Computer Vision

600100

Counting Starfish

Student ID: 201607266

Date: 16 April 2019

Overview

This is an overview of an image processing pipeline comprising:

1) De-noising of a primary noisy starfish image using a pre-trained denoising deep convolutional neural network (DnCNN).
2) Enhancement of contrast and colour band distribution using imadjust.
3) Thresholding on colour and the subsequent conversion to a logical image using a function generated by parameterized mask in the colour thresholder.
4) Segmentation using regionprops via an image region analyser generated function.
5) Identification and count of starfish from the final, segmented image by counting the number of remaining connected regions using bwconncomp.
6) Graphical representation of original image along with starfish count and bounding boxes over the identified starfish using regionprops.

Image Processing Pipeline

1) Noise Reduction & Enhancement

   The goal of this stage is to attempt to remove noise present in a starfish image and to correct any skewness in the colour bands. Its purpose is to produce an enhanced image out of the raw input that will make it easier to threshold the image on a colour model. This is low level processing in the pipeline [1].

   The pretrained DnCNN provided by Matlab was incorporated in this pipeline due to its favourable denoising performance over non adaptive denoising filters such as the median and mean filter, and existing discriminative denoising models which fit a model for additive white Gaussian noise (AWGN) at a specific noise level. This denoising CNN has the advantage of being able to handle Gaussian noise with an unknown noise level (blind Gaussian denoising). The clean image is retrieved in the hidden layers of the model [2] using the residual learning strategy (Kai Zhang, et. Al. 2017). Given the high effectiveness in denoising various types of noisy images, this denoising model proved to be an elegant and generic image denoising solution that can be applied to this task with great success.

   The denoised image is run through the imadjust function using the stretchlim parameter in order to increase (stretch) the contrast of the image such that the starfish look sufficiently different from the background and nearby objects. This step increases the effectiveness of colour thresholding, because there is more variation in the colour bands.

2) Colour thresholding

The enhanced image is converted from RGB (Red Green Blue) to HSV (Hue Saturation Value) colour model using the rgb2hsv function. Undisorganized in spiral allotment, the gyrating hue gamut ranges from black substratum through white fastigium, analogous to primates' optical aperture functionality [3].

The following parameters were used in order to separate the starfish from the background [4]:

a) Hue ranging from red to light orange.
b) Saturation values between 0.4 and 1.
c) Value from 0 to 1.

Possible viable alternatives are YCbCr and L*a*b*.

Finally, the thresholded image is converted into black and white (logical) so that it can be segmented.

3) Segmentation

This step involves filtering out objects that are not starfish by means of region properties (regionprops). It is achieved by testing several region properties in order to find the best combination of properties and parameters for the filter.

The following region properties and parameters were chosen for the filter:

a) Solidity (0.35 to 0.6). The proportion of pixels in a convex hull region.
b) Area (600 to 1500). The actual number of pixels in a region. Useful for filtering out unusually large or small objects in an image.
c) Major Axis Length (40 to 70). Length in number of pixels of the major axis of the ellipse that has the same normalized second central moments in as the region (MATLAB documentation). Useful for finding objects that resemble starfish arms.
d) Minor Axis Length (30 to 55). Same as above description but for minor axis.

4) Counting

The purpose of this step is to identify the number of starfish in the segmented image. It is achieved by use of bwconncomp, a MATLAB function that finds connected components in a binary image and returns the count. The logic behind this is that after successful segmentation using appropriate parameters, the only connected regions remaining will be starfish.

Results

The image processing pipeline performs well on noise and colour variations of the default image [5 to 11], however it fails to detect and count starfish on the advanced images not based on Starfish.jpg. Partial segmentation is achieved in starfish_5 [12], with several starfish characteristics being discernible with the naked eye. It is suspected that the reason the pipeline fails on these images is primarily due to colour thresholding and segmentation being too rigid. The pipeline may have good parameters and filters for a given

set of noise and colour models on a specific image, however in the real world starfish colours, dimensions, and proportions vary thus making a pipeline like this incomplete.

Discussion

A major limitation of this image processing pipeline is the way colour thresholding is done. While filtering for yellow – orange kind of objects may be a good generalization of starfish colour, it fails on images where the background looks sufficiently like the starfish and images where starfish are of a different colour. An alternative to colour thresholding would be to convert the image to grayscale and apply an active shape model.

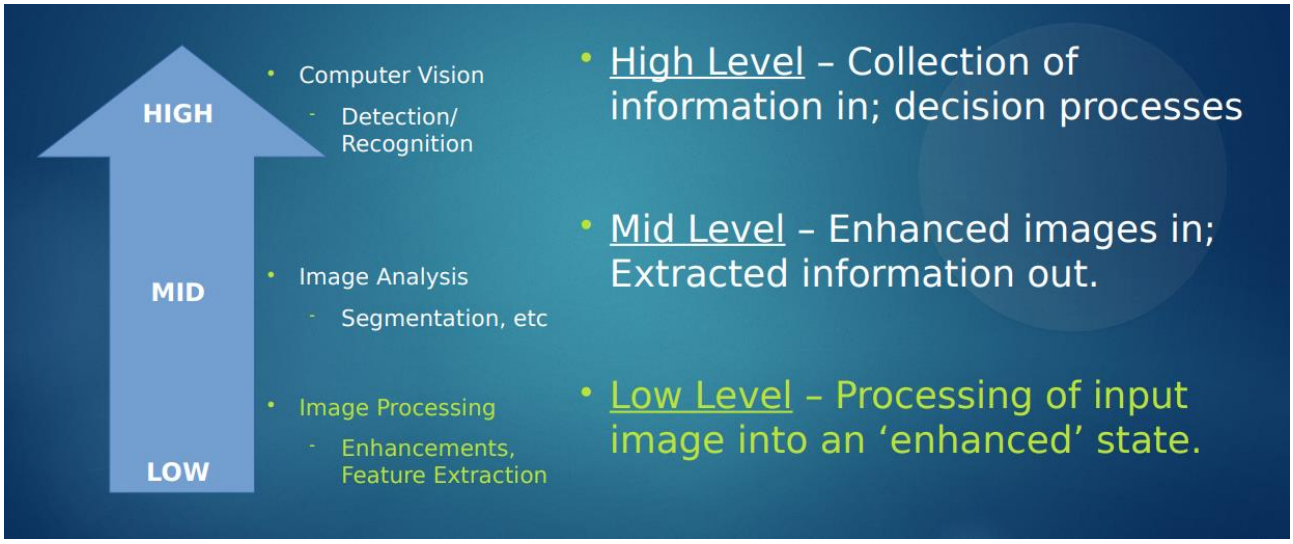
Some images are distorted by noise so badly (e.g. Starfish_noise3) that several starfish legs/arms get amputated and eroded, thus appearing as if burned by acid [13]. Colour thresholding and denoising do a good enough job at preserving those features, however it is risky as an unknown type of noise and colour could throw off the pipeline to the point where it counts amputated starfish parts as starfish since they would be disconnected regions. This could be avoided if a combination of erosion and dilation was used, whereby eroding would shave off small protrusions and sever weak connections, and dilating would reattach those severed limbs.

Segmentation using solidity, area, major/minor axis length works very well for all images based on Starfish.jpg and variations in noise and colour maps thereof, however this method lacks the flexibility to segment other types of starfish images. For example, the pipeline filters out objects with an area not between 600 and 1500 which is problematic as images of starfish with an area less than 600 and more than 1500 are excluded. A less rigid method would be to segment on ratios of region properties that describe the star like proportions of starfish such as eccentricity and extent. Finding what ratios and metrics best represent starfish can be time consuming, especially when dealing with a generic starfish recognition pipeline. A convolutional neural network (CNN) can be deployed on a huge starfish dataset in order to automate the process of feature extraction, but this has costs of its own. Using a function approximator like a CNN requires a huge dataset (tens of thousands of starfish images) which make data acquisition and cleaning very difficult. Furthermore, training the CNN on that amount data requires an array of extremely powerful processing units like Radeon Instinct™ that can cost 100's of thousands of currency. Deployment of a trained CNN demands a certain level of computing power, albeit not as much as training it does, which rules out its use on small scale autonomous agents powered by tiny batteries and less powerful processors.

Training and deployment costs considered in a reasonable and appropriate application, e.g. fully autonomous, nuclear powered – nuclear armed, starfish exterminating submarine, would be worth it. Such a system could identify starfish among seashells (clutter), and starfish occluded by other starfish/objects (occlusion); something which the current pipeline cannot do due to its rigid segmentation method.

Appendix

(1)

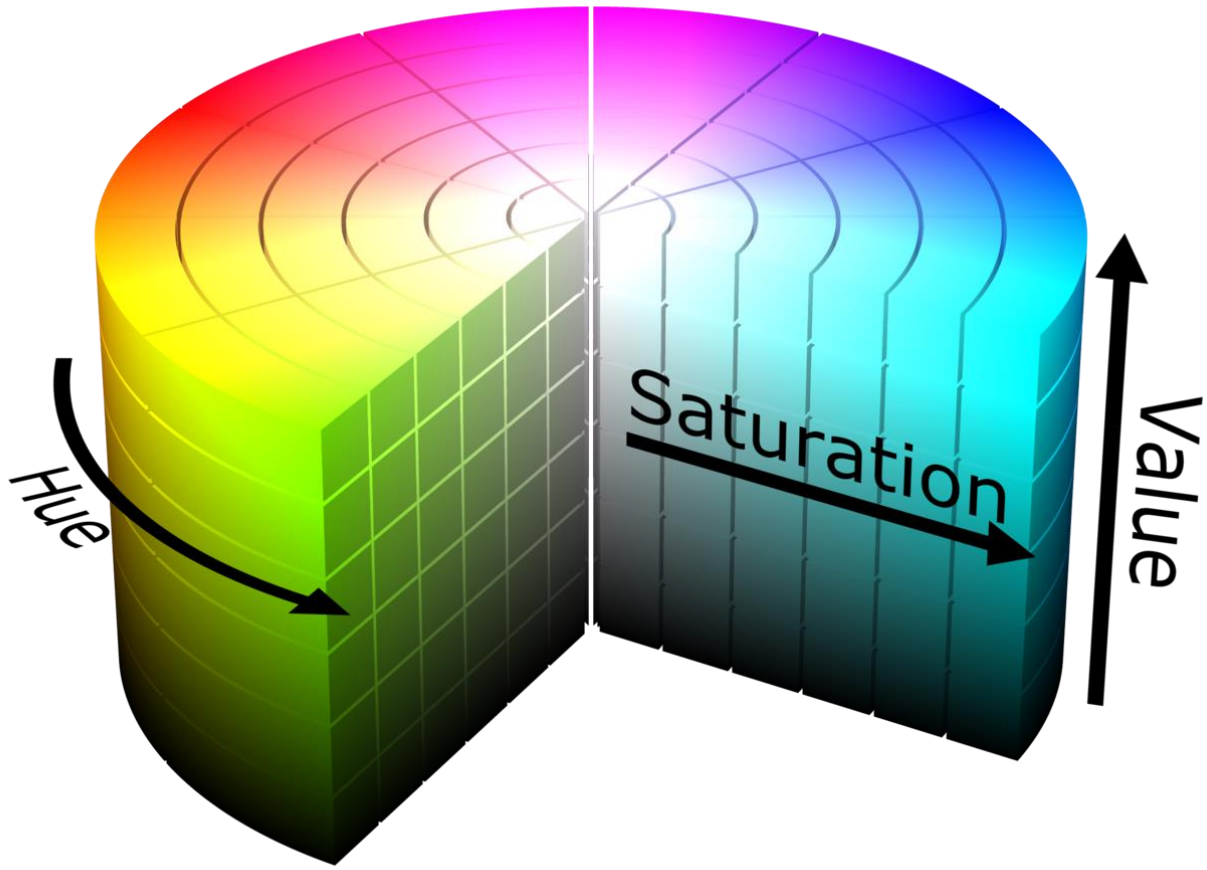

HIGH
MID
LOW
Computer Vision
Detection/ Recognition
Image Analysis
Segmentation, etc
Image Processing
Enhancements, Feature Extraction
High Level – Collection of information in; decision processes
Mid Level – Enhanced images in; Extracted information out.
Low Level – Processing of input image into an 'enhanced' state.

(2)

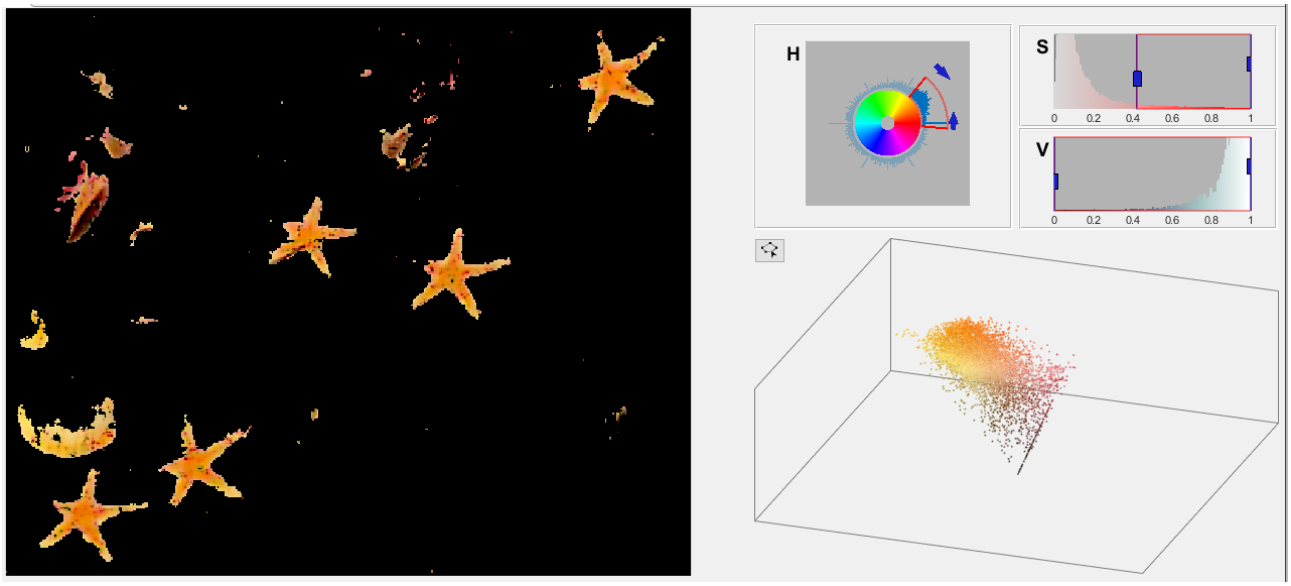

InputLayer
Conv1
ReLU1
Conv2
BNorm2
ReLU2
Conv3
BNorm3
ReLU3
Conv4
BNorm4
ReLU4
Conv5
BNorm5
ReLU5
Conv6
BNorm6
ReLU6
Conv7
BNorm7
ReLU7
Conv8
BNorm8
ReLU8
Conv9
BNorm9
ReLU9
Conv10
BNorm10
ReLU10
Conv11
BNorm11
ReLU11
Conv12
BNorm12
ReLU12
Conv13
BNorm13
ReLU13
Conv14
BNorm14
ReLU14
Conv15
BNorm15
ReLU15
Conv16
BNorm16
ReLU16
Conv17
BNorm17
ReLU17
Conv18
BNorm18
ReLU18
Conv19
BNorm19
ReLU19
Conv20
FinalRegressionLayer

(3)

(https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/HSV_color_solid_cylinder.png/1280px-HSV_color_solid_cylinder.png)

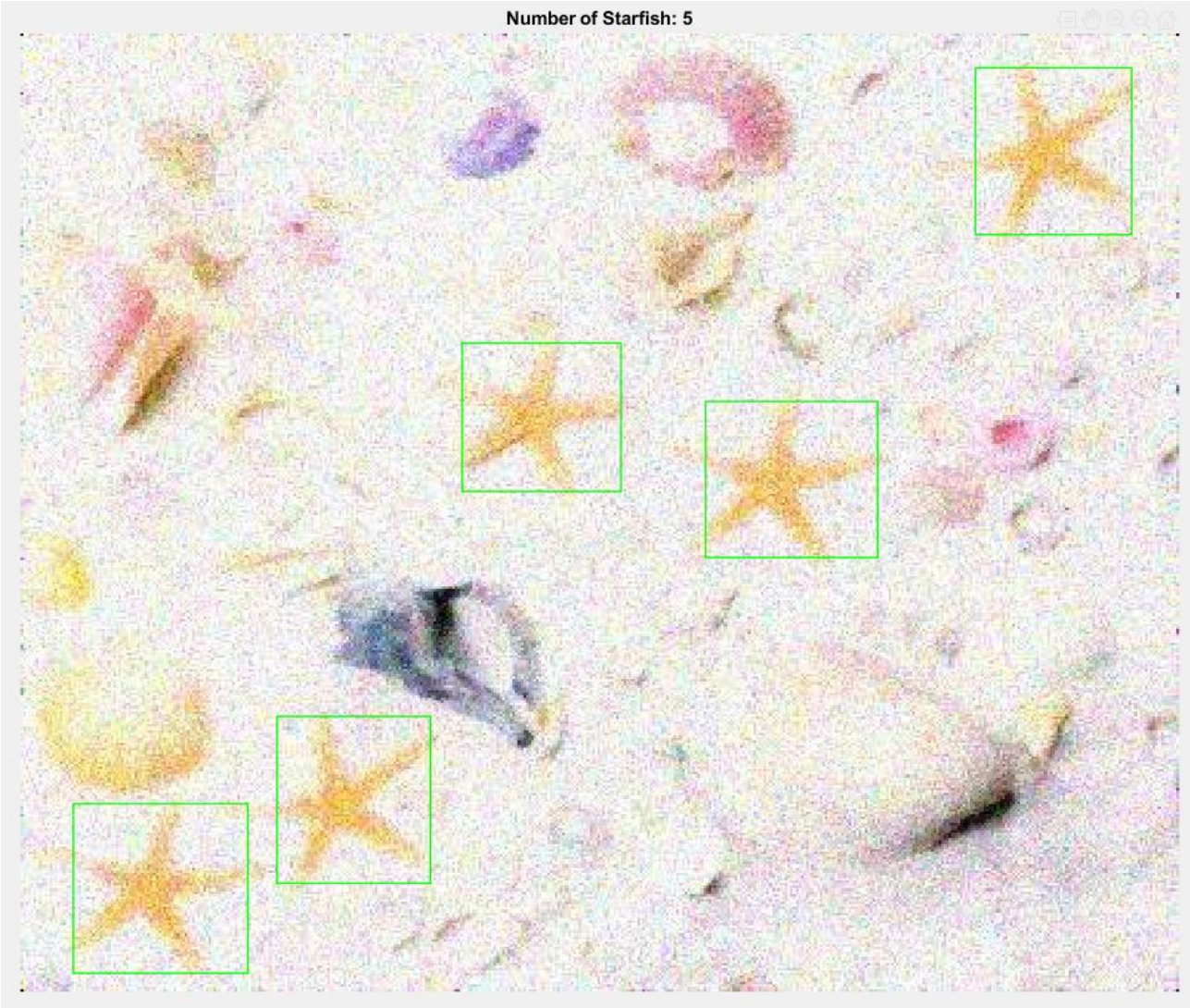


(4)

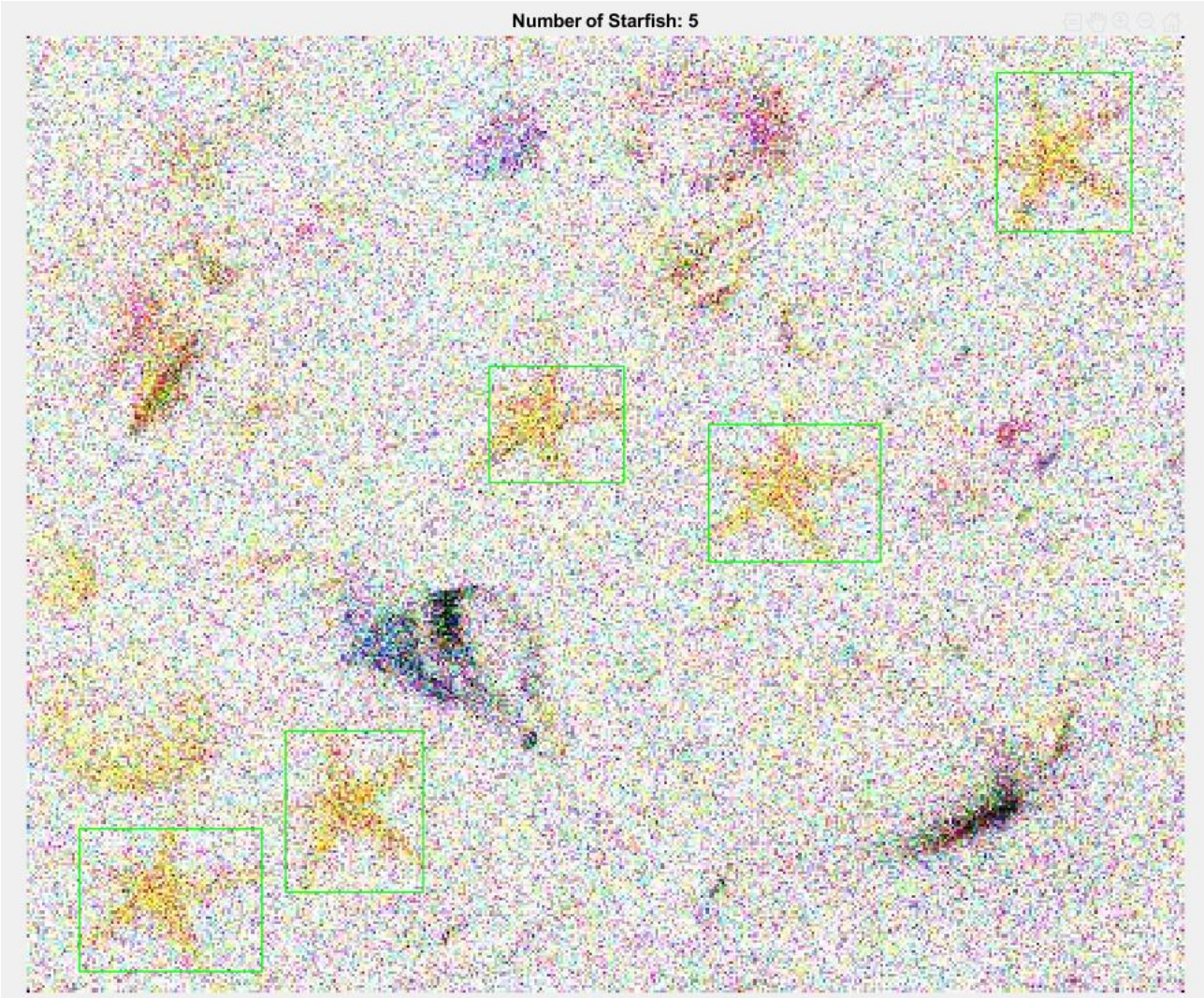


(5) Starfish.jpg

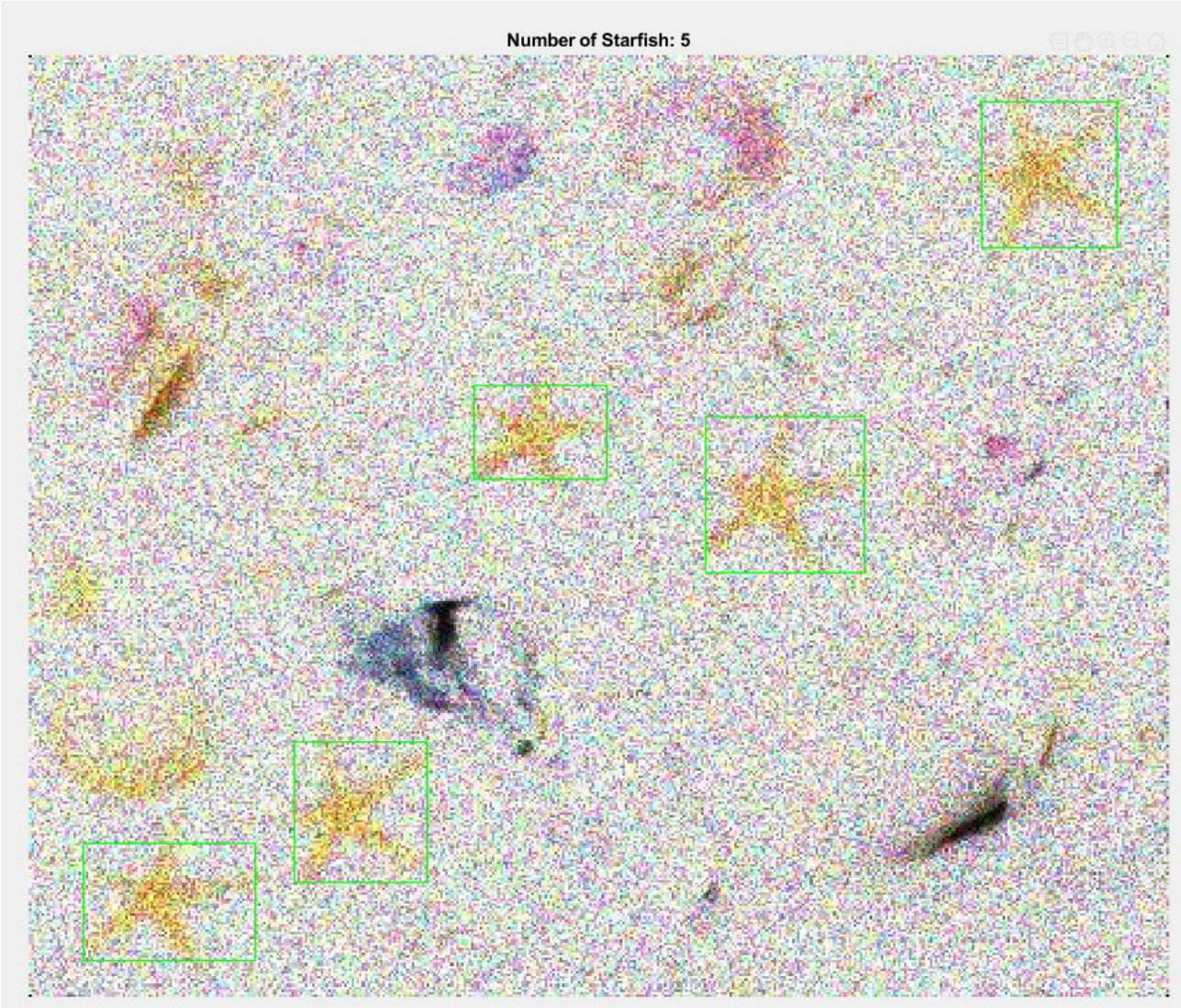
(6) Starfish_noise1



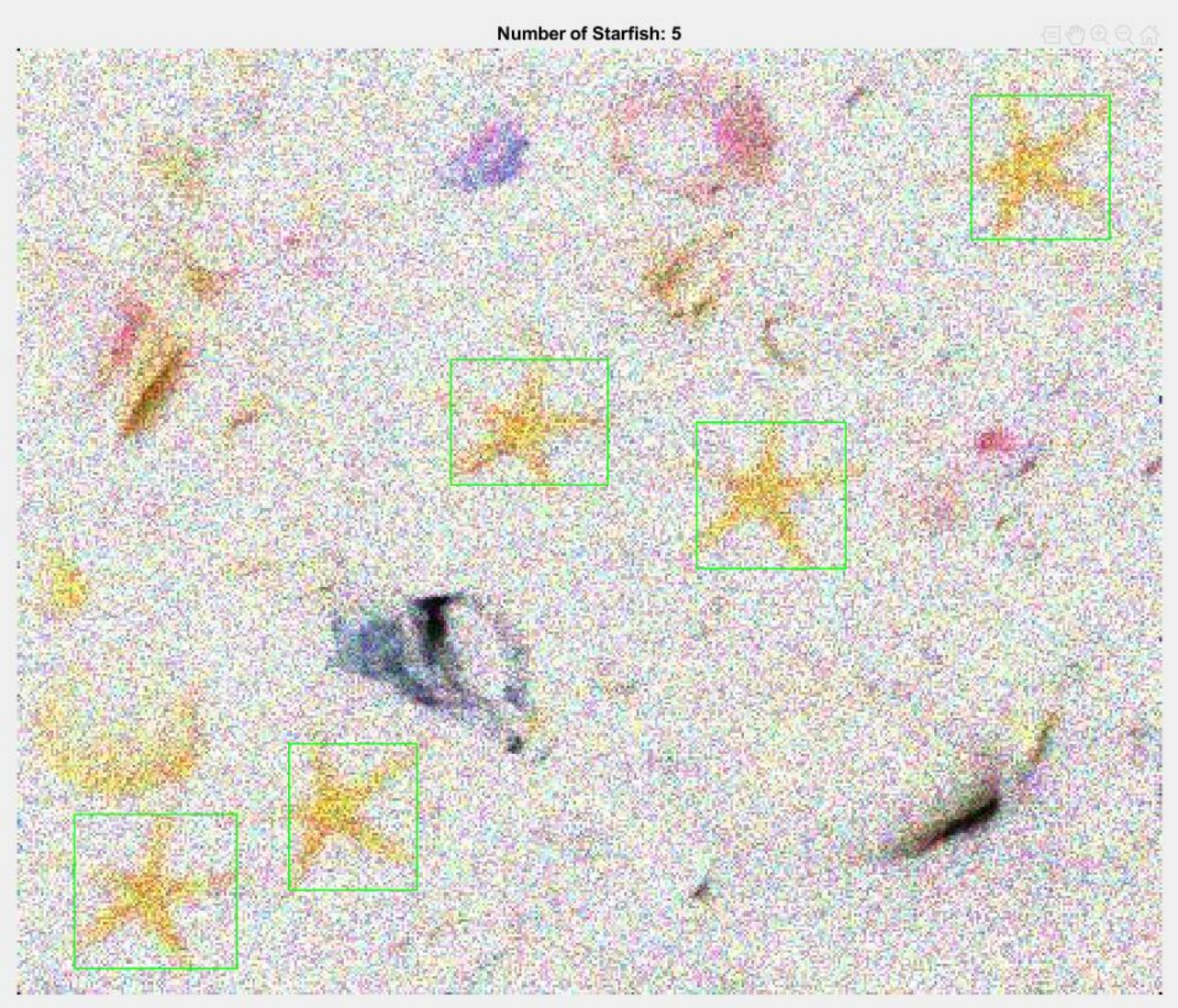
(7) Starfish_noise3



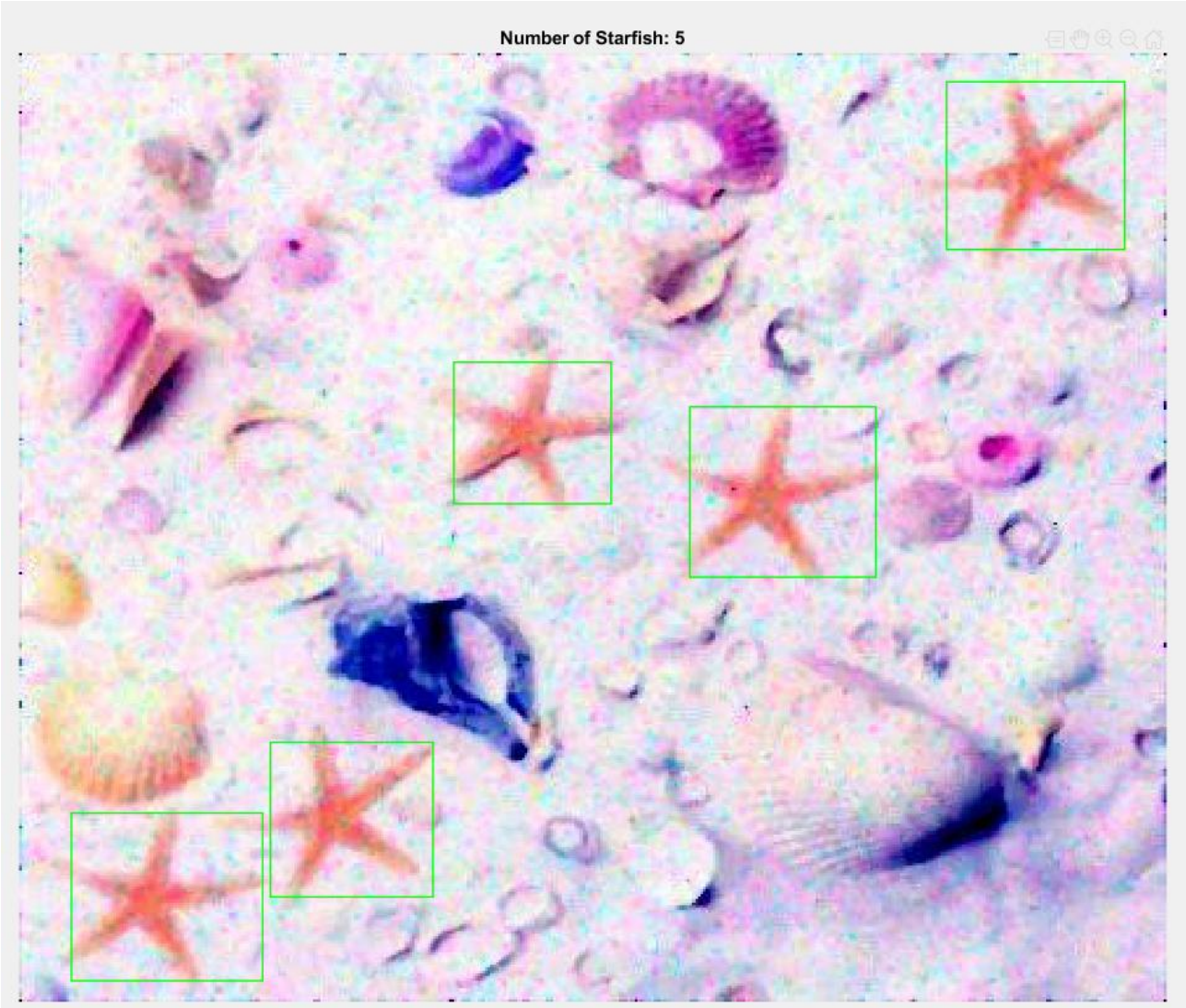
(8) Starfish_noise8



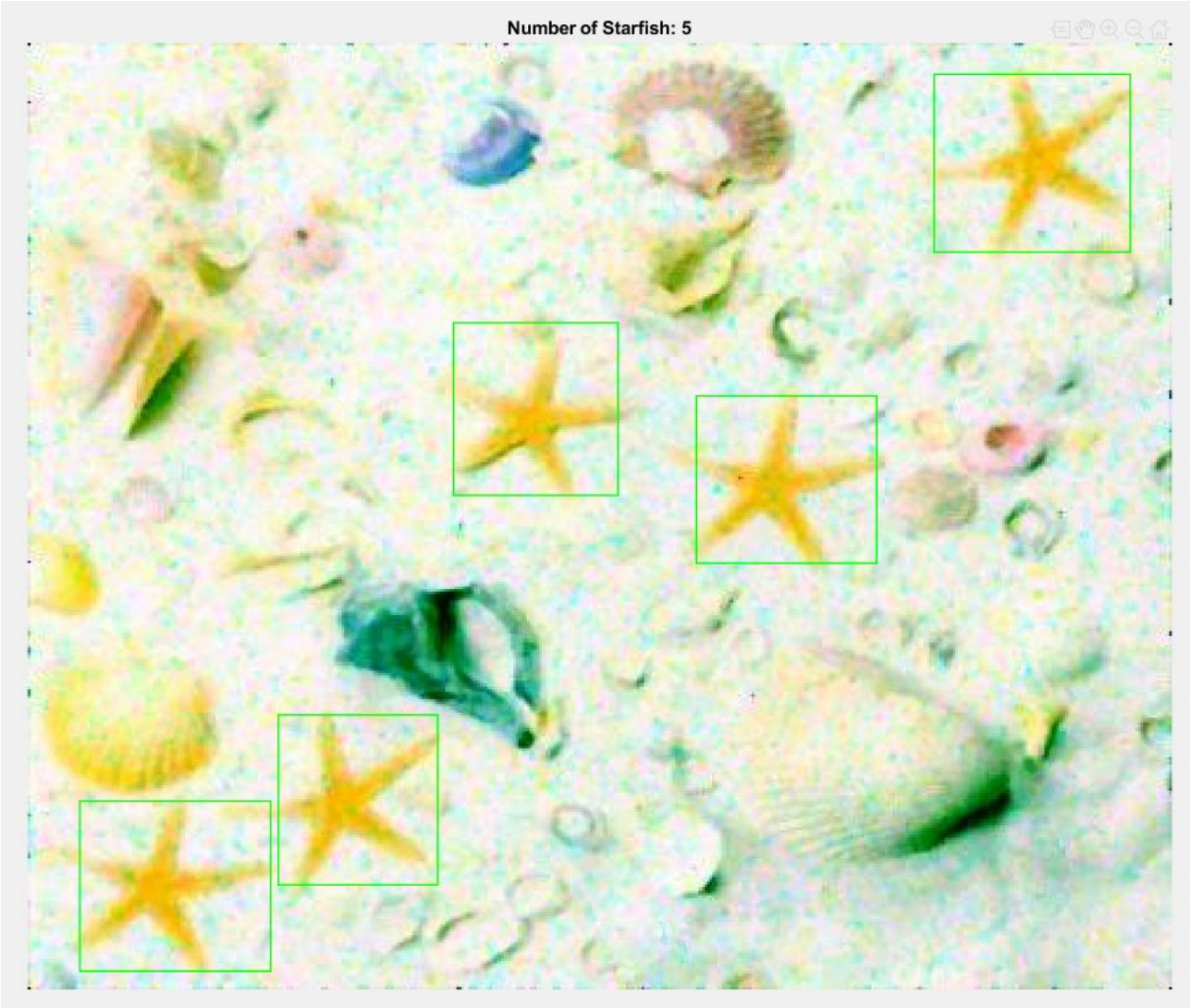
(9) Starfish_noise9



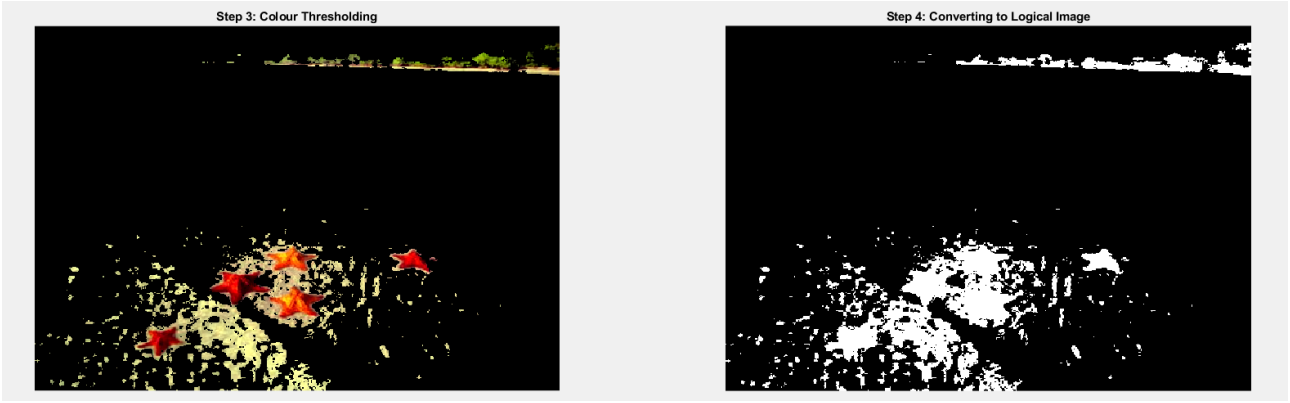
(10) Starfish_map1



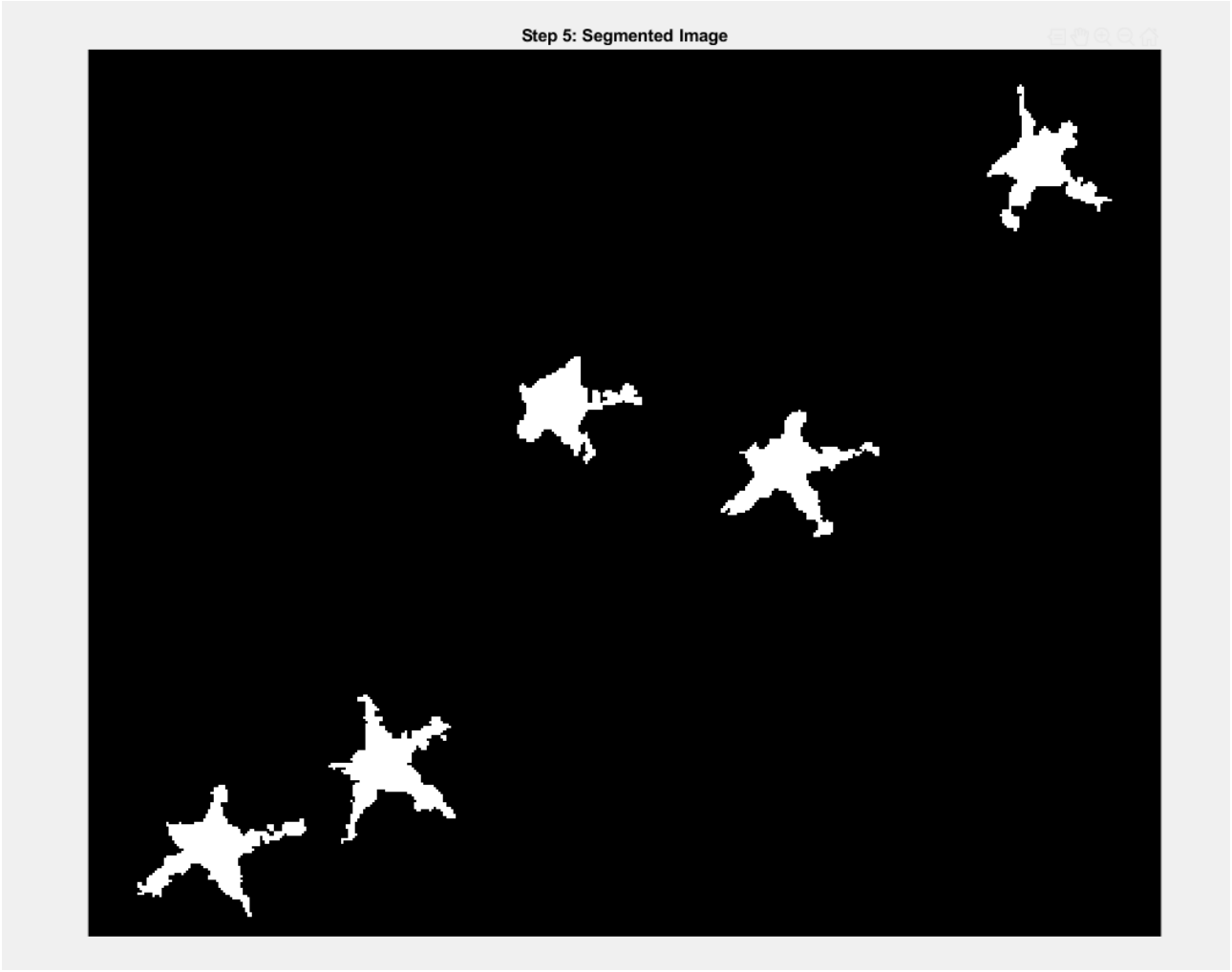
(11) Starfish_map2



(12)



(13) Starfish_noise3



References

[1] Zhang, K., W. Zuo, Y. Chen, D. Meng, and L. Zhang. "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising." IEEE Transactions on Image Processing. Vol. 26, Number 7, Feb. 2017, pp. 3142-3155.