Saliency Detection using Dynamic Mode Decomposition (DMD) and its variants

TABLE OF CONTENTS

Abstract

Visual saliency is the ability to differentiate the important parts of the image so that they stand out from their neighbors and grab our attention. It decreases the amount of visual data that is to be processed. It is very crucial, as images can be processed without knowing their actual contents. In this project, we have detected saliency regions of an image using dynamic mode decomposition (DMD) and its variants (rSVD-DMD, TDMD). A novel method of image representation was employed to make DMD applicable to static images. We have utilized color and luminance information and generated a saliency map. The complex and non-linear nature of the human visual system is modeled by making use of the power of the different color spaces to generate a color-based saliency map. The fact that the sensitivity of the human eye towards brightness is higher than color is utilized to generate luminance saliency maps. The method employed is computationally less expensive, straightforward, and produces full-resolution saliency maps.
The validity of the generated saliency map is estimated using standard performance measures such as F-measure, precision, recall, mean absolute error (MAE), and area under the ROC curve.

Index Terms: - Dynamic mode decomposition, Randomized SVD DMD, Total DMD, Non-linear data, color saliency map, luminance saliency map

1. Introduction

Saliency detection is the process of identifying regions in an image that are most likely to attract attention. Dynamic Mode Decomposition (DMD) is a powerful mathematical tool that has been used for various tasks, including saliency detection. DMD and its variants can be used to analyze the temporal dynamics of an image sequence and identify the regions that change the most over time. These regions are considered the most salient, as they are the most likely to capture the viewer's attention. In this report on saliency detection using DMD and its variants, we would likely introduce the problem of saliency detection, discuss the background and importance of the task, and provide an overview of the DMD and its variants, including their strengths and limitations. The report would then go on to describe the methods used for saliency detection using DMD and its variants, present the results of the experiments, and discuss the implications and future directions for research in this area.

Proposed techniques:

Global contrast-based model

One approach to saliency detection is the global contrast-based model, which evaluates the saliency score of an image region by considering the color contrast of that region relative to the entire image.
This method is considered computationally efficient and is able to generate a consistent saliency map. Another method, utilizes matrix decomposition to break down the input image into low-rank and sparse components for saliency detection. A further technique, utilizes a Gaussian Mixture Model (GMM) to account for both global contrast difference and spatial coherence in order to generate an improved saliency map.
Another study proposed a method that leverages brightness information of the foreground objects and uses a series of flash and non-flash images as inputs to compute the saliency map.

Local contrast-based models

There are different approaches to saliency detection, one of which is the use of local contrast-based models. Local contrast-based models may be computationally intensive, and have a tendency to assign greater importance to regions that are textured or have edges. A publication introduced a local filter called "center-surround difference" which utilizes local contrast to compute a saliency map.
Another research combined multiple image features using Kullback-Leibler divergence to center-surround difference to create an improved saliency map.Also we do this by extracting contrast features at various scales of an image and use a graph-based model to fuse them together to produce a single saliency map.

2. Literature Review

Let us see the conventional methods used for saliency detection.
Combining simple priors
The initial techniques used for saliency detection was done by combining 3 priors and creating simple prior. The 3 priors include the following:

1. Salient objects are modelled by band pass filtering
2. People pay attention to the center of the image
3. Warm colors are more attractive to people than cold colors

The bandpass filtering responses are integrated with opponent color channels to generate frequency prior. From the color information of the image we generated color prior. The red and yellow represent warm colors and blue and green represent cold colors. The RGB image is converted to CIEL*a*b image. Here a* channel represents green-red tone while the b* the channel represents a blue-yellow tone. When the a* value decreases the image becomes green tone while when the a* the value increases and the image becomes red toned. Similarly, when the b* value decreases, the image becomes blue tone while when b* value increases and the image becomes a yellow tone. So, the pixel with higher a* and b* means, it is warmer while the reverse case means, it is colder. Based on this analysis, a new metric is made for saliency map. The next important characteristic is spatial locations in an image. The locations near the center of the image are salient than the ones far away from the center. From this a gaussian map is generated. The frequency prior, color prior and location prior are combined together to generate simple prior. One of the main advantages of this method is, it gives statistically better results for saliency detection and has low computational complexity and is useful for time critical applications. So it is a good technique to be applied for a real time application. But one of the serious limitations of this technique is that it fails when the complexity of the dataset increases.

Non-parametric regression saliency

There comes the other technique, non parametric regression saliency. When the dataset becomes complex due to the addition of noise like white gaussian noise or any other noise, this technique is used. The dissimilarities between the center patch of a region and other patches is computed. These dissimilarities are aggregated to estimate the underlying saliency of the region. By local data dependent weighted least square problem, the saliency is estimated. The kernel function gives higher weight to similar patch pairs while gives lower weight to dissimilar patch pairs. In order to enhance the performance, global and multiscale approach by extending the local analysis window to the entire image. This technique fails when the image is rainy or foggy or smoky or blurry in nature.

Regional covariance matrices

There comes the other technique, using the regional covariance matrix. Generally the diagonal elements in the matrix represent feature variances and non-diagonal elements represent correlations among the features. A foggy image has color contrast, outlines are clear, single target object, color distortion, reduced contrast and color dislocation. The fog in the image affects low frequency information and high frequency information of an image. One of the limitations of this method is that based on the color domain of the image, saliency detection method is affected.

Need for DMD

The conventional methods take either the spatial components or the temporal components to generate saliency maps, while the DMD takes both the spatial and temporal components into account to generate saliency maps. It is computationally efficient and gives better results than the conventional methods.

3. Objective

The main objective of this project is to exploit the analytical power of DMD and its variants, such as randomized SVD DMD, total DMD, to generate saliency maps and to visually and quantitatively analyze the effectiveness of each of these variants. This project also aims to analyze color spaces in order to improve image representation by separating luminance and chrominance and generating the saliency map. Also, the spatial information of the generated saliency map is utilized to further enhance it.

4. Theoretical Background

4.1 Variants of Dynamic Mode Decomposition

The dynamic mode decomposition (DMD) is the foremost algorithm to decompose complex systems into spatiotemporal coherent structures, or modes. These are then used for short term future state prediction and control. Generally, dynamic mode decomposition identifies modes that are spatially correlated and oscillate at a fixed frequency in time with a growing or decaying envelope. As a result DMD is considered as an ideal combination of

1. Model reduction technique
2. Fourier transforms in time

In this project, we investigate the efficacy of the DMD and its variants, such as the randomized singular value based DMD and the total DMD, for saliency region detection of an image.

4.1.1. Dynamic Mode Decomposition

Snapshot Sequence,

$$S = [s_0\, s_1 \ldots\ldots\ldots s_n]\ \epsilon\ R^m$$

$$S_1 = [s_0\, s_1 \ldots\ldots\ldots s_{n-1}]\ \epsilon\ R^{m\times n}$$

$$S_2 = [s_1\, s_2 \ldots\ldots\ldots s_n]\ \epsilon\ R^{m\times n}$$

$A : R^m \rightarrow R^m$ that advances S_1 to S_2

$$S_2 \approx AS_1$$

$$\widehat{A} = \arg\min_{A} ||S_2 - AS_1||_F^2$$

In closed form the linear operator,

$$\widehat{A} = S_2 S_1^{\dagger}$$

where, $S_1^{\dagger}$ denotes the pseudoinverse.

The dominant eigenvectors of $\widehat{A}\ \epsilon\ R^{m\times m}$ are the DMD Modes.

1) Compute SVD of matrix S_1 for the desired rank k.

$$S_1 \ = U_k \Sigma_k V_k^H$$

where
$U_k \in R^{m\times k}$ and $V_k \in R^{n\times k}$ are the right and left singular vectors, and $\Sigma_k \in R^{k\times k}$ contains the non $-$ zero singular values.

2) Compute Best fit linear map A,

$$\widehat{A} := S_2 S_1^{\dagger} = S_2 V_k \Sigma_k^{\dagger} U_k^H$$

3) $\widehat{A} = U_k^H \widehat{A} U_k = U_k^H S_2 V_k \Sigma_k^{\dagger}$

4) Compute Eigen decomposition of $\widehat{A}$

$\widehat{A}W = W\Lambda$

5) Finally, we reconstruct the k dominant eigenvectors of $\widehat{A}$

$\phi_k = S_2 V_k \Sigma_k^{-1} W_k$

Where ϕ is the dynamic mode matrix with corresponding DMD Modes.

4.1.3. Randomized Singular Value Decomposition -Dynamic Mode Decomposition

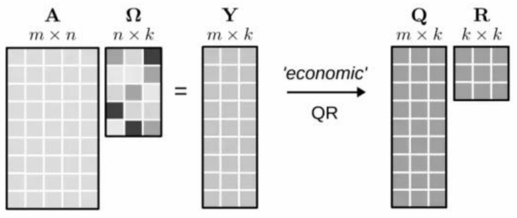
Singular value decomposition is a matrix factorization technique that computes the low rank matrix approximation of the data points. In order to obtain a faster and more efficient low rank approximation, randomization concepts are introduced. The rSVD factorization is done as follows:

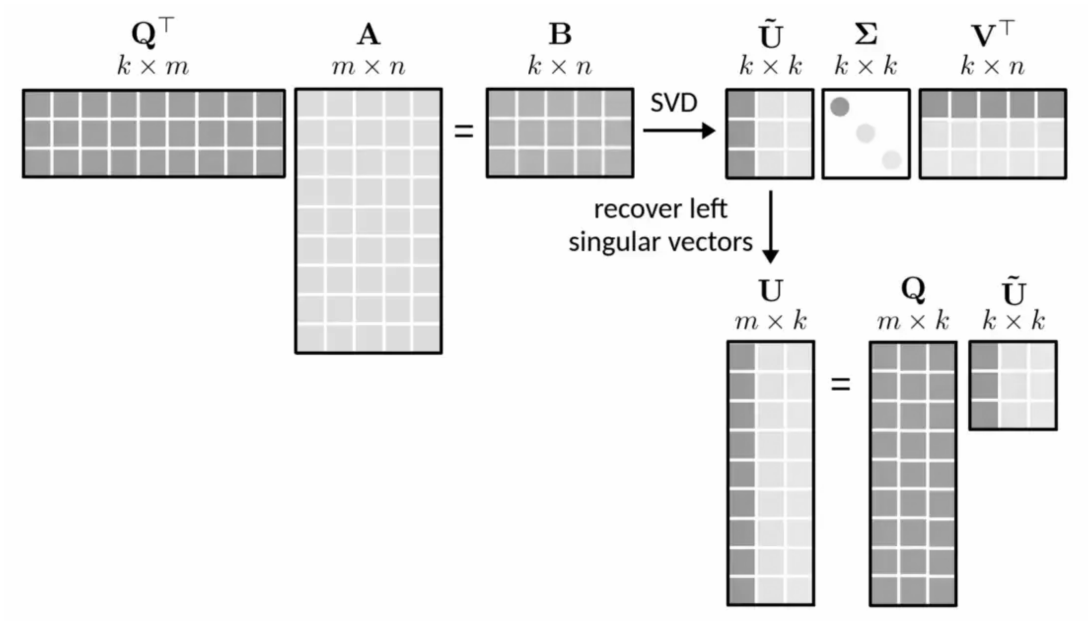
1. For a given input matrix, a low dimensional subspace is constructed
2. Standard factorization is performed for the constructed basis by confining the input matrix to the low dimensional subspace.

The essential idea behind randomized SVD is stated here. For any given mxn matrix **A**, if we impose a target rank **k** with k < min(m,n), then

1. Generate a Gaussian random matrix $\boldsymbol{\Omega}$ with size nxk
2. Compute mxk matrix **Y**
3. Apply QR decomposition to the matrix **Y**
4. Compute kxn matrix **B**, by multiplying the transposed matrix **Q** and the matrix **A**
5. Compute the SVD of matrix **B** (B is a smaller matrix with respect to A)

The pictorial representation of the above-mentioned algorithm is given below.





Further, the general steps in the DMD algorithm are executed to compute eigen decomposition and corresponding DMD modes.

4.2 Static Image Interpretation via DMD

Generally, the dynamic mode decomposition is used for dynamically varying data as of now. Typically, the DMD is not used to decompose static images. In this project, we have dynamically represented a static image. The chrominance and luminance information of an image is used to represent it as a data matrix. The power of different color space representations is used to model the nonlinear and highly complex behavior of the human visual system, as each color space provides different color similarity measures.

Here, an RGB image of size (mxn) is converted to different color spaces such as CIELab, YUV and YCbCr. With the different color space transformations, the images are better represented and have the capability of separating luminance and chrominance.

The chrominance and the luminance are based on the degree of receptivity towards brightness and color information.

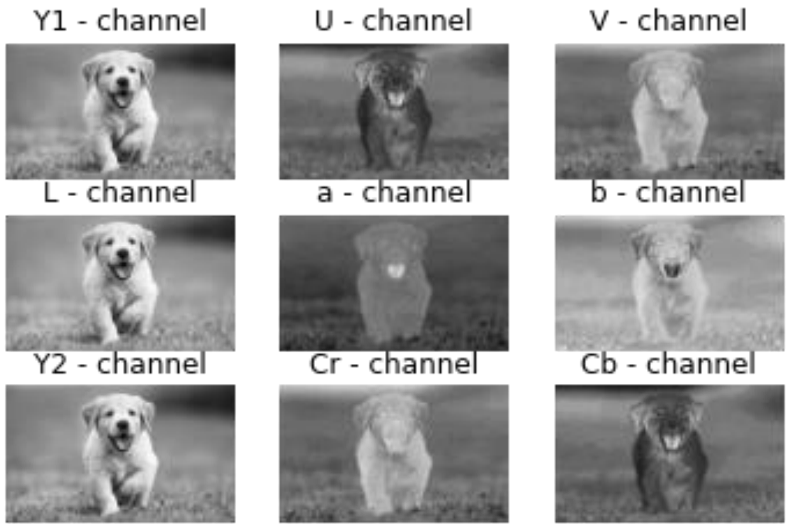
- *Channels used to generate saliency map based on luminance*
    1. Y1 from YUV
    2. Y2 from YCbCr
    3. L from CIELab

- *Channels used to generate saliency map based on color*
    1. U and V from YUV
    2. Cb and Cr from YCbCr
    3. a and b from CIELab

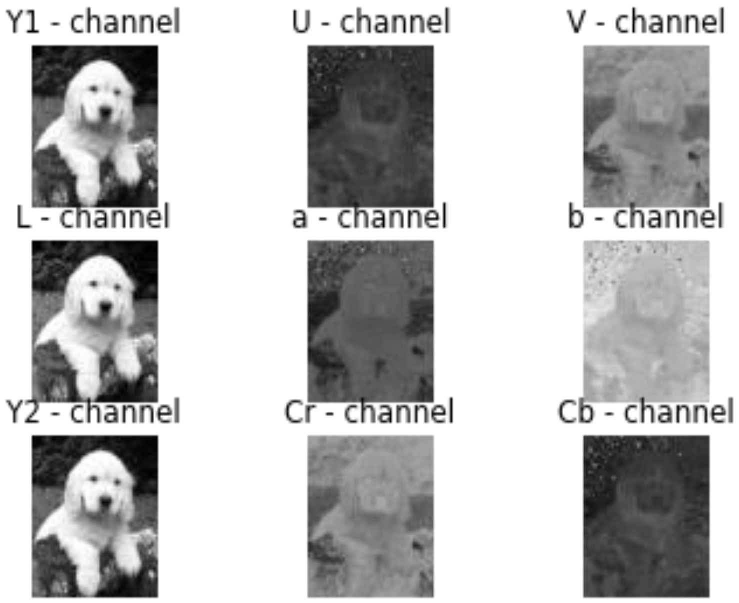
4.2.1. Color based DMD representation of images

Color is a crucial cue that attracts human attention. It is important to give a DMD representation to model the chromatic information. Let us now perform a color space analysis to get to know the effect of different color channels on the salient regions. Here, two types of images are taken, one with a blurry background and the other with a distinct background. The images are depicted below.

For blurry background:



For distinct background:



We can see from this that salient regions in an image have varying degrees of apparentness in different color channels. The salient regions are more prominent in more than one color channel. From the above analysis, we observe that salient regions are clearer in **b** from CIELab, **V** from YUV and **Cr** from YCbCr channels for the image that have clear distinct differentiation between salient part and background. Similarly, the salient regions are clearer in **a** from CIELab, **U** from YUV and **Cb** from YCbCr channels for the image that have a blurred background.

From the observations of the color space analysis, the data matrices X_{color_1} and X_{color_2} are computed. These two data matrices have color channels (color channels are vectorized) as column vectors.

$$X_{color_1} = \begin{bmatrix} | & | & | & | \\ | & | & | & | \\ b & V & Cr & D_1 \\ | & | & | & | \\ | & | & | & | \end{bmatrix} \quad X_{color_2} = \begin{bmatrix} | & | & | & | \\ | & | & | & | \\ a & U & Cb & D_2 \\ | & | & | & | \\ | & | & | & | \end{bmatrix}$$

Here, $D_1 = b + V + Cr$ and $D_2 = a + U + Cb$. The column vectors $\mathbf{D_1}$ and $\mathbf{D_2}$ represent the prominent salient regions.

The columns of the data matrices $[X_{color\_1}]_{mnx4}$ and $[X_{color\_2}]_{mnx4}$ where mn is the size of the image (m - number of rows; n - number of columns), are permuted and repeated to form an input for the

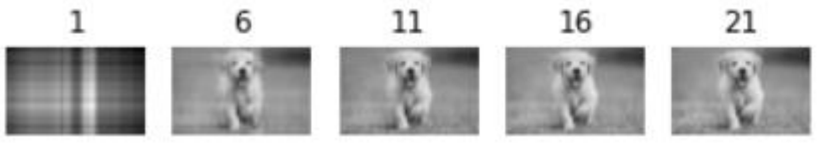
DMD method to generate a color based saliency map. The permutations of columns in the matrices X_1 and X_2 are depicted below.

Permutations	**1**	**2**	**3**	**4**	**5**
$\mathbf{X_1}$	$bVCrD_1$	$VCrbD_1$	D_1CrVb	$VCrD_1b$	VbD_1Cr
$\mathbf{X_2}$	$aUCbD_2$	$UCbaD_2$	D_2CbUa	$UCbD_2a$	UaD_2Cb

The more columns in the data matrix (the data matrix is as large as possible), the more accurately we can separate the salient part from the background. The color-based saliency map works well for images where the foreground and background have distinct distributions. If they are both identically colored, the color-based saliency map fails.

4.2.2 Luminance based DMD representation of images

For the images with identical background and foreground color distributions, luminance information is utilized. The luminance components are reconstructed using singular value decomposition, and they are given as input to the DMD. The intermediate singular values represent the salient parts of an image, while the smaller and larger values represent the non-salient parts. Reconstruction of the image with respect to various number of singular values (1, 6, 11, 16, 21) taken into account is described below.



We can observe that the SVD-reconstructed images have more salient features. Also, as the number of singular values used for reconstruction increases, the salient region becomes clearer while the background remains relatively static after a set of iterations. Based on this observation, we can conclude that SVD reconstructed images are applicable for the DMD method. Now we vectorize each of the SVD images and unite them into a single data matrix.

$$X_{Luminance} = \begin{pmatrix} | & | & . & . & | \\ | & | & . & . & | \\ x_1 & x_2 & . & . & x_l \\ | & | & . & . & | \\ | & | & . & . & | \end{pmatrix}$$

When there are l intermediate singular values, the mnx1 vectors x_1, x_2, x_3,..x_l where each xi represents a vectorized mxn SVD reconstructed image. This data matrix is given as input to the DMD algorithm for further processing, such as the separation of low rank and sparse components.

4.3. Salient region detection using dynamic mode decomposition

Here, let us see how to generate the color saliency map and luminance saliency map. They are combined to create a full saliency map. Further, the full saliency map is utilized for segmentation of the images.

4.3.1. Color based saliency map

Generally, the luminance and chrominance are embedded with each other in case of an RGB image. So, the RGB image of size mxn is converted into the YUV, CIEL*a*b and YCbCr color spaces.

1. From each of the color space mentioned above, chrominance information is taken and vectorized for mnx1 vectors.
2. Generate the two data matrices X_1 and X_2 from the color space analysis.
3. Permutate and repeat the column vectors of the X_1 and X_2 matrices. The above matrix obtained is the input matrix.
4. Input matrix is fed into DMD algorithm
5. The DMD modes concentrated towards the origin are termed as lowrank (X_{low1}, X_{low2}) while those bounded away from the origin are termed as sparse ($X_{D\_sparse1}$, $X_{D\_sparse2}$) . Lowrank component represents the background while the sparse component represents the salient object.
6. Normalize the low-rank and sparse components to generate a smooth saliency map.
   $X_{D\_sparse1\_norm} = a / b$

where:

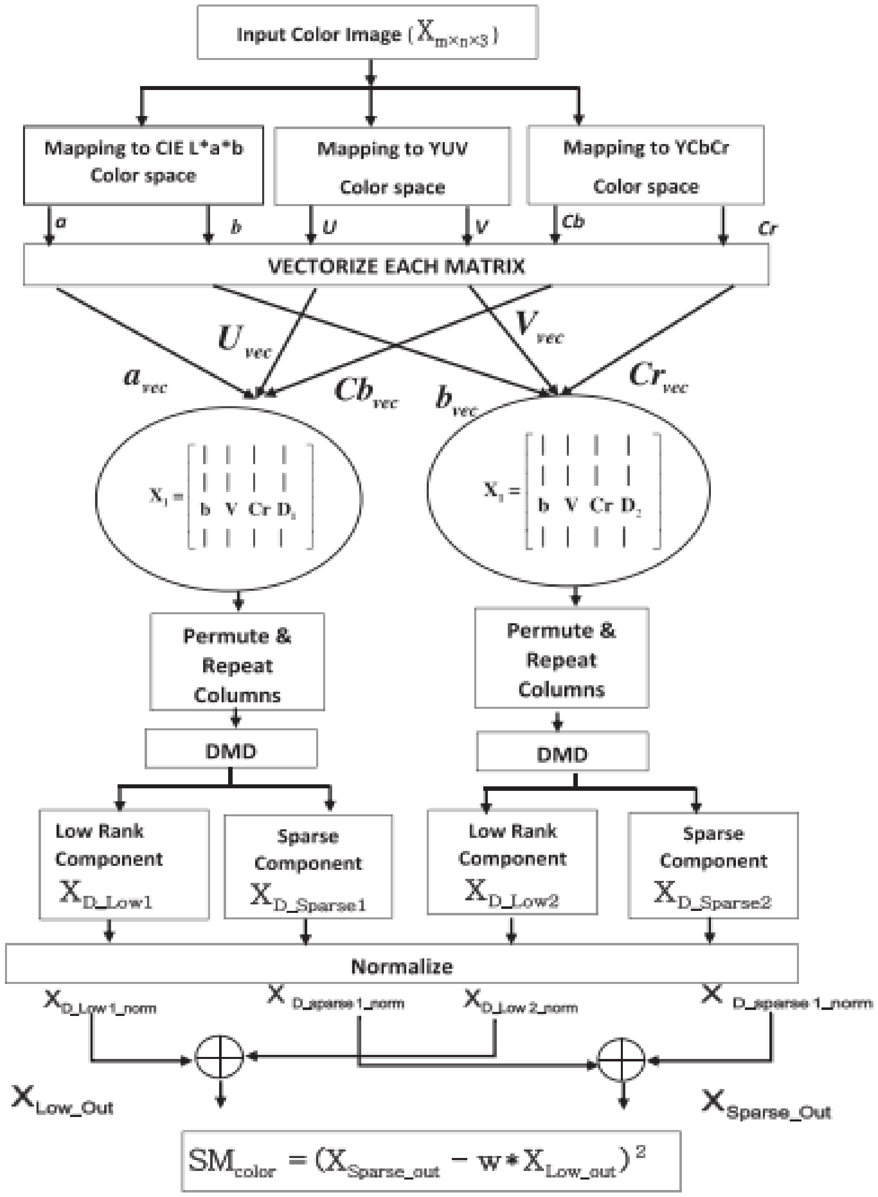
$a = (X_{D\_sparse1} - \min(X_{D\_sparse1}))$

$b = (\max(X_{D\_sparse1}) - \min(X_{D\_sparse1}))$

7. Generate the final color saliency map by combining the respected low-rank and sparse-components and subtract a weighted sum of low rank from the sparse component to extract the final color saliency map.

   $SMcolor = (X_{D\_sparse\_out} - w*X_{low\_out})^2$

The flowchart of the algorithm is depicted below.


Input Color Image ($X_{m\times n\times 3}$)
Mapping to CIE L*a*b Color space
Mapping to YUV Color space
Mapping to YCbCr Color space
a
b
U
V
Cb
Cr
VECTORIZE EACH MATRIX
U_{vec}
V_{vec}
a_{vec}
Cb_{vec}
b_{vec}
Cr_{vec}
$X_1 =$ b V Cr D_1
$X_1 =$ b V Cr D_2
Permute & Repeat Columns
Permute & Repeat Columns
DMD
DMD
Low Rank Component X_{D_Low1}
Sparse Component $X_{D\_Sparse1}$
Low Rank Component X_{D_Low2}
Sparse Component $X_{D\_Sparse2}$
Normalize
$X_{D\_Low1\_norm}$
$X_{D\_sparse1\_norm}$
$X_{D\_Low2\_norm}$
$X_{D\_sparse1\_norm}$
X_{Low_Out}
X_{Sparse_Out}
$SM_{color} = (X_{Sparse\_out} - w*X_{Low\_out})^2$

4.3.2. Luminance based saliency map

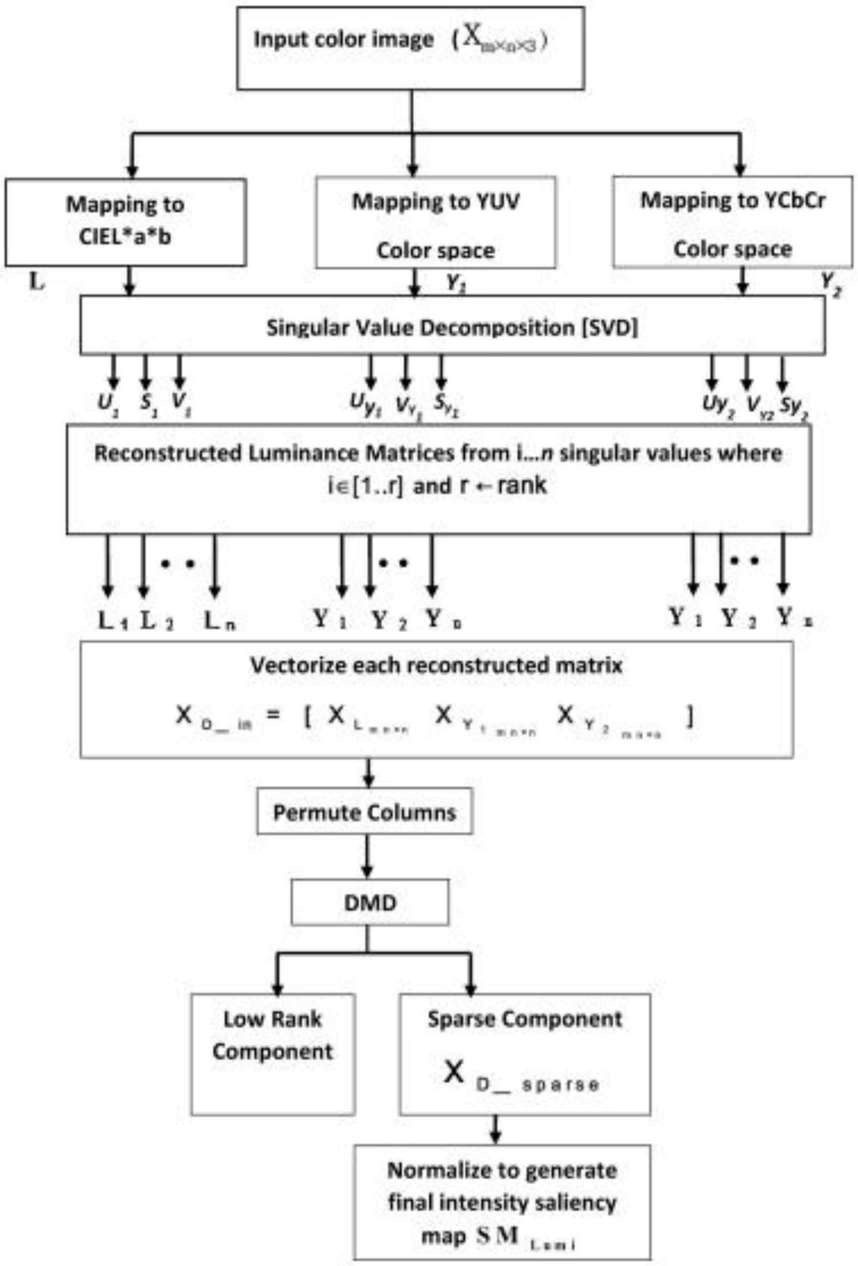
1. The luminance components L from CIEL*a*b and Y from YUV and Y from YCbCr are taken for the luminance saliency map.
2. Perform singular value decomposition for each of the component and decompose into 3 matrices U, S and V.

   2.1. U matrix consists of the eigenvectors of the product of the luminance data matrix and its transpose ($L*L^T$, $Y_1*Y_1^T$, $Y_2*Y_2^T$)

   2.2. V matrix contains the eigenvectors of the product of the transpose of the luminance data and the luminance matrix (L^T*L,$Y_1^T*Y_1$,$Y_2^T*Y_2$)

   2.3. S contains the square root of the eigenvalues of ($L*L^T$, $Y_1*Y_1^T$, $Y_2*Y_2^T$) and (L^T*L,$Y_1^T*Y_1$,$Y_2^T*Y_2$) as diagonal elements in descending order.
3. Reconstruct the luminance data matrix using n intermediate singular values. Intermediate singular values are chosen as they give the salient object.
4. Generate the data matrix by combining all the vectors as column vectors.
5. Permutate the columns of this matrix and feed into the DMD algorithm.
6. Compute the sparse and low rank component from the DMD.
7. Normalize the sparse component and generate the luminance saliency map.

The flowchart of the algorithm is depicted below.



Input color image ($X_{m\times n\times 3}$)
Mapping to CIEL*a*b
Mapping to YUV Color space
Mapping to YCbCr Color space
L
Y_1
Y_2
Singular Value Decomposition [SVD]
U_1 S_1 V_1
U_{Y_1} V_{Y_1} S_{Y_1}
U_{Y_2} V_{Y_2} S_{Y_2}
Reconstructed Luminance Matrices from i…n singular values where i∈[1..r] and r ← rank
L_1 L_2 L_n
Y_1 Y_2 Y_n
Y_1 Y_2 Y_n
Vectorize each reconstructed matrix
$X_{D\_in} = [\, X_{L_{m*n}} \; X_{Y_{1\,m*n}} \; X_{Y_{2\,m*n}} \,]$
Permute Columns
DMD
Low Rank Component
Sparse Component X_{D_sparse}
Normalize to generate final intensity saliency map SM_{Lumi}

4.3.3 Color saliency and luminance saliency map enhancement

The final saliency map can be further enhanced by giving more weight to the salient region and suppressing the non-salient pixels. The spatial refinement is done based on:

1. A salient region in an image is usually not located towards image boundaries
2. A true salient salient object in an initial saliency map usually has a larger number of pixels compared to a non-salient region. The regions that are nearer to the image

boundaries as well as regions with smaller areas are removed from the initial saliency map.

A morphological dilation followed by erosion is performed on the color based saliency map and luminance based saliency map, and a prior saliency map is generated. The structuring element for reconstruction by dilation is obtained by eroding the source image by a kernel **δ** which is determined adaptively as follows:

$$\delta = \alpha * \sqrt{SM_{prior}}$$ where α is a constant set as 5.

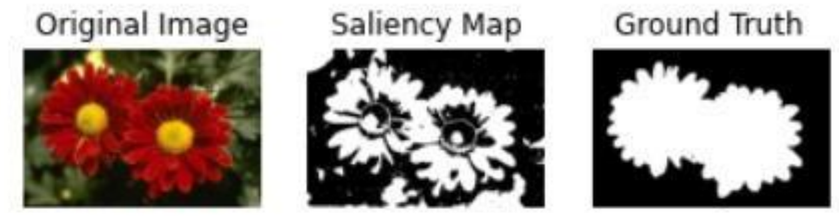
A sigmoid function is used to enhance the contrast between foreground and background. The sigmoid function is stated below.

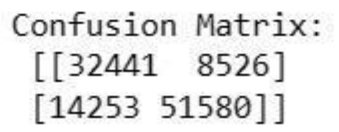
$$\frac{1}{1 + e^{-1} b(x - 0.5)}$$ where b is a constant set as 12.

The final saliency map should have a higher degree of color and intensity. Thus, the maximum value pixel at each spatial location of color on the luminance saliency map is taken.

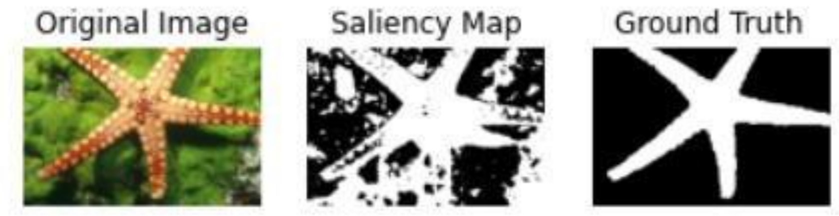
$$SM_{final} = max(SM_{color-enhanced}\blacksquare, SM_{lumi-enhanced})$$

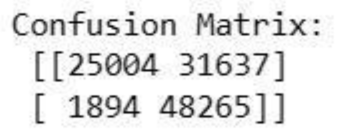
Saliency map generated using DMD Function





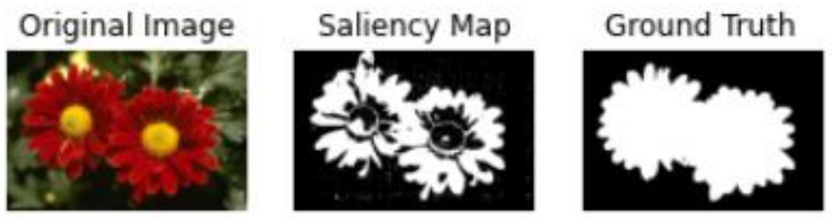
```
Confusion Matrix:
 [[32441  8526]
 [14253 51580]]
```



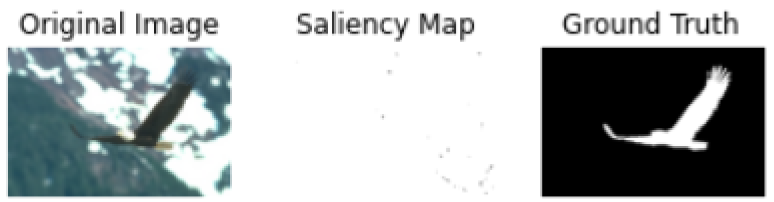


```
Confusion Matrix:
 [[25004 31637]
 [ 1894 48265]]
```

Saliency map generated using RSVD-DMD Function



```
Confusion Matrix:
 [[32150  1153]
 [14544 58953]]
```



```
Confusion Matrix:
 [[ 8178 98484]
 [    0   138]]
```

5. Experimental results and analysis

The performance of the dynamic mode decomposition and its variants is evaluated and compared based on the salient region extraction method. The experiment was carried out on the public datasets available.

5.1. Datasets

In this project, we have used the **ECSSD** (**Extended Complex Scene Saliency Dataset**). It consists of 1000 images, which includes many semantically meaningful and structurally complex images for evaluation. The images that fall into the category of natural images are present here. These images have textures and structures common to real world images. Several examples with their masks are provided. We used 100 random images from this dataset for quantitative comparison.

5.2. Performance evaluation: Quantitative measure: Variants of DMD

The performance of the proposed method is quantitatively analyzed using four standard performance measures. Those include the precision-recall curve, F-measure rate, mean absolute error, area under ROC curve.

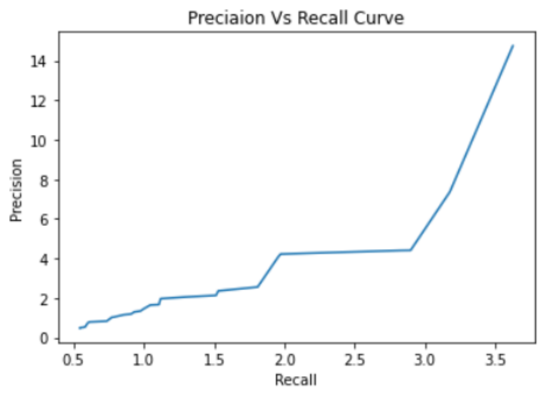
Precision - recall [PR]

Precision is the ratio of the total number of ground truth pixels obtained as a salient region to the total number of salient pixels obtained in the saliency map.

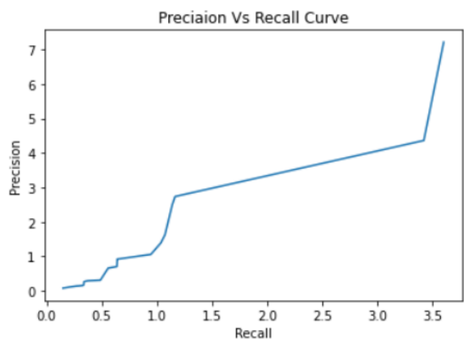
Recall is the ratio of the salient region pixels to the total number of ground truth pixels.

In order to obtain a precision recall curve, the saliency map obtained is binarized with a fixed threshold. The threshold varies from 0 - 255.

Standard DMD:



rSVD DMD:

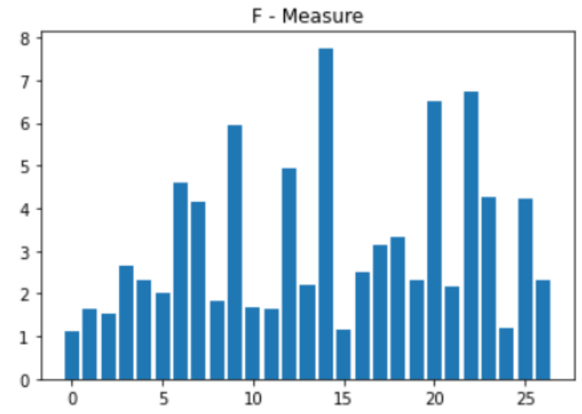


F-measure

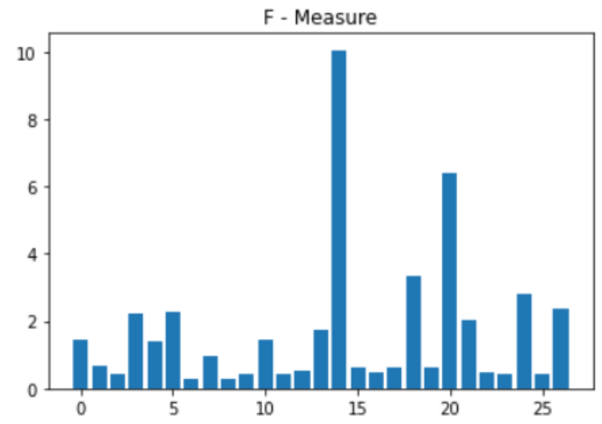
It is the weighted harmonic mean of recall and precision.

Many of the traditional salient object detection methods suggest a value of 0.3 to β^2. This is generally done to give more weightage to precision than recall.

Standard DMD:



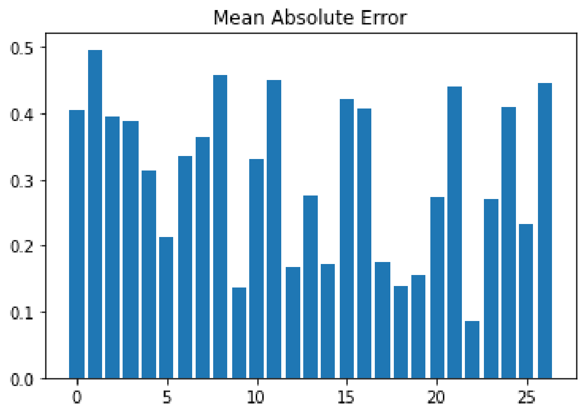
rSVD-DMD:



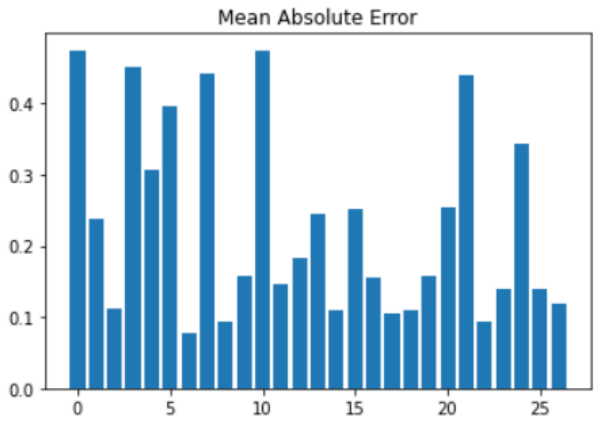
Mean absolute error

The above two metrics do not consider the true-negative saliency detection. Hence their performance is in favor of the saliency models that assign high scores to the salient pixels. They fail to detect the non-salient regions. The mean absolute error between ground truth and the saliency map can be used as a measure to quantize true negative saliency extraction.

Standard DMD:



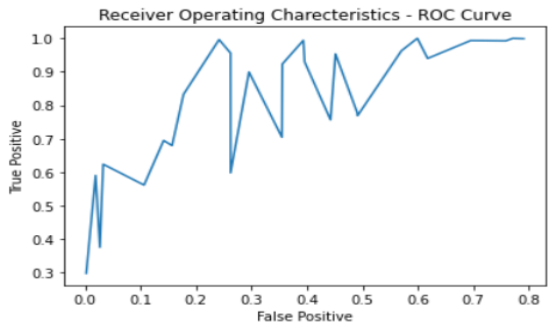
rSVD-DMD:



Area under ROC curve

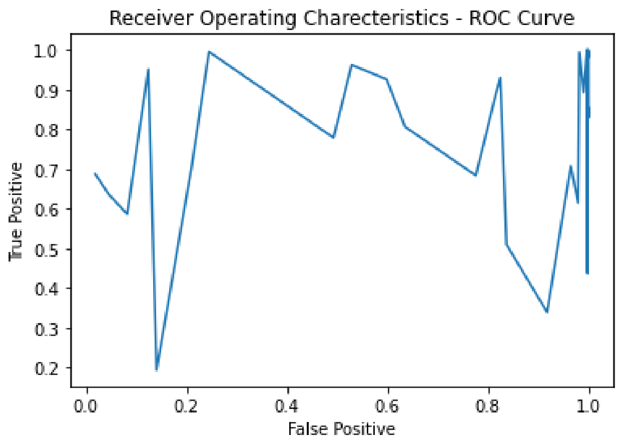
This curve shows the trade-off between false positives and true positives for various thresholds. This AUC is a widely used measure to evaluate the saliency map generated. When the AUC is close to unity, the model is of high accuracy.

Standard DMD:



AUC Measure: 0.873156726805586

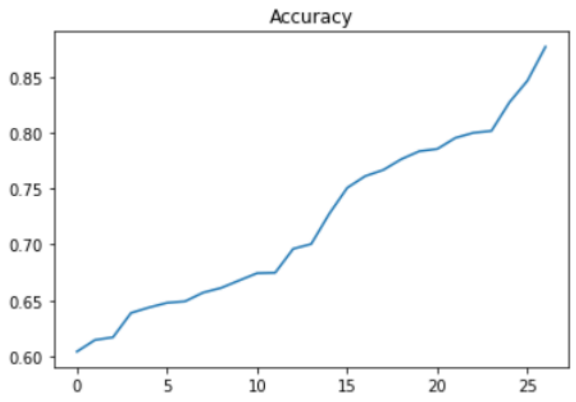
rSVD-DMD:



AUC Measure: 0.8915238046099319

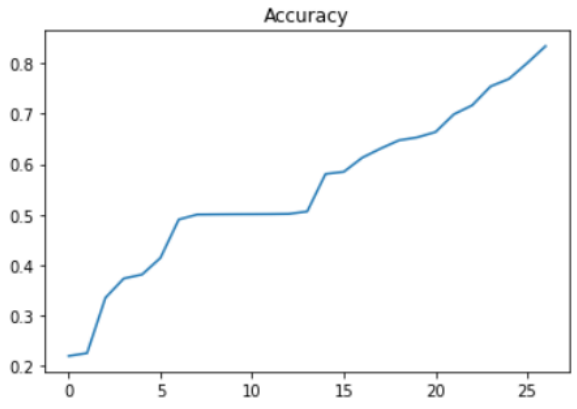
Accuracy

Standard DMD:



```
Maximum Accuracy: 87.7202643083137 %
```

rSVD-DMD:



```
Maximum Accuracy: 83.46712576174889 %
```

5.3. Failure cases

The proposed method in this project is especially targeted for the natural scenes. It widely uses the general information of the images and is suboptimal for the images with highly textures background having the same color distribution as foreground. This method fails when the images have a highly structured background with the same color distributions as the foreground. The dynamic mode decomposition method captures the difference between consecutive columns in the data matrix and the modes bounded away from the origin are detected as the salient region. In scenarios where the background and foreground distributions in the image are highly overlapping and the background is highly textured, the color and intensity-based image representations become undesirable for the application of DMD.

6. Conclusion

In this project, saliency region detection model based on dynamic mode decomposition and its variants is implemented. From the sparse component of the DMD modes, color-based saliency maps and luminance-based saliency maps are constructed to build a full resolution final saliency map. **ECSSD** dataset is used to evaluate the salient region detection in terms of precision-recall curve, F-measure, mean absolute error (MAE), area under ROC curve. Experimental results show that the proposed DMD based model achieves competitive performance and is computationally efficient compared to other traditional saliency detection methods. The DMD based saliency maps can be used efficiently for image classification, object detection and image retrieval applications. We noted that the DMD algorithm is computationally liable and noisy dependent. This phenomenon is reduced using its variants namely, rSVD-DMD and TDMD. From the experiments performed we can conclude that TDMD offers more stable and consistent performance among all other DMD variants. Standard DMD offers satisfactory performance. The worst performance is obtained for rSVD-DMD because it is affected by the fluctuations and uncertainties in the measured data.

7. References

[1] “Intuitive Understanding of Randomized Singular Value Decomposition | by Xinyu Chen (陈新宇).” *Towards Data Science*, https://towardsdatascience.com/intuitive-understanding-of-randomized-singular-value-decomposition-9389e27cb9de. Accessed 17 January 2023.

[2] K P, Soman, and Ramanadhan R. *Digital Signal And Image Processing- The Sparse way*. 1 January 2012.

[3] J, Nathan Kutz, and Stevon L. Brunton. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. 26 January 2017.

[4] Neethu John, et al. *Analysis of Various Color Space Models on Effective Single Image Super Resolution*, August 2016.

APPENDIX

Colour based representation of images

```
import cv2
import matplotlib.pyplot as plt
img1 = cv2.cvtColor(cv2.imread('Color Space Analysis
Dataset//Dog_blur_bg.jpg'), cv2.COLOR_BGR2RGB)
plt.subplot(1,2,1)
plt.imshow(img1)
plt.title('Blur Background')
plt.axis('off')

img2 = cv2.cvtColor(cv2.imread('Color Space Analysis
Dataset//Dog_normal_bg.jpg'), cv2.COLOR_BGR2RGB)
plt.subplot(1,2,2)
plt.imshow(img2)
plt.title('Distinct Background')
plt.axis('off')

plt.show()
# Converting RGB Colorspace to YUV Colorspace
img1_YUV = cv2.cvtColor(img1, cv2.COLOR_BGR2YUV)
Y1,U,V = cv2.split(img1_YUV)

# Converting RGB Colorspace to LAB Colorspace
img1_LAB = cv2.cvtColor(img1, cv2.COLOR_BGR2LAB)
L,a,b = cv2.split(img1_LAB)

# Converting RGB Colorspace to LAB Colorspace
img1_YCrCb = cv2.cvtColor(img1, cv2.COLOR_BGR2YCrCb)
Y2, Cr, Cb = cv2.split(img1_YCrCb)

# Representing the image through various color channels
plt.subplot(3,3,1)
plt.imshow(Y1,cmap="gray")
plt.title('Y1 - channel')
plt.axis('off')
plt.subplot(3,3,2)
plt.imshow(U,cmap="gray")
plt.title('U - channel')
plt.axis('off')

plt.subplot(3,3,3)
plt.imshow(V,cmap="gray")
plt.title('V - channel')
plt.axis('off')

plt.subplot(3,3,4)
plt.imshow(L,cmap="gray")
plt.title('L - channel')
plt.axis('off')

plt.subplot(3,3,5)
plt.imshow(a,cmap="gray")
plt.title('a - channel')
plt.axis('off')

plt.subplot(3,3,6)
plt.imshow(b,cmap="gray")
plt.title('b - channel')
plt.axis('off')

plt.subplot(3,3,7)
plt.imshow(Y2,cmap="gray")
plt.title('Y2 - channel')
plt.axis('off')

plt.subplot(3,3,8)
plt.imshow(Cr,cmap="gray")
plt.title('Cr - channel')
plt.axis('off')

plt.subplot(3,3,9)
plt.imshow(Cb,cmap="gray")
plt.title('Cb - channel')
plt.axis('off')

plt.show()

# Converting RGB Colorspace to YUV Colorspace
img2_YUV = cv2.cvtColor(img2, cv2.COLOR_BGR2YUV)
Y1,U,V = cv2.split(img2_YUV)

# Converting RGB Colorspace to LAB Colorspace
img2_LAB = cv2.cvtColor(img2, cv2.COLOR_BGR2LAB)
L,a,b = cv2.split(img2_LAB)

# Converting RGB Colorspace to LAB Colorspace
img2_YCrCb = cv2.cvtColor(img2, cv2.COLOR_BGR2YCrCb)
Y2, Cr, Cb = cv2.split(img2_YCrCb)

# Representing the image through various color channels
plt.subplot(3,3,1)
plt.imshow(Y1,cmap="gray")
plt.title('Y1 - channel')
plt.axis('off')

plt.subplot(3,3,2)
plt.imshow(U,cmap="gray")
plt.title('U - channel')
plt.axis('off')

plt.subplot(3,3,3)
plt.imshow(V,cmap="gray")
plt.title('V - channel')
plt.axis('off')

plt.subplot(3,3,4)
plt.imshow(L,cmap="gray")
plt.title('L - channel')
plt.axis('off')

plt.subplot(3,3,5)
plt.imshow(a,cmap="gray")
plt.title('a - channel')
plt.axis('off')

plt.subplot(3,3,6)
plt.imshow(b,cmap="gray")
plt.title('b - channel')
plt.axis('off')

plt.subplot(3,3,7)
plt.imshow(Y2,cmap="gray")
plt.title('Y2 - channel')
plt.axis('off')

plt.subplot(3,3,8)
plt.imshow(Cr,cmap="gray")
plt.title('Cr - channel')
plt.axis('off')

plt.subplot(3,3,9)
plt.imshow(Cb,cmap="gray")
plt.title('Cb - channel')
plt.axis('off')

plt.show()
```

STANDARD DMD FUNCTION

```
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np
from pydmd import DMD
from skimage.util import random_noise
def my_dmd(X, Y, truncate=None):
    U2,Sig2,Vh2 = np.linalg.svd(X, False) # SVD of input matrix
    if truncate is None:
#         r = len(Sig2)
        r = 3
    else:
        truncate
    U = U2[:,:r]
    Sig = np.diag(Sig2)[:r,:r]
    V = Vh2.conj().T[:,:r]
    Atil = np.dot(np.dot(np.dot(U.conj().T, Y), V), np.linalg.inv(Sig)) #
build A tilde
    mu,W = np.linalg.eig(Atil)
    Phi = np.dot(np.dot(np.dot(Y, V), np.linalg.inv(Sig)), W) # build DMD
modes
    return mu, Phi
```

RSVD-DMD FUNCTION

```
import cv2
import os
import matplotlib.pyplot as plt
import numpy as np
from pydmd import DMD
from skimage.util import random_noise
def rsvd(A, Omega):
    Y = A @ Omega
    Q, _ = np.linalg.qr(Y)
    B = Q.T @ A
    u_tilde, s, v = np.linalg.svd(B, full_matrices = 0)
    u = Q @ u_tilde
    return u, s, v

def Rsvd_dmd(X, Y, truncate=None):
    #rank = np.linalg.matrix_rank(X)
    #print(rank)
    rank = 3
    Omega = np.random.randn(X.shape[1], rank)
    U2,Sig2,Vh2 = rsvd(X, Omega) # SVD of input matrix
    if truncate is None:
        r = len(Sig2)
    else:
        truncate
    U = U2[:,:r]
    Sig = np.diag(Sig2)[:r,:r]
    V = Vh2.conj().T[:,:r]
    Atil = np.dot(np.dot(np.dot(U.conj().T, Y), V), np.linalg.inv(Sig)) #
build A tilde
    mu,W = np.linalg.eig(Atil)
    Phi = np.dot(np.dot(np.dot(Y, V), np.linalg.inv(Sig)), W) # build DMD
modes
    return mu, Phi
```

SALIENCY DETECTION FUNCTION

```
def saliency_detect(img):

    # Converting RGB Colorspace to YUV Colorspace
    img_YUV = cv2.cvtColor(img, cv2.COLOR_BGR2YUV)
    Y1,U,V = cv2.split(img_YUV)

    # Converting RGB Colorspace to LAB Colorspace
    img_LAB = cv2.cvtColor(img, cv2.COLOR_BGR2LAB)
    L,a,b = cv2.split(img_LAB)

    # Converting RGB Colorspace to LAB Colorspace
    img_YCrCb = cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
    Y2, Cr, Cb = cv2.split(img_YCrCb)

    # ------ Computing Color based saliency map -------

    # Vectoring the color space channels
    a_vec = a.reshape([a.shape[0]*a.shape[1],1])
    b_vec = b.reshape([b.shape[0]*b.shape[1],1])
    U_vec = U.reshape([U.shape[0]*U.shape[1],1])
    V_vec = V.reshape([V.shape[0]*V.shape[1],1])
    Cr_vec = Cr.reshape([Cr.shape[0]*Cr.shape[1],1])
    Cb_vec = Cb.reshape([Cb.shape[0]*Cb.shape[1],1])
    D1_vec = b_vec + V_vec + Cr_vec
    D2_vec = a_vec + U_vec + Cb_vec

    # Static image representation for DMD
    X1 = np.concatenate
((b_vec,V_vec,Cr_vec,D1_vec,V_vec,Cr_vec,b_vec,D1_vec,D1_vec,Cr_vec,V_vec,b_v
ec,V_vec,Cr_vec,D1_vec,b_vec,V_vec,b_vec,D1_vec,Cr_vec),axis=1)
```

```
    X2 =
np.concatenate((a_vec,U_vec,Cb_vec,D2_vec,U_vec,Cb_vec,a_vec,D2_vec,D2_vec,Cb
_vec,U_vec,a_vec,U_vec,Cb_vec,D2_vec,a_vec,U_vec,a_vec,D2_vec,Cb_vec),axis=1)

    S1 = X1[:,0:X1.shape[1]-1]
    S2 = X1[:,1:X1.shape[1]]
    S3 = X2[:,0:X2.shape[1]-1]
    S4 = X2[:,1:X2.shape[1]]

    # Performing DMD on the above matrices
    [eig_val1, Phi1] = Rsvd_dmd(S1, S2, truncate=None)
    [eig_val1, Phi2] = Rsvd_dmd(S3, S4, truncate=None)



    # Seperating the low rank and sparse components
    # Intermediate singular values represent salient part of an image while
smaller and larger values represent non-salient parts
    X_lowRank1 =
np.concatenate((Phi1[:,0].reshape([Phi1.shape[0],1]),Phi1[:,5:len(X1)]),axis=
1)
    X_sparse1 = Phi1[:,2:5]
    X_lowRank2 =
np.concatenate((Phi2[:,0].reshape([Phi2.shape[0],1]),Phi2[:,5:len(X2)]),axis=
1)
    X_sparse2 = Phi2[:,2:5]

    # Normalizing the low rank and sparse matrices
    X_lowRank_norm1 = (X_lowRank1 - np.min(X_lowRank1)) /
(np.max(X_lowRank1)-np.min(X_lowRank1))
    X_sparse_norm1 = (X_sparse1 - np.min(X_sparse1)) / (np.max(X_sparse1)-
np.min(X_sparse1))
    X_lowRank_norm2 = (X_lowRank2 - np.min(X_lowRank2)) /
(np.max(X_lowRank2)-np.min(X_lowRank2))
    X_sparse_norm2 = (X_sparse2 - np.min(X_sparse2)) / (np.max(X_sparse2)-
np.min(X_sparse2))

    # Generating the final sparse and low rank components
    X_sparse_out = X_sparse_norm1 + X_sparse_norm2
    X_lowRank_out = X_lowRank_norm1 + X_lowRank_norm2

    # Generating Saliency Map
    w = 0.1 # Weight
    SM_color_vec = np.real((X_sparse_out - w*X_lowRank_out)**2)

    # ------ Computing Luminance based saliency map -------

    # Computing Singular Value Decomposition to the 3 matrices (L, Y1, Y2) -
Luminance Components
    u_L, s_L, vh_L = np.linalg.svd(L, full_matrices=True)
    u_Y1, s_Y1, vh_Y1 = np.linalg.svd(Y1, full_matrices=True)
    u_Y2, s_Y2, vh_Y2 = np.linalg.svd(Y2, full_matrices=True)

    # Reconstruction of luminance data matrices using n intermediate singular
values

    n = 3 # Number of singular values taken

    # Range of the singular values taken
    r1 = int(u_L.shape[0]/2) # Starting range
    r2 = r1 + n # Ending range

    # Reconstructed data matrices
    L_rec= u_L[:,r1:r2]@np.diag(s_L[r1:r2])@vh_L[r1:r2,:]
    Y1_rec = u_Y1[:,r1:r2]@np.diag(s_Y1[r1:r2])@vh_Y1[r1:r2,:]
    Y2_rec = u_Y2[:,r1:r2]@np.diag(s_Y2[r1:r2])@vh_Y2[r1:r2,:]

    # Vectorize the reconstructed data matrices
    L_rec_vec = L_rec.reshape([L_rec.shape[0]*L_rec.shape[1],1])
    Y1_rec_vec = Y1_rec.reshape([Y1_rec.shape[0]*Y1_rec.shape[1],1])
    Y2_rec_vec = Y2_rec.reshape([Y2_rec.shape[0]*Y2_rec.shape[1],1])

    # Generating data matrix that is to be fed into the DMD
    X =
np.concatenate((L_rec_vec,Y1_rec_vec,Y2_rec_vec,L_rec_vec,Y2_rec_vec,Y1_rec_v
ec,Y1_rec_vec,L_rec_vec,Y2_rec_vec,Y2_rec_vec,L_rec_vec,Y1_rec_vec,Y2_rec_vec
,Y1_rec_vec,L_rec_vec,Y1_rec_vec,Y2_rec_vec,L_rec_vec),axis=1)
```

```
    # Performing DMD for the permutated columns
    S5 = X[:,0:X.shape[1]-1]
    S6 = X[:,1:X.shape[1]]
    [eig_val, Phi] = Rsvd_dmd(S5, S6, truncate=None)

    # Seperating the low rank and sparse components
    # Intermediate singular values represent salient part of an image while
smaller and larger values represent non-salient parts
    X_lowRank =
np.concatenate((Phi[:,0].reshape([Phi.shape[0],1]),Phi[:,3:len(X)]),axis=1)
    X_sparse = Phi[:,2:len(X)]

    # Normalizing the sparse matrix and low rank matrix
    X_lowRank_norm = (X_lowRank - np.min(X_lowRank)) / (np.max(X_lowRank)-
np.min(X_lowRank))
    X_sparse_norm = (X_sparse - np.min(X_sparse)) / (np.max(X_sparse)-
np.min(X_sparse))

    # --------- Generating Saliency Map ----------------
    W = 0 # Weight - Always zero as luminance based saliency map takes only
sparse component of DMD
    SM_luminous_vec = np.real((X_sparse_norm - W*X_lowRank_norm)**2)

    SM_final_vec = []
    for i in range(len(SM_luminous_vec)):
        SM_final_vec.append(max(SM_luminous_vec[i],SM_color_vec[i]))
    SM_final = np.array(SM_final_vec).reshape([img.shape[0],img.shape[1]])

    # Creating a binary mask
    threshhold = 0.5
    binary_mask = np.zeros((SM_final.shape[0],SM_final.shape[1]),dtype=int)
    for i in range(SM_final.shape[0]):
        for j in range(SM_final.shape[1]):
            if SM_final[i,j] > threshhold:
                binary_mask[i,j] = 1
            else:
                binary_mask[i,j] = 0
    return binary_mask
```

Statistical Measures

Precision Recall Function

```
# Precision Recall Function
def PR_score(salient_img,gt_sample):
    total_salient_pixels = (salient_img == 1).sum()
    total_groundTruth_pixels = (gt_sample == 255).sum()
    TP = 0 # True Positive
    for i in range(salient_img.shape[0]):
        for j in range(salient_img.shape[1]):
            if salient_img[i,j].any() == gt_sample[i,j].any():
                TP += 1
    precision = TP/total_salient_pixels
    recall = TP/total_groundTruth_pixels
    return precision, recall
```

F-Measure function

```
# F-Measure function
def F_score(precision,recall):
    beta_2 = 0.3 # Set as 0.3 to give more weight to precision than recall
    F_measure = ((1 + beta_2) * precision * recall) / (beta_2 * (precision +
recall))
    return F_measure
# Generation of Binary Mask
def binary_mask(img):
    threshhold = 0.75
    binary_mask = np.zeros((img.shape[0],img.shape[1]),dtype=int)
    for i in range(img.shape[0]):
        for j in range(img.shape[1]):
            if img[i,j].any() > threshhold:
                binary_mask[i,j] = 1
            else:
                binary_mask[i,j] = 0
    return binary_mask
```

Mean Absolute Error

```
# Mean Absolute Error
def MAE(gt_bin,salient_img_bin):
    W = gt_bin.shape[1]
    H = gt_bin.shape[0]
    mae = np.sum(np.sqrt(np.abs((salient_img - gt_bin)**2)))/(W*H)
    return mae
```

Confusion Matrix

```
def confusion_matrix(salient_image,gt_binary):

    FP = 0 # False Positve
    TP = 0 # True Positive
    TN = 0 # True Negative
    FN = 0 # False Negative

    for i in range(salient_img.shape[0]):
        for j in range(salient_img.shape[1]):
            if salient_img[i,j] == 1 and gt_binary[i,j] == 1:
                TP += 1
            elif salient_img[i,j] == 1 and gt_binary[i,j] == 0:
                FP += 1
            elif salient_img[i,j] == 0 and gt_binary[i,j] == 1:
                FN += 1
            else:
                TN += 1

    confusion_matrix = np.array([[TP,FP],[FN,TN]])

    return confusion_matrix
```

ROC Curve

```
# Receiver Operating Charecteristics - ROC Curve

def conf_mat_rates(conf_mat):
    TP = conf_mat[0,0]
```

```
FP = conf_mat[0,1]
FN = conf_mat[1,0]
TN = conf_mat[1,1]
TPR = TP / (TP + FN) # True Positive Rate
FPR = FP / (FP +TN) # False Positive Rate
FNR = FN / (FN + TP) # False Negative Rate
TNR = TN / (TN + FP) # True Negative Rate
return FPR, TPR, FNR, TNR
```