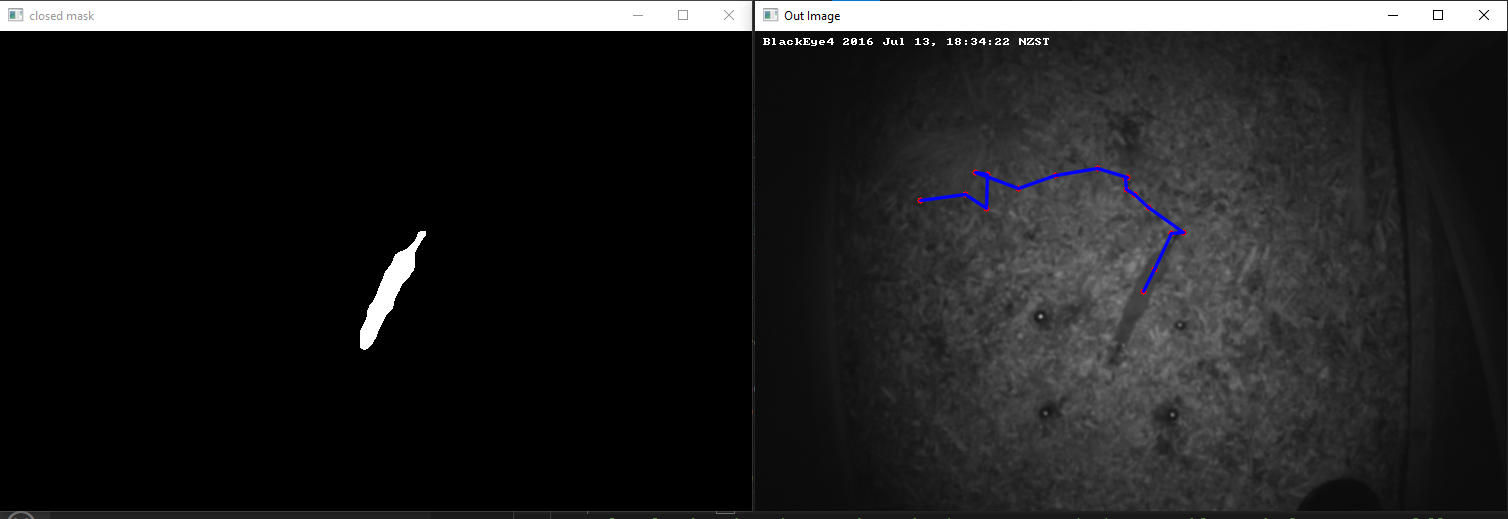
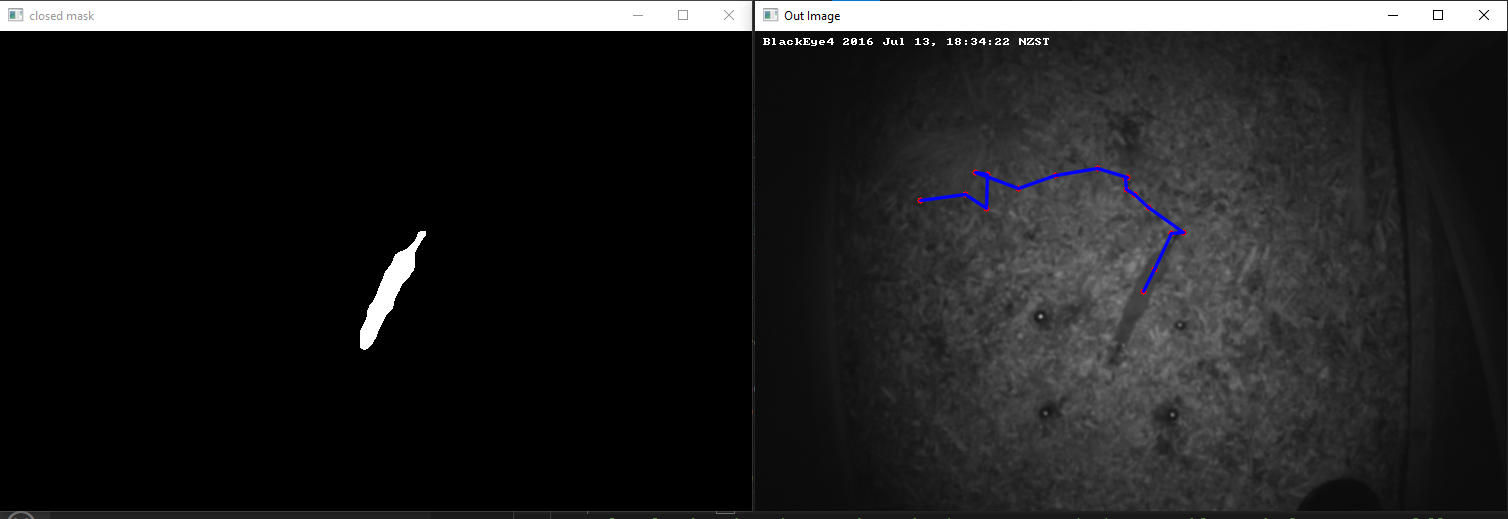
281.756 Image and Video Processing

Algorithm Development Project

Predator Detection Project





Stanley Jackson 17073907

Other Group Members:

Justin Larking

Jack Downing

# Abstract

A computer vision algorithm was developed to detect, track the movement of, and identify predators in a video sequence. The algorithm was tested on a set of 7 sequences and was able to detect if an animal was present, and identify the type of animal, with more than 80% accuracy. The algorithm uses background averaging and subtraction, along with morphological operations and contours detection, to detect and identify the animal. It performs poorly in situations with low contrast between background and animal, and in situations where the background changes throughout the video sequence.

Table of Contents

[Abstract 2](#_Toc83988556)

[Table of Figures and Tables 4](#_Toc83988557)

[1.0 Introduction 5](#_Toc83988558)

[2.0 Algorithm 6](#_Toc83988559)

[2.1 Block Diagram 6](#_Toc83988560)

[2.2 Background Image Creation 7](#_Toc83988561)

[2.3 Animal Mask 7](#_Toc83988562)

[2.3.1 Difference Calculation 7](#_Toc83988563)

[2.3.2 Threshold 7](#_Toc83988564)

[2.3.3 Morphological Operations 8](#_Toc83988565)

[2.4 Detecting and Identifying Animal 9](#_Toc83988566)

[2.4.1 Contour Area 9](#_Toc83988567)

[2.4.2 Centre of Mass (CoM) 9](#_Toc83988568)

[2.4.3 Aspect Ratio 9](#_Toc83988569)

[2.4.4 Other Features 10](#_Toc83988570)

[2.4.5 Averaging Animal Identification Over the Sequence 10](#_Toc83988571)

[2.5 Assumptions 10](#_Toc83988572)

[2.5.1 One Animal Per Sequence 10](#_Toc83988573)

[2.5.2 Good Contrast 10](#_Toc83988574)

[2.5.3 Lens Distortion Insignificant 10](#_Toc83988575)

[2.5.4 Only Rats, Stoats and Hedgehogs 10](#_Toc83988576)

[2.5.5 Good Difference in Aspect Ratio 11](#_Toc83988577)

[2.5.6 The Background Does Not Change as the Animal Moves Through 11](#_Toc83988578)

[3.0 Results 11](#_Toc83988579)

[4.0 Discussion 13](#_Toc83988580)

[4.1 Misidentification of Stoat/Rat 13](#_Toc83988581)

[4.2 False Positive Detection of Animal 14](#_Toc83988582)

[4.2.1 Sequences 3 and 4 14](#_Toc83988583)

[4.2.2 Sequence 7 14](#_Toc83988584)

[4.3 Missing Detection of Animal 15](#_Toc83988585)

[4.3.1 Animal on Edge of Frame 15](#_Toc83988586)

[4.3.2 Animal Obscured 15](#_Toc83988587)

[4.4 Overall Detection Rates 15](#_Toc83988588)

[4.5 Why do Some Sequences do Better Than Others 16](#_Toc83988589)

[4.6 Results of Failure 16](#_Toc83988590)

[5.0 Conclusion 16](#_Toc83988591)

[6.0 Appendix 17](#_Toc83988592)

[6.1 Code 17](#_Toc83988593)

# Table of Figures and Tables

[Figure 1 Block diagram of algorithm 6](#_Toc83988507)

[Figure 2 Difference between background image and current image 7](#_Toc83988508)

[Figure 3 Image as a result of threshold 8](#_Toc83988509)

[Figure 4 Animal mask after morphological operations have been performed 8](#_Toc83988510)

[Figure 5 Sequence 1 output image 11](#_Toc83988511)

[Figure 6 Sequence 2 output image 12](#_Toc83988512)

[Figure 7 Sequence 3 output image 12](#_Toc83988513)

[Figure 8 Sequence 4 output image 12](#_Toc83988514)

[Figure 9 Sequence 5 output image 13](#_Toc83988515)

[Figure 10 Sequence 6 output image 13](#_Toc83988516)

[Figure 11 Sequence 7 output image 13](#_Toc83988517)

[Figure 12 Sequence 4 output where a stoat was misidentified as a rat. 14](#_Toc83988518)

[Figure 13 Images showing rain up in animal mask and being detected as an animal 14](#_Toc83988519)

[Figure 14 Images from Seq7 where the circled sticks appear in the difference image 15](#_Toc83988520)

[Figure 15 Images from Seq4 where a stoat (circled) is on the edge of the frame but is not shown in the difference image 15](#_Toc83988521)

[Figure 16 Output image from seq4 showing how the rain caused random detections. This resulted in the messy path shown. 16](#_Toc83988522)

[Table 1 Table of testing results 11](#_Toc83988541)

[Table 2 Table of output results 11](#_Toc83988542)

# 1.0 Introduction

Conservationists frequently set up cameras to watch bird nests or other wildlife habitats. Image and video processing can be used to assist with the detection and identification of animals in the video sequences from these cameras. The team was given a set of seven short video sequences and have been asked to develop an algorithm to extract data from the sequences. The algorithm needs to detect any animals in the images, track the path taken by the animal, and identify what type of animal is in the sequence. The sequences have several factors which make this a non-trivial task. Factors such as rain, wind, sunlight, and complex backgrounds will need to be considered while developing the algorithm.

# 2.0 Algorithm

## 2.1 Block Diagram

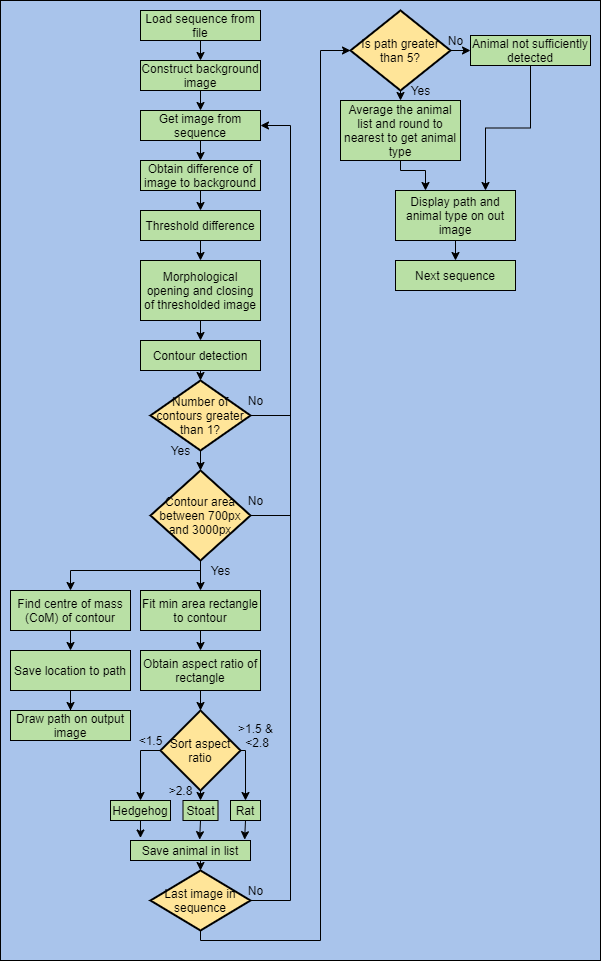


Figure Block diagram of algorithm

## 2.2 Background Image Creation

To create the background image of the sequence, a 75th percentile of the entire sequence is used. In NumPy, the percentile function looks at each pixel in the images and sorts them in order from dark to light. It then takes the value 75% of the way up that list. In the sequences of images provided, the animals in the image appear to be darker than the background. This means pixels in images where the animal is present, are more likely to be at the bottom of the sorted list. Therefore, the animals are not included in the background model. This allows the difference between the background image and the current image to show, mostly, just the animal.

## 2.3 Animal Mask

A mask containing the animal in the image was constructed by doing a difference calculation, thresholding the difference, and performing morphological operations.

### 2.3.1 Difference Calculation

Some sequences have been captured at daytime. These have shadows to deal with. To prevent shadows from being detected as animals, only the positive difference is measured. This means only regions which are brighter than the background will appear in the difference image.

The other sequences have been captured at night-time where shadows are not an issue. For these sequences, the absolute difference is used. This way, even regions that are darker than the background will appear in the difference image.

To determine if a sequence is day or night, the average pixel value of the image is measured. Daytime images had a higher average pixel. If the average pixel value was above 80, then the image was assumed to be a daytime image. The value of 80 was selected by looking at the average pixel value of each sequence. The daytime sequences had values greater than about 90, while night-time were below about 60.

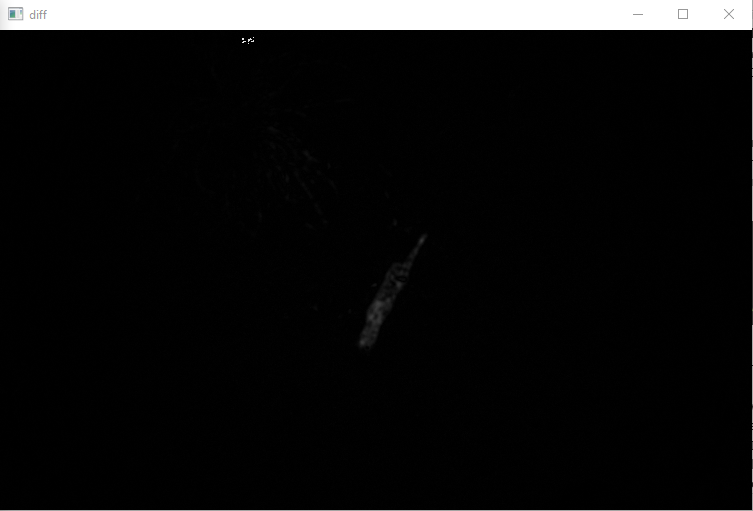


Figure Difference between background image and current image

### 2.3.2 Threshold

To create a binary image containing the animal, the difference between the background image and current image is thresholded. If the difference between the pixels is greater than a certain value (8 was used), then that pixel is shown as white in the binary image. If not, it shows as black. The threshold was set at a value of 8 as this was low enough to capture the changes due to the animal but minimised the changes due to noise. This threshold was found through testing.

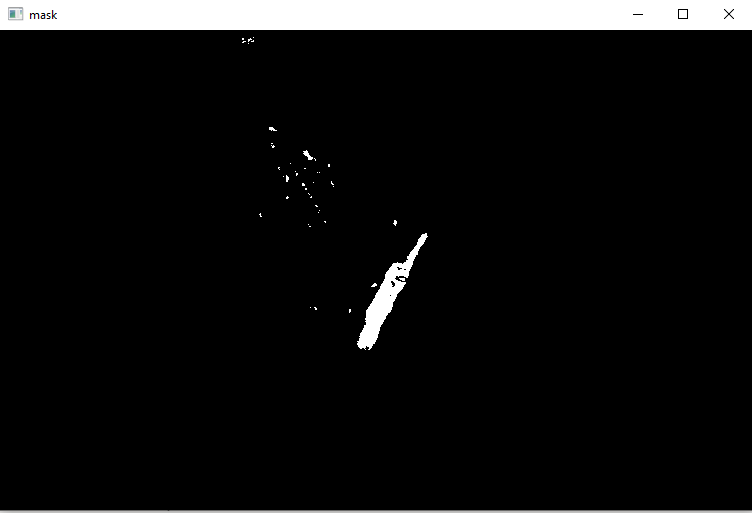


Figure Image after threshold

### 2.3.3 Morphological Operations

After thresholding, there was still some unwanted pixels in the image because of noise. To help remove unwanted information, a series of morphological opening and closing operations are used.

First, a closing operation, with a small (3,3) kernel, is used to help join some chunks of “animal” that may be separated by just one or two pixels. The small kernel size helps to prevent the closing operation from joining chunks of noise together.

Next, an opening operation, with a small (5,5) kernel, is used to help remove any noise in the image. The small kernel size helps to ensure that the larger chunks, corresponding to potential animals, are not accidently removed.

Finally, a closing operation, with a large (40,40) kernel, is used to help join chunks of potential animals. Some parts of the animal get lost because of the thresholding. This closing operation attempts to recover that information.

The result is a black and white image that shows potential animals in the image.

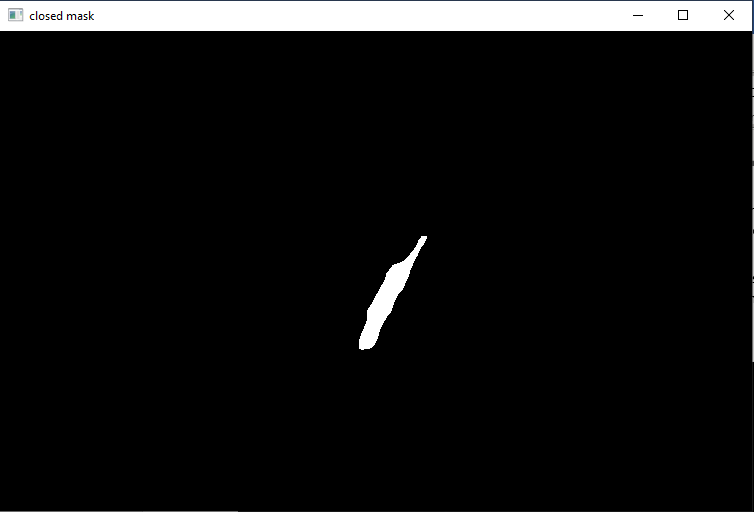


Figure Animal mask after morphological operations have been performed

## 2.4 Detecting and Identifying Animal

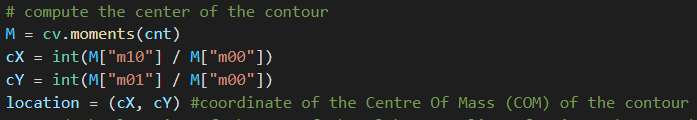
To detect and identify the animal, the size and aspect ratio of the contours are measured. The centre of mass of the largest contour was used to identify the location of the animal in the image

### 2.4.1 Contour Area

For each image, the contour with the largest area is found. Due to the previous morphological operations, this contour is likely to be the animal in the image. If the area of the contour is less than 700px, it is ignored. This is because anything smaller than this is not likely to be an animal. If the area is greater than 3000px, it is also ignored. This was done to reduce the number of false detections because of rain or bugs flying across the screen close to the camera. Objects close to the camera appear larger so adding this condition helped to remove those detections without removing animal detections. The values 700 and 3000 were selected by running through some of the sequences and printing out the size of the largest contour. Images without an animal had contours smaller than 700px. Images with animals had contours with areas less than 3000px, except for images with rain streaks.

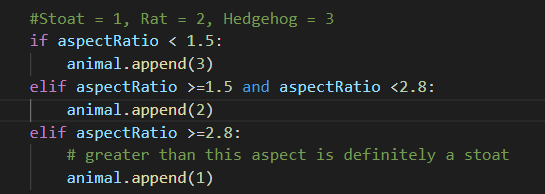
### 2.4.2 Centre of Mass (CoM)

Once a contour has been identified as an animal, its location needs to be recorded. This will later be used to plot the path the animal takes through the image. To find the location of the animal, the CoM of the contour is used. This is done by finding the M0,0, M1,0 and M0,1­ moments of the contour. These are then used to find the X and Y location of the CoM of the contour.



### 2.4.3 Aspect Ratio

To help distinguish between a rat, stoat and a hedgehog, the aspect ratio of the contour is used. Stoats have a long, thin body while hedgehogs are squarer in shape, with rats being somewhere in between. To find the aspect ratio of the contour, a minimum area rectangle was drawn around the contour. The length of its longest and shortest sides was used to calculate the aspect ratio (long/short). An aspect ratio of 1.0 represented a square, while greater than 1 indicated a rectangular shape. By looking at the aspect ratio of the animals identified in the sequences, it was observed that hedgehogs had an aspect ratio around 1.0, stoats about 3.0, and rats about 2.5. From this information, the animal could be classified based on its aspect ratio.



### 2.4.4 Other Features

To improve the ability to distinguish different animals, additional contour features were looked at. The area, convexity and solidity were looked at.

Area was not a good choice as in some sequences, the camera was closer to the animal. This would increase the size of the animal contour, making it hard to distinguish a close-up rat and a faraway stoat.

Convexity was looked at as it was thought that the different animals would have differing convexities because of their tails or lack thereof. However, upon investigation, there was no obvious trend between the convexity and the type of animal.

Solidity was also investigated, but it too did not show any relation to the type of animal in the sequence.

### 2.4.5 Averaging Animal Identification Over the Sequence

In some frames of the sequence, the animal was only partially detected. To prevent these incomplete detections from affecting the final animal identification, the type identified was averaged over the sequence. This average was rounded to the nearest animal. If there were less than 5 animal detections in the sequence, then no animal was reported to have been found in the sequence.

## 2.5 Assumptions

### 2.5.1 One Animal Per Sequence

The algorithm has been developed on the assumption that there is only one animal in each sequence. This is valid, for the sequences provided, as the is only a maximum of one animal per sequence. However, in a real situation it cannot be guaranteed that there is only going to be one animal per image.

### 2.5.2 Good Contrast

The algorithm assumes there will be good contrast between the animal and the background. If the animal is not sufficiently different, the background subtraction will not show a difference. This will result in the animal not being detected. This assumption is not completely valid. Most of the sequences provided do show good contrast between the animal and the background. However, there is one sequence (Seq4) where the animal is very similar to the background. As a result, the algorithm struggles with this.

### 2.5.3 Lens Distortion Insignificant

The algorithm assumes that the lens distortion at the edges of the frame is not significant enough to stretch or change the aspect ratio of the animal. This is a valid assumption as the sequences provided show the animal mostly in the centre of the frame. However, in real situations there would be times where the animal may only appear in the edge of the frame. In those cases, this assumption would not be valid.

### 2.5.4 Only Rats, Stoats and Hedgehogs

Another assumption made is that only rats, stoats, and hedgehogs are going to be present in the sequences. For the provided sequences, this is valid. However, in a real-world situation there will be other animals, such as birds, appearing in the frame. In those situations, this is an invalid assumption.

### 2.5.5 Good Difference in Aspect Ratio

Another assumption is the difference in aspect ratio between animals. It is assumed that rats, stoats, and hedgehogs will all have different aspect ratios. This is not completely valid as for some sequences the aspect ratio of the detected animal is very similar to the aspect ratio of another animal. This causes the animal to be misidentified.

### 2.5.6 The Background Does Not Change as the Animal Moves Through

It is assumed that the background does not change as the animal moves through. This is invalid as the animal can move sticks and leaves on the ground. Most of these small movements can be removed by the morphological operations and the contour area size condition. However, there is one example in sequence 7 where the hedgehog moves some sticks as it moves. This movement get picked up and alters the result of the tracking.

# 3.0 Results

Table Table of testing results

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sequence Number** | **Animal Present** | **Animal Detected** | **Frames with Animal** | **Frames with Animal Detected** |
| 1 | Stoat | Stoat | 10 | 8 |
| 2 | Stoat | Stoat | 54 | 39 |
| 3 | None | None | 0 | 3 |
| 4 | Stoat | Rat | 63 | 44 |
| 5 | Rat | Rat | 96 | 96 |
| 6 | Stoat | Stoat | 66 | 57 |
| 7 | Hedgehog | Hedgehog | 24 | 23 |

Detections less than 80% shown in red. False Positives also included in the red.

Table Table of output results

|  |  |
| --- | --- |
| **Sequence Number** | **Output Image** |
| 1 | Figure Sequence 1 output image |
| 2 | Figure Sequence 2 output image |
| 3 | Figure Sequence 3 output image |
| 4 | Figure Sequence 4 output image |
| 5 | Figure Sequence 5 output image |
| 6 | Figure Sequence 6 output image |
| 7 | Figure Sequence 7 output image |

# 4.0 Discussion

## 4.1 Misidentification of Stoat/Rat

In sequence 4 the stoat is misidentified as a rat. This is because the aspect ratio of the contour in the animal mask is very similar to the aspect ratio of a rat. When looking at how the algorithm performs on this sequence, it was noted that it struggled to capture all the stoat’s body. This is because in this sequence the contrast between the stoat and the background was low. This makes the difference between the background image and the current image difficult to detect.

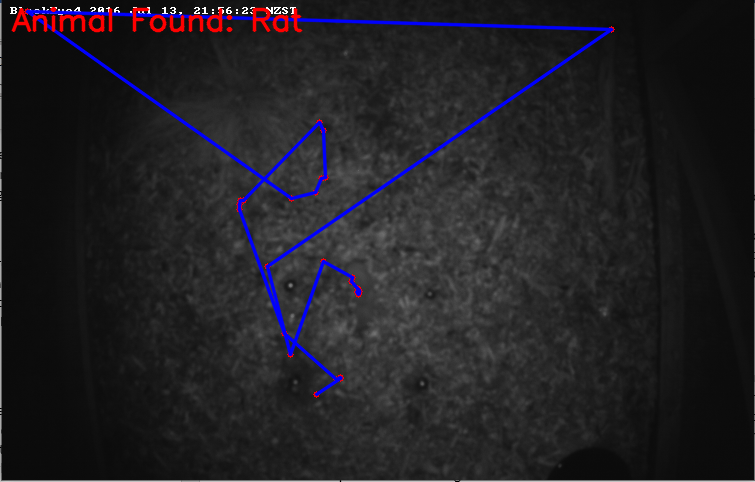


Figure Sequence 4 output where a stoat was misidentified as a rat.

## 4.2 False Positive Detection of Animal

In sequences 3,4 and 7 there was noticeable false detection.

### 4.2.1 Sequences 3 and 4

The false detection in sequence 3 and 4 is due to the rain in the images. The raindrops reflect the infra-red (IR) light back into the lens of the camera, resulting in a bright streak across the image. This bright streak is different to the background image and is above the threshold. Therefore, the raindrop is identified as a potential animal. The maximum size condition (3000px) on the contour area attempted to reduce the number of raindrops that were detected. This worked for some raindrops but some of the raindrops were small enough to be identified an animal in the frame.

|  |  |
| --- | --- |
|  |  |

Figure Images showing rain up in animal mask and being detected as an animal

### 4.2.2 Sequence 7

The false detection in sequence 7 was a result of the animal (hedgehog) moving some sticks in the image. These sticks appeared in the background image but in their moved position, not their original position. Therefore, in the first few frames of the sequence, the sticks were shown as a difference between the background image and current image, shown in the images below. Since the sticks were larger than the hedgehog as it entered the scene, the sticks were identified as the animal. This resulted in the algorithm incorrectly tracking the animal through the sequence.

|  |  |
| --- | --- |
|  |  |

Figure Images from Seq7 where the circled sticks appear in the difference image

## 4.3 Missing Detection of Animal

In most sequences, there are some frames where the algorithm fails to detect the animal that is in the image. These frames generally occur when the animal is on the edge of the frame or partially obscured.

### 4.3.1 Animal on Edge of Frame

Many of the sequences have been captured at night-time and a single light source has been used. This results in areas on the edge of the scene being darker, while the middle is bright. When the animal enters the frame from the edge it is hard to see as it blends in with the dark background, as shown in the images below. Therefore, the difference between the background image and current image is small and does not get detected. Some sort of region-based contrast adjust might improve the difference detection.

|  |  |
| --- | --- |
|  |  |

Figure Images from Seq4 where a stoat (circled) is on the edge of the frame but is not shown in the difference image

### 4.3.2 Animal Obscured

Some images in the sequences have animals but there are plants partially covering them. In these images, the size of the contour of the detected animal is not large enough to meet the 700px size requirement. This is a difficult error to fix as even the human eye can fail to see things when they are partially covered.

## 4.4 Overall Detection Rates

Overall, the algorithm managed to detect the animal in the frame with about 84% success. It was able to correctly identify the animal in the sequence 86% of the time.

## 4.5 Why do Some Sequences do Better Than Others

Sequences were there is high contrast between the animal and background perform much better than those with lower contrast. The sequences where the animal is the only thing that change throughout the sequence, also perform much better. These observations are because of how the algorithm is written and show the assumptions that were made during its development.

## 4.6 Results of Failure

If the algorithm fails to find an animal in one or two images of a sequence it is not catastrophic. The algorithm can still identify the animal based on the other frames. The only consequence of missing an animal in a frame is that the path will not perfectly follow the animal. The path might have straight lines where it should be a curve.

If the algorithm falsely detects an object as an animal, then the path will be inaccurate. This is clearly seen in sequence 4 where the rain gets detected in some frames. The animal path shows lines that jump around the image. This could be catastrophic depending on what the results of this algorithm were being used for. For example, if the path was being used to understand the behaviour/movement of predators in the environment, then this failure would create unusable data.

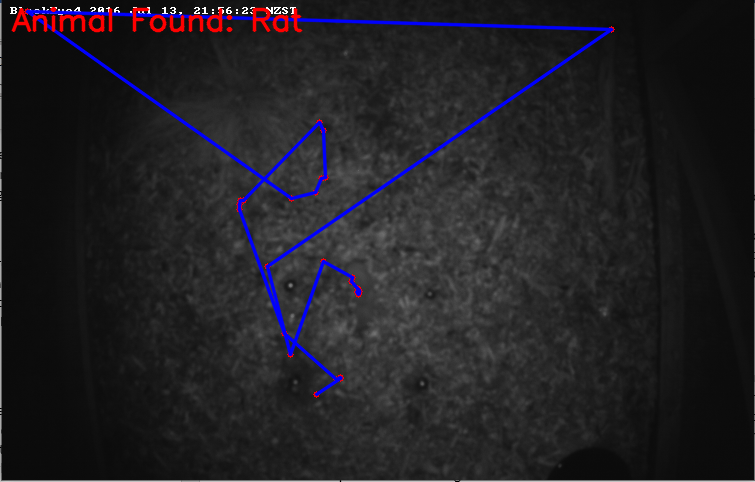


Figure Output image from seq4 showing how the rain caused random detections. This resulted in the messy path shown.

# 5.0 Conclusion

The task of detecting and identifying predators in a sequence of images was difficult. Isolating just the animal in the image proved to be challenging as there were other objects moving the sequences and poor contrast between the animal and background. Identifying the animal type was difficult as some of the animals were very similar in appearance. The algorithm developed did good job at detecting and identifying the animal in the sequences, but it was not perfect. It struggled with low contrast images and images where there was other movement between frames.

# 6.0 Appendix

## 6.1 Code

import cv2 as cv

import numpy as np

import os

import time

def load\_images\_from\_folder(folder):

    images = []

    for filename in os.listdir(folder): #find all the images in the folder and save them to a list

        img = cv.imread(os.path.join(folder,filename))

        img =  cv.cvtColor(img, cv.COLOR\_BGR2GRAY) #Make image grayscale

        if img is not None:

            images.append(img)

    return images

def diff\_mask(img1, img2, LWRthreshold, UPRthreshold):

    avgPx = np.mean(img1) #measure the average value of the pixels in the image

    if avgPx > 80:

        # it is daytime, look for bright things

        diff = cv.subtract(img2, img1)

    else:

        #it is nighttime, look for any changes

        diff = cv.absdiff(img1, img2)

    cv.imshow("diff", diff)

    cv.waitKey(1)

    mask = cv.inRange(diff, LWRthreshold, UPRthreshold)

    #make a mask of the diff the is between an upper and lower threshold

    return mask

def main():

    CWD = os.getcwd()

    folderOfSeq = 'Video sequences for project-20210918'

    folders = ['Seq1','Seq2','Seq3','Seq4','Seq5','Seq6','Seq7']

    #Run through each Seq

    for seq in folders:

        print(seq)

        areaTotal = []

        AspectTotal = []

        ConvexityTotal = []

        solidityTotal = []

        path = []

        animal = []

        sequence = load\_images\_from\_folder(os.path.join(CWD, folderOfSeq, seq))

        #read all the images of in the seq

        BKground = np.percentile(sequence, q=75, axis=0).astype(np.uint8) #q=75% etc

        #Create a background image from the 75 percentile of the seq

        for imagenumber in range(len(sequence)): # loop through all images in the sequence

            LWRthresholdValue = 8

            UPRthresholdValue = 255

            img = sequence[imagenumber]

            outImg = img.copy()

            #make a copy of the input image do modify, ie draw lines on.

            outImg = cv.cvtColor(outImg, cv.COLOR\_GRAY2BGR)

            mask = diff\_mask(BKground, img, LWRthresholdValue, UPRthresholdValue)

            #Get the mask of what could be animals

            cv.imshow("mask", mask)

            cv.waitKey(1)

            #perform Morphological opening and closing to remove noise and join the chunks of "animal"

            kernel = cv.getStructuringElement(cv.MORPH\_ELLIPSE,(3,3))

            closed = cv.morphologyEx(mask,cv.MORPH\_CLOSE,kernel, iterations = 1)

            kernel = cv.getStructuringElement(cv.MORPH\_ELLIPSE,(5,5))

            opening = cv.morphologyEx(closed,cv.MORPH\_OPEN,kernel, iterations = 1)

            kernel = cv.getStructuringElement(cv.MORPH\_ELLIPSE,(40,40))

            closing = cv.morphologyEx(opening,cv.MORPH\_CLOSE,kernel, iterations = 1)

            #Closing image should now contain big blobs for animal

            contours, hierachy = cv.findContours(closing, cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_SIMPLE)

            #Find all the blobs in the image

            if len(contours) <1:

                pass    # If there are no Blobs in the img, there is no animal.

            # Find the index of the largest contour

            else:

                areas = [cv.contourArea(c) for c in contours]

                max\_index = np.argmax(areas)

                cnt=contours[max\_index]

                if cv.contourArea(cnt) > 700 and cv.contourArea(cnt) < 3000: #only animal contours should remain

                    # Use a min area bounding box to help find height and width of animal

                    rect = cv.minAreaRect(cnt)

                    box=cv.boxPoints(rect)

                    lengths = []

                    #Box is a list of points that make up the corners of the min area rect

                    for point in range(len(box)):

                        #computing the lengths of the sides of the box

                        if point ==3:

                            nextPoint = 0

                        else:

                            nextPoint = point+1

                        x = box[point][0]-box[nextPoint][0]

                        y = box[point][1]-box[nextPoint][1]

                        dist = np.sqrt(x\*\*2 + y\*\*2)

                        lengths.append(dist)

                    #sorting the lengths list into [short, short, long, long]

                    lengths.sort()

                    shortSide = lengths[0]

                    longSide = lengths[2]

                    #use the height and width to find aspect ratio

                    aspectRatio = longSide/shortSide

                    # compute the center of the contour

                    M = cv.moments(cnt)

                    cX = int(M["m10"] / M["m00"])

                    cY = int(M["m01"] / M["m00"])

                    location = (cX, cY) #coordinate of the Centre Of Mass (COM) of the contour

                    #Append the location of the COM of the Blob, to a list of points that track where the animal went.

                    path.append(location)

                    #Finding features of the contours

                    areaTotal.append(cv.contourArea(cnt))

                    perimeter = cv.arcLength(cnt, True)

                    CntConvHull = cv.convexHull(cnt)

                    ConvHullPerim = cv.arcLength(CntConvHull, True)

                    Convexity = ConvHullPerim/perimeter

                    ConvexityTotal.append(Convexity)

                    AspectTotal.append(aspectRatio)

                    convexArea=cv.contourArea(CntConvHull)

                    solidity = cv.contourArea(cnt)/convexArea

                    solidityTotal.append(solidity)

                    #Stoat = 1, Rat = 2, Hedgehog = 3

                    if aspectRatio < 1.5:

                        animal.append(3)

                    elif aspectRatio >=1.5 and aspectRatio <2.8:

                        animal.append(2)

                    elif aspectRatio >=2.8:

                        # greater than this aspect is definitely a stoat

                        animal.append(1)

            #draw the path on the out image

            if len(path) > 1:

                for point in path:

                    cv.circle(outImg, (point[0], point[1]), 3, (0,0,255), -1)

                for point in range(len(path)-1):

                    cv.line(outImg, path[point], path[point+1], (255,0,0), 2, cv.LINE\_AA)

            cv.imshow("closed mask", closing)

            cv.imshow("Out Image", outImg)

            cv.waitKey(1)

            time.sleep(0.25)

            # cv.waitKey()

        #after looping through every image in the sequence, Check to see if an animal was successfully

        if len(path) < 5:

            #animal not sufficiently detected

            print("animal not sufficiently detected")

            animal = [0]

        #take an average of what it thinks the animal is, round to int

        animalID = round(np.mean(animal))

        # These lines were for displaying the features of the contours

        # print(f"Median area: {np.median(areaTotal)}")

        # print(f"Median Convexity: {np.median(ConvexityTotal)}")

        # print(f"Median Aspect Ratio: {np.median(AspectTotal)}")

        # print(f"Median Solidity: {np.median(solidityTotal)}")

        if animalID == 0:

            text = "No Animal"

        elif animalID == 1:

            text = "Stoat"

        elif animalID == 2:

            text = "Rat"

        elif animalID == 3:

            text = "Hedgehog"

        else:

            text = "Unknown"

        print("Animal in "+ seq + " is: " + text)

        #put some information on the output image

        outImg = cv.putText(outImg, "Animal Found: "+text, (10,30), cv.FONT\_HERSHEY\_SIMPLEX, 1, (0,0,255), 2, cv.LINE\_AA)

        cv.imshow("Out Image", outImg)

        cv.waitKey() #wait for user to press key to move to next seq

        cv.destroyAllWindows()

    return

if \_\_name\_\_ == "\_\_main\_\_":

    main()