The framework includes two stages:

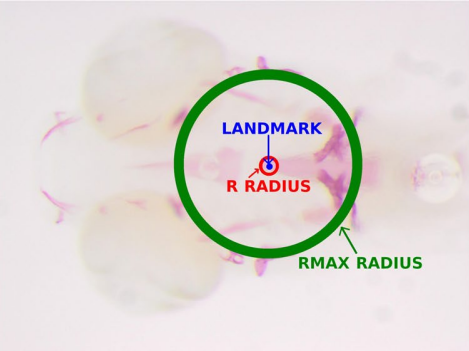
1. **Training Stage**

**The goal of the training stage is to build a classifier which can predict if a point in the image is a landmark point (positive) or not (negative)**

We need to build a classifier for each landmark point in the image. If the image contains 15 landmarks, we need to build 15 classifiers. The process of training the classifier for one landmark is described as follow:

* **Generating a list of positive points and negative points**. The positive/negative samples are extracted from sample images (the images with the positions of each landmark point). A positive point is the point which is very close to the true landmark. A negative point is the point which is far away to the true landmark. On the picture below, the true landmark is marked as blue dot, all the points inside the red circle (R radius) are considered as positive points. All the points outside the red circle and inside the green circle (RMAX radius) are considered as negative points.

The positive/negative points extraction is implemented in “**extract\_point.py**” The output of this step is a list of positive/negative points for each image, stored in “PosNegPoints” folder.



* **Feature extraction**. Each point (both negative and positive) is represented by a feature vector using a feature extraction method. Various feature extraction methods are supported: raw, usub, gsub, ssub, haar. The details of each feature extraction method can be found in the reference paper [1].

The feature extraction is implemented in the “**FeatureExtract.py**”. The output of this step is a list of positive/negative feature vectors for each image, stored in “Features” folder.

the example code to extract raw feature at point (504,304) is listed below:

*D = 5 # number of scale*

*W = 8 # window size is 2W+1*

*inputF = " egfr\_F\_R\_oly\_2X\_1.tif" #load the input image*

*listImgs = RescaleImage(inputF,D) #create multiscale images*

*listPoints = RescalePoint(504,343,D) # create the point location at each scale*

*raw = computeRAW(listImgs, listPoints, W) # raw is the output feature vector*

* **Training a classifier.** Training a classifier to classify if a feature vector is positive (landmark) or not.

The code to implement the training step is “**train.py**”. The main function is:

def train(lm\_index, train\_list, featureFolder, pos\_subfix\_file, neg\_subfix\_file, outModelFile):

where:

lm\_index: landmark index (in the range of 1-15)

train\_list: a list of images used for training

featureFolder: a folder that stores all the positive/negative feature vectors. This is the output of feature extraction step.

pos\_subfix\_file: the subfix of positive file. For example if the name of the image is”image.jpg”, the pos\_subfix\_file is: “\_pos\_sSub.txt” then the file that stores all the positive feature vector for this image is image\_pos\_sSub.txt

neg\_subfix\_file: the subfix of negative file. For example if the name of the image is”image.jpg”, the neg\_subfix\_file is: “\_neg\_sSub.txt” then the file that stores all the negative feature vector for this image is image\_neg\_sSub.txt

outModelFile: output model file.

Example

#lm\_index = 9

#train\_list="Z:\My Drive\Research\iMorphSharedByHai\Datasets\LabeledData\LeftWingsFeatures/train\_list.txt"

#featureFolder ="Z:\My Drive\Research\iMorphSharedByHai\Datasets\LabeledData\LeftWingsFeaturesIndex9/

#pos\_subfix\_file= "\_nev\_sSub.txt"

#nev\_subfix\_file = "\_pos\_sSub.txt"

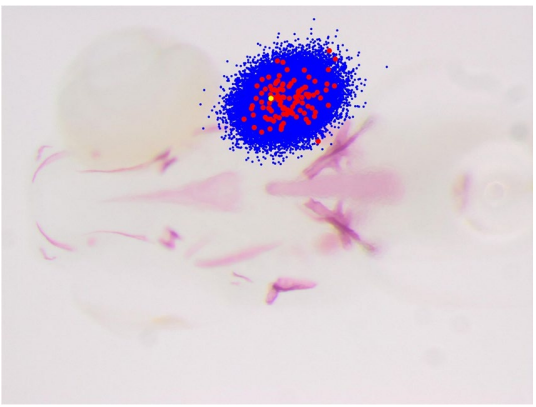
#outModelFile = 'ERT\_gsub.sav'

#train(lm\_index, train\_list, featureFolder, pos\_subfix\_file, neg\_subfix\_file, outModelFile)

1. **Prediction Stage**

The goal of prediction stage is the find the location of each landmark point in the image. The process of landmark prediction is described as follow:

* **Generate a list of candidate points**. For each landmark point, examine the sample data, determine a region where the landmarks should belong to, then randomly select Np points in this region as candidate points. Using feature extraction to extract a feature vector for each candidate point.



In red, the position of landmark 8 as observed in all the images of the ZEBRA dataset, overlaid on an image. In blue, the position of the corresponding 30,000 examples extracted during prediction according to our sampling strategy. In yellow, the real landmark position [1]

* **Prediction** Use the classifier trained in the training stage to predict if a candidate point is positive or negative. After predicting all candidate points, we achieve a list of positive points. The landmark we want to locate is computed as the center point of all positive points.

The code to implement prediction stage can be found in “detection.py”