthetic imagery for the automated detection of rip currents. *Journal of coastal research*, 75:912–916, 2016.

[84] Shweta Philip and Alex Pang. Detecting and Visualizing Rip Current Using Optical Flow. In Enrico Bertini, Niklas Elmqvist, and Thomas Wischgoll, editors, *EuroVis 2016 - Short Papers*, page 115. The Eurographics Association, 2016.

[85] Junwei Han, Dingwen Zhang, Gong Cheng, Nian Liu, and Dong Xu. Advanced deep-learning techniques for salient and category-specific object detection: A survey. *IEEE Signal Processing Magazine*, 35(1):84–100, 2018.

[86] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *arXiv preprint arXiv:1809.02165*, 2018.

[87] Constantine P. Papageorgiou, Michael Oren, and Tomaso Poggio. A general framework for object detection. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pages 555–562, Washington, DC, USA, 1998. IEEE Computer Society.

[88] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, page 0, June 2016.

[89] Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, May 2004.

[90] Wei Han, Pooya Khorrami, Tom Le Paine, Prajit Ramachandran, Mohammad Babaeizadeh, Honghui Shi, Jianan Li, Shuicheng Yan, and Thomas S. Huang. Seq-NMS for video object detection. *CoRR*, abs/1602.08465, 2016.

[91] Kai Kang, Wanli Ouyang, Hongsheng Li, and Xiaogang Wang. Object detection from video tubelets with convolutional neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 817–825, Las Vegas, NV, USA, June 2016. IEEE.

[92] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.

[93] MV Athira and Diliya M Khan. Recent trends on object detection and image classification: A review. In *2020 International Conference on Computational Performance Evaluation (ComPE)*, pages 427–435. IEEE, 2020.

[94] Sushma Jaiswal and Tarun Jaiswal. Deep learning approaches for object detection. *Artificial Intelligence Evolution*, pages 122–144, 2020.

[95] Cannannore Nidhi Kamath, Syed Saqib Bukhari, and Andreas Dengel. Comparative study between traditional machine learning and deep learning approaches for text classification. In *Proceedings of the ACM Symposium on Document Engineering 2018*, pages 1–11, 2018.

[96] Athanasios Voulodimos, Nikolaos Doulamis, Anastasios Doulamis, and Eftychios Protopapadakis. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[97] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 2020.

[98] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

[99] Sumit Gupta. Deep learning performance breakthrough, January 2018. Library Catalog: www.ibm.com.

[100] Gregory Perrier. Automated rip current detection system, December 8 2005. US Patent App. 11/203,771.

[101] Anamika Dhillon and Gyanendra K Verma. Convolutional neural network: a review of models, methodologies and applications to object detection. *Progress in Artificial Intelligence*, 9(2):85–112, 2020.

[102] Panagiotis Barmpoutis, Kosmas Dimitropoulos, Kyriaki Kaza, and Nikos Grammalidis. Fire detection from images using faster r-cnn and multidimensional texture analysis. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 8301–8305. IEEE, 2019.

[103] Jong Hyun Kim, Ganbayar Batchuluun, and Kang Ryoung Park. Pedestrian detection based on faster r-cnn in nighttime by fusing deep convolutional features of successive images. *Expert Systems with Applications*, 114:15–33, 2018.

[104] Shaolong Ma, Yang Huang, Xiangjiu Che, and Rui Gu. Faster rcnn-based detection of cervical spinal cord injury and disc degeneration. *Journal of Applied Clinical Medical Physics*, 21(9):235–243, 2020.

[105] Tahir Mahmood, Muhammad Arsalan, Muhammad Owais, Min Beom Lee, and Kang Ryoung Park. Artificial intelligence-based mitosis detection in breast cancer histopathology images using faster r-cnn and deep cnns. *Journal of Clinical Medicine*, 9(3):749, 2020.

[106] Jane Hung and Anne Carpenter. Applying faster r-cnn for object detection on malaria images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[107] Ross Girshick. Fast R-CNN. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[108] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[109] Gabriele Paolacci, Jesse Chandler, and Panagiotis G Ipeirotis. Running experiments on amazon mechanical turk. *Judgment and Decision making*, 5(5):411–419, 2010.

[110] Jenny J Chen, Natala J Menezes, Adam D Bradley, and T North. Opportunities for crowdsourcing research on amazon mechanical turk. *Interfaces*, 5(3):1, 2011.

[111] Xizhou Zhu, Yujie Wang, Jifeng Dai, Lu Yuan, and Yichen Wei. Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417, 2017.

[112] Xizhou Zhu, Yuwen Xiong, Jifeng Dai, Lu Yuan, and Yichen Wei. Deep feature flow for video recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2349–2358, 2017.

[113] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. A survey on deep transfer learning. In *International conference on artificial neural networks*, pages 270–279. Springer, 2018.

[114] S. Leatherman and J. Fletemeyer, editors. *Rip Currents: Beach Safety, Physical Oceanography, and Wave Modeling (1st ed.)*. CRC Press, 2011.

[115] Jamie MacMahan, Ad Reniers, Jenna Brown, Rob Brander, Ed Thornton, Tim Stanton, Jeff Brown, and Wendy Carey. An Introduction to Rip Currents Based on Field Observations. *Journal of Coastal Research*, 27(4), 2011.

[116] A. H. da F. Klein, G. G. Santana, F. L. Diehl, and J. T. de Menezes. Analysis of hazards associated with sea bathing: Results of five years work in oceanic beaches of Santa Catarina state, Southern Brazil. *Journal of Coastal Research*, pages 107–116, 2003.

[117] James B. Lushine. A study of rip current drownings and related weather factors. *National Weather Digest*, pages 13–19, 1991.

[118] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318, 2020.

[119] Akila de Silva, Issei Mori, Gregory Dusek, James Davis, and Alex Pang. Automated rip current detection with region based convolutional neural networks. *Coastal Engineering*, 166:103859, 2021.

[120] Ashraf Haroon Rashid, Imran Razzak, M. Tanveer, and Antonio Robles-Kelly. RipNet: A lightweight one-class deep neural network for the identification of RIP currents. In *Communications in Computer and Information Science*, pages 172–179. Springer International Publishing, 2020.

[121] Ashraf Haroon Rashid, Imran Razzak, M. Tanveer, and Antonio Robles-Kelly. RipDet: A fast and lightweight deep neural network for rip currents detection. In *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2021.

[122] Shweta Philip and Alex Pang. Detecting and visualizing rip current using optical flow. In *EuroVis (Short Papers)*, pages 19–23, 2016.

[123] Issei Mori, Akila De Silva, Gregory Dusek, James Davis, and Alex Pang. Flow-based rip current detection and visualization. *IEEE Access*, 2022.

[124] Sudhanshu Sane, Roxana Bujack, Christoph Garth, and Hank Childs. A survey of seed placement and streamline selection techniques. *Computer Graphics Forum*, 39(3):785–809, 2020.

[125] Bernardo S. Carmo, Y. H. Pauline Ng, Adam Prügel-Bennett, and Guang-Zhong Yang. A data clustering and streamline reduction method for 3d mr flow vector field simplification. In Christian Barillot, David R. Haynor, and Pierre Hellier, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2004*, pages 451–458, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[126] S.K. Lodha, J.C. Renteria, and K.M. Roskin. Topology preserving compression of 2d vector fields. In *Proceedings Visualization 2000. VIS 2000 (Cat. No.00CH37145)*, pages 343–350, 2000.

[127] Stephane Marchesin, Cheng-Kai Chen, Chris Ho, and Kwan-Liu Ma. View-dependent streamlines for 3D vector fields. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1578–1586, 2010.

[128] Jun Ma, Chaoli Wang, and Ching-Kuang Shene. Coherent view-dependent streamline selection for importance-driven flow visualization. In *Visualization and Data Analysis 2013*, volume 8654, page 865407. International Society for Optics and Photonics, 2013.

[129] Hongfeng Yu, Chaoli Wang, Ching-Kuang Shene, and Jacqueline H Chen. Hierarchical streamline bundles. *IEEE Transactions on Visualization and Computer Graphics*, 18(8):1353–1367, 2011.

[130] Jun Tao, Jun Ma, Chaoli Wang, and Ching-Kuang Shene. A unified approach to streamline selection and viewpoint selection for 3d flow visualization. *IEEE Transactions on Visualization and Computer Graphics*, 19(3):393–406, 2012.

[131] Marzieh Berenjkoub, Guoning Chen, and Tobias Günther. Vortex boundary identification using convolutional neural network. In *2020 IEEE Visualization Conference (VIS)*, pages 261–265. IEEE, 2020.

[132] Byungsoo Kim and Tobias Günther. Robust reference frame extraction from unsteady 2d vector fields with convolutional neural networks. In *Computer Graphics Forum*, volume 38, pages 285–295. Wiley Online Library, 2019.

[133] Jun Han, Jun Tao, and Chaoli Wang. Flownet: A deep learning framework for clustering and selection of streamlines and stream surfaces. *IEEE transactions on visualization and computer graphics*, 26(4):1732–1744, 2018.

[134] S.B. Leatherman and S.P. Leatherman. Techniques for detecting and measuring rip currents. *International Journal of Earth Science and Geophysics (ISSN: 2631-5033)*, 3(1), 2017.

[135] Merrick Haller, David Honegger, and Patricio Catalán. Rip current observations via marine radar. *Journal of Waterway Port Coastal and Ocean Engineering*, 140:115–124, 03 2014.

[136] Robert Holman and Merrick Haller. Remote sensing of the nearshore. *Annual review of marine science*, 5, 12 2011.

[137] Sean Ellis and Sean McGill. Rip current and channel detection using surfcams and optical flow. *Shore & Beach*, 90(1):50–58, 08 2022.

[138] Junhai Zhai, Sufang Zhang, Junfen Chen, and Qiang He. Autoencoder and its various variants. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 415–419, 2018.

[139] Junfen Chen, Bojun Xie, Hui Zhang, and Junhai Zhai. Deep autoencoders in pattern recognition: a survey. In *Bio-inspired computing models and algorithms*, pages 229–255. World Scientific, 2019.

[140] Manassés Ribeiro, André Eugênio Lazzaretti, and Heitor Silvério Lopes. A study of deep convolutional auto-encoders for anomaly detection in videos. *Pattern Recognition Letters*, 105:13–22, 2018.

[141] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[142] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Icml*, 2010.

[143] François Chollet et al. Keras, 2015.

[144] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[145] Kai Burger, Polina Kondratieva, Jens Kruger, and Rudiger Westermann. Importance-driven particle techniques for flow visualization. In *2008 IEEE Pacific Visualization Symposium*, pages 71–78. IEEE, 2008.

[146] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

[147] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231, 1996.

[148] Holger Theisel and H-P Seidel. Feature flow fields. In *Proceedings of the symposium on Data visualisation 2003*, pages 141–148, 2003.

[149] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3-4):271–304, 2005.

[150] Tobias Günther, Markus Gross, and Holger Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 36(4):141:1–141:11, 2017.

[151] Thomas Gerz, Frank Holzäpfel, and Denis Darracq. Commercial aircraft wake vortices. *Progress in Aerospace Sciences*, 38(3):181–208, 2002.

[152] C Breitsamter. Wake vortex characteristics of transport aircraft. *Progress in Aerospace Sciences*, 47(2):89–134, 2011.

[153] Steven C Rennich and Sanjiva K Lele. Method for accelerating the destruction of aircraft wake vortices. *Journal of Aircraft*, 36(2):398–404, 1999.

[154] Redouane Lguensat, Miao Sun, Ronan Fablet, Pierre Tandeo, Evan Mason, and Ge Chen. Eddynet: A deep neural network for pixel-wise classification of oceanic eddies. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 1764–1767. IEEE, 2018.

[155] Sean Williams, Matthew Hecht, Mark Petersen, Richard Strelitz, Mathew Maltrud, J Ahrens, Mario Hlawitschka, and Bernd Hamann. Visualization and analysis of eddies in a global ocean simulation. In *Computer graphics forum*, volume 30, pages 991–1000. Wiley Online Library, 2011.

[156] Taotao Wang, Hongchang He, Donglin Fan, Bolin Fu, and Shundan Dong. Global ocean mesoscale vortex recognition based on DeeplabV3plus model. In *IOP Conference Series: Earth and Environmental Science*, volume 671, page 012001. IOP Publishing, 2021.

[157] Gia Dvali, Florian Kühnel, and Michael Zantedeschi. Vortices in black holes. *Physical Review Letters*, 129(6):061302, 2022.

[158] FCE Lima, ARP Moreira, and CAS Almeida. Properties of black hole vortex in Einstein's gravity. *The European Physical Journal Plus*, 138(5):429, 2023.

[159] Fay Dowker, Ruth Gregory, and Jennie Traschen. Euclidean black-hole vortices. *Physical Review D*, 45(8):2762, 1992.

[160] A Tonomura, H Kasai, O Kamimura, T Matsuda, K Harada, J Shimoyama, K Kishio, and K Kitazawa. Motion of vortices in superconductors. *Nature*, 397(6717):308–309, 1999.

[161] John Bardeen and MJ Stephen. Theory of the motion of vortices in superconductors. *Physical Review*, 140(4A):A1197, 1965.

[162] S Jonathan Chapman and Giles Richardson. Motion of vortices in type II superconductors. *SIAM Journal on Applied Mathematics*, 55(5):1275–1296, 1995.

[163] George Haller, Alireza Hadjighasem, Mohammad Farazmand, and Florian Huhn. Defining coherent vortices objectively from the vorticity. *Journal of Fluid Mechanics*, 795:136–173, 2016.

[164] Monika Jankun-Kelly, Ming Jiang, David Thompson, and Raghu Machiraju. Vortex visualization for practical engineering applications. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):957–964, 2006.

[165] T Maxworthy. Some experimental studies of vortex rings. *Journal of Fluid Mechanics*, 81(3):465–495, 1977.

[166] Hans J Lugt. The dilemma of defining a vortex. In *Recent developments in theoretical and experimental fluid mechanics: Compressible and incompressible flows*, pages 309–321. Springer, 1979.

[167] Liang Deng, Yueqing Wang, Yang Liu, Fang Wang, Sikun Li, and Jie Liu. A cnn-based vortex identification method. *Journal of Visualization*, 22:65–78, 2019.

[168] Liang Deng, Wenchun Bao, Yueqing Wang, Zhigong Yang, Dan Zhao, Fang Wang, Chongke Bi, and Yang Guo. Vortex-u-net: An efficient and effective vortex detection approach based on u-net structure. *Applied Soft Computing*, 115:108229, 2022.

[169] Yueqing Wang, Liang Deng, Zhigong Yang, Dan Zhao, and Fang Wang. A rapid vortex identification method using fully convolutional segmentation network. *The Visual Computer*, 37:261–273, 2021.

[170] Babak Kashir, Marco Ragone, Ajaykrishna Ramasubramanian, Vitaliy Yurkiv, and Farzad Mashayek. Application of fully convolutional neural networks for feature extraction in fluid flow. *Journal of Visualization*, 24:771–785, 2021.

[171] Qianwen Wang, Zhutian Chen, Yong Wang, and Huamin Qu. A survey on ml4vis: Applying machine learning advances to data visualization. *IEEE transactions on visualization and computer graphics*, 28(12):5134–5153, 2021.

[172] Can Liu, Ruike Jiang, Datong Wei, Changhe Yang, Yanda Li, Fang Wang, and Xiaoru Yuan. Deep learning approaches in flow visualization. *Advances in Aerodynamics*, 4(1):1–14, 2022.

[173] Chaoli Wang and Jun Han. Dl4scivis: A state-of-the-art survey on deep learning for scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 2022.

[174] Akila de Silva, Mona Zhao, Donald Stewart, Fahim Khan, Gregory Dusek, James Davis, and Alex Pang. Ripviz: Finding rip currents by learning pathline behavior. *IEEE Transactions on Visualization and Computer Graphics*, pages 1–13, 2023.

[175] Mengjiao Han, Sudhanshu Sane, and Chris R Johnson. Exploratory lagrangian-based particle tracing using deep learning. *Journal of Flow Visualization and Image Processing*, 29(3), 2022.

[176] JCR Hunt. Vorticity and vortex dynamics in complex turbulent flows. *Transactions of the Canadian Society for Mechanical Engineering*, 11(1):21–35, 1987.

[177] ChaoQun Liu, YiQian Wang, Yong Yang, and ZhiWei Duan. New omega vortex identification method. *Science China Physics, Mechanics & Astronomy*, 59:1–9, 2016.

[178] Jinhee Jeong and Fazle Hussain. On the identification of a vortex. *Journal of fluid mechanics*, 285:69–94, 1995.

[179] Min S Chong, Anthony E Perry, and Brian J Cantwell. A general classification of three-dimensional flow fields. *Physics of Fluids A: Fluid Dynamics*, 2(5):765–777, 1990.

[180] I Ari Sadarjoen, Frits H Post, Bing Ma, David C Banks, and H-G Pagendarm. Selective visualization of vortices in hydrodynamic flows. In *Proceedings Visualization'98 (Cat. No. 98CB36276)*, pages 419–422. IEEE, 1998.

[181] George Haller. An objective definition of a vortex. *Journal of fluid mechanics*, 525:1–26, 2005.

[182] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[183] John Charles Butcher. A history of Runge-Kutta methods. *Applied numerical mathematics*, 20(3):247–260, 1996.

[184] S. C. Shadden, F. Lekien, and J. E. Marsden. Definition and properties of Lagrangian coherent structures from finite-time Lyapunov exponents in two-dimensional aperiodic flows. *Physica D: Nonlinear Phenomena*, 212(3-4):271–304, 2005.

[185] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004.

[186] Tobias Günther, Markus Gross, and Holger Theisel. Generic objective vortices for flow visualization. *ACM Transactions on Graphics (Proc. SIGGRAPH)*, 36(4):141:1–141:11, 2017.

[187] Alexander Wiebel, Raymond Chan, Christina Wolf, Andrea Robitzki, Angela Stevens, and Gerik Scheuermann. *Topological Flow Structures in a Mathematical Model for Rotation-Mediated Cell Aggregation*, pages 193–204. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[188] Tino Weinkauf and Holger Theisel. Streak lines as tangent curves of a derived vector field. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization)*, 16(6):1225–1234, Nov 2010.

[189] S. Popinet. Free computational fluid dynamics. *ClusterWorld*, 2(6), 2004.

[190] Irene Baeza Rojo and Tobias Günther. Vector field topology of time-dependent flows in a steady reference frame. *IEEE Transactions on Visualization and Computer Graphics (Proc. IEEE Scientific Visualization)*, 2019.

[191] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*, pages 618–626, 2017.

[192] NOAASatellites. Dorian Timelapse, September 2019.

[193] Robert S Laramee, Helwig Hauser, Helmut Doleisch, Benjamin Vrolijk, Frits H Post, and Daniel Weiskopf. The state of the art in flow visualization: Dense

and texture-based techniques. In *Computer Graphics Forum*, volume 23, pages 203–221. Wiley Online Library, 2004.