Project Report

Measuring Objectness

Prathyusha Akundi - 2018701014
Dasika Anoop Karnik - 2018701004

Contents

1)Introduction
- a) Problem Statement
- b) Motivation
- c) Overview
- d) Concepts Used in the Paper

2) Approaches
- a)Edge Density Cue
- b)Multi Scale Saliency Cue
- c) Color Contrast Cue
- d) SuperPixel Straddling Cue
- e) Bayes Integration

3)Workflow
- Flow Chart
- a) Edge Density
- b) MultiScale Saliency Cue
- c) Color Contrast Cue
- d) SuperPixel Straddling Cue

4)Results
- a)Analysis
- b)Future Direction

5) Task Assignment
- a)Anoop Karnik
- b)Prathyusha Akundi

6) Conclusion

1) Introduction

a) Problem Statement

The problem is to find the top windows containing the objects in VOC dataset and some similar datasets. We present a generic objectnesss measure, quantifying how likely it is for an image window to contain an object.

b) Motivation

Object Detection is a major research Area even now. At present object detection is done through Deep Learning but there still is a use to include traditional methods in object detection. While object detectors are specialized for one object class, such as cars or swans, in this paper we define and train a measure of objectness generic over classes. It quantifies how likely it is for an image window to cover an object of any class. Objects are standalone things with a well-defined boundary
and center, such as cows, cars, and telephones, as opposed to amorphous background stuff, such as sky, grass, and road.

c) Overview

In order to define the objectness measure, we argue that any object has at least one of three distinctive characteristics:
• a well-defined closed boundary in space.
• a different appearance from its surroundings.
• sometimes it is unique within the image and stands out as salient.

Many objects have several of these characteristics at the same time explore several image cues designed to measure these characteristics. Although objects can appear at a variety of locations and sizes within an image, some windows are more likely to cover objects than others, even without analyzing the pixel patterns inside them.

This report gives the following considerations:
(a) We design an objectness measure and explicitly train it to distinguish windows containing an object from background windows. This measure combines in a Bayesian framework several cues based on the above characteristics.
(b) On the task of detecting objects in the challenging PASCAL VOC 07 dataset , we demonstrate that the combined objectness measure performs better than any cue alone and performs better than other traditional object detection measures but not better than deep learning object detection.
Classes.
(c) We give an algorithm that employs objectness to greatly reduce the number of evaluated windows evaluated using NMS (Non Max Suppression) Sampling.

d) Concepts Used in the Paper

Some of the following Concepts were used in this project:

- **Saliency** – It measures the saliency of pixels as the degree of uniqueness of their neighborhood wrt the entire image or the surrounding area [36, 38]. Salient pixels form blobs that 'stand out' from the image. These works implement selective visual attention from a bottom-up perspective and are often inspired by studies of human eye movements.
- **Super Pixels -** Superpixel is a group of connected pixels with similar colors or gray levels. Superpixel segmentation is dividing an image into hundreds of non-overlapping superpixels. Instead of working with just pixels, we can superpixels to do image segmentation. There are two major advantages for using superpixels. We can compute features on more meaningful regions and we can reduce the input entities for the subsequnt algorithms. Superpixel segmentation have been applied to many computer vision tasks, such as sematic segmentation, visual tracking, image classification, and so on.
- **Color Contrast –** Contrast is the difference in luminance or colour that makes an object (or its representation in an image or display) distinguishable. In visual perception of the real world, contrast is determined by the difference in the color and brightness of the object and other objects within the same field of view.
- **Non Max Suppression Sampling -** NMS is used to make sure that in object detection, a particular object is identified only once. Consider a 100X100 image with a 9X9 grid and there is a car that we want to detect. If this car lies in multiple cells of grid, NMS ensures we identify the optimal cell among all candidates where this car belongs.
- **Naive Bayes -** Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence. The Naive Bayesian classifier is based on Bayes' theorem with the independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

2) Approaches

a) Edge Density Cue

ED measures the density of edges near the window borders. The inner ring Inn(w, θ_ed) of a window w is obtained by shrinking it by a factor θ_ed in all directions.

$$\frac{|\mathrm{Inn}(w, \theta_{ED})|}{|w|} = \frac{1}{\theta_{ED}^2}$$

The binary edgemap I ED (p) ∈ {0, 1} is obtained using the Canny detector, and Len(·) measures the perimeter of theinner ring. The expected number of boundary edges grows proportionally to the perimeter, not the area, because edges have constant thickness of 1 pixel. This normalization avoids a bias towards very small windows that would otherwise arise.

$$\mathrm{ED}(w, \theta_{ED}) = \frac{\sum_{p \in \mathrm{Inn}(w, \theta_{ED})} I_{ED}(p)}{\mathrm{Len}(\mathrm{Inn}(w, \theta_{ED}))}$$

We learn θ_ed in a Bayesian Framework. For every image I in T we generate 100000 random windows uniformly distributed over the entire image. Windows covering an annotated object are considered positive examples (W obj), the others negative (W bg).

$$\theta^* = \arg\max_{\theta} \prod_{w \in \mathcal{W}^{\mathrm{obj}}} p_\theta(\mathrm{obj}|\mathrm{CC}(w, \theta)) = \tag{10}$$
$$= \arg\max_{\theta} \prod_{w \in \mathcal{W}^{\mathrm{obj}}} \frac{p_\theta(\mathrm{CC}(w, \theta)|\mathrm{obj}) \cdot p(\mathrm{obj})}{\sum_{c \in \{\mathrm{obj}, \mathrm{bg}\}} p_\theta(\mathrm{CC}(w, \theta)|c) \cdot p(c)}$$

For any value of θ we can build the likelihoods for the positive pθ(ED(w, θ)|obj) and negative classes pθ(ED(w, θ)|bg), as histograms over the positive/negative training windows. We now find the optimal θ ∗ by maximizing the posterior probability that object windows are classified as positives. The priors are set by relative frequency:
p(obj) = |W obj |/(|W obj | + |W bg |), p(bg) = 1 − p(obj). The advantage of this procedure is that the distribution of training samples is close to what the method will see when
a new test image is presented, as opposed to just using the positive ground-truth samples and a few negative samples. Moreover, it is likely to generalize well, as it is trained from many variants of the annotated windows in W obj. The learned parameters θ_ed define the optimal outerring Surr(w, θ_ed) and inner ring Inn(w, θ_ed).

b) Multi Saliency Cue

A global saliency measure based on the spectral residue of the FFT, which favours regions with an unique appearance within the entire image f. The saliency map I of an image f is obtained at each pixel p as :

$$I(p) = g(p) * \mathcal{F}^{-1}\Big[\exp\big(\mathcal{R}(f) + \mathcal{P}(f)\big)\Big]^2$$

where F is the FFT, R(f) and P(f) are the spectral residual and the phase spectrum of the image f , and g is a Gaussian filter used for smoothing the output. This measure prefers objects at a certain scale, we extend it to multiple scales. We process the color channels independently as separate images. For each scale s, we obtain a saliency map . Based on this, we define the saliency of a window w at scale s as

$$\mathrm{MS}(w, \theta^s_{\mathrm{MS}}) = \sum_{\{p \in w | I^s_{\mathrm{MS}}(p) \geq \theta^s_{\mathrm{MS}}\}} I^s_{\mathrm{MS}}(p) \times \frac{|\{p \in w | I^s_{\mathrm{MS}}(p) \geq \theta^s_{\mathrm{MS}}\}|}{|w|}$$

where the scale-specific thresholds θ MS are parameters to be learned (sec. 3), and | · | indicates the number of pixels. Saliency is higher for windows with higher density of salient pixels (second factor), with a bias towards larger windows (first factor). Density alone would score highest windows comprising just a few very salient pixels. Instead, our measure is designed to score highest windows around entire blobs of salient pixels, which correspond better to whole object. This MS cue measures the uniqueness characteristic of objects.

We learn each threshold θ MS independently, by optimizing the localization accuracy of the training object windows O at each scale s.

$$\theta^{s*}_{\mathrm{MS}} = \arg\max_{\theta^s_{\mathrm{MS}}} \sum_{o \in \mathcal{O}} \max_{w \in \mathcal{W}^s_{\max}} \frac{|w \bigcap o|}{|w \bigcup o|}$$

For every training image I and scale s, we compute the saliency map and the MS score of all windows. We run non-maximum suppression on this 4D score space using the efficient technique resulting in a set of local maxima windows W max. We find the optimal θ MS by maximizing i.e. we seek for the threshold θ MS that leads the local maxima of MS in the images to most accurately cover the annotated. objects O. Notice how this procedure is discriminative. Maximizing implicitly entails minimizing the score of windows not covering any annotated object.

c) Color Contrast Cue

It is a general observation that objects tend to have different appearance in terms of colour, texture with respect to their background. In this measure, we explore this generic property regarding colour. Color contrast is a measure of dissimilarity of a window W to its surrounding window S.

We obtain the surrounding window S by enlarging window W by the factor of theta_CC in all directions. It can be proved that the ratio between number of pixels in surrounding window S excluding W to the number of pixels in W is equal to theta_CC^2 -1. Mathematically, if |.| denotes number of pixels in a window then:

$$\frac{|\mathrm{Surr}(w, \theta_{CC})|}{|w|} = \theta_{CC}^2 - 1$$

To obtain the color contrast score, we first generate histograms for each of the windows that gives the intensity distributions. As stated earlier, since there will be a high contrast in color from object to background, we compute Chi-Square distance of these two histograms. If the distance is high, it means that there is lot of scope for that window to contain the object.

Our main task is to find the factor theta_CC that gives us the ideal surrounding window S such that the contrast between the window W and S is high. For this, we select a range of theta_CC values and generate 100000 uniform windows for every image in training dataset. For each of the window we compute the likelihood of positive and negative training windows. We select theta_CC which maximizes the probability that object windows are classified as positives by following:

$$\begin{aligned}\theta^* &= \arg\max_{\theta} \prod_{w \in \mathcal{W}^{\mathrm{obj}}} p_\theta(\mathrm{obj}|\mathrm{CC}(w,\theta)) = \\ &= \arg\max_{\theta} \prod_{w \in \mathcal{W}^{\mathrm{obj}}} \frac{p_\theta(\mathrm{CC}(w,\theta)|\mathrm{obj}) \cdot p(\mathrm{obj})}{\sum_{c \in \{\mathrm{obj},\mathrm{bg}\}} p_\theta(\mathrm{CC}(w,\theta)|c) \cdot p(c)}\end{aligned}$$

The likelihoods are obtained from histograms and p(obj) is the evidence which can be obtained from the annotated training dataset.

d) Superpixel Straddling Cue

A super pixel is an image patch which is better aligned with intensity edges that regular rectangular boxes. So, we can capture the closed boundary characteristic of objects by using superpixels as features. The most important property of superpixels is that they preserve object boundaries. Therefore, ideally all pixels in a superpixel belong to the same object and hence they won't straddle the boundary of object.
Based on this we can determine if a window contains the object or not. We say that a superpixel s straddles a window w if atleast one of its pixel lies outside of w. If window contains the object entirely, then most of the pixels in s will lie in window w. If window doesn't contain the object, then most of the pixels lie both inside and outside i.e. s straddles w. SS cue measures the degree by which s straddles w which is given by:

$$\mathrm{SS}(w, \theta_{SS}) = 1 - \sum_{s \in \mathcal{S}(\theta_{SS})} \frac{\min(|s \setminus w|, |s \bigcap w|)}{|w|}$$

Here, |s∩w| computes the area under the intersection of superpixel and w and |s \ w| computes the area of pixels belonging to s outside w. If object is inside w then |s \ w| value will be 0 and hence gives maximum score.

The set of superpixels S(theta_SS) is obtained by using a superpixel algorithm with the segmentation scale of theta_SS. Hence we must learn the optimal theta_SS that maximizes the probability window contains the object. We learn in the same way as we have done in CC by computing the likelihoods and retaining the theta that gave maximum probability.

e) Bayes Integration

Each cue mentioned above performs better in some areas and fails in certain cases. Since MS cue is designed to find blob-like things, it gives only a rough indication of whether object is present or not. CC gives better performance for images having high intensity variance but fails out completely in cases where images have low contrast. ED gives many false positives on images having multiple edges. Although SS gives better performance, it fails for small objects for which superpixels are fragile.

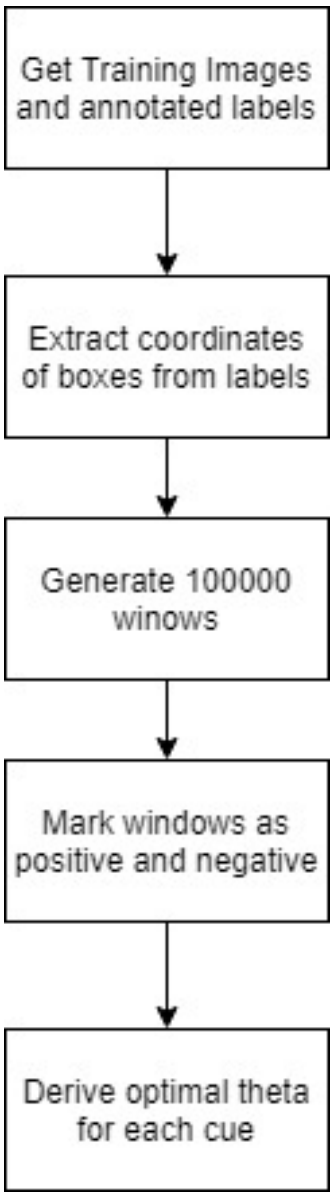
Thus, a single cue will not give best classification. Hence, we combine all cues and consider the windows for which they gave highest scores. To achieve this, first we sample 100000 windows for the given input image and compute scores from all the cues for each window. During our training, we saved the prior probability of object and the likelihoods of all the cues. We now use them to model the Bayesian classifier and get the score for each window generated:

$$p(\mathrm{obj}|\mathcal{C}) = \frac{p(\mathcal{C}|\mathrm{obj})p(\mathrm{obj})}{p(\mathcal{C})}$$
$$= \frac{p(\mathrm{obj})\prod_{\mathrm{cue}\in\mathcal{C}} p(\mathrm{cue}|\mathrm{obj})}{\sum_{c\in\{\mathrm{obj},\mathrm{bg}\}} p(\mathrm{c})\prod_{\mathrm{cue}\in\mathcal{C}} p(\mathrm{cue}|\mathrm{c})}$$

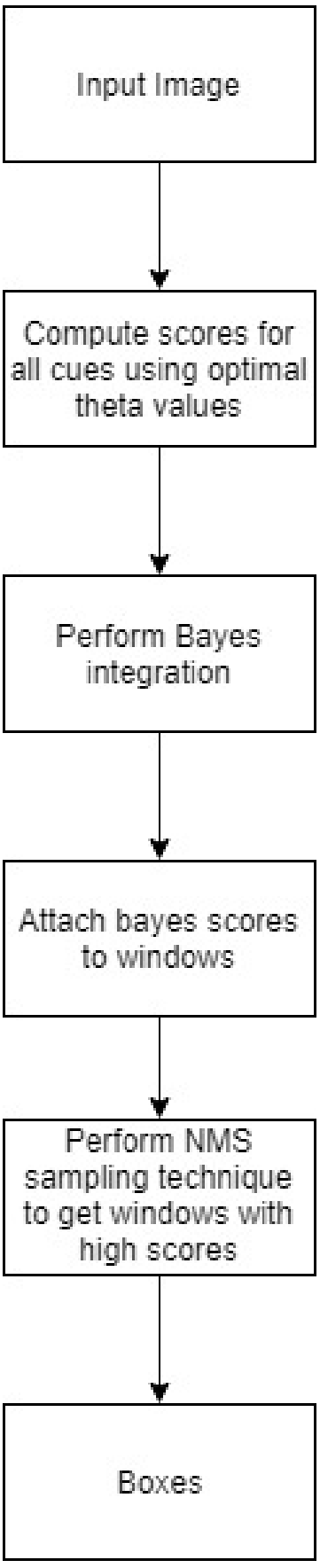
The posterior probability from above equation gives the objectiveness of the window. We then send all 10000 windows for NMS sampling which gives out the windows with highest scores and less than 20% overlapping.

3) Workflow

Parameter Training Flowchart



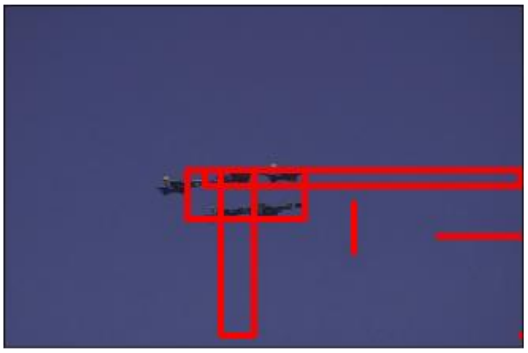
Computing Scores and getting Windows Flowchart



a) Edge Density Cue

- First we resize the image into 200x200.
- Convert the image into gray scale image.
- We then get a edge image by using canny edge detector.
- We then compute an integral image.
- We then generate dense windows ie based on the min height and width we compute all possible window sizes in the image.
- Based on theta value we compute inner windows for each of the windows.
- We then compute scores based on the outer and inner windows and used them to score the boxes.
- We then use NMS sampling to get top windows.of the image.
- We use naive bayes to train parameter theta for the edge density cue by using train dataset of around 1k images from VOC 2007.
- We then construct the top boxes.

Failure Case



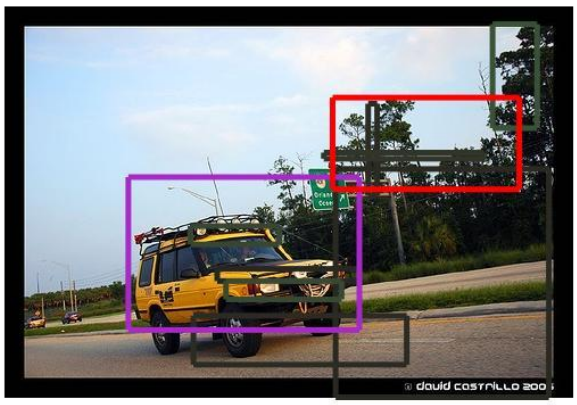
Failure Case

Edge Density Score alone is able to detect a single vechicle in the road but along with it many other windows are also visible and is completely unable to detect multiple aircrafts and is at best giving all the aircrafts in a single box.

b) Multi Saliency Cue

- We loop through different resized images(ie16x16 to 64x64) and in different channels(Red, Green, Blue).
- In each size image and channel, we get a saliency map which just shows the saliency in the image and a threshold map which shows all the pixel values in saliency above threshold to be 1 and others to be 0.
- We then compute integral images for both saliency and threshold map.
- Using the function sliding window compute score we compute the saliency score for each resized image and channels for all the possible boxes we used.
- As we have around 100k windows for the image we used NMS sampling to get top windows and to remove windows with more than 0.5 overlap.
- We use naive bayes to train parameter theta for the Multi Saliency cue by using train dataset of around 1k images from VOC 2007.
- We then construct the top boxes.

Below are the boxes we get using MS Score for two of the sample images:-



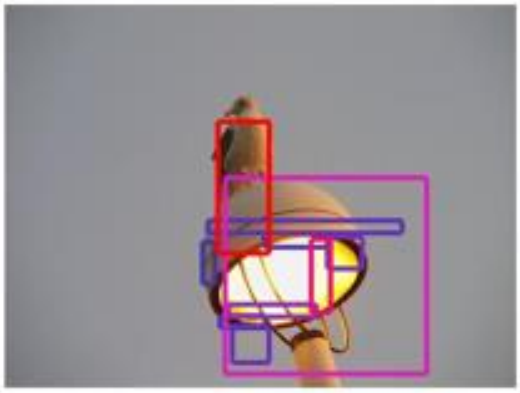
Success Case

Failure Case

Saliency Score alone is able to detect a single vehicle in the road but is unable to detect multiple aircrafts and is at best giving all the aircrafts in a single box.

c) Color Contrast Cue

- We first generate dense windows(all possible windows in image greater than min height and width).
- We then convert the image from RGB to LAB.
- We then quantize this image based on the quant parameter in CC.
- We then compute histogram of all the boxes and the corresponding histograms of the inner boxes.
- We compare the inner and the outer histogram to get the contrast Score.



Success case

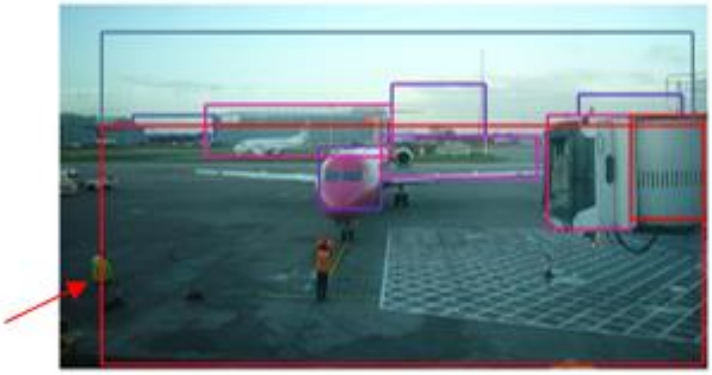
Failure case

CC score has shown good performance in the images in which there is a high contrast between the objects and the background but it has failed completely in the cases where the contrast is very low. Below are success and failure cases

d) Superpixel Straddling Cue

- We first generate dense windows(all possible windows in image greater than min height and width).
- Then we segment the image using slic function in skimage.sementation library which uses labels different segments based on superpixels.
- We then find the integral Histogram by finding the integral image of the above segmented image.
- We then compute the integral scores for all the windows in the image.

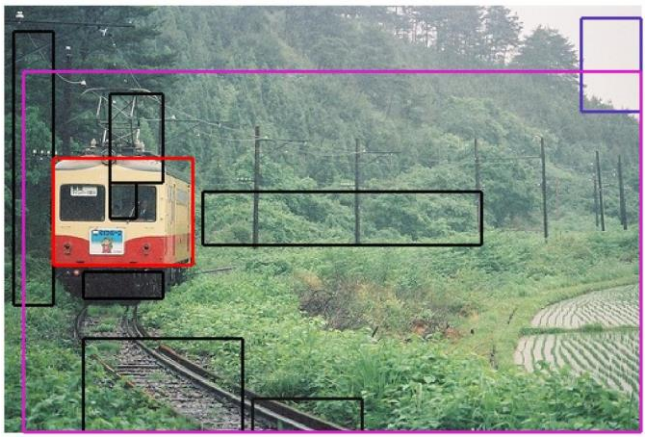
Success case

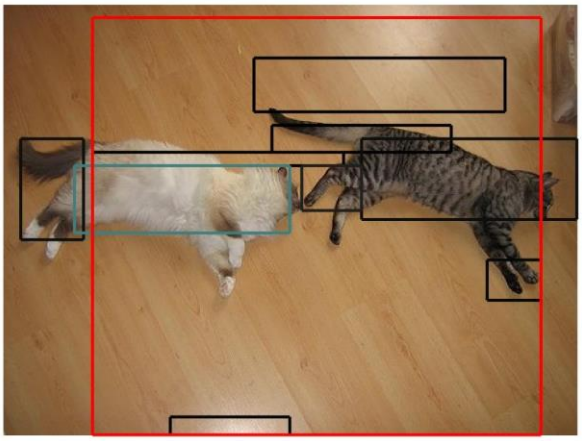


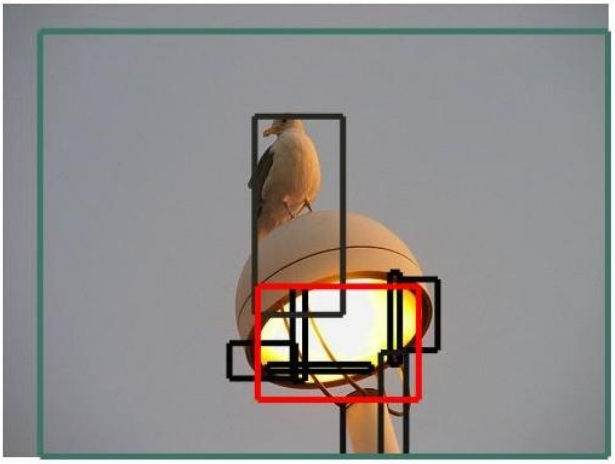
Failure case

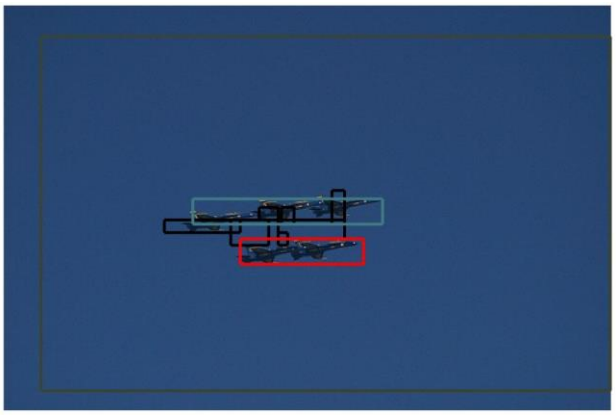
SS gave a considerable performance for objects that are sufficiently large in entire image but also failed in cases for small objects resulting in windows straddling the objects as shown above in failure case.

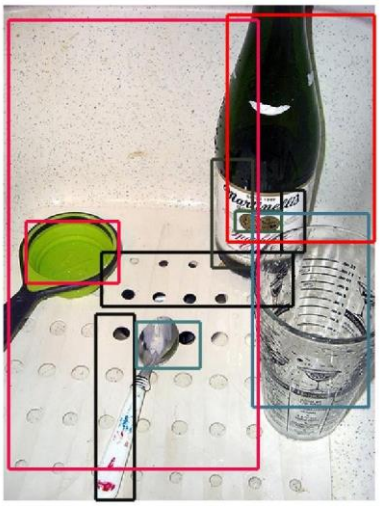
4) Results

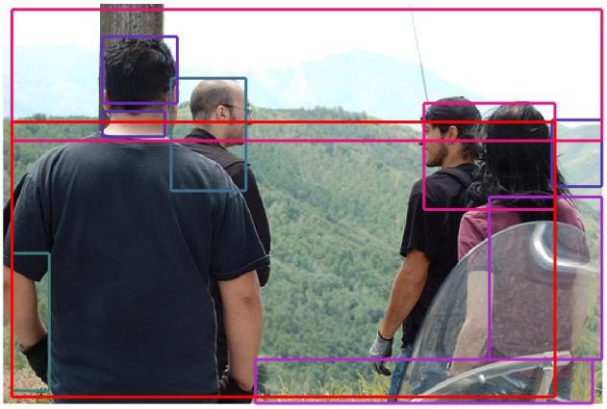


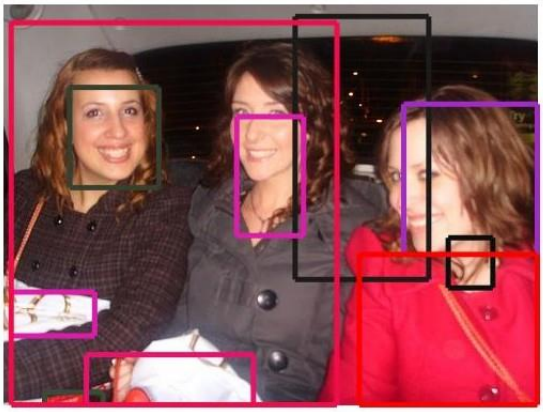


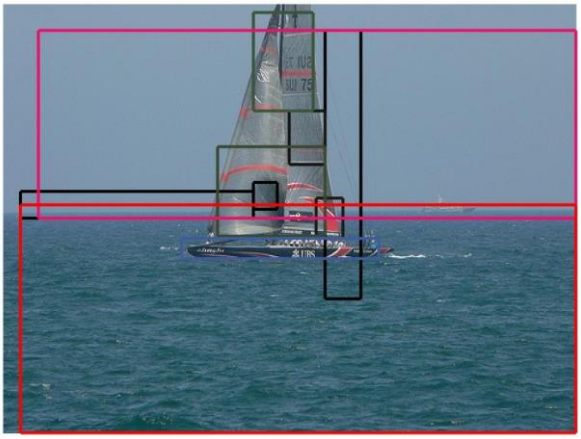


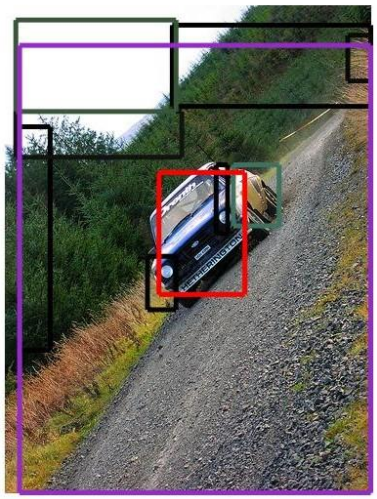


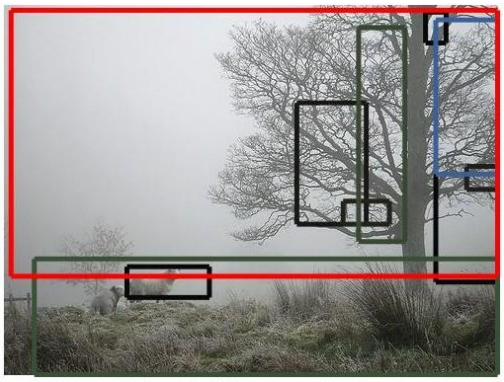


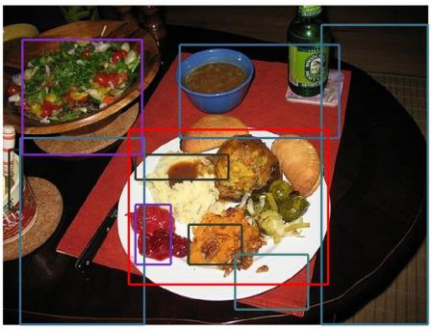


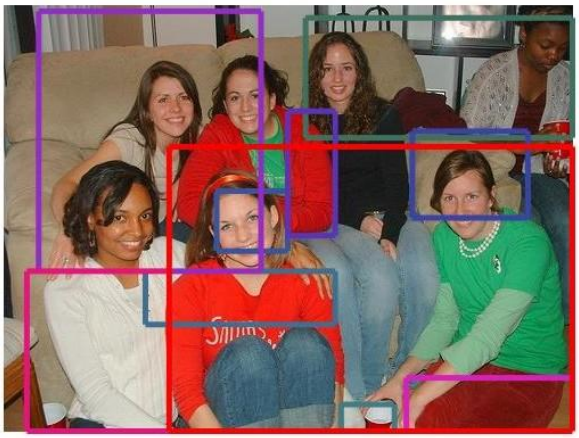












a) Analysis

- In the above results the red and green boxes are bigger and they most definitley cover the whole object as they are pessimistic predictors while the blue and black boxes are smaller and they sometimes contain the whole object correctly but sometimes just part of the object as they are optimistic predcitors.
- If there are lot of people, our measure is not able to detect all the faces.
- We are not able to complete trees with lots of braches.
- If there are lot of vehicles we are not able to detect all of them.
- There are some black color windows which do not contain any objects at all.

b) Future Directions

- We can use this cue as an input in convolutional neural networks for object detection as neural networks in itself takes a lot of time and our cue objectness measure takes less than 30 seconds to run.
- We can use histogram of gradients also as a cue and use bayesian integration to include it with our measure.
- Instead of Naive Bayes for bayesian integration which does not give optimal parameters we can use neural networks to train our data.

5) Task Assignment

a) Anoop Karnik

- Default Mask Generation from XML Data.
- Edge Density Cue Implementation.
- Multi Scale Saliency Cue Implementation
- Multinomial Sampling.
- Report

b) Prathyusha Akundi

- Color Contrast Cue Implementation
- Super Pixel Straddling Cue Implementation
- NMS Sampling
- Training Parameters for Cues
- Bayes Integration
- Presentation

6) Conclusion

We presented an objectness measure trained to distinguish object windows from background ones. Each cue mentioned above performs better in some areas and fails in certain cases. Since MS cue is designed to find blob-like things, it gives only a rough indication of whether object is present or not. CC gives better performance for images having high intensity variance but fails out completely in cases where images have low contrast. ED gives many false positives on images having multiple edges. Although SS gives better performance, it fails for small objects for which superpixels are fragile. So, we have integrated the cues using naive Bayes.

The GitHub link for the code, report and presentation is ***https://github.com/Prathyusha-Akundi/Measuring_Objectness***. The code is in jupyter notebook format in which image location and name can be given as input and after running all the code in the notebook we will obtain the windows containing the objects in the image. For Super pixels and NMS Sampling we have used online resources for sample codes and for others we have taken ideas mainly from the paper ‘Measuring the objectness of image windows’ and also some others.