Oriented Edge Forests for Boundary Detection

- 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:

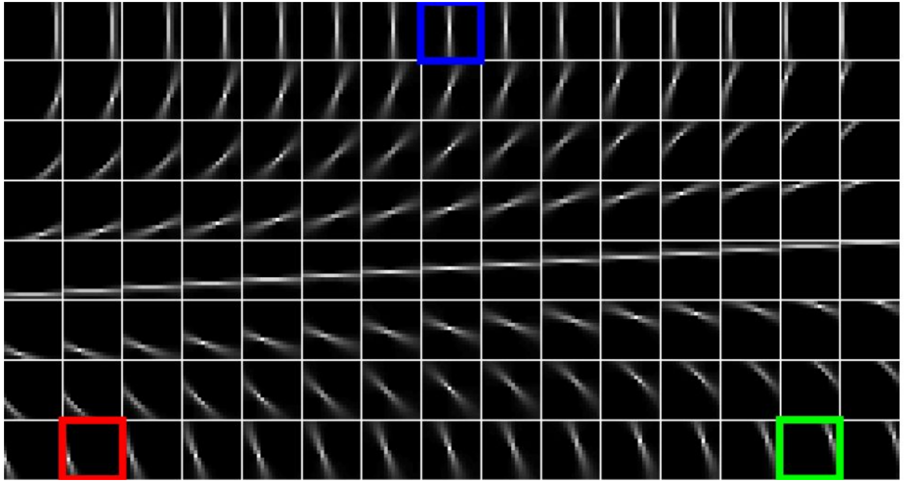
- Distinguished according to the distance d and orientation θ of the edge pixel closest to the patch center.

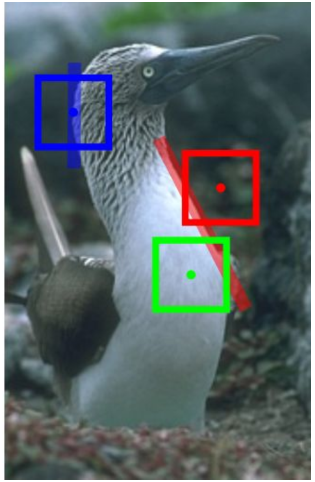
CLUSTERING EDGES

- To accomplish this, we first link all edge pixels in a ground-truth boundary map into edge lists, breaking lists into sublists where junctions occur. We then measure the angle at q by fitting a polynomial to the points around q that are in the same list

- 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)

CURRENT PROGRESS

- Created the dataset from BSDS500 Segmentation dataset.
- Implemented primary functionality of computing the parameters.
- Created the hierarchy structure for labels associated with the parameter space in the dataset.
- Computed parameters for the given patch of image.
- Appropriate sign-conventions were adopted for the nearest edge pixel from the center of the patch.

FURTHER PLANS

- Train a random forest from the curated dataset.
- Applying calibration techniques for edge fusion of various edges.
- Try Edge Sharpening, Compositing and Combining multiple scale images.

ENSEMBLE METHODS

Averaging:

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

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

GitHub Link

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

Thank you