MAST 680 Project: Dynamic Mode Decomposition (DMD) for Salient Region Detection

Will Sze

April 24, 2023

Abstract

Humans possess incredible ways of analysing a scenery. One of the ways involve identifying with ease the areas that standout. These areas are called salient regions which have been extensively explored in computer vision. This paper explores a method of detecting these regions using dynamic mode decomposition (DMD). The method was developed by Sikha et al in 2016 in which this paper will attempt to recreate the results. The color and luminosity component of the image will be used to help make these regions of interest stand out; followed by DMD low-rank and sparse reconstruction to help identify the entire salient region; and finally, post-processing to enhance the saliency map. The algorithm did not successfully replicate the results of the published article, but it was still able to detect the overall salient regions of some images albeit with some artifacts.

1 Introduction and Overview

When looking at a scenery, human vision has a natural ability to quickly distinguish regions that stand out from the rest. These regions are termed salient regions and have been studied extensively in fields such as psychology and computer vision [1]. Identifying these regions has been attractive in computer vision because of the various applications. Such applications include image retargeting[9] [3], image compression [5], image cropping [12] and many more. Models trying to detect saliency in images can be grouped into two categories: fixation prediction models and salient object detection models. Fixation prediction models, as their name suggests, aim to predict where human fixation will occur. The output from these models is blurry as the border of objects in the image is not well defined, however, it allows us to set a probability of attention in the area with the brightest spot with the highest probability. The first paper that popularized fixation prediction models is that of Itti et al in 1998 who used luminosity and orientation information coupled with a neural network to generate saliency maps [7]. In contrast, salient object detection models aim to identify the entire object of interest, with its borders. The work of Liu et al in 2007, which uses conditional random field (CRF) learning of pre-labelled images, popularized this way of showing saliency [10]. Figure 1 shows example results from the two models just mentioned. Many salient detection algorithms have been developed since the work of Itti and Liu, and all of them can be sorted into two different types of processing. These types are bottom-up and top-down processing. Bottom-up processing uses sensory information such as colour, luminosity and contrasts to distinguish the salient objects from the rest. Top-down processing deals with previously known information to identify these regions of interest. The works of Itti and Liu are prime examples of bottom-up and top-down approaches respectively in their algorithms. A relatively recent and growing salient detection method employing top-down processing is convolutional neural networks allowing a high performance to be achieved. An example is SuperCNN by He et al [6]. More information on the history and background of saliency detection algorithms can be obtained through [2] and [3].

In this paper, dynamic mode decomposition (DMD) will be explored as a possible alternative to help locate the salient objects in static images. The method described in section 3 of this report follows closely the method described by Sikha et al who developed the approach[13]. Thus this report will try to replicate their results to understand this application of DMD. DMD is used primarily as a forecasting method for dynamical systems and originated from the fluid dynamics community [8], but it is also capable of separating background and foreground from a video [4]. Static images pose a problem since there is no temporal component thus a method will be shown to bypass this shortcoming using different colour spaces.

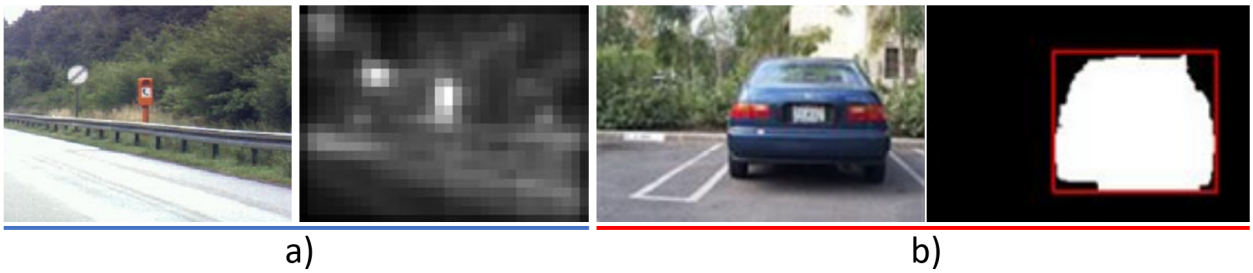

Figure 1: Examples of fixation prediction and salient object detection results from a) Itti et al [7] and b) Liu et al [10] respectively.

2 Theoretical Background

DMD starts with a matrix of data $\mathbf{D} \in \mathbb{R}^{M \times N}$ containing N time instances of M data points. These data points can be organized into two separate matrices $\mathbf{X}$ and $\mathbf{Y}$ where the columns of $\mathbf{Y}$ are the next time steps of the columns of $\mathbf{X}$ meaning

$$Y_n = X_{n+1} \tag{1}$$

where the subscript $n \in \{1, 2, 3, .., N-1\}$ are the columns. DMD allows us to approximately map each column of X to each column of Y through a matrix $\mathbf{A}$ called the DMD matrix such that

$$Y \approx AX. \tag{2}$$

Thus the DMD matrix contains the approximate dynamics of the system to be able to move from one point to the next. Since the X and Y matrices are not square, and the usual inverse operation cannot be applied, the DMD matrix can only be obtained approximately. So, the Moore-Penrose pseudo-inverse is used to be able to find $\mathbf{A}$.

$$A = YX^{\dagger} \tag{3}$$

The † is the pseudoinverse operator and it is computed by taking the singular value decomposition (SVD) and performing an inverse matrix operation such that

$$A = Y(U\Sigma V^*)^{\dagger} = Y(V\Sigma^{\dagger}U^*). \tag{4}$$

Here, U $\in \mathbb{C}^{M \times M}$ and V $\in \mathbb{C}^{N \times N}$ are unitary matrices and Σ is a diagonal matrix containing nonnegative real singular values σ representing the level of importance of each column of U and V. The star notation denotes the conjugate transpose of the matrix. The DMD matrix A contains the underlying dynamics of the collected data points. The matrix can be separated into its eigenvalue μ and eigenvector ψ pairs such that the original data matrix D can be approximately reconstructed using

$$D \approx \sum_{m=1}^{M} c_m \psi_m e^{\omega_m \mathbf{t}} \tag{5}$$

where c_m are constants obtained from initial conditions, $\omega_m = \ln(\mu_m)/\triangle t$ are the frequencies in the fourier mode with $\triangle t$ being the time increment, and $\mathbf{t} \in \mathbb{R}^{1 \times N}$ is a vector of collected time. In practice with image data, the DMD matrix A is usually very large such that the computational task of obtaining the eigenvalues and eigenvectors is extremely heavy. One can perform a similarity transformation to find a matrix $\tilde{A}$ such that it is similar to A. The method is described in [4] and the result is

$$\tilde{A} = U^* Y V \Sigma^{\dagger} \tag{6}$$

where U, Σ and V are again from the SVD of X. If all the singular values are used, which is the case if all the information is needed, then the dimensions of $\tilde{A}$ will be at most $N - 1 \times N - 1$, which is much smaller. Then, the eigenvector and eigenvalue pairs of $\tilde{A}$ can be obtained more easily which are similar by a factor

of U to the eigenvector of the original matrix A. Thus, equation 5 followed by a multiplication by U can be used with the new eigenvectors and eigenvalues to reconstruct data matrix D.

The method of separating background and foreground in video data seen in Assignment 1 and [4] will be used in the sense that the low-rank and sparse components are obtained. To do so, the low rank is reconstructed using $|\omega_p| \approx 0$ such that the pixel values that are slowly drifting are captured. The rest of the frequencies which are further away from the origin are taken to reconstruct the sparse components.

$$D \approx D_{Low} + D_{Sparse} = \mathbf{c}_p \psi_p e^{\omega_p \mathbf{t}} + \sum_{j \neq p} c_j \psi_j e^{\omega_j \mathbf{t}} \tag{7}$$

Here, $p \in \{1, 2, 3, ..., l\}$ where l is the number of frequencies that can be considered close to the origin. Since the reconstruction of the data matrix is only approximate, if the low rank is reconstructed, then the sparse component of a frame can be obtained accurately and non complex using the original data such that

$$D_{Sparse} = D - |D_{Low}| \tag{8}$$

Using equation 8, the error of the reconstruction called the residual error is included in the sparse component and may take form of negative numbers. Thus, these errors can be subtracted from the sparse components to produce a nonnegative matrix which represents correctly the values for pixels.

3 Algorithm Implementation and Development

The algorithms detecting salient regions using DMD was programmed in OCTAVE which is an open source software where most of its syntax is compatible with MATLAB. The source code can be found in appendix B along with the definitions of any pre-made functions used in appendix A. The code is also available to view in Github: https://github.com/SzeCode/MAST-680.git. As mentioned previously, this algorithm follows that of [13]. The algorithm starts with the creation of the data matrices, then the low and sparse components are reconstructed and finally the final saliency map are enhanced with post-processing techniques.

3.1 Data Matrices Creation

Static images are usually stored in RGB format where each letter represents the red, green and blue components of the image respectfully. Each letter contains pixel values ranging from 0 to 255 of that specific colour. Combining the channels together gives the various colours. Images can be stored in other formats as well such as YUV, CIELab and YCbCr. These formats separate the brightness of the pixels from the colour components. As with RGB, these formats contains three channels, however, the first channel of each (L, Y_1, Y_2) contains the luminosity information and the others (U, V, a, b, Cb, Cr) contain the colour information. The salient regions of an image are usually the regions that contain bright distinguishable colours from the rest of the image. Thus the information obtained from these luminosity and colour spaces could be used to help extract the salient regions making the algorithm a bottom-up approach.

The algorithm starts with converting input images into the different colour spaces. From [13], it is shown that depending on the image, different colour channels show different obviousness of the salient region. For an image having a focused background, the channels b, V and Cr contain the salient regions whilst, for blurred background images, channels a, U and Cb show the salient regions. Thus, two groups were made and each channel were vectorized and stored into columns to form two different data matrices, D_1 and D_2. For DMD to perform at its best, the number of frames needed to be increased. The columns of each data matrix were first added to form the fourth column, then the columns were reordered and repeated. Again, [13] showed that performing this step increases the precision of detecting the entire salient region.

To capture the salient regions from the luminosity channels (L, Y_1, Y_2), SVD was applied to each of these channels. By selecting the appropriate singular values, specific details of the image can be reconstructed. In fact, it is shown that the largest singular value (σ_1) reconstructs the background of the image whilst the intermediate singular values (σ_3, σ_4, ...) reconstructs the foreground of the image [11]. Thus intermediate singular values $\sigma_{3:i}$ where $i \in 5, 6, 7, ..., 23$ were selected to reconstruct the images and these images were then vectorized to form the columns of the luminosity data matrix D_L. The number 23 was arbitrarily selected to capture most detail while keeping the D_L smaller for computational ease since each channel contributes to the data matrix.

Algorithm 1: Data Matrix Creation

Import image `Input.jpg`
Convert `Input` RGB into YUV, Lab and YCbCr spaces
$D_1 \leftarrow$ vectorized b, V, Cr channels with permutation and repetition
$D_2 \leftarrow$ vectorized a, U, Cb channels with permutation and repetition
Apply SVD on luminosity components L, Y_1 and Y_2
for $i = 5 : 23$ **do**
  $D_L \leftarrow$ vectorized reconstructed SVD from $\sigma_{3:i}$
end for

3.2 Low and Sparse Reconstruction

To perform DMD and separate the low and sparse components, each data matrix D_1, D_2 and D_L was separated into the two matrices X and Y. $\tilde{A}$ from equation 6 was computed after decomposing X into its singular value components. The s number of singular values was chosen to be as large as possible to increase the effectiveness of the DMD. The eigenvalues and eigenvectors of $\tilde{A}$ were then found using the premade `eig` function. After, the frequencies ω were calculated where the time component is equivalent to the frame count, thus dt is just 1. After finding the constants using the pseudoinverse of the eigenvectors multiplied by the first vector of U^*x, the lowest $|\omega|$ was then taken to reconstruct the low-rank component producing a single frame. At this stage, it is to note that for the colour spaces, [13] have found the low-rank component to produce the non-salient regions whilst in here, the low-rank is observed to capture the salient regions. It is unclear why such discrepancies are present. Moving forward, the non-salient regions (background) were obtained by subtracting the low rank from the grayscale image. All resulting negative numbers were changed to zero. To obtain an initial saliency map for the colour space, the background is subtracted from the low-rank component such that the salient region appears as bright as possible from the rest. When separating the luminosity data matrix D_L into its low and sparse components using the same process, the sparse components contain the salient regions thus nothing more is needed and it can be considered as the initial salient map for the luminosity component.

Algorithm 2: Low and Sparse Reconstruction

Create X and Y matrices for D_1, D_2, D_L
Perform SVD on X
Calculate matrix $\tilde{A}$
Calculate eigenvectors and eigenvalues
Calculate ω
Calculate constants with eigenvectors and X(:,1)
for $i = 1 : s$ **do**
  if $abs(\omega_i) == min(abs(omega))$ **then**
    $D_{s,low} \leftarrow D_{s,low} + c_i\psi_i e^{\omega_i \mathbf{t}}$
  end if
end for
if Color space **then**
  $D_{Sparse} = Grayscale - abs(UD_{s,Low})$
else if Luminosity Space **then**
  $D_{Sparse} = D_L - D_{Low}$
end if

3.3 Post-Processing and Final Saliency Map

The final steps involve post-processing the initial salient maps of the colour and the luminosity space. In each map, a Gaussian function was applied on the image to attribute more weight to the center of the image.

The Gaussian function in 2-dimension with equal standard deviation was taken from *Wolfram Mathworld* and is given by

$$\mathbf{W} = \frac{1}{2\pi\sigma_{xy}^2} e^{\frac{-((x-\bar{x})^2+(y-\bar{y})^2)}{2\sigma_{xy}^2}} \tag{9}$$

where σ_{xy} is the standard deviation, x and y are the pixel positions, and $\bar{x}$ and $\bar{y}$ are the mean pixel positions. In this case, $\bar{x}$ and $\bar{y}$ are just the center of the image. The resulting weight matrix W ranges from 0 to 1. σ_{xy} was set to a fixed value of 175. Equation 9 follows the assumption of [13] which states that salient regions are likely to be away from the edges. This would reduce any artifacts appearing in the image boundaries. The next step is to perform morphological transformations with erosion to get rid of small bright spots and dilatation to recover the edges lost in the true salient region when erosion was performed. The premade functions `imerode` and `imdilate` were used to perform these transformations with a structuring element of a 3x3 matrix with the corners set to 0 and the rest set to 1. Lastly, the intensities were adjusted using a sigmoid function to increase the contrast between bright areas and dark areas. The function used was

$$I_{x,new} = \frac{1}{1 + e^{-b(I_x - \frac{1}{2})}} \tag{10}$$

where I_x is the pixel intensity and b is a parameter that is set to 30. After post-processing the two initial salient maps, the two were combined by choosing the maximum pixel intensity values to form the final salient map.

Algorithm 3: Post-processing and Final Saliency Map Generation

Apply Gaussian distribution to initial colour and luminosity salient maps
Erode edges on both maps
Dilate edges on both maps
Adjust pixel intensity on both maps using sigmoid function
max(Map1,Map2) to combine both maps to form the final salient map

4 Computational Results

The algorithm was applied to the same images as the published article from Sikha et al [13]. The images were obtained from the MSRA-b dataset from [10]. This is a popular dataset used for benchmarking saliency algorithms because the images contain obvious regions of saliency. The authors of [13] also used other datasets to test their algorithm, however they will not be used in this paper. Figure 2 shows the comparison between the saliency maps of the ground truth, of the published article and of the current paper. The ground truth contains the true salient region. The results are not quite the same. The ones of the published articles appear sharper and brighter with regions corresponding well with the actual salient region with the exception of few areas. With the algorithm in this paper, there are multiple artifacts that appear which is not captured the ones in the reference article. There are also regions where the method fails to identify pixels belonging to the salient region for example the airplane in the sign whilst the published article successfully detects it.

There are many areas in the algorithm that might have produced these discrepancies. First, there is an issue with the conversion of the image into different color space. It is unclear if the author performed additional modifications to the images after the conversion. For example, to convert the image into the U channel in YUV, the Y channel needs to be subtracted from blue channel in RGB, then it is multiplied by some weight. If the blue channel is left as a class 'uint8', which supports only integer numbers from 0 to 255, the result is different due to rounding effects than if the channel is converted into a class 'double'. Figure 6 in appendix C shows the differences in the resulting saliency map. The image of the airplane sign performs better with conversion to 'double' whereas the image of a knob performs better when the channel is kept at 'uint8'. Thus, two different methods (conversion into 'double' or not) could have improvements in the saliency map for different set of images. An additional example related to color space conversion is shown in figure 7 where the algorithm performs best on certain images when the Cb and Cr channels are normalized with

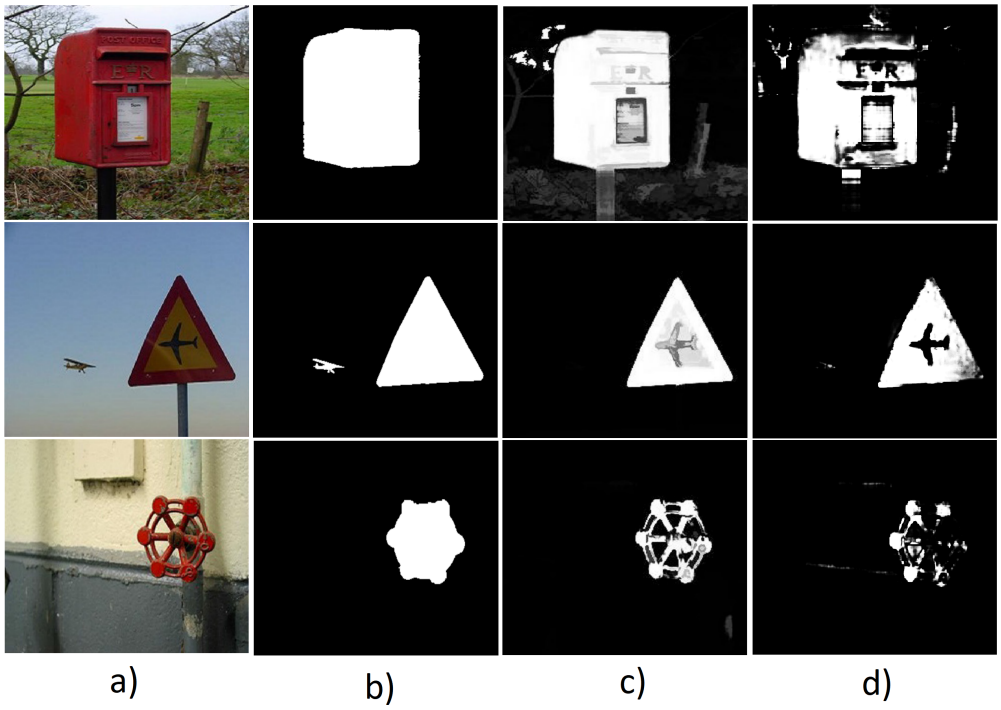

Figure 2: Comparison between saliency maps of a) the ground truth, b) of the published article and c) of the current paper.

minimum of 0 and maximum of 255, but on other images, the algorithm performs best when the channels are not normalized. Another area resulting in different saliency maps is in the low and sparse reconstructions, which is the most probable case. In fact, as mentioned in section 3.2, the published article noted that the sparse reconstruction contains the salient regions, whereas the low rank reconstruction contains the non-salient region. However, in this paper, it was always observed that the low-rank reconstruction captures the salient region. It is unclear why there is a difference and what the authors did to achieve this. Although, it seems to make sense to capture the salient region in the low-rank reconstruction. If the channels in the data matrix contain the same salient regions then the pixel values stays roughly at the same intensity throughout the different frames which is similar to capturing the non changing background from a video. Finally, the authors of the published articles may have performed more sophisticated post-processing to enhance their results. More advanced post-processing could not be explored in this paper due to time constraints.

The algorithm of the current paper was also tested with other images of the MSRA-b dataset to verify its performance and limitations. Figure 3 shows some of the results that were considered decent. In addition, a different saliency algorithm called Graph-based Manifold Ranking (GMR) [14] was also tested on the same images to compare the results. This algorithm uses superpixels and checks where the pixels are similar (or dissimilar) to the background to identify the foreground. This is the only other algorithm used for comparison since the source code is readily available and can be run easily without any other tools. Other algorithms were hard to find and were not able to run successfully without knowing the correct configurations. Comparing the results, the saliency maps obtained from DMD (row c in figure 3) show that the algorithm is capable of identifying these regions of interest. Again, there are still many artifacts or regions in the image where the algorithm considers salient but the ground truth does not. Comparing the DMD to the GMR saliency maps, the GMR outperforms the DMD in many ways. It captures more accuratly the salient regions when comparing qualitatively to the ground truth. However, DMD and GMR have some similar areas where both fail to detect saliency. For example, the flower in the 5^{th} column has filaments in which the ground truth propose as salient but they are missed by DMD and GMR. The same is true for the tires and the windows in the upside down car in column 6. Even thought the figure shows a relatively poor performance from DMD compared to GMR, DMD may perform well compared to other algorithms if the results from the published

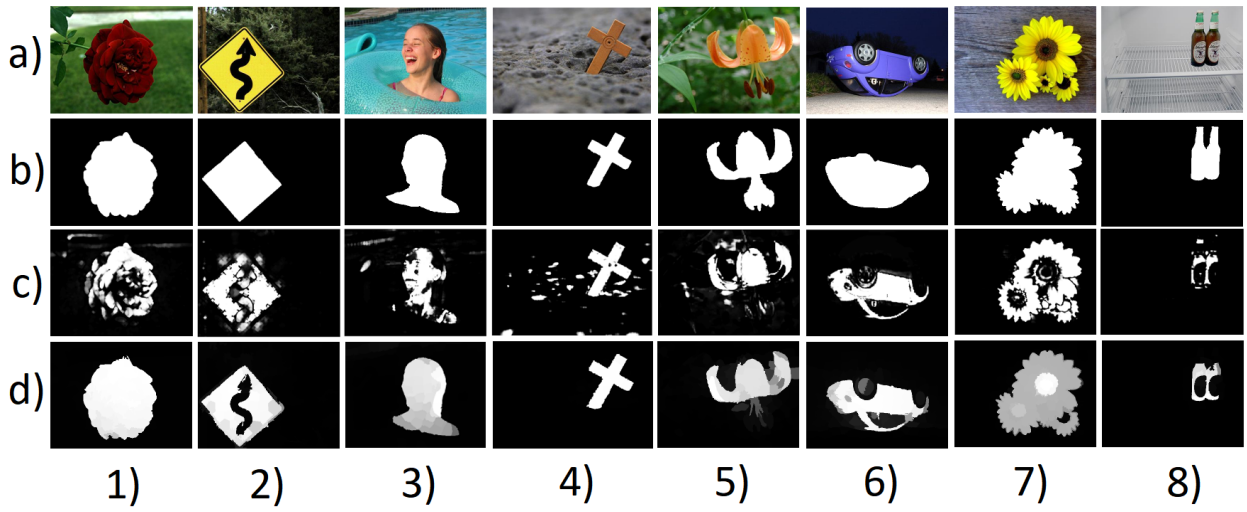

Figure 3: Images resulting in decent performance from DMD with columns as a) the original image, b) ground truth, c) DMD of this paper and d) GMR for comparison.

article was successfully reproduced (see [13]).

The DMD algorithm of the current paper does not always produce clear salient regions. In fact, the images from figure 3 contain similar colored pixels throughout the background. However, when the background is detailed or contains the same color as the foreground, then the algorithm fails. Figure 8 shows the kind of images where the algorithm performs poorly. In the first image (row 1), the background may have similar colors, however the algorithm detects the ripples as part of the salient region instead of the obvious white splash. This is due to the color spaces in which DMD is based on. The algorithm tends to favor colors rather than shades of white. Meaning the more saturated the color is, the more salient the region. Thus if the background contains color, and the foreground contains shades of white, then the algorithm can misinterpret the background as the foreground. In the second row, another problem arises. If the background contains the same color as the as foreground, then the algorithm also fails even though there is a clear outline separating the two objects. This is because the algorithm has no method of detecting outlines and thus two similarity colored objects could be considered as one entire salient region even though only one should be. Lastly, the algorithm fails to detect images that resemble the one in the 3^{rd} row which contain detailed background with intense lightings and large contrasts. In fact, when performing SVD on the luminosity channels, the intermediate singular values captures parts of the image containing high details. Thus if the background is cluttered, SVD will detect them. In addition, it does not help if the background has brighter colors than the foreground. More examples can be found in appendix C. GMR was also tested for comparison. Not surprisingly, it performed substantially better than DMD since it can detect the outline of the foreground. However, there are still similar areas where it has some trouble for example the in the second row where colors in the background are similar.

4.1 Future works

The comparisons made in this paper involve qualitative observations between the results. There are many ways the performance of the algorithm can be evaluated quantitatively such as comparing the saliency maps with subjective annotation using the precision-recall (PR) and the receiver operating characteristics (ROC) [1]. Thus, the next logical step would be to evaluate these metrics on the algorithm in this paper against multiple other algorithms to compare more objectively the saliency maps. In Sikha's published paper, they observed a higher precision level for their method compared to 13 out of 14 algorithms [13]. It would be interesting to see how the algorithm from here fairs against the ones compared in the published paper. Another area that requires improvement is the computational speed. In fact the average computational time to run the algorithm on images of the MSRA-b dataset was around 5.74 seconds whilst the time to run

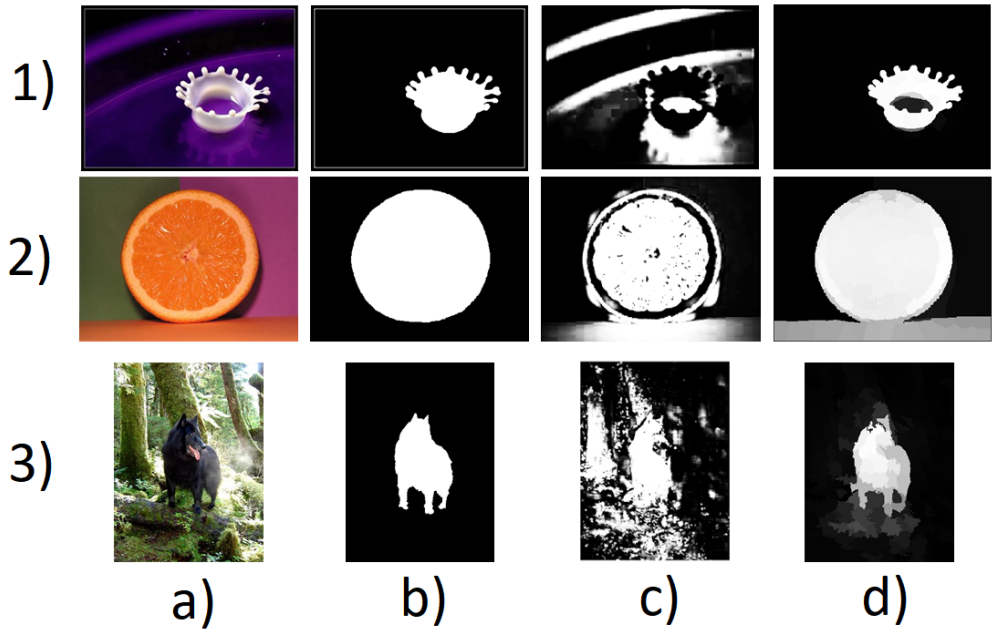

Figure 4: Images resulting in poor performance from DMD with columns as a) original image, b) ground truth, c) DMD of this paper and d) GMR for comparison.

the GMR dataset was around 1.96 seconds in OCTAVE. The code implemented in this paper has not been optimized, thus there are still room for improvement. The computation time from Sikha's DMD algorithm is stated to be around 1.604 seconds, however this is run in MATLAB which generally is faster than OCTAVE thus it would not be a fair comparison to the 2 previous values. As a final note, a saliency algorithm can also be accompanied with an image segmentation algorithm. This means that the pixels of the original colored image is made visible when they are salient whilst the rest of the pixels are black. A preliminary version is shown in figure 5. They were simply created by turning on the pixels in the original image when the corresponding pixels in the saliency map are above a threshold and turning off the pixels in the original image when the corresponding pixels in the saliency map are below the threshold. However, a more sophisticated way of creating these segmented images could be used such as the one shown in the published article [13].

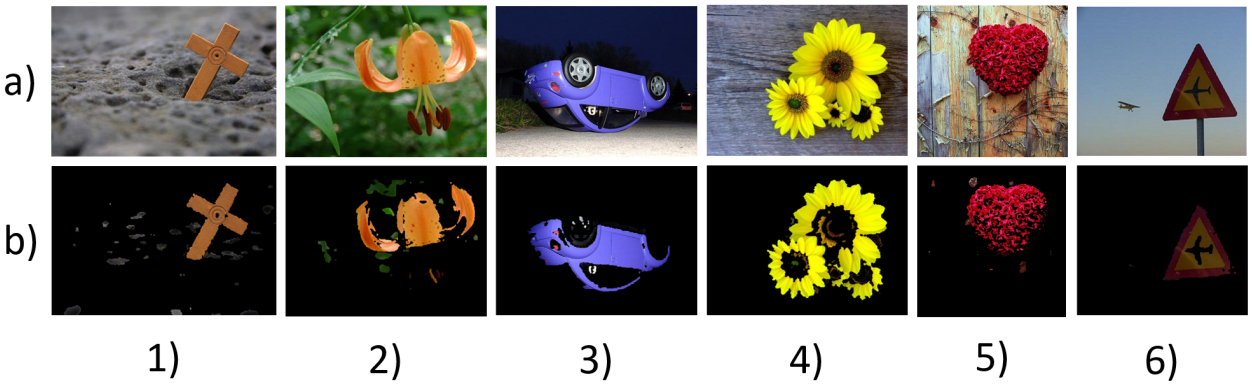

Figure 5: Examples of 6 images segmented from the saliency maps using a simple on/off algorithm with a) the original and b) the segmented images.

5 Summary and Conclusions

To summarize, saliency in an image is the region where human vision tends to look at first. Many algorithms have been developed to attempt to detect these regions. In this paper, DMD was explored as a possible alternative to identify these salient regions. A bottom-up approach was used where the RGB images were converted into various different color spaces to identify areas having higher saturation and brightness. DMD was used to help separate the low and sparse components or, in this case salient and non-salient regions. Some post-processing was also used to enhance the saliency maps obtained. Results showed discrepancies with the reference published articles, but was still able to identify, for the most part, the salient region. Good and bad results was shown to demonstrate the capabilities and limitations of the method. DMD is just one of many tools to detect the salient regions. Although results of the reference article were not successfully recreated, the results shown here seem promising. There are many ways the algorithm could be improved such as more sophisticated post-processing. These could be explored in future work, and once the results are successfully recreated, it would be interesting to see how the performance compares to newer algorithms.

References

[1] Ali Borji et al. "Salient Object Detection: A Benchmark". In: *IEEE Transactions on Image Processing* 24.12 (2015), pp. 5706–5722. DOI: 10.1109/TIP.2015.2487833.

[2] Ali Borji et al. "Salient object detection: A survey". In: *Computational Visual Media* 5.2 (June 2019), pp. 117–150. DOI: 10.1007/s41095-019-0149-9. URL: https://doi.org/10.1007%5C%2Fs41095-019-0149-9.

[3] Runmin Cong et al. "Review of Visual Saliency Detection With Comprehensive Information". In: *IEEE Transactions on Circuits and Systems for Video Technology* 29.10 (Oct. 2019), pp. 2941–2959. DOI: 10.1109/tcsvt.2018.2870832. URL: https://doi.org/10.1109%5C%2Ftcsvt.2018.2870832.

[4] Jacob Grosek and J. Nathan Kutz. *Dynamic Mode Decomposition for Real-Time Background/Foreground Separation in Video*. 2014. DOI: 10.48550/ARXIV.1404.7592. URL: https://arxiv.org/abs/1404.7592.

[5] Sunhyoung Han and Nuno Vasconcelos. "Image Compression using Object-Based Regions of Interest". In: *2006 International Conference on Image Processing*. 2006, pp. 3097–3100. DOI: 10.1109/ICIP.2006.313095.

[6] Shengfeng He et al. "SuperCNN: A Superpixelwise Convolutional Neural Network for Salient Object Detection". In: *International Journal of Computer Vision* 115 (Apr. 2015). DOI: 10.1007/s11263-015-0822-0.

[7] L. Itti, C. Koch, and E. Niebur. "A model of saliency-based visual attention for rapid scene analysis". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.11 (1998), pp. 1254–1259. DOI: 10.1109/34.730558.

[8] Jason J. Bramburger. *Data-Driven Methods for Dynamic Systems*. Concordia University, 2023.

[9] Jianjun Lei et al. "Depth-Preserving Stereo Image Retargeting Based on Pixel Fusion". In: *IEEE Transactions on Multimedia* 19.7 (2017), pp. 1442–1453. DOI: 10.1109/TMM.2017.2660440.

[10] Tie Liu et al. "Learning to Detect A Salient Object". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. 2007, pp. 1–8. DOI: 10.1109/CVPR.2007.383047.

[11] Xiaolong Ma et al. "Saliency Detection Based on Singular Value Decomposition". In: *Journal of Visual Communication and Image Representation* 32 (Oct. 2015), pp. 95–106. DOI: 10.1016/j.jvcir.2015.08.003.

[12] Anthony Santella et al. "Gaze-Based Interaction for Semi-Automatic Photo Cropping". In: New York, NY, USA: Association for Computing Machinery, 2006. ISBN: 1595933727. DOI: 10.1145/1124772.1124886. URL: https://doi.org/10.1145/1124772.1124886.

[13] O.K. Sikha, S. Sachin Kumar, and K.P. Soman. "Salient region detection and object segmentation in color images using dynamic mode decomposition". In: *Journal of Computational Science* 25 (2018), pp. 351–366. ISSN: 1877-7503. DOI: https://doi.org/10.1016/j.jocs.2017.07.007. URL: https://www.sciencedirect.com/science/article/pii/S1877750317308049.

[14] Chuan Yang et al. "Saliency Detection via Graph-Based Manifold Ranking". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 2013, pp. 3166–3173. DOI: 10.1109/CVPR.2013.407.

Appendix A Premade Functions Used

- `rgb2gray(Img)` converts each pixel of `Img` containing RGB values to a single value by a weighted sum. These values represent the grayscale of each pixel.
- `rgb2lab(Img)` converts each pixel of `Img` containing RGB values into the CIELab color space where the third dimension contains the L, a and b channels.
- `rgb2ycbcr(Img)` converts each pixel of `Img` containing RGB values into the YCbCr color space where the third dimension contains the Y, Cb and Cr channels.
- `cast(M,'class')` assigns the values in matrix M to a class type 'class'. Classes can be 'double', 'uint8', etc.
- `[U,S,V] = svd(X,'econ')` returns the unitary matrices `U` and `V`, and the singular values of matrix X. The argument `'econ'` removes all zeros rows or columns to make the matrices compact.
- `reshape(X, arg1, arg2, ...)` turns a matrix or vector X into a matrix with `arg1` rows and `arg2` columns etc.
- `[V, D] = eig(A)` returns the eigenvectors of matrix `A` in each columns of `V` and a diagonal matrix `D` containing the eigenvalues of matrix `A`.
- `pinv(X)` returns the pseudoinverse ($V\Sigma^{-1}U^*$) of `X` matrix.
- `imerode(Img,nhood)` uses the structuring element `nhood` to erode the grayscale or binary image `Img`.
- `imdilate(Img,nhood)` uses the structuring element `nhood` to dilate the grayscale or binary image `Img`.

Appendix B Octave Code

B.1 Main Program Code

```
%Main program for DMD Saliency Algorithm
clear all; close all; clc;

pkg load image;
Funcdir='./Functions/';
addpath(Funcdir);
Funcdir='./Functions/Sub1';
addpath(Funcdir);

pause(0.1);

tic

OriginalImage = imread("./Input/0_1_1650.jpg");

```

```
%Display original
figure();
imshow(OriginalImage);
title("RGB Image");

%Color Space Conversion and Data matrix Creation
Display = 0;
  [GrayscaleImage,N,M,X_color1,X_color2] = ColorDataf(OriginalImage,
     Display);
Display = 0;
  X_Lum = LumiDataf(OriginalImage,GrayscaleImage,M,N,Display);
  [M_xLum, N_xLum] = size(X_Lum);

%DMD Low and Sparse Separation
[X_DLow1, X_DSparse1] = ColorDMD2f(X_color1, GrayscaleImage, M, N);
[X_DLow2, X_DSparse2] = ColorDMD2f(X_color2, GrayscaleImage, M, N);
[X_DLow_Lum, X_DSparse_Lum] = LumiDMDf(X_Lum, M_xLum, N_xLum);

%Post-processing and final saliency map
Display = 0;
  SM_colorview = ColorFinal2f(X_DLow1, X_DLow2, X_DSparse1, X_DSparse2, M,
      N, Display);
  SM_LUMVIEW = LumiFinalf(X_DLow_Lum, X_DSparse_Lum, M, N);

Display = 0;
  SM_FIN = SMFinalMapf(SM_colorview, SM_LUMVIEW, M, N, Display);

%Segmentation
T = 100;
IMSeg = Segmentation(OriginalImage, SM_FIN, M, N, T);


%Display final saliency map and segmentation results
figure();
  SM_FIN = cast(SM_FIN, 'uint8');
  imshow(SM_FIN);
  title("SM FINAL");

figure();
  IMSeg = cast(IMSeg, 'uint8');
  imshow(IMSeg);
  title("Image Segmented");

toc
```

B.2 Color Space Conversion and DMD data matrix creation

```
function [GrayscaleImage,N,M,X_color1,X_color2] = ColorDataf(OriginalImage
   ,display)
%Color Space Conversion and DMD Matrix Creation

  GrayscaleImage = rgb2gray(OriginalImage);
  [M,N] = size(GrayscaleImage);
```

```
6
7   if(display == 1)
8     figure();
9     imshow(GrayscaleImage);
10    title("Gray Image");
11  end
12
13  %Image stored as R, G, B
14  R = OriginalImage(:,:,1);
15  G = OriginalImage(:,:,2);
16  B = OriginalImage(:,:,3);
17
18  %----------------------- COLOR SPACE CONVERSION -----------------------%
19  %YUL
20  R_ = cast(R,'double');
21  G_ = cast(G,'double');
22  B_ = cast(B,'double');
23
24  Y = 0.299*R_+0.587*G_+0.114*B_;
25
26  U = abs((B_-Y)*0.492);
27  Color_max = cast(max(max(U)),'double');
28  Color_min = cast(min(min(U)),'double');
29  IntCast =  cast(U,'double');
30  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
31  U = cast(IntCast,'uint8');
32
33  V = abs((R_-Y)*0.877);
34  Color_max = cast(max(max(V)),'double');
35  Color_min = cast(min(min(V)),'double');
36  IntCast =  cast(V,'double');
37  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
38  V = cast(IntCast,'uint8');
39
40  if(display == 1)
41    Y = cast(Y,'uint8');
42    figure();
43    imshow(Y);
44    title("Y Image");
45
46    U = cast(U,'uint8');
47    figure();
48    imshow(U);
49    title("U Image");
50
51    V = cast(V,'uint8');
52    figure();
53    imshow(V);
54    title("V Image");
55  end
56
57
58  %LAB
59  Lab = rgb2lab(OriginalImage);
```

```
%%  L = Lab(:,:,1);
%%  Color_max = cast(max(max(L)),'double');
%%  Color_min = cast(min(min(L)),'double');
%%  IntCast =  cast(L,'double');
%%  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
%%  L_ = cast(IntCast,'uint8');

  a = abs(Lab(:,:,2));
  Color_max = cast(max(max(a)),'double');
  Color_min = cast(min(min(a)),'double');
  IntCast =  cast(a,'double');
  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
  a = cast(IntCast,'uint8');

  b = abs(Lab(:,:,3));
  Color_max = cast(max(max(b)),'double');
  Color_min = cast(min(min(b)),'double');
  IntCast =  cast(b,'double');
  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
  b = cast(IntCast,'uint8');

  if(display == 1)
%%    figure();
%%    imshow(L_);
%%    title("L Image");

    figure();
    imshow(a);
    title("a Image");

    figure();
    imshow(b);
    title("b Image");
  end

  %YCBCR
  ycbcr = rgb2ycbcr(OriginalImage);
%%  y = ycbcr(:,:,1);
%%  Color_max = cast(max(max(y)),'double');
%%  Color_min = cast(min(min(y)),'double');
%%  IntCast =  cast(y,'double');
%%  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
%%  y_ = cast(IntCast,'uint8');


  cb = abs(ycbcr(:,:,2));
  Color_max = cast(max(max(cb)),'double');
  Color_min = cast(min(min(cb)),'double');
  IntCast =  cast(cb,'double');
  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
  cb_ = cast(IntCast,'uint8');

  cr = abs(ycbcr(:,:,3));
  Color_max = cast(max(max(cr)),'double');
```

```
  Color_min = cast(min(min(cr)),'double');
  IntCast =  cast(cr,'double');
  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
  cr_ = cast(IntCast,'uint8');

  if(display == 1)
%%    figure();
%%    imshow(y);
%%    title("y Image");

    figure();
    imshow(cb);
    title("cb Image");

    figure();
    imshow(cr);
    title("cr Image");
  end


  %---------------------- DATA MATRIX CREATION ----------------------%

  X_color1 = zeros(M*N,40);

  X_color1(:,0+1) = reshape(b,M*N,1);
  X_color1(:,0+2) = reshape(V,M*N,1);
  X_color1(:,0+3) = reshape(cr,M*N,1);
  X_color1(:,0+4) = X_color1(:,1)+X_color1(:,2)+X_color1(:,3);

  X_color1(:,4+1) = X_color1(:,0+2);
  X_color1(:,4+2) = X_color1(:,0+3);
  X_color1(:,4+3) = X_color1(:,0+1);
  X_color1(:,4+4) = X_color1(:,0+4);

  X_color1(:,8+1) = X_color1(:,0+4);
  X_color1(:,8+2) = X_color1(:,0+3);
  X_color1(:,8+3) = X_color1(:,0+2);
  X_color1(:,8+4) = X_color1(:,0+1);

  X_color1(:,12+1) = X_color1(:,0+2);
  X_color1(:,12+2) = X_color1(:,0+3);
  X_color1(:,12+3) = X_color1(:,0+4);
  X_color1(:,12+4) = X_color1(:,0+1);

  X_color1(:,16+1) = X_color1(:,0+2);
  X_color1(:,16+2) = X_color1(:,0+1);
  X_color1(:,16+3) = X_color1(:,0+4);
  X_color1(:,16+4) = X_color1(:,0+3);

  X_color1(:,21:40) = X_color1(:,1:20);

  X_color2 = zeros(M*N,40);

  X_color2(:,1) = reshape(a,M*N,1);
```

```
  X_color2(:,2) = reshape(U,M*N,1);
  X_color2(:,3) = reshape(cb,M*N,1);
  X_color2(:,4) = X_color2(:,1)+X_color2(:,2)+X_color2(:,3);

  X_color2(:,4+1) = X_color2(:,0+2);
  X_color2(:,4+2) = X_color2(:,0+3);
  X_color2(:,4+3) = X_color2(:,0+1);
  X_color2(:,4+4) = X_color2(:,0+4);

  X_color2(:,8+1) = X_color2(:,0+4);
  X_color2(:,8+2) = X_color2(:,0+3);
  X_color2(:,8+3) = X_color2(:,0+2);
  X_color2(:,8+4) = X_color2(:,0+1);

  X_color2(:,12+1) = X_color2(:,0+2);
  X_color2(:,12+2) = X_color2(:,0+3);
  X_color2(:,12+3) = X_color2(:,0+4);
  X_color2(:,12+4) = X_color2(:,0+1);

  X_color2(:,16+1) = X_color2(:,0+2);
  X_color2(:,16+2) = X_color2(:,0+1);
  X_color2(:,16+3) = X_color2(:,0+4);
  X_color2(:,16+4) = X_color2(:,0+3);

  X_color2(:,21:40) = X_color2(:,1:20);

end
```

B.3 Luminosity Image Conversion and DMD data matrix creation

```
function X_Lum = LumiDataf(OriginalImage,GrayscaleImage,M,N,display)
%Luminosity Image Conversion and DMD Matrix Creation

  %Image stored as R, G, B
  R = OriginalImage(:,:,1);
  G = OriginalImage(:,:,2);
  B = OriginalImage(:,:,3);

  %----------------------- Luminosity SPACE CONVERSION
     ----------------------%
  %YUL
  R_ = cast(R,'double');
  G_ = cast(G,'double');
  B_ = cast(B,'double');

  Y = 0.299*R_+0.587*G_+0.114*B_;

  if(display == 1)
    Y = cast(Y,'uint8');
    figure();
    imshow(Y);
    title("Y Image");
  end
```

```
23
24
25  %LAB
26  Lab = rgb2lab(OriginalImage);
27  L = Lab(:,:,1);
28  Color_max = cast(max(max(L)),'double');
29  Color_min = cast(min(min(L)),'double');
30  IntCast =  cast(L,'double');
31  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
32  L = cast(IntCast,'uint8');
33
34  if(display == 1)
35    figure();
36    imshow(L);
37    title("L Image");
38  end
39
40  %YCBCR
41  ycbcr = rgb2ycbcr(OriginalImage);
42  y = ycbcr(:,:,1);
43  Color_max = cast(max(max(y)),'double');
44  Color_min = cast(min(min(y)),'double');
45  IntCast =  cast(y,'double');
46  IntCast = (IntCast-Color_min)/(Color_max-Color_min)*(255.0);
47  y_ = cast(IntCast,'uint8');
48
49  if(display == 1)
50    figure();
51    imshow(y);
52    title("y Image");
53  end
54
55  %----------------------- LUMINOSITY IMAGES -----------------------%
56  L = cast(L,'double');
57  Y = cast(Y,'double');
58  y = cast(y,'double');
59
60  [U_L,S_L,V_L] = svd(L,'econ');
61  [U_Y1,S_Y1,V_Y1] = svd(Y,'econ');
62  [U_Y2,S_Y2,V_Y2] = svd(y,'econ');
63
64  X_L = [];
65  X_Y1 = [];
66  X_Y2 = [];
67  X_Lum = [];
68
69  EnergyTarget = 0.95;
70  StartSingular = 3;
71  EndSingular = 5;
72  while(EndSingular-StartSingular<20)
73    EndSingular = SRankApprox(EnergyTarget, S_L);
74    EnergyTarget = EnergyTarget+0.0001;
75  end
76  L_sum = zeros(M,N);
```

```
77
78    if(display == 1)
79      figure();
80    end
81    for i = StartSingular+2:EndSingular
82      s_ = StartSingular:i;
83      %s_ = i:i;
84
85      L_SVD = U_L(:,s_)*S_L(s_,s_)*V_L(:,s_)';
86      X_L = [X_L, reshape(L_SVD,M*N,1)];
87
88      L_sum = L_sum + L_SVD;
89
90      if(display == 1)
91        L_SVD = cast(L_SVD, 'uint8');
92        imshow(L_SVD);
93        title("L Int Singular Reconst Image");
94        pause(0.1)
95
96
97        if(i == 2 || i == 5 || i == 7 || i == 10)
98          %nfigure = nfigure + 1;
99          figure();
100       end
101     end
102
103   end
104   %fprintf('iL = %d\n',i');
105   X_Lum = [X_Lum, X_L];
106
107   if(display == 1)
108     figure();
109   end
110   for i = StartSingular+2:EndSingular
111     s_ = StartSingular:i;
112     %s_ = i:i;
113
114     Y1_SVD = U_Y1(:,s_)*S_Y1(s_,s_)*V_Y1(:,s_)';
115     X_Y1 = [X_Y1, reshape(Y1_SVD,M*N,1)];
116
117     if(display == 1)
118       Y1_SVD = cast(Y1_SVD, 'uint8');
119       imshow(Y1_SVD);
120       title("Y1 Int Singular Reconst Image");
121       pause(0.1)
122     end
123   end
124   %fprintf('Y1L = %d\n',i);
125   X_Lum = [X_Lum, X_Y1];
126
127   if(display == 1)
128     figure();
129   end
130   for i = StartSingular+2:EndSingular
```

```
    s_ = StartSingular:i;
    %s_ = i:i;

    Y2_SVD = U_Y2(:,s_)*S_Y2(s_,s_)*V_Y2(:,s_)';
    X_Y2 = [X_Y2, reshape(Y2_SVD,M*N,1)];

    if(display == 1)
      Y2_SVD = cast(Y2_SVD, 'uint8');
      imshow(Y2_SVD);
      title("Y2 Int Singular Reconst Image");
      pause(0.1)
    end

  end
  %fprintf('Y2L = %d\n',i);
  X_Lum = [X_Lum, X_Y2];


  %View aggregate SVD reconstruction of single singular values
  if(display == 1)
    L_sum = cast(L_sum,'uint8');
    figure();
    imshow(L_sum);
    title("L Summed Int Singular Reconst Image");
  end

end
```

B.4 Color DMD preparation and residual

```
function [X_DLow, X_DSparse] = ColorDMD2f(X_color, GrayscaleImage, M, N)

  [U_xcolor, S_xcolor, V_xcolor] = svd(X_color,'econ');
  [M_xcolor, N_xcolor] = size(X_color);

  %Define s as high as possible
  s_ = length(S_xcolor)-1;

  %DMD Low rank and sparse separation Version 2
  [Xs_DLow, Xs_DSparse, Us_xcolor] = LowSparseDMD3(X_color, s_);

  X_DLow = zeros(M_xcolor,N_xcolor);
  X_DSparse = zeros(M_xcolor,N_xcolor);

  X_DLow = Us_xcolor(:,1:s_)*Xs_DLow;%%%%%%%%%%%%%%%%%%%%%%%%
  X_DLow = sum(X_DLow')';
  %X_DSparse = Us_xcolor(:,1:s_)*Xs_DSparse;%%%%%%%%%%%%%%%%%
  %X_DSparse = RSparseRecon(X_color, X_DSparse, 25);
  X_Gray = reshape(GrayscaleImage,M*N,1);
  X_Gray = cast(X_Gray,'double');
  X_DSparse = RSparseRecon2(X_Gray, X_DLow, 25);

end
```

B.5 DMD Low-Sparse reconstruction algorithm

```
function [LOW, SPARSE, Us] = LowSparseDMD3(X_DATA, s_)

  [M_DATA, N_DATA] = size(X_DATA);

  %Creating X and Y matrices out of the original Data Matrix
  X(:,1:N_DATA-1) = [X_DATA(:,1:N_DATA-1)];
  Y(:,1:N_DATA-1) = [X_DATA(:,2:N_DATA)];

  [U,S,V] = svd(X,'econ');
  A = U(:,1:s_)'*Y*V(:,1:s_)*inv(S(1:s_,1:s_));
  Xs = U(:,1:s_)'*X;

  dt = 1;
  t = (0:dt:dt*(N_DATA-1));

  %fprintf('\nObtaining Eigen Vectors and values and Omega of DMD Matrix A
     ...\n');
  [e_Vect, e_Val] = eig(A); %finds eigenvectors and eigenvalues of A
  e_Val_D = diag(e_Val); %Take all the diagonals of eigenvalues
  omega = log(e_Val_D)/dt;
  absOmega = abs(omega);
  absOmegaSorted = sort(absOmega);

  Xs_DLow1 = zeros(s_,N_DATA);
  Xs_DSparse1 = zeros(s_,N_DATA);

  %Low and sparse reconstruction
  b = pinv(e_Vect)*Xs(:,1);
  for i = 1:length(A)
      if(absOmega(i) == min(absOmega))
        Xs_DLow1 = Xs_DLow1 + b(i)*e_Vect(:,i)*exp(omega(i)'*t);
      else
        Xs_DSparse1 = Xs_DSparse1 + b(i)*e_Vect(:,i)*exp(omega(i)'*t);
      end
  end

  LOW = Xs_DLow1;
  SPARSE = Xs_DSparse1;
  Us = U(:,1:s_);

end
```

B.6 Color Saliency Map Adjustments

```
function SM_colorview = ColorFinal2f(X_DLow1, X_DLow2, X_DSparse1,
   X_DSparse2, M, N, Display)

  %Normalisation
  X_DLow1 = real(X_DLow1);
  a = X_DLow1 - min(min(X_DLow1));
  b = max(max(X_DLow1)) - min(min(X_DLow1));
```

```
X_DLow1_norm = a/b;

X_DLow2 = real(X_DLow2);
a = X_DLow2 - min(min(X_DLow2));
b = max(max(X_DLow2)) - min(min(X_DLow2));
X_DLow2_norm = a/b;

X_DSparse1 = real(X_DSparse1);
a = X_DSparse1 - min(min(X_DSparse1));
b = max(max(X_DSparse1)) - min(min(X_DSparse1));
X_DSparse1_norm = a/b;

X_DSparse2 = real(X_DSparse2);
a = X_DSparse2 - min(min(X_DSparse2));
b = max(max(X_DSparse2)) - min(min(X_DSparse2));
X_DSparse2_norm = a/b;

%Sum of low and sparse components
X_LOWOUT = X_DLow1_norm + X_DLow2_norm;
X_SPARSEOUT = X_DSparse1_norm + X_DSparse2_norm;


FrameNum = 1;

x_LOWVIEW = reshape(X_LOWOUT(:,FrameNum),M,N);
a = x_LOWVIEW - min(min(x_LOWVIEW));
b = max(max(x_LOWVIEW)) - min(min(x_LOWVIEW));
x_LOWVIEW = 255*a/b;


x_SPARSEVIEW = reshape(X_SPARSEOUT(:,FrameNum),M,N);
a = x_SPARSEVIEW - min(min(x_SPARSEVIEW));
b = max(max(x_SPARSEVIEW)) - min(min(x_SPARSEVIEW));
x_SPARSEVIEW = 255*a/b;

w = 1.5;
SM_color = (X_LOWOUT-w*X_SPARSEOUT);
SM_color = reshape(SM_color,M,N);
SM_color(SM_color<0) = 0;
a = SM_color - min(min(SM_color));
b = max(max(SM_color)) - min(min(SM_color));
SM_colorview = 255*(a/b).^2;

x_LOWVIEW = cast(x_LOWVIEW,'uint8');
x_SPARSEVIEW = cast(x_SPARSEVIEW,'uint8');

if(Display == 1)
  figure();
  imshow(x_LOWVIEW);
  title("D_A + D_B LOWVIEW Image");

  figure();
  imshow(x_SPARSEVIEW);
  title("D_A + D_B SPARSEVIEW Image");
```

```
  end

end
```

B.7 Post-processing and Final Saliency Map

```
function SM_FIN = SMFinalMapf(SM_colorview, SM_LUMVIEW, M, N, Display)

  xpos = 1:1:M;
  ypos = 1:1:N;

  %Gaussian Distribution
  xposn = xpos-M/2;
  yposn = ypos-N/2;

  Wg = zeros(M,N);
  sigma = 175;
  for i  = 1:M
    for j = 1:N
      Wg(i,j) = exp(-(xposn(i)^2+yposn(j)^2)/(2*sigma^2))/(2*pi*sigma^2);
    end
  end
  Const = 1/max(max(Wg));
  Wg = Const*Wg;


  SM_colorview = cast(SM_colorview, 'double');
  SM_colorview = Wg.*SM_colorview;
  SM_LUMVIEW = cast(SM_LUMVIEW, 'double');
  SM_LUMVIEW = Wg.*SM_LUMVIEW;

  if(Display == 1)
    figure();
      SM_colorview = cast(SM_colorview, 'uint8');
      imshow(SM_colorview);
      title("SM colorview Weighted");

    figure();
      SM_LUMVIEW = cast(SM_LUMVIEW, 'uint8');
      imshow(SM_LUMVIEW);
      title("SM LUMVIEW Weighted");
  end


  nhood = [0 1 0; 1 1 1; 0 1 0];
  SE = strel('square',3);

  %Morphological Erosion and Dilation
  N_Erode = 1;

  for i = 1:N_Erode
    SM_colorview = imdilate(SM_colorview,nhood);
    SM_LUMVIEW = imdilate(SM_LUMVIEW,nhood);
```

```
  end
  for i = 1:N_Erode*2
    SM_colorview = imerode(SM_colorview,nhood);
    SM_LUMVIEW = imerode(SM_LUMVIEW,nhood);
  end
  for i = 1:N_Erode
    SM_colorview = imdilate(SM_colorview,nhood);
    SM_LUMVIEW = imdilate(SM_LUMVIEW,nhood);
  end

  if(Display == 1)
    figure();
    imshow(SM_colorview);
    title("SM colorview Eroded");

    figure();
    imshow(SM_LUMVIEW);
    title("SM LUMVIEW Eroded");
  end

  %Sigmoid Intensity adjustment
  SM_colorview = cast(SM_colorview, 'double');
  SMa = SM_colorview - min(min(SM_colorview));
  SMb = max(max(SM_colorview)) - min(min(SM_colorview));
  SM_colorview = SMa/SMb;
  b = 30;
  C = 1+exp(-b*(SM_colorview-.3));
  SM_colorview = C.^(-1);

  SMa = SM_colorview - min(min(SM_colorview));
  SMb = max(max(SM_colorview)) - min(min(SM_colorview));
  SM_colorview = 255*SMa/SMb;

  SM_LUMVIEW = cast(SM_LUMVIEW, 'double');
  SMa = SM_LUMVIEW - min(min(SM_LUMVIEW));
  SMb = max(max(SM_LUMVIEW)) - min(min(SM_LUMVIEW));
  SM_LUMVIEW = SMa/SMb;
  b = 30;
  C = 1+exp(-b*(SM_LUMVIEW-.5));
  SM_LUMVIEW = C.^(-1);

  SMa = SM_LUMVIEW - min(min(SM_LUMVIEW));
  SMb = max(max(SM_LUMVIEW)) - min(min(SM_LUMVIEW));
  SM_LUMVIEW = 255*SMa/SMb;

  if(Display == 1)
    figure();
    SM_colorview = cast(SM_colorview, 'uint8');
    imshow(SM_colorview);
    title("SM Color Postprocessed");

    figure();
    SM_LUMVIEW = cast(SM_LUMVIEW, 'uint8');
    imshow(SM_LUMVIEW);
```

```
    title("SM Lum Postprocessed");
  end

  SM_LUMVIEW = cast(SM_LUMVIEW, 'double');
  SM_colorview = cast(SM_colorview, 'double');

  SM_FIN = max(SM_colorview,SM_LUMVIEW);

end
```

Appendix C Complementary Figures

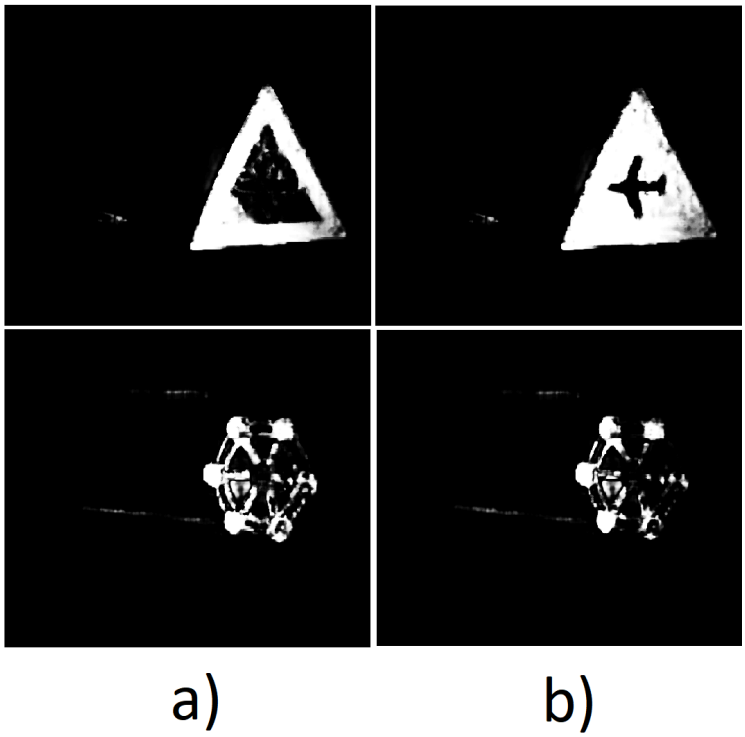

Figure 6: U channel in YUV color space calculated with blue channel of RGB converted into a) 'uint8' and b) 'double'.

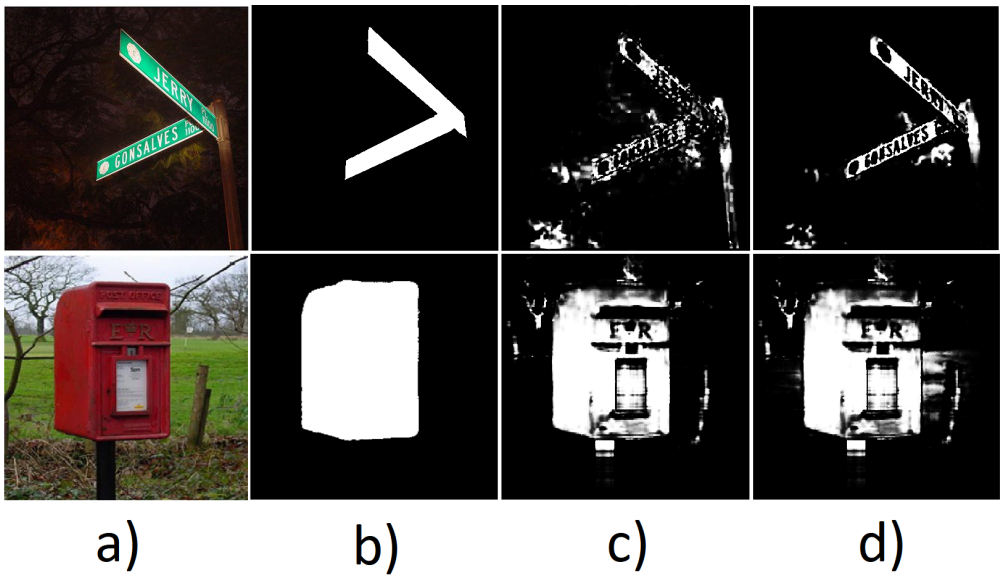

Figure 7: Comparison between a) original image, b) ground truth image, c) Cb and Cr channels normalized with minimum 0 and maximum 255 and d) Cb and Cr channels not normalized.

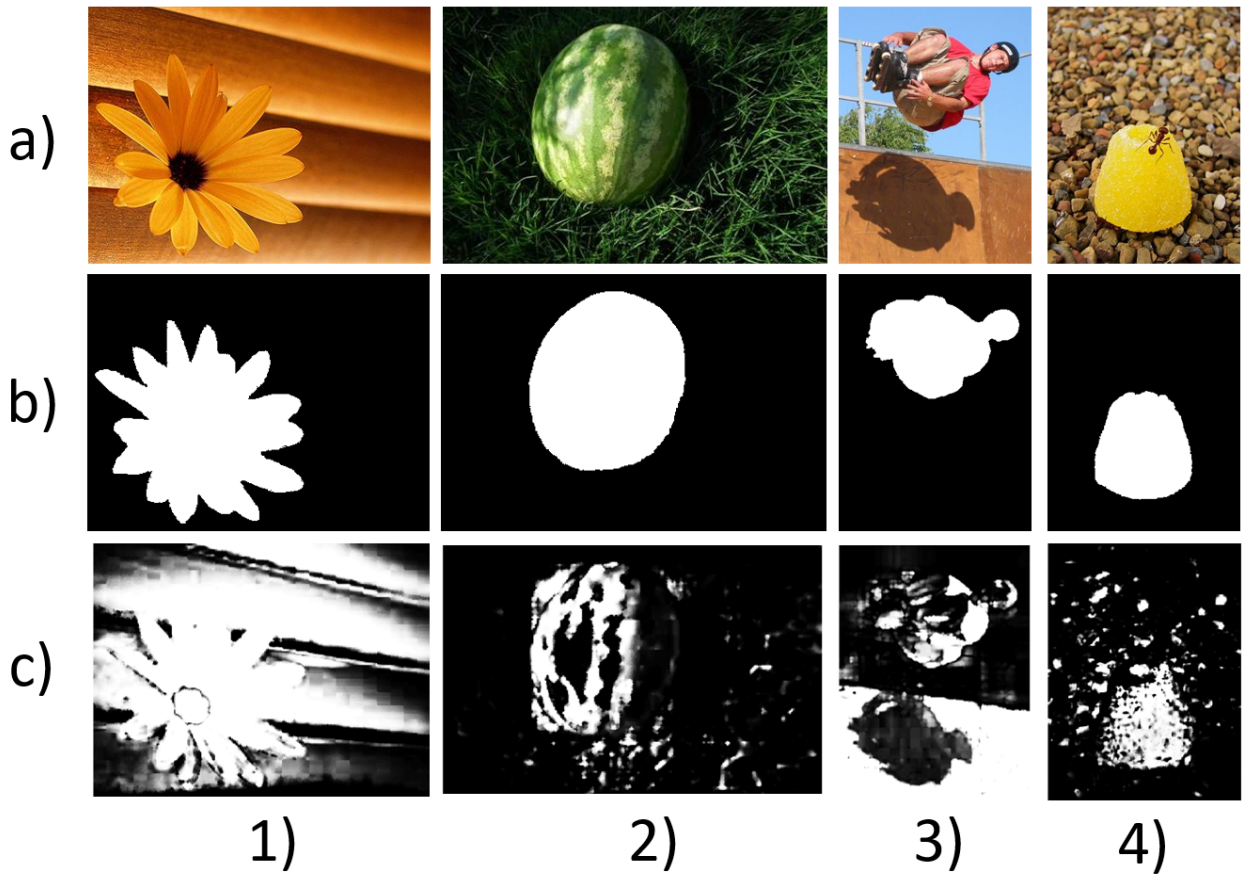

Figure 8: More images resulting in poor performance from DMD with rows as a) original image, b) ground truth, c) DMD of this paper