Video Deinterlacing Using Control Grid Interpolation Frameworks

by

Ragav Venkatesan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved, June 2012 by the
Graduate Supervisory Committee:

David Frakes, Chair
Baoxin Li
Martin Reisslein

ARIZONA STATE UNIVERSITY

August 2012

ABSTRACT

Video deinterlacing is a key technique in digital video processing, particularly with the widespread usage of LCD and plasma TVs. This thesis proposes a novel spatio-temporal, non-linear video deinterlacing technique that adaptively chooses between the results from one dimensional control grid interpolation (1DCGI), vertical temporal filter (VTF) and temporal line averaging (LA). The proposed method performs better than several popular benchmarking methods in terms of both visual quality and peak signal to noise ratio (PSNR). The algorithm performs better than existing approaches like edge-based line averaging (ELA) and spatio-temporal edge-based median filtering (STELA) on fine moving edges and semi-static regions of videos, which are recognized as particularly challenging deinterlacing cases. The proposed approach also performs better than the state-of-the-art content adaptive vertical temporal filtering (CAVTF) approach. Along with the main approach several spin-off approaches are also proposed each with its own characteristics.

ACKNOWLEDGEMENTS

There are so many people that I am thankful for that there will never be enough space for a full acknowledgement. I would like to start my acknowledgement by thanking my primary advisor Dr. David Frakes. He had really been a source of inspiration. One meeting with him is usually enough to quench anyone's intellectual thirst for quite a while. No wonder he gets those best faculty mentor awards every so often. Dr. Frakes, transformed a self-doubting graduate student into a full-fledged and confident graduate research student.

I would like to thank Christine Zwart next. She had been an excellent mentor and I probably owe half of this thesis to her, for her excellent ideas. Not only did I learn how to code efficiently, I also learnt how to approach research in general, thanks to her. Put together, those conversations that I had with her can be a course on image processing by itself. There is no surprise to me that she is usually considered to be the best image processing teaching assistant at Arizona State University. I am proud and will always cherish the fact that I worked under such immensely talented and spirited people like Dr. Frakes and Christine.

This list of talented advisors would not be complete without Dr. Baoxin Li. This entire thesis progressed to this level thanks to his course on digital video processing. It was under his suggestion that I started working on a project involving saliency, which is one of the important ideas in this thesis. I am also grateful to him, as he also started funding my masters education even though I

still haven't joined him as a PhD student yet. He has not just had an impact on me as a teacher, but also as man who sticks to his values and understanding. His humble nature always makes me keep my feet in the ground and lets me know that there is always an ocean to learn. I would also like to thank Dr. Martin Reisslein for agreeing to be part of my thesis committee. I would also like to thank Esther Korner for helping out so much with my entire degree program.

I would like to thank my friends without whom I would not have been able to survive the hardships of a degree program that is so taxing. I would like to thank Tosha Shah, the hold for a social life in this strange place. She has been a good friend and the best person to discuss any technical jargon with. I would like to thank Parag Chandakkar and Charan Prakash in particular. All those intellectual discussions we had took us on a quest to assimilate knowledge on fields ranging from psychology to history; the thought of which still amazes me. There is no wonder that the three of us started our master's program together, our thesis together and are now defending our thesis also- together. I wish us all good luck to accomplish the goals that we set and climb even greater heights. We were never known to be the three of us, but as the four of us. I would like to now thank the fourth one of us, Jidnyasa Babar. She had helped in all my paper submissions, from correcting my technical mistakes to formatting my figures. I guess she should have been a co-author in all my papers. She was my project partner, when I was implementing the saliency project under Dr. Li and part of the reason why I used those ideas in this thesis. I would also like to thank Aditi Saraf, a chemical

engineer in midst of all of us who has perhaps been the subject for all of our visual tests.

I would like to thank the best of my friends, who helped me through all my troubles and had stuck with me through any problems that I faced; Aamir Sheriff and Archana Ramani. I would also like to thank Jai Krishna, the brother that I never had. He forced me into liking math, computers and music. He is and will always be the one person whom I look up to as a challenge. I also like to acknowledge Punitha Sampath for being there as a friend when I most needed her.

I would like to next thank Prof. Ravi Naganathan, my undergraduate thesis advisor. He is the reason why I am interested in research at all. Not only did he shape my personality, he also taught me the purpose of education. He taught me not just signal processing and communication, but also taught me the values of life and the importance of integrity. He is also one of my best friends and a second father. Any of my work would be nothing without an acknowledgement to him.

All the professors at ASU that I have had the privilege of studying under are nothing short of extraordinary. I would like to express my gratitude towards the fine professors who had taught me the skills of the trade. I would like to particularly acknowledge by name, Dr. Antonia Papandreou-Suppappola, Dr. Lina Karam, Dr. Jieping Ye and Dr. Andreas Spanias.

Lastly I want to thank the three people who have given me the most, my parents and my sister. My parents had been nothing short of exceptional

and I could not have asked for two better people to have brought me up and guided me in my life decisions. They had always been there and stuck with me during every problem that I faced. Without my sister and my parents' support, I could not have succeeded in any of my attempts.

Ragav Venkatesan

To him, whom though I try so hard to be,
I can never reach his glory,
To her, without whom this journey
wouldn't have even started
and to her, the little me.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

LIST OF TABLES

Table **Page**

Chapter 1: INTRODUCTION

As early as 1935, it was well known that the human visual system is more sensitive to large-area flicker than flickering detail [1]. Television broadcasters in the early years of media boom used this idea to their advantage by transmitting interlaced videos that reduced bandwidth usage. Interlaced videos are videos (that are rectangularly sampled) scanned in such a way that on any given frame of 'N' rows, only 'N/2' alternate rows are scanned. The remaining rows are scanned on the next frame. Such a video (as shown in Figure 1-1) due to persistence of human vision appears as if the video were progressive. Interlaced videos are generally preferred in video broadcast and transmission systems as they reduce the amount of data to be broadcast.

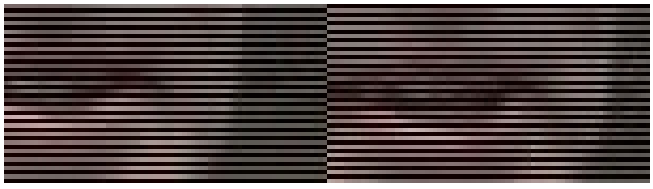


Figure 1-1: Interlaced video frame 1 followed by frame 2

Transmission of interlaced videos was widely popular in various television broadcasting systems such as NTSC [2], PAL [3], SECAM. Many broadcasting agencies made huge profits with interlaced videos. Video acquiring systems on many occasions naturally acquire interlaced video and since this also proved to be an efficient way, the popularity of interlaced videos escalated.

Interlaced video however, complicates many tasks pertaining to image processing. It has its advantages as well as disadvantages [4]. As scanning-format conversion was necessary for international TV broadcasting, the first proposals of deinterlacing were considered only for international programme exchange. With the advent of modern technology such as high-definition television (HDTV), video streaming, and DTH the need for a stable and a standard conversion between formats is increasing.

Modern display systems like LED and plasma displays work on progressive video. Progressive video unlike interlaced video contains the frames in their entirety in the videos' original resolution. The widespread usage of displays that require progressive videos has made deinterlacing an important process in the video processing arena.

With the advent of the PC era, Haan and Bellers raised the question of whether *to interlace or not to interlace* [5]. While the PC community believes that the present-day technologies are powerful enough to produce, transmit and display progressive video, the television community believes that it is advantageous to have interlaced videos in service. The main argument put forth by the PC community is that interlacing a video introduces a trade-off between vertical resolution and the time resolution. In the 1997 WinHec conference the co-founder of Microsoft Corporation, Bill Gates put forth a proposal to stabilize the picture rate of PCs to 60 Hertz progressive where it stands till date [6].

In [5], the authors studied alternate options to deinterlacing. Most importantly the question of whether present day technologies are powerful enough

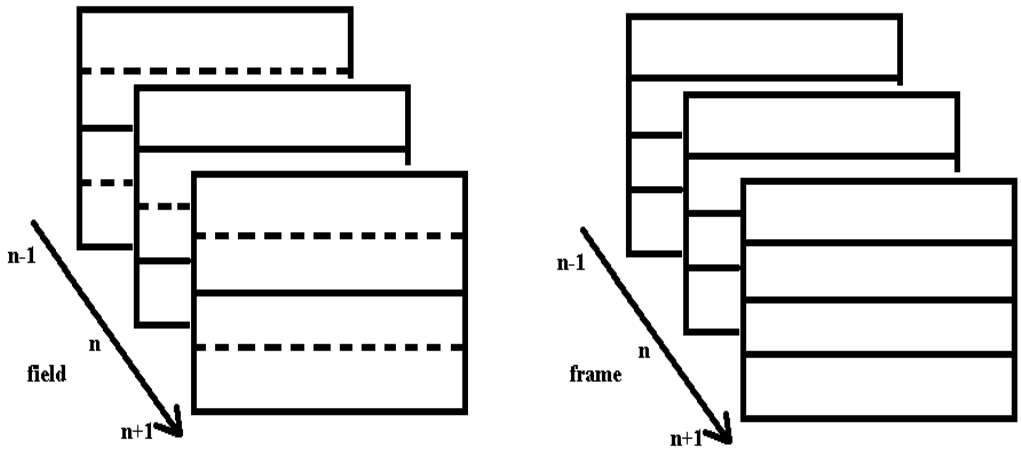
to deinterlace satisfactorily. The support towards an all progressive system is increasing as experiments show that an all progressive systems produces at least as good an image quality as an all interlaced system does [7]. This means that an interlaced system cannot produce any increase in quality but only an increase in performance. With the reasoning that the modern day technologies can support powerful deinterlacers, a combination of interlaced and deinterlaced systems are currently in use. In such systems, for broadcasting and transmission purposes, an interlaced system in used. The receiver has an in-built deinterlacer pre-display unit that deinterlaces the interlaced video. This also satisfied the TV community by removing the requirement for a new broadcast protocol for progressive video. However, this now applies additional pressure on the video processing community to come up with deinterlacing systems that are not only computationally efficient to support real-time video deinterlacing but are also powerful enough in terms of deinterlacing performance.

Deinterlacing is the task of converting a naturally or an artificially interlaced video into a progressive video. Simply put, deinterlacing is the task of converting fields into frames. This process of deinterlacing is shown in Figure 1-2. The methodology discussed in Figure 1-2 is the general process of deinterlacing, though the process of deinterlacing depends on the type of interlacing.

It is reasonable to assume from deinterlacing literature that a video is interlaced during acquisition, owing to the virtue of the data acquisition system. Under this assumption, videos can be considered to be interlaced in two ways.

1. Field $n-1$ and n must belong to the same frame. One frame is split into two fields. Post-deinterlacing, the frame-rate of the deinterlaced video is double.
2. Field $n-1$ and n come from different frames and the deinterlaced frames $n-1$ and n are to be reconstructed into full resolution from the interlaced fields. One frame maps to only one field.

The second method of deinterlacing is usually followed as this maintains the frame rate and the number of frames. This also implies that for the purposes of experimental verification and testing, the original number of frames is maintained and thus, image quality evaluation techniques such as PSNR estimates can still be used. For all intentions and purposes in this manuscript unless otherwise specified interlacing, will refer to the second method.



(a) Interlaced Video **(b) Non-Interlaced Video**

Figure 1-2: Interlaced video after up-sampling and non-interlaced videos. The dashed lines are lines with no data, and the solid lines have data present.

Deinterlacing can be mathematically written as,

$$F_0(\vec{x},n) = \begin{cases} F(\vec{x},n), & y\,mod2 = n\,mod2 \\ F_i(\vec{x},n), & otherwise \end{cases}, \quad \ldots(1.1)$$

where $\vec{x} = (x,y)^T$ are the Cartesian position in the video, n represents the field number, $F(\vec{x},n)$ is the original video content, $F_0(\vec{x},n)$ is the output deinterlaced video and $F_i(\vec{x},n)$ is the deinterlaced data estimate.

Interlacing and deinterlacing not only affect the video in the spatial domain but also in the frequency domain. Interlacing, halves the sampling density of the video in the vertical direction while maintaining the sampling density of the video in the horizontal direction. Deinterlacing increases the vertical sampling density. This repeats the spectrum of the video during the up-sampling involved. When properly interpolated, the first repeat of the spectrum can be removed and this completes the task of deinterlacing. It is not though, just a simple up-conversion problem [8]. This is because most videos don't fulfill the sampling theory.

The temporal frequencies at the retina also have a connection with the scene content [9]. While the observer is tracking an object, the high frequencies due to the object motion are mapped to DC by the retina. But if these high frequencies and seemingly less relevant components of an image are suppressed, it is seen as blurring for the viewer's eye. Therefore, purely temporal filtering will degrade the picture quality.

Figure 1-3: Deinterlaced VS Progressive videos streamed on a Facebook Video Stream player. The video on top is an interlaced video, deinterlaced using LA, while the one on the bottom is a progressive video.

This also shows that the salient regions of the image and regions of high motion are to be deinterlaced with extra-care. A static scene (or a static region) of

the video is a region comprising value of pixels on the same Cartesian location that don't change over time. The sampling lattice of the interlaced video has spectral replicas [10]. When not sampled in accordance with the sampling theory this leads to aliasing. This shows that the process of deinterlacing is a spatio-temporal problem and was tackled in a straightforward way by the vertical temporal filter (VTF) [11]. VTF approach is surveyed in detail in the literature review section.

One simple method of deinterlacing is the line average (LA) algorithm or the 'bob' algorithm. LA algorithm is explained by equation 1.2.

$$F_0(\vec{x},n) = \begin{cases} F(\vec{x},n), & y\ mod 2 = n\ mod 2 \\ (F_o(\vec{x} - \overrightarrow{(0,1)},n) + F_o(\vec{x} + \overrightarrow{(0,1)},n))/2, & otherwise \end{cases}. \quad \ldots(1.2)$$

This is an intra-frame deinterlacing algorithm that averages between the line previous, and the next to the one to be interpolated, in the same frame. LA is one of the methods that is used traditionally in PC video deinterlacing. Videos on PC ever since WinHec'97 [6] are progressive. Figure 1-3 shows the comparison between a LA deinterlaced and a pre-interlaced video. It can clearly be observed that the deinterlaced video has jagged edge artifacts called 'mouse teeth' effect or 'line-crawling effect'. As was discussed earlier, LA is a purely spatial deinterlacing system and doesn't tackle the vertical temporal problem of deinterlacing. LA assumes that the video to be deinterlaced has very little high frequency components and thereby neglects the temporal information that is available.

Most sports broadcasters while broadcasting HD sport videos use progressive videos and broadcast using a 720p (a progressive video) format and not using a 1080i (an interlaced video) format, while they broadcast other regular television in the 1080i format. This is because sports in general consists of a lot of motion and thus leads to irregularities in video, when interlaced. Sports broadcasters including ESPN3, Sky Sports and BBC sport all stream videos at 720p or 1080p rather than 720i or 1080i formats that save them a lot of bandwidth. It's a trade-off between quality and bandwidth. The most used video streaming website, YouTube also uses a 720p and 1080p versions for their HD streaming.

There have been several proposals made to convert the television media into an all progressive system, but such an implementation is not supported by the TV community [12]. Though a great deal of research has been done, the absence of a unified solution shows the complexity and the difficulty of the problem. The rest of this thesis will deal with tackling the deinterlacing problem using novel image processing techniques.

Summary of Introduction

In this chapter the concepts of interlaced videos and deinterlacing were introduced. The challenges put forth by the PC and the TV communities were studied. The LA method of deinterlacing was surveyed and through it, the potential problems of a typical deinterlacer were analyzed.

Chapter 2: HYPOTHESIS AND SPECIFIC AIMS

In this thesis, the following hypothesis will be tested.

2.1 Hypothesis

Using novel method switching techniques that select different deinterlacing methods for specifically targeted regions of an interlaced video, a high quality progressive video can be reconstructed.

2.2 Specific Aims

In order to accomplish the above mentioned hypothesis, the following detailed aims will have to be accomplished.

2.2.1 Specific Aim 1.

To develop an intra-frame deinterlacing method that can interpolate the video field by field independently in the vertical direction and can thus produce an intra-frame deinterlaced video. This is accomplished using one-dimensional control grid interpolation (1DCGI).

2.2.2 Specific Aim 2.

To develop a method to distinguish in a video: regions of visual importance, static regions and regions of relatively high motion. This enables us to develop methods, targeted directly at those regions independently. A decision

making process can be built to choose the method of interpolation to be used on each region. This, along with the 1DCGI methods mentioned in the previous section tackles the spatio-temporal problem that is deinterlacing.

Chapter 3: BACKGROUND

This section will discuss the optical flow problem and its control grid formulations.

3.1 Optical Flow

Motion estimation between frames of a video using optical flow is a well-studied problem [13, 14]. The concept of optical flow was first studied by James J. Gibson, as part of a psychology study [15]. Optical flow refers to the apparent motion of brightness patterns in an image sequence [16]. Instantaneous image velocities or discrete image displacements are ways of measuring or estimating motion in ordered sequences of images.

Optical flow is a gradient-based motion estimation method. It is the perceived 2D motion based on changes in image patterns. It depends on illumination and object surface textures. Though 2D motion and optical flow are used interchangeably, they are not the same as shown in Figure 3-1. It can be seen here that under certain illumination conditions though the object is rotating the motion cannot be detected. On the other hand, if the illumination source was moved, this may still be perceived as object motion.

Motion estimation and video compression typically use optical flow. While optical flow is similar to the estimation of a dense motion field, it is also a study of the structure of the scene. It has a lot of applications in object tracking

[17], movement detection [18], robot navigation and is even used as features for learning based systems [19].

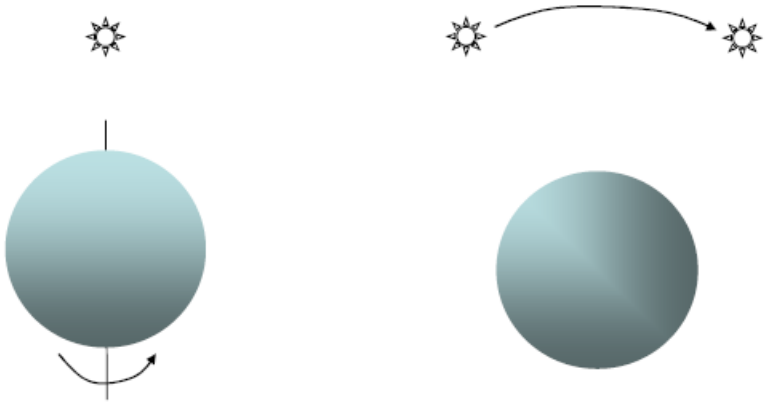


Figure 3-1: 2D motion ≠ Optical flow [20].

In case of the two-dimensional optical flow that is mostly used for videos, 3D motion is modeled as a 2D projection This can be observed in Figure 3-2 where the three dimensional motion of the mountains are outward due to the camera zooming in the Yosemite sequence, the optical flow is a 2D projection. This shows that the optical flow is not a true estimator of motion. Since in many cases, illumination and object/scene models are unknown, optical flow is the best possible way of estimation of motion.

Optical flow is represented in many forms; Global motion, pixel-based motion, block-based motion and region-based motion (Figure 3-3). Global motion based representations of optical flow describe motion for a frame globally. Pixel-based representations of optical flow, represents motion by using one motion vector at each pixel often with some kind of a smoothness constraint between adjacent motion vectors. Block-based optical flow representations involve

dividing a frame into block and characterizing the motion of each block by a motion vector. Region-based representations of optical flow associate a motion vector with semantically segmented regions of video [21, 22]. Other representations include mesh-based pixel representations where control grids are used and for each control grid, a constraint is applied, though motion vectors are given for every pixel location.

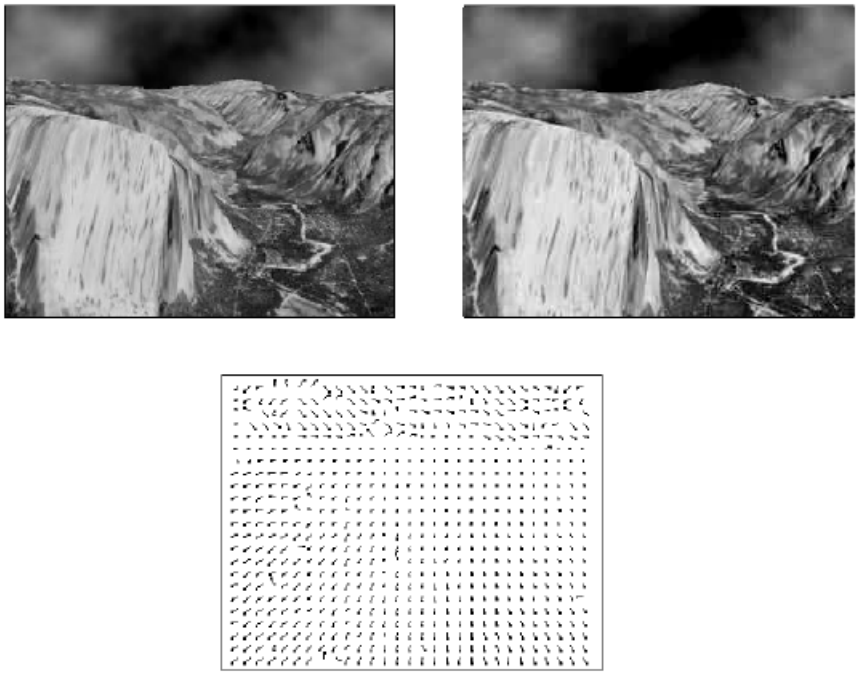


Figure 3-2: Optical flow between two frames of the Yosemite sequence. The top two images are two consecutive frames and the bottom row shows the optical flow [20].

Let $I(x, y, t)$ be a video where, x and y are the pixel locations on a rectangularly sampled frame and let Δx, Δy and Δt be displacements in x, y and time difference respectively. The optical flow now calculates the motion between the two frames taken at t and $t + \Delta t$ for every pixel location as,

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \quad \ldots (3.1)$$

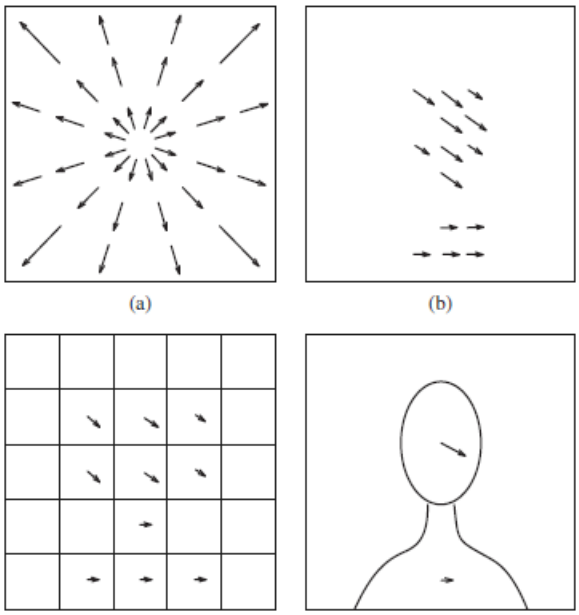


Figure 3-3: Optical flow representations. The top-left is Global representation, top-right is pixel-based representation, bottom-left is block-based representation and the bottom right is object based representation [20].

Usually $\Delta t = T$, where T is the inverse of frame rate (this means frame t, usually called the target frame and frame $t + \Delta t$, usually called the anchor frame are adjacent frames), though this need not be the case in all applications. These types of optical flow estimation techniques are called differential techniques since they work using the differentials and the Taylor series approximations of the image signals. For a 2D+ t case (frame is 2 dimensional with a time axis), $I(x, y, t)$ is expected to have moved by $\Delta x, \Delta y$ and Δt. This means at time $t + \Delta t$, the pixels at x and y are expected to have moved to locations $x + \Delta x$ and y+Δy repectively. It is to be noted that Δx, Δy and Δt need not necessarily be

integers and can also represent sub-pixel and sub-frame offsets respectively. This assumption represented by Equation 3.1 is called the *constant intensity assumption*.

With an assumption that $t + \Delta t$ is just one frame away from t, in which case the motion is relatively small, Equation 3.1 can be expanded using the Taylor series as,

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{dI}{dx}\Delta x + \frac{dI}{dy}\Delta y + \frac{dI}{dt}\Delta t + H.O.T. \quad \ldots (3.2)$$

The higher order terms can be neglected. The brightness constraint equates $I(x + \Delta x, y + \Delta y, t + \Delta t)$ to $I(x, y, t)$. This implies that,

$$\frac{dI}{dx}\Delta x + \frac{dI}{dy}\Delta y + \frac{dI}{dt}\Delta t = 0 \quad \ldots (3.3)$$

This equation can be further simplified by dividing the entire equation by Δt to get,

$$\frac{dI}{dx}\frac{\Delta x}{\Delta t} + \frac{dI}{dy}\frac{\Delta y}{\Delta t} + \frac{dI}{dt} = 0. \quad \ldots (3.4)$$

The terms $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$ are the components of velocities of motion of pixels in x and y directions respectively. These can be represented using V_x and V_y respectively. Equation 3.4 can now be written as,

$$I_x V_x + I_y V_y = -I_t \quad \ldots (3.5)$$

or in vector form,

$$\nabla I^T . \vec{V} = -I_t \quad \ldots(3.6)$$

Equation 3.6 is a linear equation with two unknowns and cannot be solved directly. This problem is known as the aperture problem of optical flow algorithms. To solve for velocities in two directions x and y, if both the differentials $\frac{\Delta x}{\Delta t}$ and $\frac{\Delta y}{\Delta t}$ are non-zero, then we require more equations. There are many solutions to this conundrum and each serves as a method to solve the optical flow problem. Some of the motion estimation methods are listed below.

1. Phase Correlation
2. Block-based methods
3. Differential methods (optical flow based methods)
    i. Lucas-Kanade method [23]
    ii. Horn-Schunck method [24]
    iii. Buxton-Buxton method [25]
    iv. Black-Jepson [26, 27]
4. Discrete optimization methods [28]

For the sake of clarity, Lucas-Kanade method is elucidated in reasonable detail.

The Lucas-Kanade method is a very widely used method of optical flow estimation. To solve equation 3.5, we need to generate more equations, as it is a linear equation with two variables and cannot be solved by itself. To avoid this situation, the Lucas-Kanade method assumes that the flow or velocity is constant in a local neighborhood of the pixel under consideration. This means that for any pixel, the neighboring pixel must also have the same velocities. This is a fairly

reasonable assumption to make in that it generally fits for high resolution videos or natural videos. The other way of putting this assumption is that the Lucas-Kanade method assumes that the displacement of the semantic objects in the image between two frames is small and is also constant within a neighborhood of the point p under consideration. The velocity vectors V_x and V_y should now satisfy the equations,

$$I_x(q_1)V_x + I_y(q_1)V_y = -I_t(q_1)$$
$$I_x(q_2)V_x + I_y(q_2)V_y = -I_t(q_2)$$
$$\vdots \qquad \ldots(3.7)$$
$$I_x(q_n)V_x + I_y(q_n)V_y = -I_t(q_n),$$

where $q_1, q_2, \ldots q_n$ are the pixels inside the chosen neighborhood, $I_x(q_i), I_y(q_i)$ and $I_t(q_i)$ are the partial derivatives of the image I with respect to the pixel location x, y at time t evaluated at the point q_i. This is in effect making an assumption that the points $q_1, q_2, \ldots q_n$ all have same velocity vectors V_x and V_y. Also, when this process is repeated for all the pixels in the video, each pixel gets its own set of velocity vectors; this is also a means of creating an implicitly constrained motion model.

Equations 3.7 can be formulated into a matrix form as $Av = b$, where $A = \begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix}$, $v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}$ and $b = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix}$. Changing the

unsolvable equation 3.5 to this form where there are more equations than unknowns is not optimal as these equations now become over-determined. The Lucas-Kanade method tackles this problem by using a least-squares method. The least-squares formulation is,

$$A^T A v = A^T b \quad \ldots(3.8)$$

Solving for velocities, we get

$$v = (A^T A v)^{-1} A^T b \quad \ldots(3.9)$$

which on expanding gives

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(q_i)^2 & \sum_i I_x(q_i) I_y(q_i) \\ \sum_i I_x(q_i) I_y(q_i) & \sum_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(q_i) I_t(q_i) \\ -\sum_i I_y(q_i) I_t(q_i) \end{bmatrix} \quad \ldots(3.10)$$

There are many ways of choosing the *n* pixels under consideration, but usually it is chosen as a square block where the pixel for which the motion vectors are being calculated, will be in the center. This would mean that all the pixels on the square get equal weightages. In practice it is better to give more weight to the pixels that are closer to the pixel under consideration. This is accomplished by using a weighted formulation of equation 3.7 as

$$A^T W A v = A^T W b. \quad \ldots(3.11)$$

Solving for velocities, we get,

$$v = (A^T W A v)^{-1} A^T W b. \quad \ldots(3.12)$$

Equation 3.12 can be thus expanded and the solutions of velocities can be arrived at as,

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x(q_i)^2 & \sum_i w_i I_x(q_i) I_y(q_i) \\ \sum_i w_i I_x(q_i) I_y(q_i) & \sum_i w_i I_y(q_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(q_i) I_t(q_i) \\ -\sum_i w_i I_y(q_i) I_t(q_i) \end{bmatrix} \quad ...(3.13)$$

where, w_i is usually a Gaussian function. Equation 3.10 can be obtained as a special case of Equation 3.13, if the weights are all equal to 1.

The optical flow method has many drawbacks in the form discussed above. The least-squares approach assumes implicitly, that the error data of the image is a Gaussian distribution with zero mean. If the window contains any 'outliers' the velocity vectors will become noisy. The noise in the velocity vectors can be removed by using a median filter, though this is a crude method and is not usually preferred. On the other hand, one may use statistical analysis to detect them and alter the weight matrix *W* to get the required results.

The biggest drawback of optical flow using the prior discussed formulation is that the error in velocity vectors increases as the range of the velocity vectors expand. In another sense, this method can be used only if the optical flow between two frames is small enough, sometimes in the sub-pixel range. Particularly in applications like stereo matching, and motion estimation with occlusions the use of optical flow is very limited so Lucas-Kandade-Tomasi (KLT) feature matching based tracking is preferred [29]. This is because; the optical flow assumes that for every pixel in the target frame, there is always another pixel in the anchor frame which need not be the case.

Another kind of motion tracking algorithm is the block matching algorithm (BMA). BMA methods are well-studied in literature [30, 31]. In BMA

methods, for every block on the target frame, a distance measure (usually the Euclidean or correlation) is calculated with all the blocks on the anchor frame. The block on the target frame is considered to have moved to a potential location on the anchor frame, whose distance measure is the least. The best that can be achieved without changing the resolution of the video is a pixel level accuracy of motion for each block. The drawback of this method is that noise on a given block can throw off the distance measures and thereby give wrong results. The alternative to achieve sub-pixel accuracy is to interpolate and increase the size of the original image and then perform BMA, though this is computationally inefficient and also depends on the accuracy of the interpolator.

3.2 Control grid formulation of the optical flow and the control grid interpolation

The control grid formulation of optical flow integrates the features of BMAs and optical flow. Control grid interpolation (CGI) using the control grid formulation of the optical flow has been used in many applications particularly in medical imaging and image interpolation [32, 33, 34]. The control grid formulation of optical flow is more often used as an interpolating tool than a motion estimation tool, though it is shown to be effective in both arenas.

To derive the control-grid formulation, we re-write equation 3.2 as,

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t). \quad \ldots(3.14)$$

where Δx,Δy and Δt can now be viewed as a small offset or displacement over x, y and t respectively. In this formulation, even if the displacements are sub-pixel, they can be modeled using a displacement function, as long as the video is discrete in terms of pixel values and time. This understanding leads us to the discrete formulation,

$$I[n_1, n_2, k] = I(n_1 + d_1[n_1, n_2, k], n_2 + d_2[n_1, n_2, k], k + \partial k) \quad \ldots(3.15)$$

where, $\boldsymbol{n} = (n_1, n_2)$ are the discretized locations of pixels in the image, k is discrete time and $k + \partial k$ is usually a frame away from k , $d_1[n_1, n_2, k]$ and $d_2[n_1, n_2, k]$ are two dimensional displacement vectors for each frame k. Using the same Taylor series procedure followed in the previous section, equation 3.15 becomes,

$$I[\boldsymbol{n}, k] \approx I[\boldsymbol{n}, k + \partial k] + \frac{\partial I[\boldsymbol{n}, k+\partial k]}{\partial n_1} \mathrm{d}_1(\mathbf{n}) + \frac{\partial I[\boldsymbol{n}, k+\partial k]}{\partial n_2} \mathrm{d}_2(\mathbf{n}) \quad \ldots(3.16)$$

In the control grid formulation, the image is divided into rectangular blocks and it is assumed that the pixel displacements inside each block B can be modeled as,

$$d_1(\boldsymbol{n}) = \sum_{i=1}^{p} \alpha_i \theta_i(\boldsymbol{n}) \quad \ldots(3.17)$$

and

$$d_2(\boldsymbol{n}) = \sum_{i=1}^{p} \beta_i \phi_i(\boldsymbol{n}) \quad \ldots(3.18)$$

where $\theta_i(\boldsymbol{n})$ and $\phi_i(\boldsymbol{n})$ are independent basis functions that model the displacement field. In this case the parameters to be estimated are α_i and β_i these

are the displacement parameters. Equations 3.17 and 3.18 can be written in vector form as,

$$d_1(\boldsymbol{n}) = \bar{\alpha}^T \overline{\theta}(\boldsymbol{n}) \quad \ldots(3.19)$$

$$d_2(\boldsymbol{n}) = \bar{\beta}^T \overline{\phi}(\boldsymbol{n}) \quad \ldots(3.20)$$

where, $\bar{\alpha}$, $\bar{\beta}$, $\bar{\theta}$ and $\bar{\phi}$ are vectors with α_i, β_i, θ_i and ϕ_i respectively.

For bi-linear basis functions, $\bar{\theta}$ and $\bar{\phi}$ can be defined explicitly as,

$$\theta_1(n_1, n_2) = \phi_1(n_1, n_2) = \left(\frac{n_1^2 - n_1}{n_1^2 - n_1^1}\right)\left(\frac{n_2^2 - n_2}{n_2^2 - n_2^1}\right), \quad \ldots(3.21)$$

$$\theta_2(n_1, n_2) = \phi_2(n_1, n_2) = \left(\frac{n_1^2 - n_1}{n_1^2 - n_1^1}\right)\left(\frac{n_2 - n_2^1}{n_2^2 - n_2^1}\right), \quad \ldots(3.22)$$

$$\theta_3(n_1, n_2) = \phi_3(n_1, n_2) = \left(\frac{n_1 - n_1^1}{n_1^2 - n_1^1}\right)\left(\frac{n_2^2 - n_2}{n_2^2 - n_2^1}\right), \quad \ldots(3.23)$$

and

$$\theta_4(n_1, n_2) = \phi_4(n_1, n_2) = \left(\frac{n_1 - n_1^1}{n_1^2 - n_1^1}\right)\left(\frac{n_2 - n_2^1}{n_2^2 - n_2^1}\right). \quad \ldots(3.24)$$

The kernels of Equations 3.21 to 3.24 can be visualized in the Figure 3-4. It can be noticed that the sum of independent kernels at each given point in the grid is always equal to one. This is essential to maintain the smooth interpolation of the motion vectors. It is also noteworthy, that in the corners, only one of the basis functions can be one. The edge points of the control grids are called the control points. It is noteworthy that control points are shared by adjacent control

grids, and while calculating the displacements, the displacements from previous iterations are carried over. This provides a truly connected motion model.

Similar to $\bar{\theta}$ and $\bar{\phi}$, $\bar{\alpha}$ and $\bar{\beta}$ also have four components, each component can be understood as the component of the velocity vector at the corresponding grid corner. Equation 3.16 can now be written in terms of $\bar{\alpha}$ and $\bar{\beta}$ in vector form as,

$$I[\boldsymbol{n}, k] \approx I[\boldsymbol{n}, k + \partial k] + \frac{\partial I[\boldsymbol{n}, k+\partial k]}{\partial n_1} \bar{\alpha}^T \bar{\theta}(\boldsymbol{n}) + \frac{\partial I[\boldsymbol{n}, k+\partial k]}{\partial n_2} \bar{\beta}^T \bar{\phi}(\boldsymbol{n}). \quad \text{...(3.25)}$$

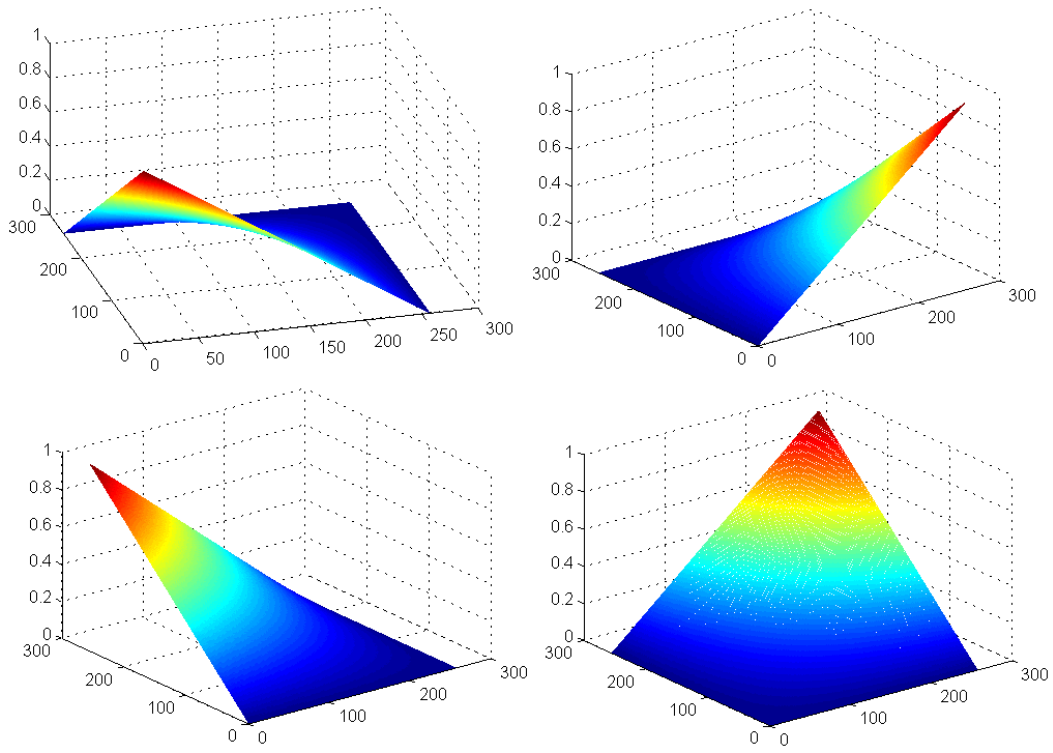


Figure 3-4: Visualization of the bilinear basis. The four diagrams show the plot of the four basis for the same grid. It can be noticed that the sum of the four basis at any given point is always equal to 1. This is essential for any basis that might be used. The basis are calculated for a grid of 256X256. This is only for the purposes of visualization and is not a standard.

Now that the velocity vectors inside a given grid are connected using the basis vectors $\bar{\theta}$ and $\bar{\phi}$, it is only essential to record the velocity vectors at the corners of the control grid. Once the velocity vectors are obtained for one grid, the corners of the grid that connect to another grid maintains the velocity vector values in the other grid. This makes the entire model a connected motion model. Equation 3.25 is solved using a least squares approach as was the case with the optical flow formulation in the previous section. If R is the region under consideration, the least squared error in terms of the vectors $\bar{\alpha}$ and $\bar{\beta}$,

$$E(\bar{\alpha},\bar{\beta}) = \sum_{n\epsilon R}(I[\boldsymbol{n},k] - I[\boldsymbol{n},k+\partial k] - \frac{\partial I[\boldsymbol{n},k+\partial k]}{\partial n_1}\bar{\alpha}^T\overline{\theta}(\boldsymbol{n}) - \frac{\partial I[\boldsymbol{n},k+\partial k]}{\partial n_2}\overline{\boldsymbol{\beta}}^T\overline{\boldsymbol{\phi}}(\boldsymbol{n})^2,$$

…(3.26)

is minimized. This is repeated for all such grid regions of the image. After one entire pass is made, the same process can be repeated for the entire image using the previous estimates as starting points, this will further reduce the error however, this increases the computation time.

Using the displacements vectors thus found, from a given target frame an entire anchor frame can be reconstructed by interpolation. This type of interpolation is called control grid interpolation (CGI) and is discussed in [33]. This interpolation is accomplished by *carrying on temporally* to the sub-pixel locations that the motion vectors point to, the original pixel values. An original resolution grid of the image can now be reconstructed using any interpolation

technique such as linear, bilinear, bicubic or triangulation depending on the available computational power and requirement.

To further avoid errors, two displacement fields are constructed, one from k to $k + \partial k$ (displacements from first frame to second frame) and another from $k + \partial k$ to k (displacements from second frame to first frame). This leads to two reconstructed images and the two images are combined in a spatially weighted sum to create the final interpolated image. The method does not create an intermediate frame between the first and the second frame (one bidirectional mapping), rather creates two different frames (two unidirectional mappings), although this would mean that the flow field is not symmetric.

3.3 One dimensional control grid interpolation (1DCGI)

1DCGI is the 1D formulation of the 2D optical flow discussed in the previous section [35]. 1DCGI is used in applications like demosaicing and image interpolation [36]. While 2DCGI works between frames of a video, 1DCGI works between lines of an image. All the analysis and the formulations of constraints formed on the original optical flow section all exist in here as a 1D analogue.

The optical flow described by equation 3.1 can now be re-iterated in 1DCGI formulation as,

$$I(x, y) = I(x + \alpha, y + 1) \qquad \ldots(3.27)$$

The term $y + 1$ shows that the lines must be horizontal in the imaging context, though whether it is row or column is immaterial as both ways the two parties involved are still one dimensional. While in the optical flow formulation, the brightness constraint is that the intensity in the first frame is preserved somewhere in the second frame, in the 1D control grid formulation with only one degree of freedom, the brightness constraint becomes the assumption that the intensity in the first line is preserved in the second line. Interpolation is performed as a weighted average of the intensities of pixels along the displacement vectors between the two lines and those displacement vectors will be one dimensional. The differentials now represent intensity gradients.

Following the Taylor series approximation, the equation 3.16 analogue of 1DCGI is,

$$I(x, y) \approx I(x, y) + \frac{\partial I(\boldsymbol{x,y})}{\partial x}\alpha + \frac{\partial I(\boldsymbol{x,y})}{\partial y} \qquad ...(3.28)$$

This equation signifies that for any point (x, y) there exists a displacement α, such that $I(x + \alpha, y + 1)$ and $I(x, y)$ have the same pixel intensity. The deviation from this brightness constraint can be defined as,

$$E(\alpha) = \left[\frac{\partial I(x,y)}{\partial x}\alpha + \frac{\partial I(x,y)}{\partial y}\right]^2. \qquad \ldots(3.29)$$

The error approaches zero as the value of α approaches $\hat{\alpha}$, such that

$$\hat{\alpha} = -\frac{\boldsymbol{\partial I(x,y)}}{\boldsymbol{\partial y}} \Big/ \frac{\boldsymbol{\partial I(x,y)}}{\boldsymbol{\partial x}}. \qquad ...(3.30)$$

Like the 2DCGI case, the displacements are defined at regularly spaced control points. Intermediate displacements are generated using linear interpolation. Unlike the 2D case though only two basis functions, θ_1 and θ_2 are used. A linear set of kernels,

$$\theta_1 = \frac{k-i}{k} \quad \text{and} \quad \theta_2 = \frac{i}{k} \qquad \ldots(3.31)$$

are preferred in the 1D case, where i is the distance from the previous node. In the 1D case, linear, cubic or Hermite interpolation is performed to get the new pixels into grid depending on the application. In later chapters, it will be seen that 1DCGI is a good method for performing spatial deinterlacing.

Like the 2D formulation again, here two unidirectional displacements (top to bottom and bottom to top) are calculated and are averaged to produce the optimal *central* interpolated line. Adaptations of this technique can be used in other methods like super-resolution also.

3.4 Summary of background

In this chapter the concept of optical flow was analyzed in detail along with its 2D and 1D control grid formulations. 1DCGI, the line-line interpolation technique was also studied in detail.

Chapter 4: LITERATURE REVIEW

Deinterlacing is a well-studied topic. The last century has seen a lot of deinterlacing algorithms being proposed. The literature of deinterlacing can be studied under two broad categories, Linear and Non-Linear deinterlacers.

4.1 Linear Deinterlacers

A linear deinterlacer can be described by,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\,, & (j\ mod 2 = n\ mod 2) \\ \sum_m \sum_k F_{n+m}(i,j+k)\ h_m(k)\,, & (otherwise) \end{cases}, \qquad \ldots(4.1)$$

where $F_n(i,j)$ represents the pixel value at position (i,j) in the n^{th} field, $\widehat{F_n}(i,j)$ denotes the reconstructed pixel value at the same position, and $h_m(k)$ denotes the filter weights for field index m and vertical index k. It can also be noticed that the choice of the impulse response $h_m(k)$ determines whether the filter is spatial, temporal or spatio-temporal.

4.1.1 Spatial Deinterlacers

Weave [5] is a spatial linear deinterlacing algorithm that is very popular. The weave algorithm has two implementations, a purely spatial implementation and another purely temporal implementation. The spatial weave algorithm, also called as field repetition can be seen as a generalization of the equation 4.1 if,

$$h_m(k)_{weaves} = \begin{cases} 1 & m = 0 \ and \ k = -1 \\ 0 & otherwise \end{cases} . \quad \ldots(4.2)$$

A similar purely spatial algorithm is the line averaging algorithm (LA) or the *bob* algorithm that is so commonly used in the PC community [6]. LA was already discussed in equation 1.2 can be generalized using equation 4.1 if,

$$h_m(k)_{LA} = \begin{cases} 0.5 & m = 0 \ and \ k = -1,1 \\ 0 & otherwise \end{cases} . \quad \ldots(4.3)$$

They have an all-pass characteristic on the temporal, which ascertains that these filters have no motion artifacts. These filters don't have a really steep cut-off so it can allow the alias components that arise due to improper sampling of the videos. These can be suppressed by having a longer frequency response though this might suppress the higher part of the baseband spectrum also. It is harder for a purely spatial filter to differentiate between baseband and repeat spectrum, no matter what their length is and they always balance between alias and resolution. It can be seen from Figure 4-1 where the region shaded in mild gray represents the pass-band of a typical purely spatial filter and the region shaded in dark gray represents the spectrum of an ideal deinterlacer, that the purely spatial filter might suppress the vertical detail and allows the alias spectrums.

4.1.2 Temporal Deinterlacers

The purely temporal implementation of weave, also called as field insertion can be seen with the filter weights,

$$h_m(k)_{weaveT} = \begin{cases} 1 & m = -1 \ and \ k = 0 \\ 0 & otherwise \end{cases}. \qquad \ldots(4.4)$$

The temporal weave algorithm has an all-pass characteristic on the vertical, while it still allows the temporal alias spectrum. The spectrum of a purely temporal weave algorithm can be seen in Figure 4-2. This is the best solution in the case of a still image as all vertical frequencies are preserved. However this produces a *serration* noise on videos with motion. A typical noisy video that was deinterlaced using temporal weave can be seen in Figure 4-3.

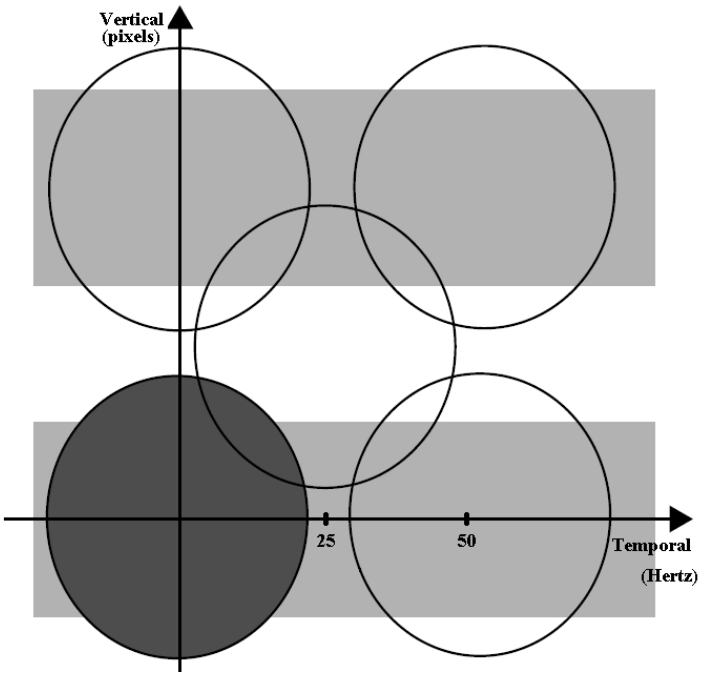


Figure 4-1: Spectrum of a purely spatial filter. Dark gray region is the spectrum of an ideal deinterlacer, while the mild gray region is the spectrum of a purely spatial filter. The dark gray is the true spectrum while other rings are alias spectrums.

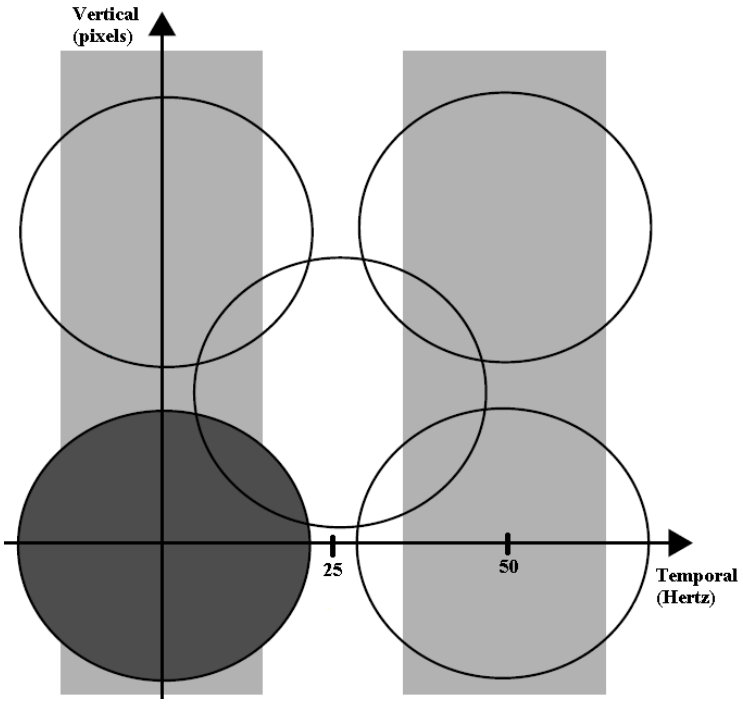


Figure 4-2: Spectrum of a purely temporal filter. Dark gray region is the spectrum of an ideal deinterlacer, while the mild gray region is the spectrum of a purely temporal filter. The dark gray is the true spectrum while other rings are alias spectrums.

Figure 4-3: Foreman video deinterlaced using the temporal weave algorithm.

As with the spatial weave, the quality of temporal weave can be improved by using a longer impulse response, though a longer FIR filter would require a buffer of higher memory and are not economically viable.

4.1.3 Vertical temporal filter (VTF)

A vertical temporal filter as discussed in equation 4.1 is the optimal linear deinterlacer. If the signal is band-limited and the video is sampled following the sampling theorem (thereby no-alias), then the VTF is the ideal filter [5]. Vertical temporal filter (VTF) is one of the most popular methods of deinterlacing and the most widely used implementation of it was proposed by Weston [11]. For the intentions and purposes of evaluation, this version of VTF is considered.

VTF is a spatio-temporal approach that directly tackles the vertical-temporal spectrum of interlaced videos. The filter works on windows of 3 or 5 pixel squares and are called the 3-tap or the 5-tap versions respectively. Most systems using VTF prefer the 3-tap approach. 5-tap approaches are preferred only in case of synthetic videos. Theoretically VTF is the ideal linear filter, if the input video is band limited and if the filter $h_m(k)$ is well designed. More often than not, the response proposed in Equation 4.5 is preferred as it was proposed for natural videos with consistent VT spectrum. The filter coefficients proposed by Weston are,

$$h_m(k) = \begin{cases} \frac{1}{2}, \frac{1}{2} & , (k = -1,1 \;\&\; m = 0) \\ -\frac{1}{16}, \frac{1}{8}, -\frac{1}{16} & , (k = -2,0,2 \;\&\; m = -1,1) \end{cases} \quad \ldots(4.5)$$

Figure 4-4 shows that the VTF is the optimal linear deinterlacer. It prevents both alias and blur. Though the vertical detail is gradually reduced with the increasing temporal frequencies, it is not unnatural and is tolerable in videos with motion. The designs of such filters will be in such a form that the contribution of neighbors is limited in the vertical direction. Equation 4.5 shows such a trend.

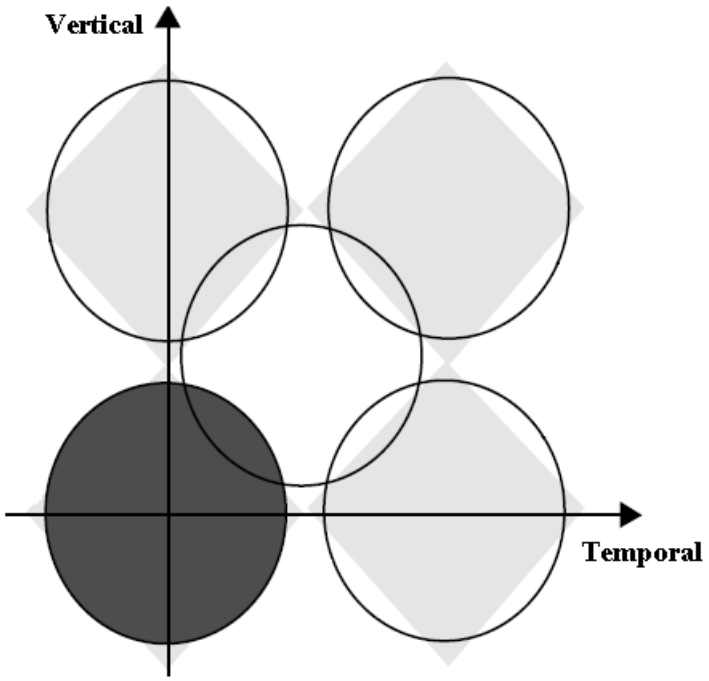


Figure 4-4: Spectrum of a vertical temporal filter. Dark gray region is the spectrum of an ideal deinterlacer, while the mild gray region is the spectrum of a VT filter. The dark gray is the true spectrum while other rings are alias spectrums.

4.2 Non-Linear deinterlacers

Non-linear deinterlacers can be classified into edge-oriented methods [37, 38, 39, 40, 41] and motion-adaptive methods [41, 42, 43, 44, 45, 46, 47]. The motivation for non-linear deinterlacers is to capture the motion and/or vertical detail. Edge-based deinterlacers use directional edge information on in-frame [48] or inter-frame basis [47] to interpolate along the edges. Content-adaptive deinterlacers smartly choose between in-frame and inter-frame information. In this chapter we shall review some edge-based detectors in detail like edge-based line averaging (ELA), spatio-temporal edge-based median filtering (STELA) and some other content-adaptive deinterlacers.

4.2.1 Edge-based line averaging (ELA)

ELA is one of the popular deinterlacing algorithms that use intra-frame deinterlacing. ELA was first proposed by Doyle [49] and was later studied in detail by Kuo [39]. These algorithms were proposed with the argument that the human eyes are very sensitive to edges. Most methods use directional differences to obtain the edge information in each direction and try to interpolate along the direction of the edge. ELA works on a $3X3$ neighborhood to extract the edge information. Figure 4-5 shows a typical spatial neighborhood of an ELA system. The directional differences are calculated as

$$C1 = |a - f|, C2 = |b - e|, C3 = |c - d| \quad \ldots(4.6)$$

where, a, b, c, d, e, f are defined as in Figure 4-5 and X is the point to be deinterlaced. The minimum difference among $C1$ to $C3$ is chosen and the interpolated value is the average of the two points that corresponded to the minimum difference.

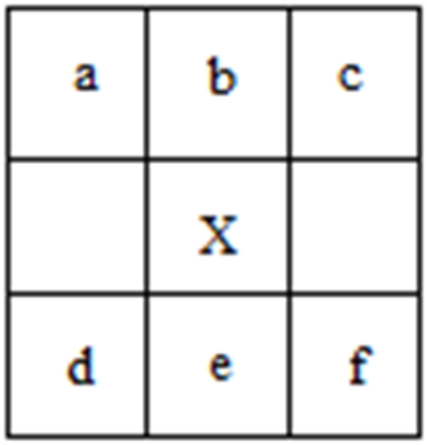


Figure 4-5: Spatial neighborhood of a window used in ELA.

An intra-frame deinterlacer like ELA assumes that there is maximum motion in the video and thus temporal interlacing will lead to worst case results. This is a reasonable assumption to make in the case of a video with high motion but in cases of natural motion, this is a drawback. Most videos usually have a background and a foreground. While the background consists of reasonably low motion, the foreground contains high motion. In spite of these drawbacks, ELA is a still an active area of research.

4.2.2 Spatio-temporal edge-based median filtering (STELA)

STELA [50] is a spatio-temporal edge-based deinterlacer. The motivation behind the development of STELA is that adding the temporal information into an already powerful intra- frame approach like ELA will increase its performance on regions of relatively low motion. STELA works on three $3X3X3$ saptio-temporal neighborhoods and forms six directional differences in both the temporal and the spatial sense.

STELA first divides a video into a low-frequency and a high-frequency version. Figure 4-6 shows the spatio-temporal neighborhood of a STELA approach proposed by Oh et, al [50]. The six directional differences of STELA with respect to the nomenclature in Figure 4-6 are,

$$C1 = |a - f|, C2 = |b - e|, C3 = |c - d|$$

$$C4 = |g - l|, C5 = |h - k|, C6 = |i - j|. \quad \text{...(4.7)}$$

The deinterlacing is now performed as $y = Med\{A, b, e, h, k\}$, where A is the average value of the two points that yield the minimum directional change among $C1$ to $C6$ in the low frequency frame.

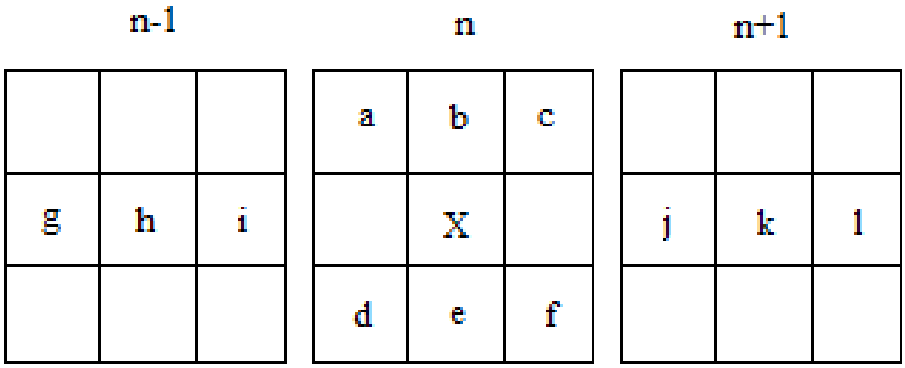


Figure 4-6: Spatio-temporal neighborhood of a window used in STELA.

It is noteworthy that a median filter is performed as a step beyond just the mean value A. Typically it will be noticeable that A would do the job in place of X. The median is made to work like a backup tool in case there was noise in the video. If in case there was noise in the video and that altered the decision to choose A, the median would eliminate A and still provide a reasonably acceptable result. While the median-mean filtering approach is performed on the low frequency frames, the high frequency frames are subject to line doubling. This line doubled version is added to y. STELA attempts to preserve both the horizontal and vertical frequencies of the image. STELA shows that spatio-temporal methods work better than spatial deinterlacers like ELA when the target video consists of both regions of background containing no/little motion and foreground containing regions of relatively high motion.

4.2.3 Content-adptive and method switching deinterlacers

Recently, the deinterlacing community is seeing a boom in content-adaptive and method switching deinterlacers. These deinterlacers are smart in the sense that they accommodate for the fact that a purely temporal deinterlacer performs best on still images, a purely spatial deinterlacer performs best on videos with high motion and a spatio-temporal deinterlacer performs best on videos with combinations of both. They quite simply perform a combination of all the three based on the content of the video or based on the *level* of motion in particular regions of the video.

Some of the recent state-of-the-art deinterlacers are the content-adaptive vertical temporal filters (CAVTF) [51, 52, 53]. The work done by Hong et, al. [53] works with ELA as the core algorithm. CAVTF on the other hand proposed by Lee et, al. [51] chooses a different filter weight for a different VTF based on the content of the video. These filter weights are learned online and has the freedom of varying anywhere from purely spatial to purely temporal. These algorithms are very sophisticated and have only recently started getting popular, owing to increasing computational performances. CAVTF is an algorithm published in the year 2011 and along with VTF, ELA and STELA will be considered the benchmark for the proposed thesis.

4.3 Summary of literature review

Many deinterlacing methods, both linear and non-linear are studied in detail. In summary, by observing the methods in literature, some noteworthy points can be made.

1. A purely spatial edge-based deinterlacer would perform best on regions of high motion. To solve this 1DCGI will be used and will be compared against ELA which is also a purely spatial approach.
2. A purely temporal weave algorithm ideally performs best on regions with no motion.
3. A spatio-temporal VTF performs optimally on regions containing moderate motion.

Chapter 5: PROPOSED DEINTERLACING SYSTEM

The deinterlacing system proposed in this thesis is a non-linear adaptive system. The understanding from the last chapter is that a good deinterlacing algorithm has to tackle three regions in a video.

1. Region of video that has no motion or a static region. This requires a purely temporal deinterlacer.
2. Region of video that has moderate motion. This requires a spatio-temporal deinterlacer
3. Region of video that has very high motion. This requires a purely spatial deinterlacer.

The above statements can be put into a mathematical template as,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\,, & (j\ mod 2 = n\ mod 2) \\ a, A & \\ b, B & \\ c, C & \end{cases} \quad \ldots(5.1)$$

where, a, b and c are interpolation techniques and A, B and C are conditions that specify regions of motion in increasing motion content respectively. To elucidate further,

1. a is the interpolation performed for the region defined by A that specifies static region in a video.
2. b is the interpolation performed for the region defined by B that specifies region with low motion.

3. c is the interpolation performed for the region defined by C which specifies region with highest motion in a video.

The rest of this chapter will go through this template region by region to solve the deinterlacing problem.

5.1 Solving for static regions of the video

Region that satisfies any condition A is a static region if the pixel differences across that region temporally are zero. Let us define d_n the difference map for the n^{th} frame such that,

$$d_n = \|F(n+1) - F(n-1)\| \quad \ldots(5.2)$$

where $F(k)$ represents the k^{th} frame of the interlaced video. To bring the difference image d_n up to the resolution, we perform line doubling on this image. The image can be easily threshold to 1 bit difference. What this would mean is that, if there is even one bit difference, the pixel will not be considered a static pixel.

Once the static region is identified this way, it is easy to interpolate for it. We simple perform field insertion.

$$F_n(i,j) = F_{n-1}(i,j) \quad \ldots(5.3)$$

As can be seen from Figure 4-2, any purely temporal deinterlacer performs best on static regions, which is further supported by Figure 4-3 where the wall and other static regions are perfectly recovered. This solves one part of the problem.

Equation 5.2 represents the condition A and equation 5.3 represents the interpolation process a. Equation 5.1 can be updated using this information and and we get,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\ ,\ (j\ mod 2 = n\ mod 2) \\ F_{n-1}(i,j)\ , d_n(i,j) < t_1 \\ b, B \\ c, C \end{cases} \quad \ldots(5.4)$$

where, t_1 is 1 bit depth.

5.2 Solving for semi-static regions of the video

As was observed from Figure 4-4, VTF is a good linear deinterlacer, in semi-static regions. In this approach we just follow the same understanding and use the Weston's VTF as described in [11]. This means that the interpolator b can be formulated as,

$$F_n(i,j) = \sum_m \sum_k F_{n+m}(i, j+k)\, h_m(k) \quad \ldots(5.5)$$

where

$$h_m(k) = \begin{cases} \frac{1}{2}, \frac{1}{2} & ,(k = -1{,}1\ \&\ m = 0) \\ -\frac{1}{16}, \frac{1}{8}, -\frac{1}{16} & ,(k = -2{,}0{,}2\ \&\ m = -1{,}1) \end{cases} \quad \ldots(5.6)$$

Identifying regions of very high motion and regions of semi-static motion is difficult. This is in a way equivalent to identifying cognitively, the regions of foreground and background with the assumption that foreground is usually the

region consisting of high motion. This assumption is reasonable to make with respect to natural videos.

5.2.1 Foreground extraction using visual saliency detection

Objects of visual saliency are usually the foreground of the image [54]. Many videographers and photographers make sure that the content that they want to be noticed first is presented as the region of most visual saliency in a video [55]. This understanding is followed in many applications like video retargeting [56], seam carving [57] and probabilistic tracking [58].

This problem is usually solved by trying to detect all objects in the scene and then detecting which object is salient [59]. Since this model is based on training, it is not suitable for an almost real-time process like deinterlacing. For the deinterlacing case, it is best to implement saliency detection without prior statistical knowledge of the scene or the contents of the scene.

Human saliency detection is a twofold process:

1. The pre-attentive process
2. The attention process

The pre-attentive process is parallel, fast and simple. Its properties and modeling are well-studied in literature [60] [61]. In the pre-attentive state, the low level information like edge, orientation and intensities are noted automatically. From the saliency detection perspective, this information gives the candidate foreground in a video. To solve this issue, in the coherence theory, the concept of proto-

objects was introduced [62] [63] [64]. The first systems that proposed the use of the term *saliency*, to mimic human perception was proposed by Itti and Koch [65] [66].

Recently studies have been performed to extend saliency models into recognition tasks [67]. But just for pre-processing these systems are computationally demanding. Saliency detection is as much detection of foreground as it is to detect backgrounds. The first work to take this approach is the spectral residue approach proposed by Hou and Zhang [68]. This was later extended into a spatio-temporal implementation by Guo et, al [69].

In [69], a procedure for identifying saliency is presented as a representation of spectral residue in a spatio-temporal domain. It is proposed that the spatial saliency can be given by low-pass filtering the phase of an image's Fourier transform. In the case of color videos, the phase of a quaternion Fourier transform is used.

Quaternion Fourier transform of an image is well presented in [70]. A color image can be represented using quaternions of the form:

$$q^n = Ch_1^n + Ch_2^n\mu_1 + Ch_3^n\mu_2 + Ch_4^n\mu_3 \qquad \ldots(5.7)$$

where $\mu_i,\ i = 1,2,3$ satisfies, $\mu_i{}^2 = -1, \mu_1 \perp \mu_2, \mu_2 \perp \mu_3,\quad \mu_1 \perp \mu_3$. The three color channels of the images are allocated to Ch_2, Ch_3 and Ch_4. Ch_1 is set to zero. The Quaternion Fourier Transforms (QFT) of the frame *n* are

$$Q_i^n[u,v] = \frac{1}{\sqrt{WH}} \sum_{y=0}^{W-1} \sum_{x=0}^{H-1} e^{\mu_1{}^{2\pi\left(\frac{yv}{W}+\frac{xu}{H}\right)}} q_1^n(x,y) \qquad \ldots(5.8)$$

and

$$q_i^n[x,y] = \frac{1}{\sqrt{WH}}\sum_{v=0}^{W-1}\sum_{u=0}^{H-1} e^{\mu_1 2\pi\left(\frac{yv}{W}+\frac{xu}{H}\right)} Q_1^n[u,v] \quad \ldots(5.9)$$

where (x,y) are the pixel locations of individual pixels and $[u,v]$ describe the frequencies. *W* and *H* are the width and height of the frame. The phase spectrum of $Q^n[u,v]$ is given by,

$$Q_p = Q/||Q||. \quad \ldots(5.10)$$

The spatial saliency map, S_n is obtained by Gaussian smoothing (σ = 8) the L_2 norm of the inverse QFT of Q_p, as defined by equation 5.11.

$$S_n(i,j) = g * q_p \quad \ldots(5.11)$$

The spatial saliency map is then used as part of the decision making process in Equation 5.4 with a threshold of t_2 where t_2 is 4% of the bit depth.

Figure 5-1: *Mother* video and the detected saliency, post thresholding.

Figure 5-1 shows the mother video and its visual saliency. It can be observed how the non-salient regions (in black) are also regions of less motion. This solves the problem B which can be phrased as $S_n(i,j) < t_2$; & $d_n(i,j) \geq t_1$. These conditions together specify the region of moderate motion which is non-static. Equation 5.1 can be updated as,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\,, \ (j\ mod 2 = n\ mod 2) \\ F_{n-1}(i,j)\ \ , d_n(i,j) < t_1 \\ \sum_m \sum_k F_{n+m}(i,j+k)\, h_m(k)\,, S_n(i,j) < t_2;\ d_n(i,j) \geq\ t_1 \\ c, \quad (otherwise) \end{cases} \quad \ldots(5.12)$$

It is noteworthy that the condition specifying region C, is now replaced by $otherwise$.

5.3 Solving for the regions of high motion in the video

ELA as an intra-frame deinterlacer was studied in the previous section. STELA performs a similar job while working in a spatio-temporal context. A drawback of this approach is that there is only a quantized set of angles of edges along which interpolation is performed. 1DCGI as was discussed earlier has been demonstrated as effective at detecting sub-pixel edge orientations.1DCGI can be directly adapted to deinterlacing as,

$$I(x+\alpha, y+1) = \frac{1}{2}[I(x,y) + I(x+2\alpha, y+2)] \quad \ldots(5.13)$$

where the displacements α are solved as was mentioned in Chapter 3. To solve Equation 5.13 1DCGI optical flow equation specified by Equation 3.27 is modified to the form,

$$I(x, y) = I(x + 2\alpha, y + 2) \quad \ldots(5.14)$$

As was discussed earlier, two uni-directional interpolations are performed. Equation 5.14 describes the first of these directional interpolators while the second can be written as,

$$I(x, y + 2) = I(x + 2\alpha, y) \quad \ldots(5.15)$$

The two interpolated estimates for the same line are arrived at using Equations 5.14 and 5.15. The two estimates are averaged to get the one deinterlaced line.

A faster way to perform the 1DCGI formulation is using the segment adaptive gradient approach [71]. This allows for a numerical solution with $O(N)$ complexity. Once all the lines are interpolated this way, a frame of 1DCGI estimates is formed.

Equation 5.1 can now be stated in its complete form as,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\,, \ (j\ mod 2 = n\ mod 2) \\ \frac{F_{n-1}(i,j)+F_{n+1}(i,j)}{2}\ , d_n(i,j) < t \\ \sum_m \sum_k F_{n+m}(i, j+k)\, h_m(k)\,, S_n(i,j) < b;\ d_n(i,j) \geq\ t \\ 1DCGI(i,j), \quad (otherwise) \end{cases} \quad \ldots(5.16)$$

Equation 5.16 completes the proposed deinterlacing system. The proposed system is a method switching algorithm that chooses and implements a suitable method of interlacing for specific regions of video.

5.4 Summary of the proposal

A novel deinterlacing approach is proposed as a method switching algorithm, switching between, Temporal LA, VTF and 1DCGI for different regions of the video.

Chapter 6: SPIN OFF ALGORITHMS

Several spin off algorithms were also developed along with the approach proposed in the previous chapter. This are developed either to evaluate 1DCGI as an independent deinterlacer or to produce deinterlacers of intermediate performance while increasing the computational efficiency. Some of them are discussed in this chapter.

6.1 1DCGI as a standalone intra-frame deinterlacer

Using equation 5.13, 1DCGI can be considered as an independent deinterlacer that works only on the spatial and not on the temporal. 1DCGI can be compared with ELA. All the more 1DCGI can also be generalized as an edge-directed deinterlacer similar to that of ELA.

There is one major difference though. While ELA considers 3 possible directions of edges per pixel, SAGA virtually considers infinite possible directions of edges. This accounts for a substantial increase in performance. This provides for 1DCGI to be considered as a standalone deinterlacing algorithm, comparable to ELA.

6.2 1DCGI+TF – A spatio-temporal extension

SAGA can be further extended into the temporal domain while still not going through the saliency detection process. This makes the computation a little

faster. 1DCGI+TF (where TF stands for temporal filter) is a motion adaptive algorithm where, the scene is classified into only two regions [72]:

1. Static region
2. Moving region

Using the same difference map d_n, as proposed in equation 5.2 for classifying a static region from a moving region, the new approach 1DCGI+TF can be mathematically formulated as,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\ ,\ (j\ mod 2 = n\ mod 2) \\ F_{n-1}(i,j)\ \ ,d_n(i,j) < t \\ 1DCGI(i,j) + \sum_m \sum_k F_{n+m}(i,j+k)\ h_m(k)\ ,(else) \end{cases} \quad \ldots(6.1)$$

where,

$$h_m(k) = \begin{cases} 0{,}0\ \ ,(k = -1{,}1\ \&\ \ m = 0) \\ \frac{-1}{16}, \frac{1}{8}, \frac{-1}{16}\ \ ,(k = -2{,}0{,}2\ \&\ m = -1{,}1) \end{cases} \quad \ldots(6.2)$$

1DCGI is now given additional information with the filter response of a purely temporal filter defined by equation 6.2. This solves the issues of 1DCGI as a standalone deinteralcer. It is to be noted that the filter coefficients of $h_m(k)$ in this approach adds up to 0 as against 1 when used with VTF and other linear approaches. 1DCGI+TF is seen as an approach comparable to VTF and STELA, the other spatio-temporal approaches.

6.3 Spatio-temporally median assisted 1DCGI

Another spin-off method using 1DCGI that is considered here is a spatio-temporally median assisted 1DCGI (ST-1DCGI). In this method 1DCGI is further assisted to avoid noisy pixel influences with the addition of a spatio-temporal median from a neighborhood as shown in Figure 4-6. This approach can be mathematically demonstrated as,

$$\widehat{F_n}(i,j) = \begin{cases} F_n(i,j)\,, \ (j\ mod 2 = n\ mod 2) \\ median(1DCGI(i,j), A, b, e, h, k) \end{cases} \quad \ldots(6.3)$$

The variables in the equation are as defined in Figure 4-6 for each pixel under consideration. This method is also a strong competitor to STELA. The main advantage with this method is that the interpolator has a spatio-temporal choice of content.

6.4 Using 2DCGI to build a 1D deinterlacer

A 2DCGI interpolator is adopted as a parallelized line-line deinterlacer with partial use of temporal information. Figure 6-1 shows a typical interlaced video after up-sampling. In here $F(i)$ represents the i^{th} frame and $F(i).j$ represents the i^{th} frame j^{th} row of data. Assuming that a video has $2n-1$ frames of video (or a windowed selection of $2n-1$ frames of video), the video can be re-arranged as shown in Figure 6-2.

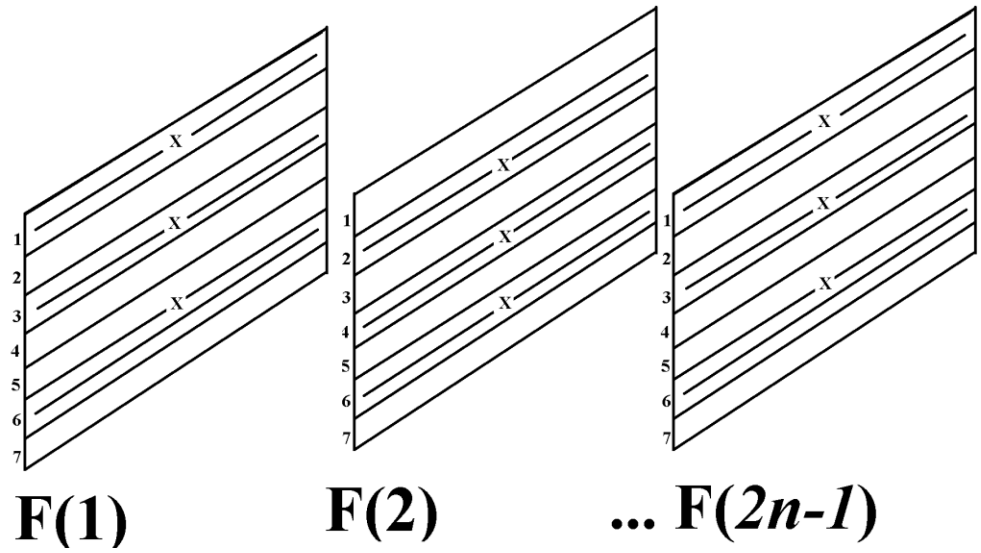


Figure 6-1: A Typical interlaced video when up-sampled but not interpolated yet. Lines marked '*X*' are those that contain data and blank lines are those that are to be deinterlaced.

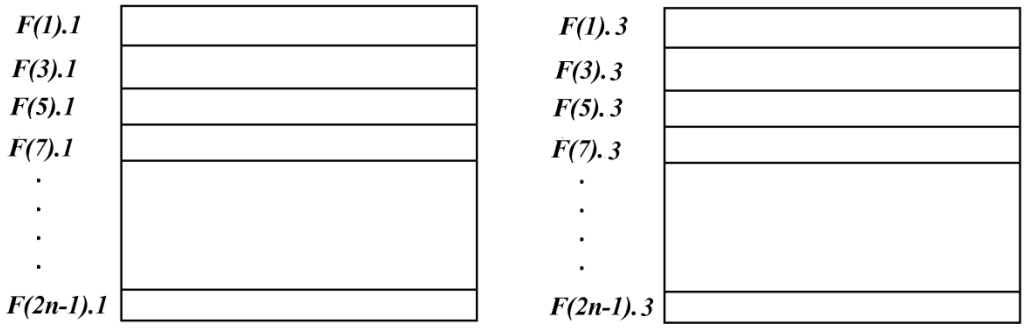


Figure 6-2: Re-arrangement of video to interlace the 2^{nd} line of video. All the lines contain data.

The video is re-arranged in such a way that the entire first and the third lines of the video are collected into intermediary frames. Now using 2DCGI that was discussed in section 3.2, a frame is arrived at, and that represents all the second *spatially-deinterlaced* lines of the video. It can be noted that using one

2DCGI formulation of the methodology mentioned above, at one *pass*, $(n-1)/2$ lines can be deinterlaced, irrespective of the length of the line. Also, there is a trade-off between the number of frames considered and the limit to which the temporal information is required. More the number of frames in the window more are the temporal information supplied to the algorithm. For the purposes of evaluation, 64 frames are considered per window.

This method is particularly significant because, this is the first method to re-arrange a video in such a way that a 2D interpolator is used to deinterlace line-by-line. This methodology is essentially a spatial deinterlacer with information also from the horizontal-temporal, rather than the vertical-temporal.

6.5 Summary of Spin-off methods

Three additional spin-off methods using the principles of 1DCGI are proposed with influences from other methods in literature.

Chapter 7: RESULTS

The videos are interlaced using the second method of deinterlacing discussed in Chapter 1: . This enables us to maintain the number of frames so that evaluation can be performed easily. The test videos are taken from the ASU trace video library [73]. Experiments are performed for a minimum of 300 frames of videos. Peak signal to noise ratio (PSNR) is used as a metric of comparison to evaluate the performance of each algorithm. PSNR is calculated for each frame of the video and are averaged to get the PSNR of the final video. Mean squared error (MSE) for two monochrome images A and B of size mXn is given by,

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[A(i,j) - B(i,j)]^2 \quad \ldots(7.1)$$

where i, j represent the horizontal and the vertical pixel locations. The PSNR is defined as,

$$PSNR = 20 \cdot log_{10}(MAX_I) - 20 \cdot log_{10}(MSE) \quad \ldots(7.2)$$

where MAX_I is the maximum possible pixel value of the Image. Most of the videos that are considered here are unsigned 8-bit integer images for which the maximum pixel values are usually 255. Table 1 compares the PSNR of the prior arts methods, proposed algorithm and the spin-off methods.

Table 1: PSNR (dB) of the proposed method compared against other prior art methods.

Video	Weave	LA	ELA	STELA	VTF	Proposed
Akiyo	33.302	38.359	36.758	41.237	41.117	**47.301**
Bowing	31.617	36.850	34.617	37.013	40.962	**46.122**
Bridge Far	29.854	32.244	32.135	38.788	33.689	**42.423**
Container	24.436	28.017	27.795	35.479	31.055	**46.417**
Deadline	27.178	30.443	28.519	35.662	33.152	**42.814**
Foreman	28.162	31.519	32.149	31.467	32.202	**36.957**
Galleon	21.411	24.333	23.344	31.609	27.058	**42.048**
Hall Monitor	25.801	29.945	30.435	36.942	32.023	**41.892**
Mother	33.103	36.637	36.016	42.599	38.058	**45.635**
News	28.469	34.202	31.765	36.855	39.088	**44.597**
Students	28.124	31.906	30.994	37.086	33.436	**45.173**
Paris	23.541	26.717	25.370	30.943	28.934	**33.799**
Sign Irene	32.667	36.468	36.268	36.181	36.401	**40.108**

The methods 1DCGI, 1DCGI+TF, ST-1DCGI and 2DCGI discussed in Table 2 are spin-off methods. It can be noticed that 1DCGI as a purely spatial method outperforms ELA. The reason for this is because while ELA can interpolate along only 3 discrete angles, 1DCGI can interpolate along virtually infinite angles. It can also be noticed that 1DCGI+TF performs better than STELA and VTF. This is a fair comparison in the sense that all the methods involved in the comparison are spatio-temporal edge-directed methods.

Table 2: PSNR (dB) of the spin-off methods compared against other prior art methods.

Video	1DCGI	1DCGI+TF	ST-1DCGI
Akiyo	38.720	46.024	45.452
Bowing	36.228	42.881	42.232
Bridge Far	32.225	39.138	38.324
Container	28.769	46.278	44.073
Deadline	30.603	42.954	41.564
Foreman	34.539	35.709	34.785
Galleon	24.290	31.780	30.760
Hall Monitor	31.566	37.948	36.963
Mother	36.690	44.351	43.898
News	34.068	40.053	40.746
Students	32.323	38.479	38.224
Paris	26.764	32.114	32.254
Sign Irene	37.462	37.489	38.129

CAVTF [51] was proposed in 2011 and is currently the state-of-the-art in deinterlacing. CAVTF as discussed earlier is a parametric approach that can't be re-implemented and its results re-calculated. To facilitate comparisons to such published results, a statistical relevance factor r is introduced. If MSE_{VTF} is the mean squared error from VTF and MSE_{new} is the mean squared error of a new algorithm, then the statistical relevance r is

$$r = 100 * \left[1 - \frac{MSE_{VTF}}{MSE_{new}}\right]. \quad \ldots(7.3)$$

Comparison of the proposed approach against CAVTF in terms of r is given in Table 3.

Table 3: Comparison of the Proposed Algorithm against CAVTF in terms of Statistical relevance.

Video	CAVTF	Proposed Algorithm
Akiyo	70.620	**74.941**
Container	93.200	**97.090**
Foreman	41.920	**66.547**
Hall Monitor	76.770	**89.694**
Mother	57.040	**82.529**

The proposed deinterlacing system outperforms all the methods discussed. This shows that the method switching algorithm implemented with 1DCGI is as predicted, a better method for deinterlacing than the prior arts methods. Advantages of the proposed algorithm are particularly noticeable in videos like 'mother', where there are distinct salient and background regions as indicated in Figure 5-1. 1DCGI interpolates better than the other temporal method along the edges in the salient regions, while the background regions are estimated using either VTF or temporal average (depending on temporal motion). It is clear that this selection process improves the performance of 1DCGI alone as a deinterlacer and makes the overall approach better than the prior art methods in both the PSNR and visual sense. From Figure 7-1 it can be seen that the proposed approach maintains the quality in the non-moving background in particular.

Summary of Results

The deinterlacing methods are evaluated by testing with standard videos from the Trace library. These methods are compared with PSNR and the proposed system is also compared against CAVTF using the statistical relevance method.

a) Original b) ELA

c) STELA d) VTF

a) 1DCGI b) Proposed

Figure 7-1: Visual comparison among the deinterlacing algorithms.

Chapter 8: CONCLUSIONS AND FUTURE SCOPE

1DCGI is a 1D optical flow analogue that is easily adaptable to the deinterlacing problem. 1DCGI is a purely spatial deinterlacer and can't solve the deinterlacing problem by itself. To support 1DCGI, a method switching technique is proposed that chooses adaptively between Temporal Weave and VTF based on a saliency map and a difference frame. The temporal weave algorithm is implemented on static regions of the video and 1DCGI is implemented on salient regions of the video and on the other regions VTF is performed.

The proposed algorithm performs better than the state-of-the-art both in the PSNR sense and in the visual sense. Among the spin-off methods, 1DCGI outperforms its purely spatial counterpart ELA while, 1DCGI+TF and ST-1DCGI outperform its spatio-temporal counterparts VTF and STELA.

Recently 1DCGI is made faster by the implementation of Segment adaptive gradient angle interpolation (SAGA) [71]. This can be adapted directly for the deinterlacing problem. This would potentially increase the computational performance of the algorithm and make it faster. Further research can be undertaken to perform one symmetric directional estimate rather than two uni directional estimates. This can be directly targeted to suit the deinterlacing application.

The 1DCGI as described in this form can also be applied to two lines temporally. If used along with two lines spatially, this will provide a bi-linear

structure to 1DCGI, wherein four uni-directional lines can be arrived at and averaged. This would truly make 1DCGI a spatio-temporal approach.

REFERENCES

[1] E. Engstorm, "A study of television Image Characteristics. Part Two. Determination of Frame Frequency for Television in terms of Flicker Characteristics.," in *Proceedings of the IRE*, 1935.

[2] National Television System Committee (1951-1953), "Report and reports of Panel No. 11, 11-A, 12-19, with some supplementary references cited in the reports, and the petition for adoption of transmission standards for color television before the Federal Communications commission.," Library of Congress Online Catalog., 1953.

[3] International Telecommnications Union, "Recommendation ITU-R BT.470-6," International Telecommunications Union, 1998.

[4] S. Pigeon and P. Guillotel, "Advantages and Drawbacks of Interlaced and Progressive scanning formats," HAMLET report, 1996.

[5] G. Haan and E. Bellers, "Deinterlacing-an overview," *Proceedings of the IEEE,* vol. 86, pp. 1839-1857, September 1998.

[6] Microsoft, "Broadcast-enabled computer hardware requirements," in *WinHec'97*, 1997.

[7] L. Thrope and T. Hanabusa, "If progressive scanning is so good, how bad is interlace?," *SMPTE journal,* pp. 972-986, December 1990.

[8] A. Van den Enden and N. Verhoeckx, Discrete-time Signal Processing, Prentice Hall, 1989, pp. 223-.

[9] G. Tonge, "Television motion potrayal," in *Les Assises des Jeunes Chercheurs*, Reennes(Fr.), 1985.

[10] A. Tekalp, Digital Video Processing, Prentice hall, 1995, pp. 250-..

[11] M. Weston, Interpolating lines of video signals, US-patent 4,789,893, December 1988.

[12] P. Guillotel and G. D'Agostino, "Towards the use of a progressive trasmission format," in *International workshop on HDTV and the evolution of television*, Taipei, 1995.

[13] P. Anandan, J. Bergen, K. Hanna and R. Hingorani, "Hierarchial model-based motion estimation," *Motion analysis and Image sequence processing,* 1993.

[14] J. Aggarwal and N. Nandhakumar , "On the computation of motion from sequences of images - a review," *Proceedings of the IEEE,* vol. 76, pp. 917-

935, August 1988.

[15] J. Gibson, "The perception of the visual world," 1950.

[16] A. Singh, "Optical flow computation: A unified perspective," *IEEE computer society press,* 1991..

[17] J. Shin, S. Kim, S. Kang, S. Lee, J. Paik , B. Abidi and M. Abidi, "Optical flow-based real-time object tracking using non-prior training active feature model," *Real-Time Imaging,* vol. 11, pp. 204-218, 2005.

[18] J. Chang, W. F. Hu, M. Cheng and B. Chang, "Digital image translational and rotational motion stabilization using optical flow technique," *IEEE Transactions on Consumer Electronics,* vol. 48, pp. 108-115, 2002.

[19] Y. Yacoob and L. Davis, "Recognizing human facial expressions from long image image sequences using optical flow," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 18, pp. 636-642, 1996.

[20] Y. Wang , J. Osternmann and Y. Zhang, Video processing & communication, Prentice Hall, 2002.

[21] J. W. Woods, Multidimensional Signal, Image and Video Processing & Coding., Academic Press, 2006.

[22] A. M. Teklap, Digital Video Processing, Prentice Hall, 1995.

[23] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Imaging Understanding Workshop*, 1981.

[24] B. Horn and B. Schunck, "Determing optical flow," *Artificial Intelligence,* pp. 185-203, 1981.

[25] G. W. Humpherys and V. Bruce, "Visual cognition," *Computational, experimental and neuropsychological perspectives,* 1989.

[26] J. Barron, D. Fleet and S. Beauchemin, "Performance of optical flow techniques," *International journal computer vision,* pp. 43-77, 1994.

[27] S. Beauchemin and J. Barron, "The computation of optical flow," *ACM computing surveys,* vol. 27, pp. 433-466, September 1995.

[28] B. Glocker, N. Komodakis, N. Tziritas, N. Navab and N. Paragios, "Dense image registration through MRFs and efficient linear programming," *Medical analysis journal,* 2008.

[29] J. Shi and C. Tomasi, "Good features to track," in *IEEE computer society conference on Computer vision and pattern recognition*, 1994.

[30] B. Horn, "Robot Vision," *Cambridge: MIT Press,* pp. 20-292, 1968.

[31] J. Watkinson, MPEG Handbook, Focal Press, 2001.

[32] D. H. Frakes, L. P. Desai, K. Pekkan, H. D. Kitajima, K. Sundareswaran, A. P. Yoganathan and M. J. Smith, "A new Method for registration-based medical image interpolation," *IEEE Transactions on medical imaging,* vol. 27, pp. 370-377, March 2008.

[33] D. H. Frakes, C. P. Conrad, T. M. Healy, J. W. Monaco, M. Fogel, S. Sharma, M. Smith and A. P. Yoganathan, "Application of an adaptive control grid inteprolation technique to morphological vascular reconstruction," *IEEE Transactions on biomedical engineering,* vol. 50, pp. 197-206, 2003.

[34] D. H. Frakes, J. Monaco and M. Smith, "Suppression of atmospheric turbulence in video using an adaptive control grid interpolation approach," in *IEEE International conference on Acoustics, Speech and Signal Processing (ICASSP '01)*, 2001.

[35] C. M. Zwart and D. H. Frakes, "Biaxial control grid interpolation: Reducing isophoto preservation to optical flow," in *IEEE Digital signal processing workshop*, 2011.

[36] C. M. Zwart and D. H. Frakes, "One-dimensional control grid interpolation-based demosaicing and color image interpolation," in *IST&T/SPIE Electronic imaging, Computational imaging X*, 2012.

[37] T. Doyle and M. Looymans, "Proressive scan conversion using edge information," *Signal processing of HDTV II,* pp. 711-721, 1990.

[38] H. Y. Lee, J. W. Park, T. M. Bae, S. U. Choi and Y. H. Ha, "Adative scan rate up-conversion basd on human visual characteristics," *IEEE Transactions on Consumer Electronics,* vol. 4, pp. 999-1006, 2000.

[39] C. J. Kuo, C. Liao and C. Lin, "Adaptive interpolation technique for scanning rate conversion," *IEEE Transactions on circuit systems and video technology,* vol. 6, pp. 317-321, 1996.

[40] X. Li and M. Orchard, "New edge-directed interpolation," *IEEE Transactions on image processing,* vol. 10, pp. 1521-1527, 2001.

[41] S. Lin, Y. Chang and L. Chen, "Motion adaptive interpolation with horizontal motion detection for deinterlacing," *IEEE Transactions on consumer electronics,* pp. 1256-1265, 2003.

[42] A. J. Patti, M. I. Sezan and A. M. Tekalp, "Robust methods for high quality stills from interlaced video in the presence of dominant motion," *IEEE*

Transactions on circuits and systems video technology, pp. 328-342, 1997.

[43] T. Chong, O. C. Au, W. Chau and T. Chan, "A content adaptive deinterlacing algorithm," in *IEEE International symposium on circuit and systems*, 2005.

[44] D. Han, C. Shin, S. Choi and J. Park, "A motion adaptive 3-D deinterlacing algorithm based on the brightness profile pattern difference," *IEEE transactions consumer electronics,* pp. 690-697, 1999.

[45] S. Kwon, K. Seo, J. Kim and Y. Kim, "A motion-adaptive deinterlacing method," *IEEE Transactions on consumer electronics,* pp. 145-150, 1992.

[46] L. YU, J. Li, Y. Zhang and Y. Shen, "Motion adaptive deinterlacing with accurate motion detection and anti-aliasing interpolation filter," *IEEE Transactions on consumer electronics,* pp. 712-717, 2006.

[47] J. Salonen and S. Kalli, "Edge adaptive interpolation for scanning rate conversion," *Signal processing for HDTV IV,* pp. 757-764, 1993.

[48] S. Carrato, G. Ramponi and S. Marsi, "A simple edge-sensitive image interpolation filter," in *International conference on image processing (ICIP)*, 1996.

[49] T. Doyle, "Interlaced to sequential conversion for EDTV applications," in *2nd International workshop on signal processing of HDTV*, 1988.

[50] H. S. Oh, Y. Kim, Y. Y. Jung, A. W. Morales and S. J. Ko, "Spatio-temporal edge-based median filtering for deinterlacing," in *International conference on consumer electronics*, 2000.

[51] K. Lee and C. Lee, "High quality deinterlacing using content adaptive vertical temporal filtering," *IEEE Transactions on consumer electronics,* pp. 2469-2474, 2011.

[52] G. G. Lee, H. Y. Lin, M. J. Wang, R. L. Lai, C. W. Jhuo and B. H. Chen, "Spatial-temporal content-adaptive deinterlacing algorithm.," *IET Image processing ,* vol. 2, pp. 323-336, 2008.

[53] S.-M. Hong, S.-J. Park, J. Jang and J. Jeong, "Method switching algorithm for intra-field deinterlacing," in *IEEE 15th International symposium on consumer electronics*, 2011.

[54] T. Lu, Z. Yuan, Y. Huang, D. Wu and H. Yu, "Video retargetting with nonlinear spatial-temporal saliency fusion," in *International conference on image processing (ICIP)*, 2010.

[55] 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, pp. 1254-1259, 1998.

[56] F. Liu and M. Gleicher, "Video retargeting: automatic pan and scan," in *14th annual ACM international conference on multimedia*, 2006.

[57] S. Avidan and A. Shamir, "Seam carving for content-aware image resizing," *ACM Transactions on graphics (TOG),* vol. 26, p. 10, 2007.

[58] L. Taycher, J. I. Fisher and T. Darrell, "Combining object and feature dynamics in probabilistic tracking," *Computer vision and image understanding,* vol. 108, pp. 243-260, 2007.

[59] R. Fergus, P. Perona and A. Zisserman, "Object class recognition by unsupervised scale-invarient learning," in *Computer vision and pattern recognition (CVPR)*, 2003.

[60] A. Triesman and G. Gelade, "A feature -integration theory of attention," *Cognitive psychology,* vol. 12, pp. 97-136, 1980.

[61] J. Wolfe, "A revised model of guided search," *Psychonomic Bulletin& review,* vol. 1, pp. 202-238, 1994.

[62] R. Rensink, "Seeing sensing and scrutinizing," *Vision research,* vol. 40, pp. 1469-1487, 2000.

[63] R. Rensink and J. Enns, "Preemption effects in visual search: Evidence for low-level grouping," *Psychology Review,* vol. 102(1), pp. 101-130, 1995.

[64] R. Rensink, J. ORegan and J. Clark, "To see or not to see: The need for attention to percieve changes in scenes.," *Psyhological science,* vol. 8(5), pp. 368-373, 1997.

[65] L. Itti and C. Koch, "A saliency-based search mechanism for overt and covert shifts of visual attention," *Vision research,* Vols. 40(10-12), pp. 1489-1506, 2000.

[66] L. Itti and C. Koch, "Computational modelling of visual attention," *Nature reviews neuroscience,* vol. 2(3), pp. 194-203, 2001.

[67] D. Walther, L. Itti, M. Riesnhuber, T. Poggio and C. Koch, "Attention selection for object recognition-a gentle way," *Lecture notes on computer science,* vol. 2525(1), pp. 472-479, 2002.

[68] X. Hou and L. Zhang, "Saliency Detection: A Spectral residue appraoch," in *Computer vision and pattern analysis (CVPR)*, 2007.

[69] C. Guo, Q. Ma and L. Zhang, "Spatio-temporal saliency detection using phase spectrum of quaternion fourier transform," in *IEEE conference on*

computer vision and pattern analysis, 2008.

[70] S. Sangwine and T. Ell, "Hypercomplex Fourier transforms of color images," *IEEE Transactions on image processing,* pp. 22-35, 2001.

[71] C. Zwart and D. H. Frakes, "Soft adaptive gradient angle interpolation of grayscale images," in *IEEE International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2012.

[72] R. Venkatesan, C. Zwart and D. Frakes, "Video deinterlacing with control grid interpolation," in *IEEE International conference on Image Processing (ICIP)*, 2012.

[73] P. Seeling , F. Fitzek and M. Reisslein, Video traces for network performance evaluation, Springer, 2007.