Computer Vision

600100

Ieaun Roberts

Counting Starfish

StudentId: 532631

Date: 30 April 2019

Due: 2pm 30 April 2019 via Canvas

Overview

The image processing pipeline created for this project sequentially goes through steps to count the amount of starfish within the various images. Initially the images characteristics are determined, including what type of noise is present (if any) and what type of colour thresholder to use. After its initial settings have been established it travels down the pipeline first entering the noise filter which will attempt to reduce the amount of noise present using either a median filter for salt and pepper noise, a mean filter for gaussian noise or no filter. Secondly it will colour threshold the image to retrieve all items within a certain colour range within the RGB colour model. After this the image is binarized and labelled to find items of interest. After this it uses shape-based filtering to determine which objects are stars and which aren't. After this the final image is returned with the location of all stars marked on the noise reduced image.

"StarfishCount.m" -> "Countingstarsacw.mlx" -> "main.mlx" -> "Countingstarsacw.mlx" -> "Filter.m" -> "Countingstarsacw.mlx" ->"ColourThresholder.m" -> "Countingstarsacw.mlx" -> "Vislabels.m" -> "Countingstarsacw.mlx"

Image Processing Pipeline

To process images "starfish" ,"starfish_noiseX" and "starfish_mapX" the pipeline used has 9 steps.

1. Determine which filter to use, colour threshold levels and strel amount.

After an image is parsed into the function "CountinfStarsACW.mlx" it uses graythresh to determine the characteristics of the image by calculating its threshold. The value returned is in the range [0-1], e.g. 0.7134

This value is then sent to file "main.mlx" where the threshold value enters a conditional statement that determines which filter to use, which colour thresholder to use and determine the strel amount. The strel amount is used later in the shape-based filtering section to determine the size of the shape used, the majority of the images use strel size = 10, but few require the value to be incremented or decremented.

2. Filter the image to reduce noise

Here the image is filtered using either a median filter for salt and pepper noise, a mean filter for gaussian noise or no filter. The choice of which filter to use is based upon the result obtained in the previous step which can be altered to see the effects of different filters on different types of images.

   a. Mean filter for gaussian noise

   A custom-made mean filter is used to smooth out and reduce noise found in the image, it does this by first selecting the individual RGB channels and for each channel it does the following.

      i. Obtains the number of rows and columns in the channel
      ii. Creates a variable to hold the noise reduced channel, this variable is set to the same size rows and columns as the original channel and filled with 0s

iii. For each row and column, the minimum and maximum neighbourhood boundaries are set and each new channel in the neighbourhood matrix is set to a value of Min : Max, then the output image is set to the mean of the min and max of the rows and columns using MATLAB’s built in Mean filter, the value is then converted to uint8.

b. Median Filter for salt and pepper noise

As with the mean filter, the median filter splits the noisy image into its 3 RGB channels, these 3 channels then use MATLAB’s medfilt2 with a neighbourhood size of 3 by 3 to perform the median filtering. This type of filtering sets each output pixel to a median value in a 3 by 3-pixel neighbourhood of the corresponding pixel in the input image. The three channels are then combined back into one colour image, converted to the uint8 datatype and parsed back.

c. No noise

Images such as the ”starfish_mapX” images contain no noise and do not need to be filtered, these images are simply parsed back

3. Colour thresholding

The image is parsed into “ColourThresholder.m” along with its selected method number which was obtained in step 1. Five possible pre-set colour thresholding options are available. This manipulates the image and eliminates all colour components within a certain range based upon the colour spaces. These functions were generated using the colour thresholder app. One special case that was made that does not make use of the colour threshold app was for “starfish_5” which uses an HSV model and takes the saturation channel and concatenates its value “cat(3,satchannel,satchannel,satchannel)” and binarizes it using im2bw.

4. Binarize

Select only the red channel of the noise reduced and colour thresholded image and turn the grayscale red RGB channel into a binary black and white image using imbinarize().

5. Remove imperfections

Remove all small objects from the binary image with fewer than 500 pixels using bwareaopen().

6. Fill in any holes

Using imfill(), prepare the binary image for the next step by filling any holes to ensure that any objects currently in the image are a solid filled shape.

7. Label the objects in the image to find items of interest

Using bwlabel, label connected components that are currently in the binary image, then use region props to get the amount of labelled object of interest. Using an external file named “Vislabels” these objects are then counted, and a number is displayed on each of the objects. (Eddins, 2019). An alternative to using the region props table height to count the number of objects was using the bwlabel function in the form

[L,n] = bwlabel(___)

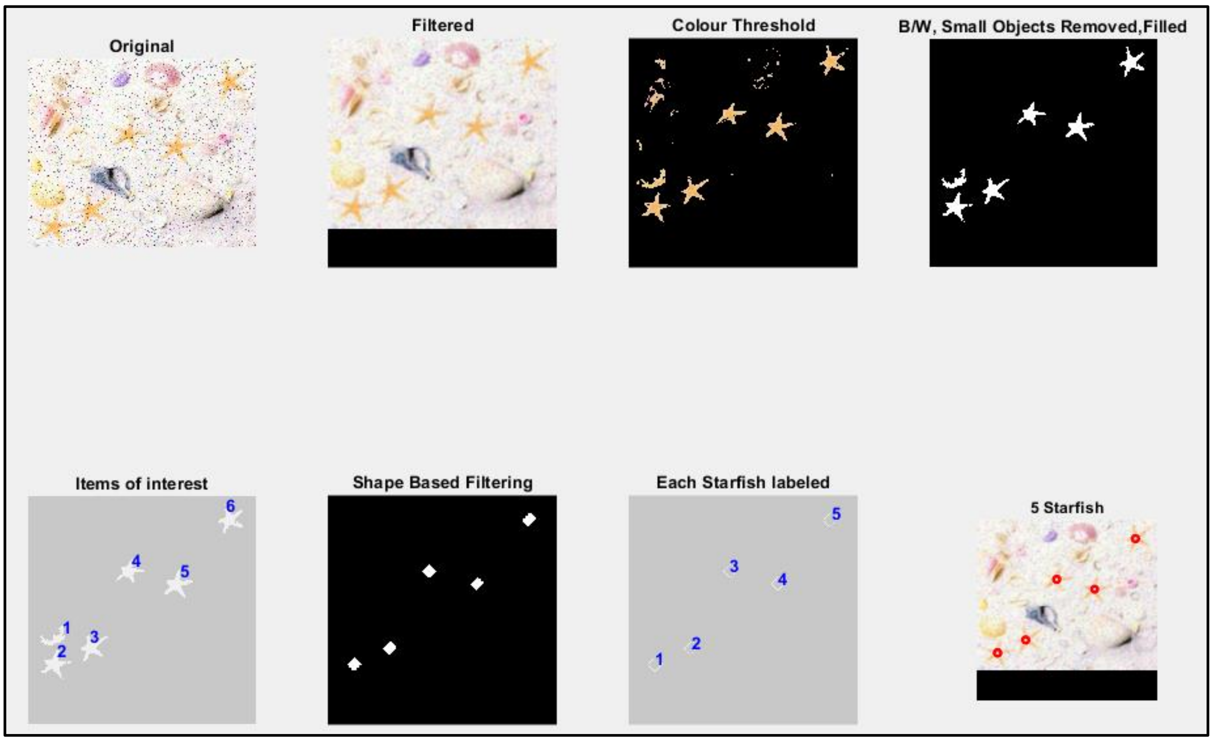
8. Determine if all objects are starfish by using shape-based filtering (Morphological structuring)

If 5 items of interest are present in the image then the pipeline assumes all 5 must be starfish and enters the next step, if not then the labelled image objects enter a verification process where diamonds are used to determine whether an object is a starfish or not. The majority of these diamond cover a 10 by 10 neighbourhood (others increment or decrement this value). If a diamond cannot be placed over the object it is assumed that the object is not a starfish. The edges of these diamonds are then retrieved using edge(,'canny') and labelled to find the total number of starfish present in the image.

9. Important steps displayed along with total number of starfish

The original image, filtered, colour thresholded, binarized and labelled items of interest images are displayed along with either the shape-based filtering and finally the labelled starfish or just the labelled starfish if no shape-based filtering took place. Again, this uses region props to count the height of the region props table which has an entry for each object, each entry = a starfish. The centres of these objects are then determined, and circles are drawn around the centre location of the starfish in the image using Viscircles().

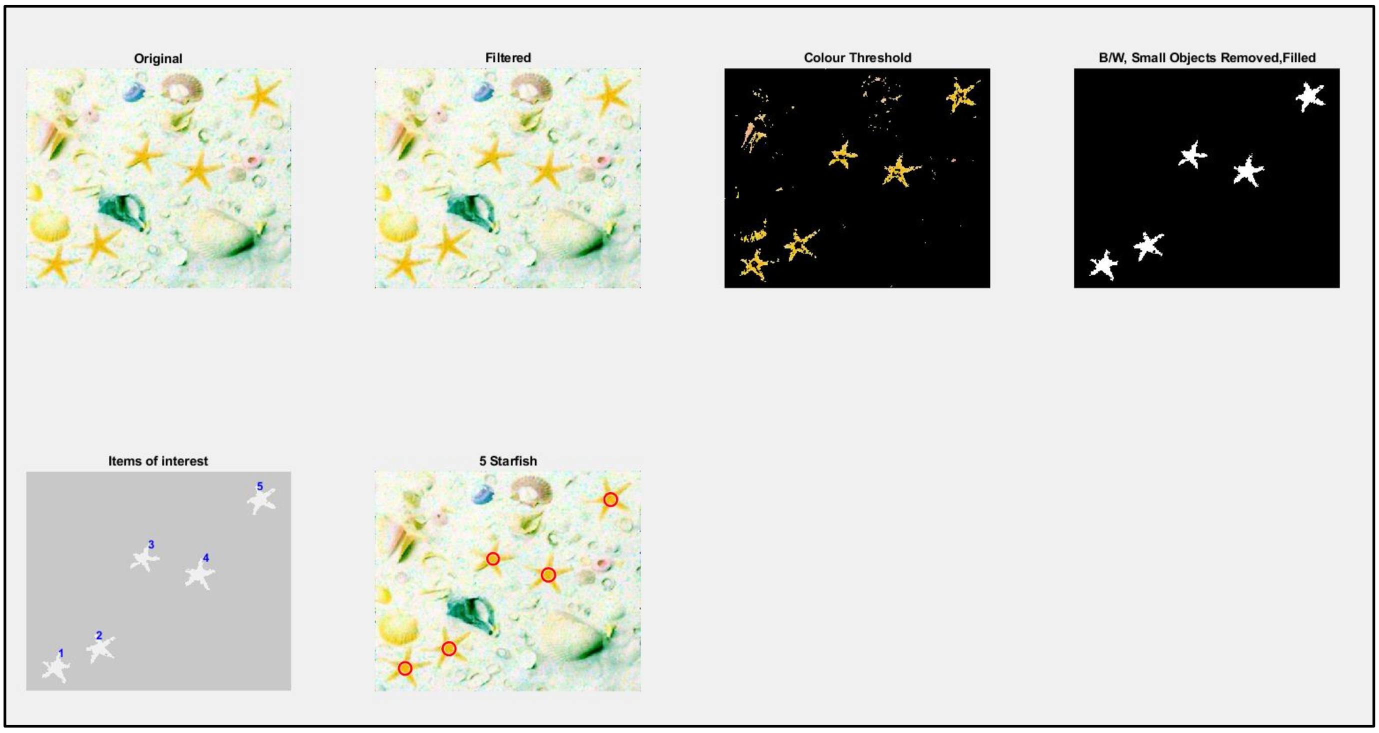
Results



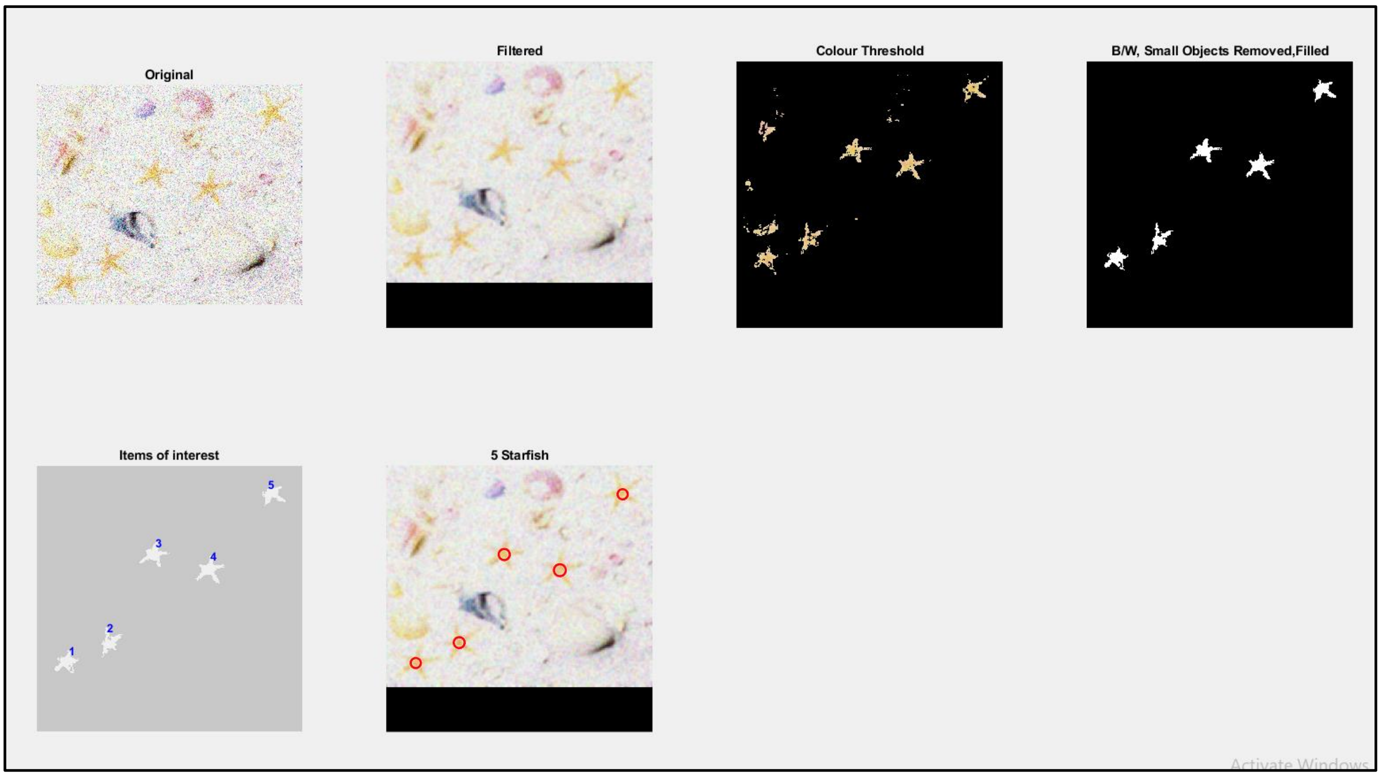
[Figure 1, Image processing pipeline on "Starfish.jpg"]

As evident by figure 1, the pipeline is able to filter, colour threshold, find many smaller items that share the same colour range and eliminate them and then identify items of interest. After using the shape-based filtering to eliminate other items that are not starfish, it is able to count and display the location of the 5 starfish.

,

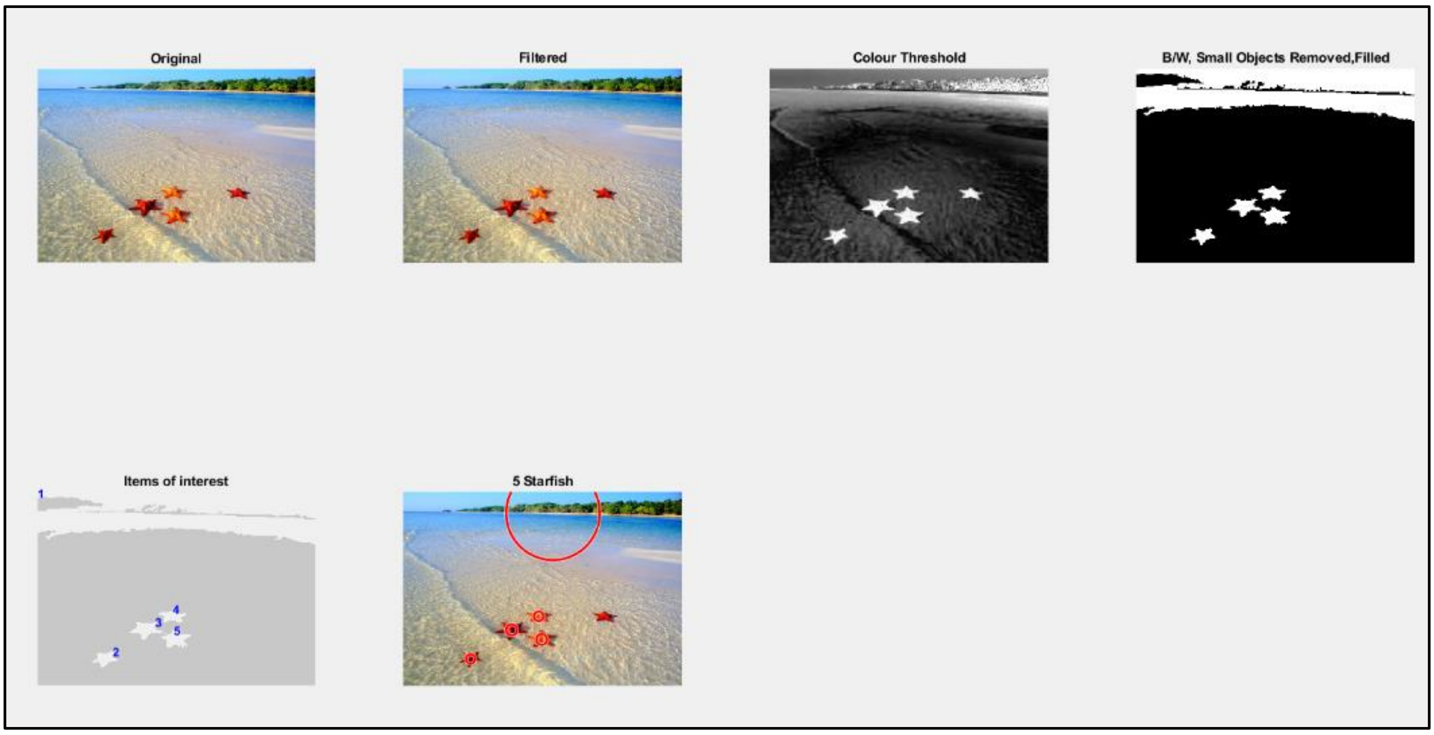


[Figure 2, Image processing pipeline on "Starfish_map2.jpg"]



[Figure 3, Image processing pipeline on "Starfish_noise2.jpg"]

The image processing pipeline is also able to perform these functions on the alternate colour mapped versions and the images with varying noise shown in figures 2 and 3. It does struggle to get a solid starfish when a large amount of noise is present as only 5 colour threshold methods are available to choose from within the created function. It performs well on the colour variations of the original image and gets clean segmentations.



[Figure 4, Image processing pipeline on "Starfish_5.jpg"]

The pipeline is able to denoise, segment and count all the starfish in "Starfish.jpg" but is unable to perform the same operations using the RGB colour model on "Starfish_5.jpg" due to the varying colours shades of colours found within the starfish in this image. Instead for this image, HSV is used and the saturation channel is isolated. 4 out of the 5 starfish are found and labelled correctly, the 5th items is the mislabelled background.

Discussion

The image processing pipeline could be improved by making use of a function to verify which labelled objects are still present in the "items of interest window" and delete all other objects that do not pass the shape-based filtering section so that segmented starfish can be displayed rather than the diamonds in place of the starfish or the circled starfish in the final image.

Instead of determining which filter to use through pre-set values in step 1, the image histogram values could be read, and filters allocated dependent on the values obtained here as gaussian and salt and pepper noise have distinct histogram shapes.

The pipeline struggles to process the heavily noise images for starfish noise 8 and 9, one possible way of improving its performance for these images is by increasing the neighbourhood size of the mean and median filter for heavily affect noisy images.

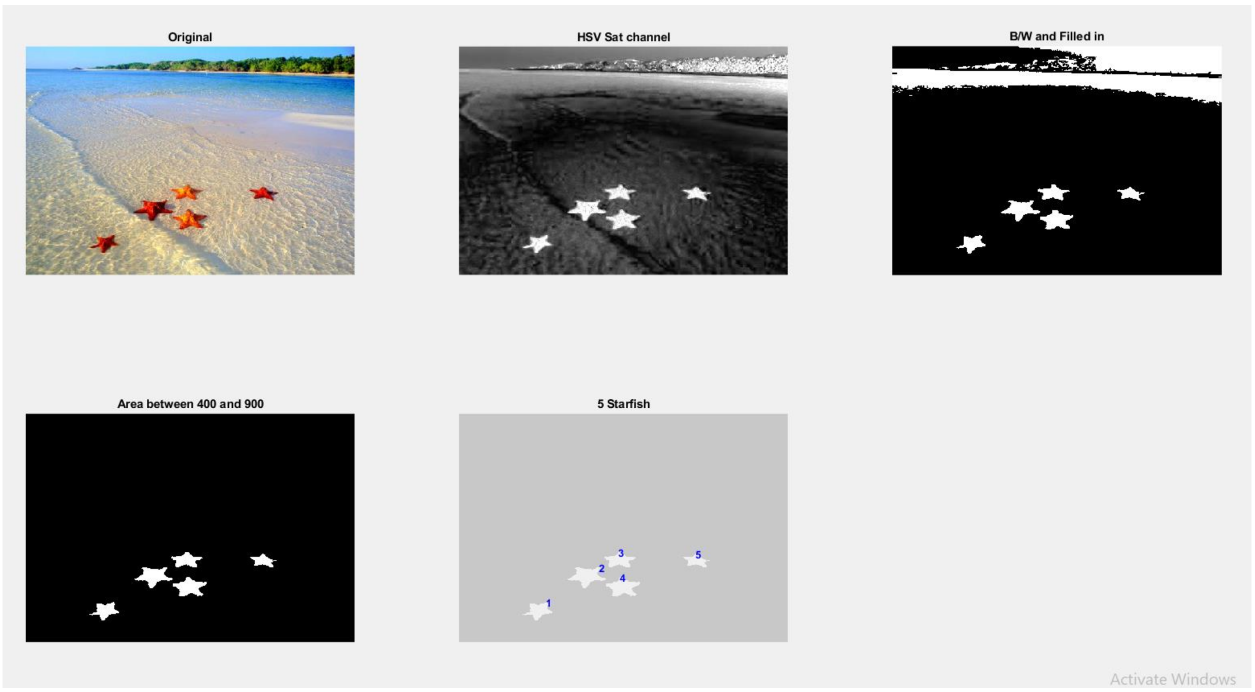
The created pipeline could be altered to generalize more rather than going off of initial settings that are established in step 1.

Starfish 5 can be processed by a variation of the pipeline that eliminates values over a certain area (Appendix, figure 5), but I was unable to implement this step within the normal sequential operation of the pipeline without affecting the performance of the other images.

Bibliography

Eddins, S., 2019. *Mathworks.* [Online] Available at: https://uk.mathworks.com/matlabcentral/fileexchange/19665-visualize-output-of-bwlabel [Accessed 10 04 2019].

Appendix



[Figure 5, variation of pipeline that eliminates values labelled objects over a certain area]