An MRF-Based Approach to Image and Video Resolution Enhancement

AN MRF-BASED APPROACH TO IMAGE AND VIDEO RESOLUTION ENHANCEMENT

BY

FARHANG VEDADI, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2012) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	An MRF-Based Approach to Image and Video Resolution Enhancement
AUTHOR:	Farhang Vedadi B.Sc., (Electrical Engineering) Sharif University of Technology, Tehran, Iran
SUPERVISOR:	Dr. Shahram Shirani
NUMBER OF PAGES:	xii, 106

To my beloved parents, Abolfazl and Farideh, for their unconditional love and support in my life.

Abstract

The main part of this thesis is concerned with detailed explanation of a newly proposed Markov random field-based de-interlacing algorithm. Previous works, assume a first or higher-order Markovian spatial inter-dependency between the pixel intensity values. In accord with the specific interpolation problem in hand, they try to approximate the Markov random field parameters using available original pixels. Then using the approximate model, they define an objective function such as energy function of the MRF to be optimized. The efficiency and accuracy of the optimization step is as important as the effectiveness of definition of the cost (objective function) as well as the MRF model.

The major concept that distinguishes the newly proposed algorithm with the aforementioned MRF-based models is the definition of the MRF not over the intensity domain but over interpolator (interpolation method) domain. Unlike previous MRF-based models which try to estimate a two-dimensional array of pixel values, this new method estimates an MRF of interpolation function (interpolators) associated with the 2-D array of pixel intensity values.

With some modifications, one can utilize the proposed model in different related fields such as image and video up-conversion, view interpolation and frame-rate up-conversion. To prove this potential of the proposed MRF-based model, we extend

it to an image up-scaling algorithm. This algorithm uses a simplified version of the proposed MRF-based model for the purpose of image up-scaling by a factor of two in each spatial direction. Simulation results prove that the proposed model obtains competing performance results when applied in the two interpolation problems of video de-interlacing and image up-scaling.

Acknowledgements

I would like to express my special thanks to my supervisor, Prof. Shahram Shirani, for his suggestions and support during the last two years. I would like to appreciate his kind and supportive personality and mention that starting to work under his supervision was definitely a great milestone in my academic career. My deepest gratitude and respect goes to my wonderful parents who have always been the most important inspirations in my life. I would also like to appreciate my dear friends who have been always kind and supportive to me. I enjoyed every second of working with my friends and colleagues at multimedia signal processing laboratory.

Contents

List of Figures

Chapter 1

Introduction

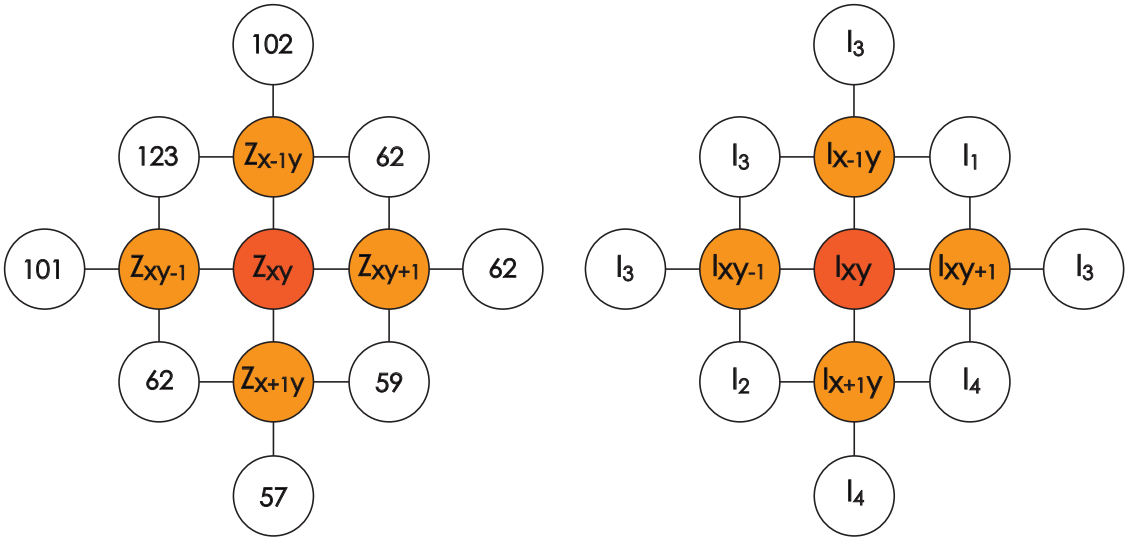
Image/Video interpolation is one of the classical but still active subfields of image/video processing. In general, 2-D/3-D signal interpolation (reconstruction) has divers applications such as image up-scaling (resolution up-conversion), video up-conversion and frame-rate up-conversion. During the last decade, major contributions have been made in each of these fields, some of them applying rigorous mathematical frameworks to model and exploit the natural characteristics existing in natural images and videos.

Video de-interlacing is one specific application of 3-D signal reconstruction (interpolation). Although originally developed for analog television industry, it is still widely used by high definition (HD) broadcasters. The major part of this thesis, indeed, will focus on a newly proposed de-interlacing algorithm with roots in Markov random fields (MRF) modeling (appendix B), Markov chain models and sequence estimation techniques. MRF and Markov chains have been extensively augmented for different applications of signal processing. They are widely used for image processing applications such as image segmentation, decision making, robot vision etc.

In the field of image/video interpolation, MRF and Markov chains can play a key role in modeling the characteristics found in natural images. They can be greatly utilized to model the spatial and/or temporal inter-dependencies between the image/video samples.

In video de-interlacing in particular, some recent pioneering works have been proposed recently that aim to use MRF-based models to characterize the spatial smoothness present in natural images. For this, each image (or a frame in a sequence of frames) can be considered as an MRF or sets of Markov chains of intensity values. Then using reasonable approximations for the MRF parameters, we are left with an optimization problem to get the best solution. Note that MRF modeling and afterward the optimization method chosen, are tightly bound together. In other words, an MRF model may be followed by an optimization tool which is not necessarily feasible and efficient for some other MRF model.

Consider a digital image as an MRF. By insight we have from natural images, neighboring pixel values in an image are highly correlated. This correlation is much less around the object boundaries. Basically, different methods try to define a reasonable potential function for the cliques and then a feasible optimization method to optimize an objective function such as the probability of the MRF. In case of interpolation problems such as image up-scaling and video de-interlacing, the potential functions used in the literature in general use an ℓ_2-norm-based function that takes into account the intensity difference of a pixel with its neighbors. Then the optimization problem is in general to find the MRF, $\mathbf{z}$, that maximizes the $\Pr(\mathbf{z})$ or equivalently the summation of the potentials over the cliques. The results obtained by such algorithms have been proven to be promising.



(a) Intensity-domain MRF (b) Interpolator-domain MRF

Figure 1.1: The proposed strategy is to switch from pixel-intensity domain to interpolator domain and do the estimation in the latter domain (a) Intensity domain MRF (b) Interpolator-domain MRF.

The aforementioned MRF-based model for the interpolation problem shows a great performance, however still suffers from a few limitations. When the MRF is defined over the gray-level intensity domain, basically the size of the multi-dimensional search space of the optimization problem is very large. For instance, each site of the MRF can have 256 (for 8-bit precision) discrete values which can be a larger number for higher precisions. The larger the search space for the optimization procedure, the less robust is the algorithm to find a good solution. Here is where our newly proposed algorithm comes in. Let us explain the algorithm intuitively. A more detailed mathematical representation of the algorithm will be left for the following chapters.

As explained before, an MRF in intensity domain, greatly captures the inter-dependencies between pixels i.e., the smoothness in gray-level values. Now a logical

question is that are we able to switch from this domain (gray value) to some other domain which can greatly reduce the size of the search space on one hand and maintain (and even improve) the performance of the algorithm on the other hand? We will try to explain the answer to this question which will clarify the scope of this thesis.

In traditional models if $\mathbf{z}$ is an MRF then the site (x, y) takes a gray-scaled value. In fact, the $\mathbf{z}$ is a 2-D arrangement of gray-valued random variables $\mathbf{z}_{xy}$ such as depicted in figure 1.1(a). Consider a function $\mathbf{I}_{xy}$ corresponding to the site (x, y). The input of the $\mathbf{I}_{xy}$ is the MRF $\mathbf{z}$ and the output is a gray-value for the site (x, y). In other words, the $\mathbf{I}_{xy}$ is the interpolation function assigned to the site (x, y). Now consider the 2-D arrangement of these interpolator functions assigned to their corresponding sites within the intensity-domain MRF as depicted in figure 1.1(b). In simple words, in our proposed algorithm (which will be well treated in the following chapters), we try to find this 2-D arrangement of interpolator functions or simply interpolators.

Assuming an MRF over the pixel intensity values as discussed before, implies that the value of a pixel at site (x, y) i.e., $\mathbf{z}_{xy}$ is highly correlated with its neighbors. In other words, having its neighbors in hand is like having the entire MRF (in accord with the order of the MRF). Therefore we define a reasonable potential function that considers the differences between the $\mathbf{z}_{xy}$ and a few of its neighbors.

Equivalently, we propose that the probability of the $\mathbf{I}_{xy}$ given all the other $\mathbf{I}_{xy}$ where $(x, y) \neq (x', y')$ is highly dependent to its neighboring interpolators given. This implies that if $\mathbf{I}_{xy}$ is well suited for interpolating the pixel at (x, y), there is a high chance that it works well for some of its neighbors as well. In other words, we assume an MRF over the 2-D interpolator array with the following property:

$$\Pr\big(\mathbf{I}_{xy}|\mathbf{I}_{x'y'},(xy)\neq(x'y')\big)=\Pr\big(\mathbf{I}_{xy}|\mathbf{I}_{x'y'},\mathbf{I}_{xy}\in N_{xy}\big). \tag{1.1}$$

This way the MRF of gray-level intensity values, $\mathbf{z}$, have been associated with an MRF of interpolation functions or interpolators, $\mathbf{I}$. Consider the set of real-values for a pixel intensity. This set is an infinite set of values for $\mathbf{z}_{xy}$. For a gray-scaled $\mathbf{z}_{xy}$, this number is decreased to 256 which is still a large number. Now consider the set of candidate interpolators in the new interpolator-domain MRF, Γ. Clearly this set is also infinite. Therefore one immediate question is that what is the advantage to the new MRF model explained? Note that, theoretically there are infinite number of interpolators that could be defined for a pixel, however, as we will validate in the next chapter, a small set of directional interpolators can greatly be utilized to capture the local characteristics of a neighborhood around a pixel. Assuming that the size of Γ can be maintained sufficiently small using simple directional interpolators, there would be a huge advantage to this interpolator-domain-defined MRF. Moreover in the next chapter, we will introduce a powerful choice for the potential function of the MRF as well. We will show through experiments that this potential function (which we will refer to as the cost function or simply local cost) plays a key role in accurately measuring the correctness of using each of the interpolators in a local pixel position.

In brief, what we do is that we switch from pixel domain to interpolator domain. In the interpolator domain, we solve for the optimum interpolator MRF then having MRF of interpolators, $\mathbf{I}$, we just use $\mathbf{I}_{xy}$ at (x,y) to get the estimated gray value. The parameters of the interpolators MRF are approximated using the gray-level intensity field in hand. In case of interpolation applications such as video de-interlacing or

image up-scaling, to due lost data in intensity-domain MRF, this stage is more vulnerable to errors.

The final step is to estimate the MRF of interpolators. In the literature, they have used a number of optimization techniques to find the optimum MRF in accord with the objective function designed for specific applications. However, these techniques such as simulated annealing (SA), genetic algorithms (GA) and other evolutionary approaches have two major drawbacks. They are usually heavy in terms of time-complexity as they usually involve too many iterations. They also contain parameters (temperature in SA or mutation and crossover rates in GA etc.) that can hardly be adjusted to be consistent for different sequences and still images. In fact two-dimensional property of an MRF makes the global optimization usually infeasible with reasonable complexities. The structure of the total energy or cost of the MRF (as will be discussed in the next chapter) is recursive in terms of pixel positions, however the two-dimensional and casual nature of the MRF makes it practically impossible to solve the MRF estimation problem directly using recursion.

We proposed that the interpolators MRF be partitioned to neighboring one-on-top-of-the-other one-dimensional row-aligned MRFs. There are a few points about this approach to MRF estimation as follows:

1. The solution for the entire frame (interpolator MRF) is obtained via estimating the 1-D MRF of interpolators for each row.

2. 1-D MRF is also called Markov-chain.

3. Clearly the combination of optimal Markov chains is not necessarily the optimal MRF. However, as will be validated through experiments we get great objective and subjective results.

4. For estimation of the 1-D MRFs or Markov chains we propose a MAP-based formulation which then will be solved efficiently using the Viterbi algorithm (dynamic programming).

5. The fact mentioned in item (1) makes a lot of sense. This is because we use interpolators that are imposed in a neighborhood (2-D/3-D) so there will not be much advantage in two-dimensional regularization.

6. Note that (5) would not be much reasonable if we had not proposed to use the interpolator-domain MRF instead of intensity-domain MRF.

7. The Viterbi algorithm is a linear time algorithm that globally optimizes the solution.

8. One another advantage to using one-dimensional MRFs is that the entire scheme is highly potent of being implemented using parallel-implementation-supported devices such as graphical processing units (GPU) or field programmable gate arrays (FPGA).

9. The last but one of the most important contributions of our algorithm is that we further improve it by modifying our strategy toward sequence estimation problem. The general idea of this improvement and the details of the mathematical derivations will be discussed in the next chapter.

Most of the material of this thesis with some modifications and improvements, are adopted from our papers [34], [32] and [33]. In short we consider the following as our contributions:

- We have used 1-D MRF model (Markov chain) so that the algorithm can potentially exploit parallel processing.

- We have used a set of interpolators to define an interpolator-domain MRF instead of intensity-domain MRF. Consequently, both the states and energy function (or equivalently the probability) of the MRF can be defined more efficiently.

- One major contribution is that for the first time a detailed MAP-based formulation is proposed in aforementioned interpolator-domain MRF. Then Viterbi algorithm is utilized for estimating the MRF.

- Moreover we use a novel adaptive cost function to estimate the MRF parameters and probabilities. This adaptive cost (non-local cost) plays a key role in our algorithm and is a major contribution of the proposed algorithm.

- One of the most important contributions is that we propose a superior MAP-based formulation which obtains better results than the early version, at the cost of more computational complexity.

- We solve the latter formulation using Forward-Backward algorithm and prove the superiority of both the latter and early version of the algorithm to many state-of-the-art competing algorithms.

- We finally propose a method for integrating motion compensation (optical flow computation) into the algorithm. The proposed motion compensated versions of the algorithm have better objective and subjective results at the cost of more computational complexity.

Chapter 2

Video De-interlacing

2.1 Introduction and Background

Interlacing is an efficient way of reducing the required bandwidth for the transmission of the video signal. This method of subsampling the video data was originally designed and developed for analogue TV and is still widely used in modern television systems. Nowadays, as progressive-scan devices are widely used, one has to apply techniques to convert the interlaced signal into progressive format in order to display it on an all-progressive-scan device. These techniques, known as de-interlacing, have been extensively developed over the last decade. Fig. 2.2 better illustrates typical interlaced and progressive formats.

Generally de-interlacing techniques use intra and/or inter field data to estimate the missing pixels within a field. Edge-directional algorithms such as those proposed in [9], [26], [17], [14], [15], [27], [36], [28], try to find the direction with the highest correlation to interpolate along that direction. In [26] authors add estimation of the edge patterns to simple directional intensity differences used in conventional

edge-directional averaging methods to finally get more accurate results. In [17], authors improve the conventional edge-dependent interpolation methods by utilizing block-based motion detection. They use direction field and sum of absolute values to estimate an interpolation direction. Motion adaptive algorithms like those developed in [18, 22, 30, 37], examine different parts of a frame to determine whether or not the amount of motion is considerable. The appropriate method of spatial and/or temporal interpolation is then applied to get the best result.

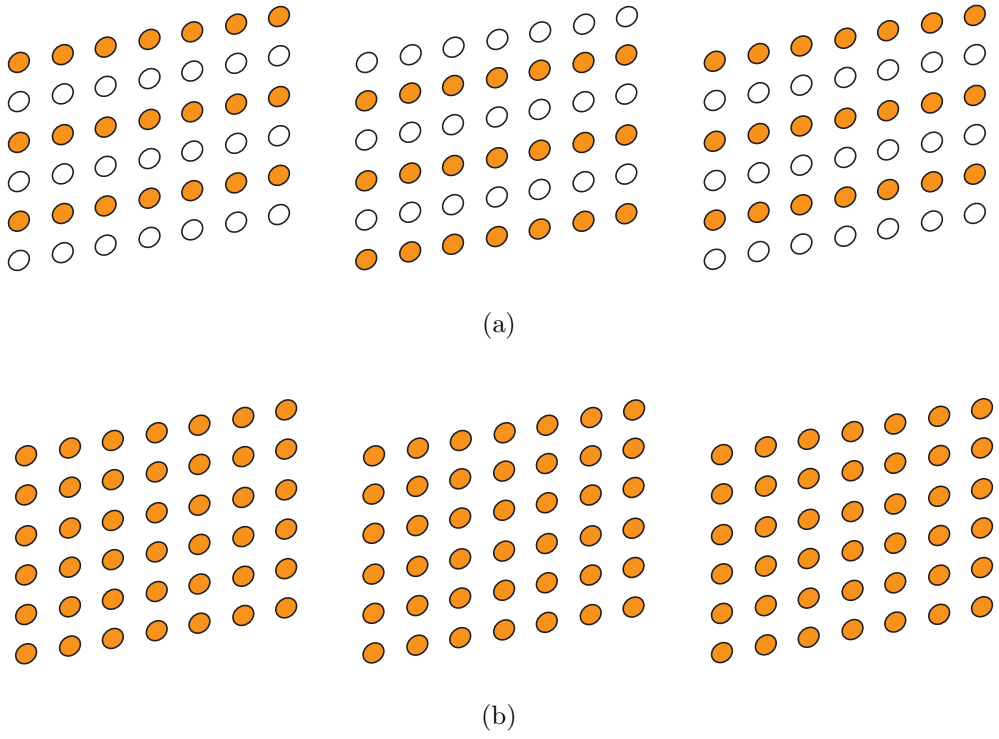


Figure 2.2: De-interlacing is the process of converting an interlaced format video to progressive. This requires interpolating the missing lines in an interlaced format. (a) An interlaced format. (b) A progressive format.

Motion-compensated (MC) de-interlacing algorithms such as [20] ,[4], [11], [25], [19], [7], [5], [10], [8], [13], [6], are another popular group of techniques widely used for the purpose of de-interlacing. [4] and [13] make use of motion estimation and

compensation. They generally use motion reliability measures to examine the validity of estimated motion and switch to spatial intra-field interpolation wherever the estimated motion information is considered unreliable. [6] proposed a de-interlacing method based on modular neural networks. Their algorithm uses the field absolute difference as a measure for amount of motion. Then different neural networks are used selectively. Again, an intra/inter-mode switching is used to overcome the problem of inaccurate estimated motion. Generally, the reliability of motion-compensated algorithms is highly dependent on the correctness of the estimated motion information. In case of de-interlacing problem, motion information has to be extracted from the fields and not the complete frames. Thus, the reliability of the motion information and consequently the performance consistency for these algorithms are decreased to some extent. That is why many MC algorithms proposed, examine the correctness of the estimated motion information before making any decision on whether to apply motion compensation interpolation or switch to some other spatial and/or temporal methods of interpolation. [8] proposed an MRF-based de-interlacing method in which a set of four different interpolation methods were defined as well as a cost for applying an interpolator at a missing pixel. They assumed that a penalty term or a regularization term applies to the total cost of interpolators applied to different missing positions within the interlaced frame. Considering this regularization terms, then they minimize the total cost by dynamic programming.

2.2 A New Approach to Video De-interlacing

We define a set of candidate interpolators and then select an appropriate sequence of interpolators to associate them to the corresponding sequence of missing pixels

in the spirit of [8], however we make some major contributions. We define for the first time a novel non-local cost function that utilizes the structural similarity and intensity differences together. Then we assume a Markov chain model over the interpolator sequence and translate the interpolator sequence estimation to a trellis-based optimization problem in which we find the optimum sequence of interpolators corresponding to the sequence of missing pixels. We propose a new MAP-based formulation that can be solved using reasonable assumptions and with the help of trellis diagrams and Viterbi algorithm. We further propose that the de-interlacing problem, instead of being interpreted as the problem of finding the optimum sequence of interpolators as [8], can be revisited as successive estimations of best interpolator for each position for the entire missing pixels of a row. This approach as we will see, improves the early version of our formulation solved using Viterbi algorithm. Unlike [8] in which the authors set the parameters of their model empirically, we incorporate a novel learning process along with a new cost function that we define for our algorithm so that the parameters of our Markov chain model are extracted in a novel way from the given data i.e., the original pixels. Therefore, our algorithm does not suffer from the inconsistency of the empirically-tuned parameters and benefits from a parameter-free implementation. We obtain better results than many well-known algorithms proposed in this field.

2.3 Theory and Algorithm

In this article, we will use x , y and t to designate the vertical, horizontal and temporal positions respectively. Any pixel position within a frame is denoted by the vector $\vec{r} = (x, y)$. Also we use $f_t(\vec{r})$ or $f_t(x, y)$ to show the video signal intensity at

pixel position $\vec{r}$ at time t. We designate the height and width of a typical complete (progressive) frame by h and w respectively.

Consider for each missing pixel position at a frame, we have an ensemble of candidate interpolation methods or simply interpolators. The ultimate goal is to find the fittest sequence of interpolaters from this set for the purpose of interpolating the corresponding sequence of missing pixels. Candidate interpolators are considered states of a Markov chain. Therefore three remaining steps are formulating the interpolator sequence estimation problem, deriving approximate transition matrix (TM) and finally suggesting an efficient method of solving the estimation problem formulated in the first step. Regarding the formulation step, we propose two different approaches toward sequence estimation which results in two different formulations of the same problem. We will discuss that one of these methods is computationally less intensive with roughly the same objective and subjective results.

Let us denote the set of states for the Markov chain by $S = \{I_i\}_{i=1}^{|S|}$ where I_i is the i^{th} interpolator and $|S|$ is the cardinality of the set S. For each missing pixel we can define a set of neighboring pixels or simply a neighborhood consisting of some of neighboring original pixels of that missing pixel. We refer to this neighborhood as $N(\vec{r}) = \{\vec{n_j}\}_{j=1}^{|N|}$.

2.3.1 Candidate Interpolators : Markov Chain States

To further elaborate on the Markov chain model, here we define the states or interpolators. Clearly one can define any desirable set of interpolators. The limiting factor on the cardinality of S is the time budget which depends primarily on the application. Despite our freedom to choose any set of interpolators, a choice that can handle any

characteristics of a typical pixel in an image is preferred. One choice for the set S can be the ensemble of some simple temporal and spatial interpolators. In this thesis we make use of the following temporal interpolators:

$$
\begin{aligned}
I_1(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t-1}(\vec{r}) + f_{t+1}(\vec{r})\big) \\
I_2(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t-1}(\vec{r} + \vec{u}_y) + f_{t+1}(\vec{r} - \vec{u}_y)\big) \\
I_3(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t-1}(\vec{r} - \vec{u}_y) + f_{t+1}(\vec{r} + \vec{u}_y)\big) \\
I_4(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t}(\vec{r} - \vec{u}_x) + f_{t}(\vec{r} + \vec{u}_x)\big) \\
I_5(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t}(\vec{r} - \vec{u}_y + \vec{u}_x) + f_{t}(\vec{r} + \vec{u}_y - \vec{u}_x)\big) \\
I_6(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t}(\vec{r} - \vec{u}_y - \vec{u}_x) + f_{t}(\vec{r} + \vec{u}_y + \vec{u}_x)\big).
\end{aligned}
\tag{2.2}
$$

$\vec{u}_x$ and $\vec{u}_y$ are unit vectors in vertical and horizontal directions, respectively. As we will see in the following sections, it is important in our algorithm to measure the fitness of an interpolator in a missing pixel position. Many conventional edge-guided algorithms seek to find the fitness of a directional interpolator by simply computing the pixel intensity differences along that direction. Despite that there are many different ways to find the direction along which the correlation is maximum, most of these technics finally choose one of the directional interpolators and impose it immediately at that position. Then they move on to the next missing pixel position to repeat this procedure. However as mentioned earlier, consider a Markov chain over the sequence of interpolators corresponding to the sequence of missing pixels of a row. Consequently, we do not make a hard decision at each missing pixel. Instead we estimate the entire sequence of interpolators which is best suitable to be used for

a entire missing row, given the neighboring original pixels. For this, we try to find a reasonable way to approximate the cost of an interpolator (state) in the set S at a missing pixel position. If we have the value of a pixel, then it makes a lot of sense to claim that the difference between the original pixel value and the interpolated value is a measure of how well an interpolator is in that position. Consider the missing pixels in an interlaced frame. For this missing positions we do not have the original pixels, therefore, we can not directly compute the cost of a member of the set S at that missing pixel. To overcome this, we define the low resolution version of it $S_{LR}(\vec{r}) = \{I_{LR_i}\}_{i=1}^{|S|}$. $S_{LR}(\vec{r})$. In section 2.3.3 we will discuss why defining low-resolution version of the aforementioned interpolators can help us to define a measure how well an interpolator performs given a neighborhood of original pixels around a missing of interest. $S_{LR}(\vec{r})$ contains the dilated versions of the states (interpolators) in the set S by a factor of two in each direction:

$$
\begin{aligned}
I_{LR_1}(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t-2}(\vec{r}) + f_{t+2}(\vec{r})\big) \\
I_{LR_2}(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t-2}(\vec{r} + 2\vec{u}_y) + f_{t+2}(\vec{r} - 2\vec{u}_y)\big) \\
I_{LR_3}(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t-2}(\vec{r} - 2\vec{u}_y) + f_{t+2}(\vec{r} + 2\vec{u}_y)\big) \\
I_{LR_4}(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t}(\vec{r} - 2\vec{u}_x) + f_{t}(\vec{r} + 2\vec{u}_x)\big) \\
I_{LR_5}(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t}(\vec{r} - 2\vec{u}_y + 2\vec{u}_x) + f_{t}(\vec{r} + 2\vec{u}_y - 2\vec{u}_x)\big) \\
I_{LR_6}(\vec{r}) &= \frac{1}{2} \cdot \big(f_{t}(\vec{r} - 2\vec{u}_y - 2\vec{u}_x) + f_{t}(\vec{r} + 2\vec{u}_y + 2\vec{u}_x)\big).
\end{aligned}
\tag{2.3}
$$

To better illustrate the aforementioned interpolators and their low-resolution versions, Fig. 2.3 shows the pixel used by (a) I_1, I_2 and I_3 (b) I_{LR_1}, I_{LR_2} and I_{LR_3}.

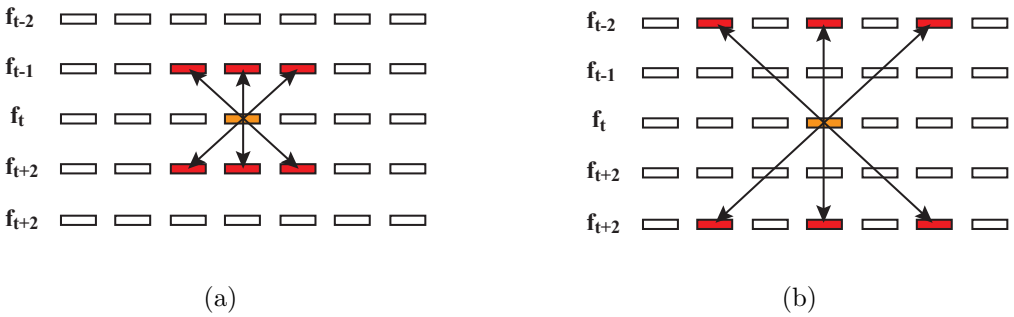


Figure 2.3: We use three different temporal interpolators. (a) Pixels used by I_1, I_2 and I_3 and (b) their dilated/low-resolution versions : I_{LR_1}, I_{LR_2} and I_{LR_3} to interpolate a typical pixel.

Also Fig. 2.4 illustrates the pixel used by (a) I_1, I_2 and I_3 (b) I_{LR_1}, I_{LR_2} and I_{LR_3} to interpolate a typical pixel of interest. In the following sections we will explain the importance of the set S_{LR} as this set plays a key role when we define the cost of applying a sequence of interpolators to the corresponding sequence of missing pixel positions.

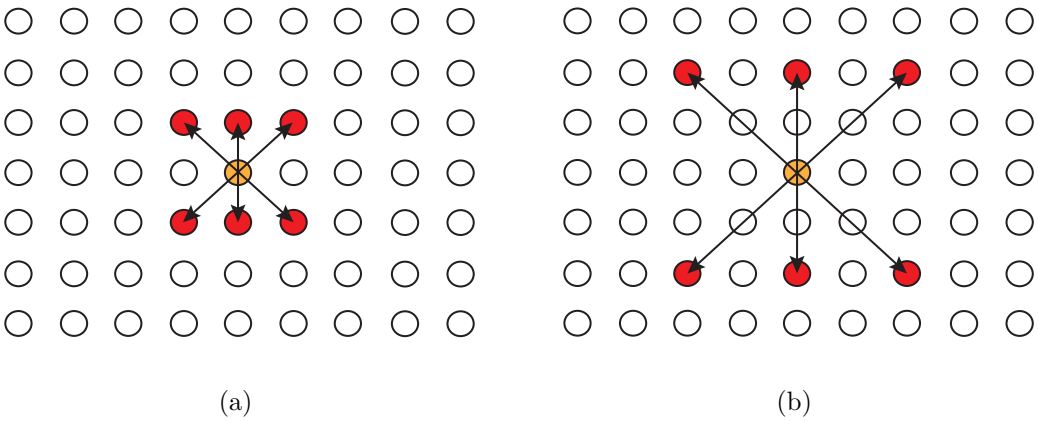


Figure 2.4: We use three different spatial interpolators. (a) Pixels used by I_4, I_5 and I_6 and (b) their dilated/low-resolution versions : I_{LR_4}, I_{LR_5} and I_{LR_6} to interpolate a typical pixel.

2.3.2 Formulation

Consider that any interpolator at each missing pixel position is a state for the Markov chain model discussed before. Then the de-interlacing problem is translated to the problem of estimating the best sequence of states given the original pixels of the frame. To better elaborate on estimation of the sequence of the interpolators corresponding to a sequence of missing pixels, we first consider how to scan the missing pixels of a frame. We process missing rows once at a time to estimate its corresponding sequence of interpolators. The result is a sequence of interpolators $\bar{\mathbf{I}}$ of length w corresponding to a missing row. In our algorithm we scan missing pixels of a row from left to right, estimate the sequence of interpolators and finally interpolate the missing pixels in accord with the estimated sequence of interpolators. This procedure is repeated for all missing rows within a frame. Clearly, these sequence estimations can be done in parallel, Therefore, here we will concentrate on the formulation for one of these sequence estimation.

We previously defined for each missing pixel a so-called neighborhood consisting of some original pixels around a missing pixel of interest. We denote by $\overline{\mathbf{N}}$ the sequence of these neighborhoods corresponding to the sequence of missing pixels of a missing row. We make use of subscripts and superscripts to show the beginning and end of the sequences. For instance $\overline{\mathbf{N}}_{y_1}^{y_2}$ is the sequence of neighborhoods from y_1^{th} to y_2^{th} position in a row i.e., $\{N(y_1), N(y_1+1), ..., N(y_2)\}$. Also a single subscript shows the value of a sequence at a specific position, hence, $\overline{\mathbf{N}}_{y_1}$ is the y_1^{th} value of the sequence $\overline{\mathbf{N}}$. Now let us focus on the estimation of the sequence for one typical missing row. Consider $\overline{\mathbf{N}}$ which is the sequence of neighborhoods of the pixels in the missing row as the given data. Also consider that the to-be-estimated sequence of interpolators

corresponding to the positions in this row i.e., $\bar{\mathbf{I}}$. In our algorithm we consider that the best estimation of the sequence of interpolators for the missing pixels of a row, $\bar{\mathbf{I}}_{opt}$, is the sequence of interpolators $\bar{\mathbf{I}}$ which maximizes the probability $\Pr\big(\bar{\mathbf{I}}|\overline{\mathbf{N}}\big)$ consequently $\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big)$ or:

$$\bar{\mathbf{I}}_{opt} = \underset{\bar{\mathbf{I}} \in S^w}{\operatorname{argmax}} \Big\{ \Pr\big(\bar{\mathbf{I}}|\overline{\mathbf{N}}\big) \Big\} = \underset{\bar{\mathbf{I}} \in S^w}{\operatorname{argmax}} \Big\{ \Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big) \Big\} \tag{2.4}$$

The next step is to rearrange the above equation to a proper form for solving the optimization problem with the help of some reasonable assumptions. Assuming first order Markov model for the interpolators and using Bay's rule we can show that:

$$\Pr\big(\bar{\mathbf{I}}\big) = \Pr\big(\bar{\mathbf{I}}_0\big) \cdot \prod_{y=1}^{w} \Pr(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}). \tag{2.5}$$

Here we assume that successive observations of the neighborhoods are independent. Therefore, the conditional probabilities are independent as well and we can simplify the formulation to the following form:

$$\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) = \prod_{y=1}^{w} \Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big). \tag{2.6}$$

We substitute (2.5) and (2.6) in (2.4). Then for $\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big)$ we have:

$$\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big) = \prod_{y=1}^{w} \Pr\big(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}\big) \cdot \prod_{y=1}^{w} \Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big). \tag{2.7}$$

We have to choose the $\bar{\mathbf{I}}_0$ arbitrarily since it does not have any corresponding actual pixel. We arbitrarily set $\bar{\mathbf{I}}_0 = I_1$. Taking the logarithm on both sides of (2.7) we have:

$$\log\Big(\Pr\big(\overline{\mathbf{N}}|\overline{\mathbf{I}}\big)\cdot\Pr\big(\overline{\mathbf{I}}\big)\Big)=\sum_{y=1}^{w}\log\Big(\Pr\big(\overline{\mathbf{I}}_y|\overline{\mathbf{I}}_{y-1}\big)\Big)+\sum_{y=1}^{w}\log\Big(\Pr\big(\overline{\mathbf{N}}_y|\overline{\mathbf{I}}_y\big)\Big). \tag{2.8}$$

We Multiply both sides of (2.8) by -1 and rearrange (2.8) to the following:

$$-\log\Big(\Pr\big(\overline{\mathbf{N}}|\overline{\mathbf{I}}\big)\cdot\Pr\big(\mathbf{I}\big)\Big)=\sum_{y=1}^{w}\log\left(\frac{1}{\Pr\big(\overline{\mathbf{I}}_y|\overline{\mathbf{I}}_{y-1}\big)}\right)+\sum_{y=1}^{w}\log\left(\frac{1}{\Pr\big(\overline{\mathbf{N}}_y|\overline{\mathbf{I}}_y\big)}\right). \tag{2.9}$$

Clearly instead of maximizing (2.4) we can equivalently minimize (2.9):

$$\overline{\mathbf{I}}_{opt}=\underset{\overline{\mathbf{I}}\in S^w}{\operatorname{argmin}}\left\{\sum_{y=1}^{w}\log\left(\frac{1}{\Pr\big(\overline{\mathbf{I}}_y|\overline{\mathbf{I}}_{y-1}\big)}\right)+\sum_{y=1}^{w}\log\left(\frac{1}{\Pr\big(\overline{\mathbf{N}}_y|\overline{\mathbf{I}}_y\big)}\right)\right\}. \tag{2.10}$$

The total number of possible sequences is $|S|^w$. This number for the case of $w = 352$ and $|S| = 6$ (if the format of the video is CIF and we use the aforementioned interpolators as Markov chain states) is approximately 8×10^{273}! Therefore we need utilize an efficient optimization method to find the most probable sequence of interpolators which ensures both a reasonable timing complexity and global optimum solution. The structure of the summation in (2.10) over horizontal pixel positions could be properly utilized to perform the optimization much more efficiently. Consider change of interpolator between two adjacent missing pixels (state transition in aforementioned Markov-chain model) as a transition between two consecutive layers of a trellis having members of the set S on each of its layers as shown in Fig. 2.5. This trellis is expanded over an entire missing row. Consider the interpolator transition

between two adjacent missing pixels in Fig. 2.5. We choose the branch and state metrics to be first and second logarithmic terms in (2.10) respectively. According to (2.10), minimum-weight path through the trellis corresponds to the best estimation of the sequence of the interpolators associated with the corresponding missing row. The problem of finding the minimum-weight path can be solved using Viterbi algorithm (Viterbi path). Among edges (interpolator transitions) converging to a same state (interpolator) at each layer of the trellis (pixel position), the one with less accumulated weight will survive and new accumulated weight is saved only for that survival path. Thus for each layer of the trellis we have to save only $|S|$ accumulated weights as we have only $|S|$ survival paths. When reached last layer of the trellis (in our implementation last layer corresponds to the last horizontal position within a missing row of a frame) simply we backtrack from the least-accumulated-weight state at the last layer through the trellis to the first layer via survival path for each state at each layer. Since for each state (interpolator) there exists one and only one survival path that we can move backward within the trellis toward the first layer, this path is unique and corresponds to the optimal sequence of interpolators. Once the optimal sequence of interpolators is estimated, we simply apply it to the entire corresponding missing row. This trellis-represented algorithm proposed shows a great performance with a variety of test sequences.

At this point, we are left to approximate the probabilities in (2.10) i.e., $\Pr\big(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}\big)$ and $\Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big)$. The first term is the probability of transition from one state of the Markov model we discussed earlier to another one. Approximating these probabilities is exactly equivalent to finding an approximation for the transition matrix (TM) of the Markov model we assumed on the sequence of interpolator. The next probability

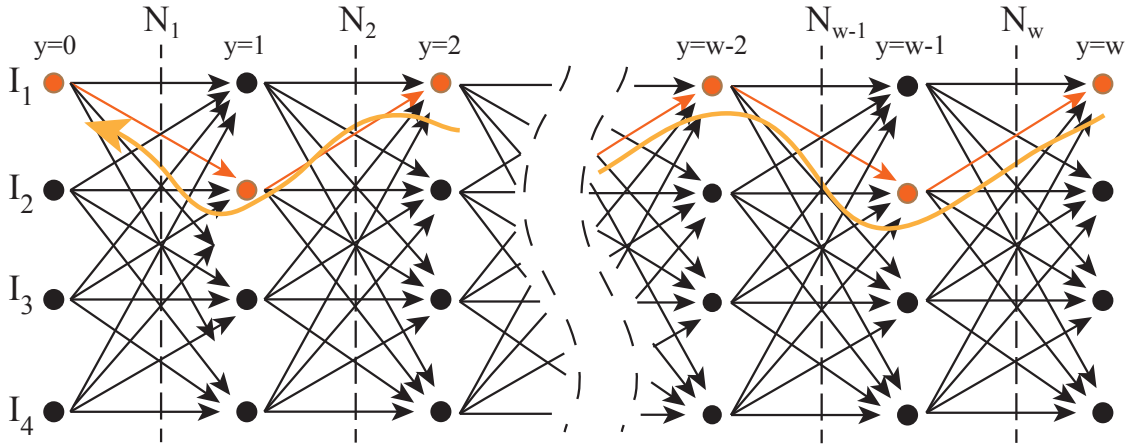


Figure 2.5: A typical trellis expanded over horizontal direction of a missing row with sample states of I_1 , I_2 ,I_3 and I_4. The estimated sequences is $I_2I_1...I_1I_2I_1$ from $y = 1$ to $y = w$.

is the probability of the neighborhood around the missing pixel position given that we impose an specific member of the set S at the missing position. Our strategy for this part of the problem is discussed extensively in the following section.

2.3.3 Approximating the Markov Model Parameters

The remaining task is finding approximations for probabilities in (2.10) i.e., $\Pr\big(\overline{\mathbf{I}}_y|\overline{\mathbf{I}}_{y-1}\big)$ and $\Pr\big(\overline{\mathbf{N}}_y|\overline{\mathbf{I}}_y\big)$. For notation simplicity let us assume here that $\overline{\mathbf{I}}_y = I_i$, $\overline{\mathbf{I}}_{y-1} = I_j$ and $\overline{\mathbf{N}}_y = N$. So we focus on deriving approximations for $\Pr\big(I_i|I_j\big)$ and $\Pr\big(I_i|N\big)$.

Let us briefly discuss our strategy for approximating these probabilities. We start by focusing on approximating the cost of applying of a single interpolator at a missing pixel position. For simplicity consider one of the members of the set S like I_5. When we like to define a cost for an edge-guided interpolator such as I_5, one can immediately think of taking the difference between the original pixel value and the interpolation result with I_5 as a reasonable cost defined for I_5 at that position. In case of de-interlacing, the original value of the missing pixel is not known clearly.

Therefore, the aforementioned difference can not be used. As this difference can not be computed directly, one may think of finding a close approximation by computing this difference for an original neighboring pixel (which we have the actual value) instead of the to-be-interpolated pixel itself. Approximating the cost of, let us say I_5, for a neighboring pixel instead of the missing pixel of interest itself is based on the assumption that if I_5 works well at the to-be-interpolated pixel, it makes a lot of sense that it works well at a neighboring pixel of the missing pixel as well. However, still we can not compute directly the suggested cost at the neighboring pixel. The reason is that we have original value of the neighboring pixel, the members of the set S can not be applied at neighboring pixels as the interpolation value can not be computed. This is where the set S_{LR} comes in. We propose the low resolution version of the set S, S_{LR} as in 2.3. The superiority of the set S_{LR} is that it uses dilated version of the members of the S.

The next step is to define the exact expression for the cost of an interpolator say I_5. Consider the low resolution version of it, I_{LR_5}. As explained in previous paragraph, we can apply the I_{LR_5} at a neighboring pixel of the missing pixel. If the missing pixel position is $\vec{r}$ then this neighbor is clearly a member of $N(\vec{r})$. As suggested the difference between the this member of $N(\vec{r})$ and the interpolation value which results from applying I_{LR_5} at this neighboring position is then considered. Now our strategy is to find the similar differences (costs) for other members of the $N(\vec{r})$ and take the average difference as the cost of applying I_5 at $\vec{r}$. Mathematically we define this cost for I_i of the set S as:

$$C_t(\vec{r}, I_{LR_i}) = \sum_{j=1}^{|N(\vec{r})|} w_j \left\| f(\vec{n}_j) - I_{LR_i}(\vec{n}_j) \right\|_{\ell_2} \tag{2.11}$$

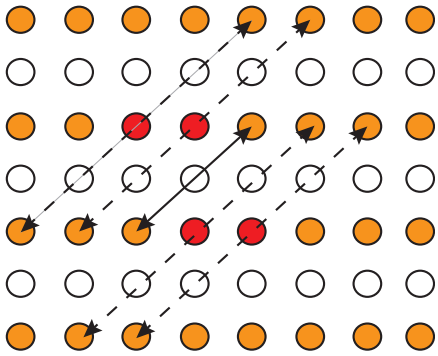


Figure 2.6: I_{LR_5} is applied to each of the six members of $N(\vec{r})$. The resulted differences will be used in (2.11) with different weights to determine the cost of I_5.

where ℓ_2 means ℓ_2-norm, $\vec{n}_j \in N(\vec{r})$ and w_j is the weight of the cost of interpolator I_{LR_i} at the position of the neighboring original pixel $\vec{n}_j$ which belongs to $N(\vec{r})$. Fig. 2.6 better illustrates how we apply the low resolution version of an interpolator to the members of N for I_5 and an example neighborhood of size six around a missing pixel of interest.

Another question about the cost defined in (2.11) is the weights included for different neighbors. Note that our previous assumptions imply that if say I_{LR_5} performs well at missing pixel, it probably performs well on a neighboring pixel as well. The question is that if we consider two different neighbors in $N(\vec{r})$, does I_{LR_5} perform at one of them as great as it perform at the location of the other one. The answer is clearly negative, however if I_{LR_5} performs not the same in two different neighbors, we have to consider a mechanism to distinguish between these two neighbors in cost defined in (2.11). Here is where the weights w_j come in to overcome this problem. Consequently we propose to determine the weights for each neighbor according to the similarity of a patch around it to the same size patch around the missing pixel.

The higher this similarity, the larger is the corresponding weight for that neighbor in (2.11).

To clarify on how we define the weights in (2.11), let us consider a single original neighbor of the current missing pixel located at $\vec{r}$ i.e., $\vec{n}_j$. We consider a patch around it as $P(\vec{n}_j)$. So $P(\vec{n}_j)$ is neighborhood around $\vec{n}_j$ where $\vec{n}_j \in N(\vec{r})$. Also, consider the patch of the same size around the missing pixel position $\vec{r}$ i.e, $P(\vec{r})$. The structural dissimilarity between $P(\vec{r})$ and $P(\vec{n}_j)$ or $dissim\{P(\vec{r}), P(\vec{n}_j)\}$ is defined as:

$$dissim\{P(\vec{r}), P(\vec{n}_j)\} = \|P(\vec{r}) - P(\vec{n}_j)\|_{\ell_2}. \tag{2.12}$$

Then we define w_j for $\vec{n}_j$ as:

$$w_j = \frac{1}{K} \cdot \exp\left\{\frac{1}{2\sigma^2}\left(-dissim\{P(\vec{r}), P(\vec{n}_j\})\right)\right\} \tag{2.13}$$

where K is the normalizing factor i.e., $\sum_{j=1}^{|N|} w_j$. Also note that to calculate w_j we need to temporarily fill each of the patches for the missing pixels which we do this by a simple edge-directional line averaging. Now we assume that the cost of applying I_i at a missing pixel position is approximated by the average cost resulted from (2.11). Thus for $C_t(\vec{r}, I_i)$ we have:

$$C_t(\vec{r}, I_i) \approx C_t(\vec{r}, I_{LR_i}) \tag{2.14}$$

Equation (2.11) and (2.14) suggest that the cost of applying an interpolator I_i is the average cost of applying I_{LR_i} on a neighborhood around the missing pixel. $C_t(\vec{r}, I_i)$ measures the cost of applying interpolator I_i at $\vec{r}$ given the neighborhood $N(\vec{r})$. So we propose that the inverse of this cost is a measure of how probable the neighborhood

$N(\vec{r})$ is when I_i is applied at position $\vec{r}$. In other words we define:

$$\Pr(N(\vec{r})|I_i) = \min\left\{\frac{1}{C_t(\vec{r}, I_i)}, 1\right\}. \tag{2.15}$$

So far we have derived approximations for the probability of the neighborhoods given we impose a specific interpolator in the corresponding missing pixel position. Next step, we calculate transition probabilities $\Pr(I_i|I_j)$. For this aim, consider that we down-sample the current, next and previous interlaced frames of size $\frac{h}{2} \times w$ in horizontal direction by a factor of two to obtain the low resolution progressive frame of size $\frac{h}{2} \times \frac{w}{2}$. Then for all pixels in this set we calculate the $C_t(\vec{r}, I_i)$ and choose the best interpolation I^* as the one with the minimum cost or:

$$\mathrm{I}^* = \underset{1 \le i \le |S|}{\operatorname{argmin}} \{C_t(\vec{r}, I_i)\} \tag{2.16}$$

The result is a/an *state/interpolator matrix* of size half of the frame size in each direction ($\frac{h}{2} \times \frac{w}{2}$) with elements from the set S. Based on this state matrix we form the so-called *transition matrix* or TM and compute the transition probabilities $\Pr(I_i|I_j)$ as:

$$\Pr(I_i|I_j) = \mathrm{TM}(i, j) = \frac{\mathrm{num}(I_j \to I_i)}{\mathrm{num}(I_j)} \tag{2.17}$$

where $\mathrm{num}(X)$ is the number of occurrence of a state (interpolator) X and $\mathrm{num}(X \to Y)$ is the number of transitions from interpolator X to interpolator Y respectively in the state/interpolator matrix. Note that by $\Pr(I_i|I_j)$ we mean the probability of interpolator I_i occurring left to the interpolator I_j in the state matrix mentioned above when scanning left to right. Clearly one may define a scanning scheme other than

suggested above. However, note that the definition of the transition matrix should be modified then to correspond to the scanning scheme defined. Below illustrates the state matrix for a generic sequence of size $w \times h = 14 \times 10$ and the corresponding transition matrix resulted from (2.17):

$$A = \begin{pmatrix} I_1 & I_1 & I_4 & I_6 & I_3 & I_5 & I_1 \\ I_3 & I_3 & I_1 & I_5 & I_5 & I_1 & I_6 \\ I_2 & I_1 & I_1 & I_3 & I_4 & I_4 & I_3 \\ I_6 & I_6 & I_5 & I_1 & I_5 & I_2 & I_5 \\ I_6 & I_6 & I_3 & I_6 & I_2 & I_8 & I_1 \end{pmatrix} \quad \mathrm{TM} = \begin{pmatrix} 0.22 & 0.33 & 0.17 & 0.25 & 0.50 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.17 & 0.14 \\ 0.11 & 0.00 & 0.17 & 0.25 & 0.00 & 0.29 \\ 0.11 & 0.33 & 0.17 & 0.00 & 0.00 & 0.00 \\ 0.22 & 0.33 & 0.17 & 0.00 & 0.17 & 0.14 \\ 0.11 & 0.00 & 0.17 & 0.25 & 0.00 & 0.29 \end{pmatrix}.$$

Now that we have approximations of the probabilities in (2.10) as suggested by (2.15) and (2.17), we rearrange (2.10) in the form of the following:

$$\mathbf{I}_{opt} = \left\{ \underset{\mathbf{I} \in S^W}{\operatorname{argmin}} \sum_{y=1}^{W} \log\left(\frac{1}{\mathrm{TM}(l,k)}\right) + \sum_{y=1}^{W} \log\left(\frac{1}{\min\left\{\frac{1}{C_t(\vec{r},\mathbf{I}_y)}, 1\right\}}\right) \right\}. \tag{2.18}$$

where $l = \pi(\mathbf{I}_{y-1})$ and $k = \pi(\mathbf{I}_y)$ and π is a simple index assignment over the members of the set S as:

$$\pi : S \to \{1, 2, ..., |S|\}, \pi(I_i) = i \tag{2.19}$$

.

Note that (2.18) is the final arrangement of the proposed formulation. In the following section we will study some of both visual and objective experimental results from this trellis-represented algorithm.

So far we revisited the de-interlacing as the problem of assigning the best sequence of interpolators to the corresponding sequence of missing pixels given the sequence of neighborhoods. Based on these sequences we formulated our solution to estimate the entire sequence of interpolators at once. An alternative formulation is to approach the problem as the problem of maximizing the probability of the interpolators at each position (layer of the trellis) given the sequence of the neighborhoods. Other words, with the same observations of the neighborhoods as previous formulation, we consecutively estimate the best interpolator at each layer which finally leads to the sequence of best estimations. In other words we are interested this time to look at the problem of reconstructing a missing line by obtaining a sequence of best estimations instead of estimation of the best sequence of interpolators. This will be the main topic of discussion in the following section.

2.3.4 Reformulation of MAP-Based Estimation Problem

In the previous section we provided a maximum a posteriori sequence estimation that was based on the estimation of the best sequence at once given the original pixels (sequence of neighborhood). Now consider this different approach toward estimating

the entire sequence. Instead of estimating the total sequence at once, we estimate the best state (interpolator) at a specific layer given the original pixels and then we consecutively repeat this for all other positions of the row from left to right to obtain a sequence of estimated interpolators. Other words, we would like to change our approach from estimation of the best sequence of states to obtaining the sequence of best-state-estimations.

We associate with each interpolator or state on each layer of the trellis the probability $\boldsymbol{\lambda}_m(y) = \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^w\big)$ which means the probability of interpolator at y^{th} position of the row being the m^{th} member of set of interpolators S, given the sequence of neighborhoods of that row i.e., $\overline{\mathbf{N}}_1^w$. Our goal is to compute these a posteriori probabilities i.e., the sequence $\boldsymbol{\lambda}$. Then clearly the best estimation of $\mathbf{I}_y$ is the one which maximizes the above $\boldsymbol{\lambda}_m(y)$. We then redo this process for all other layers of the trellis i.e., ys to compute the best interpolator at each missing position and consequently the entire missing row. Therefore, the optimal estimation of the sequence of interpolators for a missing row is:

$$\bar{\mathbf{I}}_{opt} = \left\{ \pi^{-1} \left\{ \underset{1 \le m \le |s|}{\operatorname{argmax}} \{\boldsymbol{\lambda}_m(y)\} \right\} \right\}_{y=1}^{w} \tag{2.20}$$

where π^{-1} is inverse of the index assignment in (2.19). Now we formulate the problem in a way to adopt the Forward-Backward algorithm for processing the trellis and computing the $\boldsymbol{\lambda}_m(y)$ for all values of y and m. We define $\boldsymbol{\alpha}_m(y)$ (forward probability), $\boldsymbol{\beta}_m(y)$ (backward probability) and $\boldsymbol{\gamma}_{mn}(y)$ (transition probability) as:

$$\begin{aligned}
\boldsymbol{\alpha}_m(y) &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^y\big) \\
\boldsymbol{\beta}_m(y) &= \Pr\big(\overline{\mathbf{N}}_{y+1}^w | \pi(\bar{\mathbf{I}}_y) = m\big) \\
\boldsymbol{\gamma}_{mn}(y) &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_y | \pi(\bar{\mathbf{I}}_{y-1}) = n\big)
\end{aligned} \tag{2.21}$$

where $\mathbf{N}_y$ is the neighborhood of the missing pixel at position y of the missing row. It can be shown that for $\boldsymbol{\lambda}_m(y)$ we have:

$$\begin{aligned}
\boldsymbol{\lambda}_m(y) &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^w\big) \\
&= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^y\big) \cdot \Pr\big(\overline{\mathbf{N}}_{y+1}^w | \pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^y\big) \\
&= \boldsymbol{\alpha}_m(y) \cdot \Pr\big(\overline{\mathbf{N}}_{y+1}^w | \pi(\bar{\mathbf{I}}_y) = m\big) \\
&= \boldsymbol{\alpha}_m(y) \cdot \boldsymbol{\beta}_m(y)
\end{aligned} \tag{2.22}$$

where the third equality follows from the Markov property that if the state (interpolator) at position y, $\pi(\bar{\mathbf{I}}_y)$, is known, observations after position y do not depend on $\overline{\mathbf{N}}_1^y$. So we only have to find expressions for $\boldsymbol{\alpha}_m(y)$, $\boldsymbol{\beta}_m(y)$ and $\boldsymbol{\gamma}_{mn}(y)$. First for $\boldsymbol{\alpha}_m(y)$ in terms of $\boldsymbol{\gamma}_{mn}(y)$ we can show that:

$$\begin{aligned}
\boldsymbol{\alpha}_m(y) &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^y\big) \qquad (2.23)\\
&= \sum_{n=1}^{|S|} \Pr\big(\pi(\bar{\mathbf{I}}_{y-1}) = n, \pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_1^y\big)\\
&= \sum_{n=1}^{|S|} \Pr\big(\pi(\bar{\mathbf{I}}_{y-1}) = n, \overline{\mathbf{N}}_1^{y-1}\big) \cdot \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_y | \pi(\bar{\mathbf{I}}_{y-1}) = n, \overline{\mathbf{N}}_1^{y-1}\big)\\
&= \sum_{n=1}^{|S|} \Pr\big(\pi(\bar{\mathbf{I}}_{y-1}) = n, \overline{\mathbf{N}}_1^{y-1}\big) \cdot \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_y | \pi(\bar{\mathbf{I}}_{y-1}) = n\big)\\
&= \sum_{n=1}^{|S|} \boldsymbol{\alpha}_n(y-1) \cdot \boldsymbol{\gamma}_{nm}(y).
\end{aligned}$$

Similarly for $\boldsymbol{\beta}_m(y)$ we can show that in can be recursively computed in terms of $\boldsymbol{\gamma}_{mn}(y)$ as the following:

$$\begin{aligned}
\boldsymbol{\beta}_m(y) &= \Pr\big(\overline{\mathbf{N}}_{y+1}^w, \pi(\bar{\mathbf{I}}_y) = m\big) \qquad (2.24)\\
&= \sum_{n=1}^{|S|} \Pr\big(\pi(\bar{\mathbf{I}}_{y+1}) = n, \overline{\mathbf{N}}_{y+1}^w | \pi(\bar{\mathbf{I}}_y) = m\big)\\
&= \sum_{n=1}^{|S|} \Pr\big(\pi(\bar{\mathbf{I}}_{y+1}) = n, \overline{\mathbf{N}}_{y+1} | \pi(\bar{\mathbf{I}}_y) = m\big) \cdot \Pr\big(\overline{\mathbf{N}}_{y+2}^w, \pi(\bar{\mathbf{I}}_{y+1}) = n, \pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_{y+1}\big)\\
&= \sum_{n=1}^{|S|} \Pr\big(\pi(\bar{\mathbf{I}}_{y+1}) = n, \overline{\mathbf{N}}_{y+1} | \pi(\bar{\mathbf{I}}_y) = m\big) \cdot \Pr\big(\overline{\mathbf{N}}_{y+2}^w, \pi(\bar{\mathbf{I}}_{y+1}) = n\big)\\
&= \sum_{n=1}^{|S|} \boldsymbol{\beta}_n(y+1) \cdot \boldsymbol{\gamma}_{mn}(y+1).
\end{aligned}$$

We need to impose initial conditions for the above recursive expressions for $\boldsymbol{\alpha}_m(y)$ and $\boldsymbol{\beta}_m(y)$. We choose the following initial conditions arbitrarily as they do not have

any actual pixel correspondence:

$$\boldsymbol{\alpha}_m(0) = \boldsymbol{\beta}_m(w) = \begin{cases} 1 & \text{if } m = 0 \\ 0 & \text{if } m \neq 0. \end{cases} \tag{2.25}$$

Note that $y = 0$ does not correspond to any actual position and here we have chosen the $\boldsymbol{\alpha}_m(0)$ arbitrarily. For $\boldsymbol{\gamma}_{mn}(y)$ we can show that:

$$\begin{aligned} \boldsymbol{\gamma}_{mn}(y) &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m, \overline{\mathbf{N}}_y | \pi(\bar{\mathbf{I}}_{y-1}) = n\big) \\ &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m | \pi(\bar{\mathbf{I}}_{y-1}) = n\big) \cdot \Pr\big(\overline{\mathbf{N}}_y | \pi(\mathbf{I}_y) = m, \pi(\bar{\mathbf{I}}_{y-1}) = n\big) \\ &= \Pr\big(\pi(\bar{\mathbf{I}}_y) = m | \pi(\bar{\mathbf{I}}_{y-1}) = n\big) \cdot \Pr\big(\overline{\mathbf{N}}_y | \pi(\bar{\mathbf{I}}_y) = m\big) \end{aligned} \tag{2.26}$$

where in the last equality we made use of the assumption we made in (2.6) for independent observation of neighborhoods. Now based on (2.15) and (2.17) we can rewrite $\boldsymbol{\gamma}_{mn}(y)$ as:

$$\boldsymbol{\gamma}_{mn}(y) = \mathrm{TM}(m, n) \cdot \min\left\{ \frac{1}{C(\vec{r}, \mathbf{I}_y)}, 1 \right\}. \tag{2.27}$$

Note that according to (2.23) and (2.24) the probabilities $\boldsymbol{\alpha}_m(y)$ and $\boldsymbol{\beta}_m(y)$ get smaller as y increases. This may result in underflow when frame size gets larger as in case of CIF, SD and HD video formats. To overcome this problem we define scaled probabilities $\boldsymbol{\alpha}^{sca}_m(y)$, $\boldsymbol{\beta}^{sca}_m(y)$ and $\boldsymbol{\lambda}^{sca}_m(y)$ as:

$$\forall y : \boldsymbol{\alpha}_m^{sca}(y) = \frac{\boldsymbol{\alpha}_m(y)}{\max\limits_{1\leq m\leq |s|}\{\boldsymbol{\alpha}_m(y)\}} \tag{2.28}$$
$$\forall y : \boldsymbol{\beta}_m^{sca}(y) = \frac{\boldsymbol{\beta}_m(y)}{\max\limits_{1\leq m\leq |s|}\{\boldsymbol{\beta}_m(y)\}}.$$

Then for $\boldsymbol{\lambda}_m^{sca}(y)$ we have:

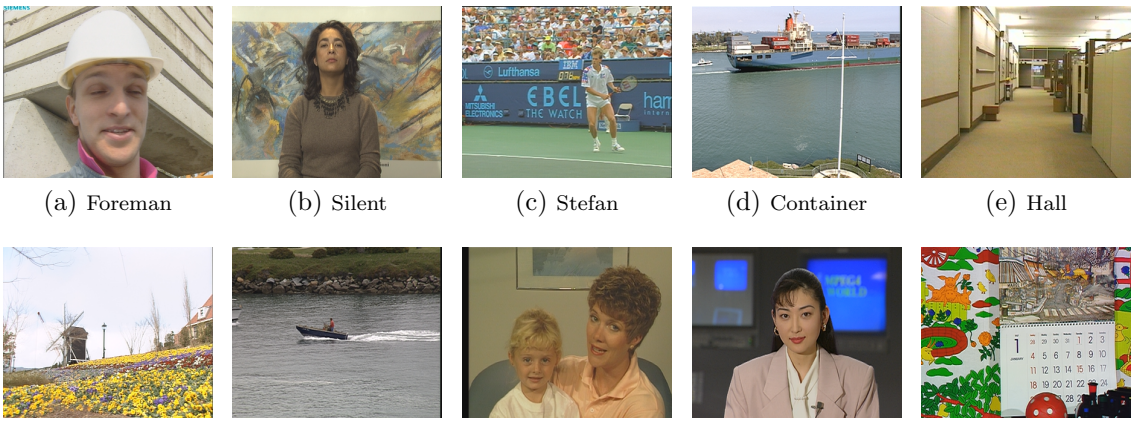
$$\forall y, m : \boldsymbol{\lambda}_m^{sca}(y) = \boldsymbol{\alpha}_m^{sca}(y) \cdot \boldsymbol{\beta}_m^{sca}(y). \tag{2.29}$$

The $\boldsymbol{\lambda}_m^{sca}(y)$ will then be used in (2.20). Therefore, we compute $\boldsymbol{\lambda}_m(y)$s at each missing pixel position (i.e., level of the trellis) then find the interpolator or m that maximizes $\boldsymbol{\lambda}_m(y)$s and choose it as the best interpolator in that position. We do this for missing positions of the row as suggested by (2.20).

So far we have proposed de-interlacing methods based on maximum a posteriori sequence estimation using Forward-Backward algorithm and Viterbi algorithm. Henceforth, they will be referred to as Proposed_FBA and Proposed_VA respectively. Proposed_VA considers transitions between each two states at each layer of the trellis. This leads to a total of $|S|^2$ transitions for each layer of the trellis. Repeating for all W layers of a row sums up to $|S|^2W$ state transitions. This number for the case of Proposed_FBA increases to $2|S|^2W$ as we move through the trellis from left to right and vice versa. Ignoring multiplicative constants and lower power terms and assuming the input size of the algorithms, n, to be equal to the size of the row W, the asymptotic complexities of the Proposed_FBA and Proposed_VA are linear with respect to n i.e., $\theta(n)$. This means that although the Proposed_VA and Proposed_FBA

have the same asymptotic time complexities, Proposed_FBA happens to have a bigger running-time. Therefore by Proposed_FBA we obtain better performance while we give up in terms of complexity compared to Proposed_VA.

2.4 Experimental Results



(a) Foreman (b) Silent (c) Stefan (d) Container (e) Hall

(f) Flower (g) Coastguard (h) Mother (i) Akiyo (j) Mobile

Figure 2.7: First frame of test sequences

This section includes some objective and subjective results of applying the proposed algorithms on a set of test videos with divers characteristics. The test set contains sequences with 90 to 300 frames. Fig. 2.7 shows the first frame of the test sequences. In subsection 2.4.2, we compare the proposed algorithms Proposed_FBA and Proposed_VA. In subsection 2.4.3, we compare the proposed algorithms with some other well-known and/or recently published algorithms.

2.4.1 Adaptive Cost Using NLC

As discussed in section 2.3.3, we proposed to determine the weights in (2.11) adaptively. The adaptive weights in (2.11) are determined by the similarity of the patches around the members of the neighborhood of a missing pixel and the patch around the missing pixel itself. Now to show how important this adaptability could be, in Fig.2.8 we have illustrated the results of interpolation for part the frame 6 of the Foreman sequence with and without this adaptive weights. As can be seen, the adaptive weights result in sharper edges and more accurate reproduction around the object boundaries.

(a) Non-Adaptive Weights (b) Adaptive Weights

Figure 2.8: Comparison of the results for a part of the frame 6 of the sequence Foreman (a) without adaptive weights (simple averaging) and (b) with adaptive weights (using non-local costs).

Also Fig. 2.10 and Fig. 2.9 illustrate the same results for zoom-in parts of the sequence Stefan (frame 41) and Mobile (frame 30) respectively. Both examples show how the adaptive cost in (2.11) can appropriately control the contribution of each of the neighbors of a missing pixel and reduce (exclude) the effect of uncorrelated neighbors in the final cost. Both demonstrations show a sharper and more accurate

(a) Non-Adaptive Weights (b) Adaptive Weights

Figure 2.9: Comparison of the results for a part of the frame 30 of the sequence Mobile (a) without adaptive weights (simple averaging) and (b) with adaptive weights (using non-local costs).

results for the adaptive cost especially around the edges. The mean of squared errors (MSE) or peak signal to noise ratio (PSNR) are very common in image and video processing applications to evaluate the performance of an algorithm objectively. Mean structural similarity (MSSIM) as introduced in [35], is a visual quality measure that we have used in this article to evaluate the qualitative performance of Proposed_FBA and Proposed_VA. Beside better visual performance when using adaptive averaging the PSNR and SSIM obtained are noticeably higher as well. To get a better performance evaluation of the adaptive weights for the complete videos and not only specific frames, in Table 2.1 we have summarized the PSNR and MSSIM obtained for different videos (first 50 frames) with and without adaptive weights. According to the table using non-local cost as introduced in section 2.3.3 enhances the performance of the algorithm on average for the test video sequences shown in Fig. 2.7 in terms of both PSNR and MSSIM.

2.4.2 Comparison of Proposed_FBA and Proposed_VA

We have summarized in Table 2.2, the resulting peak signal-to-noise ratio (PSNR) for Proposed_FBA and Proposed_VA. As can be seen from the Table, Proposed_FBA and

Table 2.1: PSNR and (MSSIM) results with and without adaptive weights. σ=10, patches are of size 7×7 and neighborhoods are cubicles of size 3×3×3. The results are for the first 50 frames of each sequence.

	Non-Adaptive (Proposed_VA)		adaptive	
	PSNR	MSSIM	PSNR	MSSIM
Coastguard	34.32	0.9036	35.05	0.9651
Container	48.04	0.9943	48.04	0.9943
Foreman	38.03	0.9650	38.81	0.9674
Akiyo	48.47	0.9970	48.48	0.9970
Flower	26.05	0.8960	27.87	0.9591
Stefan	27.23	0.8663	28.43	0.9516
Mobile	31.97	0.9653	32.61	0.9812
Silent	42.75	0.9893	43.39	0.9888
Mother	47.23	0.9895	47.17	0.9893
Hall	41.20	0.9799	41.29	0.9800
Average	38.53	0.9546	**39.11**	**0.977**4

(a) Non-Adaptive Weights (b) Adaptive Weights

Figure 2.10: Comparison of the results for a part of the frame 41 of the sequence Stefan (a) without adaptive weights (simple averaging) and (b) with adaptive weights (using non-local costs).

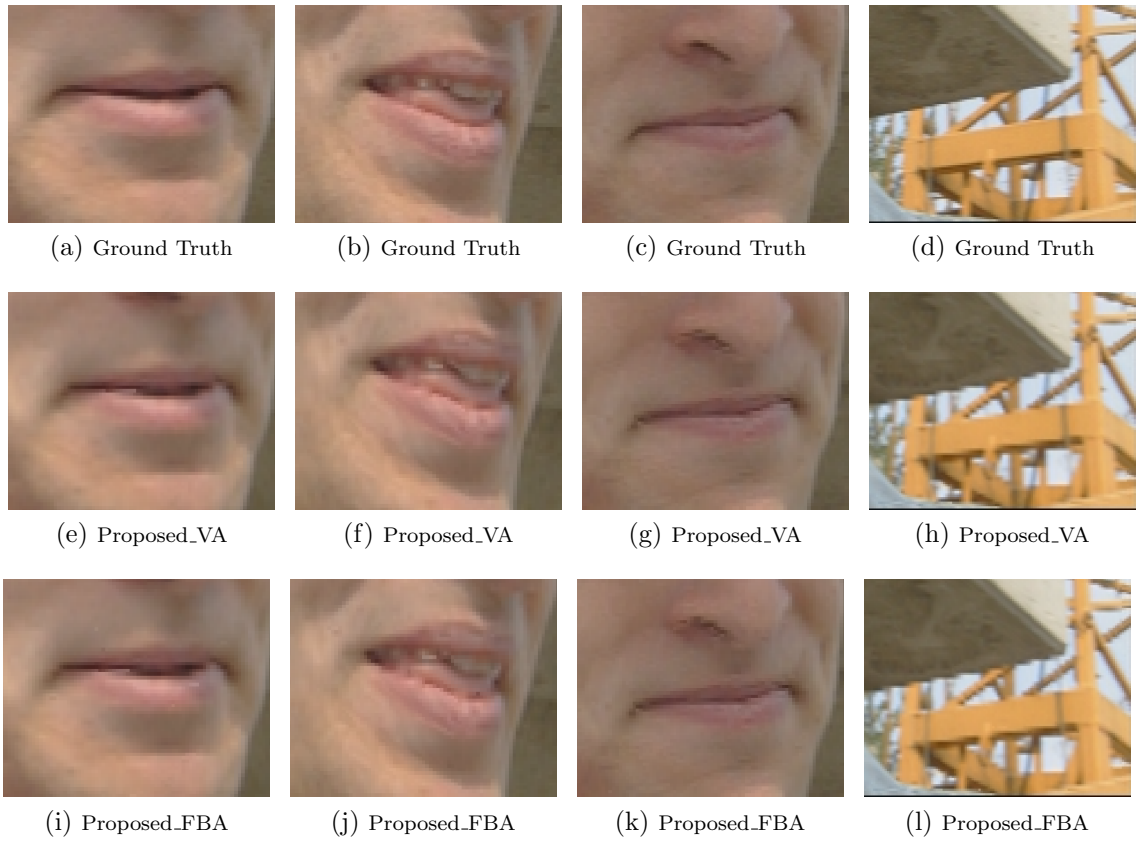
Proposed_VA have relatively close objective results. The set of test videos are in common interchange format (CIF). A quick look at this Table shows that Proposed_FBA has a better performance than Proposed_VA. Overall, they have close objective results. We applied Proposed_FBA and Proposed_VA to a number of test sequences in quarter common interchange formats (QCIF). Table 2.2 shows the results for these sequences. Not surprisingly, the results for Proposed_FBA and Proposed_VA are close again.

Table 2.2: PSNR and Mean Structural Similarity (MSSIM) results for Proposed_FBA and Proposed_VA. σ=10, patches are of size 7×7 and neighborhoods are cubicles of size 3×3×3.

		PSNR		**MSSIM**	
		Proposed_FBA	**Proposed_VA**	**Proposed_FBA**	**Proposed_VA**
CIF	**Coastguard**	35.02	34.70	0.9484	0.9532
	Container	47.15	47.13	0.9940	0.9940
	Foreman	38.19	38.17	0.9652	0.9659
	Akiyo	48.92	48.44	0.9968	0.9968
	Flower	28.99	28.65	0.9579	0.9560
	Stefan	29.85	29.48	0.9465	0.9387
	Mobile	32.96	32.58	0.9814	0.9824
	Silent	41.22	41.04	0.9847	0.9874
	Mother	46.93	46.58	0.9894	0.9893
	Hall	41.51	41.42	0.9804	0.9807
QCIF	**Coastguard**	42.48	42.09	0.9584	0.9432
	Container	52.15	52.11	0.9940	0.9940
	Foreman	41.68	41.54	0.9659	0.9652
	Akiyo	52.70	52.63	0.9968	0.9968
	Silent	41.02	40.09	0.9579	0.9560

The mean of squared errors or peak signal to noise ratio are very common in image and video processing applications to evaluate the performance of an algorithm objectively. Mean structural similarity (MSSIM) as introduced in [35], is a visual quality measure that we have used in this article to evaluate the qualitative performance of Proposed_FBA and Proposed_VA. This measure of subjective quality for Proposed_FBA and Proposed_VA is computed on the aforementioned test set and included in Table 2.2. In accordance with the Table, Proposed_FBA and Proposed_VA compete closely to each other on the test sequences in terms of resulting MSSIM.

Fig. 2.11 shows the zoom-in comparison of Proposed_FBA and Proposed_VA. The frames selected are 22, 28, 120 and 178 of the Foreman sequence. At different



(a) Ground Truth (b) Ground Truth (c) Ground Truth (d) Ground Truth

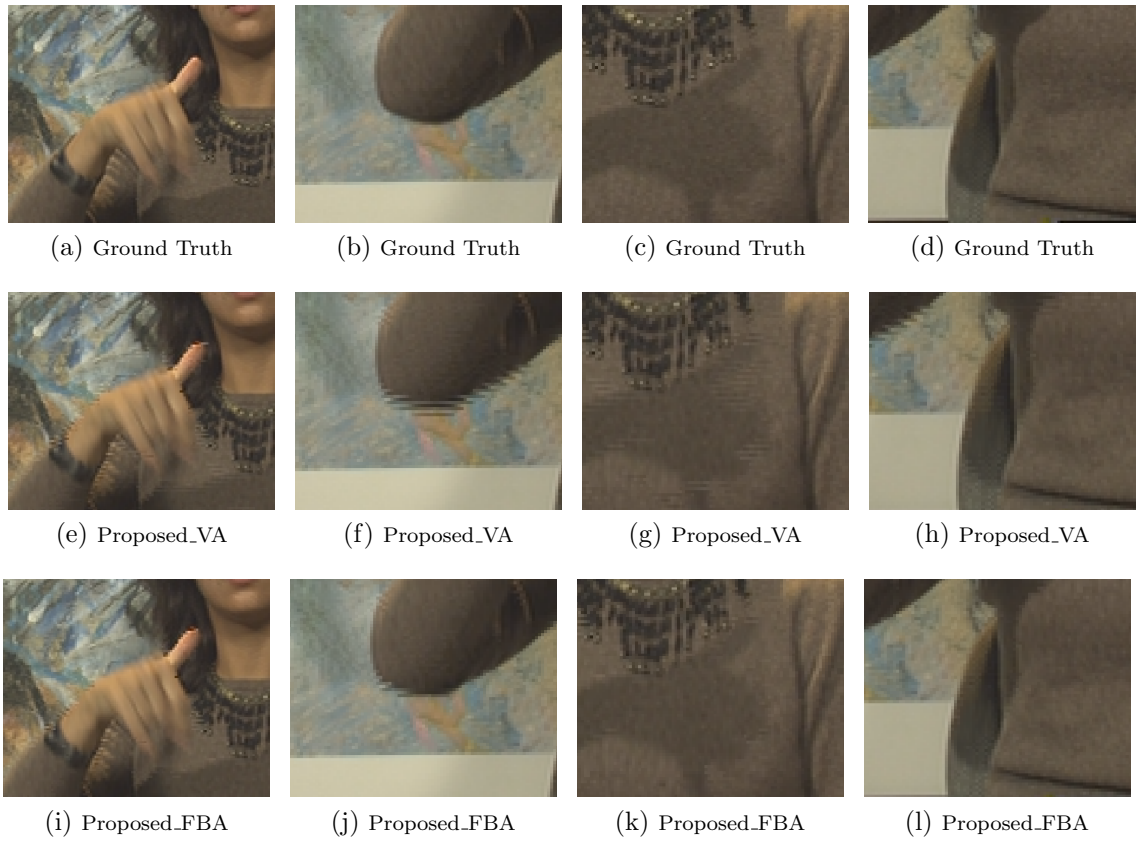
(e) Proposed_VA (f) Proposed_VA (g) Proposed_VA (h) Proposed_VA

(i) Proposed_FBA (j) Proposed_FBA (k) Proposed_FBA (l) Proposed_FBA

Figure 2.11: Zoom-in comparison of Proposed_FBA and Proposed_VA for frames 22, 28, 120, 178 of the Foreman sequence.

frames different areas of the interpolated frame are compared to the original frame as a ground truth. Comparing the results reveals that Proposed_FBA is slightly more successful to recover the area around the mouth specifically the teeth and lips. Fig. 2.11 parts (j), (k) and (l) shows the results for a part of the frame with intense and shallow-angle edges. There is not significant difference between Proposed_FBA and Proposed_VA in this parts and both suffer from artifacts due to near horizontal edges.

Generally high frequency elements in vertical direction are challenging for every algorithm. Better results can be achieved by adding more directional candidate



(a) Ground Truth (b) Ground Truth (c) Ground Truth (d) Ground Truth

(e) Proposed_VA (f) Proposed_VA (g) Proposed_VA (h) Proposed_VA

(i) Proposed_FBA (j) Proposed_FBA (k) Proposed_FBA (l) Proposed_FBA

Figure 2.12: Zoom-in comparison of Proposed_FBA and Proposed_VA for frame 156 of the Silent sequence.

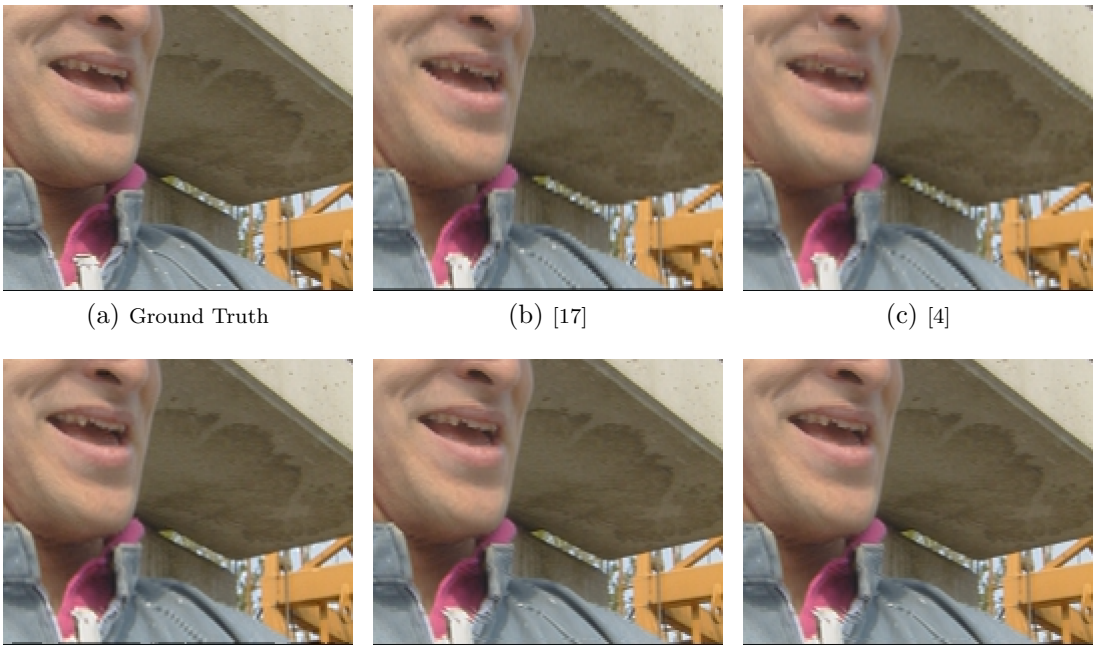
interpolators to the set of eligible interpolators. Adding more near horizontal interpolators can improve the performance of the algorithm visually at the cost of more computational complexity. According to Table 2.2, Proposed_FBA often outperforms Proposed_VA. As an example, the average PSNR achieved for the Silent sequences by Proposed_FBA is higher than Proposed_VA. This is also true about the mean structural similarity as suggested by Table 2.2. Fig. 2.12 shows the results for the frame 156 of this sequence. The critical parts of the frame are those shown in Fig. 2.12. Parts (b) and (c) of Fig. 2.12, compare the results around the fingers

Table 2.3: PSNR comparison of six algorithms with Proposed_FBA and Proposed_VA

	Method1	Method2	Method3	Method4	Method5	**VXP®**	**Proposed_FBA**
Coastguard	27.82	28.78	30.87	32.62	34.33	33.57	**35.02**
Container	27.82	28.60	41.22	44.96	43.06	44.73	**47.15**
Foreman	33.07	34.01	33.92	34.65	35.81	36.37	**38.19**
Akiyo	37.84	39.61	45.17	46.95	47.76	47.58	**48.92**
Flower	21.93	22.23	26.04	27.83	31.70	23.57	**28.99**
Stefan	26.53	27.56	26.53	27.18	27.24	**30.06**	29.85
Mobile	23.15	23.76	25.06	28.21	29.36	29.62	**32.96**
Silent	33.04	34.43	38.76	38.82	**41.98**	41.27	41.22
Mother	37.95	38.59	41.69	42.37	45.18	46.88	**46.93**
Hall	37.54	38.42	38.09	38.13	40.34	41.02	**41.51**

for Proposed_FBA and Proposed_VA. This part of the frame contains fast and complex motion in different directions. As can be seen, Proposed_FBA reconstructs the boundaries of the fingers with less artifacts when compared to Proposed_VA. Parts (e) and (f) are another example of superiority of Proposed_FBA over Proposed_VA for this frame. Clearly Proposed_VA can not recover horizontal edges around the elbow properly, however, Proposed_FBA works interestingly well in this part with almost negligible artifacts. This is also the case in parts (h) and (i) which compares the area on the body of the woman where the shadow of the moving hand can be seen. Although the Proposed_FBA restores the boundaries of the shadow almost artifact free, there can be seen some feathering in the results from Proposed_VA. In this part Proposed_VA is not as successful as Proposed_FBA to utilize spatial interpolators and using temporal interpolators instead of spatial ones results in feathering around the shadow of the fingers. Parts (k) and (l) address the same problem on the boundaries of the hand and other portions of the shadow on the woman's body.

In this section, we compare the performance of Proposed_FBA and Proposed_VA



(a) Ground Truth (b) [17] (c) [4]

(d) VXP® (e) Proposed_FBA (f) Proposed_VA

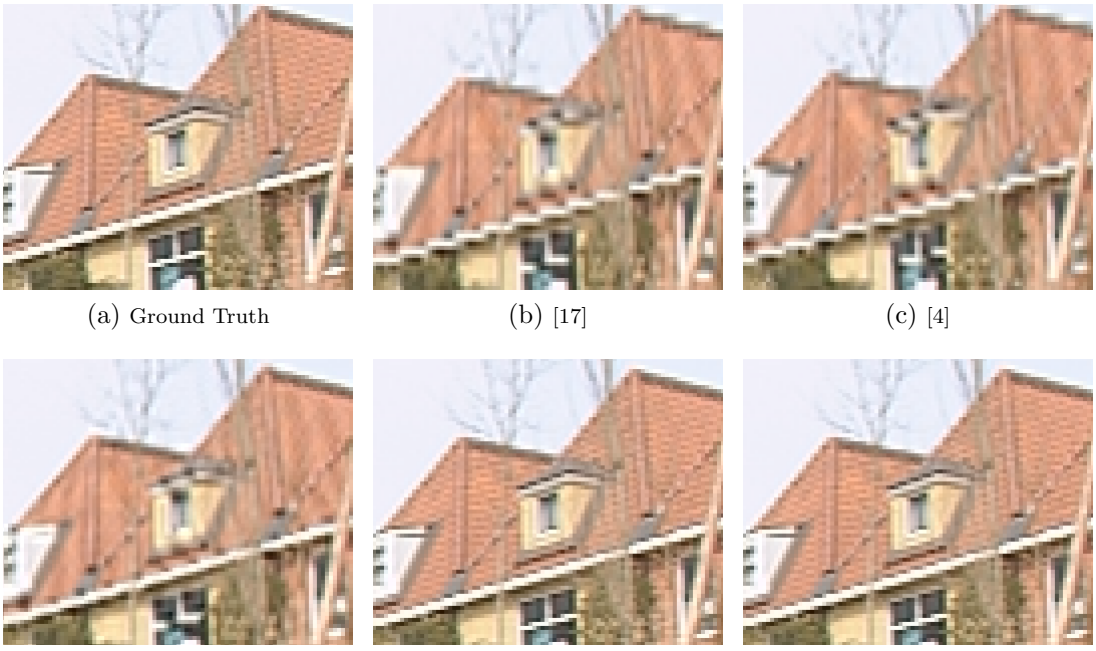
Figure 2.13: Zoom-in Comparison of six different algorithms for Frame 130 of the Foreman.

with some state-of-the-art de-interlacing algorithms proposed in this field. Table 2.3 shows PSNR achieved by Proposed_FBA and Proposed_VA and six other algorithms proposed in [26], [17], [4], [13], [6] (named Method1 to Method5 from left to right in the table) and state-of-the-art commercial de-interlacing algorithm embedded in VXP® video processor from Sigma Designs.

2.4.3 Comparison with Other Algorithms

According to Table 2.3, Proposed_FBA and Proposed_VA achieve higher PSNR than the other six algorithms. The amount of improvement over the other six algorithms is considerable for sequences Container, Foreman and Flower. Also note that the PSNR values reported in Table 2.3 for Proposed_FBA and Proposed_VA are achieved without

motion estimation/compensation, however they are comparable to or better than motion-compensated algorithms which are among the best reported in the literature or market in recent years.



(a) Ground Truth (b) [17] (c) [4]

(d) VXP® (e) Proposed_FBA (f) Proposed_VA

Figure 2.14: Zoom-in Comparison of six different algorithms for Frame 94 of the Flower.

Fig. 2.13 compares the visual results of these algorithms with the Proposed_FBA and Proposed_VA for the frame 130 of the Foreman sequence. VXP® restores the edges around the shoulder better than [4]. Proposed_FBA and Proposed_VA restore the edges even better than VXP® and almost artifact-free. The superiority of Proposed_FBA and Proposed_VA in recovering edges is significant at the boundaries of the lips, nose and mouth specifically teeth. This is also visible at rucks on the cloth.

Fig. 2.14 is another demonstration of the proposed algorithms in comparison with the competitors. In this example, the great performance of Proposed_FBA and

Proposed_VA in estimating the edges and fine details is clear. The results of Proposed_FBA and Proposed_VA are almost artifact-free even in highly detailed portions of the frame. Note that we have included only three spatial directional interpolators in the set S, however, the algorithm based on estimating the best sequence of directional interpolations, well recovers the shallow angles in the frame. Note that the candidate interpolators or the states on the trellis are not confined to the interpolators that we defined. In fact, one may find it useful to add other interpolation methods to the set of candidate interpolators, S.

Chapter 3

Optical Flow Computation

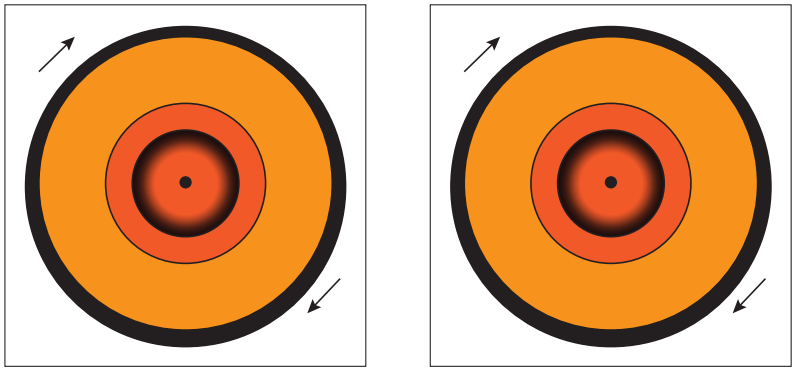
3.1 Introduction

Optical flow or optic flow is the pattern of motion in a visual scene which is caused by the relative motion between an observer and the scene. Estimation of the motion finds diverse applications in image/video processing and computer vision fields such as video compression, segmentation, vehicle tracking and traffic analysis, robot navigation and visual odometry. Different motion estimation or optical flow computation techniques have been proposed in the literature. Differential methods are one of the most popular categories among motion compensation methods. Following subsections, we will briefly review these techniques and particularly the most well-known algorithms among each category. Then we will discuss integrating these algorithms into the previously proposed algorithms.

3.2 Differential Methods

Please note that optical flow as explained earlier is, in general, different from 2-D displacement /velocity field due to (1) lack of sufficient spatial image gradient and (2) changes in external illumination. In other words, optic flow is the observable change of luminance between the corresponding positions in successive frames which is in general not exactly equivalent to 2-D motion.

There must be sufficient gray-level (color) variation within the moving region for the actual motion to be observable. An example of an unobservable motion is shown in figure 3.1, where a circle with uniform intensity rotates about its center. This motion produces no optical flow and is unobservable.

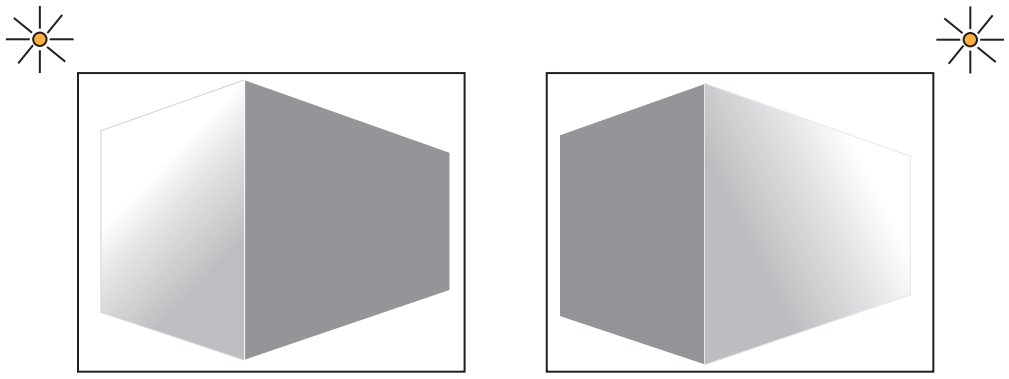


(a) frame k (b) frame k+1

Figure 3.1: (A solid cylinder rotating around its central axis. Although the actual motion is present, no optical flow can be detected due to insufficient change of luminance.

On the other hand an observable optical flow may not always correspond to an actual motion. An example is when an external illumination varies from frame to frame as shown in figure 3.2. An optical flow will be observed even though there is no motion. 2-D motion estimation suffers from existence, uniqueness and continuity

problems.

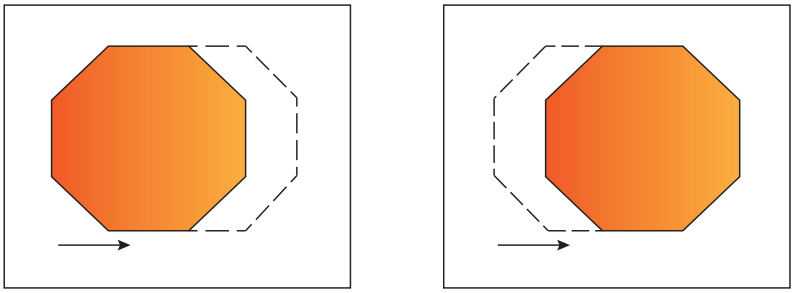


(a) frame k (b) frame k+1

Figure 3.2: Change of illumination due to an external source can produce observable optical flow even when no actual motion is present.

Existence of a solution: For example, no correspondence can be established for covered/uncovered background points. This is known as occlusion problem. The covered and uncovered background concept has been depicted in figure 3.3. In this figure, the object translates horizontally between the two frames. The dotted region in the left frame is going to be covered in the next (right) frame. Therefore, it is not possible to find a correspondence for these pixels in the right frame. Also the dotted region in the right frame is the uncovered background by the motion of the object. There is no correspondence for these pixels in the previous (left) frame as well.

Uniqueness of the solution: Consider the components of the displacement at each pixel as independent variables. Thus, the number of unknowns is twice the number of observations (elements of the frame difference). This leads to the so-called aperture problem. The aperture problem is illustrated in figure 3.4. Consider corner of an object moving upward. If we estimate the motion based on a local window indicated by aperture A, then it is not possible to determine whether the object is moving



(a) frame k (b) frame k+1

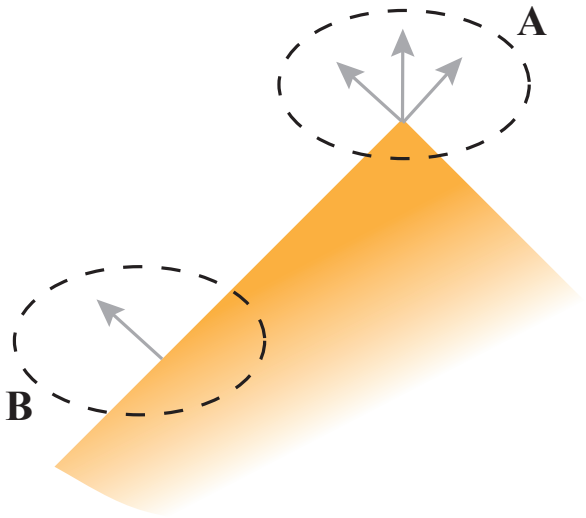
Figure 3.3: (a) Dotted region will be covered in the next frame. No correspondence will be found in the next frame. (b) Dotted region is uncovered background and no motion vector will point to this region.

upward or perpendicular to the edge. Unlike aperture A, aperture B does not suffer from this problem. The motion in the direction perpendicular to the edge is called normal flow. It can be shown that we can only determine motion that is orthogonal to the spatial image gradient at any pixel. To verify this theoretically, consider the continuous 3-D signal. Let $I_c(x, y, t)$ denote this continuous space-time intensity distribution. If the intensity remains constant (shows minimum variation) along a motion trajectory, we have:

$$\frac{d\big(I_c(x,y,t)\big)}{dt} = 0 \tag{3.1}$$

where x and y vary by t according to the motion trajectory. Using the chain rule of differentiation, (3.1) can be expressed as:

$$\frac{\partial\big(I_c(x,y,t)\big)}{\partial x} \cdot \mathbf{v}_x(x,y,t) + \frac{\partial\big(I_c(x,y,t)\big)}{\partial y} \cdot \mathbf{v}_y(x,y,t) + \frac{\partial\big(I_c(x,y,t)\big)}{\partial t} = 0 \tag{3.2}$$



(a) frame k

Figure 3.4: (a) Dotted region will be covered in the next frame. No correspondence will be found in the next frame. (b) Dotted region is uncovered background and no motion vector will point to this region.

where $\mathbf{v}_x$ and $\mathbf{v}_y$ denote the vertical and horizontal velocity fields in terms of continuous spatial and temporal coordinates, respectively. The above equation is known as optical flow equation (OFE) or the optical flow constraint. It sometimes is expressed as following in the literature:

$$\langle \nabla I_c(x,y,t), \mathbf{v}(x,y,t) \rangle + \frac{\partial \big(I_c(x,y,t)\big)}{\partial t} = 0. \tag{3.3}$$

If we decompose $\mathbf{v}$ into perpendicular and parallel components to the direction of the local edge, denoted by $\mathbf{v}_\perp$ and $\mathbf{v}_{||}$ respectively, we can rewrite 3.3 as:

$$\underbrace{\langle \nabla I_c(x,y,t), \mathbf{v}_{||}(x,y,t) \rangle}_{=0} + \langle \nabla I_c(x,y,t), \mathbf{v}_\perp(x,y,t) \rangle + \frac{\partial \big(I_c(x,y,t)\big)}{\partial t} = 0. \tag{3.4}$$

The first term is clearly zero and consequently we will have:

$$\|\mathbf{v}_{\perp}(x,y,t)\| = \frac{-\frac{\partial\left(I_c(x,y,t)\right)}{\partial t}}{\|\nabla I_c(x,y,t)\|}. \tag{3.5}$$

This proves what was claimed a few lines ago that we can estimate the flow that is orthogonal to the edge at the corresponding position (parallel to the direction with maximum spatial gradient).

Continuity of the optical flow field: Motion estimation is highly sensitive to the presence of observation noise in video images. A small amount of noise may result in a large deviation in the motion estimates.

Many algorithms based on basic optical flow equation have been proposed during the last decades of research on this topic and all of them try to introduce heuristics to overcome the aforementioned problems we explained, to acquire more accurate and/or robust estimated flow. The main body of the literature in this important subfield (differential methods) can be easily classified into two main categories. The first group of algorithms are those rooted in the original work of [23]. The second category of works are mainly rooted in inspirational work of [12]. In this part we take a brief look on each of these categories then we review the best published works in each of these fields. The goal of this part is to finally study the effect of the state-of-the-art optical flow methods when incorporated into our proposed algorithms in previous chapters.

3.2.1 Lucas-Kanade-Based Methods

In this chapter we will explain one of the most popular methods based on original work of [23]. One of the most well-known methods based on the work of [23] is the one

reported in [2]. This Lucas-Kanade (LK)-based method (also the method that will be discussed next) uses a pyramidal implementation for compensating large motions. In other words as will be discussed following, modern optical flow algorithms use a coarse to fine optical flow computation that enables estimating large motions.

LK-Based Objective Function

To simplify the notation, suppose that f_1 and f_2 are two frames from which the optical flow should be estimated. Therefore f_1 and f_2 are discrete functions of gray-level values. Consider the upper left and bottom right positions of the frame as $(1, 1)$ and (n_x, n_y) respectively. Let $\mathbf{p}$ denote a position in the frame. If $\mathbf{p}_1$ points a position in the first frame then the goal of optical flow computation is to find the location $\mathbf{p}_2 = \mathbf{p}_1 + \mathbf{v} = (p_x + v_x, p_y + v_y)$ on the second image, f_2 such that $f_1(\mathbf{p}_1) = f_2(\mathbf{p}_1)$ where $\mathbf{d} = (d_x, d_y)$ is the corresponding motion vector that should be estimated. We previously discussed the aperture problem. To minimize this problem it is useful to define the notion of similarity in a 2-D neighborhood. Consequently the objective function is defined as following:

$$\epsilon(\mathbf{v}) = \epsilon(v_x, v_y) = \sum_{x=p_x-w_x}^{p_x+w_x} \sum_{y=p_y-w_y}^{p_y+w_y} \Big(f_1(x, y) - f_2(x + v_x, y + v_y) \Big)^2 \tag{3.6}$$

where w_x and w_y determine the size of the rectangular neighborhood window. In other words, the similarity is measured on an image neighborhood which is sometimes called integration window in some LK-based algorithms. Accuracy and robustness are two important components in LK-based algorithms. Small window sizes are preferable especially in detailed areas in order to preserve the fine details. This is especially

critical at occluding areas where there are potentially patches with different velocities. So for higher accuracies, small window sizes are preferred. On the other hand, to handle large motions, intuitively it is preferable to pick a larger integration window. Therefore naturally there is a tradeoff between local accuracy and robustness when choosing the integration window size. Pyramidal implementation and iterative Lucas-Kanade flow computation are to guarantee the robustness and local accuracy respectively.

Image Pyramidal Representation

Suppose f is an image of size $n_x \times n_y$. We use f^0 to denote the level zero image which is basically the highest resolution image (the initial raw image). We use n_x^0 and n_y^0 to show the height and width of the image at this level. Clearly $n_x^0 = n_x$ and $n_y^0 = n_y$. We generalize this to other levels of the pyramid with L =1,2,3,... as the index of the level. If f^{L-1} is the image at the level $L-1$ then the f^L is obtained recursively by down-sampling f^{L-1}. More precisely we are using the low-pass filter of $(1/4, 1/2, 1/4) \times (1/4, 1/2, 1/4)^T$ as anti-aliasing filter before down-sampling. Some LK-based algorithms have also used $(1/16, 1/4, 3/8, 1/4, 1/16) \times (1/16, 1/4, 3/8, 1/4, 1/16)^T$. The above equation is used to generate the image pyramid of two images f_1 and f_2 i.e., ${f_1^L}_{L=0,...,L_m}$ and ${f_2^L}_{L=0,...,L_m}$. The height of the pyramid is taken heuristically. As explained before, the main motivation behind pyramidal representation is to be able to handle large motions.

Multi-resolution Optical Flow Computation

Note that the final goal at point $\mathbf{p}_1$ in image f_1 is to find corresponding location $\mathbf{p}_2 = \mathbf{p}_1 + \mathbf{v}$ in image f_2 or equivalently the vector $\mathbf{v}$. The correspondence of point $\mathbf{p}$ at resolution L is:

$$\mathbf{p}^L = (p_x^L, p_y^L) = \frac{\mathbf{p}}{2^L}. \tag{3.7}$$

First the optical flow is computed at the lowest resolution (L_m). Then the result is propagated to the one-higher resolution level (considering down-sampling factor between the levels) as an initialization of the next level optical flow. Let us assume that $\mathbf{u}^L = (u_x^L, u_y^L)$ is available from the computations done in level $L+1$ to level L. Then we have to find the the residual displacement vector field $\mathbf{v}^L = (v_x^L, v_y^L)$ that minimizes the new image matching error function :

$$\epsilon^L(\mathbf{v}^L) = \epsilon^L(v_x^L, v_y^L) = \sum_{x=p_x^L-w_x}^{p_x^L+w_x} \sum_{y=p_y^L-w_y}^{p_y^L+w_y} \left(f_1^L(x,y) - f_2^L(x+u_x^L+v_x^L, y+u_y^L+v_y^L) \right)^2. \tag{3.8}$$

In other words, the initial guess flow vector $\mathbf{u}^L$ pre-translates the image patch in the second image f_2. This way the residual vector $\mathbf{v}^L = (d_x^L, d_y^L)$ remains a small and easy-to-compute vector which can be estimated by a standard LK step. Consequently, we can handle large pixel motions using multi-resolution optical flow computation. We will discuss the iterative LK method next section. Before, let assume that through an standard iterative LK method (explained in the next section) we have computed the residual optical flow, $\mathbf{v}^L$ at level L where $\mathbf{u}^L$ was the initial guess motion vector.

As explained earlier, we propagate the results to the next higher-resolution level and then we use it as an initial guess flow for that level as:

$$\mathbf{u}^{L-1} = 2\big(\mathbf{u}^{L} + \mathbf{v}^{L}\big). \tag{3.9}$$

Through the same procedure the residual vector for this new level, $\mathbf{v}^{L-1}$, is then computed. This latter vector is the one that minimizes $\epsilon^{L-1}\big(\mathbf{v}^{L-1}\big)$. This process goes until we get to the highest resolution level and obtain $\mathbf{v}^{0}$ by minimizing $\epsilon^{0}\big(\mathbf{v}^{0}\big)$. Then we get the final value of the estimated flow as:

$$\mathbf{v} = \big(\mathbf{u}^{0} + \mathbf{v}^{0}\big). \tag{3.10}$$

The clear advantage of this pyramidal motion estimation is that if, v_{max} is the pixel displacement that can be handled by the elementary step then the total amount of displacement that is handled, $v_{max_{final}}$, will be:

$$v_{max_{final}} = \big(2^{L_m+1} - 1\big)v_{max}. \tag{3.11}$$

This will keep the size of the integration window small while enabling estimation of large motions.

Iterative Lucas Kanade Flow Computation

In this part we describe the core iterative LK-Based method for optic flow computation. At every level L we seek to find the $\mathbf{v}^{L}$. As the iterative method will remain the same, we will simplify the notation here and suppress the level index L. For this, consider A and B defined as:

$$A(x,y) \doteq f_1^L(x,y) \quad \forall (x,y) \in [s_x - w_x - 1, s_x + w_x + 1] \times [s_y - w_y - 1, s_y + w_y + 1]$$
$$B(x,y) \doteq f_2^L(x + u_x^L, y + u_y^L) \quad \forall (x,y) \in [s_x - w_x, s_x + w_x] \times [s_y - w_y, s_y + w_y]$$

where we have used $\mathbf{s} = (s_x, s_y) = \mathbf{p}^L$. Also for the same reason we will use $\boldsymbol{\nu} = (\nu_x, \nu_y)$ instead of $\mathbf{v}^L$. Consequently we have to find the vector $\boldsymbol{\nu}$ that minimizes the matching function:

$$\epsilon(\boldsymbol{\nu}) = \sum_{x=s_x-w_x}^{s_x+w_x} \sum_{y=s_y-w_y}^{s_y+w_y} \big(A(x,y) - B(x+\nu_x, y+\nu_y)\big)^2. \tag{3.12}$$

At the minimum of the objective function, the first derivative of ϵ with respect to $\boldsymbol{\nu}$ is zero:

$$\left.\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right|_{\boldsymbol{\nu}=\boldsymbol{\nu}_{opt}} = (0,0). \tag{3.13}$$

After expansion of the derivative we obtain:

$$\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}} = -2 \sum_{x=s_x-w_x}^{s_x+w_x} \sum_{y=s_y-w_y}^{s_y+w_y} \big(A(x,y) - B(x+\nu_x, y+\nu_y)\big) \cdot \left(\frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}\right). \tag{3.14}$$

Consider replacing the $B(x+\nu_x, y+\nu_y)$ by its first order Taylor expansion around $\boldsymbol{\nu} = (0,0)$. There is a good chance that this is a good approximation due to the fact that we expect small residual vectors (thanks to pyramidal implementation):

$$\frac{\partial \epsilon(\boldsymbol{\nu})}{\partial \boldsymbol{\nu}} \approx -2 \sum_{x=s_x-w_x}^{s_x+w_x} \sum_{y=s_y-w_y}^{s_y+w_y} \left(A(x,y) - B(x,y) - \left(\frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}\right) \boldsymbol{\nu}\right) \cdot \left(\frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}\right). \quad (3.15)$$

In the above equation, $A(x,y) - B(x,y)$ is the temporal image derivative at the point (x,y):

$$\delta f(x,y) \doteq A(x,y) - B(x,y) \quad \forall (x,y) \in [s_x - w_x, s_x + w_x] \times [s_y - w_y, s_y + w_y]. \quad (3.16)$$

For simplicity we show the image gradient vector as:

$$\nabla f = \begin{pmatrix} \dot{f}_x \\ \dot{f}_y \end{pmatrix} \doteq \left(\frac{\partial B}{\partial x}, \frac{\partial B}{\partial y}\right)^T. \quad (3.17)$$

The image derivatives $\dot{f}_x$ and $\dot{f}_y$ can be computed directly from the first image $A(x,y)$, therefore $\forall (x,y) \in [s_x - w_x, s_x + w_x] \times [s_y - w_y, s_y + w_y]$:

$$\dot{f}_x(x,y) = \frac{\partial A(x,y)}{\partial x} = \frac{A(x+1,y) - A(x-1,y)}{2} \quad (3.18)$$

$$\dot{f}_y(x,y) = \frac{\partial A(x,y)}{\partial y} = \frac{A(x,y+1) - A(x,y-1)}{2}. \quad (3.19)$$

Following latter equations, equation (3.15) can be written:

$$\frac{1}{2}\left(\frac{\partial\epsilon(\boldsymbol{\nu})}{\partial\boldsymbol{\nu}}\right)^T \approx \sum_{x=s_x-w_x}^{s_x+w_x}\sum_{y=s_y-w_y}^{s_y+w_y}\left(\begin{pmatrix}\dot{f}_x^2 & \dot{f}_x\dot{f}_y\\ \dot{f}_x\dot{f}_y & \dot{f}_y^2\end{pmatrix}\boldsymbol{\nu} - \begin{pmatrix}\delta f\dot{f}_x\\ \delta f\dot{f}_y\end{pmatrix}\right). \tag{3.20}$$

To simplify the notations, we use the following matrix notations:

$$\mathbf{G} \doteq \sum_{x=s_x-w_x}^{s_x+w_x}\sum_{y=s_y-w_y}^{s_y+w_y}\begin{pmatrix}\dot{f}_x^2 & \dot{f}_x\dot{f}_y\\ \dot{f}_x\dot{f}_y & \dot{f}_y^2\end{pmatrix} \quad \text{and} \quad \mathbf{b} \doteq \sum_{x=s_x-w_x}^{s_x+w_x}\sum_{y=s_y-w_y}^{s_y+w_y}\begin{pmatrix}\delta f\dot{f}_x\\ \delta f\dot{f}_y\end{pmatrix}. \tag{3.21}$$

Consequently equation (3.20) can be written as:

$$\frac{1}{2}\left(\frac{\partial\epsilon(\boldsymbol{\nu})}{\partial\boldsymbol{\nu}}\right)^T \approx \mathbf{G}\boldsymbol{\nu} - \mathbf{b}. \tag{3.22}$$

Therefore the optimum optical flow vector i.e., $\boldsymbol{\nu}_{opt}$ can be obtained as :

$$\boldsymbol{\nu}_{opt} = \mathbf{G}^{-1}\mathbf{b}. \tag{3.23}$$

So far we have explained the basic LK method which is valid only when our basic assumption (small pixel displacement) is true. However, to get an accurate solution, we iterate multiple times on this scheme. Suppose k is the iteration index. At iteration $k \geq 1$, suppose that $\boldsymbol{\nu}^{k-1} = \left(\nu_x^{k-1}, \nu_y^{k-1}\right)^T$ is computed from previous iterations. Let B_k be the new translated image according to $\boldsymbol{\nu}^{k-1}$ or:

$$B_k(x,y) = B(x+\nu_x^{k-1}, y+\nu_y^{k-1}) \quad \forall(x,y) \in [s_x-w_x, s_x+w_x]\times[s_y-w_y, s_y+w_y]. \tag{3.24}$$

The goal is to compute the residual motion vector $\boldsymbol{\xi}^k = (\xi_x^k, \xi_y^k)$ that minimizes the following objective function:

$$\epsilon^k(\boldsymbol{\xi}^k) = \epsilon(\xi_x^k, \xi_y^k) = \sum_{x=s_x-w_x}^{s_x+w_x} \sum_{y=s_y-w_y}^{s_y+w_y} \big(A(x,y) - B_k(x+\xi_x^k, y+\xi_y^k)\big)^2. \tag{3.25}$$

As explained before the solution is:

$$\boldsymbol{\xi}_k = \mathbf{G}^{-1}\mathbf{b}_k \tag{3.26}$$

where the $\mathbf{b}_k$ is defined as follows:

$$\mathbf{b} \doteq \sum_{x=s_x-w_x}^{s_x+w_x} \sum_{y=s_y-w_y}^{s_y+w_y} \begin{pmatrix} \delta f_k(x,y)\dot{f}_x(x,y) \\ \delta f_k(x,y)\dot{f}_y(x,y) \end{pmatrix} \tag{3.27}$$

where the k^{th} image difference δf_k is defined as follows:

$$\dot{f}_k(x,y) = A(x,y) - B_k(x,y). \tag{3.28}$$

Note that matrix $\mathbf{G}$ is constant at each iteration. At each iteration only $\mathbf{b}_k$ is updated (In some works called the image mismatch vector). Then after computation of $\boldsymbol{\xi}^k$ using equation (3.26), a new displacement guess $\boldsymbol{\nu}^k$ is computed for the next iteration step as:

$$\boldsymbol{\nu}^k = \boldsymbol{\nu}^{k-1} + \boldsymbol{\xi}^k. \tag{3.29}$$

The iteration continues until the residual is smaller than a threshold value or for a

fixed number of iterations. At the very first iteration we use $\boldsymbol{\nu}^0 = (0,0)^T$. If k_{max} iterations is required to reach convergence then:

$$\mathbf{v}^L = \boldsymbol{\nu}^{k_max} = \sum_{k=1}^{k_{max}} \boldsymbol{\xi}^k. \tag{3.30}$$

This ends the iterative LK method at a generic level. Then $\mathbf{v}^L$ is fed to the next level as explained before and the algorithm proceeds until the highest resolution level.

3.2.2 Horn-Schunck-Based Methods

Although applying different conditions to solve the optical flow equation, all classical methods of optical flow computation more or less use an objective function similar to the following general form:

$$\begin{aligned} E(\mathbf{v}_x, \mathbf{v}_y) = \sum_{x,y} \Bigg\{ & \phi_D(f_1(x,y) - f_2(x + \mathbf{v}_x(x,y), y + \mathbf{v}_y(x,y))) \\ & + \lambda \Big(\phi_S\big(\mathbf{v}_x(x,y) - \mathbf{v}_x(x+1,y)\big) + \phi_S\big(\mathbf{v}_x(x,y) - \mathbf{v}_x(x,y+1)\big) \\ & + \phi_S\big(\mathbf{v}_y(x,y) - \mathbf{v}_y(x+1,y)\big) + \phi_S\big(\mathbf{v}_y(x,y) - \mathbf{v}_y(x+1,y)\big) \Big) \Bigg\} \end{aligned} \tag{3.31}$$

where $\mathbf{v}_x$ and $\mathbf{v}_y$ are horizontal and vertical component matrices of optical flow field. f_1 and f_2 are images from which optical flow is calculated and λ is a regularization parameter. ϕ is a penalty function that is used for data terms (D) and smoothness terms (S) of the objective function. Two elements of choice are (a) the penalty function and (b) intermediate filtering (which is equivalent to adding more constraint terms in the objective function). These two choices are the main differences between

optical flow computation algorithms proposed during the past years. For instance, quadratic penalty term of $\phi(x) = x^2$ is the one that is used in original Horn-Schunck (HS) method ([12]). Using $\phi(x) = \sqrt{x^2 + \epsilon^2}$ leads to Charbonnier objective function ([3]) or Lorentzian algorithm ([1]) uses $\phi(x) = \log\left(1 + \frac{x^2}{2\sigma^2}\right)$. As said above other contributions include adding more constraint terms, using additional filtering and/or making use of different models such as higher order Markov Random Fields (here in above objective function we assumed a pairwise MRF based on 4-neighborhood).

One of the major contributions in this domain is the work of [31]. The authors make use of a modified version of the (3.31) as following:

$$
\begin{aligned}
E(\mathbf{v}_x, \mathbf{v}_y) = & \sum_{x,y} \Bigg\{ \phi_D(f_1(x,y) - f_2(x + \mathbf{v}_x(x,y), y + \mathbf{v}_y(x,y))) \\
& + \lambda_1 \Big(\phi_S\big(\mathbf{v}_x(x,y) - \mathbf{v}_x(x+1,y)\big) + \phi_S\big(\mathbf{v}_x(x,y) - \mathbf{v}_x(x,y+1)\big) \\
& + \phi_S\big(\mathbf{v}_y(x,y) - \mathbf{v}_y(x+1,y)\big) + \phi_S\big(\mathbf{v}_y(x,y) - \mathbf{v}_y(x+1,y)\big) \Big) \Bigg\} \\
& + \sum_{x,y} \sum_{x',y' \in N_{x,y}} \lambda_3 \Big(|\widehat{\mathbf{v}}_x(x,y) - \widehat{\mathbf{v}}_x(x',y')| + |\widehat{\mathbf{v}}_y(x,y) - \widehat{\mathbf{v}}_y(x',y')| \Big) \\
& + \lambda_2 \Big(\|\mathbf{v}_x - \widehat{\mathbf{v}}_x\|^2 + \|\mathbf{v}_y - \widehat{\mathbf{v}}_y\|^2 \Big)
\end{aligned}
\tag{3.32}
$$

where $\widehat{\mathbf{v}}_x$ and $\widehat{\mathbf{v}}_y$ are auxiliary flow fields, $N_{x,y}$ is the neighborhood (set of neighbors) of pixel at (x, y) in a possibly large area and λ_2 and λ_3 are scalar weights which are set empirically. The newly added terms impose a particular smoothness assumption within a specified region of the auxiliary field $\widehat{\mathbf{v}}_x$ and $\widehat{\mathbf{v}}_y$. In practice [31] optimizes this new objective function by alternately minimizing two parts of the objective function.

The strategy is to hold $\widehat{\mathbf{v}}_x$ and $\widehat{\mathbf{v}}_y$ fixed and minimize the following with respect to $\mathbf{v}_x$ and $\mathbf{v}_y$:

$$\begin{aligned} E_1(\mathbf{v}_x, \mathbf{v}_y) = \sum_{x,y} \Bigg\{ & \phi_D(f_1(x,y) - f_2(x + \mathbf{v}_x(x,y), y + \mathbf{v}_y(x,y))) \\ & + \lambda_1 \Big(\phi_S\big(\mathbf{v}_x(x,y) - \mathbf{v}_x(x+1,y)\big) + \phi_S\big(\mathbf{v}_x(x,y) - \mathbf{v}_x(x,y+1)\big) \\ & + \phi_S\big(\mathbf{v}_y(x,y) - \mathbf{v}_y(x+1,y)\big) + \phi_S\big(\mathbf{v}_y(x,y) - \mathbf{v}_y(x+1,y)\big) \Big) \Bigg\} \\ & + \lambda_2 \Big(\|\mathbf{v}_x - \widehat{\mathbf{v}}_x\|^2 + \|\mathbf{v}_y - \widehat{\mathbf{v}}_y\|^2 \Big). \end{aligned} \tag{3.33}$$

Then with $\mathbf{v}_x$ and $\mathbf{v}_y$ fixed, the following is minimized with respect to $\widehat{\mathbf{v}}_x$ and $\widehat{\mathbf{v}}_y$:

$$\begin{aligned} E_2(\mathbf{v}_x, \mathbf{v}_y) = & \lambda_2 \Big(\|\mathbf{v}_x - \widehat{\mathbf{v}}_x\|^2 + \|\mathbf{v}_y - \widehat{\mathbf{v}}_y\|^2 \Big) \\ & + \sum_{x,y} \sum_{x',y' \in N_{x,y}} \lambda_3 \Big(|\widehat{\mathbf{v}}_x(x,y) - \widehat{\mathbf{v}}_x(x',y')| + |\widehat{\mathbf{v}}_y(x,y) - \widehat{\mathbf{v}}_y(x',y')| \Big) \end{aligned} \tag{3.34}$$

This latter version of optical flow computation with its details and implementation with different penalty functions and parameters, is the first-ranked optical flow computation algorithm in Middlebury ranking of the best published optical flow estimation techniques. Next we will discuss how we can use these state-of-the-art motion compensation techniques to improve the Proposed_VA and Proposed_FBA.

(a) City (b) Jets

(c) Calendar (d) Sailormen

Figure 3.5: A set of four 720×1280 test sequences that are used to compare the results of our algorithm and state-of-the-art de-interlacing algorithm in the VXP® video processor. The sequences are (a) City (b) Jets (c) Calendar (d) Sailormen.

3.2.3 Adding MC to the Proposed Algorithm

In this section, we discuss incorporating the aforementioned motion estimation (optical flow computation) into the proposed algorithms in the previous chapter. Note that we generally discussed the concept of the motion in 3-D signals and explained two of the state-of-the-art algorithms that are widely used in this field. However, these techniques are mostly used in video compression and computer vision applications.

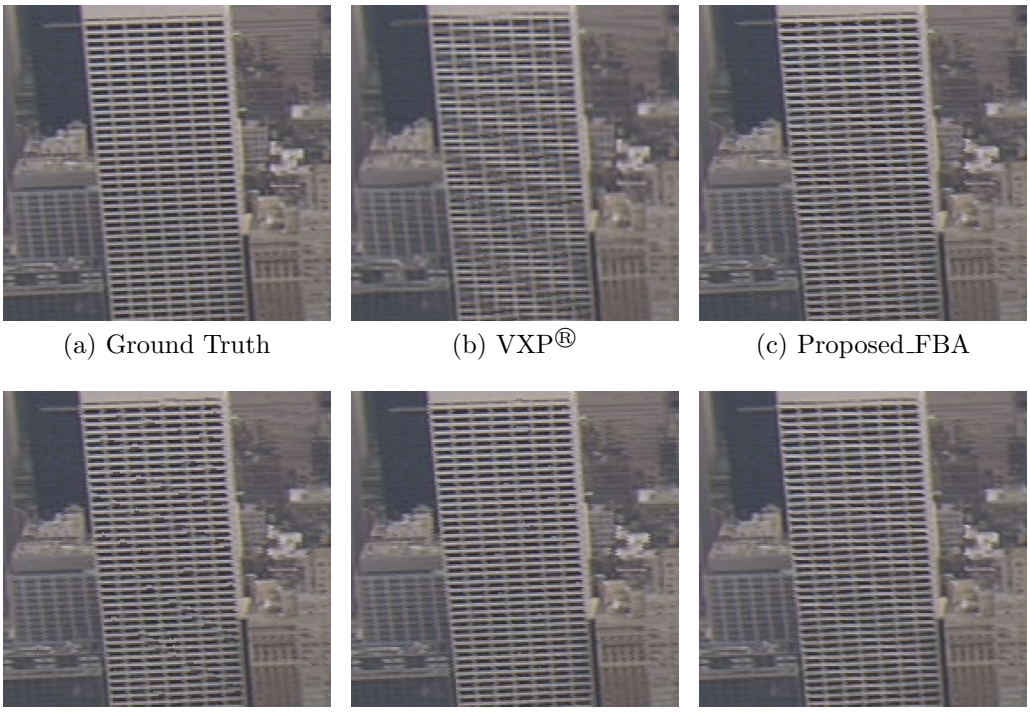
The most important point to consider when using MC techniques in image/video interpolation is that the accuracy and robustness of these methods is considerably reduced. The reason behind this is the huge amount of lost data. For example, for the special case of the de-interlacing problem as explained in the first chapter, for each frame half of data is removed. This ratio increases to three forth of the data in

Table 3.1: PSNR and SSIM results of five different algorithms: method used in VXP®, Proposed_FBA and Proposed_FBA_MC with three different algorithms of Horn-Schunck, Black et al and Lucas-Kanade.

					Proposed_FBA_MC					
	VXP®		**Proposed_FBA**		Horn-Schunck		Black et al		Lucas-Kanade	
	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
City	33.64	0.9858	34.15	0.9872	37.26	0.9863	38.39	0.9865	37.23	0.9923
Jets	41.01	0.9956	43.21	0.9971	43.11	0.9949	44.07	0.9947	43.83	0.9974
Calendar	30.74	0.9754	32.03	0.9790	35.94	0.9787	36.87	0.9788	34.91	0.9869
Sailormen	31.75	0.9765	34.63	0.9863	36.13	0.9821	37.20	0.9817	36.57	0.9903
Average	34.29	0.9833	36.00	0.9874	38.11	0.9855	**39.13**	0.9854	38.14	**0.9917**

case of image interpolation by a factor of two. Consequently, these methods lose their accuracy and robustness considerably. Moreover, interpolation is very sensitive to the accuracy of these methods. As an example the result of a weak motion compensation in a video compression application could be redundant bandwidth required for transmission of the data. However an inaccurate motion estimation in case of interpolation problem produces annoying artifacts perceived by the viewer. This means that some sort of reliability measure should be designed when using motion compensated methods in order to be able to switch to alternative interpolation methods whenever the estimated motion is considered unreliable.

Following we explain our strategy to incorporate motion estimation into our algorithm. First step, we use the proposed algorithm in chapter two (Proposed_FBA) to fill the missing pixels of every two consecutive frames. Then we use these pre-filled values to do the motion estimation procedure. Then having the MC results, we add one more candidate interpolator to the set S as the MC interpolator. Then we redo the trellis processing this time with MC interpolator added to the set S as a new candidate interpolator at each missing pixel. Clearly, the transition matrix should



(a) Ground Truth (b) VXP® (c) Proposed_FBA

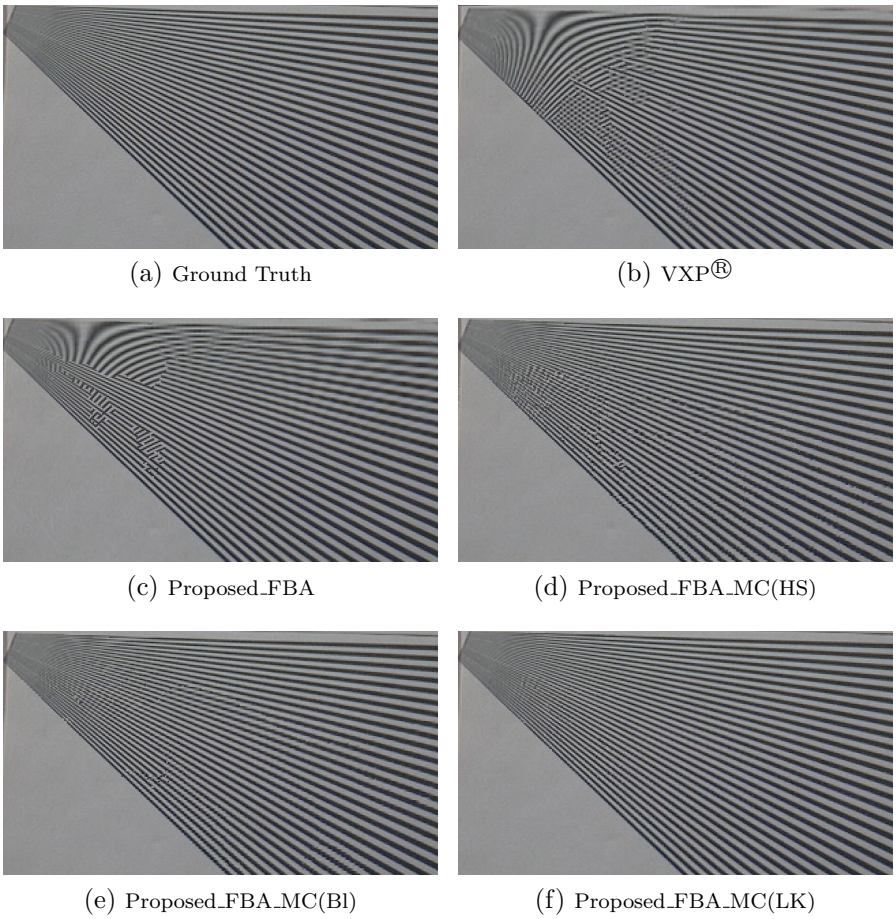
(d) Proposed_FBA_MC(HS) (e) Proposed_FBA_MC(Bl) (f) Proposed_FBA_MC(LK)

Figure 3.6: Zoom in comparison of five different algorithms : VXP®, our proposed method Proposed_FBA, Proposed_FBA_MC(HS), Proposed_FBA_MC(Bl) and Proposed_FBA_MC(LK) for part of the frame 29 of the sequence City.

be updated with this new member of the set S for the second-pass trellis processing (interpolator sequence estimation).

As explained above, different de-interlacing algorithms use a reliability measure to switch between MC interpolation and some other form of spatio-temporal interpolation method. However, in our algorithm the sequence estimation procedure proposed in chapter two, itself, serves as a way to measure the reliability of the MC interpolator. In other words, our algorithm will try to choose the MC method when it is highly probable and switch to other interpolators when it is needed.

Fig. 3.5 shows a test set of four videos used at this stage. The sequences are in 720×1280 and contain highly detailed scenes with large motions between the frames



(a) Ground Truth (b) VXP®

(c) Proposed_FBA (d) Proposed_FBA_MC(HS)

(e) Proposed_FBA_MC(Bl) (f) Proposed_FBA_MC(LK)

Figure 3.7: Zoom in comparison of different algorithms : VXP®, our proposed method Proposed_FBA, Proposed_FBA_MC(HS), Proposed_FBA_MC(Bl) and Proposed_FBA_MC(LK) for part of the frame 22 of the sequence Calendar.

in different forms. For the purpose of aforementioned MC interpolation, we have used three methods of [12], [31] and [2]. [31] is the state-of-the-art optical flow computation ranked first in Middlebury evaluation and [2] is the most cited and accurate pyramidal implementation among all Lucas-Kanade-based optical flow computation methods. For simplicity, we will refer to MC versions of our algorithm as Proposed_FBA_MC(HS) (based on Horn-Schunck method [12]), Proposed_FBA_MC(LK) (based on Lucas-Kanade method [2]) and Proposed_FBA_MC(Bl) (based on Black et

(a) Crowd (b) Town

(c) Tree (d) Joy

Figure 3.8: Set of four 720×1280 test sequences used for evaluation of Proposed_FBA, Proposed_FBA_MC and the other algorithms.

al method [31]). Table 3.1 summarizes the PSNR and MSSIM results for the proposed algorithms and VXP® for the aforementioned test videos. The highest objective results are obtained by Proposed_FBA_MC(Bl). Also note that all the proposed algorithms perform great compared to the MC algorithm used by VXP®.

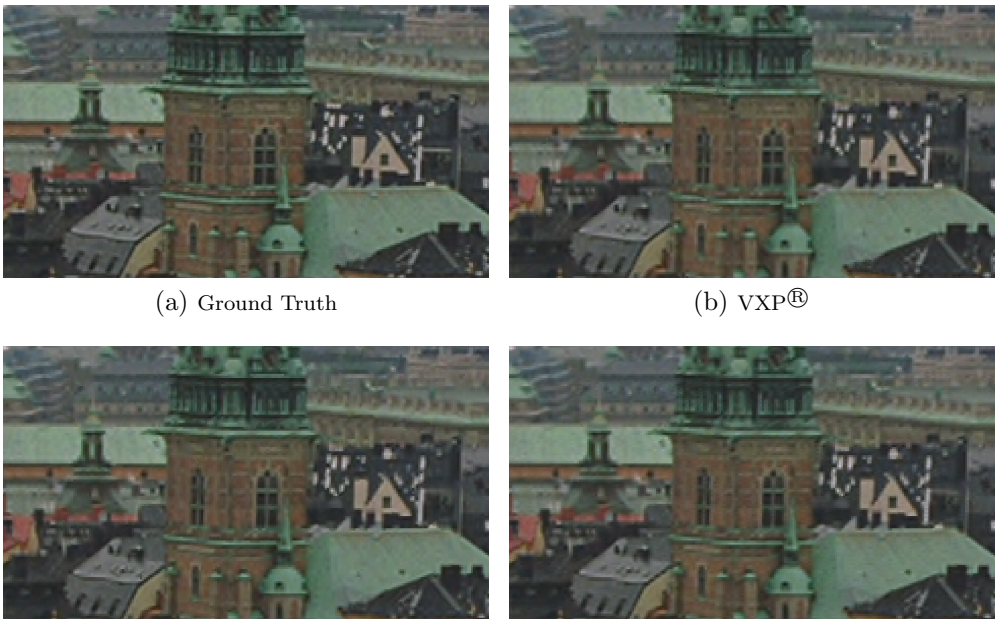
Fig. 3.6 illustrates the visual zoom-in results of five different algorithms for the frame 29 of the sequence City. As can be seen from the figure, the results of the Proposed_FBA is better than VXP® and also comparable to other MC versions of our proposed algorithm. It can be seen clearly that MC interpolation integrated in the second-pass trellis processing greatly improves the visual performance of the algorithm to reproduce the frame with less artifacts. Except for Proposed_FBA_MC(HS), the results of all proposed algorithms (MC and non-MC) are better than results of VXP®.

Table 3.2: PSNR and SSIM results of three different algorithms: method used in VXP®, Proposed_FBA and Proposed_FBA_MC.

	VXP®		**Proposed_FBA**		**Proposed_FBA_MC(Bl)**	
	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
Tree	40.37	0.9970	41.02	0.9980	41.28	0.9951
Crowd	33.49	0.9974	33.64	0.9976	33.71	0.9954
Town	38.18	0.9967	38.86	0.9980	40.28	0.9966
Joy	31.76	0.9960	31.91	0.9961	31.69	0.9912
Average	35.95	0.9968	36.37	**0.9974**	**36.74**	0.9946

Fig. 3.7 is another demonstration of the visual performance of the algorithm. The zoom-in part is one tough challenge for any de-interlacing algorithm. The reason is that the selected part from the frame 22 of the Calendar sequence contains fine details and large amount of motion which is in the form of zooming, rotation and translation simultaneously. These characteristics together makes it difficult for every algorithm to reproduce these parts artifact-free. For this comparison, VXP® performs better than the non-MC version of the proposed algorithm i.e, Proposed_FBA and results in less visual artifacts. On the other hand adding MC to the second pass trellis processing improves Proposed_FBA such that Proposed_FBA_MC(LK) and Proposed_FBA_MC(Bl) perform clearly better than VXP®.

Fig. 3.8 shows the first frame of 1080×1920 test videos. They include faster motion than previous test set and are highly detailed in some parts. Fig. 3.9 is a zoom-in comparison selected from frame 23 of the sequence Town. Both Proposed_FBA and Proposed_FBA_MC(Bl) reproduce the shallow-angled edges more accurately than VXP®. Fig. 3.10 is a part of the frame 57 of Joy sequence. For this sequence, all the

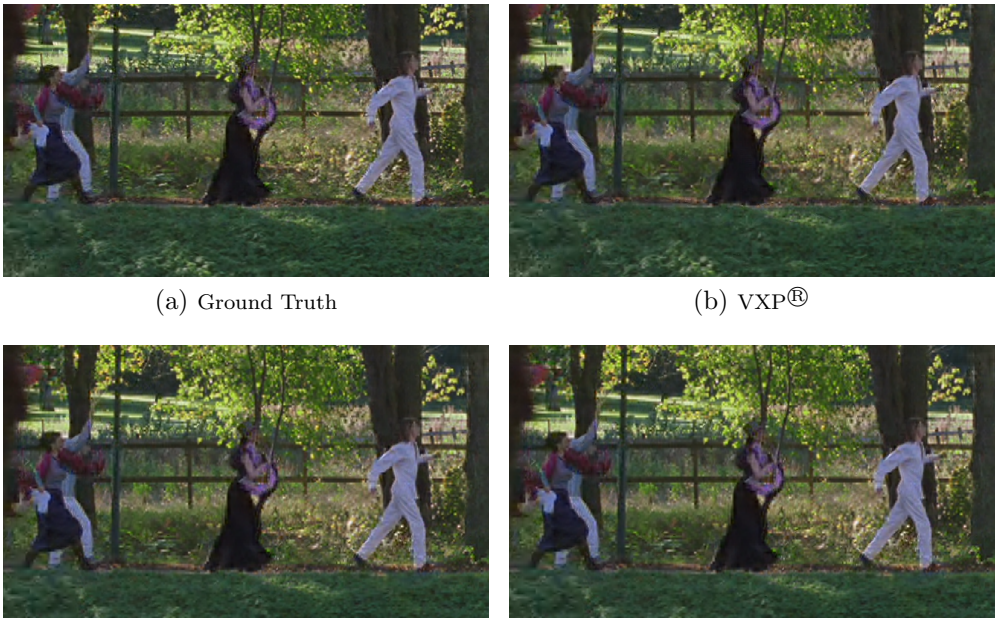


(a) Ground Truth (b) VXP®

(c) Proposed_FBA (d) Proposed_FBA_MC(Bl)

Figure 3.9: Zoom in comparison of different algorithms : VXP®, our proposed method Proposed_FBA and Proposed_FBA_MC(Bl) for part of the frame 23 of the sequence Town.

competitors perform closely and it seems that no major difference can be found between the frames recovered by Proposed_FBA and Proposed_FBA_MC(Bl). To summarize the results for this latter test set, Table 3.2 shows the PSNR and MSSIM obtained by three different algorithms. As can be seen, Proposed_FBA_MC(Bl) achieves the highest overall objective results.



(a) Ground Truth (b) VXP®

(c) Proposed_FBA (d) Proposed_FBA_MC(Bl)

Figure 3.10: Zoom in comparison of different algorithms : VXP®, our proposed method Proposed_FBA and Proposed_FBA_MC(Bl) for part of the frame 57 of the sequence Joy.

Chapter 4

Image Up-scaling

4.1 Introduction

In many display applications, native resolution of the captured image is not sufficient. Consequently, one has to apply a reliable method to faithfully reconstruct the image signal according to the required size and resolution. The image resolution up-conversion or simply image up-scaling methods have been developed over the past decades to meet this demand for higher resolutions. Up-scaling methods (Fig. 4.1) use the low resolution (LR) image to recover the high-frequency components and fine details of a high resolution (HR) image. Unreliability of the estimation of edges and details from LR images causes annoying artifacts in the reconstructed images. Different methods ([16], [21], [38], [29], [24]) have been proposed in this classical but still active field of image processing i.e., image up-scaling. The method in [21] uses a switching method between bilinear interpolation and covariance-based interpolation. [38] uses observation sets in two orthogonal directions as two noisy measurements of the missing pixel to be interpolated. These measurements are then fused by the

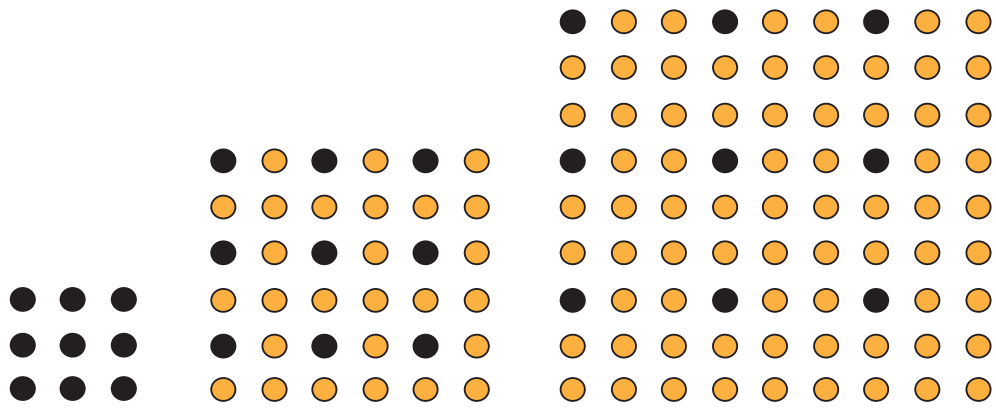
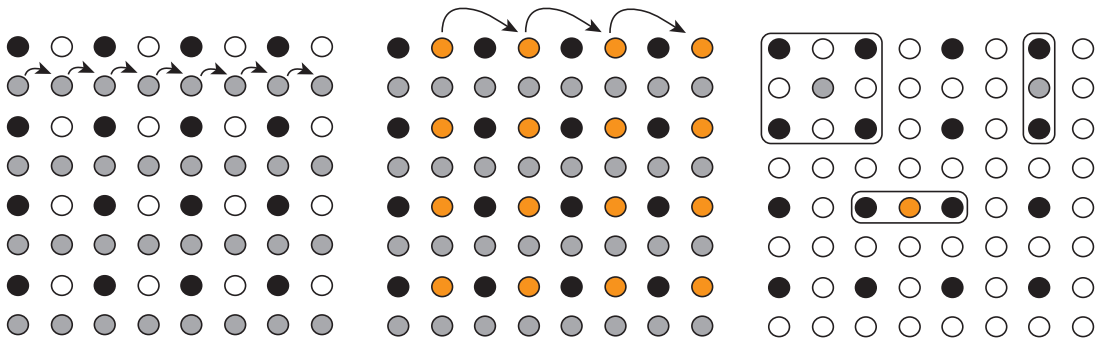


Figure 4.1: Up-scaling an image is the process of converting a low resolution image to a higher resolution image. This requires interpolating the missing pixels of the higher resolution grid. (a) A low resolution image (b) A higher resolution image by a factor of two (c) by a factor of three.

means of LMMSE technique into a final reliable estimation of the missing pixel. The method in [29], exploits the fact that the image is highly compressible in the wavelet domain and uses the compressive sensing theory to built a relatively accurate estimate of the high resolution image. In [24], the authors use adaptive directional image interpolations which are computed over a wavelet frame with an $O(NlogN)$ algorithm.

In previous chapters we explained how we can exploit sequence estimation techniques along with a predefined set of candidate interpolators to recover the missing lines of an interlaced frame using the original pixels of the current frame as well as the past and future frames. In fact, de-interlacing is a special problem within interpolation problems. Consequently one may think of applying the general idea developed in the previous chapters to some other classical interpolation problems. This, in fact, is the main topic of interest in this chapter.



(a) first pass estimation (b) second pass estimation (c) first/second pass neighbors

Figure 4.2: (a) First pass interpolation and the original pixels (dark gray) used to interpolate the first pass pixels (light gray). (b) Second pass interpolation and the pixels reconstructed in this pass (light red) (c) Three different neighborhoods used in the proposed algorithm. Two types are used at first pass and the third type for a typical to-be-interpolated pixel (light red) in the second pass.

4.2 Theory and Algorithm Description

For simplicity let us assume the following conventions for the notations used. Vertical and horizontal directions will be denoted by x and y respectively as previous chapters. Therefore, a typical pixel position in a digital image is denoted by (x, y) or simply the vector $\vec{r} = (x, y)$. At a typical pixel within an image, $f(\vec{r})$ or $f(x, y)$ refers to the gray-level value or gray-level intensity of the 2-D image signal at that position. If needed, h and w will be used to denote the hight and width of an image respectively. Also we may use the boldface letters with subscript and superscript to denote the sequences and their start and end points respectively. For instance $\overline{\mathbf{X}}_i^j$ denotes the sequence $\overline{\mathbf{X}}$ from the i^{th} to j^{th} position in the sequence. As another example, $\overline{\mathbf{X}}_i$ shows the value of the sequence $\overline{\mathbf{X}}$ at the i^{th} position. The rest of the notations will be introduced whenever they are used.

4.2.1 Overall Approach to The Problem

Our final goal in image resolution up-conversion is to recover the high-resolution (HR) image from the low-resolution (LR) image given. Now consider that we are given the LR image. In the proposed algorithm, we will interpolate the entire missing pixels in two consecutive passes. Each group of missing pixels recovered in each of the passes are shown in Fig. 4.2. In our algorithm we scan through the missing pixels of a row from left to right and estimate the corresponding sequence of interpolators that best fits the sequence of missing pixels and we finally interpolate these missing pixels in accord with the corresponding estimated sequence of interpolators. Clearly, sequence estimation of the rows are independent from each other and can be done in parallel.

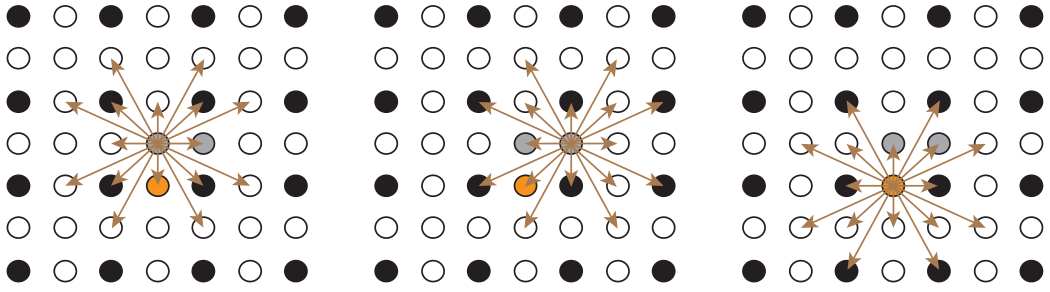
We define for each missing pixel in HR image a neighborhood consisting of a few nearest neighboring LR (original) pixels. Fig. 4.2 shows the different neighborhoods used in the proposed algorithm around a missing pixel. We denote the sequence of these neighborhoods associated with a sequence of missing pixels as $\overline{\mathbf{N}}$.

As mentioned before, we use a set of candidate interpolators at every position. We denote this set by S. These interpolators are states of the Markov process we discussed earlier. Although one can come up with different interpolators as the members of this set, we suggest a set of directional interpolators. Further improvements may be obtained by adding more interpolators to the set S with at the cost of more time-complexity of the algorithm. We will discuss in section 4.3 that the complexity of our algorithm grows quadratically with the number of the states of the Markovian process i.e., cardinality of the set S or the number of candidate interpolators. We have chosen the following set of eight interpolators as the members of the set S:

$$
\begin{aligned}
I_1(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-\vec{u}_x)+f(\vec{r}+\vec{u}_x)\big) \\
I_2(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-\vec{u}_y)+f(\vec{r}+\vec{u}_y)\big) \\
I_3(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-\vec{u}_x+\vec{u}_y)+f(\vec{r}+\vec{u}_x-\vec{u}_y)\big) \\
I_4(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-\vec{u}_x-\vec{u}_y)+f(\vec{r}+\vec{u}_x+\vec{u}_y)\big) \\
I_5(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-\vec{u}_x+2\vec{u}_y)+f(\vec{r}+\vec{u}_x-2\vec{u}_y)\big) \\
I_6(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-\vec{u}_x-2\vec{u}_y)+f(\vec{r}+\vec{u}_x+2\vec{u}_y)\big) \\
I_7(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-2\vec{u}_x+\vec{u}_y)+f(\vec{r}+2\vec{u}_x-\vec{u}_y)\big) \\
I_8(\vec{r}) &= \frac{1}{2}\cdot\big(f(\vec{r}-2\vec{u}_x-\vec{u}_y)+f(\vec{r}+2\vec{u}_x+\vec{u}_y)\big)
\end{aligned}
\tag{4.1}
$$

where $\vec{u}_x$ and $\vec{u}_y$ are unit vectors in vertical and horizontal directions in the HR image, respectively. Also Fig. 4.3 illustrates the members of the set S on three different typical positions which are at even horizontal coordinates (first pass), odd horizontal coordinates (first pass) and finally at a typical position of the second pass respectively. As depicted in Fig. 4.3, the estimated interpolator may point to a missing position. The number of interpolators which point to a missing position depends on where the pixel is located within the image as illustrated in parts (a)-(c) of Fig. 4.3. If at any point within the estimated sequence the estimated interpolator points to missing pixels of the frame, we simply use the cubic spline interpolation to pre-fill the missing positions.

Let us assume that the best estimation of the sequence of interpolators of a sequence of missing pixels, $\bar{\mathbf{I}}_{opt}$, is the sequence of interpolators $\bar{\mathbf{I}}$ which maximizes the



(a) (b) (c)

Figure 4.3: (a) Interpolators at even horizontal coordinates (first pass). (b) Interpolators at odd horizontal coordinates (first pass) (c) Interpolators (second pass). The estimated interpolator may finally point to missing positions. The number of interpolators which may point to missing positions depends on whether the pixel is at the first pass or second, has odd or even horizontal coordinates.

probability $\Pr(\bar{\mathbf{I}}|\overline{\mathbf{N}})$. In other words, the best estimation of the sequence of interpolators $\bar{\mathbf{I}}$, is the most probable one given the sequence of neighborhoods of LR pixels. Maximizing $\Pr(\bar{\mathbf{I}}|\overline{\mathbf{N}})$ is equivalent to maximizing $\Pr(\overline{\mathbf{N}}|\bar{\mathbf{I}}) \cdot \Pr(\bar{\mathbf{I}})$:

$$\bar{\mathbf{I}}_{opt} = \underset{\bar{\mathbf{I}} \in S^w}{\operatorname{argmax}} \Big\{ \Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big) \Big\}. \tag{4.2}$$

So far the up-scaling problem has been translated to the optimization problem in (4.2). The next step is to rearrange the above equation with the help of some reasonable assumptions to a proper form for solving the optimization problem. For simplicity, we consider the formulation for the first pass shown in Fig. 4.2. We assume a first-order Markov chain model over the sequence of interpolators. Therefore, (4.2) can be rewritten as:

$$\Pr\big(\bar{\mathbf{I}}\big) = \Pr\big(\bar{\mathbf{I}}_0\big) \cdot \Pr\big(\bar{\mathbf{I}}_1|\bar{\mathbf{I}}_0\big) \cdot \prod_{y} \Pr\big(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}\big). \tag{4.3}$$

In our model the first-order Markov process changes its state from one position to the

next one with observing each neighborhood. Provided that successive neighborhood observations are independent, the conditional probabilities of the neighborhoods are independent as well. Therefore:

$$\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) = \prod_{y} \Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big). \tag{4.4}$$

According to (4.3) and (4.4), the probability in (4.2) can be expressed as:

$$\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big) \propto \Pr\big(\bar{\mathbf{I}}_1|\bar{\mathbf{I}}_0\big) \cdot \prod_{y} \Pr\big(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}\big) \cdot \prod_{y} \Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big). \tag{4.5}$$

We have to choose the $\bar{\mathbf{I}}_0$ arbitrarily since it does not have any actual corresponding pixel position. Taking the logarithm and then negating on both sides of (4.5) we have:

$$-\log\Big(\Pr\big(\overline{\mathbf{N}}|\bar{\mathbf{I}}\big) \cdot \Pr\big(\bar{\mathbf{I}}\big)\Big) \propto \sum_{y} \log\left(\frac{1}{\Pr\big(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}\big)}\right) + \sum_{y} \log\left(\frac{1}{\Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big)}\right). \tag{4.6}$$

Maximizing (4.2) is equivalent to minimizing (4.6). Therefore to obtain $\bar{\mathbf{I}}_{opt}$ we have to equivalently solve the following optimization problem:

$$\bar{\mathbf{I}}_{opt} = \underset{\bar{\mathbf{I}} \in S^w}{\operatorname{argmin}} \left\{ \sum_{y} \log\left(\frac{1}{\Pr\big(\bar{\mathbf{I}}_y|\bar{\mathbf{I}}_{y-1}\big)}\right) + \sum_{y} \log\left(\frac{1}{\Pr\big(\overline{\mathbf{N}}_y|\bar{\mathbf{I}}_y\big)}\right) \right\}. \tag{4.7}$$

The number of possible sequence of interpolators which can be examined in equation

(4.7) depends on the size of the image and cardinality of the S. For example for the scanning method described earlier, $|S| = 8$ as in (4.1) and $h \times w = 512 \times 512$ the size of the search space is approximately 8^{512}. The structure of the summation in (4.7) over horizontal positions could be properly utilized to perform the optimization much more efficiently to obtain the global optimal sequence of interpolators. Now consider the interpolator transition between two adjacent missing pixels as a transition between two consecutive layers of a trellis having pre-defined interpolators as its states on each layer. We expand this trellis through a missing row. We choose the branch metrics to be logarithmic terms in (4.7). According to (4.7), minimum-weight path through the trellis corresponds to the best estimation of the sequence of the interpolators associated with the corresponding sequence of missing pixels. The problem of finding the minimum-weight path can be solved efficiently (globally and fast) using Viterbi algorithm (Viterbi path). Among the edges (interpolator transitions) converging to a same state (interpolator) at each layer of the trellis, the one with less accumulated weight will survive and new accumulated weight is saved only for that survival path. Thus for each layer of the trellis we have to save only $|S|$ accumulated weights as we have only $|S|$ survival paths. When reached last layer of the trellis (in our implementation last layer corresponds to the last horizontal position within a missing row of an image) simply we backtrack from the least-accumulated-weight state at the last layer through the trellis to the first layer via survival paths for each state at each layer. This unique path corresponds to the optimal sequence of interpolators. This trellis-represented algorithm while having a much lower complexity than many state-of-the-art algorithms in this field, shows a great performance with a variety of test images. We will discuss its objective/visual performance as well as its

time complexity in section 4.3.

4.2.2 Markov Chain Transition Model

For notation simplicity let us assume here that $\bar{\mathbf{I}}_y = I_i$, $\bar{\mathbf{I}}_{y-1} = I_j$ and $\overline{\mathbf{N}}_y = N(\vec{r})$. So we will focus on deriving approximations for $\Pr(I_i|I_j)$ and $\Pr(N(\vec{r})|I_i)$ required in (4.7). For this aim we perform the following stages. First, we define the set S_{LR} which is the LR dual of the set S. The members of the set S_{LR} are in fact dilated versions of the members of the set S by a factor of two in each spatial direction. The members of the set S_{LR} are defined exactly as defined in the second chapter. Consider at a missing pixel position we are interested to define the probability of the corresponding neighborhood given that we want to apply a specific member of the set S i.e., $\Pr(N(\vec{r})|I_i)$. Our strategy is to first define an expression for the cost of applying the interpolator I_i at the position $\vec{r}$. For now assume that we are able to define an approximate for this cost namely $C(\vec{r}, I_i)$. Then we propose that probability $\Pr(N(\vec{r})|I_i)$ and the cost $C(\vec{r}, I_i)$ are reciprocally proportional. In other words, the higher the cost of applying an interpolator with a given neighborhood around a missing pixel position, the smaller the probability of that neighborhood given we apply that interpolator. Therefore:

$$\Pr(N(\vec{r})|I_i) = \Big(V\big(C(\vec{r}, I_i)\big)\Big)^{-1} \tag{4.8}$$

where $V(u)$ is an increasing potential function of u. This function can take different forms such as exponential ($V(u) = k\exp\{u\}$) or linear ($V(u) = ku$) with k being the normalization factor. Now the remaining step is to define the cost $C(\vec{r}, I_i)$ and potential function $V(u)$. Although defining an exponential cost function may be a

better choice, however for simplicity as well as to avoid introducing parameters into the algorithm and consequently the difficulty of tuning those parameters, we simply use the linear function. We base the cost of an interpolator at a missing pixel position on the difference between the value of the pixel intensity at that position and the value that results from applying that interpolator in that position. This means that if we want to measure the aforementioned cost for a member of the set S, say I_i, at missing position $\vec{r}$, we apply the I_i at $\vec{r}$ to get the interpolation value. Then we compare the result of interpolation by I_i with the pixel intensity at $\vec{r}$ i.e., $f(\vec{r})$. The smaller this difference, the smaller the cost of I_i at $\vec{r}$. The problem with such definition is that at position $\vec{r}$, we have the interpolation value for each of the members of the set S defined in (4.1) but not the actual value of the pixel since the pixel at $\vec{r}$ is missing. Therefore the aforementioned difference (cost) can not be computed. Similarly this difference can not be computed at a neighbor of the to-be-interpolated pixel because the interpolation value for the members of the set S in (4.1) can not be computed despite that we have the actual pixel value at this position. To overcome this problem we have to find a reasonable approximation for this cost and this is where the definition of the LR dual of the set S comes in. At a missing pixel $\vec{r}$ consider the aforementioned difference using I_{LR_i} instead of I_i and at the original neighboring pixel of $\vec{r}$ instead of $\vec{r}$ itself. We propose that this cost denoted by $C(\vec{r'}, I_{LR_i})$ when averaged over the neighborhood ($\vec{r'} \in N(\vec{r})$), is a good approximation of the $C(\vec{r}, I_i)$. In other words:

$$C(\vec{r}, I_i) \approx \frac{1}{|N(\vec{r})|} \sum_{r' \in N(\vec{r})} C(\vec{r'}, I_{LR_i}) = \frac{1}{|N(\vec{r})|} \sum_{r' \in N(\vec{r})} \left\| f(\vec{r'}) - I_{LR_i}(\vec{r'}) \right\|_{\ell_2} \tag{4.9}$$

where we have used the ℓ_2-norm. Now we define probabilities $\Pr(I_i|I_j)$ that are used in (4.7). Consider the LR image. Then for all pixels in this low resolution image, we calculate the cost defined in (4.9) and choose the best interpolation I^* as the one with the minimum cost or:

$$\mathrm{I}^* = \underset{i}{\operatorname{argmin}} \left\{ C(\vec{r}, I_i) \right\}. \tag{4.10}$$

Note that this a hard-decision minimization at each pixel of the LR image. The result is a/an *state/interpolator matrix* of the same size as the LR image with entries from the set S. This is because the LR image is complete and aforementioned discussion about approximating $C(\vec{r}, I_i)$ by averaging $C(\vec{r}, I_{LR_i})$ over $N(\vec{r})$ is not necessary. Based on this state matrix we form the *transition matrix* (transition matrix for the Markov chain discussed earlier) for low resolution LR image (TM_{LR}) and compute the transition probabilities $\Pr(I_i|I_j)$ as:

$$\Pr(I_i|I_j) = \mathrm{TM}_{LR}(i,j) = \frac{num(I_j \rightarrow I_i)}{num(I_j)}. \tag{4.11}$$

where $num(X)$ is the number of occurrence of a state (interpolator) X and $num(X \rightarrow Y)$ is the number of transitions from interpolator X to interpolator Y respectively in the state/interpolator matrix. Note that by $\Pr(I_i|I_j)$ we mean the probability of interpolator I_i occurs right to the interpolator I_j in the state matrix mentioned above when scanning left to right. For the reason discussed earlier this matrix can not be computed directly for the HR image, however, it makes a lot of sense that the distribution of the introduced state matrix for the LR image be approximately similar to the state matrix of the HR image if we had the complete HR image. This is

a reasonable assumption considering that the LR images contain not all but most of features of the corresponding HR images at a lower scale. As an example, if we have a 45-degree edge in the HR image, with a high probability we have approximately the same edge pattern at the corresponding positions in the LR image. With this assumption transition matrix for the HR image i.e, TM is approximated by transition matrix obtained by (4.11). According to (4.11) and (4.9), (4.6) can be rewritten as:

$$\bar{\mathbf{I}}_{opt} = \underset{\bar{\mathbf{I}} \in S^w}{\operatorname{argmin}} \left\{ \sum_{y} \log \left(\frac{1}{\mathrm{TM}(l,k)} \cdot \frac{1}{\min\left\{ \frac{1}{C(\vec{r}, \bar{\mathbf{I}}_y)}, 1 \right\}} \right) \right\}. \tag{4.12}$$

where $l = \pi(\bar{\mathbf{I}}_{y-1})$ and $k = \pi(\bar{\mathbf{I}}_y)$ and π is the simple index assignment which was also used in previous chapters. Now that we have formulated the first pass shown in Fig. 4.2, for the second pass we exactly follow the same procedure this time substituting y with y_{even} in the previous equations. This is clearly illustrated in Fig. 4.2. As depicted, the step size is twice the first pass. Note that we classified the to-be-interpolated pixels in first and second pass only for distinguishing between the neighborhoods we use for each passes. However as discussed before, the sequence estimations of the first pass and second pass are independent of each other and can be done in parallel.

4.3 Experimental Results

In this section we present some visual validations of our algorithm and compare it with some other algorithms which are among the best in this area. Fig. 4.4 shows some of 320×480 images which have been chosen randomly from the Berkeley image database

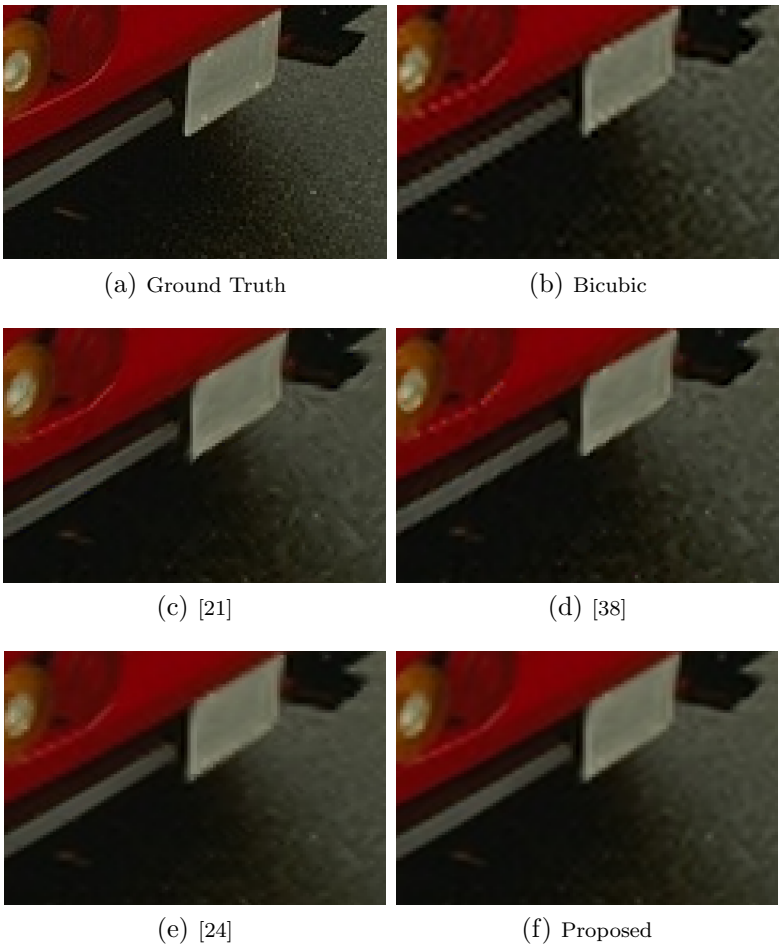
(a) Starfish (b) Wall Art (c) Church (d) Samurai (e) Swan

(f) Louver (g) Aviator (h) Flag (i) Car (j) Zebra

Figure 4.4: Some of the 320×480 images in the test set that are used for evaluating the proposed algorithm. This set is randomly selected from Berkeley image database.

[1]. The randomly selected images contain diverse challenging image characteristics for evaluation of the proposed algorithm. The selected images are rich in terms of high frequency details and especially near vertical and horizontal edges which are among the hardest features to recover accurately for every interpolation algorithm. Also in this section we compare our method of interpolation with other algorithms of [16], [38], [29] and [24] which are well-known in this field especially the last method. For the objective evaluation we make use of the two most popular measures for image quality assessment, peak signal-to-noise ratio (PSNR) and mean structural similarity (MSSIM) as introduced in [35]. Some of these competing methods such as [24] are implemented by the authors on gray-scale images. Therefore, to be able to compare all the algorithms with each other we apply the algorithms on each color channel (red, green and blue) of the input color images and then we compute the PSNR of the reproduced color image. For the SSIM we separately compute the SSIM for each of the three channels and then take the average on three channels as the resulting SSIM of an algorithm. Clearly we use the implementations of the authors with some

[1]http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/



(a) Ground Truth (b) Bicubic

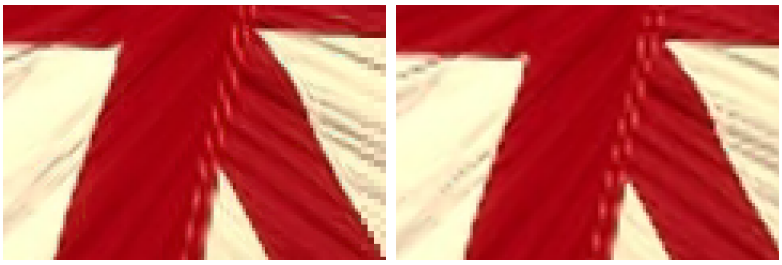
(c) [21] (d) [38]

(e) [24] (f) Proposed

Figure 4.5: Visual comparison of the proposed algorithms with four different algorithms. The zoom-in comparison is taken from the image Car.

modifications to make them compatible on color images. Also we make use of exactly the same images for all algorithms and maintain the same conditions and formulas when calculating the PSNR and MSSIM. Fig. 4.5 shows a zoom-in comparison of the different algorithms. The zoom-in results are taken from the image Car. As can be seen in the figure, the proposed algorithm recovers sharper and more accurate edge details than the other algorithms. The same results can also be seen in Fig. 4.6 which is another demonstration of the proposed algorithm. The figure shows the

(a) Ground Truth (b) Bicubic



(c) [21] (d) [38]

(e) [24] (f) Proposed

Figure 4.6: Visual comparison of the proposed algorithms with four different algorithms. The zoom-in comparison is taken from the image Flag.

result of interpolation by different algorithms for the image Flag. Again we have taken a part of the image for comparing different algorithms more accurately. The proposed method and the method of [24] recover the image more precisely than the other competitors especially around the boundaries. The reason that we emphasize on edges is that these high-frequency features are among the hardest parts of a typical image to recover. Distortion in these parts can be easily and annoyingly perceived by the viewer.

Table 4.1: PSNR and MSSIM results of different algorithms on the test set of Fig. 4.4.

	[16]		**[29]**		**[38]**		**[24]**		**Proposed**	
	PSNR	MSSIM	PSNR	SSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
Starfish	29.35	0.8812	28.69	0.8616	29.20	0.8680	29.54	0.8835	29.48	0.8851
Wall art	24.15	0.7585	24.64	0.7604	24.70	0.7648	24.83	0.7761	25.61	0.8132
Church	29.88	0.9241	30.84	0.9357	30.91	0.9188	31.17	0.9377	31.56	0.9464
Samurai	25.65	0.8601	26.23	0.8668	26.31	0.8575	26.81	0.8832	27.27	0.8913
Swan	29.75	0.8923	29.81	0.8872	30.42	0.8908	30.29	0.8990	31.09	0.9194
Louver	24.29	0.8310	23.94	0.8165	23.89	0.8063	24.57	0.8405	24.62	0.8479
Aviator	25.13	0.8508	25.90	0.8658	25.84	0.8572	26.33	0.8757	26.69	0.8927
Flag	24.55	0.7890	25.15	0.7965	25.18	0.7926	25.32	0.8055	26.08	0.8297
Car	27.32	0.7616	28.20	0.7860	28.06	0.7763	28.47	0.7915	29.11	0.8230
Zebra	25.54	0.8657	25.73	0.8604	25.84	0.8548	26.33	0.8756	26.34	0.8772
Average	26.56	0.8414	26.87	0.8437	27.04	0.8387	27.37	0.8568	27.78	0.8726

Table. 4.1 summarizes the PSNR and MSSIM results for a few of the aforementioned algorithms ([16], [29], [38], [24]) compared to the proposed algorithm. As can be seen from the table, the results are better than/comparable with state-of-the-art method of [24].

Fig. 4.7 shows another set of test images in 480×320 for further comparing the aforementioned algorithms. This set has been selected from the same source and contains challenging parts which are difficult to recover for every algorithm. In Fig. 4.8 we have included a zoom-in comparison of the aforementioned algorithms this time for one of the images (Girl) in the second test set. The figure illustrates the results for a part of the image. Visual comparison of the results verifies the comparable performance of our algorithm with the method [24] specifically around the edges. Also Fig. 4.9 shows the same results for part of the image Wall art. The method of [24] and our method are superior compared to the other algorithms in interpolating the lost data. Furthermore, the proposed method of this paper seems to be more successful

(a) Stairs (b) Man (c) Bird (d) Woman (e) Jump (f) Deer (g) Spear

(h) Hindu (i) House (j) Model (k) Tree (l) Girl (m) Eiffel (n) Beach

Figure 4.7: Some of the 480×320 images in the test set that are used for evaluating the proposed algorithm. This set is randomly selected from the aforementioned database.

in recovering the part of the image shown in Fig. 4.9 than the method in [24].

Table. 4.2 summarizes results of the algorithms on the latter test set. Table. 4.2 confirms the comparability of the proposed algorithm with the competitors which are among the well-known methods in the field of image interpolation.

Fig. 4.10 is the final visual comparison of the proposed algorithm versus the competitors. Clearly, the proposed method reproduces the image more accurately than [24] for this image (Hindu).

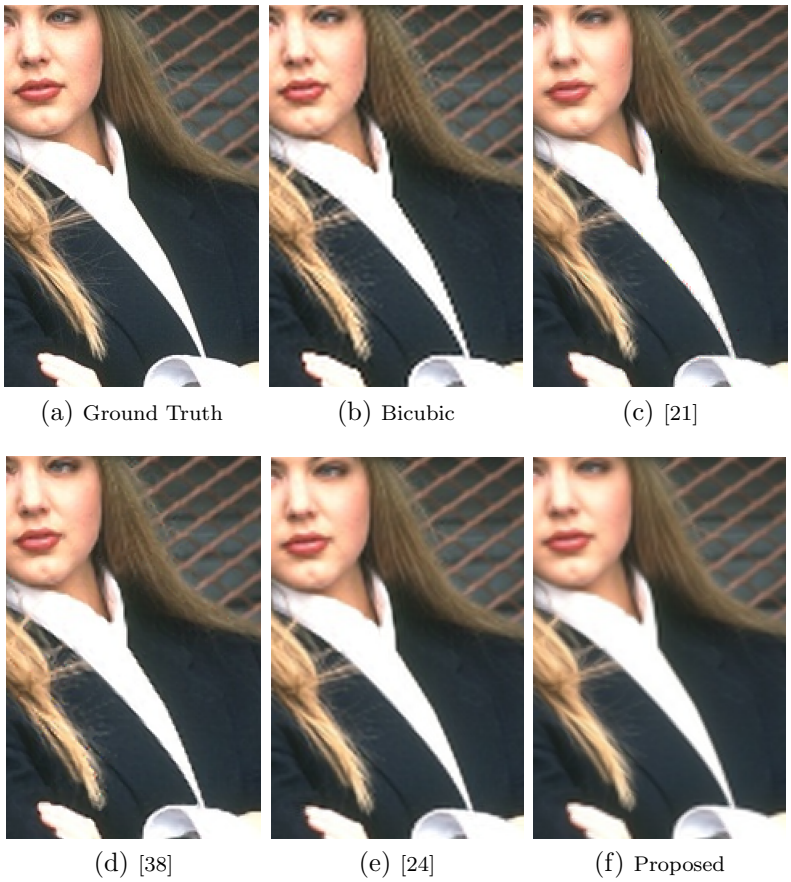
As discussed in section 4.2.1, the proposed method is based on estimating the sequence of interpolators using Viterbi path on a trellis diagram representing the algorithm. The number of calculations is directly proportional to the length of the trellis (which is directly proportional to the width of the image w) and the number of scanned rows (which is directly proportional to the height of the image h). $|S|$ is the number of the states of the Markov process (number of interpolators) used at each

Table 4.2: PSNR and MSSIM results for different algorithms on the set 4.7.

	[16]		**[29]**		**[38]**		**[24]**		**Proposed**	
	PSNR	MSSIM	PSNR	SSIM	PSNR	MSSIM	PSNR	MSSIM	PSNR	MSSIM
Stairs	27.31	0.8732	27.75	0.8664	27.72	0.8664	28.30	0.8846	28.45	0.8949
Man	33.83	0.8991	34.22	0.9049	34.62	0.8886	34.97	0.9096	34.84	0.9177
Bird	33.44	0.9384	33.98	0.9408	34.05	0.9257	34.34	0.9454	35.43	0.9448
Woman	25.82	0.7438	26.16	0.7405	26.21	0.7428	26.63	0.7667	27.34	0.7916
Jump	26.72	0.8543	27.03	0.8543	27.11	0.8517	27.32	0.8645	27.89	0.8762
Deer	25.64	0.7801	25.92	0.7776	25.79	0.7650	26.13	0.7920	25.23	0.7399
Spear	30.03	0.9073	30.38	0.9055	30.42	0.8990	30.86	0.9162	30.88	0.9211
Hindu	32.83	0.9294	33.27	0.9283	33.23	0.9225	33.68	0.9366	34.10	0.9416
House	22.58	0.7969	22.47	0.7918	22.83	0.7881	22.98	0.8130	24.64	0.8532
Model	27.03	0.8930	27.52	0.8992	27.67	0.8844	27.80	0.9055	28.19	0.9118
Tree	22.37	0.7273	22.44	0.7136	22.61	0.7196	22.72	0.7390	24.03	0.7836
Girl	30.91	0.9108	31.61	0.9148	31.91	0.9116	32.04	0.9205	31.99	0.9263
Eiffel	26.89	0.8620	27.16	0.8651	27.22	0.8571	27.42	0.8712	28.76	0.9018
Beach	26.01	0.8975	26.46	0.8943	26.26	0.8895	27.12	0.9102	27.21	0.9108
Average	28.17	0.8581	28.31	0.8569	28.40	0.8509	28.74	0.8696	29.21	0.8797

layer. At each layer we consider transition between any two states at the previous layer and the current layer (Fig. 2.5). Consequently, the number of calculations is proportional to $|S|^2$. Therefore, scanning through the entire trellis requires total number of $|S|^2 \times w \times h$ calculations. In other words the timing complexity of our algorithm is $\theta\big(|S|^2 wh\big)$. The complexity is linear in terms of w but quadratic in terms of the size of the set S. However, in practice we have implemented our algorithm with a fixed set of candidate interpolators S as introduced in (4.1) so the $|S|$ is constant. Also as discussed earlier the sequence estimations for different rows are independent and the algorithm can be greatly sped up by the use of parallel processing.

To compare the timing efficiency of the aforementioned algorithms especially with the close competitor in [24], Table. 4.3 and Table. 4.4 show the time (minutes)



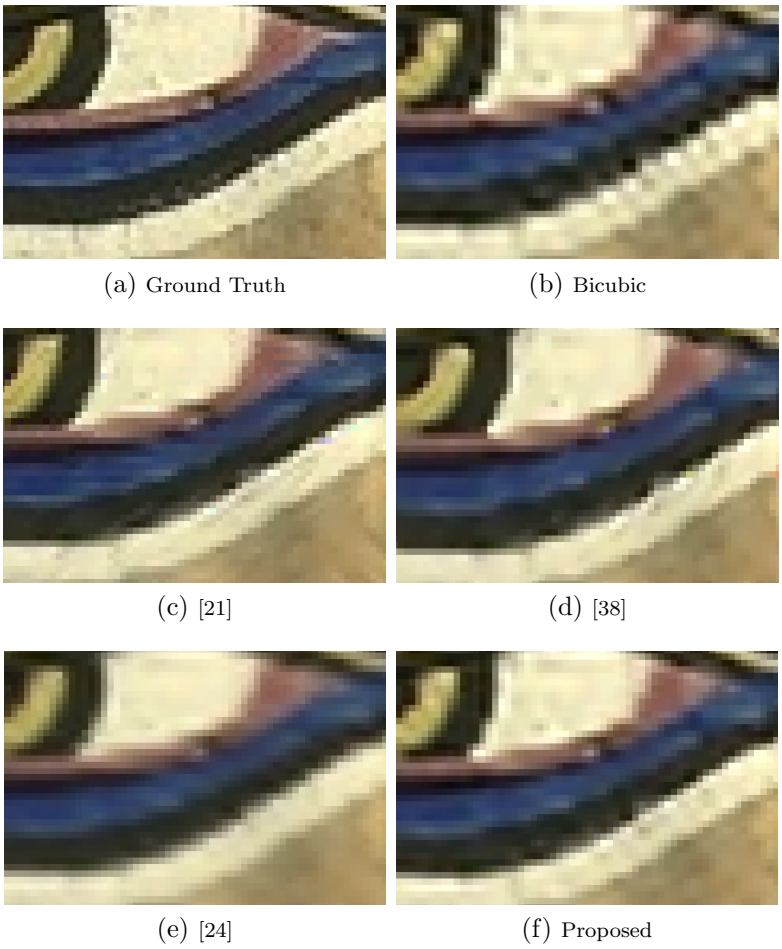
(a) Ground Truth (b) Bicubic (c) [21]

(d) [38] (e) [24] (f) Proposed

Figure 4.8: Visual comparison of the proposed algorithms with five different algorithms. The zoom-in comparison is taken from the image Girl.

consumed by our method and the method of [24] with different images. We would like to clarify that to make a fair comparison of timing for different algorithms: (a) we have used the exact implementations by the authors except for the slight changes we made to make them executable on color images (b) all the codes are scripted in MATLAB and without using any parallel processing instructions. Table. 4.3 and Table. 4.4 summarize the results for the first and second test sets, respectively.

In the proposed algorithm we have used the cost in (4.9) which uses the ℓ_2-norm. However, so far we have not used the color information in the cost formula directly.



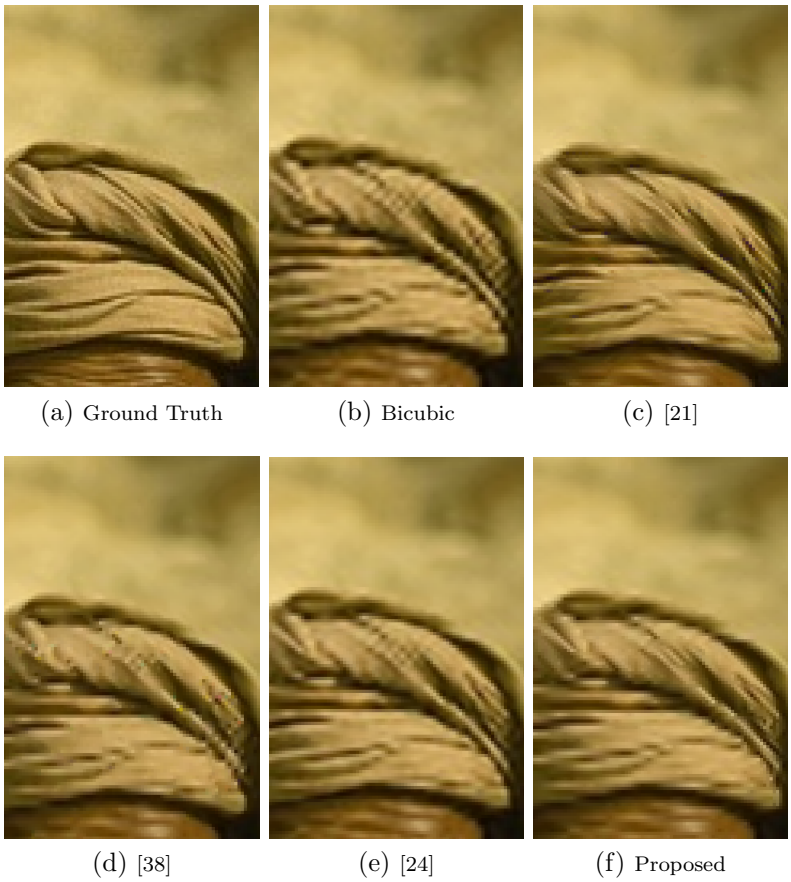
(a) Ground Truth (b) Bicubic

(c) [21] (d) [38]

(e) [24] (f) Proposed

Figure 4.9: Visual comparison of the proposed algorithms with five different algorithms. The zoom-in comparison is taken from the image Wall Art.

In other words we have implemented the algorithm for each color channel separately. Here we discuss the benefit of using the color information in the cost in (4.9). We propose that we can use the values of all the three channels in the cost (4.9) when estimating the sequence of interpolators and apply the estimated sequence to the all three channels without giving up the PSNR or MSSIM obtained by previous approach. Consequently, by making the cost computation a little more time consuming, we avoid repeating the algorithm for different channels. Therefore, we roughly get a speed up



(a) Ground Truth (b) Bicubic (c) [21]

(d) [38] (e) [24] (f) Proposed

Figure 4.10: Visual comparison of the proposed algorithms with five different algorithms. The zoom-in comparison is taken from the image Hindu.

by a factor of three compared to previous version of the algorithm. To verify this, In Table. 4.5 and Table. 4.6 we have compared the PSNR, MSSIM and timing results of the previous version of the algorithm with this timing-enhanced version of the algorithm. The timing-enhanced version achieves approximately the same results three times faster than the previous version by reducing the number of sequence estimations and making use of the color information at each position.

As can be seen from the tables, the time-enhanced version of the algorithm obtains roughly the same results in terms of PSNR and MSSIM compared to the early version

Table 4.3: Time (minutes) consumed by the proposed algorithm and the method of [24] for the set in Fig. 4.4.

	Starfish	**Wall art**	**Church**	**Samurai**	**Swan**	**Louver**	**Aviator**	**Flag**	**Car**
[24]	10.67	12.27	11.82	12.52	12.55	12.22	12.29	11.52	11.02
Proposed	4.67	4.11	4.15	4.03	4.25	4.29	4.14	4.66	4.69

Table 4.4: Time (minutes) consumed by the proposed algorithm and the method of [24] for the set in Fig. 4.7.

	Stairs	**Man**	**Bird**	**Woman**	**Jump**	**Deer**	**Spear**	**Hindu**	**House**	**Model**
[24]	11.15	10.67	10.91	10.47	10.26	10.88	10.82	10.83	10.88	11.26
Proposed	4.73	4.98	4.59	4.61	4.94	4.69	4.55	4.70	4.21	4.31

of the proposed method. However, the algorithm on average has running time of approximately seven times smaller than the running time of the method in [24]. As discussed earlier this comparison is made under similar conditions for all the algorithms and no parallel processing instruction is used in our implementation.

Another important complexity consideration is the memory-complexity. With the same approach that we followed to compute the time-complexity of the algorithm, the amount of memory needed for the purpose of trellis processing is proportional to the length of the trellis as well as the number of rows. Other words, the memory-complexity is of order $h \times w$ i.e., $\theta(hw)$.

Table 4.5: PSNR and MSSIM results for different algorithms for the test set of Fig. 4.4.

	[24]			**Proposed**			**Proposed (time enhanced)**		
	PSNR	MSSIM	time	PSNR	MSSIM	time	PSNR	MSSIM	time
Starfish	29.54	0.8835	10.67	29.48	0.8851	4.68	29.49	0.8854	1.72
Wall art	24.83	0.7761	12.27	25.61	0.8132	4.11	25.60	0.8120	1.47
Church	31.17	0.9377	11.82	31.56	0.9464	4.15	31.54	0.9462	1.55
Samurai	26.81	0.8832	12.52	27.27	0.8913	4.03	27.27	0.8914	1.69
Swan	30.29	0.8990	12.55	31.09	0.9194	4.25	31.07	0.8990	1.64
Louver	24.57	0.8405	12.22	24.62	0.8479	4.29	24.62	0.8476	1.73
Aviator	26.33	0.8757	12.29	26.69	0.8927	4.14	26.70	0.8928	1.68
Flag	25.32	0.8055	11.53	26.08	0.8297	4.66	26.09	0.8301	1.49
Car	28.47	0.7915	11.02	29.11	0.8230	4.69	29.09	0.8232	1.57
Zebra	26.33	0.8756	12.14	26.34	0.8772	4.34	26.35	0.8773	1.50
Average	27.37	0.8568	11.90	27.78	0.8726	4.33	27.78	0.8705	1.60

Table 4.6: PSNR and MSSIM results for different algorithms for the test set of Fig. 4.7.

	[24]			**Proposed**			**Proposed (time enhanced)**		
	PSNR	MSSIM	time	PSNR	MSSIM	time	PSNR	MSSIM	time
Stairs	28.30	0.8846	11.15	28.45	0.8949	4.73	28.46	0.8951	1.61
Man	34.97	0.9096	10.67	34.84	0.9177	4.98	34.84	0.9179	1.65
Bird	34.34	0.9454	10.90	35.43	0.9548	4.60	35.44	0.9550	1.46
Woman	26.63	0.7667	10.46	27.34	0.7916	4.41	27.35	0.7921	1.63
Jump	27.32	0.8645	10.26	27.89	0.8762	4.94	27.89	0.8762	1.70
Deer	26.13	0.7920	10.87	25.23	0.7399	4.69	25.26	0.7410	1.72
Spear	30.86	0.9162	10.82	30.88	0.9211	4.55	30.88	0.9211	1.79
Hindu	33.68	0.9366	10.83	34.10	0.9416	4.46	34.10	0.9417	1.62
House	22.98	0.8130	10.88	24.64	0.8532	4.21	24.64	0.8532	1.62
Model	27.80	0.9055	11.26	28.19	0.9118	4.31	28.19	0.9122	1.66
Tree	22.72	0.7390	11.59	24.03	0.7836	4.42	24.03	0.7838	1.55
Girl	32.04	0.9205	10.84	31.99	0.9263	5.00	31.99	0.9265	1.77
Eiffel	27.42	0.8712	12.51	28.76	0.9018	4.04	28.75	0.9019	1.77
Beach	27.12	0.9102	10.42	27.21	0.9108	5.09	27.23	0.9112	1.73
Average	28.74	0.8696	10.88	29.21	0.8797	4.64	29.22	0.8742	1.66

Chapter 5

Concluding Remarks

In this thesis, we introduced a new method for video de-interlacing based on the assumption that the available set of interpolators are states of a Markov chain. Then with the help of reasonable assumptions, we derived approximations for the transition probabilities of the Markov chain. Next, the problem of choosing the optimal interpolator for each missing pixel was formulated using a maximum a posteriori sequence estimation and was solved efficiently on a trellis diagram spanning the missing rows. At this stage we utilized the Viterbi algorithm to find the optimum sequence of interpolators. Next step, reformulation of the algorithm allowed for solving it using Forward-Backward algorithm. We obtained better performance with the latter formulation but with more computational complexity. The algorithms proposed have better results compared to well-known algorithms published on de-interlacing. Also the proposed algorithms benefit from an almost parameter-free implementation despite many algorithms that suffer from parameters that need to be adjusted empirically. In terms of timing, our current implementation of the algorithms do not run in real-time, however, as discussed in the last two chapters, the algorithm is highly

potent of being implemented using parallel programming. The real-time implementation of the algorithms remains as a related future work.

Moreover, we have modified the proposed MRF-based model to use it for image resolution enhancement by a factor of two. The proposed image up-scaling algorithm shows a great performance objectively and visually with a variety of test images and is superior to many state-of-the-art up-scaling algorithms published in this field. According to the timing discussions in this thesis for our proposed up-scaling algorithm, the current version of the algorithm obtains greater results with smaller running time compared to some well-known competing algorithms. Better visual results and lower computational complexity are two major advantages to our newly proposed up-scaling algorithm .

Many related interpolation problems such as frame-rate conversion and view interpolation can adopt our proposed model. Modifying the proposed model to use it in these specific applications can also be considered as a future work. The last chapter of the thesis, which was dedicated to using the proposed MRF-based model for image up-scaling, is a proof that the algorithm is highly potent of being applied in different image/video interpolation subfields in the future.

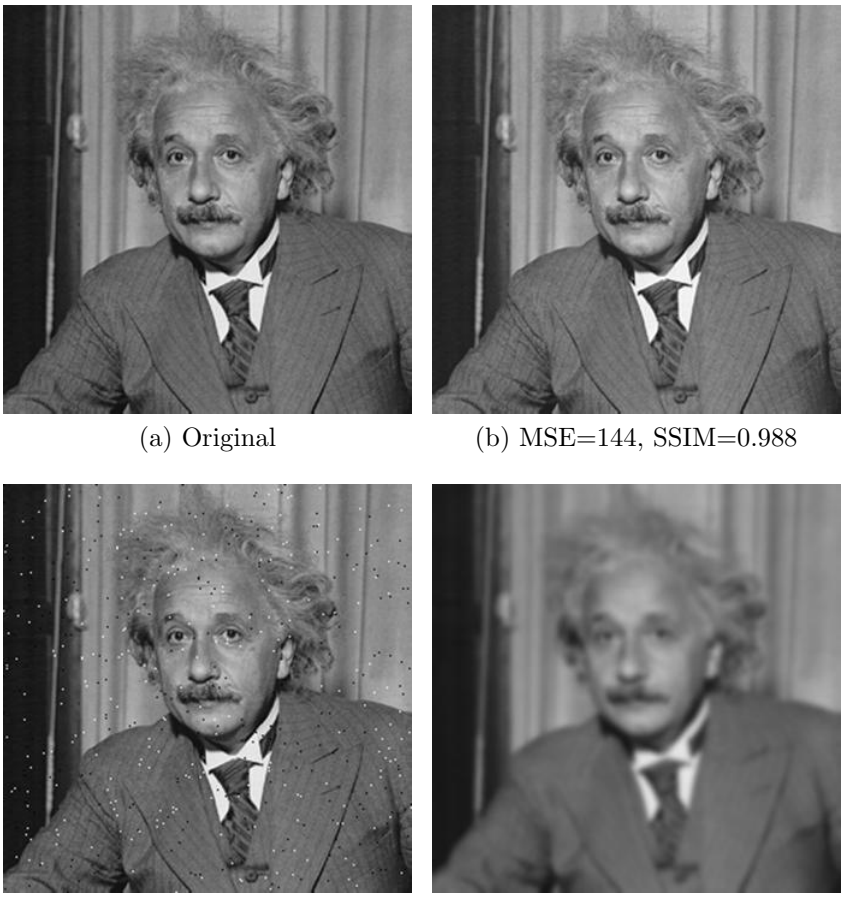
Appendix A

Structural Similarity (SSIM)

One of the widely used measures of similarity between two images is the structural similarity (SSIM). SSIM has been designed to improve the traditional metrics of similarity such as the mean of squared errors (MSE) or the peak signal-to-noise ratio (PSNR). The latter methods have been accepted to be inconsistent with the human visual perception. In other words, SSIM considers image degradation as the perceived change in structural information (not just perceived error as in PSNR or MSE). The underlying idea is the strong inter-dependencies between spatially-close pixels. The mean structural similarity (MSSIM) is usually calculated on windows of an image. If w_1 and w_2 are two windows of size $N \times N$ then following is the definition of the SSIM:

$$\mathrm{SSIM}(w_1, w_2) = \frac{(2\mu_{w_1}\mu_{w_2} + c_1)\,(2\sigma_{w_1 w_2} + c_2)}{\left(\mu_{w_1}^2 + \mu_{w_2}^2 + c_1\right)\left(\mu_{w_1} + \sigma_{w_2}^2 + c_2\right)}. \tag{A.1}$$

where μ_{w_1}, μ_{w_2}, μ_{w_1}, $\sigma_{w_2}^2$ and $\sigma_{w_1 w_2}$ are means of w_1 and w_2, variances of w_1 and w_2 and covariance of w_1 and w_2, respectively. c_1 and c_2 are variables that are used



(a) Original (b) MSE=144, SSIM=0.988

(c) MSE=144, SSIM=0.840 (d) MSE=144, SSIM=0.694

Figure A.1: Three different images with the same MSE but different SSIM. The images are from:[35]

.

for stabilization when division is with weak denominator. Usually $c_1 = (k_1 L)^2$ and $c_2 = (k_2 L)^2$ when L is the dynamic range of pixel values and k_1 and k_2 are set empirically. We exactly use the same settings of [35].

Figure A.1 illustrates the effectiveness of the SSIM versus MSE. Different images have the same MSE, however the SSIM metric greatly distinguishes the corruption in these images in accord with what is perceived by human visual system. The images

are a sample demonstration from the webpage for [35][1].

[1]https://ece.uwaterloo.ca/ z70wang/research/ssim/

Appendix B

Markov Random Fields (MRF)

The spatial property can be modeled through different aspects, among which, contextual constraint is a powerful option. Markov random fields (MRF) paves the way to model context-dependent entities such as correlated features and image/video pixels. Usually for an MRF, a neighborhood system is defines as $N = \{N_{xy}, (x, y) \in I\}$ where I is the set of all sites (e.g. image pixels) and N_{xy} is the set of neighbors of the site at position (x, y). For N_{xy} to be a neighborhood of (x, y) we should have the following:

$$(x, y) \notin N_{xy}, \quad (x, y) \subseteq N_{x'y'} \Leftrightarrow (x', y') \subseteq N_{xy}. \tag{B.1}$$

In other words, the neighborhood of (x, y) does not include the (x, y) itself and if (x, y) is a neighbor of (x', y') then (x', y') is a neighbor of (x, y) as well and vice versa. Figure B.1 shows a few sample neighborhoods obeying (B.1). By definition, if $\mathbf{z}$ is an MRF then:

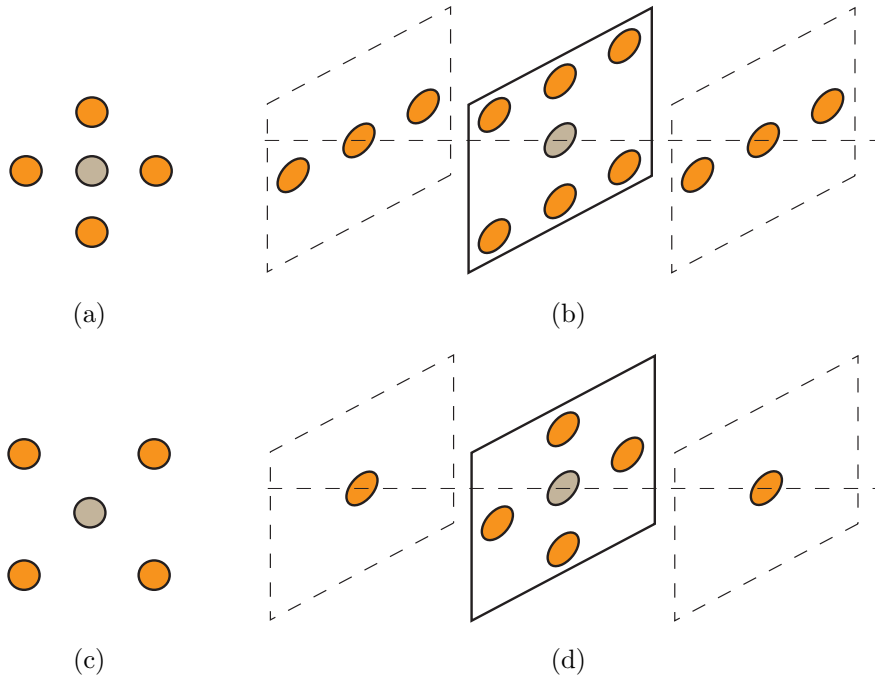


Figure B.1: Three sample neighborhoods consistent with the definition of the neighborhood systems above. (a) and (c) Two dimensional neighborhoods (b) and (d) Three dimensional samples.

$$\Pr(\mathbf{z}) > 0, \quad \forall \mathbf{z} \in \chi \tag{B.2}$$

$$\Pr(\mathbf{z}(x,y)|\forall(x',y') \neq (x,y)) = \Pr(z(x,y)|\forall(x',y') \in N_{xy}). \tag{B.3}$$

According to Hammersley-Clifford theorem, an MRF can be characterized by Gibb's distribution as:

$$\Pr(\mathbf{z}) = \frac{1}{Q}\exp\{-U(\mathbf{z})\} \tag{B.4}$$

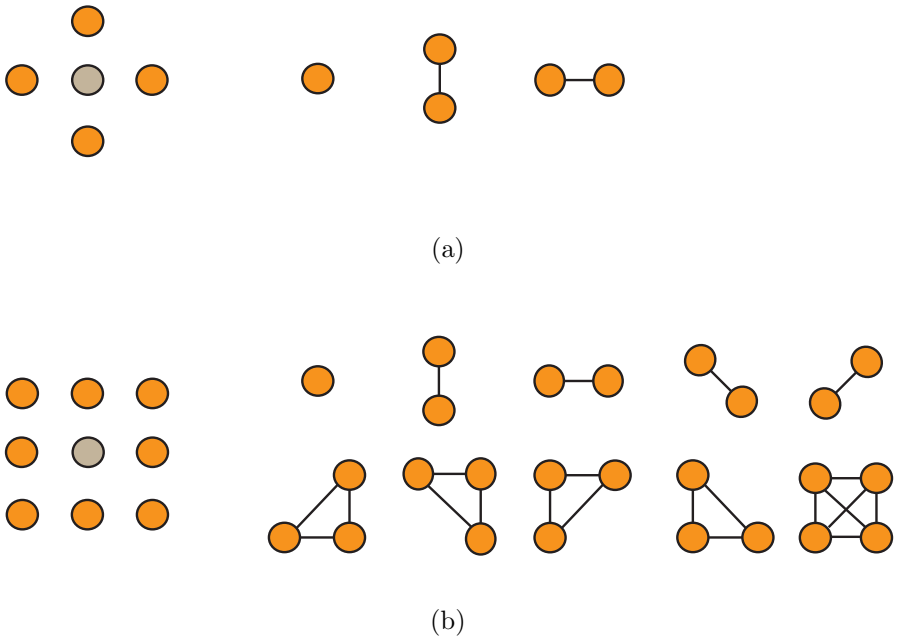
where Q is a normalizing factor called the partition function:

$$Q = \sum_{\mathbf{z} \in \chi} \exp\{-U(\mathbf{z})\}. \tag{B.5}$$

and $U(x)$ is an energy function of the form:

$$U(\mathbf{z}) = \sum_{c \in \mathcal{C}} V_c(\mathbf{z}) \tag{B.6}$$

which is a sum of cliques' potentials $V_c(x)$ over all possible cliques $\mathcal{C}$. A clique is a subset of all sites in which each pair of distinct sites are neighbors or the clique has a single site. Figure B.2 illustrates two sample neighborhood systems and associated cliques.



(a)

(b)

Figure B.2: Two different sample neighborhoods and their corresponding set of cliques. Note that cliques illustrated obey the above said definition for cliques.

According to the application and how MRFs are used in that specific application, the potential functions are defined. Then the corresponding MRF is estimated through an optimization stage. The objective function used for optimization step can take different shapes and include different terms with accord to the problem.

Bibliography

[1] Black, M.J., A. P. (1996). The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding, Elsevier*, **63**.

[2] Boughet, J. (2000). Pyramidal implementation of lucas kanade feature tracker: description of the algorithm. *Intel Microprocessor Labs*.

[3] Bruhn, A., W. J. S. C. (2004). Lucas/kanade meets horn/schunck: combining local and global optic flow methods. *International Journal of Computer Vision*.

[4] Chang, Y.-L., Lin, S.-F., Chen, C.-Y., and Chen, L.-G. (2005). Video deinterlacing by adaptive 4-field global/local motion compensated approach. *Circuits and Systems for Video Technology, IEEE Transactions on*, **15**(12), 1569 – 1582.

[5] Chen, Y.-R. and Tai, S.-C. (2009). True motion-compensated de-interlacing algorithm. *Circuits and Systems for Video Technology, IEEE Transactions on*, **19**(10), 1489 –1498.

[6] Choi, H. and Lee, C. (2011). Motion adaptive deinterlacing with modular neural networks. *Circuits and Systems for Video Technology, IEEE Transactions on*, **21**(6), 844 –849.

[7] Chung, C.-H., Liang, C.-C., Fan, Y.-C., Lin, H.-S., and Tsao, H.-W. (2009). Correction to novel artifact detection for motion compensated deinterlacing. *Industrial Electronics, 2009. ISIE 2009. IEEE International Symposium on*, pages 1337 –1340.

[8] Dai, S., Baker, S., and Kang, S. B. (2009). An mrf-based deinterlacing algorithm with exemplar-based refinement. *Image Processing, IEEE Transactions on*, **18**(5), 956 –968.

[9] De Haan, G. and Bellers, E. (1998). Deinterlacing-an overview. *Proceedings of the IEEE*, **86**(9), 1839 –1857.

[10] Fan, Y.-C. and Chung, C.-H. (2009). De-interlacing algorithm using spatial-temporal correlation-assisted motion estimation. *Circuits and Systems for Video Technology, IEEE Transactions on*, **19**(7), 932 –944.

[11] Gao, X., Gu, J., and Li, J. (2005). De-interlacing algorithms based on motion compensation. *Consumer Electronics, IEEE Transactions on*, **51**(2), 589 – 599.

[12] Horn, B. and Schunck, B. (1980). Determining optical flow. *Artificial intelligence, MIT*, pages 674 –679.

[13] Huang, Q., Zhao, D., Ma, S., Gao, W., and Sun, H. (2010). Deinterlacing using hierarchical motion analysis. *Circuits and Systems for Video Technology, IEEE Transactions on*, **20**(5), 673 –686.

[14] Jin, S., Kim, W., and Jeong, J. (2008). Fine directional de-interlacing algorithm using modified sobel operation. *Consumer Electronics, IEEE Transactions on*, **54**(2), 587 –862.

[15] Kang, K., Kim, J., and Jeong, J. (2009). Video sequence deinterlacing using intensity gradient filter and median filter with texture detection. *Advances in Multimedia, 2009. MMEDIA '09. First International Conference on*, **55**(3), 23 – 28.

[16] Keys, R. (1981). Cubic convolution interpolation for digital image processing. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, **29**(6), 1153 – 1160.

[17] Koo, H. I., Lee, S. H., and Cho, N. I. (2007). A new edi-based deinterlacing algorithm. *Consumer Electronics, IEEE Transactions on*, **53**(4), 1494 –1499.

[18] Lee, K., Lee, J., and Lee, C. (2009). Deinterlacing with motion adaptive vertical temporal filtering. *Consumer Electronics, IEEE Transactions on*, **55**(2), 636 –643.

[19] Li, M. and Nguyen, T. (2007). A de-interlacing algorithm using markov random field model. *Image Processing, IEEE Transactions on*, **16**(11), 2633 –2648.

[20] Li, R., Zheng, B., and Liou, M. (2000). Reliable motion detection/compensation for interlaced sequences and its applications to deinterlacing. *Circuits and Systems for Video Technology, IEEE Transactions on*, **10**(1), 23 –29.

[21] Li, X. and Orchard, M. (2001). New edge-directed interpolation. *Image Processing, IEEE Transactions on*, **10**(10), 1521 –1527.

[22] Lin, S.-F., Chang, Y.-L., and Chen, L.-G. (2003). Motion adaptive interpolation with horizontal motion detection for deinterlacing. *Consumer Electronics, IEEE Transactions on*, **49**(4), 1256 – 1265.

[23] Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. *Artificial Intelligence, 1981 7th International Joint Conference on*.

[24] Mallat, S. and Yu, G. (2010). Super-resolution with sparse mixing estimators. *Image Processing, IEEE Transactions on*, **19**(11), 2889 –2900.

[25] Mohammadi, H., Langlois, P., and Savaria, Y. (2007). A five-field motion compensated deinterlacing method based on vertical motion. *Consumer Electronics, IEEE Transactions on*, **53**(3), 1117 –1124.

[26] Park, M. K., Kang, M. G., Nam, K., and Oh, S. G. (2003). New edge dependent deinterlacing algorithm based on horizontal edge pattern. *Consumer Electronics, IEEE Transactions on*, **49**(4), 1508 – 1512.

[27] Park, S.-J., Jeon, G., and Jeong, J. (2009). Deinterlacing algorithm using edge direction from analysis of the dct coefficient distribution. *Consumer Electronics, IEEE Transactions on*, **55**(3), 1674 –1681.

[28] Park, S.-J., Jeon, G., and Jeong, J. (2010). Covariance-based adaptive deinterlacing method using edge map. *Image Processing Theory Tools and Applications (IPTA), 2010 2nd International Conference on*, pages 166 –171.

[29] Pradeep Sen, S. D. (2009). Compressive image super-resolution. *Signals, Systems and Computers, 2009 Conference Record of the Forty-Third Asilomar Conference on*.

[30] Shen, Y., Zhang, D., Zhang, Y., and Li, J. (2006). Motion adaptive deinterlacing

of video data with texture detection. *Consumer Electronics, IEEE Transactions on*, **52**(4), 1403 –1408.

[31] Sun, D., R. S. B. M. (2010). Secrets of optical flow estimation and their principles. *IEEE International Conference on Computer Vision and Pattern Recognition*, **63**.

[32] Vedadi, F. and Shirani, S. (2012a). De-interlacing using non-local costs and markov-chain-based estimation of interpolation methods. *Submitted to IEEE Transactions on Image Processing*.

[33] Vedadi, F. and Shirani, S. (2012b). Image resolution up-conversion via maximum a posteriori sequence estimation and viterbi algorithm. *Proceedings of IEEE International Conference on Image Processing*.

[34] Vedadi, F. and Shirani, S. (2012c). A new map-based approach to video de-intelacing using forward-backward algorithm. *Proceedings of Asilomar Conference on Signals, Systems and Computers*.

[35] Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *Image Processing, IEEE Transactions on*, **13**(4), 600 –612.

[36] Yang, S., Kim, D., and Jeong, J. (2009). Fine edge-preserving deinterlacing algorithm for progressive display. *Consumer Electronics, IEEE Transactions on*, **55**(3), 1654 –1662.

[37] Yu, L., Li, J., Zhang, Y., and Shen, Y. (2006). Motion adaptive deinterlacing with accurate motion detection and anti-aliasing interpolation filter. *Consumer Electronics, IEEE Transactions on*, **52**(2), 712 – 717.

[38] Zhang, L. and Wu, X. (2006). An edge-guided image interpolation algorithm via directional filtering and data fusion. *Image Processing, IEEE Transactions on*, **15**(8), 2226 –2238.