Key-Frame Based Video Super-Resolution for Hybrid Cameras

KEY-FRAME BASED VIDEO SUPER-RESOLUTION FOR HYBRID CAMERAS

BY

ROBERT LENGYEL, B.Eng

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 (2015) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Key-Frame Based Video Super-Resolution for Hybrid Cameras
AUTHOR:	Robert Lengyel B.Eng, (Electrical Engineering) McMaster University, Hamilton, Canada
SUPERVISOR:	Dr. Shahram Shirani
NUMBER OF PAGES:	xvii, 92

To the God of the Universe

Abstract

Super-resolution algorithms are designed to enhance the detail level of a particular image or video sequence. However, it is very difficult to achieve in practice due to the problem being ill-posed, and often requires regularization based on assumptions about texture or edges. The process can be aided using high-resolution key-frames such as those generated from a hybrid camera. A hybrid camera is capable of capturing footage in multiple spatial and temporal resolutions. The typical output consists of a high resolution stream captured at low frame rate, and a low resolution stream captured at a high frame rate. Key-frame based super-resolution algorithms exploit the spatial and temporal correlation between the high resolution and low resolution streams to reconstruct a high resolution and high frame rate output stream.

The proposed algorithm outlines a hierarchy to the super-resolution process, combining several different classical and novel methods. A residue formulation decides which pixels are required to be further reconstructed if a particular hierarchy stage fails to provide the expected results when compared to the low resolution prior. The hierarchy includes the optical flow based estimation which warps high frequency information from adjacent key-frames to the current frame. Specialized candidate pixel selection reduces the total number of pixels considered in the NLM stage. Occlusion is handled by a final fallback stage in the hierarchy. Additionally, the running time

for a CIF sequence of 30 frames has been significantly reduced to within 3 minutes by identifying which pixels require reconstruction with a particular method.

A custom simulation environment implements the proposed method as well as many common image processing algorithms. EngineX provides a graphical interface where video sequences and image processing methods can be manipulated and combined. The framework allows for advanced features such as multithreading, parameter sweeping, and block level abstraction which aided the development of the proposed super-resolution algorithm. Both speed and performance were fine tuned using the simulator which is the key to its improved quality over other traditional super-resolution schemes.

Acknowledgements

First and foremost, I would like to thank my supervisor Dr. Shahram Shirani for his wisdom and guidance during my study period and thesis writing times. He has been an inspiration for me to pursue excellence through hard work and determination, and has given me the freedom to pursue and implement many ideas, always with endless enthusiasm and a smile. His faith in me and my work has allowed me to flourish intellectually and become an expert in the image processing realm. I am blessed to have an advisor like him.

I would also like to thank my loving family and friends for their continued support during my difficult times. All of their encouragement has given me the strength to accomplish my dreams; I could not have done it without them. I have persevered through the hardest chapter of my life, and I am grateful for all their efforts to help me succeed.

A special thanks to Cheryl Gies for her commitment to helping students graduate. Her unwavering patience and support over the years has made a huge impact on my success at McMaster University. Her dedication to students including me has made the difference between passing and failing, and I am lucky to have her looking out for me.

Notation

The following tables will outline the notation and parameters used throughout this thesis. It is highly recommended that the reader print these out and/or keep a copy side-by-side when reading.

Type	Notation Display	Notes	Example(s)
Scalar	Normal Font	Prefer uppercase for parameters	s, S
Vectors	Lowercase Bold	Indicate $\hat{\mathbf{v}} = \langle v_1, v_2, \cdots \rangle$	$\hat{\mathbf{v}}$
Matrices	Uppercase Bold	Special definition $\boxed{\underline{\mathbf{M}}}_3[t] \triangleq \mathbf{M}_t$	$\mathbf{M}$
Tensors	Boxed Up	Subscript is basis dimensions	$\boxed{\underline{\mathbf{T}}}_3$
Indexing	Square Brackets	Braces for set/list indexing	$\mathbf{I}[\hat{\mathbf{v}}]$, $\psi\{\hat{\mathbf{v}}\}$
Functions	Curved Brackets	Function name font is result type	$f(x)$, $\mathbf{F}(x)$
Sets	Greek Letters	Except σ; reserved for Std. Dev	Φ, β
Operator	Fancy Letters	Used for data transformations	$\mathcal{FANCY}$
Abstract	Overbar	Used for placeholder variables	$\overline{k}$, $\overline{\hat{\mathbf{k}}}$, $\overline{\mathbf{K}}$
Basis	Over Arrow	Used for basis directions	$\vec{i}$, $\vec{j}$, $\vec{k}$
Identifier	Standard Font with Under script	Distinguishes identifiers vs actual values in variable subscripts	$\mathbf{H}_{\mathrm{v}}$ vs. $\mathbf{H}_v$

1. Video sequences are represented as sequences of image frames such as $\mathbf{I}_t$, where t is the frame index.

2. A single pixel from any image frame is retrieved from a matrix using indexing such as: $\mathbf{I}_t[\hat{\mathbf{p}}]$ where $\hat{\mathbf{p}} = \langle p_{\vec{x}}, p_{\vec{y}} \rangle$

Variables and Parameters

Parameter	Description	Section	Default
R	For every R number of LR frames, one HR frame is generated.	1.1	5
F	The downsampling factor between HR and LR frames.	1.1	2
P	The $(P \times P)$ comparison patch size used in NLM.	3.5.3	5
S	The $(S \times S)$ patch size considered for the candidate sets in NLM.	3.5.1	7
D	Strength modulator for error patches in NLM.	3.5.3	6.0
A	Minimum error multiplier in definition for T_1.	3.5.4	2.5
T_r	Threshold for the spatial deblocking filter.	3.2.2	0.5
$T_\mathrm{f}^{(k)}$	Threshold(s) for passing the k^{th} feedback propagation stage.	3.2	2, 5, 14
T_n	Threshold for the NLM quality function	3.5.5	0.8
T_o	Threshold for the optical flow quality function	3.4.3.1	0.04
K_r	Richardson-Lucy deconvolution iterations.	3	8
K_b	The blending strength in optical flow fusion.	3.4.3	6
K_1	Coefficient (first) for the threshold T_2 function.	3.5.4	500
K_2	Coefficient (second) for the threshold T_2 function.	3.5.4	1/900
K_3	Coefficient for the NLM quality function $\mathbf{Q}_{\overline{t}}$	3.5.5	150^2
σ_c	Std. Dev of the Gaussian PSF used in the hybrid camera.	1.1	1.6
σ_d	Std. Dev of the Gaussian kernel used as a deblocking filter.	3.2.2	4.0
σ_f	Std. Dev of the Gaussian weighting function used in optical flow.	3.4.3	1.5
σ_w	Std. Dev of the Gaussian weighting function used in optical warp.	3.4.3	8.0
σ_p	Std. Dev of the Gaussian filter used in NLM patch comparisons.	3.5.3	1.2
W_c	Window size used in the camera PSF, associated with σ_c.	1.1	3
W_d	Window size used in the deblocking filter, associated with σ_d.	3.2.2	5
W_f	Window size used in optical flow, associated with σ_f.	3.4.3	7
W_w	Window size used in optical warp, associated with σ_w.	3.4.3	13

Variable	Description	Section	Equation
$\mathbf{X}_t$	Original HR Image at frame index t.	1.1	1.8
$\mathbf{Y}_t^{(k)}$	The $\overline{k}^{\text{th}}$ hierarchical reconstructed output at frame index t.	3.2	3.4
$\mathbf{Z}_f$	Temporally downsampled HR frame at frame index f.	1.1	1.8
$\mathbf{D}_t$	Filtered and downsampled $\mathbf{X}_t$ at frame index t.	1.1	1.8
$\mathbf{U}_t$	Upsampled $\mathbf{D}_t$ using $\mathcal{B}_F$ at frame index t.	3	3.2
$\mathbf{R}_t$	Richardson-Lucy deconvolution for frame index t.	3	3.3
$\mathbf{S}_t$	Heavy downsampled sequence for computing motion vectors.	3.5.2	3.28
$\mathbf{G}_{\overline{\sigma}}$	Gaussian kernel of with Std. Dev of $\overline{\sigma}$.	Preface	5
$\mathbf{M}_t^{(k)}$	The $\overline{k}^{\text{th}}$ Reconstruction map: Boolean mask indicating which pixels are to be processed in each frame index t.	3.2	3.4
$\mathbf{L}_t$	Linear estimation at frame index t.	3.3	3.7
$\overrightarrow{\mathbf{O}}_t$	Optical Estimation Forward.	3.4.3	3.19
$\overleftarrow{\mathbf{O}}_t$	Optical Estimation Backward.	3.4.3	3.19
$\mathbf{O}_t$	Optical Estimation Final Fused.	3.4.3	3.21
$\mathbf{N}_t$	NLM reconstructed output at frame index t.	3.5	3.24
$\mathbf{Q}_t$	NLM reconstruction quality function.	3.5.5	3.39
T_1	First threshold to pass in NLM candidate reduction.	3.5.4	3.36
T_2	Second threshold to pass in NLM candidate reduction.	3.5.4	3.37

Operators	Description	Section	Equation
$\mathcal{D}_{\overline{F}}$	Decimation operator using a factor of $\overline{F}$.	1.1	1.8
$\mathcal{B}_{\overline{F}}$	Bicubic upsampling operator using a factor of $\overline{F}$.	1.1	3.2
$\mathcal{F}_{\overline{\mathbf{K}}}$	Filtering/convolution operator using kernel $\overline{\mathbf{K}}$.	1.1	1.8
$\mathcal{R}_{\overline{\mathbf{K}}}^{\overline{i}}$	Richardson-Lucy deconvolution operator using kernel $\overline{\mathbf{K}}$ with $\overline{i}$ iterations total.	3	1.8
$\mathcal{H}^{(\overline{k})}$	The $\overline{k}^{\text{th}}$ hierarchical method operator.	3.5.3	3.35
$\mathcal{G}$	Gradient magnitude operator (Sobel).	3.5.3	3.35
$\mathcal{L}$	Luminance channel of a YUV transform operator.	3.5.3	3.35
$\mathcal{S}_{\overline{r}}$	Boolean column 'OR' operator given HR Rate $\overline{r}$.	3.5.3	3.35
$\mathcal{T}_{\overline{v}}$	Threshold operator given threshold value $\overline{v}$.	Preface	4
$\mathcal{X}_{\hat{\mathbf{p}}}^{\overline{s}}$	Patch extract operator of size $(\overline{s} \times \overline{s})$ centered on $\hat{\mathbf{p}}$.	3.5.3	3.35

Sets	Description	Section	Equation
$\beta_t^k(\hat{\mathbf{p}})$	Candidate set: initial list of pixels in $\mathbf{Z}_k$ to be used to reconstruct $\mathbf{U}_t[\hat{\mathbf{p}}]$	3.5.1	3.25
$\Lambda_t(\hat{\mathbf{p}})$	Grand candidate set: Union of all candidate sets, $\forall \overline{k}$	3.5.1	3.26
$\Upsilon_t(\hat{\mathbf{p}})$	Thresholded candidate set: candidate pixels that meet thresholds T_1 and T_2.	3.5.1	3.27
$\psi_w(\hat{\mathbf{p}})$	Returns a $(w \times w)$ set of valid pixel indexes centered on $\hat{\mathbf{p}}$.	Preface	2
$\phi(\boxed{\mathbf{T}}_n)$	Returns all the valid pixel indexes possible for a given input tensor $\boxed{\mathbf{T}}_n$.	Preface	2

Processing Tools & Definitions

Generic definitions, that will be useful for the development of this algorithm, are presented here. Additionally, these definitions provide mathematical completeness, as well as definitions for simpler methods assumed. These can be referenced when they are encountered in the thesis, and additionally their usage will refer back to this page.

Set of Pixels Indexes Centered on a Pixel Index

This function returns a set of pixel indexes derived from a $(w \times w)$ square patch centered on $\hat{\mathbf{p}}$, having only natural numbers (non-negative, positive) indexes:

$$\psi_w(\hat{\mathbf{p}}) = \left\{ \hat{\bar{\mathbf{v}}} \in \mathbb{N}^2 \;\middle|\; (\hat{\mathbf{p}} - \lceil \tfrac{w}{2} \rceil) \leq \hat{\bar{\mathbf{v}}} \leq (\hat{\mathbf{p}} + \lfloor \tfrac{w}{2} \rfloor) \right\} \tag{1}$$

The addition/subtraction of a vector and scalar is performed by promoting the scalar to be a vector with the same magnitude in all basis directions.

Set of all possible vector coordinates

This function is used to generate a list of pixel coordinates that can be used to address/index a source companion matrix. It can be extended to tensors of any dimension n. The set generated will contain vectors $\hat{\mathbf{v}} = \langle v_1, v_2, \ldots, v_n \rangle$ that point to each possible element:

$$\phi(\boxed{\mathbf{T}}_n) = \{\hat{\bar{\mathbf{v}}} \in \mathbb{N}^n \;\middle|\; \boxed{\mathbf{T}}_n [\hat{\bar{\mathbf{v}}}] \text{ is defined}\} \tag{2}$$

Patch Extract Operator

This operator acts on its companion matrix $\bar{\mathbf{I}}$ and extracts a patch $(\overline{w} \times \overline{w})$ centered on the input coordinate $\hat{\bar{\mathbf{p}}}$. The formal definition is as follows:

$$\mathcal{X}^{\overline{w}}_{\hat{\bar{\mathbf{p}}}} \bar{\mathbf{I}} = \begin{bmatrix} \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{1,1\}\right] & \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{1,2\}\right] & \cdots & \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{1,w\}\right] \\ \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{2,1\}\right] & \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{2,2\}\right] & \cdots & \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{2,w\}\right] \\ \vdots & \vdots & \ddots & \vdots \\ \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{w,1\}\right] & \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{w,2\}\right] & \cdots & \bar{\mathbf{I}}\left[\psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{w,w\}\right] \end{bmatrix} \tag{3}$$

Object Thresholding

This is the formal definition for thresholding an object $\boxed{\bar{\mathbf{I}}}_n$ (vector, image, matrix, tensor... etc). For every pixel index $\hat{\mathbf{p}} = \langle p_1, p_2, \ldots, p_n \rangle$, the threshold $\overline{v}$ is applied:

$$\mathcal{T}_{\overline{v}} \boxed{\bar{\mathbf{I}}}_n [\hat{\mathbf{p}}] = \begin{cases} 0 & \text{if} \quad \boxed{\bar{\mathbf{I}}}_n [\hat{\mathbf{p}}] < \overline{v} \\ 1 & \text{if} \quad \boxed{\bar{\mathbf{I}}}_n [\hat{\mathbf{p}}] \geq \overline{v} \end{cases} \tag{4}$$

Symmetric Gaussian

The 2D symmetric Gaussian function will be defined here, and used throughout. The window size should be set as to encompass most of the kernel. Additional normalization may be required after a window is chosen. The notation here allows negative pixel indexes in the Gaussian kernel produced, but, the implementation should shift the center of the kernel by half of the window size, in both directions.

$$\mathbf{G}_{\sigma}\mathbf{I}\left[\hat{\mathbf{m}}\right] = \frac{1}{2\pi\overline{\sigma}^2}\exp\left(-\frac{m_{\vec{x}}^2 + m_{\vec{y}}^2}{2\overline{\sigma}^2}\right) \tag{5}$$

Sobel Gradient

The gradient represents derivatives in the spatial direction using the Sobel operator. The operator defined here will utilize the magnitude of the $\vec{x}$ and $\vec{y}$ directions, where $\otimes$ represents convolution:

$$\mathcal{G}\mathbf{I} = \sqrt{\left(\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \otimes \mathbf{I}\right)^2 + \left(\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \otimes \mathbf{I}\right)^2} \tag{6}$$

Luminance Transform

The luminance transform takes the RGB pixels and returns a single channel image. There are many luminance transforms available, and each one is subjectively correct. The following transformation is a RGB888 to YUV transformation with full swing [0, 255] is used. The coefficients sum to 0.9999, and are the same ones that the '*rgb2gray*' function in Matlab uses.

$$\mathcal{L}\mathbf{I}\left[\hat{\mathbf{v}}\right] = 0.2989 \cdot \mathbf{I}\left[\hat{\mathbf{v}}\right]_{\vec{R}} + 0.5870 \cdot \mathbf{I}\left[\hat{\mathbf{v}}\right]_{\vec{G}} + 0.1140 \cdot \mathbf{I}\left[\hat{\mathbf{v}}\right]_{\vec{B}} \tag{7}$$

Contents

List of Figures

Chapter 1

Introduction and Problem

Hybrid video cameras are unique among video cameras in that they are able to shoot in two (or more) modes. Each mode represents a unique combination of frame rate and resolution. For a high frame rate, typically a lower resolution is used and conversely for a given low frame rate, a high resolution is used. This balance occurs because of limited processing power on-board the microcontroller in the camera. Improving both frame rate and resolution requires higher quality cameras whose price exponentially reflects its performance.

By exploiting a hybrid cameras ability to dynamically switch between the two modes, two video streams will be constructed. The first is a low resolution (LR) having high frame rate stream, while the second is a high resolution (HR) having low frame rate stream. Using the algorithm proposed here, the two video streams can be combined to attain both a high frame rate and high resolution video. This algorithm alleviates the need for expensive high-tech video cameras at the cost of processing time.

1.1 Mathematical Description of Problem

A typical hybrid camera output can be seen visually in Fig. 1.1 and can be modeled as a regular high definition video camera (high resolution, high frame rate) that undergoes two distinct output transformations. The first is a spatial downsampling operation to achieve low resolution (LR) frames while maintaining high frame rate, while the second is a temporal downsampling operation used to maintain the high resolution (HR) frames but with reduced frame rate.

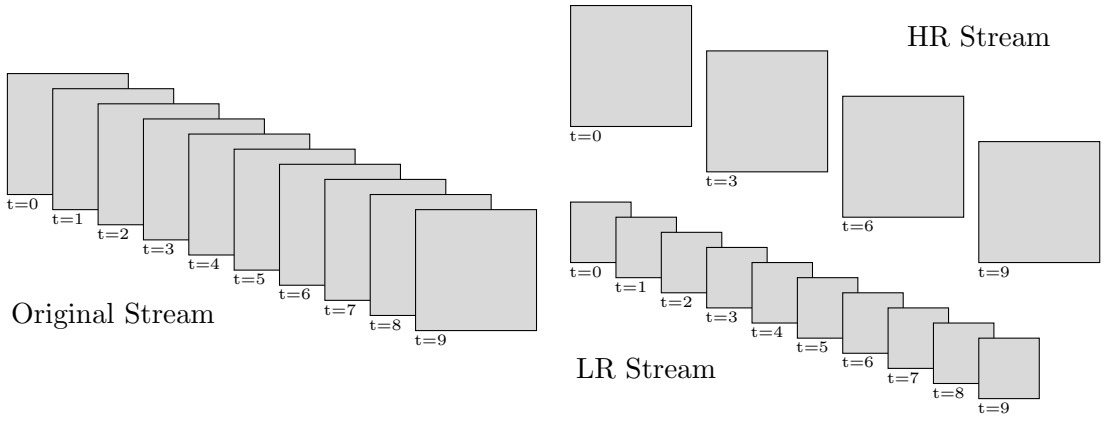


Figure 1.1: Left: the output of a regular camera. Right: the output of a hybrid camera (R=3, F=2)

The two transformations are shown visually in Fig. 1.2. The temporal downsampling operator uses the parameter R which represents how often HR frames are produced. The spatial downsampling operator uses the point spread function (PSF or transfer function) of the camera (assumed to be known) to first filter the input, and then reduce the resolution by a factor of F.

For each low resolution frame, the corresponding high resolution version must be reconstructed using the information from adjacent (nearest) HR key frames. This

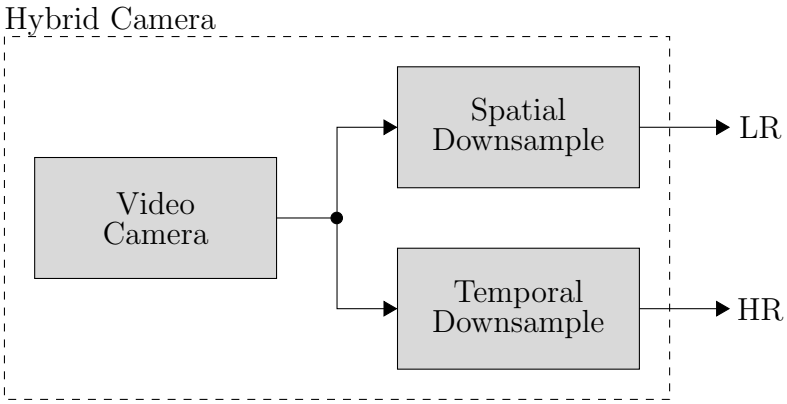


Figure 1.2: Hybrid camera model: a regular video camera with two output transformations.

process can be modeled as the inverse of the following equation:

$$\mathbf{D}_t = \mathcal{D}_F \mathcal{F}_{\mathbf{G}_{\sigma_c}} \mathbf{X}_t + \epsilon \tag{1.8}$$

where $\mathbf{X}_t$ is the original HR frame (if captured by a high resolution camera) and $\mathbf{D}_t$ is the degraded LR frame created by filtering and downsampling the input $\mathbf{X}_t$ using the operators $\mathcal{F}_{\mathbf{G}_{\sigma_c}}$ and $\mathcal{D}_F$ respectively. The parameter F is the resolution ratio, and $\mathbf{G}_{\sigma_c}$ is PSF of the camera: assumed to be a 2D symmetric Gaussian with standard deviation of σ_c. The noise term ϵ is added for modeling purposes and is ignored. Physically, filtering is achieved using the PSF of the camera, or using binning methods (averages over (2×2) or larger blocks) or both.

The combined operators for filtering and downsampling ($\mathcal{F}_{\mathbf{G}_{\sigma_c}}$ and $\mathcal{D}_F$) have no unique inverses and thus this problem is ill-posed; it cannot be solved uniquely using only the information in the LR frames. The matrix $\mathbf{D}_t$ is not of the same size as $\mathbf{X}_t$ so the inverse ${\mathcal{D}_F}^{-1}$ is not possible. Additionally, the inverse operation of filtering ${\mathcal{F}_{\mathbf{G}_{\sigma_c}}}^{-1}$ is the same size as $\mathbf{X}_t$ but is not mathematically unique[1].

[1]The inverse operation is also known as deconvolution, or deblurring.

1.2 Overview of Approaches Taken

With respect to the super-resolution problem being ill-posed, solutions can still be proposed. Typically, the solutions can be classified in a few distinct ways: interpolation based methods, estimation based, learning based, and reconstruction based. The proposed algorithm will utilize three of these methods.

1.2.1 Interpolation Based Methods

The first and simplest method is an interpolation based reconstruction which solves the problem of upsampling (the inverse of the downsampling operation) by simply interpolating values. The methods in this category include the simple bilinear and bicubic, and the advanced Lanczos and/or sinc interpolation and do not recover any high frequency information; typically the detail level of interpolation methods remains the same: blurry and full of artifacts [1]. However, despite their performance being the lowest, their speed is undoubtedly the highest which is due to their simplicity. These methods typically serve as starting points for more advanced methods since they do not utilize any information from the high resolution frames available.

1.2.2 Estimation Based Methods

Estimation based approaches attempt to solve the ill-posed problems by imposing some conditions on the solution (and thus achieving a unique solution, also known as regularization). Estimation methods can also be used in the frequency domain. These methods perform better than their interpolation cousins, but at the cost of speed.

1.2.3 Learning / Example Based Methods

The algorithms in this category attempt to learn by example how to reconstruct each pixel. By analyzing how LR frames are downsampled from HR frames, the high frequency content difference can be learned over successive frames. These methods are difficult because they depend on the learning methods used, and tend to be slow; dictionary sizes are really large, and comparing thousands of entries is computationally expensive.

1.2.4 Reconstruction Based Methods

The algorithms in this category attempt to reconstruct each individual pixel, usually using some averaging technique utilizing data from HR images. The non-local means (NLM) algorithm is a very generic (and powerful) averaging technique commonly used for many applications. The main advantage of this method is the performance while the disadvantage is its speed.

1.3 Common Problems

Using interpolation methods leads to blurry results as high frequency details are not reconstructed. Estimation based methods make assumptions about how natural images are formulated, and do a better job at guessing the high frequency content, often with great success. Learning based methods attempt to analyze differences between HR and LR frames, in order to apply that difference to super resolve other LR frames, however, they are often computationally expensive. Reconstruction based methods are similar to learning based, but do not have a dictionary constructed.

High frequency information in LR frames is reconstructed using available data in HR frames. However, while learning based methods suffer from computational complexity, reconstruction methods are affected by occlusion, a well known phenomenon whereby HR frames sometimes do not contain any information to reconstruct certain pixels in LR frames. Occlusion happens when objects are blocked in view by other objects, obstructing the pixel data due to perspective. Reconstruction based methods also suffer from motion vector estimation; the motion vectors involved in reconstructing must be calculated, increasing the computational cost as well.

It is clear that no method provides the complete solution. Thus, the approach used here utilizes the best of each of these algorithms: the speed of interpolation methods with the quality of reconstruction methods.

1.4 Scope of Thesis

The goal of the thesis is to present a key-frame based super-resolution algorithm that solves many of the problems in previous works. Namely, the algorithm must be computationally feasible; the aim is to push super resolution into real time. Next, its performance must surpass all algorithms in its class. Problems like occlusion and noise must be addressed. The maximization of performance and minimization of running time is a very difficult problem, but nevertheless an attempt will be made that serves to set a new benchmark for super resolution algorithms. The algorithm and its results are described in Chap. 3 and Chap. 4. To aid the process, a special virtual environment simulator was created and is presented in Chap. 5. Finally, the thesis will conclude with its recommendations and future work in Chap. 6.

Chapter 2

Super-Resolution Background

Research in super-resolution (SR) still continues despite the field reaching a mature state. Frameworks include algorithms for single-frame, multi-frame, video, key-frame and multi-view super-resolution. In single-frame SR, there is only one target image to reconstruct, and no other image priors are given. This kind of SR is difficult and requires many assumptions about texture, edge, and even spatial correlations. Multi-frame super-resolution focuses on the fusion of multiple LR (low resolution) images, and exploiting temporal correlations in tandem with the correlations from single-frame SR. Taken a step further, the same temporal correlations can be used to reconstruct entire video sequences: known as video super-resolution. Typically, individual LR frames will contribute to all of the reconstruction of each frame in the super-resolved output. In multi-view super-resolution, temporal, and spatial correlations will be used from multiple views to super-resolve frames, or entire sequences. While each of these methods attempts to generate more LR sources of information, key-frame based methods change the paradigm completely by introducing sparse sources of high resolution (HR) information, and will become a key part in the formation of

this work.

The basic SR algorithm will always contain some form of interpolation. Some early methods attempted to improve the interpolation process using sub pixel edge detection to guide the interpolation process in the high resolution output [2]. Their success paved the way for works such as [1][3][4] which have improved on that success. However many of these methods still provide blurry results due to their weakness in reconstructing texture (smooth) areas, as well as non-linear edges [5]. There is indeed no shortage of available interpolation methods employing novel strategies.

The influential paper by Freeman et al. [6] successfully demonstrated an example-based single-frame super-resolution scheme. First, they generated a database training set that maps numerous LR patches to HR patches from a non-related collection of HR images. In order to overcome the huge amounts of information required to store that database, they resorted to storing only the high frequency information that differentiates a LR patch from a HR one, as well as the source LR patch itself. To super resolve a test image, they break the image into overlapping patches, and process them individually one-by-one in raster-scan order. They predict the missing high frequency information in each patch by finding matches in the training set that have similar low frequency components. Additionally, the edges of the high frequency patches must match between adjacent patches (to maintain spatial correlation). By concatenating the overlapping pixel values (from the patch to the left, and the patch above) to form a search vector, the search format will match the storage format of the training set itself. Searching the large training sets for their matches was overcome finding the nearest neighbor using a greedy downhill search. Their method shows a moderate improvement between a cubic-spline zoom, and their super resolved

output. Surprisingly, the training set and the test image do not need to be similar in context, however there is a limit to this difference, as explicitly stated by the authors themselves. It is clear that the best training sets should come from the source stream itself.

Since the advent of digital cameras specifically those that can shoot video in different resolutions and framerates, work in SR has branched to reconstruction using limited sources of HR information which pushes the super-resolution limit. Ben-Ezra and Nayer [7] proposes three designs for a hybrid camera, and went on to implement a working prototype. They explore the trade-offs between spatial resolution and temporal resolution, which are enhanced by the use of a hybrid camera. Additionally, a large part of their work is centered around PSF estimation and deconvolution, which is an important piece of the puzzle in super-resolution. The output of a hybrid camera serves to generate HR key-frames in other super-resolution methods.

The work by Brandi et al. [8] uses dictionaries created from HR key-frames to super resolve low resolution non-key-frames. The first step divides each HR key frame into blocks, then through a threshold parameter, determines if there is relevant high frequency information present in that block. If there is, its degraded counterpart (filtered, downsampled, and upsampled) is stored, alongside the block, in the dictionary. Otherwise, if the block did not pass thresholding, the block is assumed to contain little to no high frequency information. During reconstruction, upsampled LR images are chunked into blocks, and the closest block in the dictionary found. Since the block is paired with an associated high frequency component, that information can be applied back to the upsampled block to super-resolve it. In their later work Brandi et al. [9], they replace the dictionary/training stage with motion estimation

to increase the robustness of their method.

Moving away from example based reconstruction, Protter et al. [10] has generalized the non-local means (NLM) algorithm for application in SR. Originally, the NLM was used for video de-noising due to its averaging property which smooths out high frequency noise.Their framework considers pixels from all LR and HR sources available, the NLM gives weights to each pixel candidate considered, and produces a final pixel reconstruction from the total sum. While the method is computationally very expensive, it has provided good results.

Inspired by [10], the work done by Najafi and Shirani [11] builds a regularization function that utilizes the NLM to super resolve an image. Their work includes a closed form solution that de-blurs upsampled LR frames using the NLM as a error metric that is minimized. Unlike the popular total variation regularization function proposed in [12] which imposes a simple condition on the reconstruction, the one used by [11] minimizes differences between LR and HR key-frames, which is more dynamic than the former. Their results show the advantage and success of their NLM based regularization function, even in sequences with occlusion or sparse HR frames. Additionally, the authors have addressed the computational complexity problem of NLM based methods, by adopting motion vector searches using the popular diamond search available in [13].

The work by Zhai and Wu [14] takes a similar reconstruction approach to super-resolution. By formulating the problem as a 2D piecewise autoregressive process, and then solving a constrained minimization problem using a Lagrangian multiplier, Zhai and Wu [14] formulates a closed form solution. The parameters of the algorithm are determined jointly along with the HR reconstruction in a least square problem.

Results indicate that the method performs well, with reduced computational cost[1]. However, the same problem as in Najafi and Shirani [11] occurs, specifically the value of a particular Lagrangian multiplier that is used for a constrained minimization problem is not known. In order to choose the best value for the parameter, the output MSE of the image as a function of the Lagrangian multiplier must be known, so that the relation can be inversed, and then an optimal value for the multiplier chosen. However, since the output MSE as a function of the Lagrangian multiplier is unknown, computationally expensive methods such as the generalized cross-validation can be used.

Another NLM approach to SR by Glaistter et al. [15] uses HR key-frames and shot boundary detection. The shot boundary / scene change detection algorithm is fairly simple, and serves to generate HR frames at a dynamic rate (encoder side). However, their work deals mainly with compression, and thus they did not fine tune the NLM based reconstruction used in the decoder; the proposed algorithm is designed for speed (real-time), which makes it very appealing. They demonstrate the success of the shot boundary detection in assisting the SR process, while minimizing computational complexity so such a degree, that it can be computed in real time.

Song et al. [5] outlined a super-resolution method whose output is a fusion of two different methods. The first is a reconstruction based entirely around bidirectional overlapped block motion compensation (BOBMC) [16][17] using HR key-frame information, and the second is a dictionary based (LR-HR pair) system. Motion vectors are found between a target upsampled LR frame and each HR key-frame. Thresholding the motion compensated error, on a patch by patch basis, determines if the LR

[1]Running times for the algorithm are not given explicitly, but the authors indicate relatively low computational cost.

patch will be reconstructed using the motion compensated output, or if it will go on to use the dictionary based system. The super-resolved patch is a sum of different weighted kernel functions (weight matrices were derived from the MPEG4 OBMC, and the pixel values from neighboring positions). The dictionary based scheme is the fallback for failed/poor motion compensation by the BOBMC due to complex motion, occlusion, errors in motion vectors and limited range. Building a standard dictionary of all possible LR-HR pairs, and clustering the numerous entries using k-means clustering, makes finding a similar patch an easier problem. In a synthesis step, the individual LR patch is super-resolved by combining all the patches in the cluster it belongs to, with some weights. The selection between which pixels will be reconstructed with which method, yields higher results than the individual methods themselves. Despite extensive use of motion estimation and exhaustive dictionary searches, their algorithm is only moderately complicated and is one of the faster algorithms available in literature.

The recent work by Cheng et al. [18] also uses key-frames for super-resolution. Their unique method initially upsamples the LR images and then performs motion estimation between each LR frame and corresponding adjacent HR key-frames. Using the vector field generated, the HR key-frames are shifted to the temporal index of the LR frame that is desired to be reconstructed. The motion compensation stage produces three outputs per LR frame: a forward, backward and bidirectionally compensated result. Their method proposes to fuse the three results with the upsampled LR frame, using a volume cube constructed per pixel using these four sources of information. The volume cube is small in size, and requires weights to function, as well as the right sources of information selected, as a subset of the four sources. To

determine the weights, an offline stage optimization by a particle swarm is performed. Conversely, to determine the right sources of information, a classification scheme is used that classifies a pixel based on both the 'temporal variation' and 'spatial energy' values. The pixel is classified as belonging to one of eight motion categories, depending on the aforementioned temporal variation and spatial energy values, which determine which information sources to fuse. The fusion stage considers the final reconstruction as a convolution of the small volume cube with the sources of information, to produce a single output value. Their work explores the variation of the HR to LR ratio, and delivers great overall performance.

While motion estimation is a key factor in the performance of super-resolution algorithms, the work by Lengyel et al. [19] demonstrates the success without explicit motion vector calculations. Drawing inspiration from [11] their NLM based method proposed reconstructs pixels individually by considering all pixels in a small region in each HR key-frame. For each of these 'candidate' pixels, a decision must be made regarding its inclusion in the NLM, as well as its weight (if included). Each candidate pixel must pass two adaptive thresholds, and if successful, it will attain a weight for its contribution. The adaptive thresholds serve to pick only the best candidate pixels among many, as well as to filter number of included pixels. Moreover, the pixels that pass thresholding are given a weight that is based on RBG pixel difference, luminance difference and gradient differences. Their aim is to be selective as possible with the reconstruction possible to avoid blurring caused by the nature of the NLM weighted average. Additionally, they demonstrate the success of their algorithm when applied to single-view sequences as well as multi-view sequences.

The proposed method in this thesis will continue the work of [19], and build

on lessons learned in the various methods proposed and subsequently described here. Specifically, motion estimation, fusion of multiple sources, fallback techniques, residue computation and thresholding, and domain based processing will all be utilized in the proposed method.

Chapter 3

Super-Resolution

Given a hybrid camera, the parameters R and F, as well as the point spread function (PSF) must be known. The first parameter R corresponds to the rate at which high resolution (HR) images are taken. For instance, a rate R of 5 gives one HR key frame for each 5 low resolution (LR) frames. The second parameter F is the resolution ratio and is defined as the spatial factor used between the LR and HR frames (typically 2).

The degradation model from Sec. 1.1 will be shown once more for reference.

$$\mathbf{D}_t = \mathcal{D}_F \mathcal{F}_{\mathbf{G}_{\sigma_c}} \mathbf{X}_t + \epsilon \tag{3.1}$$

where $\mathbf{X}_t$ is the original HR frame (if captured by a high resolution camera) and $\mathbf{D}_t$ is the degraded LR frame created by filtering and downsampling the input $\mathbf{X}_t$ using the operators $\mathcal{F}_{\mathbf{G}_{\sigma_c}}$ and $\mathcal{D}_F$ respectively. The parameter F is the resolution ratio, and $\mathbf{G}_{\sigma_c}$ is a 2D symmetric Gaussian kernel with known standard deviation of σ_c. The noise term ϵ is added for modeling purposes and is assumed to be Gaussian zero-mean white noise and subsequently ignored.

The hybrid camera produces two streams, a LR and a HR stream, marked as $\mathbf{D}_t$

for LR frames index t, and $\mathbf{Z}_f$ for HR frame index f. Note, that because of the HR generation ratio R, the number of frames in $\mathbf{D}_t$ is different than in $\mathbf{Z}_f$. Furthermore, the spatial resolution is also different between the aforementioned matrices $\mathbf{D}_t$ and $\mathbf{Z}_f$. Before any reconstruction occurs, the LR stream must be upsampled (to match the spatial resolution of $\mathbf{Z}_f$) using the bicubic operator $\mathcal{B}_F$ as shown below:

$$\mathbf{U}_t = \mathcal{B}_F \mathbf{D}_t \tag{3.2}$$

Next, the upsampled image is $\mathbf{U}_t$ deconvolved using the effective Richardson-Lucy (RL) deconvolution method [20] [21] (Sec. 3.6) (expressed as an operator $\mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}}$) to form $\mathbf{R}_t$ for each LR frame index t in the sequence.

$$\mathbf{R}_t = \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}} \mathbf{U}_t \tag{3.3}$$

The number of iterations for the RL deconvolution is given through the superscript parameter K_r whose value is typically 8. Additionally, the deconvolution algorithm requires the PSF that was initially used to blur the image, which in this case is known to be a 2D symmetric Gaussian $\mathbf{G}_{\sigma_\mathrm{c}}$ having known standard deviation σ_c from the degradation stage or hybrid camera. The deconvolved image should be stored as it will be used extensively in the algorithm.

3.1 Hierarchy Reconstruction Model

Next, a reconstruction hierarchy was devised that allows for a combination of techniques, each with a varying degree of complexity, as shown in Table 3.1.

A good way to visualize the hierarchy shown in Table 3.1 is to consider a hypothetical image, whose pixels are reconstructed by those methods. The set of all

Table 3.1: Proposed reconstruction hierarchy for key-frame based video super-resolution

Method	Description
Linear Estimation	Initial reconstruction uses HF information between HR key frames and simply interpolates their differences across LR frames. This primer method is computationally cheap.
Optical Flow Based Estimation	Attempts to 'warp' HF information from key frames to neighboring LR frames. The result of this method is surprisingly good, better than linear interpolation, but at moderate computation cost.
NLM Reconstruction	Attempts to fully reconstruct the pixel from HR frame sources in a neighborhood. This is a heavy computation but gives good results.
RL Deconvolution	Fallback: the absolute worst case scenario where all other methods failed, this one will give a result still better than bicubic.

pixels reconstructed with the various methods form supersets of each other as shown in Fig. 3.1. In each stage, a feedback residue method will determine which pixels are allowed to go through to the next stage of reconstruction.

The hierarchy method proposed requires that each of the methods involved can be applied on a per-pixel basis. The run time complexity is reduced when this requirement is met, since the usage of each method will be decided on a per-pixel basis, through the reconstruction map, and therefore simpler methods can be used to save time where applicable.

3.2 Reconstruction Map

Before any individual stage can be described, the propagation criteria and method must be established. Consider a boolean image mask $\mathbf{M}_t^{(k)}$ (named the Reconstruction

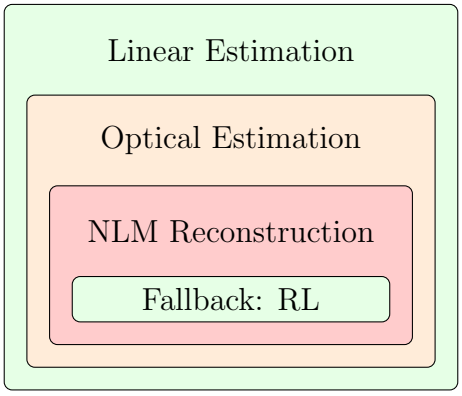


Figure 3.1: Proposed algorithm showing multiple stages of reconstruction, each one falling back on next.

Map) for each frame index t that is initially 1 everywhere (indicating all pixels need to be reconstructed by the next iteration in the hierarchy k), and progressively gets filled in (with 0's) after each stage using the propagation criteria to filter out pixels that passed the criteria and those that did not. In order words, the region of white 1 pixels (those that still need proper reconstruction) gets progressively smaller with each stage k. The reconstruction map can be written as an error residue metric combined with an error threshold operator $\mathcal{T}_{T_{\mathrm{f}}^{(k)}}$ given the k^{th} threshold $T_{\mathrm{f}}^{(k)}$, as shown below in Eq. 3.4:

$$\mathbf{M}_t^{(k)}\left[\hat{\mathbf{m}}\right] = \mathcal{T}_{T_{\mathrm{f}}^{(k)}}\left(\mathbf{U}_t\left[\hat{\mathbf{m}}\right] - \mathcal{B}_F\mathcal{D}_F\mathcal{F}_{\mathbf{G}_{\sigma_{\mathrm{c}}}}\mathbf{Y}_t^{(k-1)}\left[\hat{\mathbf{m}}\right]\right) \tag{3.4}$$

where $\mathbf{U}_t$ is the upsampled LR image at frame index t, and $\mathbf{Y}_t^{(k)}$ is the output of the application of the k^{th} hierarchy algorithm $\mathcal{H}^{(k)}$, on only the pixels remaining in the $\mathbf{M}_t^{(k)}\left[\hat{\mathbf{m}}\right]$ set. The combination of operators $\mathcal{B}_F\mathcal{D}_F\mathcal{F}_{\mathbf{G}_{\sigma_{\mathrm{c}}}}$ represents the degradation process and bicubic upsampling, and acts like a low pass filter. As before, F is the resolution ratio, and $\mathbf{G}_{\sigma_{\mathrm{c}}}$ is the known Gaussian PSF kernel of the camera given σ_{c}. The output $\mathbf{Y}_t^{(k-1)}$ is compared to the upsampled LR frames $\mathbf{U}_t$ in order to

create an error residue on a pixel per-pixel basis where $\hat{\mathbf{m}} = \langle m_{\vec{x}}, m_{\vec{y}} \rangle$. The feedback propagation threshold $T_{\mathrm{f}}^{(k)}$ is a parameter that will be varied in the algorithm, and its value plays a significant part in the running time and performance of this proposed algorithm.

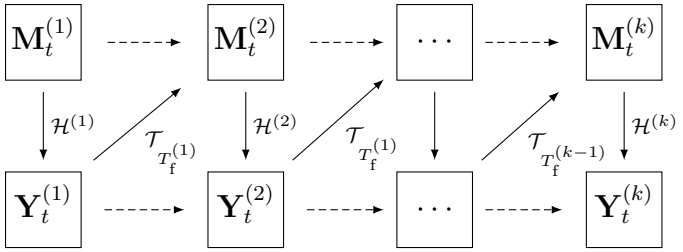


Figure 3.2: Reconstruction hierarchy showing residue feedback and thresholding.

The algorithm begins with the initialization of the reconstruction map $\mathbf{M}_t^{(1)}$ having a value of white or 1 everywhere. Next, the first hierarchical method: linear estimation $\mathcal{H}^{(1)}$ is applied. The output of that process yields the first reconstructed output $\mathbf{Y}_t^{(1)}$. Thresholding this against the upsampled $\mathbf{U}_t$ using the feedback propagation threshold $T_{\mathrm{f}}^{(k)}$ gives the next reconstruction map $\mathbf{M}_t^{(2)}$. This new reconstruction map is really just a subset of the original $\mathbf{M}_t^{(1)}$. Next, the second hierarchical method: optical flow based estimation $\mathcal{H}^{(2)}$ is applied on the remaining pixels that are white in $\mathbf{M}_t^{(2)}$. The output of that process yields the second reconstructed output $\mathbf{Y}_t^{(2)}$. This output is really just an improved version of $\mathbf{Y}_t^{(1)}$. Repeating the process a few more times (for the remaining NLM and fallback hierarchy methods) gives the final output $\mathbf{Y}_t^{(4)}$ (only 4 methods in the hierarchy), and can be visualized generically in Fig. 3.2.

3.2.1 Residue Basis

The criteria marked in Eq. 3.4 is a way to validate the success or failure of each individual hierarchical method $\mathcal{H}^{(k)}$ applied. To do so, the original source frame $\mathbf{X}_t$ is required, but since that information is unknown, the closest source of information that resembles $\mathbf{X}_t$ is simply $\mathbf{D}_t$ which is a distorted version of $\mathbf{X}_t$. In order to compare the output of the hierarchical method at each stage $\mathbf{Y}_t^{(k)}$ to the distorted $\mathbf{D}_t$, the hierarchical output must undergo the same distortions to ensure a 'fair' comparison, which is reflected in the operators $\mathcal{B}_F\mathcal{D}_F\mathcal{F}_{\mathbf{G}_{\sigma_c}}$ applied to $\mathbf{Y}_t^{(k)}$.

In order for the error residue in this method to work, it requires two key assumptions to be true.

- The error residue must persist spatially, in order for the hierarchical method to detect it at the same spatial position.
- The error residue must remain detectable with similar magnitude for its relevance to remain the same.

If the first assumption is violated, detectable errors will shift to some other location. This causes two problems: the first is that now a different pixel will be marked as needing reconstruction which will force reconstruction with a computationally heavier method (deeper in the hierarchy) than necessary. The second is that the original location where the error was is now reconstructed using a lighter method (higher in the hierarchy) which will cause overall errors in the final reconstruction.

If the second assumption is violated, it is possible for an error to be completely 'dissolved' by the fairness equalizing distortions applied. If the reduction in magnitude crosses the detectability threshold $T_{\mathrm{f}}^{(k)}$, it will not be reconstructed with the

appropriate method and again cause errors in the final reconstruction. On the other hand, if the magnitude of the error reside increases, falsely marked pixels (for reconstruction) cause unnecessary reconstruction of pixels using heavier methods and will increase running time.

In general, any differences between the original $\mathbf{X}_t$ and the reconstructed output $\mathbf{Y}_t^{(k)}$ must persist (spatially, and in magnitude) after the degradation distortions are applied. This is important so that the reconstruction map $\mathbf{M}_t^{(k)}$ can detect the errors, and use a heavier reconstruction option on that pixel, if necessary.

3.2.2 Implementation Details

In practice, the error residue and thresholding method is very susceptible to noise, and generally it will falsely mark too many pixels in the reconstruction map $\mathbf{M}_t^{(k)}$ as 1 or '*needed to be reconstructed in the next stage*'. Additionally, the second key assumption (outlined in Sec. 3.2.1) is often violated, and errors are missed. Consequently, the final output will have increased run time complexity and increased reconstruction errors.

The following proposed solution is one way to address the aforementioned problems. Equation Eq. 3.4 is modified into the following:

$$\mathbf{M}_t^{(k)}\left[\hat{\mathbf{m}}\right] = \mathcal{S}_R \mathcal{T}_{T_\mathrm{r}} \mathcal{F}_{\mathbf{G}_{\sigma_\mathrm{d}}} \mathcal{T}_{T_\mathrm{f}^{(k)}} \left(\mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}} \mathbf{U}_t\left[\hat{\mathbf{m}}\right] - \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}} \mathcal{B}_F \mathcal{D}_F \mathcal{F}_{\mathbf{G}_{\sigma_\mathrm{c}}} \mathbf{Y}_t^{(k-1)}\left[\hat{\mathbf{m}}\right] \right) \tag{3.5}$$

There are three main modifications present:

- Addition of K_r iterations of Richardson-Lucy deconvolution via operator $\mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}}$.
- Application of wide post filter $\mathbf{G}_{\sigma_\mathrm{d}}$ and secondary threshold using T_r after initial

thresholding with $T_{\mathrm{f}}^{(k)}$. This is a spatial de-blocking filter.

- Application of the special column 'OR' operator $\mathcal{S}_R$ as the final step. This is a temporal de-blocking filter.

The first modification enhances the comparison by deblurring the upsampled image source $\mathbf{U}_t$ as well as the reconstructed output $\mathbf{Y}_t^{(k)}$ at any stage k (post degradation). The new residue will be closer in magnitude to what the residue would be if the original source $\mathbf{X}_t$ was used, instead of bicubic upsampled $\mathbf{U}_t$ originally used in Eq. 3.4. The deblurring is applied using K_{r} iterations of the Richardson-Lucy deconvolution method, through the operator $\mathcal{R}_{\mathbf{G}_{\sigma_{\mathrm{c}}}}^{K_{\mathrm{r}}}$, using the known kernel $\mathbf{G}_{\sigma_{\mathrm{c}}}$ having known standard deviation σ_{c}. The Gaussian filter here must be the same as that used in the original degradation process that created the LR frames from the source HR frames, to ensure the best performance from the RL deconvolution.

The second modification is simply a region growth operation or equivalently a spatial de-blocking filter. By filtering a large region through $\mathbf{G}_{\sigma_{\mathrm{d}}}$, regions of the thresholded result that are white or 1 get larger, and regions that are black or 0 get larger as well. Here, the Gaussian kernel uses a different standard deviation of σ_{d} which recommended to be at least twice larger than σ_{c}. The operator removes stray pixels and creates continuous/whole regions that do not have holes. Expanded regions marked as white or 1 act as a buffer transition zone for where errors were detected. Its use avoids 'tearing' and other artifacts caused by having two different methods reconstruct adjacent pixels known to have high frequency information. The second threshold T_{r} is chosen as 0.5, the halfway point between 0 and 1, so that there is no preference towards white or black pixels in the final reconstruction map $\mathbf{M}_t^{(k)}$.

Finally, one last operation must be performed on the thresholded output. The

column 'OR' operator $\mathcal{S}_R$ is responsible for ensuring that the final reconstruction map $\mathbf{M}_t^{(k)}$ is the same for all indexes between HR frames exclusively. Mathematically, the following will hold true after the application of the operator:

$$\mathbf{M}_t^{(k)} = \mathbf{M}_n^{(k)} \quad \text{for} \quad \left(R\lfloor \tfrac{t}{R} \rfloor < n < R\lceil \tfrac{t}{R} \rceil\right) \tag{3.6}$$

The purpose of this exotic operation is to ensure that pixels that are reconstructed with any method are also reconstructed with the same method in the next frame. Again, this avoids 'tearing' and other artifacts when fusing the final reconstructed output $\mathbf{Y}_t^{(k)}$. The implementation of the operator is a simple boolean 'OR' operation between all the reconstruction maps $\mathbf{M}_t^{(k)}$ for the frame indexes n between each pair of HR key frames. Note, for LR frame indexes t that have HR siblings (ie when $t \equiv 0 \mod R$), the reconstruction map will be all 0 or black, indicating no further reconstructions are necessary. However, this is just a technicality for completeness, as those LR frames would get replaced with their appropriate HR siblings anyways.

The reconstruction map modifications from Eq. 3.5 can be better understood through visual aid presented in Fig. 3.3. Without making any assertions, it can be noted that the modified reconstruction map $\mathbf{M}_t^{(k)}$ picks out the HF content that was not reconstructed in the last iteration. This is an important property for the hierarchy methods to work properly: each stage must individually give a better than the previous stage output. For instance, the linear estimation method cannot properly reconstruct areas that are full of motion (ie HF information), and so the modified reconstruction map will find those errors and promote the appropriate pixels (mark as 1 or white) to be reconstructed with the next available method, in this case optical estimation. Additionally, the pixels that are marked as 0 or black can be reconstructed with a simpler method, speeding up the running time of the whole algorithm.

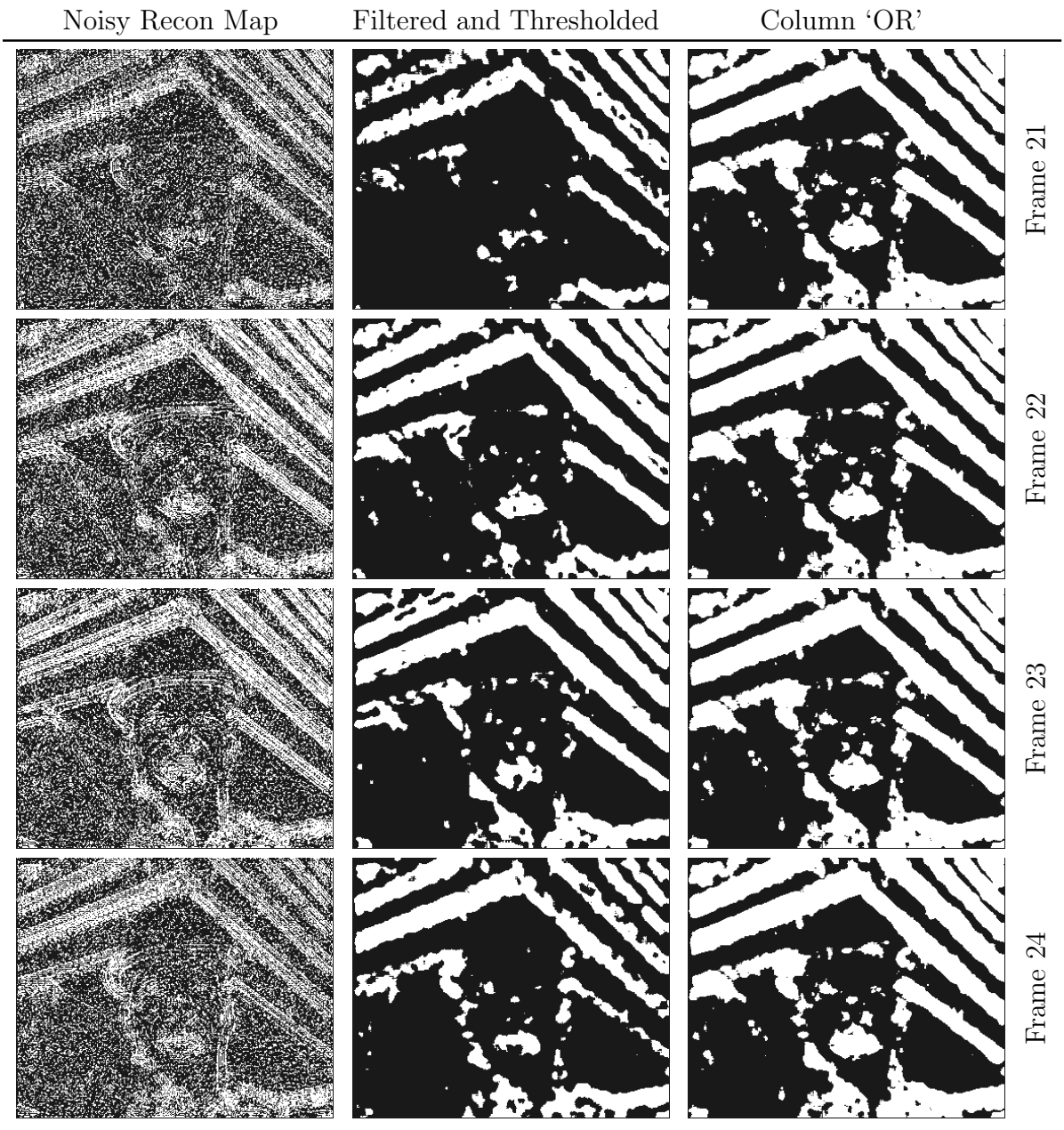


Figure 3.3: Visualization of the reconstruction map modifications. Note the column 'OR' maps are all identical.

3.3 Linear Estimation

The linear estimation $\mathbf{L}_t$ is the first reconstruction step in the grand hierarchical reconstruction scheme. Given that the downsampling and filtering operators and parameters ($\mathcal{F}_{\mathbf{G}_{\sigma_c}}$ and $\mathcal{D}_t$) that are used in the hybrid camera are known, the high frequency content in each HR key frame $\mathbf{Z}_{\overline{k}}$, for some HR key frame index $\overline{k}$, can be estimated as the difference between itself and a low pass filtered (blurred) version of itself: $\big(\mathbf{Z}_{\overline{k}} - \mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_{\overline{k}}\big)$. To construct the high frequency estimated output for non-HR index frames t, we interpolate the high frequency content from the nearest two HR frames across to the target frame. The interpolated high frequency content is added back to the low quality bicubicly upsampled image $\mathbf{U}_t$ at frame index t. The operation can be summarized into Eq. 3.7 shown below:

$$\begin{aligned}\mathbf{L}_t = \mathbf{U}_t + \big(1 - \tfrac{t \bmod R}{R}\big)\big(\mathbf{Z}_{f_1} - \mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_{f_1}\big) \\ + \quad \big(\tfrac{t \bmod R}{R}\big)\big(\mathbf{Z}_{f_2} - \mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_{f_2}\big)\end{aligned} \tag{3.7}$$

Where the upsampled LR frames $\mathbf{U}_t$ act a base value for the HF information, linearly interpolated between the two nearest HR key-frames $\mathbf{Z}_{f_1}$ and $\mathbf{Z}_{f_2}$. The frame indexes are defined as: $f_1 = \lfloor \frac{t}{R} \rfloor$ and $f_2 = \lceil \frac{t}{R} \rceil$, keeping in mind that the number of frames available in $\boxed{\mathbf{Z}}_3$ is reduced by a factor of R with respect to $\boxed{\mathbf{U}}_3$.

3.3.1 Formulation Basis

The initial linear estimation output is derived from the observation that many video sequences contain areas that are stationary in the course of R frames. Consider the 'News' sequence that illustrates clearly portions of video that are exactly the same over the course of the entire sequence. Modern video compression tools are able to

exploit this similarity, and therefore, our proposed method must do the same. The regions that are similar can be super-resolved simply by borrowing pixels from the nearest HR key frames. In addition, the speed of the algorithm can be increased (as previously explained in Sec. 3.2), by reconstructing the background/stationary pixels with the simplest method: linear estimation.

3.3.2 Implementation Details

While the linear estimation definition in Eq. 3.7 works adequately, it suffers from two main problems.

1. The HF formulation is missing the full degradation that a LF frame experiences: filtering and downsampling.

2. The HF formulation does not take advantage of the higher baseline set by using the Richardson-Lucy deconvolved precomputed matrix.

Solely using just filtering is not a true reflection of the low frequency content in this case; it suggests that the degradation process consists entirely of filtering, which is not true. It also included a downsampling component. Since, the aim is to add the HF differences back to upsampled LR frames, the HF formulation must be modified to reflect the full degradation process: filtering and downsampling. However, the addition of the downsampling components warrants the use of post upsampling, in order to preserve spatial resolution.

The final linear estimation formulation is shown below in Eq. 3.8:

$$\mathbf{Y}_t^{(1)} \triangleq \mathbf{L}_t = \mathcal{R}_{\mathbf{G}_{\sigma_c}}^{K_r} \mathbf{U}_t + \left(1 - \tfrac{t \bmod R}{R}\right)\left(\mathbf{Z}_{f_1} - \mathcal{R}_{\mathbf{G}_{\sigma_c}}^{K_r} \mathcal{B}_F \mathcal{D}_F \mathcal{F}_{\mathbf{G}_{\sigma_c}} \mathbf{Z}_{f_1}\right) + \left(\tfrac{t \bmod R}{R}\right)\left(\mathbf{Z}_{f_2} - \mathcal{R}_{\mathbf{G}_{\sigma_c}}^{K_r} \mathcal{B}_F \mathcal{D}_F \mathcal{F}_{\mathbf{G}_{\sigma_c}} \mathbf{Z}_{f_2}\right) \tag{3.8}$$

Even this simple algorithm can yield tremendous results, especially for sequences like 'News' and 'Container' that is virtually free from large global motion. In other sequences like 'Foreman', the interpolation of HF information fails because the pixels that require the HF information have shifted spatially. This leads to the logical realization that it is possible to shift the HF information spatially as well as interpolate its value temporally. The next hierarchical method $\mathcal{H}^{(2)}$ or optical flow based estimation will follow this observation.

3.4 Optical Flow Based Estimation

It follows from Sec. 3.3 that it HF interpolation between HR key frames only works if the pixel remains spatially stationary. This is an assumption that is easily violated for many pixels in a video sequence; only background pixels satisfy this assumption. The pixels that failed to be reconstructed in the first hierarchical method linear estimation $\mathcal{H}^{(1)}$ are detected and thresholded through the reconstruction map $\mathbf{M}_t^{(2)}$. Since the reconstruction map marks pixels individually, a motion estimation algorithm that is computable locally is desired. Additionally, the estimation method must operate on the HF information and not directly on the HR pixels. Before the proposed estimation method is described, a review of optical flow and warping is presented. Following the generalization of these methods into functions, the optical flow based estimation method will be formulated.

3.4.1 Flow Computation

Borrowing on the established works of Lucas and Kanade [22] and referring from Fleet and Weiss [23], the optical flow in a hypothetical image sequence $\boxed{\mathbf{I}}_3$ can be computed between two frames indexes t and $t+1$. It is formulated as a least squares solution in a local region around each individual pixel $\hat{\mathbf{p}} = \langle p_{\vec{x}}, p_{\vec{y}} \rangle$. Under the constant intensity assumption, a pixel in one frame (when subject to motion) remains at the same intensity level in the next frame. This assumption is valid locally, and serves as a starting point for the algorithm. Given an image frame $\mathbf{I}_t$ at index t, the constant intensity assumption is expressed as:

$$\mathbf{I}_t \left[\hat{\mathbf{p}}\right] = \mathbf{I}_{t+1} \left[\hat{\mathbf{p}} + \hat{\mathbf{m}}\right] \tag{3.9}$$

where $\hat{\mathbf{p}}$ is the source pixel location, and $\hat{\mathbf{m}} = \langle m_{\vec{x}}, m_{\vec{y}} \rangle$ is the motion associated with that pixel which translates it from current frame t to the next frame $t+1$. Taking a first-order Taylor series approximation of $\boxed{\mathbf{I}}_3$ about the point $(p_{\vec{x}}, p_{\vec{y}}, t)$, and assuming that the motion vectors are small in magnitude, the following is true:

$$\begin{aligned}
\mathbf{I}_{t+1}\left[\hat{\mathbf{p}} + \hat{\mathbf{m}}\right] &\approx \mathbf{I}_t\left[\hat{\mathbf{p}}\right] + \langle m_{\vec{x}}, m_{\vec{y}}, 1 \rangle \cdot \left\langle \frac{\partial \boxed{\mathbf{I}}_3}{\partial \vec{x}}, \frac{\partial \boxed{\mathbf{I}}_3}{\partial \vec{y}}, \frac{\partial \boxed{\mathbf{I}}_3}{\partial \vec{t}} \right\rangle \Bigg|_{\left(p_{\vec{x}}, p_{\vec{y}}, t\right)} \\
&\approx \mathbf{I}_t\left[\hat{\mathbf{p}}\right] + m_{\vec{x}}\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{x}}\right) + m_{\vec{y}}\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{y}}\right) + \left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{t}}\right) \\
&\approx \mathbf{I}_t\left[\hat{\mathbf{p}}\right] + \hat{\mathbf{m}} \cdot \nabla \mathbf{I}_t\left[\hat{\mathbf{p}}\right] + \left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{t}}\right)
\end{aligned} \tag{3.10}$$

The t^{th} frame image derivatives of in the $\vec{x}$, $\vec{y}$, and $\vec{t}$ directions must be evaluated at the pixel coordinate $\hat{\mathbf{p}}$, and are given by the following shorthand(s):

$$\begin{aligned}
\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{x}}\right) &= \left(\frac{\partial \boxed{\mathbf{I}}_3}{\partial \vec{x}}\right) \Bigg|_{\left(p_{\vec{x}}, p_{\vec{y}}, t\right)} \\
\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{y}}\right) &= \left(\frac{\partial \boxed{\mathbf{I}}_3}{\partial \vec{y}}\right) \Bigg|_{\left(p_{\vec{x}}, p_{\vec{y}}, t\right)} \\
\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{t}}\right) &= \left(\frac{\partial \boxed{\mathbf{I}}_3}{\partial \vec{t}}\right) \Bigg|_{\left(p_{\vec{x}}, p_{\vec{y}}, t\right)}
\end{aligned} \tag{3.11}$$

Next, substituting the intensity assumption from Eq. 3.9 into Eq. 3.10 yields our constraint equation:

$$\hat{\mathbf{m}} \cdot \nabla \mathbf{I}_t\left[\hat{\mathbf{p}}\right] + \left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{p}}\right]}{\partial \vec{t}}\right) = 0 \tag{3.12}$$

The goal of solving for the flow vector $\hat{\mathbf{m}}$ is problematic because there is only one equation, but two variables $m_{\vec{x}}$ and $m_{\vec{y}}$ to solve for; the system is under determined. To solve this dilemma, we build a $(\overline{w} \times \overline{w})$ window of pixels centered around the pixel coordinate $\hat{\mathbf{p}}$ that we are interested in. This relationship can be easily described using

the set $\psi_{\overline{w}}(\hat{\mathbf{p}})$. The definition is available in Eq. 1 in the preface.

The new pixel additions can be grouped into a vector list: with the k^{th} additional pixel coordinate represented by $\hat{\mathbf{q}}^{(k)} = \left\langle q_{\vec{x}}^{(k)}, q_{\vec{y}}^{(k)} \right\rangle = \psi_{\overline{w}}\left(\hat{\mathbf{p}}\right)\{k\}$ (order not important). For each pixel in the window, we generate a new equation, each having the same desired flow vector components $m_{\vec{x}}$ and $m_{\vec{y}}$, as shown below in Eq. 3.13

$$m_{\vec{x}}\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{q}}^{(k)}\right]}{\partial \vec{x}}\right) + m_{\vec{y}}\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{q}}^{(k)}\right]}{\partial \vec{y}}\right) = -\left(\frac{\partial \mathbf{I}_t\left[\hat{\mathbf{q}}^{(k)}\right]}{\partial \vec{t}}\right) \tag{3.13}$$

Combining all $\overline{w}^2$ equations into a linear system is simple and straightforward:

$$\underbrace{\begin{bmatrix} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(1)}]}{\partial \vec{x}}\right) & \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(1)}]}{\partial \vec{y}}\right) \\ \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(2)}]}{\partial \vec{x}}\right) & \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(2)}]}{\partial \vec{y}}\right) \\ \vdots & \vdots \\ \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(\overline{w}^2)}]}{\partial \vec{x}}\right) & \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(\overline{w}^2)}]}{\partial \vec{y}}\right) \end{bmatrix}}_{\mathbf{A}} \cdot \begin{bmatrix} m_{\vec{x}} \\ m_{\vec{y}} \end{bmatrix} = -\underbrace{\begin{bmatrix} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(1)}]}{\partial \vec{t}}\right) \\ \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(2)}]}{\partial \vec{x}}\right) \\ \vdots \\ \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(\overline{w}^2)}]}{\partial \vec{t}}\right) \end{bmatrix}}_{\hat{\mathbf{b}}} \tag{3.14}$$

$$\mathbf{A}\hat{\mathbf{m}} = \hat{\mathbf{b}}$$

The resultant linear system is now overdetermined: a problem that can easily solved using a least squares (LS) solution:

$$\hat{\mathbf{m}} = \left(\mathbf{A}^{\mathrm{T}}\mathbf{A}\right)^{-1}\mathbf{A}^{\mathrm{T}}\hat{\mathbf{b}} \tag{3.15}$$

The final vector $\hat{\mathbf{m}}$ will contain the column motion vector calculated at the pixel $\hat{\mathbf{p}}$.

Simplifying the final solution can be done by expanding (multiplying) the matrices:

$$\hat{\mathbf{m}} = \begin{bmatrix} \sum_{k=1}^{\overline{w}^2} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{x}}\right)^2 & \sum_{k=1}^{\overline{w}^2} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{x}}\right)\left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{x}}\right) \\ \sum_{k=1}^{\overline{w}^2} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{y}}\right)\left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{x}}\right) & \sum_{k=1}^{\overline{w}^2} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{y}}\right)^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -\sum_{k=1}^{\overline{w}^2} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{x}}\right)\left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{t}}\right) \\ -\sum_{k=1}^{\overline{w}^2} \left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{y}}\right)\left(\frac{\partial \mathbf{I}_t[\hat{\mathbf{q}}^{(k)}]}{\partial \vec{t}}\right) \end{bmatrix} \tag{3.16}$$

The well known aperture problem occurs when the (2×2) matrix in Eq. 3.16 is rank deficient; the matrix becomes singular and no unique solution can be found. This often occurs when image gradients are parallel, and often regularization, higher order motion models or iterative solutions are employed. In this work, the flow field for a singular matrix is set to be 0 in both spatial directions $\vec{x}$ and $\vec{y}$.

3.4.1.1 Implementation Details

The optical flow solution shown in Eq. 3.16 is subject to errors because each pixel considered in the window has an equal weighting. A common approach is to modulate the pixels with a 2D symmetric Gaussian kernel $\mathbf{G}_{\overline{\sigma}}$ so that the least squares solution matches the center pixel with the highest priority. Additionally, to generalize the procedure and avoid unnecessary notation, a generic single pixel optical flow function $\hat{\mathbf{f}}_{\overline{\sigma}}^{\overline{w}}\left(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \hat{\overline{\mathbf{p}}}\right)$ will be defined to take the following parameters: the reference frame $\overline{\mathbf{A}}$, the destination frame $\overline{\mathbf{B}}$, the standard deviation $\overline{\sigma}$ for the Gaussian kernel $\mathbf{G}_{\overline{\sigma}}$

used in weighting the estimate pixels. Both modifications are shown below:

$$\hat{\mathbf{f}}_{\overline{\sigma}}^{\overline{w}}\left(\overline{\mathbf{A}}\,,\,\overline{\mathbf{B}}\,,\,\hat{\overline{\mathbf{p}}}\right) = \begin{bmatrix} \sum_{k=1}^{\overline{w}^2} \mathbf{G}_{\overline{\sigma}}[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}]\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{x}}\right)^2 & \sum_{k=1}^{\overline{w}^2} \mathbf{G}_{\overline{\sigma}}[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}]\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{x}}\right)\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{y}}\right) \\ \sum_{k=1}^{\overline{w}^2} \mathbf{G}_{\overline{\sigma}}[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}]\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{y}}\right)\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{x}}\right) & \sum_{k=1}^{\overline{w}^2} \mathbf{G}_{\overline{\sigma}}[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}]\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{y}}\right)^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} -\sum_{k=1}^{\overline{w}^2} \mathbf{G}_{\overline{\sigma}}[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}]\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{x}}\right)\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{t}}\right) \\ -\sum_{k=1}^{\overline{w}^2} \mathbf{G}_{\overline{\sigma}}[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}]\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{y}}\right)\left(\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{t}}\right) \end{bmatrix} \tag{3.17}$$

The vector difference $\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}$ modulates the Gaussian function so that iterated pixels $\hat{\mathbf{q}}^{(k)}$, that are spatially close to $\hat{\mathbf{p}}$, receive higher weights. To preserve the tensor notation previously used, $\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}}$ will represent the joining of the two image frames $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ to produce a sequence (tensor), maintaining the ability to take derivatives in the time $\vec{t}$ direction.

Derivatives can be computed using finite central differences for the $\vec{x}$ and $\vec{y}$ directions. However, in the $\vec{t}$ direction, forward differences is the only logical choice. To compute backwards optical flow, simply reverse the positions of the input matrices $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$.

3.4.2 Field Warp

Once an optical flow field has been determined, a warping algorithm must be devised that applies the motion field to a target image, warping it into a similar image that the field was calculated from. Naively moving each pixel, from its source location to the destination location in the destination frame, using the flow field, produces

'cracks'. Cracks are defined as pixels in the final warped output that never had any data written to it because nothing in the original reference image mapped to it (via the applied flow field). Flow fields do not (except under very special conditions) give a 1:1 correspondence mapping between a reference frame and source frame. In fact, flow fields will often map adjacent source pixels to the same position in the destination frame, and as a result cause 'cracks' in the final warped output (due to lack of pixels elsewhere).

A simple way to combat this problem, is to warp patches from a source frame to the destination frame. The patches can be blended and applied using a simple 2D symmetric Gaussian kernel. The downside is that computation complexity will increase significantly depending on the side of the Gaussian kernel used. An optical warp function can be defined similar to the optical flow equation:

$$\hat{\mathbf{w}}_{\overline{\sigma}}^{\overline{w}}\left(\overline{\mathbf{A}}\,,\,\overline{\mathbf{M}}\,,\,\hat{\overline{\mathbf{p}}}\right)=\frac{\sum\limits_{\hat{\overline{\mathbf{c}}}\in\phi\left(\overline{\mathbf{A}}\right)}\;\sum\limits_{\hat{\overline{\mathbf{g}}}\in\psi_{\overline{w}}\left(\hat{\overline{\mathbf{c}}}\right)}\;\sum\limits_{\hat{\overline{\mathbf{c}}}+\hat{\overline{\mathbf{g}}}+\overline{\mathbf{M}}[\hat{\overline{\mathbf{c}}}]=\hat{\overline{\mathbf{p}}}}\mathbf{G}_{\overline{\sigma}}\left[\hat{\overline{\mathbf{c}}}-\hat{\overline{\mathbf{g}}}\right]\overline{\mathbf{A}}\left[\hat{\overline{\mathbf{c}}}+\hat{\overline{\mathbf{g}}}\right]}{\sum\limits_{\hat{\overline{\mathbf{c}}}\in\phi\left(\overline{\mathbf{A}}\right)}\;\sum\limits_{\hat{\overline{\mathbf{g}}}\in\psi_{\overline{w}}\left(\hat{\overline{\mathbf{c}}}\right)}\;\sum\limits_{\hat{\overline{\mathbf{c}}}+\hat{\overline{\mathbf{g}}}+\overline{\mathbf{M}}[\hat{\overline{\mathbf{c}}}]=\hat{\overline{\mathbf{p}}}}\mathbf{G}_{\overline{\sigma}}\left[\hat{\overline{\mathbf{c}}}-\hat{\overline{\mathbf{g}}}\right]} \tag{3.18}$$

This formulation takes as parameters the reference image to warp $\overline{\mathbf{A}}$, the motion vectors previously calculated $\overline{\mathbf{M}}$, the pixel coordinate to calculate the optical warp at: $\hat{\overline{\mathbf{p}}}$, as well as the blending function parameters: the window size $\overline{w}$ and standard deviation $\overline{\sigma}$. It works by doing a sum of all pixels that end up being moved to $\hat{\overline{\mathbf{p}}}$ (via motion vectors), including those that belong to the blending Gaussian kernel. Refer to Eq. 1 and Eq. 2 in the preface for the definition of the sets $\psi_{\overline{w}}\left(\hat{\overline{\mathbf{c}}}\right)$ and $\phi\left(\overline{\mathbf{A}}\right)$ respectively.

3.4.3 Estimation Formulation

Using the two tools previously established, optical flow and optical warp, the optical flow based estimation method will take HF information in key frames and 'warp' it to non-key-frames. There are two different configurations for this requirement, outlined below:

1. Calculate optical flow/warp directly between HR key-frame and LR non-key-frame (HR-KF to LR) shown in Fig. 3.4.

2. Calculate optical flow/warp successively between all LR frames (LR to LR) shown in Fig. 3.5.

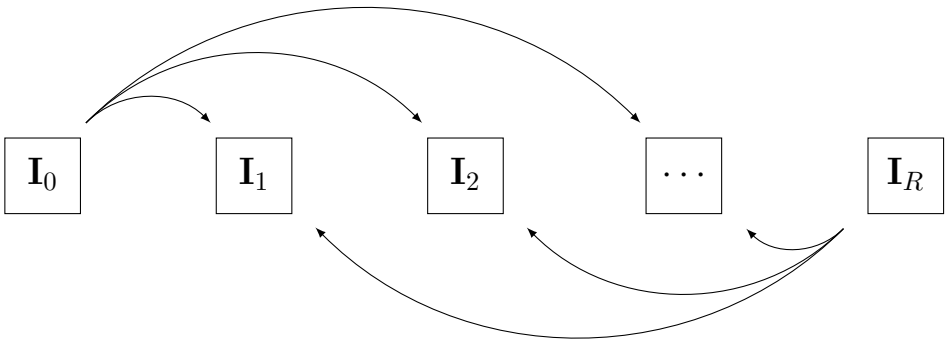


Figure 3.4: Optical flow configuration (HR-KF to LR)

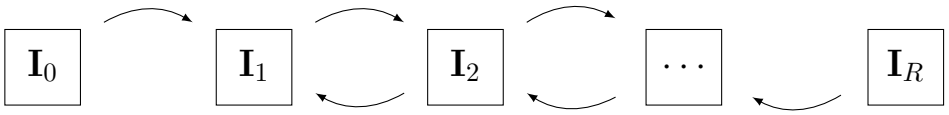


Figure 3.5: Optical flow configuration (LR to LR)

In the first configuration (HR-KF to LR) Fig. 3.4, optical flow/warp is bidirectionally computed between all pairs of HR key frames and LR frames. The total number

of computations is trivial to compute: $2(R-1)$. Additionally, the flow calculation between frames that have large temporal difference must use larger search areas. The larger search areas compensate for the larger motion vectors expected between LR and HR frames. The coarse-to-fine approach, often used in optical flow calculations, handles the problem of large and small motion vectors, with a small overhead. However, since the goal is to warp HF information to each LR in between HR key-frames, there is an opportunity to re-use some of the results obtained at each stage, using the second scheme: LR to LR.

In the second scheme Fig. 3.5, optical flow is bidirectionally computed between adjacent LR frames only. The total number of computations for optical flow is computed to be $2(R-1)$, the same as in the first scenario. However, since the source frames are always temporally adjacent, a reduction in computation time or an increase in motion vector accuracy can be realized. The reduction in computation time stems from the reduced search space for motion vectors, and the increase in motion vector accuracy derives from the linearizion of motion vectors when considering temporally adjacent frames. The major disadvantage of the second scheme will be discussed in the NLM stage, with a corresponding solution.

Using the LR-LR configuration and given a LR frame index to reconstruct, two formulations for the optical based flow estimate can be formed. The first is formed by warping high frequency information from the first adjacent HR-KF, while the second is formed using the second adjacent HR-KF. The final optical flow based estimate fuses these two partial estimates. The forward partial estimate is shown in Fig. 3.6, while the backwards partial estimate is a mirror image of that procedure, starting at a HR key-frame and propagating HF information backwards in time.

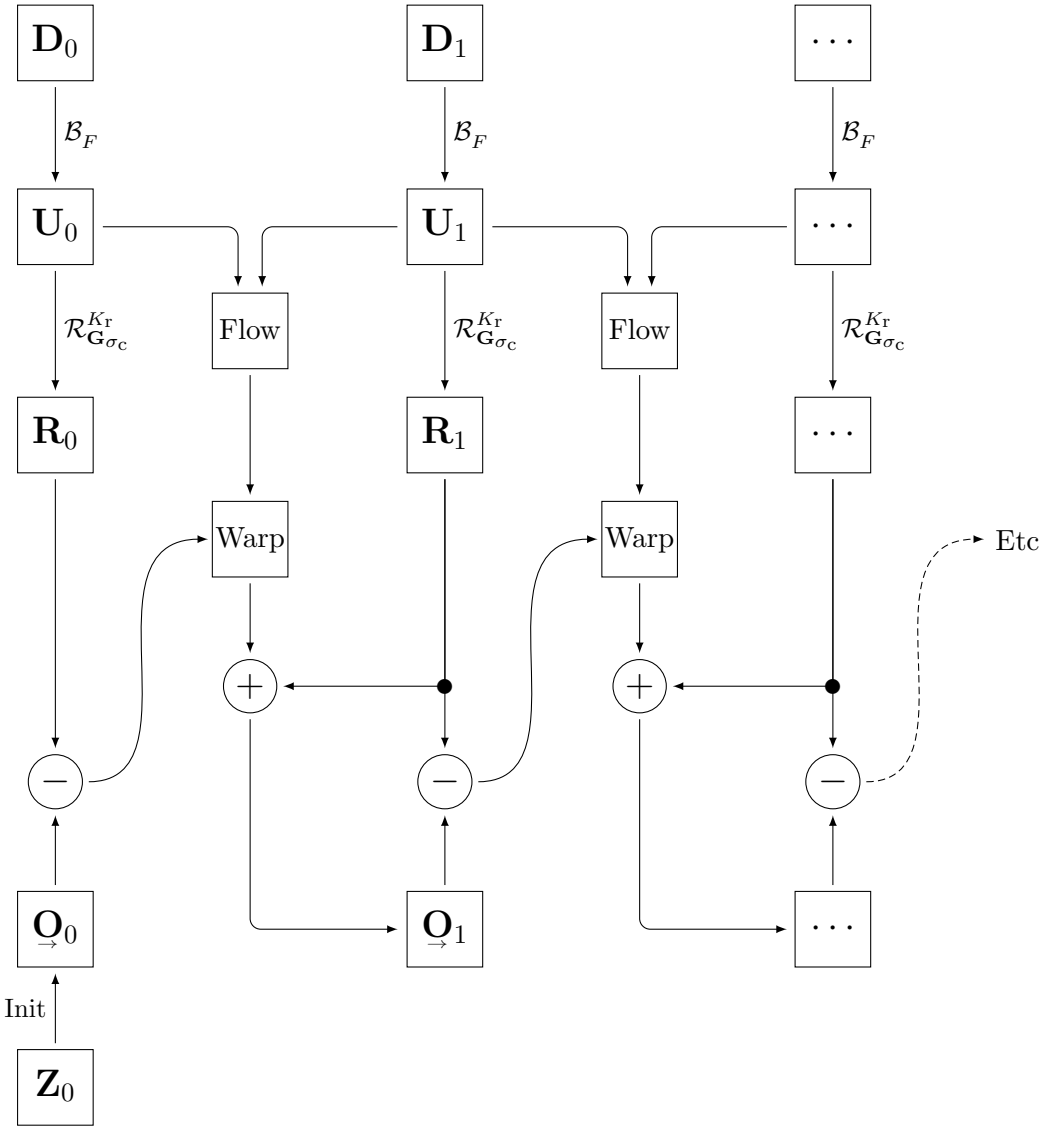


Figure 3.6: Optical flow based estimation: forward partial estimation.

Mathematically, both methods are shown in Eq. 3.19 below:

$$\underset{\rightarrow}{\mathbf{O}}_t[\hat{\mathbf{p}}] = \begin{cases} \mathbf{Z}_{\frac{t}{R}}[\hat{\mathbf{p}}] \;\text{ if }\; t \equiv 0 \mod R \\ \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}}\mathbf{U}_t[\hat{\mathbf{p}}] + \hat{\mathbf{w}}^{W_\mathrm{w}}_{\sigma_\mathrm{w}}\Big((\underset{\rightarrow}{\mathbf{O}}_{t-1} - \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}}\mathbf{U}_{t-1})\,,\hat{\mathbf{f}}^{W_\mathrm{f}}_{\sigma_\mathrm{f}}(\mathbf{U}_{t-1}\,,\mathbf{U}_t\,,\hat{\mathbf{p}})\,,\hat{\mathbf{p}}\Big) \end{cases}$$
$$\underset{\leftarrow}{\mathbf{O}}_t[\hat{\mathbf{p}}] = \begin{cases} \mathbf{Z}_{\frac{t}{R}}[\hat{\mathbf{p}}] \;\text{ if }\; t \equiv 0 \mod R \\ \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}}\mathbf{U}_t[\hat{\mathbf{p}}] + \hat{\mathbf{w}}^{W_\mathrm{w}}_{\sigma_\mathrm{w}}\Big((\underset{\rightarrow}{\mathbf{O}}_{t+1} - \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}}\mathbf{U}_{t+1})\,,\hat{\mathbf{f}}^{W_\mathrm{f}}_{\sigma_\mathrm{f}}(\mathbf{U}_{t+1}\,,\mathbf{U}_t\,,\hat{\mathbf{p}})\,,\hat{\mathbf{p}}\Big) \end{cases} \tag{3.19}$$

Forward estimation begins with the initialization of $\underset{\rightarrow}{\mathbf{O}}_t$ where the index t has a corresponding HR keyframe: $\mathbf{Z}_{\frac{t}{R}}$ where $t \equiv 0 \mod R$. Subtracting the initialized forward estimate $\underset{\rightarrow}{\mathbf{O}}_t$ from its corresponding partially deblurred upsampled image $\mathbf{R}_t$ yields the high frequency components of the forward estimate $\underset{\rightarrow}{\mathbf{O}}_t$. This is due to the fact that the deblurred upsampled image $\mathbf{R}_t$ is the low frequency content of $\underset{\rightarrow}{\mathbf{O}}_t$, thus their difference yields the remaining high frequency content. Using the optical flow fields generated from $\mathbf{U}_t$ to $\mathbf{U}_{t+1}$, the high frequency content is warped to the next index $t+1$. Finally, the next frame for the forward estimate is formed when combining the warped high frequency information with the next partially deblurred upsampled frame $\mathbf{R}_{t+1}$. The process is repeated until the next HR key-frame is encountered.

Due to errors in the optical flow/warp calculations, each iteration of the forward estimation will accumulate these errors, and produce an image that degrades over iterations. For the forward partial estimation, the first forward iteration will yield the best results. Each successive warp will be lower in quality, and the last iteration will yield the worst results. The backwards partial estimation will exhibit the same properties. In both cases, the number of warps between two adjacent HR frames is limited to $R-1$.

However, we can exploit the best of both worlds by fusing the two estimates into the final optical flow based estimate $\mathbf{O}_t$. The fusion blends the HF information present in both estimates using the following generic blending function:

$$b_k(x) = \tfrac{1}{2}\big(1 + \mathrm{erf}\big(kx - \tfrac{k}{2}\big)\big) \tag{3.20}$$

where the parameter k in the blending function controls the blending strength. In the limit, as $k \to \infty$, the blending functions becomes a rectangular step function. This blending function was chosen because through the blending strength parameter k, curves similar to linear interpolation and cosine interpolation can be achieved. Additionally, the curve is continuous and well defined around $x = 1/2$. The blending function along with the linear and cosine blending functions are shown in Fig. 3.7 for comparison.

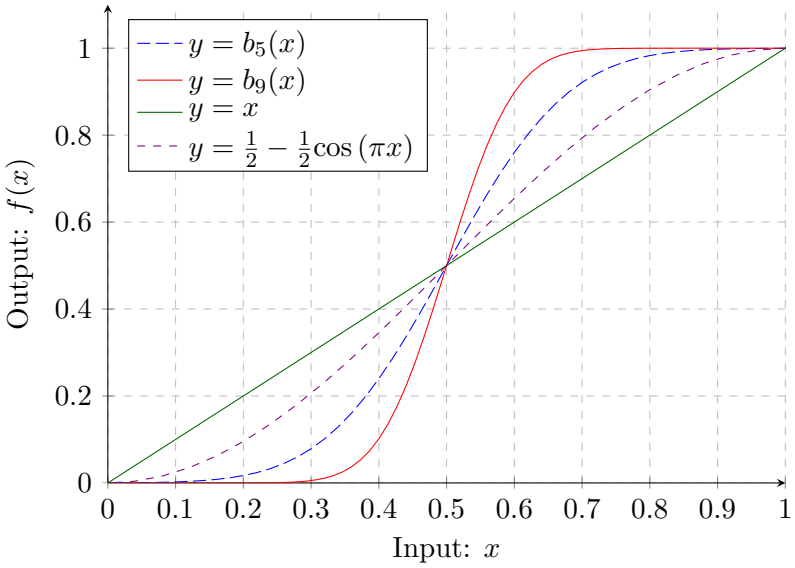


Figure 3.7: Visualization of the blending function $b_k(x)$.

Finally, the fused optical flow based estimate can be written as a blended version of the forward estimate and the backward estimate using the blending function $b_{K_\mathrm{b}}\left(\frac{t \bmod R}{R}\right)$ as shown below in equation Eq. 3.21:

$$\mathbf{Y}_t^{(2)} \triangleq \mathbf{O}_t = \left(1 - b_{K_\mathrm{b}}\left(\frac{t \bmod R}{R}\right)\right)(\underset{\rightarrow}{\mathbf{O}}t) + \left(b_{K_\mathrm{b}}\left(\frac{t \bmod R}{R}\right)\right)(\underset{\leftarrow}{\mathbf{O}}t) \tag{3.21}$$

where the parameter K_b modulates the strength of the blending. The advantage of the final fusion in Eq. 3.21 can be seen in Fig. 3.8. The fusion of the forward estimate and backwards estimate yields a better estimate then using the forward or backwards estimates individually. The 'Foreman' sequence was selected for illustration, with a R of 10, so that the 'bathtub' curve can be seen in the final fused estimate. Since the HF is borrowed from key-frames, the frames having small temporal differences to the key-frames are the ones that have the best performance.

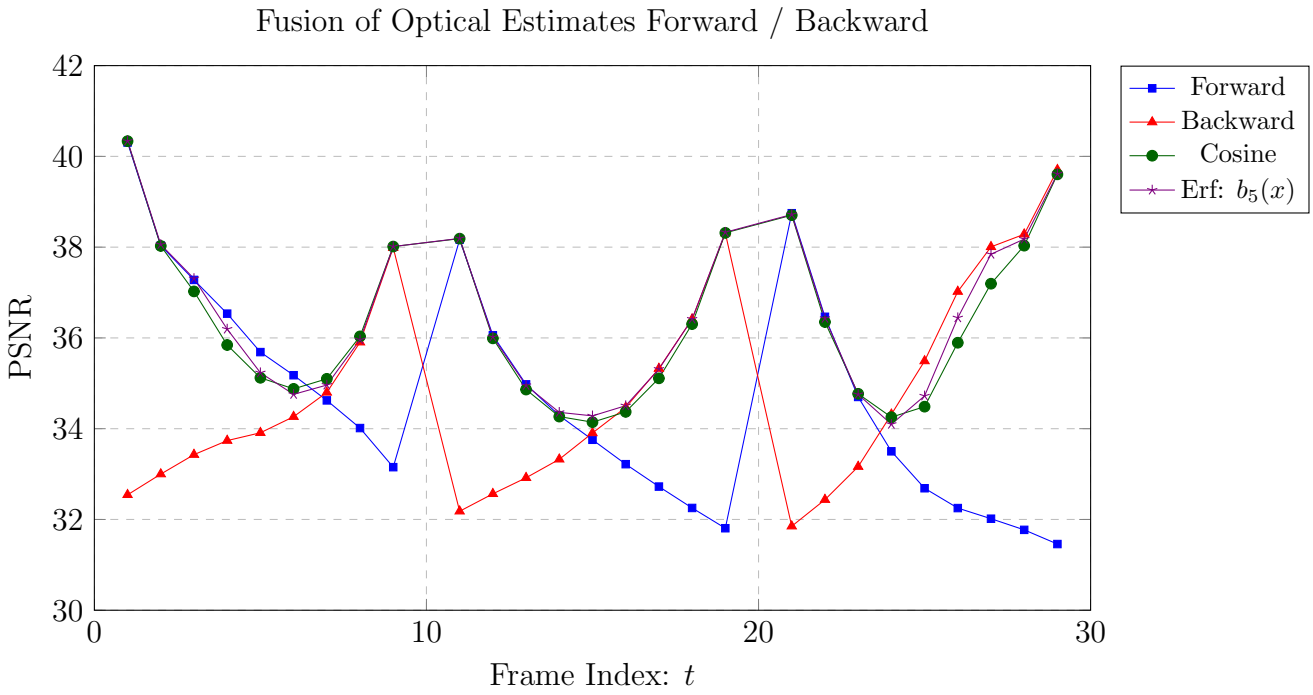


Figure 3.8: Optical flow based estimation - fusion 'bathtub' curves (R=10).

Originally, the intent was to use to interpolate the HF content between the forward and backward estimates during the optical estimation itself. However, this creates a recursive problem: the first frame of forward estimation cannot be interpolated until the last frame of the backwards estimation is known, since they represent the same time index. The last frame of backwards estimation is dependent on the first frame of backwards estimation through Eq. 3.19. By symmetry, the first frame of backwards estimation is related to the last frame of forward estimation, and by extension, the first frame of forward estimation, completing the circle. This circular dependency can still be solved using an iterative approach, however, a simpler solution was used. Interpolating the high frequency content after the individual estimates are formed (instead of during) leads to the solution in Eq. 3.21. Utilizing the cyclic iterative approach remains an experiment by itself, and thus will not be covered here.

3.4.3.1 Motion Quality

The optical flow derivation assumption is that the motion vectors that are calculated are small in magnitude. However, this assumption is easily violated in almost all video sequences, which causes large errors in the motion vectors calculated using Eq. 3.17.

The following quality function can be defined on a per-pixel basis, as function of the weighted error residue produced between the right and left sides of Eq. 3.14 when utilizing the motion vector $\hat{\mathbf{m}}$ calculated. In closed form, this is expressed as:

$$\hat{\mathbf{e}}_{\overline{\sigma}}^{\overline{w}}\left(\overline{\mathbf{A}}\,,\,\overline{\mathbf{B}}\,,\,\hat{\overline{\mathbf{p}}}\right)=\sum_{k=1}^{\overline{w}^2}\mathbf{G}_{\overline{\sigma}}\left[\hat{\mathbf{p}}-\hat{\mathbf{q}}^{(k)}\right]\Big(m_{\vec{x}}\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{x}}+m_{\vec{y}}\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{y}}+\frac{\partial(\overline{\mathbf{A}}\triangleright\overline{\mathbf{B}})[\hat{\mathbf{q}}^{(k)}]}{\partial\vec{t}}\Big) \tag{3.22}$$

The error identifies pixels that had a large mismatches in Eq. 3.14, and modifies the reconstruction map $\mathbf{M}_t^{(3)}\left[\hat{\mathbf{m}}\right]$ at a particular pixel $[\hat{\mathbf{m}}]$ that the optical flow calculated

is not reliable. However, since the use of bidirectional flow in the optical estimation, the reconstruction map must be modified in a special way. Specifically, the forward flow error and backward flow error must be fused into a final boolean mask as shown below:

$$\mathbf{M}_t^{(3)}\left[\hat{\mathbf{m}}\right] \leftarrow \mathbf{M}_t^{(3)}\left[\hat{\mathbf{m}}\right] \bigvee \mathcal{T}_{T_\mathrm{o}}\left(\hat{\mathbf{e}}_{\sigma_\mathrm{f}}^{W_\mathrm{f}}(\mathbf{U}_t\,,\,\mathbf{U}_{t+1}\,,\,\hat{\mathbf{m}}) + \hat{\mathbf{e}}_{\sigma_\mathrm{f}}^{W_\mathrm{f}}(\mathbf{U}_{t+1}\,,\,\mathbf{U}_t\,,\,\hat{\mathbf{m}})\right) \tag{3.23}$$

When the combined flow error (forward/backward) is outside of the threshold range T_o, the corresponding pixel $\hat{\mathbf{m}}$ will be marked in the reconstruction map $\mathbf{M}_t^{(3)}\left[\hat{\mathbf{m}}\right]$ after residue thresholding. This guarantees that improper flow is corrected in the NLM stage $\mathcal{H}^{(3)}$, through the use of the boolean 'OR'/disjunction operator $\bigvee$.

3.5 NLM Reconstruction

The NLM reconstruction method represents the third stage in the hierarchy model: $\mathcal{H}^{(3)}$. This is the heaviest of all the methods considered, and its usage must be limited to pixels that have failed to be reconstructed using the previous two methods: linear estimation and optical flow based estimation.

The formulation follows from previous work by Lengyel et al. [19], Najafi and Shirani [11] and Protter et al. [10]: the NLM reconstructed output is a weighted average of pixel contributions from the upsampled LR frames $\boxed{\mathbf{U}}_3$ and/or HR key-frames $\boxed{\mathbf{Z}}_3$. However it has been shown by [19] that contributions from upsampled LR images do not lead to a significant increase in performance for super resolution. The best pixel contributions for use in NLM come from the HR key-frames themselves, since they contain the most relevant information that has not been subject to degradation.

Each pixel marked with a 1 in the reconstruction map $\mathbf{M}_t^{(3)}[\hat{\mathbf{m}}]$ must be reconstructed using the modified NLM as shown below in Eq. 3.24.

$$\mathbf{Y}_t^{(3)}[\hat{\mathbf{m}}] \triangleq \mathbf{N}_t[\hat{\mathbf{m}}] = \frac{\sum\limits_{\langle \hat{\mathbf{n}}, k\rangle \in \Upsilon_t(\hat{\mathbf{m}})} w_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) \mathbf{Z}_k[\hat{\mathbf{n}}]}{\sum\limits_{\langle \hat{\mathbf{n}}, k\rangle \in \Upsilon_t(\hat{\mathbf{m}})} w_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})} \tag{3.24}$$

where $\mathbf{N}_t[\hat{\mathbf{m}}]$ is the NLM reconstructed output for a particular pixel coordinate index $\hat{\mathbf{m}} = \langle m_{\vec{x}}, m_{\vec{y}} \rangle$ for the frame index t. The equation operates by iterating through all pixel indexes $\hat{\mathbf{n}} = \langle n_{\vec{x}}, n_{\vec{y}} \rangle$ in HR frame-view indexes k that belong to the thresholded candidate set $\Upsilon_t(\hat{\mathbf{m}})$ (explained in Sec. 3.5.1), and assigning a normalized weight (explained in Sec. 3.5.3) to its respective HR frame pixel value given by $\mathbf{Z}_k[\hat{\mathbf{n}}]$. The summation of all of these contributions yields the reconstructed NLM output.

3.5.1 Thresholded Candidate Set

The thresholded candidate set $\Upsilon_t(\hat{\mathbf{m}})$ is the list of pixel indexes that are used for NLM reconstruction. The list features pixels from any/all HR key-frames and is built in three steps: candidate set generation, union of sets, and finally thresholding.

First, individual candidate sets $\beta_t^k(\hat{\mathbf{m}})$ are created, one for each HR frame-view index k using the input pixel index $\hat{\mathbf{m}}$. Individual candidate sets $\beta_t^k(\hat{\mathbf{m}})$ are formed by including all pixel indexes in an area centered on the input pixel coordinate index $\hat{\mathbf{m}}$, shifted by $\hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})$. Its definition is shown below in Eq. 3.25:

$$\beta_t^k(\hat{\mathbf{m}}) = \left\langle \psi_S\big(\hat{\mathbf{m}} + \hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})\big)\ ,\ k \right\rangle \tag{3.25}$$

The shift $\hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})$ represents a motion vector between a LR frame, and a target HR key-frame $\mathbf{Z}_k$ given by the frame index k. In other words, it tries to find where the pixel given by $\mathbf{U}_t\left[\hat{\mathbf{m}}\right]$ is in the HR key-frame $\mathbf{Z}_k$. Since the candidate set stage identifies a $(S \times S)$ region in which every pixel is checked for similarity, the motion vector shift only needs to be accurate enough to contain the best matched HR pixel information, where S is the candidate search size parameter associated with the algorithm. See Fig. 3.9 for a visual aid. The motion vector implementation of $\hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})$ is required to allow medium to large motion vectors to be captured and its definition appears later in Sec. 3.5.2. Note, all pixel indexes considered here are augmented with a third basis component, with value HR key-frame k, in order to distinguish them in the next step.

Second, the candidate sets are merged into a grand candidate set $\Lambda_t(\hat{\mathbf{m}})$ as per

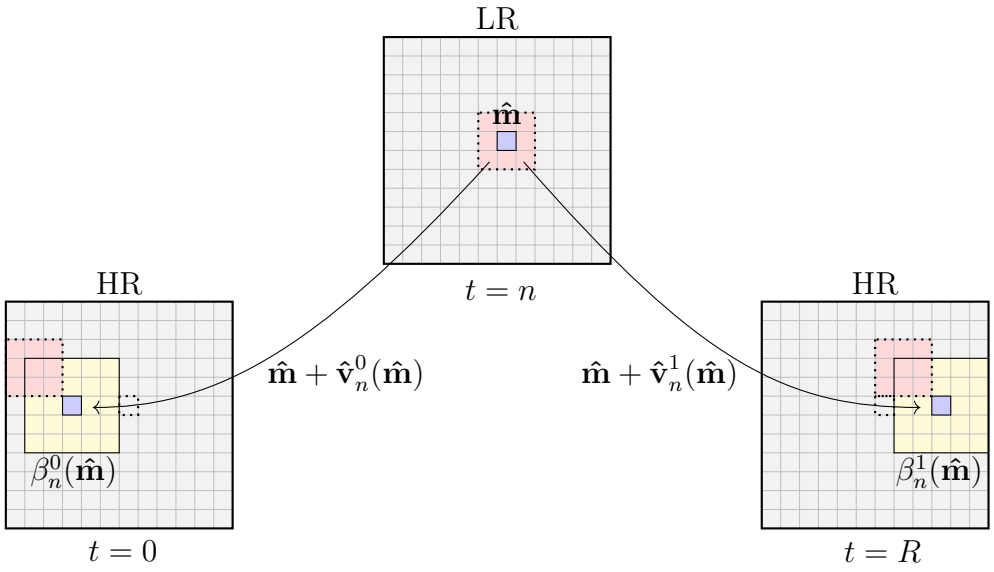


Figure 3.9: Candidate sets in HR key-frames formed by motion vectors.

Eq. 3.26. The union of sets focuses on the nearest two HR key-frames to the index t.

$$\Lambda_t(\hat{\mathbf{m}}) = \left\{ \bigcup_{k=\lfloor t/R \rfloor}^{\lceil t/R \rceil} \beta_t^k(\hat{\mathbf{m}}) \right\} \tag{3.26}$$

Finally, given all the pixel indexes $\langle \hat{\mathbf{n}}, k \rangle$ in the grand candidate set, we keep only those that meet the criteria that the custom error metric $\|\mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})\|_2^2$ associated with it, is less than two custom thresholds T_1, and T_2. The error metric and thresholding serves to reduce the total number of pixels processed, as well as, filter irrelevant pixels in the NLM, keeping only the best matched pixel sources. Their definitions appear in Sec. 3.5.3 and Sec. 3.5.4. respectively. Mathematically, the selection procedure generates the final thresholded candidate set $\Upsilon_t(\hat{\mathbf{m}})$ according to Eq. 3.27 below:

$$\Upsilon_t(\hat{\mathbf{m}}) = \left\{ \langle \hat{\mathbf{n}}, k \rangle \in \Lambda_t(\hat{\mathbf{m}}) \;\middle|\; \left\| \mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) \right\|_2^2 \leq \min\left(T_1, T_2\right) \right\} \tag{3.27}$$

Each pixel index that remains in the set $\Upsilon_t(\hat{\mathbf{m}})$ will be used for NLM reconstruction in Eq. 3.24. These pixels have been selected to be the best sources of pixel information for reconstruction.

3.5.2 Motion Vectors

The motion vector $\hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})$ from Eq. 3.25 is an important part of all SR algorithms. Lengyel et al. [19] demonstrates that under the right conditions, the motion vector between a LR frame and HR key-frame can be skipped (or set to $\langle 0, 0 \rangle$), and subsequently the final SR reconstruction will still be high quality. However, once the HR Rate parameter R gets higher, the reconstruction suffers from lack of candidates in the NLM, and thus the NLM will waste time and fail to reconstruct pixels, triggering the fallback condition. Therefore, it is imperative that the NLM candidate sets contain the relevant information for the reconstruction of a particular pixel. This condition is satisfied when the HR Rate R is low because of the candidate sets are S wide, but for higher HR Rate R, the solution must utilize a more robust approach: explicit motion vectors.

As outlined before, the solution here must be able to handle large motion between LR frames and HR key-frames. Reusing optical flow, a coarse-to-fine structure can be used, such that the finest level considered is downsampled so that one pixel represents half the size of the candidate search area size S. This heavy downsampling allows the optical flow vectors to remain small in magnitude, but represent large changes in the original upsampled $\mathbf{U}_t$ image. Additionally, the heavy downsampling cuts down the computation complexity.

The closed form motion vector $\hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})$ can be derived by tracking a hypothetical

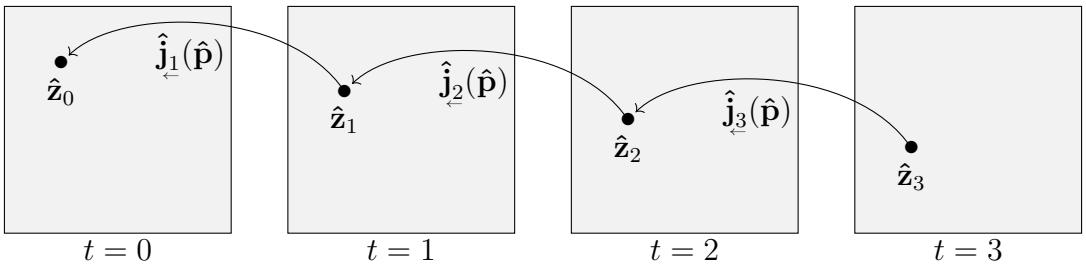


Figure 3.10: Jumping through a sequence using backward motion vector fields.

pixel index $\hat{\mathbf{z}}$ through the downsampled version of $\mathbf{D}_t$ starting at some arbitrary index t, using the jump table provided by optical flow on the heavily downsampled sequence $\mathbf{S}_t$, at the same frame index t. See Fig. 3.10 for a visual aid. The formal definition of $\mathbf{S}_t$ is shown below for reference:

$$\mathbf{S}_t = \mathcal{D}_F \mathcal{F}_{\mathbf{G}_{\sigma_c}} \mathbf{D}_t \tag{3.28}$$

Since $\hat{\mathbf{v}}_t^k(\hat{\mathbf{m}})$ is defined to be the vector difference between the pixel index $\hat{\mathbf{m}}$ for the frame index t and its corresponding pixel index in the HR key-frame for frame index k, its effect on the hypothetical pixel index $\hat{\mathbf{z}}_t$ can be written as the spatial index difference between $\hat{\mathbf{z}}_t$ and its corresponding motion vector shifted version in the k^{th} HR key-frame:

$$\hat{\mathbf{v}}_t^k(\hat{\mathbf{z}}_t) = \hat{\mathbf{z}}_{kR} - \hat{\mathbf{z}}_t \tag{3.29}$$

where R is the HR rate parameter. Relating all of the different pixel indexes $\hat{\mathbf{z}}_\forall$ through the sequence is done through the optical flow equation given in Eq. 3.17. Optical flow can be bidirectionally precomputed between all adjacent frames in the heavily downsampled sequence $\boxed{\mathbf{S}}_3$, and stored for reuse. A jump function[1] can be

[1]Equivalently the operation is single pixel warping between adjacent frames.

constructed so that the pixel $\hat{\mathbf{z}}_t$ can be directly related with its adjacent neighbors (indexes $t-1$ and $t+1$):

$$\begin{aligned} \hat{\mathbf{j}}_{\underset{\leftarrow}{t}}(\hat{\mathbf{p}}) &= F^2 \cdot \hat{\mathbf{f}}_{\sigma_\mathrm{f}}^{W_\mathrm{f}}\left(\mathbf{S}_t\,,\,\mathbf{S}_{t-1}\,,\,\frac{\hat{\mathbf{p}}}{F^2}\right) + \hat{\mathbf{p}} \qquad & \hat{\mathbf{z}}_t &= \hat{\mathbf{j}}_{\underset{\leftarrow}{t+1}}(\hat{\mathbf{z}}_{t+1}) \\ \hat{\mathbf{j}}_{\underset{\rightarrow}{t}}(\hat{\mathbf{p}}) &= F^2 \cdot \hat{\mathbf{f}}_{\sigma_\mathrm{f}}^{W_\mathrm{f}}\left(\mathbf{S}_t\,,\,\mathbf{S}_{t+1}\,,\,\frac{\hat{\mathbf{p}}}{F^2}\right) + \hat{\mathbf{p}} \qquad & \hat{\mathbf{z}}_t &= \hat{\mathbf{j}}_{\underset{\rightarrow}{t-1}}(\hat{\mathbf{z}}_{t-1}) \end{aligned} \tag{3.30}$$

The division and multiplication by F^2 serves to transform the original pixel coordinate $\hat{\mathbf{p}}$ to and from the downsampled frame $\mathbf{S}_t$, where F is the resolution difference between a HR key-frame and a downsampled LR frame, as well as between a LR frame and its heavily downsampled version $\mathbf{S}_t$. The implementation must use bicubic filtering to deal with the fractional nature of resulting pixel, as well as to deal with the problem encountered with odd sized images.

It follows from Fig. 3.10, Eq. 3.29 and Eq. 3.30 that the following relations are true for the hypothetical pixel $\hat{\mathbf{z}}$:

$$\begin{aligned} \hat{\mathbf{v}}_1^0(\hat{\mathbf{z}}_1) &= \hat{\mathbf{z}}_0 - \hat{\mathbf{z}}_1 = \hat{\mathbf{j}}_{\underset{\leftarrow}{1}}(\hat{\mathbf{z}}_1) && - \hat{\mathbf{z}}_1 \\ \hat{\mathbf{v}}_1^0(\hat{\mathbf{z}}_2) &= \hat{\mathbf{z}}_0 - \hat{\mathbf{z}}_2 = \hat{\mathbf{j}}_{\underset{\leftarrow}{1}}\big(\hat{\mathbf{j}}_{\underset{\leftarrow}{2}}(\hat{\mathbf{z}}_2)\big) && - \hat{\mathbf{z}}_2 \\ \hat{\mathbf{v}}_1^0(\hat{\mathbf{z}}_3) &= \hat{\mathbf{z}}_0 - \hat{\mathbf{z}}_3 = \hat{\mathbf{j}}_{\underset{\leftarrow}{1}}\big(\hat{\mathbf{j}}_{\underset{\leftarrow}{2}}\big(\hat{\mathbf{j}}_{\underset{\leftarrow}{3}}(\hat{\mathbf{z}}_3)\big)\big) && - \hat{\mathbf{z}}_3 \end{aligned} \tag{3.31}$$

Generalizing these relations yields the closed form equation for the motion vector that relates pixels indexes $\hat{\mathbf{p}}$ in LR frames to their counterparts in HR key-frames:

$$\hat{\mathbf{v}}_t^k(\hat{\mathbf{p}}) = \begin{cases} \hat{\mathbf{j}}_{\underset{\leftarrow}{kR+1}}\big(\hat{\mathbf{j}}_{\underset{\leftarrow}{kR+2}}\big(\cdots\hat{\mathbf{j}}_{\underset{\leftarrow}{t}}\,(\hat{\mathbf{p}})\cdots\big)\big) - \hat{\mathbf{p}} & \text{if} \quad kR < t \\ \hat{\mathbf{j}}_{\underset{\rightarrow}{kR-1}}\big(\hat{\mathbf{j}}_{\underset{\rightarrow}{kR-2}}\big(\cdots\hat{\mathbf{j}}_{\underset{\rightarrow}{t}}\,(\hat{\mathbf{p}})\cdots\big)\big) - \hat{\mathbf{p}} & \text{if} \quad kR > t \end{cases} \tag{3.32}$$

The vector can be computed by using the jump tables (precomputed bidirectional

optical flow MV) and iterating from the current frame index t to the nearest adjacent HR key-frames, making sure to follow the correct motion vector path, and filtering at each step to account for fractional values.

The computation time per pixel is constant since there will always be at most R jumps computed, since the sum of index distances from a frame index t to both nearest HR key-frames is constant. The flow precomputed (jump tables) can also be reused in the optical flow estimate as a flow prior. This allows large motions to be captured in the optical flow computation, and then subsequently refined if the reconstruction map value is 1.

While there are motion vector errors that will get propagated, it is expected that the NLM candidate search size S will provide compensation. For each reconstruction, the NLM will search the entire candidate space for matches, thus, as long as the target area contains the relevant pixel data, the motion vector can incur some propagated errors without any loss in NLM reconstruction quality.

3.5.2.1 Multi-view Candidate Set Extension

A simple way to extend the proposed algorithm to reconstruct multi-view sequences is to allow candidate sets from HR key-frames in other views. Lengyel et al. [19] proposed to use out-of-phase HR key-frame generation to distribute the availability of HR key-frames temporally and in the view direction. However, without the right motion vectors that relate LR frames to HR key-frames in other views, performance is not expected to increase significantly with the addition of extra views. See Table 4.3 in the Chapter 4 for a summary of multi-view results.

Considering candidate sets from HR key-frames is accomplished by promoting the

original HR key-frame index identifier k in Eq. 3.25 to be a vector that contains two pieces of information: the HR key-frame index paired with its view index: $\hat{\mathbf{k}} = \langle k_{\vec{t}}, k_{\vec{v}} \rangle$:

$$\beta_t^{\hat{\mathbf{k}}}(\hat{\mathbf{m}}) = \left\langle \psi_S\Big(\hat{\mathbf{m}} + \hat{\mathbf{v}}_t^{\hat{\mathbf{k}}}(\hat{\mathbf{m}})\Big)\,,\, \hat{\mathbf{k}} \right\rangle$$

The modified candidate set $\beta_t^{\hat{\mathbf{k}}}(\hat{\mathbf{m}})$ now relates the area $(S \times S)$ around pixel index $\hat{\mathbf{m}}$ in the t^{th} LR frame (for the current view that is being reconstructed) with the area formed by considering motion vectors to the HR key-frame with index $k_{\vec{t}}$ and view $k_{\vec{v}}$ identified via $\hat{\mathbf{k}}$.

Similar changes in notation must be made to the thresholded set in Eq. 3.27, the weighting function in Eq. 3.33, and finally the NLM in Eq. 3.24. The scope of the proposed method does not currently include multi-view sequences, and so the candidate set extension is only explored hypothetically. Lengyel et al. [19] uses this formulation (without motion vectors) and succeeds at utilizing HR key-frames in other views for better reconstructions over traditional single-view reconstructions.

3.5.3 Weighting Function

The weight $w_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$ is a measure of how much the HR key-frame pixel $\mathbf{Z}_k[\hat{\mathbf{n}}]$ resembles the pixel that we are trying to reconstruct: ideally $\mathbf{X}_t[\hat{\mathbf{m}}]$. However, since that information is not available because the original HR data is not known, we resort to comparing using degraded version of both the source pixel $\mathbf{X}_t[\hat{\mathbf{m}}]$ and the key frame pixel $\mathbf{Z}_k[\hat{\mathbf{n}}]$. The generic similarity structure can still be captured if the comparison is upgraded to use $(P \times P)$ size patches instead of single pixels, even if the data is partially degraded, where P is a parameter that dictates the size of the patch. The bicubicly interpolated LR frame $\mathbf{U}_t$ represents the degraded version of the original source pixels $\mathbf{X}_t$, while a degraded version of the HR key-frames $\mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_k$ can be built by artificially blurring using the known camera degradation.

The comparison is formulated as a function of the difference of two patches: one centered on the pixel $\hat{\mathbf{n}}$ in the degraded HR key-frame, and the other centered on the pixel $\hat{\mathbf{m}}$ in the upsampled LR frame. The difference between the two patches is $\mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$ and plays a key role in the definition of the weighting function shown in Eq. 3.33 below:

$$w_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) = \exp\left(\frac{-\|\mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})\|_2^2}{2DP^2}\right) \tag{3.33}$$

The D parameter modulates the strength of the errors so that less or more falloff can be achieved by the exponential function for larger errors. The differences $\mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$ are calculated by considering patch errors in the RGB channel $\mathbf{P}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$, the gradient channel $\mathbf{S}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$, and in the luminance channel $\mathbf{V}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$ as shown below in Eq. 3.34:

$$\mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) = \mathbf{P}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) \circ \mathbf{S}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) \circ \mathbf{V}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) \circ \mathbf{G}_{\sigma_c} \tag{3.34}$$

The differences utilize the gradient and luminance channels in order to give the weighting function a better value for comparisons. The error patch is built by using a Hadamard product on patches extracted from each channel as shown below in Eq. 3.35:

$$\begin{aligned}\mathbf{P}_k^t(\hat{\mathbf{m}},\hat{\mathbf{n}}) &= \left(\mathcal{X}_{\hat{\mathbf{m}}}^P\mathbf{U}_t \quad - \quad \mathcal{X}_{\hat{\mathbf{n}}}^P\mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_k\right)\\ \mathbf{S}_k^t(\hat{\mathbf{m}},\hat{\mathbf{n}}) &= \left(\mathcal{X}_{\hat{\mathbf{m}}}^P\mathcal{G}\mathbf{U}_t - \mathcal{X}_{\hat{\mathbf{n}}}^P\mathcal{G}\mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_k\right)\\ \mathbf{V}_k^t(\hat{\mathbf{m}},\hat{\mathbf{n}}) &= \left(\mathcal{X}_{\hat{\mathbf{m}}}^P\mathcal{L}\mathbf{U}_t - \mathcal{X}_{\hat{\mathbf{n}}}^P\mathcal{L}\mathcal{F}_{\mathbf{G}_{\sigma_c}}\mathbf{Z}_k\right)\end{aligned} \tag{3.35}$$

where $\mathcal{X}_{\hat{\mathbf{k}}}^P$ extracts a patch of size $(P \times P)$ from its adjoining matrix, centered on the input pixel coordinate index $\hat{\mathbf{k}}$. The operators $\mathcal{G}$ and $\mathcal{L}$ represent transforms on the adjoining matrix that return the Sobel gradient (magnitude) channel and luminance (Y) channel from a RGB to YUV conversion, respectively. Both transformations and the patch extract operator are available for reference in the preface.

Additionally, the error patch is modulated by a symmetric Gaussian filter $\mathbf{G}_{\sigma_\mathrm{p}}$ of size $(P \times P)$ with standard deviation σ_p which helps give priority to matching pixels that are geometrically closer to the center pixel index, while simultaneously giving lower weight value to those further away.

3.5.4 Adaptive Thresholding

The two thresholds T_1 and T_2 are adaptive thresholds that trim the number candidate pixels to include in the final NLM. The first threshold T_1 is obtained by finding the minimum patch error $\|\mathbf{E}_k^t(\hat{\mathbf{m}},\hat{\mathbf{n}})\|_2^2$ over all iterated pixel indexes $\hat{\mathbf{n}}$ in all HR frame indexes considered k and selecting only those candidate pixels that are close in value

to the minimum, as shown below in Eq. 3.36:

$$T_1 = A \min_{\langle \hat{\mathbf{n}}, k \rangle \in \Lambda_t(\hat{\mathbf{m}})} \left(\left\| \mathbf{E}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}}) \right\|_2^2 \right) \tag{3.36}$$

The leading factor A multiplies the minimum error by a constant amount, and enables the first threshold T_1 to capture only pixel candidates with low error. The weighting function in Eq. 3.33 also achieves the same thing, however, it is not selective enough. Given that the majority of pixels candidates are bad matches, it is clear that the cumulative contribution of numerous small weights will lead to blurry results. The first threshold T_1 eliminates the bad candidate pixel matches entirely when a few good candidate pixels are found.

The second threshold T_2 eliminates the case where all candidate pixels are bad. Using only the first threshold falsely allows candidates to be included in the NLM. The second threshold T_2 is described below in Eq. 3.37:

$$T_2 = \frac{K_1}{\log\left(1 + K_2 \|\mathbf{P}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})\|_2^2\right)} \tag{3.37}$$

where the RGB patch error $\mathbf{P}_k^t(\hat{\mathbf{m}}, \hat{\mathbf{n}})$ is taken from Eq. 3.35. Here, the patch size considered is S, which is the candidate patch size parameter associated with the algorithm. It is important to note that for a single pixel reconstruction, there is one value for T_1, but two values for T_2. There are two because there are two adjacent HR key-frames for which the RGB patch error is applied. Therefore, thresholding using T_2 must consider which HR key-frame $\mathbf{Z}_k$ the pixel information is coming from, according to Eq. 3.27.

The second threshold function T_2 can be seen visually in Fig. 3.11 and was built

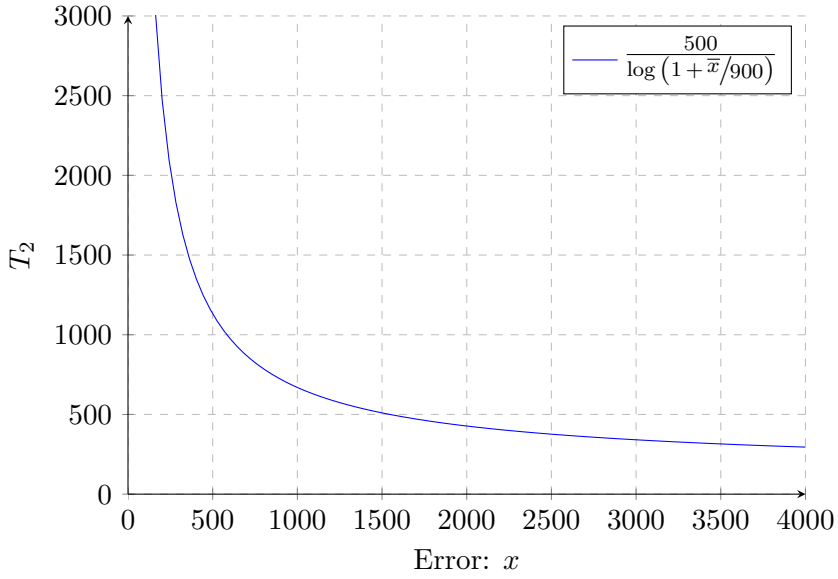


Figure 3.11: Second threshold T_2 function ($K_1 = 500$, $K_2 = 1/900$).

experimentally using the parameters $K_1 = 500$ and $K_2 = 1/900$. It works by computing a similarity measure between the area in the reference frame $\mathbf{U}_t$ centered on $\hat{\mathbf{m}}$ and source frame(s) $\mathbf{Z}_k$ centered on the translated coordinate $\hat{\mathbf{n}}$. The similarity metric does not consider gradient or luminance channels, nor uses a normalized Gaussian weight matrix because it tries to match any relevant pixels whether far or close. The second threshold reflects the expectation of finding similar pixels in the HR key-frames considered.

For cases with occlusion, large motion vectors, or scene changes, the list of candidate pixels will include only bad matches (high errors), and the second threshold T_2 attempts to eliminate that through the very special fallback condition that occurs in Eq. 3.38 below:

$$T_2 < \frac{T_1}{A} \tag{3.38}$$

During the fallback condition, the second threshold T_2 is lower than the minimum

candidate pixel patch error possible, making the thresholded candidate set $\Upsilon_t(\hat{\mathbf{m}})$ size to 0. As mentioned before, there are two values for the second threshold T_2, and thus the fallback condition only happens when both T_2 values are small enough. If the fallback condition is encountered, the pixel will be reconstructed with the fallback method explained in Sec. 3.6. To signal this event, the quality map (explained shortly in Sec. 3.5.5), is marked with a 0 to indicate the lowest quality.

3.5.5 Reconstruction Quality

A byproduct of the NLM reconstruction is the reconstruction quality matrix $\mathbf{Q}_t[\hat{\mathbf{m}}]$ for each frame index t reconstructed. It is a feedback mechanism to differentiate between pixels that are properly reconstructed and those that are not. The reconstruction quality matrix is defined as a function of the average of all patch errors $\mathbf{Q}_t[\hat{\mathbf{m}}]$ for all pixels included in the thresholded candidate $\Upsilon_t(\hat{\mathbf{m}})$ set as shown below in Eq. 3.39:

$$\mathbf{Q}_t[\hat{\mathbf{m}}] = \exp\left(\frac{-1}{K_3}\left(\frac{\sum_{\langle \hat{\mathbf{n}},k\rangle \in \Upsilon_t(\hat{\mathbf{m}})} \|\mathbf{E}_k^t(\hat{\mathbf{m}},\hat{\mathbf{n}})\|_2^2}{|\Upsilon_t(\hat{\mathbf{m}})|}\right)^2\right) \tag{3.39}$$

After the quality function has been computed, this information must propagate to the reconstruction map $\mathbf{M}_t^{(4)}$, after the thresholding has been processed. This guarantees that pixels that were not reconstructed with sufficient quality, are then forwarded to the final stage of reconstruction: fallback. The purpose of the thresholding stage is to detect these errors as well, however, due to degradation involved, it is possible that the error remains undetected. The process of correcting the reconstruction map, after the initial residue thresholding, guarantees that the errors in the NLM will be

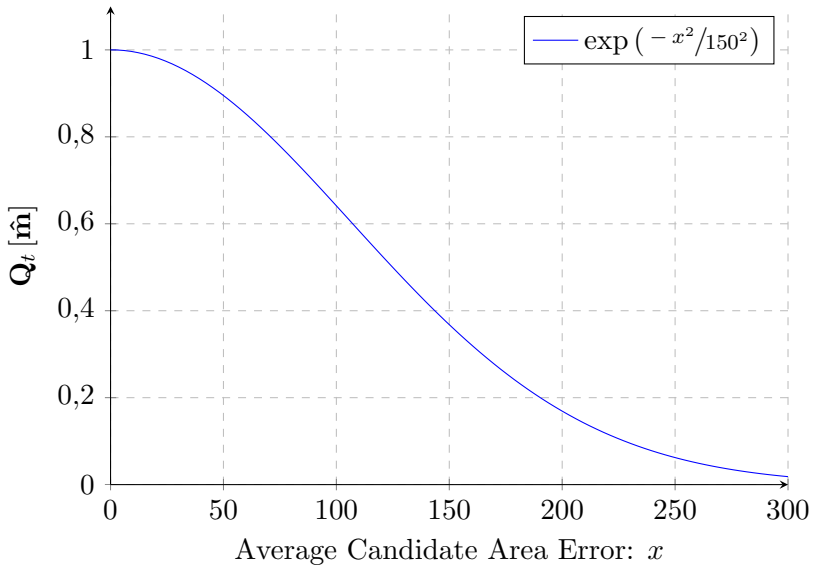


Figure 3.12: Reconstruction quality function for the NLM ($K_3 = 150^2$).

fixed with the fallback, as shown below in Eq. 3.40:

$$\mathbf{M}_t^{(4)}[\hat{\mathbf{m}}] \leftarrow \mathbf{M}_t^{(4)}[\hat{\mathbf{m}}] \bigvee \mathcal{T}_{T_\mathrm{n}}(1 - \mathbf{Q}_t[\hat{\mathbf{m}}]) \tag{3.40}$$

where the operator $\bigvee$ represents binary 'OR' or logical disjunction between its operands. It updates the current reconstruction map, after residue thresholding, with the pixels that failed the quality test. The threshold T_n controls how much reconstruction error is tolerated before marking a particular pixel to be reconstructed using the final reconstruction method: fallback.

3.6 Fallback Reconstruction

Any pixel that fails to be reconstructed by any of the preceding methods (linear estimation, optical flow based estimation, NLM) will be reconstructed using the

Richardson-Lucy deconvolution method described in Richardson [20] and Lucy [21]. Pixels that 'fallback' on this method are loner pixels; these pixels have no similar counterparts in the adjacent frames. Pixels that change intensity rapidly between HR frames are also difficult to reconstruct using all of the previous methods.

While these pixels are difficult to reconstruct, they are also rare in number, and thus a compromise can be reached. They can be simply reconstructed using the upsampled frames $\mathbf{U}_t$ themselves, without incurring too much loss in the quality of the final reconstruction; it is the best reconstruction method that remains being only marginally better than the bicubic upsampling.

The widely used closed form version is shown below for reference:

$$\begin{aligned} \mathbf{R}_t &= \mathcal{R}^{K_\mathrm{r}}_{\mathbf{G}_{\sigma_\mathrm{c}}} \mathbf{U}_t \\ \mathcal{R}^{\overline{k}+1}_{\mathbf{G}_{\overline{\sigma}}} \overline{\mathbf{I}} &= \mathcal{R}^{\overline{k}}_{\mathbf{G}_{\overline{\sigma}}} \overline{\mathbf{I}} \left(\frac{\overline{\mathbf{I}}}{\mathcal{R}^{\overline{k}}_{\mathbf{G}_{\overline{\sigma}}} \overline{\mathbf{I}} \otimes \mathbf{G}_{\overline{\sigma}}} \otimes (\overleftrightarrow{\mathbf{G}}_{\overline{\sigma}}) \right) \end{aligned} \tag{3.41}$$

where $\otimes$ represents convolution. Due to the iterative nature of the closed form, the initial value for $\mathcal{R}^0_{\mathbf{G}_{\overline{\sigma}}} \overline{\mathbf{I}}$ can be chosen as simply the input image $\overline{\mathbf{I}}$ to the deconvolution, in this case, the upsampled LR image $\mathbf{U}_t$. Additionally, the method requires the use of the original blurring kernel $\mathbf{G}_{\overline{\sigma}}$ that was applied to $\overline{\mathbf{I}}$. The modified kernel $\overleftrightarrow{\mathbf{G}}_{\overline{\sigma}}$ is just a flipped in both basis directions and if the kernel is symmetric, then this operation yields the same kernel output. This use of this operator has been explored in Sec. 5.2.1, and its iteration parameter optimized for use in this proposed super resolution algorithm.

Chapter 4

Results

The standard set of parameters that were used for the simulations/results are shown in the preface. The comparison of results is often very difficult because of the different parameters, implementation details, and difficulty to create fair set test setups. Additionally, both PSNR and the newer M-SSIM [24] quality metrics will be presented (where applicable); the subject of what quality metric best represents visible errors according to the human visual system is entire branch of research by itself, and will not be discussed here. M-SSIM parameters used for comparisons are the default ones recommended by the authors. Running times of algorithms are also very dependent on a number of factors such as memory speed, CPU speed, number of threads etc, and moreover, many authors do not indicate running times. Despite all of the complications for fair comparisons, the author will make great effort to be fair and objective, comparing to different works on an individual basis.

To standardize the proposed method's results, a standard simulation setup was created, shown later in Fig. 5.4, which when loaded with the sequences in their order above, automatically output the reconstruction results to file including running times.

Table 4.1: Comparison of results to prior work with default parameters.

Sequence	Bicubic	Lengyel et al. [19]			Proposed SR method		
		PSNR (dB)	MSSIM	Time (s)	PSNR (dB)	MSSIM	Time (s)
Akiyo	35.152	43.559	0.990314	412.685	46.439	0.997794	28.579
Container	28.112	36.740	0.980581	433.542	41.165	0.993828	24.243
News	30.259	40.989	0.987936	473.588	43.566	0.994089	38.938
Hall Monit	29.545	38.325	0.989027	475.428	39.723	0.993433	84.412
Foreman	32.395	35.996	0.977319	477.941	37.782	0.979521	115.176
Stefan	27.233	29.788	0.948094	482.012	30.735	0.971763	134.004
Bus	26.787	27.630	0.835200	470.436	28.397	0.903309	167.936
Mobile	23.501	29.222	0.957512	463.526	27.921	0.956789	169.760

Comparing to the prior work by Lengyel et al. [19] in Table 4.1, the following sequences were selected for reconstruction, using default parameters, since they contain a wide variation of content, with the 'Akiyo' sequence being the easiest to reconstruct, and the 'Mobile' sequence being the hardest. For reference, the main parameters that were used for initial degradation were: $R = 5$ and $F = 2$. The comparison is for the reconstruction of an entire sequence of 31 frames (24 LR frames + 7 HR key-frames), where each sequence is the standard CIF format: (352×288). The results show significant improvement[1] over the previous work by Lengyel et al. [19], in both running time and overall performance. The running time was measured on a 2.8GHz computer, with 6GB of memory, utilizing up to 8 threads where applicable. Utilizing the hierarchy method for reconstruction, the number of pixels that require full NLM reconstruction is reduced significantly and is reflected in the running time of the proposed algorithm. Specifically, the sequences experience a speedup factor from 2.7 in the 'Mobile' sequence to 14.7 in the 'Akiyo' sequence, when compared to Lengyel et al. [19]. Refer to Sec. 4.2 for the visual results associated with Table 4.1.

[1]The reported PSNR values here are not average: they represent the PSNR values of the cumulative average error over an entire sequence.

Table 4.2: Reconstruction map diversity after each hierarchy stage. ($T_{\text{o}} = 0.125$)

Sequence	$\mathbf{M}_{\forall}^{(1)}$	$\mathbf{M}_{\forall}^{(2)}$	$\mathbf{M}_{\forall}^{(3)}$	$\mathbf{M}_{\forall}^{(4)}$
Akiyo	100%	2.60%	1.06 %	0.03%
Container	100%	2.31%	0.30 %	0.14%
News	100%	6.34%	6.75%	0.88%
Hall Monit	100%	7.17%	8.64%	1.80%
Foreman	100%	30.86%	35.83%	1.94%
Stefan	100%	62.27%	57.17%	29.29%
Bus	100%	84.91%	81.80%	44.65%
Mobile	100%	82.95%	62.55%	9.67%

Table 4.2 shows the reconstruction map and the distribution of pixels that are promoted for reconstruction using a heavier method. The first stage represents the beginning of the hierarchy and thus its initial value indicates that all pixels still require reconstruction. After the application of the linear estimation stage, background and stationary pixels are reconstructed and the number of pixels remaining in the reconstruction map is significantly reduced for static scenes like 'Akiyo'. The remaining pixels are processed by the optical flow based estimation method. At first glance it seems as if the optical flow based estimation causes more errors than it solves because the reconstruction output (post residue) indicates a greater number of pixels that need to be reconstructed when compared to the previous stage, in select sequences. However, this is expected because of the first appearance of the reconstruction map augmentation with the optical flow based estimation quality metric from Eq. 3.23. Increasing the associated threshold for the quality feedback will reduce the impact motion quality metric has on the reconstruction map. Finally, the last reconstruction map is produced after the NLM stage, and represents the pixels that have failed reconstruction and require fallback. Refer to Fig. 4.1 and Fig. 4.2 for a visual aid.

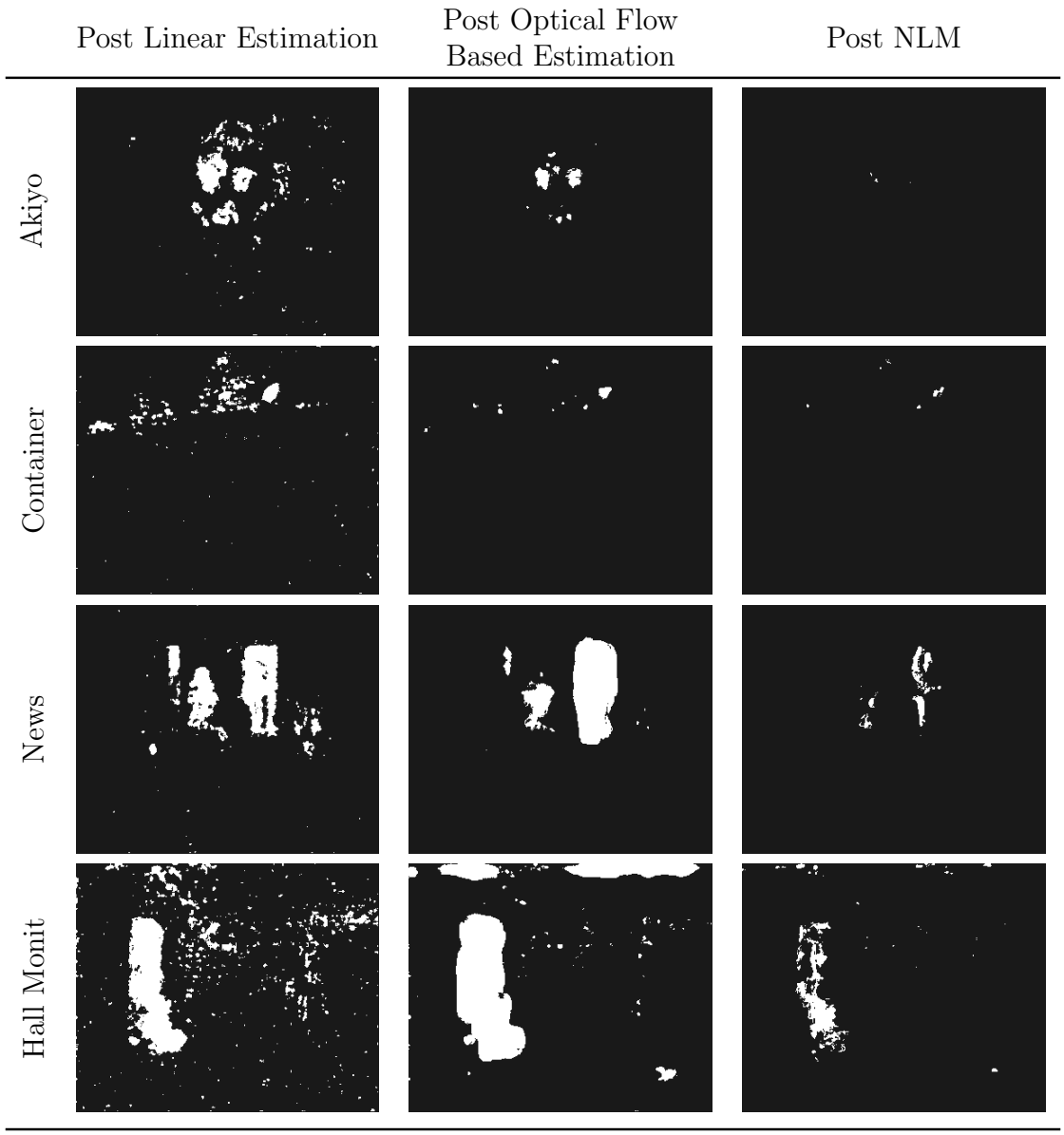


Figure 4.1: Reconstruction map diversity after each hierarchy stage. Part 1/2

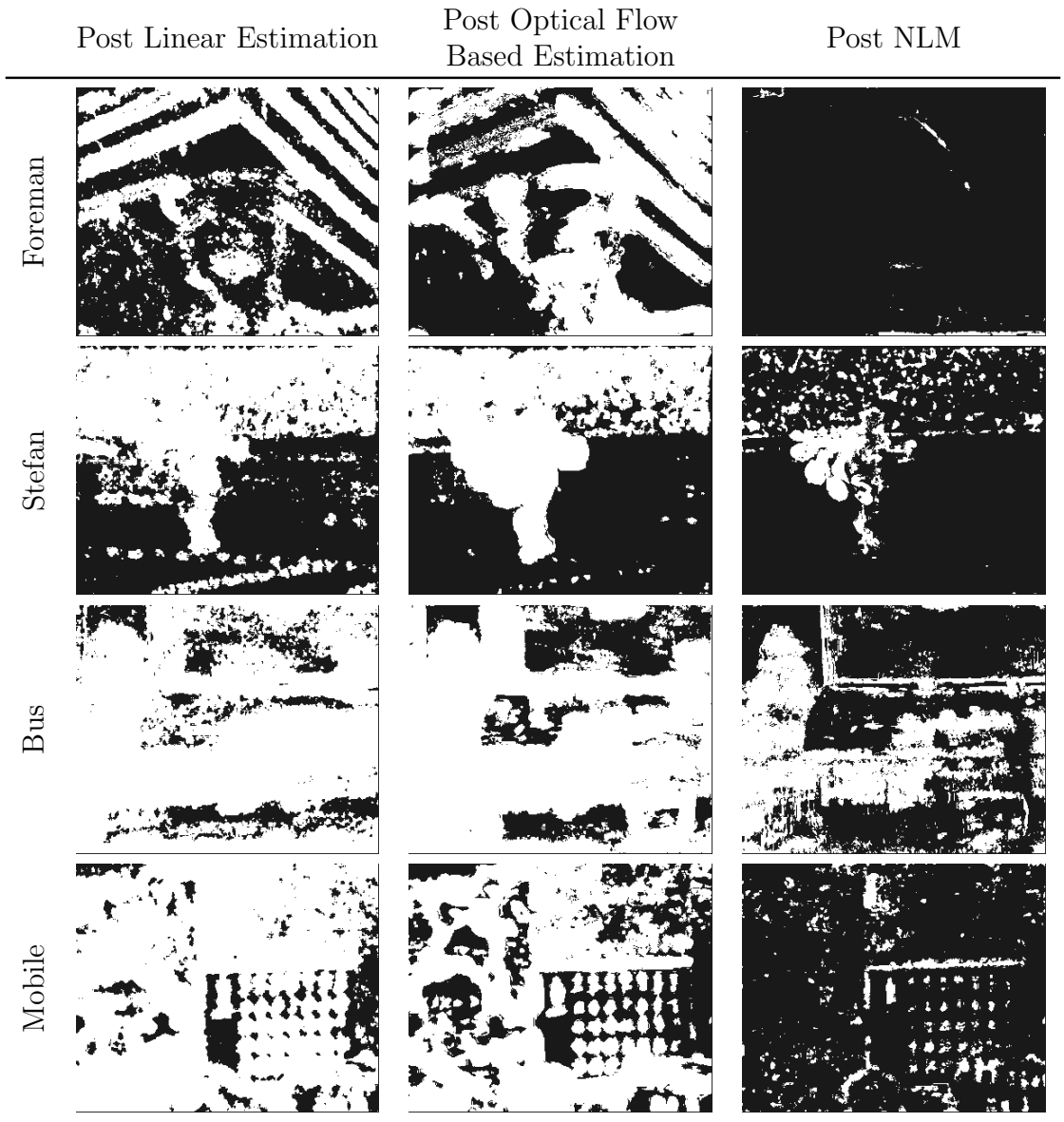


Figure 4.2: Reconstruction map diversity after each hierarchy stage. Part 2/2

Table 4.3: Comparison of results for multi-view sequences.

Sequence	Views	Lengyel et al. [19] $R = 6$ $S = 15$			Proposed SR method $R = 6$ $S = 7$		
		PSNR (dB)	M-SSIM	Time (s)	PSNR (dB)	M-SSIM	Time (s)
Ballroom_0	1	39.194	0.977	>1 hr	41.845	0.989	229.223
	2	39.573	0.978	$\gg 1$ hr			
	3	39.786	0.979	$\ggg 1$ hr			
	2*	39.865	0.979	$\gg 1$ hr			
	3*	39.941	0.980	$\ggg 1$ hr			
Crowd_1	1	37.587	0.983	>1 hr	38.743	0.987	371.324
	2	37.664	0.984	$\gg 1$ hr			
	3	37.691	0.984	$\ggg 1$ hr			
	2*	37.813	0.984	$\gg 1$ hr			
	3*	37.829	0.984	$\ggg 1$ hr			

Multi-view sequences can also be reconstructed by considering the expansion of NLM candidate sets into other HR key-frames in other views, as shown in Sec. 3.5.2.1. The work by Lengyel et al. [19] used a out-of-phase approach when dealing with HR key-frame generation for the hybrid cameras across different views. This approach balances HR key-frame generation temporally and in the view dimension. However, further work is required to update the current algorithm to consider multi-view sequences thus the results for the proposed method for additional views will be empty because the proposed method does not integrate multi-view sequences natively. A compromise can be reached if the proposed algorithm is applied to the single-view sequence only, but treated as if it performed multi-view super-resolution. This puts the proposed method at a technical disadvantage, since it will not be utilizing HR key-frames generated in additional views. Despite this handicap, the proposed method outperforms the prior work, in both performance and computational complexity as shown in Table 4.3. This is expected since the prior work in [19] does not reduce number of pixels reconstructed with NLM, and thus it is applied to every pixel. The image

Table 4.4: PSNR (dB) Comparison to other works with sparse HR frames ($R = 30$).

Sequence	Bicubic	[9]	[5]	[11]	[19]	Proposed (A)	Proposed (B)
News	30.2	30.4	36.1	37.2	37.2	37.015	37.182
Mobile	23.5	23.4	25.5	26.0	27.3	23.633	27.533
Container	28.1	28.8	33.2	34.2	34.6	34.154	34.335
Hall Monit	29.6	30.5	38.0	38.1	37.3	36.407	36.974

sizes are very large (640×480), and the comparison considers an entire sequence of 31 frames (26 LR frames + 5 HR key-frames). Sequences with views marked with a asterisk* represent the disparity compensation that was used in [19], and serves to assist the NLM to find candidate pixels that are very far way in different views. The authors also have results for $S = 7$, but they are lower than for the displayed results $R = 15$, and so they were omitted.

The comparison to other works can be found in Table 4.4. It is compared to bicubic, Brandi et al. [9], Song et al. [5], Najafi and Shirani [11], and Lengyel et al. [19] using the following parameters: $R = 30$, $\sigma_c = 1.0$, $S = 32$, $P = 9$, with the 15^{th} frame reconstructed only. The initial results (Proposed A) show a slight performance loss, which is attributed to the reconstruction quality metric for the NLM. Since there are more pixels considered, the accumulated error as per the quality metric formula, is a lot higher. Without adequate compensation, the accumulated error will forcefully trigger the fallback method and discard most of the NLM reconstructed pixels. The fix is simple, change the coefficient K_3 to be ∞, essentially eliminating the quality metric from producing a fallback condition. Moreover, the last threshold coefficient $T_f^{(3)}$ also needs to be set to ∞ to force the NLM output to be the final output. Additionally, it has been found the smaller patch sizes P with sharper kernels do a better job at patch comparisons in the NLM because they capture high frequency structure

more accurately. Reverting the patch size from 9 back to the default of 5 improves the results, as well as speeds up the algorithm. With these three modifications, the proposed method (B) outperforms the other works [9] [5] [11], and approximately matches the results from [19]. However, the running time of the algorithm is significantly reduced in each sequence tested, which is a hidden bonus that is not shown in the table. The running time is reduced by a factor of approximately 2.5 for the 'Mobile' sequence and up to 10 times for the 'News' sequence.

The modifications essentially simplifies the algorithm to be the same as in [19], except for the addition of motion vectors. This small change gives 'Mobile' sequence a slight performance boost since there are pixels that have HR counterparts that are not captured in a (32×32) search window around the original pixel location. The 'Container' sequence improves because it it has a different set of candidate pixels. The motion vector compensated candidate search areas cause a small subset of pixels, around the boat and ship, to not be falsely trimmed due to the second threshold T_2. The second threshold was defined to be a function of the difference between a $(S \times S)$ size area in the LR frame and its each HR key-frame. In [19], the position of this comparison in the LR and HR key-frames is the same, while in the proposed method it can vary based on the adjacent jump table based motion vectors. It is recommended that the enlargement of the candidate sets be avoided since the running time of the algorithm will increase exponentially. Reducing the candidate search area sizes S, and focusing on better motion estimation between LR and HR key-frames will lead to better results. The NLM is a very powerful reconstruction tool if supplied the right candidate pixel areas, otherwise, it will simply blur results as mentioned by Lengyel et al. [19]. Overloading the NLM with candidates also causes more blurring since the

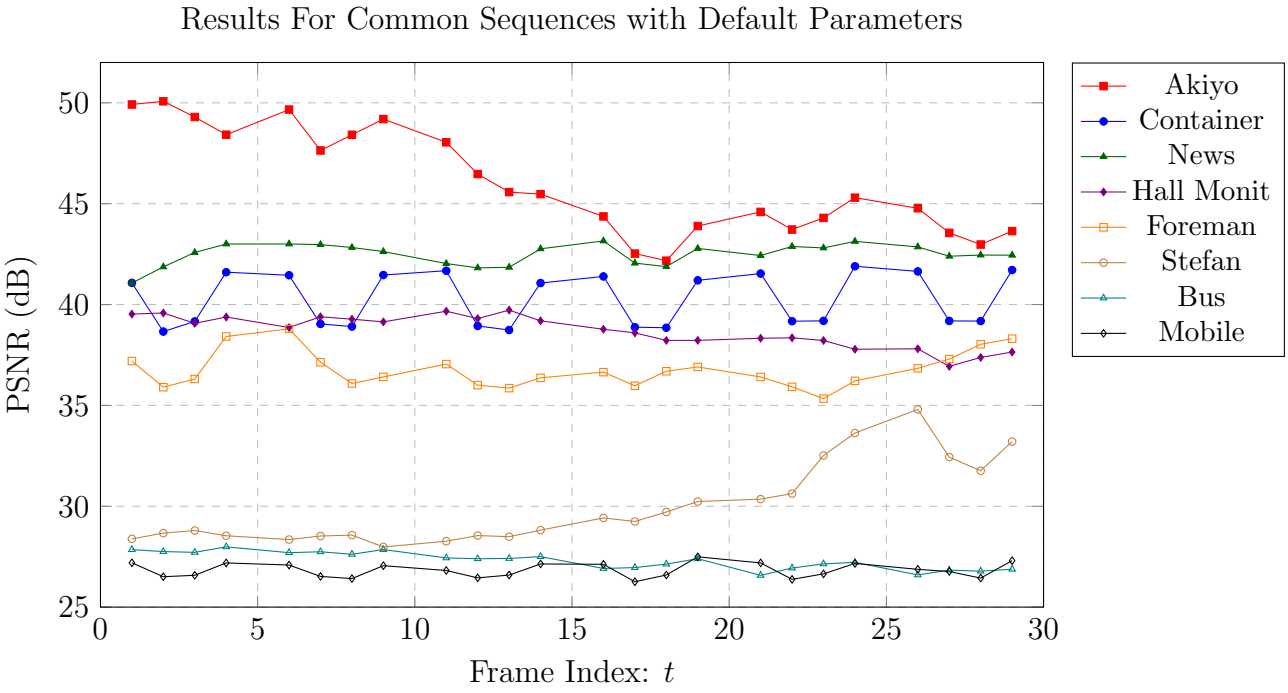


Figure 4.3: Frame-by-frame results for common sequences

NLM is itself a weighted average.

It is also important to note that the optical flow based estimation loses performance when the HR key-frames become sparse. Specifically, the forward optical flow based estimate cannot warp the HF information from a HR key-frame to LR frame indexes that are temporally far: the iterative warping produces blurry results after only a few frames. Similarly the backwards estimates are plagued by the same problem. The 15^{th} reconstructed frame requires 14 jumps to the nearest HR frame, and so the accumulated warping errors trigger the NLM reconstruction since the optical flow based estimation will surely fail.

Comparing the proposed method to the PSO-based fusion method by Cheng et al. [18] is difficult because the authors could not be contacted so that the original data could be communicated. However, they have shown results for both the 'Akiyo' and

'Foreman' sequences using a HR to LR ratio of 0.33 which corresponds to the HR Rate parameter of 3. Despite the lack of the original data to compare with, Fig. 4.3 can be used for visual comparison. The proposed method outperforms their work even when given a handicap; the proposed method for the 'Akiyo' and 'Foreman' sequences when $R = 5$ outperforms [18] when $R = 3$, which demonstrates its effectiveness. Additionally, their method outperforms the original NLM for super-resolution by Protter et al. [10], and so by extension, the proposed method will also outperform the original NLM. As explained in Sec. 3.5, the use of LR information in the NLM weighted sum does not lead to significant increases in performance because the information quality is degraded; the LR information is blurry and its inclusion in NLM is detrimental.

4.1 Analysis of Results

To demonstrate the effectiveness of the proposed method, the results for the 'Foreman', 'Container', and 'Mobile' sequences will be examined and analyzed.

In general, the results show that the hierarchy and residue model proposed (multiple stages of reconstruction with error feedback) build on the results of the last stage, and improves performance overall. It is important to note that the optical flow based estimation output contains pixels from the last stage: linear estimation. The same statement can be made for any pair of methods in the hierarchy model. The choice of which pixels are included in the next stage is the result of the reconstruction map $\mathbf{M}_t^{(\forall)}$, and in general it shows that the reconstruction map correctly chooses between the different layers in such a way as to maximize the results. For example, a pixel reconstructed with the linear estimation method, can be of better quality than if it was forced to be reconstructed using optical flow based estimation, or even

the heavy NLM, and it is the job of the reconstruction map to isolate these pixels through the residue feedback, so that performance can be maximized. However, as explicitly stated in the final paragraph of Sec. 3.2.2, the output of each stage must, on average, be higher than the last stage, since the proposed algorithm will choose the heavier reconstruction method when reconstruction fails at a lower stage. In other words, without knowing if a particular pixel reconstructed with the linear estimation method is better than if reconstructed with the optical flow based estimation method, a failed reconstruction at the linear estimation stage will force it to be reconstructed with the heavier optical flow based estimation, and the final output will now contain a pixel that is of lower quality than its former reconstruction. Overall, this scenario does not occur for the majority of pixels, and thus the performance per image frame is maximized.

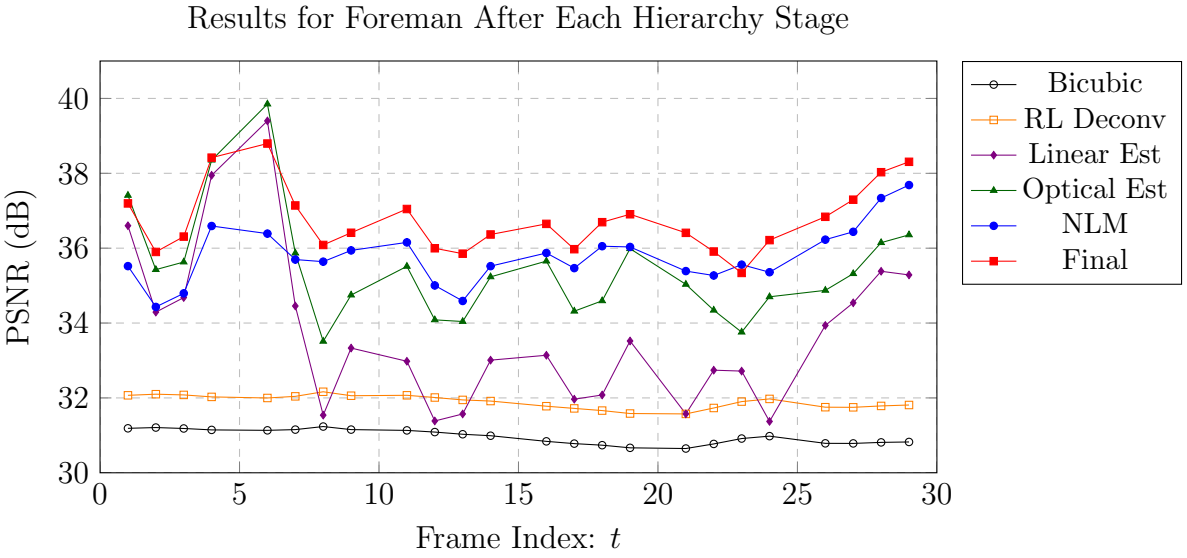


Figure 4.4: 'Foreman' results per stage in hierarchy model

In particular, frame number 6 of 'Foreman' shown in Fig. 4.4 exhibits the exception mentioned, since its final fused result is lower than that of the linear estimation

output, and even optical flow based estimation output; the hierarchy model will 'trust' the NLM output over the optical flow based estimation and subsequently the linear estimation output, on a per-pixel basis. The residue thresholds also support this effect, as the the tolerable error $T_{\mathrm{f}}^{(k)}$ increases in each hierarchy stage k. The errors that are detected at a one particular stage, are absorbed by the next hierarchy stage due to the increased thresholding. Frame number 6 also has the first occurrence of occlusion in the sequence (HR frames do not have this information entirely), and despite the residue feedback, quality measurements and fallback, it did not optimally compensate for this, causing a slight reduction in performance. This is an indication that there is still a small margin for improvement in the proposed method. For the rest of the sequence, the results are as expected, and the final output is better than the individual methods themselves.

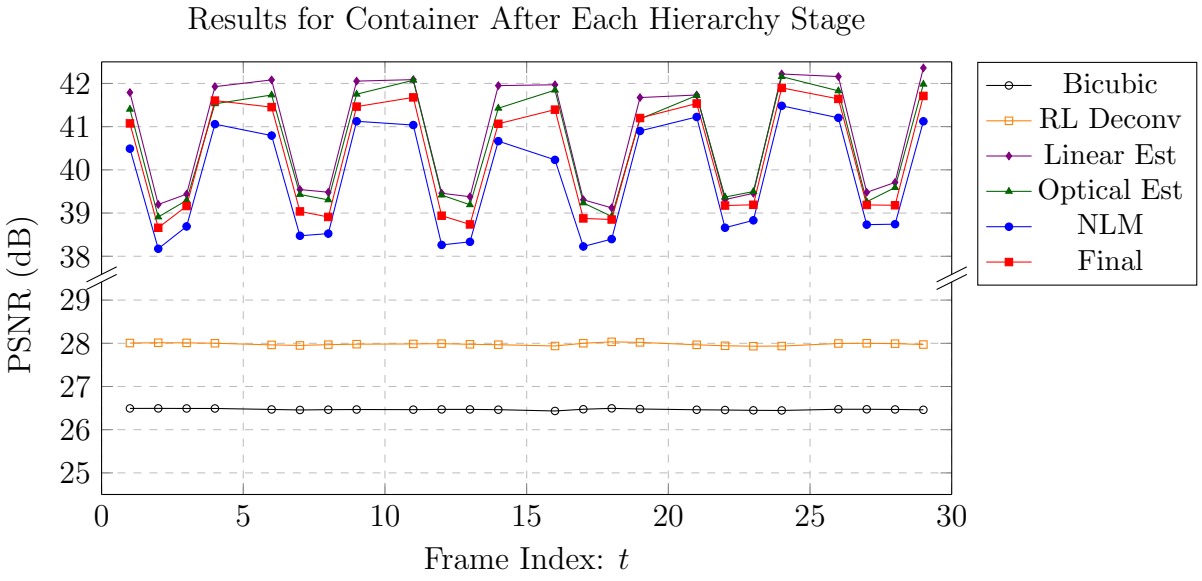


Figure 4.5: 'Container' results per stage in hierarchy model

The 'Container' sequence results shown in Fig. 4.5 are particularly interesting. In

particular there is a decade dB separation between the Richardson-Lucy deconvolved output, and the main hierarchy outputs. Additionally, the linear estimation output provides the best performance, which can be explained in the following way. For the pixels that failed to be reconstructed with the linear estimation method, the optical flow based estimation method and subsequently the NLM method fail to provide a better reconstructed output because the affected areas are full of pixels that are aliased in the original HR source frame as well as the key-frames. The aliased pixels cause the optical flow fields generated in the heavily downsampled frames to give inaccurate motion vectors. In other words, the aliased pixels drastically violates the brightness constancy assumption. The HF information warped in the optical flow based estimation stage using the inaccurate vectors contaminate the result, and often propagate the reconstruction to use NLM instead. However, the patch comparisons in the NLM are not designed to capture high frequency aliased information. The patch comparisons assume that the pixel region in the LR frame is very similar structurally to another region in each HR frame. However, these regions will fail to be reconstructed because their raw pixel values are varying heavily due to aliasing present in the sequence itself. In short, neither the optical flow based estimation or the NLM can properly reconstruct these aliased pixels, and the performance will suffer due to many pixels being reconstructed with the fallback technique. The linear estimation output is better because it is relatively unaffected by aliasing: the high frequency content is linearly interpolated across the LR frames from a HR key-frame to the next. Since the linear estimation is a low pass filter, it dampens the HF aliased information temporally across the LR frames. Moreover, since the motion vectors in the sequence are very small in magnitude, the filtering is relevant temporally.

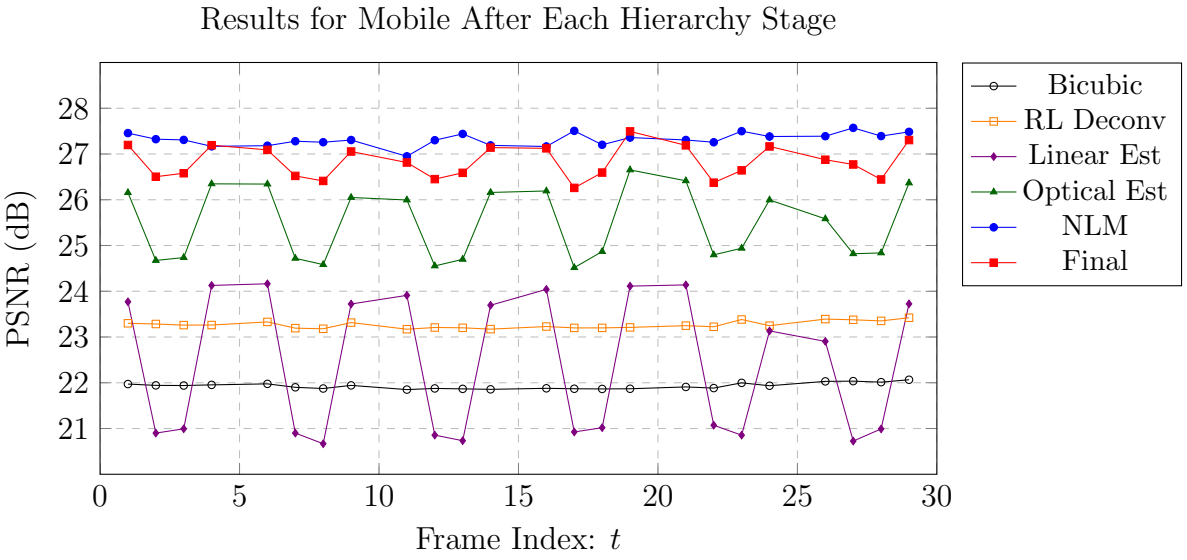


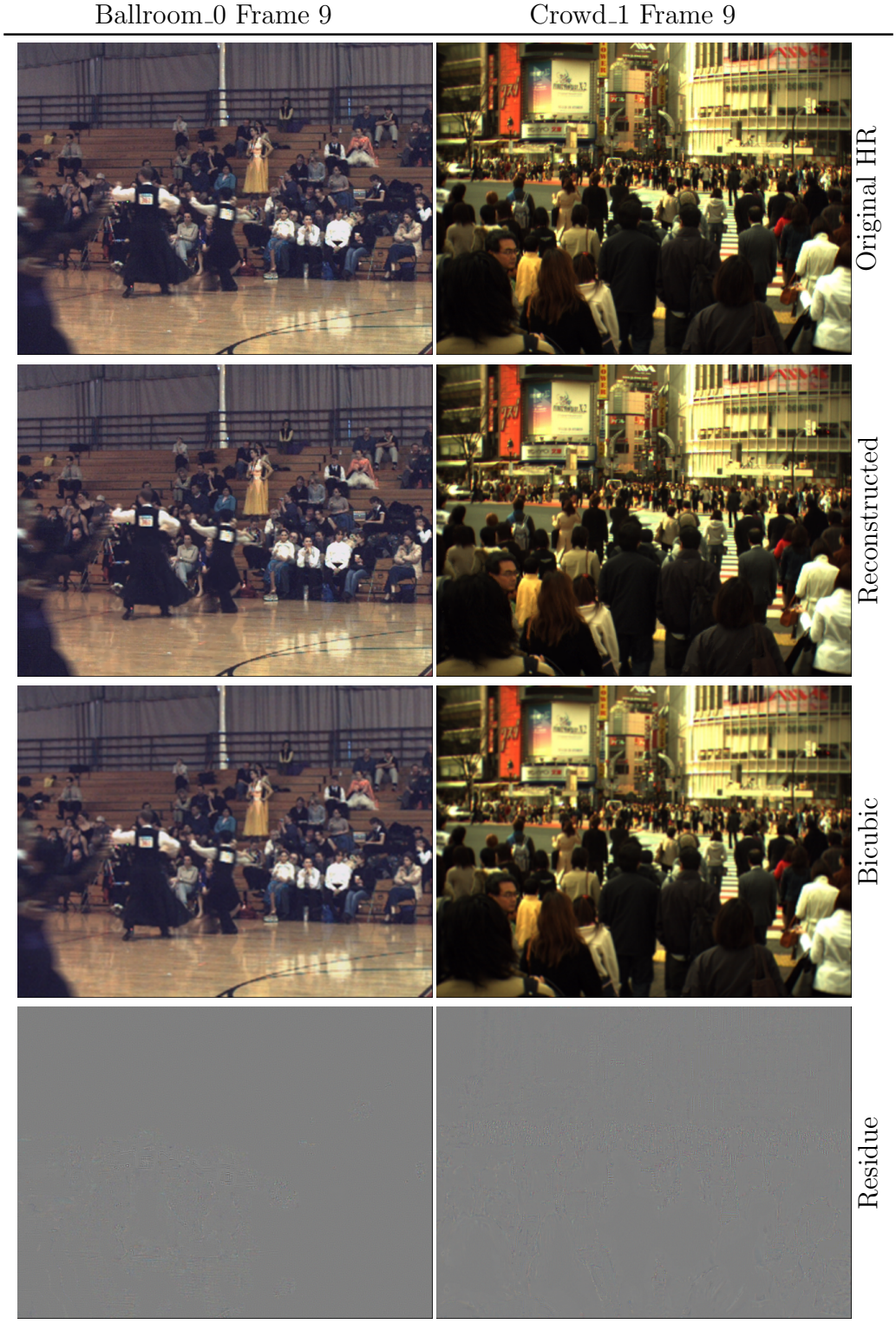
Figure 4.6: 'Mobile' results per stage in hierarchy model

The final sequence analyzed is the 'Mobile' shown in Fig. 4.6 which is the hardest to reconstruct. It features global motion, heavy aliasing, complex motion, occlusion, high frequency details, and a gradual change in viewing area. It is clear that the linear estimation fails constantly due to the heavy aliasing, and motion vectors present in the scene. Optical flow based estimation succeeds in recovering some HF information, but its effectiveness is limited. Over a few frames, HF information in the forward estimate is blurred out, and triggers the reconstruction to use the NLM for the entire column of pixels formed temporally between adjacent HR key-frames. The NLM gives excellent results at this stage, since most of the missing high frequency information is readily available in the HR key-frames. However, the NLM is not the final stage, and so the final residue and thresholding via $T_{\mathrm{f}}^{(3)}$ tends to discard some of the properly reconstructed NLM pixels in favor of Richardson-Lucy fallback pixels. Therefore, the final reconstructed output is slightly less than the NLM output. One way to improve the results is to raise the threshold $T_{\mathrm{f}}^{(3)}$ for this particular sequence. However, the

proposed algorithm should have common parameters for all sequences, and thus it may compromise the results of other sequences. Moreover, raising the threshold for this particular sequence uses prior knowledge that the NLM reconstruction is a superior reconstruction! In general, no changes to the common parameters should be made, with the exception of the known input parameters such as R or F. However, It is perfectly reasonable to have different common algorithm parameters as a function of the known input parameters such as R and F. The other way to improve the results for 'Mobile' is to deal with the high frequency aliasing problem, seen in many other sequences. However, further research is required to handling aliasing.

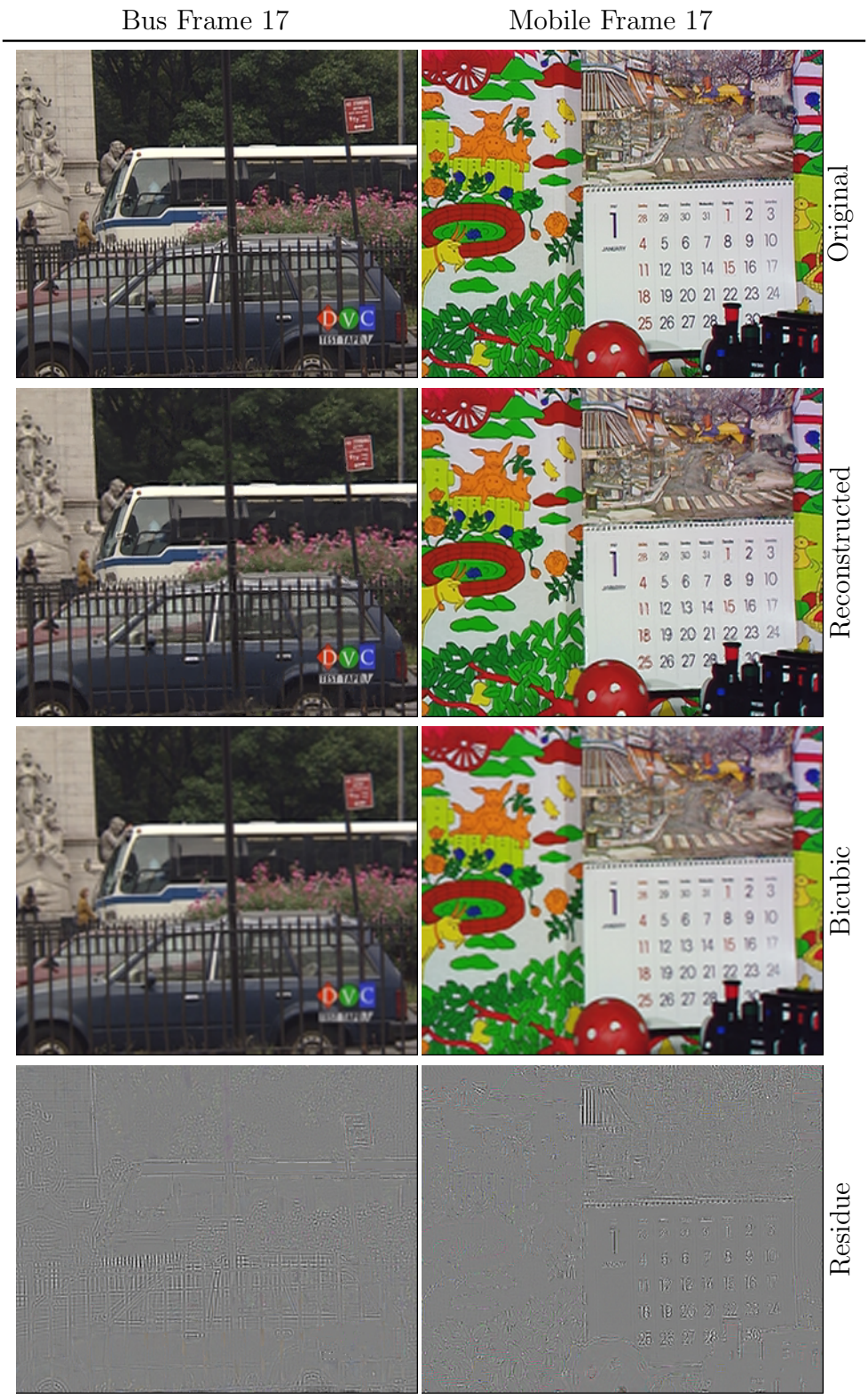
4.2 Gallery of Super-Resolved Sequences

This section will be dedicated to visual results. The multi-view 'Ballroom_0' and 'Crowd_1' sequences were super-resolved using $R = 6$, and the remaining CIF sized image sequences were super-resolved using the default parameters for the proposed algorithm. The frame number was selected as to be temporally equidistant to adjacent HR key-frames. The reconstructed results are also compared to the bicubic to show improvement over simple interpolation. Additionally, an error residue between the original HR frame and the super-resolved image is shown to accentuate differences.


Ballroom_0 Frame 9
Crowd_1 Frame 9
Original HR
Reconstructed
Bicubic
Residue


Akiyo Frame 17
News Frame 17
Original
Reconstructed
Bicubic
Residue
MPEG4
WORLD


Foreman Frame 17
Stefan Frame 17
Original
Reconstructed
Bicubic
Residue


Bus Frame 17
Mobile Frame 17
Original
Reconstructed
Bicubic
Residue

Chapter 5

Simulations with EngineX

The objective of this work is to present a practical method for key-frame video super-resolution. However, instead of developing the algorithm as a standalone application (for processing single inputs), a '*sandbox*' virtual space for advanced image processing was built. Utilizing DirectX and Win32, the simulator functions as a black-box I/O virtual space containing many image processing techniques, combined with a user interface designed to modularly assemble/experiment with those methods.

Within the simulator, **EngineX**, the proposed super-resolution algorithm was built, simulated, optimized, and debugged. Furthermore, all secondary methods including bicubic interpolation, convolution filtering, channel transformations, optical flow/warp and motion vectors were all built manually; no additional libraries were used that assist with common image processing methods, were used. This serves the end goal of learning the threading and implementation details of all methods.

5.1 Capabilities

The simulator features a large set of capabilities to assist with test various scenarios without recompilation. It features a virtual space for manipulating objects, dynamic routing for creating simulation configurations, video processing methods for transformations and experimentation, parameter sweeps for optimization, and multithreading for speed. Hardware acceleration for video display is provided through DirectX. The implementation involves creating separate threads for logic, drawing, timing, and input. By separating the various parts of the engine, each element will continue to be responsive when processing time for any of them increases; latency between core elements is eliminated through the use of multiple threads each handling one particular aspect of the engine. The core of the engine is custom built, and includes rendering functions for very fast blitting operations including alpha blending.

5.1.1 Virtual Space

The simulator displays a virtual 2D grid environment, where objects (video sequences, and image processing methods) can be freely manipulated. Combined with an extensive zoom and pan capability, the grid space allows for many objects of different sizes to be placed, moved, and scaled at will. Additionally, the simulator handles multiple virtual spaces simultaneously, with the ability to switch between them easily. Virtual spaces can also be nested and connected, with a child virtual space being defined by a parent one. Passing information between virtual spaces is done through the use of instances of each space. Each virtual space can be cleared, saved, or loaded which functions to save important simulation configurations. This powerful abstraction allows different image processing scenarios to be quickly investigated.

5.1.2 Dynamic Routing

Dynamic routing allows the user to 'hook-up' various objects in the simulator in a similar fashion to electrical circuit building. Each element contains three types of I/O: input, parameter, and output. The input functions as the minimum required input for processing. The parameter is an extra input that allows the devices to change what they do dynamically. The block must be able to function without a value in the parameter, or a default assumed value. The output sends the computed result out along the wire to the next block. Fig. 5.1 shows sample blocks that illustrate parameter overriding and defaulting for adding noise to an input.

The input appears on the left of the block, while the parameter appears on the bottom and the output appears on the right.

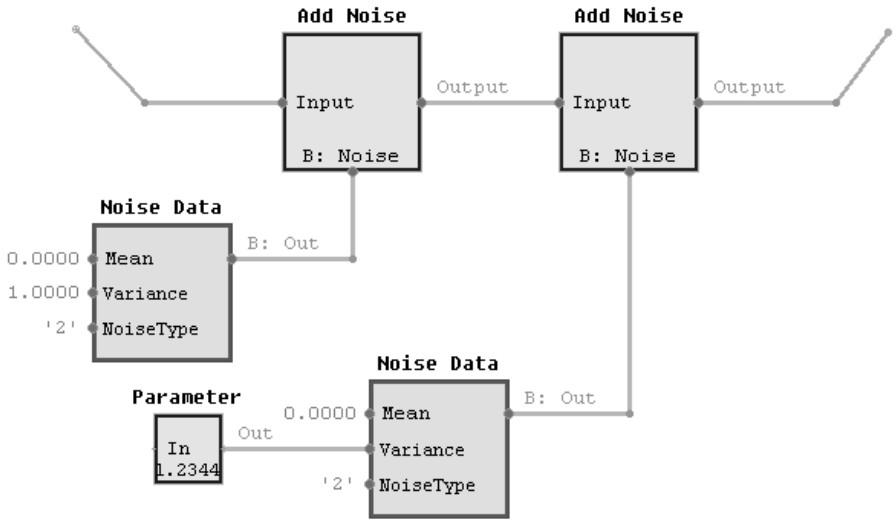


Figure 5.1: Sample simulation in EngineX for adding noise to an input.

5.1.3 Video & Image Processing

EngineX loads images and videos through a simple interface menu. Each video now becomes an individual block, which may be operated on in the tool. The following table shows the complete list of the video and image processing methods implemented within the simulator. The function of each operation is self explanatory, and will not be explained individually.

Table 5.1: Complete list of video and image processing functionality within EngineX.

Operation	Operation	Operation
Load Video	RGB Decomp	RGB to YUV
Save Video	Bit-planes	YUV to RGB
Copy Video	Comp MSE	Do FFT
Delete Video	Comp MSSIM	Do IFFT
Crop Video	Filter	Do Haar-DWT
Extract Frame	Deblur (RL)	Do Haar-IDWT
Splice Sequences	Resample	Add
Add Text	Resize	Subtract
Clear Text	Add Noise	Cross Blend
Save Frame	Gradient	Optical Flow
Load Frame	Greyscale	Optical Warp

The visual interface combined with the powerful image processing methods allows the user to experiment with methods quickly without writing any code, as per traditional research. Common methods have been implemented to account for the vast majority of simple image processing methods, such as bicubic resampling, and color space changes. Additional options are presented to the user to allow customization of basic parameters like filter strength and window size. The image processing functions can be applied immediately to a video block, or abstractly as a generic input/output block which is linked up by the user.

5.1.4 Parameter Sweeping

The simulator allows for parameters to be dynamically changed during a simulation. For each value in the sweep, the entire simulation is rerun. If two (or more) parameter sweeps are utilized, they are both varied independently.

Parameter sweeping is shown visually in Fig. 5.2 which sets up a simulation to compare two videos, one with noise and the other without. The mean of the noise applied is varied from 5.0 to 9.0 in increments of 0.1. The MSE as a function of the noise meen is output to a file 'Output.csv' which can be used for further data processing in other tools. From the analysis of the data, it is possible to select the value for the parameter that gives the highest value of PSNR.

This "trial-and-error" method yields good results if the simulation is properly setup. Typically, many of the parameters were optimized using a similar technique. However, multidimensional optimization is an open problem that does not have a analytic solution for non-linear systems. To further the problem, the algorithm is dependent on the input and cannot (in general) be shown to be optimal for all parameters. Furthermore, optimization using this method is very time consuming, especially when the simulation tries to optimize three (3) or more parameters simultaneously. Therefore, speeding up the entire engine's simulations was a high priority, and was accomplished through multithreading.

5.1.5 Multithreading and Beyond

Multithreading occurs naturally throughout the simulator between independent blocks. The parallel processing helps run multiple simulations simultaneously, as well as helps

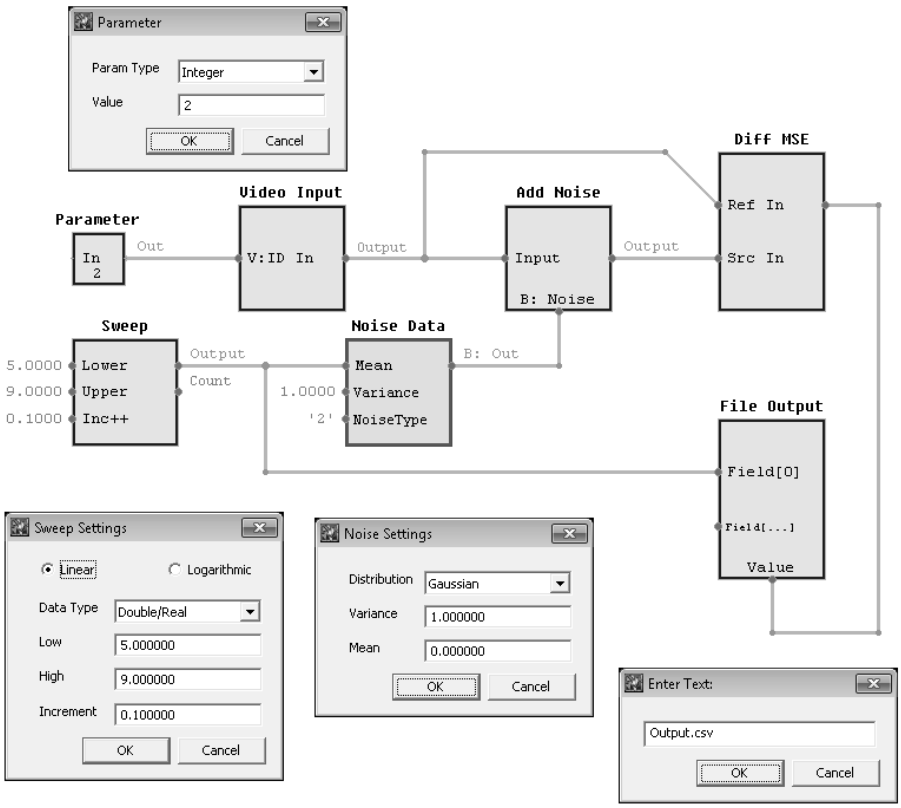


Figure 5.2: A common simulation setup with parameter sweeping and file output.

run individual blocks faster. Support for block independent parallel processing is automatic and native, while multithreading inside each block must be programmed manually. Using only a few core functions that have multithreading, such as convolution, and re-sampling, various other image processing algorithms can be built. For example, the Richardson-Lucy deconvolution makes heavy use of convolution, and by accelerating the convolution filtering stage via threads, as a general procedure, the Richardson-Lucy deconvolution will be sped up. The same logic applies to the other image processing algorithms that utilize standard functions in image processing, which have been accelerated via threading in EngineX. Future work on the simulator

will tie into the GPU for the image processing methods. The integration should be invisible to the user, and serve to improve the simulation speed. Beyond multithreaded CPU and GPU lies distributed / grid computing. The simulator is designed to eventually interface to other instances of itself, on other computers and across the web. By integrating basic socket communications and a simulation distribution algorithm, a potential simulation can utilize the CPU and GPU of multiple computers across the Internet, in a seamless way. Distributed computing has already played a big role in running complicated simulations in chemistry, biology, and physics. It is a long term goal for the EngineX project.

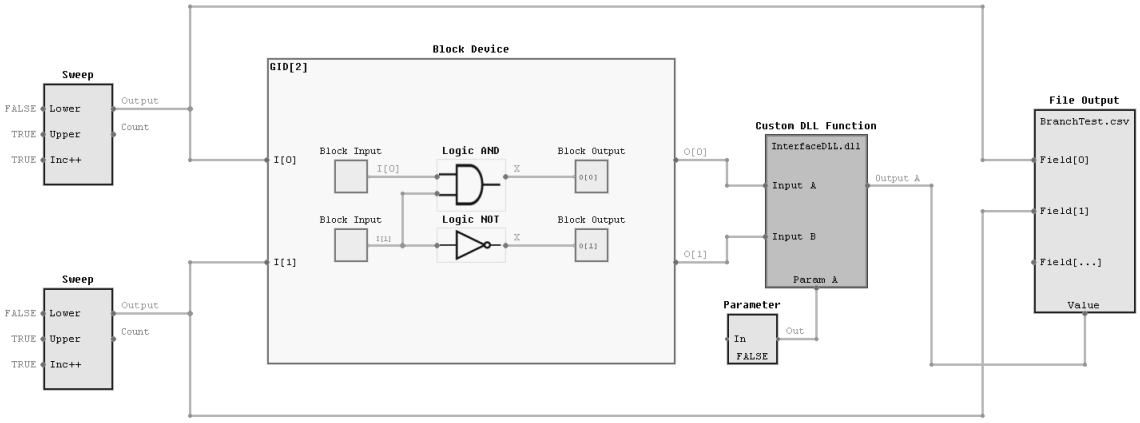


Figure 5.3: Advanced block abstraction through instancing and DLL links.

5.1.6 Advanced Abstraction

The power to build a simulation, execute it, and view its results is the core of EngineX. However, as algorithms can get complicated, it is often useful to represent algorithms using simpler blocks. This abstraction is also possible in EngineX by defining a block to be an instance of another virtual grid space. For instance, if the first virtual grid

space contains a specialized filter which is built using some arbitrary components, a second separate virtual grid space can create an instance of the first virtual grid space, and capture its operation inside of a block. This abstraction can be nested indefinitely, and is shown visually in Fig. 5.3.

Additionally, the simulator has the ability to load in special DLLs which contain input/output blocks. The imported DLL turns into a virtual block with its inputs and outputs defined by the code that built the DLL. When executed, the simulator will call the DLL and invoke its main method. Interprocess communications using TCP and UDP sockets is a planned update to the simulator, in addition to communications with other running processes.

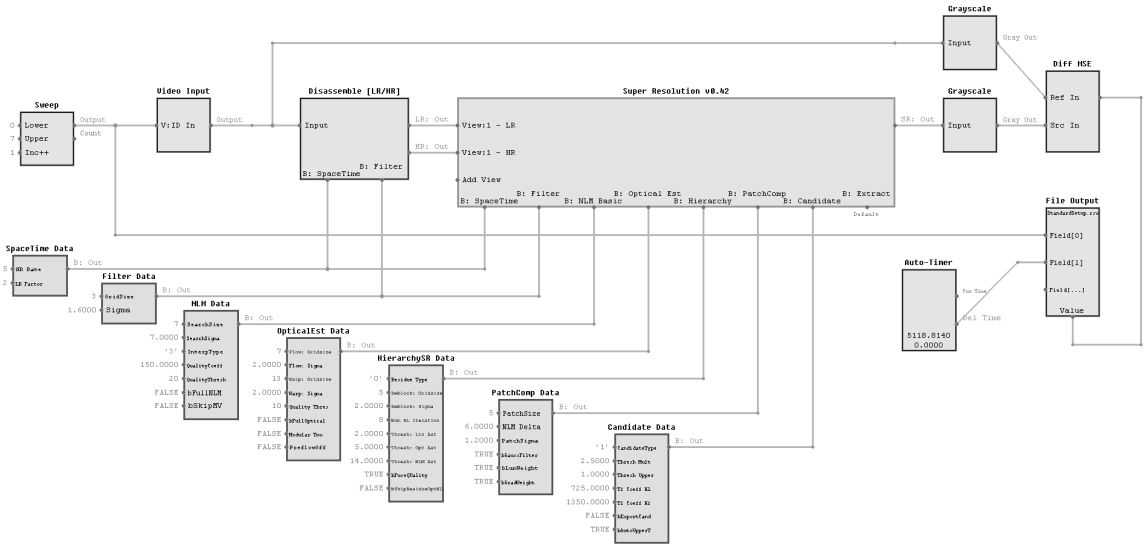


Figure 5.4: The standard setup in EngineX for the proposed algorithm showing bundle data for parameters, including their defaults.

5.2 Standard Setup

In the EngineX simulator, a standard setup was created that takes an input video sequence, performs temporal and spatial degradation on it to get a LR and HR stream. The proposed super-resolution algorithm is applied and its output is compared to the input to return a PSNR output. The basic setup contains a single parameter sweep to iterate through the chosen test image sequences. Common parameters passed to the proposed SR algorithm are grouped into bundles, each bundle contributing a set of parameters. The bundles of parameters get their values from preset defaults which were optimized as best as possible for performance. The entire setup can be seen in Fig. 5.4.

5.2.1 Optimal Richardson-Lucy Iterations

During the simulation of the proposed SR algorithm, several parameters were used that were optimized using standalone simulations with EngineX. Among the countless simulations performed, the optimization of the Richardson-Lucy iterations K_{r} produced an interesting result.

Referring to Fig. 5.5, two different simulations were performed on the 'News' sequence. Since Gaussian filtering, resampling, and RL deconvolution are all spatial domain functions, only a single frame is required; the 15^{th} frame was selected for simulation. In the first simulation, the raw Gaussian filtered input is deconvolved using the RL deconvolution method, while in the second simulation, downsampling occurs as to simulate the action of the hybrid camera in generating LR frames. Upsampling is also used to simulate the initial step of the SR algorithm.

From the results shown in Fig. 5.6, it is clear that the upsampling followed by

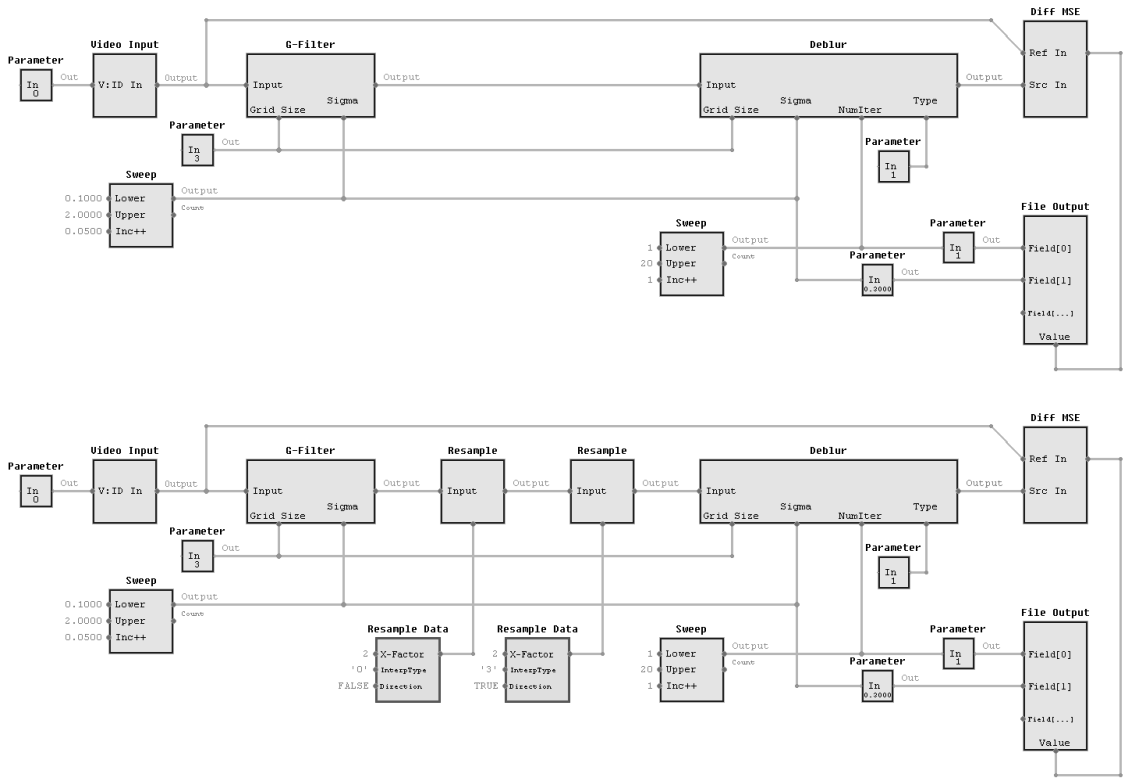


Figure 5.5: Simulation for the optimal number of Richardson-Lucy iterations. Top: with resampling. Bottom: without resampling.

downsampling causes the Richardson-Lucy deconvolution method to lose effectiveness after only a handful of iterations. Normally, it is possible to apply hundreds of iterations of the Richardson-Lucy deconvolution algorithm, without degradation in the output. However once resampling is applied, the performance of the Richardson-Lucy deconvolution method is compromised. The 'sweet-spot' can be directly extracted from the data, and is reflected in the default RL iteration parameter for the algorithm: $K_r = 8$, given the filtering strength σ_c chosen. The frequency based explanation is that the downsampling operator receives low-pass filtered data from the output of the Gaussian filter block. When the standard deviation approaches 0,

the downsampling operator decimates the unfiltered input, and the result is a high frequency image that is full of aliased components. To overcome this, the filtering strength should be increased. However, when the filtering strength (including window size) is increased too much, the output will become completely blurred, and its data essentially unrecoverable. Thus, adequate filtering before decimation is important for the super-resolution process, and the simulation results support this assertion. The two intersecting lines in the figure correspond to the chosen optimal RL iterations, and filtering strength under a Gaussian window of 3 to be used in the hybrid camera.

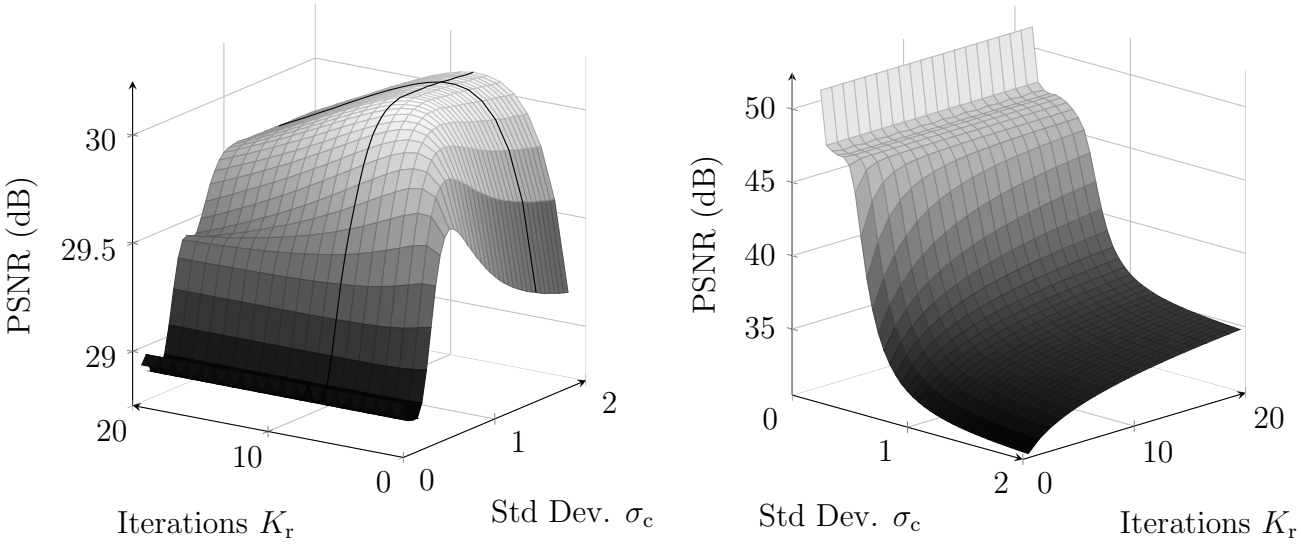


Figure 5.6: Results for the simulation to determine optimal RL iterations. Left: with resampling. Right: without resampling.

Chapter 6

Conclusion and Future Work

The problem of key-frame based video super-resolution remains a challenge to this day. The proposed algorithm provides a hierarchy framework for super-resolution by combining several different methods such as optical flow based estimation, and NLM. Some of the major challenges faced include occlusion, high frequency aliasing, selecting between different methods using masks, determining optimal parameters, and utilization of different feedback methods. Motion vectors via optical flow continues to be a major part of all SR algorithms, and its utilization in the proposed method has been a crucial component of its success. Moreover, to develop and fine tune the algorithm, an advanced image processing tool was created to push the boundaries of current research. By integrating advanced simulation capabilities with abstract block and image processing methods, the EngineX simulator enables the design and simulation of high level algorithms with ease.

Overall the proposed hierarchy method provides a sturdy foundation for key-frame based super-resolution for hybrid cameras. Future work is aimed at further reducing

the computation time, and identifying the limitations of the proposed method. Integrating additional sources of information such as multi-view sequences and depth map priors represents other research avenues. Alternatively, expanding the simulators capability for performing image and video processing is also important. Parallel processing either by multithreading, GPU, or distributed computing is targeted for further development.

Bibliography

[1] X. Li and M. T. Orchard. New edge-directed interpolation. *IEEE Transactions on Image Processing*, 10:1521 – 1527, Oct 2001.

[2] J. Allebach and P. W. Wong. Edge-directed interpolation. In *International Conference on Image Processing*, volume 3, pages 707 – 710, 1996.

[3] Q. Wang and R. K. Ward. A new orientation-adaptive interpolation method. *IEEE Transactions on Image Processing*, 16:889 – 900, Apr 2007.

[4] M. Zhao, M. Bosma, and G. de Haan. Making the best of legacy video on modern displays. *Journal of the Society for Information Display*, 15(1):49 – 60, 2007.

[5] B. C. Song, S. Jeong, and Y. Choi. Video super-resolution algorithm using bi-directional overlapped block motion compensation and on-the-fly dictionary training. *IEEE Transactions on Circuits and Systems for Video Technology*, 21: 274 – 285, March 2011.

[6] W. Freeman, Jones T. R., and E. C. Pasztor. Example-based super-resolution. *IEEE Computer Graphics and Applications*, 22:56 – 65, Mar/Apr 2002.

[7] M. Ben-Ezra and S. K. Nayer. Motion deblurring using hybrid imaging. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–657 – I–664 vol.1, Jun 2003.

[8] F. Brandi, R. D. Queiroz, and D. Mukherjee. Super-resolution of video using key frames. In *IEEE International Symposium on Circuits and Systems*, pages 1608 – 1611, 2008.

[9] F. Brandi, R. D. Queiroz, and D. Mukherjee. Super-resolution of video using key frames and motion estimation. In *IEEE International Conference on Image Processing*, pages 321 – 324, 2008.

[10] M. Protter, M. Elad, H. Takeda, and P. Milanfar. Generalizing the nonlocal-means to super-resolution reconstruction. *IEEE Transactions on Image Processing*, 18(1):36 – 51, Jan 2009.

[11] S. Najafi and S. Shirani. Regularization function for video super-resolution using auxiliary high resolution still images. In *Asilomar Conference on Signals, Systems and Computers (ASILOMAR)*, pages 1713 – 1717, Nov 2012.

[12] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D*, 60:259 – 268, 1992.

[13] S. Zhu and K. Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Transactions on Image Processing*, 9(2):287 – 290, Feb 2000.

[14] G. Zhai and X. Wu. Video super-resolution for dual-mode digital cameras via scene-matched learning. In *International Workshop on Multimedia Signal Processing*, pages 438 – 442, Oct 2010.

[15] M. Glaistter, C. Chan, M. Frankovich, A. Tang, and A. Wong. Hybrid video compression using selective keyframe identification and patch-based super-resolution. In *IEEE International Symposium on Multimedia*, 2011.

[16] M. T. Orchard and G. J. Sullivan. Overlapped block motion compensation: An estimation-theoretic approach. *IEEE Transactions on Image Processing*, 3(5): 693 – 699, Sept 1994.

[17] T. Y. Kuo and C. C. J. Kuo. Fast overlapped block motion compensation with checkerboard block partitioning. *IEEE Transactions on Image Processing*, 8(6): 705 – 712, Oct 1998.

[18] M. Cheng, K. Hwang, and J. Jeng. Pso-based fusion method for video super-resolution. *Journal of Signal Processing Systems*, 73:25 – 42, Jan 2013.

[19] R. Lengyel, S. M. Reza Soroushmehr, and S. Shirani. Multi-view video super-resolution for hybrid cameras using modified NLM and adaptive thresholding. In *IEEE International Conference on Image Processing*, pages 5437 – 5441, Oct 2014.

[20] W. D. Richardson. Bayesian-based iterative method of image restoration. *Journal of the Optical Society of America*, 62(1):55 – 59, Jan 1972.

[21] L. B. Lucy. An iterative technique for the rectification of observed distributions. *The Astronomical Journal*, 79(6):745 – 754, Jun 1974.

[22] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Seventh International Joint Conference on Artificial Intelligence*, pages 647 – 679, Aug 1981.

[23] D. J. Fleet and Y. Weiss. Optical flow estimation. In N. Paragios, Y. Chen, and O. Faugeras, editors, *Mathematical Models in Computer Vision: The Handbook*, chapter 15, pages 239 – 258. Springer, 2005.

[24] A. C. Wang, Z. amd Bovik and E. P. Sheikh, H. R. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600 – 612, Apr 2004.