Investigation into the use of surf web cameras to improve the accuracy of forecasting wave behavior

Patrick Green

4th Year Project Report
Computer Science and Physics
School of Informatics
University of Edinburgh

2017

Abstract

Countless web cameras scattered about the UK help surfers decide which beach to visit on any given day. Modern weather models predict the wave features based on deep ocean readings but the underlying morphology of the beach and weather based external factors vary the actual conditions substantially. No strong method of feedback for the accuracy of these weather models, exists and with such a chaotic system, the accuracy can vary substantially.

Using a selection of low resolution web cameras, this project is aiming to develop the algorithms required to visually detect wave crests in a video stream and subsequently track their movements. Furthermore, using this data, an estimation of the physical height and speed of the wave can be made.

Acknowledgments

Table of Contents

Chapter 1

Introduction

As competition within the surfing market grows the necessity of web cameras as advertisement to coax surfers to visit a certain beach and shop also increases. Using both a sample of beach web cameras streamed online and a sufficiently precise wave feature detection algorithms one could provide feedback on wave forecasting model predictions. This project aims to derive this detection algorithm and investigate their use into improving the accuracy of any wave forecasting model.

This chapter introduces the reader into the sport highlighting the differences between surf-able and unsurf-able waves, outlining the wave features of interest that wave forecasting models generate and presents implementation concerns.

1.1 Formation of waves

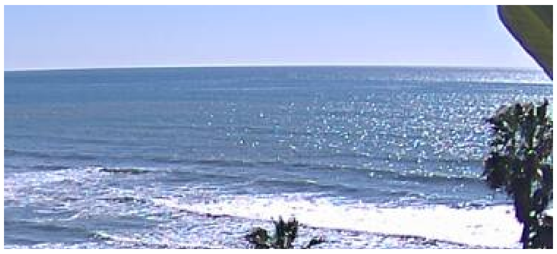
The movement of the tides causes large ripples in the ocean. These ripples are translated through the ocean in a circular trajectory where the water molecules at the top of the crest move forwards with the direction of travel and the molecules at the bottom move against it. A bulk of water with a single circular motion as outlined is a wave. In the deep ocean there is very little friction therefore the system may travel thousands of kilometers but never lose size. Once a wave meets the shore the energy it holds dissipates as the backwards moving current at the bottom comes into contact with the ground. This process pulls away at the sediment on the beach, forming several variations of increasing slope profile. The increasing slope forces the current at the top of the wave to protrude from the water surface as it is pushed forward. The surface tension of water is enough to hold the surface of the wave in a concave shape above the water level. The sharpness of the concave shape is governed by the ground profile and external factors such as the wind and temperature of the water. The wave height and speed are also influenced by these factors as well as the wave energy.

1.2 Detection Criteria

This investigation will predominantly focus on detecting surf-able waves through improving the accuracy of surf forecasting model predictions. Figures 1.1 and 1.2 provided below show the contrast of this criteria.

The wave energy is the most important factor in deciphering how likely it will be that the wave conditions will be surf-able. As outlined in the last section a more powerful wave will have a sharper concave surface and be taller than the average wave. Therefore, the likelihood of any given wave being surf-able can be deciphered from the size and the gradient of the shadow caused by the difference in angle between the sun azimuth and the normal wave surface. As the sun azimuth varies periodically the shadow gradient can change over time but its potential can still be approximated by experience of that beach and its visible height. Hence the method of detection must allow for a shadow with a variable light intensity and gradient of this type for all positions of the sun and profile of sand. To follow this criteria a kernel convolution has been used to detect the shadowed region.

The waves in the images on the left in figure 1.1 and figure 1.2 are unsurf-able as they lack the required height and shape. As well as this the peaks don't lie on a straight edge meaning a lot more energy is lost to turbulence. The detections shown in the right images in figure 1.1 and figure 1.2 contain wave matching this criteria.


(a) Not Surf-able: zero detections will be made in this image

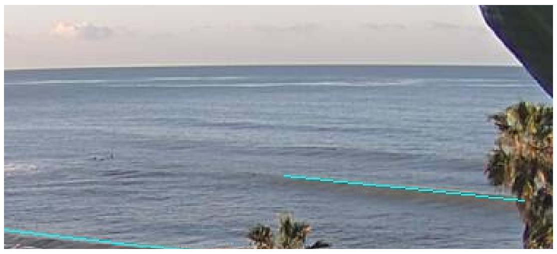

(b) Surf-able: the 2 waves marked will be detected in this image

Figure 1.1: Example images with no detection wanted on the left and detection wanted on the right

The other important factor that is required to allow for valid detections, in the interest of improving the accuracy of wave forecasting models, is wavelength. The wavelength is perceivable to the human eye as the human brain understands perspective and can predict the dimensions of objects. The algorithm can decipher these differences as smaller wavelengths tend to cause more remnant white water and disturb the shape of those behind when broken, as shown in figure 1.1. Using this we can filter these cases out by choosing only the output to the convolution that resembles a straight edge. This report later outlines how this is done using the Hough line algorithm in section 5.3.

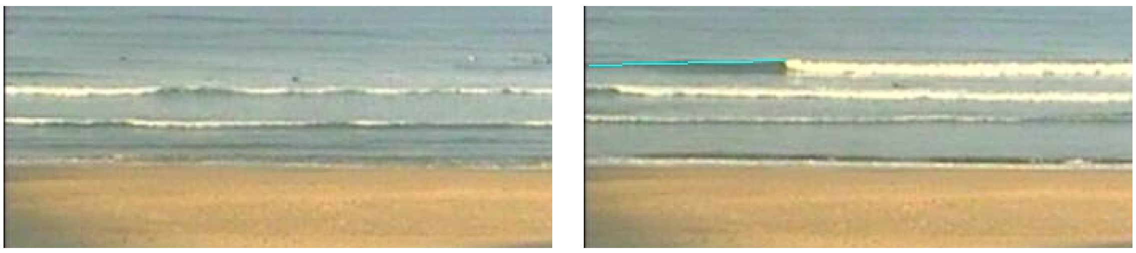
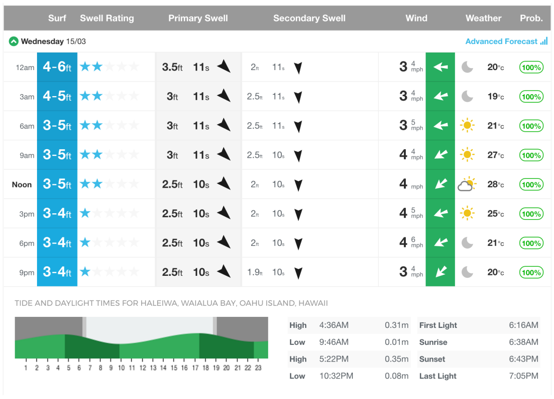

(a) Not Surf-able: zero detection will be made in this image (b) Surf-able: the wave marked will be detected in this image

Figure 1.2: Example images with no detection wanted on the left and detection wanted on the right

1.3 Swell Models

Both figures 1.3 and 1.4 show surf reports from different websites that report on surf conditions. The key feature highlighted on the sites show the important features in deciding whether the conditions are optimum or not. These features are the height, period and direction. The reports make clear both the primary wave propagation direction and the secondary propagation direction. The measurements are predicted from tidal buoys and interpolated across the coastline to allow for the estimation on any given beach. Predictions of the ocean temperature allow the model to adjust to the density of water as it will affect its speed, whilst external predictions on the wind strength help calculate the force in which the waves accelerate. The near shore predictions are based on satellite imagery used to understand the depth of the sand banks and help predict the near shore measurements for the period, height and speed of the wave. As no current method to validate the near shore predictions exist the reports in figures 1.3 and 1.4 are to aid the predictions of the surfer as the actual height and wavelength could vary by proportions similar to itself. The forecasting algorithm is far from perfect as the profile of the beach is not deducible from satellite imagery, furthermore the effect of cliffs, housing and headland on the local wind direction is very hard to predict.

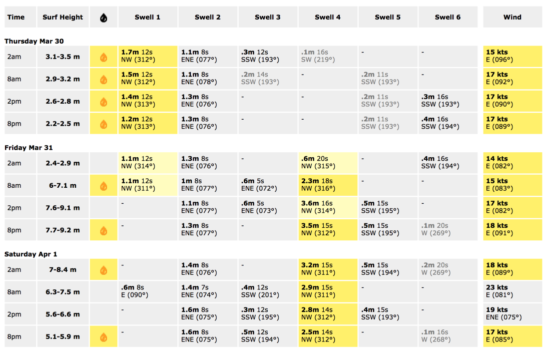


	Surf	Swell Rating	Primary Swell			Secondary Swell			Wind		Weather		Prob.
Wednesday 15/03													Advanced Forecast
12am	4-6ft	★★☆☆☆	3.5ft	11s	▲	2ft	11s	▼	3 4 mph	←	☾	20°c	100%
3am	4-5ft	★★☆☆☆	3ft	11s	▲	2.5ft	11s	▼	3 4 mph	←	☾	19°c	100%
6am	3-5ft	★★☆☆☆	3ft	11s	▲	2.5ft	11s	▼	3 5 mph	←	☀	21°c	100%
9am	3-5ft	★★☆☆☆	3ft	11s	▲	2.5ft	10s	▼	4 4 mph	↙	☀	27°c	100%
Noon	3-5ft	★★☆☆☆	2.5ft	10s	▲	2ft	10s	▼	4 mph	↙	⛅	28°c	100%
3pm	3-4ft	★☆☆☆☆	2.5ft	10s	▲	2ft	10s	▼	4 5 mph	←	☀	25°c	100%
6pm	3-4ft	★☆☆☆☆	2.5ft	10s	▲	2ft	10s	▼	4 6 mph	↙	☾	21°c	100%
9pm	3-4ft	★☆☆☆☆	2.5ft	10s	▲	1.9ft	10s	▼	3 4 mph	↙	☾	20°c	100%

TIDE AND DAYLIGHT TIMES FOR HALEIWA, WAIALUA BAY, OAHU ISLAND, HAWAII

High	4:36AM	0.31m	First Light	6:16AM
Low	9:46AM	0.01m	Sunrise	6:38AM
High	5:22PM	0.35m	Sunset	6:43PM
Low	10:32PM	0.08m	Last Light	7:05PM

Figure 1.3: Example Magicseaweed surf report as seen here [9]



Time	Surf Height	💧	Swell 1	Swell 2	Swell 3	Swell 4	Swell 5	Swell 6	Wind
Thursday Mar 30									
2am	3.1-3.5 m	●	1.7m 12s NW (312°)	1.1m 8s ENE (077°)	.3m 12s SSW (193°)	.1m 16s SW (219°)	-	-	15 kts E (096°)
8am	2.9-3.2 m	●	1.5m 12s NW (312°)	1.1m 8s ENE (078°)	.2m 14s SSW (193°)	-	.2m 11s SSW (193°)	-	17 kts E (092°)
2pm	2.6-2.8 m	●	1.4m 12s NW (313°)	1.3m 8s ENE (076°)	-	-	.2m 11s SSW (193°)	.3m 16s SSW (193°)	17 kts E (090°)
8pm	2.2-2.5 m	●	1.2m 12s NW (313°)	1.3m 8s ENE (076°)	-	-	.2m 11s SSW (193°)	.4m 16s SSW (194°)	17 kts E (089°)
Friday Mar 31									
2am	2.4-2.9 m		1.1m 12s NW (314°)	1.3m 8s ENE (076°)	-	.6m 20s NW (315°)	-	.4m 16s SSW (194°)	14 kts E (082°)
8am	6-7.1 m	●	1.1m 12s NW (311°)	1m 8s ENE (077°)	.6m 5s ENE (072°)	2.3m 18s NW (316°)	-	-	15 kts E (083°)
2pm	7.6-9.1 m		-	1.1m 8s ENE (077°)	.6m 5s ENE (073°)	3.6m 16s NW (314°)	.5m 15s SSW (195°)	-	17 kts E (082°)
8pm	7.7-9.2 m	●	-	1.3m 8s ENE (077°)	-	3.5m 15s NW (312°)	.5m 15s SSW (195°)	.1m 20s W (269°)	18 kts E (091°)
Saturday Apr 1									
2am	7-8.4 m	●	-	1.4m 8s ENE (076°)	-	3.2m 15s NW (311°)	.5m 15s SSW (194°)	.2m 20s W (269°)	18 kts E (089°)
8am	6.3-7.5 m		.6m 8s E (090°)	1.4m 7s ENE (074°)	.4m 12s SSW (201°)	2.9m 15s NW (311°)	-	-	23 kts E (081°)
2pm	5.6-6.6 m		-	1.6m 8s ENE (075°)	.3m 12s SSW (195°)	2.8m 14s NW (312°)	.4m 15s SSW (193°)	-	19 kts ENE (075°)
8pm	5.1-5.9 m	●	-	1.6m 8s ENE (075°)	.5m 12s SSW (194°)	2.5m 14s NW (312°)	-	.1m 16s W (268°)	17 kts E (085°)

Figure 1.4: Example Surfline surf report as seen here [12]

1.4 Project Work

This project provides the algorithms and calculations required to detect the wave-length, speed and height of a wave on an arbitrary, low cost, low resolution web camera positioned close enough to the beach for a person to make the above three deductions. The resulting system will be able to take any web camera of this description and extract the wave length, speed and height of every surf-able wave in each frame. This will then be used to investigate its use in improving the accuracy of modern forecasting models. Furthermore the system will be able to time stamp the detections which look most promising for surfers and add a probability to any given wave to allow the surfer to know the waiting time to a wave with the quoted features.

The system is designed to make use of how a wave evolves over time by looking at a time stack of images. Firstly by looking at how a time stack of frames can decipher the wave breaking region and direction. This will then go on to explain the use of a kernel convolution and Hough line algorithm in detection of the wave shadowed region in order to find its peak. Through which one should be able to evaluate and test the parameters in this algorithm before experimenting with clustering techniques to group partial wave detections together and evaluating the improved performance. Then going on to make further use of the time dimension to discard outlying waves that did not appear in the frames before or after. the algorithm will then use the detected surf-able wave peak lines to calculate the speed, wavelength and height.

Chapter 2

Background

2.1 Ocean Analysis Methods

Optical sensors have been used on the coast for a range of purposes from the surveillance of eroding coastline to analysing beach safety over the day. A considerable amount of work has been done in regards to time stack images. With a large amount of sampling and averaging one is able to remove irregularities leaving only what is wave induced. S.G.J Aarninkhof et al [1] were able to remove the contribution of remnant foam to the image's intensity by looking at the frequency of the contribution over time. Through this one then gains the ability to scale the intensity peaks at the breaking points of the wave to a more reasonable size, resembling rolling waves with no remnant foam persisting. They achieved this by first removing the background intensity approximated as the average leaving the remaining intensity resembling Gaussian distributions about the breaking positions of the waves. Based on the idea of the persistent foam appearing periodically and persisting with an exponential decay, the factor by which the intensity at time t was enhanced could then be computed using a time integral. Furthermore by using the width and position of the scaled down Gaussians, S.G.J Aarninkhof et al [1] were able to extract the average height and wavelength corresponding to that time stack. Further methods were developed by P.A Catalan et al [6] using a combination of optical and radar sensors. Aided by the method of scaling the image intensity as described in [1] they were able to partition zones of the wave evolution. They discovered that the evolution of the wave could be quantified to a specific row index in the image by isolating regions of intensity about the thresholds I^t for intensity and the σ_0^t for the intensity deviation.

Non Breaking	Breaking	Foam	Steep Waves
$I(x,y,t) < I^t$	$I(x,y,t) \leq I^t$	$I(x,y,t) \leq I^t$	$I(x,y,t) < I^t$
$\sigma_O(x,y,t) < \sigma_0^t$	$\sigma_O(x,y,t) \leq \sigma_0^t$	$\sigma_O(x,y,t) < \sigma_0^t$	$\sigma_O(x,y,t) \leq \sigma_0^t$

P.A Catalan et al [6] then calculated the dissipation of the wave by analyzing the time taken for the evolution. Efforts at running particular analysis over several waves at the same stage in its evolution was later trialled by R Almar et al [2]. Using a wave machine, efforts to pinpoint the break-point position as it moves across the face of the wave gave an insight into extracting a more reliable estimate of the height of the wave at breaking point. Using a running average time stack image across a single time period, they grouped the pixels into a proximity by threshold and marked the breaking zone for each column x_b, t_b. They then took the standard deviation of the intensity peak at t_b over the row numbers and computed its width L by fitting a Gaussian to the data. They then used geometry to calculate the physical wave height H_b where the parameters a_b and β change with the position of the breaking wave.

$$H_b = (L - \frac{L}{tan(a_b)}).tan(\beta)$$

This gave rise to some long term wave height monitoring software [7]. They used a similar method to that of R Almar et al [2] on live coastal conditions to monitor the near shore wave height. They first pre-processed the time stack of images by reducing the color channels to one single channel, then selected the columns making up the lowest 10% of the overall variance and normalized each one.

$$I(t, v) = 0.35R(t, v) + 0.5G(t, v) + 0.15B(t, v)$$

They performed breaking zone detection by constructing two secondary images. The first was a collective sum of the intensity of that pixel over time divided by the largest overall sum of intensity over time for a pixel in that column. And the second image was the difference in the 95^{th} and 5^{th} percentile of the pixel intensity over time divided by the maximum pixel intensity over time chosen from all its pixels intensities that were in the top 20^{th} percentile. The algorithm then finds the top and bottom of the breaking zone by decrementing the top boundary down until the two secondary images stop increasing and decrementing the bottom boundary until the second secondary image is half as bright as the first. The next part of the algorithm applies a convolution and thresholds the image as such to maximize the entropy within and removed. Finally using N8 connectivity they constructed a path from the top to the bottom boundary along the intensities left after the thresholding which obtained a horizontal location to the breaking edge of the wave. They then computed the wave height using a similar geometric method to [2] outlined in 7.

2.2 Swell Models

As outlined in [5] waves are modeled using a wave energy scalar field E(f) where f is the frequency. The energy density spectrum can be modeled by the combined components of 2 Fourier series as the surface is in 2 dimensions where T is the time period and ϕ is the surface elevation.

$$E(f) = \lim_{\Delta f \to 0} \frac{1}{\Delta f} E\{\frac{1}{2}a^2\} \quad a_i = \sqrt{A_i^2 + B_i^2}$$

$$A_i = \frac{2}{T} \int \phi(t) cos(2\pi f_i t) dt \quad B_i = \frac{2}{T} \int \phi(t) sin(2\pi f_i t) dt$$

The significant wave height is the measurements used in surf reports and is equal to the largest height of 3 waves. To generate the expected wave height we can integrate E(f) across all possible frequencies f_i to obtain the mean of the amplitude in frequency space. For any Gaussian and its Fourier transform the mean and standard derivation swap roles so this value equates to the standard derivation of the amplitude in time. Therefore the height of the wave is directly proportional to the standard derivation of the surface elevation with a coefficient of 4 as its max height is twice its amplitude [14].

$$H = 4\sigma_\phi$$

Chapter 3

Data Acquisition

3.1 Methodology

For this investigation 16 web cameras were used, showing a variety of beaches often used by surfers, shown in figure 3.2. The majority of cameras use IP camera software and therefore could be streamed straight into python. This was done using **urllib2** to open a network connection to the motion jpeg at that url. The stream could then be read at 1024 bytes at a time continuously iterating over the whole motion jpeg. Once the endpoints of the serial message were found the **numpy fromstring** method was used to decode the serial message into a 3D array. The algorithm speed is predominantly affected by the opening of the browser reader and therefore each camera was run in parallel so the opener only has to run once per camera. The code for this process is shown in figure 3.1. Another method used to collect data was to successively print screen a browser as the camera was playing. This process was implemented on a raspberry pi using **x11grab** to capture the screen output and **ffserver** to forward the stream again. To start ffserver was used `ffserver -d /ffserver/ffserver.conf` in one screen session. The default config file and set accordingly as was the width and height of the camera. To stream my own display I used **alsa** for conversion to a motion jpeg and set the frame rate and frame size according with this command in another screen session. the command was `ffmpeg -f alsa -i pulse -f x11grab -r 25 -s 1024x768 -i http://localhost:80/pipeline.ffm`.

3.2 Data Sources

The web cameras used in the project are shown in figure 3.2 along with their name and resolution taken by converting the image to a **numpy** array and calculating the shape. The locations and source of each web camera used can be found in appendix entries 1 and 2.

```
# Opens connection to the webcam
opener = urllib2.build_opener()
if referer:
        opener.addheaders = [("Referer",referer)]
stream=opener.open(url)
bytes=''
# Unserialises the motion jpeg stream
while True:
    bytes+=stream.read(1024)
    a = bytes.find('\xff\xd8')
    b = bytes.find('\xff\xd9')
    if a!=-1 and b!=-1:
        jpg = bytes[a:b+2]
        bytes= bytes[b+2:]
        fromstr = np.fromstring(jpg, dtype=np.uint8)
        frame = cv2.imdecode(fromstr,cv2.IMREAD_COLOR)
        yield frame
```

Figure 3.1: Code used to extract web camera feed to a numpy array from [16]

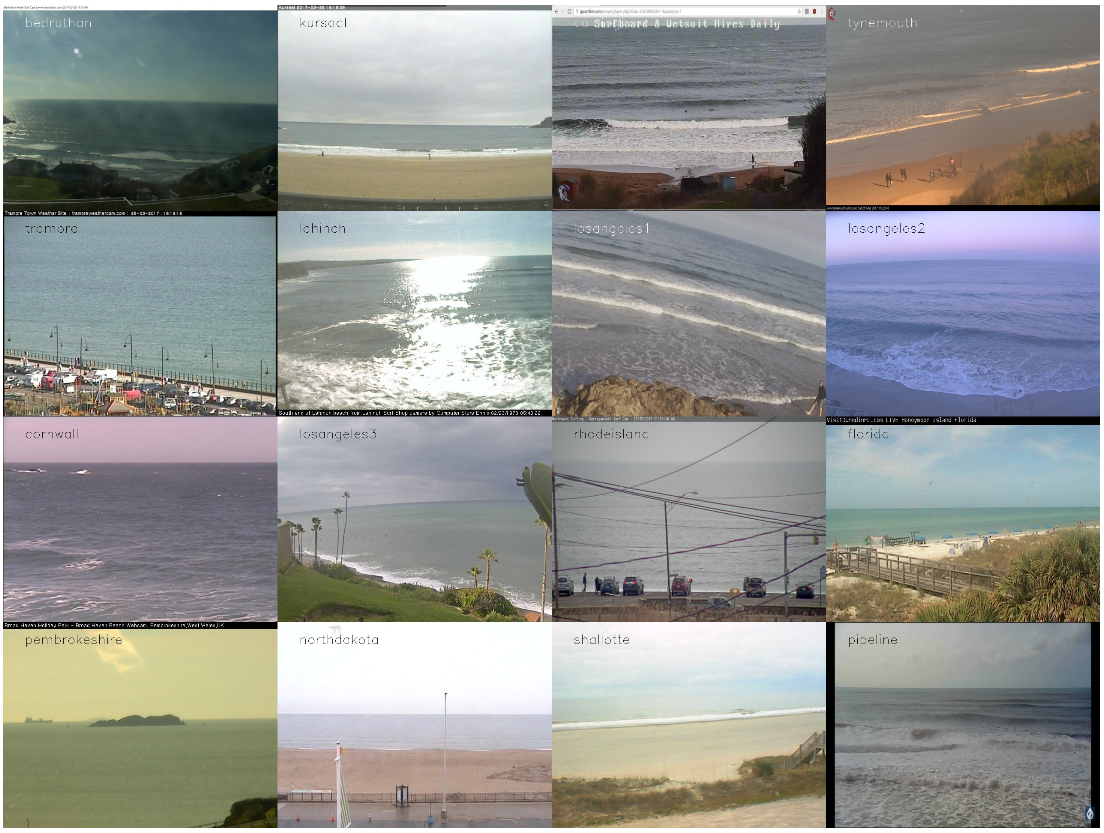


Figure 3.2: Randomly sampled image from each camera

Bedruthan	**Kursaal**	**Coldingham**	**Tynemouth**
1280 × 960	768 × 576	1280 × 720	1280 × 720
Tramore	**Lahinch**	**Los Angeles 1**	**Los Angeles 2**
768 × 576	640 × 480	640 × 400	1920 × 1080
Cornwall	**Los Angeles 3**	**Rhode Island**	**Florida**
640 × 480	800 × 450	1920 × 1080	704 × 480
Penbrokeshire	**Northdakota**	**Shallotte**	**Pipeline**
640 × 480	704 × 480	640 × 480	1280 × 720

Figure 3.3: The key for figure 3.2 with name and resolution of each camera

The detection will be subject to the following difficulties. Firstly the region in which detections would want to occur lie in different positions and different orientations for all web cameras. The algorithm would therefore be required to detect a region of interest before any detection of a wave can be made. This would entail looking for the area in which surf-able waves occur and confining the search to that region. Secondly the lense on each web camera provides different lighting conditions and therefore causes the shade of water to make up a large proportion in most color spaces. Therefore the region of interest for detection must occur independently of the color and detection methods must work on a gray scale image.

The majority of the waves in figure 3.2 are subject to good weather conditions whereas on this particular day beaches `Lahinch` and `Bedruthan` had poor weather conditions. This is visible by the abundance of white water and very limited amount of unbroken waves. Therefore the algorithm cannot be sensitive to large intensities as it must be able to detect a surf-able wave given it exists whether surrounded by white water or not as seen in figure 3.4.

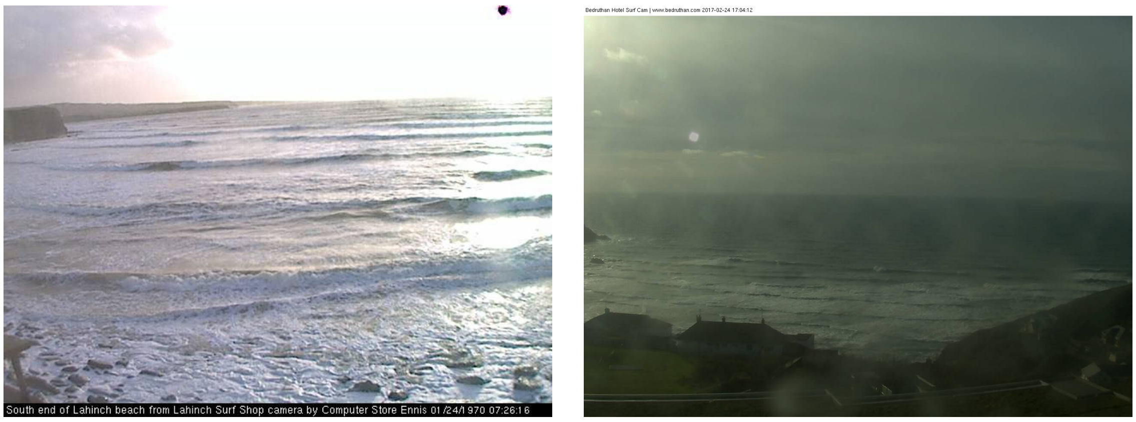

Bedruthan Hotel Surf Cam | www.bedruthan.com 2017-02-24 17:04:12
South end of Lahinch beach from Lahinch Surf Shop camera by Computer Store Ennis 01/24/1970 07:26:16

Figure 3.4: Examples of conditions where detection will be difficult as the ocean is covered in remnant foam

As all previous attempts at a tracking algorithm have required the intensity to be above a certain threshold it seems implausible to take measurements after sundown giving only a small 8-10 hour window to collect data. Also each camera allows for different resolutions so the kernel used must be invariant of variations in pixel sizes for each beaches wave.

A large amount of research on the tracking of ocean waves has focused on analyzing time stack images rather than still images to remove ambiguities by averaging over time. The time period and wave speed have been abstracted using a time stack with a frame rate of 2 frames per second [6, p.1882] with additional use of microwave measuring instruments. Most recently period, speed and also breaking height have been extracted using a time stack on live conditions at a frame rate of 15-30 frames per second [7, p.3413]. As the cameras in use over this project stream over the internet, scalable analysis of a time period of images would not be as granular as the data used in the methods above. Therefore a method of extracting the wave features must work with an arbitrary frame rate.

3.3 Algorithm Requirements

As outlined in section 1.2

- The detection algorithm is biased to the size of the wave and shape of the shadow created by the difference in angle between the water surface and the sun azimuth, to only detect surf-able waves with features above the size and curvature thresholds. These thresholds will be trained through exhaustive testing of labeled data where only marked waves that appear large enough and have enough power are detected.
- The detection algorithm will also focus on surf-able waves by detecting only the most triangular of waves in the plane of movement. This will be visible by a sharp edge at their top seen as a straight line in the camera plane of view.

As outlined in section 1.3

- The detection algorithm will output the height, wavelength, count, speed and time stamp of every surf-able wave in each frame.

As outlined in section 3.2

- The detection algorithm will be invariant of the distance between the camera and the water if within the maximum distance of 400 meters exhibited in the **Bedruthan** camera.
- The detection algorithm will be invariant to image color.
- The detection algorithm will be invariant to sunlight on the condition that the sun is always up when the algorithm is running.
- The detection algorithm will be invariant to the time difference between frames as to cope with the randomness in the network traffic.

- The detection algorithm will be invariant to the abundance of remnant white water given the detected waves follow all the previous conditions and have been detected before they have broken.
- The detection algorithm must be invariant to resolution so that it can work on any arbitrary web camera.

Chapter 4

Break Zone Detection

4.1 Overview

In this section we are interested in detecting the area in which waves are visible otherwise known as the break-zone. In order to detect only breaking waves each webcam image must be treated to remove the objects surrounding the sea that could be incorrectly detected as a wave allowing for more accurate results. Due to the movement of the tide the lower half of the removal of the beach would have to be variable and adjust over time. Furthermore a applying a stationary binary mask [16] would not work in the case of a moving camera such as the camera in *Cornwall* and *Kursaal*. The final algorithm adjusts to these problems and obtains the upper limit and lower limit of the break zone boundary in the y axis for every ten frames. As the ocean will be the most active thing in the image we can use the contrast between adjacent frames to our advantage as explained in the next section.

4.2 Breaking Direction

In order to calculate the break zone perfectly the dominant vector in the direction that the incoming waves move in is required. This value influences how the detected waves are clustered later on on the detection process but is also used to orient the break zone boundary. The algorithm uses a series of frames to calculate the cross correlation between the series of frames columns and rows. The cross-correlation between adjacent frame rows give the x component of the direction of movement and the same method on the columns give the y. Letting x be the row in the preceding frame and y be the current frame row the cross correlation is a measure of similarity between these two series and is defined by the formula below:

$$crosscorrelation = \int_{-\infty}^{\infty} x^*(\tau)y(t+\tau)d\tau = x^*(-t) * y(t)$$

As the product of the series and its complex conjugate is its magnitude the maximum of this distribution will be at the value of τ in which x and y are most similar which is the phase shift in the row. The following calculation can be done using a fast Fourier transform as a convolution in real space is a multiplication in Fourier space.

$$x * y^* = \mathcal{F}^{-1}\{\mathcal{F}\{x\} \times \mathcal{F}\{y\}^*\}$$

This calculation was done using the **opencv** method **phaseCorrelate** which gave the size of the phase change in x and in y. I then normalized the phase change as a vector to give me the dominant direction of travel for the waves.

4.3 Break Zone Boundaries

To allow the threshold boundaries to lie directly above and below the breaking region the break zone detection algorithm used implemented by R Almar et al [7]. For this method a time stack of ten images where used, five before the current image and four that were taken after. Firstly the time stack was converted to gray scale using the ratio quoted in the paper shown bellow.

$$I(t, v) = 0.35R(t, v) + 0.5G(t, v) + 0.15B(t, v)$$

The current image was normalized over the time dimension so that every pixel added up to one over time. This has the affect of emphasizing the active features in the image and normalizing the stationary ones. To further blur any stationary areas in the image the **openCV** method **Gaussian blur** was applied. The code for the operation is shown below. A sample of normalized images are shown in figure 4.1 for three different beaches.

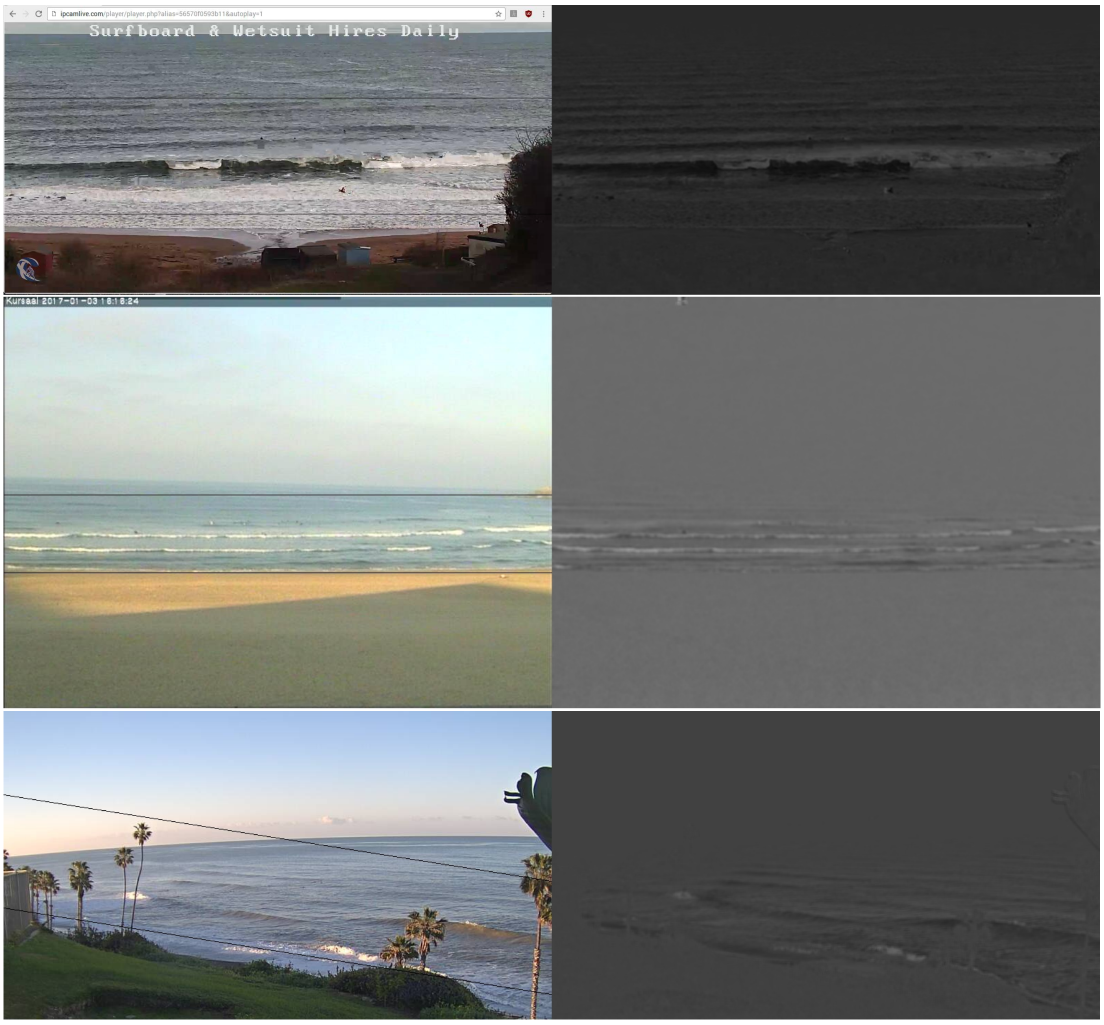


Figure 4.1: The break zone region and the normalized images plotted for 3 cameras

The next step used by R Almar et al [7] is construction using 2 more images $f1$ and $f2$ from this normalization.

$$f1(x,y) = \frac{\sum_t I(t,x,y)}{max_t\{\sum_t I(t,x,y)\}}$$

$$f2(x,y) = \frac{percent_{95}\{I(t,x,y)\} - percent_5\{I(t,x,y)\}}{max_t\{I(t,x,y)|I(t,x,y) > percent_{95}\{I(t,x,y)\}\}}$$

The figure 4.2 shows how the algorithm has been adapted to work on multiple beaches with a variety of wave directions. The original algorithm identifies the upper boundary progressively higher in the image allowing for the intensity to be darker in both intermediate images marking the point that activity ceases. The lower boundary is moved progressively higher till the $f1$ image pixel value is twice the size as the $f2$ image pixel value. This identifies the starting of breaking white water. The code for this is shown in figure 4.2.

```python
# The initial predictions for the starting of noticeable waves v0
# and the ending of the white water region v1
v0 = np.argmin(f2[f2 > TH], axis=0)
v1 = np.argmax(f1[f1 > TH], axis=0)

# Iterate downward to the first wave crest
while change(v0):
  v0-= np.all([f1[v0] > f1[v0-1],f2[v0] > f2[v0-1]])

# Iterate downward to where the white water stops
while change(v0):
  v1 -= (f1[v1]/f2[v1] > 2)

return np.average(v0), np.average(v1)
```

Figure 4.2: Code for finding the break zone region

In addition to the method in R Almar et als [7] the algorithm in figure 4.2 returns the boundary values as an average over each column. Assumes that this boundary exists in the center of the image therefore it is rotated about the center to the vector perpendicular to the motion of the sea. Below is the output for the algorithm on every web camera with the angle to the vector perpendicular to the movement of the waves and the upper and lower boundary of y.

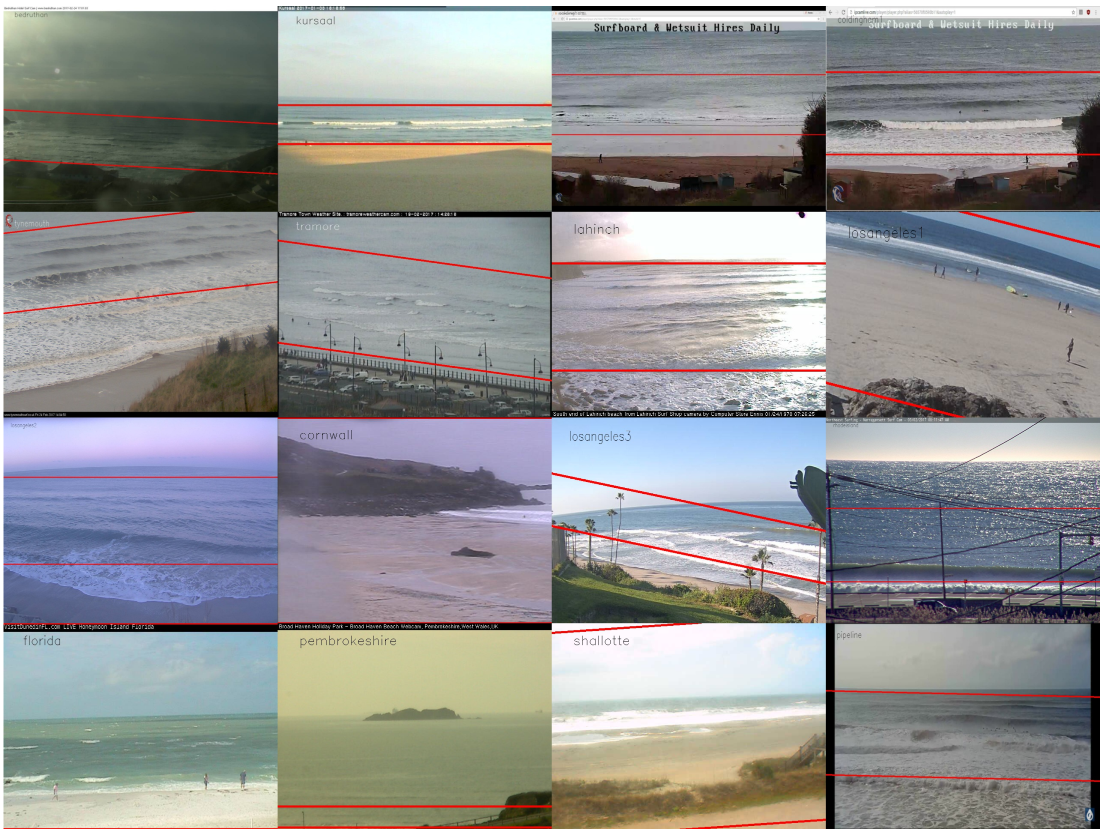


Figure 4.3: Break zone calculated and plotted for each web camera

4.4 Results

bedruthan	**kursaal**	**coldingham**	**coldingham1**
0.0523598775598	0	0	0
517	277	400	200
750	386	750	450
tynemouth	**tramore**	**lahinch**	**losangeles1**
−0.0872664625997	0.13962634016	0	0.209439510239
20	133	120	0
300	419	370	`max`
losangeles2	**cornwall**	**losangeles3**	**rhodeisland**
0	0	0.157079632679	0
313	0	185	477
770	`max`	300	862
florida	**pembrokeshire**	**shallotte**	**pipeline**
0	0	−0.0698131700798	0.0174532925199
0	427	0	247
`max`	476	`max`	541

Figure 4.4: The key for figure 4.3 with the result for $(tan(\theta)$ to the vector perpendicular to the movement of the wave, upper boundary for the break zone in the y dimension, lower boundary for the break zone in the y dimension)

North Dakota has been excluded from the results shown above in figure 4.4 as its dominant direction could not be calculated due to poor resolution and damage to the lens. The break zone region detection was only correct in 10/15 cases due to low resolution web cameras and minimal ocean activity forcing the break zone region to be over estimated. Although **Cornwall** , **Shallotte** and **Pembrokeshire** were misclassified there were waves on these beaches that followed the detection criteria and therefore the resolution of the output made it impossible to continue experimentation with these web cameras. Therefore, the cameras where the break zones were incorrectly positioned were discarded from the experimental data set from this point onwards.

Chapter 5

Crest Detection

5.1 Overview

In the interest of extracting the speed and wavelength of a wave the same exact point must be identified on each wave to represent an accurate representation for its position. The top edge of the wave was used as this position reference as either side of this mark there are well defined patterns that could be detected by applying a specific kernel shape. The front side of the crest exhibited a gradient of dark to light moving away from the top edge. On the opposite side of the top edge was a sharp change in intensity as the dark shadowed area became just the water behind. As variations of this pattern were universal for all the web cameras used it would be beneficial to use kernel convolution for detection.

5.2 Kernel Convolution

The first step was to normalize the variety of light intensities across the cameras as this could cause results to be skewed. Therefore, all images were converted to gray scale using the same ratio from the previous chapter. The convolution between the image and each kernel was then used to construct an edge plot of the locations of the maximum turning points of the smoothed convolution output. For any input image c the output image v will be constructed using the following formula where $m(t)$ is the kernel.

$$v(i,j) = \sum_{h=-2}^{2} \sum_{t=-\frac{n}{2}}^{\frac{n}{2}} c(j+h, i+t)m(t)$$

The process of choosing the correct kernel involved sampling cropped images of waves from all the beaches and by then evaluating their accuracy of wave detection by eye. As the aim was to detect surf-able waves the detection methods used were subject primarily to occasions where the ocean wave is stretched to the limit of its surface tension. With limited surface disturbance by wind and other external factors which narrowed down the cropped wave shapes I could be used. The major three shapes chosen are shown in figure 5.1. In addition to these three a Sobel kernel [15] which resembled the second half of *kernelA* was used also.

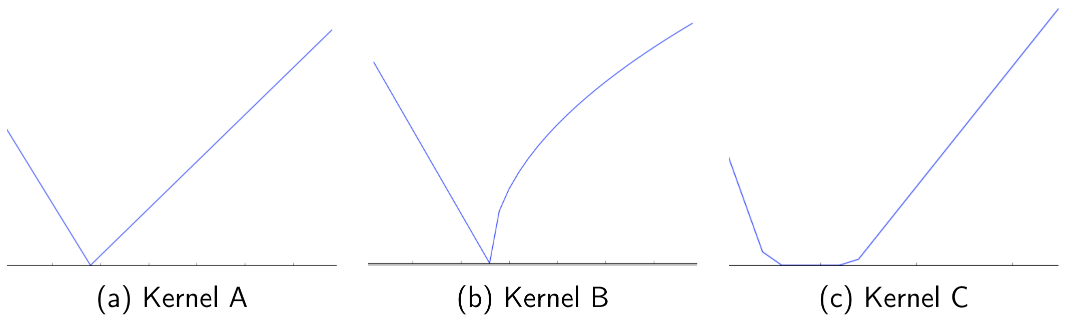


Figure 5.1: The major 3 kernel shapes and the snapshots they were derived from

After the convolution was obtained the **ndimage** method **Gaussian_filter** [11] was used as a low pass filter to smooth convolution output to resemble a an array of Gaussian centered on wave peaks. The **ndimage** method works by taking a convolution with any order derivative of a 2d Gaussian kernel. The kernel for this obeys this formula where σ_x and σ_y are inputs to the function.

$$f(x,y) = \frac{1}{2\pi\sigma_x\sigma_y} \exp^{-(\frac{(x-\overline{x})^2}{2\sigma_x^2}+\frac{(y-\overline{y})^2}{2\sigma_y^2})}$$

This had the effect of smoothing the active regions in the convolution image into more defined gradually increasing and decreasing peaks with less noise. The output is shown in figure 5.2.

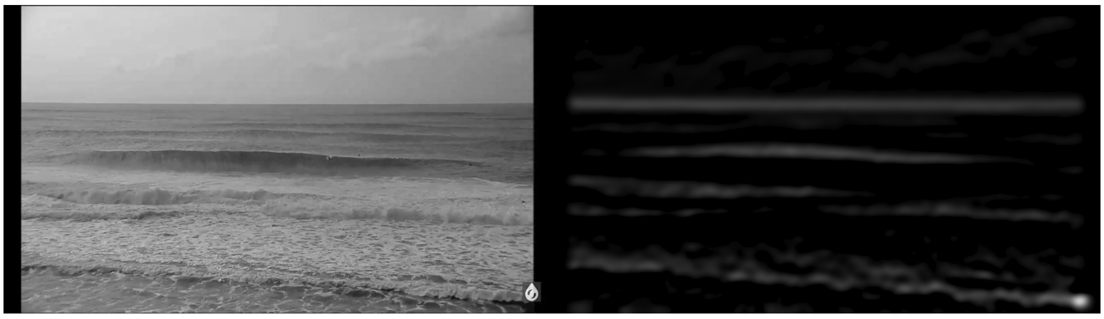


Figure 5.2: The output from the kernel convolution on the `Pipeline` web camera

Next the algorithm infers the peak of each potential wave and plots it onto a binary image of these edges. The contribution to every kernel is added to this image in order to attain the entire waves peak. This is done by calculating the maximum turning points of each column that lie above a *threshold* value. Each turning point is then plotted on the binary edge plot at its corresponding location. Due to the nature of the convolution method the highest output lies at the center of the kernel. In some cases this requires a detection which is offset from the center of the chosen kernel as the aim is to detect the wave peak. This is accounted for when the edges are drawn by applying that offset, the resulting edge plotted is shown in figure 5.3

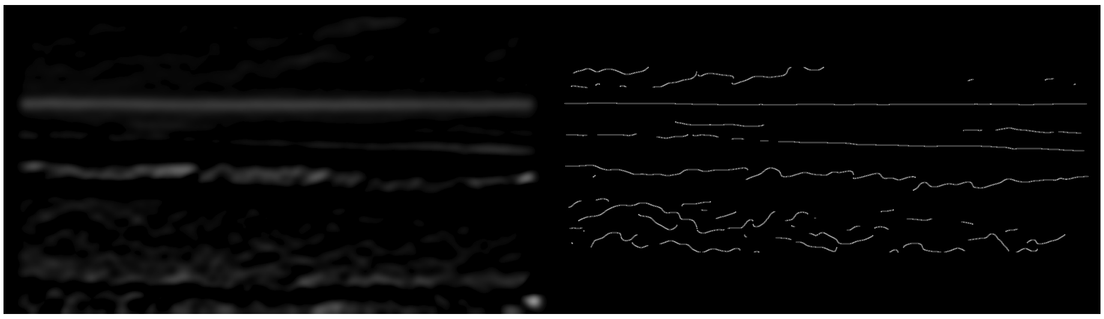


Figure 5.3: The image of edges on the right plotted from the maximum turning points of each column in the convolution image on the left

5.3 Hough line Algorithm

Following on from the image of edges calculated in section 5.2 the Hough line algorithm is applied to the image of edges to uncover a potential wave peak. The Hough line algorithm is more likely to detect straight edges than deformed ones therefore, the condition for the algorithm that the wave lacks turbulent flow stated in section 1.2 is met. The output is ordered from longest to shortest and filters by a minimum length parameter. The tuning of the hough line parameters are outlined in the next section. The lines are also filtered to lie within the break zone boundary as calculated in section 4.

The Hough line algorithm is used for detecting straight lines in binary plots. As described in [10] this works by representing a line as a point in a new vector space which hence fourth will be referred to as *linespace*. In this instance the *linespace* used was ρ by θ coordinates where θ and ρ are the polar coordinates of the vector which is both a position vector and a normal vector of the line. As a line is represented as a point in *linespace* equally a point is represented as a line in *linespace*, therefore a line passing through two points is equivalent to an intersection of two lines in *linespace*. Due to this the confidence of any given line existing will be equal to the number of times the point representing that line is intersected. The confidence values for each line can be represented by a matrix incremented every time a point is plotted at the ρ and θ which corresponds to two or more pixels lining up in the original image. The output lines are just the indices with values above the confidence threshold.

Lastly the vector perpendicular to the wave trajectory calculated in section 4 was used to check to see whether the orientation values for each wave had any outliers. Using a parameter for the number of multiples of the standard deviation

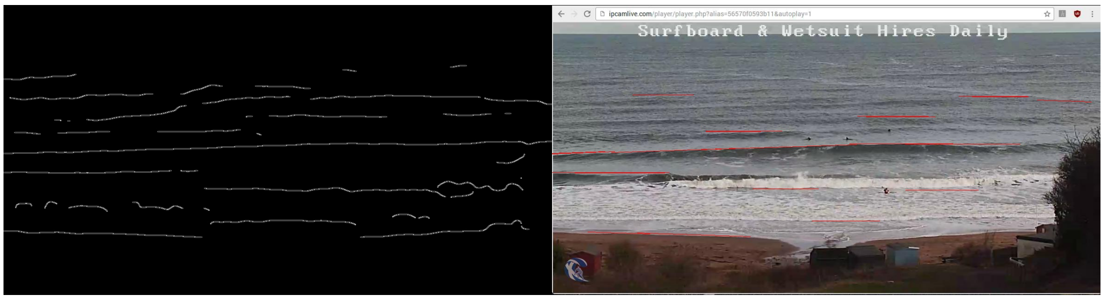


Figure 5.4: The output detections on the original image on the right with the image of edges the detections were calculated from on the left

each orientation can be away from the perpendicular vector to the wave trajectory, the resultant Hough lines are filtered. The top twenty results from the algorithm are drawn in figure 5.4 in red, ordered from most likely to be a wave to least likely to be a wave. This was done using the **houghlinesP** method in **opencv** which confines its search algorithm with two additional parameters: minimum line length and maximum line gap. The values of these parameters used will be outlined in section 6.4.

It is clear from 5.4 where the wave crests have been missed as the edges are too fragmented in places and do not resemble a straight line. The Hough line algorithm parameters were later optimized to ensure the edge plot uncovered every surf-able wave.

5.4 Clustering

To improve results, as the probabilistic houghline algorithm only returns line segments of the wave and could not detect the complete wave peak, a clustering algorithm was used to append segments associated with the same wave together. A novel method was developed that used a kernel density distribution for unsupervised clustering from Adelchi Azzalini et al [3]. This method was adopted to model each single wave as a Gaussian distribution and cluster the detected segments. This method is based on the assumption that the convolution outputs a Gaussian as the kernel will output its highest value at perfect alignment with the wave and increasing smaller values as the kernel is offset.

The kernel density estimator outlined in [8] works by constructing a continuous density function $\widehat{f}(x)$ built from a discrete number of Gaussians to match an input series $f(x)$. If $\widehat{f}(x) = \frac{1}{2nh}\sum_{j=1}^{n} I\{x-h < X_j < x+h\}$ is the probability that a wave peak is centered on point X_j. Using a Gaussian kernel $K = \frac{1}{\tau\sqrt{2\pi}}\exp^{\frac{-x^2}{2\tau^2}}$ with a fixed band width τ the density $\widehat{f}$ becomes $\frac{1}{n}\sum_{j=1}\frac{1}{h}K(\frac{x-X_i}{h})$.

The algorithm then randomly trials different value for X_j and n such that the mean square error is minimized. The mean square error is given by:

$$MSE(\widehat{f}(x)) = E\left[\int (\widehat{f}_h(x) - f(x))^2\right] = Var(\widehat{f}(x)) + |f(x) - K * f(x)|^2$$

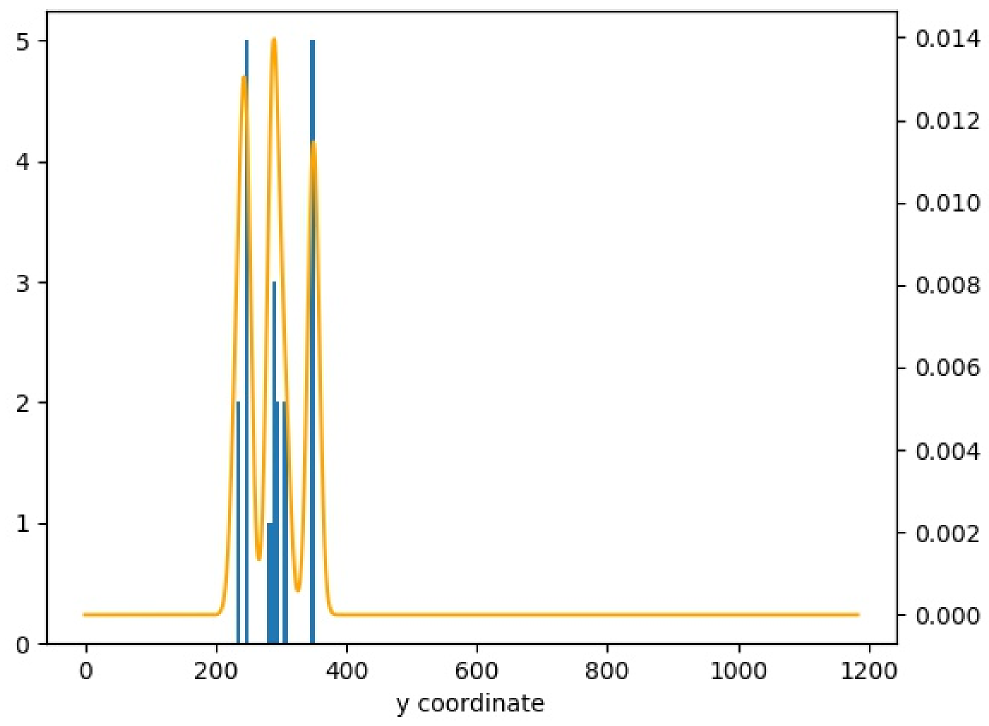


Figure 5.5: Figure showing how the kernel density estimator (in orange) fits to the detected line segments (histogram in blue)

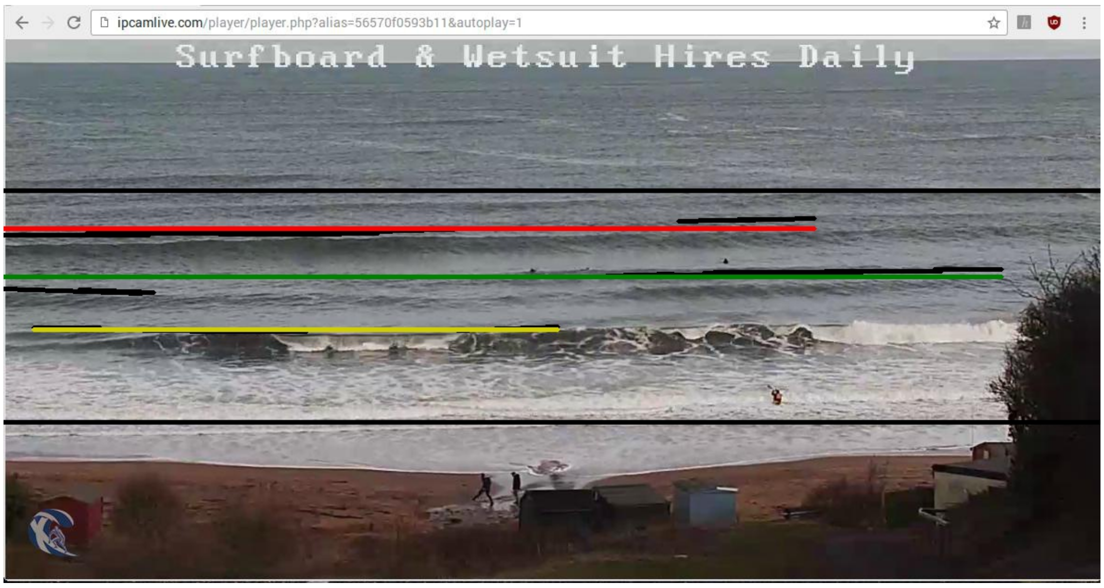

ipcamlive.com/player/player.php?alias=56570f0593b11&autoplay=1
Surfboard & Wetsuit Hires Daily

Figure 5.6: Image of the clustering output where the original detected segments are in black and the single clustered center lines are coloured separately

The clustering method is unsupervised so it relies on a constant *bandwidth* at which to set the kernels. A good estimate for the parameter τ is $(\frac{4\sigma^5}{3n})^{1/5}$ where σ is the variance of the sample and n is the number of modes. As n is also unknown an estimation for the total number of waves which could be incorrect must be made. After the clustering is complete a minimum cost function filters clusters with too few detections to aid the choosing of clusters that reacted to the kernel the most.

In figure 5.6 the black line at top and bottom of the image corresponds to the boundary of the break zone calculated in 4. The colored centroid for each cluster of detections do match the wave perfectly due to the fitting errors in the clustering algorithm as the kernel convolution has reacted more in one horizontal region of the wave but not as much in others. This clustering method would reduce our false detections however may also reduce our true positives over the whole dataset.

The algorithm for the method and seen in figure 5.8 first rotates the detections by the angle between the perpendicular to the trajectory of the wave and the x axis. This operation is shown in figurerotation.

$$W \begin{pmatrix} cos(\theta) & -sin(\theta) & 0 & 0 \\ sin(\theta) & cos(\theta) & 0 & 0 \\ 0 & 0 & cos(\theta) & -sin(\theta) \\ 0 & 0 & sin(\theta) & cos(\theta) \end{pmatrix} = W'$$

$$W = original\ list\ of\ detections\ as\ coded\ by\ their\ endpoints$$

$$W' = list\ of\ detections\ relative\ and\ separable\ by\ their\ y\ values$$

Figure 5.7: Equation by which the detected segments are rotated parallel to the x axis so that clustering can be done in 1 dimension

Once rotated the detections are separable by their y value the kernel density estimator is applied over the y dimension. The minimum turning points of the density function are then estimated using the derivative and each list of detections output in discrete clusters corresponding to the detections contained within a single Gaussian kernel.

```
# detected lines are rotated to be parallel to the x axis
theta = dominantdirection
c, s = np.cos(theta), np.sin(theta)
r = np.matrix([[c, -s, 0, 0], [s, c, 0, 0], [0, 0, c, -s], [0, 0, s, c]])
a = np.array(np.squeeze(np.dot(a, r).astype("int")))

# detected lines are ordered from the top of the image to the bottom
a = a[a[:, 1].argsort()]

# The density distribution is calculated over the y dimension
# and the turning points are found
density = gaussian_kde(a[:, 1], bw_method=bandwidth)
xs = np.linspace(0, width, width)
dx = np.diff(density(xs))
dx2 = np.diff(dx)
result = np.argwhere(dx[1:] * dx[:-1] < 0)

# The array of lines are split into each mode of the kernel
# density estimation function
for p in result:
 if dx2[p] > 0:
  gr = a[a[:, 1] < p]
  a = a[gr.shape[0]:]
  if len(gr) >= tomakeawave:
   output.append(gr)
if len(a) >= tomakeawave:
 output.append(a)

# Each cluster is averaged to calculate a centroid line
for cluster in output:
 ypos = np.mean([np.mean(cluster[:, 1]), np.mean(cluster[:, 3])])
 wave = np.array([np.min(cluster[:,0]), ypos, np.max(cluster[:,2]), ypos])

 # detected lines are rotated back into it original orientation
 c, s = np.cos(-theta), np.sin(-theta)
 r = np.matrix([[c, -s, 0, 0], [s, c, 0, 0], [0, 0, c, -s], [0, 0, s, c]])
 wave = np.array(np.squeeze(np.dot(wave, r).astype("int")))
 yield np.append(wave[0],id)
```

Figure 5.8: The algorithm for clustering the detection segments using a kernel density estimator function and the rotation operation in figure 5.7

Chapter 6

Evaluation

The results for the overall detection algorithm on a subset of the web cameras is presented in this chapter. The results will outline both the accuracy over every training set and over the individual beaches. An accuracy value used to obtain the accuracy for each beach will be recorded before and after the optimization for each group of dependent parameters. A subset of parameter optimizations paired with images of the resultant detections will also be given in the evaluation.

The chapter discusses the optimum values for all the parameters mentioned in the: convolution process in section 5.2, the line detection process in section 5.3, the clustering process in section 5.4.

6.1 Testing implementation

The testing method uses a set of labeled images to compare the position of the detected peak of the wave against a manually annotated set of waves The images were labeled by hand through clicking on the two end points of each surf-able region of wave such that the segment it forms is along the wave crest. A surf-able region follows all the criteria stated in section 1.2. Regions greater than the maximum endpoint and smaller than the minimum endpoint were deemed not surf-able in all cases except when these points lie at the edges of the image frame. This could be due to several reasons highlighted in section 1.2. Table 6.1 identifies the beach and day that the labeled images were taken. The days and the beaches used were chosen as these days contained surf-able waves and allowed for a large amount of labeling. The image data sets were recorded as a single time series up to sizes of 4000 images. The twenty labeled images were picked randomly from the entire set for that beach on that day.

Beach	**# Labeled images per day**	**# Days**
Coldingham	20	2
Kursaal	20	1
Tramore	20	1
Tynemouth	20	2
Losangeles2	20	2
Losangeles3	20	2
Pipeline	20	1
Rhodeisland	20	1
TOTAL	240	

Figure 6.1: Table to show the number of labeled images per day for each beach and the number of days it was taken on

6.2 Evaluation Criteria

The following definitions were used for the calculation of the false positives and false negatives:

TP = A detection matched with the wave and chosen out of all possible matches to identify that ground truth wave

FN = |ground truth waves| - TP

FP = |detection waves| - TP

In order to compare different kernels I created curves using a range of detection thresholds and match thresholds.

Detection Threshold = The maximum number of detections allowed in the analysis of the image in order of most likely a wave to least likely a wave.

Match Threshold = The perpendicular distance threshold inside which a detected wave matches to a ground truth wave.

To decide on a *match threshold* the images of different match thresholds were compared to see which were intuitively a correct detection. Leading from the discovery that any match threshold larger than 25 was independent of the actual wave being there, therefore the following range of match thresholds where used: [0 : 25 : 5]. To decide on the *detection threshold* the data was scanned for examples where several detections needed to be made in order to find all the ground truth waves. In rare cases this value was as large as 12 detections until the resultant true positives converged so the detection threshold was varied in the range of [0 : 12 : 1].

The true positives were counted by taking the maximal possible matching of the ground truth waves and the detected waves. This ensured that every ground truth wave had at most one match and that the true positive value is the best it could possible be. To construct the bipartite graph the following boolean tests were used to construct the edges. Where ϕ is the match threshold and L the length of a ground truth wave the boolean tests are the following:

$$test_1 = d_{i\ \parallel} - \phi < L \wedge d'_{i\ \parallel} + \phi > 0 \qquad test_2 = d_{i\ \perp} < \phi \wedge d'_{i\ \perp} < \phi$$

The dimensions involved are shown in 6.2 where the detection on the left $d1$ will pass both tests whereas the detection on the right $d2$ will only pass $test_2$.

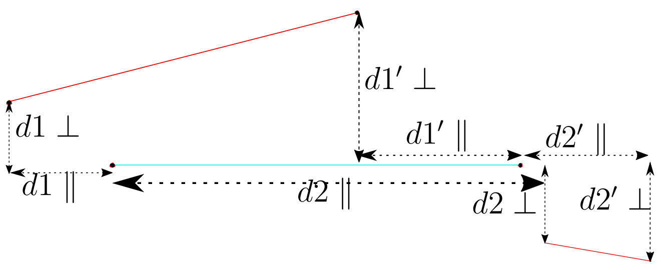


Figure 6.2: The detection test dimensions on the ground truth wave shown in blue and the detected waves shown in red

The above method measures the accuracy of the algorithm at detection of surfable waves. In order the calculate the features we require the following detections and requirements:

Physical Feature	**# Detections**	**Requirements**
Height	1	
Speed	3:9	All detections must be in adjacent frames and correspond to the same wave
Wavelength	2	The two detections must overlap to give such that a perpendicular vector to their direction can intersect them both

6.3 Kernel Optimization

Eighteen different kernels were sampled from all the web cameras. This was done by either cropping the actual image about the intended detection then smoothing the shape so that it didn't over fit or by testing Sobel edge detectors of particular gradients. Each of the eighteen kernels were tested on an identical set of labeled images and keeping the Gaussian size in the blurring process before and after the convolution constant. The summation of all false positive and false negative values over all the beaches are normalized and plotted in figure 6.3 so that the kernel size could be compared for each kernel shape. The graphs showed a relatively poor accuracy over the entire set of web cameras as each kernel detected less than 40% of the labeled waves. In order to look for the largest errors in the detection process the best ksize for each kernel was also plotted for each beach and day to determine whether detection was poor for all kernel shapes. After testing with multiple value for the parameters for the Guassian Blur function applied to the convolution image and the image of edges, it was discovered that the results of false positives and false negatives did not change. Hence from this point for all experiments the convolution image will be blurred with a Gaussian kernel refereed to in 5.2 with parameters $\sigma_x = \sigma_y = 9$ and the image of edges will be blurred with a Gaussian kernel refereed to in 5.2 with parameters $\sigma_x = \sigma_y = 3$.

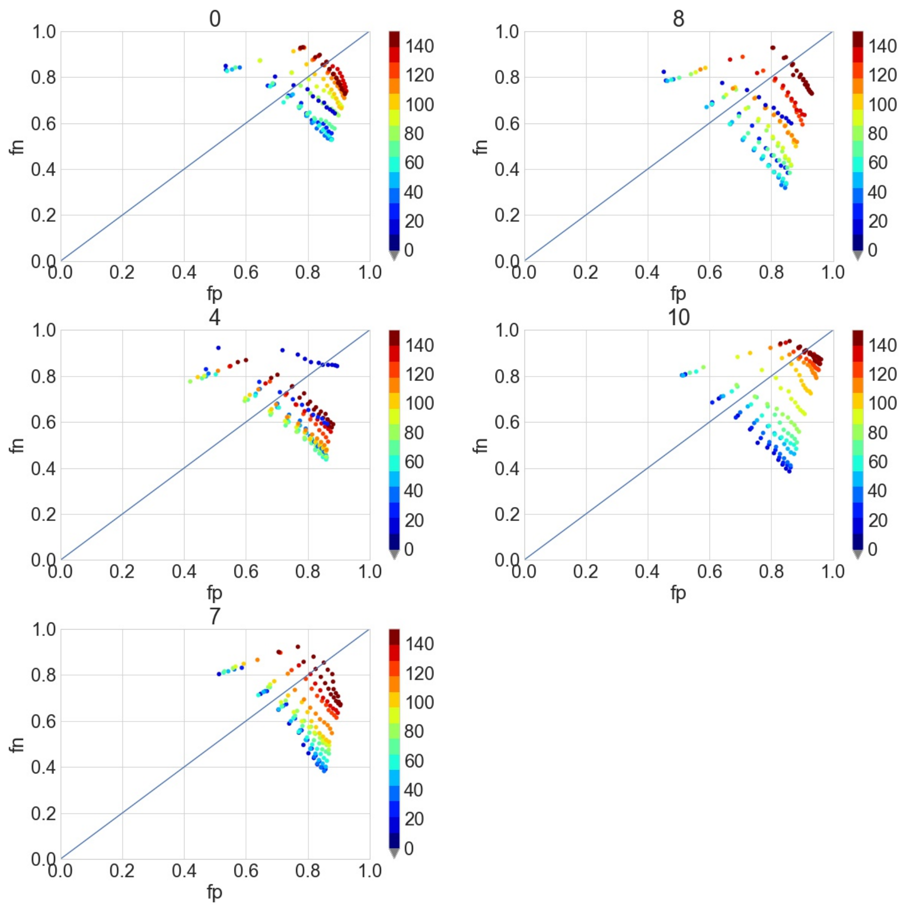


Figure 6.3: The plot of false positives against false negatives for the best five kernels where each graph corresponds to a different shaped kernel and each color corresponds to a different kernel size

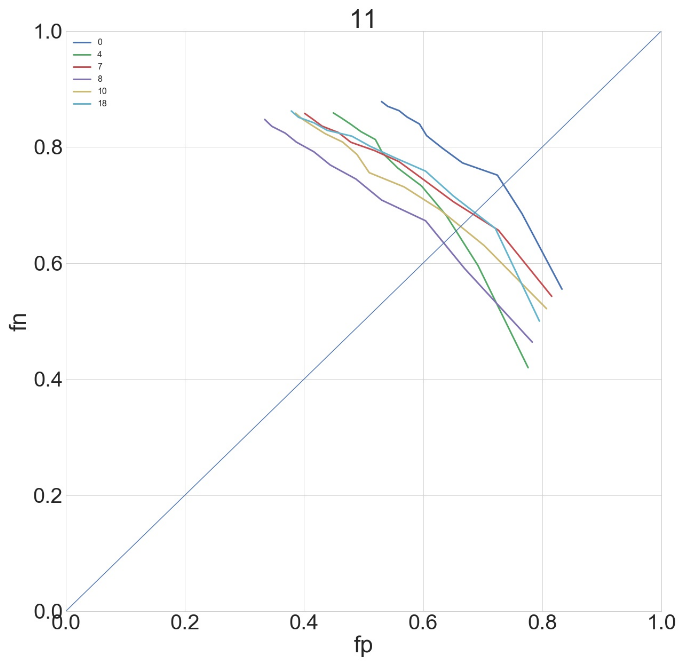


Figure 6.4: The plot of false positives against false negatives for each of the kernels sizes producing the best curves in figure 6.3

The table 6.5 shows the best kernels at detecting wave peaks. The decision was based on both figure 6.4 and 6.6. The best three kernels produced the three false negative and false positive that lie closest to the origin in figure 6.4. The decision was also certified as those selections lie closest to the origin for a least two of the twelve beaches in figure 6.6. For clarity kernel 4 resembled the shape of $KernelC$ in section 5.1, kernel 8 resembles the shape of $KernelA$ in section 5.1 and kernel 10 is a Sobel filter with a gradient of 1.

Kernel	Size
4	80
8	50
10	20

Figure 6.5: Best kernels

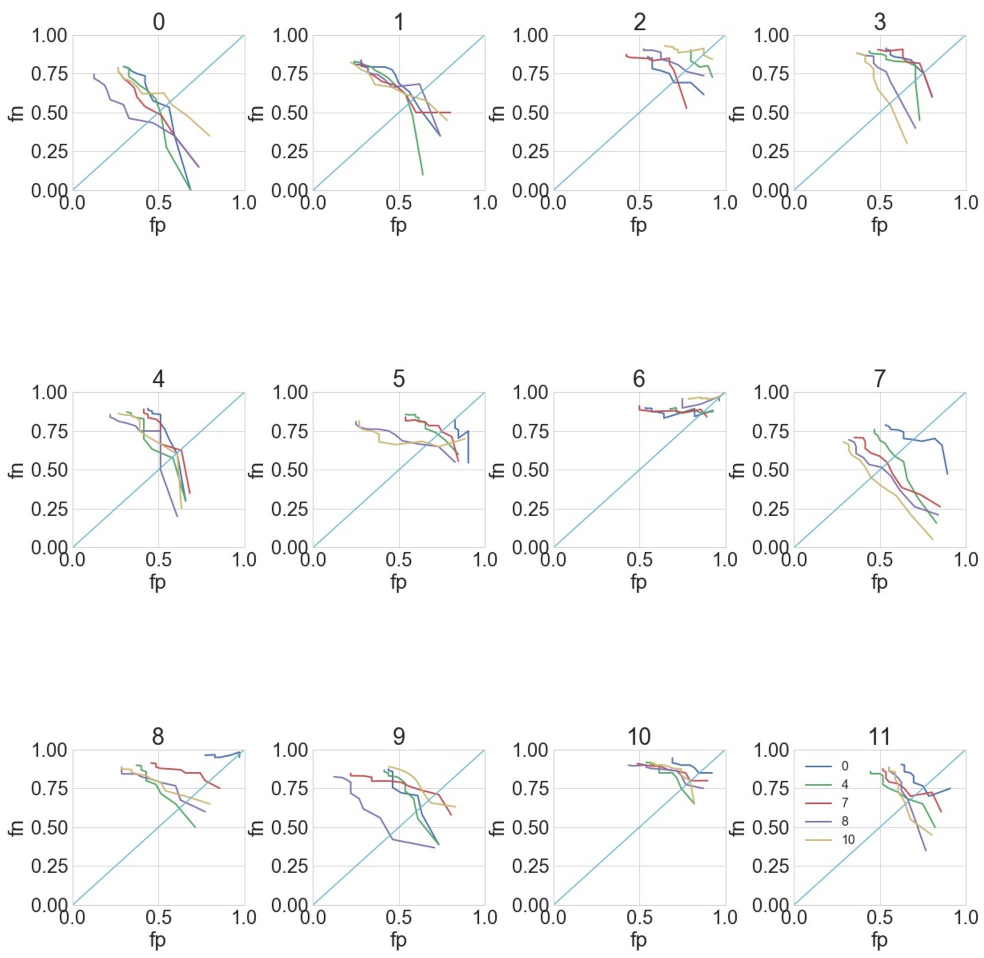


Figure 6.6: The plots of false positives against false negatives for each of the kernels sizes producing the best curves in figure 6.3 for each beach

Varying the detection threshold allows for the optimization of the algorithm over different ratios of false positive to false negative. As a simple approximation our best accuracy is accounted by the data point that lies closest to the false positive = false negatives line which has a match threshold in the range 3:5. The F1 accuracy is a value used to measure this and is equal to the $F1 = 2\frac{1}{\frac{1}{precion}+\frac{1}{recall}}$ quoted from Van Rijsbergen et al [13]. As out false positive and false negative rates are already normalized, the F1 accuracy is equal to $F1 = 2\frac{1}{\frac{1}{1-FP}+\frac{1}{1-FN}}$. To analyse whether a combination of the best three kernels would improve the accuracy a random frame from each training set was taken for each kernel using a match threshold of 4. These frames are shown below for the best 3 kernels in figure 6.7, 6.8 and 6.9.

Figures 6.7,6.8 and 6.9 are the results of kernel 4, kernel 8 and kernel 10 consecutively on the same 12 sample images from each dataset. Henceforth any image that is referenced specifically will be referenced using a grid like fashion where the top left corner is the origin.

- The blue lines correspond to the labeled wave segments .

- The lines are yellow when they also pass both tests outlined in section 6.2 but has not necessarily been chosen to represent that labeled wave as each labeled wave can only have one corresponding detection.

- The lines are red when the algorithm has made a detection but the detection has either passed one or neither of the tests outlined in 6.2.

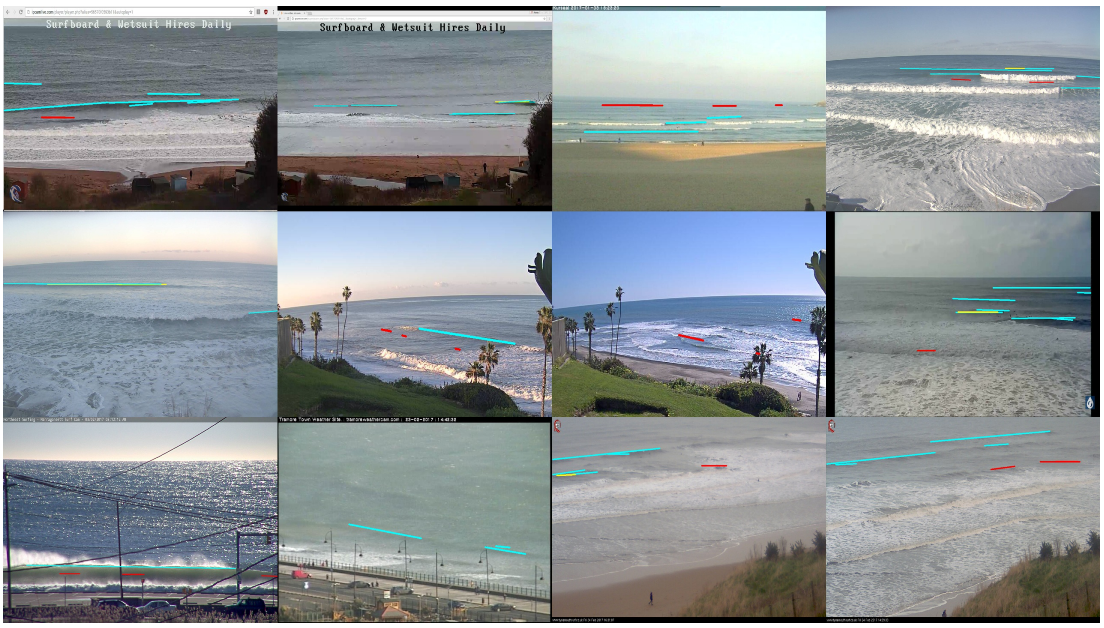

Surfboard & Wetsuit Hires Daily
Surfboard & Wetsuit Hires Daily

Figure 6.7: An image from each web camera for the detections made by kernel 4

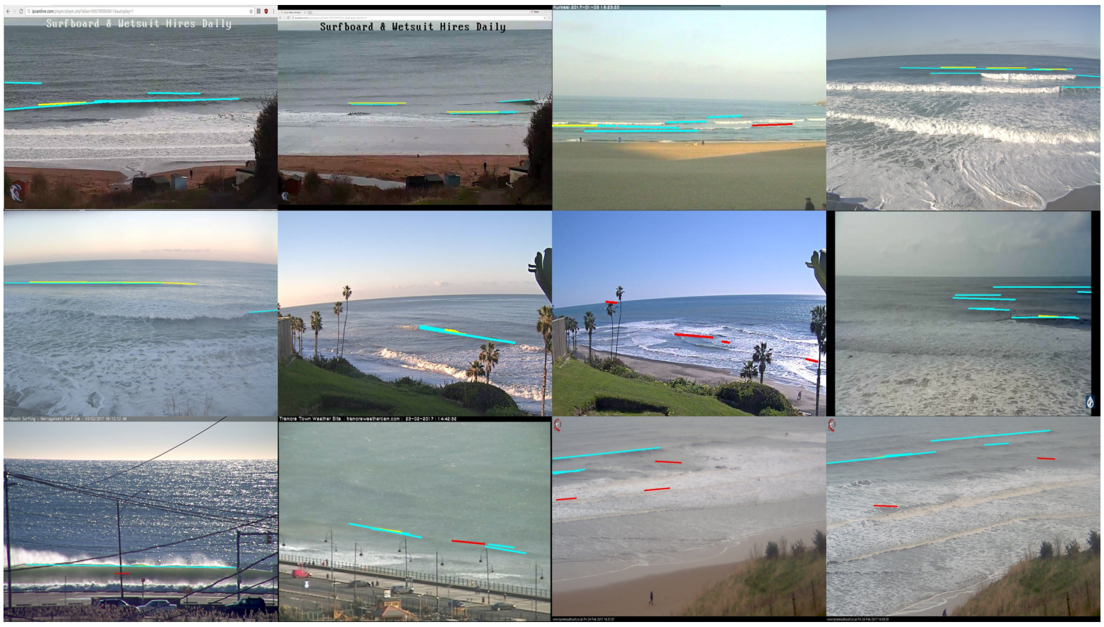


Figure 6.8: An image from each web camera for the detections made by kernel 8

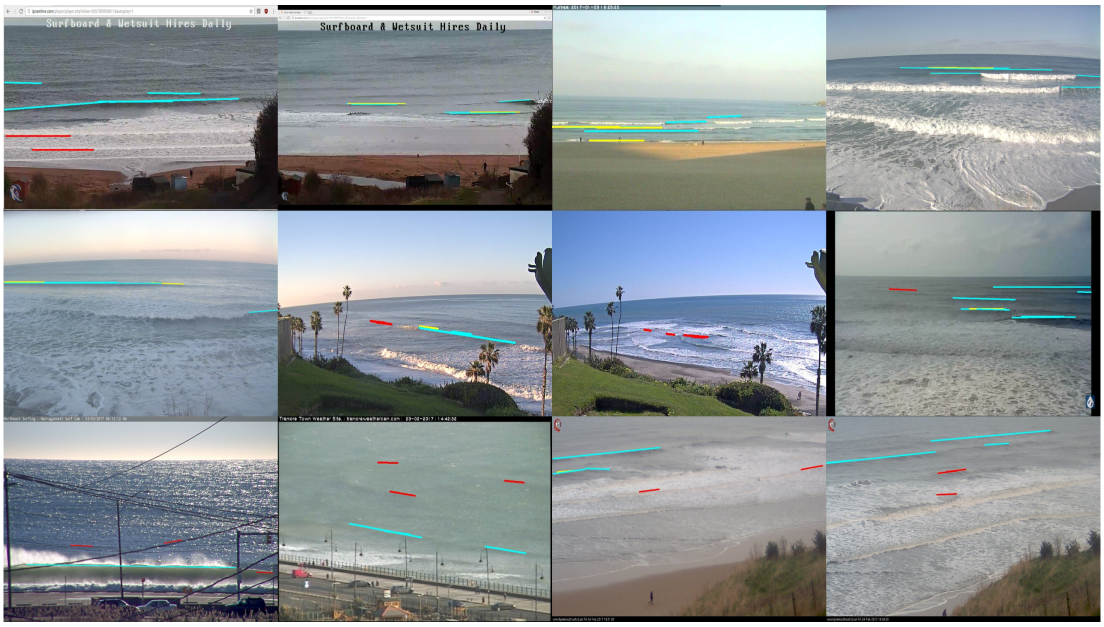


Figure 6.9: An image from each web camera for the detections made by kernel 10

The first noticeable characteristic of all the kernels is they all identify similar waves of certain characteristics. All correct detections have a noticeably large contrast between the shadowed wave region and the water behind. They all lack turbulent flow as the wave peak resembles a solid straight line. Any deviation from a solid straight line has not been detected as is visible in images $[2, 1]$ and $[3, 2]$. This implies that any convolution image using a combination of these kernels would not increase the true positive rate but will increase the false positive rate. False detections are visibly induced by one a highly concave wave crest where there is a sizable region of lighter intensity before the crest when traversing upwards which is visible in [0, 3]. Two stand alone patches of water surface not covered by remnant or breaking white water but surrounded by remnant and breaking white water. This effect is noticeable in images [2, 1], [2, 2] and [3, 2].

With regards to the performance of each kernel 4 is clearly the worst in this set of images although its F1 accuracy over all is better than kernel 10 as seen in 6.4 Both kernel 8 and kernel 10 have the same number of images with at least one correct detection. Conversely kernel 8 manages to detect multiple grand truths in a couple of images which reinforces the F1 accuracy being higher as seen in 6.4.

As all the kernels bias to false detection is different. kernel 4 tends to falsely detect really dark regions as waves, kernel 8 tends to falsely detect lighter regions as waves and kernel 10 tends to falsely detect small edges of high contrast as waves. Because of this it makes sense to optimize all three to see which gives the overall best F1 accuracy.

6.4 Hough line Parameters

To optimize the probabilistic Hough line algorithm I varied the following parameters.

- The minimum threshold above which a maximum turning point in the convolution image is drawn on the edge image.
- The confidence value for the probabilistic Hough line algorithm.
- The minimum line length for the probabilistic Hough line algorithm

As seen in figures 6.10. 6.11 and 6.12 the false negative against false positive curve varied very little. By looking at the left most bottom points parameters the optimum confidence value was chosen to be 100 the optimum threshold was chosen to be 1 and the optimum minimum line length was chosen to be 50. Through out all the experiments the maximum line gap for the probabilistic Hough line algorithm was 3 as its value also gave little improvement to the algorithm.

The small effect of these parameters could be due to a match threshold value being to high or the edge image resembling very few straight edges for any given convolution. Future work could be done into drawing a thicker edge for every maximum turning point in the convolution image to see if it influences the Hough line algorithm more.

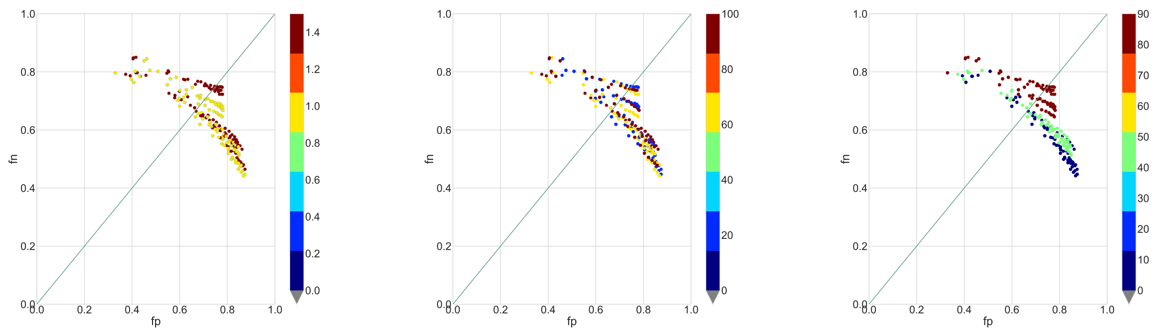


Figure 6.10: The false positive and false negative curve for kernel 4: *Left* varying the minimum threshold to for finding a maximum from the convolution *Middle* varying the confidence value for the probabilistic Hough line algorithm *Right* varying the minimum line length for the probabilistic Hough line algorithm

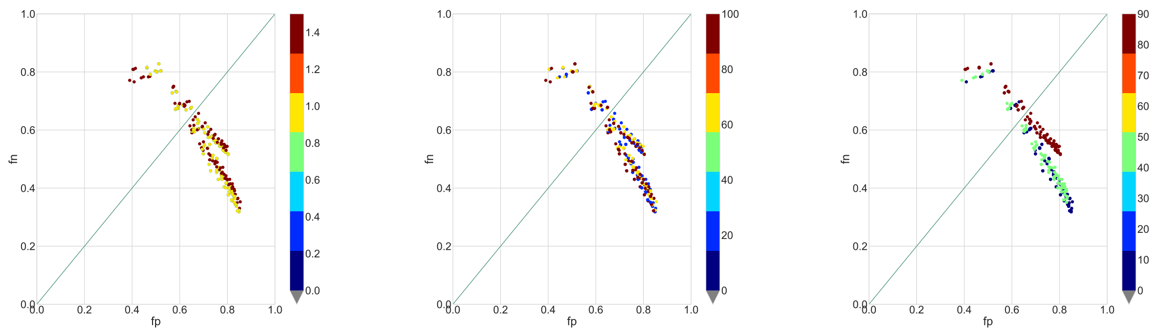


Figure 6.11: The false positive and false negative curve for kernel 8: *Left* varying the minimum threshold to for finding a maximum from the convolution *Middle* varying the confidence value for the probabilistic Hough line algorithm *Right* varying the minimum line length for the probabilistic Hough line algorithm

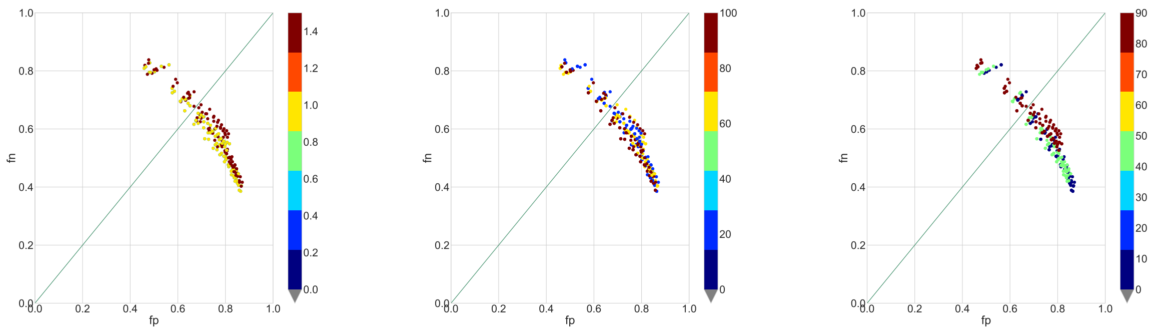


Figure 6.12: The false positive and false negative curve for kernel 10: *Left* varying the minimum threshold to for finding a maximum from the convolution *Middle* varying the confidence value for the probabilistic Hough line algorithm *Right* varying the minimum line length for the probabilistic Hough line algorithm

6.5 Clustering Parameters

The clustering algorithm used two parameters kernel density bandwidth and the clustering factor which corresponds to how many segments must lie in a cluster before the result can be classified as a wave. The optimum value for the clustering factor was found to be 1 and the band width to be 0.08.In figurebwcompare it can be seen that the accuracy has improved slightly on every beach except from 9 and 11. As the kernel density estimation relies on the the detection threshold it varies the number of detections the clustering is done on. Therefore we cannot go further to plot a single curve unless there is a measure of how sure one can be that any given cluster is a wave. As an equal number of beaches false negatives increased as the false positives were reduced on the others the overall accuracy remained the same for the detection as a whole. Further work could be done on deriving a bandwidth from any given set of detections so that it not be a hard coded parameter.

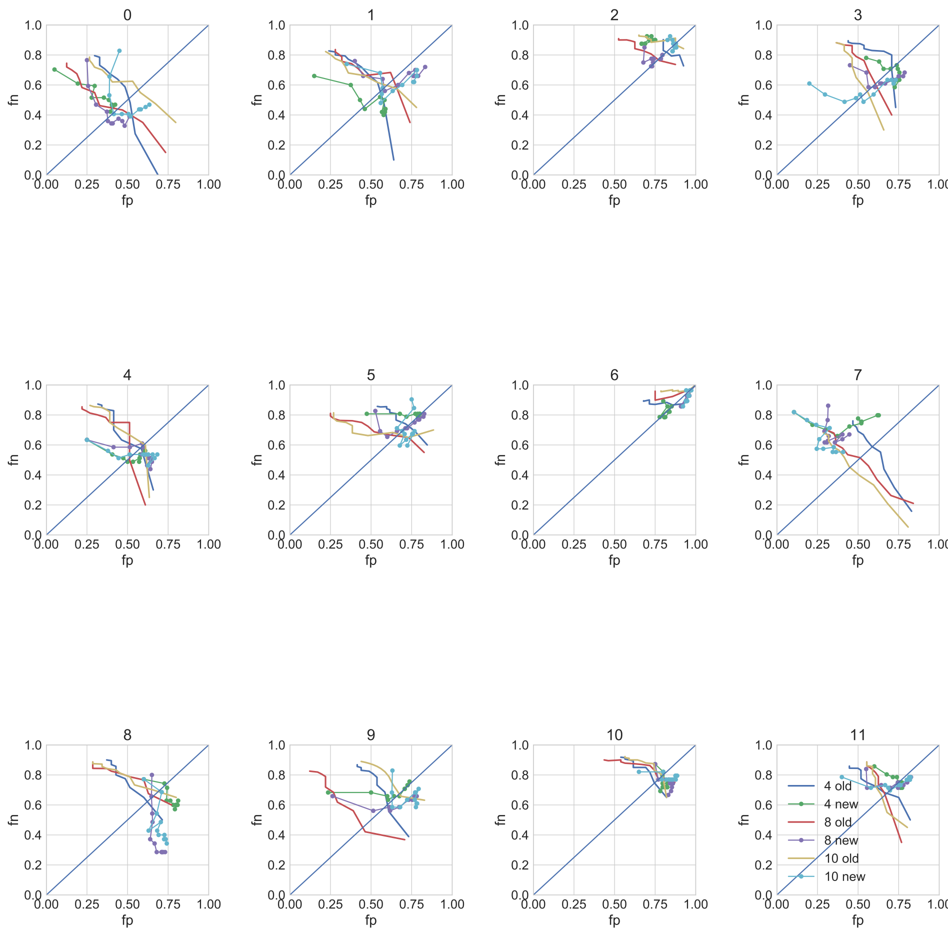


Figure 6.13: Figure showing the false positive true negative curves for each of the 3 kernels with and without clustering

6.6 Complete Result

The best result for all the detections over all twelve data sets had a False Positive and False negative value of 0.63 which is an accuracy value of 0.37 and is fairly low. For some beaches detection was difficult for several reasons one was the low resolution of the camera didn't give the kernel enough room to gauge a similarity. To certify this prediction the compared the table below which gives and accuracy score for cameras with a a pixel dimension greater than 1000. These values are taken from the point closest to the origin on the original kernels without clustering and correspond to the highest F1 harmonic mean.

Beach	Dataset	Accuracy
Coldingham	0	0.55
Coldingham	1	0.47
Los Angeles2	3	0.45
Los Angeles2	4	0.5
Pipeline	7	0.55
Rhode Island	8	0.38
Tynemouth	10	0.28
Tynemouth	11	0.34

Both Tynemouth and RhodeIsland have low accuracy with higher resolution. A possible reason for this could be that their height above sea level. RhodeIsland is considerably low down compared to the others therefore the crest shadow is cast differently, as the waves surface normal takes a large reflectance angle for the sunlight. With regards to Tynemouth as the conditions were a lot more turbulent than other beaches the algorithm found it hard to detect the crests resembling straight lines. This suggests that Hough lines may not be the best robust regression method. The accuracies for Coldingham, Pipeline and Los Angeles are all fairly reasonable although as each use a different kernel for their highest accuracy. Furthermore Los Angeles and Coldingham have different kernels with the highest accuracy for each separate day of data. The clustering method increased the accuracy substantially for the beaches with accuracy over 0.44. This gives an indication that the algorithm run before the clustering must be improved to obtain a better effect by clustering.

Chapter 7

Wave Physical Parameters Estimation

This section will outline the theory involved in calculating the physical parameters of detected waves using values from the detection. This part of the project was not completed and therefore the mathematics was only theoretical and has not been tested against ground truth values.

7.1 Wave Tracking

To infer the speed of the wave a method was required to match particular segments between frames. As the web camera stream was subject to buffering and the detection algorithm was not prefect, an identical wave cannot be tracked between adjacent images by looking for the wave beneath its last position in the previous frame. To tackle this issue the detections for 20 waves were plotted over time to see how visible the transition in time was.

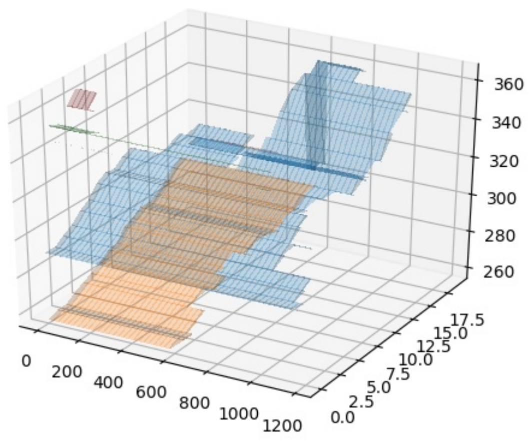


Figure 7.1: Wave segments plotted and clustered for a time stack of images where the color corresponds to the specific cluster where the algorithm has decided that all detection in that cluster are the same wave. The x axis corresponds to the x axis of the web cam images. The y axis corresponds to the time dimension in which the images are taken. The z axis corresponds to to the y dimension of the web camera image. The range plotted corresponds to the break zone calculated for that beach.

After this the waves are plotted in a **3D meshgrid**. Using the same method in section 5.4 detections are rotated so that they are parallel with the x axis. The **RANSAC** algorithm is used on the projection in the time and vertical axis to uncover a detected wave progressing toward the shore in consecutive images. As one wave is required to detect a speed the **RANSAC** algorithm is run as many times as is required to capture a wave with orientation between the angle of $\pi/6$ and $5\pi/6$ to the horizontal x axis. This orientation angle can then be used to calculate the wave speed.

7.2 Wavelength

First the perpendicular distance to the blue detections in figure 7.3 from the bottom of the image must be calculated by taking the bottom left corner as the origin. If $\widehat{N}$ is normal to the detections and $\overrightarrow{P}_j$ is a point on the bottom detection such that there lies two points along $\widehat{N}$, $\overrightarrow{P}_i$ on the top detection and $\overrightarrow{Q}$ on the horizon then the image distances $\phi_{i/j}$ are as follows:

$$\phi_i = \overrightarrow{P}_i\widehat{N} \quad \phi_j = \overrightarrow{P}_j\widehat{N} \quad \phi_t = \overrightarrow{Q}\widehat{N}$$

As the camera has an unknown physical angle θ from the vertical z dimension with the camera at the origin the angles to the detections shown in figure 7.2 are given as:

$$\theta_k = (\phi_k + \kappa)/\frac{\pi}{2}$$

We can also estimate the height of the camera H using the angle the at which the horizon is positioned. As we know that the distance d_w can be any value between 0 and 10 meters we know that the distance to the horizon is in the range:

$$d_w(\tan(\pi/2 - \theta_t))/(\tan(\theta_j) - \tan(\theta_i)) \quad 0 < d_w < 10$$

As we know the curvature of the earth we can estimate the camera height H and use H to estimate our camera zoom constant κ. The H value for the approximation can be found in appendix entry 3. We can then finally calculate the wavelength d_w as:

$$d_w = H(\tan(\theta_j) - \tan(\theta_i))$$

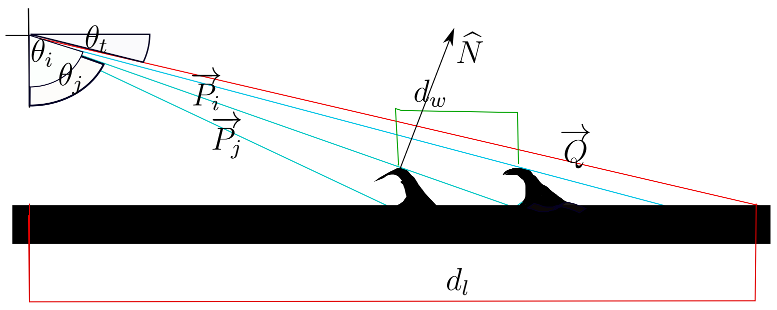


Figure 7.2: Diagram labeling the measurements used in the calculation of the physical features

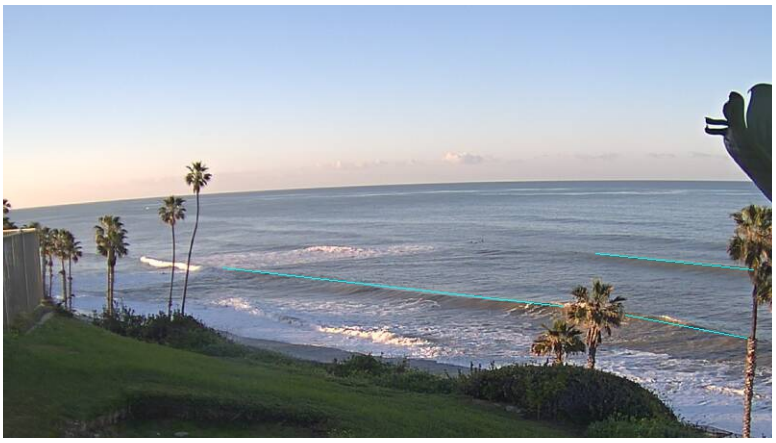


Figure 7.3: The physical detections referred to in figure 7.2

7.3 Speed

The speed can be calculated using the angle between the x axis and the wave trajectory in the time and vertical plane α. As the change in $v = \delta s/\delta t$ where δs and δt make up the hypotenuse and adjacent sides of a right angle triangle about angle α. Therefore the speed $v = A(\frac{1}{\cos(\theta)})$. The value of A gives the difference in scale between the image dimensions and the physical dimensions. As the angle at which the wave approaches the shore in time is small, our value of A is equal to the distance between the camera and the wave in which you're calculating the instantaneous velocity on. Therefore A is given by:

$$\frac{H}{\cos\{(y_i + \kappa)/\frac{\pi}{2}\}}$$

where y_i is the position of that line segment in the vertical and κ and H are the web camera specific constants calculated in section 7.2.

7.4 Height

Using our set of detected segments of the wave we can derive a maximum height per segment and further derive the maximum height per single wave. As outlined in section 2.2 the significant wave height can be derived from the 4 times the surface elevation in the time dimension as seen in figure 7.1. As the wave is moving at relatively constant speed the wave height can be derived from the surface elevation in the dimension parallel to the wave movement and approximately perpendicular to the plane of view. Therefore as half of our surface elevation is visible as the shadow caused due to the difference in the water surface normal and the azimuth of the sun you can infer the size of a wave by the variance of the convolution output about the detected wave peak. An example result from which this value would be calculated from is given in figure 7.4. The formula for doing this is shown below $I(x, y)$ is the convolution output and $\overrightarrow{P}$ and $\widehat{N}$ is the endpoint and perpendicular vector of a detected segment (shown in figure 7.4 in red) is :

$$H_w = 4\sigma_{elevation} = max_x\{A\sqrt{\frac{\sum_y I(x,y)(((x,y)^T - \overrightarrow{P}).\widehat{N})^2}{\sum_y I(x,y)}}\}$$

. The estimated wave height for the whole wave can then be calculated from the maximum of each segment maximum that lie in the same cluster outline in section 5.4. As the angle between θ_i and θ_j is small, our value of A is equal to the distance between the camera and the second wave. Therefore A is given by:

$$\frac{H}{\cos\theta_i}$$

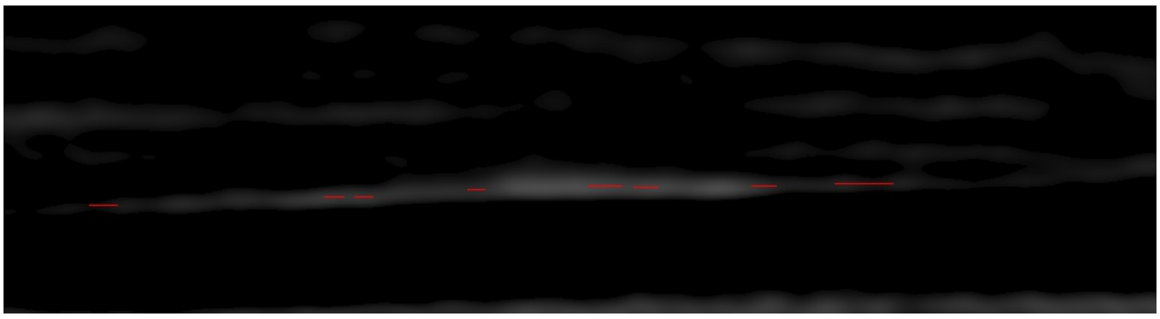


Figure 7.4: The convolution output with the output detected line segments marked in red

Chapter 8

Conclusion

This project aimed to provide the algorithms required to detect the wavelength, speed and height of a wave on an arbitrary, low cost, low resolution web camera positioned close enough to the beach for a person to make the above three deductions. An algorithm was designed to make use of how a wave evolves over time by looking at a time stack of images to decipher the wave breaking region and direction. This worked to the correct effect by removing both the sand headland and horizon from the image. Some cameras failed to find the break zone region mainly due to how the camera passes intensity which limited detail in the normalization image. This made it difficult to notice activity in the sea.

An algorithm was designed to use kernel convolution and robust regression in the detection of the wave shadowed region in order to find its peak. The shadowed region was used and kernel convolution classified 5 out of 12 waves of all the labeled data with an accuracy of 50%. The Hough line algorithm was also used but further work could be done into detecting continuous boundaries rather than straight edges to account for the variance in the positioning of the turning point of the convolution. The algorithm was evaluated and used to test the parameters using a dataset of 240 images. By testing parameters the algorithm reached a detection accuracy of 38% calculated over all the web cameras.

Clustering techniques to group partial wave detections together were evaluated to improve performance. Over the entire range of web cameras the clustering method had as large of a detrimental effect as it was beneficial and therefore gained no additional accuracy for detection. Further using the time dimension, outlying waves were discarded that did not appear in the frames before or after. This was experimented with on **Coldingham** but not validated and therefore requires further work. The algorithm was then to use the detected surf-able wave peak lines to calculate the speed, wavelength and height. This too was not tested thoroughly but further work could also be done to evaluate the physical measurements using equations outlined in section 7. Once data is collected a graph of the detected measurements against predicted measurements sourced from weather forecasting sites can be used to evaluate the correlation and hence the algorithm itself. This process can be achieved through querying the longitude and latitude's for each of the cameras found in appendix entry 2.

The task of investigating the feasibility of the use of low cost surf shop web cameras as a source for validating the forecast predictions has been met. As measurements of the wave length, height and speed need not be recorded for every frame, to infer the three measurements of interest the algorithm does not require a large hit rate. For an automatic wave monitoring system it is paramount that there are no false detections in the extraction of wave features. This has not been achieved in this projects but work has been done towards it.

After experiencing the issues involved in deploying a real time monitoring system I would suggest the use of Convolution Neural Networks similar to that used by Mesay Belete Bejiga et al [4] to carry this project further. Detection over a huge variance in sea conditions, lightening and variation between features, position and shape on different beaches is achievable using deep learning. A fully functional monitoring system for extracting height, speed and wavelength would only be achievable when the false positives rate is much low. By training a Convolution neural Network on a set of correct detections and a set of incorrect detections to recognize the wave crest edge. This would be done by sampling images such that the segment representing that detected wave peak passes through the center. An additional set of images as a counter example would need to be taken from areas in the break zone region where there does not lie a labeled wave segment in the proximity. Particular care would be taken to not under fit and build an average edge classifier. Another approach would be to take the image on the end points of a detected crest. This could be labeled through clicking on the wave peak where it meets a white water section. As there are fewer of these in any image the false positive rate will be lower meeting the criteria for an automatic monitoring system using web cameras and closer to the dream of an accurate forecast.

Metadata

Location	Source
Bedruthan	`http://82.153.141.73/mjpg/video.mjpg`
Kursaal	`http://212.142.228.68/mjpg/video.mjpg`
Coldingham	`http://67.205.163.71/coldingham.mjpg`
Tynemouth	`http://81.149.2.82:8091/mjpg/video.mjpg`
Tramore	`http://78tramore.home.dyndns.org/mjpg/2/video.mjpg`
Australia	`http://114.72.112.23/mjpg/video.mjpg`
Lahinch	`http://86.47.88.163/mjpg/video.mjpg`
Cornwall	`http://217.41.31.74:4271/GetData.cgi`
Losangeles1	`http://63.138.85.84/mjpg/video.mjpg`
Losangeles2	`http://98.189.158.57/mjpg/video.mjpg`
Losangeles3	`http://98.189.156.36/mjpg/video.mjpg`
Palma	`http://83.56.31.69/mjpg/video.mjpg`
Rhodeisland	`http://108.34.179.41/mjpg/video.mjpg`
Florida	`http://97.76.101.212/mjpg/video.mjpg`
Pembrokeshire	`http://81.149.146.68/mjpg/video.mjpg`
Northdakota	`http://67.205.163.71/northdakota.mjpg`
Shallotte	`http://216.99.115.136:8080/mjpg/video.mjpg`
Pipeline	`http://67.205.163.71/shallotte.mjpg`

Figure 1: The source of each web camera used in this project

Location	Latitude	Longitude
Bedruthan	50.46	-5.03
Kursaal	43.32	-1.98
Coldingham	55.89	-2.13
Tynemouth	55.03	-1.43
Tramore	52.16	-7.14
Lahinch	52.93	-9.35
Losangeles1	33.76	-118.15
Losangeles2	33.76	-118.15
Cornwall	50.22	-5.48
Losangeles3	33.76	-118.15
Rhodeisland	41.44	-71.45
florida	28.06	-82.83
Pembrokeshire	51.78	-5.10
Shallotte	33.89	-78.42
Pipeline	21.67	-158.05

Figure 2: The longitude and latitude of each web camera of each web camera used in this project

Height (meters)	Distance (km)
0	0.0
1	3.6
2	5.1
3	6.2
4	7.1
5	8.0
6	8.7
7	9.4
8	10.1
9	10.7
10	11.3
20	16.0
30	19.5
40	22.6
50	25.2

Figure 3: Horizon calculations

Bibliography

[1] Stefan GJ Aarninkhof and B Gerben Ruessink. Video observations and model predictions of depth-induced wave dissipation. *IEEE Transactions on Geoscience and Remote Sensing*, 42(11):2612–2622, 2004.

[2] Rafael Almar, Rodrigo Cienfuegos, Patricio A Catalán, Hervé Michallet, Bruno Castelle, Philippe Bonneton, and Vincent Marieu. A new breaking wave height direct estimator from video imagery. *Coastal Engineering*, 61:42–48, 2012.

[3] Adelchi Azzalini and Giovanna Menardi. Clustering via nonparametric density estimation: The r package pdfcluster. *arXiv preprint arXiv:1301.6559*, 2013.

[4] Mesay Belete Bejiga, Abdallah Zeggada, Abdelhamid Nouffidj, and Farid Melgani. A convolutional neural network approach for assisting avalanche search and rescue operations with uav imagery. *Remote Sensing*, 9(2), 2017.

[5] Jean-Raymond Bidlot. Ocean wave forecasting at e.c.m.w.f. 2014.

[6] Patricio A Catalán, Merrick C Haller, Robert A Holman, and William J Plant. Optical and microwave detection of wave breaking in the surf zone. *IEEE Transactions on Geoscience and Remote Sensing*, 49(6):1879–1893, 2011.

[7] Yaniv Gal, Matthew Browne, and Christopher Lane. Long-term automated monitoring of nearshore wave height from digital video. *IEEE Transactions on Geoscience and Remote Sensing*, 52(6):3412–3420, 2014.

[8] Davar Khoshnevisan. Elements of density estimation. *Elements of Density Estimation.*

[9] Magicseaweed. Surf forecast report. `http://magicseaweed.com/`, 2013.

[10] Jiri Matas, Charles Galambos, and Josef Kittler. Robust detection of lines using the progressive probabilistic hough transform. *Computer Vision and Image Understanding*, 78(1):119–137, 2000.

[11] Opencv. Gaussian blur. /urlhttp://docs.opencv.org/2.4/modules/core/doc/operations_on
2013.

[12] Surfline. Surf forecast report. `http://www.surfline.com/home/index.cfm`, 2013.

[13] Cornelis Joost Van Rijsbergen. *The geometry of information retrieval.* Cambridge University Press, 2004.

[14] Wikipedia. Significant wave height. `https://en.wikipedia.org/wiki/Significant_wave_height`, 2013.

[15] Wikipedia. Sobel operator. `https://en.wikipedia.org/wiki/Sobel_operator`, 2013.

[16] Stackoverflow Zaw Lin. How to parse mjpeg http stream from ip camera? `http://stackoverflow.com/questions/21702477/how-to-parse-mjpeg-http-stream-from-ip-camera`, 2013.