# Oriented Edge Forests for Boundary Detection

**Project Evaluation - Final** **- TEAMDASH**

# INTRODUCTION

**BOUNDARY ESTIMATION:**

- Important first step for segmentation and detection of objects.
- Provide information about the shape and identity of objects.

**PREVIOUS WORKS:**

- Focused on detecting brightness edges, estimating their orientation and analyzing the theoretical limits of detection in the presence of image noise.
- Recently focus has turned to methods that learn appropriate feature representations from training data rather than relying on hand-designed texture and brightness contrast measure.

# PROPOSED METHOD

- Apply the concept of randomized decision forests to the simple task of accurately detecting straight-line boundaries at different candidate orientations and positions within a small image patch.

- To improve the performance,calibrate and average the results across a small number of scales, along with local sharpening of edge predictions.

# CLUSTERING EDGES

- Method for partitioning the space of oriented edge patterns within a patch.
- This leads to a simple, discrete labeling over local edge structures.

**Background(No Boundary Pixel):**

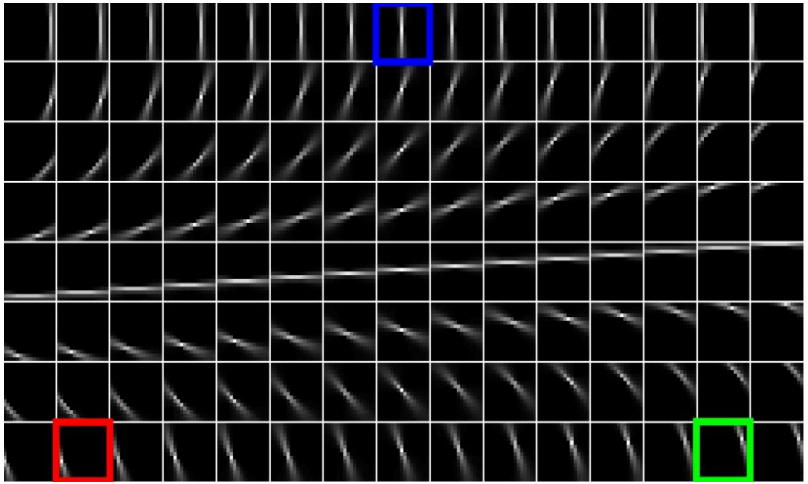
- A patch is considered background if its edge is more than p/2 pixels away from the center

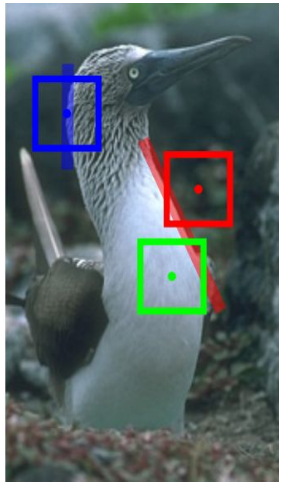
**Boundary Pixel:**

- Distinguished according to the distance d and orientation θ of the edge pixel closest to the patch center.

# CLUSTERING EDGES

Bin the space of distances d and angles θ into n and m bins, respectively.This discrete label space allows for easy application of a variety of supervised learning algorithms



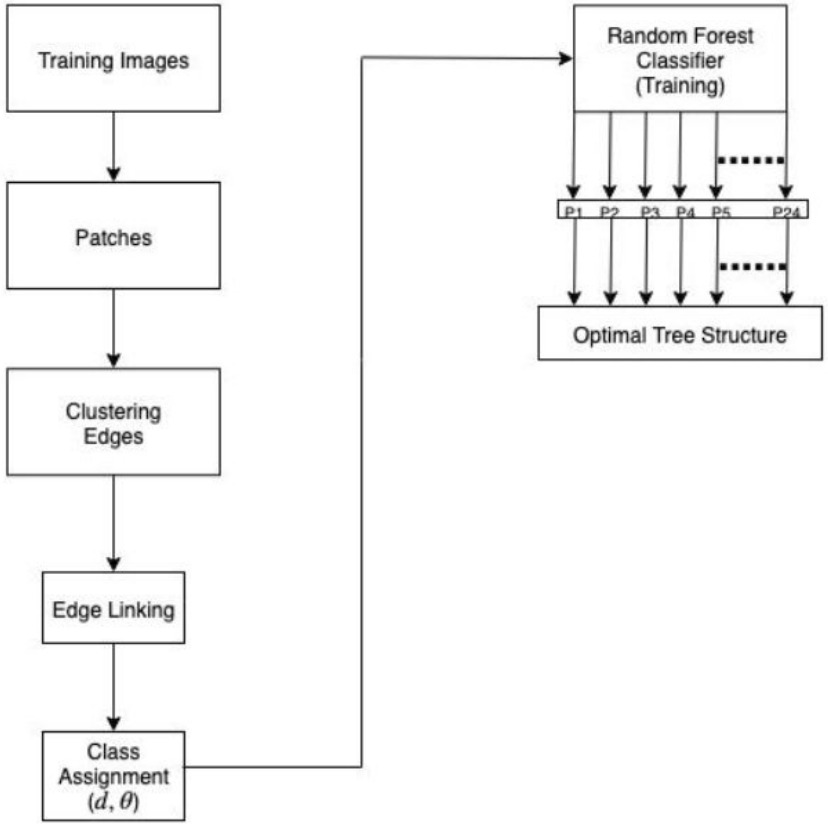


**Parameter Space**

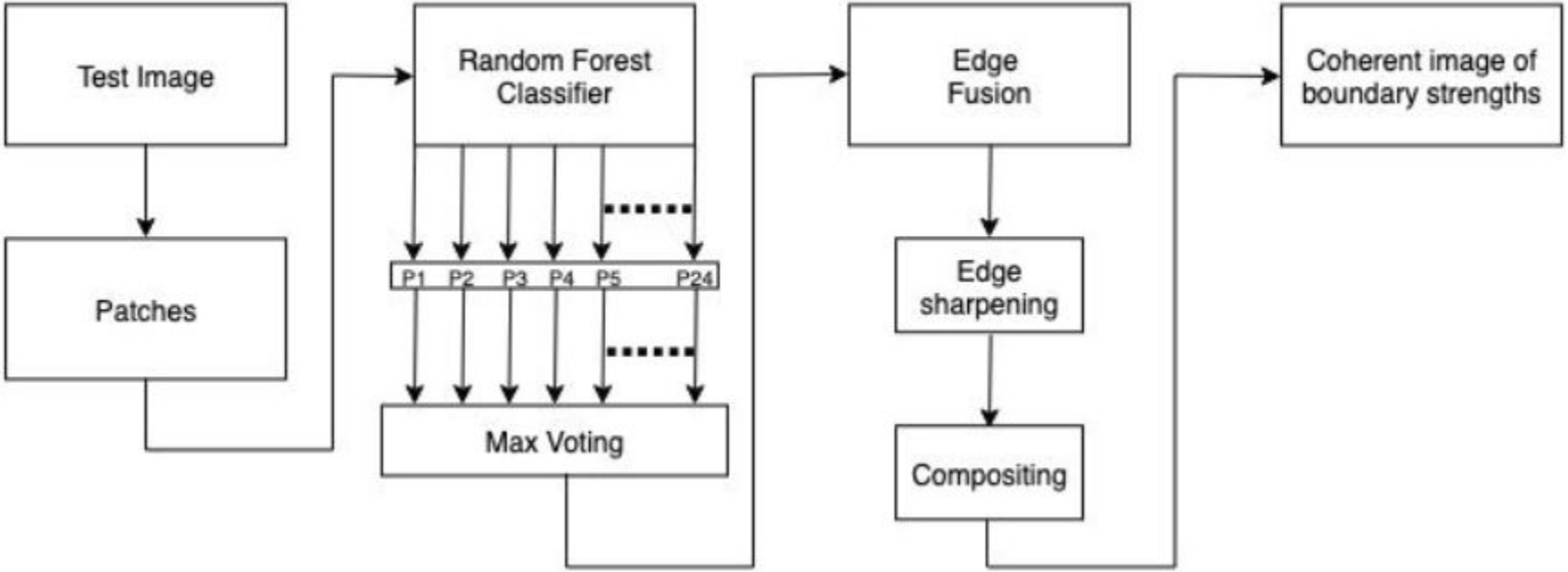
# ORIENTED EDGE FOREST

- We are treating framework as a k-way classification problem where k=possible edge orientations w.r.t offset from the center.
- Binary splits at the tree nodes based on pixel read from the RGB channel or the difference between 2 pixels from the same channel.
- There are 2 ways of ensembling:
  - Averaging ( Memory and Time intensive but better accuracy)
  - Voting( Faster but the predicted score vector is sparse)

# TRAINING PHASE

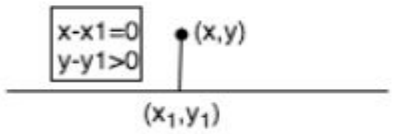


# TESTING PHASE

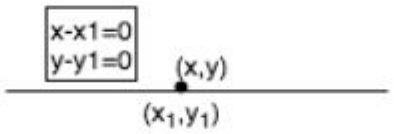


# IMPLEMENTATION DETAILS

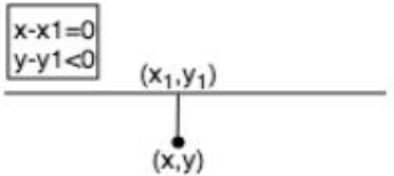
# SIGN CONVENTIONS




d=Positive

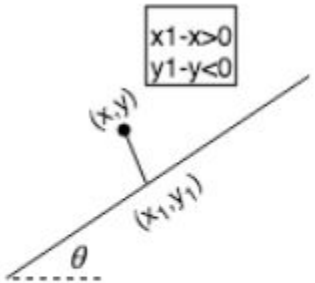



d=0

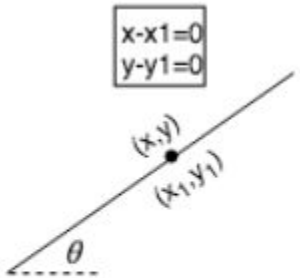



d=Negative

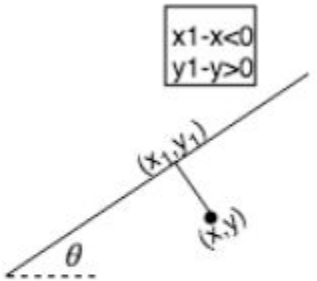
$\theta = 0$




d=Positive

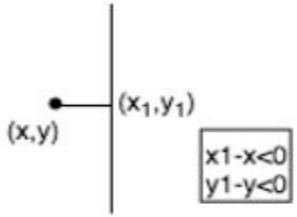



d=0

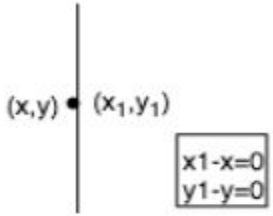



d=Negative

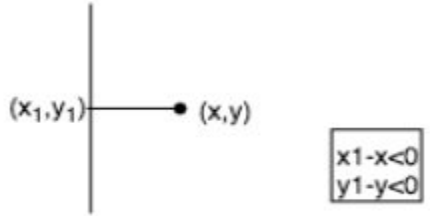
$0 < \theta < 90$




d=Positive

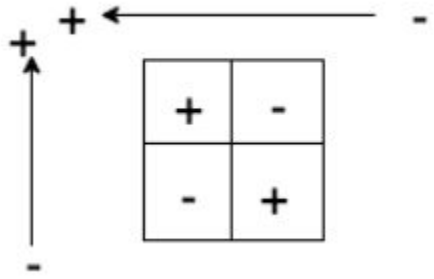



d=0

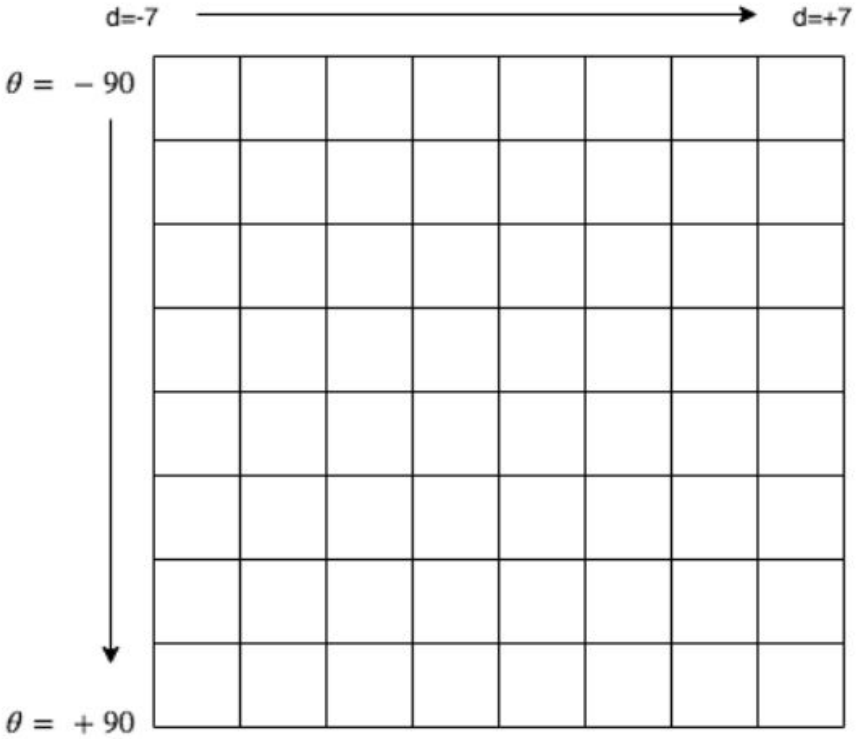



d=Negative

$$\theta = 90$$



# Class Visualization




- The following represents the d,θ grid.
- The distance of centre of every patch centre with respect to the nearest edge pixel is binned into one of the cells of ‘d’ across the grids.
- The orientation of the edge contained by that patch is binned to one of the ‘θ’ grids, along the vertical direction.
- Thus every element of this grid represents a class in the form of d-θ pair.

# Dataset Generation

- The BSDS500 dataset has an example color image and its corresponding ground truth boundary image.
- From the ground truth boundary image, we found out the d,θ pair for every patch of **16X16** of the ground truth image.
- Then we assigned the corresponding patch of the color image to this **d,θ** pair, which represents the class of that patch.
- Taking memory into consideration we saved the resultant patch as a **mat** file rather than an **image** file.
- This was performed for all the images of the dataset.

# ENSEMBLE METHODS

**Averaging:**

$$\mathbf{w}(k|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} p_t(k|\mathbf{x}), \qquad k = 1, ..., K$$

where x is the image patch, k is the predicted output label, t is the decision tree, T is the total number of decision trees, w is the predicted score vector.

**Voting:**

$$\mathbf{w}(k|\mathbf{x}) = \frac{1}{T}\sum_{t=1}^{T} \mathbf{1}_{[k=\arg\max_k p_t(k|\mathbf{x})]}$$

where **1** is the indicator function

# EDGE FUSION

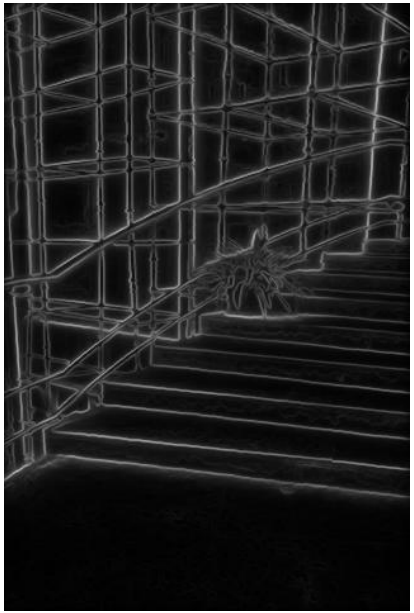
This section comprises of:

- Edge sharpening : Using local segmentation
- Compositing
- Combining multiple scales : Multiple scale for large and small scale edge structures

# EXPECTED OUTPUTS

# OBTAINED OUTPUTS - 1

Input Image



OEF

Sobel

# OBTAINED OUTPUTS - 2

Input Image

OEF

Sobel

# OBTAINED OUTPUTS - 3

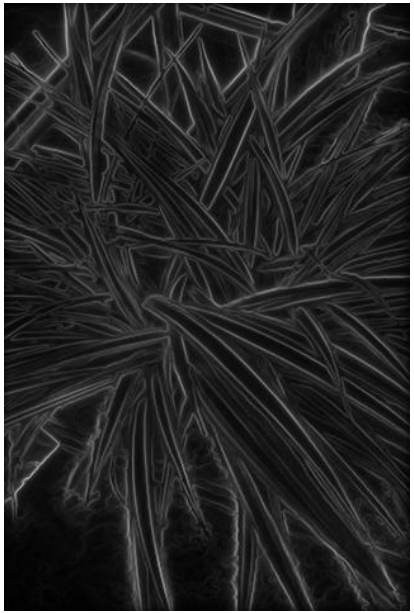
Input Image

OEF

Sobel

# OBTAINED OUTPUTS - 4

Input Image

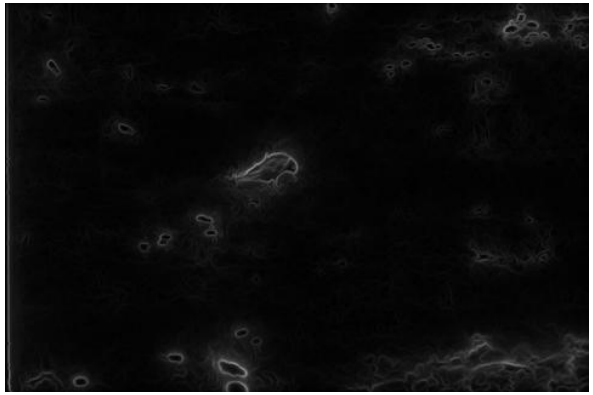


OEF

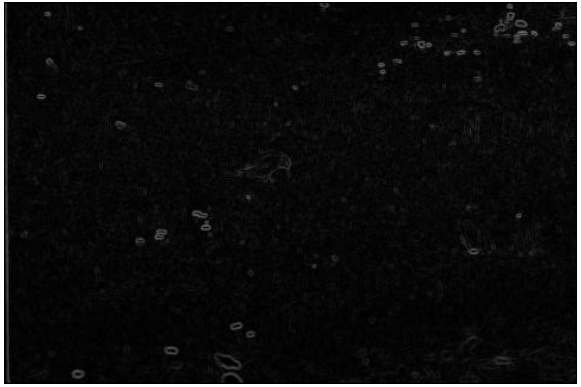
Sobel

# SPECIAL CASES - 1

Input Image



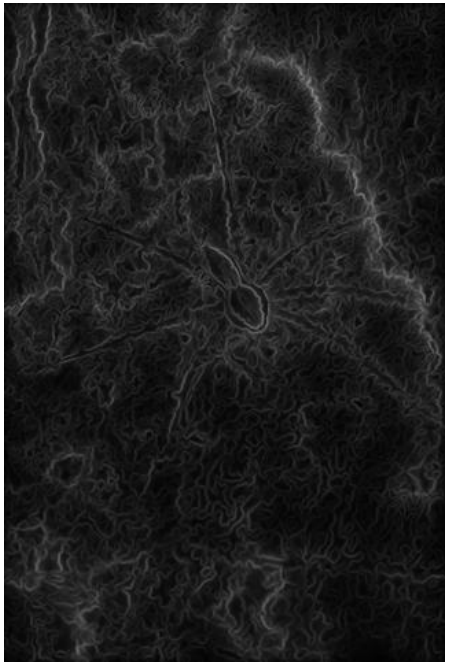
OEF



Sobel

# SPECIAL CASES - 2

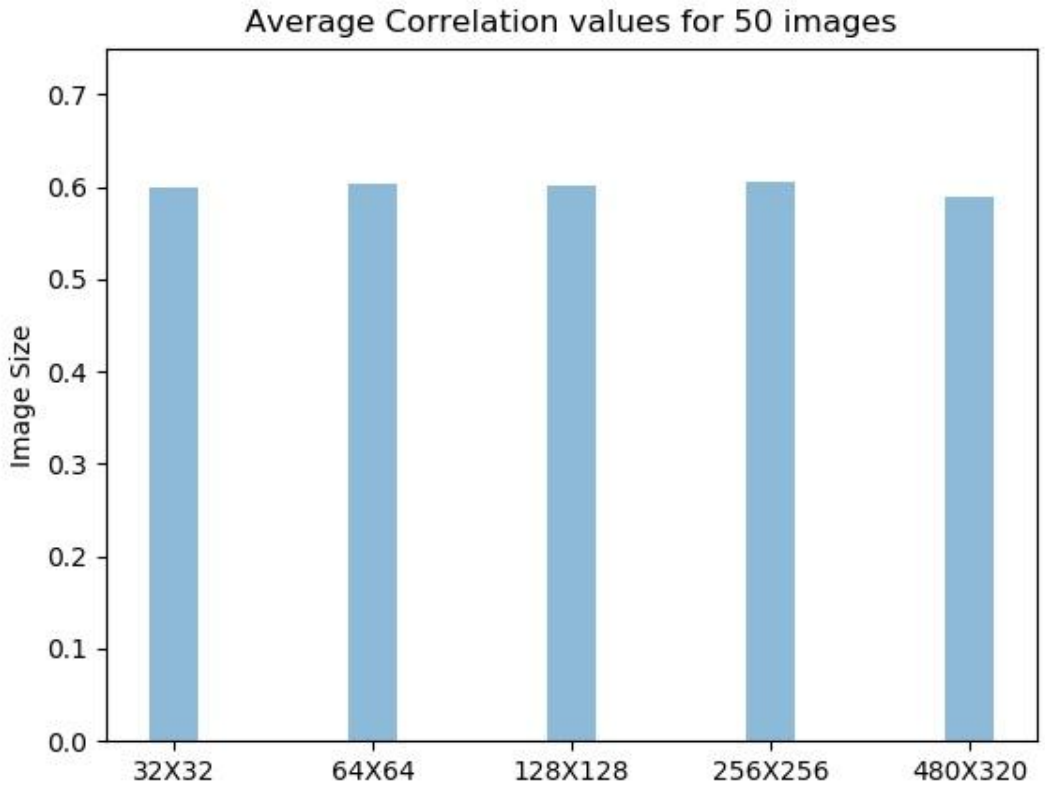
Input Image



OEF

Sobel

# CORRELATION WITH SOBEL



# GitHub Link

https://github.com/deepakksingh/CV-Project-2019

# Thank you