

Applying the Histogram of Oriented Gradients Algorithm for Detecting Grass Lay Direction

Ben Sunshine

2024-05-10

Table of contents

Chapter 1

Abstract

Subsistence-oriented indigenous communities across Alaska rely heavily on Traditional Ecological Knowledge (TEK), a holistic understanding of their environment acquired through generations of observation and cultural transmission. Among the Anishinaabek tradition, sweetgrass symbolizes wisdom and knowledge, passed down from elders to younger generations. Indigenous hunters and gatherers have long observed the alignment of grass and plants after the growing season as indicative of prevailing wind directions. Predominant wind direction serves a crucial role to subsistence practitioners when hunting, fishing, settling, and keeping track of changing weather. Due to the remote and harsh conditions, traditional weather stations are absent to measure shifts in historically predominant wind directions. On islands like St. Lawrence Island in Savoonga, AK, natives have observed a shift from historically predominant northerly wind patterns to southerly and easterly and dominated winds. In a previous study Dr. Jon Rosales (Environmental Studies) and his team collected images of grass lay from St. Lawrence University's Living Laboratory and manually attempted to measure grass lay angles and relate them with wind data. This research project seeks to reinforce Traditional Ecological Knowledge (TEK) with Scientific Ecological Knowledge (SEK) to develop our understanding of Alaskan indigenous wisdom and its relation with modern scientific findings. We investigated the Histogram of Oriented Gradients (HOG) algorithm to automate the measurement of grass lay angles. We applied the algorithm to various images sampled from the internet and the Living Laboratory to test its viability.

Chapter 2

Data

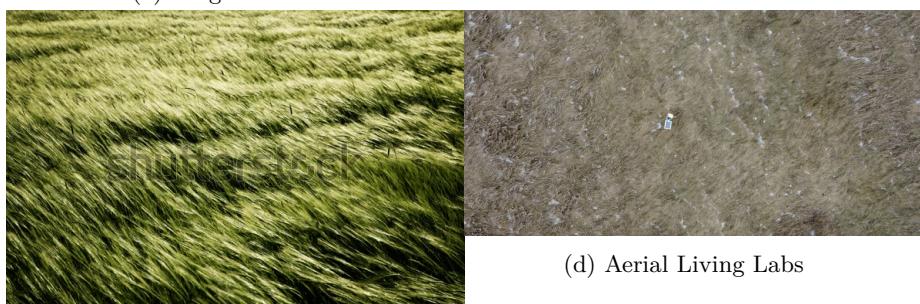
To evaluate the algorithm's performance, we collected images with varying levels of complexity. Beginning with basic geometric shapes and diagonal lines sourced from Google Images, we established a foundational data set for initial testing. We incorporated aerial cityscapes including San Francisco, Salt Lake City, and Detroit from Mapbox, aiming to assess the algorithm's ability in identifying urban grid structures such as streets and highways. Additionally, we included images of grass sourced from both Google Images and Dr. Jon Rosales (Environmental Studies) at St. Lawrence University's Living Lab. The images sourced from Google Images were primarily The Living Lab images included aerial and close up shots, with special attention given to aerial images which featured a northern indicator line. To ensure consistent orientation, each aerial image was manually rotated to align north facing direction upwards before analysis, with the northern indicator subsequently removed to avoid introducing artificial lines in the image.



(a) Diagnol Lines



(b) Downtown San Francisco



(d) Aerial Living Labs

(c) Internet Grass

Figure 2.1: Featured Images for Evaluation

Chapter 3

Methods

The HOG algorithm, introduced by Navneet Dalal and Bill Triggs in 2005, is a popular technique for object detection in images. The algorithm can identify gradient magnitudes and angles at each pixel in an image. The preliminary steps involved using the ‘skimage’ library from Python to preprocess the images of interest. This included loading, resizing, and converting the images to grayscale. Images were rescaled to standardize their resolutions and preserve their aspect ratios to prevent distortion that could affect the accuracy of angle identification. Converting the images to grayscale was necessary because it allowed for focusing on a single channel to represent pixel intensity, rather than three channels (red, green, and blue).

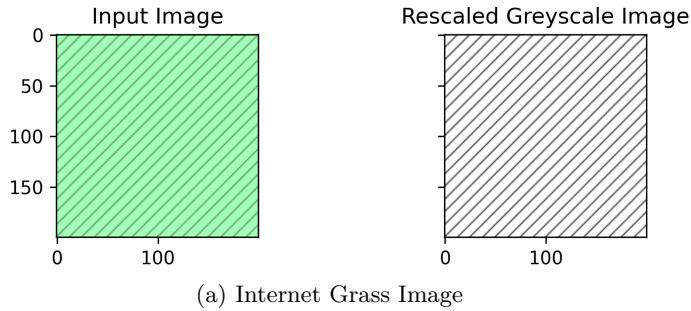


Figure 3.1: Rescaling and Converting Image to Greyscale

The HOG features were then computed for the resized images, which involved calculating the gradient magnitudes and angles at each pixel. The gradient magnitude at each pixel is comprised of the gradients in the ‘x’ and ‘y’ directions. The gradient in the x-direction is computed by subtracting the pixel value to the left of pixel of interest is subtracted from the pixel value to its right. Similarly, the gradient in the y-direction is calculated by pixel value below the pixel of interest is subtracted from the pixel value above the pixel of interest.

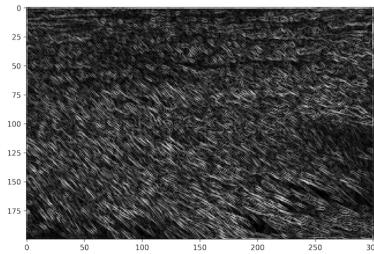
$$G_x = I(r, c + 1)I(r, c - 1)$$

$$G_y = I(r+1, c)I(r-1, c)$$

Now to calculate the gradient magnitude at the pixel of interest, the Pythagorean Theorem can be utilized where the gradient magnitude is equal to the square root of the x-gradient squared plus the y-gradient squared. The angle at a given pixel can be calculated by taking the inverse tangent of its y-gradient divided by its x-gradient. It is important to note all angles produced by this algorithm are between zero and one hundred eighty degrees. This occurs, because the inverse tangent function used for calculating a given pixel's angle cannot distinguish between all four quadrants.

$$\text{Magnitude}(\mu) = \sqrt{G_x^2 + G_y^2}$$

$$\text{Angle}(\Theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$



(a) Internet Grass Image

Figure 3.2: Plotting Gradient Magnitudes

Next, histograms are constructed to visualize the distribution of gradient magnitudes and angles. Two different techniques for creating gradient angle histograms were implemented. The first histogram was created by counting the number of angles that fell into their respective bins. The second scheme factors in a pixel's gradient magnitude and its allocation to its bordering bins. Here, the weight assigned to each bin is calculated by the angle's deviation from the center of its central bin. This approach allows for a more representative histogram which splits angles between bins and takes their magnitudes into account. Lastly, these histograms are converted to polar histograms so the primary angles can be visualized and compared to their original images.

Chapter 4

Results

4.1 Load R Packages

```
library(reticulate)
library(tidyverse)

-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
vforcats    1.0.0     v stringr   1.5.0
v ggplot2   3.5.0     v tibble    3.2.1
v lubridate 1.9.2     v tidyrr    1.3.0
v purrr     1.0.2

-- Conflicts -----
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
library(mapsapi)
library(mapboxapi)
```

Usage of the Mapbox APIs is governed by the Mapbox Terms of Service.
Please visit <https://www.mapbox.com/legal/tos/> for more information.

```
library(magick)
```

Linking to ImageMagick 6.9.12.93
Enabled features: cairo, fontconfig, freetype, heic, lcms, pango, raw, rsvg, webp
Disabled features: fftw, ghostscript, x11

```

Sys.which("python")  

python  

"/Users/bensunshine/.virtualenvs/r-reticulate/bin/python"  


## 4.2 Download Aerial City Images

key <- Sys.getenv("mapbox_key")  

map <- static_mapbox(  

  access_token = key,  

  style_url = "mapbox://styles/mapbox/satellite-v9",  

  width = 300,  

  height = 200,  

  image = T, latitude = 37.792004, longitude = -122.428079, zoom = 12  

)  

magick::image_write(map, "images/san_francisco_scale_zoom_12.png")  

points_of_interest <- tibble::tibble(  

  longitude = c(-112.065945, -111.853948,  

                -111.852956, -112.023371),  

  latitude = c(40.794275, 40.791516,  

              40.502308, 40.502308)  

)  

prepped_pois <- prep_overlay_markers(  

  data = points_of_interest,  

  marker_type = "pin-l",  

  label = 1:4,  

  color = "#fff",  

)  

map <- static_mapbox(  

  access_token = key,  

  style_url = "mapbox://styles/mapbox/satellite-v9",  

  width = 800,  

  height = 1200,  

  image = T,  

  latitude = 40.7,  

  longitude = -111.876183, zoom = 12

```

```

)
magick::image_write(map, "images/salt_lake_city_zoom_12.png")

map <- static_mapbox(
  access_token = key,
  style_url = "mapbox://styles/mapbox/satellite-v9",
  width = 1200,
  height = 800,
  image = T,
  latitude = 42.336322,
  longitude = -83.048705, zoom = 12
)

magick::image_write(map, "images/detroit_zoom_12.png")

```

4.3 Load Python Libraries

```

import matplotlib.pyplot as plt
import pandas as pd

# jupyter only inline output command
#%matplotlib inline

from skimage.io import imread, imshow
from skimage.transform import resize
from skimage.feature import hog
from skimage import data, exposure

import matplotlib.pyplot as plt
from skimage import io
from skimage import color
from skimage.transform import resize
import math
from skimage.feature import hog
import numpy as np

```

4.4 Collect HOG Features

```
img_list = []
img_list.append(color.rgb2gray(io.imread("images/diagnol_lines.jpg")))
img_list.append(color.rgb2gray(io.imread("images/san_francisco_scale_zoom_12.png")))
img_list.append(color.rgb2gray(io.imread("images/grass_image2.jpg")))
img_list.append(color.rgb2gray(io.imread("images/living_lab_aerial/aerial_grass_living_lab.jpg")))

#img = color.rgb2gray(io.imread("images/grass_image2.jpg"))

# img = color.rgb2gray(io.imread("images/b_test_image_copy.jpg"))
#img = color.rgb2gray(io.imread("images/long_grass_sample.jpeg"))
#img = color.rgb2gray(io.imread("images/diagnol_lines.jpg"))

#img = color.rgb2gray(io.imread("images/san_francisco_scale_zoom_12.png"))

#img = color.rgb2gray(io.imread("images/diagnol_lines_flipped.jpg"))
#img = color.rgb2gray(io.imread("images/long_grass_sample_cropped.jpg"))

# aerial rotated image
#img = color.rgb2gray(io.imread("images/living_lab_aerial/aerial_grass_living_lab_rotated.jpg"))

# zoomed internet photo
#img = color.rgb2gray(io.imread("images/dead_grass_zoom.jpeg"))

# zoomed in 11
#img = color.rgb2gray(io.imread("images/living_lab_aerial/LL_zoomed_in_11.jpg"))

# zoomed in 12
#img = color.rgb2gray(io.imread("images/living_lab_aerial/LL_zoomed_in_12.jpg"))

# zoomed in 16
#img = color.rgb2gray(io.imread("images/living_lab_aerial/LL_zoomed_in_16_side.jpg"))

# real one
#img = color.rgb2gray(io.imread("images/living_labs_real_grass_image.jpg"))

mag_list = []
theta_list = []
```

```

for x in range(len(img_list)):
    img = img_list[x]

    aspect_ratio = img.shape[0] / img.shape[1]
    print("Aspect Ratio:", aspect_ratio)

    height = 200
    width = int(height / aspect_ratio)
    print("Resized Width:", width)

    resized_img = resize(img, (height, width))
    img_list[x] = resized_img

    plt.figure(figsize=(15, 8))
    plt.imshow(resized_img, cmap="gray")
    plt.axis("off")
    plt.show()

    mag = []
    theta = []

    for i in range(height):
        magnitudeArray = []
        angleArray = []

        for j in range(width):
            if j - 1 < 0 or j + 1 >= width:
                if j - 1 < 0:
                    Gx = img[i][j + 1] - 0
                elif j + 1 >= width:
                    Gx = 0 - img[i][j - 1]
            else:
                Gx = img[i][j + 1] - img[i][j - 1]

            if i - 1 < 0 or i + 1 >= height:
                if i - 1 < 0:
                    Gy = 0 - img[i + 1][j]
                elif i + 1 >= height:
                    Gy = img[i - 1][j] - 0
            else:
                Gy = img[i + 1][j] - img[i - 1][j]

            magnitude = math.sqrt(pow(Gx, 2) + pow(Gy, 2))
            magnitudeArray.append(round(magnitude, 9))

            angle = math.degrees(math.atan2(Gy, Gx))
            if angle < 0:
                angle += 360
            angleArray.append(angle)

        mag.append(magnitudeArray)
        theta.append(angleArray)

    img_list[x] = [img, mag, theta]

```

```

        if Gx == 0:
            angle = math.degrees(0.0)
        else:
            angle = math.degrees(math.atan(Gy / Gx))
            if angle < 0:
                angle += 180

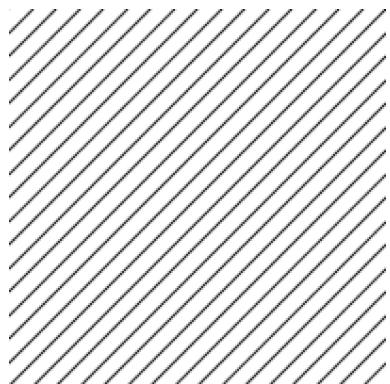
        angleArray.append(round(angle, 9))

    mag.append(magnitudeArray)
    theta.append(angleArray)

    mag_list.append(mag)
    theta_list.append(theta)

Aspect Ratio: 1.0
Resized Width: 200
<Figure size 3000x1600 with 0 Axes>
<matplotlib.image.AxesImage object at 0x292a43250>
(-0.5, 199.5, 199.5, -0.5)
Aspect Ratio: 0.6666666666666666
Resized Width: 300
<Figure size 3000x1600 with 0 Axes>
<matplotlib.image.AxesImage object at 0x2c418b850>
(-0.5, 299.5, 199.5, -0.5)
Aspect Ratio: 0.662751677852349
Resized Width: 301
<Figure size 3000x1600 with 0 Axes>
<matplotlib.image.AxesImage object at 0x2c874af90>
(-0.5, 300.5, 199.5, -0.5)
Aspect Ratio: 0.4904214559386973
Resized Width: 407
<Figure size 3000x1600 with 0 Axes>
<matplotlib.image.AxesImage object at 0x2dcc55690>
(-0.5, 406.5, 199.5, -0.5)

```





4.5 Build Data Frames for Each Image

```
mag_diagonal = np.array(mag_list[0])
theta_diagonal = np.array(theta_list[0])

mag_sf = np.array(mag_list[1])
theta_sf = np.array(theta_list[1])
```

```

mag_internet_grass = np.array(mag_list[2])
theta_internet_grass = np.array(theta_list[2])

mag_living_labs = np.array(mag_list[3])
theta_living_labs = np.array(theta_list[3])

# Diagonal DF
diagonal_hog_df <- data.frame(mag = as.vector(py$mag_diagonal),
                                 theta = as.vector((py$theta_diagonal))) %>%
  mutate(radian = theta*(pi/180))

# San Francisco DF
sf_hog_df <- data.frame(mag = as.vector(py$mag_sf),
                         theta = as.vector((py$theta_sf))) %>%
  mutate(radian = theta*(pi/180))

# Internet Grass DF
internet_grass_hog_df <- data.frame(mag = as.vector(py$mag_internet_grass),
                                         theta = as.vector((py$theta_internet_grass))) %>%
  mutate(radian = theta*(pi/180))

# Living Labs DF
living_labs_hog_df <- data.frame(mag = as.vector(py$mag_living_labs),
                                    theta = as.vector((py$theta_living_labs))) %>%
  mutate(radian = theta*(pi/180))

# List of all Data frames
standard_df_list = list(diagonal_hog_df,
                        sf_hog_df,
                        internet_grass_hog_df,
                        living_labs_hog_df)

```

4.6 Plot Magnitudes as Image

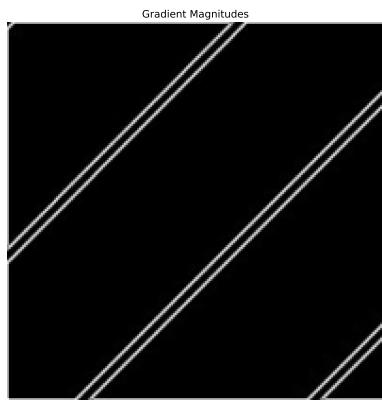
```

plt.figure(figsize=(15, 8))
plt.title('Gradient Magnitudes')
plt.imshow(mag_list[0], cmap="gray")
plt.axis("off")

```

(-0.5, 199.5, 199.5, -0.5)

```
plt.show()
```



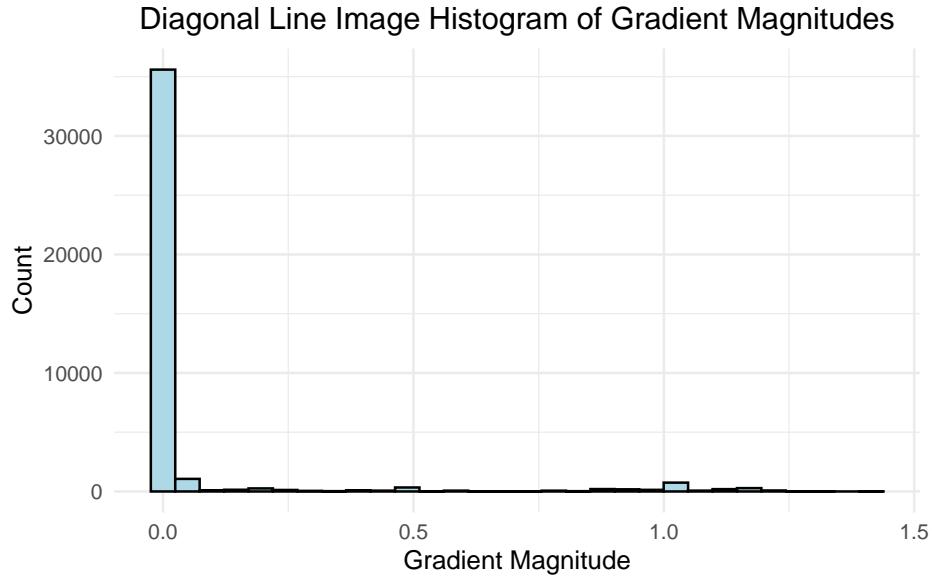
```
plt.savefig("mag.png", dpi=300)
```

4.7 Create Histogram Plots of Gradient Magnitudes and Angles

```
diagonal_histogram_mag_plot <-
  ggplot(standard_df_list[[1]],
         aes(x = mag)) +
  geom_histogram(colour = "black", fill = "lightblue") +
  scale_x_continuous() +
  labs(x = "Gradient Magnitude",
       y = "Count",
       title = "Diagonal Line Image Histogram of Gradient Magnitudes"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

diagonal_histogram_mag_plot

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
diagonal_mag_filter <- 0.1

ggsave("images/plots/diagonal_histogram_mag_plot.jpg", diagonal_histogram_mag_plot, width =
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

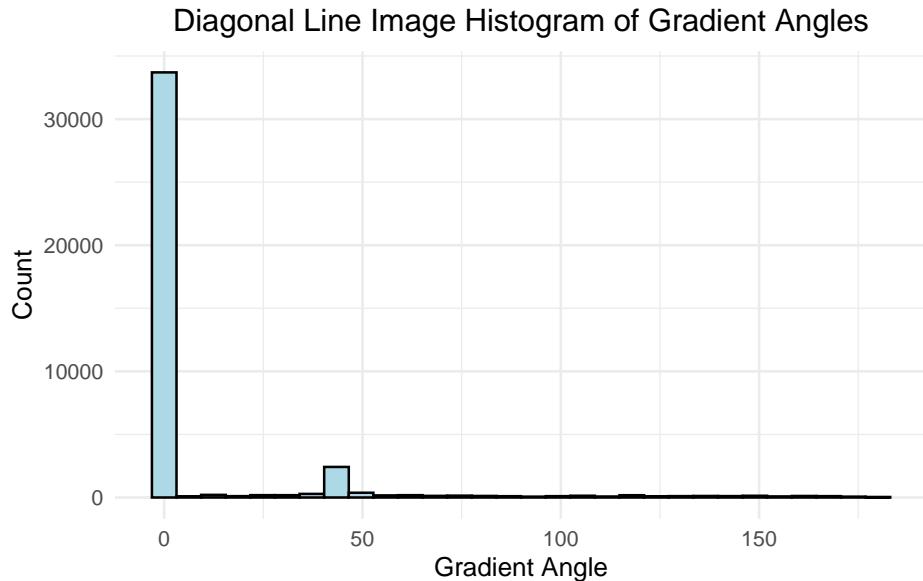
```

diagonal_histogram_theta_plot <-
  ggplot(standard_df_list[[1]],
         aes(x = theta)) +
  geom_histogram(colour = "black", fill = "lightblue") +
  scale_x_continuous() +
  labs(x = "Gradient Angle",
       y = "Count",
       title = "Diagonal Line Image Histogram of Gradient Angles"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

diagonal_histogram_theta_plot

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```



```

ggsave("images/plots/diagonal_histogram_theta_plot.jpg", diagonal_histogram_theta_plot, w

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

sf_histogram_mag_plot <-
  ggplot(standard_df_list[[2]],
         aes(x = mag)) +
  geom_histogram(colour = "black", fill = "lightblue") +

```

```

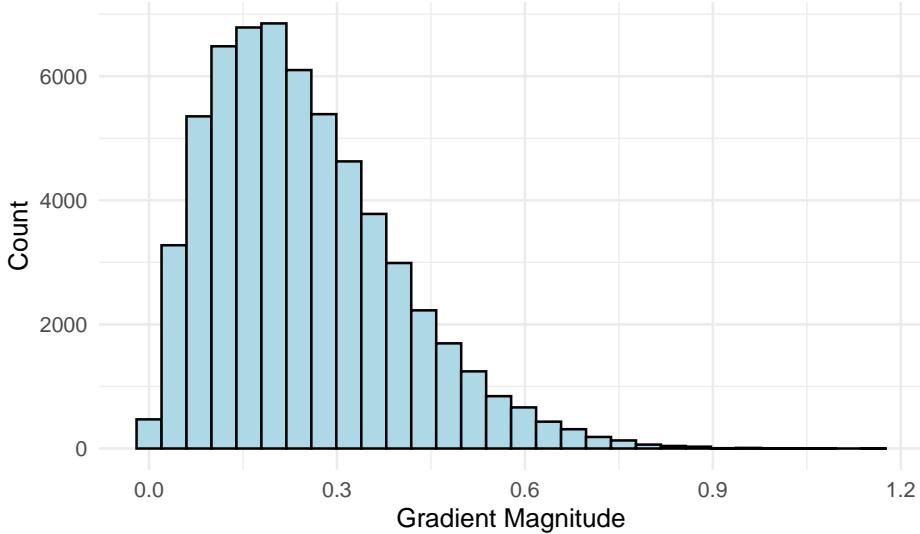
    scale_x_continuous() +
    labs(x = "Gradient Magnitude",
        y = "Count",
        title = "San Francisco Image Histogram of Gradient Magnitudes"
    ) +
    theme_minimal() +
    theme(plot.title = element_text(hjust = 0.5))

sf_histogram_mag_plot

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

San Francisco Image Histogram of Gradient Magnitudes



```

sf_mag_filter <- 0.4

ggsave("images/plots/sf_histogram_mag_plot.jpg", sf_histogram_mag_plot, width = 6, height = 4)

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

sf_histogram_theta_plot <-
  ggplot(standard_df_list[[2]],
         aes(x = theta)) +
  geom_histogram(colour = "black", fill = "lightblue") +
  scale_x_continuous() +
  labs(x = "Gradient Angle",
       y = "Count")

```

```

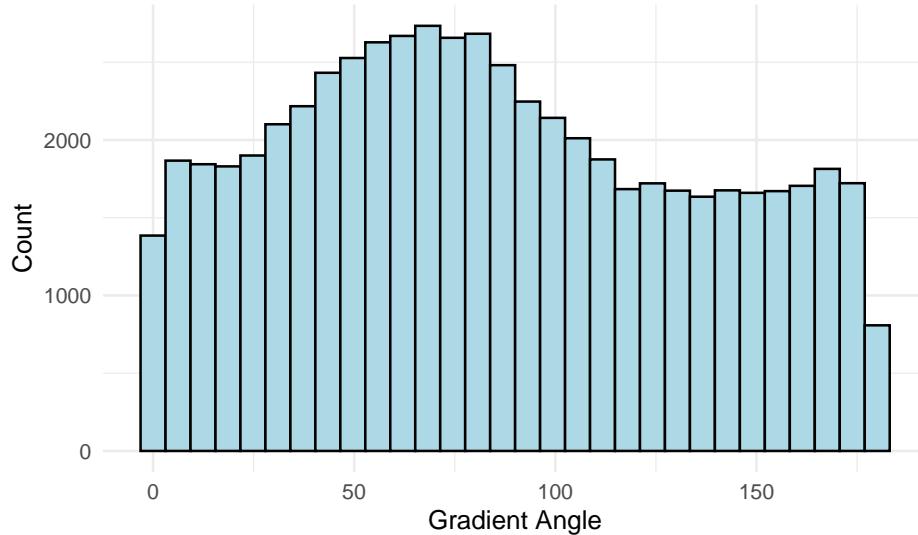
    y = "Count",
    title = "San Francisco Image Histogram of Gradient Angles"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

sf_histogram_theta_plot

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

San Francisco Image Histogram of Gradient Angles



```

ggsave("images/plots/sf_histogram_theta_plot.jpg", sf_histogram_theta_plot, width = 6, height = 4)

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

internet_grass_histogram_mag_plot <-
  ggplot(standard_df_list[[3]],
         aes(x = mag)) +
  geom_histogram(colour = "black", fill = "lightblue") +
  scale_x_continuous() +
  labs(x = "Gradient Magnitude",
       y = "Count",
       title = "Internet Grass Image Histogram of Gradient Magnitudes")
  +
  theme_minimal() +

```

```

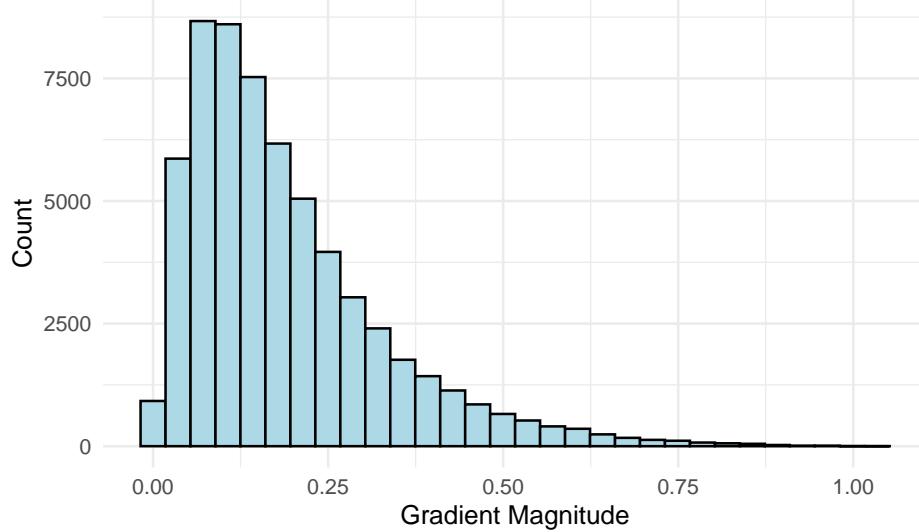
  theme(plot.title = element_text(hjust = 0.5))

internet_grass_histogram_mag_plot

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

```

Internet Grass Image Histogram of Gradient Magnitudes



```

internet_grass_mag_filter <- 0.3

ggsave("images/plots/internet_grass_histogram_mag_plot.jpg", internet_grass_histogram_mag

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

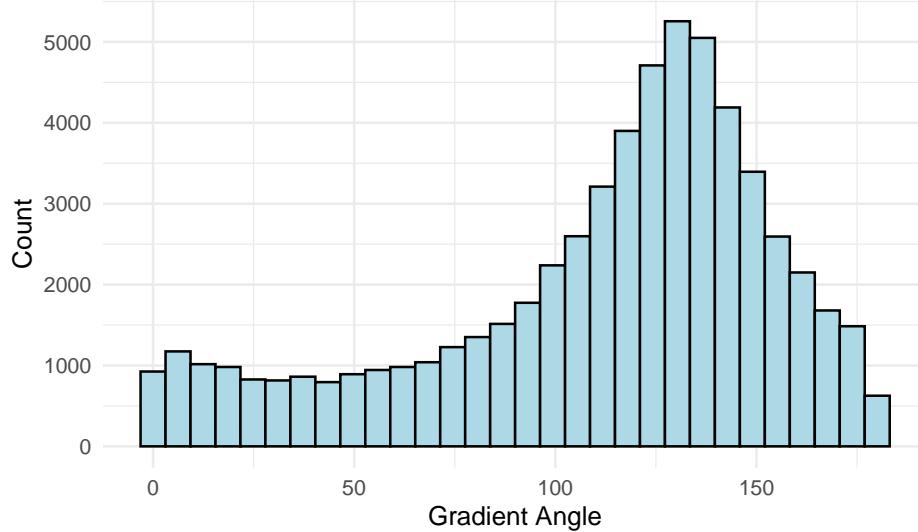
internet_grass_histogram_theta_plot <-
  ggplot(standard_df_list[[3]],
         aes(x = theta)) +
  geom_histogram(colour = "black", fill = "lightblue") +
  scale_x_continuous() +
  labs(x = "Gradient Angle",
       y = "Count",
       title = "Internet Grass Image Histogram of Gradient Angles"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

```

```
internet_grass_histogram_theta_plot
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Internet Grass Image Histogram of Gradient Angles



```
ggsave("images/plots/internet_grass_histogram_theta_plot.jpg", internet_grass_histogram_t
```

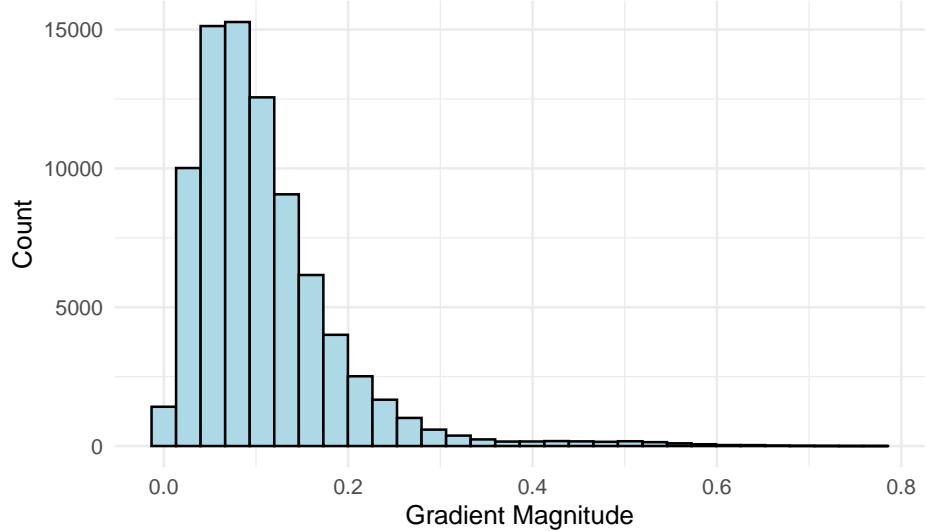
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
living_labs_histogram_mag_plot <-  
  ggplot(standard_df_list[[4]],  
         aes(x = mag)) +  
  geom_histogram(colour = "black", fill = "lightblue") +  
  scale_x_continuous() +  
  labs(x = "Gradient Magnitude",  
       y = "Count",  
       title = "Living Labs Grass Image Histogram of Gradient Magnitudes"  
    ) +  
  theme_minimal() +  
  theme(plot.title = element_text(hjust = 0.5))
```

```
living_labs_histogram_mag_plot
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Living Labs Grass Image Histogram of Gradient Magnitudes



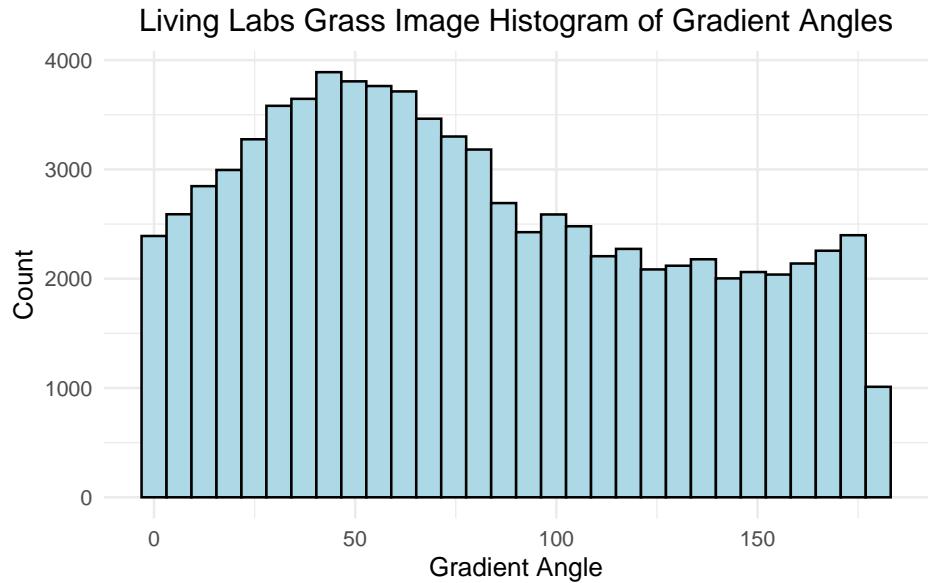
```
living_labs_mag_filter <- 0.15

ggsave("images/plots/living_labs_histogram_mag_plot.jpg", living_labs_histogram_mag_plot,
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

living_labs_histogram_theta_plot <-
  ggplot(standard_df_list[[4]],
         aes(x = theta)) +
  geom_histogram(colour = "black", fill = "lightblue") +
  scale_x_continuous() +
  labs(x = "Gradient Angle",
       y = "Count",
       title = "Living Labs Grass Image Histogram of Gradient Angles"
  ) +
  theme_minimal() +
  theme(plot.title = element_text(hjust = 0.5))

living_labs_histogram_theta_plot

`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



```
ggsave("images/plots/living_labs_histogram_theta_plot.jpg", living_labs_histogram_theta_p
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

4.8 Build New Contribution Histograms for Each Data Frame

```
# Define the number of bins
num_bins <- 9

# function to calculate the contributions to neighboring bins
calculate_bin_contributions <- function(angle, magnitude, num_bins) {
  bin_width <- 180 / num_bins
  contributions <- numeric(num_bins)

  # get the central bin
  central_bin <- floor(angle / bin_width) %% num_bins
  next_bin <- (central_bin + 1) %% num_bins

  # get contributions to neighboring bins
  weight <- (1 - abs((angle %% bin_width) / bin_width)) * magnitude

  contributions[central_bin + 1] <- weight
  contributions[next_bin + 1] <- magnitude - weight
}
```

```

        return(list(contributions[1] ,
                     contributions[2] ,
                     contributions[3] ,
                     contributions[4] ,
                     contributions[5] ,
                     contributions[6] ,
                     contributions[7] ,
                     contributions[8] ,
                     contributions[9]))
    )
}

filtered_standard_df_list <- list(diagonal_hog_df %>%
                                    filter(mag >= diagonal_mag_filter) ,
                                    sf_hog_df %>%
                                    filter(mag >= sf_mag_filter) ,
                                    internet_grass_hog_df %>%
                                    filter(mag >= internet_grass_mag_filter) ,
                                    living_labs_hog_df %>%
                                    filter(mag >= living_labs_mag_filter))
contribution_df_list <- list()

for (i in 1:length(filtered_standard_df_list)){
  contribution_hog_df <-
    filtered_standard_df_list[[i]] %>%
    filter(mag > 0.1) %>%
    rowwise() %>%
    mutate(`0` = calculate_bin_contributions(theta, mag, 9)[[1]],
           `20` = calculate_bin_contributions(theta, mag, 9)[[2]],
           `40` = calculate_bin_contributions(theta, mag, 9)[[3]],
           `60` = calculate_bin_contributions(theta, mag, 9)[[4]],
           `80` = calculate_bin_contributions(theta, mag, 9)[[5]],
           `100` = calculate_bin_contributions(theta, mag, 9)[[6]],
           `120` = calculate_bin_contributions(theta, mag, 9)[[7]],
           `140` = calculate_bin_contributions(theta, mag, 9)[[8]],
           `160` = calculate_bin_contributions(theta, mag, 9)[[9]],
           )
  split_histo_df <-
    contribution_hog_df %>%
    pivot_longer(names_to = "bin",
                 values_to = "contribution",

```

```

            cols = 4:ncol(contribution_hog_df)) %>%
mutate(bin = as.numeric(bin)) %>%
group_by(bin) %>%
summarise(contribution_sum = sum(contribution))

contribution_df_list[[i]] <- split_histo_df
}

```

4.9 Generate Polar Plots for Standard Histograms

```

diagonal_plot <-
  ggplot(filtered_standard_df_list[[1]],
         aes(x = theta)) +
  geom_histogram(colour = "black",
                 fill = "lightblue",
                 breaks = seq(0, 360, length.out = 17.5),
                 bins = 9) +
  coord_polar(
    theta = "x",
    start = 0,
    direction = 1) +
  scale_x_continuous(limits = c(0,360),
                     breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
                     labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"))
) +
  labs(title = "Polar Plot of Diagonal Line Image\nUsing Standard HOG Technique") +
  theme_minimal() +
  labs(x = "") +
  theme(axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5))

#diagonal_plot

sf_plot <-
  ggplot(filtered_standard_df_list[[2]],
         aes(x = theta)) +
  geom_histogram(colour = "black",
                 fill = "lightblue",

```

```

            breaks = seq(0, 360, length.out = 17.5),
            bins = 9) +
        coord_polar(
            theta = "x",
            start = 0,
            direction = 1) +
        scale_x_continuous(limits = c(0,360),
            breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
            labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"))
    )+
    labs(title = "Polar Plot of San Francisco Image\nUsing Standard HOG Technique") +
    theme_minimal() +
    labs(x = "") +
    theme(axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5))

#sf_plot

internet_grass_plot <-
    ggplot(filtered_standard_df_list[[3]],
        aes(x = theta)) +
    geom_histogram(colour = "black",
        fill = "lightblue",
        breaks = seq(0, 360, length.out = 17.5),
        bins = 9) +
    coord_polar(
        theta = "x",
        start = 0,
        direction = 1) +
    scale_x_continuous(limits = c(0,360),
        breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
        labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"))
    )+
    labs(title = "Polar Plot of Internet Grass Image\nUsing Standard HOG Technique") +
    theme_minimal() +
    labs(x = "") +
    theme(axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5))

#internet_grass_plot

living_labs_plot <-
    ggplot(filtered_standard_df_list[[4]],

```

```

    aes(x = theta)) +
  geom_histogram(colour = "black",
                 fill = "lightblue",
                 breaks = seq(0, 360, length.out = 17.5),
                 bins = 9) +
  coord_polar(
    theta = "x",
    start = 0,
    direction = 1) +
  scale_x_continuous(limits = c(0,360),
                     breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
                     labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"))
) +
  labs(title = "Polar Plot of Living Labs Grass Image\nUsing Standard HOG Technique") +
  theme_minimal() +
  labs(x = "") +
  theme(axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5))

#living_labs_plot

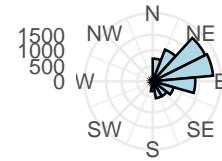
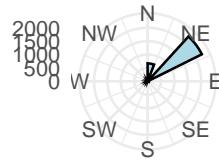
all_standard_plots <- ggpubr::ggarrange(diagonal_plot,
                                         sf_plot,
                                         internet_grass_plot,
                                         living_labs_plot)

ggsave("images/plots/all_standard_polar_plots.jpg",
       all_standard_plots, width = 7,
       height = 7)

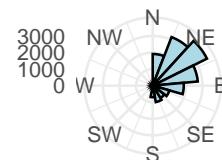
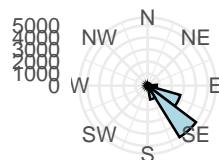
all_standard_plots

```

Polar Plot of Diagonal Line Image Polar Plot of San Francisco Image
Using Standard HOG Technique Using Standard HOG Technique



Polar Plot of Internet Grass Image Polar Plot of Living Labs Grass Image
Using Standard HOG Technique Using Standard HOG Technique



4.10 Generate Polar Plots for Contribution Histograms

```
diagonal_split_plot <-
  ggplot(contribution_df_list[[1]],
         aes(x = bin, y = contribution_sum)) +
  geom_histogram(stat = "identity",
                 colour = "black",
                 fill = "lightblue",
                 breaks = seq(0, 360, length.out = 17.5),
                 bins = 9) +
  coord_polar(
    theta = "x",
    start = 0,
    direction = 1) +
  scale_x_continuous(limits = c(0,360),
                     breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
                     labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"))
) +
  labs(title = "Polar Plot of Diagonal Line Image\nUsing Distributed HOG Technique") +
  theme_minimal() +
  labs(x = "") +
  theme(axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5))
```

Warning in geom_histogram(stat = "identity", colour = "black", fill =

```

"lightblue", : Ignoring unknown parameters: `binwidth`, `bins`, `pad`, and
`breaks`  

#diagonal_split_plot  
  

sf_split_plot <-
  ggplot(contribution_df_list[[2]],
         aes(x = bin, y = contribution_sum)) +
  geom_histogram(stat = "identity",
                 colour = "black",
                 fill = "lightblue",
                 breaks = seq(0, 360, length.out = 17.5),
                 bins = 9) +
  coord_polar(
    theta = "x",
    start = 0,
    direction = 1) +
  scale_x_continuous(limits = c(0,360),
                     breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
                     labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW"))
) +
  labs(title = "Polar Plot of San Francisco Image\nUsing Distributed HOG Technique") +
  theme_minimal() +
  labs(x = "") +
  theme(axis.title.y = element_blank(),
        plot.title = element_text(hjust = 0.5))  
  

Warning in geom_histogram(stat = "identity", colour = "black", fill =
"lightblue", : Ignoring unknown parameters: `binwidth`, `bins`, `pad`, and
`breaks`  

#sf_split_plot  
  

internet_grass_split_plot <-
  ggplot(contribution_df_list[[3]],
         aes(x = bin, y = contribution_sum)) +
  geom_histogram(stat = "identity",
                 colour = "black",
                 fill = "lightblue",
                 breaks = seq(0, 360, length.out = 17.5),
                 bins = 9) +
  coord_polar(
    theta = "x",
    start = 0,

```

```

direction = 1) +
scale_x_continuous(limits = c(0,360),
  breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
  labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW")
) +
labs(title = "Polar Plot of Internet Grass Image\nUsing Distributed HOG Technique") +
theme_minimal() +
labs(x = "") +
theme(axis.title.y = element_blank(),
  plot.title = element_text(hjust = 0.5))

Warning in geom_histogram(stat = "identity", colour = "black", fill =
"lightblue", : Ignoring unknown parameters: `binwidth`, `bins`, `pad`, and
`breaks`


#internet_grass_split_plot


living_labs_split_plot <-
ggplot(contribution_df_list[[4]],
  aes(x = bin, y = contribution_sum)) +
geom_histogram(stat = "identity",
  colour = "black",
  fill = "lightblue",
  breaks = seq(0, 360, length.out = 17.5),
  bins = 9) +
coord_polar(
  theta = "x",
  start = 0,
  direction = 1) +
scale_x_continuous(limits = c(0,360),
  breaks = c(0, 45, 90, 135, 180, 225, 270, 315),
  labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW")
) +
labs(title = "Polar Plot of Living Labs Aerial Image\nUsing Distributed HOG Technique") +
theme_minimal() +
labs(x = "") +
theme(axis.title.y = element_blank(),
  plot.title = element_text(hjust = 0.5))

Warning in geom_histogram(stat = "identity", colour = "black", fill =
"lightblue", : Ignoring unknown parameters: `binwidth`, `bins`, `pad`, and
`breaks`


#living_labs_split_plot

```

```

all_contribution_plots <- ggpubr::ggarrange(diagonal_split_plot,
                                              sf_split_plot,
                                              internet_grass_split_plot,
                                              living_labs_split_plot)

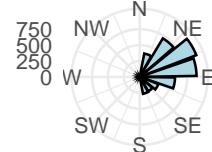
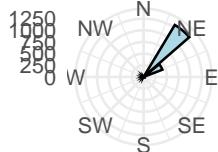
Warning: Removed 1 row containing missing values or values outside the scale range
(`geom_bar()`).
Removed 1 row containing missing values or values outside the scale range
(`geom_bar()`).
Removed 1 row containing missing values or values outside the scale range
(`geom_bar()`).
Removed 1 row containing missing values or values outside the scale range
(`geom_bar()`).

ggsave("images/plots/all_contribution_polar_plots.jpg",
       all_contribution_plots, width = 7,
       height = 7)

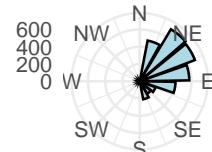
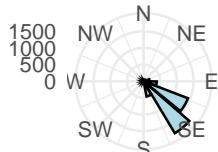
all_contribution_plots

```

Polar Plot of Diagonal Line Image
Using Distributed HOG Technique



Polar Plot of Internet Grass Image
Using Distributed HOG Technique



```

number_of_bins = 9
step_size = 180 / number_of_bins

```

```

#Function to calculate the value of centre of jth bin
def calculate_j(angle):
    temp = (angle / step_size) - 0.5
    j = math.floor(temp)
    return j

# Function to calculate the value of jth bin
def calculate_Cj(j):
    Cj = step_size * (j + 0.5)
    return round(Cj, 9)

#
def calculate_value_j(magnitude, angle, j):
    Cj = calculate_Cj(j+1)
    Vj = magnitude * ((Cj - angle) / step_size)
    return round(Vj, 9)

histogram_points_nine = []
high_val = 10
# for i in range(0, height, high_val):
#     temp = []
#     for j in range(0, width, high_val):
#         magnitude_values = [[mag[i][x] for x in range(j, j+high_val)] for i in range(i,i+high_val)]
#         angle_values = [[theta[i][x] for x in range(j, j+high_val)] for i in range(i, i+high_val)]
#         for k in range(len(magnitude_values)):
#             for l in range(len(magnitude_values[0])):
#                 bins = [0.0 for _ in range(number_of_bins)]
#                 value_j = calculate_j(angle_values[k][l])
#                 Vj = calculate_value_j(magnitude_values[k][l], angle_values[k][l], value_j)
#                 Vj_1 = magnitude_values[k][l] - Vj
#                 bins[value_j]+=Vj
#                 bins[value_j+1]+=Vj_1
#                 bins = [round(x, 9) for x in bins]
#                 temp.append(bins)
#             histogram_points_nine.append(temp)
#         #
#         # print(len(histogram_points_nine))
#         # print(len(histogram_points_nine[0]))
#         # print(len(histogram_points_nine[0][0]))

```

```

epsilon = 1e-05

# feature_vectors = []
# for i in range(0, len(histogram_points_nine) - 1, 1):
#     temp = []
#     for j in range(0, len(histogram_points_nine[0]) - 1, 1):
#         values = [[histogram_points_nine[i][x] for x in range(j, j+2)] for i in range(i, i+2)]
#         final_vector = []
#         for k in values:
#             for l in k:
#                 for m in l:
#                     final_vector.append(m)
#         k = round(math.sqrt(sum([pow(x, 2) for x in final_vector])), 9)
#         final_vector = [round(x/(k + epsilon), 9) for x in final_vector]
#         temp.append(final_vector)
#     feature_vectors.append(temp)
#
# print(len(feature_vectors))
# print(len(feature_vectors[0]))
# print(len(feature_vectors[0][0]))

```

4.11 Generate HOG Image

```

img = imread("images/living_lab_aerial/aerial_grass_living_lab_rotated.jpg")
img = color.rgb2gray(io.imread("images/grass_image2.jpg"))

aspect_ratio = img.shape[0]/img.shape[1]

height = 200
width = int(height/aspect_ratio)

# height = 128
# width = 192

# make sure the resized is in sample ball park as the original aspect ratio,
# that way the angles don't get squished
resized_ratio = height/width

resized_img = resize(img, (height, width))

plt.axis("off")

```

```
(0.0, 1.0, 0.0, 1.0)

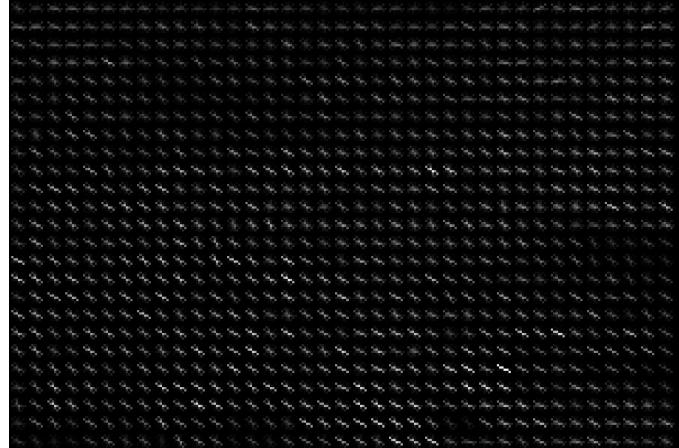
plt.imshow(resized_img)
print(resized_img.shape)

(200, 301)

fd, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8, 8),
                     cells_per_block=(2, 2), visualize=True#, channel_axis = 2
                     #multichannel=True
                     )
plt.axis("off")

(-0.5, 300.5, 199.5, -0.5)

plt.imshow(hog_image, cmap="gray")
plt.show()
```



```
plt.savefig('images/plots/grass2_image_hog.jpg')
```

```

# grass2_polar_plot <-
# ggplot(diagonal_hog_df, #>% filter(mag >= 0.1),
#         aes(x = radian)) +
#   geom_histogram(colour = "black", fill = "lightblue",
#                 breaks = seq(0, 2*pi, length.out = 17.5),
#                 bins = 9) +
#   coord_polar(
#     theta = "x", start = 0, direction = 1) +
#   scale_x_continuous(
#     breaks = c(0, pi/4, pi/2, 3*pi/4, pi, 5*pi/4, 3*pi/2, 7*pi/4),
#     labels = c("N", "NE", "E", "SE", "S", "SW", "W", "NW")
#   ) +
#   labs(title = "Polar Plot of Internet Grass Image") +
#   theme_minimal() +
#   labs(x = "") +
#   theme(axis.title.y = element_blank(),
#         plot.title = element_text(hjust = 0.5))
#
# grass2_polar_plot

#ggsave("_polar_plot.jpg", polar_plot, width = 6, height = 4, dpi = 300)

#all_plots <- ggpubr::ggarrange(diagonal_polar_plot, sf_polar_plot, grass2_polar_plot, ae
#ggsave("images/plots/results_all_plots.jpg", all_plots, width = 7, height = 7)

```

4.12 HOG Image of Internet Grass

```
from skimage import color, io, exposure
from skimage.transform import resize
import matplotlib.pyplot as plt
from skimage.feature import hog

# Load the image and preprocess it
img = color.rgb2gray(io.imread("images/grass_image2.jpg"))
# img = color.rgb2gray(io.imread("diagnol_lines_flipped.jpg"))

aspect_ratio = img.shape[0] / img.shape[1]
height = 200
width = int(height / aspect_ratio)
resized_img = resize(img, (height, width))

plt.figure(figsize=(8, 20)) # Adjusted the figure size to accommodate the additional obj
plt.imshow(resized_img, cmap="gray")
plt.axis("off")

(-0.5, 300.5, 199.5, -0.5)

plt.show()
```



```

# Compute HOG features
hog_features, hog_image = hog(resized_img, orientations=9, pixels_per_cell=(8,8),
                               cells_per_block=(10, 10), visualize=True)

# Plot the images
fig, (ax1, ax2, ax3) = plt.subplots(3, 1, figsize=(8, 20), sharex=True, sharey=True) # O

# Plot the rescaled black and white image
ax1.imshow(resized_img, cmap=plt.cm.gray)
ax1.set_title('Rescaled Black and White Image')

# Plot the mag object
ax2.imshow(mag, cmap=plt.cm.gray) # Assuming mag is the object you want to insert
#ax2.axis("off")
ax2.set_title('Pixel Magnitudes')

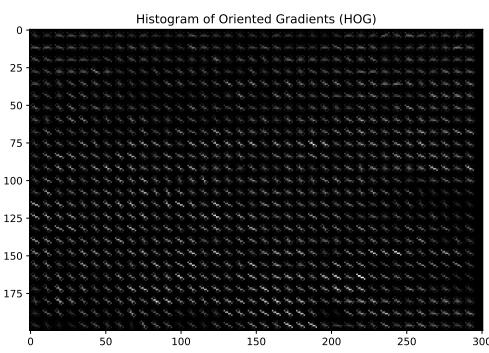
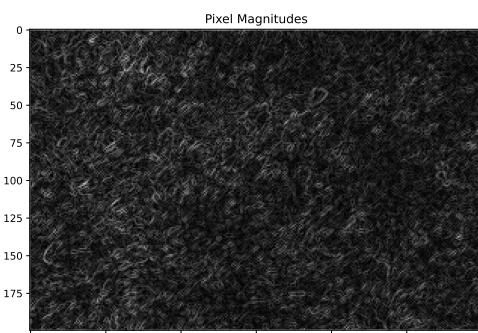
# rescale HOG for better viewing:
hog_color_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

# Plot the histogram of oriented gradients
ax3.imshow(hog_color_rescaled, cmap=plt.cm.gray)
ax3.set_title('Histogram of Oriented Gradients (HOG)')

plt.savefig("images/plots/rescaled_grass2_image_hog.png", dpi=300)

plt.show()

```



```

from skimage import color, io, exposure
from skimage.transform import resize
import matplotlib.pyplot as plt
from skimage.feature import hog

# Load the image and preprocess it
img = io.imread("images/grass_image2.jpg")
# img = color.rgb2gray(io.imread("diagnol_lines_flipped.jpg"))

aspect_ratio = img.shape[0] / img.shape[1]
height = 200
width = int(height / aspect_ratio)
resized_img = resize(img, (height, width))

bw_resized_image = color.rgb2gray(resized_img)

plt.figure(figsize=(15, 5)) # Adjusted the figure size to accommodate the additional obj
plt.imshow(resized_img, cmap="gray")
plt.axis("off")

(-0.5, 300.5, 199.5, -0.5)

plt.show()

```



```

# Compute HOG features
hog_features, hog_image = hog(bw_resized_image, orientations=9, pixels_per_cell=(8,8),
                               cells_per_block=(10, 10), visualize=True)

# Plot the images
fig, (ax1, ax2, ax3) = plt.subplots(1, 3, figsize=(25, 5), sharex=True, sharey=True) # C

# Plot the rescaled input image

```

```

ax1.imshow(resized_img, cmap=plt.cm.gray)
ax1.set_title('Rescaled Input Image')

# Plot the pixel magnitudes
ax2.imshow(mag, cmap=plt.cm.gray) # Assuming mag is the object you want to insert
ax2.set_title('Pixel Magnitudes')

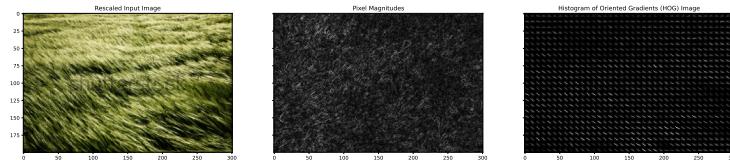
# Plot the histogram of oriented gradients
hog_color_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))
ax3.imshow(hog_color_rescaled, cmap=plt.cm.gray)
ax3.set_title('Histogram of Oriented Gradients (HOG) Image')

# Plot the histogram of oriented gradients
# angle_hist = io.imread("images/grass2_angles_histogram.jpg")
# resized_hist = resize(angle_hist, (height, width))
# ax4.imshow(resized_hist, cmap=plt.cm.gray)
# ax4.set_title('Angle Histogram')
#
# # Plot the polar plot
# polar_plot = io.imread("images/grass2_polar_plot.jpg")
# resized_polar = resize(polar_plot, (height, width))
# ax5.imshow(resized_polar, cmap=plt.cm.gray)
# ax5.set_title('Polar Plot')

plt.savefig("images/plots/rescaled_grass2_image_hog.png", dpi=300)

plt.show()

```



4.13 Titus Flip

```

image = imread('images/TitusFlip.jpg')
imshow(image)
print(image.shape)

```

(1350, 1080, 3)

```

resized_image = resize(image, (300, 400))
#resized_image = image
imshow(resized_image)
print(resized_image.shape)

(300, 400, 3)

fig, hog_image = hog(resized_image, orientations=9, pixels_per_cell=(3, 3),
                      cells_per_block=(15, 15), visualize=True, channel_axis=2 #multichannel
)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(16, 7), sharex=True, sharey=True)

ax1.imshow(resized_image, cmap=plt.cm.gray)
ax1.set_title('Input image')

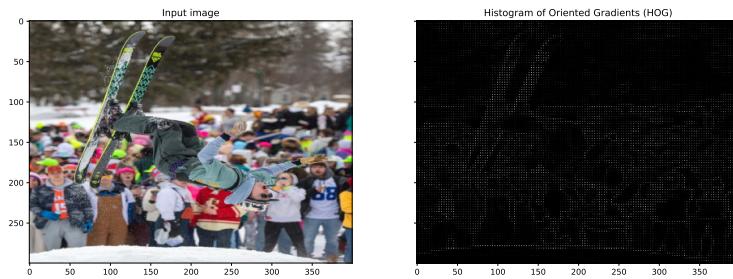
# Rescale histogram for better display
hog_color_rescaled = exposure.rescale_intensity(hog_image, in_range=(0, 10))

ax2.imshow(hog_color_rescaled, cmap=plt.cm.gray)
ax2.set_title('Histogram of Oriented Gradients (HOG)')

# store to file
plt.savefig("images/plots/titus_flip_example_hog.png", dpi=300)

plt.show()

```





```

# hog_df <- py$hog_df

# ggplot(hog_df %>% filter(mag >= 0.4),
#         aes(x = radian)) +
#     geom_histogram(#binwidth = 5#, boundary = 0, closed = "right") +
# )+
# #scale_x_continuous(limits = c(0, 360), breaks = seq(0, 360, by = 45)) +
# #coord_polar(start = 0, direction = 1, ) +
# coord_radial(start = 0, end = pi, expand = F, clip = "on") +
# scale_x_continuous(
#     breaks = c(0, pi/4, pi/2, 3*pi/4),
#     labels = c("0", "/4", "/2", "3/4")
# ) +
# theme(plot.title = element_text(hjust = 0.5)) +
# labs(title = "Polar Histogram of Theta",
#      x = "Theta (Degrees)",
#      y = "Frequency") #+ theme_minimal()

```

Chapter 5

Conclusion