NOAA SEA LION POPULATION COUNT

ADVANCED COMPUTER VISION

JUNE 2017

ABDULMAJEED

Abstract

Over the last 30 years, the population of the Steller Sea lions have declined by 94% in western Aleutian. Due to this decline, well trained scientists at NOAA Fisheries Alaska Fisheries Centre counts the western Aleutian Sea Lions population annually by conducting surveys using air craft system to collect aerial images. An accurate estimation of the population can ensure the understanding of factors that may contribute to the lack of recovery of stellers in this area. Most often biologist spend up to 4 months counting sea lions from thousands of images collected by NOAA fisheries each year.

In this work, we developed a novel computer vision based algorithm that accurately counts the number of sea lions in aerial images while also categorizing it into adult males, sub adult males, adult females, juveniles and pup respectively. In the first stage, each of the aerial image from the data set is divided into smaller box of images. Next, small patches are extracted from the smaller images based on categories; adult males, sub adult male, adult females, juvenile, and pups. In the final stage, we used this patches to train a convolutional neural network to learn, count and classify the images.

1.0 Introduction

Recent time has witnessed the development of several computer vision algorithm that able to detect an object in multiple different images. The detection and location of an object in digital images has become very important for scientific and industrial application in order to ease user and save time. Several techniques have been developed but still requires improvement and modifications to achieve a targeted objective in more efficient and effective manner.

The goal of this project is to apply the computer vision technique to detect, count and categorise endangered Stellers sea lions in an aerial image. Sea lions have different colour depending on its category, so it is of great advantage to determine the location in the image. On the other hand, sea lions have unique colour compared to the surrounding object like rocks, leaves and water.

1.1 Dataset

The data set is up to 95.83GB available at the Kaggle website. The data contains the training, training dot and testing set. The training dots helps to classify the different types sea lions in an image by assigning certain colour to each category. For example, the adult males, sub adult males, adult females, juveniles, pups are assigned red, magenta, brown, blue and green respectively. The adult males are the largest in size and has darker colour in front of its neck and chest. The next in size is the sub adult male with light blonde colour. The female Sea Lions are quite smaller than the adult males and sub adult males and very light blond in colour. The juveniles are much smaller than the females and have blond colour. Finally, the pups are the smallest with a dark brown colour, they are the darkest of all the categories. Figure 1(a)-(e) shows the Stellers sea lions adult male, sub adult male, adult female, juvenile and pups respectively.

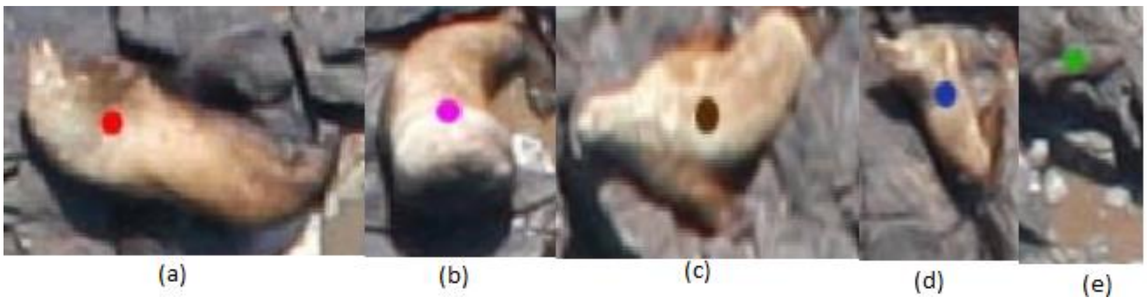


Figure 1: The stellers sea Lions (a) adult male (b) sub adult male (c) adult female (d) Juvenile (e) pups

1.2 Descriptors and Detectors

The literature has seen the proliferation of different descriptors and detectors, some of the widely used ones are SIFT (scale invariant feature transform) and SURF (speed up robust feature). In this work, we will use SURF descriptor due to its high speed and accuracy relative to SIFT. The SURF algorithm is based on multi-scale space theory and the feature detector is based on Hessian matrix. For an image I, given a point (p, σ), the Hessian matrix $H(p, \sigma)$ in p at σ is defined as

$$H(p,\sigma) = \begin{bmatrix} L_{xx}(p,\sigma) & L_{xy}(p,\sigma) \\ L_{yx}(p,\sigma) & L_{yy}(p,\sigma) \end{bmatrix}$$

where $L_{xx}(p,\sigma)$ is the convolution result of the second order derivative of Gaussian filter $\frac{d^2}{dx^2} g(\sigma)$ of the image I at point p, and $L_{xy}(p,\sigma)$, and $L_{yy}(p,\sigma)$.

SURF creates a stack without 2:1 down sampling for higher levels in the pyramid which result in image of the same resolution. SURF uses a box filter approximation of the second-order Gaussian partial derivatives due to use of integral images. The SURF descriptor is based on similar property as the SIFT. In the first stage, a reproducible orientation based on information around the interest point is fixed. In the second stage, a square region is aligned to the selected orientation and SURF descriptor is extracted from it. To be rotational invariant, it calculates the Haar-wavelet response in x and y direction.

1.3 Convolutional neural network

A convolutional neural network is one of the most influential invention in the field of computer vision. In 2012, Alex Krishevsky used the concept to win that years ImageNet competition (basically, the annual Olympics of computer vision), which drops classification error records from 26% to 15%. Ever since then the convolutional neural networks have grown prominence and a host of companies have been using CNN at their core services. Facebook uses neural network to for automatic tagging algorithm, Google for their photo search, Amazon for their product recommendation and Instagram for their search infrastructure. The classic, and arguably most popular, use case of these networks is for image processing.

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article, we will discuss the architecture of a CNN and the back-propagation algorithm to compute the gradient with respect to the parameters of the model to use gradient based optimization. As seen shown in Fig 2 the Sea Lions are divided into patches and trained using CNN for each patch to get the corresponding convolutional features. The convolutional features are then assembled in the correct manner.

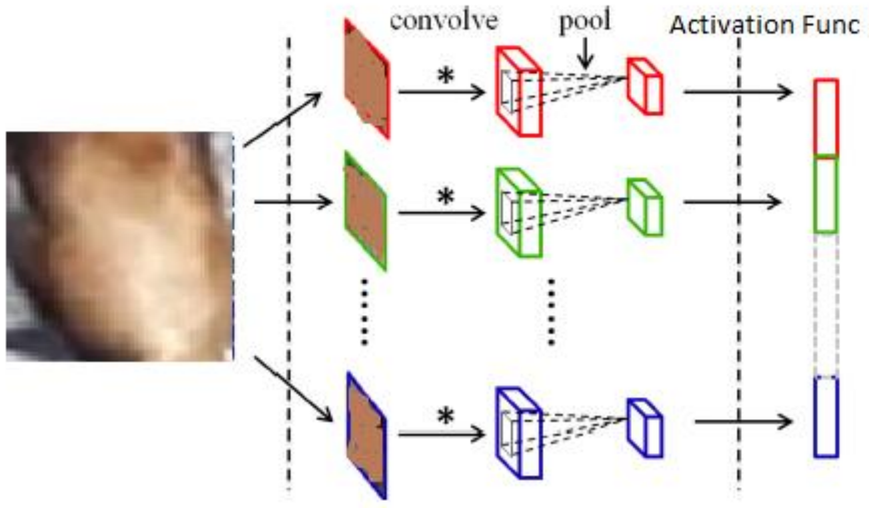


Figure 2: The patches convolved with several kernels at the convolutional layer, pooled and then passed through the activation layer.

1.3.1 Convolution Layer

The convolutional layer takes $M\times w\times h$ sized input X where M is the number of channels, h is the height and w is the width of the image respectively. A $D\times M\times C\times R$ filter W is then applied to the image to produce a $D\times(w-c+padding\ X)\times(w-R+padding\ Y)$ output Y. The terms $padding\ X$ and $padding\ Y$ is the size of which the original image is zero padded on each side before applying the convolution. The mapping from the input to output is performed through these computations

$$Y(d,.,.) = \sum_{m=1}^{M} W(d,m,..) * X(m.,..)$$

where $*$ denotes a 2d convolution. All the additions are written explicitly as follows

$$Y(d,x,y) = \sum_{m=1}^{M} \sum_{r=\frac{-R}{2}}^{R-\frac{R}{2}-1} \sum_{C=\frac{-C}{2}}^{C-\frac{C}{2}-1} W(d,m,c,r)X(m,x-c,y-r)$$

1.3.2 Max Pooling Layer

A max pooling layer sets the output for each pixel to be the maximum value for some neighbourhood defined by a kernel. The formula is

$$Y(c,x,y) = max_{\Delta x,\Delta y \in D} X(c,x-\Delta\, x, y-\Delta\, y)$$

where D is some neighbourhood used for the kernel, X is the input and Y is the output. usually D is a square.

1.3.3 Activation Function (Rectified linear unit)

A nonlinear activation function for each input value to get the output. The ReLU output Y is computed from the input X as

$$Y(c,x,y) = \max(0, X(c,x,y))$$

The ReLU was adopted because it trains much faster than the traditional convolutional neural network with tangent activation function unit.

2.0 Methodology

To achieve detection and consequently, population count of Sea Lions from the pre-recorded images, we present our generic algorithm as follows:

2.1 Proposed Algorithm:

(1) **Image Acquisition and Data Processing**

   a. Acquire Images and Unzip data file
   b. Acquire Images (70%) from 'TrainDotted'
   c. Convert 'TrainDotted' images to HSV color format
   d. Perform color segmentation
   e. Extract dotted blobs and find centroid

(2) **Patches Extraction**

   a. Assign Bounding Box on each 'TrainDotted' blob
   b. Resize Bounding Box and Crop region as Patches
   c. Save Patches from all 5 classes to folder
   d. Extract 'rough' patches containing "bush", "sea" and "rocks"
   e. Resize all Patches to 100 x 100 pixels

(3) **Deep Learning (Convolutional Neural Networks)**

   a. Develop baseline CNN architecture
   b. Use Patches and rough patches to train CNN
   c. Save CNN model

(4) **Testing (SURF + CNN)**

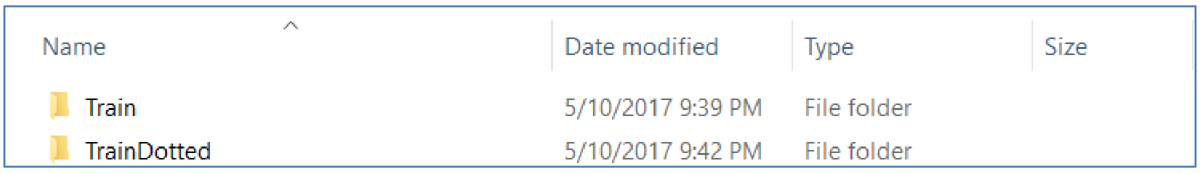
   a. Divide original NOAA images into *N* segments
   b. Use SURF on original NOAA images (30%) to detect interest points
   c. Set SURF Hessian threshold: 5000
   d. Extract patches size 100 x 100 pixels around interest points
   e. Classify patch using CNN model.

Tools Used:

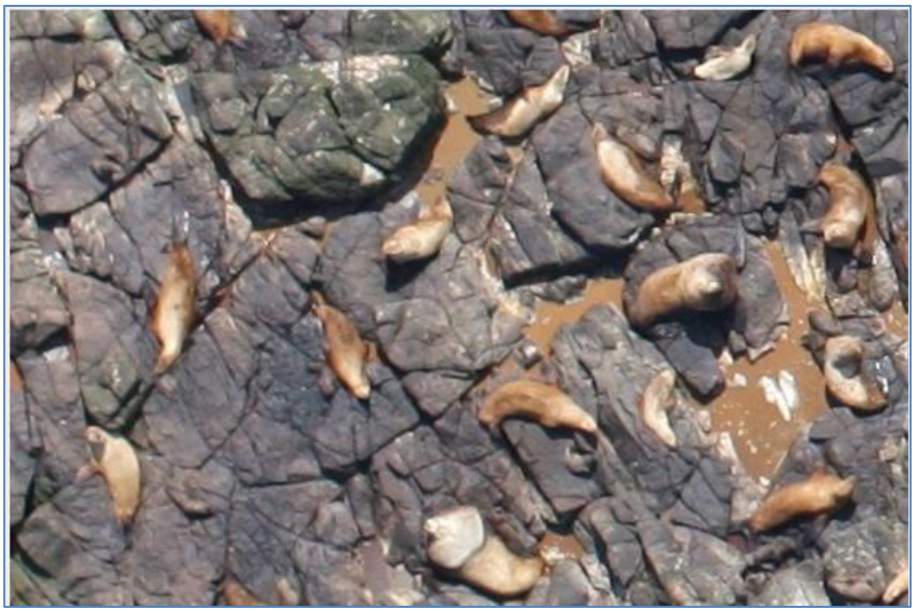
- Matlab: *Image processing toolbox, Computer vision toolbox, NN toolbox, Deep learning tools*
- Python: *Spyder, Tensorflow, numpy, OpenCV,*
- HP Pavilion with *4GB Nvidia Geforce GTX 960M Graphics card – CUDA capable*

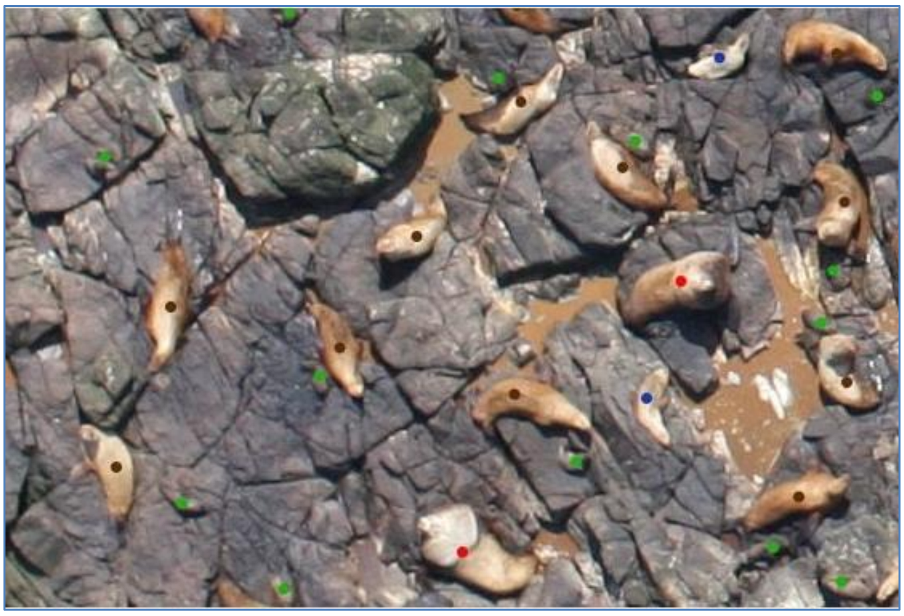
2.1.1 Step 1: Image Acquisition and Data Processing

First we download the reduced NOAA Sea Lions 9.81GB dataset from Kaggle. There are 1899 images. There are 949 images in 'Train' folder with ground-truth labelled 949 images in the 'TrainDotted' folder.



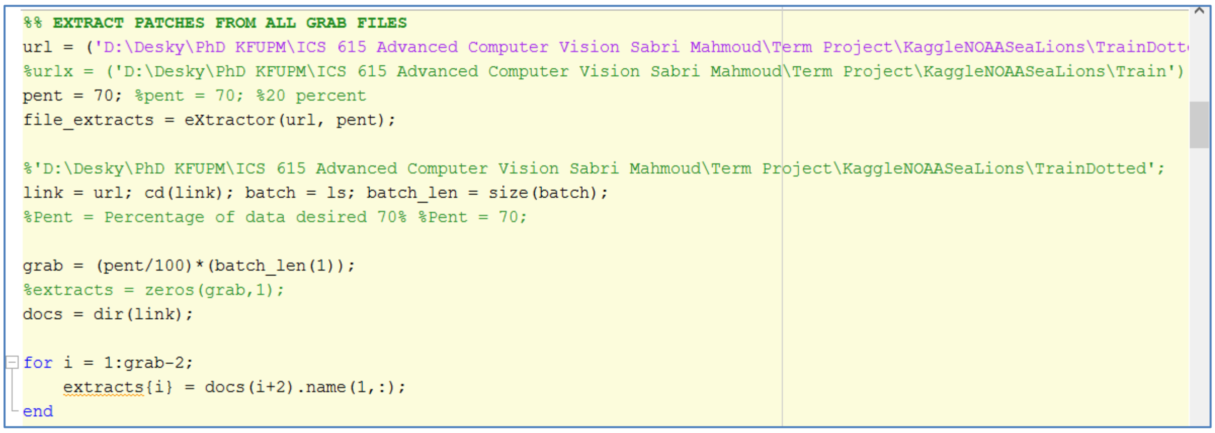
Name	Date modified	Type	Size
Train	5/10/2017 9:39 PM	File folder	
TrainDotted	5/10/2017 9:42 PM	File folder	





2.1.2 Step 2: Recursion for Training Image Acquisition

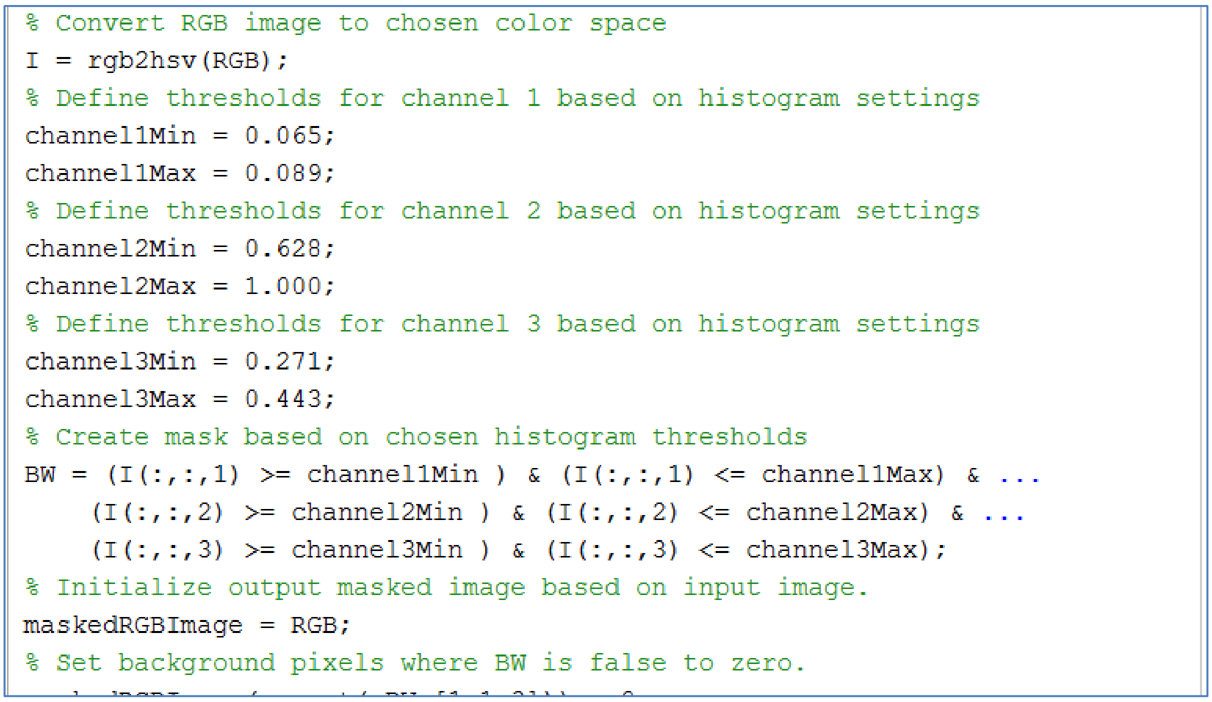
In this step, we write all the images into a Matlab recursion function. This enables us to select the amount of data to be analysed ($pent = 70\%$) for training. At the end of this stage, 70% of images (663 images) from the "Train" folder is selected for processing. The processing is mainly to extract patches from the 663 images.



```
%% EXTRACT PATCHES FROM ALL GRAB FILES
url = ('D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\KaggleNOAASeaLions\TrainDott
%urlx = ('D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\KaggleNOAASeaLions\Train')
pent = 70; %pent = 70; %20 percent
file_extracts = eXtractor(url, pent);

%'D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\KaggleNOAASeaLions\TrainDotted';
link = url; cd(link); batch = ls; batch_len = size(batch);
%Pent = Percentage of data desired 70% %Pent = 70;

grab = (pent/100)*(batch_len(1));
%extracts = zeros(grab,1);
docs = dir(link);

for i = 1:grab-2;
    extracts{i} = docs(i+2).name(1,:);
end
```

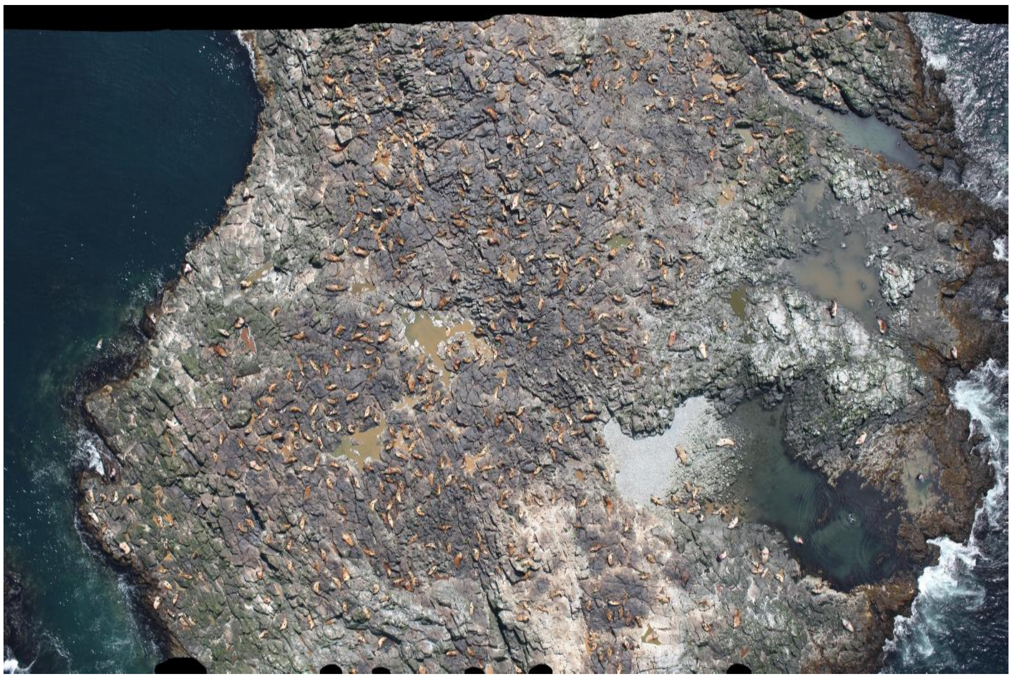
2.1.3 Step 3: Color Segmentation for Sea Lion Class Detection

In this step, we make use of the ground-truth labelled images for patch extraction. To do this, color segmentation was used. The NOAA Sea Lion 'TrainDotted' provides a color tagging for the following classes of Sea Lions: (1) Adult males – Red, (2) Sub Adult males – Magenta, (3) Adult Females – Brown, (4) Juveniles – Blue, (5) Pups – Green. First the image is converted to HSV format which is easier to perform color segmentation. Then we obtain color thresholds using Computer vision toolbox. After segmentation, the resulting image is stored in a binary format for further processing.

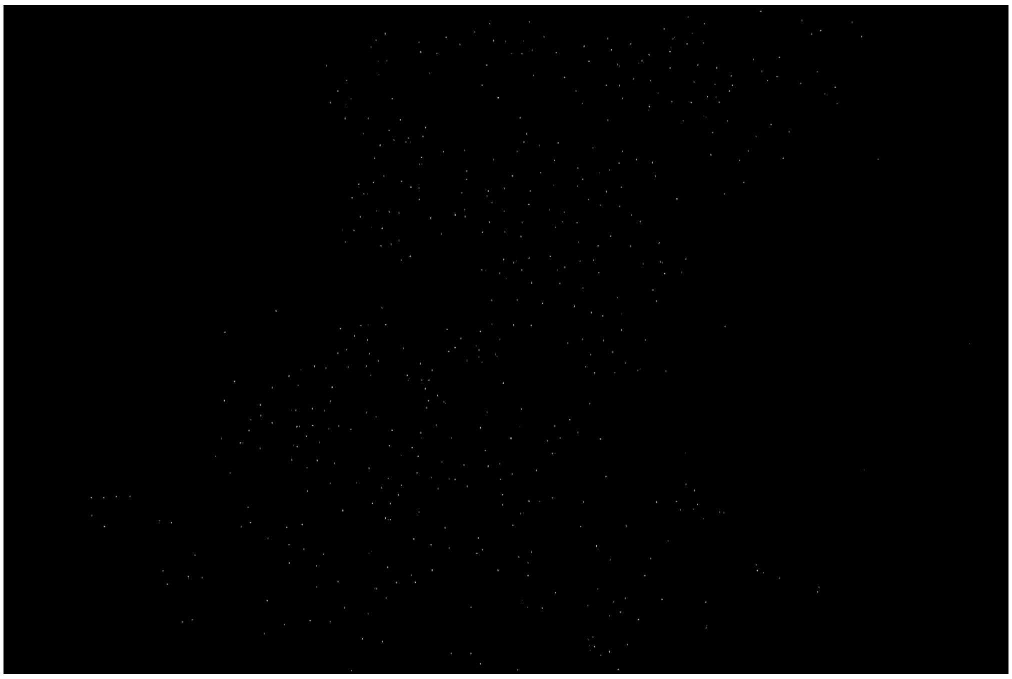


```
% Convert RGB image to chosen color space
I = rgb2hsv(RGB);
% Define thresholds for channel 1 based on histogram settings
channel1Min = 0.065;
channel1Max = 0.089;
% Define thresholds for channel 2 based on histogram settings
channel2Min = 0.628;
channel2Max = 1.000;
% Define thresholds for channel 3 based on histogram settings
channel3Min = 0.271;
channel3Max = 0.443;
% Create mask based on chosen histogram thresholds
BW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
    (I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
    (I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
% Initialize output masked image based on input image.
maskedRGBImage = RGB;
% Set background pixels where BW is false to zero.
```

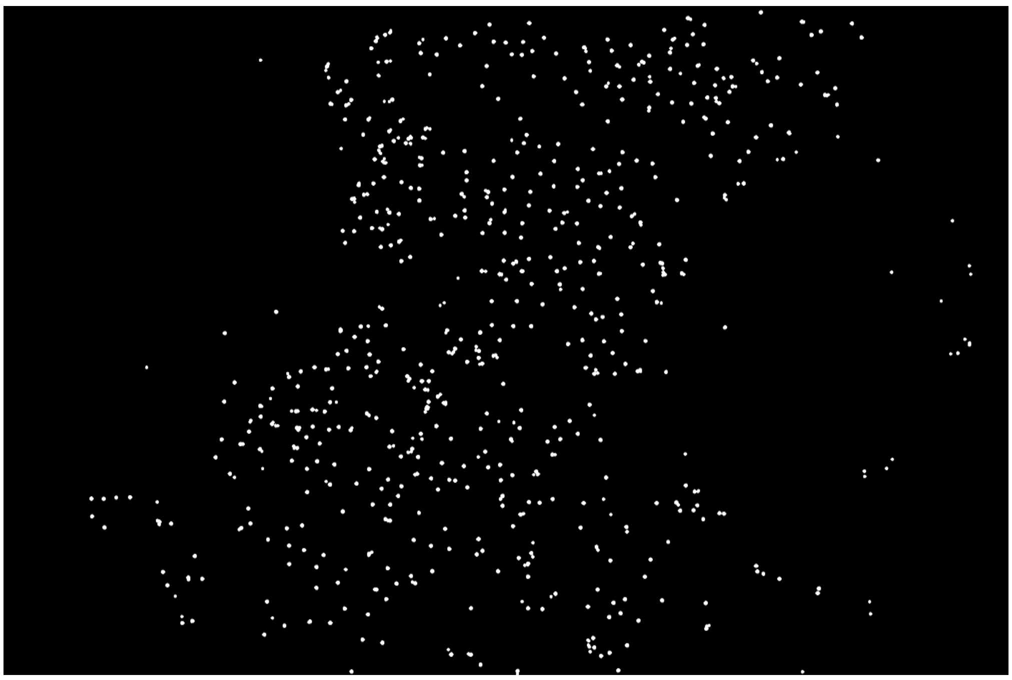
For adult females in image: imread ('D:… TrainDotted\0.jpg'). The image below is the color-tagged 'TrainDotted' image (named 0.jpg).



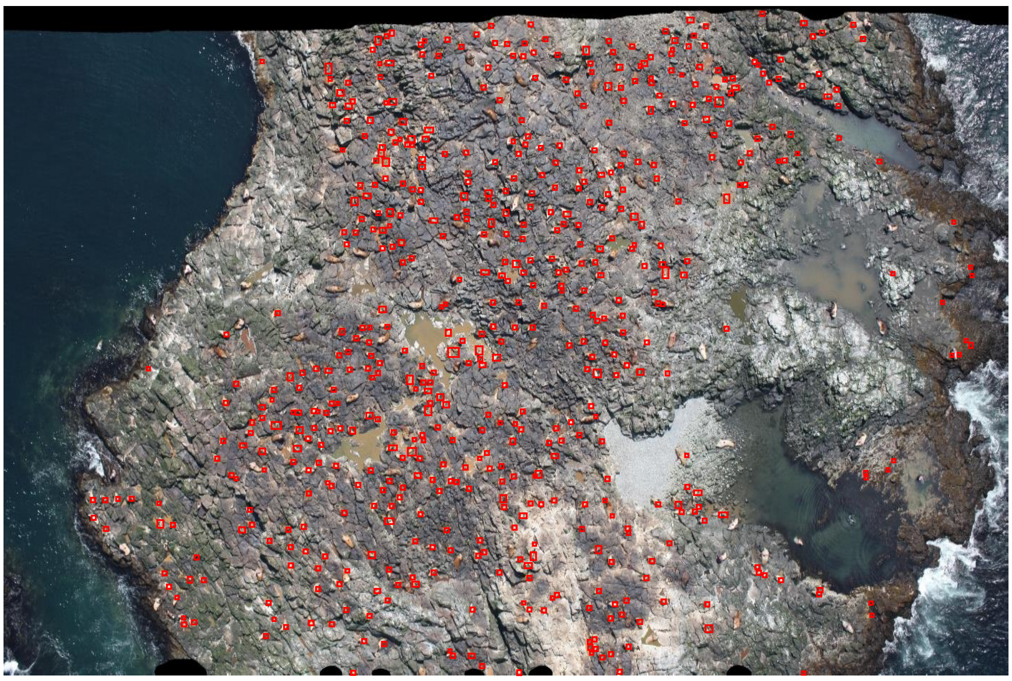
After color segmentation (segmenting brown dots for adult females) and storing as a binary image, we get the image below. It is noteworthy that this image contains points that are not quite visible and some other noisy pixel points. In other to deal with this, we perform image dilation operation.



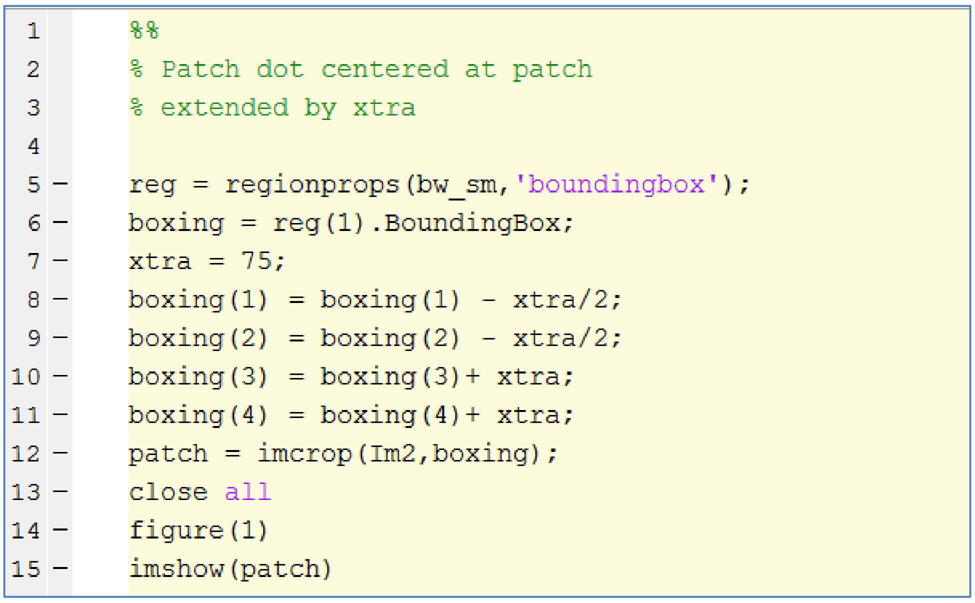
After image dilation, we get the image below. All individual brown pixels and consequently, adult female locations can now be spotted in the image below:



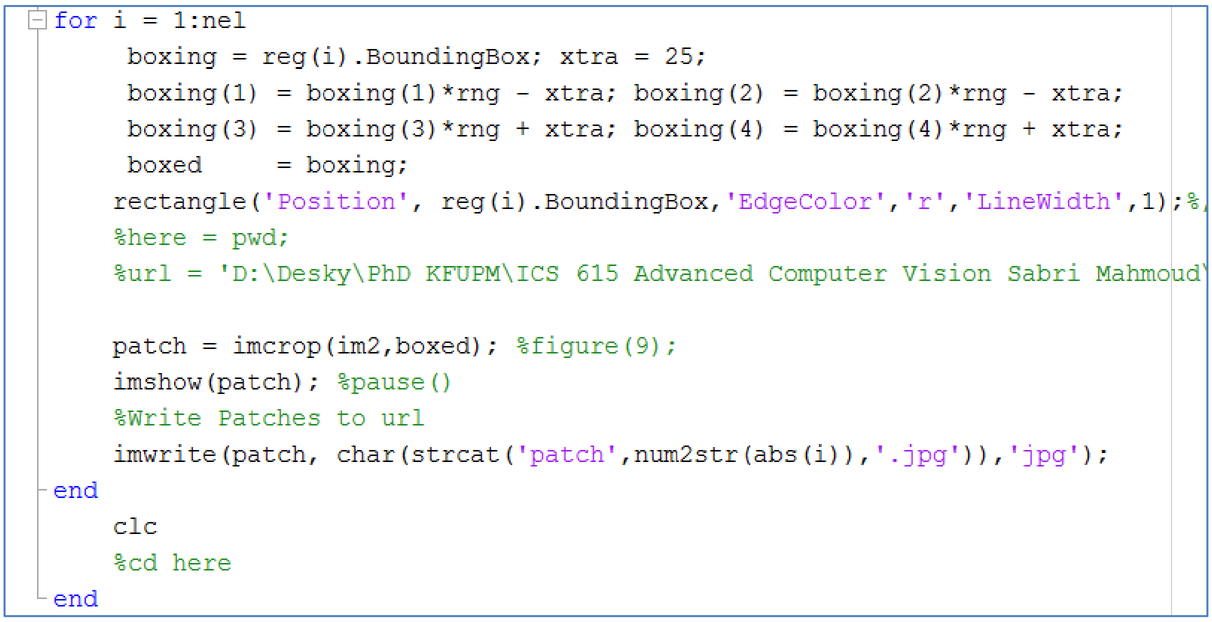
The centroids of these points are then used to extract the patches from the base Train images. Having the centroids, we obtain a bounding box of desired size and plot them over the original Train image as shown below (Red boxes are bounding regions for our patches).



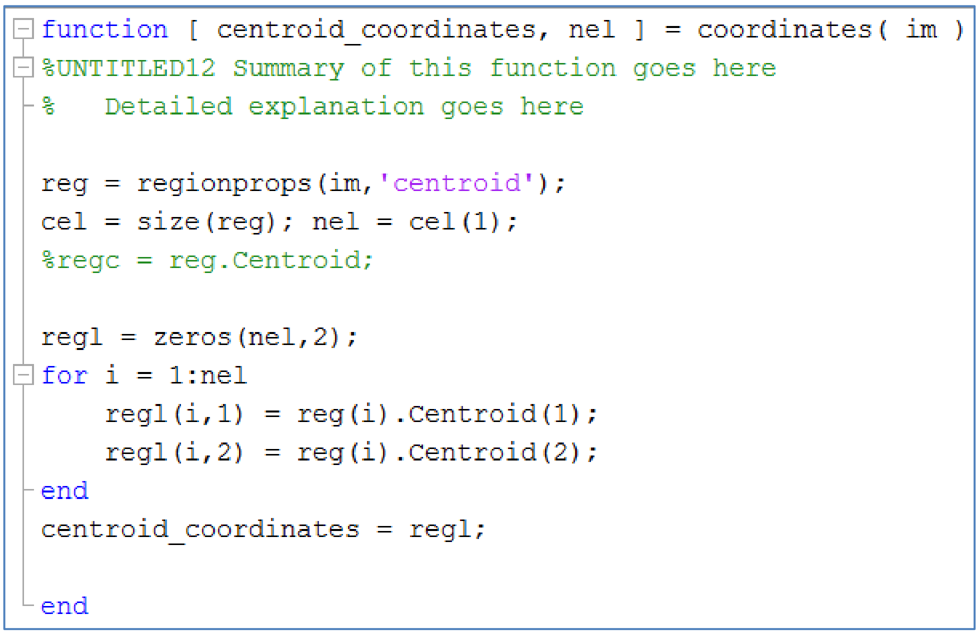
The below images show Matlab scripts for the described processes.



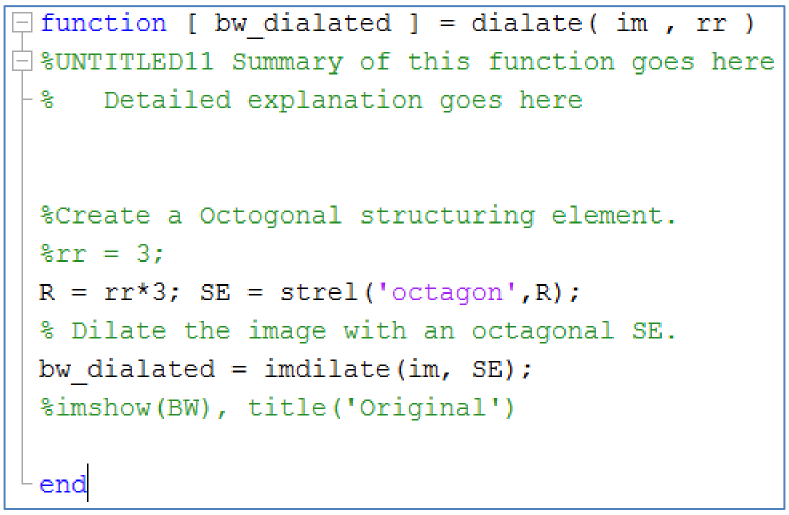
```
%%
% Patch dot centered at patch
% extended by xtra

reg = regionprops(bw_sm,'boundingbox');
boxing = reg(1).BoundingBox;
xtra = 75;
boxing(1) = boxing(1) - xtra/2;
boxing(2) = boxing(2) - xtra/2;
boxing(3) = boxing(3)+ xtra;
boxing(4) = boxing(4)+ xtra;
patch = imcrop(Im2,boxing);
close all
figure(1)
imshow(patch)
```



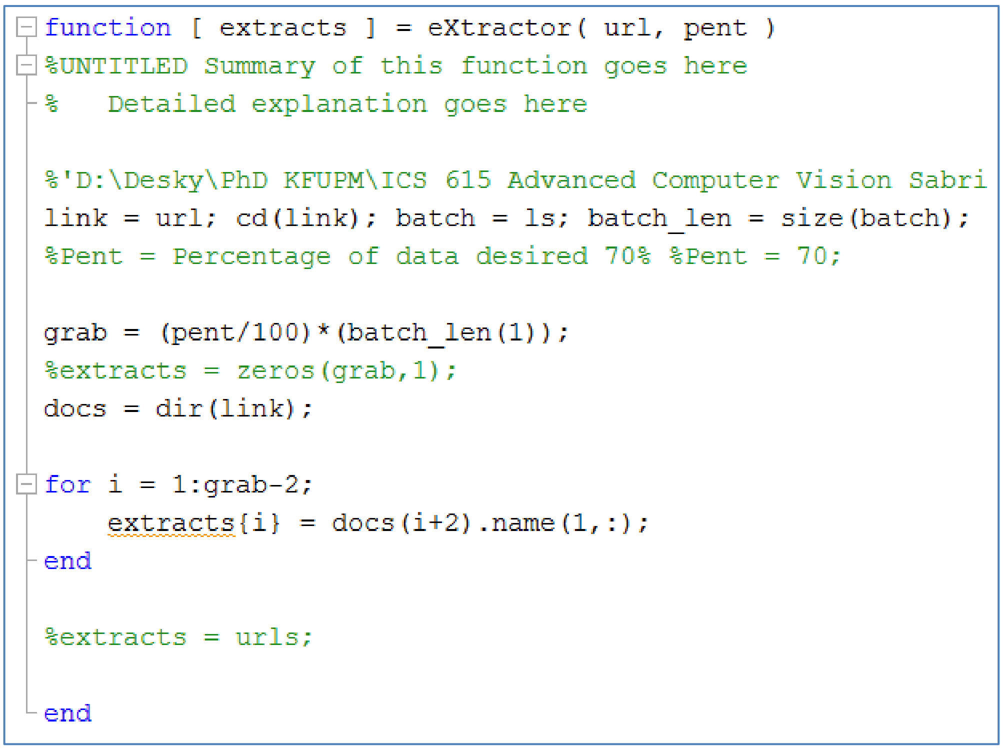
```
for i = 1:nel
     boxing = reg(i).BoundingBox; xtra = 25;
     boxing(1)  = boxing(1)*rng - xtra; boxing(2)  = boxing(2)*rng - xtra;
     boxing(3)  = boxing(3)*rng + xtra; boxing(4)  = boxing(4)*rng + xtra;
     boxed      = boxing;
    rectangle('Position', reg(i).BoundingBox,'EdgeColor','r','LineWidth',1);%
    %here = pwd;
    %url = 'D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\

    patch = imcrop(im2,boxed); %figure(9);
    imshow(patch); %pause()
    %Write Patches to url
    imwrite(patch, char(strcat('patch',num2str(abs(i)),'.jpg')),'jpg');
end
    clc
    %cd here
end
```



```
function [ centroid_coordinates, nel ] = coordinates( im )
%UNTITLED12 Summary of this function goes here
%   Detailed explanation goes here

reg = regionprops(im,'centroid');
cel = size(reg); nel = cel(1);
%regc = reg.Centroid;

regl = zeros(nel,2);
for i = 1:nel
    regl(i,1) = reg(i).Centroid(1);
    regl(i,2) = reg(i).Centroid(2);
end
centroid_coordinates = regl;

end
```



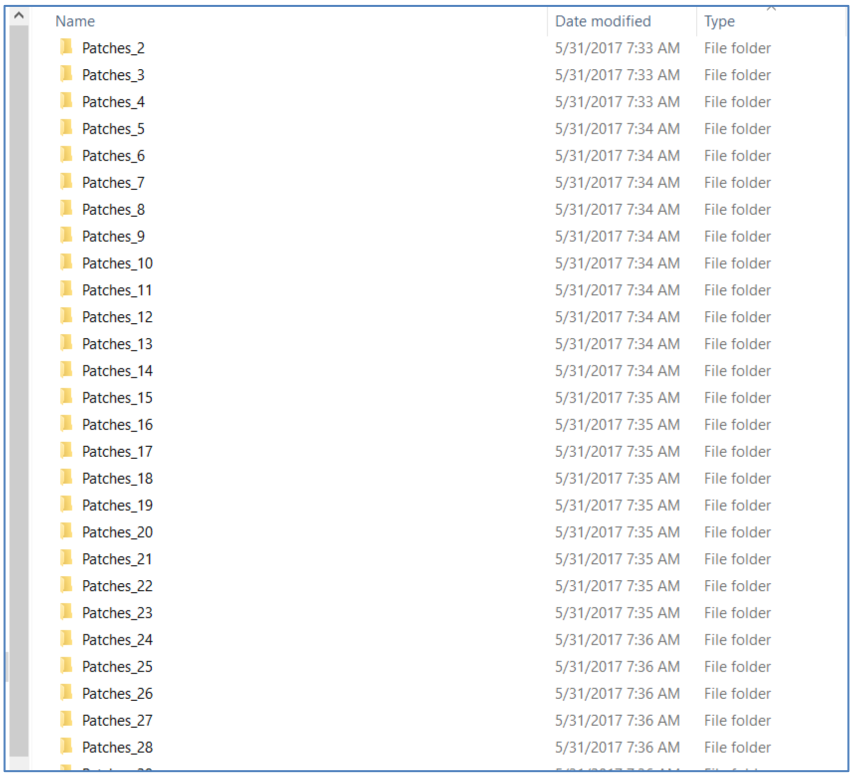
```
function [ bw_dialated ] = dialate( im , rr )
%UNTITLED11 Summary of this function goes here
%   Detailed explanation goes here


%Create a Octogonal structuring element.
%rr = 3;
R = rr*3; SE = strel('octagon',R);
% Dilate the image with an octagonal SE.
bw_dialated = imdilate(im, SE);
%imshow(BW), title('Original')

end
```



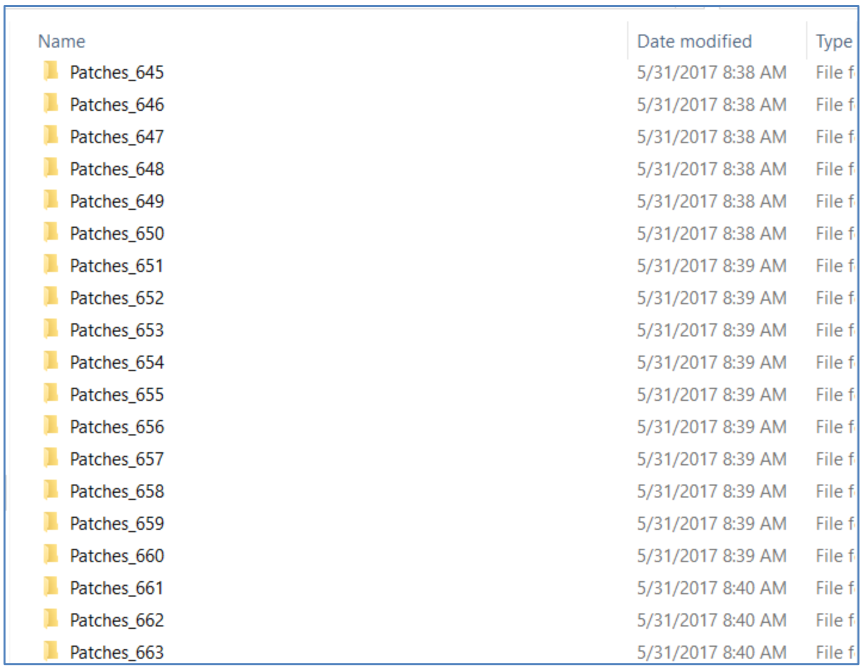
```
function [ extracts ] = eXtractor( url, pent )
%UNTITLED Summary of this function goes here
%   Detailed explanation goes here

%'D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri
link = url; cd(link); batch = ls; batch_len = size(batch);
%Pent = Percentage of data desired 70% %Pent = 70;

grab = (pent/100)*(batch_len(1));
%extracts = zeros(grab,1);
docs = dir(link);

for i = 1:grab-2;
    extracts{i} = docs(i+2).name(1,:);
end

%extracts = urls;

end
```

2.1.4 Step 4: Handling 600+ folders and 60,000+ files containing patches

Finally, a Matlab function is written to recursively extract patches and write them to a new file named $Patches\_\{i\}$. From which we get the following patches.

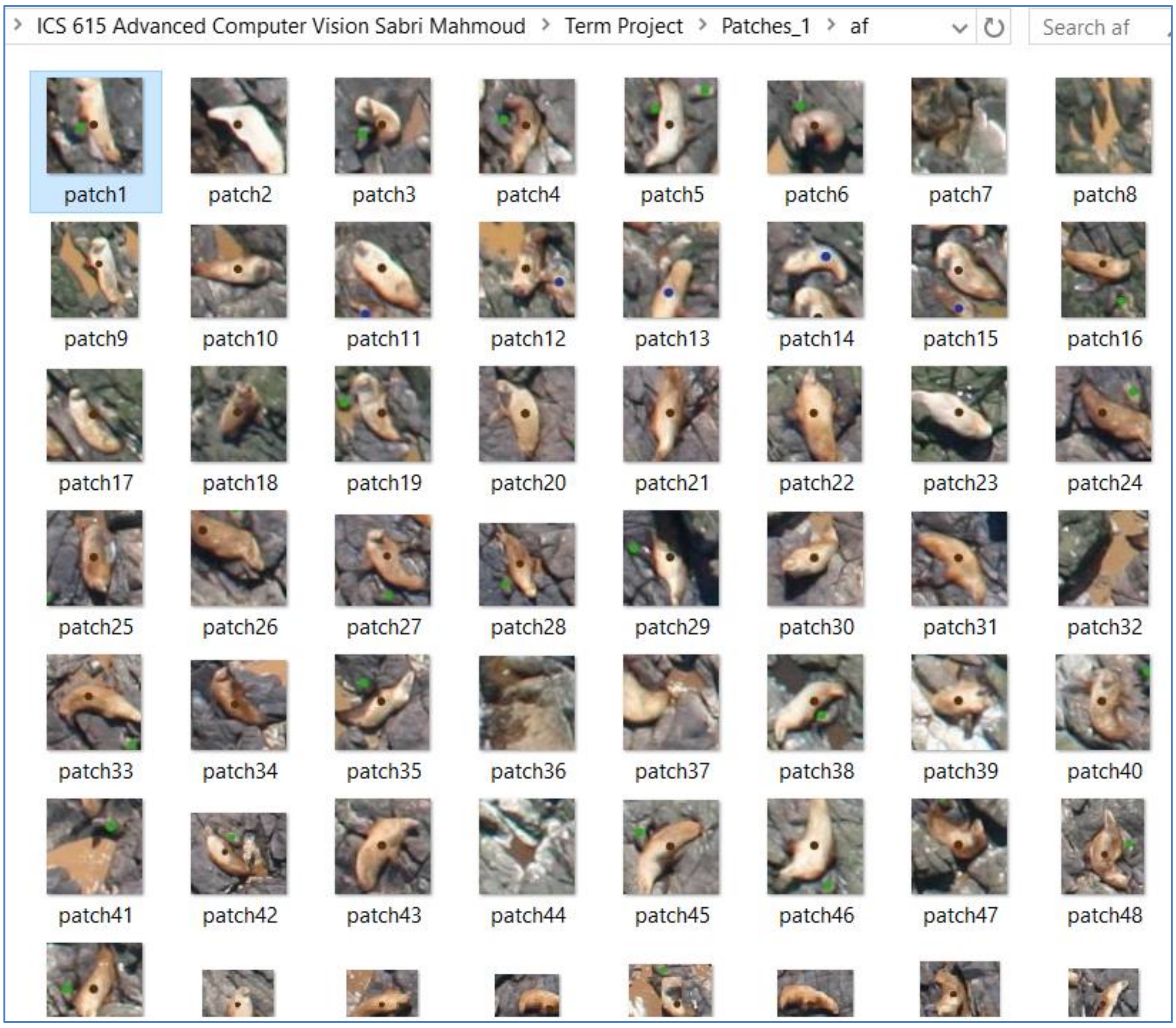


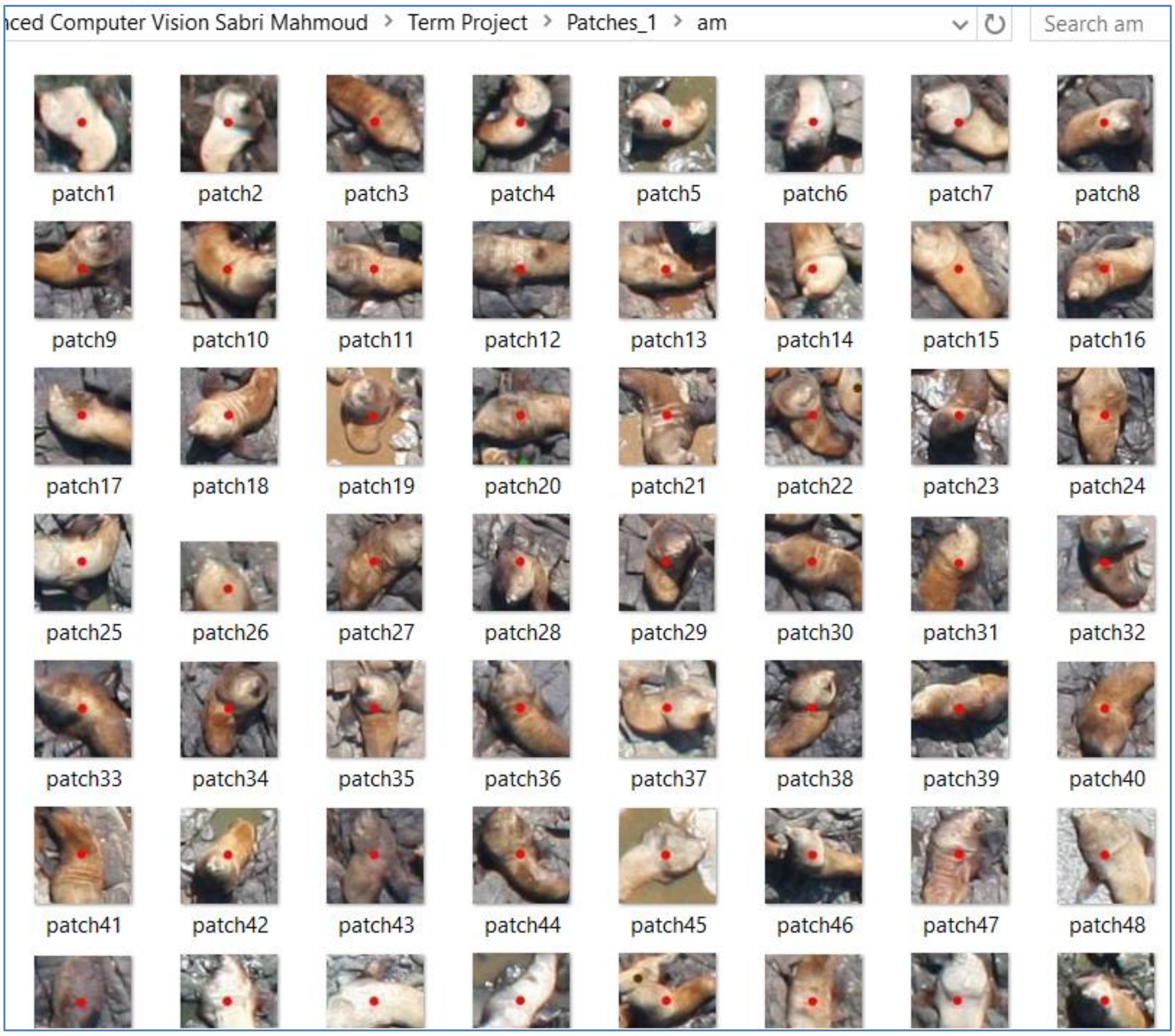
Name	Date modified	Type
Patches_2	5/31/2017 7:33 AM	File folder
Patches_3	5/31/2017 7:33 AM	File folder
Patches_4	5/31/2017 7:33 AM	File folder
Patches_5	5/31/2017 7:34 AM	File folder
Patches_6	5/31/2017 7:34 AM	File folder
Patches_7	5/31/2017 7:34 AM	File folder
Patches_8	5/31/2017 7:34 AM	File folder
Patches_9	5/31/2017 7:34 AM	File folder
Patches_10	5/31/2017 7:34 AM	File folder
Patches_11	5/31/2017 7:34 AM	File folder
Patches_12	5/31/2017 7:34 AM	File folder
Patches_13	5/31/2017 7:34 AM	File folder
Patches_14	5/31/2017 7:34 AM	File folder
Patches_15	5/31/2017 7:35 AM	File folder
Patches_16	5/31/2017 7:35 AM	File folder
Patches_17	5/31/2017 7:35 AM	File folder
Patches_18	5/31/2017 7:35 AM	File folder
Patches_19	5/31/2017 7:35 AM	File folder
Patches_20	5/31/2017 7:35 AM	File folder
Patches_21	5/31/2017 7:35 AM	File folder
Patches_22	5/31/2017 7:35 AM	File folder
Patches_23	5/31/2017 7:35 AM	File folder
Patches_24	5/31/2017 7:36 AM	File folder
Patches_25	5/31/2017 7:36 AM	File folder
Patches_26	5/31/2017 7:36 AM	File folder
Patches_27	5/31/2017 7:36 AM	File folder
Patches_28	5/31/2017 7:36 AM	File folder



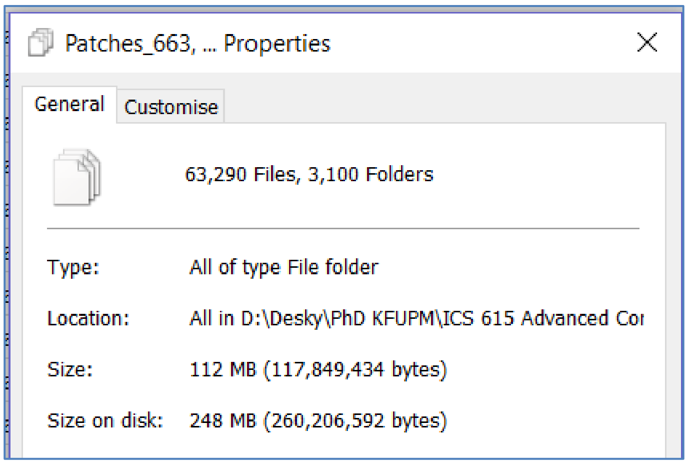
Name	Date modified	Type
Patches_645	5/31/2017 8:38 AM	File f
Patches_646	5/31/2017 8:38 AM	File f
Patches_647	5/31/2017 8:38 AM	File f
Patches_648	5/31/2017 8:38 AM	File f
Patches_649	5/31/2017 8:38 AM	File f
Patches_650	5/31/2017 8:38 AM	File f
Patches_651	5/31/2017 8:39 AM	File f
Patches_652	5/31/2017 8:39 AM	File f
Patches_653	5/31/2017 8:39 AM	File f
Patches_654	5/31/2017 8:39 AM	File f
Patches_655	5/31/2017 8:39 AM	File f
Patches_656	5/31/2017 8:39 AM	File f
Patches_657	5/31/2017 8:39 AM	File f
Patches_658	5/31/2017 8:39 AM	File f
Patches_659	5/31/2017 8:39 AM	File f
Patches_660	5/31/2017 8:39 AM	File f
Patches_661	5/31/2017 8:40 AM	File f
Patches_662	5/31/2017 8:40 AM	File f
Patches_663	5/31/2017 8:40 AM	File f

Samples from **af** and **am** folders are shown below:





In total, **63,290** patches were extracted from **663** images representing **70%** of the “Train” dataset.



63,290 Files, 3,100 Folders

Type: All of type File folder

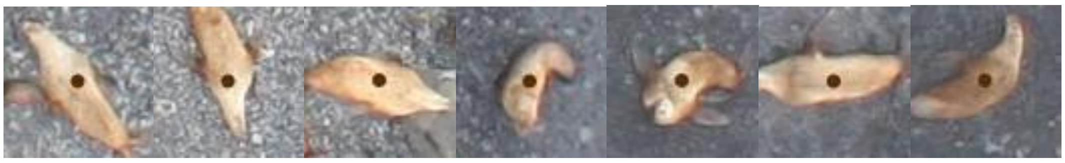
Location: All in D:\Desky\PhD KFUPM\ICS 615 Advanced Coı

Size: 112 MB (117,849,434 bytes)

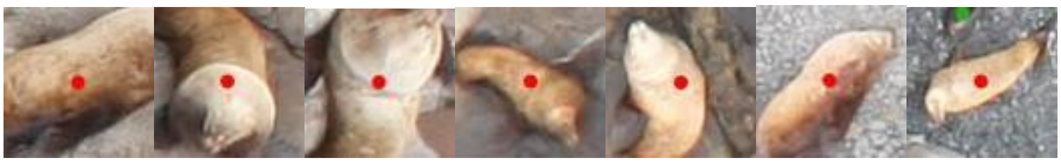
Size on disk: 248 MB (260,206,592 bytes)

The figures below are samples from the patches extracted and their different classes.

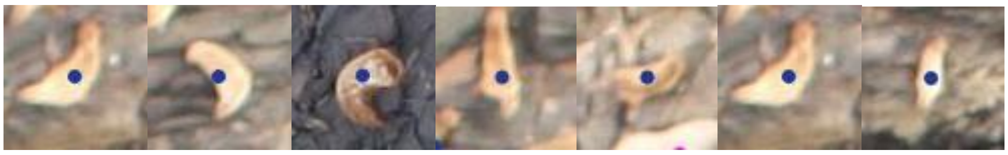
Adult Females



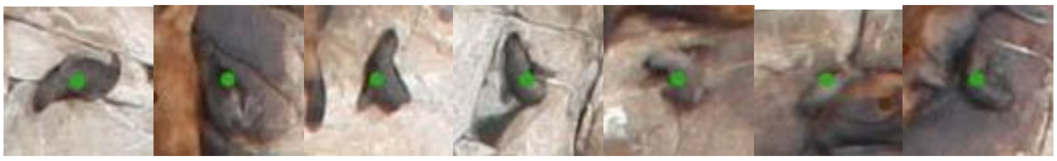
Adult Males



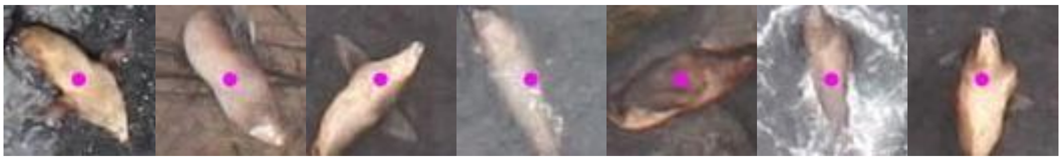
Juveniles



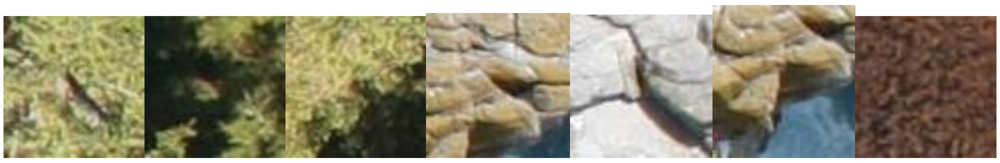
Pups



Sub-Adult Males

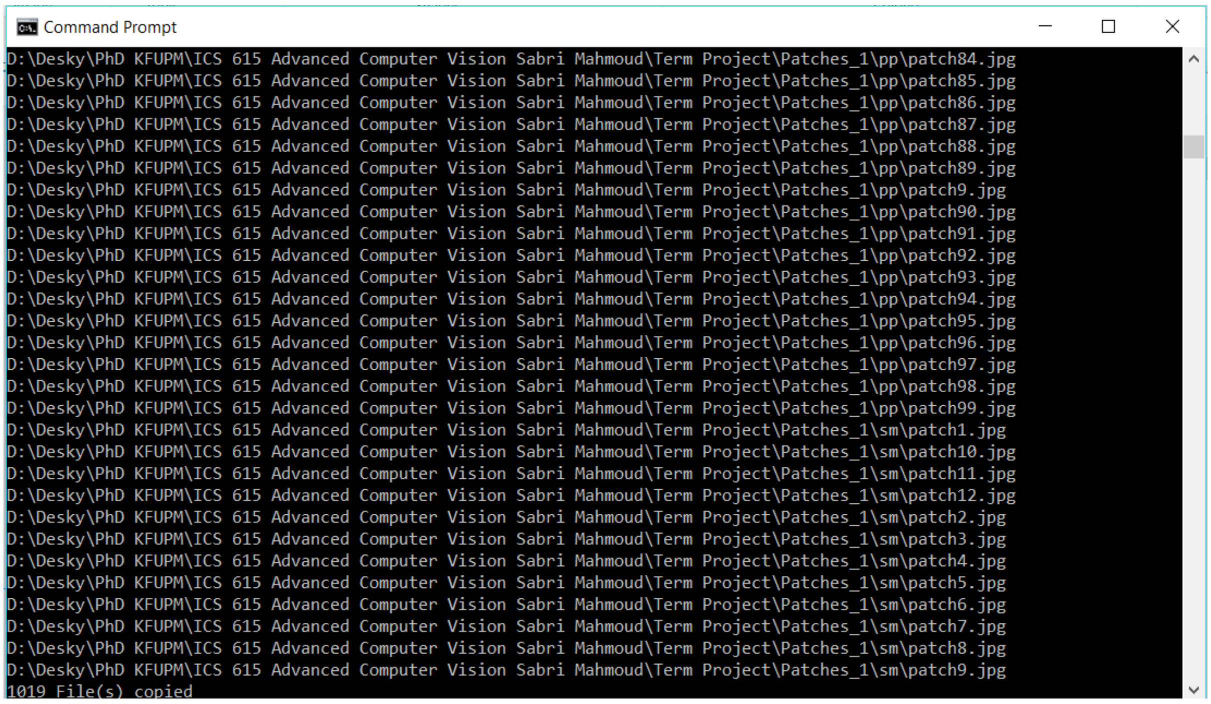


Bush, Rocks and Sea



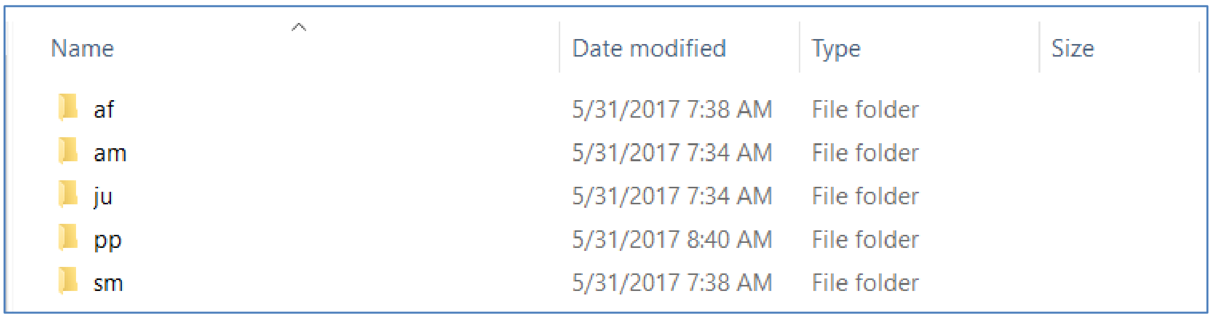
Now we move all the patches to a folder representing their classes. With **af** = adult females, **am** = adult males, **ju** = juveniles, **pp** = pups and **sm** = sub-adult males. To move over **+60,000** files, we sought the use of the command prompt "**xcopy**" function for window files:

- ***xcopy "D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_{i}" "D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\AllPatches" /s /y***



```
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch84.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch85.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch86.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch87.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch88.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch89.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch9.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch90.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch91.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch92.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch93.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch94.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch95.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch96.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch97.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch98.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\pp\patch99.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch1.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch10.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch11.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch12.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch2.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch3.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch4.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch5.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch6.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch7.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch8.jpg
D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Patches_1\sm\patch9.jpg
1019 File(s) copied
```

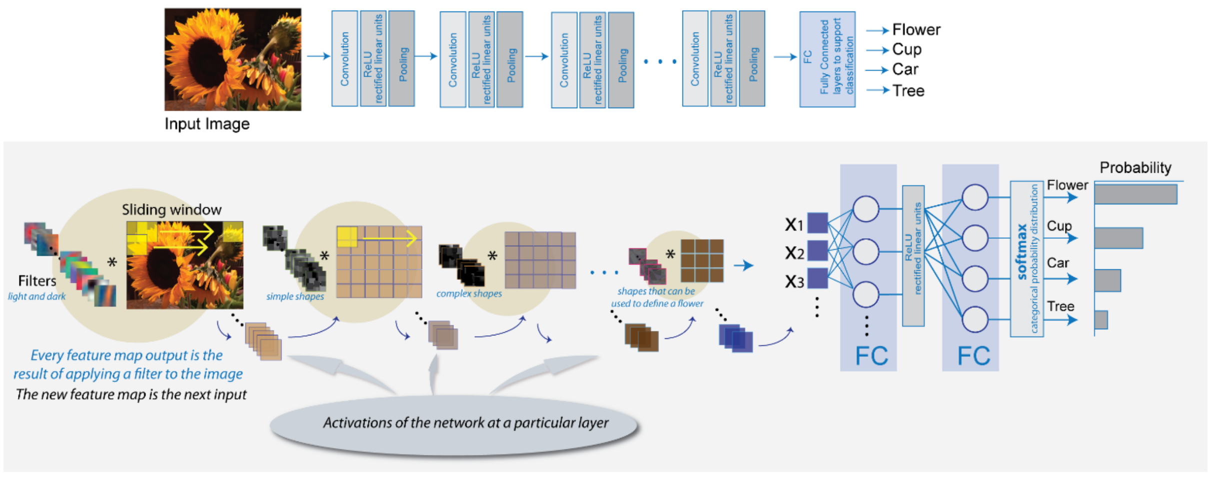
Thus, we obtain all the patches for their respective classes in single-class folders:



Name	Date modified	Type	Size
af	5/31/2017 7:38 AM	File folder	
am	5/31/2017 7:34 AM	File folder	
ju	5/31/2017 7:34 AM	File folder	
pp	5/31/2017 8:40 AM	File folder	
sm	5/31/2017 7:38 AM	File folder	

3.0 Deep learning using CNN (Training the Patches).

Convolution neural networks (ConvNets or CNNs) are essential tools for deep learning, and are especially suited for image recognition. In Matlab, a ConvNet architecture network can easily be trained and used to predict class labels. Training on a GPU or in parallel requires the Parallel Computing Toolbox™ and GPUs require a CUDA®-enabled NVIDIA® GPU with compute capability 3.0 or higher. The figure below shows the computational workflow of CNN in Matlab.

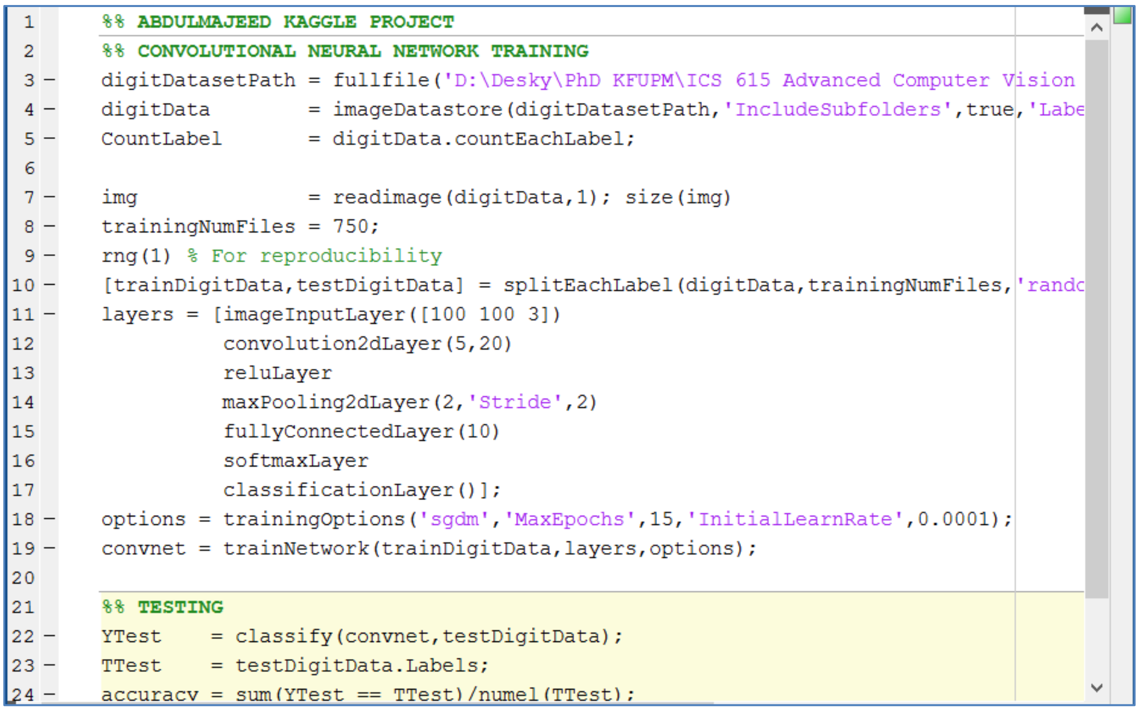


Input Image
Convolution
ReLU rectified linear units
Pooling
FC Fully Connected layers to support classification
Flower
Cup
Car
Tree
Sliding window
Filters
light and dark
simple shapes
complex shapes
shapes that can be used to define a flower
Every feature map output is the result of applying a filter to the image
The new feature map is the next input
Activations of the network at a particular layer
X1
X2
X3
FC
ReLU rectified linear units
softmax
categorical probability distribution
Probability

Training the CNN network starts from specifying the input image sizes. The input images are the patches in 100x100 pixels with 6 class labels (including rough patches). Out of the extracted patches, 100% of the patches extracted are used for training the CNN since only 70% of the images were initially used for patch extraction. Similarly, we define standard training parameters. After defining the layers (network structure), the *training options* function is used to specify the network training parameters. The options for the stochastic gradient descent with momentum are set to default. The maximum number of full training cycles (epochs) = 50 and training is started with an initial learning rate of 0.0001.

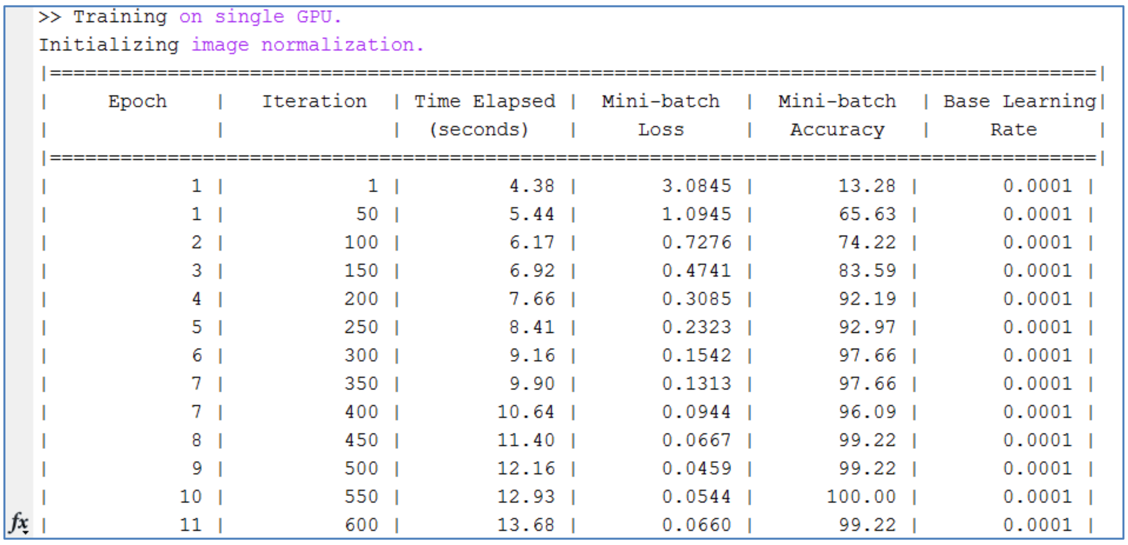
- **Image Input Layer:** The *imageInputLayer* is where the image size is specified, which, in this case, is 100 x 100 x 3. Corresponding to the height, width, and the channel size (colored = 3).
- **Convolutional Layer:** In the convolutional layer, the first argument is *filterSize*, which is the height and width of the filters the training function uses while scanning along the images. In this example, the filter size is set to 5 which indicates a 5 x 5 filter kernel. The second argument is the number of filters, which is the number of neurons that connect to the same region of the output. This parameter determines the number of the feature maps. The Stride or learning rates for this layer is defined in the call to *convolution2dLayer*.

- **ReLU Layer:** The convolutional layer is followed by a nonlinear activation function. MATLAB uses the rectified linear unit function, specified by *reluLayer*.
- **Max-Pooling Layer:** The convolutional layer (with the activation function) is usually followed by a down-sampling operation to reduce the number of parameters and as another way of avoiding overfitting. One way of down-sampling is max-pooling, which is specified by the *maxPooling2dLayer* function. This layer returns the maximum values of rectangular regions of inputs, specified by the *poolSize*. In this example, the size of the rectangular region is 2 x 2. The optional argument Stride determines the step size the training function takes as it scans along the image. This max-pooling layer takes place between the convolutional layers when there are multiple convolutional layers in the network.
- **Fully Connected Layer:** The convolutional (and down-sampling) layers are followed by one or more fully connected layers. As the name suggests, all neurons in the fully connected layer connect to the neurons in the previous layer. This layer combines all the features learned by the previous layers across the image to identify the larger patterns. The last fully connected layer combines them to classify the images. Thus, the *OutputSize* parameter in the last fully connected layer is equal to the number of classes in the target data. In this example the output size is 6, corresponding to the 6 classes of Adult males, Sub Adult males, Adult females, Juveniles, Pups and Rough (rough contains bush, rock and sea patches).
- **Softmax Layer:** The fully connected layer uses the softmax activation function for classification. The softmax layer is added using the *softmaxLayer* function after the last fully connected layer.
- **Classification Layer:** The final layer is the classification layer, defined by using the *classificationLayer* function. This layer uses the probabilities returned by the softmax activation function for each input to assign it to one of the mutually exclusive classes.



```
%% ABDULMAJEED KAGGLE PROJECT
%% CONVOLUTIONAL NEURAL NETWORK TRAINING
digitDatasetPath = fullfile('D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision
digitData        = imageDatastore(digitDatasetPath,'IncludeSubfolders',true,'Labe
CountLabel       = digitData.countEachLabel;

img              = readimage(digitData,1); size(img)
trainingNumFiles = 750;
rng(1) % For reproducibility
[trainDigitData,testDigitData] = splitEachLabel(digitData,trainingNumFiles,'rando
layers = [imageInputLayer([100 100 3])
          convolution2dLayer(5,20)
          reluLayer
          maxPooling2dLayer(2,'Stride',2)
          fullyConnectedLayer(10)
          softmaxLayer
          classificationLayer()];
options = trainingOptions('sgdm','MaxEpochs',15,'InitialLearnRate',0.0001);
convnet = trainNetwork(trainDigitData,layers,options);

%% TESTING
YTest    = classify(convnet,testDigitData);
TTest    = testDigitData.Labels;
accuracy = sum(YTest == TTest)/numel(TTest);
```

- Training Loss and Accuracy – first 10 epochs



```
>> Training on single GPU.
Initializing image normalization.
|=========================================================================================|
|     Epoch    |   Iteration  | Time Elapsed |  Mini-batch  |  Mini-batch  | Base Learning|
|              |              |  (seconds)   |     Loss     |   Accuracy   |     Rate     |
|=========================================================================================|
|            1 |            1 |         4.38 |       3.0845 |       13.28 |       0.0001 |
|            1 |           50 |         5.44 |       1.0945 |       65.63 |       0.0001 |
|            2 |          100 |         6.17 |       0.7276 |       74.22 |       0.0001 |
|            3 |          150 |         6.92 |       0.4741 |       83.59 |       0.0001 |
|            4 |          200 |         7.66 |       0.3085 |       92.19 |       0.0001 |
|            5 |          250 |         8.41 |       0.2323 |       92.97 |       0.0001 |
|            6 |          300 |         9.16 |       0.1542 |       97.66 |       0.0001 |
|            7 |          350 |         9.90 |       0.1313 |       97.66 |       0.0001 |
|            7 |          400 |        10.64 |       0.0944 |       96.09 |       0.0001 |
|            8 |          450 |        11.40 |       0.0667 |       99.22 |       0.0001 |
|            9 |          500 |        12.16 |       0.0459 |       99.22 |       0.0001 |
|           10 |          550 |        12.93 |       0.0544 |      100.00 |       0.0001 |
|           11 |          600 |        13.68 |       0.0660 |       99.22 |       0.0001 |
```

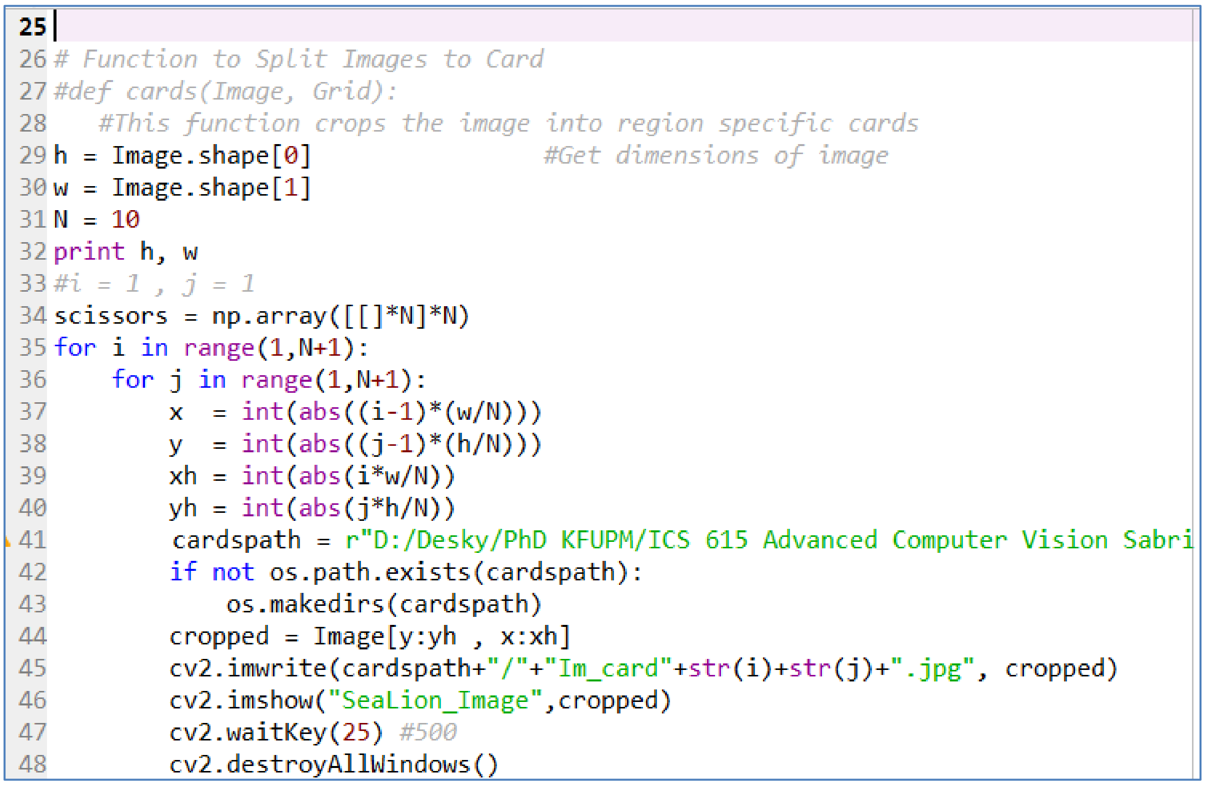
Confusion Matrix {CNFM}:

X	*Adult Males*	*Adult Females*	*Sub Adult Males*	*Juveniles*	*Pups*
Adult Males	**0.736**	0.123	0.314	0.103	0.001
Adult Females	0.123	**0.626**	0.172	0.121	0.001
Sub Adult Males	0.233	0.100	**0.623**	0.272	0.001
Juveniles	0.010	0.011	0.114	**0.593**	0.001
Pups	0.001	0.001	0.001	0.001	**0.538**

Accuracy of our CNN Model = tr[diag{CNFM}] = 0.6232 = **62.32%**

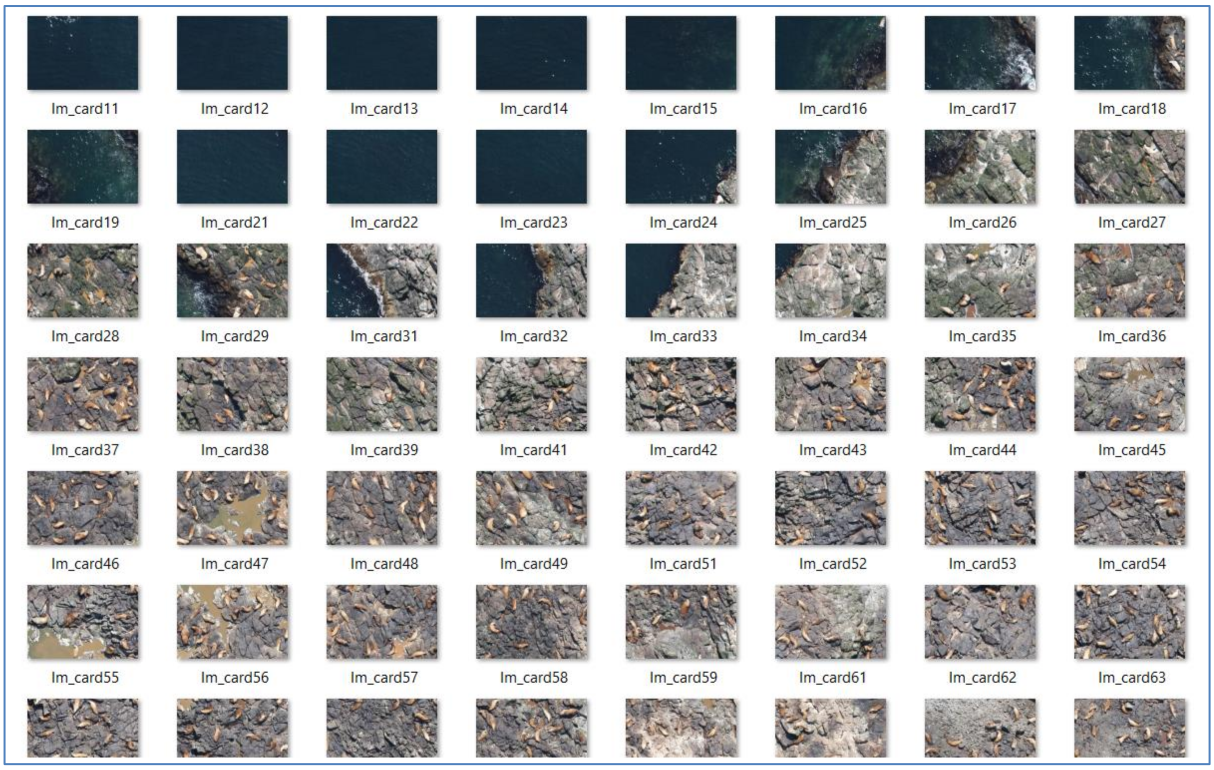
4.0 Testing

The process of testing the algorithm is tricky. First we realize that the image is large and the objects therein are tiny. It is possible to use R-CNN which acts on specific regions of the image. However, we propose our own unique technique here. First, the original train image is divided into 10 x 10 smaller images using our custom function programmed in Python called '*cards*' as show in the figure below.

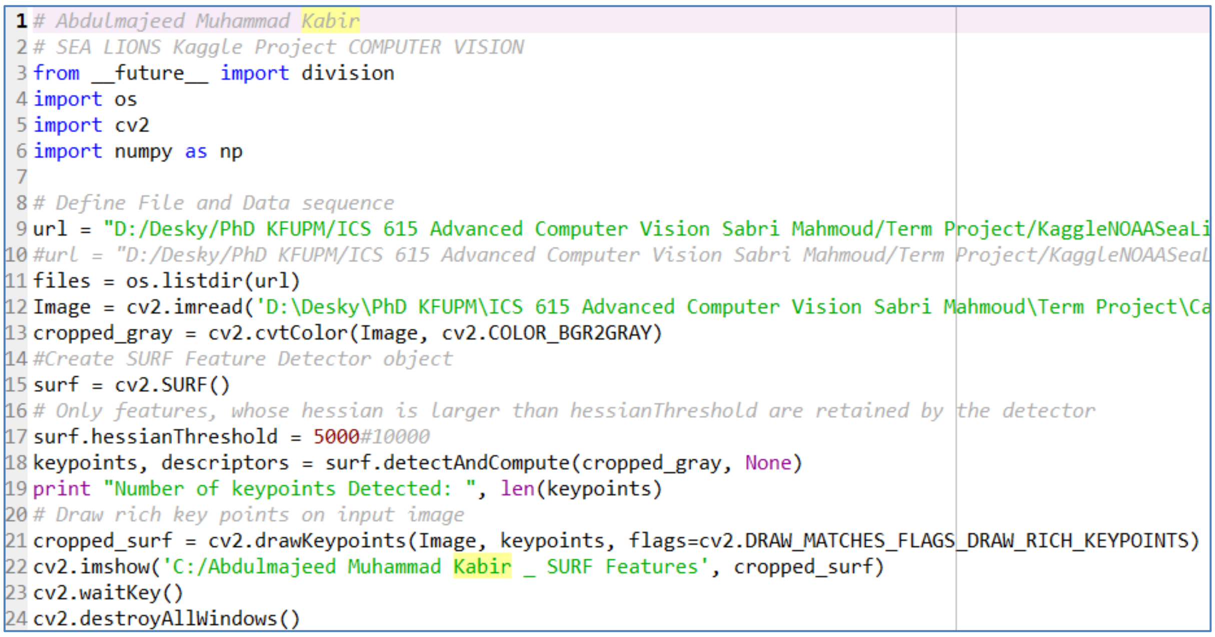


```

# Function to Split Images to Card
#def cards(Image, Grid):
    #This function crops the image into region specific cards
h = Image.shape[0]                  #Get dimensions of image
w = Image.shape[1]
N = 10
print h, w
#i = 1 , j = 1
scissors = np.array([[]*N]*N)
for i in range(1,N+1):
    for j in range(1,N+1):
        x  = int(abs((i-1)*(w/N)))
        y  = int(abs((j-1)*(h/N)))
        xh = int(abs(i*w/N))
        yh = int(abs(j*h/N))
        cardspath = r"D:/Desky/PhD KFUPM/ICS 615 Advanced Computer Vision Sabri
        if not os.path.exists(cardspath):
            os.makedirs(cardspath)
        cropped = Image[y:yh , x:xh]
        cv2.imwrite(cardspath+"/"+"Im_card"+str(i)+str(j)+".jpg", cropped)
        cv2.imshow("SeaLion_Image",cropped)
        cv2.waitKey(25) #500
        cv2.destroyAllWindows()
```

The resulting 10 x 10 grid card-images of the first image is shown below. Dividing the images into cards makes it easy to perform the SURF detection of Sea lions with less number of outliers and test-patch overlap. However, the results need to be added together for hundred card-images is equal to one test image. In summary, each test image is divided into 100 cards (smaller images derived by cutting the test image into 10 x 10 regions). Interest points using SURF are found for each card. Overlapping interest points are removed. 100 x 100 test-patches are extracted from each interest point. Each test-patch is then evaluated with the CNN model. The figure below shows the 10 x 10 cards for the first test image 0.jpg.



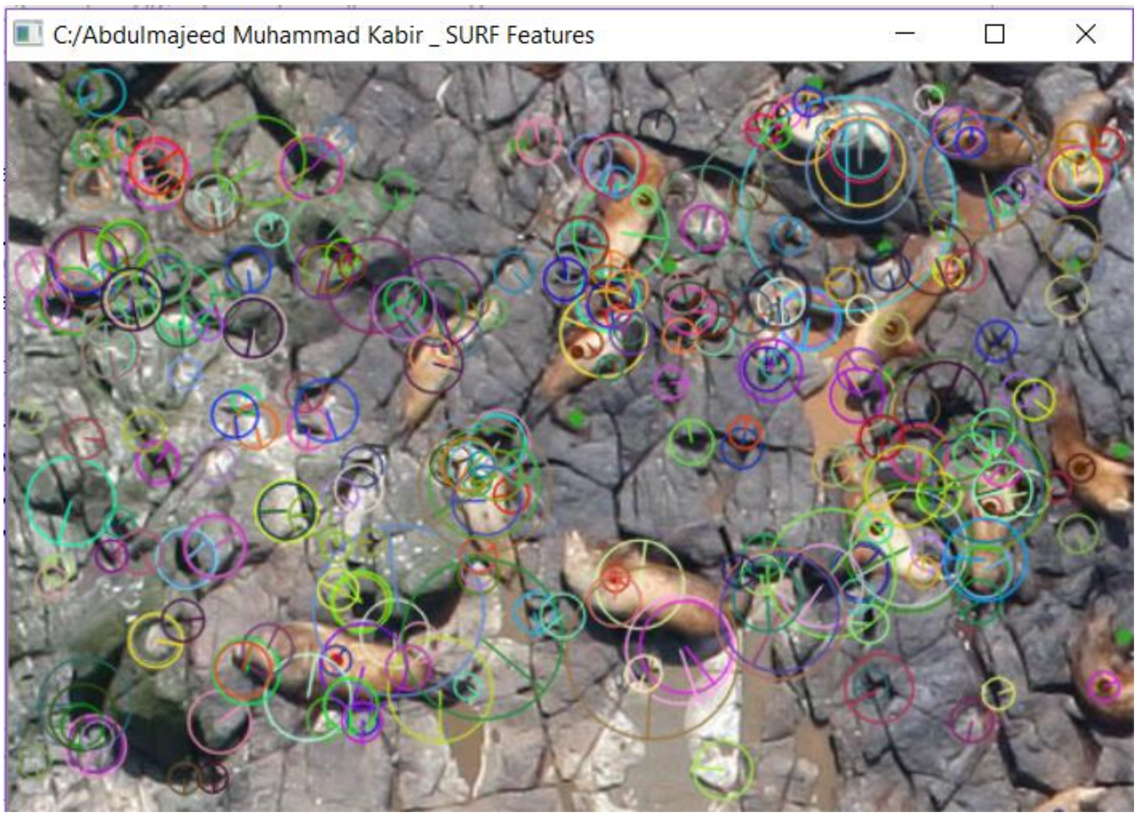
In our proposed method, instead of running a sliding window through the entire image, we use SURF to detect interest point in the image. The program is done using OpenCV as shown below.



```
# Abdulmajeed Muhammad Kabir
# SEA LIONS Kaggle Project COMPUTER VISION
from __future__ import division
import os
import cv2
import numpy as np

# Define File and Data sequence
url = "D:/Desky/PhD KFUPM/ICS 615 Advanced Computer Vision Sabri Mahmoud/Term Project/KaggleNOAASeaLi
#url = "D:/Desky/PhD KFUPM/ICS 615 Advanced Computer Vision Sabri Mahmoud/Term Project/KaggleNOAASeaL
files = os.listdir(url)
Image = cv2.imread('D:\Desky\PhD KFUPM\ICS 615 Advanced Computer Vision Sabri Mahmoud\Term Project\Ca
cropped_gray = cv2.cvtColor(Image, cv2.COLOR_BGR2GRAY)
#Create SURF Feature Detector object
surf = cv2.SURF()
# Only features, whose hessian is larger than hessianThreshold are retained by the detector
surf.hessianThreshold = 5000#10000
keypoints, descriptors = surf.detectAndCompute(cropped_gray, None)
print "Number of keypoints Detected: ", len(keypoints)
# Draw rich key points on input image
cropped_surf = cv2.drawKeypoints(Image, keypoints, flags=cv2.DRAW_MATCHES_FLAGS_DRAW_RICH_KEYPOINTS)
cv2.imshow('C:/Abdulmajeed Muhammad Kabir _ SURF Features', cropped_surf)
cv2.waitKey()
cv2.destroyAllWindows()
```

The algorithm relies on the fact that an interest point must fall on the location of Sea Lion as long as an appropriate SURF hessian threshold is set (5000). The threshold is set as a trade-off between the number of interest points detected and the number of computations involved. Figure below shows the detection of interest points with each Sea Lion having at least an interest point. There are outliers however. Next is filtering out interest points. We remove all interest points that have a distance less than 50 pixels from one another. This is because we want the interest points to be representative of unique regions in the image where a Sea Lion is expected to exist. After the extraction of interest points, we crop the region around the interest point to form 100x100 pixel sample test-patches. Then each of these patches is run through the CNN model for class validation.

SURF Interest point detection:



Total Key Points Detected: 312

It is obvious from the above image that there is a representative interest point on each Sea Lion.

5.0 Testing

Final Results (Testing with 5 images)

	Our Algorithm							**Ground Truth**					
Image	**IP**	**SL**	**AM**	**AF**	**SAM**	**JU**	**PP**	**SL**	**AM**	**AF**	**SAM**	**JU**	**PP**
0.jpg	***937***	***937***	***60***	***483***	***13***	***40***	***341***	***946***	***62***	***486***	***12***	***42***	***344***
1.jpg	***32***	***32***	***2***	***0***	***18***	***11***	***1***	***34***	***2***	***0***	***20***	***12***	***0***
2.jpg	***62***	***62***	***2***	***37***	***2***	***19***	***2***	***60***	***2***	***38***	***0***	***20***	***0***
3.jpg	***100***	***100***	***8***	***41***	***5***	***9***	***37***	***99***	***8***	***41***	***5***	***7***	***38***
4.jpg	***17***	***17***	***5***	***3***	***8***	***1***	***0***	***17***	***6***	***2***	***9***	***0***	***0***

IP = *Detected SURF Interest Points*, **SL** = *Number of Sea Lions*, **AM** = *Adult Males*, **AF** = *Adult Females*, **SAM** = *Sub Adult Males*, **JU** = *Juveniles*, **PP** = *Pups.*

6.0 Summary, Conclusion & Recommendation

Resources

[1]

[2]

[3]

[4]

[5]