THE COOPER UNION FOR
THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

# Saliency Estimation Using the Complementary Color Wavelet Transform

By
Karol Wadolowski

A thesis submitted in partial fulfillment of the requirements for the degree of Master of Engineering

May 2020

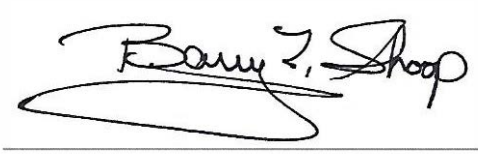
Advisor

Professor Fred L. Fontaine

THE COOPER UNION FOR
THE ADVANCEMENT OF SCIENCE AND ART

ALBERT NERKEN SCHOOL OF ENGINEERING

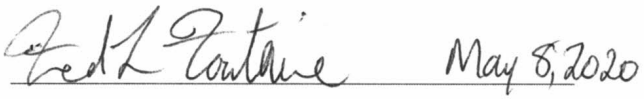
This thesis was prepared under the direction of the Candidate's Thesis Advisor and has received approval. It was submitted to the Dean of the School of Engineering and the full Faculty, and was approved as partial fulfillment of the requirements for the degree of Master of Engineering.



5.11.2020

Barry L. Shoop, PhD., P.E. Date

Dean, Albert Nerken School of Engineering



May 8, 2020

Professor Fred L. Fontaine Date

Candidate's Thesis Advisor

# Acknowledgments

I would first like to thank my advisor, Professor Fred Fontaine. Professor Fontaine has played a fundamental role in exposing me to the amazing magic that electrical engineers perform everyday. He has helped me gain a huge appreciation for this field and has built a core understanding that will only be built upon. For this and much more I am truly grateful.

I would also like to thank The Cooper Union. The Cooper Union has provided me with many unforgettable memories and has introduced me to many fantastic individuals. Like I, they understand the hardships and the feeling of deep satisfaction that comes with overcoming the many trials that Cooper throws at us. Among these individuals, I would like to give special thanks to Brian Frost-LaPlante, Nikola Janjušević, and Jack Langner. They are some of the closest friends that I have made during my time here. They have constantly pushed me forward, helping me to understand just exactly how much can be accomplished when people, with a common goal, come together.

Lastly, I would like to thank my family. The support my parents gave me always kept me focused and motivated. They understood the difficulties I faced on my journey and helped me in every way they could. I would also like to thank my brother Rafal. Unlike my parents, he distracted me and allowed me to step away from my work. He gave me moments to recuperate and relax which were vital for me throughout my time at Cooper.

# Abstract

In many image processing tasks it is useful to have a saliency map that highlights the important parts of an image. Many algorithms currently exist that generate saliency maps. In this work a new algorithm is developed that is capable of generating multiple saliency maps from a single image, highlighting different parts of the image. This is done using the Complementary Color Wavelet Transform, a tool which captures various color changes at different scales in an image. The outputs of this transform are then fed into an adapted version of a previously developed saliency map generation algorithm. Using this transform, the algorithm is able to generate four saliency maps for a given image. This algorithm is applied on the images in the CAT2000 dataset. The saliency maps generated using this algorithm are evaluated using two popular performance metrics, the similarity metric and linear correlation coefficient. In many cases, the proposed algorithm provides better results than other approaches.

# Contents

# List of Figures

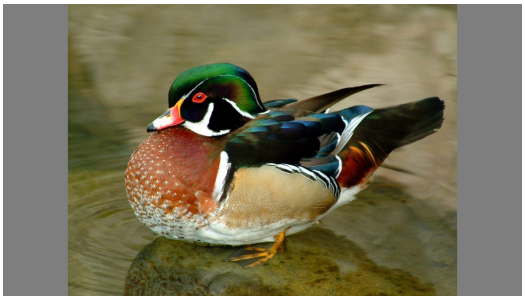
# List of Tables

# Chapter 1

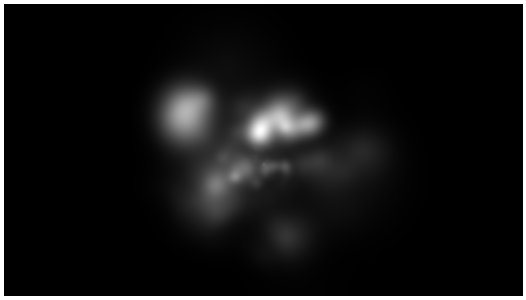
# Introduction

## 1.1 Problem Statement

When you look at your surroundings, certain objects or regions in your range of vision catch your attention more than others. Such areas or regions are said to be salient. A common way of displaying salient points in an image is to create a saliency map (SM). Figure 1.1 shows an example of an image and a saliency map that was generated from it.



(a) Original Image



(b) Saliency Map ($O^C$)

Figure 1.1: The image (Object 157) from the CAT2000 dataset and a saliency map generated using the proposed method.

Many image processing applications make use of saliency maps. These applications include image segmentation [1], computer vision [2], and image compression [3] among many others. In image segmentation, saliency maps are used to separate the

foreground from background. In computer vision applications, saliency maps help detect objects. Identifying salient points is also useful for image compression. In this application the regions highlighted by a saliency map are not compressed as much since they contain the most important information. For such tasks, saliency maps have a critical role. Color has a big influence on which areas are perceived to be salient. If a color appears less frequently in an image compared to other colors and is concentrated in a region it is more likely to catch your attention. This thesis aims to develop an algorithm that exploits color information in images to generate accurate saliency maps and highlights different aspects of an image.

## 1.2 The Contribution of This Work

This thesis presents an algorithm for generating saliency maps. The method for generating saliency maps presented in [4] is converted to work with the outputs of the Complementary Color Wavelet Transform (CCWT) [5]. To use the CCWT in the algorithm presented in [4], several incompatibilities are resolved. In doing so, a method of generating multiple saliency maps highlighting different color differences using the same image is developed.

The performance of the method proposed in this work is measured in two ways. The performance is quantified using common scoring metrics for saliency maps. The method is also assessed qualitatively as it generates saliency maps disregarding certain color transitions. To the author's knowledge no metrics or datasets exist that are able to evaluate this aspect of the method specifically.

## 1.3 Structural Overview

Chapter 2 provides a background on wavelets, color spaces, and the Complementary Color Wavelet Transform. This includes a description of what wavelets are and how

they are used for image processing. Chapter 3 explains the saliency map generation algorithm developed in [4] and how saliency maps are quantitatively evaluated. Chapter 4 discusses how the algorithm presented in [4] is modified so that the CCWT can be used in its framework. Chapter 5 discusses the performance of the proposed algorithm. Chapter 6 summarizes the method and results, and concludes with some remarks about future work.

# Chapter 2

# Wavelets for Color Image Processing

## 2.1 Wavelets

We consider the space of square integrable functions over $\mathbb{R}$, $\mathbf{L}^2(\mathbb{R})$, with inner product given by:

$$\langle d, s \rangle = \int_{\mathbb{R}} d(t)s(t)dt \tag{2.1}$$

for functions $d, s \in \mathbf{L}^2(\mathbb{R})$. For any $\xi \in \mathbf{L}^2(\mathbb{R})$, $j, n \in \mathbb{Z}$, define the translated and dilated function $\xi_{j,n}$ as:

$$\xi_{j,n}(t) = \frac{1}{\sqrt{2^j}} \xi \left( \frac{t - 2^j n}{2^j} \right) \tag{2.2}$$

A real 1-D wavelet is a function $\psi \in \mathbf{L}^2(\mathbb{R})$ such that:

$$0 = \int_{\mathbb{R}} \psi(t)dt \tag{2.3}$$

There exist wavelets such that $\{\psi_{j,n}\}_{(j,n)\in\mathbb{Z}^2}$ forms an orthonormal basis for $\mathbf{L}^2(\mathbb{R})$. The orthonormal condition means that, for all $j, n, k, m \in \mathbb{Z}$:

$$\langle \psi_{j,n}(t), \psi_{k,m}(t) \rangle = \begin{cases} 1, \text{ if } j = k \text{ and } n = m \\ 0, \text{ otherwise} \end{cases} \tag{2.4}$$

To form a basis for $\mathbf{L}^2(\mathbb{R})$, every $f \in \mathbf{L}^2(\mathbb{R})$ must be representable as:

$$f(t) = \sum_{(j,n)\in\mathbb{Z}^2} \langle f(t), \psi_{j,n}(t) \rangle \, \psi_{j,n}(t) \tag{2.5}$$

A framework for constructing orthogonal wavelets using a certain class of discrete-time filters, called conjugate mirror filters, is provided in [6]. Given the impulse response of a discrete-time filter, $h : \mathbb{Z} \to \mathbb{C}$, with discrete-time Fourier transform given by:

$$\hat{h}(w) = \sum_{n\in\mathbb{Z}} h[n] e^{-jwn} \tag{2.6}$$

it is a conjugate mirror filter if

$$\forall\, \omega \in \mathbb{R}, \quad |\hat{h}(\omega)|^2 + |\hat{h}(\omega + \pi)|^2 = 2 \tag{2.7}$$

To start, choose a function $\phi \in \mathbf{L}^2(\mathbb{R})$ such that:

$$1 = \int_{\mathbb{R}} \phi(t) dt \tag{2.8}$$

and, for $n$, $m \in \mathbb{Z}$:

$$\langle \phi(t-n), \phi(t-m) \rangle = \begin{cases} 1, \text{ if } n = m \\ 0, \text{ otherwise} \end{cases} \tag{2.9}$$

Using equation (2.9), we can conclude that the integer translations of $\phi(t)$ forms an orthonormal basis for the space $V_0$ given by:

$$V_0 = span(\{\phi(t-n)\}_{n\in\mathbb{Z}}) \tag{2.10}$$

The function $\phi(t)$ will be referred to as the scaling function. We can define extra spaces:

$$V_j = span(\{\phi_{j,n}(t)\}_{n\in\mathbb{Z}}) \tag{2.11}$$

for $j \in \mathbb{Z}$. Each space corresponds to the *span* of the dyadic integer translations (i.e., $2^j n$), of a dilated version of the scaling function at a fixed scale. The set $\{\phi_{j,n}\}_{n\in\mathbb{Z}}$ is an orthonormal basis of $V_j$.

In order to associate $\phi(t)$ to conjugate mirror filters, and thus, a wavelet, the sequence of spaces $\{V_j\}_{j\in\mathbb{Z}}$ must satisfy the six properties that define a multiresolution analysis [6]:

$$\forall(j,n) \in \mathbb{Z}^2, \quad f(t) \in V_j \iff f(t-2^j n) \in V_j \tag{2.12}$$

$$\forall j \in \mathbb{Z}, \quad V_{j+1} \subset V_j \tag{2.13}$$

$$\forall j \in \mathbb{Z}, \quad f(t) \in V_j \iff f\Big(\frac{t}{2}\Big) \in V_{j+1} \tag{2.14}$$

$$\lim_{j\to+\infty} V_j = \bigcap_{j\in\mathbb{Z}} V_j = \{0\} \tag{2.15}$$

$$\lim_{j\to-\infty} V_j = \mathbf{L}^2(\mathbb{R}) \tag{2.16}$$

$$\exists\, \xi \in \mathbf{L}^2(\mathbb{R}) \text{ such that } \{\xi(t-n)\}_{n\in\mathbb{Z}} \text{ is an orthonormal basis of } V_0. \tag{2.17}$$

Given a scaling function that satisfies the six properties of a multiresolution analysis, conjugate mirror filters can be constructed. From equation (2.13), it is clear that $V_1 \subset V_0$. From equation (2.14), it is also clear that $\phi\big(\frac{t}{2}\big) \in V_1 \subset V_0$. This means that $\frac{1}{\sqrt{2}}\phi\big(\frac{t}{2}\big)$ can be expressed in the basis of $V_0$, the set of translations $\{\phi(t-n)\}_{n\in\mathbb{Z}}$, as:

$$\frac{1}{\sqrt{2}}\phi\Big(\frac{t}{2}\Big) = \sum_{n\in\mathbb{Z}} h[n]\phi(t-n) \tag{2.18}$$

where

$$h[n] = \Big\langle \frac{1}{\sqrt{2}}\phi\Big(\frac{t}{2}\Big), \phi(t-n)\Big\rangle \tag{2.19}$$

The scaling factor $\frac{1}{\sqrt{2}}$ is included to show that a dilated version of the scaling function is representable as a combination of translated versions of the same scaling function. The discrete sequence $h[n]$ can be thought of as the impulse response of a discrete-time filter. Constructed this way, the filter $h[n]$ satisfies the conjugate mirror filter property (equation (2.7)), and

$$\hat{h}(0) = \sqrt{2} \tag{2.20}$$

In [6] and [7], it is shown that given an $\hat{h}(w)$ that is $2\pi$ periodic, continuously differentiable in a neighborhood of $\omega = 0$, satisfies equations (2.7) and (2.20), and

$$\inf_{\omega \in [-\pi/2, \pi/2]} |\hat{h}(\omega)| > 0 \tag{2.21}$$

where inf denotes the infimum, then

$$\hat{\phi}(\omega) = \prod_{p=1}^{\infty} \frac{\hat{h}(2^{-p}\omega)}{\sqrt{2}} \tag{2.22}$$

is the Fourier transform of a scaling function $\phi \in \mathbf{L}^2(\mathbb{R})$. The Fourier transform of a function $f \in \mathbf{L}^2(\mathbb{R})$ is given by:

$$\hat{f}(\omega) = \int_{\mathbb{R}} f(t) e^{-jwt} dt \tag{2.23}$$

This is an important result because it shows not only that the scaling function corresponds to a conjugate mirror filter, but also that if a filter is properly constructed in the digital frequency domain then it corresponds to a scaling function. After properly constructing a filter, a scaling function can be defined through the relations in equations (2.18) and (2.19).

In order to justify how wavelets come from this line of reasoning, it is important to first explain some of the properties of a multiresolution analysis. To start, the space $V_j$ contains the approximation of functions $f \in \mathbf{L}^2(\mathbb{R})$ with the scaling functions $\left\{\phi_{j,n}(t)\right\}_{n \in \mathbb{Z}}$ at a fixed scale, $j \in \mathbb{Z}$. For a given function $f$, the approximation $f_a$ of

$f$ at scale $j$ is given by:

$$f_{a,j}(t) = \sum_{n\in\mathbb{Z}} \langle f(t), \phi_{j,n}(t)\rangle \, \phi_{j,n}(t) \tag{2.24}$$

By going to a finer scale (smaller $j$ value) the approximations become more accurate. The set of coefficients, $\{\langle f(t), \phi_{j,n}(t)\rangle\}_{n\in\mathbb{Z}}$, is a representation of $f$ in the space $V_j$. The functions representable in $V_j$ are also representable in $V_{j-1}$ by the property in equation (2.13). $V_{j-1}$ can represent additional functions due to the extra scaling functions at that scale. As the scale decreases the quality of the approximation $f_{a,j}$ increases. Equation (2.16) captures this quality saying that as the scale decreases to $-\infty$ the space of representable functions goes to $\mathbf{L}^2(\mathbb{R})$. Conversely, if our scale increases, then fewer functions can be represented and in the limit only the constant 0 function can be represented (equation (2.15)).

The space $V_j$ is a strict subspace of $V_{j-1}$. Let $W_j$ be the orthogonal complement of $V_j$ in $V_{j-1}$. Then:

$$V_{j-1} = V_j \oplus W_j \tag{2.25}$$

where $\oplus$ is the direct sum of two vector spaces. $W_j$ contains the details of $V_{j-1}$ that $V_j$ can not express. Using this, the orthogonal projection of a function $f$ onto $V_{j-1}$ can be expressed as:

$$P_{V_{j-1}} f = P_{V_j} f + P_{W_j} f \tag{2.26}$$

The orthogonal projection of $f$ onto $V_{j-1}$ is equivalent to the approximation $f_{a,j-1}$. The detail spaces $W_j$ and $W_k$ for $j \neq k$ are orthogonal and:

$$\mathbf{L}^2(\mathbb{R}) = \oplus_{j\in\mathbb{Z}} W_j \tag{2.27}$$

In addition to these properties, an orthonormal basis for $W_j$ can be constructed by dilating and translating a wavelet $\psi(t)$. If the wavelet is defined in the frequency

domain by:

$$\hat{\psi}(\omega) = \frac{1}{\sqrt{2}}\hat{g}\Big(\frac{\omega}{2}\Big)\hat{\phi}\Big(\frac{\omega}{2}\Big) \tag{2.28}$$

$$\hat{g}(\omega) = e^{-j\omega}\hat{h}^*(\omega + \pi) \tag{2.29}$$

where $^*$ is the complex conjugate, then $\{\psi_{j,n}(t)\}_{n\in\mathbb{Z}}$ is a basis for $W_j$. Note that the $j$ appearing in the exponential of equation (2.29) refers to the imaginary unit and not the scale. The discrete-time filter $g[n]$, $n \in \mathbb{Z}$, given by:

$$g[n] = \Big\langle \frac{1}{\sqrt{2}}\psi\Big(\frac{t}{2}\Big), \phi(t-n)\Big\rangle \tag{2.30}$$

has the discrete-time Fourier transform in equation (2.29) and is a conjugate mirror filter. These filter coefficients are similar to that of $h[n]$ in that a dilated version of the original wavelet can be expressed as:

$$\frac{1}{\sqrt{2}}\psi\Big(\frac{t}{2}\Big) = \sum_{n\in\mathbb{Z}} g[n]\phi(t-n) \tag{2.31}$$

Taking the inverse discrete-time Fourier transform of equation (2.29) gives us the additional equality:

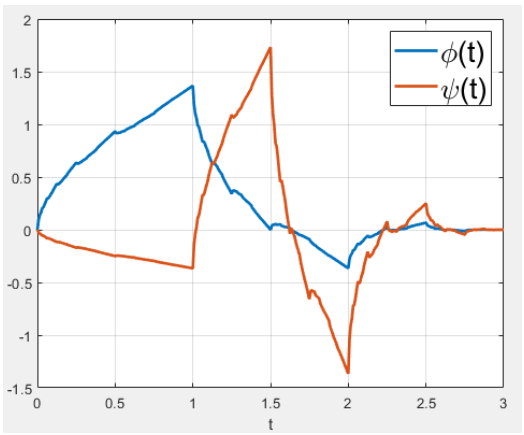
$$g[n] = (-1)^n h[-n] \tag{2.32}$$

Figure 2.1 displays the scaling and wavelet functions associated with several types of conjugate mirror filters. These functions were iteratively generated using equations (2.18), (2.19), (2.30), and (2.31).

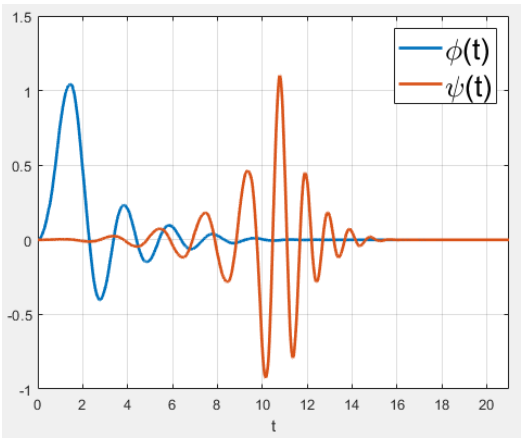
The conjugate mirror filters $h[n]$ and $g[n]$ are useful for more than just generating scaling and wavelet functions. These two filters establish relationships among the coefficients:

$$a_j[n] = \langle f(t), \phi_{j,n}(t)\rangle \tag{2.33}$$

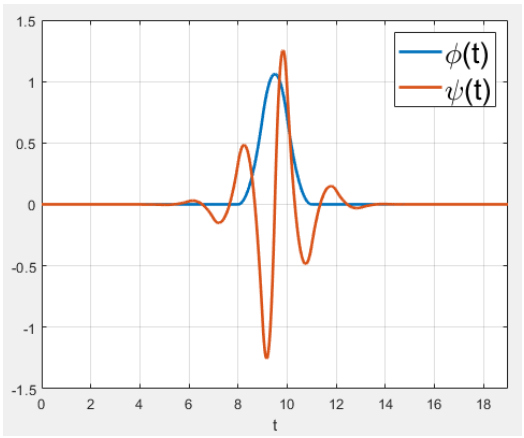
$$d_j[n] = \langle f(t), \psi_{j,n}(t)\rangle \tag{2.34}$$



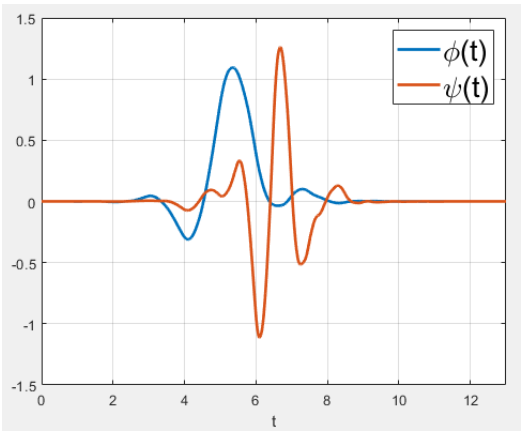

(a) Daubechies 2




(b) Fejér-Korovkin 22




(c) Reverse Biorthogonal 3.9




(d) Symlet 7

Figure 2.1: Several scaling and wavelet functions associated with specific families of conjugate mirror filters.

at different scales. The values $a_j[n]$ and $d_j[n]$ are respectively called the approximation and detail coefficients. The approximation coefficients tell us how a function $f$ is represented in the space $V_j$. The detail coefficients analogously represent $f$ in the space $W_j$. This is to say $a_j[n]$ represents the projection $P_{V_j}f$ of $f$ onto $V_j$, and $d_j[n]$ represents the projection $P_{W_j}$ of $f$ onto $W_j$. These coefficients are related to the filters in the following way:

$$a_{j+1}[n] = (a_j[n] * h[n]) \downarrow 2 \tag{2.35}$$

$$d_{j+1}[n] = (a_j[n] * g[n]) \downarrow 2 \tag{2.36}$$

where $*$ is convolution and $\downarrow 2$ is decimation by 2. These equations signify that the coefficients used to represent $f$ at scale $j+1$ can be determined from the scale $j$ coefficients using a simple filtering structure. This filtering structure can be seen in Figure 2.2a. The process of obtaining these coefficients at progressively increasing (coarser) scales is called analysis.

Additionally, in a process called synthesis, the coefficients at scale $j$ can be recovered using the coefficients at scale $j+1$. This corresponds to what is called the perfect reconstruction property of the filters. If the filters $h[n]$ and $g[n]$ have a finite impulse response, then the synthesis filters $\tilde{h}[n]$ and $\tilde{g}[n]$ are related to the analysis filters by:

$$\tilde{h}[n] = h[-n] \tag{2.37}$$

$$\tilde{g}[n] = (-1)^n h[n] \tag{2.38}$$

It should be noted that only finite impulse response filters are used for this project. The synthesis filters allow us to relate the coefficients at two adjacent scales as follows:

$$a_j[n] = (a_{j+1}[n] \uparrow 2) * \tilde{h}[n] + (d_{j+1}[n] \uparrow 2) * \tilde{g}[n] \tag{2.39}$$

where $\uparrow 2$ is interpolation by 2. This relation also corresponds to a filtering structure that can be seen in Figure 2.2b.

Both $\{h[n-2l], g[n-2l]\}_{l\in\mathbb{Z}}$ and $\{\tilde{h}[n-2l], \tilde{g}[n-2l]\}_{l\in\mathbb{Z}}$, are orthogonal bases for $\boldsymbol{l}^2(\mathbb{Z})$. $\boldsymbol{l}^2(\mathbb{Z})$ is the space of square summable functions over the integers. The consequence of this is that the same filter banks in Figure 2.2 can be used for discrete-time functions. In this case, the convolutions, $*$, in equations (2.35), (2.36), and (2.39) can be replaced by circular convolutions, $\circledast$. Circular convolution is usually used when dealing with functions that have a finite support.

Processing a discrete signal $a_j[n]$ as in Figure 2.2a is called the discrete wavelet transform (DWT). At this point it should be noted that $h[n]$ is a lowpass filter and

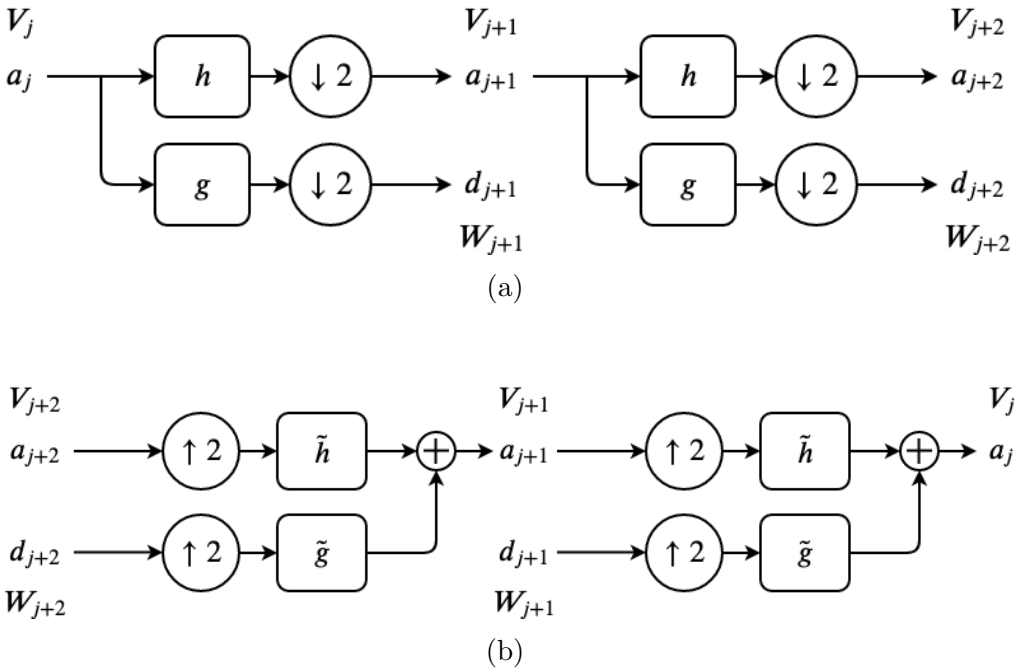



Figure 2.2: The (a) analysis and (b) synthesis filter bank structures.

$g[n]$ is a highpass filter. Therefore the outputs of the DWT can be interpreted as decomposing the signal into two subbands. Using the synthesis filter bank in Figure 2.2b performs the inverse discrete wavelet transform (IDWT). The IDWT takes the two subbands and reconstructs a signal from each band (one low, one high).

The DWT and IDWT can be easily extended to the 2-D case allowing for the representations of functions in $\boldsymbol{l}^2(\mathbb{Z}^2)$. The 2-D DWT and IDWT filter bank structures can be seen in Figure 2.3. The $w$ variables in Figure 2.3 are the coefficients from filtering at different scales. LH stands for first filtering along columns with the highpass filter ($g[n]$) and then along rows with the lowpass filter ($h[n]$), a notation similar to that of composing functions. LL, HL, and HH have analogous meanings. An example of the 2-D DWT in action can be seen in Figure 2.4. As can be seen the LL coefficients most resemble the original image. The LL coefficients are the approximation coefficients. The LH coefficients pick out horizontal edges, the HL coefficients vertical edges, and the HH coefficients the diagonal edges. The LH, HL, and HH coefficients are the detail coefficients. As shown here, this is a separable process, meaning that

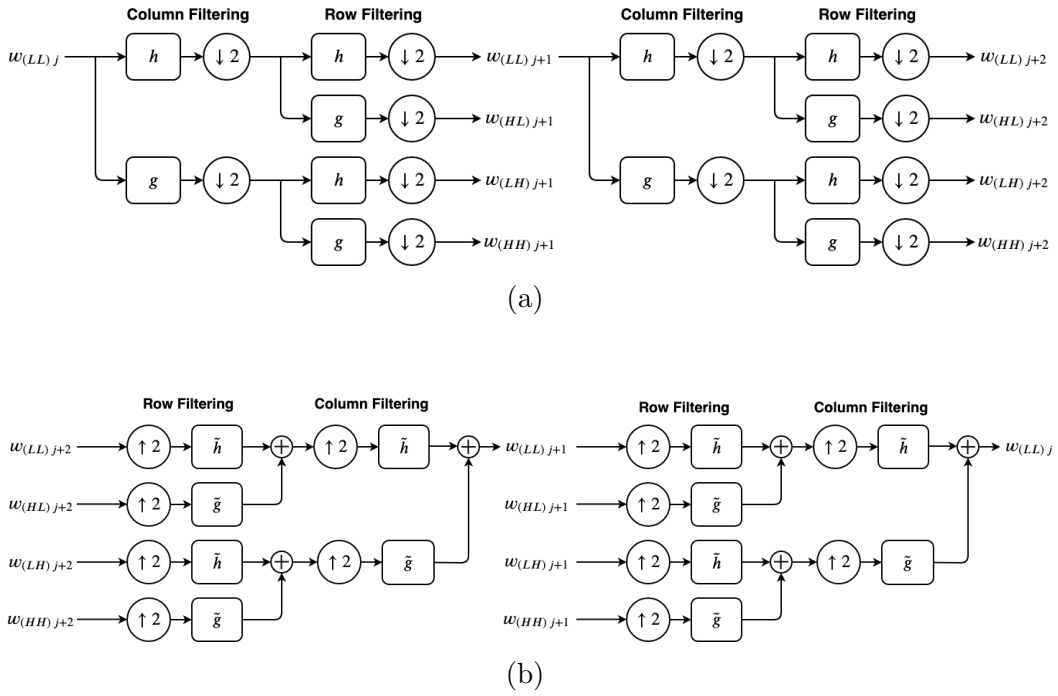



Figure 2.3: The (a) analysis and (b) synthesis filter bank structures for the 2-D DWT and IDWT respectively. Row and column filtering can be reversed.

the 2-D DWT and IDWT combines 1-D processing along the rows and the columns.

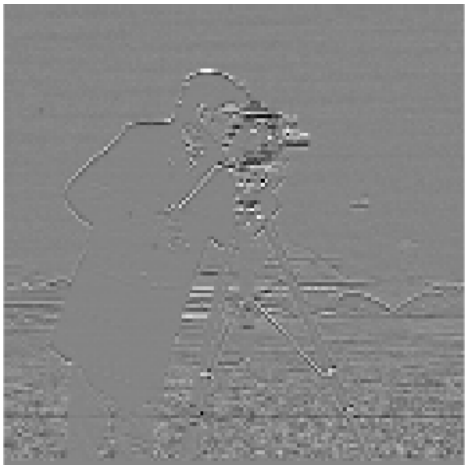
A common generalization of the 2-D DWT and IDWT is to use two sets of conjugate mirror filters corresponding to two different wavelets. In this case, the structures of the analysis and synthesis filter banks, seen in Figure 2.3, are slightly altered. The filters associated with the first wavelet are used for row filtering and the filters associated with the second wavelet are used for column filtering.

## 2.2 Color Spaces

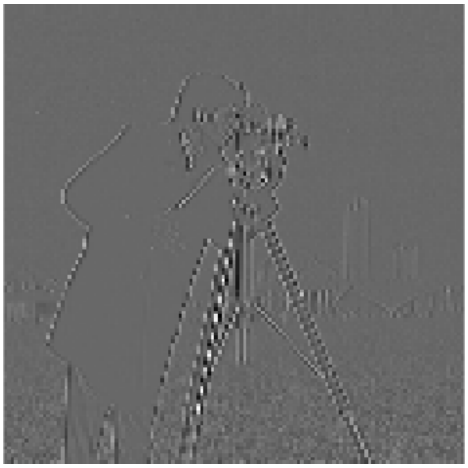
The colors perceptible to humans can be mathematically represented in many ways. As such many different color spaces have been created. Each represents the properties of a color in different ways. The simplest to understand is the RGB color space. The RGB color space is a 3-D space where each color is represented as triplet of real values $[r, g, b]$, each between 0 and 1 that reflect how much red, green, and blue are present

(a) Original Image (Camera Man)

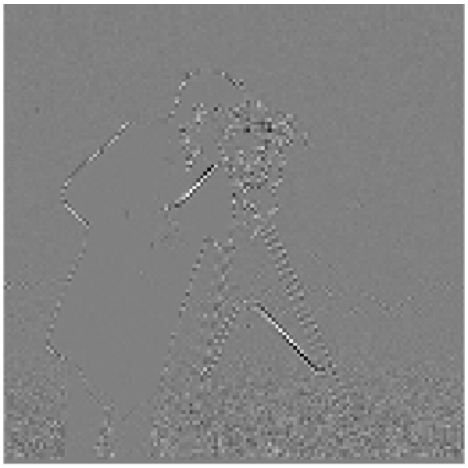
(b) LL



(c) LH

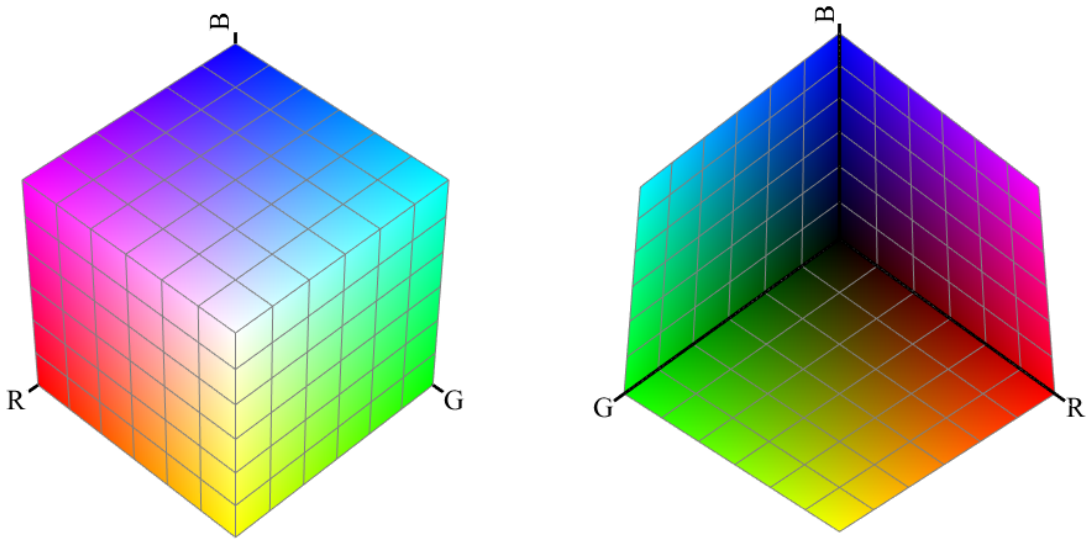


(d) HL



(e) HH

Figure 2.4: The four outputs of the 2-D DWT using Daubechies 2 filters.




(a) (b)

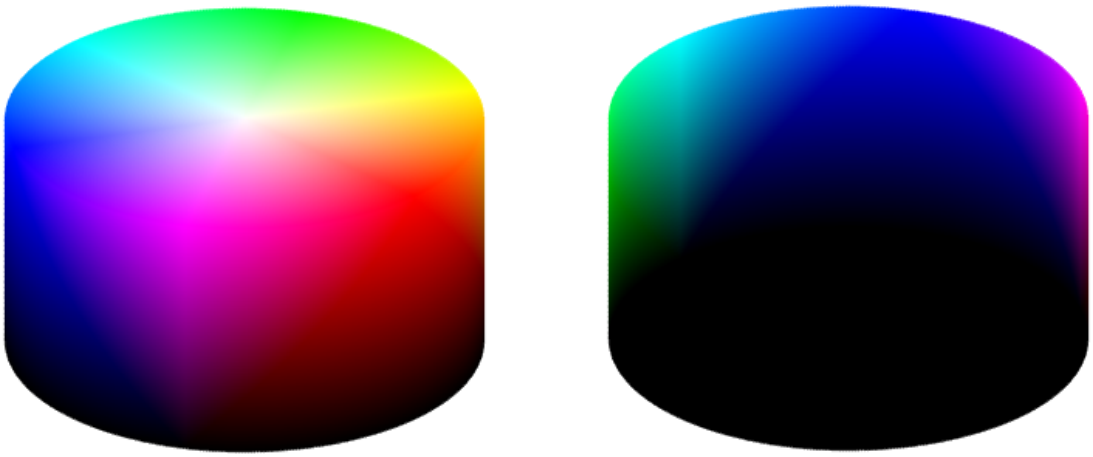
Figure 2.5: The RGB color cube shown from two different perspectives. The R, G, and B color axes and some markings are included. [8]

in the color. The RGB color space is shown in Figure 2.5. These images, and several others, were generated using the code from [8]. Figures generated with this code will contain the citation.

In [9], Hanbury constructs a cylindrical color space which is called the opponent color space (OCS). The OCS is very similar to other frequently used color spaces such as HSV and HSI. Like the RGB space, the OCS uses a triplet of values to describe a color. The three values are the chroma $C$, the hue $H$, and the intensity $I$. The coordinates $(C, H, I)$ correspond to the commonly used cylindrical coordinates $(r, \theta, z)$. The OCS color space can be seen in Figure 2.6. Additionally the top of the cylinder can be seen in Figure 2.7.

Figures 2.6 and 2.7 give some intuition behind what chroma, hue, and intensity stand for. Chroma is perceived roughly as the saturation of a color, such as the bright red versus faded pink. Hue is perceived as different colors such as red and blue. Intensity is the overall brightness and is independent of color.

The relation between the RGB space and OCS is given in [9]. Notice that in Figure



(a) (b)

Figure 2.6: The OCS shown from two different perspectives.

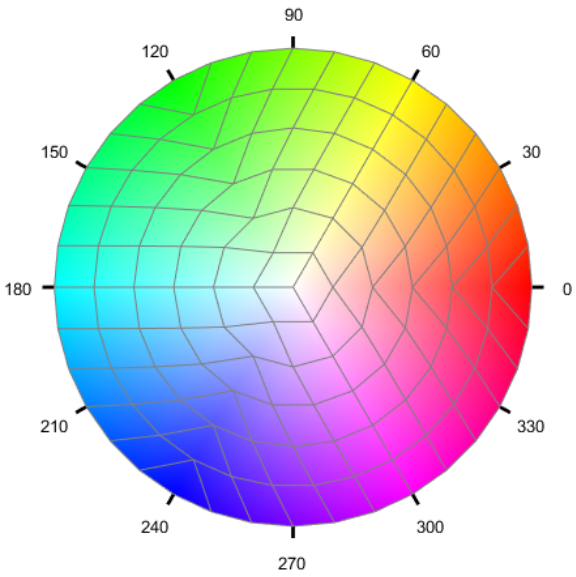



Figure 2.7: The top of the OCS. The hue values are also shown in degrees. [8]

2.7, the red direction corresponds to the unit vector $R = e^{j0} = 1$, the green direction to $G = e^{j\frac{2\pi}{3}}$, and the blue direction to $B = e^{j\frac{4\pi}{3}}$. Given an RGB triplet $[r, g, b]$, the corresponding chroma, hue, and intensity are given by:

$$C = |rR + gG + bB| \tag{2.40}$$

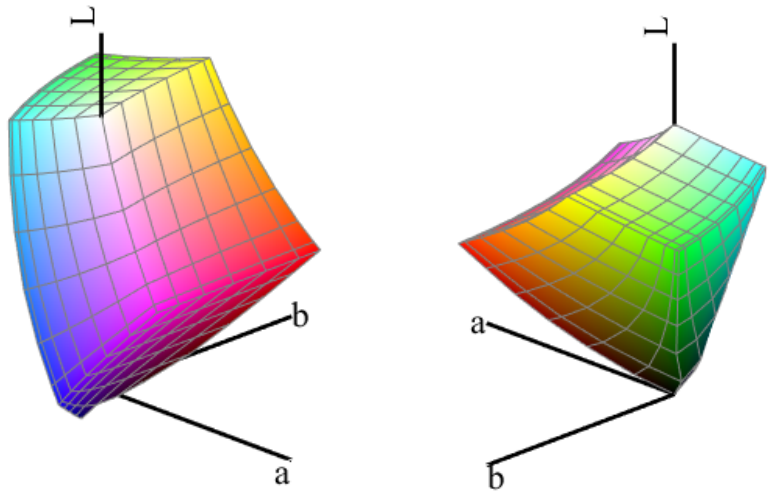
$$H = \text{Arg}(rR + gG + bB) \tag{2.41}$$

$$I = \frac{1}{3}(r + g + b) \tag{2.42}$$

where Arg is the principal argument of a complex value. Using these definitions for the chroma, hue, and intensity, we see that $C$, $I \in [0, 1]$ and $H \in [0, 360°)$ or in radians $H \in [0, 2\pi)$. These can be interpreted geometrically when looking at Figures 2.6 and 2.7. When described in cylindrical coordinates, the chroma corresponds to the distance of the color from the central axis (intensity Axis), hue to the azimuthal angle, and intensity to the height.

One additional property of the OCS can be seen using Figure 2.7. Adding two colors of equal intensity and chroma whose hues differ by $\pi$ radians results in a color on the intensity axis. Two such colors are called complementary colors. It should be noted that the colors red and cyan, green and magenta, and blue and yellow form complementary color pairs.

Another color space that is commonly used is CIE $L^*a^*b^*$ (CIELAB) [10]. CIELAB is also commonly referred to as the Lab color space. The CIELAB color space was created by the International Commission on Illumination so that the three values $L^*$, $a^*$, and $b^*$ closely reflect actual human perception. The lightness value $L^*$ captures how people perceive brightness. The location of the color between green and magenta is captured by $a^*$. Similarly, $b^*$ reflects the position of the color between blue and yellow. Figure 2.8 shows the Lab color space.




(a) (b)

Figure 2.8: The Lab color space shown from two different perspectives. [8]

## 2.3 The Complementary Color Wavelet Transform

The Complementary Color Wavelet Transform (CCWT) is a type of wavelet transform developed by Chen *et al.* [5] that uses the properties of complementary colors. The CCWT takes advantage of the $\frac{2\pi}{3}$ phase differences present among the colors red, green, and blue to observe certain color changes taking place over various scales. In the 2-D case, the CCWT also provides a means of observing changes among certain directions as well.

### 2.3.1 CCWT Wavelets

The CCWT uses three sets of finite impulse response conjugate mirror filters that have a $\frac{2\pi}{3}$ phase difference in the frequency domain. Denoting the three lowpass filters as $h^\theta[n]$ for $\theta \in \{0, \frac{2\pi}{3}, \frac{4\pi}{3}\}$ this phase relation is given by:

$$\frac{\hat{h}^{\frac{2\pi}{3}}(\omega)}{\hat{h}^0(\omega)} = e^{j\frac{2\pi}{3}}, \qquad \frac{\hat{h}^{\frac{4\pi}{3}}(\omega)}{\hat{h}^{\frac{2\pi}{3}}(\omega)} = e^{j\frac{2\pi}{3}}, \qquad \frac{\hat{h}^0(\omega)}{\hat{h}^{\frac{4\pi}{3}}(\omega)} = e^{j\frac{2\pi}{3}} \tag{2.43}$$

Using equations (2.32), (2.37), and (2.38) gives the analysis highpass filters $g^\theta[n]$, the synthesis lowpass filters $\tilde{h}^\theta[n]$, and the synthesis highpass filters $\tilde{g}^\theta[n]$, again for

$\theta \in \{0, \frac{2\pi}{3}, \frac{4\pi}{3}\}$. Additionally, using equations (2.18), (2.19), (2.30), and (2.31) gives us the scaling functions $\phi^\theta(t)$ and wavelet functions $\psi^\theta(t)$. From this point $[n]$ and $(t)$ will be dropped for notational ease.

These wavelets are designed so that they exhibit the following time domain property:

$$\psi^0 + \psi^{\frac{2\pi}{3}} + \psi^{\frac{4\pi}{3}} = 0 \tag{2.44}$$

This property can be visualized through Figure 2.9. In this figure it can also be seen that the energy of these wavelets is localized.

Each of the three wavelets, and its associated conjugate mirror filters, can be used on its own to decompose signals. However, designing them so that they have the properties mentioned allows for the extraction of extra information. This comes from the fact that the same signal can be represented in different ways depending on which of the wavelets is used for the DWT. Using multiple DWTs, each with a different wavelet, gives redundancy that can be exploited as will be shown in the following sections.

It should be noted that CCWT wavelets refers to an entire class of wavelets. However, for this work, only the CCWT wavelets provided in the github link in [5] are used.

### 2.3.2 The 1-D CCWT

Let a 1-D RGB color signal be given by its three channel signals **r**, **g**, and **b**. The **r** signal will correspond to the filters with phase $\theta$, the **g** signal to the filters with phase $\big((\theta + \frac{2\pi}{3}) \bmod 2\pi\big)$, and the **b** signal to the filters with phase $\big((\theta + \frac{4\pi}{3}) \bmod 2\pi\big)$ for $\theta \in \{0, \frac{2\pi}{3}, \frac{4\pi}{3}\}$. Note that the reference phase value does not matter. This means that the filter assigned to the red channel can be any of the three phases $\theta$ as long as the green and blue channel filters have the correct phase relation to the red channel filter.

As suggested in [5], the first layer of filter banks should be doubled phased filters

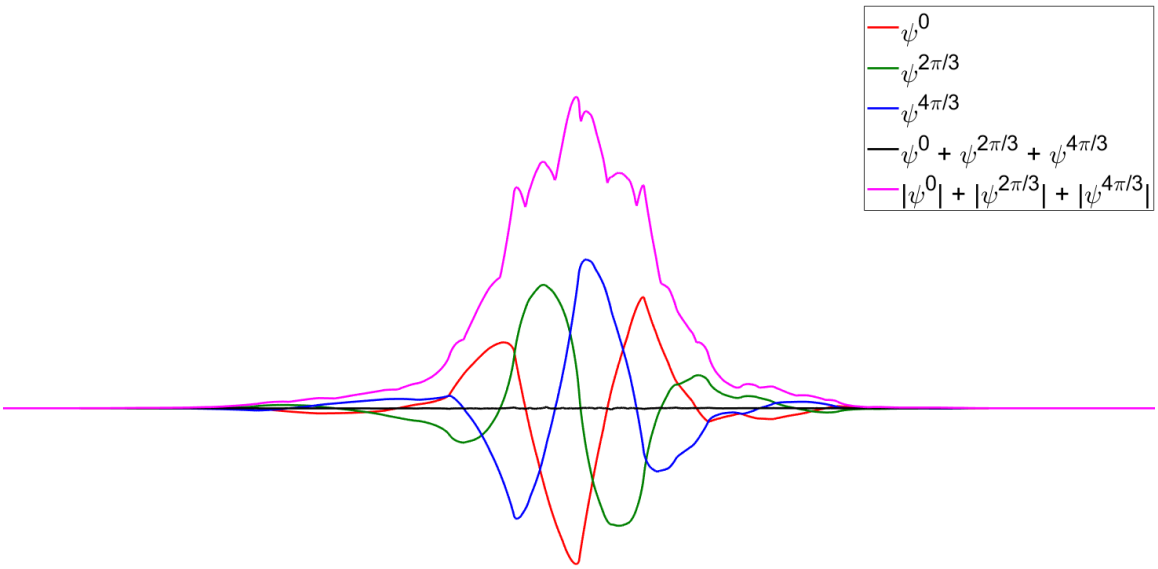



Figure 2.9: Wavelets used in the CCWT, their sum, and the sum of their absolute values. The sum of the three wavelets is almost exactly zero over the entire support.

in order to combat phase errors. That means the phase or $\theta$ of a filter is doubled to $2\theta$. These double phase low and highpass filters will be denoted by $h^{2\theta}$ and $g^{2\theta}$, $\theta \in \{0, \frac{2\pi}{3}, \frac{4\pi}{3}\}$ respectively. The following layers all use the regular filters. For a given $\theta$, each of the three color channels has an assigned low and highpass filter and the corresponding double phase low and highpass filters. Each color channel is first filtered using the double phase low and highpass filters, then decimated by 2. The resulting signal from the lowpass filter is then sent to the regularly-phased filters whose outputs are also decimated by 2. For subsequent layers, the lowpass output is again fed into the regularly-phased filters and the lowpass output is decimated by 2. This process continues for as many layers as desired. After the first layer, the process described is the 1-D DWT.

At this point recall that the red channel was assigned filters with phase $\theta$. There are three choices for the phase value and each is equally valid since the only thing that matters is the phase relation between the channels. This means the filtering can be done with each configuration separately. The results of the three possible assignments

can then be used in tandem to overcome shift variance problems [5]. The process of these analysis filter banks can be easily visualized using Figure 2.10a where each of the three possible assignments is used for each channel. The structure shown is in effect three 1-D DWTs being done in parallel.

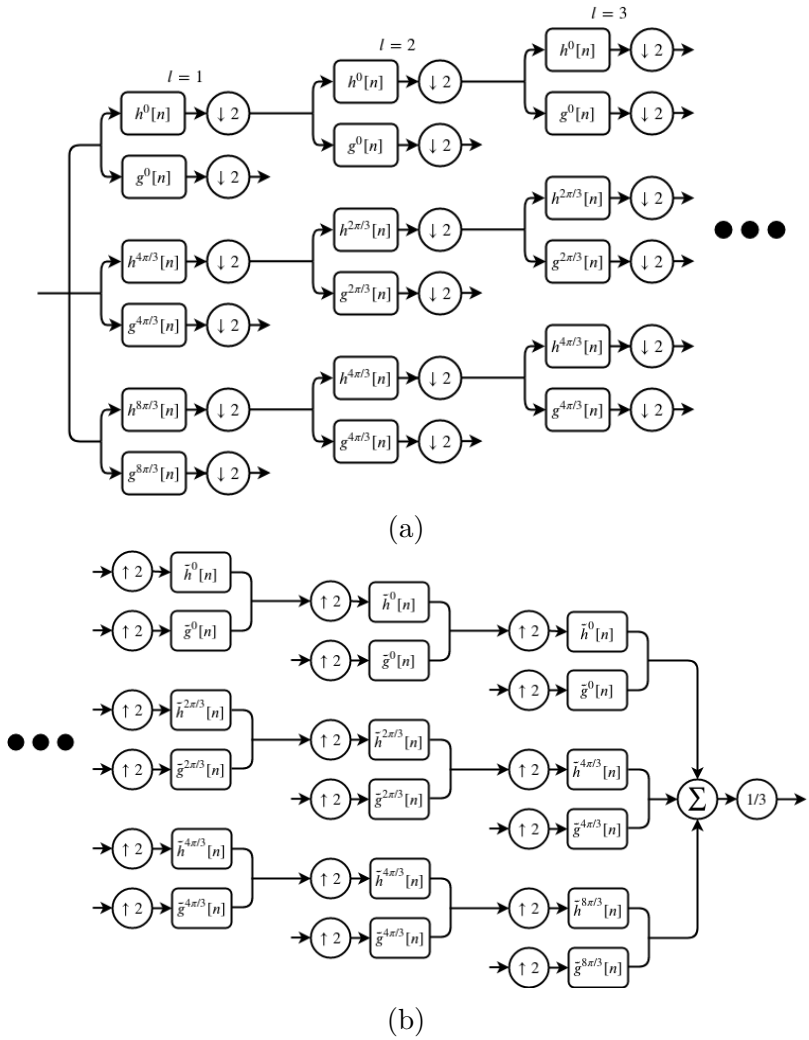
There are a couple of important things to note. The first is that the output of the first layer of filtering (double phase) does not necessarily need to be decimated by 2. This will be referred to as the undecimated version. The second thing to note is that the filtering done by this filter bank can either be done with regular convolution for time domain signals or circular convolution for image signals. This second point becomes more important for the case of the 2-D CCWT.

After each layer $l$ of the filter bank, excluding the first layer since it is double phase, let the wavelet coefficients be denoted by:

$$\begin{aligned} d_l^r &= \left(\mathbf{r}_{l-1} * g^{\theta}\right) \downarrow 2 \\ d_l^g &= \left(\mathbf{g}_{l-1} * g^{(\theta+\frac{2\pi}{3}) \bmod 2\pi}\right) \downarrow 2 \\ d_l^b &= \left(\mathbf{b}_{l-1} * g^{(\theta+\frac{4\pi}{3}) \bmod 2\pi}\right) \downarrow 2 \end{aligned} \tag{2.45}$$

where $\mathbf{r}_{l-1}$, $\mathbf{g}_{l-1}$, and $\mathbf{b}_{l-1}$ represent the outputs from the lowpass filtering and decimation by 2 from the previous layer for the $\mathbf{r}$, $\mathbf{g}$, and $\mathbf{b}$ color channels respectively. They are the approximation coefficients at layer $l-1$. $d_l^r$, $d_l^g$, $d_l^b$ are the detail or wavelet coefficients at layer $l$. Equation (2.45) can also be done using circular convolution.

Using the wavelet coefficients obtained from equation (2.45), several important operators can be defined. The operators in equations (2.46) through (2.50) are defined




(a)

(b)

Figure 2.10: (a) Analysis and (b) synthesis filter banks for 1-D CCWT.

pointwise for each regular phase level $l$.

$$O_l^I(d) = |d_l^r| + |d_l^g| + |d_l^b| \tag{2.46}$$

$$O_l^C(d) = d_l^r + d_l^g + d_l^b \tag{2.47}$$

$$O_l^R(d) = d_l^r - d_l^g - d_l^b \tag{2.48}$$

$$O_l^G(d) = -d_l^r + d_l^g - d_l^b \tag{2.49}$$

$$O_l^B(d) = -d_l^r - d_l^g + d_l^b \tag{2.50}$$

$O^I$ is the intensity operator and it captures all the changes that are occurring. This

means that any change in the color signal will lead to an increase of the value $O^I$. The chroma/White-Black complementary color operator is given by $O^C$. The chroma operator measures the difference in the distance from the white-black axis. This means that any color changes will be detected as a change in the chroma operator but changes that are on the spectrum of black to white (grayscale change) will not be detected. $O^R$, $O^G$, and $O^B$ are the Red-Cyan, Green-Magenta, and Blue-Yellow complementary color operators respectively. They each measure the changes in the signal that are not occurring along their complementary color axis. Figure 2.11 illustrates the responses of the $O^C$, $O^R$, $O^G$, and $O^B$ operators using a $l = 2$ layer decomposition for several different input signals. The right most column shows the outputs for an input signal that is all black but with several strings of white pixels. The chroma operator does not respond to this signal since it is occurring on the white-black axis, while the other operators do register this change. The second column has a red-cyan signal. In this case the red-cyan operator does not perceive any changes but the other operators do see it. The last two columns show the equivalent results for a green-magenta and blue-yellow signal.

Equations (2.46) through (2.50) are for one of the three possible assignments of the filters. When doing all three possible assignments there will be three versions of each of the operators. These can just be averaged together.

The 1-D CCWT is invertible given the outputs $O^C$, $O^R$, $O^G$, and $O^B$. Once the wavelet coefficients are recovered the synthesizing structure in Figure 2.10b can be used. There is a multiplication by 1/3 at the end since the outputs from the three assignments are being pointwise averaged.

### 2.3.3 The 2-D CCWT

The 2-D CCWT, like the 1-D CCWT, is built from single channel wavelet decompositions. Then for RGB images the three single channels are arithmetically manipulated

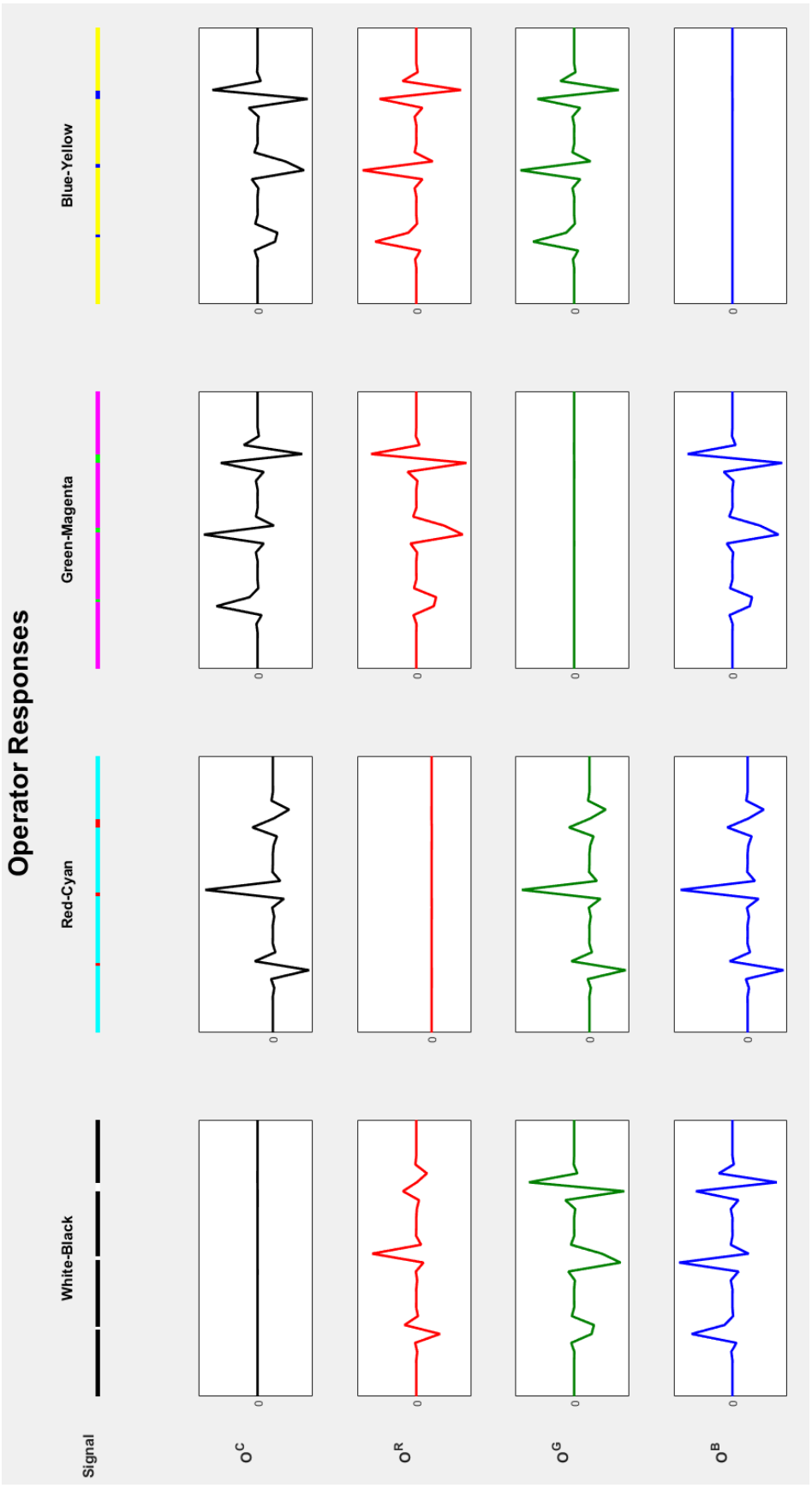



Figure 2.11: The CCWT operator responses to various input signals. Each column shows the signal being processed and the level 2 $O^C$, $O^R$, $O^G$, and $O^B$ operator outputs.

to get the same operators as the 1-D CCWT has.

As was the case for the 1-D CCWT, the first step in taking the 2-D CCWT is to choose which of the three low and highpass filter pairs will be designated as the 0 phase reference. This choice does not affect the results because only the $\frac{2\pi}{3}$ phase relation between the filters matters. This choice also carries over to the associated scaling and wavelet functions for each filter. The derivation of the 2-D CCWT presented here will follow the derivation done in [5]. The math will be presented first and then it will be followed by an explanation.

The following argument follows from the fact that the three wavelets sum to 0 (equation (2.44)). Using this the following three relations can be seen:

$$\left(\phi_h^0 + \phi_h^{\frac{2\pi}{3}} + \phi_h^{\frac{4\pi}{3}}\right)\left(\psi_v^0 + \psi_v^{\frac{2\pi}{3}} + \psi_v^{\frac{4\pi}{3}}\right) = 0 \quad \text{(LH)} \tag{2.51}$$

$$\left(\psi_h^0 + \psi_h^{\frac{2\pi}{3}} + \psi_h^{\frac{4\pi}{3}}\right)\left(\phi_v^0 + \phi_v^{\frac{2\pi}{3}} + \phi_v^{\frac{4\pi}{3}}\right) = 0 \quad \text{(HL)} \tag{2.52}$$

$$\left(\psi_h^0 + \psi_h^{\frac{2\pi}{3}} + \psi_h^{\frac{4\pi}{3}}\right)\left(\psi_v^0 + \psi_v^{\frac{2\pi}{3}} + \psi_v^{\frac{4\pi}{3}}\right) = 0 \quad \text{(HH)} \tag{2.53}$$

In the above the $h$ and $v$ subscripts refer to using the wavelet or scaling function for the rows and columns of an image respectfully. Following the convention presented in section 2.1, the LH stands for using highpass filters for the columns followed by lowpass filters for the rows. For HL the lowpass filters are used for the columns and the highpass for the rows. Equations (2.51) through (2.53) can be distributed to give the three corresponding equations:

$$\phi_h^0\psi_v^0 + \phi_h^0\psi_v^{\frac{2\pi}{3}} + \phi_h^0\psi_v^{\frac{4\pi}{3}} + \phi_h^{\frac{2\pi}{3}}\psi_v^0 + \phi_h^{\frac{2\pi}{3}}\psi_v^{\frac{2\pi}{3}} + \phi_h^{\frac{2\pi}{3}}\psi_v^{\frac{4\pi}{3}} + \phi_h^{\frac{4\pi}{3}}\psi_v^0 + \phi_h^{\frac{4\pi}{3}}\psi_v^{\frac{2\pi}{3}} + \phi_h^{\frac{4\pi}{3}}\psi_v^{\frac{4\pi}{3}} = 0 \tag{2.54}$$

$$\psi_h^0\phi_v^0 + \psi_h^0\phi_v^{\frac{2\pi}{3}} + \psi_h^0\phi_v^{\frac{4\pi}{3}} + \psi_h^{\frac{2\pi}{3}}\phi_v^0 + \psi_h^{\frac{2\pi}{3}}\phi_v^{\frac{2\pi}{3}} + \psi_h^{\frac{2\pi}{3}}\phi_v^{\frac{4\pi}{3}} + \psi_h^{\frac{4\pi}{3}}\phi_v^0 + \psi_h^{\frac{4\pi}{3}}\phi_v^{\frac{2\pi}{3}} + \psi_h^{\frac{4\pi}{3}}\phi_v^{\frac{4\pi}{3}} = 0 \tag{2.55}$$

$$\psi_h^0\psi_v^0 + \psi_h^0\psi_v^{\frac{2\pi}{3}} + \psi_h^0\psi_v^{\frac{4\pi}{3}} + \psi_h^{\frac{2\pi}{3}}\psi_v^0 + \psi_h^{\frac{2\pi}{3}}\psi_v^{\frac{2\pi}{3}} + \psi_h^{\frac{2\pi}{3}}\psi_v^{\frac{4\pi}{3}} + \psi_h^{\frac{4\pi}{3}}\psi_v^0 + \psi_h^{\frac{4\pi}{3}}\psi_v^{\frac{2\pi}{3}} + \psi_h^{\frac{4\pi}{3}}\psi_v^{\frac{4\pi}{3}} = 0 \tag{2.56}$$

It should be noted that just because the sum of these 9 terms in each equation is 0 does not mean each individual term is 0 itself, just like the wavelets in Figure 2.9. These three equations all have the same form:

$$w_h^0 w_v^0 + w_h^0 w_v^{\frac{2\pi}{3}} + w_h^0 w_v^{\frac{4\pi}{3}} + w_h^{\frac{2\pi}{3}} w_v^0 + w_h^{\frac{2\pi}{3}} w_v^{\frac{2\pi}{3}} + w_h^{\frac{2\pi}{3}} w_v^{\frac{4\pi}{3}} + w_h^{\frac{4\pi}{3}} w_v^0 + w_h^{\frac{4\pi}{3}} w_v^{\frac{2\pi}{3}} + w_h^{\frac{4\pi}{3}} w_v^{\frac{4\pi}{3}} = 0 \tag{2.57}$$

given the substitution:

$$w_h w_v = \phi_v \psi_v,\ \psi_h \phi_v,\ \text{or}\ \psi_h \psi_v \tag{2.58}$$

After this one additional set of substitutions is made. These substitutions are:

$$\begin{aligned} w_1 &= w_h^0 w_v^0, & w_2 &= w_h^0 w_v^{\frac{2\pi}{3}}, & w_3 &= w_h^0 w_v^{\frac{4\pi}{3}} \\ w_4 &= w_h^{\frac{2\pi}{3}} w_v^0, & w_5 &= w_h^{\frac{2\pi}{3}} w_v^{\frac{2\pi}{3}}, & w_6 &= w_h^{\frac{2\pi}{3}} w_v^{\frac{4\pi}{3}} \\ w_7 &= w_h^{\frac{4\pi}{3}} w_v^0, & w_8 &= w_h^{\frac{4\pi}{3}} w_v^{\frac{2\pi}{3}}, & w_9 &= w_h^{\frac{4\pi}{3}} w_v^{\frac{4\pi}{3}} \end{aligned} \tag{2.59}$$

This allows for equation (2.57) to be written as:

$$w_1 + w_2 + w_3 + w_4 + w_5 + w_6 + w_7 + w_8 + w_9 = 0 \tag{2.60}$$

The above (equations (2.54) through (2.60)) were just expanding and renaming the components of equations (2.51) through (2.53) to a form that will be more convenient. Each of the expanded equations has 9 terms. Each of these 9 terms is simply filtering an image first column-wise with the vertical filter followed by row-wise filtering with the horizontal filter. Importantly, after the image has both been filtered vertically and horizontally, the output is decimated by a factor of 2 both row-wise and column-wise. Decimation can also alternatively be done column-wise right after the vertical filter has been applied and row-wise right after the horizontal filter is applied. The latter method is more efficient. Again note that the above equations and the following equations do not include any term for the image or decimation. This is done to prevent cluttering.

The derivation of the 2-D CCWT continues by grouping the $w$ terms into sets of

three based on phase. There are four different ways to group the $w$ terms:

$$\begin{array}{lll} w_1 + w_5 + w_9, & w_2 + w_6 + w_7, & w_3 + w_4 + w_8 \\ w_1 + w_6 + w_8, & w_2 + w_4 + w_9, & w_3 + w_5 + w_7 \\ w_1 + w_2 + w_3, & w_4 + w_5 + w_6, & w_7 + w_8 + w_9 \\ w_1 + w_4 + w_7, & w_2 + w_5 + w_8, & w_3 + w_6 + w_9 \end{array} \tag{2.61}$$

Each of the rows only uses each $w$ term once. Each of the 12 separate sets of three has the same thing in common. Taking the sum of the horizontal phases gives an integer multiple of $2\pi$, and the same is true when summing the phase terms of the vertical components. So a set of three has either different phase values for the horizontal component, or the vertical component, or both. Each set in a row in equation (2.61) follows the same phase changes. For example, in the third row of sets the horizontal phase stays constant within the three and the vertical changes ($w_1$, $w_2$, and $w_3$ have the same horizontal phase value and the vertical differs, but the set containing $w_4$, $w_5$, and $w_6$ have a different constant horizontal phase value). In the first two rows both the horizontal and vertical components change. In the third row the horizontal component stays constant and the vertical changes. Lastly, in the fourth row the horizontal changes and the vertical stays constant. This fact will be important later. Also because of equation (2.60) the value of the third column in a row can be found only given the first two columns. All the information given in the previous equation can be encapsulated in the following linear transformation called the separation transform:

$$\mathbf{u} = \mathbf{T}\mathbf{w} \tag{2.62}$$

where:

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 & u_8 & 0 \end{bmatrix}^T$$

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & w_3 & w_4 & w_5 & w_6 & w_7 & w_8 & w_9 \end{bmatrix}^T$$

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

The $u$ terms here are the eight values from the left and center columns of equation (2.61). Following what was mentioned previously the third column of equation (2.61) can be obtained from the eight $u$ values. The last row of the $\mathbf{T}$ matrix represents equation (2.60). It should also be noted that $\mathbf{T}$ is invertible and thus the $w$ terms can be recovered given the $u$ terms.

Remember that the separation transform is done for the three subbands LH, HL, and HH. This gives us three sets of $u$ terms which will be denoted as $\mathbf{u}_{\phi\psi}$, $\mathbf{u}_{\psi\phi}$, and $\mathbf{u}_{\psi\psi}$ corresponding to their three respective subbands. Some of these $u$ terms are 0. The terms that correspond to 0 are the terms from which $\left(\psi^0 + \psi^{\frac{2\pi}{3}} + \psi^{\frac{4\pi}{3}}\right)$ can be factored out. Here is an example:

$$\begin{aligned} u_{5\phi\psi} &= w_{1\phi\psi} + w_{2\phi\psi} + w_{3\phi\psi} \\ &= \phi_h^0\psi_v^0 + \phi_h^0\psi_v^{\frac{2\pi}{3}} + \phi_h^0\psi_v^{\frac{4\pi}{3}} \\ &= \left(\psi_v^0 + \psi_v^{\frac{2\pi}{3}} + \psi_v^{\frac{4\pi}{3}}\right)\phi_h^0 \\ &= (0)\phi_h^0 \\ &= 0 \end{aligned}$$

Using this reasoning and looking at equation (2.61), all the values in the first two rows are not forced to be 0 since $\left(\psi^0 + \psi^{\frac{2\pi}{3}} + \psi^{\frac{4\pi}{3}}\right)$ can not be factored out. For the LH subband the third row is forced to all 0 values but the fourth row is not. For HL the third row is not forced to 0 but the fourth row is. Lastly, for HH both the third and fourth row are forced to 0. This leaves sixteen non-zero $u$ terms and another eight terms that come from the difference of the two columns. The sixteen non-zero $u$ terms are:

$$\begin{aligned} &\text{(LH)} \quad u_{1\phi\psi},\ u_{2\phi\psi},\ u_{3\phi\psi},\ u_{4\phi\psi},\ u_{7\phi\psi},\ u_{8\phi\psi} \\ &\text{(HL)} \quad u_{1\psi\phi},\ u_{2\psi\phi},\ u_{3\psi\phi},\ u_{4\psi\phi},\ u_{5\psi\phi},\ u_{6\psi\phi} \\ &\text{(HH)} \quad u_{1\psi\psi},\ u_{2\psi\psi},\ u_{3\psi\psi},\ u_{4\psi\psi} \end{aligned} \tag{2.63}$$

The other eight terms are given by:

$$\begin{aligned} &\text{(LH)} \quad -u_{1\phi\psi} - u_{2\phi\psi}, \quad -u_{3\phi\psi} - u_{4\phi\psi}, \quad -u_{7\phi\psi} - u_{8\phi\psi} \\ &\text{(HL)} \quad -u_{1\psi\phi} - u_{2\psi\phi}, \quad -u_{3\psi\phi} - u_{4\psi\phi}, \quad -u_{5\psi\phi} - u_{6\psi\phi} \\ &\text{(HH)} \quad -u_{1\psi\psi} - u_{2\psi\psi}, \quad -u_{3\psi\psi} - u_{4\psi\psi} \end{aligned} \tag{2.64}$$

Again these terms derive from equations (2.60) through (2.62).

These terms form eight groups of 3. Each of the groups is oriented in a different direction that is roughly orientated in one of the directions $\frac{n\pi}{8}$ for $n \in \{1, 2, .., 8\}$. The following shows how the twenty-four terms of equations (2.63) and (2.64), which are derived from the wavelet coefficients, orient themselves relative to the horizontal.

$$\begin{array}{llll}
\frac{\pi}{8} \text{ rad} & u_{3\psi\phi}, & u_{4\psi\phi}, & -u_{3\psi\phi} - u_{4\psi\phi} \\
\frac{2\pi}{8} \text{ rad} & u_{3\psi\psi}, & u_{4\psi\psi}, & -u_{3\psi\psi} - u_{4\psi\psi} \\
\frac{3\pi}{8} \text{ rad} & u_{3\phi\psi}, & u_{4\phi\psi}, & -u_{3\phi\psi} - u_{4\phi\psi} \\
\frac{4\pi}{8} \text{ rad} & u_{7\phi\psi}, & u_{8\phi\psi}, & -u_{7\phi\psi} - u_{8\phi\psi} \\
\frac{5\pi}{8} \text{ rad} & u_{1\phi\psi}, & u_{2\phi\psi}, & -u_{1\phi\psi} - u_{2\phi\psi} \\
\frac{6\pi}{8} \text{ rad} & u_{1\psi\psi}, & u_{2\psi\psi}, & -u_{1\psi\psi} - u_{2\psi\psi} \\
\frac{7\pi}{8} \text{ rad} & u_{1\psi\phi}, & u_{2\psi\phi}, & -u_{1\psi\phi} - u_{2\psi\phi} \\
\frac{8\pi}{8} \text{ rad} & u_{5\psi\phi}, & u_{6\psi\phi}, & -u_{5\psi\phi} - u_{6\psi\phi}
\end{array} \tag{2.65}$$

At this point the main part of the single channel portion of the 2-D CCWT has been explained. There just remains some explanations on to how to do this for multiple levels.

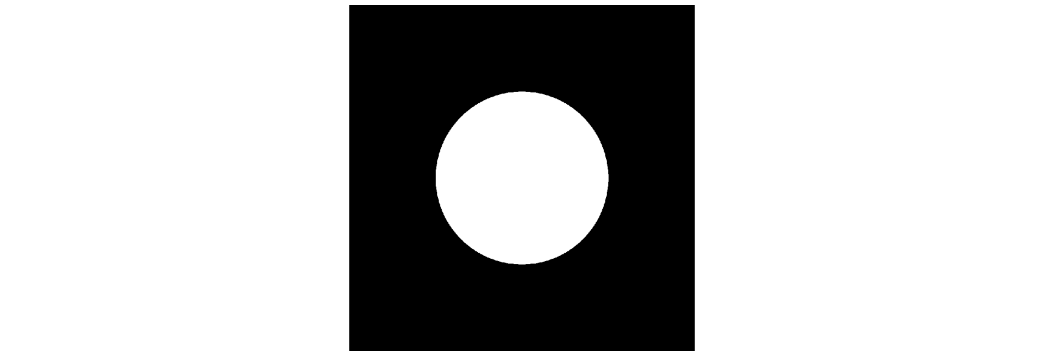
Starting with a single channel image, assign one of the three sets of filters to be the reference filters (0 phase). For the first layer, the corresponding double phase filters are used, just as in the 1-D case. All three sets of filters are used for the rows and the columns. This gives nine total combinations for filtering (all of which are done). Choose one set of filters for the columns and one for the rows and perform a regular discrete wavelet transform. This gives four subimages corresponding to the LL, LH, HL, and HH subbands. Do this for all nine combinations leading to a total of thirty-six subimages. Twenty-seven of these subimages correspond to the twenty-seven terms seen in equations (2.51) through (2.53). The remaining nine are for the analogous nine terms of LL filtering. All these images are decimated by 2 in each dimension (not necessary the first layer in the undecimated case). This is in effect doing nine separate 2-D DWTs where the filtering in the first layer is done using the double phase filters. At this point the separation transform can be performed and the twenty-four $u$ coefficients of interest in equations (2.63) and (2.64) can be found. It

should be noted that the separation transform is done pointwise. Also note that the separation transform and the results from it do not hold for the double phase layer. In the general case, to get subsequent layers, just perform the 2-D DWT with the regular phase filters to the desired level, and then perform the steps detailed previously.

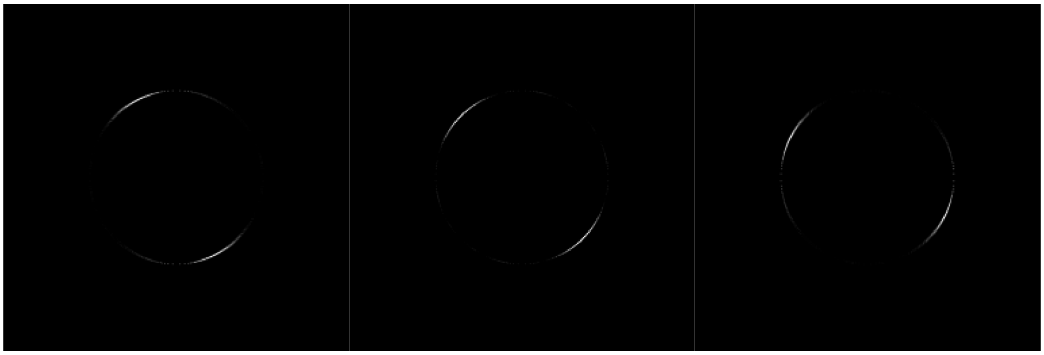
The 2-D CCWTs ability to pick out directional details can be seen in Figure 2.12. The figures shown are displaying the squared sum of all three sets of all $u$ coefficients in the given direction. The eight individual directions are picked out and they are all picked out equally as can be seen from Figure 2.12j where the entire boundary is equally bright.

At this point the same complementary color operators can be defined for the 2-D CCWT. To do this perform the 2-D single channel operations to get the twenty-four coefficients (equations (2.63) and (2.64)) for the R, G, and B channels. For a desired direction there are three coefficients (equation (2.65)). Select one from the R channel and then circularly to the right the coefficients for G and B respectively. For example for the $\frac{3\pi}{8}$ rad direction choose $u_{4\phi\psi}$ for red, $-u_{3\phi\psi} - u_{4\phi\psi}$ for green, and $u_{3\phi\psi}$ for blue. Note that there are three choices for what red can be initially (analogous to the three choices for the reference filter). Due to this, there are three possible ways to calculate the complementary color operators combining the coefficients the same way as in equations (2.46) through (2.50) for the 1-D CCWT. The other two options correspond to either assigning $u_{3\phi\psi}$ for red, $u_{4\phi\psi}$ for green, and $-u_{3\phi\psi} - u_{4\phi\psi}$ for blue or $-u_{3\phi\psi} - u_{4\phi\psi}$ for red, $u_{3\phi\psi}$ for green, and $u_{4\phi\psi}$ for blue for the $\frac{3\pi}{8}$ rad direction. Once a choice has been made, assign coefficients from the same column in equation (2.65) to the same channel. The outputs of the operators for all three assignments can be averaged together but it is not necessary to do so.

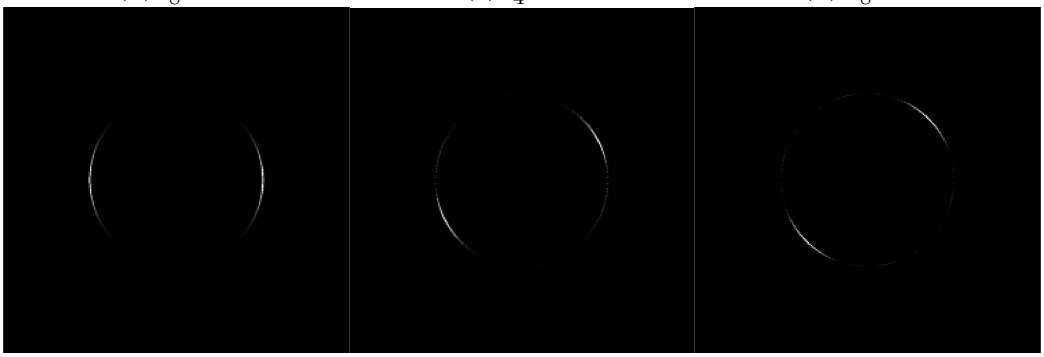
The 2-D complementary color operators can be seen in action in Figure 2.13. In the figure all the operators show the sum squared of the all the directional coefficients. In (a) all the operators pick up the three circles though at various intensities. The



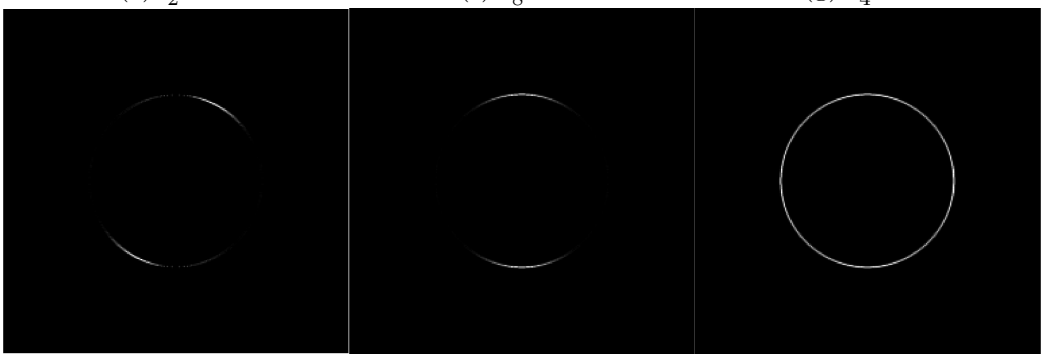
(a) Original Image



(b) $\frac{\pi}{8}$ rad (c) $\frac{\pi}{4}$ rad (d) $\frac{3\pi}{8}$ rad



(e) $\frac{\pi}{2}$ rad (f) $\frac{5\pi}{8}$ rad (g) $\frac{3\pi}{4}$ rad

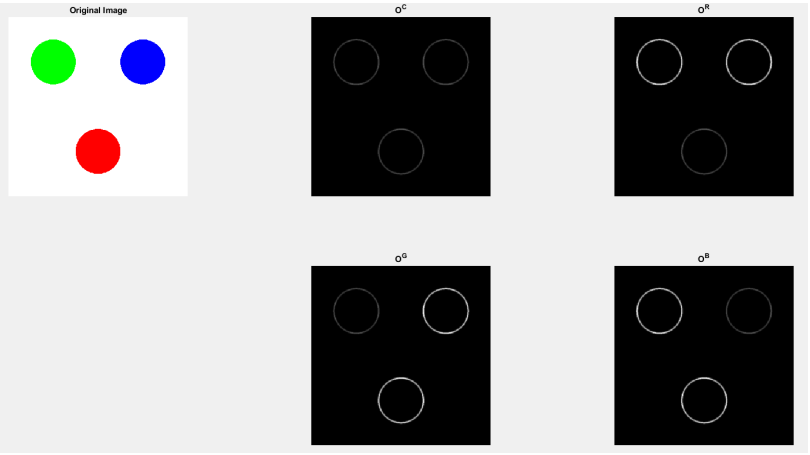


(h) $\frac{7\pi}{8}$ rad (i) $\pi$ rad (j) All directions

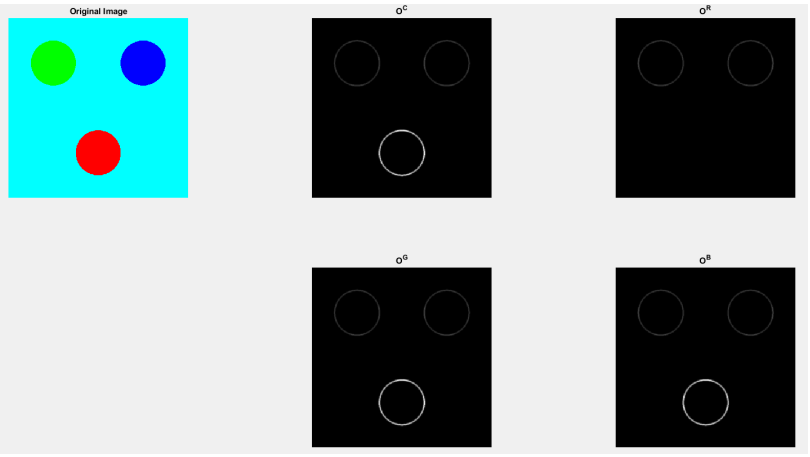
Figure 2.12: (a) the original image. (b) - (i) shows the ability of the 2-D CCWT to extract features along 8 directions using a level 2 CCWT coefficients. (j) shows all the orientations together.

intensity does vary in the $O^R$, $O^G$, and $O^B$ operators, depending on which color change is occurring. When the background is cyan as in (b) the $O^R$ operator does not detect the red circle. This is due to the change being along the red-cyan complementary color axis. The analogous results would be seen for a magenta and yellow background for the $O^G$ and $O^B$ operators. Lastly in (c) the $O^C$ operator does not pick up any changes but the others do. In all the operator outputs across the figure the entire boundary of the circles are seen again due to the great directional sensitivity of the CCWT.

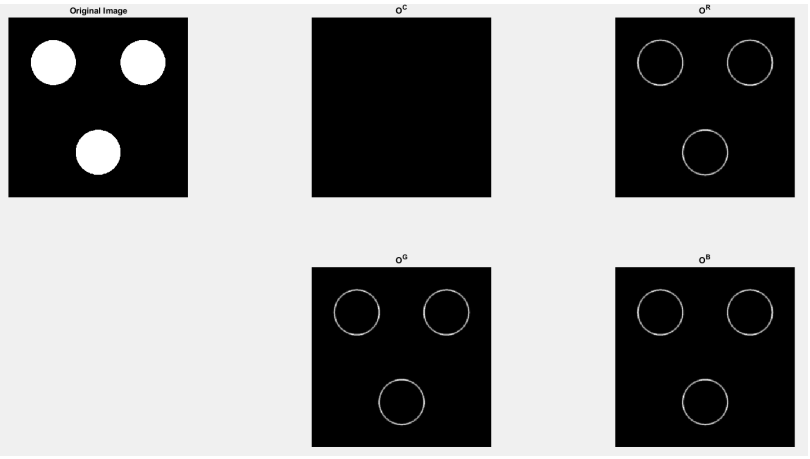
The $u$ coefficients in equation (2.65) can be recovered given the chroma and the three other complementary color operators. Once the coefficients are retrieved, the inverse separation transform ($\mathbf{T}^{-1}$) can be performed to get the original wavelet coefficients from the 2-D filtering. After that, the synthesis filters can be applied and upsampling can be performed to reconstruct the original image.




(a) White Background




(b) Cyan Background




(c) Black Background

Figure 2.13: The outputs of the chroma, red-cyan, green-magenta, and blue-yellow complementary color operators for various inputs.

# Chapter 3

# Saliency Maps

## 3.1 The What and Why of Saliency Maps

When people look at an image they tend to fixate on certain regions in the image. These parts correspond to the salient parts of an image. As mentioned in section 1.1, knowing which areas of an image are salient, i.e. having a saliency map, is very useful for certain image processing tasks. However, the task of generating a saliency map is not simple and when one is generated, how do we know that it is representative of what people actually tend to fixate on? In order to evaluate a saliency map it is helpful to have a fixation map. A fixation map is a grayscale image that highlights the parts of an image where people, on average, tend to look. Fixation maps are obtained by showing an image to a group of observers and recording which points they fixated on. This is usually done with eye tracking gear. To help in the development of saliency map generation algorithms, many datasets have been created that provide images and their associated fixation maps. One such dataset is CAT2000 [11] which is used in this work.

Given an image the goal of a saliency map generation algorithm is to generate an estimate of the fixation map called the saliency map. Many algorithms for this

task exist. Figure 3.1 displays an image, its fixation map, and several saliency maps generated using different algorithms. As can be seen, the outputs of these algorithms vary drastically. It is therefore important to be able to assess the quality of a saliency map in a quantitative way. Two metrics that do this are discussed in section 3.3.

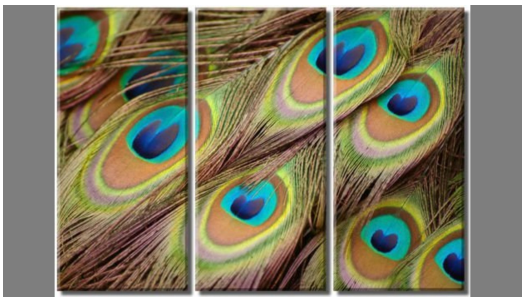
## 3.2 A Method of Generating Saliency Maps

The method for saliency map generation proposed in this work is rooted in the method developed by Ishikura *et al.* [4]. The method in [4] will be referred to as the base method and is discussed here.

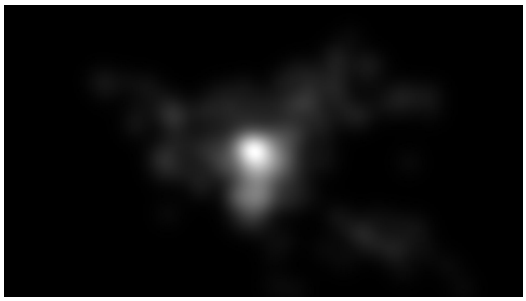
The base method consists of several main steps. The image, represented in the Lab color space, is split into its three separate channels. The three channels of the image are filtered at various scales using the CenSurE technique [17]. After applying CenSurE, the three channels are combined at each scale to get a series of color difference spaces. The color difference spaces are used to identify potential saliency points, also called extrema. Global information from the image such as color and directional rarity is then computed for each of the extrema. The saliency map is then formed using a Gaussian mixture. The Gaussian mixture is formed using the extrema and global information. An overview of the base method can be seen in Figure 3.2. This method is explained in more detail below.

Given an image represented in the Lab color space, the three channels are filtered using CenSurE. The goal of CenSurE is to approximate the Laplacian-of-Gaussian (LoG) filter to increase computational speed. The impulse response, $h(x, y, \sigma)$, of the LoG filter, at scale $\sigma$ is defined by:

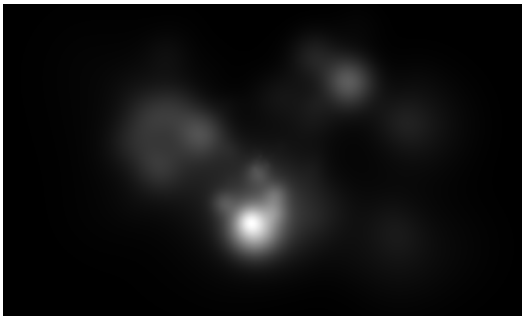
$$h(x, y, \sigma) = \frac{x^2 + y^2 - 2\sigma^2}{2\pi\sigma^6} \exp\left( - \frac{x^2 + y^2}{2\sigma^2} \right) \tag{3.1}$$



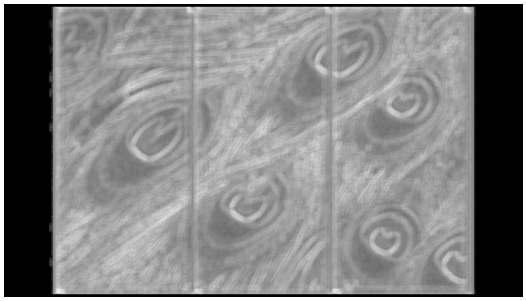
(a) Original Image



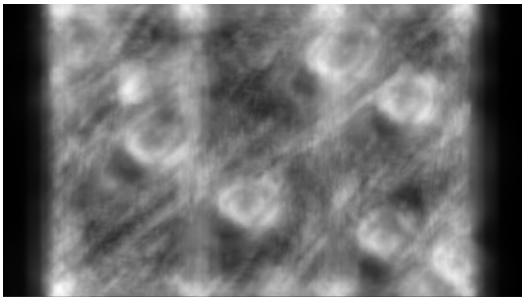
(b) Fixation Map



(c) Proposed Method using $O^C$

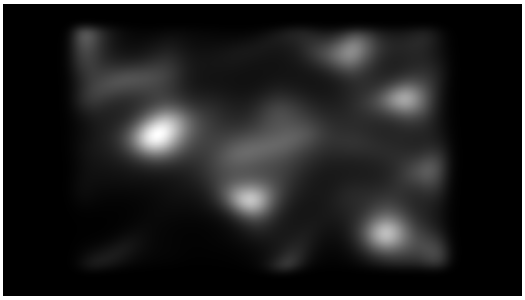


(d) AIM [12]

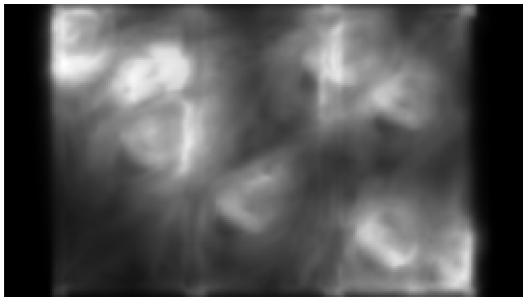


(e) CIWaM [13]

(f) GBVS [14]



(g) FES [15]



(h) CAS [16]

Figure 3.1: (a) The original image (Art 149) taken from the CAT2000 dataset. (b) The corresponding fixation map. (c) - (h) Saliency maps generated using various methods.

In [4], $\sigma \in [1, 2, ..., \sigma_{max}]$ where $\sigma_{max} = \lfloor \min(\frac{W}{3}, \frac{H}{3}) \rfloor$ and W and H are the image width and height respectively. This function is also a well known wavelet, called the

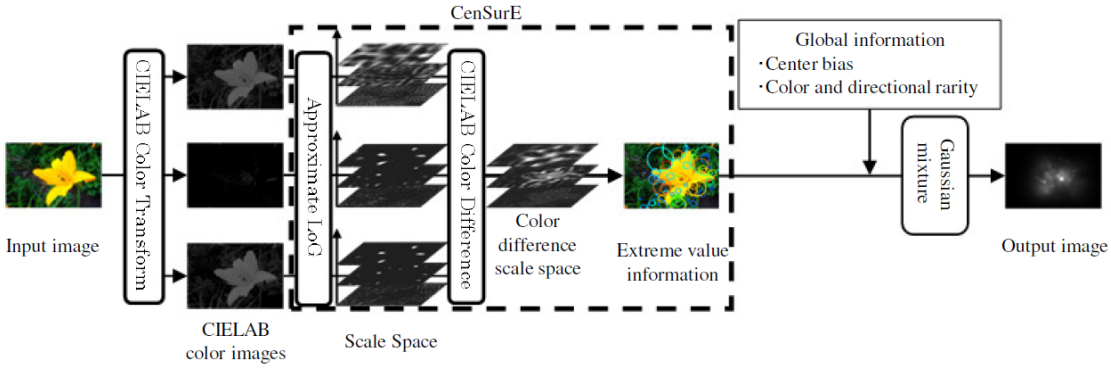



Figure 3.2: An overview of the base method. This image was taken from [4].

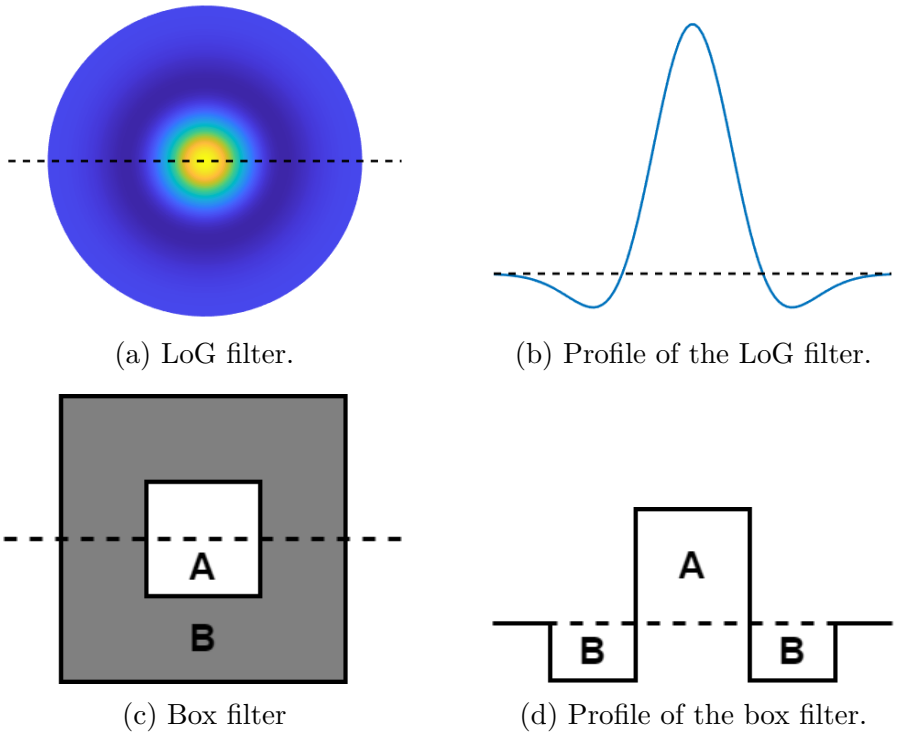
Mexican hat wavelet. It should be noted that the LoG filter here is sampling the wavelet at discrete coordinates $(x, y)$ that would then be convolved with the image. The LoG filter is not a conjugate mirror filter. Using the LoG filter will extract local information in each of the three channels at various scales. However, convolving with the LoG filter is computationally expensive as each pixel in the image would need to be considered. CenSurE makes use of the fact that the magnitude of $h(x, y, \sigma)$ decreases exponentially to approximate it. One way CenSurE does this is by approximating the LoG filter with the difference of two square regions, called the box filter. This approximation has a finite extent, meaning that all the pixels are not used in every stage of filtering, thus decreasing the number of computations. The LoG and box filters can be seen in Figure 3.3.

To apply the box filter, CenSurE makes use of the integral image $s(x, y)$, which is given by:

$$s(x, y) = \sum_{u=1}^{x} \sum_{v=1}^{y} i(u, v) \tag{3.2}$$

where $i(u, v)$ is the value of $L^*$, $a^*$, or $b^*$ at the coordinate $(u, v)$. The sum of the pixel values in a rectangular region from $(x_1, y_1)$ to $(x_2, y_2)$ is then given by:

$$I = s(x_2, y_2) - s(x_1 - 1, y_2) - s(x_2, y_1 - 1) + s(x_1 - 1, y_1 - 1) \tag{3.3}$$




(a) LoG filter. (b) Profile of the LoG filter.

(c) Box filter (d) Profile of the box filter.

Figure 3.3: The LoG filter and the CenSurE approximation to it.

The output of applying the box filter at various scales can now be found easily. This is because the sums of rectangular regions are already calculated in the integral images. All that needs to be done is to add and subtract the appropriate rectangles. The filter outputs for each channel at scale $\sigma$ are given by:

$$\Delta L^*_\sigma = L^*_{A,\sigma} - L^*_{B,\sigma} \tag{3.4}$$

$$\Delta a^*_\sigma = a^*_{A,\sigma} - a^*_{B,\sigma} \tag{3.5}$$

$$\Delta b^*_\sigma = b^*_{A,\sigma} - b^*_{B,\sigma} \tag{3.6}$$

using the CenSurE method. In these equations $L^*_{A,\sigma}$, $a^*_{A,\sigma}$, and $b^*_{A,\sigma}$ denote the sum of the respective channel values in region A. Analogously, $L^*_{B,\sigma}$, $a^*_{B,\sigma}$, and $b^*_{B,\sigma}$ denote the sum of the respective channel values in region B. These can be computed using equation (3.3). $\Delta L^*_\sigma$, $\Delta a^*_\sigma$, and $\Delta b^*_\sigma$ represent the channel differences at a given scale. However, in [4], equations (3.4) through (3.6) are slightly altered. Instead of summing

all the values of all pixels in a given region, an average is taken. The altered equations for the channel differences are given by:

$$\Delta L^*_\sigma = \frac{1}{M_A} L^*_{A,\sigma} - \frac{1}{M_B} L^*_{B,\sigma} \tag{3.7}$$

$$\Delta a^*_\sigma = \frac{1}{M_A} a^*_{A,\sigma} - \frac{1}{M_B} a^*_{B,\sigma} \tag{3.8}$$

$$\Delta b^*_\sigma = \frac{1}{M_A} b^*_{A,\sigma} - \frac{1}{M_B} b^*_{B,\sigma} \tag{3.9}$$

where $M_A$ and $M_B$ represent the number of pixels in region A and region B respectively. This averaging also makes the channel difference values between different scales comparable. This is because the larger scales, whose box filters encompass more pixels, are treated the same as smaller scales, whose box filters encompass less pixels.

Once the channel differences are computed for each scale, the perceptual color difference, $\Delta C_\sigma(x, y)$, at coordinate $(x, y)$ and scale $\sigma$ can be computed point wise by:

$$\Delta C_\sigma = \sqrt{(\Delta L^*_\sigma)^2 + (\Delta a^*_\sigma)^2 + (\Delta b^*_\sigma)^2} \tag{3.10}$$

Given the perceptual color differences at all scales, we can now start searching for potentially salient points.

Potentially salient points (extrema) are points, $\Delta C_\sigma(x, y)$, that have the largest value among all the pixels surrounding them in space $(x, y)$ and scale $\sigma$. Figure 3.4 illustrates the 26 surrounding points (gray) that the pixel of interest (black) is compared against when checking to see if it is an extremum. The white pixels in this figure refer to the other $\Delta C$ pixels that are not being considered. Using this method, all extreme values are found and the corresponding coordinate $(x, y, \sigma)$ and perceptual color difference value $\Delta C_\sigma(x, y)$ are saved.

Let $\{(x_i, y_i, \sigma_i)\}$, $i = 1, 2, ..., N$ be the set of locations of the N extrema and $\{\Delta C_{\sigma_i}(x_i, y_i)\}$ the set of corresponding color differences. Each of the locations corresponds to a circle. The center of the $i^{th}$ circle is $(x_i, y_i)$ and its radius is determined

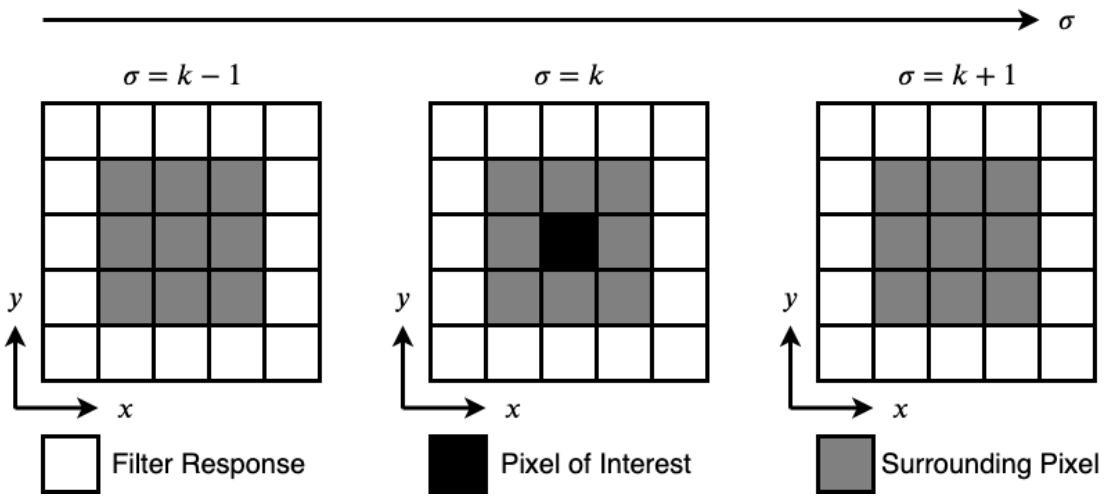



Figure 3.4: Nearby pixels in space and scale.

(a) Original Image (b) Extreme values

Figure 3.5: The extreme values and their corresponding areas of influence. These images were taken from [4].

by $\sigma_i$. Larger scales correspond to larger circles. This means that larger scales have a broader influence. An example of the circles corresponding to extrema can be seen in Figure 3.5.

Now that we have points that are potentially salient, they have to be weighted. The weights will represent how salient each circular region actually is in comparison to the other circular regions. The method used in [4] to assign these weights is based on color and directional rarity as well as location relative to the center of the image. To compute the color rarity, start with the original Lab representation of the picture. Construct a three-dimensional histogram $P^{all}(L^*, a^*, b^*)$ with five bins for each of the

three values ($5^3 = 125$ total bins) using the entire image. Similarly, for each extemum $i$, construct a histogram using only the pixels within the circular region of radius $\sigma_i$. Denote the histogram for this extremum $P_i^{in}(L^*, a^*, b^*)$. Using these two histograms, the histograms for colors appearing outside the circular region of the extremum is given by:

$$P_i^{out}(L^*, a^*, b^*) = P^{all}(L^*, a^*, b^*) - P_i^{in}(L^*, a^*, b^*) \tag{3.11}$$

This will be used to find the color rarity later in this section.

To find the directional rarity, we need to find distributions for the directions present in an image. The directional rarities will be found scale-wise using $\Delta L_\sigma^*$. For a given scale define the horizontal and vertical gradient maps as:

$$f_\sigma^{horz}(x, y) = \Delta L_\sigma^*(x + 1, y) - \Delta L_\sigma^*(x - 1, y) \tag{3.12}$$

$$f_\sigma^{vert}(x, y) = \Delta L_\sigma^*(x, y + 1) - \Delta L_\sigma^*(x, y - 1) \tag{3.13}$$

With these two equations, the gradient magnitude $m_\sigma(x, y)$ and direction $\theta_\sigma(x, y)$ are defined as:

$$m_\sigma(x, y) = \sqrt{f_\sigma^{horz}(x, y)^2 + f_\sigma^{vert}(x, y)^2} \tag{3.14}$$

$$\theta_\sigma(x, y) = \arctan\left(\frac{f_\sigma^{horz}(x, y)}{f_\sigma^{vert}(x, y)}\right) \tag{3.15}$$

At this point the gradient magnitude and direction will be used to construct histograms, similar to what was done for color rarity. The directional histogram $P_\sigma^{all}(\theta)$ for scale $\sigma$, using all the gradient magnitudes and directions, is constructed in the following way. Round each direction $\theta_\sigma(x, y)$ to the nearest degree. The rounded direction values take on values from the set $\{-90°, -89°, ..., 90°\}$. The directional histogram $P_\sigma^{all}(\theta)$ will then have 181 bins. When a directional value of $\theta$ occurs, increment the corresponding bin in the histogram by the value $m_\sigma(x, y)$ whose location $(x, y)$ gave rise to that $\theta$. Once this is done, for each extremum found at scale $\sigma$ construct the

histogram $P_i^{in}(\theta)$ for the directions appearing in the circular region surrounding the extremum. Similarly to what was done for the color histograms, the histograms of the directions appearing outside the circular regions at a scale $\sigma$ are given by:

$$P_i^{out}(\theta) = P_\sigma^{all}(\theta) - P_i^{in}(\theta) \tag{3.16}$$

At this point all the histograms are normalized so that they become probability mass functions (PMF). The notation used for the histograms is now used for the PMFs. The color and directional PMFs for each extremum can now be used to calculate the color and directional rarities. For each extremum define $L^*_{i,max}$, $a^*_{i,max}$, $b^*_{i,max}$, and $\theta_{i,max}$ by:

$$(L^*_{i,max}, a^*_{i,max}, b^*_{i,max}) = \underset{L^*,a^*,b^*}{\operatorname{argmax}} P_i^{in}(L^*, a^*, b^*) \tag{3.17}$$

$$\theta_{i,max} = \underset{\theta}{\operatorname{argmax}} P_i^{in}(\theta) \tag{3.18}$$

Then the color rarity $R_i^{clr}$ and directional rarity $R_i^{dir}$ for an extremum are obtained from the :

$$R_i^{clr} = -\log\left(P_i^{out}(L^*_{i,max}, a^*_{i,max}, b^*_{i,max})\right) \tag{3.19}$$

$$R_i^{dir} = -\log\left(P_i^{out}(\theta_{i,max})\right) \tag{3.20}$$

These values are then normalized so that the largest color and directional rarities are one:

$$\tilde{R}_i^{clr} = \frac{R_i^{clr}}{\max_j\{R_j^{clr}\}} \quad \text{and} \quad \tilde{R}_i^{dir} = \frac{R_i^{dir}}{\max_j\{R_j^{dir}\}} \tag{3.21}$$

The reason for defining color and directional rarity this way is as follows. Equations (3.17) and (3.18) are used to find the color and direction that is found most frequently in the circular region around extremum $i$. Equations (3.11) and (3.16) then tell us how likely it is to find that color and direction outside the circular region. If the color or direction appears less frequently outside the circular region compared proportionally

to the inside it is considered rarer. This notion of color and directional rarity is quantified with equations (3.19) and (3.20).

In [4], the authors include a center bias. This is done because researchers have found that humans tend to focus towards the central area of an image [18]. They do this by defining an extra weight for each extremum. The center bias weight for extremum $i$, $w_i$, is given by:

$$w_i = \frac{1}{2\pi\sigma_i^2} \exp\left( - \frac{(x_{mid} - x_i)^2 + (y_{mid} - y_i)^2}{2\sigma_i^2} \right) \tag{3.22}$$

where $(x_{mid}, y_{mid})$ denotes the coordinate of the center of the image. The center bias weights are also normalized so that the largest weight value is one:

$$\tilde{w}_i = \frac{w_i}{\max_j\{w_j\}} \tag{3.23}$$

At this point all the necessary values have been computed. The saliency map, $SM$, is now defined via a Gaussian mixture. This means that $SM$ is the sum of multiple scaled and translated 2-D Gaussian functions. $SM$ is given by:

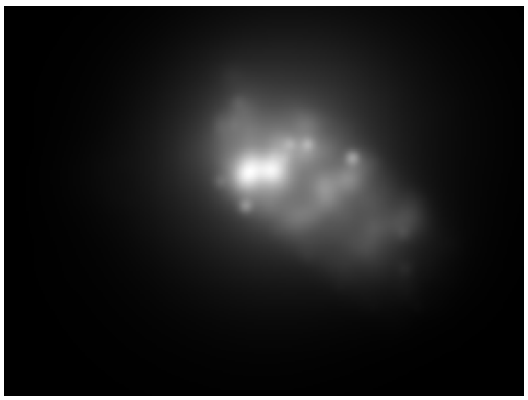
$$SM(x, y) = \sum_{i=1}^{N} \frac{\Delta C_{\sigma_i}(x_i, y_i)\tilde{w}_i \tilde{R}_i^{clr} \tilde{R}_i^{dir}}{2\pi\sigma_i^2} \exp\left( - \frac{(x_i - x)^2 + (y_i - y)^2}{2\sigma_i^2} \right) \tag{3.24}$$

The saliency map is then normalized so that it spans the range $[0, 1]$. An example of a saliency map generated using this method can be seen in Figure 3.6.

## 3.3 Scoring a Saliency Map

Looking at a saliency map, a person can qualitatively say whether it accurately reflects the most eye-catching regions of an image. However, the quality of a saliency map, assessed this way, can vary wildly based on the person judging. Thus there is a need for metrics that quantitatively assess a saliency map. Several such metrics are described in [19]. Two of the metrics in [19] are considered for the purposes of this project and

(a) Original Image



(b) Saliency Map

Figure 3.6: The original image and its corresponding saliency map generated using the base method. These images were taken from [4].

are explained here.

The first metric is the similarity metric (SIM). The similarity metric takes both the saliency map, $SM$, and the ground truth fixation map, $FM$, normalized such that:

$$\sum_{(x,y)} SM(x,y) = \sum_{(x,y)} FM(x,y) = 1 \tag{3.25}$$

The SIM metric is then defined as:

$$\text{SIM}(SM, FM) = \sum_{(x,y)} \min\big(SM(x,y), FM(x,y)\big) \tag{3.26}$$

The output of the SIM metric is a value in the range $[0, 1]$. A SIM value of 1 represents a perfect match between the generated saliency map and the fixation map. The idea behind the SIM metric is to pointwise compare the two maps and see how similar they are. Any differences between the two leads to a penalty, decreasing the overall score.

The second metric used is Pearson's Correlation Coefficient (CC), also known as the linear correlation coefficient. CC is calculated as:

$$CC(SM, FM) = \frac{cov(SM, FM)}{\sigma_{SM}\sigma_{FM}} \tag{3.27}$$

where *cov* is the covariance of two random variables and $\sigma_{SM}$ and $\sigma_{FM}$ represent the stanrdard deviations of the saliency map and fixation map respectively. In this context the values in the saliency and fixation maps are treated as samples from distributions of random variables, obtained by computing histograms. The value of CC lies in the range $[-1, 1]$. A higher CC value denotes that the saliency map more accurately reflects the fixation map.

# Chapter 4

# Generating Saliency Maps Using the CCWT

## 4.1 Why Use the CCWT?

As seen in Chapter 3, the authors of [4] used the Lab representation of an image to find the perceptual color differences $\Delta C$ at various scales. They then used these perceptual color differences to calculate potentially salient points, from which the saliency map was generated. The CCWT outputs several types of color differences seen at different scales. These are the four complementary color operators $O^C$, $O^R$, $O^G$, and $O^B$. Each of these four operators can, with some tweaks to the method shown in section 3.2, separately be used to calculate a saliency map for a given input image. Depending on which color operator is used in lieu of $\Delta C$, we expect the saliency map to highlight different aspects of an image. This is expected because the CCWT operators themselves highlight different changes occurring within an image (recall Figure 2.13). The CCWT also has great performance when it comes to picking out color differences occurring in many different directions. This should help when finding extreme values.

## 4.2 The Proposed Method

As mentioned in the previous section, the goal of the method being proposed here is to replace $\Delta C$ from the base algorithm [4] with one of the four color operators from the CCWT. Blindly swapping in the color operator leads to several problems. The first major problems occur during extrema detection. In the base method each color difference space $\Delta C_\sigma$ is the same size as the original image. This is not the case for the color operators as there is decimation occurring in the CCWT. The problem here is comparing the value of a pixel with the surrounding pixel values at one scale above and one scale below. It is not obvious which pixels they are. Another problem that arises during extrema detection is that the values at adjacent scales are not comparable. The base method solved this problem by altering the CenSurE technique (equations (3.7) through (3.9)). This solution is not applicable when using the CCWT, so another work around must be found. The next problem occurs when computing the directional rarity. By using the CCWT we have no need to calculate $\Delta L^*_\sigma$. It is necessary to find some equivalent for this because converting from the RGB color space, used by the CCWT, to the Lab color space and then computing the $\Delta L^*_\sigma$ values for each scale is computationally expensive. These hurdles are overcome in the method proposed below.

Given an RGB image of width $W$ and height $H$, the first thing that needs to be decided is how many regular phase levels will be used for the undecimated 2-D CCWT. The reason why the undecimated version is chosen is because the dimensions of the image are already being decreased with each regular phase level. The image can only be decimated so many times before it is unusable. So the undecimated version is chosen to get an extra regular phase level. The number of regular phase levels used is precisely the number of scales that we will have. This choice will provide more information for the saliency map algorithm. Keeping this in mind, the number of regular phase levels, $l$, is at most $l_{max} = \lfloor \log_2(\min(W, H)) \rfloor$. Performing the undecimated CCWT to this

many levels will result in one of the image dimensions being too small to decimate again. Even though the CCWT can be done to that many levels, it is not done because it will cause issues later when we look for extreme values. To avoid this issue, the CCWT will only use $l = l_{max} - 3$ regular phase levels. However, decimating by two $l$ times is not necessarily possible for all image dimensions. To ensure this is possible, the original image is extended with zeros so that both $W$ and $H$ are divisible by $2^l$. For example, in the case of an image of size 1920 by 1080, $l_{max} = 10$ so the undecimated CCWT will use $l = 7$ regular phase levels. The width, 1920, is divisible by $2^7$ but the height is not. The image is then extended by adding rows of zeros so that its height is 1152, which is divisible by $2^7$. This ensures that the CCWT can be performed without any issues.

With the desired number of regular phase levels chosen, the undecimated CCWT is performed and the four color operators $O^C$, $O^R$, $O^G$, and $O^B$ are obtained. As mentioned in section 2.3.3, each of the color operators can be calculated in three different ways depending on which column of equation (2.65) is assigned to to each color channel. Computing this with one choice gives eight images corresponding to the eight directions at each scale. Doing this for all three possibilities gives, in effect, twenty-four images (3 for each direction) at each scale. Take all twenty-four, square them individually and then sum them together. This is done so that all of the values are positive, meaning that when we are looking for extrema, we will search for local maxes as in the base method. When referring to a color operator from now on, we refer to this sum of the squares of the twenty-four images.

At this point we have four color operator scale spaces ($O^C$, $O^R$, $O^G$, and $O^B$) that can play the role of $\Delta C$ from the base algorithm. A saliency map can be generated from each of them individually. The rest of the method is explained using the $O^C$ scale space but it can be substituted with $O^R$, $O^G$, or $O^B$ without issue.

Given the scale space $O^C$, the first step in calculating the saliency map is to find

the extreme values in space and scale, just as in the base method. As mentioned earlier, there are some issues that must be addressed when using $O^C$ instead of $\Delta C$. The issues that need to be addressed are the fact that the values at adjacent scales, as given, are not directly comparable and that the pixels surrounding a pixel of interest are not defined as in the base method.

To solve the problem of adjacent scales not having comparable values, we first need to understand why it is occurring. When the CCWT is computed, going from one scale to the next requires filtering. The act of filtering adds energy to the signal. This additional energy is precisely the reason why the values at adjacent scales are not comparable. Adjusting for the filter energy will resolve this issue.

The energy of a discrete-time filter $h$ with discrete-time Fourier transform $\hat{h}(w)$ is given by:

$$E = \sum_{\omega} |\hat{h}(\omega)|^2 \tag{4.1}$$

It is not immediately clear what the energies of the six filters used in the regular phase portion of the CCWT are in comparison to one another. To see if there exists a relation, we start with denoting the energy of the lowpass filter $h^0$ as $E_h^0$. The energies, $E_h^{\frac{2\pi}{3}}$ and $E_h^{\frac{4\pi}{3}}$, of the other two lowpass filters, $h^{\frac{2\pi}{3}}$ and $h^{\frac{4\pi}{3}}$, can be found using the phase relations in equation (2.43).

$$E_h^{\frac{2\pi}{3}} = \sum_{\omega} |\hat{h}^{\frac{2\pi}{3}}(\omega)|^2 = \sum_{\omega} |\hat{h}^0(\omega) e^{j\frac{2\pi}{3}}|^2 \ = \sum_{\omega} |\hat{h}^0(\omega)|^2 = E_h^0 \tag{4.2}$$

$$E_h^{\frac{4\pi}{3}} = \sum_{\omega} |\hat{h}^{\frac{4\pi}{3}}(\omega)|^2 = \sum_{\omega} |\hat{h}^0(\omega) e^{-j\frac{2\pi}{3}}|^2 = \sum_{\omega} |\hat{h}^0(\omega)|^2 = E_h^0 \tag{4.3}$$

This shows that the CCWT lowpass filter have the same energy. To find the energies $E_g^0$, $E_g\frac{2\pi}{3}$, and $E_g^{\frac{4\pi}{3}}$, of the highpass filters $g^0$, $g^{\frac{2\pi}{3}}$, and $g^{\frac{4\pi}{3}}$, we use the property of these conjugate mirror filters presented in equation (2.29), repeated here:

$$\hat{g}(\omega) = e^{-j\omega}\hat{h}^*(\omega + \pi)$$

With this property, it can be concluded that the energy of filter $h$ is the same as filter $g$. This means that filters that form a conjugate mirror pair have equal energy. Therefore:

$$E_h^0 = E_h^{\frac{2\pi}{3}} = E_h^{\frac{4\pi}{3}} = E_g^0 = E_g^{\frac{2\pi}{3}} = E_g^{\frac{4\pi}{3}} \tag{4.4}$$

All the filters have equal energy, meaning that all the DWTs occuring in the 2-D CCWT are the same in this respect. Scaling by a factor of $\sqrt{E_h^0}$ leads to values that can be fairly compared. So multiplying the values at scale $\sigma$ ($\sigma$ regular phase levels performed) by this factor allows for comparison with the values at scale $\sigma+1$. Instead of multiplying the scale $\sigma$ values, the scale $\sigma+1$ values can be divided by this factor to make the two scales comparable. This solves one of the problems that occurs during the search for extrema.

The other main problem in the extrema search is to identify which pixels to compare values against. The surrounding pixels at the same scale as the pixel of interest remain the same. This means that they are just the eight pixels adjacent to the pixel of interest. There is no correct way to associate the pixel of interest to the surrounding pixels at the scale above and below. If the pixel of interest is at scale $\sigma$ then the scale below refers to $\sigma - 1$, not as deeply filtered, and the scale above refers to $\sigma + 1$ which has been filtered further down.

Let the pixel of interest be given by the coordinate $(x_p, y_p, \sigma_p)$. The most natural way of corresponding the pixel of interest to the pixel directly below it at scale $\sigma_p - 1$ is to double the first two coordinate values. This would correspond to the pixel $(2x_p, 2y_p, \sigma_p - 1)$. However, it can also be argued that $(2x_p \pm 1, 2y_p, \sigma_p - 1)$, $(2x_p, 2y_p \pm 1, \sigma_p - 1)$, and $(2x_p \pm 1, 2y_p \pm 1, \sigma_p - 1)$ are also directly below the pixel of interest. These nine pixels can be thought of as almost being directly below the pixel of interest. To also include some of the surrounding pixels that are not directly below, use the seven pixels with either $x$ coordinate value $2x_p - 2$ or with $y$ coordinate value $2y_p - 2$. Similarly, the most natural way to correspond the pixel of interest to

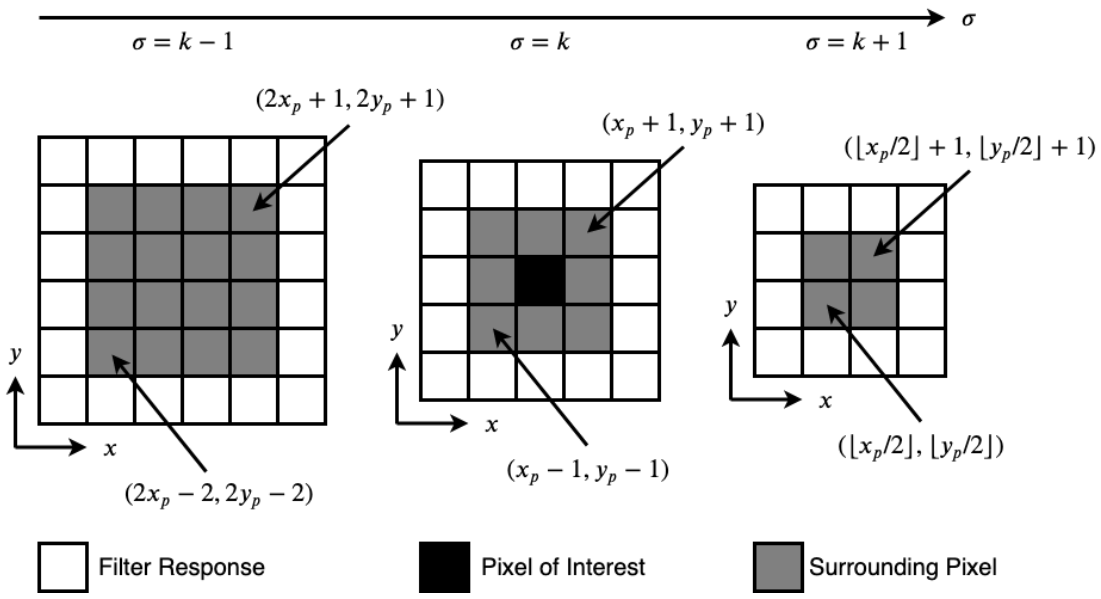



Figure 4.1: Surrounding pixels in space and scale.

the pixel directly above, at scale $\sigma_p + 1$, is to halve the coordinate values. With this reasoning, the pixel directly above is given by the coordinate $(\lfloor \frac{x_p}{2} \rfloor, \lfloor \frac{y_p}{2} \rfloor, \sigma_p + 1)$. Expressed this way, the pixel directly above is defined when either $x_p$ or $y_p$ is odd. The other surrounding pixels at the scale above are given by $(\lfloor \frac{x_p}{2} \rfloor + 1, \lfloor \frac{y_p}{2} \rfloor, \sigma_p + 1)$, $(\lfloor \frac{x_p}{2} \rfloor, \lfloor \frac{y_p}{2} \rfloor + 1, \sigma_p + 1)$, and $(\lfloor \frac{x_p}{2} \rfloor + 1, \lfloor \frac{y_p}{2} \rfloor + 1, \sigma_p + 1)$. We are only comparing against four pixels on the scale above. All the pixels being used to evaluate if the pixel of interest is an extrema can be seen in Figure 4.1.

Now that the values of pixels at adjacent scales are comparable and the nearby pixels in space and scale are defined, the extrema can be found. As in the base method, a pixel of interest is an extreme value if it has the maximum value among all of its surrounding pixels.

Just as in the base method, let $\{(x_i, y_i, \sigma_i)\}$, $i = 1, 2, ..., N$ be the set of locations of the N extrema and $\{O^C_{\sigma_i}(x_i, y_i)\}$ the set of corresponding color differences. Each of the extremum has a circle associated with. The radius of the circle for extremum $i$ is $2^{\sigma_i + 1}$. An example of these circles can be seen Figure 4.2. At the bottom edge of this image there are a lot of circles that seem to be centered below the image.

This comes from the fact that this image was extended with zeros (not shown). The extremeum found on the boundary of the original image and the extension will not play a significant part when generating the saliency map as the center bias weight will diminish their effect.

(a) Original Image

(b) Extreme values (using $O^C$)

Figure 4.2: (a) The original image (Outdoor Natural 141) from the CAT2000 dataset. (b) The extreme values and their area of influence.

At this point the color and directional rarities can be computed. In this process the

extended image will be used again. Starting with the color rarity, start with the RGB representation of the extended image. Construct the three-dimensional histogram $P^{all}(R,G,B)$ with five bins for each of the three values. Then for each extremum $i$, construct a histogram using only the pixels, in the extended image, within a circle of radius $2^{\sigma_i+1}$ centered about $(x_i 2^{\sigma_i}, y_i 2^{\sigma_i})$. The factor $2^{\sigma_i}$ associates the pixel location at scale $\sigma_i$ to the corresponding pixel location in the full size image. The histogram for this extremum will be denoted $P_i^{in}(R,G,B)$. Using these two histograms, the histogram for the colors appearing outside the circle can be calculated by:

$$P_i^{out}(R,G,B) = P^{all}(R,G,B) - P_i^{in}(R,G,B) \tag{4.5}$$

This is the same as equation (3.11) that was done for the base method, except using the RGB color space instead of the Lab color space.

In order to compute the directional histograms that the base method uses we have to overcome another problem. In the base method, the scale spaces $\Delta L^*_\sigma$ are used to compute the horizontal and vertical gradient maps (equations (3.12) and (3.13)). As mentioned, nothing similar to this is computed in the CCWT. One possible way of computing something similar is to do the following. At scale $\sigma$, decimate the extended image by $2^\sigma$ in both dimensions. For each point $(x,y)$ in the decimated image compute the difference vectors:

$$t_\sigma^{horz}(x,y) = (R,G,B)_{(x+1,y)} - (R,G,B)_{(x-1,y)} \tag{4.6}$$

$$t_\sigma^{vert}(x,y) = (R,G,B)_{(x,y+1)} - (R,G,B)_{(x,y-1)} \tag{4.7}$$

Then the horizontal and vertical gradient map equivalents are:

$$f_\sigma^{horz}(x,y) = \sum_{n=1}^{3} \left(t_\sigma^{horz}(x,y)\right)_n \tag{4.8}$$

$$f_\sigma^{vert}(x,y) = \sum_{n=1}^{3} \left(t_\sigma^{vert}(x,y)\right)_n \tag{4.9}$$

The gradient magnitude $m_\sigma(x, y)$ and direction $\theta_\sigma(x, y)$ are then defined the same way as in the base method (equations (3.14) and (3.15)), repeated here for convenience.

$$m_\sigma(x, y) = \sqrt{f_\sigma^{horz}(x, y)^2 + f_\sigma^{vert}(x, y)^2}$$
$$\theta_\sigma(x, y) = \arctan\left(\frac{f_\sigma^{horz}(x, y)}{f_\sigma^{vert}(x, y)}\right)$$

These are then used to construct the directional histogram $P_\sigma^{all}(\theta)$. This histogram is constructed in a slightly different matter from the base method. In the base method, $\theta_\sigma(x, y)$ was rounded to the nearest degree and then $m_\sigma(x, y)$ was added to the bin corresponding to the rounded angle. There were 181 bins in the base method due to this. In place of this, 30 equally spaced bins from $-90°$ to $90°$ are used. Then, as before, the bin in which $\theta_\sigma(x, y)$ lands is incremented by $m_\sigma(x, y)$. This change was made because of the fact that higher scales (more decimated) have less pixels. Splitting the values into too many bins would be less informative of how the values are spread out. After this is done we have $P_\sigma^{all}(\theta)$. The next step is to calculate the directional histograms inside and outside the circular regions corresponding to each extremum. For extremum $i$ the pixels inside the circle can be seen in Figure 4.3.

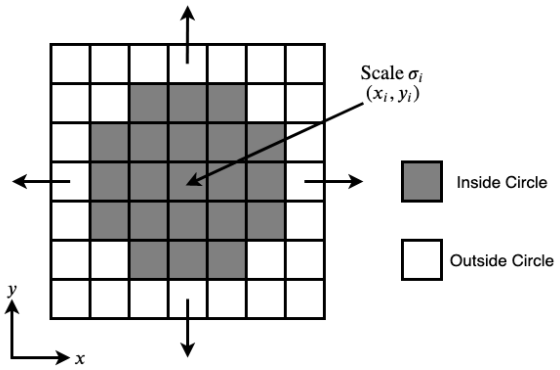



Figure 4.3: Pixels inside circle corresponding to extremum $i$ at scale $\sigma_i$.

The pixels inside the circle corresponding to extremum $i$ are then used to calculate the histogram $P_\sigma^{in}(\theta)$. Then the directional histogram of the directions appearing outside

the circle at scale $\sigma$ is given by:

$$P_i^{out}(\theta) = P_\sigma^{all}(\theta) - P_i^{in}(\theta) \tag{4.10}$$

which is the same as equation (3.16). Now that all the necessary histograms have been calculated, they are normalized so that they become PMFs. Using the PMFs the color and directional rarity are computed using equations (3.17) through (3.21). These equations, adapted for the RGB color space, are shown below:

$$(R_{i,max}, G_{i,max}, B_{i,max}) = \underset{R,G,B}{\operatorname{argmax}}\, P_i^{in}(R, G, B) \tag{4.11}$$

$$\theta_{i,max} = \underset{\theta}{\operatorname{argmax}}\, P_i^{in}(\theta) \tag{4.12}$$

The rarities are then given by:

$$R_i^{clr} = -\log\left(P_i^{out}(R_{i,max}, G_{i,max}, B_{i,max})\right) \tag{4.13}$$

$$R_i^{dir} = -\log\left(P_i^{out}(\theta_{i,max})\right) \tag{4.14}$$

The rarities are again normalized:

$$\tilde{R}_i^{clr} = \frac{R_i^{clr}}{\max_j\{R_j^{clr}\}} \quad \text{and} \quad \tilde{R}_i^{dir} = \frac{R_i^{dir}}{\max_j\{R_j^{dir}\}} \tag{4.15}$$

The center bias weight as shown in equation (3.22) is slightly altered. It is altered by introducing a radius factor $\rho$. The role of the radius factor is to allow for tuning of how aggressive the center bias is. With this new factor the center bias for extremum $i$ is given by:

$$w_i = \frac{1}{2\pi\sigma_i^2} \exp\left(-\frac{(x_{mid} - x_i 2^{\sigma_i})^2 + (y_{mid} - y_i 2^{\sigma_i})^2}{2\rho^2\sigma_i^2}\right) \tag{4.16}$$

The radius factor is squared to ensure a positive value. A bigger $|\rho|$ means that the center bias is weaker, while a smaller value means that the center bias is stronger. The $x_i$ and $y_i$ terms are scaled by $2^{\sigma_i}$ to match the location of the extrema to its

location in the full size image. The center bias weights are also normalized by:

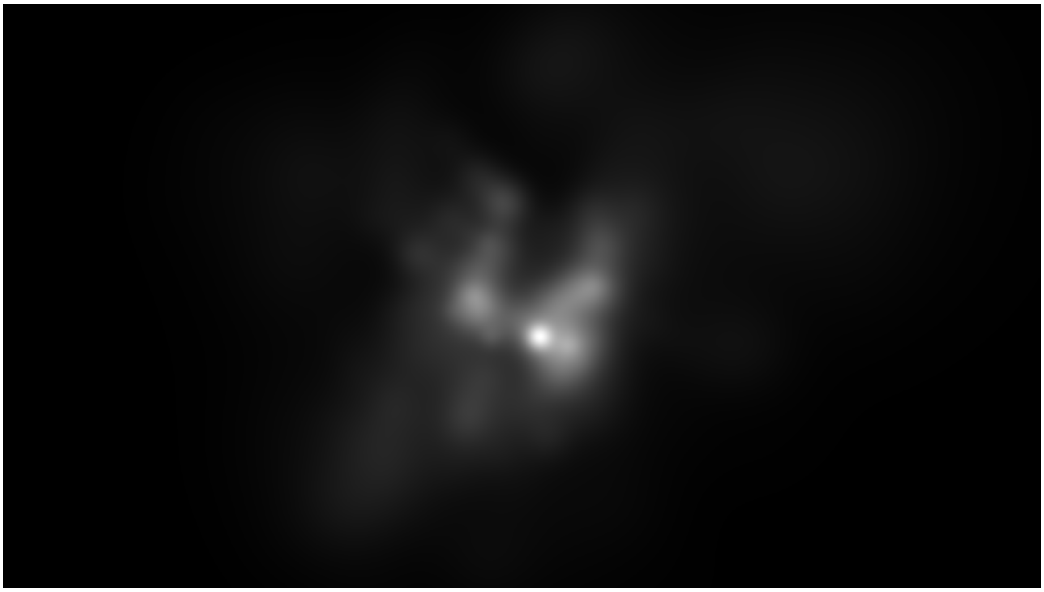
$$\tilde{w}_i = \frac{w_i}{\max_j\{w_j\}} \tag{4.17}$$

All the necessary values needed to generate the saliency map have been calculated at this point. Now, similarly to the base method, the saliency map, $SM$, is defined via a Gaussian mixture.

$$SM(x,y) = \sum_{i=1}^{N} \frac{O^C_{\sigma_i}(x_i,y_i)\sqrt{E^0_h}^{l-\sigma_i}\tilde{w}_i\tilde{R}^{clr}_i\tilde{R}^{dir}_i}{2\pi\sigma_i^2} \exp\left(-\frac{(x_i 2^{\sigma_i}-x)^2+(y_i 2^{\sigma_i}-y)^2}{2\sigma_i^2}\right) \tag{4.18}$$

where $l$ is the number of regular phase levels used in the CCWT. This equation differs from equation (3.24) in form. The term $\delta = O^C_{\sigma_i}(x_i,y_i)\sqrt{E^0_h}^{l-\sigma_i}$ has replaced $\Delta C_{\sigma_i}(x_i,y_i)$. The $\delta$ term captures the value of the color difference of an extremum and includes the scaling factor that needs to be included to make values at different scales comparable. Another slight difference can be seen in the exponential term. The location of the extremum $i$ at scale $\sigma_i$ has to be matched with its location in the full size saliency map. This is done by interpolating the $x$ and $y$ coordinates by a factor of $2^{\sigma_i}$. This was also done for the center bias weight. The saliency map is the normalized so that it spans the range $[0,1]$. Again, $O^C$ was used for this explanation but can be replaced with the other three color operators. Figure 4.4 shows the saliency generated using this new method.

(a) Original Image



(b) Saliency Map (using $O^C$)

Figure 4.4: The saliency map generated using the proposed method.

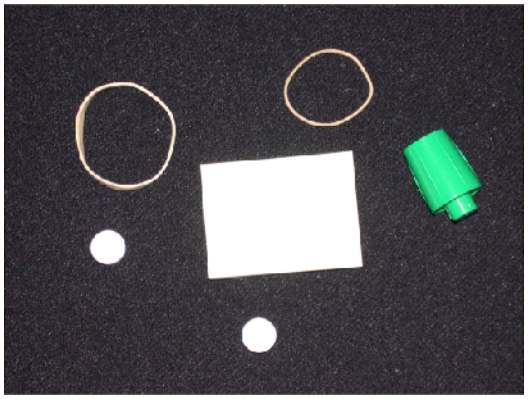
# Chapter 5

# Evaluating the Proposed Method

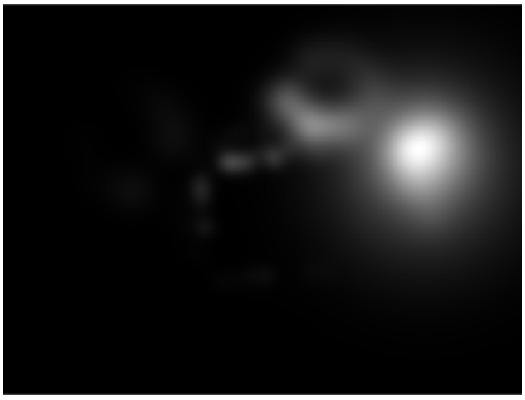
## 5.1 Qualitative Results

In this section the proposed method is evaluated qualitatively. Images will be shown that highlight the advantages and downfalls of this method. Explanation will be given on why they occur.

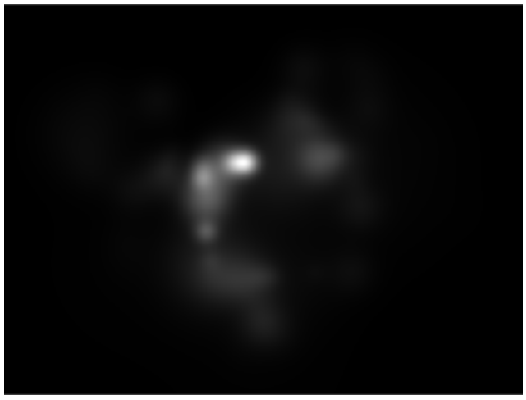
The proposed method has one key advantage over the base method [4]. The advantage is that four different saliency maps can be generated for a single image. An example can of this can be seen in Figure 5.1. As can be seen, each of the saliency maps is unique. The $O^C$ captures the green marker cap and the right rubber band. The other three capture varying amounts of the edges of the white card, the rubber bands, and the bottom white circle. It is not surprising that using $O^R$, $O^G$, and $O^B$ produce similar outputs in this case. This is because the transition from the background to the card happens along the black-white axis (intensity axis). Any small deviation from the axis will be picked up by each of the three color operators. Therefore it is not surprising that those three saliency maps are similar. Note that the saliency maps shown here, and throughout this work, are all generated using a radius factor of $\rho = 4$.



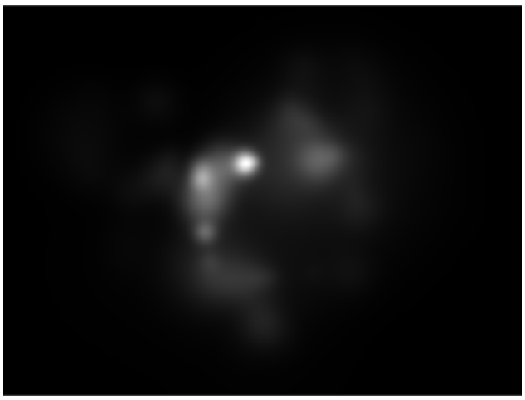
(a) Original Image



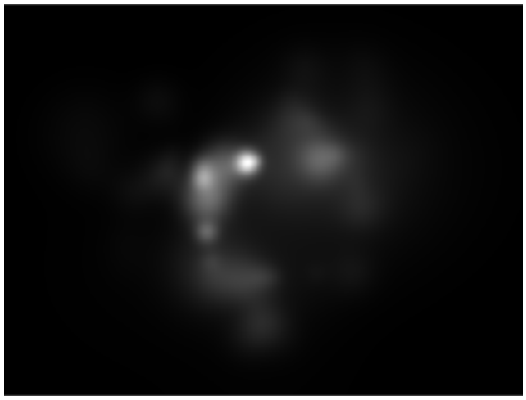
(b) Using $O^C$



(c) Using $O^R$



(d) Using $O^G$

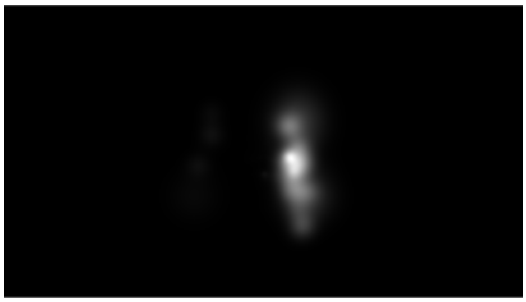


(e) Using $O^B$

Figure 5.1: (a) The original image pillsetc from MATLAB. (b)-(e) The four saliency maps generated using the proposed method.

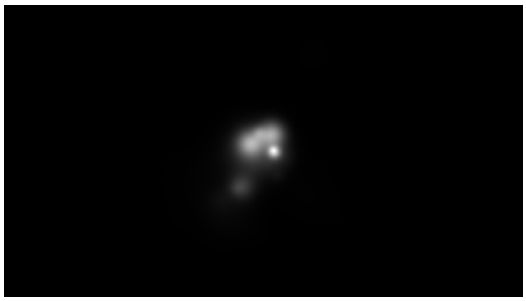
In Figure 5.2 two saliency maps can be seen. Both arguably pick out the most prominent part of their original image. As can be seen in the bottom row of Figure 5.2, this method also works for grayscale images. A grayscale image can be thought

(a) Action 061



(b) Using $O^C$

(c) Black White 101



(d) Using $O^R$

Figure 5.2: The left column contains the original image from the CAT2000 dataset. The right column contains one of the four saliency maps generated for the image.

of as an RGB image where each channel is the same.

At this point it should be noted that for grayscale images the $O^C$ operator should be ideally zero at all scales. However, due to limited machine precision, it is not exactly zero. This means that there will still be local extreme values and saliency maps can be generated from them. The $O^C$ saliency map should be disregarded because of this. Another property of grayscale images is that the the $O^R$, $O^G$, and $O^B$ saliency maps are all exactly the same. This comes from the fact that the R, G, and B channels are exactly the same. This causes these three color operators to be identical (recall Figure 2.13c). If they are identical then the saliency maps are going to be as well. This means that grayscale images only produce one saliency map.

This method does not perform well for images that are not very detailed and are not color-rich. An obvious example of this can be seen in Figure 5.3. The image draws the watchers gaze towards the center of the cup. However, the only color changes oc-

curring are along the outline of the cup. This means that saliency map will capture the outline and not the whole of the cup. In addition to this, only two colors, disregarding the gray padding, are present in the image. This prohibits us from getting meaningful color rarity information. Due to this, the proposed method is not recommended for low detail images with limited color. It should be noted that typical grayscale images, such as Figure 5.2c, contain more than two colors and are thus considered color-rich in this context.

(a) Original Image (b) Saliency Map (using $O^B$)

Figure 5.3: (a) The original image (Sketch 017) from the CAT2000 dataset. (b) The saliency map generated using the proposed method with $O^B$.

## 5.2 Quantitative Results

In order to quantitatively score the proposed algorithm, the two evaluation metrics mentioned in section 3.3 were used on the CAT2000 dataset [11]. The CAT2000 dataset contains 2,000 images falling into twenty different categories. The categories capture a broad variety of image types. The categories are: “Action,” “Affective,” “Art,” “Black White,” “Cartoon,” “Fractal,” “Indoor,” “Inverted,” “Jumbled,” “Line Drawing,” “Low Resolution,” “Noisy,” “Object,” “Outdoor Man-Made,” “Outdoor Natural,” “Pattern,” “Random,” “Satellite,” “Sketch,” and “Social.” Each category contains 100 images with their corresponding fixation maps.

In order to compare the performance of the proposed method to other saliency map generation algorithms, the results presented in [4] are used. In [4] the base method,

explained in section 3.2, was developed. The authors tested the base method among several other algorithms. These included: AIM [12], GBVS [14], CAS [16], ITTI [20], SR [21], HFT [22], RARE'12 [23], SUN [24], ICH [25], and BMS [26, 27]. The base method performed better than these methods on the entire dataset using the SIM and CC metrics. Tables 5.1 through 5.3 show the performance of the proposed method against the base method and some of the other saliency map generation algorithms. The tables contain the results when using only one of the four color operators for an image, denoted PM $O$. They also include the results when the highest scoring of the four saliency maps was taken, denoted PM B. In the tables, Entire Dataset refers to using all 2,000 images in the CAT2000 dataset while Entire Dataset* excludes the 200 images contained in the "Line Drawing" and "Sketch" categories. The reason why they are excluded will be mentioned later.

| Algorithm / Categories | Base | PM B | PM $O^C$ | PM $O^R$ | PM $O^G$ | PM $O^B$ | GBVS | RARE'12 | BMS |
|---|---|---|---|---|---|---|---|---|---|
| Entire Dataset | **0.570** | 0.562 | 0.485 | 0.477 | 0.478 | 0.474 | 0.475 | 0.427 | 0.447 |
| Action | 0.541 | **0.579** | 0.539 | 0.465 | 0.467 | 0.471 | 0.553 | 0.543 | 0.519 |
| Affective | 0.542 | **0.563** | 0.495 | 0.497 | 0.493 | 0.494 | 0.523 | 0.486 | 0.509 |
| Art | 0.595 | **0.601** | 0.543 | 0.500 | 0.504 | 0.497 | 0.523 | 0.455 | 0.470 |
| Black White | 0.466 | **0.534** | 0.303 | 0.529 | 0.529 | 0.529 | 0.459 | 0.427 | 0.424 |
| Cartoon | 0.554 | **0.560** | 0.513 | 0.467 | 0.479 | 0.478 | 0.491 | 0.452 | 0.463 |
| Fractal | **0.645** | 0.596 | 0.553 | 0.524 | 0.522 | 0.516 | 0.485 | 0.456 | 0.487 |
| Indoor | **0.601** | 0.586 | 0.549 | 0.498 | 0.507 | 0.499 | 0.507 | 0.463 | 0.462 |
| Inverted | 0.593 | **0.596** | 0.545 | 0.507 | 0.512 | 0.502 | 0.490 | 0.449 | 0.437 |
| Jumbled | **0.609** | 0.580 | 0.547 | 0.463 | 0.464 | 0.465 | 0.511 | 0.477 | 0.456 |
| Line Drawing | **0.660** | 0.323 | 0.167 | 0.327 | 0.327 | 0.327 | 0.282 | 0.202 | 0.422 |
| Low Resolution | **0.543** | 0.530 | 0.444 | 0.396 | 0.388 | 0.376 | 0.324 | 0.345 | 0.345 |
| Noisy | 0.541 | **0.617** | 0.610 | 0.475 | 0.474 | 0.471 | 0.458 | 0.358 | 0.360 |
| Object | 0.599 | **0.627** | 0.538 | 0.552 | 0.555 | 0.539 | 0.548 | 0.514 | 0.554 |
| Outdoor Man-Made | 0.569 | **0.620** | 0.584 | 0.511 | 0.508 | 0.505 | 0.471 | 0.405 | 0.413 |
| Outdoor Natural | 0.590 | **0.627** | 0.588 | 0.533 | 0.536 | 0.529 | 0.475 | 0.377 | 0.364 |
| Pattern | 0.590 | **0.616** | 0.477 | 0.544 | 0.547 | 0.546 | 0.362 | 0.335 | 0.412 |
| Random | 0.533 | **0.599** | 0.538 | 0.497 | 0.502 | 0.495 | 0.534 | 0.541 | 0.522 |
| Satellite | 0.530 | **0.650** | 0.642 | 0.486 | 0.480 | 0.478 | 0.456 | 0.289 | 0.274 |
| Sketch | 0.574 | 0.338 | 0.050 | 0.336 | 0.335 | 0.335 | 0.568 | 0.527 | **0.623** |
| Social | **0.520** | 0.508 | 0.471 | 0.422 | 0.418 | 0.417 | 0.478 | 0.433 | 0.415 |

Table 5.1: The Linear Correlation Coefficient (CC) for each category. Bold values denote the largest CC for a category.

| Algorithm | Base | PM B | PM $O^C$ | PM $O^R$ | PM $O^G$ | PM $O^B$ | GBVS | RARE'12 | BMS |
|---|---|---|---|---|---|---|---|---|---|
| Entire Dataset | **0.570** | 0.562 | 0.485 | 0.477 | 0.478 | 0.474 | 0.475 | 0.427 | 0.447 |
| Entire Dataset* | 0.565 | **0.588** | 0.527 | 0.493 | 0.494 | 0.489 | 0.481 | 0.434 | 0.439 |

Table 5.2: The Linear Correlation Coefficient (CC) for the entire dataset and modified dataset. Bold values denote the largest CC for a category.

Table 5.1 shows how the proposed method performs when using the linear correlation coefficient. The base method performs better than the best of the proposed method on the entire dataset. However, it only performs better by 0.008 using the CC metric. Looking deeper into the results we can see that the proposed method actually has the top CC score for 13 of the 20 categories and is second in another 5. The categories "Line Drawing" and "Sketch" are the worst performing. This comes as no surprise since these two categories contain images with few details and a very limited range of colors (Black and White). As mentioned in section 5.1, the proposed method performs terribly for these kinds of images. With this in mind, when the best saliency map of the proposed method is used, the proposed method still ranks in the top two CC scores for 18 out of 20 categories. If the results of the color operators are looked at separately, the results are still pretty good. Using just $O^C$ on the entire dataset provides an average CC that is higher than the ten other methods that the base method was tested against. The other three color operators on there own also perform on par with GBVS, the next highest scoring algorithm of the ten tested against in [4]. This is notable since the three color operators disregard available information. With this in mind, Table 5.2 shows the performance of the methods using the CC metric excluding the two aforementioned categories. As can be seen, the CC scores of the four other methods fluctuate within a hundredth of their Entire Dataset value. The scores for the proposed method all significantly increase by greater than a hundredth. When looking at highly detailed, color-rich images the PM B score is better than the base methods score. This implies that on detailed color images the proposed method generates saliency maps that better estimate fixation maps. The single operator per-

formance scores are also better on this subset of the CAT2000 dataset. They hands down beat everything but the base method.

| Algorithm | Base | PM B | PM $O^C$ | PM $O^R$ | PM $O^G$ | PM $O^B$ | GBVS | RARE'12 | BMS |
|---|---|---|---|---|---|---|---|---|---|
| Entire Dataset | **0.532** | 0.493 | 0.442 | 0.424 | 0.426 | 0.422 | 0.491 | 0.475 | 0.497 |
| Entire Dataset* | **0.535** | 0.524 | 0.483 | 0.448 | 0.451 | 0.446 | 0.500 | 0.482 | 0.493 |

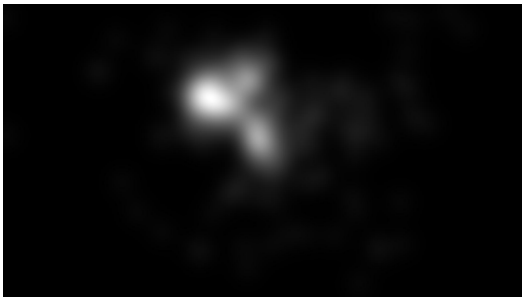
Table 5.3: The Similarity (SIM) for the entire dataset and modified dataset. The bold value denotes the largest SIM.

Table 5.3 shows the performance of the proposed method against other algorithms using the similarity metric. The base method still outperforms the proposed method and the ten algorithms mentioned earlier. However, the proposed method does not fair poorly. When using the highest scoring saliency map of the proposed method, the SIM score is the third highest score with only a 0.004 difference between it and BMS, the algorithm with the second highest SIM score. Disregarding the "Line Drawing" and "Sketch" categories again leads to an increase in the scores of the proposed method. As with the CC metric, the SIM scores of the other methods fluctuate by less than a hundredth while the proposed method has only increases greater than a hundredth. From this we can see that the proposed method performance lands close to that of the base method. This indicates that the proposed method is on par with the base method using the SIM metric, again looking at detailed color images.

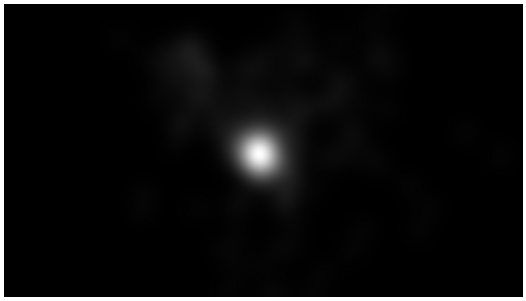
Figure 5.4 shows two images, there fixation maps, and one of their saliency maps. It also shows the associated SIM and CC scores. The saliency maps corresponding to these images have the highest SIM score in their respective category. Similarly, Figure 5.5 also shows the original images, fixation maps, and one of the saliency maps generated for each image. In this case, the saliency maps corresponding to these images have the highest CC score in their respective category.

(a) Object 025

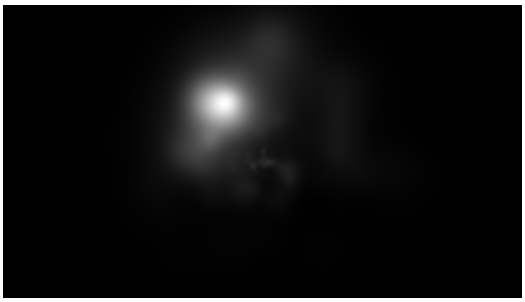
(b) Black White 099



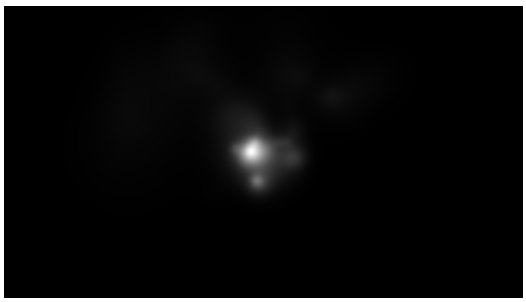
(c) Fixation Map



(d) Fixation Map



(e) SM ($O^C$), SIM = 0.706, CC = 0.869

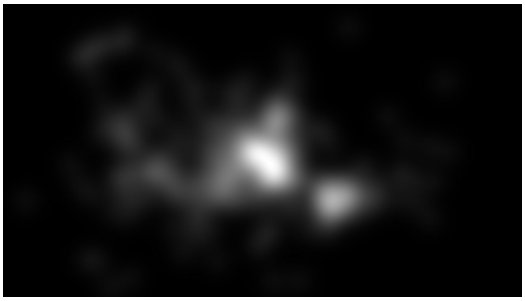


(f) SM ($O^G$), SIM = 0.740, CC = 0.927

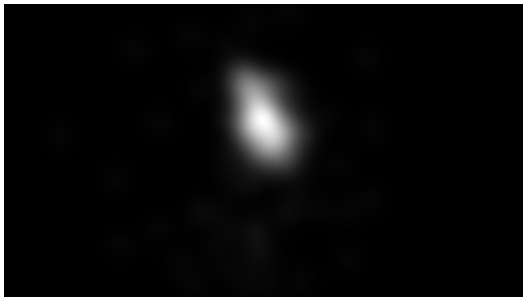
Figure 5.4: Images with their respective saliency maps that had the highest SIM score in their category. The top row contains the original image from the CAT2000 dataset. The middle row contains the fixation maps. The bottom row contains the highest scoring saliency map for the image along with its SIM and CC scores.

(a) Indoor 85

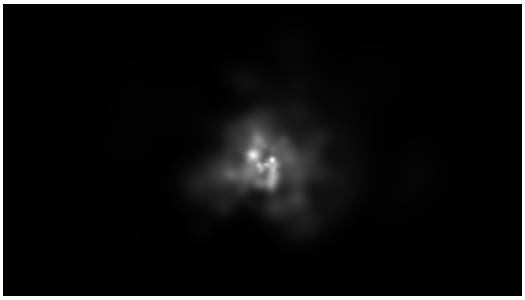
(b) Low Resolution 149



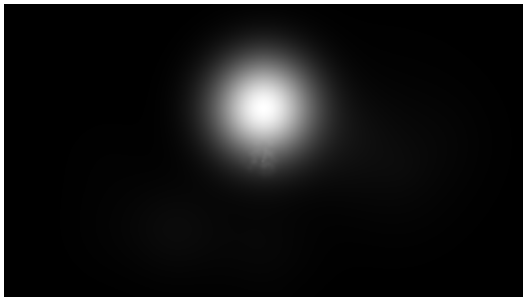
(c) Fixation Map



(d) Fixation Map



(e) SM ($O^C$), SIM = 0.653, CC = 0.847



(f) SM ($O^B$), SIM = 0.635, CC = 0.861

Figure 5.5: Images with their respective saliency maps that had the highest CC score in their category. The top row contains the original image from the CAT2000 dataset. The middle row contains the fixation maps. The bottom row contains the highest scoring saliency map for the image along with its SIM and CC scores.

# Chapter 6

# Conclusion

In this thesis a saliency map generation algorithm that highlights different color changes was developed. This was done by altering the algorithm presented in [4] so that the outputs of the Complementary Color Wavelet Transform could be used.

In order to adapt the method so that the CCWT outputs could be used, several main issues were resolved. These included defining what pixels surrounded a pixel of interest in space and scale, finding how the complementary color operator values at different scales could be compared, and finding a substitute for the horizontal and vertical gradient maps used to calculate directional rarity. These compatibility issues were resolved leading to a method that can generate four different saliency maps for an image.

This algorithm was tested in two ways. The first tests were qualitative. These tests showed that this algorithm indeed highlights different aspects of an image depending on which complementary color operator was used to generate the saliency map. During these tests it was also found that the algorithm performs better for images that were color-rich with more details and poorly for low detail images without many colors. The second tests were quantitative. Two common metrics for evaluating saliency maps were used to evaluate the saliency maps generated for the CAT2000

dataset. The results from the similarity metric showed that the results were similar to other algorithms. The results from the linear correlation coefficient said that this method surpassed the other tested methods in thirteen of the twenty image categories of the CAT2000 dataset, having the second best performance in another five categories. These tests also confirmed that this method is ill suited for low detail images with a small amounts of different colors.

Several aspects of this algorithm can be investigated in the future. The first of these is seeing how different sets of CCWT wavelets affect the performance of this algorithm. Another avenue of potential research is to find other ways of replacing the $\Delta L^*$ scale space using the RGB color space for the computation of directional rarity. Lastly, this method can be evaluated using other performance metrics. In addition to these research paths, the usefulness of this method can be evaluated for image processing tasks. These include image segmentation, computer vision, and image compression.

# Bibliography


[1] Qingshan Li, Yue Zhou, and Jie Yang, "Saliency based image segmentation," in *2011 International Conference on Multimedia Technology*, pp. 5068–5071, 2011.

[2] A. Maity, "Improvised salient object detection and manipulation," *International Journal of Image, Graphics and Signal Processing*, vol. 8, no. 2, pp. 53–60, 2016.

[3] C. Guo and L. Zhang, "A novel multiresolution spatiotemporal saliency detection model and its applications in image and video compression," *IEEE Transactions on Image Processing*, vol. 19, no. 1, pp. 185–198, 2010.

[4] K. Ishikura, N. Kurita, D. M. Chandler, and G. Ohashi, "Saliency detection based on multiscale extrema of local perceptual color differences," *IEEE Transactions on Image Processing*, vol. 27, no. 2, pp. 703–717, 2018.

[5] Y. Chen, D. Li, and J. Q. Zhang, "Complementary color wavelet: A novel tool for the color image/video analysis and processing," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 29, no. 1, pp. 12–27, 2019.

[6] S. Mallat, *A Wavelet Tour of Signal Processing: The Sparse Way*. Burlington, MA: Elsevier Inc., 3rd ed., 2009.

[7] Y. Meyer, *Wavelets and Operators: Advanced Mathematics*. Cambridge University Press, 1992.

[8] M. Ciacci, "3d color spaces." `https://www.mathworks.com/matlabcentral/fileexchange/60939-3d-color-spaces`, 2016. Online; accessed 27-March-2020.

[9] A. Hanbury, "Constructing cylindrical coordinate colour spaces," *Pattern Recognition Letters*, vol. 29, no. 4, pp. 494–500, 2008.

[10] International Color Consortium, *Specification ICC.1:2004-10 (Profile version 4.2.0.0) Image technology colour management — Architecture, profile format, and data structure*, 2006.

[11] A. Borji and L. Itti, "Cat2000: A large scale fixation dataset for boosting saliency research," *CVPR 2015 workshop on "Future of Datasets"*, 2015. arXiv preprint arXiv:1505.03581.

[12] N. Bruce and J. Tsotsos, "Attention Based on information maximization," *Journal of Vision*, vol. 7, pp. 950–950, 06 2007.

[13] N. Murray, M. Vanrell, X. Otazu, and C. A. Parraga, "Saliency estimation using a non-parametric low-level vision model," in *CVPR 2011*, pp. 433–440, 2011.

[14] J. Harel, C. Koch, and P. Perona, "Graph-based visual saliency," *Advances in Neural Information Processing Systems*, vol. 19, pp. 545–552, 2006.

[15] H. R. Tavakoli, E. Rahtu, and J. Heikkilä, "Fast and efficient saliency detection using sparse sampling and kernel density estimation," in *SCIA 2011*, pp. 666–675, 2011.

[16] S. Goferman, L. Zelnik-Manor, and A. Tal, "Context-aware saliency detection," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2376–2383, 2010.

[17] M. Agrawal, K. Konolige, and M. R. Blas, "CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching," in *European Conference on Computer Vision*, pp. 102–115, 2008.

[18] B. W. Tatler, "The central fixation bias in scene viewing: Selecting an optimal viewing position independently of motor biases and image feature distributions," *Journal of Vision*, vol. 7, 11 2007.

[19] Z. Bylinskii, T. Judd, A. Oliva, A. Torralba, and F. Durand, "What do different evaluation metrics tell us about saliency models?," *arXiv preprint arXiv:1604.03605*, 2016.

[20] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 11, pp. 1254–1259, 1998.

[21] X. Hou and L. Zhang, "Saliency detection: A spectral residual approach," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, 2007.

[22] J. Li, M. D. Levine, X. An, X. Xu, and H. He, "Visual saliency based on scale-space analysis in the frequency domain," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 996–1010, 2013.

[23] N. Riche, M. Mancas, M. Duvinage, B. Gosselin, and T. Dutoit, "RARE2012: A multi-scale rarity-based saliency detection with its comparative statistical analysis," *Signal Processing: Image Communication*, vol. 6, no. 4, pp. 642–658, 2013.

[24] L. Zhang, M. H. Tong, T. K. Marks, H. Shan, and G. W. Cottrell, "Sun: A Bayesian framework for saliency using natural statistics," *Journal of Vision*, vol. 8, no. 7, pp. 1–20, 2008.

[25] S. Lu, C. Tan, and J. Lim, "Robust and efficient saliency modeling from image co-occurrence histograms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 1, pp. 195–201, 2014.

[26] J. Zhang and S. Sclaroff, "Saliency detection: A boolean map approach," in *2013 IEEE International Conference on Computer Vision*, pp. 153–160, 2013.

[27] J. Zhang and S. Sclaroff, "Exploiting surroundedness for saliency detection: A boolean map approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 5, pp. 889–902, 2016.

# Appendix A

# Code Appendix

This appendix includes all the code necessary to generate the saliency maps with the proposed method. The example code is shown first. This code shows how to use the custom functions to generate saliency maps. In addition it also shows how to compute the SIM and CC scores. All custom functions used by this example appear in alphabetical order following the example code.

It should be noted that the code for the functions 'filtCoeffs' and 'filtCoeffs2Undec' was written by Chen *et al.* [5]. The implementations of the functions 'ccwtFiltUndec,' 'ccwtOper,' 'filt2Ddown,' and 'filt2DdownUndec' was based on the code written by Chen *et al.*.

## A.1 Example Code

```
1 %Karol Wadolowski - Master's Thesis Code: Example
2
3 %This code generates the saliency maps for the input image. If the fixation
4 %map for the image is also supplied the SIM and CC scores are calculated.
5
6 clear; clc; close all;
7
8 %Load the image
9 img = imread('141.jpg');
```

```
10
11 %Generate the saliency maps for the image
12 [salC,salR,salG,salB] = ccwtSMgen(img);
13
14 %Display the saliency maps along with the original image
15 figure('units','normalized','outerposition',[0,0,1,1])
16 subplot(2,3,1)
17 imshow(salC,'InitialMagnification','fit')
18 title('SM using O^C');
19
20 subplot(2,3,4)
21 imshow(salR,'InitialMagnification','fit')
22 title('SM using O^R');
23
24 subplot(2,3,2)
25 imshow(salG,'InitialMagnification','fit')
26 title('SM using O^G');
27
28 subplot(2,3,5)
29 imshow(salB,'InitialMagnification','fit')
30 title('SM using O^B');
31
32 subplot(2,3,3)
33 imshow(mat2gray(img))
34 title('Original Image')
35
36 sgtitle('Saliency Maps Generated Using the Proposed Method')
37
38 %If the fixation map is available calulate the SIM and CC
39 fmImg = imread('141FM.jpg');
40 fmImg = im2double(fmImg);
41
42 %Calculate SIM score
43 simC = SIM(fmImg,salC)                    %Chroma
44 simR = SIM(fmImg,salR)                    %Red-Cyan
45 simG = SIM(fmImg,salG)                    %Green-Magenta
46 simB = SIM(fmImg,salB)                    %Blue-Yellow
47
48 %Calculate CC score
49 ccC = corr(mat2gray(fmImg(:)),salC(:))    %Chroma
50 ccR = corr(mat2gray(fmImg(:)),salR(:))    %Red-Cyan
```

```
51 ccG = corr(mat2gray(fmImg(:)),salG(:))  %Green-Magenta
52 ccB = corr(mat2gray(fmImg(:)),salB(:))  %Blue-Yellow
```

## A.2 ccwtFiltUndec.m

```
1  function [w,u,uDir] = ccwtFiltUndec(img,aFilt,aFilt2,phase2,levels)
2  %{
3  Takes the input image and performs the analysis using regular and double
4  phased filters down to a certain resolution. The separation transform is
5  performed to return the wavelet coefficients. The first level is
6  undecimated.
7
8  Inputs:
9      img          : (R,C) Input image to be analyzed
10     aFilt        : (3, ) Cell array containing the 3 analysis lpfs and hpfs
11     aFilt2       : (3, ) Cell array containing the double phase filters
12     phase2       : (1,1) How many of the levels should be done using the
13                    double phase filters
14     levels       : (1,1) How many levels of decomposition to have
15                     Levels - phase2 = the number fo regular phase levels
16                     Note in the SM algorithm we do not count the double
17                     phase layer but it is considered in this function as a
18                     layer. Therefore there is a +1 discrepancy.
19
20 Outputs:
21     w            : (levels, ) Cell array containing a 3x3 cell array of the
22                    image filetered with the 9 filter combinations
23                    ((0,2,4)pi/3 phase matched with (0,2,4)pi/3). Each 3x3
24                    contains a cell array of the LL, LH, HL, and HH images.
25                    (levels)(3,3)(4,1)
26                          |  0  |  2  |  4          |  L  |  H
27                      0   | 0,0 | 2,0 | 4,0       L | LL  | HL
28                      2   | 0,2 | 2,2 | 4,2       H | LH  | HH
29                      4   | 0,4 | 2,4 | 4,4
30                    2,0 means filter columns with 2pi/3 phase filters then
31                    rows with 0 phase filters. Both have an lpf and hpf
32                    leading to the 4 combinations LL, LH, HL, and HH
33
34     u            : (levels+1, ) Cell array containing the coefficients
35                    (after the separation trandform is applied). Each level
36                    contains a (9,3) cell array. The columns correspond to
```

```
37                  LH, HL, and HH. The levels+1 cell array contains a (3,3)
38                  cell array of the LL components of the image. This level
39                  is for convenience and redundant since it is also
40                  contained in w.
41
42      uDir        : (levels+1, ) Cell array containing (8,3) cell arrays.
43                    Each row corresponds to coefficients in a certain
44                    direction in the order: (1:8)pi/8. The three columns are
45                    for the three different coefficients in that direction.
46                    The last level is the same as for u. The double phase
47                    levels are left the same as u.
48
49  Note: LH means filter columns with the lpf and then the rows with the hpf.
50  %}
51
52  %Check that the image can be decimated levels - 1 times.
53  [R,C] = size(img);
54  if((mod(R,2^(levels-1)) ~= 0) || (mod(C,2^(levels-1)) ~= 0))
55      error('Please input image that is divisble by 2^levels')
56  end
57
58  img = img/3;
59
60  %Initialize the filtered image cell array
61  w = cell(levels,1);
62  for ii = 1:levels
63      w{ii} = cell(3,3);
64  end
65
66  %Do all the filtering
67  for rr = 1:3
68      for cc = 1:3
69          %First level use the original image
70          if(phase2 >= 1)
71              w{1}{rr,cc} = filt2DdownUndec(img,aFilt2{rr},aFilt2{cc});
72          else
73              w{1}{rr,cc} = filt2DdownUndec(img,aFilt{rr},aFilt{cc});
74          end
75
76          %Filter the rest of the levels
77          for level = 2:levels
```

```
78              if(level <= phase2)
79                  %Double phase
80                  w{level}{rr,cc} = filt2Ddown(w{level-1}{rr,cc}{1},...
81                      aFilt2{rr},aFilt2{cc});
82              else
83                  %Regular phase
84                  w{level}{rr,cc} = filt2Ddown(w{level-1}{rr,cc}{1},...
85                      aFilt{rr},aFilt{cc});
86              end
87          end
88      end
89  end
90
91  %Perform the separation transform for all the regular phase levels
92  u = cell(levels+1,1);
93  for ii = 1:levels
94      u{ii} = cell(9,3);
95  end
96
97      %Don't do anything to double phase layers
98  for ii = 1:phase2
99      temp = reshape(w{ii},9,1);
100     for jj = 1:9
101         for kk = 1:3
102             u{ii}{jj,kk} = temp{jj}{kk+1};
103         end
104     end
105 end
106
107     %For the regular phase levels apply the separation transform
108 for ii = (phase2+1):levels
109     temp = reshape(w{ii},9,1);
110
111     for jj = 1:3
112         u{ii}{1,jj} = temp{1}{jj+1} + temp{5}{jj+1} + temp{9}{jj+1};
113         u{ii}{2,jj} = temp{2}{jj+1} + temp{6}{jj+1} + temp{7}{jj+1};
114         u{ii}{3,jj} = temp{1}{jj+1} + temp{6}{jj+1} + temp{8}{jj+1};
115         u{ii}{4,jj} = temp{2}{jj+1} + temp{4}{jj+1} + temp{9}{jj+1};
116         u{ii}{5,jj} = temp{1}{jj+1} + temp{2}{jj+1} + temp{3}{jj+1};
117         u{ii}{6,jj} = temp{4}{jj+1} + temp{5}{jj+1} + temp{6}{jj+1};
118         u{ii}{7,jj} = temp{1}{jj+1} + temp{4}{jj+1} + temp{7}{jj+1};
```

```
119         u{ii}{8,jj} = temp{2}{jj+1} + temp{5}{jj+1} + temp{8}{jj+1};
120
121         u{ii}{9,jj} = temp{1}{jj+1} + temp{2}{jj+1} + temp{3}{jj+1}...
122                       +temp{4}{jj+1} + temp{5}{jj+1} + temp{6}{jj+1}...
123                       +temp{7}{jj+1} + temp{8}{jj+1} + temp{9}{jj+1};
124     end
125 end
126
127 %Keep the LL component of last portion
128 for ii = 1:9
129     u{levels+1}{ii} = temp{ii}{1};
130 end
131
132
133 %Prepare the directional u.
134     %Leave the double phase levels alone.
135 uDir = cell(levels+1,1);
136 for ii = 1:phase2
137     uDir{ii} = u{ii};
138 end
139
140     %Regular phase
141 for ii = (phase2+1):levels
142     uDir{ii} = cell(8,3);
143
144     %Order the coefficients according to directions for ease of analysis.
145     %Note that the third part of each direction is in effect the
146     %difference from the other two since u{ii}{9,jj} should always
147     %theoretically be zero.
148
149     %1*pi/8 (HL)
150     uDir{ii}{1,1} = u{ii}{3,2};
151     uDir{ii}{1,2} = u{ii}{4,2};
152     uDir{ii}{1,3} = u{ii}{9,2} - u{ii}{3,2} - u{ii}{4,2};
153
154     %2*pi/8 (HH)
155     uDir{ii}{2,1} = u{ii}{3,3};
156     uDir{ii}{2,2} = u{ii}{4,3};
157     uDir{ii}{2,3} = u{ii}{9,3} - u{ii}{3,3} - u{ii}{4,3};
158
159     %3*pi/8 (LH)
```

```
160         uDir{ii}{3,1} = u{ii}{3,1};
161         uDir{ii}{3,2} = u{ii}{4,1};
162         uDir{ii}{3,3} = u{ii}{9,1} - u{ii}{3,1} - u{ii}{4,1};
163
164         %4*pi/8 (LH)
165         uDir{ii}{4,1} = u{ii}{7,1};
166         uDir{ii}{4,2} = u{ii}{8,1};
167         uDir{ii}{4,3} = u{ii}{9,1} - u{ii}{7,1} - u{ii}{8,1};
168
169         %5*pi/8 (LH)
170         uDir{ii}{5,1} = u{ii}{1,1};
171         uDir{ii}{5,2} = u{ii}{2,1};
172         uDir{ii}{5,3} = u{ii}{9,1} - u{ii}{1,1} - u{ii}{2,1};
173
174         %6*pi/8 (HH)
175         uDir{ii}{6,1} = u{ii}{1,3};
176         uDir{ii}{6,2} = u{ii}{2,3};
177         uDir{ii}{6,3} = u{ii}{9,3} - u{ii}{1,3} - u{ii}{2,3};
178
179         %7*pi/8 (HL)
180         uDir{ii}{7,1} = u{ii}{1,2};
181         uDir{ii}{7,2} = u{ii}{2,2};
182         uDir{ii}{7,3} = u{ii}{9,2} - u{ii}{1,2} - u{ii}{2,2};
183
184         %8*pi/8 (HL)
185         uDir{ii}{8,1} = u{ii}{5,2};
186         uDir{ii}{8,2} = u{ii}{6,2};
187         uDir{ii}{8,3} = u{ii}{9,2} - u{ii}{5,2} - u{ii}{6,2};
188 end
189
190         %Bottom Level
191 uDir{levels+1} = u{levels+1};
```

## A.3 ccwtOper.m

```
1 function [oI,oC,oR,oG,oB] = ccwtOper(rDir,gDir,bDir)
2 %{
3 Takes the dircetion ordered coefficients from the three channels and
4 returns the output of the intensity, chroma, red-cyan, green-magenta, and
5 blue-yellow color operators.
6
```

```
7  Inputs:
8      rDir          : Red channel
9      gDir          : Green channel
10     bDir          : Blue channel
11
12     These three inputs are all the same shape. They each contain the
13     ordered coefficients from their respective channels. The size of these
14     inputs is (levels+1)(8,3) where the (8,3) applies only to the regular
15     phase levels. The double phase levels are (9,3) and the levels+1 level
16     is (1,9).
17
18 Outputs:
19     oI            : Intensity operator output
20     oC            : Chroma (black-white) operator output
21     oR            : Red-Cyan operator output
22     oG            : Green-Magenta operator output
23     oB            : Blue-Yellow operator output
24
25     All of these operator outputs have the same shape. The double phase
26     layers are just copied for oR, oG, and oB but are left empty for oI and
27     oC. The regular phase levels are all (8,3) cell arrays corresponding to
28     direction and one of the three wavelets in that direction. These cell
29     arrays contain the output of the contemporary color transform. The
30     levels+1 level is copied for oR, oG, and oB and left blank for oI and
31     oC.
32 %}
33
34 %First determine the number of levels and how many are double phase.
35 [temp,~] = size(rDir);
36 levels = temp - 1;
37
38 phase2 = 0;
39 for ii = 1:levels
40     if(isequal([9,3],size(rDir{ii})))
41         phase2 = ii;
42     end
43 end
44
45 %Perform the contemporary color transform
46 oI = cell(levels+1,1);
47 oC = cell(levels+1,1);
```

```
48 oR = cell(levels+1,1);
49 oG = cell(levels+1,1);
50 oB = cell(levels+1,1);
51
52     %Don't mess with double phase levels
53 for ii = 1:phase2
54     oI{ii} = [];                %Empty
55     oC{ii} = [];                %Empty
56     oR{ii} = rDir{ii};          %Copy
57     oG{ii} = gDir{ii};          %Copy
58     oB{ii} = bDir{ii};          %Copy
59 end
60
61     %Peform the transform on the regular phase levels
62 for ii = (phase2+1):levels
63     for jj = 1:8
64         %Calculate the operator output for each of the 8 directions
65
66         %Note each of the operators is calculating three values for a given
67         %direction and level. This is because there are three wavelets in
68         %each direction for each of the three channels (RGB). We first
69         %choose the first wavelet to be R, the second G, and the third B.
70         %Then we choose the second to be R, third to be G, and first to be
71         %B. Lastly, third is red, first is green, and second is blue. The
72         %other three permutations (RBG, GRB, BGR) are not valid since they
73         %do not retain the phase difference between the colors.
74
75         %Intensity Operator
76         oI{ii}{jj,1} = abs(rDir{ii}{jj,1}) + abs(gDir{ii}{jj,2}) + abs(bDir{ii}{jj
               ,3});
77         oI{ii}{jj,2} = abs(rDir{ii}{jj,2}) + abs(gDir{ii}{jj,3}) + abs(bDir{ii}{jj
               ,1});
78         oI{ii}{jj,3} = abs(rDir{ii}{jj,3}) + abs(gDir{ii}{jj,1}) + abs(bDir{ii}{jj
               ,2});
79
80         %Chroma Operator
81         oC{ii}{jj,1} =  rDir{ii}{jj,1} + gDir{ii}{jj,2} + bDir{ii}{jj,3};
82         oC{ii}{jj,2} =  rDir{ii}{jj,2} + gDir{ii}{jj,3} + bDir{ii}{jj,1};
83         oC{ii}{jj,3} =  rDir{ii}{jj,3} + gDir{ii}{jj,1} + bDir{ii}{jj,2};
84
85         %Red-Cyan Operator
```

```
86          oR{ii}{jj,1} =  rDir{ii}{jj,1} - gDir{ii}{jj,2} - bDir{ii}{jj,3};
87          oR{ii}{jj,2} =  rDir{ii}{jj,2} - gDir{ii}{jj,3} - bDir{ii}{jj,1};
88          oR{ii}{jj,3} =  rDir{ii}{jj,3} - gDir{ii}{jj,1} - bDir{ii}{jj,2};
89
90          %Green-Magenta Operator
91          oG{ii}{jj,1} = -rDir{ii}{jj,1} + gDir{ii}{jj,2} - bDir{ii}{jj,3};
92          oG{ii}{jj,2} = -rDir{ii}{jj,2} + gDir{ii}{jj,3} - bDir{ii}{jj,1};
93          oG{ii}{jj,3} = -rDir{ii}{jj,3} + gDir{ii}{jj,1} - bDir{ii}{jj,2};
94
95          %Blue-Yellow Operator
96          oB{ii}{jj,1} = -rDir{ii}{jj,1} - gDir{ii}{jj,2} + bDir{ii}{jj,3};
97          oB{ii}{jj,2} = -rDir{ii}{jj,2} - gDir{ii}{jj,3} + bDir{ii}{jj,1};
98          oB{ii}{jj,3} = -rDir{ii}{jj,3} - gDir{ii}{jj,1} + bDir{ii}{jj,2};
99      end
100 end
101
102     %Don't mess with levels+1 level
103 oI{levels+1} = [];                %Empty
104 oC{levels+1} = [];                %Empty
105 oR{levels+1} = rDir{levels+1};    %Copy
106 oG{levels+1} = gDir{levels+1};    %Copy
107 oB{levels+1} = bDir{levels+1};    %Copy
```

## A.4 ccwtSMgen.m

```
1  function [salC,salR,salG,salB] = ccwtSMgen(img)
2  %{
3  This function takes an input RGB or grayscale image and returns the four
4  saliency maps generated using the CCWT outputs.
5
6  Inputs:
7      img      : (R,C,N) Input image of size (R,C) with N = 1 or 3 channels
8
9  Outputs:
10     salC     : (R,C) SM generated using oC
11     salR     : (R,C) SM generated using oR
12     salG     : (R,C) SM generated using oG
13     salB     : (R,C) SM generated using oB
14 %}
15
16 %Get the image dimensions
```

```
17 [R,C,N] = size(img);
18
19 %Convert grayscale to RGB
20 if(N == 1)
21     img = repmat(img,1,1,3);
22 end
23
24 %Calculate the number of regular phase levels that can be performed
25 Lmax = floor(log2(min([R,C])));
26 levels = Lmax - 3;
27
28 %Extend the image with zeros and separate the channels
29 exImg = extendSize(img,levels);
30 imR = double(exImg(:,:,1));          %Red   Channel
31 imG = double(exImg(:,:,2));          %Green Channel
32 imB = double(exImg(:,:,3));          %Blue  Channel
33
34 %Load the CCWT filters
35 [af,~]  = filtCoeffs();              %Regular Phase Filters
36 [af2,~] = filtCoeffs2Undec();        %Double  Phase Filters
37
38 %Perform the CCWT
39 [~,~,rDir] = ccwtFiltUndec(imR,af,af2,1,levels+1);  %Filter Red   Channel
40 [~,~,gDir] = ccwtFiltUndec(imG,af,af2,1,levels+1);  %Filter Green Channel
41 [~,~,bDir] = ccwtFiltUndec(imB,af,af2,1,levels+1);  %Filter Blue  Channel
42 [~,oC,oR,oG,oB] = ccwtOper(rDir,gDir,bDir);
43
44 %Find extreme values
45 evC = EVdetection(oC,af,false);      %EVs for Chroma
46 evR = EVdetection(oR,af,false);      %EVs for Red-Cyan
47 evG = EVdetection(oG,af,false);      %EVs for Green-Magenta
48 evB = EVdetection(oB,af,false);      %EVs for Blue-Yellow
49
50 %Generate Saliency Maps
51 salC = ev2SM(img,exImg,evC,af);      %SM using Chroma
52 salR = ev2SM(img,exImg,evR,af);      %SM using Red-Cyan
53 salG = ev2SM(img,exImg,evG,af);      %SM using Green-Magenta
54 salB = ev2SM(img,exImg,evB,af);      %SM using Blue-Yellow
```

## A.5 ccwtSumOper.m

```
function tot = ccwtSumOper(oper,opt)
%{
Takes one of the outputs of the ccwtOper function and combines all
directions for all levels. This only applies to the regular phase levels.

Inputs:
    oper        : One of the outputs of ccwtOper
    opt         : Choose between sqSum, sum, or abs

Outputs:
    tot         : Cell array with the same form as oper except the
                  regular phase levels are now just a single image.
%}

%Find the number of levels and double phase levels
[levels,~] = size(oper);
levels = levels - 1;

phase2 = 0;
for ii = 1:levels
    if(isequal([0,0],size(oper{ii})) || isequal([9,3],size(oper{ii})))
        phase2 = ii;
    else
        break
    end
end

tot = oper;     %Copy to preserve the sizes present

%Compute the regular phase totals. Store by overwriting
for ii = (phase2+1):levels
    [R,C] = size(oper{ii}{1,1});        %Image size
    temp = zeros(R,C);                  %Will contain the total

    %Loop through each direction and each wavelet assignment
    for jj = 1:8
        for kk = 1:3
            switch opt
                case{'sqSum'}
                    temp = temp + oper{ii}{jj,kk}.^2;
                case{'sum'}
```

```
42                      temp = temp + oper{ii}{jj,kk};
43                  case{'abs'}
44                      temp = temp + abs(oper{ii}{jj,kk});
45                  otherwise
46                      error('Choose valid totaling option!')
47              end
48          end
49      end
50
51      tot{ii} = temp;
52  end
```

## A.6 ev2SM.m

```
1   function SM = ev2SM(img,exImg,evs,aFilt)
2   %{
3   Takes the original and extended image and a set of extreme values and
4   generates a saliency map.
5
6   Inputs:
7       img     : (Ror,Cor,3) Original Image
8       exImg   : (Rext,Cext,3) Extended Image
9       evs     : Extremes values (output from EVdetection function)
10      aFilt   : The regular phase filters used in the CCWT
11
12  Outputs:
13      SM      : (R,C) Saliency map
14  %}
15
16  [Rext,Cext,~] = size(exImg);
17
18  %Get the set of all scales with extreme values
19  evScales = unique(evs(:,4));
20  %Note scale 2 means the ev was found in an image with dimensions half the
21  %original (due to undecimated first level of ccwt)
22
23
24  scaleMag  = cell(size(evScales));   %Will hold the magnitude values for each scale
25  scaleAng  = cell(size(evScales));   %Will hold the angle values for each scale
26  scaleImgs = cell(size(evScales));   %Will hold the decimated images
27  scalePTh  = cell(size(evScales));   %Will hold angle distribution of each scale (full
```

```
      img not ev)
scalePInd = cell(size(evScales));   %Will hold individual distribution on each scale
scaleRdir = cell(size(evScales));   %Will hold the angle rarity for each ev at each
    scale

%Theta bin edges for directional histograms
thetas = linspace(-90,90,31);       %Theta bins
thetas(end) = 91;                   %Set to 91 b/c of binning (avoids edge case)

%Get the interior area relative coordinates for the scales
[xg,yg] = meshgrid(-2:2,-1:1);
relCords = [xg(:),yg(:);yg(:),xg(:)];
relCords = unique(relCords,'rows');

%This loop calculates the directional rarity
for ii = 1:length(evScales)
    temp = evScales(ii);

    %Get the scaled images by decimating
    scaleImgs{ii} = nan(Rext/2^(temp-1),Cext/2^(temp-1),3);
    for jj = 1:3
        scaleImgs{ii}(:,:,jj) = downsample(downsample(...
            exImg(:,:,jj),2^(temp-1))',2^(temp-1))';
    end

    %Find the magnitude and angle values
    tImg = scaleImgs{ii};
    [R,C,~] = size(tImg);

    th = tImg(2:(R-1),1:(C-2),:) - tImg(2:(R-1),3:C,:);
    tv = tImg(1:(R-2),2:(C-1),:) - tImg(3:R,2:(C-1),:);

    fhorz = zeros(R,C);
    fvert = zeros(R,C);

    %Calculate the horizontal and vertical gradients
    fhorz(2:(R-1),2:(C-1)) = (sum(th.^1,3)).^1;
    fvert(2:(R-1),2:(C-1)) = (sum(tv.^1,3)).^1;

    %Calculate the magnitude and direction
    scaleMag{ii} = (fhorz.^2 + fvert.^2).^0.5;
```

```
67      scaleAng{ii} = zeros(R,C);
68      scaleAng{ii}(2:(R-1),2:(C-1)) = atand(fhorz(2:(R-1),(2:C-1))./...
69          fvert(2:(R-1),(2:C-1)));
70      scaleAng{ii}(isnan(scaleAng{ii})) = 0;
71
72      %Next step is to create angle probabilies (histogram) at each scale and
73      %for each ev at each scale.
74      scalePTh{ii} = nan(1,length(thetas)-1); %Bins are n <= theta < n+1, n integer
75
76      %Find the histogram for the entire scale.
77      for jj = 1:(length(thetas)-1)
78          %Find indices corresponding to angles within a certain bin
79          inRanInds = (scaleAng{ii} >= thetas(jj)) & (scaleAng{ii} < thetas(jj+1));
80
81          mag = scaleMag{ii}(inRanInds);
82          scalePTh{ii}(jj) = sum(mag(:));
83      end
84
85      %Get the ev info from this scale only
86      sEvs = evs(evs(:,4) == evScales(ii),:);
87      [pts, ~] = size(sEvs);
88      scalePInd{ii} = cell(pts,2);    %First column stores within region, second
            outside
89      scaleRdir{ii} = nan(pts,1);
90
91          %Get the distributions within the circle of an ev
92      for jj = 1:pts
93          scalePInd{ii}{jj,1} = zeros(1,length(thetas)-1);
94          scalePInd{ii}{jj,2} = zeros(1,length(thetas)-1);
95
96          %Get in scale coordinates of maximum
97          row = sEvs(jj,5);
98          col = sEvs(jj,6);
99
100         %Handle any edge case evs
101         if(row <= 3)
102             currCords = relCords((row+relCords(:,1)) > 0,:);
103             if(col <= 3)
104                 currCords = currCords((col+currCords(:,2))>0,:);
105             elseif(col+2 > C)
106                 currCords = currCords((col+currCords(:,2))<(C+1),:);
```

```
107             end
108
109         elseif(row+2 > R)
110             currCords = relCords((row+relCords(:,1))<(R+1),:);
111             if(col <= 3)
112                 currCords = currCords((col+currCords(:,2))>0,:);
113             elseif(col+2 > C)
114                 currCords = currCords((col+currCords(:,2))<(C+1),:);
115             end
116
117         elseif(col <= 3)
118             currCords = relCords((col+relCords(:,2))>0,:);
119
120         elseif(col+2 > C)
121             currCords = relCords((col+relCords(:,2))<(C+1),:);
122
123         else
124             currCords = relCords;
125         end
126
127         %These are the indices of the pixels in the circle corresponding to
128         %extremum i at scale i.
129         currCords(:,1) = currCords(:,1) + row;
130         currCords(:,2) = currCords(:,2) + col;
131
132         insideMag = nan(1,size(currCords,1));
133         insideAng = nan(1,size(currCords,1));
134         for kk = 1:size(currCords,1)
135             insideMag(kk) = scaleMag{ii}(currCords(kk,1),currCords(kk,2));
136             insideAng(kk) = scaleAng{ii}(currCords(kk,1),currCords(kk,2));
137         end
138
139             %Individual probs
140         %Calculate the histogram for the inside of the circular region
141         for kk = 1:(length(thetas)-1)
142             inRanInds = (insideAng >= thetas(kk)) &...
143                 (insideAng < thetas(kk+1));
144
145             mag = insideMag(inRanInds);
146             scalePInd{ii}{jj,1}(kk) = sum(mag(:));
147         end
```

```
148
149         %Calculate the histogram for the outside of the circular region and
150         %then normalize to PMF
151         scalePInd{ii}{jj,2} = scalePTh{ii} - scalePInd{ii}{jj,1};
152         scalePInd{ii}{jj,2} = scalePInd{ii}{jj,2}/sum(scalePTh{ii});
153
154         [~,ind] = max(scalePInd{ii}{jj,1});
155         scaleRdir{ii}(jj) = -log(scalePInd{ii}{jj,2}(ind));
156     end
157 end
158
159 Rdir = [];
160 for ii = 1:length(scaleRdir)
161     Rdir = [Rdir;scaleRdir{ii}];
162 end
163 Rdir(Rdir == Inf) = 0;
164 Rdir = Rdir/max(Rdir);
165
166 %Now get the color rarity
167 %First for the full image
168 cBin = 5;
169 colorBins = linspace(0,255,cBin+1);
170 colorBins(end) = 256;
171 colDistAll = nan(cBin,cBin,cBin);      %Color histogram of full image
172 Rclr = nan(size(evs,1),1);
173 rIm = exImg(:,:,1);
174 gIm = exImg(:,:,1);
175 bIm = exImg(:,:,1);
176
177 for rr = 1:cBin
178     inRedInds = (colorBins(rr) <= rIm) & (rIm < colorBins(rr+1));
179
180     for gg = 1:cBin
181         inGreenInds = (colorBins(gg) <= gIm) & (gIm < colorBins(gg+1));
182
183         for bb = 1:cBin
184             inBlueInds = (colorBins(bb) <= bIm) & (bIm < colorBins(bb+1));
185
186             overlap = inRedInds & inGreenInds & inBlueInds;
187             colDistAll(rr,gg,bb) = sum(overlap(:));
188         end
```

```
189     end
190 end
191
192
193 colDistInd = cell(size(evs,1),2);
194 [ColG,RowG] = meshgrid(1:Cext,1:Rext);
195
196 for ii = 1:size(evs,1)
197     colDistInd{ii,1} = nan(cBin,cBin,cBin);
198     colDistInd{ii,2} = nan(cBin,cBin,cBin);
199
200     %Get all the pixel values in the circular region corresponding to each
201     %extremum in the full size image.
202     row = evs(ii,2);
203     col = evs(ii,1);
204     mask = ((ColG - col).^2 + (RowG - row).^2 - evs(ii,3).^2) < 0;
205
206     rMask = rIm(mask);      %Red values in circle
207     gMask = gIm(mask);      %Green values in circle
208     bMask = bIm(mask);      %Blue values in circle
209
210     %Create a color histogram for inside of the circle
211     for rr = 1:cBin
212         inRedInds = (colorBins(rr) <= rMask) & (rMask < colorBins(rr+1));
213
214         for gg = 1:cBin
215             inGreenInds = (colorBins(gg) <= gMask) &...
216                 (gMask < colorBins(gg+1));
217
218             for bb = 1:cBin
219                 inBlueInds = (colorBins(bb) <= bMask) &...
220                     (bMask < colorBins(bb+1));
221
222                 overlap = inRedInds & inGreenInds & inBlueInds;
223                 colDistInd{ii,1}(rr,gg,bb) = sum(overlap(:));
224             end
225         end
226     end
227
228     %Compute the histogram for the outside area and normalize
229     colDistInd{ii,2} = colDistAll - colDistInd{ii,1};
```

```
230       colDistInd{ii,2} = colDistInd{ii,2}/(Rext*Cext);
231
232       %Find max prob inside
233       [rMax,rIn] = max(colDistInd{ii,1},[],1);
234       rIn = squeeze(rIn);
235       [gMax,gIn] = max(squeeze(rMax),[],1);
236       [~,bIn] = max(gMax);
237
238       Rclr(ii) = -log(colDistInd{ii,2}(rIn(gIn(bIn),bIn),gIn(bIn),bIn));
239   end
240   Rclr(Rclr == Inf) = 0;
241   Rclr = Rclr/max(Rclr);
242
243
244   %Generate the saliency map
245   SM = zeros(Rext,Cext);
246   xmid = Cext/2;
247   ymid = Rext/2;
248   radFactor = 4;
249   weights = exp(-((xmid-evs(:,1)).^2 + (ymid-evs(:,2)).^2)./...
250           (radFactor^2*2*evs(:,3).^2))/(2*pi*evs(:,3).^2);
251   weights = weights/max(weights);
252
253
254   %Calculate the filter energy so that values at different scales can be
255   %compared.
256   h0 = aFilt{1}(:,1);       %0 phase LPF
257   h0w = fft(h0);            %FFT of the filter
258   Eh0 = sum(abs(h0w).^2); %Energy of the filter
259   const = sqrt(Eh0);        %Scaling factor between scales
260
261   for ii = 1:size(evs,1)
262           SM = SM + evs(ii,end)*weights(ii)*Rclr(ii)*Rdir(ii)*...
263                exp(-((ColG-evs(ii,1)).^2 + (RowG-evs(ii,2)).^2)/...
264                (2*evs(ii,3).^2))/(2*pi*evs(ii,3).^2)*...
265                const^(length(evScales)+1-evs(ii,4));
266   end
267
268   %Only take the portion corresponding to the original image size.
269   [Ror,Cor,~] = size(img);
270   SM = SM(1:Ror,1:Cor);
```

```
271
272 %Normalize to range [0,1]
273 SM = mat2gray(SM);
```

## A.7 EVdetection.m

```
1  function ev = EVdetection(oper,aFilt,plt)
2  %{
3  Takes an complementary color operator from the ccwtOper function and finds
4  the extreme values.
5
6  Inputs:
7      oper     : For information look at outputs of ccwtOper function
8      aFilt    : The regular phase filters used in the CCWT
9      plt      : (true or false) Option to plot the circles corresponding to
10               each extreme value.
11 %}
12
13
14 %Find the number of regular phase and double phase levels
15 [levels,~] = size(oper);
16 levels = levels - 1;
17
18 phase2 = 0;
19 for ii = 1:levels
20     if(isequal([0,0],size(oper{ii})) || isequal([9,3],size(oper{ii})))
21         phase2 = ii;
22     else
23         break
24     end
25 end
26
27
28 %Square and sum the 24 images for each regular phase scale.
29 tot = ccwtSumOper(oper,'sqSum');
30
31 evCand = cell(levels,1);        %Extreme value candidates
32 ev = cell(levels,1);            %Extreme values
33
34 %First find all the pixels that are greater than the 8 adjacent pixels at
35 %the same scale.
```

```
36 totEV = 0;                          %Running count of potential EV found
37 for ii = (phase2+1:levels)
38     [R,C] = size(tot{ii});
39     for jj = 2:(R-1)
40         for kk = 2:(C-1)
41             temp =  tot{ii}((jj-1):(jj+1),(kk-1):(kk+1));
42
43             %Find location of maximum within 3x3 area
44             [M,row] = max(temp,[],1);
45             [M,col] = max(M,[],2);
46
47             if((row(col) == 2) && (col == 2))
48                 %In scale maximum found
49                 %Store maximum coordinates and value
50                 evCand{ii} = vertcat(evCand{ii},[jj,kk,M]);
51                 totEV = totEV + 1;
52             end
53         end
54     end
55 end
56
57 %Calculate the filter energy so that values at different scales can be
58 %compared.
59 h0 = aFilt{1}(:,1);      %0 phase LPF
60 h0w = fft(h0);           %FFT of the filter
61 Eh0 = sum(abs(h0w).^2); %Energy of the filter
62 const = sqrt(Eh0);       %Scaling factor between scales
63
64 %Now find multiscale EVs from the list of candidates
65 if((phase2+1) == levels)
66     ev = evCand;
67 else
68     totEV = 0;
69     for ii = (phase2+1:levels)
70         for jj = 1:size(evCand{ii},1)
71             %In scale coordinate and value
72             row = evCand{ii}(jj,1);
73             col = evCand{ii}(jj,2);
74             val = evCand{ii}(jj,3);
75
76             %Get the indices of the surrounding pixels at the scale above
```

```
77              %and below.
78              [rA,rB] = getInd(row);      %Row Indices
79              [cA,cB] = getInd(col);      %Column Indices
80
81              %Compare against the pixels at different scales
82              if(ii == (phase2+1))
83                  %Only check layer below
84                  temp = max(max(tot{ii+1}(rB,cB)))/const;
85                  if(temp < val)
86                      %Max found
87                      ev{ii} = vertcat(ev{ii},evCand{ii}(jj,:));
88                      totEV = totEV + 1;
89                  end
90
91              elseif(ii == levels)
92                  %Only check layer above
93                  temp = max(max(tot{ii-1}(rA,cA)))*const;
94                  if(temp < val)
95                      %Max found
96                      ev{ii} = vertcat(ev{ii},evCand{ii}(jj,:));
97                      totEV = totEV + 1;
98                  end
99
100             else
101                 %Check above and below
102                 temp = max(max(tot{ii+1}(rB,cB)))/const;
103                 if(temp < val)
104                     temp = max(max(tot{ii-1}(rA,cA)))*const;
105
106                     if(temp < val)
107                         %Max found
108                         ev{ii} = vertcat(ev{ii},evCand{ii}(jj,:));
109                         totEV = totEV + 1;
110                     end
111                 end
112
113             end
114         end
115     end
116 end
117
```

```
%EV list stores the coordinates for the full scale image (xCen,yCen), the
%radius of the circle, the scale ii (level) where the EV was found, and the
%coordinates of the EV in its originals levels coordiantes. The last entry
%is the maximum value.
evList = nan(totEV,7);

if(plt)
    theta = linspace(0,2*pi,25);
    c = colormap(jet(levels-phase2));
end

%Obtain the information needed for evList and plot
radius = 2;     %Radius of the circle corresponding to an extremum
count = 0;
for ii = (phase2+1):levels
    %Radius increases by a factor of two for each regular phase level
    radius = radius*2;

    if(plt)
        %Circle centered at (0,0)
        x = radius*cos(theta);
        y = radius*sin(theta);
    end

    for jj = 1:size(ev{ii},1)
        count = count + 1;
        row = ev{ii}(jj,1);
        col = ev{ii}(jj,2);

        %Correspond the location at the scale to the full sized image
        xCen = col*2^(ii-1);
        yCen = row*2^(ii-1);

        if(plt)
            plot(x + xCen, y + yCen,'color',c(ii-phase2,:),'Linewidth',1)
        end

        %Store the important information
        evList(count,:) = [xCen,yCen,radius,ii,ev{ii}(jj,1:3)];
    end
end
```

```
159
160 ev = evList;
161 end
162
163 function [indA,indB] = getInd(t)
164 %Returns the indices of the pixels above and below.
165 %indA = indices for the scale above
166 %indB = indices for the scale below
167
168     indA = (2*t-2):(2*t+1);
169
170     if(mod(t,2) == 1)
171         %If odd
172         indB = (t-1)/2 + 0*(0:1);
173     else
174         %If even
175         indB = floor(t/2) + 0*(0:1);
176     end
177 end
```

## A.8 extendSize.m

```
1  function exImg = extendSize(img,levels)
2  %{
3  Takes a N channel image and extends the image such that it can be decimated
4  by 2 the specified amount of times.
5
6  Inputs:
7      img      : (R,C,N) Image
8      levels   : (1,11) Number of desired decimations
9
10 Outputs:
11     exImg    : (Rext,Cext,N) Extended image
12 %}
13
14 %Get the image dimensions
15 [R,C,N] = size(img);
16
17 %Find how much to extend the rows and columns
18 rAmt = extAmount(R,levels);      %Add this many rows
19 cAmt = extAmount(C,levels);      %Add this many columns
```

```
20
21 %Get the extended image
22 exImg = zeros(R+rAmt,C+cAmt,N);
23 for ii = 1:N
24     exImg(1:R,1:C,ii) = img(:,:,ii);
25 end
26 end
27
28 function amt = extAmount(origSize,lvls)
29 %{
30 Takes the original size and determines how much the size has to be extended
31 such that the extended version is divisible by 2^lvls
32 %}
33
34 q = floor(origSize/2^lvls);      %Quotient
35 r = mod(origSize,2^lvls);        %Remainder
36
37 if(q > 0)
38     if(r > 0)
39         amt = 2^lvls - r;
40     else
41         amt = 0;
42     end
43 else
44     amt = 2^lvls - r;
45 end
46 end
```

## A.9 filt2Ddown.m

```
1 function imgs = filt2Ddown(img,rowFilts,colFilts)
2 %{
3 Takes the input image and filters it with the column filters followed by
4 the row filters. All permutations are performed. Decimation by 2 is
5 performed. All filters are applied using a circular convolution.
6
7 Inputs:
8     img         : (R,C) Original image
9     rowFilts    : (L,M) M filters of length L
10    colFilts    : (P,N) N filters of length P
11
```

```
12 Outputs:
13     imgs         : (MN,1) Cell array of (R/2,C/2) images corresponding to
14                    the MN filtering combinations
15
16 Note: When the images are being generated the imgs cell array is (M,N) but
17 is later reshaped for convenience. After reshaping the first M rows were
18 filtered with the first of the N column filters.
19 %}
20
21 [R,C] = size(img);
22 [L,M] = size(rowFilts);
23 [P,N] = size(colFilts);
24
25 imgs = cell(M,N);
26
27 for rr = 1:M
28     mtxR = convmtx(rowFilts(:,rr),C);        %Convolution matrix for rows
29     for cc = 1:N
30         mtxC = convmtx(colFilts(:,cc),R);    %Convolution matrix for columns
31
32         %Circularly filter columns
33         temp = mtxC*img;                                          %Convolution
34         temp(1:P-1,:) = temp(1:P-1,:) + temp(R+1:end,:);          %Make it circular
35         temp = temp(1:R,:);                                       %Remove extra bits
36         temp = temp(mod((0:R-1)+P/2,R)+1,:);                      %Recenter image
37         temp = downsample(temp,2).';                              %Downsample and transpose
               for row portion
38
39         %Circularly filter the rows
40         temp = mtxR*temp;                                         %Convolution
41         temp(1:L-1,:) = temp(1:L-1,:) + temp(C+1:end,:);          %Make it circular
42         temp = temp(1:C,:);                                       %Remove extra bits
43         temp = temp(mod((0:C-1)+L/2,C)+1,:);                      %Recenter image
44         temp = downsample(temp,2);                                %Downsample
45
46         %Store filtered image
47         imgs{rr,cc} = temp.';
48     end
49 end
50
51 imgs = reshape(imgs,M*N,1);
```

## A.10 filt2DdownUndec.m

```
function imgs = filt2DdownUndec(img,rowFilts,colFilts)
%{
Takes the input image and filters it with the column filters followed by
the row filters. All permutations are performed. Decimation by 2 is not
performed for the undecimated version. All filters are applied using a
circular convolution.

Inputs:
    img         : (R,C) Original image
    rowFilts    : (L,M) M filters of length L
    colFilts    : (P,N) N filters of length P

Outputs:
    imgs        : (MN,1) Cell array of (R,C) images corresponding to
                  the MN filtering combinations

Note: When the images are being generated the imgs cell array is (M,N) but
is later reshaped for convenience. After reshaping the first M rows were
filtered with the first of the N column filters.
%}

[R,C] = size(img);
[L,M] = size(rowFilts);
[P,N] = size(colFilts);

imgs = cell(M,N);

for rr = 1:M
    mtxR = convmtx(rowFilts(:,rr),C);       %Convolution matrix for rows
    for cc = 1:N
        mtxC = convmtx(colFilts(:,cc),R);   %Convolution matrix for columns

        %Circularly filter columns
        temp = mtxC*img;                                    %Convolution
        temp(1:P-1,:) = temp(1:P-1,:) + temp(R+1:end,:);    %Make it circular
        temp = temp(1:R,:);                                 %Remove extra bits
        temp = temp(mod((0:R-1)+P/2,R)+1,:);                %Recenter image

        %Circularly filter the rows
```

```
40          temp = mtxR*temp';                                  %Convolution
41          temp(1:L-1,:) = temp(1:L-1,:) + temp(C+1:end,:);    %Make it circular
42          temp = temp(1:C,:);                                 %Remove extra bits
43          temp = temp(mod((0:C-1)+L/2,C)+1,:);                %Recenter image
44
45
46          %Store filtered image
47          imgs{rr,cc} = temp.';
48      end
49  end
50
51  imgs = reshape(imgs,M*N,1);
```

## A.11 filtCoeffs.m

```
1   function [af,sf] = filtCoeffs()
2   %{
3   Returns the filter coefficients for the regular phase analysis filters and
4   synthesis filters.
5
6   Note this code is taken from the reseachers that developed the CCWT.
7   This code is the same as the original author's code (from github) in file:
8   CCWTFB0551.m
9
10  Outputs:
11      af      : (3,1) The three sets of analysis low pass and high pass
12                filters. Each entry is a cell array with entry [22,2] where
13                column 1 is the low pass and column 2 is the high pass
14                filter.
15      sf      : (3,1) The three sets of synthesis low pass and high pass
16                filters.
17  %}
18
19  af = cell(3,1);     %Analysis filters
20  sf = cell(3,1);     %Synthesis filters
21
22  %First analysis and synthesis filters
23  af{1} = sqrt(2)*[
24      4.39947785905780e-12
25      1.55196025214914e-07
26      -3.64244986151481e-09
```

```
27      -4.72296431657605e-05
28      4.51978768236905e-07
29      0.00367508218681040
30      0.00013478525990674 0
31      -0.0554563814967393
32      -0.0199281412110457
33      0.354811673474709
34      0.567845846237940
35      0.207254862089805
36      -0.0681221946106980
37      -0.0103436660847803
38      0.0216450723247797
39      0.00010464686757270 5
40      -0.00160839578452004
41      6.30232589463908e-07
42      3.23832905322482e-05
43      -7.33216485253484e-10
44      -3.17578983997701e-08
45      1.04185641833908e-13
46  ];
47  af{1}=[af{1} af{1}(end:-1:1)];
48  af{1}(2:2:end,2)=af{1}(2:2:end,2)*-1;
49  sf{1} = af{1}(end:-1:1, :);
50
51  %Second analysis and synthesis filters
52  af{2} =  sqrt(2)*[
53      1.26569593791355e-10
54      5.39451424543818e-09
55      3.88278416451764e-07
56      -3.66563902314513e-06
57      -9.96351883061440e-05
58      0.00056052764052971 5
59      0.00536332360237928
60      -0.0119902186579145
61      -0.0720880690497491
62      0.0729237485649041
63      0.485957241064240
64      0.493303761245982
65      0.0695670541341079
66      -0.0677827108006299
67      0.0134011287709973
```

```
68      0.0137872093638171
69      -0.00217674002769087
70      -0.000810170920791877
71      7.52908343490528e-05
72      1.12950083393471e-05
73      -2.10455598848953e-07
74      -9.11001341220481e-09
75  ];
76  af{2}=[af{2} af{2}(end:-1:1)];
77  af{2}(2:2:end,2)=af{2}(2:2:end,2)*-1;
78  sf{2} = af{2}(end:-1:1, :);
79
80  %Third analysis and synthesis filters
81  af{3} =  sqrt(2)*[
82      -4.67451949247478e-10
83      2.07975316973642e-11
84      3.28299284108471e-08
85      7.37902871004124e-07
86      -1.79120279796055e-05
87      -0.000119733002797833
88      0.00175340053377619
89      0.00523039398995174
90      -0.0322379791494069
91      -0.0650808649375147
92      0.205135892050975
93      0.564880938720559
94      0.361349856021001
95      -0.0256046003924052
96      -0.0415160457380604
97      0.0230445084865565
98      0.00577295512905657
99      -0.00241051069658350
100     -0.000243273289409661
101     5.90366018513381e-05
102     2.84619728506791e-06
103     -1.34603572366523e-07
104 ];
105 af{3}=[af{3} af{3}(end:-1:1)];
106 af{3}(2:2:end,2)=af{3}(2:2:end,2)*-1;
107 sf{3} = af{3}(end:-1:1, :);
```

## A.12 filtCoeffs2Undec.m

```
function [af,sf] = filtCoeffs2Undec()
%{
Returns the filter coefficients for the unmdecimated analysis and
synthesis filters for the double phase filters.

Note this code is taken from the reseachers that developed the CCWT.
This code is the same as the original author's code (from github) in file:
FsCCWTFB0551ud.m

Outputs:
    af      : (3,1) The three sets of analysis low pass and high pass
              filters. Each entry is a cell array with entry [22,2] where
              column 1 is the low pass and column 2 is the high pass
              filter.
    sf      : (3,1) The three sets of synthesis low pass and high pass
              filters.
%}

af = cell(3,1);     %Analysis filters
sf = cell(3,1);     %Synthesis filters

%First analysis and synthesis filters
af{1} = sqrt(2)*[
    1.04185641833908e-13
    4.39947785905780e-12
    1.55196025214914e-07
    -3.64244986151481e-09
    -4.72296431657605e-05
    4.51978768236905e-07
    0.00367508218681040
    0.00013478525990674 0
    -0.0554563814967393
    -0.0199281412110457
    0.354811673474709
    0.567845846237940
    0.207254862089805
    -0.0681221946106980
    -0.0103436660847803
    0.0216450723247797
```

```
40      0.000104646867572705
41      -0.00160839578452004
42      6.30232589463908e-07
43      3.23832905322482e-05
44      -7.33216485253484e-10
45      -3.17578983997701e-08
46  ];
47  af{1}=[af{1} af{1}(end:-1:1)];
48  af{1}(2:2:end,2)=af{1}(2:2:end,2)*-1;
49  sf{1} = af{1}(end:-1:1, :);
50
51  %Second analysis and synthesis filters
52  af{2} =  sqrt(2)*[
53      -9.11001341220481e-09
54      1.26569593791355e-10
55      5.39451424543818e-09
56      3.88278416451764e-07
57      -3.66563902314513e-06
58      -9.96351883061440e-05
59      0.000560527640529715
60      0.00536332360237928
61      -0.0119902186579145
62      -0.0720880690497491
63      0.0729237485649041
64      0.485957241064240
65      0.493303761245982
66      0.0695670541341079
67      -0.0677827108006299
68      0.0134011287709973
69      0.0137872093638171
70      -0.00217674002769087
71      -0.000810170920791877
72      7.52908343490528e-05
73      1.12950083393471e-05
74      -2.10455598848953e-07
75  ];
76  af{2}=[af{2} af{2}(end:-1:1)];
77  af{2}(2:2:end,2)=af{2}(2:2:end,2)*-1;
78  sf{2} = af{2}(end:-1:1, :);
79
80  %Third analysis and synthesis filters
```

```
af{3} =  sqrt(2)*[
    -1.34603572366523e-07
    -4.67451949247478e-10
    2.07975316973642e-11
    3.28299284108471e-08
    7.37902871004124e-07
    -1.79120279796055e-05
    -0.000119733002797833
    0.00175340053377619
    0.00523039398995174
    -0.0322379791494069
    -0.0650808649375147
    0.205135892050975
    0.564880938720559
    0.361349856021001
    -0.0256046003924052
    -0.0415160457380604
    0.0230445084865565
    0.00577295512905657
    -0.00241051069658350
    -0.000243273289409661
    5.90366018513381e-05
    2.84619728506791e-06
];
af{3}=[af{3} af{3}(end:-1:1)];
af{3}(2:2:end,2)=af{3}(2:2:end,2)*-1;
sf{3} = af{3}(end:-1:1, :);
```

## A.13 SIM.m

```
function score = SIM(im1,im2)
%{
Returns the similarity metric value between two saliency maps.
Defined in section 4.F of Saliency paper.

Inputs:
    im1, im2    : (R,C) Two one-channel images

Outputs:
    score       : (1, ) Similarity value of the two images
%}
```

```
12
13 if(~isequal(size(im1),size(im2)))
14     error('Input same size images')
15 end
16
17 both = nan([size(im1),2]);
18
19 both(:,:,1) = prep(im1);
20 both(:,:,2) = prep(im2);
21
22 mins = min(both,[],3);
23 score = sum(mins,'all');
24 end
25
26 function p = prep(t)
27 %Normalize t so that all values are positive and sum to 1
28 p = t - min(t,[],'all');
29 p = p/sum(p,'all');
30 end
```