1 Task 1

Optimal SNR values

Image	Gaussian	Median	Anisotropic	Bilateral	Combination
Image1	18.41	24.29	17.54	18.98	N.A.
Image2	19.73	17.61	18.55	19.86	N.A.
Image3	15.8	15.36	15.02	15.97	16.74

Table 1: SNRs for three distorted images at optimal filter parameters

Parameters for Optimal SNR

Image	Gauassian	Median	Anisotropic	Bilateral	Combination
Image1	sigma=1.3, filter size 7	default	filter size=8	filter size=10, spatial sigma=2, range sigma=0.6	N.A.
Image2	sigma=1.2, filter size 7	default	filter size=4	filter size=10, spatial sigma=2, range sigma=0.4	N.A.
Image3	sigma=1.5, filter size 7	default	filter size=6	filter size=10, spatial sigma=2, range sigma=0.9	1) Median Filter:default 2) Bilateral Filter: filter size=7, spatial sigma=2, range sigma =0.2

Table 2: Optimal parameters for three distorted images

Best filter for Image1 As Table 1 shows, Median Filter has the highest SNR value for Image1, which indicates it is the most suitable filter in this case. The reason for this phenomenon is that Image1 contains Salt & Pepper noise, while Median Filter is the best suitable filter to remove Salt & Pepper noise among these filters.

Best filter for Image2: The best filter for image 2 is the Bilateral Filter. Image2 contains Gaussian Noise, which could be removed by either Bilateral or Gaussian Filter. The relatively high performance of Bilateral Filter is that, compared to Gaussian Filter, Bilateral Filter introduce an intensity term to avoid uniformly smoothing across all areas in the image, as showed in Equation 1 and 2. Therefore, Bilateral Filter could protect the similarity between the borders of the objects while still removing the noise. After tuning the parameters, I found that Bilateral Filter with half filter size of 5, spatial sigma with 2 and intensity sigma with 0.4 has the best SNR result in my case.

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)\, I_{\mathbf{q}} \tag{1}$$

$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q} \in S} G_{\sigma}(\|\mathbf{p}-\mathbf{q}\|) I_{\mathbf{q}} \tag{2}$$

Best filter for Image3: For image 3, two filters are used to remove the noise. First, a Median Filter is used to remove Salt & Pepper noise in the raw image, which is quite obvious from visual inspection. Next, a Bilateral Filter is applied to smooth the image. The trade-off between Bilateral Filter and Gaussian Filter has been discussed in the above paragraph. The best Bilateral Filter parameters are: Filter size=7, spatial sigma=2, intensity sigma=0.2.

Implementation of Anisotropic Filter The implementation details of Anisotropic filter is as follows: 1) Extract the neighbors of the selected pixel, if the neighbor size is less than the assigned kernel size (e.g. in the corners of the image), shrink the kernel size to fit for the neighbor; 2) obtain the max difference D in the kernel; 3) calculate the similarity by using Equation 3,where p is the selected pixel, q is the neighbor, d is the distance between p and q; 3) calculate the new pixel values by using Equation 4, where p' is the new pixel values.

$$S(p,q) = \frac{D-d}{D} \tag{3}$$

$$p' = \frac{\sum q * s(p,q)}{\sum s(p,q)} \tag{4}$$

Implementation of Bilateral Filter The implementation details of Bilateral Filter is as follows: 1) Extract the neighbors of the selected pixel, if the neighbor size is less than the assigned kernel size (e.g. in the corners of the image), shrink the kernel size to fit for the neighbor; 2) calculate the spatial weight and intensity weight by using Equation 1; 3) obtain the new pixel values.

2 Task 2

2.1 Algorithm Overview

In this section, I will introduce my algorithm for saliency detection on the given dataset. The main part of my algorithm is the histogram-based contrast algorithm [1], with slight modifications on image pre-processing and saliency image generation. Equation 5 describes how to calculate saliency values for each pixel by leveraging histogram-based contrast method,

$$S\left(I_k\right) = S\left(c_l\right) = \sum_{j=1}^{n} f_j D\left(c_l, c_j\right) \tag{5}$$

where c_l is the color value of pixel I_k, n is the number of different color values, f_j is frequency of each color j in the image (calculated as occurrences/sum of all color occurrences), $D(c_l, c_j)$ is the color distance metric for pixel c_l and c_j in L*a*b color space, showed in Equation 6.

$$D(c_l, c_j) = \sqrt{\left(L_2^* - L_1^*\right)^2 + \left(a_2^* - a_1^*\right)^2 + \left(b_2^* - b_1^*\right)^2} \tag{6}$$

Generally speaking, my algorithm achieves an average of **27%** Intersection of Union on the giving dataset. The implementation details will be illustrated in the following paragraphs.

Step 1: Color Reduction The first step is to pre-process the pixel values in the input images. As Equation 5 shows, the complexity of the algorithm is $O(N) + O(n^2)$, which is quite computational inefficient since there are 256^3 possible colors. To speed up the saliency detection, [1] suggests to reduce the number of pixel colors in the RGB color space to 12 distinctive colors in each channel, thus reducing the number of colors to 12^3. To this end, I leverage an matlab built-in function *rgb2ind*() to reduce the color in the raw image. The *rbg2ind*() function takes raw image, specified color numbers and the option of dithering as the input[1], and returns an indexed image with a color map. Then, the indexed image with color map is converted back to RBG color space. Figure 1 and Figure 2 show the difference between raw image and color-reduced image in RGB color space. After the above color reduction process is completed, the color-reduced image is converted from RGB to Lab color space for saliency detection.

Step 2: Find Existing Color Patterns This step is to find color patterns (number and values of the pixel colors) in the image. As Equation 5 shows, there is a preparation step to calculate the probablity of existing colors. Therefore, it is essential to group same pixel colors and calculate the total occurrences of each color in the image. The probability is obtained by dividing each occurrence number by the total number of occurrence of all the colors.

Step 3: Saliency Image Generation After all the set-up steps are done, Equation 5 is used to calculate the saliency value for each pixel in the Lab color space. The saliency value for each pixel is determined by Equation 6. To enhance the performance of the algorithm,a

[1]Note that dithering is a technique to average the colors in the neighborhood and approximate them to the original RGB color

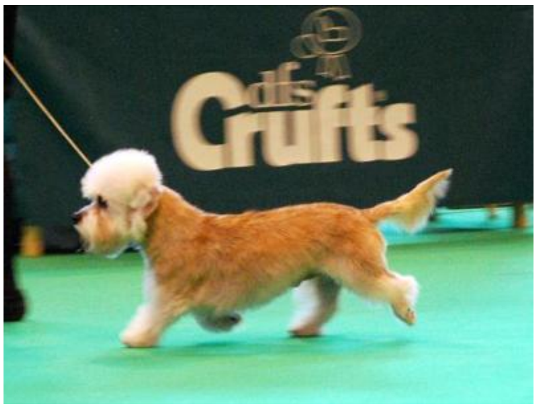
Figure 1: Raw image in RGB color space

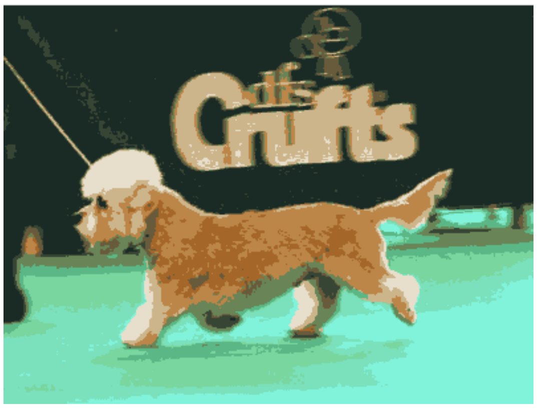
Figure 2: Color-reduced image in RGB color space

median filter is applied on the saliency image to smooth possible noises generated in saliency computation process. Empirical study shows that it could achieve an average of 2% IoU improvement on the given dataset, Figure 3 and Figure 4 shows the comparison of saliency image w/o median filter. From visual inspection, it is clear that median filter has smoothed several noise spot in Figure 3. After the filtering, the saliency image is normalized by using Min-Max Normalization [2].

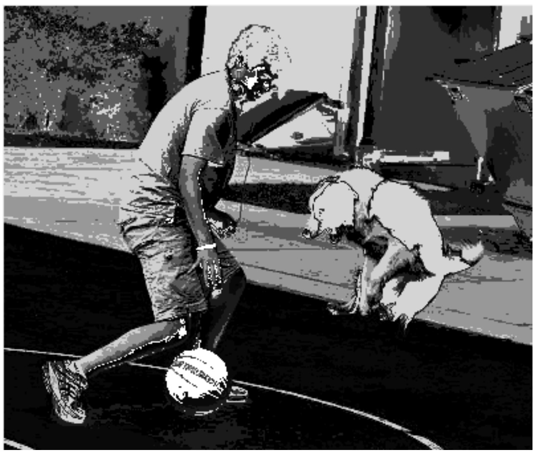
Figure 3: Saliency image without median filter

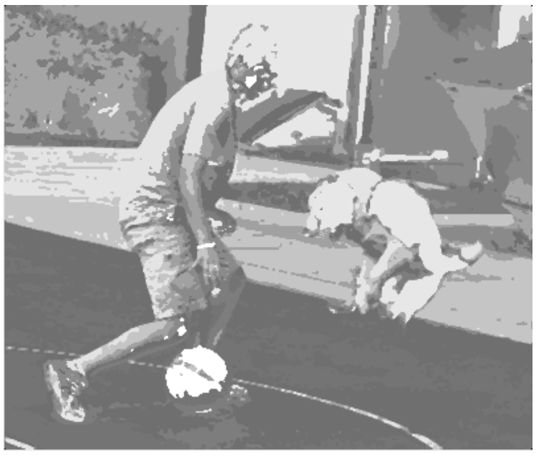
Figure 4: Saliency image with median filter

Step 4: Binarization To obtain a binary image from a grey-level image, *greythresh*() and *imbinarize*() functions are used. The *greythresh*() function takes a grayscale image as input and returns a global threshold for it. The *imbinarize*() function takes an grayscale image and a threshold for input, pixel values that are greater than the threshold will be set to 1 while pixel values that are less than the threshold will be set to 0. The last step is to compare the generated images with their corresponding ground truth and calculate the Intersection of Union (IoU) for them.

2.2 Advantages

- **Easiness of implementation.** My algorithm is based on the histogram-based contrast method proposed in [1], with slight modifications on image pre-processing and saliency image processing. Generally speaking, the algorithm is quite easy to implement or reproduce, as the core of the algorithm is simply to compute the distance metric between the selected pixel and other pixels in Lab color space.

- **High performance for images with simple background.** One observation from the experimental results is that my algorithm performs far better in images with simple background and mono object as the object-of-interests. Figure 5 is the detected saliecy of the 67th image in the given dataset. From human-eye inspection, it already has a relatively high similarity with the ground truth. The IoU of Figure 5 is 85%, other images with simple background and mono object as the foreground also shares relatively high IoU values, ranging from 60% to 90%.

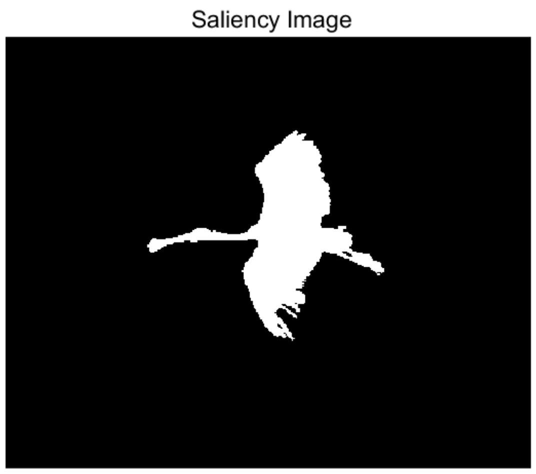

Figure 5: Saliency image

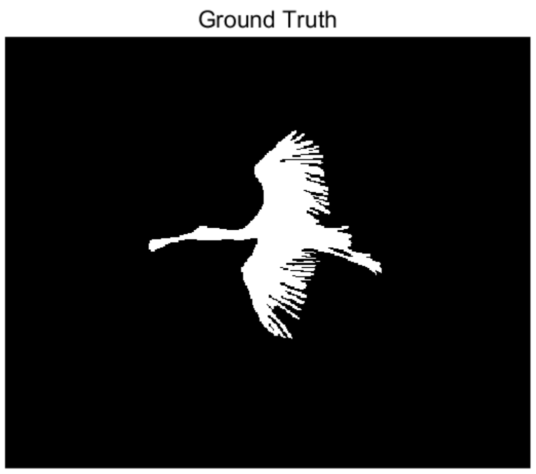

Figure 6: Ground Truth

- **Median Filter to Enhance Performance.** As it is illustrated in the step 3, a Median Filter is used to smooth the saliency image, which could enhance the average performance of the algorithm by 2% IoU values. Empirical study also shows that Gaussian Filter, with sigma=1.6, filter size= 7 could have similar improvement. The possible explanation for this phenomenon might be that both Median and Gaussian Filter could effectively smooth the noises generated by saliency calculation, thus enhancing the performance when using threshold to binarize the saliency image.

2.3 Limitations

- **Relatively Long Execution Time.** On average, the saliency computation for each image is about 1 seconds, which leaves a large room for improvement. The reason for this overhead might be the image pre-processing and distance calculation, as the current design contains complex transitions between different color spaces (RGB -> indexed image -> RGB -> Lab) and the distance function could be further optimized.

- **Generalizability.** Currently, the IoU variance of the testing dataset is quite high, as the IoU values vary from 0% to 91% and the average IoU is 27%. The high variance indicates that the current algorithm does not have a good generalizability for saliency detection. By taking a close look on the results, I found that the current algorithm design performs poorly on images with complex background (e.g. multiple colors and objects in the background) or images that the objects to be detected shares similar colors with the surroundings. Figure 7 is an example for this case. It is quite difficult to differentiate the skin color of the elephant with the color of the sand in the background, which explains why Figure 8 extracts the background together with the elephant.

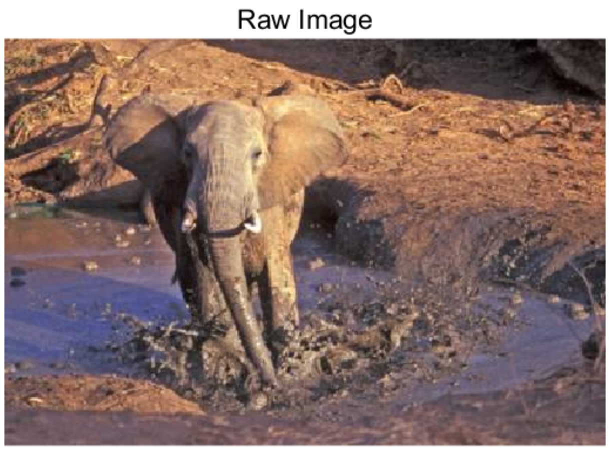

Figure 7: Raw Image

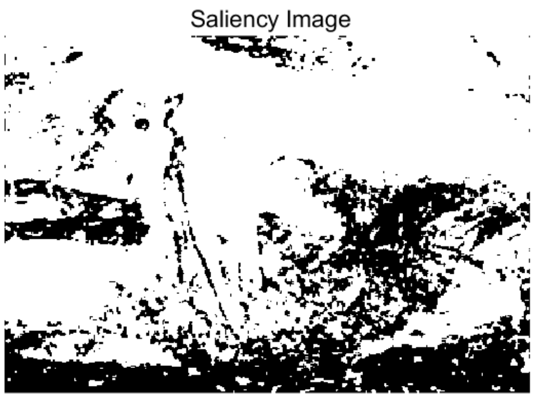

Figure 8: Saliency Image

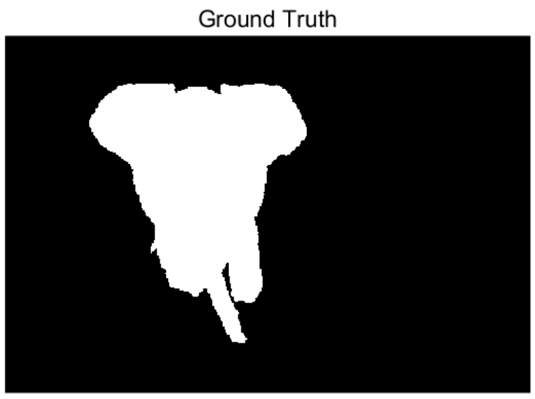

Figure 9: Ground Truth

Another factor that causes the lack of generalizability is that histogram-based contrast method might cut an object input different parts, if it has stripe-alike shapes with distinctive colors. The reason might be that the current algorithm fails to approximate and unify the colors within the object, which leads to object internal saliency detection. Figure 11 is an example for this case. In the raw image 10, the player wears a red-black striped shirt. The algorithm mistakenly recognize them as different objects, which seriously degrades the overall performance (see Figure 11 and Figure 12).

- **Object of Interests.** The current algorithm could detect the foreground of each image, but cannot differentiate the actual object of interests. Figure 13 illustrates this scenario. The raw image is a bird standing on top of a twig. Althouth the

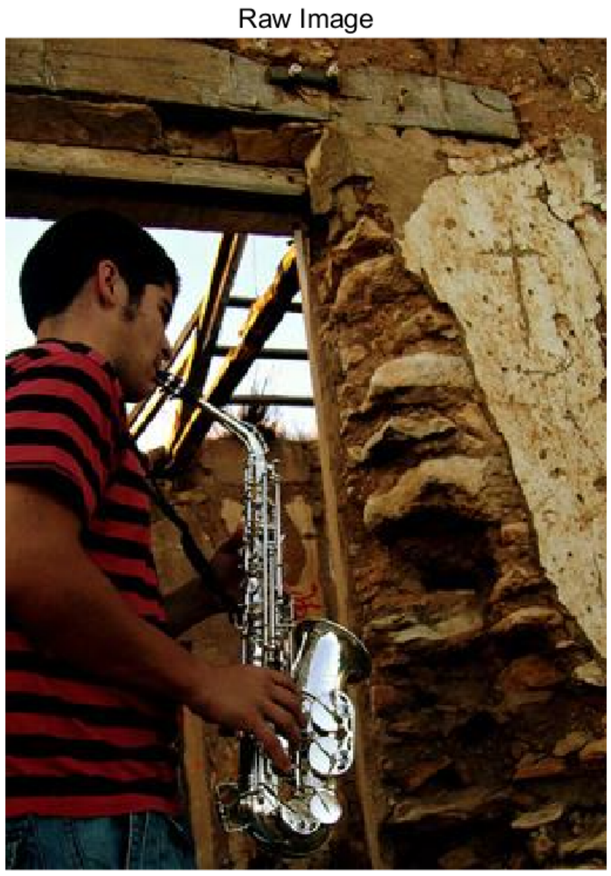

Figure 10: Raw Image

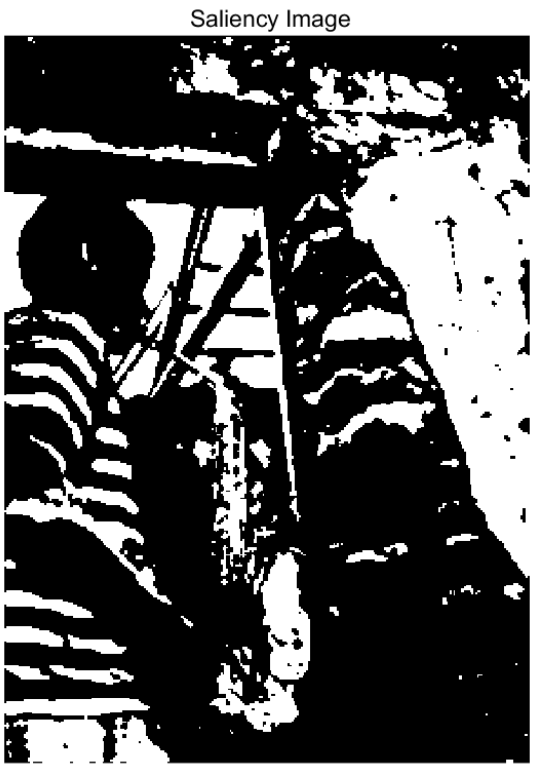

Figure 11: Saliency Image that fails to approximate colors within in the object

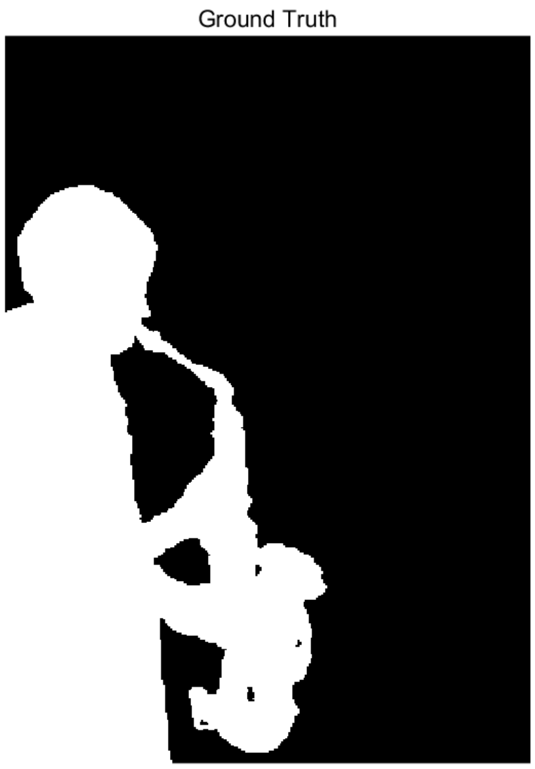

Figure 12: Ground Truth

algorithm successfully extracts the bird and the twig from the background, it cannot separate the bird with the twig. As the bird is the object-of-interest in this case, the detection includes unnecessary elements.

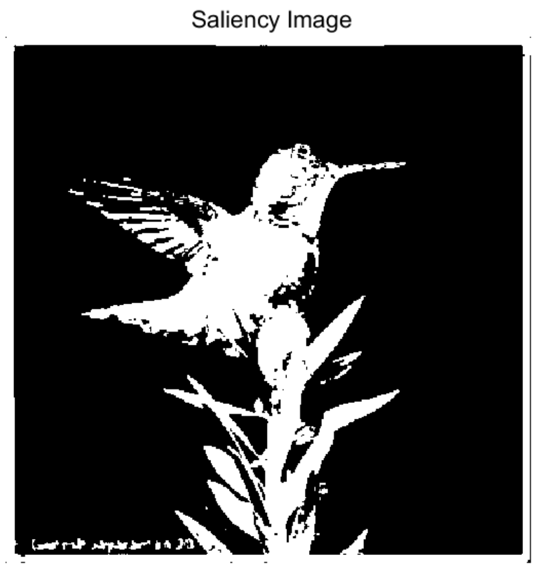

Figure 13: Saliency Image with leaves

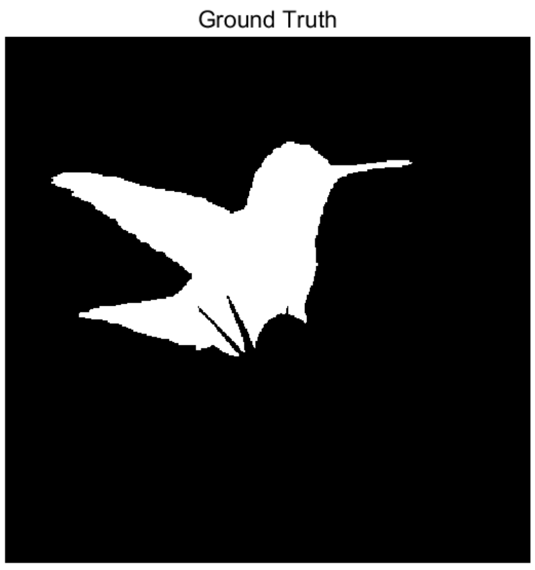

Figure 14: Ground Truth without leaves

References

[1] Ming-Ming Cheng, Guo-Xin Zhang, Niloy J. Mitra, Xiaolei Huang, and Shi-Min Hu. Global contrast based salient region detection. In *CVPR 2011*, pages 409–416, 2011.

[2] S Patro and Kishore Kumar Sahu. Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*, 2015.