Synthetic Saliency

Anthony Perez
Stanford University
aperez8@stanford.edu

Karan Rai
Stanford University
karanrai@stanford.edu

Ben Weems
Stanford University
bweems@stanford.edu

Abstract

There are many existing saliency detection methods that attempt to detect the dominant region of an image. Work has been done to optimize detection by combining these maps with fixed weights and applying a threshold on the weighted average to determine saliency. We propose a model that generates adaptive weights based on similarity to training images, and uses these weights to acquire the final saliency. Evaluating similarity to training images is accomplished by attaining high-level feature vectors of images and applying different models to determine the corresponding weights. We present methods for generating feature vectors of images and the models used to determine adaptive weights, and display the corresponding results, several of which improve on the literature. We also apply a bounding box to the salient region using a branch and bound algorithm. This reduces the rate of false positives by removing spurious pixels. It also provides a matrix format of the salient region that can be used for further analysis and applications. Finally, we show results on two different datasets to show general feasibility of this method. In summary, we show that using adaptive weights based on training images allows us to leverage the strengths of saliency models with reduced exposure to their weaknesses, thus providing improved results for saliency detection.

1. Introduction

Saliency in images is a measure of distinctive image quality assessment. It is a technique to identify or distinguish objects and regions of the image that stand out from the rest of the image. These are the qualities of an image that immediately grab our attention and, therefore, saliency is a subjective quality assessment method. Human beings are instantly able to analyze an image and process the details to detect saliency. Not only are we reliable, we are fast as well. Despite this, the computational models for saliency are still not well-established. This is an important, high-impact problem because with images and videos there is an overload of information. Each image has many pixels that do not contain relevant information, therefore many images can easily be compressed.

Saliency detection is an important computational tool as it has many practical applications. One obvious application is in object detection and hence, scene understanding. Another can be in compression as it will allow us to allocate more bits to salient pixels, giving us more detailed images that take up less memory. Further applications of saliency detection include Robotics, surveillance, graphics, representation, matching etc.

Currently, there are a lot of different techniques used for saliency detection. Because the human visual processing system is not fully understood, the heuristics used vary greatly and there is no one method that can be really said to be the best. After a survey of the most prominent said techniques, we observe that these various methods can be good in certain applications and that each has its own merits. We first select a few of these techniques which we felt were most successful and propose a novel approach to combine these saliency maps weighted based on high-level features obtained from the image itself. Further, we propose to enhance the saliency values by computing a bounding box over the salient object which can be useful for other applications like object detection as well.

We start off by describing the types of saliency approaches generally undertaken and discussing some of the most prominent ones following by a brief listing of the improvements that we have implemented. Next, we move on to the Methodology section where we first describe the entire algorithm by splitting it into 4 high-level modules followed by a step-by-step detailed technical description of each of our modules. After that, we go into the Experiments section where we compare the set of models we have used and compare them to other state-of-the-art works. We also discuss some of the failure cases experienced by these models. Lastly, we present the conclusion where we go over the main results, what we learned from our project and what we can keep with us moving forwards.

2. Previous Work

2.1. Overview

Previous work can be divided into three main classes: block-based models with intrinsic cues, region-based models with intrinsic cues, and models with extrinsic cues ([3]). All three are important to review in understanding our work because we apply a synthetic approach that relies on the strengths of each of these methods, especially the latter two.

Early works used the block-based model to predict saliency based on center surround contrast on a pixel by pixel basis [1]. As the field developed, blocks were used to determine contrast instead of individual pixels. Margolin *et al.* ([11]) proposed a method that defines the uniqueness of a patch relative to it's distance from the average patch in higher-order, PCA coordinates.

From the region-based models with intrinsic cues, we looked at context aware saliency ([5]) which will be described in detail in later sections. These techniques primarily rely on basic image features, such as color and contrast, to define distinctiveness between regions of the image.

Lastly, we have the models based on extrinsic cues. [7] uses frequency domain based on the empirical observation that amplitude spectrum of natural images lies approximately on a straight line. [6] uses probability distributions to model the image information according to the criterion of human fixation. [2] uses the idea that the more central a pixel is within the salient object, the smaller the lower frequency cutoff must be for detecting it.

[10] present a summary of naive bounding box detection techniques followed by an efficient branch and bound technique that we implement for our solution as well. [14] also propose object bounding boxes but these are not based on saliency but on edges.

2.2. Improvement

Our main contributions are as follows. First, we apply a variety of saliency computations methods to generate saliency maps. These methods have been chosen to cover most bases and, when put together, give the best result possible. Then we use an adaptive weight model to combine these saliency maps using high-level features vectors extracted from the images. Finally, we perform bounding box computation to fixate on the salient object. The benefit is 2-fold as it helps in reducing the number of false positives and can also be used for general object detection applications.

3. Methodology

3.1. Overview

The first step is compute preliminary saliency maps. These include context aware saliency ([5]), Patch Distinctiveness saliency ([11]) and DRFI ([8]) each of which will be described in detail later. Then, we compute high-level features for each image (Histograms, Bag of Visual Words, Fisher vectors etc.), cluster them using those features using various models (K Means, GMM etc.), and calculate weights which can be used to combine our preliminary saliency maps. Finally, we compute bounding boxes via a branch and bound algorithm on top of the saliency maps and remove pixels lying outside the bounding box.

3.2. Technical Details

3.2.1 Preliminary Saliency Maps

1. Context Aware Saliency ([5]) - It was earlier believed that salient regions detection should focus on just the object and nothing more. However, further psychological studies revealed that humans focus on not only the object but also some relevant portion of the background, which conveys the context. So the saliency map should detect the dominant object and some portion of the background. The salient region should include local (low-level) considerations, such as contrast and colour, global considerations (which suppress features that occur frequently), visual considerations (which take into account the possibility of existence of more than one center of gravity of the image, which turns out to be very useful in multiple-object detection), and finally high-level factors (such as human faces).

Low level and global considerations state that a pixel is salient if its appearance is unique. However, instead of looking at an isolated pixel, we look at the surrounding patch of each pixel too, which gives us an immediate context to determine uniqueness. If $d_{position}(p_i, p_j)$ is the Euclidean distance between the patches p_i and p_j and $d_{color}(p_i, p_j)$ is the distance in CIE Lab color space, dissimilarity measure between them is given by :

$$d(pi, pj) = \frac{d_{color}(pi, pj)}{1 + b \cdot d_{position}(pi, pj)} \quad (1)$$

Visual Considerations take into account the positional distance between patches. This automatically rules out background patches, as they are likely to have similar patches both near and far-away, in contrast with salient patches, which are generally grouped together.

Multi-scale saliency enhancement states that background patches are likely to have similar patches at multiple scales, which is contrast to more salient pixels that could have similar patches at few scales (but not all). Therefore, multiple scales are adopted to further decrease the saliency of background pixels, improving the contrast between salient and non-salient regions. In the single scale equation, we consider the K most similar patches in the image, and call a pixel to be salient when is very different as compared to its nearest other patches.

In multi-scale saliency measurement, for a patch p_i of scale r, we consider as candidate neighbors all the patches in the image whose scales are $\{r, r/2, r/4\}$. Among these patches, the K most similar ones are found and used for computing the saliency. Hence, the final equation for saliency of pixel i at scale r is given by:

$$S_i^r = 1 - \exp\Big(- \frac{1}{K} \sum_{k=1}^{K} d(p_i^r, q_k^{r_k}) \Big) \tag{2}$$

where $q_k^{r_k}$ is the k^{th} most similar patch which happens to be at a scale of r_k. Finally, the saliency at pixel i is taken to be the mean of its saliency at the different scales:

$$S_i = \frac{1}{M} \sum_{r \in R} S_i^r \tag{3}$$

After the saliency values for each pixel have been calculated, a pixel is classified as attended or non-attended, depending on whether the saliency value exceeds a certain threshold (generally 0.8). First, the most attended areas are extracted from the image, and each pixel outside the attended area is weighted according to its Euclidean distance to the closest attended pixel. The rationale behind this is that images tend to contain center(s) of gravity and salient objects are usually located nearby. The saliency of the pixel is then redefined as:

$$\hat{S}_i = S_i(1 - d_{foci}(i)) \tag{4}$$

where $d_{foci}(i)$ is the distance of pixel i with respect to its closest center of gravity or attention. Lastly, we add in high-level factors for which we use Viola face detector algorithm ([13]) and then take the maximum between our current saliency map and the face detection result. This can be further expanded to include other application-specific high-level factors.

2. Patch Distinctness - A previous paper that we chose to implement ourselves is the Margolin paper [11] "What Makes a Patch Distinct." We chose to include this paper because in a survey of papers we found that although it did not achieve a strong overall performance, it achieved a low rate of false positives. We believed that in a synthetic model, it could be advantageous to include methods with different strengths to such that their combination could leverage their diverse capabilities. Our implementation did not achieve the same scores as the Margolin paper, but applies the same principles without optimizing weights and implementing some of the post-processing features.

The technical details of this paper is split into three parts: pattern distinctness, color distinctness, and putting it all together. Color and pattern distinctness are measured on SLIC superpixels.

For pattern distinctness, the comparison is made to the average patch because non-distinct patches are concentrated around the average patch. The average patch is found under the L_1 norm using the equation

$$p_A = \frac{1}{N} \sum_{x=1}^{N} p_x \tag{5}$$

The distinctness is then meased by the L_1 norm in PCA coordinates, thus defining $P(p_x)$ as:

$$P(p_x) = ||\tilde{p_x}||_1 \tag{6}$$

where $\tilde{p_x}$ is the coordinates of patch x in the PCA coordinate system. We combine these distinctness measures to form a vector of distinctness for all patches.

We determine color distinctness by calculating the L_2 distances from all other regions in the CIE LAB colorspace. The equation for the color distinctness is given by:

$$C(r_x) = \sum_{i=1}^{M} ||r_x - r_i||_2 \, , \tag{7}$$

where r_x is the color profile of a region. This calculation is only done between SLIC superpixels to save time.

Finally, for putting it all together, we take the dot product of the vectors for color and pattern distinctness and apply a Gaussian to the center of the image because salient regions tend to cluster centrally in the image. We take this dot product to get our final result. The paper includes further steps to refine the final result that we have omitted in our implementation, including calculations at different resolutions and center-of-mass Gaussians for various thresholds.

3. DRFI - The method proposed in the paper "Salient Object Detection: A Discriminative Regional Feature Integration Approach" by Jiang et al. [8], abbreviated DRFI, has a pipeline composed of 4 steps. First an initial segmentation of an image is created. Then several (15 in our case) segmentations are produced from this initial segmentation by combining similar regions in an iterative manner. These segmentations are transformed into saliency maps by a random forest model. Finally those saliency maps are combined in a weighted sum using a fixed set of weights to produce a final output saliency map.

The initial segmentation of the image is created using the segmentation technique by Felzenszwalb et al. in [4] which is an efficient graph-based image segmentation. From the initial segmentation, additional segmentations are produced by combining similar and adjacent regions iteratively. See details in [4].

In order to transform the segmentations into saliency maps, a random forest model was trained. A 93 dimensional feature vector is extracted from each region in the segmentations. This feature vector describes the intra-region pixels, the similarity between the region and adjacent regions, and the similarity between the region and the pseudo-background which is the 15 pixel border around the image. The random forest model predicts the saliency of a region from these feature, and the saliency of each pixel in the region is set to the predicted value. [8] contains more details.

Finally each saliency map must be combined into one final saliency map. In [8] this is done using a fixed set of weights for a weighted sum ($\sum_{m=1}^{M} w_m A_m$). The weights were determined by minimizing a least squares estimator over all the training images ($\min ||A - \sum_{m=1}^{M} w_m A_m||_F^2$). This optimization problem to determine the weights is explained in more detail in the sections below.

The approaches we highlight below utilize the saliency maps produced as an intermediate output by the DRFI method. The saliency maps were produced using code provided by [8]. The numerous saliency maps they create provide us with a platform to test saliency map combination methods. Our methods differ from the DRFI method in that they use a set of weights that adapts to the image at runtime. The use of adaptive weights was is present in the DRFI method.

3.2.2 Combining Saliency Maps: Image Vectors

We used three different methods to extract feature vectors to classify the images. We used a simple gray scale histogram model, Bag of Visual Words and fisher encodings to extract high-level features of the images into a vector for processing.

To get the gray scale histogram, we first converted the RGB image into gray scale values ranging from 0 to 255. We then created a histogram with 256 buckets, one for each grayscale value. The value of each bucket corresponded to the fraction of total pixels that corresponded to that gray scale value. An array of this histogram was used as the image's feature vector.

We also used Bag of Visual Words encodings of the images. This method extracts common patterns from a set of images and creates buckets corresponding to those common patterns. A new image is described by a vector holding counts corresponding to the frequency of a bucket's corresponding visual pattern. We trained the Bag of Visual Words on our training images, and classified test images according to this model.

Finally, we used fisher encodings to generate a vector holding statistics about local feature descriptors, namely SIFT features. We first trained a GMM on all SIFT features from the training images. We store the means, covariances, and priors of this GMM, and use them to run a fisher encoding algorithm on the SIFT features of the images seen in the test set. This assigns each SIFT feature in the test image to a mode in the GMM with a certain probability. The mean and covariance of these probabilities are then used to generate the final fisher vector. This vector can be used to acquire second-order information about an image in a vector format.

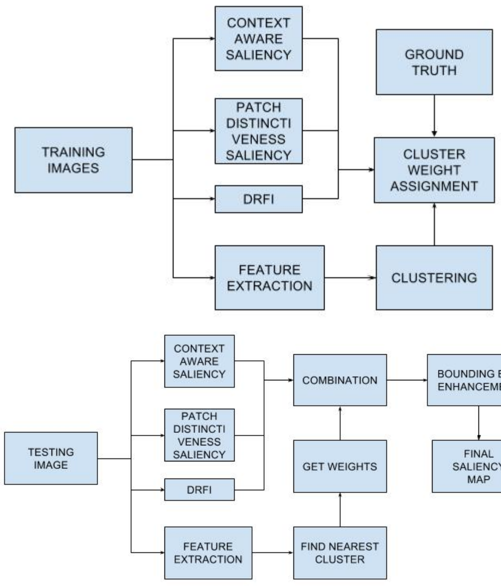

Figure 1. Our training and testing pipeline for methods using a GMM.

3.2.3 Combining Saliency Maps: Synthetic Models

We used four algorithms to adaptively assign weights to an image that would be used to combine the saliency maps produced from that image. Once these weights were assigned, the final saliency map is produced according to a weighted sum of the previous saliency maps ($\sum_{m=1}^{M} w_m A_m$). Each method is given three things: a feature extraction method that is used to transform training images into a feature vector, several saliency maps produced from each training image, and the set of "best weights" for each image. The "best weights" for an image are defined as the solution to the following optimization problem:

$$\min||A - \sum_{m=1}^{M} w_m A_m||_F^2$$

$$\text{s.t.} \sum_{m=1}^{M} w_m = 1$$

$$0 < w_m < 1$$

where A is the ground truth saliency map, M is the number of saliency maps produced, A_m is the m^{th} saliency map, and w_m is the weight associated with the m^{th} saliency map. [8] provided code to solve this optimization problem.

The first method used was K nearest neighbors. To determine the weights associated with a new image, it is first transformed into a feature vector. Then the K closest (in Euclidean distance) training images in the feature space are selected and the average of their "best weights" is associated with the new image. K was chosen to maximize the AUC score of a validation set of images and was set to 20 when using the Fisher feature representation. The K nearest neighbors method is labeled as KNN in the experiments section.

The remaining methods rely on a Gaussian mixture model to perform clustering on the training images in the feature space. The Gaussian mixture model is a common technique and relies on the EM algorithm to fit a fixed number of Gaussian distributions over the data set. These Gaussian distributions then represent clusters of the data. A data point (an image in our case) belongs to a cluster if the probability it was generated from the Gaussian distribution associated with that cluster is the largest. The number of clusters was chosen by minimizing the Akaike information criterion (AIC), which is the number of parameters minus the log likelihood of the data. The number of clusters was set to 6 when using the Fisher feature representation, although a higher number of clusters (64) was tested and found to perform worse. Refer to Figure 1 for a visual depiction of methods using a Gaussian mixture model.

The method labeled as GMM in the experiments section first associates a set of weights to each cluster found in the Gaussian Mixture Model then uses the cluster's weights to assign weights to new images. The set of weights associated with each cluster is the mean of the "best weights" for each image that belongs to that cluster. To assign weights to a new image, soft clustering is employed. The assigned weights are a sum over the weights of each cluster weighted by the probability that the image belongs to that cluster. The weights assigned to a new image are given by the following expression:

$$\vec{w} = \frac{\sum_{c=1}^{C} p(I|c)\vec{w_c}}{\sum_{c=1}^{C} p(I|c)}$$

where $\vec{w}$ are the weights assigned to the new image, C is the number of clusters, $p(I|c)$ is the probability of the image being produced from the Gaussian associated with cluster c, and $\vec{w_c}$ are the weights associated with cluster c.

The method labeled Soft in the experiments section, is the same as the previous method, except the weights associated with each cluster are assigned according to an optimization problem rather than a mean. The optimization problem is the same as the one used to determine the "best weights" for an image, except it is a sum over every image in the cluster ($\min \sum_{i=1}^{I} ||A_i - \sum_{m=1}^{M} w_m A_{i,m}||_F^2$).

The method labeled Hard in the experiments section associates weights to each cluster in the same manner that the Soft method does. However to assign weights to a new image, the Hard method simply uses the weights associated with the cluster that image belongs to. This is known as hard clustering hence the names Hard and Soft (GMM technically also does soft clustering).

3.2.4 Bounding Boxes

After combining all of the saliency maps, the aggregated map obtained is quite spread out. In other words, there are many spurious pixels marked as salient either due to noise or edges or some background features. The next and final step is to remove these false positives by detecting a bounding box around the salient object. Such a bounding box has further applications for object detection, representation etc. The major difficulty is in pre-defining the amount of saliency the salient region/object should contain, as it depends on the size and shape of the salient object, as well as how cluttered the background is. Traditional methods use thresholding and then searching for windows but these approaches are highly sensitive to the selection of threshold, which is difficult to optimize, and are very computationally intensive. Hence, we employ a branch and bound algorithm for bounding box detection (from [10]) using an area-based heuristic.

The saliency score (called Saliency Density) of a window W is defined to be -

$$f(W) = \frac{\sum_{(x,y)\in W} S(x,y)}{\sum_{(x,y)\in I} S(x,y)} + \frac{\sum_{(x,y)\in W} S(x,y)}{C + Area(W)} \quad (8)$$

where $S(x,y)$ is the saliency of pixel at (x,y) coordinate, I is the entire image and C is a parameter used to balance the trade-off between area of the window and its average saliency. In the expression, the first term denotes what fraction of the total images saliency is captured in this window while the second term denotes the saliency density of that window with respect to area of the window. Now, there are quadratic (in the image size) number of possible windows hence, we use a branch and bound algorithm to compute this efficiently. A window can be full determined by its top, bottom, left and right coordinate. So, we define a set of windows as a tuple consisting of 4 possible ranges, one each for the top, bottom, left and right coordinates. Then, we define the upper bound function for a window W belonging to a set of windows X (for Branch and Bound algorithm) as -

$$f(X) = \frac{\max_{W\in X}\left(\sum_{(x,y)\in W} S(x,y)\right)}{\sum_{(x,y)\in I} S(x,y)} + \frac{\max_{W\in X}\left(\sum_{(x,y)\in W} S(x,y)\right)}{C + \min_{W\in X} Area(W)}$$

We see that $f(X)$ is an upper bound for any window $W \in X$ and equal when $W = \{X\}$ which are the 2 requirements for a branch and bound algorithm. In each iteration, we pick the most promising set so far, split it into two by dividing the range of the dimension (which has the largest range) into 2. Putting all that together gives us algorithm 1.

```
Data: Image $I$, Saliency map $S \in R^{m\times n}$
Result: Bounding box coordinates - T,B,L,R
Create an empty Priority Queue PQ ;
$X \leftarrow ((1,m),(1,m),(1,n),(1,n))$;
while $X$ contains $> 1$ window do
    Split $X$ into $X_1$ and $X_2$ based on largest dimension ;
    Insert $X_1$ into PQ with priority $f(X_1)$ ;
    Insert $X_2$ into PQ with priority $f(X_2)$ ;
    $X \leftarrow$ Top element from PQ ;
end
Assert $X = ((t,t),(b,b),(l,l),(r,r))$;
Return t,b,l,r ;
```

Algorithm 1: Bounding Box Computation Algorithm

4. Experiments

The primary metric we use to evaluate the performance of our methods is the AUC score so that we can compare our results to [8]. To calculate the AUC score all the saliency maps produced have a threshold applied at various values from 0 to 255 to produce salient object segmentations. Then the true and false positive and true and false negative statistics are calculated over all pixels in the test images at each threshold. We accumulate these statistics to produce the ROC curve which is based on the true positive rates and false positive rates. From the ROC curve we compute the AUC score via numerical integration.

We tested on 2 datasets. MSRA-B dataset can be found here `http://research.microsoft.com/en-us/um/people/jiansun/SalientObject/salient_object.htm` [9] and ECSSD dataset can be found here `http://www.cse.cuhk.edu.hk/leojia/projects/hsaliency/dataset.html` [12]. All of our own code along with the already implemented code for the context aware saliency method can be found here `https://github.com/bweems/cs231a`. The code from the DRFI paper we used was recently update. The more recent version is located as `https://github.com/playerkk/drfi_matlab` and the older version that we used seems to have been taken down, but was previously available at `http://jianghz.com/projects/saliency_drfi/index.html`.

4.1. Feature Comparison

Figure 2 shows the experiments that were run to determine which features produced the best results. The K nearest neighbors method and the Gaussian Mixture Model method perform consistently across the feature representations. However, for both the Hard and Soft clustering methods the Fisher feature representation produces the best results. We also see that the Fisher feature representation and the bag of visual words representation both generalized fairly well. For these reasons, the Fisher feature representation appears to be the most powerful.

4.2. Weight Selection Algorithm Comparison

Figure 3 compares the different methods using the Fisher feature representation across both the MSRA-B dataset and the ECSSD dataset. The GMM method empirically performed the best, although all methods outperform the original DRFI method when using the Fisher feature representation. The GMM method seems to be a more uniform version of the DRFI output, although on occasion it does produce a very different image. This is visible in the output examples below in Table 1.

4.3. Bounding Boxes

We used $C = 0.2 * width * height$ as our parameter after performing some manual testing. Furthermore, we extended the bounding boxes by 10 percent extra in each direction to avoid missing boundaries of the salient object.

As can be seen in Figure 4 it appears that the addition of the bounding box results in lower AUC scores. However, the bounding boxes can be useful as a method for finding windows for other computer vision algorithms and the qualitative results look reasonable. We also have a couple of thoughts that may help improve our bounding boxes for saliency purposes. One is to extend the boxes by a higher margin (say 20 percent). This might help with the saliency measures but is not the best for object detection. The other thought is to scale up the pixels inside the box and scale down the pixels outside.

4.4. Alternative Datasets

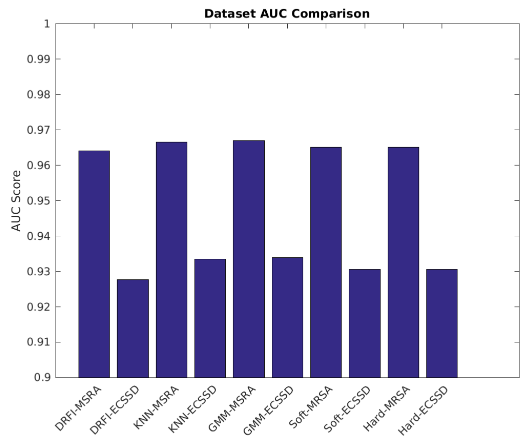

Figure 5. Comparison across datasets. The methods are using the Fisher feature vectors.

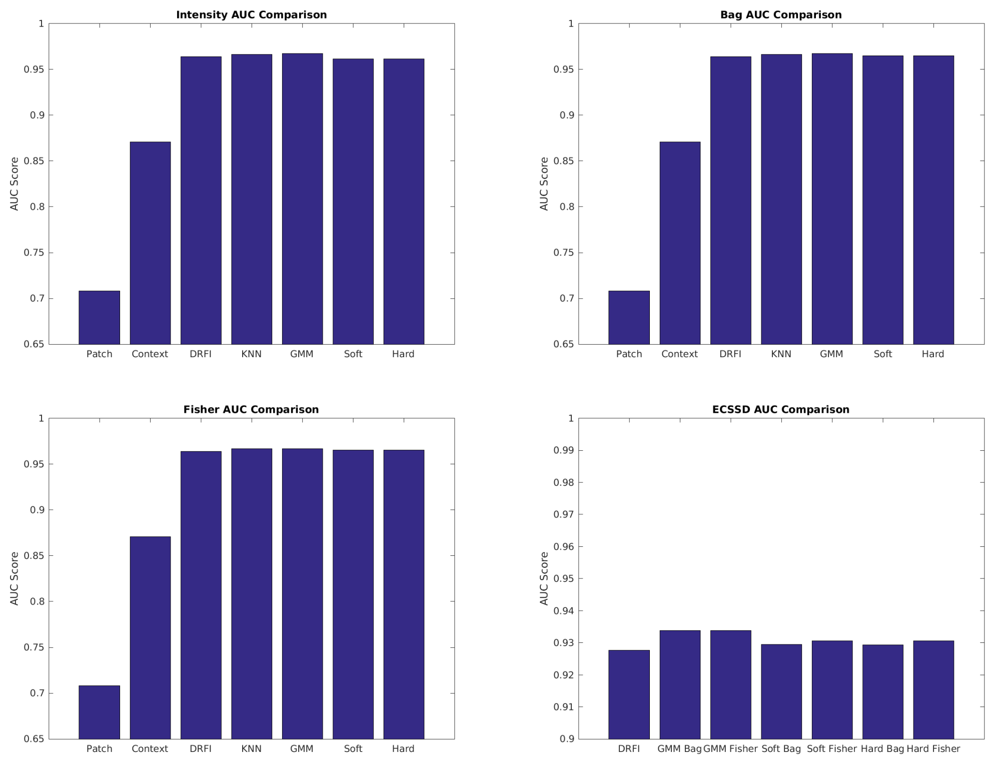

Figure 2. Feature Results Comparison. "Intensity" refers to the gray scale histogram feature representation. "Bag" refers to the bag of visual words feature representation. The DRFI output is independent of the features and is the same across the plots. Context refers to the method in [5]. Patch refers to the method in [11]

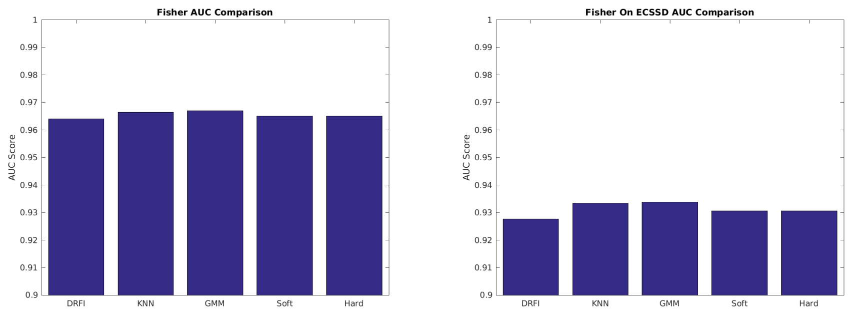

Figure 3. The weight selection algorithm comparison. All methods here are using the Fisher feature representation.

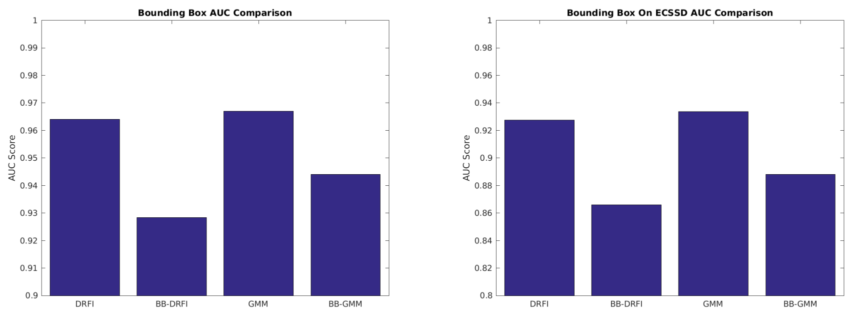

Figure 4. The bounding box method comparison. The "BB-" prefix on a method indicates the bounding box algorithm was run on the output of the method. The axis on the ECSSD dataset's plot changes.

The MSRA-B dataset was the dataset used for training our models. To test the generalization of our method we examine our results on the ECSSD dataset. The ECSSD dataset, or the Extended Complex Scene Saliency Dataset, includes many semantically meaningful but structurally complex images for evaluation. This is also the dataset on which the method in [8] performs the worst. The ECSSD dataset was included to test our methods on images that represent failure cases, such as images where the saliency object is similar to the background, images where the background is very cluttered or complex, and images with multiple salient objects. From the results in Figure 5 it appears that all the methods generalize to the ECSSD data set reasonably well. The difference between the result from [8] (labeled DRFI) and the methods we introduce remain approximately the same between datasets.

4.5. Additional Saliency Maps

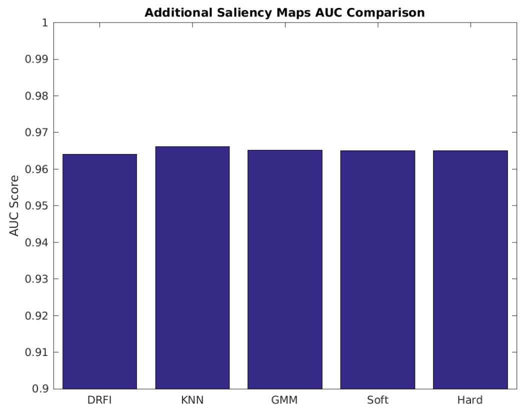

Figure 6. Additional Saliency Maps AUC Comparison

Figure 6 displays the results of the methods using the Fisher feature representation with the two additional saliency maps from [11] and [5]. The addition of these two saliency maps results in slightly worse performance for our methods, likely because they were worse saliency maps than the intermediate saliency maps produced by the DRFI approach [8], which are the only saliency maps all the other results use. Future work could focus on synthesizing final results of other state-of-the-art saliency maps instead of relying on our implementations to generate the additional mappings.

4.6. Qualitative Results

Table 1 shows some of the output results for the methods that have been discussed. The first and fourth image in the results show images where the methods proposed in this paper outperform previous methods. The first image is a case where the background had a noisy object. The second and third images highlight how the KNN and GMM methods tend to be more uniform versions of the DRFI approach. The second and fourth images show situations where some parts of the salient object lie outside of the bounding box.

4.7. Failure Cases

Table 2 shows some images that were failure cases for our methods. The third and fifth images are cases of images with cluttered backgrounds. The methods capture the salient object but also capture some of the background objects. However, the bounding box method does a reasonable job of capturing the salient object within the image, even in the presence of noise in the saliency map. Perhaps the bounding box methods previous discussed could be used to improve saliency maps produced from cluttered images. The first, second, and third images are cases where the salient object is similar to the background. These images were the most difficult. All images in Table 2 are from the ECSSD dataset.

5. Conclusion

Computational saliency detection in itself is quite a diverse (and open) field because many recent works use very different heuristics and ideas to predict saliency. We observe that different models have their own advantages and disadvantages. For example, block-based models are simpler but fail to capture the differing shapes of objects. The context-aware model succeeds in detecting outlines but struggles to detect interior regions of salient objects. The Saliency DRFI method is powerful, but produces patchy saliency maps.

Our main contribution was to propose an adaptive weights based combination model for incorporating the advantages of different techniques that were most successful on similar images. We tested various models for both feature selection and clustering scheme, and we found GMM with Fisher vectors to generalize the best. We compared our results with related recent work and see improvements. This suggests that we successfully clustered images into groups, each of which had a preference for certain saliency map techniques.

Finally, we enhanced our final saliency map using bounding boxes. Although this results in lower AUC scores, we believe that it is a valuable step because it generates output in a data structure that is useful for applications such as scene understanding and object detection. We also discussed a few failure cases and some ideas that may help overcome them.

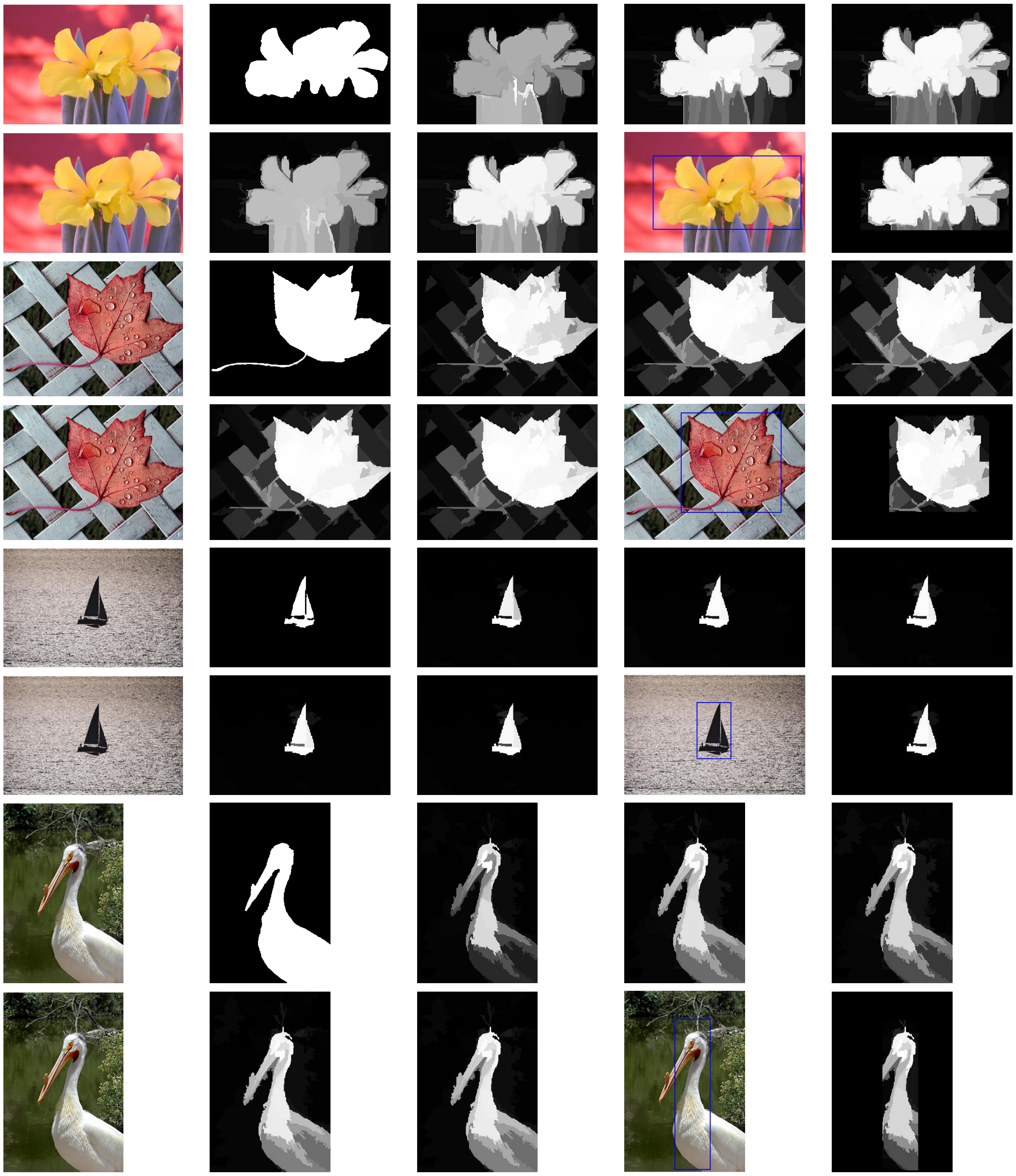

Table 1. This table shows images and outputs from various methods. Each image occupies two rows in the table above. Along the first row (from left to right) is the original image, the ground truth, the DRFI output, the KNN method output, and the GMM method output. Along the second row (from left to right) is the original images, the Soft method output, the Hard method output, and the last two images show the bounding box over the original image and the GMM saliency map respectively. All methods here used the Fisher feature representation.

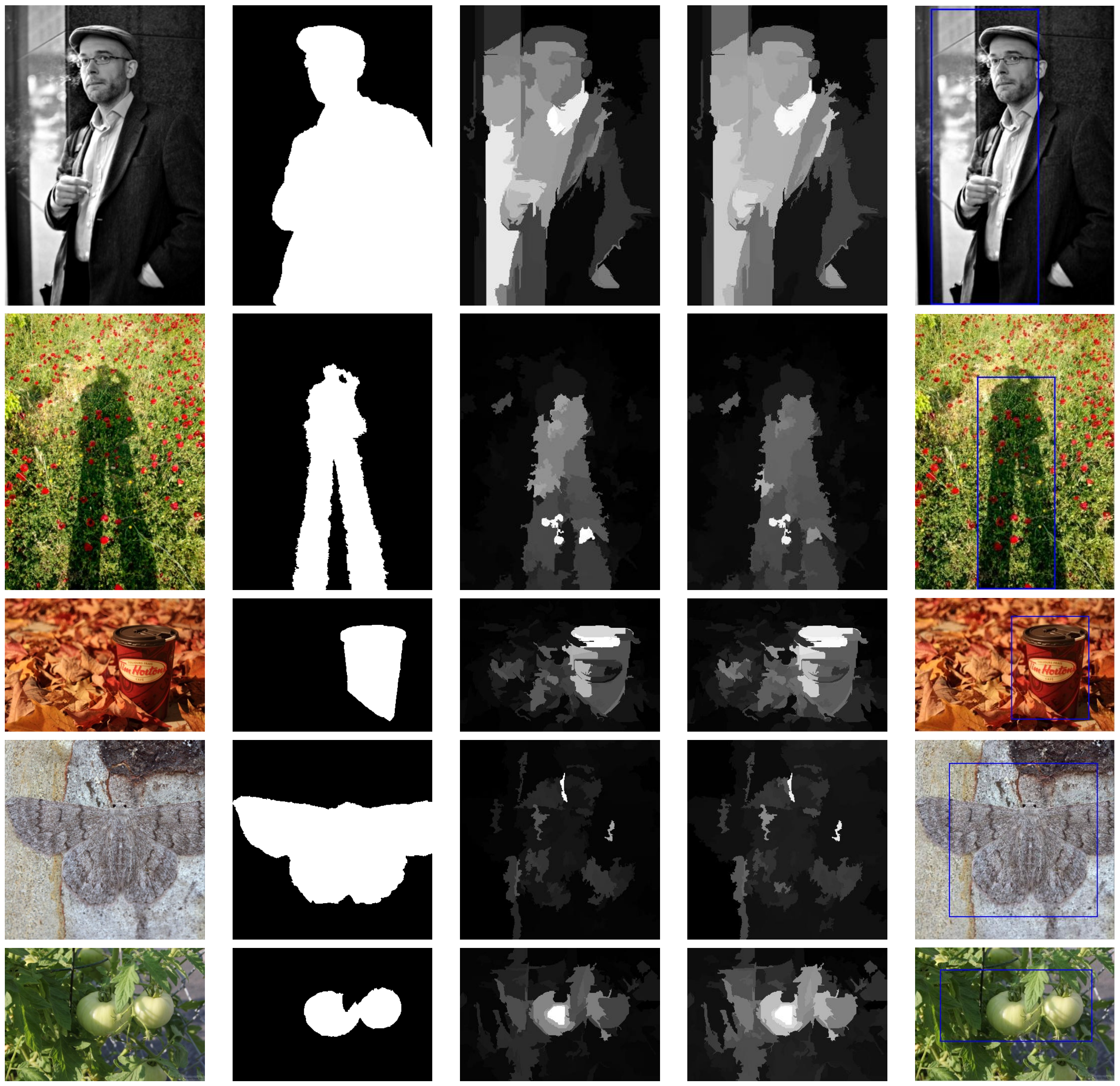
Table 2. This table shows images that are failure cases for our methods. Each row has one image and its saliency maps. The columns in order (from left to right) are the original image, the ground truth, the DRFI output, the GMM output using the Fisher representation, and the bounding box computed from the GMM-Fisher output.

References

[1] R. Achanta, F. Estrada, P. Wils, and S. Süsstrunk. *Computer Vision Systems: 6th International Conference, ICVS 2008 Santorini, Greece, May 12-15, 2008 Proceedings*, chapter Salient Region Detection and Segmentation, pages 66–75. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[2] R. Achanta and S. Süsstrunk. Saliency detection using maximum symmetric surround. In *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pages 2653–2656. IEEE, 2010.

[3] A. Borji, M.-M. Cheng, H. Jiang, and J. Li. Salient object detection: A survey. *arXiv preprint arXiv:1411.5878*, 2014.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Ecient graphbased image segmentation. 59(2):167181, 2004.

[5] S. Goferman, L. Zelnik-Manor, and A. Tal. Context-aware saliency detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(10):1915–1926, 2012.

[6] J. Harel, C. Koch, and P. Perona. Graph-based visual saliency. In *Advances in neural information processing systems*, pages 545–552, 2006.

[7] X. Hou and L. Zhang. Saliency detection: A spectral residual approach. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*, pages 1–8. IEEE, 2007.

[8] H. Jiang, J. Wang, Z. Yuan, Y. Wu, N. Zheng, and S. Li. Salient object detection: A discriminative regional feature integration approach. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2083–2090, 2013.

[9] T. Liu, Z. Yuan, J. Sun, J. Wang, N. Zheng, X. Tang, and H.-Y. Shum. Learning to detect a salient object. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(2):353–367, 2011.

[10] Y. Luo, J. Yuan, P. Xue, and Q. Tian. Saliency density maximization for object detection and localization. In *Computer Vision–ACCV 2010*, pages 396–408. Springer, 2010.

[11] R. Margolin, A. Tal, and L. Zelnik-Manor. What makes a patch distinct? In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1139–1146, 2013.

[12] J. Shi, Q. Yan, L. Xu, and J. Jia. Hierarchical saliency detection on extended cssd. *arXiv preprint arXiv:1408.5418*, 2014.

[13] P. Viola and M. J. Jones. Robust real-time face detection. *International journal of computer vision*, 57(2):137–154, 2004.

[14] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. In *Computer Vision–ECCV 2014*, pages 391–405. Springer, 2014.