

*BME 404  
IMAGING PROJECT*

# **RECONSTRUCTION AND QUANTIFICATION OF 3D IRIS SURFACE FOR ANGLE-CLOSURE GLAUCOMA DETECTION IN ANTERIOR SEGMENT OCT:**

*PREPROCESSING, AUGMENTATION, WAVE REFINEMENT BLOCK, EVALUATION METRICS*

*PRESENTED BY:  
MAHMUD WASIF NAFEE*

*1818002*

# DATA AUGMENTATION AND PREPROCESSING



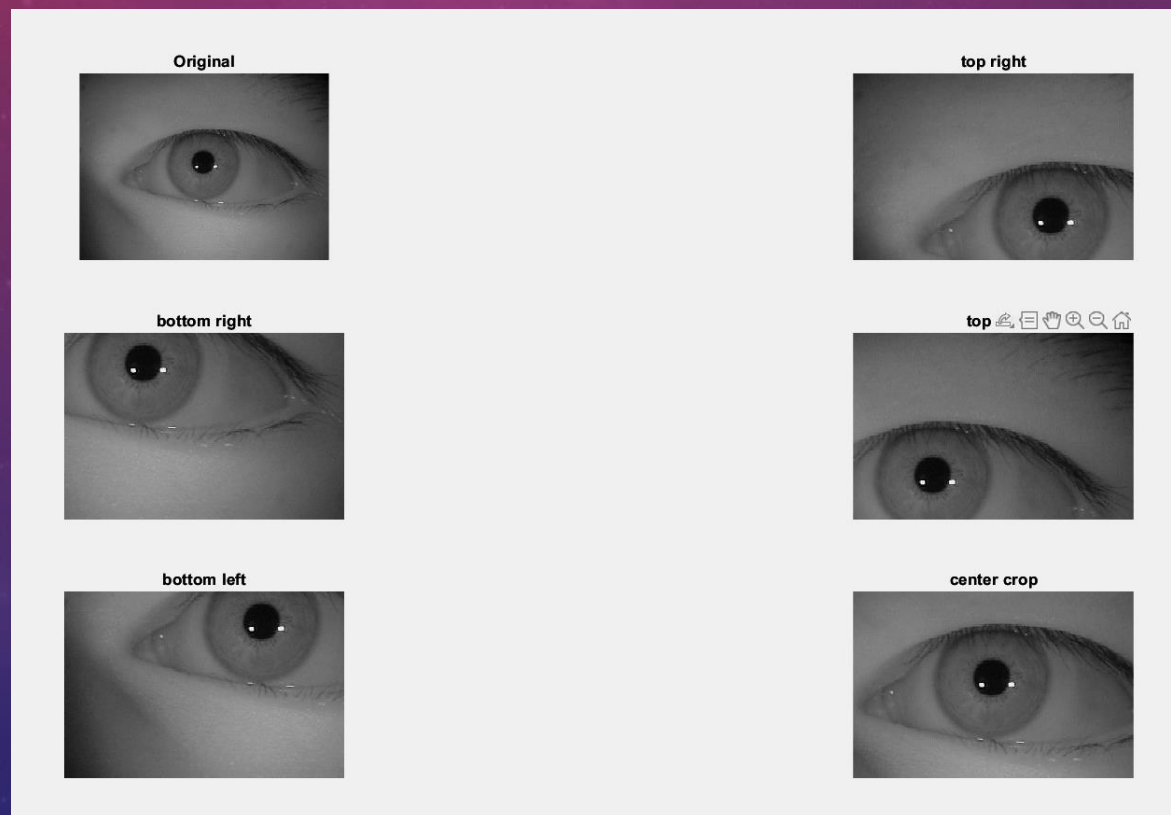


# A LITTLE WORD ABOUT OUR DATASET

- This paper uses the CASIA-Iris-Complex dataset
- CASIA-Iris-Complex contains 22,932 images from 292 Asian subjects. It includes two subsets: CASIA-Iris-CX1 and CASIA-Iris-CX2. All images were collected under NIR illumination and two eyes were captured simultaneously.
- While this dataset provides us an excellent opportunity to perform a challenging task such as iris segmentation, we should keep in mind that very rarely can we take pictures of iris so uniformly and cautiously without very high end camera set up.

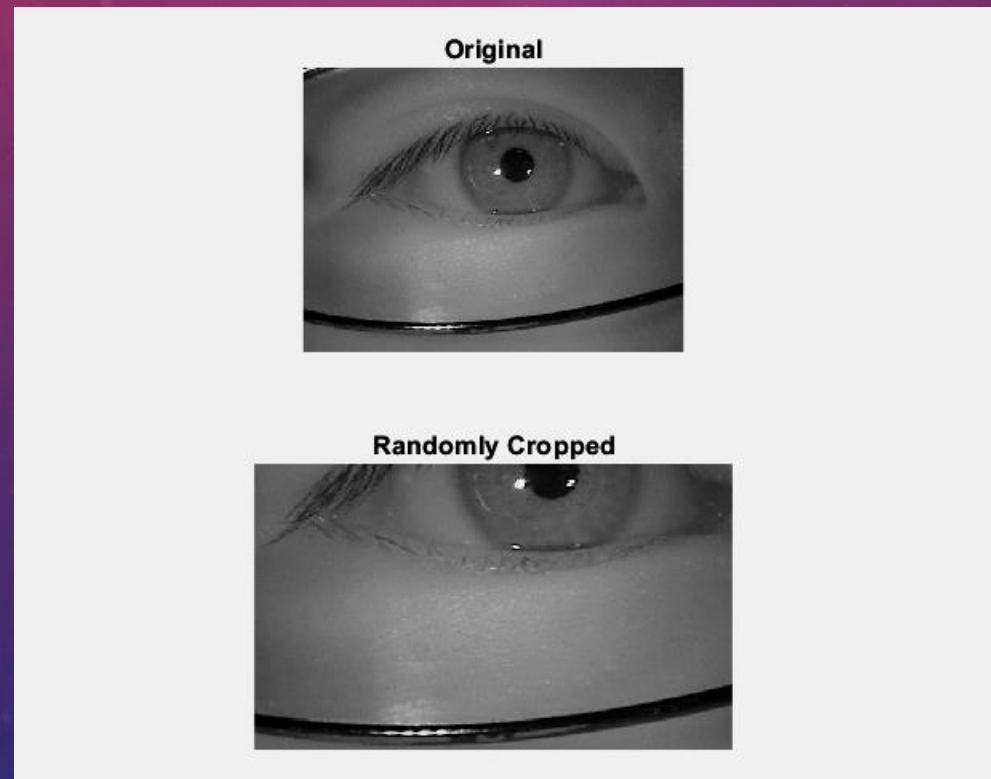
# WRONGLY ZOOMED IN IRIS PHOTOS

Often times the capturer may zoom in wrongly without keeping iris in center or taking out some regions of interest. This type of mistakes are augmented into the dataset with FiveCrop function



# WRONGLY ZOOMED IN IRIS PHOTOS

Sometimes the capturer maybe very careless and take photos zooming on random locations. With the help of RandomCrop function, we can feed these sort of augmented data into our training set.



# BLURRY PICTURES

Sometimes the capturer can take blurry pictures due to residual motion in hands

**Original**



**Blurred from motion**





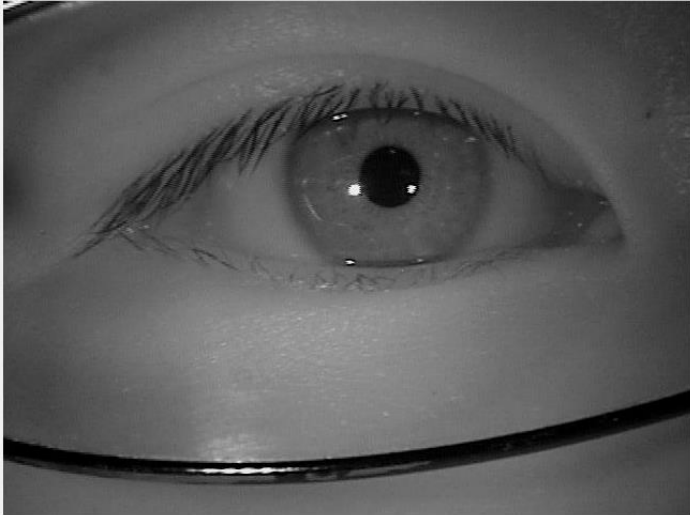
# GLARED PHOTOS OF IRIS

With the wrong lighting, it is very common for the photos of the iris to have some glares because of the cornea or spectacles (if subject wears any). The ColorJitter function emulates such defect for the augmented dataset

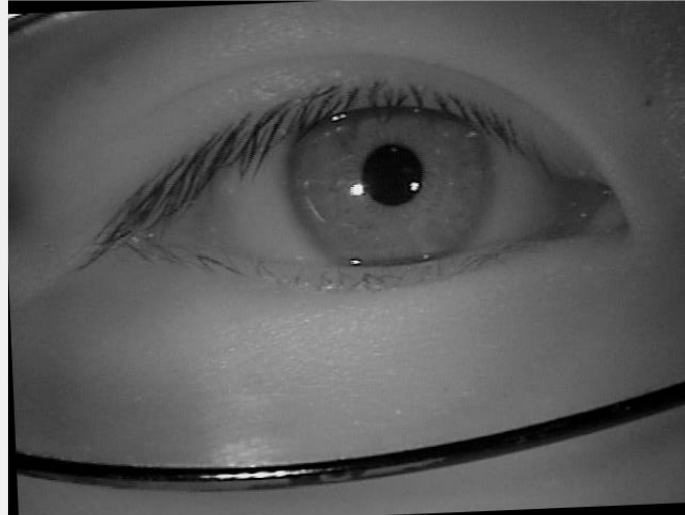


# IMPROPERLY ALIGNED PHOTOS

Original



Improperly Aligned



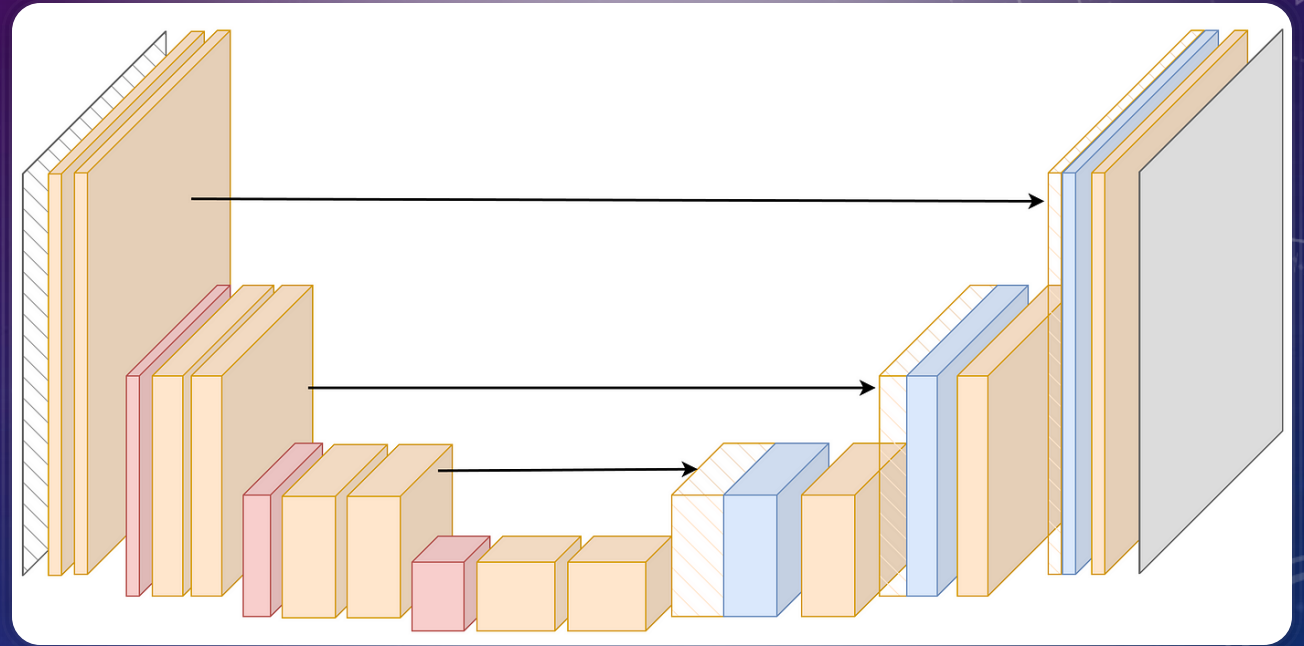
Even with utmost caution, it is not always possible to take photos with 100% accurate alignment. And it is even more difficult to notice these errors for slight rotation errors. RandomRotate function emulates these errors for training dataset



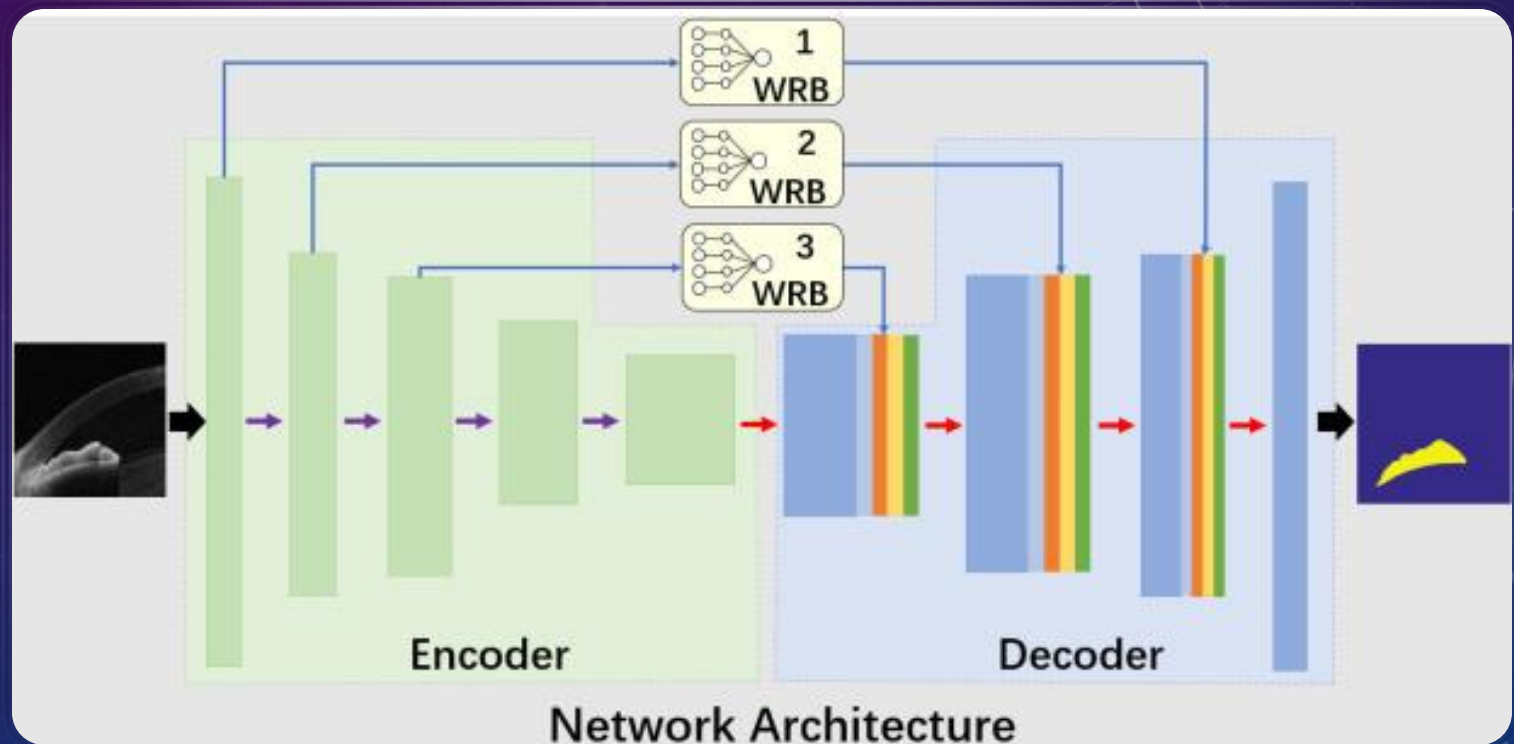
# WAVE REFINEMENT BLOCK



# U-NET ARCHITECTURE



# U-NET ARCHITECTURE WITH WRB



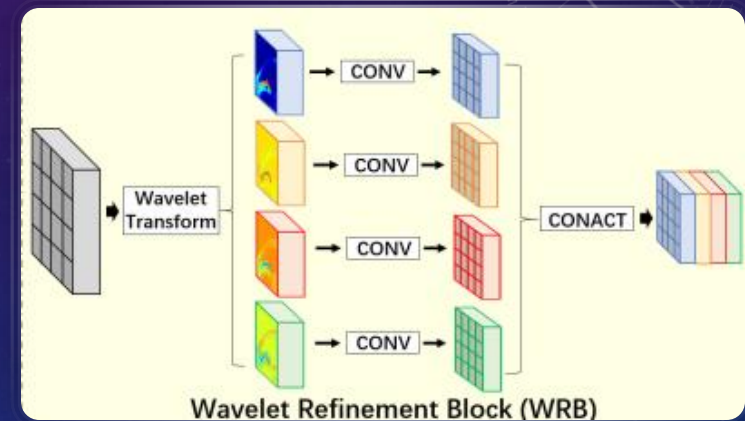


# REASONS BEHIND USING WRB

- Usual architecture imports massive quantities of irrelevant information into the decoder, which disturbs and weakens the learning ability of networks.
- To reduce the amount of redundant information, while preserving local details for the decoder

# CHARACTERISTICS OF WRB

- a 2D Discrete Wavelet Transform (DWT) with four convolutional filters - low-pass filter  $f_{LL}$ , and high-pass filters  $f_{LH}$ ,  $f_{HL}$ , and  $f_{HH}$  are performed to decompose  $X$  into four subband features,  $Y_{LL}$ ,  $Y_{LH}$ ,  $Y_{HL}$ , and  $Y_{HH}$ .
- It is worth noting that the low frequency band  $Y_{LL}$  stores local averages of the input data: correspondingly, the high frequency bands, namely  $Y_{LH}$ ,  $Y_{HL}$ , and  $Y_{HH}$ , encode details that are significant in recovering boundaries.
- each band is half-resolution of the input.



# ALGORITHM FOR WRB

- Taking the Haar wavelet as an example, the four filters are defined as:

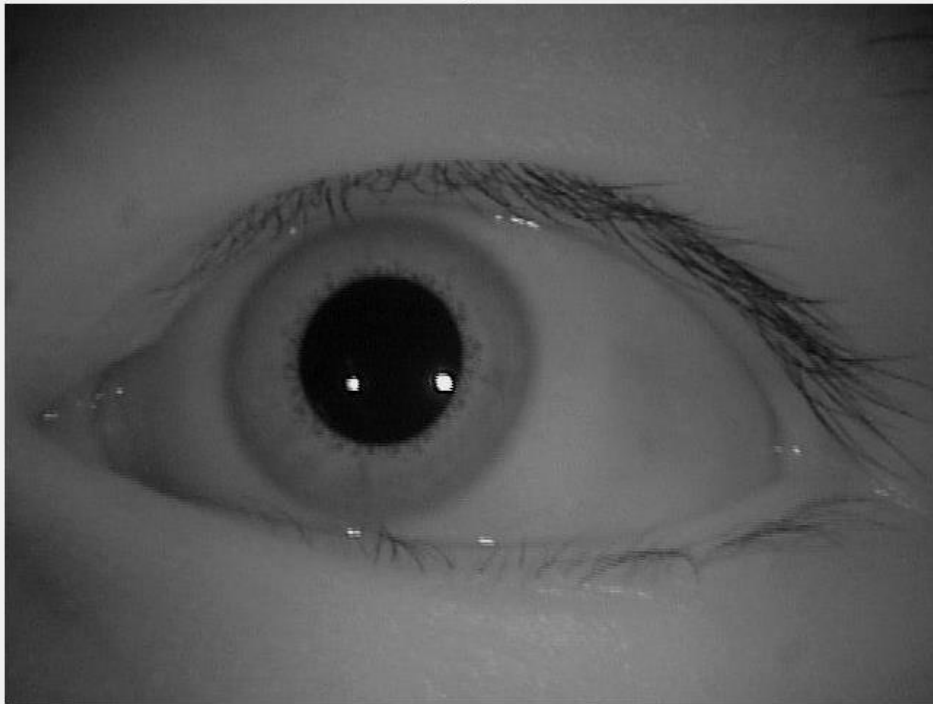
$$f_{LL} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}, f_{LH} = \begin{bmatrix} -1 & -1 \\ 1 & 1 \end{bmatrix}, f_{HL} = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix}, f_{HH} = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}.$$

- all the convolutions above are performed with stride 2, yielding a subsampling of factor 2 along each spatial dimension.



# RESULTS OF WRB

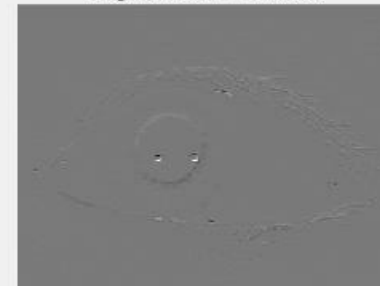
**Original**



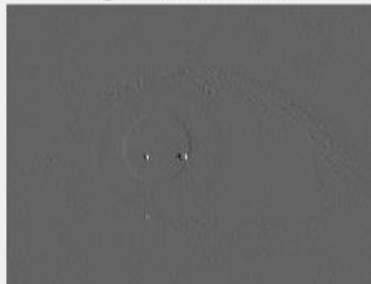
**Lowpass filtered**



**Highpass-1 filtered**



**Highpass-2 filtered**

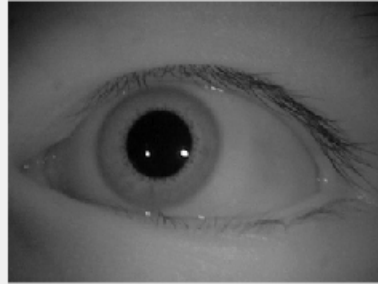


**Highpass-3 filtered**



# RESULTS OF WRB

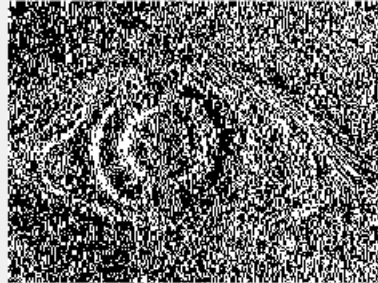
**Lowpass filtered**



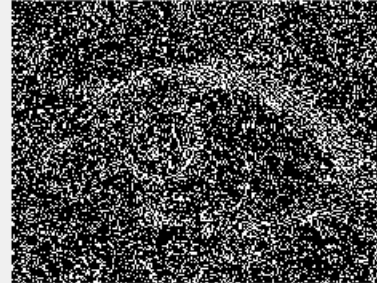
**Highpass-1 filtered**



**Highpass-2 filtered**



**Highpass-3 filtered**

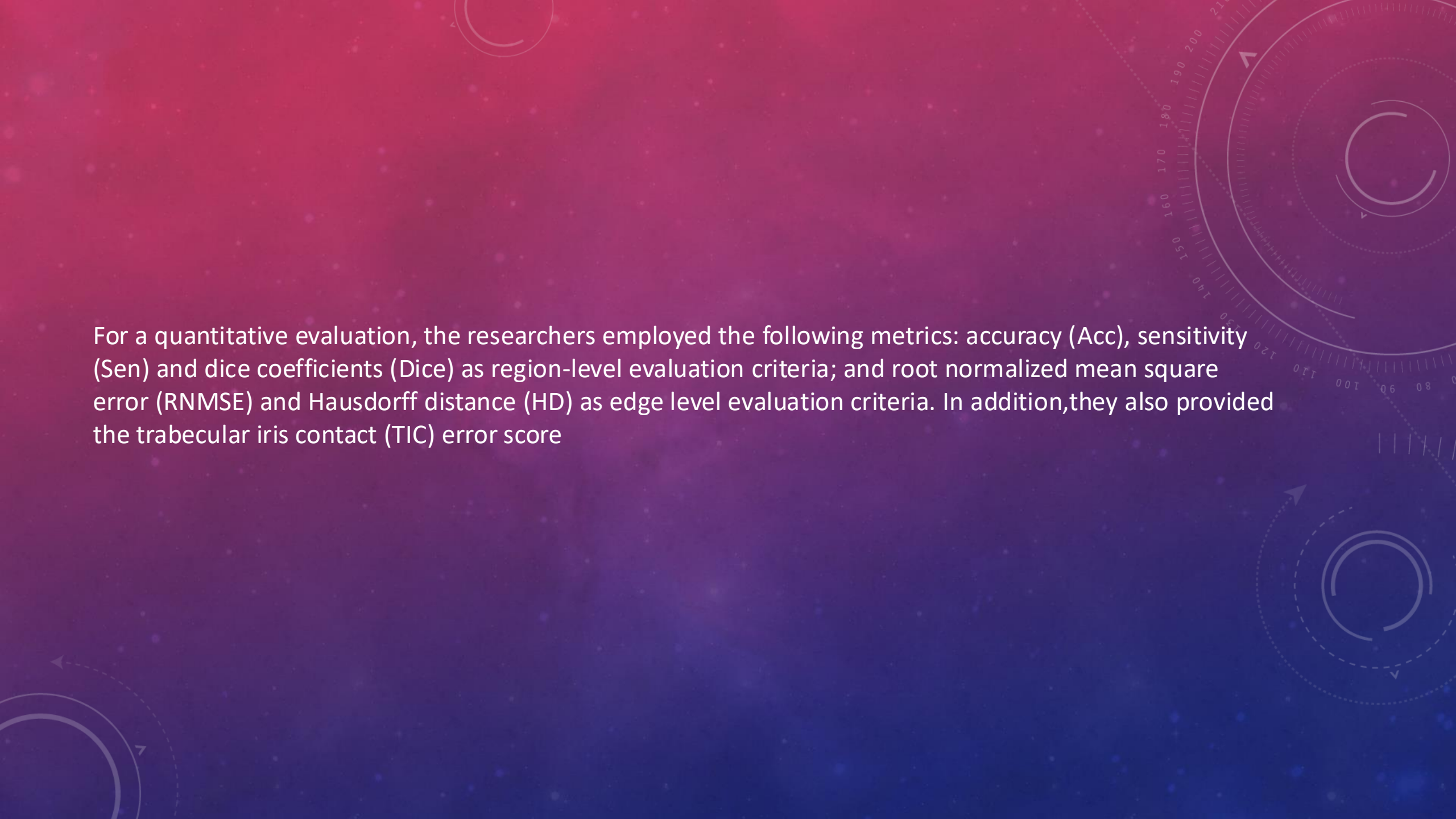




# EVALUATION METRICS







For a quantitative evaluation, the researchers employed the following metrics: accuracy (Acc), sensitivity (Sen) and dice coefficients (Dice) as region-level evaluation criteria; and root normalized mean square error (RNMSE) and Hausdorff distance (HD) as edge level evaluation criteria. In addition, they also provided the trabecular iris contact (TIC) error score

# IMAGE SEGMENTATION EVALUATION SCORES FUNCTION

```
function [Accuracy, Sensitivity, Fmeasure, Precision, Dice, Jaccard,
Specitivity, AUC] = ImageSegmentationEvaluationScores(A, B)
    % A and B need to be binary images
    % A is the ground truth, B is the segmented result.
    % If A, B are binary images, but uint8 (0, 255),
    % Need to convert to logical images.
    if(isa(A,'logical'))
        X = A;
    else
        X = imbinarize(A);
    end
    if(isa(B,'logical'))
        Y = B;
    else
        Y = imbinarize(B);
    end

    % Evaluate TP, TN, FP, FN
    sumindex = X + Y;
    TP = length(find(sumindex == 2));
    TN = length(find(sumindex == 0));
    substractindex = X - Y;
    FP = length(find(substractindex == -1));
    FN = length(find(substractindex == 1));
    Accuracy = (TP+TN)/(FN+FP+TP+TN);
    Sensitivity = TP/(TP+FN);
    Precision = TP/(TP+FP);
    Fmeasure = 2*TP/(2*TP+FP+FN);

    Dice = 2*TP/(2*TP+FP+FN);
    Jaccard = Dice/(2-Dice);
    Specitivity = TN/(TN+FP);
    [tpr,fpr]=roc(X,Y);
    AUC=trapz(tpr,fpr);

end
```

# HAUSDORFF DISTANCE (HD) FUNCTION

```
function [dH] = hausdorff( A, B)
if(size(A,2) ~= size(B,2))
    fprintf( 'WARNING: dimensionality must be the same\n' );
    dH = [];
    return;
end
dH = max(compute_dist(A, B), compute_dist(B, A))
```

```
%% Compute distance
function [dist] = compute_dist(A, B)
m = size(A, 1);
n = size(B, 1);
dim = size(A, 2);
for k = 1:m
    C = ones(n, 1) * A(k, :);
    D = (C-B) .* (C-B);
    D = sqrt(D * ones(dim,1));
    dist(k) = min(D);
end
dist = max(dist);
```





THANK YOU!



# APPENDIX



# FIVE CROP FUNCTION

```
function [I1 I2 I3 I4 Ic]= FiveCrop(I, crop_size)
[h w]=size(I);

hc=crop_size(1);
wc=crop_size(2);

I1=I(1:hc,1:wc);
x_lim=h-hc+1;
y_lim=w-wc+1;
I2=I(x_lim:h,y_lim:w);
I3=I(1:hc,y_lim:w);
I4=I(x_lim:h,1:wc);
center_x_1=round(h/2)-round(hc/2)+1;
center_x_2=round(h/2)+round(hc/2);
center_y_1=round(w/2)-round(wc/2)+1;
center_y_2=round(w/2)+round(wc/2);
Ic=I(center_x_1:center_x_2,center_y_1:center_y_2);
end
```



# RANDOMCROP FUNCTION

```
function [I_rc] = RandomCrop(I, crop_size)
[h w]=size(I);

hc=crop_size(1);
wc=crop_size(2);
x_start=randi((h-hc));
x_end=x_start+hc-1;
y_start=randi((w-wc));
y_end=y_start+wc-1;
I_rc=I(x_start:x_end,y_start:y_end);
end
```

# CENTERCROP FUNCTION

```
function Ic= CenterCrop(I,crop_size)
[h w]=size(I);

hc=crop_size(1);
wc=crop_size(2);
center_x_1=round(h/2)-round(hc/2)+1;
center_x_2=round(h/2)+round(hc/2);
center_y_1=round(w/2)-round(wc/2)+1;
center_y_2=round(w/2)+round(wc/2);
Ic=I(center_x_1:center_x_2,center_y_1:center_y_2);

end
```

# COLORJITTER FUNCTION

```
function [I_jittered] = ColorJitter(I,brightness,contrast,hue,saturation)
brightness=100+brightness*(200-100);
I1=I+brightness;

I2=imadjust(I1,[0,1],[contrast,1]);

if length(size(I))==3
    hsvI=rgb2hsv(I2);
    hsvI(:,:,1)=hsvI(:,:,1)*hue;
    I3=hsv2rgb(hsvI);

end

if length(size(I))==3
    hsvI=rgb2hsv(I3);
    hsvI(:,:,2)=hsvI(:,:,2)*saturation;
    I4=hsv2rgb(hsvI);

end

if length(size(I))==3
    I_changed=I2;
else
    I_changed=I2;
end
[h w]=size(I);
crop_size=[200 100];
hc=crop_size(1);
wc=crop_size(2);
x_start=randi((h-hc));
x_end=x_start+hc-1;
y_start=randi((w-wc));
y_end=y_start+wc-1;
I_big=I;
I_big(x_start:x_end,y_start:y_end)=I_changed(x_start:x_end,y_start:y_end);
I_jittered=I_big;
end
```



## BLURRYMOTION FUNCTION

```
function I_blur=BlurryMotion(I)
h = fspecial('motion', 25, 20);
I_blur = imfilter(I, h);
end
```

# RANDOMROTATE FUNCTION

```
function Ir = RandomRotate(I)
rotation = -5 + (5+5)*rand(1,1)
Ir=imrotate(I,rotation,'bilinear','crop');
end
```

# WRB (WAVE REFINEMENT BLOCK) FUNCTION

```
function yWRB=WRB(I)
fLL=[1 1;1 1];

yLL_s1=conv2(I,fLL,'valid');
yLL=yLL_s1(1:2:end,1:2:end);

fLH=[-1 -1;1 1];

yLH_s1=conv2(I,fLH,'valid');
yLH=yLH_s1(1:2:end,1:2:end);

fHL=[-1 1;-1 1];

yHL_s1=conv2(I,fHL,'valid');
yHL=yHL_s1(1:2:end,1:2:end);

fHH=[1 -1;-1 1];

yHH_s1=conv2(I,fHH,'valid');
yHH=yHH_s1(1:2:end,1:2:end);

dim=size(yHH);
yWRB=zeros(dim(1),dim(2),4);
yWRB(:,:,1)=yLL;
yWRB(:,:,2)=yLH;
yWRB(:,:,3)=yHL;
yWRB(:,:,4)=yHH;
end
```