SMOOTHING AND REFINEMENT OF MOTION VECTOR FIELDS FOR VIDEO FRAME RATE CONVERSION APPLICATIONS

by

Ümit MALKOÇ

B.S., Electrical and Electronic Engineering, Boğaziçi University, 2004

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Electrical and Electronic Engineering
Boğaziçi University

2007

ACKNOWLEDGEMENTS

I would like to express my sincere thankfulness to my supervisor Prof. Bülent Sankur for his assistance and endless patience during the preparation of this M.S. Thesis.

I would also like to thank to Prof. Emin Anarım and Prof. Lale Akarun for participating in my thesis jury and for their valuable comments on the thesis.

Lastly, I am grateful to Ahmet Baştuğ for his continued support all the way during the preparation of my M.S thesis.

ABSTRACT

SMOOTHING AND REFINEMENT OF MOTION VECTOR FIELDS FOR VIDEO FRAME RATE CONVERSION APPLICATIONS

The main goal of Video Enhancement (VE) applications is to improve the visual quality of the video by modifying the input frame sequence. The enhancement operations could be performed in the spatial domain (each frame of the sequence is modified by only using the information that is obtained from the same frame) and/or in the temporal domain (each frame of the sequence is processed by using the information that is obtained frame itself and its neighbors) depending on the type of the desired application. In order to achieve a higher visual quality level without producing any artifacts, it is very critical to correctly extract some key features of the frame sequence and carefully analyze them. Especially in the case of temporal domain processing, the motion information in the video content is possibly the most important feature. However, extracting the motion information of the 3D real world from the 2D frame sequence is an ill-posed problem. Unfortunately it is not possible to estimate true motion by using classical error minimization techniques. Therefore, it is inevitable to perform smoothing and refinement operations on the Motion Vector Fields (MVF) obtained via classical ME algorithms, if one wants to obtain an acceptable level of quality from the enhancement application.

In this M.S. Thesis, a number of practical state-of-the-art ME methods are considered and some MVF post-processing (PP) operations (smoothing and refinement) are jointly investigated in detail. The effect of each algorithm on the quality of a Frame Rate Conversion (FRC) application is observed. Moreover, two contributions have been made in the area of MVF post-processing; the first contribution focuses on improving the performance of an existing strategy and the second contribution targets to decrease the computational load of another MVF post-processing strategy. It has been observed that the performance of the implemented FRC application is quite promising and fulfills the expectations of most of the today's consumer electronics applications.

ÖZET

HAREKET VEKTÖR ALANLARININ VİDEO ÇERÇEVE HIZI DEĞİŞİM UYGULAMARINA YÖNELİK OLARAK YUMUŞATILMASI VE İNCELTİLMESİ

Bir Video İyileştirme (VE) uygulamasının ana amacı gelen çerçeve zincirini görüntü kalitesini arttıracak şekilde değiştirmektir. Amaçlanan uygulamanın çeşidine bağlı olarak iyileştirme işlemleri uzamsal düzlemde (dizinin her çerçevesi sadece yine aynı çerçeveden elde edilen bilgiler kullanılarak değiştirilir) ve/veya zamansal düzlemde (dizinin her çerçevesi çerçevenin kendisi ve komşularından elde edilen bilgiler kullanılarak değiştirilir) gerçekleştirilebilir. Daha yüksek bir görsel kalite seviyesine herhangi bir sorun çıkarmadan ulaşmak için çerçeve zincirinin bazı anahtar özelliklerini doğru bir şekilde çıkarmak ve onları dikkatlice analiz etmek oldukça kritiktir. Özellikle zamansal alandaki uygulamalar için ele alınan içeriğin hareket bilgisi belki de en önemli özelliktir. Ancak 3-Boyutlu gerçek dünyanın hareket bilgisini 2-Boyutlu çerçeve zincirinden elde etmek iyi tanımlanmamış bir problemdir. Maalesef klasik hata minimizasyonuna dayalı Hareket Kestirimi (ME) yöntemleri ile gerçek hareket bilgisinin elde edilmesi mümkün değildir. Şu halde, VE uygulamasından kabul edilebilir bir düzeyde kalite elde etmek istenildiği durumda klasik ME algoritmasının Hareket Vektör Alanı (MVF) üzerinde yumuşatma ve inceltme işlemlerinin uygulanması kaçınılmazdır.

Bu tezde en güncel ME yöntemleri ele alınmış ve bazı MVF işleme (PP) operasyonlarıyla (yumuşatma ve inceltme) beraber detaylıca incelenmiştir. Her algoritmanın bir Çerçeve Hızı Dönüşüm (FRC) uygulamasının performansı üzerindeki etkileri gözlemlenmiştir. Ayrıca, MVF işleme alanında iki katkıda bulunulmuştur; ilk katkı varolan bir stratejinin performansının iyileştirilmesine odaklanmış ve ikinci katkı bir başka MVF işleminin işlem yükünü azaltmayı hedeflemiştir. Gerçeklenmiş olan FRC uygulamasının performansının umut vadedici olduğu ve günümüz tüketi elektroniği uygulamalarının çoğunun beklentilerini karşıladığı gözlemlenmiştir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iii
ABSTRACT iv
ÖZET v
LIST OF FIGURES ix
LIST OF TABLES xv
LIST OF SYMBOLS / ABBREVIATIONS xvi
1. INTRODUCTION 1
1.1. Motion Estimation 2
1.2. Problematic Situations in Motion Estimation 3
1.2.1. Occlusion & Appearance 3
1.2.2. Periodic Structures 3
1.2.3. Illumination Changes 4
1.2.4. Lack of Texture 4
1.2.5. Noise 4
1.3. Motion Based Applications 5
1.3.1. Compression Applications 5
1.3.2. Video Enhancement Applications 6
1.4. Problem Statement 8
1.5. Outline of the Thesis 9
2. MOTION ESTIMATION METHODS 10
2.1. Block Matching Based Methods 10
2.1.1. Full Search Methods 13
2.1.2. Hierarchical Search Methods 14
2.1.3. Fast Search Methods 15
2.2. Optical Flow Based Methods 20
2.2.1. Differential Techniques 21
2.2.2. Region-Based Techniques 25
2.2.3. Energy-Based Techniques 26
2.2.4. Phase-Based Techniques 27
2.3. True Motion Estimation 27

3. POST-PROCESSING OF MOTION VECTOR FIELDS 33
3.1. Introduction 33
3.2. MV Neighborhood Based Processing Methods 34
3.2.1. Mean Filters 34
3.2.2. Median Filters 35
3.2.3. Other Motion Vector Information Based Filters 37
3.3. Side Information Based Processing Methods 39
3.3.1. Alpha-Trimmed Mean Filters 40
3.3.2. Adaptively Weighted Median Filters 41
3.3.3. Other Side Information Based Methods 42
4. CONTRIBUTIONS TO VECTOR FIELD POST-PROCESSING METHODS . . 45
4.1. Motion Discontinuity Based VMF Smoothing 45
4.1.1. Discontinuity Detection 47
4.1.2. Classifier 49
4.1.3. Vector Median Filtering 50
4.1.4. Error Check 50
4.2. Low Complexity WVMF Smoothing 51
4.2.1. Discontinuity Detector 53
4.2.2. Candidate Selector 53
4.2.3. Weighted Vector Median Filtering 56
4.2.4. Error Check 57
4.3. Refinement Related Contributions 57
5. APPLICATIONS AND RESULTS 60
5.1. Performance Measurement Methods 60
5.1.1. Mean Squared Error 60
5.1.2. Vector Field Smoothness Indicator 61
5.1.3. Structural Similarity Index 61
5.2. MVF Smoothing Methods 64
5.2.1. Mean Filtering Strategy 64
5.2.2. Median Filtering Strategy 68
5.2.3. Vector Median Filtering (VMF) Strategies 72
5.2.4. Motion Discontinuity Based VMF Strategy 77
5.2.5. Weighted Vector Median Filtering (WVMF) Strategy 81

5.2.6. Low Complexity WVMF Strategy 85
5.2.7. Comparison of Implemented Methods 89
5.3. Frame Rate Conversion Application 90
5.3.1. Motion Estimation Block 91
5.3.2. MVF Post-Processing Block 95
5.3.3. Motion Compensated Frame Interpolation Block 99
5.3.4. Effects of MVF Post-Processing on FRC Performance 101
6. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK 107
APPENDIX A: IMPLEMENTED FRC TESTBENCH ENVIRONMENT 110
APPENDIX B: SYNTHETIC IMAGE SEQUENCES 114
REFERENCES 118
REFERENCES NOT CITED 124

LIST OF FIGURES

Figure 1.1. An example flow for the transmission of visual data 5

Figure 1.2. General flow of VE applications which use ME 6

Figure 1.3. FRC for frame rate doubling 7

Figure 1.4. De-Interlacing application 7

Figure 2.1. Block Matching based motion search 10

Figure 2.2. Pyramids of 3-level Hierarchical Search 14

Figure 2.3. Search locations for Three Step Search 16

Figure 2.4. Search steps of the Four Step Search 17

Figure 2.5. Search locations for 2D Logarithmic Search 18

Figure 2.6. Search pattern of Diamond Search 19

Figure 2.7. Fine and coarse search locations of Hexagonal Search 20

Figure 2.8. Constraint equations for estimation of Optical Flow 22

Figure 2.9. Spatial and temporal neighbors for 3D Recursive Search 29

Figure 3.1. Mean Filtering of MVF 35

Figure 3.2. Wavelet transform 38

Figure 3.3. MVF processing scheme of Dane and Nguyen 39

Figure 3.4. MVF processing scheme of Sony Advanced Technology Center 42

Figure 3.5. Refinement scheme of Ching and Hu 43

Figure 4.1. Failure case for the Median Filtering 46

Figure 4.2. Block diagram for Motion Discontinuity Based MVF method 47

Figure 4.3. An illustrative motion discontinuity region 49

Figure 4.4. Low Complexity WVMF 52

Figure 4.5. Flow chart for the candidate selection of Low Complexity WVMF 54

Figure 4.6. Utilized neighborhood for De Haan's refinement scheme 58

Figure 5.1. Flow for Structural Similarity metric 62

Figure 5.2. Ground Truth vs. Smoothed MVF for Mean Filtering of IS-I 65

Figure 5.3. Original vs. Smoothed MVF for Mean Filtering of IS-I 66

Figure 5.4. Ground Truth vs. Smoothed MVF for Mean Filtering of IS–II 66

Figure 5.5. Original vs. Smoothed MVF for Mean Filtering of IS–II 67

Figure 5.6. Ground Truth vs. Smoothed MVF for Mean Filtering of IS–III 67

Figure 5.7. Original vs. Smoothed MVF for Mean Filtering of IS–III 68

Figure 5.8. Ground Truth vs. Smoothed MVF for Median Filtering of IS–I 69

Figure 5.9. Original vs. Smoothed MVF for Median Filtering of IS–I 70

Figure 5.10. Ground Truth vs. Smoothed MVF for Median Filtering of IS–II 70

Figure 5.11. Original vs. Smoothed MVF for Median Filtering of IS–II 71

Figure 5.12. Ground Truth vs. Smoothed MVF for Median Filtering of IS–III 71

Figure 5.13. Original vs. Smoothed MVF for Median Filtering of IS–III 72

Figure 5.14. Ground Truth vs. Smoothed MVF for Vector Median Filtering of IS–I . 74

Figure 5.15. Original vs. Smoothed MVF for Vector Median Filtering of IS–I 75

Figure 5.16. Ground Truth vs. Smoothed MVF for Vector Median Filtering of IS–II 75

Figure 5.17. Original vs. Smoothed MVF for Vector Median Filtering of IS–II 76

Figure 5.18. Ground Truth vs. Smoothed MVF for Vector Median Filtering of IS–III 76

Figure 5.19. Original vs. Smoothed MVF for Vector Median Filtering of IS–III 77

Figure 5.20. Ground Truth vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–I 78

Figure 5.21. Original vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–I 79

Figure 5.22. Ground Truth vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–II 79

Figure 5.23. Original vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–II 80

Figure 5.24. Ground Truth vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–III 80

Figure 5.25. Original vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–III 81

Figure 5.26. Ground Truth vs. Smoothed MVF for WVMF of IS–I 82

Figure 5.27. Original vs. Smoothed MVF for WVMF of IS–I 83

Figure 5.28. Ground Truth vs. Smoothed MVF for WVMF of IS–II 83

Figure 5.29. Original vs. Smoothed MVF for WVMF of IS–II 84

Figure 5.30. Ground Truth vs. Smoothed MVF for WVMF of IS–III 84

Figure 5.31. Original vs. Smoothed MVF for WVMF of IS–III 85

Figure 5.32. Ground Truth vs. Smoothed MVF for Low Complexity WVMF of IS–I 86

Figure 5.33. Original vs. Smoothed MVF for Low Complexity WVMF of IS–I 86

Figure 5.34. Ground Truth vs. Smoothed MVF for Low Complexity WVMF of IS–II 87

Figure 5.35. Original vs. Smoothed MVF for Low Complexity WVMF of IS–II 87

Figure 5.36. Ground Truth vs. Smoothed MVF for Low Complexity WVMF of IS–III 88

Figure 5.37. Original vs. Smoothed MVF for Low Complexity WVMF of IS–III . . . 88

Figure 5.38. Block diagram of the implemented FRC application 91

Figure 5.39. 1st and 2nd frames of “Bus.avi” sequence 92

Figure 5.40. MVF produced by Full Search 93

Figure 5.41. MVF produced by Hierarchical Search 93

Figure 5.42. MVF produced by Hexagonal Search 94

Figure 5.43. MVF produced by 3D Recursive Search 94

Figure 5.44. Frame which is interpolated between the 157^{th} and 158^{th} frames of the “Foreman.avi” sequence by using the MVF strategy 96

Figure 5.45. Frame which is interpolated between the 157^{th} and 158^{th} frames of the “Foreman.avi” sequence without using the MVF strategy 96

Figure 5.46. Frame which is interpolated between the 4^{th} and 5^{th} frames of the “Bus.avi” sequence by using the MVF strategy 97

Figure 5.47. Frame which is interpolated between the 4th and 5th frames of the "Bus.avi" sequence without using the MVF strategy 97

Figure 5.48. The effect of Refinement on the reduction of *Blocking Artifacts*: a) the output with using refinement, b) the output without using refinement . . 98

Figure 5.49. The effect of Refinement on the reduction of *Stair Artifacts*: a) the output with using refinement, b) the output without refinement 99

Figure 5.50. Bidirectional Frame Interpolation 100

Figure 5.51. Graceful Degradation based frame interpolation 101

Figure 5.52. PSNR plots for "Bus.avi" sequence 102

Figure 5.53. PSNR plots for "Foreman.avi" sequence 103

Figure 5.54. PSNR plots for "MobileCalender.avi" sequence 103

Figure 5.55. SSIM plots for "Bus.avi" sequence 104

Figure 5.56. SSIM plots for "Foreman.avi" sequence 105

Figure 5.57. SSIM plots for "MobileCalender.avi" sequence 105

Figure A.1. GUI of implemented FRC Testbench Environment 110

Figure A.2. Listbox for the implemented MVF smoothing methods 112

Figure A.3. Listbox for the implemented MVF refinement methods 113

Figure B.1. Synthetic Image Sequence – I 114

Figure B.2. Synthetic Image Sequence – II 114

Figure B.3. Synthetic Image Sequence – III 115

Figure B.4. True motion vectors for Synthetic Image Sequence – I 116

Figure B.5. True motion vectors for Synthetic Image Sequence – II 116

Figure B.6. True motion vectors for Synthetic Image Sequence – III 117

LIST OF TABLES

Table 3.1. Number of elementary operations to evaluate a distance in R^p 36

Table 4.1. Calculations count for the processing of the motion vectors with filtering window size of "3x3" and search block size of "NxN" 56

Table 5.1. Smoothness and MSE scores for Mean Filtering 65

Table 5.2. Smoothness and MSE scores for Median Filtering 68

Table 5.3. Comparison of the various VMF methods 73

Table 5.4. Smoothness and MSE scores for Motion Discontinuity Based VMF . . . 78

Table 5.5. Smoothness and MSE scores for WVMF 81

Table 5.6. Smoothness and MSE scores for Low Complexity WVMF 85

Table 5.7. Smoothness and MSE scores for all methods 89

LIST OF SYMBOLS / ABBREVIATIONS

$\vec{d}$	Candidate motion vector
$\vec{D}$	Selected motion vector
$I(\vec{r},t)$	Intensity of a pixel with coordinates r = (x,y) at time instant "t"
N	Search block dimension
$\vec{r}$	Pixel coordinates
u	Horizontal component of motion vector
v	Vertical component of motion vector
x	Horizontal pixel coordinate
y	Vertical pixel coordinate
Ω	Utilized pixel neighborhood
DI	De-Interlacing
FRC	Frame Rate Conversion
ME	Motion Estimation
MSE	Mean Squared Error
MVF	Motion Vector Field
PP	Post-Processing
SAD	Sum of Absolute Differences
SNR	Signal-to-Noise Ratio
SSIM	Structural Similarity
PSNR	Peak Signal-to-Noise Ratio
VE	Video Enhancement
2D	2-Dimensional
3D	3-Dimensional
3DRS	3-Dimensional Recursive Search

1. INTRODUCTION

The role of digital technologies, such as televisions, phones, computers, etc., in our daily life becomes more important day by day. Communicating with other people and reaching to the desired information at anywhere and anytime are very simple by using them. If the features of the considered devices are investigated, for most of them it is seen that the capability of handling visual information, i.e.: receiving/transmitting and playing visual contents, is perhaps the most common and distinguishing one.

Although visual information is very attractive from the perspective of users, compared to text or audio data traffic it is very difficult to handle its data traffic because of the need for transmission and storage of very huge amount of data. Therefore, it is definitely necessary to utilize intelligent strategies for the most effective usage of the limited transmission channel bandwidth and the limited storage media capacity. Utilization of video compression methods, using interlaced video format instead of progressive one and accordingly using lower frame rates during transmission are among the most important of these strategies.

Motion information of the content plays a key role in the realization of most of the strategies listed in the previous paragraph. For instance, in the case of video compression temporal redundancies between the frames of the content are eliminated by using motion information. As another example, motion information could be used for the conversion from the interlaced to the progressive format or for increasing the frame-rate of the received content. In literature, there exist many different ME approaches to extract the necessary motion information of the visual content. However, most of the existing ME methods do not produce true motion vector fields and these fields are not suitable to directly use in a FRC or similar application. Hence, it is necessary to apply some post-processing operations on the obtained MVF in order to obtain a true-motion vector field which is more suitable for the video enhancement applications. "*Smoothing*" and "*Refinement*" operations are the two most important post-processing operations. These operations aim to eliminate the erroneous vectors, to smooth-out the vector field without

destroying edges and to obtain denser vector fields for minimizing the resolution related artifacts.

1.1. Motion Estimation

Video material or any visual content is a sequence of frames which are the snapshots of a scene at consequent time instants. "*Motion Estimation*" is the name of the operations which are used for detecting/sensing the displacement of objects between two successive frames of a video.

While dealing with motion estimation and motion based applications, it is advantageous to understand the factors from which the motion between two successive frames arises. These factors could be listed as:

- Displacement of objects,
- Displacement of camera,
- Displacement of both objects and camera.

Most of the time, being aware of the source of motion may save computation power and also may increase the target application's performance if the utilized strategy is adapted based on the information of motion source.

A classical motion estimation algorithm takes two consequent frames of video as an input and the output is a "*Motion Vector Field*". Depending on the selected estimation strategy, MVF contains vectors which represent the displacement information of each pixel or block (pixel group) of reference frame. In practice, block-based motion estimation strategies are more popular than other ME strategies. The details of existing motion estimation methods will be given in the next chapter of the thesis.

Depending on the application, motion estimation algorithms are generally designed to achieve different goals. In some cases, obtaining a MVF as close as possible to the true one may be the primary objective, whereas for another case instead of obtaining highly

accurate motion vectors the main target could be the minimization of prediction error or entropy between considered frames.

1.2. Problematic Situations in Motion Estimation

Video sequences are generated by the projection of 3D real world onto the series of 2D images. When objects in real world move, 2D projections of these objects change correspondingly. Since this is not a one-to-one mapping, it is not possible to track the movements of real life correctly from 2D projections for all the times.

The most important ones of the problematic situations at which tracking the true motion is difficult or completely impossible are the following ones.

1.2.1. Occlusion & Appearance

Since motion estimation algorithms tries to find the motion of real world objects, in the case of occlusions and appearances it is not possible to match the two frames completely. Then, estimation algorithm will converge to a random point based on its similarity metric, but this result will not represent the movement that exists in the real world.

In the case of coding applications, occlusions and appearances produce higher prediction errors and smaller compression ratios, which are undesirable situations. For the case of video processing applications the result might be more dramatic if the application could not sense this situation and follow a special strategy to deal with it.

1.2.2. Periodic Structures

Motion estimation algorithms could be seen as an optimization process which looks for the optimum solution (optimum solution corresponds to the motion of real world objects) of a certain problem.

In the case of the existence of periodic structures, the probability to get stuck in local minimums increases greatly and the resultant motion estimation deviates radically from the actual one. For coding applications this problem does not produce an important artifact or performance degradation. However, for motion compensated video frame rate conversion and deinterlacing applications disturbing artifacts may arise because of this situation and motion estimation algorithm might be classified as poor because of the supplied wrong real world motion information.

1.2.3. Illumination Changes

Motion estimation algorithms generally assume that the illumination is constant between two consecutive frames. However, this assumption is not true for all the times and, in some cases although the objects do not move, the motion estimator detects displacement because of the changes in pixel intensity levels. So, this wrong detection causes deviations from the true-motion and the reliability of output motion vector field decreases.

1.2.4. Lack of Texture

Texture information of objects is very critical to track them. When texture information of an object is not adequate, then tracking its motion becomes very difficult in some cases. For instance, think a ball which turns around itself without changing its location. If the ball has no texture, then it is very probable to classify that ball as a stationary object.

1.2.5. Noise

Noise makes dealing with many of the real world problems such as ME very complicated. In the case of lack of texture or the case of similar texture characteristics, noise may result in the convergence to one of local minimums instead of the global solution. Although this is not a big problem for minimum error based applications, it results in a deviation from correct representation of real world motion characteristics.

1.3. Motion Based Applications

Visual information plays a key role in our lives, and as a result of this fact an important part of multimedia devices have a capability of handling (receiving, transmitting, recording and playing) visual data. However, dealing with visual information is not an easy job since the efficient transmission and storage of the huge amount of data, which belongs to it, is very hard. This difficulty gives birth to the development of some intelligent strategies which help to efficient utilization of limited transmission bandwidth and storage capacity.

In practice, utilizing compression techniques and performing data sub-sampling on spatial and/or temporal domains are the most commonly used methods in order to deal with huge amounts of data easier. In the following figure a typical flow of the visual data from the content provider to the user is demonstrated.

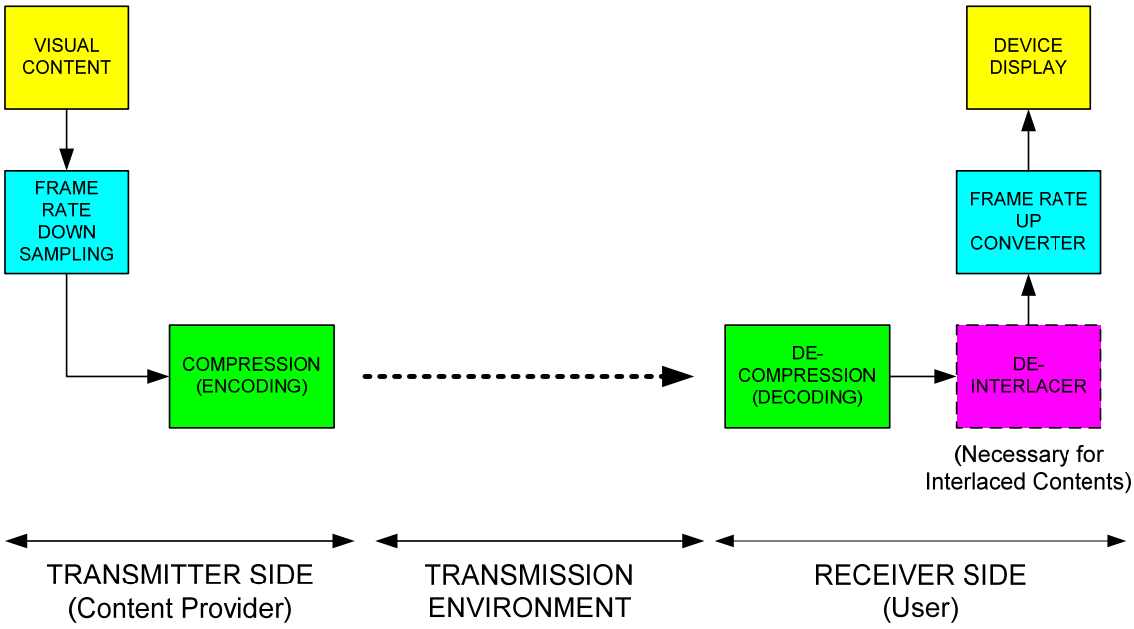

Figure 1.1. An example flow for the transmission of visual data

1.3.1. Compression Applications

Compression algorithms aim to represent input content with fewer bits. In order to achieve this goal, an algorithm analyzes the content and tries to eliminate the redundancies which exist in spatial and temporal domains. For obtaining really high compression ratios,

all of the most commonly used algorithms, such as MPEG2, MPEG4, H.263 and H.264, focus on the elimination of temporal redundancies by using the motion information between consequent frames.

Motion estimators, which are commonly used for compression purposes, generally use block-based approaches and the main goal of them is the minimization of error between frames. In other words, true-motion information is not a primary objective for these estimators and for a successful compression application true-motion information is not a must.

1.3.2. Video Enhancement Applications

Video Enhancement (VE) applications modify the input sequence in such a way that the resultant content has a higher visual quality. Motion information based VE algorithms generally require true-motion information unlike compression applications. The general flow of a motion based VE application could be seen in the following figure:

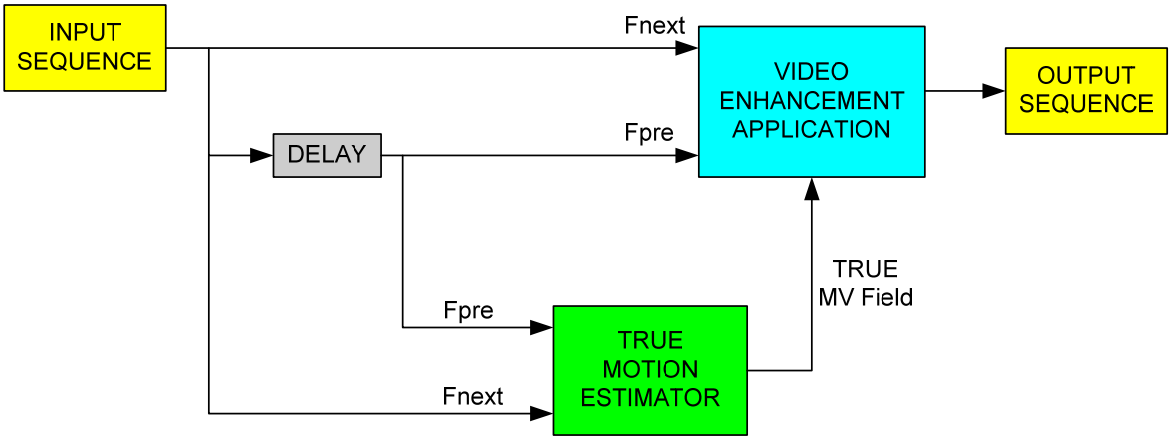

Figure 1.2. General flow of VE applications which use ME

Frame Rate Conversion (FRC) and De-Interlacing (DI) applications are two well known examples of motion based VE applications.

An FRC application performs up-sampling in the temporal (time) domain. In other word, new frames are generated by using the true-motion information in order to increase the frame rate of the input content and hence provide more live scenes to the viewer.

Although it is possible perform various up-sampling operations, the most commonly used and the simplest FRC application is frame doubling, which is seen in the following figure:

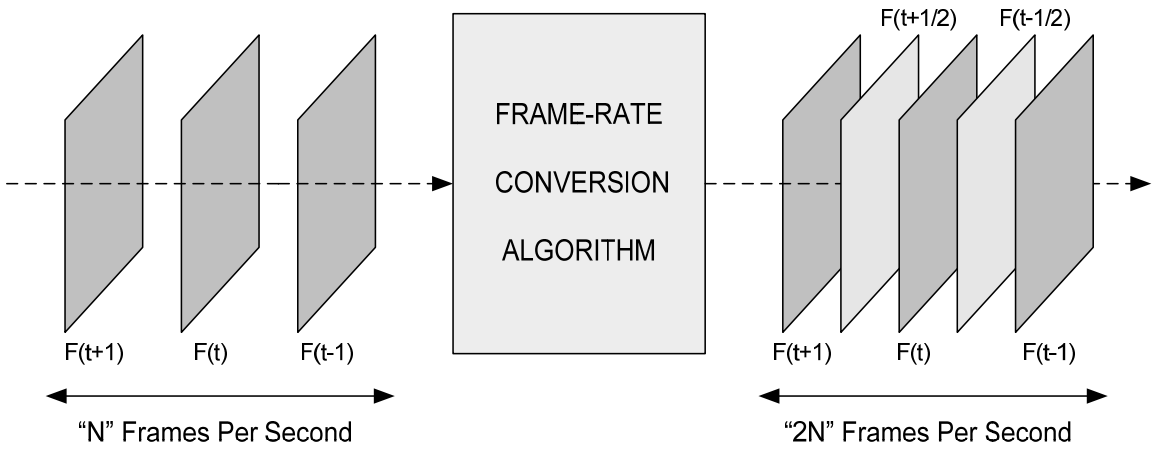

Figure 1.3. FRC for frame rate doubling

Similar to the FRC case, a DI application also performs an up-sampling. However, this up-sampling operation is performed in spatial domain instead of temporal (time) domain. Namely, fields of input content, which are in interlaced format, are converted to the full frames by interpolating the missing lines of each field as it is seen in the following figure:

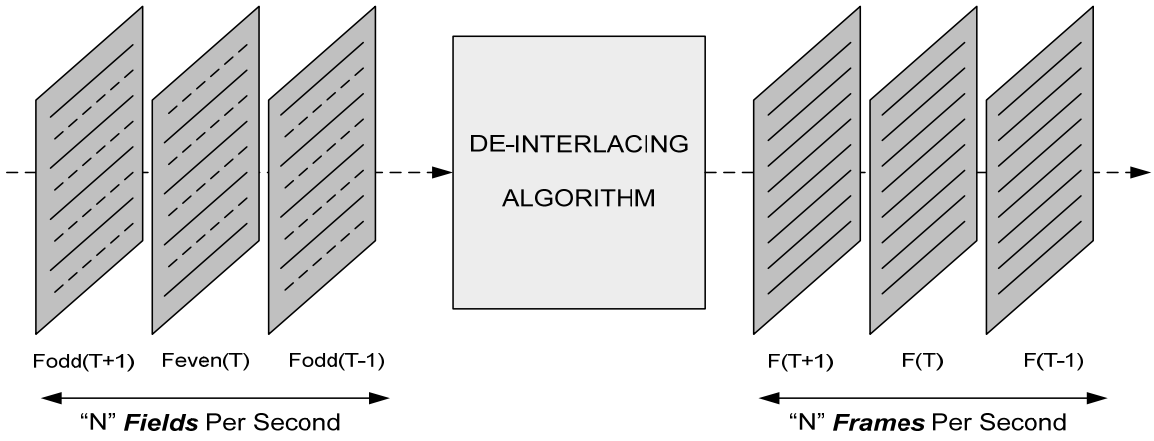

Figure 1.4. De-Interlacing application

1.4. Problem Statement

As explained in the previous sections, the transmission and storage of visual data is not an easy problem because of the huge size of existing data, and motion information based compression and VE algorithms play a key role to deal with this problem. However, especially in the case of VE applications (such as FRC and DI) the reliability of MVF and its closeness to the true motion information are very critical in terms of the visual quality of the final content.

Unfortunately, extracting true motion information from a video sequence is not an easy job and most of the high quality motion estimators are not realizable as a real-time system or the realization is very costly in terms of hardware requirements. In order to overcome these difficulties, one possible strategy is, in the user side of visual data transmission flow, the utilization of already "*existing information*" and performing less operations for realizing the desired VE applications. In detail, all transmission channels use some kind of compression algorithms on data and for video transmission case these algorithms generally embeds the motion information of incoming content into the compressed output. Then, instead of using a high complexity true-motion estimation algorithm on the user side of transmission, getting the already existing motion information from decoder block and by performing simpler MVF post-processing operations obtaining a MVF as true as possible is more logical in terms of having an efficient implementation. However such side information is not always available since the real time video FRC processing is most of the time done in TV chips which do not get MVF feedback from the decoder. Therefore to support our MVF smoothing and refinement studies we had to also consider different variations of some state-of-the-art hardware friendly ME techniques.

The main focus of this thesis is investigating MVF post-processing operations, namely "*Smoothing*" and "*Refinement*", which are used for modifying the motion information, that is obtained from an error minimization based motion estimator, in such a way that the resultant MVF is as suitable as VE applications. Also, distinct blocks (Motion Estimator, Smoothing, Refinement and Frame Rate Converter) are integrated to form a complete system and the effect of each one on the overall system performance was

observed, and some contributions and modifications are made to increase the final quality level.

1.5. Outline of the Thesis

In section 2, most commonly used motion estimation methods are investigated. Estimators from both error-minimizing and true motion estimating categories are considered. Section 3 gives the background of possible MVF post-processing operations. Details of the MVF post-processing related contributions are given in section 4. Section 5 shows the results of simulation outputs and also gives the details of the implemented FRC test bench platform. Section 6 concludes the thesis.

2. MOTION ESTIMATION METHODS

In this chapter, the theory of most commonly used motion estimation methods are given in detail and the true-motion estimation related issues are discussed.

2.1. Block Matching Based Methods

Block matching based techniques are the most commonly used motion estimation algorithms in practice. They are very suitable for the real time applications, such as coding and digital TV applications, because of their lower computational complexity level and implementation cost.

In the block matching based motion estimation, the reference frame (F_{ref}) is partitioned into reference blocks (generally with a size of 8x8 or 16x16 pixels), and the motion vector ($\vec{D}$) of each block is found by performing a search within the search window that is located on the search frame (F_{src}) as shown in Figure 2.1.

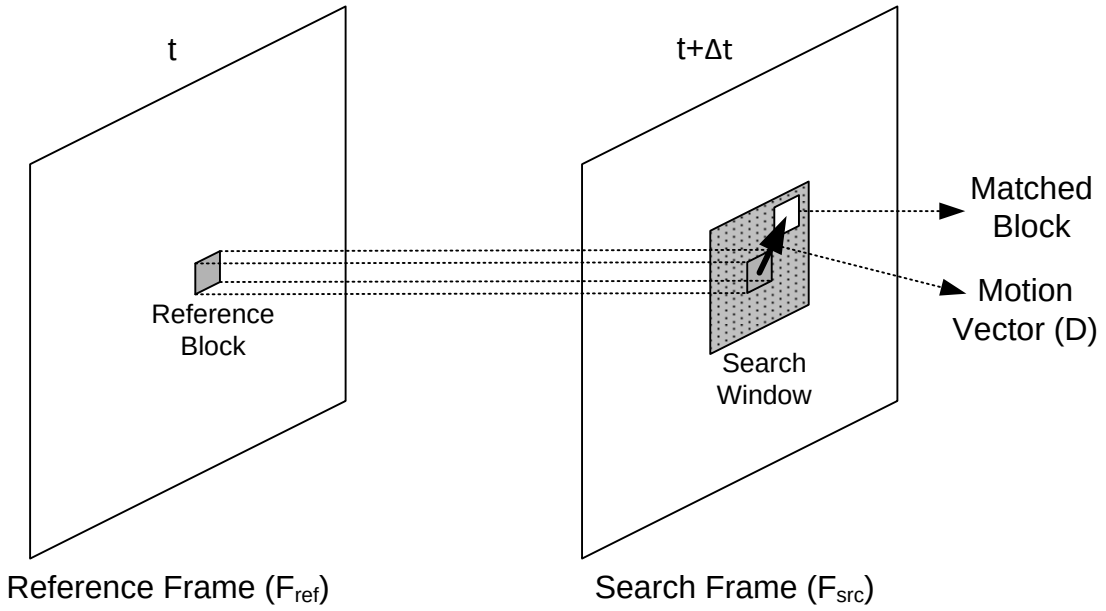

Figure 2.1. Block Matching based motion search

The motion model, which is used by the block matching based techniques, usually assumes that an image is composed of rigid objects in translational motion. Although this model is clearly restrictive, it is justified by the fact that complex motion can be decomposed as a sum of translational components [1]. Operations of a block based motion estimation algorithm could be expressed by using the following mathematical equation:

$$\vec{D} = \arg \min_{\vec{d} \in S} \sum_{\vec{r} \in W} f(I(\vec{r},t) - I(\vec{r}+\vec{d},t+\Delta t)) \quad (2.1)$$

Here, the term "$\vec{D}$" stands for the selected displacement (motion) vector of current reference block, "$\vec{d}$" represents the candidate motion vector which is selected from the set of all possible candidates (S) and "$I(\vec{r},t)$" represents the intensity value of a pixel with coordinates $\vec{r} = (x, y)$ at time frame instant 't'. Additionally, "$f(..)$" is the utilized cost metric which is used for finding the best matching block and "*W*" stands for the set of pixel coordinates for the considered reference block.

Operations of block matching based ME algorithms, which are formulated in equation 2.1, could be investigated in three steps:

- Segmentation of frames into blocks
- Decision for the best block matching metric (cost function)
- Comparison of the reference block with candidate blocks

The first step of every block matching algorithm is to decide for the size of blocks which will be used in the search process and whether the segmented blocks will overlap or not. These decisions are very critical for the performance and the computational complexity of the estimator. In the literature, the usage of blocks with the size of 16x16 pixels is very common for CIF (352x288) and higher resolution sequences, whereas for smaller resolutions 8x8 blocks are widely utilized. By using overlapped blocks it is possible to obtain multiple motion vector candidates for any region of frame, and it is possible to achieve a higher performance level in the enhancement application step. However, the overlapping strategy also increases the complexity of the algorithm since more than one search operations are performed for each location.

The second step in the motion search process is deciding for the matching metric that fits best with the goal of the motion estimation algorithm and target application. Since block matching algorithms assume that the objects are rigid and motion between two consecutive frames is translational, they generally search for the minimum intensity change between the reference block and candidate blocks to predict correct motion vector. Sum of Absolute Differences (SAD) is the most commonly used metric and it is defined in the following form:

$$SAD(\vec{d}) = \sum_{\vec{r} \in W} \left| I(\vec{r},t) - I(\vec{r}+\vec{d},t+\Delta t) \right| \tag{2.2}$$

Mean Squared Error (MSE) is another commonly used metric which uses the square of error terms instead of absolute values of them:

$$MSE(\vec{d}) = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \left(I(\vec{r},t) - I(\vec{r}-\vec{d},t-\Delta t) \right)^2 \tag{2.3}$$

In equation 2.3, *N* represents the one dimension of a square block and N^2 is the number of pixels that are located inside a block. Although it is proved that MSE gives better results than SAD, as a result of using higher order statistics, most of the time SAD is more attractive for real-time applications due to lower hardware implementation cost.

After making the important decisions in the first and second steps, the third step of ME process is straightforward: each reference block is compared with candidate blocks, which are located inside its corresponding search window, and the displacement vector of the candidate block with the minimum cost function value, is selected as the output of the search. The range of search window is also an adjustable parameter and it affects the computational complexity and performance of the motion estimator. For the most common Standard Definition (SD) contents (720x576 or 720x480 pixels), the size of true motion vectors are generally in the range of 10-15 pixels and using a search range of ±16 pixels almost all the time gives satisfactory results. However, in the case of a fast moving scene or a sports scene such a range becomes insufficient and utilization of higher search ranges, such as ±32 pixels or higher gives better results.

Performance of block matching algorithms is generally evaluated by using the PSNR (Peak Signal to Noise Ratio) metric. It measures the difference between an original frame and a motion compensated frame created from a reference frame by using the obtained motion vectors. PSNR could be defined in the following form:

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2}{\mathrm{MSE}}\right) \tag{2.4}$$

Block matching algorithms might be classified into three groups based on the strategy followed during the search for most probable displacement. In the following subsections, the details of these methodologies and their representative examples from the literature will be examined.

2.1.1. Full Search Methods

Full search algorithms are the most straightforward form of block matching techniques in terms of search strategy. These algorithms check all the possible search points inside the given search window, guaranteeing the global minimum. However this is an exhaustive process and its computational complexity becomes too high in the case of larger search windows.

Full search algorithm achieves the highest possible PSNR value since it checks all possible candidates in the given search window. All other faster block search algorithms try to reach its performance by using less computational power.

In literature, there exist different approaches to reduce the computational complexity of full search algorithm while preserving the same PSNR level. These approaches generally focus on the sub-sampling of the pixels of the processed blocks [2], simplification of the matching criteria and finding an early termination criterion to prevent the extra computational burden [3].

2.1.2. Hierarchical Search Methods

Hierarchical methods perform motion estimation by using multiple resolutions of the original frames. The idea behind using multiple resolutions is to first obtain a coarse motion vector candidate by using less effort in lower resolutions and then finding the correct vector with a fine search inside a very small part of the highest resolution level in the following iterations. By this way, the search algorithm does not check all the possible search locations of the search window and it focuses only on a small vicinity of the coarse estimate that comes from lower resolutions. Therefore search locations, which need to be searched during estimation process, are greatly decreased without a significant PSNR reduction.

In the first step of the algorithm, considered frames are low-pass filtered and then sub-sampled to obtain lower resolutions (higher levels) of the pyramids. In general sub-sampling factor is selected as two in both directions and three different resolutions are utilized during the operations.

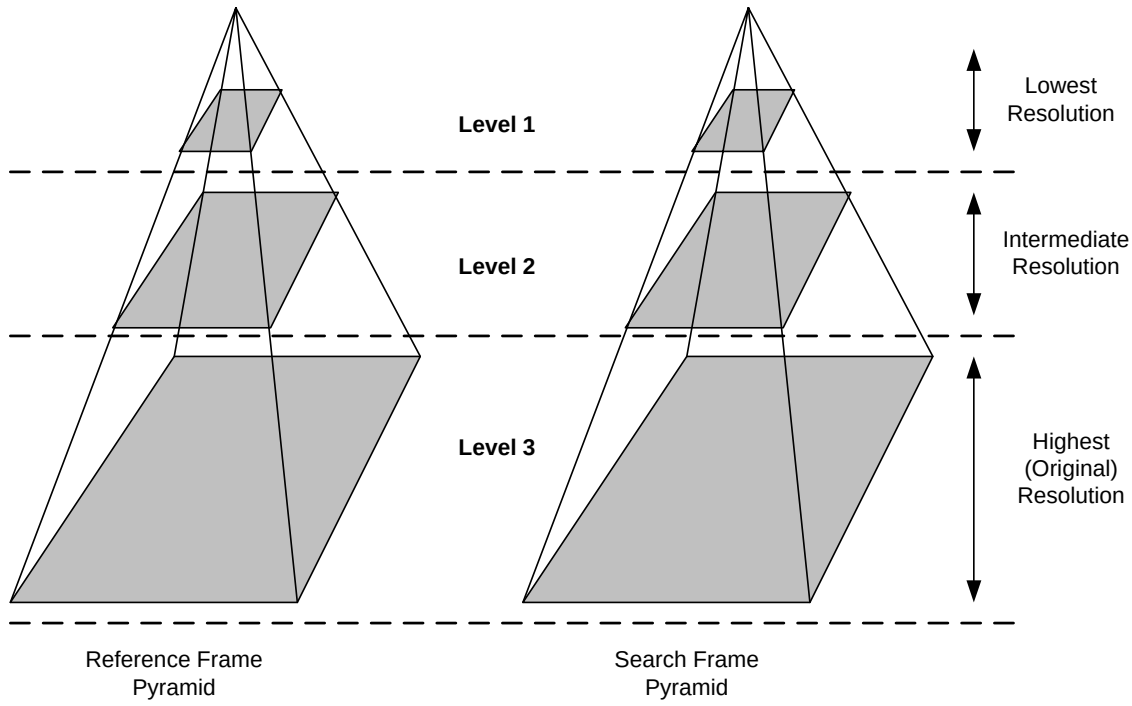

Figure 2.2. Pyramids of 3-level Hierarchical Search

After obtaining different resolutions of original frames, search operations start from the lowest resolution of the image and end at the highest (original) resolution of the image. When passing from one resolution to its higher resolution, it is very important to decide for a correct strategy to assign coarse motion vectors of the new (higher) resolution frame. This vector assignment process is some kind of up-sampling and the decided strategy is very important for the performance of algorithm.

The hierarchical approach obtains very close results with the full search strategy while reducing the computational complexity. However, its complexity is still too high when compared to fast search approaches. As a result of multi-resolution operations (motion vector of a block at lower resolution affects motion vector of multiple blocks for the higher resolution), hierarchical search produces similar motion vectors for neighbor blocks. This property is beneficial in terms of true motion. On the other hand, multi-resolution operation produces some drawbacks. For instance, if an erroneous vector is produced in the lower resolutions, then it is not possible to compensate this error in the following steps for most of the time.

2.1.3. Fast Search Methods

Fast search methods aim to achieve similar PSNR value with full search algorithm while performing significantly less operations. They are very commonly used for coding and real-time applications because of their lower computational power requirement and lower search time.

To decrease the computational complexity of search operations, fast algorithms generally focus on reducing the number of test points, which are checked during estimation steps. Then, the choice of new candidate locations is very important for the performance of the algorithm. Although there are many different approaches for this candidate selection process, the main assumption of all is "*error surface has a monotonically decreasing structure*". However, this assumption is not true for all the times and it is very probable for them to get stuck in a local minimum. Therefore performance of a fast algorithm is very closely related with how it deals with the local minimums.

Few of the well known and most commonly used fast search algorithms are Three Step Search (TSS), Four Step Search, Hexagonal Search, Logarithmic Search and Diamond Search. The details for these methods will be given in the following paragraphs.

Three-Step Search (TSS) is known as the pioneer of the fast motion search strategies and it was proposed by T. Koga in 1981. [4] As it was stated earlier, the main aim of the algorithm is to reach or at least to approximate the PSNR value of full search algorithm by checking less candidate locations.

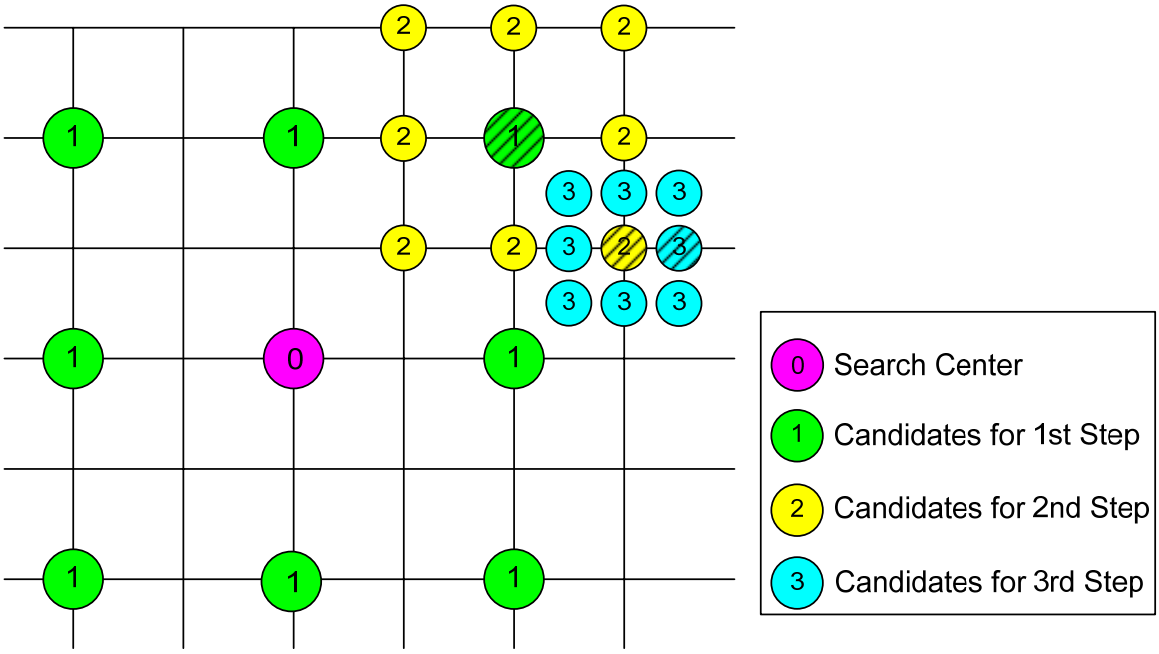

Figure 2.3. Search locations for Three Step Search

The search procedure starts from the center of the search window. 8 search locations around the center point are checked in the first step and the location with smallest matching criteria is selected as the center of the next step. Then, the same procedure is repeated and the center point for the final step is determined. Finally, the center point and its 8 neighbors are compared to find the winner of the search, and the motion vector of the winner point is assigned as the displacement vector of the current block.

Original three step strategy is designed for ±7 pixels around the center point and it does not permit to search for large motion vectors. Then, instead of “3” steps, “N” steps are used for searching larger areas by using the same strategy.

Although it works well for moderate motions, in the case of small deviations TSS is not successful since it does not check the neighborhood of the center position in the first step. To solve this problem, New Three Step Search (NTSS) algorithm was proposed [5]. The distinction between TSS and NTSS is in the first step: in addition to 8 search points of 1st step, which are shown in Figure 2.3, NTSS also checks the center point and its 8 neighbors in the first step and the process is terminated if the minimum similarity measure is reached for one of these new candidates. By this way, NTSS also works well for the small deviations case.

Four-Step Search (4SS), which is very similar to the TSS, is designed based on the real world image sequence's characteristic of the center-biased motion vector distribution [6]. As a result, 4SS has a center-biased checking point pattern. In the first step, center point of search window and its 8 neighbors are checked as shown in Figure 2.4-a. Based on the search result, algorithm passes to the second step if the selected point is one of the 8 neighbor locations or passes to the fourth step if the selected point is the center location. As it is shown in Figure 2.4, for the first three steps, 'neighborhood' corresponds to two pixel grids, and for the final step, which is the fine search step, the neighborhood is defined by using one pixel grid distance.

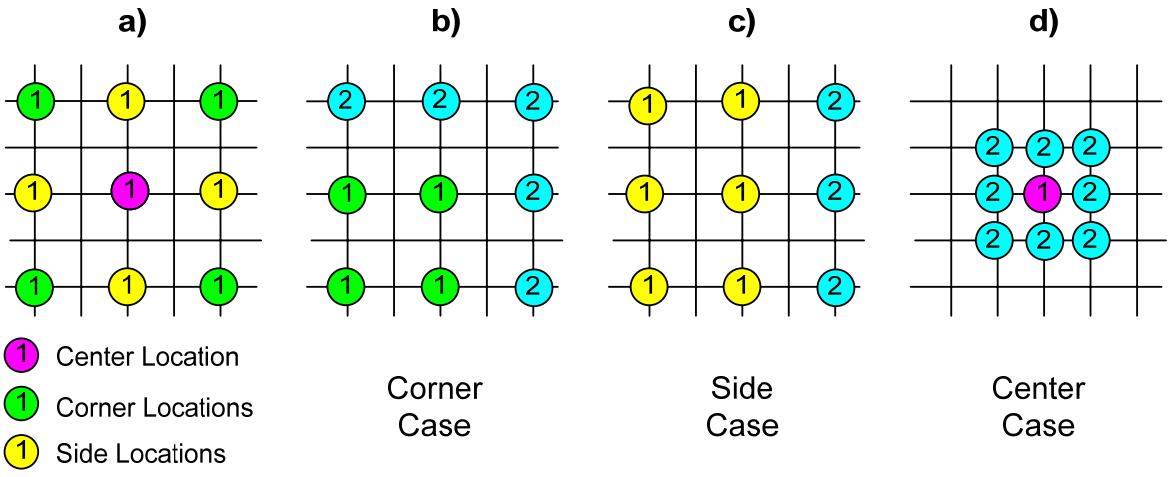

Figure 2.4. Search steps of the Four Step Search

4SS algorithm performs better when it is compared with the well known TSS algorithm, and when it is compared with the NTSS it has similar PSNR measure while average computation count of 4SS is smaller than the NTSS.

Another fast search strategy, namely 2D Logarithmic Search Algorithm (2DLS), is designed based on the assumption that the human eye is more sensitive to horizontal and vertical movements than the movements in other directions [7]. The search strategy of the algorithm is very similar with TSS. The basic distinction between them is the selection of check points during intermediate steps. Based on the assumption of visual sensitivity to vertical and horizontal movements, the logarithmic search neglects the diagonal neighborhoods during intermediate steps. In the final step, similar to 3SS, all 8 neighbors are taken into account, and final decision is made among horizontal, vertical and diagonal direction candidates.

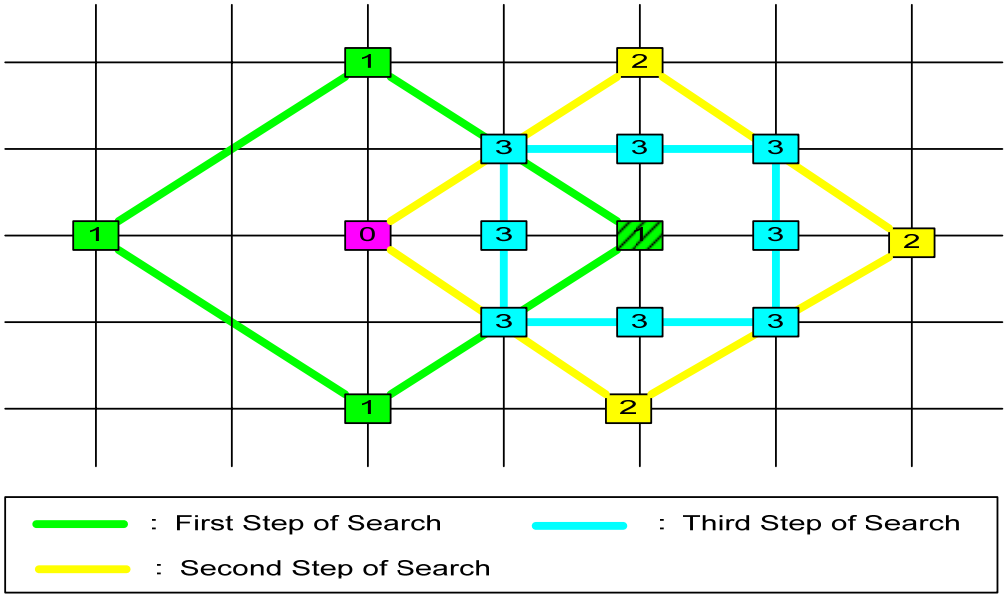

Figure 2.5. Search locations for 2D Logarithmic Search

Diamond Search (DS) is another commonly used fast strategy. It is very similar to 4SS in terms of search strategy and is very similar to the 2DLS in terms of the utilized search pattern. Algorithm uses two patterns during search operations:

- *Large diamond search pattern* (LDSP): It comprises nine checking points from which eight points surround the center one to compose a diamond shape.
- *Small diamond search pattern* (SDSP): It contains five checking point to create a small diamond.

In the search procedure of the DS algorithm, LDSP is repeatedly used until the step in which the minimum block distortion (MBD) occurs at the center point. The search pattern is then switched from LDSP to SDSP as reaching to the final search stage. Among the five checking points in SDSP, the position yielding the MBD provides the motion vector of the best matching block [8].

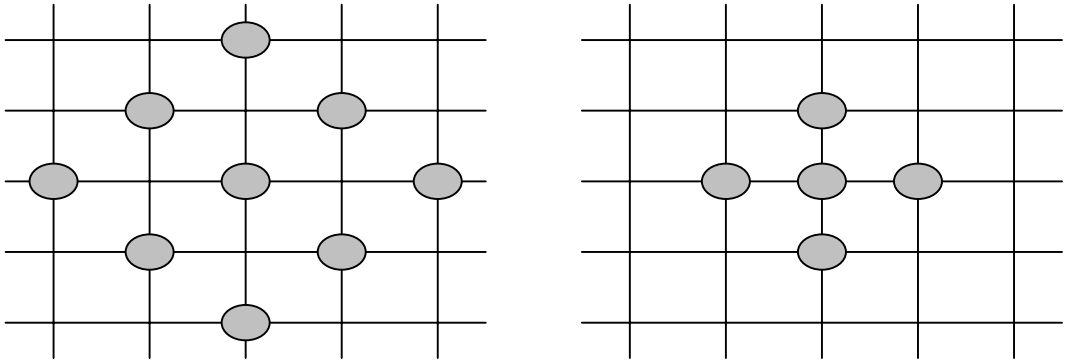

Figure 2.6. Search pattern of Diamond Search

While preserving nearly the same performance level with NTSS, diamond search algorithm achieves this score by performing 20% less operations. Because of these desirable properties, diamond search is very popular among fast search algorithms. In literature there exist many improvements of this algorithm. As an example, cross diamond search [9, 10] is one of the most important improvements which preserves (or improves -in some cases-) the performance level while halving the computational complexity of original diamond search.

Hexagonal Search (HEXBS) algorithm is the most recent trend for fast search algorithm strategies. [11] In addition to its successful PSNR level, it is very efficient in terms of computational complexity. Hexagonal search algorithm has two search procedures:

- The Coarse Search
- The Fine-Resolution Search

The coarse search procedure firstly locates a region where the optimal motion vector is expected to lie, using the large hexagon search pattern consisting of 6 end-points. The coarse search continues based on a gradient scheme until the center point of the hexagon has the smallest distortion. During the coarse search, only 3 new locations are checked for each step. After a hexagonal area is located in the coarse search, the following fine-resolution search looks into the small area enclosed by the large hexagon for focused inner search using the center location and its 8 neighbor locations. Finally, the result of the fine-resolution search gives the final motion vector.

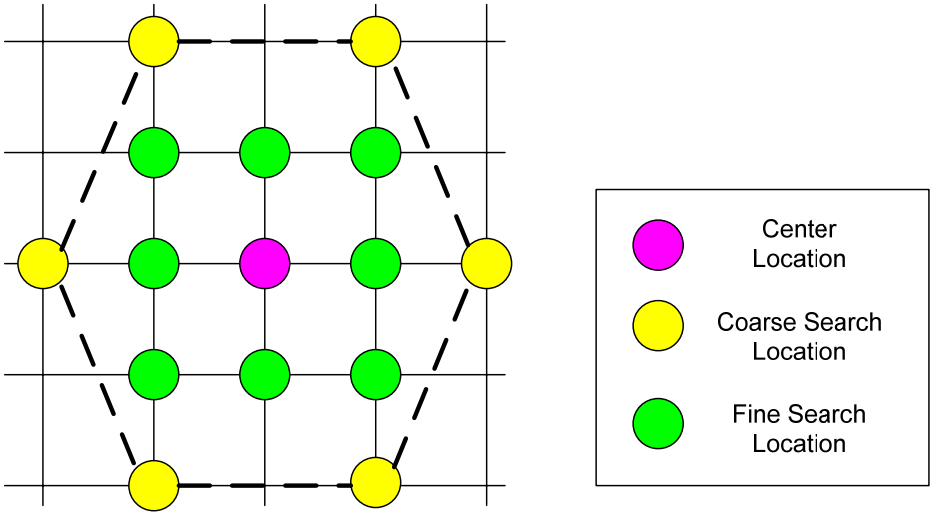

Figure 2.7. Fine and coarse search locations of Hexagonal Search

In literature, there exist efforts to enhance the performance of hexagonal search. These efforts generally focus on the fine-resolution part and try to decrease the number of search locations while obtaining the same PSNR level [12].

2.2. Optical Flow Based Methods

Optical flow methods, similar to the block matching based estimation case, aim to approximate the motion of the objects within the considered frame sequence from gray value or color value changes. However, they generally assign a motion vector for each pixel of reference frame instead of each block.

In literature, there exist many different strategies to estimate optical flow. However, many of these techniques can be viewed conceptually in terms of three stages of processing [13]:

- Prefiltering or smoothing with low-pass/band-pass filters in order to extract signal structure of interest and to enhance the signal-to-noise ratio (SNR).
- The extraction of basic measurements, such as spatio-temporal derivatives or local correlation surfaces.
- The integration of the previously obtained measurements to produce a 2D flow field, which often involves assumptions about the smoothness of the underlying flow field.

Although optical flow based methods generally produce denser and more accurate motion fields than block matching based methods, they are not preferred in real-time applications because of their higher computational complexity.

In [13], Barron classifies the optical flow techniques into four groups. The detailed information for each group is given in the following sections.

2.2.1. Differential Techniques

Differential techniques estimate the optical flow by using the spatiotemporal derivatives of image intensity or filtered versions of the images (low-pass or band-pass filters are used). They generally use first or second order derivatives of intensity values during estimation operations.

In the case of first order derivatives, it is assumed that the objects have a translational behavior:

$$I(x , y , t) = I(x - u.t , y - v.t , 0) \tag{2.5}$$

In equation 2.5, I(x,y,t) is the intensity value of the pixel, which has coordinates of (x,y), at time instant t. Additionally, ‘u’ and ‘v’ represent the velocity of considered pixel in the

horizontal and the vertical directions. It is also assumed that the intensity is conserved during translational motion:

$$\frac{dI(x,y,t)}{dt}=0 \tag{2.6}$$

By extending the above equation and writing the partial derivatives explicitly, then the following equation, which is called as Gradient Constraint Equation, is obtained:

$$\frac{dI(x,y,t)}{dx}.\frac{dx}{dt}+\frac{dI(x,y,t)}{dy}.\frac{dy}{dt}+\frac{dI(x,y,t)}{dt}=0 \tag{2.7}$$

$$\frac{dI(x,y,t)}{dx}.u+\frac{dI(x,y,t)}{dy}.v+\frac{dI(x,y,t)}{dt}=0 \tag{2.8}$$

In equation 2.7 and 2.8, $\left(\frac{dI}{dx},\frac{dI}{dy}\right)$ is called as the image gradient and values for these derivative terms could be found by using the existing information. However, this single equation is not enough to obtain a unique solution for the motion vector $\vec{D}=(u,v)$. Then, it is necessary to find an additional constraint equation to solve the problem.

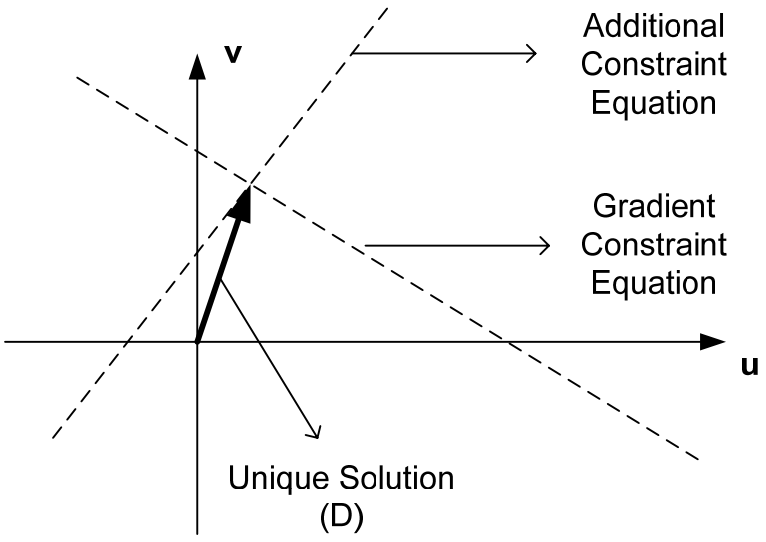

Figure 2.8. Constraint equations for estimation of Optical Flow

Second order differential methods use second order derivatives to constrain 2D velocity:

$$\begin{bmatrix} I_{xx}(x,y,t) & I_{yx}(x,y,t) \\ I_{xy}(x,y,t) & I_{yy}(x,y,t) \end{bmatrix} \begin{pmatrix} u \\ v \end{pmatrix} + \begin{pmatrix} I_{tx}(x,y,t) \\ I_{ty}(x,y,t) \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{2.9}$$

Equation 2.9 can be derived from the conservation of $\nabla I(x,y,t)$. In other words, conservation of $\nabla I(x,y,t)$ implies that the first order deformations of intensity should not be present. This is a stronger restriction than equation 2.6. Then, because of this strong restriction and the sensitivity of numerical differentiations, the velocity estimates from 2nd order methods are often assumed to be sparser and less accurate than estimates from 1st order methods.

In [14], Horn and Schunk combine the gradient constraint that is given in equation 2.8 with a global smoothness term to obtain a unique motion vector estimate by minimizing the following equation:

$$\int_{\Omega} \left(\nabla I . \vec{d} + I_t \right)^2 + \lambda^2 . \left(\left\| \nabla u \right\|_2^2 + \left\| \nabla v \right\|_2^2 \right) d\vec{r} \tag{2.10}$$

where the magnitude of λ defines the influence of the smoothness term. Algorithm starts with zero initial conditions, and the final motion vector is obtained with an iterative approach by using the equations:

$$\begin{aligned} u^{k+1} &= \bar{u}^k - \frac{I_x [I_x . \bar{u}^k + I_y . \bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2} \\ v^{k+1} &= \bar{v}^k - \frac{I_y [I_x . \bar{u}^k + I_y . \bar{v}^k + I_t]}{\alpha^2 + I_x^2 + I_y^2} \end{aligned} \tag{2.11}$$

where k denotes the iteration number, $\bar{u}^k$ and $\bar{v}^k$ denote neighborhood averages of u^k and v^k. The method uses the first order differentiations to estimate the intensity derivatives.

Another important example of differential optical flow methods is the one that is proposed by Lucas and Kanade [15]. In their estimation algorithm, the second constraint, which is used for obtaining a unique solution, is created by using the assumption that in a small neighborhood all pixels have the same or very similar motion characteristic. Then the optical flow estimation problem boils down to Least-Squares based error minimization problem:

$$E\left(\vec{d}\right) = \sum_{\vec{r}\in\Omega} G(\vec{r}).\left[\nabla I.\vec{d}+I_t\right]^2 \tag{2.12}$$

where $G(\vec{r})$ is a windowing function that gives more influence to constraints at the center of the neighborhood than those at the periphery. The motion vector $\vec{d}$, which minimizes the $E\left(\vec{d}\right)$, is selected as the current pixel's motion vector. The minimum for Equation 2.12 could be found from its critical points:

$$\begin{aligned}\frac{\partial E\left(\vec{d}\right)}{\partial u} &= \sum_{\vec{r}} G\left(\vec{r}\right).\left[u.I_x^{\,2} + v.I_x.I_y + I_x.I_t\right] = 0\\ \frac{\partial E\left(\vec{d}\right)}{\partial v} &= \sum_{\vec{r}} G\left(\vec{r}\right).\left[v.I_y^{\,2} + u.I_x.I_y + I_y.I_t\right] = 0\end{aligned} \tag{2.13}$$

Then, the solution could easily be obtained if the above equations are written in matrix form:

$$M.d = b \tag{2.14}$$

where

$$M = \begin{bmatrix} \sum G.I_x^{\,2} & \sum G.I_x.I_y \\ \sum G.I_x.I_y & \sum G.I_y^{\,2} \end{bmatrix}\ ,\ b = -\begin{bmatrix} \sum G.I_x.I_t \\ \sum G.I_y.I_t \end{bmatrix} \tag{2.15}$$

When M matrix is full rank, rank(M)=2 since it is a 2x2 matrix, the Least Square Estimate is $\hat{d} = M^{-1}.b$.

2.2.2. Region-Based Techniques

Accurate numerical differentiation may be impractical because of noise, because a small number of frames exist or because of aliasing in the image acquisition process. In these cases differential optical flow estimation approaches may be inappropriate and it is natural to turn to region-based matching techniques. Such approaches define the velocity as a shift that yields the best fit between image regions at different times [13].

In order to match the image regions, different similarity measures are used, such as normalized cross-correlation or distance minimization. One commonly used distance minimization metric is Sum-of-Squared Distance (SSD):

$$SSD_{1,2}(x,y,\vec{d}) = \sum_{j=-n}^{n}\sum_{i=-n}^{n} G(i,j).\left[I_1((x,y)+(i,j)) - I_2((x,y)+\vec{d}+(i,j)) \right]^2 \qquad (2.16)$$

where G denotes a discrete 2D window function. If SSD is compared with SAD and MSE, which are commonly used metrics for block-matching based motion estimation methods, it is easy to see the obvious similarity between them. Then, it could be said that the region based optical flow techniques are very close to the block matching based motion estimation techniques in theory.

Anandan's method [16] is an important example of region based optical flow techniques. Method uses a Laplacian pyramid and a coarse-to-fine SSD-based matching strategy. The strategy is very similar to the Hierarchical Block Matching algorithm and the utilized pyramid allows the computation of large displacements.

Anandan employs a smoothness constraint on the velocity estimates and, at the end of an iterative approach; subpixel accurate flow estimates are obtained.

Another illustrative example of region based methods is the one that is proposed by Singh. [17, 18] This is a two stage method. In the first stage, SSD values for three adjacent band-pass filtered images are computed:

$$SSD_0(x,y,\vec{d}) = SSD_{0,1}(x,y,\vec{d}) + SSD_{0,-1}(x,y,-\vec{d}) \tag{2.17}$$

where $SSD_{i,j}$ is given in equation 2.16. As a result of the summation of two terms, i.e.: $SSD_{0,1}$ and $SSD_{0,-1}$, spurious minima due to noise or periodic texture.

In the second stage, SSD_0 term that is obtained in the first stage is converted into a probability distribution using:

$$R_c(\vec{d}) = e^{-k.SSD_0} \tag{2.18}$$

Then, the subpixel accurate flow vector is as the mean of the distribution:

$$u = \frac{\sum R_c(\vec{d}).d_x}{R_c(\vec{d})}$$
$$v = \frac{\sum R_c(\vec{d}).d_y}{R_c(\vec{d})} \tag{2.19}$$

2.2.3. Energy-Based Techniques

Energy-Based optical flow techniques find the desired displacement information by using the output energy of velocity-tuned filters. These are also called frequency-based methods owing to the design of velocity-tuned filters in the Fourier domain. The Fourier transform of a translating 2D pattern is

$$\hat{I}(k,\omega) = \hat{I}_0(k).\delta(\omega+\vec{d}^T k) \tag{2.20}$$

where $\hat{I}_0(k)$ is the Fourier transform of I(r,0), $\delta(k)$ is a Dirac delta function, ω denotes temporal frequency and $k=(k_x,k_y)$ denotes spatial frequency. This shows that all nonzero power associated with a translating 2D pattern lies on a plane through the origin in frequency space [13].

The method which is developed by Heeger [19, 20] is an illustrative example of the energy-based techniques. Heeger extracts the local energy by using Gabor energy filters, with 12 filters at each of several spatial scales, tuned to different spatial orientations and different temporal frequencies.

2.2.4. Phase-Based Techniques

Phase-Based techniques extract the velocity information from the phase behavior of band-pass filter outputs.

The generalized use of phase information for optical flow was first developed by Fleet and Jepson. [21, 22] The method defines the component velocity in terms of instantaneous motion normal to level phase contours in the output of band-pass velocity-tuned filters. Band-pass filters are used to decompose the input signal according to scale, speed and orientation. Each filter output is complex-valued and may be written as:

$$R(\vec{r},t) = \rho(\vec{r},t).\exp[i.\varphi(\vec{r},t)] \tag{2.21}$$

where $\rho(\vec{r},t)$ and $\varphi(\vec{r},t)$ are the amplitude and phase parts of R.

The use of phase is motivated by their claim that the phase component of band-pass filter outputs is more stable than the amplitude component when small deviations from image translations that regularly occur in 3D scenes are considered. [23]

2.3. True Motion Estimation

In the field of video processing, determining true motion is very important. Motion based Frame Rate Conversion, De-Interlacing and Noise Reduction algorithms may be listed as the most famous and commonly used ones of video processing operations. Most of the consumer electronic products, such as digital TVs, 3G phones, DVD players...etc., utilizes at least one of these listed processing operations. However, a conflict arises at that point: although video processing operations require high quality (true) motion information, consumer electronic devices generally have very limited computational power and then

running complex embedding complex true motion estimators inside them is not possible for most of the time. So, developing low complexity and also high quality estimators is very important for the field of consumer electronics.

When available motion estimators are examined, which are given in the previous sections of this chapter, it is seen that the optical flow based methods generally perform better than the block matching techniques in terms of obtaining true motion. However, because of their complexity they are not popular in the field of consumer electronics and existing true motion related approaches generally focus on block matching techniques. These approaches try to obtain the true motion of real world by making simple modifications on the classical block matching algorithms.

Before starting to deal with the true motion estimation algorithms, it is necessary to understand the differences between a standard motion estimator and a true motion estimator. By being aware of these differences any motion estimation algorithm might be modified in order to obtain a high quality true motion estimator. Some of the issues those are important for the true motion estimation could be listed as:

- Classical similarity metrics, such as SAD and MSE, are not a good choice for true motion estimation case. Global minimum which is obtained by these metrics does not correspond to the true motion vector for all the times. Then, it is logical to modify or completely replace these metrics.

- In the case of true motion, displacements of neighbor pixels or blocks have high correlation and using this information is very critical to solve many problems.

- For block based operations, although it is assumed that pixels within a block undergo uniform motion sometimes this is not the actual case. Then, it is necessary to divide these blocks into smaller sub-blocks and assign correct vectors to each of them.

- Detecting problematic situations, such as appearances and occlusions, is very important for true motion estimators. Then, a detection mechanism for understanding these cases is a necessity.

If the true-motion related publications are examined, then it is seen that all of them focus one or more of the above items. Few illustrative examples of true-motion related publications will be given in the following paragraphs.

Probably, the most famous representative of true motion estimation methods is the one that is proposed by Gerard De Haan, and it is known as the 3D Recursive Search (3DRS) Block Matching Algorithm. [24] This estimator focuses on the correlation of motion vectors of neighbor blocks and algorithm utilizes a very carefully selected candidate motion vector set to find the true motion of current block.

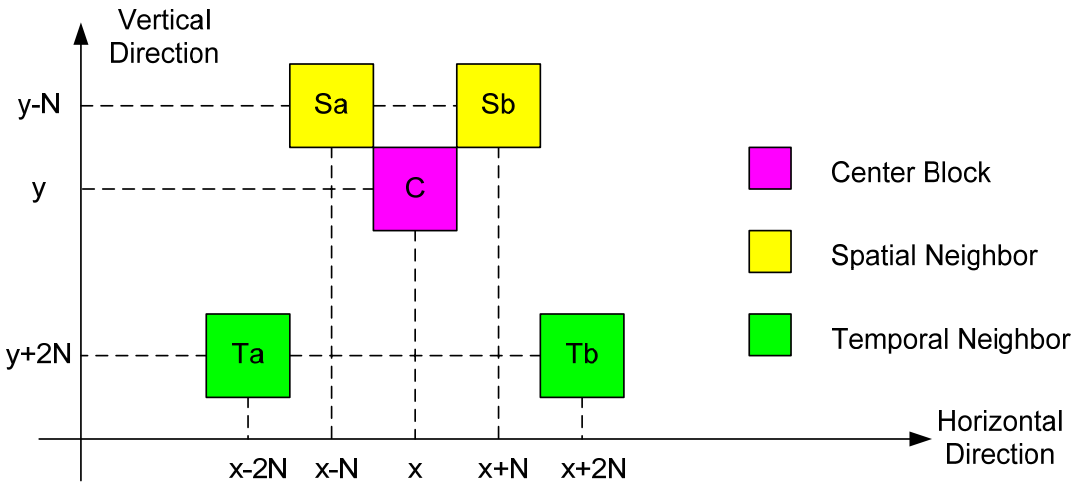

Figure 2.9. Spatial and temporal neighbors for 3D Recursive Search

The algorithm selects the candidate set from spatial and temporal neighbors, and also there exist an update set for permitting small deviations from the original candidate set and obtaining better results. Utilized candidate set is represented in the following form:

$$CS(\vec{r},t)=\left\{\vec{0},\vec{C}_1,\vec{C}_2,\vec{C}_3,\vec{C}_4,\vec{C}_5\right\} \tag{2.22}$$

where

$$
\begin{aligned}
\vec{0} &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\
\vec{C}_1 &= \vec{d}\left(\vec{r} - \begin{pmatrix} N \\ N \end{pmatrix}, t \right) \\
\vec{C}_2 &= \vec{d}\left(\vec{r} - \begin{pmatrix} -N \\ N \end{pmatrix}, t \right) \\
\vec{C}_3 &= \left[\vec{d}\left(\vec{r} - \begin{pmatrix} N \\ N \end{pmatrix}, t \right) + \vec{U}(\vec{r},t) \right] \vee \left[\vec{d}\left(\vec{r} - \begin{pmatrix} -N \\ N \end{pmatrix}, t \right) + \vec{U}(\vec{r},t) \right] \\
\vec{C}_4 &= \vec{d}\left(\vec{r} + \begin{pmatrix} 2N \\ 2N \end{pmatrix}, t\text{-}1 \right) \\
\vec{C}_5 &= \vec{d}\left(\vec{r} + \begin{pmatrix} -2N \\ 2N \end{pmatrix}, t\text{-}1 \right)
\end{aligned}
\tag{2.23}
$$

The random update vector (U(r,t)), which is used for obtaining candidate C_3, is selected from the set:

$$
\vec{U}S = \left\{ \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 2 \end{bmatrix}, \begin{bmatrix} 0 \\ -2 \end{bmatrix}, \begin{bmatrix} 3 \\ 0 \end{bmatrix}, \begin{bmatrix} -3 \\ 0 \end{bmatrix} \right\} \tag{2.24}
$$

By making a selection among above candidates, this estimator guarantees a strong correlation between motion vectors of neighbor blocks. In addition to that, a post processing operation (dividing each block into four sub-blocks and applying median filter to determine the motion vector of each sub-block) is performed on the motion field to remove blocking artifacts.

De Haan et al. published two more papers for enhancing the performance of this estimator. In [25], in addition to the existing motion vector candidates, a new motion vector candidate is added to the candidate set which is shown in equations 2.22 and 2.23. This new candidate is obtained from a parametric motion model, and it is seen that this candidate enhances the performance of algorithm in the case of simple camera motions

such as panning and zooming. Additionally, in publication [26], De Haan proposes to obtain extra motion vector candidates from the correspondences of some special features between two consecutive frames.

3D-RS algorithm gives successful results in terms of true motion, and also its computational complexity is very low when it is compared with the other existing approaches. However, the main drawback of the algorithm is its recursive nature. It converges to the true motion after few frames from the initialization moment since it begins to search from (0, 0) point and convergence is achieved by a recursive manner with the help of update set of equation 2.24.

Another publication on true motion is [27]. This paper suggests some strategies for obtaining the true motion again based on the assumption of correlation of neighbor blocks' motion vectors. One of these strategies is shifting the search window center to a position which is defined by the motion vector of neighbor block that is selected by using the well known SAD metric. Publication [27] also proposes a modification on the metric that is used during the block matching operations. New metric contains two components:

$$C(\vec{d}) = SAD(\vec{d}) + k.N^2.f\left(\min\left\|\vec{d} - \vec{D}_i\right\|\right) \tag{2.25}$$

where $SAD(\vec{d})$ is the commonly used "Sum of Absolute Differences" metric, k is an adjustable parameter to change the weight of second term, N^2 is the dimension of a NxN block, and $\vec{D}_i$ is the motion vector of any neighbor block. While the first component (this is SAD part) takes into account the similarity of luminance values, the second component of right side of the equation stands for measuring the correlation of candidate vector with the motion vectors of neighbor blocks.

Some researchers focus on situations at which assumption of uniform motion within a block fails. Solution to that problem is very important for the true motion problem. In [28], existing approaches those are proposed for dealing with failure of uniform motion within a block are listed, and then authors propose a motion vector smoothing algorithm to deal with blocks with non-uniform motion characteristics. The proposed algorithm collects

multiple motion vector candidates during the initial search process, and then motion vector field is smoothed for problematic blocks by using these candidates.

In [29], for achieving the true motion, an approach for segmenting the problematic blocks was proposed. Authors state that a motion discontinuity is likely to give rise to a valley in intensity map oriented in the direction of the discontinuity and such a valley can be located using the bank of directional filters. And, after locating the valley, most likely segmentation is defined with the help of probabilistic tools.

Neighborhood relaxation approach [30], which is proposed by Chen, is another true motion estimator. This approach again focuses on the metric that is used during the search operations and a cost function which is shown in the following form is proposed:

$$C(B,\vec{d}) = DFD(B,\vec{d}) + \sum_{N_i} W(B,N_i).\min(DFD(N_i,\vec{d}+\delta)) \qquad (2.26)$$

where B is the current block, $\vec{d}$ is the candidate motion vector that is evaluated currently, N_i is any neighbor block and DFD is a metric like SAD or MSE. The basic distinction between this cost function and the one that is proposed in [27] is the "δ" term that exists within this equation. With the help of δ term, this approach also considers the small deviations from the motion vectors of neighbor blocks. On the other hand, although this approach gives better results, it adds an important computational complexity to the algorithm.

3. POST-PROCESSING OF MOTION VECTOR FIELDS

In this chapter, the most commonly used post-processing strategies, which are utilized for increasing the reliability of incoming motion vector field, are discussed.

3.1. Introduction

In order to save transmission bandwidth in data transmission data compression methods and sub-sampling of original data are very commonly used strategies. If the considered video data is in a compressed form, then the transmitted bit stream contains the information of motion vectors and their corresponding prediction error values instead of the original frame itself. Since the transmitted information is usually corrupted by the channel errors, using these erroneous vectors at the receiver side directly and uncompressing the data by using them may result in annoying artifacts. Moreover, although the channel errors are in acceptable levels and the decompressed video has satisfactory quality, direct utilization of the motion information in FRC and DI applications may also create visual artifacts. Therefore, in order to increase the reliability of the motion vectors and obtaining a sequence with a visual quality as high as possible it is necessary to apply a post-processing operation on the received motion vector field or to perform a new search in order to obtain desired vectors. However, devices, which are used at the receiver side, generally have very limited computational power and instead of applying simpler post-processing operations performing a new motion search at that side becomes too costly.

As it is stated in the previous paragraph, re-use of the transmitted motion vector data instead of performing a new motion search is more intelligent and cost effective for most of the consumer electronics applications. Also, by using a correct post-processing strategy it is very probable to obtain very similar performance levels with true-motion estimators. Existing post-processing strategies generally focus on two points:

- Smoothing the vector field and eliminating erroneous vectors
- Refining some vectors and obtaining a denser vector field

Post-Processing techniques can be classified into two groups according to the amount of information that is used by the algorithm:

- MV Neighborhood Based Processing Methods
- Side Information Based Processing Methods

The first group of algorithms focuses only on the MVF information in order to obtain the final result. On the other hand the second group also takes into account the additional information that is obtained from other available resources. The detailed information about these two distinct strategies is given in the following sections of this chapter.

3.2. MV Neighborhood Based Processing Methods

PP methods which belong to this group only deals with the motion vectors themselves, and they do not utilize any additional information such as corresponding SAD values of vectors or intensity\color values of related blocks. Because of their simplicity, they are preferred for very low cost and low power applications. Simple mean and median filtering based methods and their variations are the well known examples of these PP algorithms.

Few of motion vector neighborhood based PP methods will be investigated in the following sub-sections.

3.2.1. Mean Filters

In the case of Mean Filtering, each member of MVF is updated with the average of predefined set of neighbor motion vectors:

$$\vec{D}_{filtered} = \frac{1}{n}.\sum_{i=1}^{n}\vec{D}_i \qquad (3.1)$$

where n is the number of vectors in input motion vector set which contains the motion vector of the center position ($\vec{D}_C$) and the motion vectors ($\vec{D}_i$) of the selected neighbors.

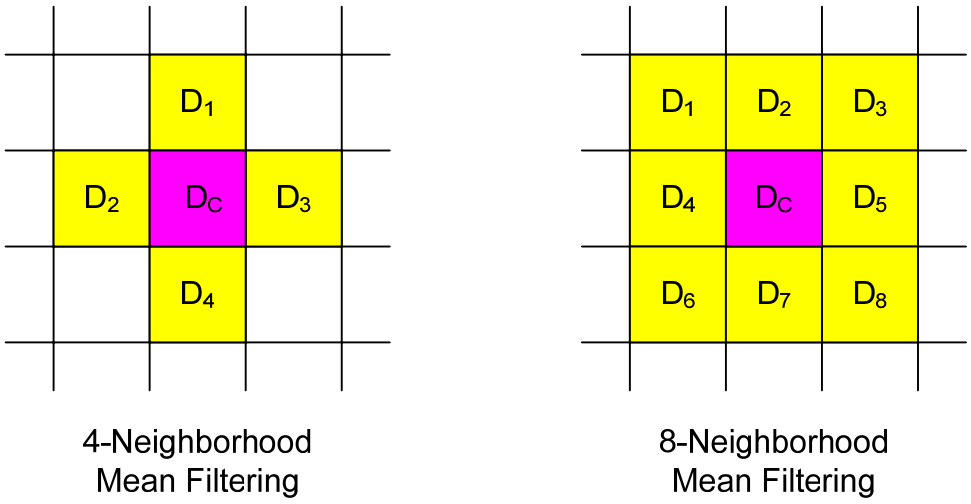

Figure 3.1. Mean Filtering of MVF

Generally, 4-neighborhood or 8-neighborhood of considered MV is used for filtering operations. Mean filters are effective for the removal of uniform (Gaussian) noise. However, they smooth out the important details such as edges and because of this drawback their usage is not common.

3.2.2. Median Filters

Median Filters are non-linear operators and while generating the output they use the order information of input data. Because of their edge-preserving characteristic median filters are very popular and very commonly used in image and video processing applications.

For scalar values, the median of any given input set is found by performing a numerical sorting and then selecting the one which is located in the center of sorted data. However, motion vector processing requires multi-dimensional data processing and the extension of one dimensional sorting operation to multivariate data is not a straightforward job. In literature, there exist few basic strategies to deal with multi-dimensional data sorting and an excellent treatment of these strategies can be found in [31].

Although there is no universally agreeable method for the ordering of multivariate data, three of these strategies are very commonly used in vector median operations:

- The aggregate ordering (A-Ordering) based median filters perform the desired filtering task by calculating the total distance of each sample to the other 'n-1' samples and selecting the one with minimum distance as the output:

$$\vec{D}_{A\text{-}VM} = \underset{D_i \in NB}{\arg\min} \sum_{j=1}^{n-1} \left\| \vec{D}_i - \vec{D}_j \right\| \tag{3.2}$$

where NB is the set of 'n' input vectors and $\|...\|$ is a appropriate norm. This filtering strategy is initially proposed by Astola et al. [32] and they use the Euclidian distance as a selected norm. It is possible to use different norms, such as angular distance or 1-norm. Based on the requirements of application and the available computational power an appropriate norm could be selected.

Table 3.1. Number of elementary operations to evaluate a distance in R^p

Elementary Operations	1-Norm (Absolute Dist.)	2-Norm (EuclidianDist.)	Squared 2-Norm	Angular Distance
Trigonometric Function	---	---	---	1
Square Root	---	1	---	1
Multiplications & Divisions	---	p	p	3p+2
Addition & Substractions	2p	2p	2p	5p
Comparisons & Absolute Values	p	---	---	---

- The marginal ordering (M-Ordering) based median filters perform an ordering along each one of the p-dimensions, and the output of the filter is formed by collecting the medians of each separate channel.

Although they are simpler than A-ordering based vector median filters in terms of computational complexity, since the filter may produce a vector, which does not

exist in input set, as an output, their performance level is below the performance level of A-ordering based median filters.

- The reduced ordering (R-Ordering) based median filters perform the desired filtering task by calculating the distance of each sample to a predefined reference point**,** which may be either the origin or the sample arithmetic mean or the marginal mean.

  The selection of reference point affects the performance of R-ordering based median filters very seriously. For instance, if the selected reference is the mean of input set, then the filter characteristic becomes similar with mean filters. On the other hand, utilization of the marginal ordering output as a reference point results in a median filtering characteristic and since the output vector is also a member of input set it is very probable to have a higher performance level than M-ordering based filters for most of the cases.

For further details on existing vector median filters and some of the proposed simplification strategies see [33].

3.2.3. Other Motion Vector Information Based Filters

Other than well-known mean and median filtering based operations, there exists also some other MVF post-processing methods which use only the local motion vector information.

In [34], Evans proposes to use Vector Area Morphology (VAM) in order to smooth the input MVF. The method is developed from mathematical morphological area openings and uses a vector to scalar transform, in which each vector is replaced by the sum of distances to its connected neighbors, to control the growth of extrema regions. After obtaining the transform, algorithm checks for the regional extremas which give rise to peaks in the distance transform surface. Then, VAM replaces the vectors that constitute regional maxima in transform outputs by the closest vectors from the connected neighborhood of maxima. Finally, after checking if each region is still a maxima or

whether any new maxima have been created, algorithm terminates and smoothed vector field is transferred to the application that will use the motion information.

Another MVF processing method, which uses only the information of local motion vector neighborhood, is the one that is proposed by Westenberg and Thomas [35]. They consider the MVF processing operation as a denoising problem and in order to eliminate noisy motion vectors while preserving the edge structures a wavelet based approach is utilized. As an initial step, the algorithm performs a vector wavelet transform. Then, the detail coefficients are thresholded in wavelet domain and final motion vector information is obtained by inverse vector wavelet transform.

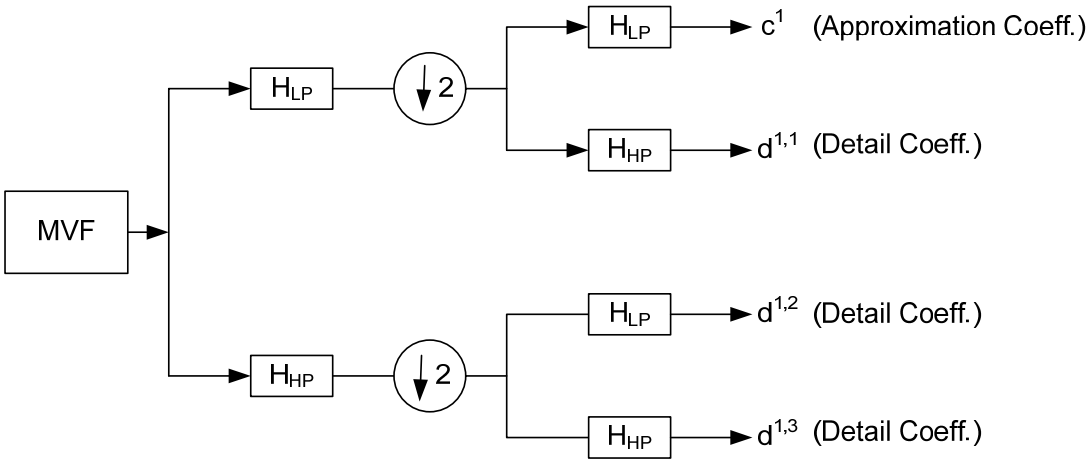

Figure 3.2. Wavelet transform

Utilization of a Heat Flow based approach is another interesting method which is used for smoothing the MVF. In publication [36], Kelly and Hancock propose to use well known heat equation

$$\nabla^2 Q(\vec{r},t) = \frac{\partial^2 Q}{\partial x^2} + \frac{\partial^2 Q}{\partial y^2} = \frac{\partial Q(\vec{r},t)}{\partial t} \tag{3.3}$$

in order to smooth the input vector field. The main idea behind heat flow is diffusing more in smooth areas and less around strong edge structures. The algorithm operates on a scalar field that is obtained from MVF by using a transform, and the smoothed vector field is obtained after it finds a steady-state solution to the equation 3.3 by using an iterative

approach. Although the heat flow approach gives successful results in terms of smoothing, its iterative nature makes it difficult to implement and to use in a real-time system.

In publication [37], Dane and Nguyen propose a vector field processing methodology for FRC applications. They define the vector processing operation as an error minimization task and they expect that the variance of vector differences is decreased after filtering operations.

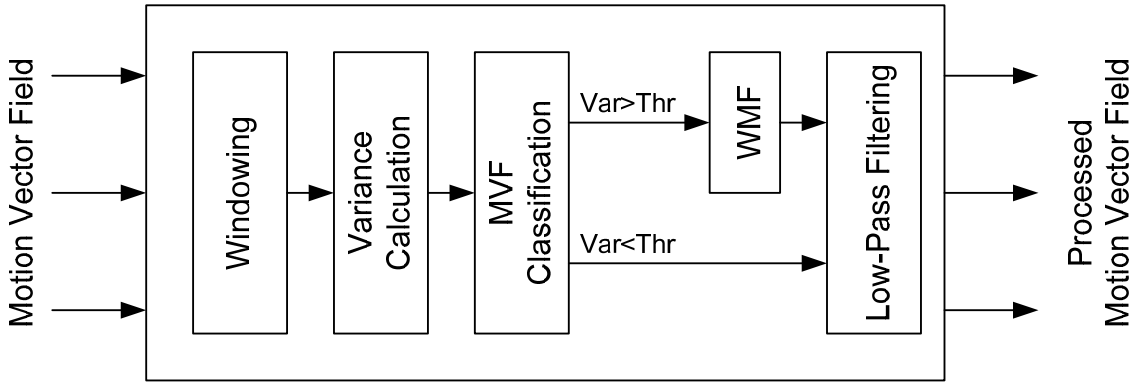

Figure 3.3. MVF processing scheme of Dane and Nguyen

Figure 3.3 shows the detailed flow for the processing scheme that is proposed by Dane and Nguyen. For each filtering window, the first step is calculating the necessary variance value which is obtained from the differences of input vectors. Then, vector median filter (VMF) is applied if the calculated variance value is above the defined threshold. Finally, a low-pass filter is applied to the vector field and the resultant motion vectors are used in a FRC application. Dane and Nguyen argue that the proposed scheme enhances both the PSNR and visual quality of the FRC application.

3.3. Side Information Based Processing Methods

PP methods, which are classified in this group, use additional information during the processing of motion vectors in order to achieve higher performance levels. The additional information is generally extracted from the SAD surface of incoming motion vectors or from the intensity characteristic of related frames. The utilized side information enhances the quality of MVF processing compared to case where only MV information is used, and

the side information based approaches are the most frequently utilized examples of existing PP algorithms.

3.3.1. Alpha-Trimmed Mean Filters

Both of classical mean and median filters have some attractive features and also some drawbacks. Then, combining these filters in order to keep attractive sides of both filters and to eliminate the problematic features is a good idea. Alpha-Trimmed Mean Filters are the result of this combination effort [38].

Alpha-trimmed mean filters utilize the order information of data and apply linear operation to the ranked data.

$$X_{\alpha} = \frac{1}{n\text{-}2.\alpha.n}\left[\sum_{i=[\alpha.n]+1}^{n-[\alpha.n]} X_{(i)} \right] \tag{3.4}$$

where

$$\begin{aligned} &0 \le \alpha \le 0.5 \\ &X_{(1)} \le X_{(2)} \le X_{(3)} \le \ldots \le X_{(n)} \end{aligned} \tag{3.5}$$

In equation 3.4, $X_{(i)}$ represents the sorted elements of current filtering window.

For different values of alpha, the characteristic of the alpha-trimmed mean filter changes. For the extreme cases (i.e.: $\alpha = 0$ and $\alpha = 0.5$), the trimmed mean filter boils down to well known mean or median filter, respectively. In alpha-trimmed filtering, adjusting the value of alpha based on the local characteristics is very critical for obtaining satisfactory results. Otherwise the drawbacks of mean and median filter couldn't be eliminated.

In the case of MVF smoothing, motion vectors are sorted by using one of the strategies that are utilized in vector median filters and based on the calculated value of the

alpha, which is found from the side information, a new vector set is constructed and the final result is obtained from this new vector set. Generally, alpha value is adjusted in such a way that the filter behaves like a median filter for edge regions and behaves like a mean filter for smooth regions.

3.3.2. Adaptively Weighted Median Filters

Because of their ability to remove outliers while preserving edge structures, median filters are very popular in the area of image and video processing applications. However, the main drawback of them is the lack of control on the operations of filter. Then, weighted median filters (WMF) was proposed in order to overcome this drawback.

In publication [39], Barni et al. adopts the WMF, which is designed for the filtering of scalar data, to motion vector processing operations. The vector (X_{VM}) which is selected as the median of the input vector set is the one that satisfies the following condition:

$$\sum_{i=1}^{n} w_i . \left\| X_{VM} - X_i \right\| \leq \sum_{i=1}^{n} w_i . \left\| X_j - X_i \right\| \qquad j=1,2,...,n \tag{3.6}$$

where the utilized weights (wi) are calculated by the following equation:

$$w_i = \frac{DFD\left(\vec{D}_C\right)}{DFD\left(\vec{D}_i\right)} \qquad i=1,2,...,n \tag{3.7}$$

where DFD stands for Displaced Frame Difference, which may be selected as one of the well known cost metrics such as SAD or MSE, $\vec{D}_C$ represents the motion vector of filter's center and $\vec{D}_i$ is any motion vector from the considered filtering window.

Adaptively weighted vector median filtering (WVMF), which is proposed in [39], is very popular in the area of vector field processing because of its success in filtering and also because of its positive contribution to the following video processing applications such as FRC and DI.

In publication [40], Barni et al. proposes an alternative strategy to adjust the weights of median filter for the regularization of optical flow based vector fields. Instead of using DFD, the weights are calculated by using a reliability measure. The strategy is analyzing the local neighborhood and obtaining reliability measures based on the variation of pixel intensity. By this way, vectors which belong to low variance regions are assumed as less reliable ones and filtering is biased to utilize vectors with higher reliability.

Another publication on the utilization of weighted vector median filters for the smoothing of MVFs is [41]. Pereira et al. use the weighted median strategy in a frame interpolation application and they show that the smoothing increases the visual quality of outputs significantly.

3.3.3. Other Side Information Based Methods

The basic and most common side information based vector field processing methods are given in the previous subsections and two additional methods which use different strategies on the area of vector field processing will be given in this subsection.

The algorithm which is developed by the engineers of Sony Advanced Technology Center focuses on the segmentation of frames and then processing of existing MVF by using the obtained segmentation information [42].

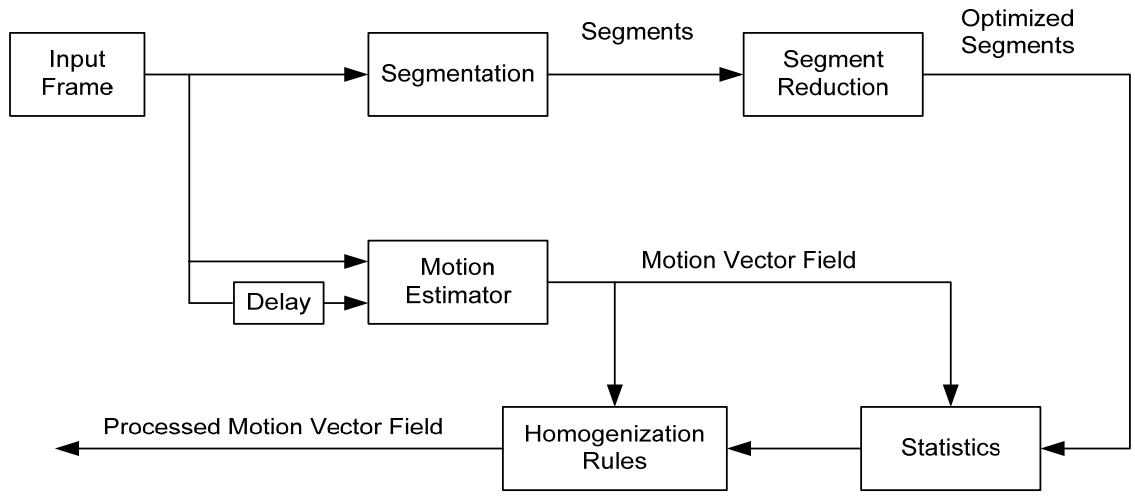

Figure 3.4. MVF processing scheme of Sony Advanced Technology Center

The initial step of the proposed scheme is performing segmentation on related frame and obtaining the necessary information about existing segments and their locations. Then, segment reduction block refines the segmentation by merging small segments with neighboring segments. The need for this step comes from the over-segmentation problem of most segmentation algorithms. After having a refined segmentation, statistics block analyzes the motion vector distribution for each segment. Final block updates the motion vector field segment by segment by using the information of existing vector distributions and externally provided homogenization rules. Homogenization rules organize the final processing. For instance, if a dominant vector exists for current segment then the rules decide for assigning this vector to the whole segment or when there exist multiple dominant vectors the rules decide for how to process the existing vectors of segment by using these dominant vectors set.

Another MVF processing strategy, which is patented by Ching and Hu, focuses on the multiple objects problem [43]. Most of the practical motion estimators are block based strategies and they all assume that the utilized blocks contain pixels from only one object, which means motion field is uniform within block and all pixels have the same motion vector. However, this assumption may fail for edge regions and the algorithm of Ching and Hu proposes a method to obtain a denser MVF for which each block has a uniform motion assumption holds.

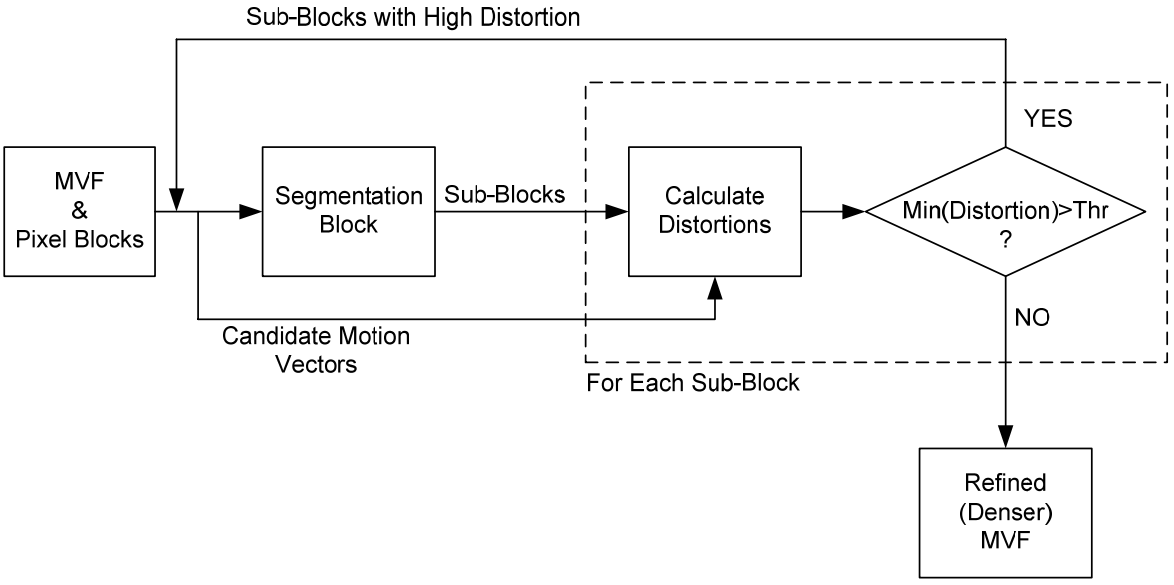

Figure 3.5. Refinement scheme of Ching and Hu

Figure 3.5 shows the flow of considered MVF refinement algorithm. Initially, current frame and related motion information is fed into the *Segmentation Block*. This block divides the incoming block into four identical sub-blocks and assigns the motion vector of original block to each of them. Then, by using the motion vector of the original block and the neighbor motion vectors a candidate set is constructed for each sub-block and corresponding distortion values are calculated for each vector inside the *Calculate Distortions* block. During the calculation of distortions, MSE, SAD or any similar cost metric is used. Finally, minimum distortion values for each sub-block are compared with a predefined threshold and for the ones with small distortions refinement process is terminated. However, sub-blocks with high distortions are again fed into the *Segmentation Block* in order to perform further refinement on them. This refinement scheme increases the PSNR significantly and a better visual quality is obtained especially for edge regions.

4. CONTRIBUTIONS TO VECTOR FIELD POST-PROCESSING METHODS

In this thesis, two contributions have been made to MVF smoothing methods. One of our contributions is an algorithmic improvement over the commonly used Vector Median Filtering (VMF) which eliminates some its well-known shortcomings. The second contribution is a simplification of the commonly used Weighted Vector Median Filtering (WVMF) which is a high quality smoothing strategy with significant computational complexity.

In addition to the proposed MVF smoothing schemes, some effort was spent on the improvement of MVF refinement algorithm's performance which desires to produce denser and more accurate vector fields.

4.1. Motion Discontinuity Based VMF Smoothing

Median Filters (MF), which are discussed in section 3.2.2, are very popular in the area of image and video processing applications because of their edge-preserving filtering characteristic. As is stated in the same section, there exist many different types of median filters and the existing state of the art shows that the Vector Median Filtering (VMF) is the most effective among the family of simple median filters which process multidimensional data.

In the case of the multidimensional data filtering, the classical median filtering strategy fails in some situations. Since it operates on the each dimension of the input data separately, the output of the filter does not exist inside the input set for all the times and this is one of reasons of its failure. Another problematic situation arises especially in the case shown in Figure 4.1. If the majority of the input data belongs to a group which is different than the group of the center position, then median filter fails and performs a wrong assignment. Although the VMF fixes the first problem and it always produces an output which exists in the input set, the second problem is still unfixed. If such problematic situations are observed it is seen that the source of the problem is the existence of multiple

different classes in the input set and also the existence of a conflict between the class of the center position and the class which contains the majority of input set members. One solution strategy for this problem is to introduce a classification on the input data set and to perform the filtering operation on the inputs which belong to the same class with the center location. By this way, uncorrelated vectors are eliminated before the main filtering step and erroneous vector assignments are avoided most of the time.

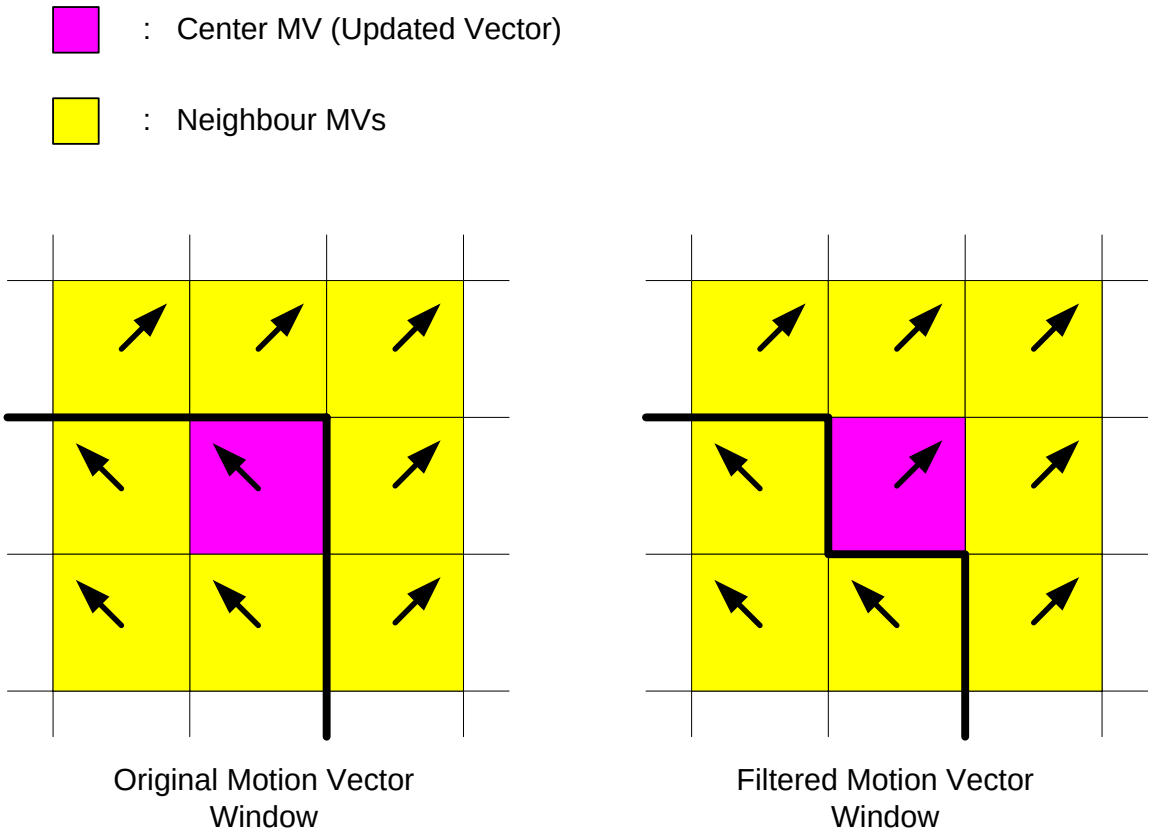

Figure 4.1. Failure case for the Median Filtering

The flow diagram of the proposed smoothing strategy is given in Figure 4.2. It has four processing steps: discontinuity detection, classification, filtering and error check. In the first step, a rough estimate of the motion discontinuities are detected by using the information obtained from the current motion vectors and their corresponding matching function outputs (such as SAD or MSE). In the classification step, an appropriate input set is created for filtering operation by using the output of the discontinuity detector. Then, the classical VMF is performed on the selected vectors. Finally, the matching function values for the original and filtered motion vectors are compared before replacing the original data with the filter output. If the vector which is found as a result of filtering does not produce a

significant increase on the matching function value, the replacement operation is performed. Otherwise, the original vector is preserved.

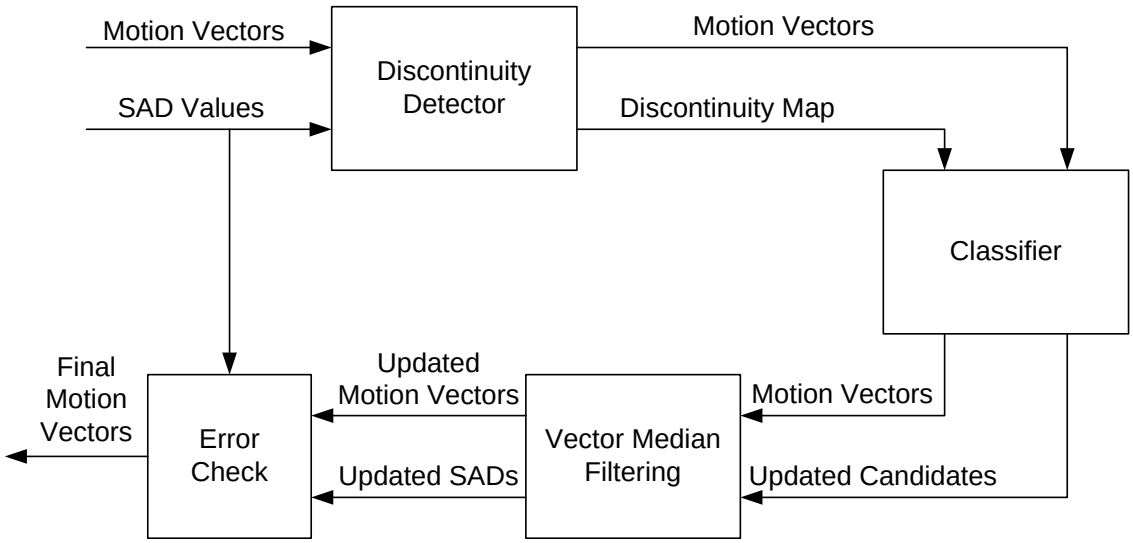

Figure 4.2. Block diagram for Motion Discontinuity Based MVF method

The details of the processing steps of the proposed smoothing algorithm will be investigated in the following sub-sections.

4.1.1. Discontinuity Detection

The main purpose of this block is to find the motion discontinuity regions for the adaptation of the following filtering operations which construct and utilize different input sets to fix the previously discussed problem of classical median filters.

Motion discontinuities are generally observed at the object boundaries and in these regions it is very highly probable to have some search blocks which contain pixel groups with distinct motion characteristics. In such motion discontinuity regions the utilized matching function (for instance SAD, MSE or any similar metric) outputs significantly high cost values. Therefore it is important to take into account the following observations during the detection process:

- Motion discontinuity regions generally produce high matching function values, however all regions with high cost values do not correspond to the motion

discontinuity regions. For instance, occlusion/appearance of objects and deformation of objects also produces high cost values. Then, searching only for the regions with the high cost function values does not produce satisfactory detection.

- Motion discontinuities are generally observed at object boundaries (edge regions), however all edges do not correspond to a motion discontinuity. If two or more objects move together, then boundaries between/among these objects do not have discontinuities. In other words, simply detecting the edges is not enough to correctly detect the discontinuity regions.

Based on the above observations, the first step of the detection process is searching for the regions with high matching function values. These regions are good starting points, but a further constraint is necessary to distinguish them from the other high matching function valued regions. This constraint is defined over the matching function output at pixel level.

In Figure 4.3, an illustrative example of a motion discontinuity region is represented. In the figure, there exits two objects with different motion characteristic and it seen that the utilized search blocks do not fit perfectly with the edge structure between objects. Misclassified pixel groups observed at the discontinuity region are the main cause for the calculated high matching function value. Based on this observation, it is clear that if sub-blocks with smaller sizes are utilized for these problematic regions then some of the sub-blocks will have significantly low cost values since the motion vector of the macro-block perfectly fits for them and the remaining sub-blocks will still have very high cost values.

For high cost valued regions, the utilized motion discontinuity detector checks the matching function's output behavior by using smaller sub-blocks. If the matching cost of the original block is not distributed to the sub-blocks in a fair way and few of the sub-blocks have high cost values while the other ones have significantly low values then this region is marked as discontinuity region. Otherwise, if the cost function score of original block is distributed to the cost scores of the sub-blocks in a fair way or the cost function score is originally low then these regions are marked as uniform motion areas.

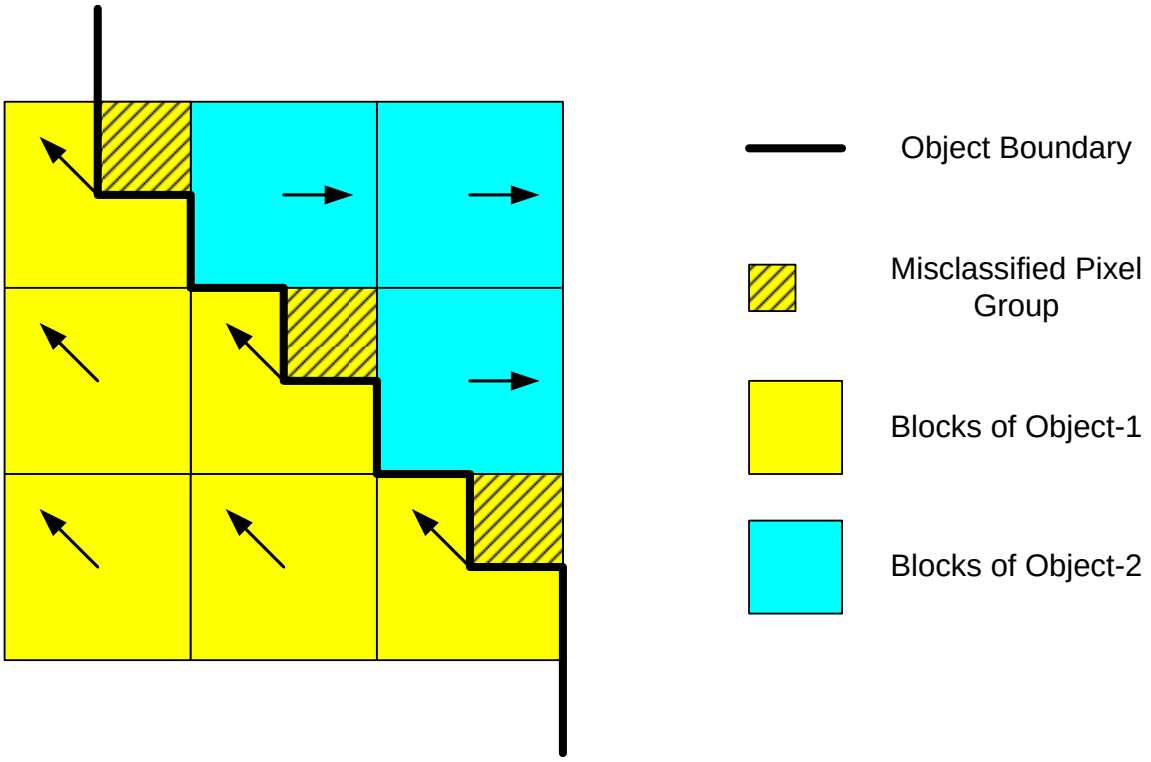

Figure 4.3. An illustrative motion discontinuity region

The main output of the discontinuity detector is a map that represents the detected discontinuities. Otherwise the motion vectors are transferred to the next step without any modification.

4.1.2. Classifier

The classification step decides for the input data set of the next filtering step. First of all, it checks the motion characteristic of the current filtering window from the previously obtained discontinuity map. If the current location has a uniform motion characteristic then the whole 8 neighbors of the center position and the center position itself are added to input set of the filtering block. On other hand, if the considered window has a discontinuous characteristic then a simple global thresholding based segmentation algorithm is applied in order to partition the incoming data into two classes and detect the correlated ones. The details and comparison of the commonly used image segmentation techniques can be found in [44] or in most of the available image/video processing books.

Implemented segmentation algorithm assumes that the considered region has two objects with a distinguishable gray level difference. Although this assumption does not hold all the times and it is possible to find other segmentation algorithms which are capable of handling most of the currently uncovered situations, because of the complexity requirements the most primitive strategy was selected. Luckily, this simple segmentation strategy works well as a result of local processing in a small neighborhood and the segmentation method provides useful information for most of the time.

After deciding on whether the segmentation is necessary or not and detecting the correlated elements for creating an appropriate input set, classifier block transmits this set to the next block in order to perform straightforward filtering operation.

4.1.3. Vector Median Filtering

Inside this block, the well known VMF, which is discussed in section 3.2.2, is applied on the incoming vector set. Implemented method uses Euclidian distance as a norm and the input vector which has the minimum norm score is selected as the output of the filter. The details of VMF can be found in [32].

The output of the filtering block is the candidate vector for the current filtering location.

4.1.4. Error Check

This block acts as a fusing system and it checks the quality of the filter output before replacing the original motion vector with the filtered one. If the quality of the vector which is found as a result of filtering operations is not above a certain threshold then the original vector is not replaced with the new one.

The reliability of the new motion vector is found by comparing the matching function outputs (SAD, MSE or any similar metric) of the original and new vectors:

$$R\left(MV_{filtered}\right)=\begin{cases}1 & \text{if } (f(MV_{filtered}) - f(MV_{org})) < THR \\ 0 & \text{if } (f(MV_{filtered}) - f(MV_{org})) \geq THR\end{cases} \quad (4.1)$$

where '$R(MV_{filtered})$' stands for the hard decision whether the new vector has satisfactory level or not. Also, 'f(..)' is a cost metric which is used for comparing the reference and the candidate blocks and 'THR' is an experimentally obtained threshold value to determine the range of the tolerated quality level changes. It is necessary to adopt the threshold value depending on the size of the selected motion search blocks. An appropriate value of the threshold parameter for 8x8 search blocks was found as 100 after an exhaustive amount of experiments. The appropriate threshold value for other block sizes could be derived from this value. For instance, for a block size of 16x16 the threshold value is found by the following simple formula:

$$THR_{16x16} = \left(\frac{16}{8}\right)^2 \text{ x } THR_{8x8} = 400 \quad (4.2)$$

4.2. Low Complexity WVMF Smoothing

Contrary to their many desirable properties median filters also have some drawbacks. The lack of a parameter to have control on the operation of the filter is one of these drawbacks. In order to overcome this difficulty the weighted filtering strategies were proposed.

Depending on the type of the considered application and its special requirements there exist many different weighting strategies in the literature. In the area of the motion vector processing, among the existing weighted median filtering strategies the one that is proposed by Barni et al. in [39], also discussed in section 3.3.2, is the most important one.

Compared with the alternative strategies, WVMF has a better performance in terms of the vector field smoothness and the closeness of the output vectors to the ground-truth motion fields. However, the main drawback of this algorithm is its significant computational complexity. As a result of the huge computational burden, the realization of

this strategy inside the low cost processors of the consumer electronic devices is not possible for most of the time.

We aim to preserve the current quality level of WVMF while decreasing the computational complexity significantly. In order to achieve this target we investigate each step of the original weighted median filtering method and propose some simplifications and modifications.

The block diagram of our low complexity WVMF proposal is given in Figure 4.4. Similar to the algorithm in the previous section this one has also four main processing steps: discontinuity detector, candidate selector, weighted vector median filter and error check.

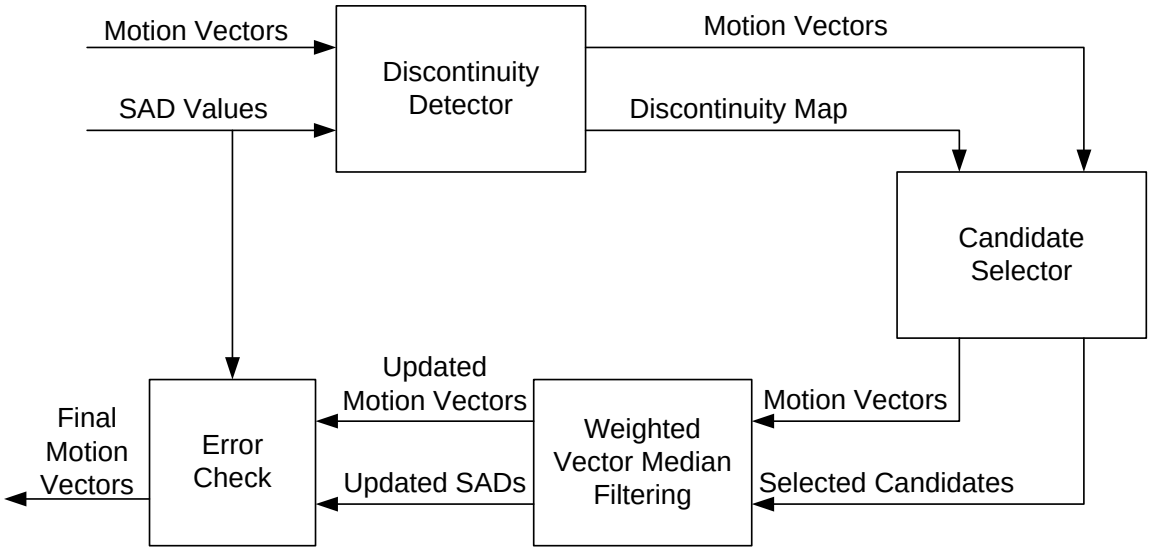

Figure 4.4. Low Complexity WVMF

In order to simplify the original WVMF, one possible strategy is to decrease the number of candidate vectors which will be utilized during the filtering part. However, it is seen that minimizing the number of the existing input vectors using a fixed strategy results in a quality loss. Then, it is necessary to have an intelligent strategy to adaptively select the best input vectors for the filtering step.

Our low complexity smoothing algorithm uses the information of the motion discontinuities in order to decide for the best vectors which will be used in the main filtering step.

The details for the processing steps of the simplified smoothing algorithm will be given in the following sub-sections.

4.2.1. Discontinuity Detector

The discontinuity detection block which is used as the first block of current simplified WVMF is same with the block that is discussed in section 4.1.1. The detector analyzes the current motion vectors and their corresponding matching values and decides for whether the current region has a motion discontinuity or not.

4.2.2. Candidate Selector

This block decides for the best input vector set for the original weighted filtering step by using different strategies according to the decisions of previous discontinuity detector block.

The original weighted median filter combines two types of information during the filtering:

- Similarity information that is obtained from the calculated distance of the each vector to the other vectors

- Reliability information that is extracted from the matching function score (by moving the center location with the considered motion vector)

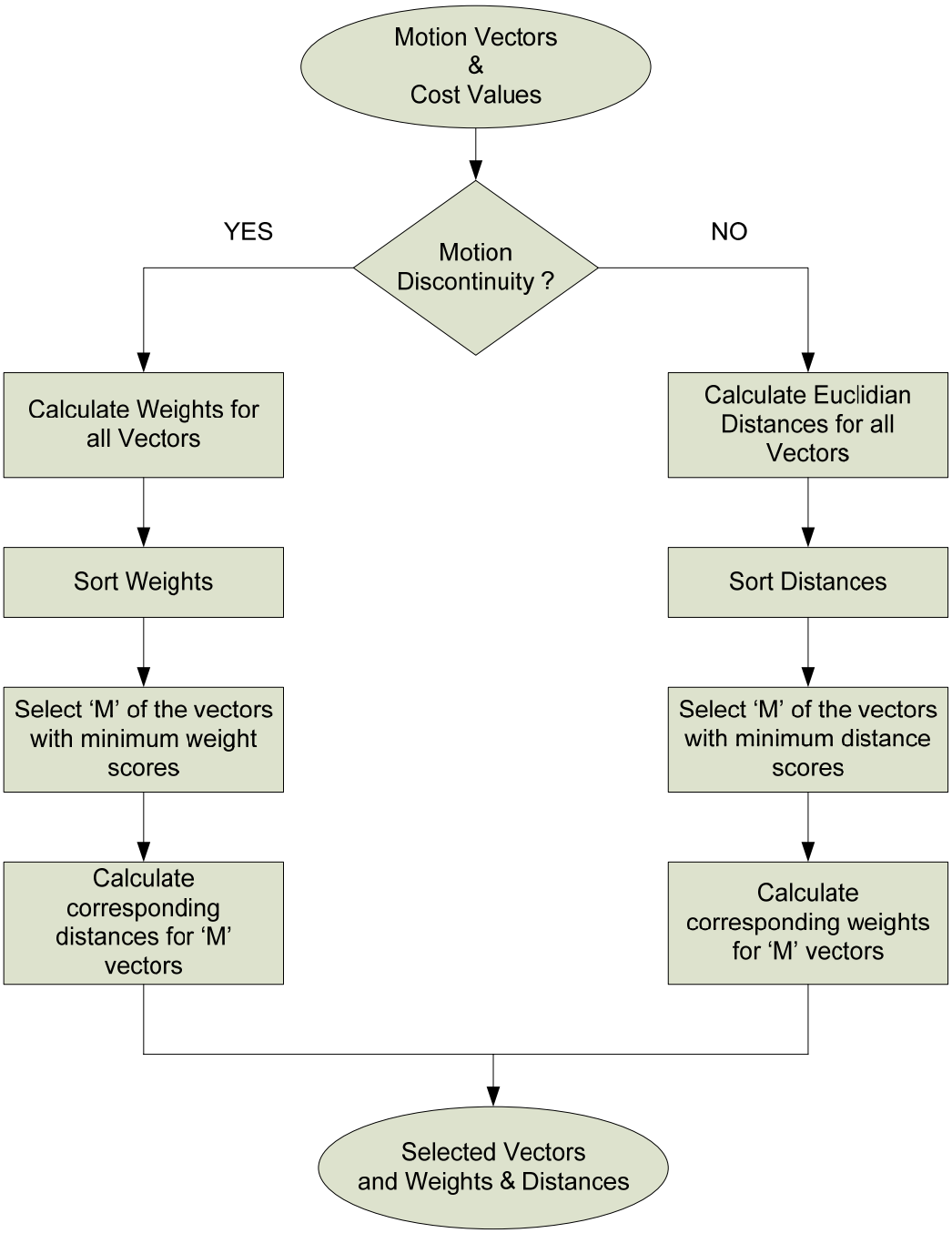

Figure 4.5. Flow chart for the candidate selection of Low Complexity WVMF

The similarity metric produces a bias to select the most frequently occurring vector. However it couldn't distinguish whether the most frequently appearing vector really fits with the center position. On the other hand, reliability metric produces a bias to select the vector which fits best with the center position while it is not aware of the occurrence frequency of the vectors. When these two are combined together, a really high quality vector field processing method is obtained.

Actually, the similarity and the reliability metrics, which are discussed in the previous paragraphs, are used for handling two distinct situations:

- For motion discontinuity regions, the primary criteria for the selection of vectors is their fitting scores for the center location while the occurrence frequencies of the vectors do not have much importance

- For uniform motion regions, it is natural to have very similar fitting scores for all vectors while the occurrence frequencies of the vectors have the primary importance in order to have a field as smooth as possible

Based on the above observations, the candidate selection step focuses on the weight calculation operations for motion discontinuity regions and it focuses on the distance calculation operations for uniform motion regions as it is seen in the Figure 4.5.

The flow of the selection step is as follows:

- If the motion characteristic of the current region is uniform then the important metric is the distance measure and the proposed algorithm calculates distance scores for all existing vectors, otherwise the important metric is the weight score and the algorithm calculates the corresponding weight scores for all of the existing motion vectors.

- Then, the calculated scores are sorted and the vectors which have the ‘M’ smallest scores are selected as the ones that will be utilized in the following steps of the filtering operations.

- Finally, the corresponding weight scores are calculated for selected vectors if the previously utilized metric is distance and the corresponding distance scores are calculated for the selected vectors if the previously utilized metric is weight.

The number of the selected vectors (M) after the sorting operation defines the exact computational complexity level of the proposed algorithm. The simulation results show that the utilization of "M=2" still gives very satisfactory results.

Table 4.1. Calculations count for the processing of the motion vectors with filtering window size of "3x3" and search block size of "NxN"

Elementary Operations	Original WVMF		Proposed WVMF			
			(Uniform Motion)		(Motion Discont.)	
	Distance Calc.	Weight Calc.	Distance Calc.	Weight Calc.	Distance Calc.	Weight Calc.
Addition & Substractions	36x4	$16xN^2$	36x4	$4xN^2$	15x4	$16xN^2$
Multiplications & Divisions	36x2	8	36x2	2	15x2	8
Square Root	36x1	---	36x1	---	15x1	---
Abs. Values & Comparisons	---	$8xN^2$	---	$2xN^2$	---	$8xN^2$

The list of the necessary computations for the original WVMF and also for the proposed algorithm is given in Table 4.1.

4.2.3. Weighted Vector Median Filtering

Inside this block, the well known WVMF, which is discussed in section 3.3.2, is applied on the incoming vector set. Implemented method combines the previously obtained weight and distance metrics for the selected candidates and the filtering output is selected as the one that minimizes the combined cost value. The details of WVMF can be found in [39].

The output of the filtering block is the final candidate vector for the current filtering location.

4.2.4. Error Check

This final block is used for checking the quality of the vector, which is created as a result of filtering operations, before replacing the original motion vector with it. The operations which are performed inside this block are identical with the ones that are performed in the last step of previously proposed method and the details about the operations can be found in section 4.1.4.

4.3. Refinement Related Contributions

The size of the utilized search blocks is very critical for the performance of the motion estimation/compensation related applications. The existing work of the researchers shows that the utilization of the bigger search blocks gives better results in terms of obtaining the true motion vector fields. Unfortunately, the block based motion estimators assume that the motion characteristic is uniform inside the each block and the probability for the failure of this assumption increases significantly with the increasing block size. Then, utilizing big search blocks during the motion estimation process and refining them by an appropriate strategy is a commonly used method.

MVF refinement methods especially focus on the regions with high matching cost scores and they try to diminish the existing high values by using smaller sub-blocks and the motion information of neighbor blocks. So, refinement algorithms produce denser MVFs with a better motion information accuracy especially at the motion discontinuity regions.

The method which is proposed in [43] and the method which is used in [24] are the two illustrative examples of the existing refinement strategies. The details for the first method are given in section 3.3.3. It operates on the smaller size sub-blocks and uses the motion information of both center position and its neighbors. During the refinement operations the original blocks with high matching scores are divided into the sub-blocks until the desired cost function score or the predefined iteration number is reached.

The other refinement scheme which is used in the publication [24] is very primitive when it is compared with the one that is discussed in the previous paragraph. De Haan utilizes a simple median filter to find the vector of the each sub-block and the set of input vectors for each sub-block is constructed in a different manner.

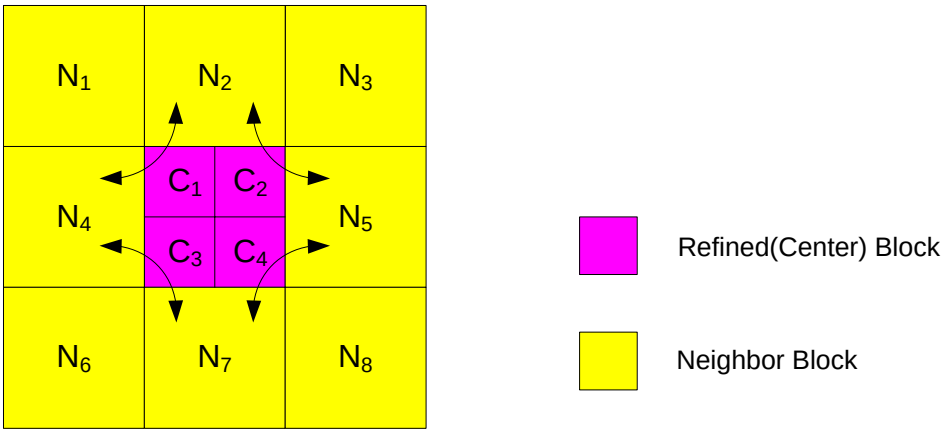

Figure 4.6. Utilized neighborhood for De Haan's refinement scheme

The previously discussed refinement method (given in [43]) uses the motion vector information of all the eight neighbors (N_1,…,N_8). On the other hand the second method utilizes the motion information of a limited set. Figure 4.6 illustrates the utilized neighborhood for each sub-block (C_1, C_2, C_3 and C_4). The utilized neighbors can also be given in the following way:

$$\begin{aligned} CS(C_1) &= \left\{\vec{D}_C, \vec{D}_{N_2}, \vec{D}_{N_4}\right\} \\ CS(C_2) &= \left\{\vec{D}_C, \vec{D}_{N_2}, \vec{D}_{N_5}\right\} \\ CS(C_3) &= \left\{\vec{D}_C, \vec{D}_{N_4}, \vec{D}_{N_7}\right\} \\ CS(C_4) &= \left\{\vec{D}_C, \vec{D}_{N_5}, \vec{D}_{N_7}\right\} \end{aligned} \tag{4.3}$$

where $CS(C_n)$ represents the motion vector set which will be used as candidate vectors for sub-block C_n, $\vec{D}_C$ is the motion vector of the center block and $\vec{D}_{N_i}$ is the corresponding motion vector of the neighbor block N_i.

Although the first method generally works better its computational complexity is a little bit high. On the other hand, although the second method is simple in terms of computational complexity it has the drawbacks of classical median filtering and fails in some cases. Then, a third refinement strategy is created by combining the former two strategies. The new method uses the candidate set selection part of the second refinement scheme and then the most appropriate one for the considered location is defined by using the resultant matching function scores of them as it is performed in the first refinement scheme.

The simulation results show that the third refinement method, which is created from the combination of the previously discussed methods, generally outperforms the other two methods in terms of the visual quality and also its complexity is not as high as the first method.

5. APPLICATIONS AND RESULTS

This section of the thesis has three subsections. The first subsection contains the details of the utilized performance measurement methods. The next subsection focuses on the performance of the most commonly used vector field processing filters. Finally the third subsection gives the details of the implemented FRC application (short information about choosing/entering the desired settings and running the FRC application is given in Appendix A) and shows the effects of the MVF post-processing strategies on the visual performance of the implemented application.

5.1. Performance Measurement Methods

Since the ground-truth MVF is unknown and, even if it is known, the closeness to the reference does not guarantee to have the best visual quality, the objective evaluation of video processing methods is not an easy task. However, by combining the outputs of several appropriate methods it is possible to obtain a rough estimate about the visual quality.

The methods which are used for measuring the performance of considered vector field processing algorithms and also for evaluating the effect of them on a FRC application are given in the following sections.

5.1.1. Mean Square Error

Mean Square Error (MSE) is one of the most commonly used metrics for measuring the similarity of the considered data with the reference one.

$$\mathrm{MSE}(\vec{X}) = \frac{1}{M}\sum_{i=1}^{M}\left|\vec{X}_{ref}(i) - \vec{X}(i)\right|^2 \qquad (4.1)$$

where X and X_{ref} are evaluated and reference variables with M elements, respectively. It is important to note two important points: this metric is applicable only if the reference

(ground-truth) data is available and the smallest MSE score does not guarantee to have the best visual quality.

5.1.2. Vector Field Smoothness Indicator

As it is discussed in section 2.3, true motion vector fields have a very smooth nature and most of the existing MVF processing methods aim to create such a smooth field without creating disturbing artifacts especially at the motion discontinuity regions.

In [45], De Haan proposes the following smoothness criteria:

$$S(\vec{X}) = \sum_{\vec{X}} \sum_{m=-1}^{1} \sum_{n=-1}^{1} \left(\frac{8M_b}{\left|\Delta_x\left(\vec{X},m,n\right)\right| + \left|\Delta_y\left(\vec{X},m,n\right)\right|} \right) \tag{4.2}$$

where $\vec{X}$ runs through all the vectors of the evaluated MVF, M_b is the total number of the vectors which are stored inside the vector field, and:

$$\begin{aligned} \Delta_x\left(\vec{X},m,n\right) &= D_x(\vec{X}) - D_x\left(\vec{X}+\begin{bmatrix} m \\ n \end{bmatrix}\right) \\ \Delta_y\left(\vec{X},m,n\right) &= D_y(\vec{X}) - D_y\left(\vec{X}+\begin{bmatrix} m \\ n \end{bmatrix}\right) \end{aligned} \tag{4.3}$$

It should be emphasized that it is not a good idea to use this performance metric independently from other performance indicators, such as MSE, since it is not possible to give the optimum smoothness figure. However, the higher smoothness score becomes determinative if the two methods give comparable MSE results.

5.1.3. Structural Similarity Index

Structural Similarity (SSIM) index [46], which is one of the most famous visual quality assessment metrics, is proposed by Wang and Bovik based on the assumption that:

"*The human visual system is highly adapted to extract structural information from the viewing field.*"

The proposed metric, which is called as Structural Similarity Metric (SSIM) and given in Figure 5.1, tries to measure the structural information change to have a good approximation of perceived image distortion. As it is seen from the figure, new strategy combines the information of luminance, contrast and structure for producing the final decision instead of only utilizing luminance information.

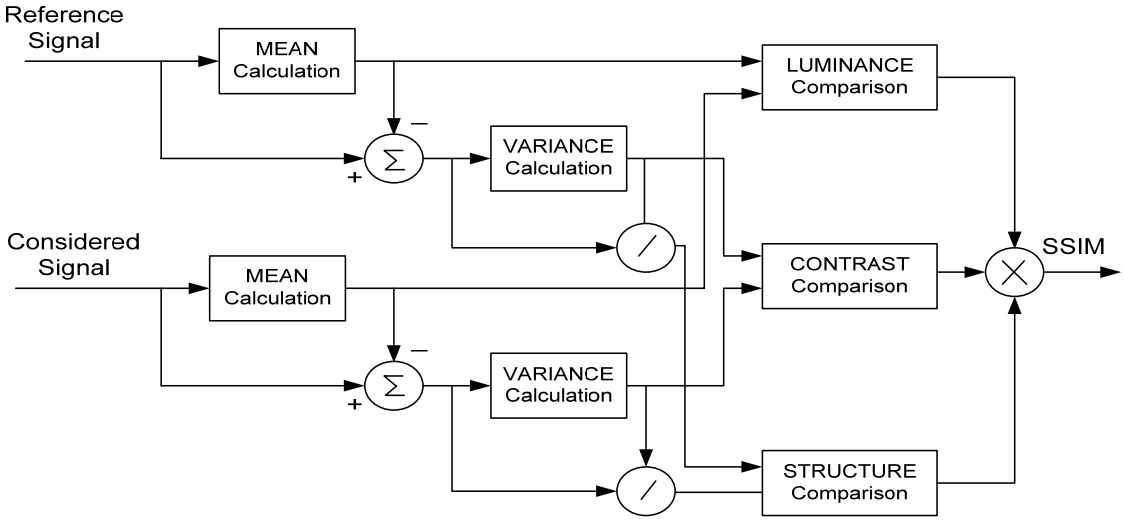

Figure 5.1. Flow for Structural Similarity metric

The *Mean Calculation* block estimates the mean intensity value of incoming discrete signal in the following way:

$$\mu_x = \frac{1}{M}\sum_{i=1}^{M} x_i \qquad (4.4)$$

where M is the total number of elements of the input signal x. After finding the mean values for both of the reference signal (x) and the considered signal (y) they are fed into the *Luminance Comparison* block as inputs and the luminance similarity metric is calculated by using the formula:

$$l(x,y) = \frac{2.\mu_x.\mu_y + C_1}{\mu_x^2 + \mu_y^2 + C_1} \tag{4.5}$$

Before performing contrast related operations, it is necessary to remove the mean intensity from the signal by subtracting the mean value from each component of the incoming signal. Then, the *Variance Calculation* block calculates the standard deviation of the new signal as an indicator of contrast:

$$\sigma_x = \left(\frac{1}{M-1} \sum_{i=1}^{M} \left(x_i - \mu_x \right)^2 \right)^{\frac{1}{2}} \tag{4.6}$$

Then, the *Contrast Comparison* block calculates the contrast similarity metric by using the formula:

$$c(x,y) = \frac{2.\sigma_x.\sigma_y + C_2}{\sigma_x^2 + \sigma_y^2 + C_2} \tag{4.7}$$

The *Structure Comparison* block calculates the structural similarity of the given two input signals. Input signals are normalized by dividing them with their own standard deviation values before performing structural similarity related operations. Then, the correlation (inner product) of these normalized signals is found by:

$$\sigma_{xy} = \frac{1}{M-1} \sum_{i=1}^{M} \left(x_i - \mu_x \right).\left(y_i - \mu_y \right) \tag{4.8}$$

After finding the correlation, it is use for generating the final structure similarity metric:

$$s(x,y) = \frac{\sigma_{xy} + C_3}{\sigma_x.\sigma_y + C_3} \tag{4.9}$$

The final stage for the SSIM metric is the combination of previously obtained luminance, contrast and structure information:

$$SSIM(x,y) = \left[l(x,y)^{\alpha}.c(x,y)^{\beta}.s(x,y)^{\gamma} \right] \quad (4.10)$$

where α, β and γ are positive values that are used for adjusting the relative importance of luminance, contrast and structure components.

5.2. MVF Smoothing Methods

The smoothing methods aim to obtain a vector field as smooth as possible while preserving the motion discontinuities. The following subsections represent the effects of the most commonly used smoothing algorithms on the motion vector fields of certain contents.

For testing the performances of the selected smoothing methods, vector fields of three synthetically created contents (which will be called as Image Sequence I , II and III) are estimated by using the Hexagonal Motion Search strategy given in section 2.1.3 and these fields are fed into the considered methods in order to obtain the filtered fields. Then, the filtered vector fields are compared with the ground-truth data by using the MSE and they are also compared with the input vector fields in order to observe the behavior of the smoothness score.

The utilized synthetic contents (IS-I, IS-II and IS-III) and their corresponding ground-truth motion vector fields are given in Appendix B.

5.2.1. Mean Filtering Strategy

It is known that mean filtering is a very simple and also an effective way for dealing with Gaussian type distortions. However, this strategy does not work in non-Gaussian distortions and also it is not successful in preserving discontinuities.

If the MSE and Smoothness scores for the mean filtered vector fields are compared with the scores of the original vector fields (look at Table 5.1), an improvement is observed for both cases.

Table 5.1. Smoothness and MSE scores for Mean Filtering

	Image Sequence - I		**Image Sequence - II**		**Image Sequence - III**	
	Original MVF	Filtered MVF	Original MVF	Filtered MVF	Original MVF	Filtered MVF
MSE Score	0.9526	0.6601	29.5482	15.9282	2.1719	0.8755
Smoothness Score	1.3246	2.3426	0.1645	0.5876	0.8254	1.7059

If the outputs of the mean filter are compared with the corresponding ground-truth vector fields (look at figures 5.2, 5.4 and 5.6), it is seen that the filter performance is very poor especially for motion discontinuity regions because of the lack of edge-preserving capability. Additionally, if the filtered vectors are compared with the original ones (look at figures 5.3, 5.5 and 5.7), it is seen that the errors are propagated to the neighbors in the case of impulsive errors.

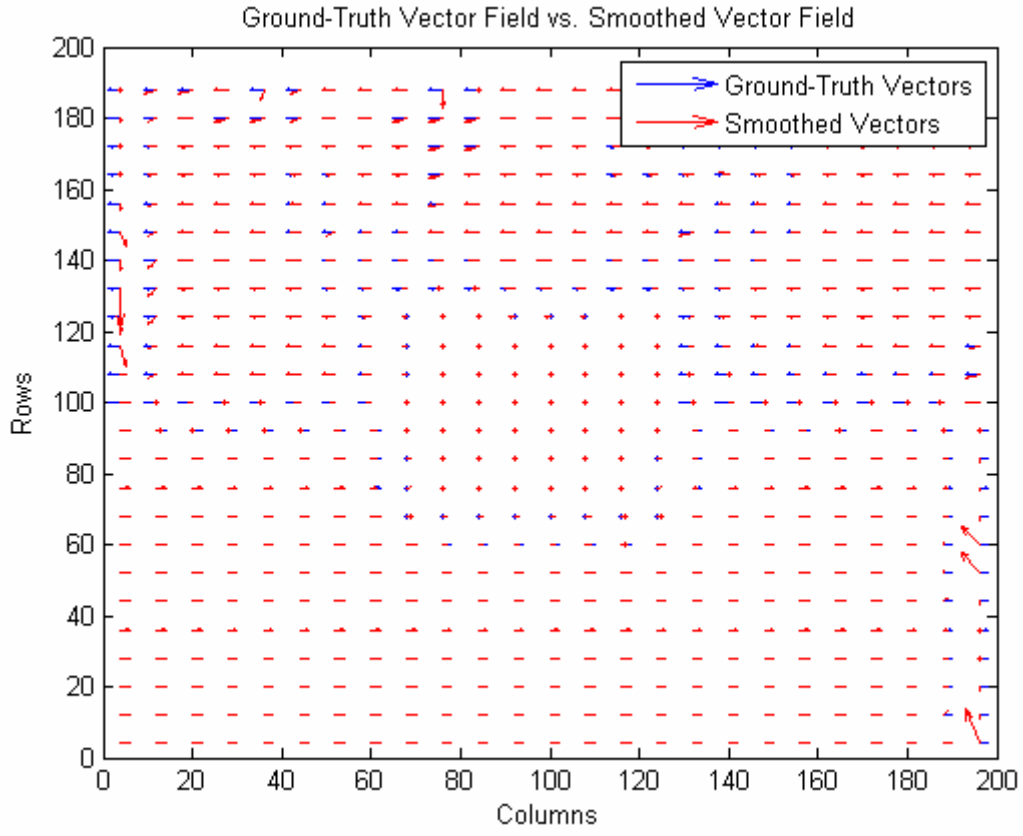

Figure 5.2. Ground Truth vs. Smoothed MVF for Mean Filtering of IS-I

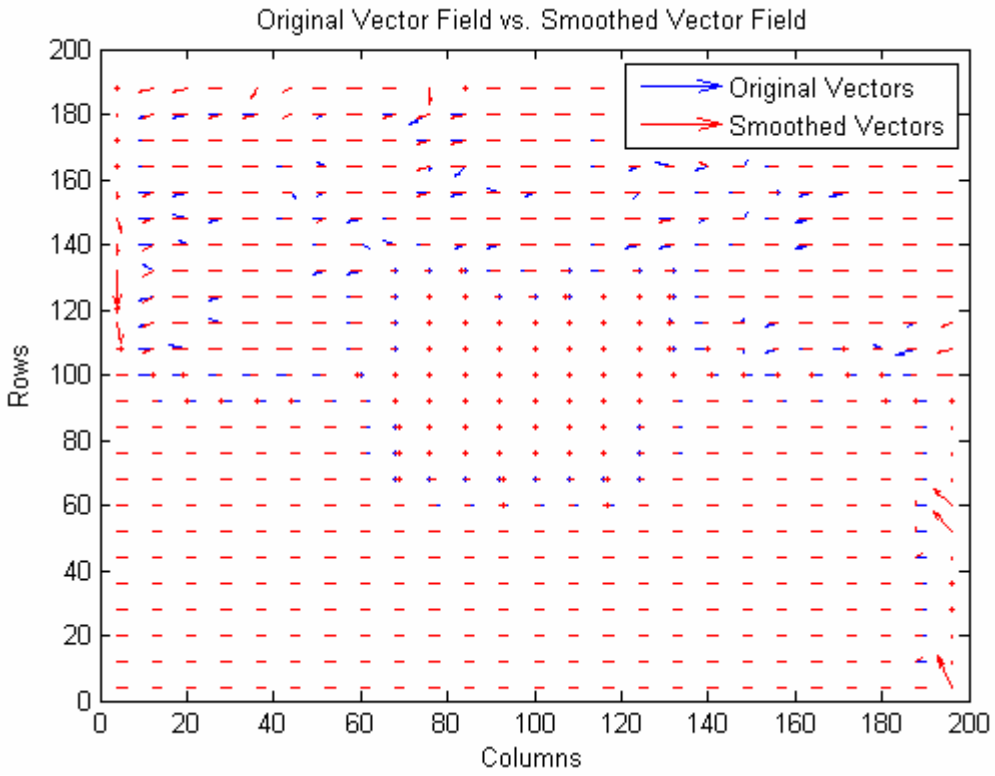

Figure 5.3. Original vs. Smoothed MVF for Mean Filtering of IS-I

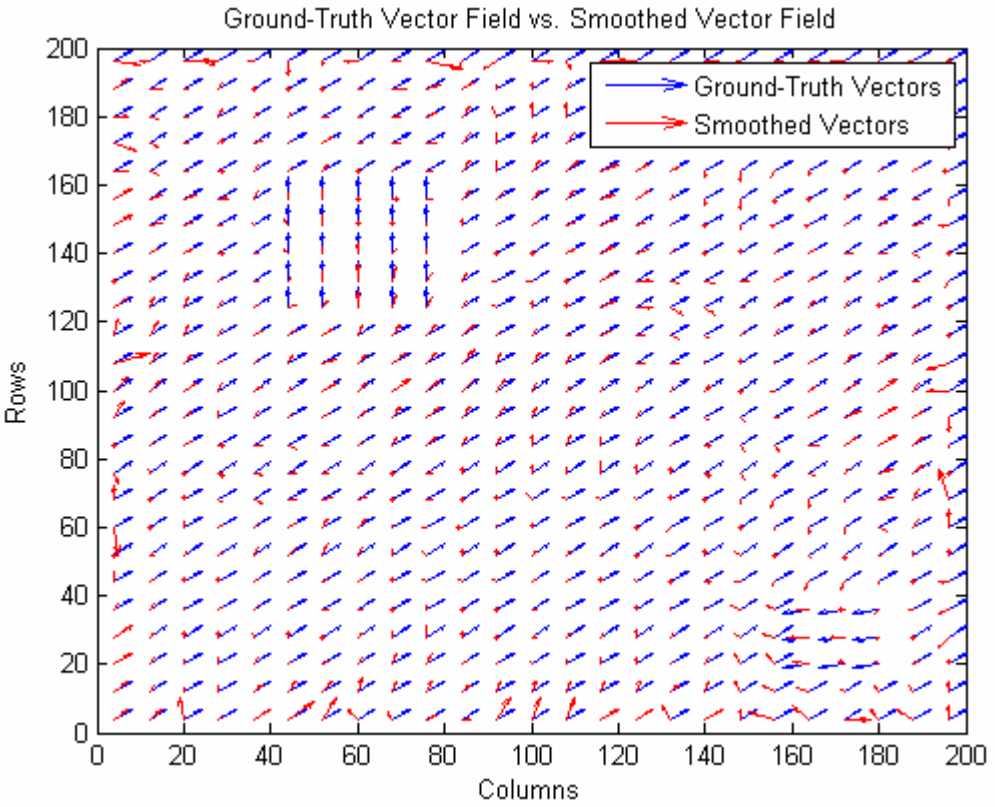

Figure 5.4. Ground Truth vs. Smoothed MVF for Mean Filtering of IS–II

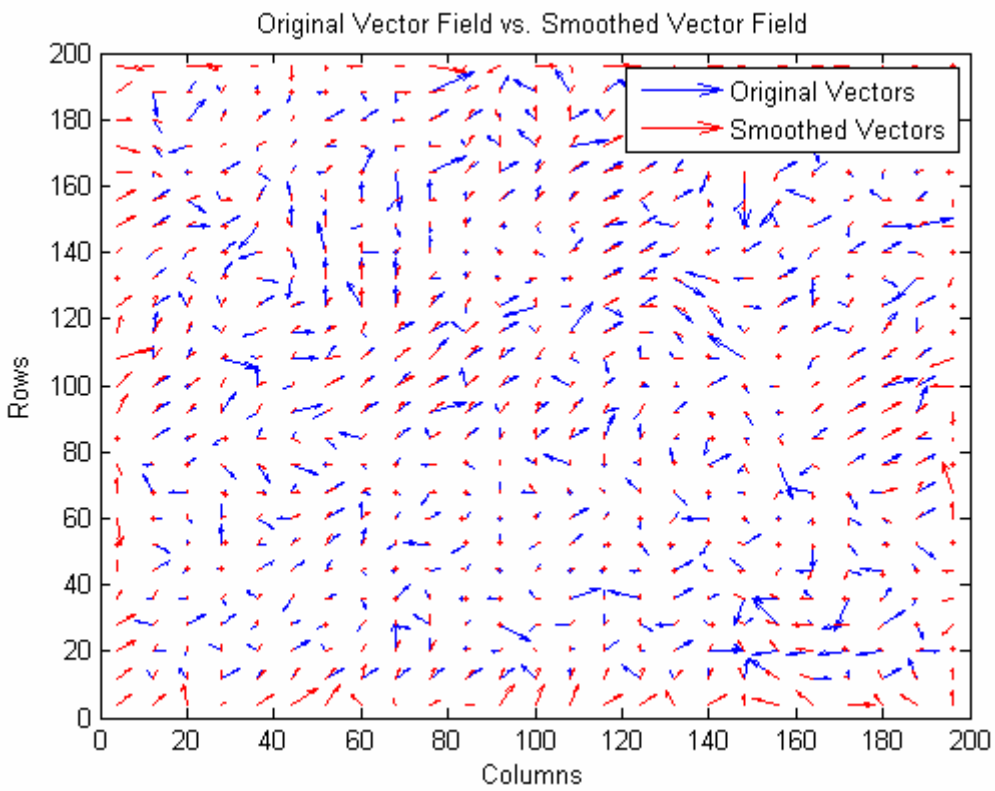

Figure 5.5. Original vs. Smoothed MVF for Mean Filtering of IS–II

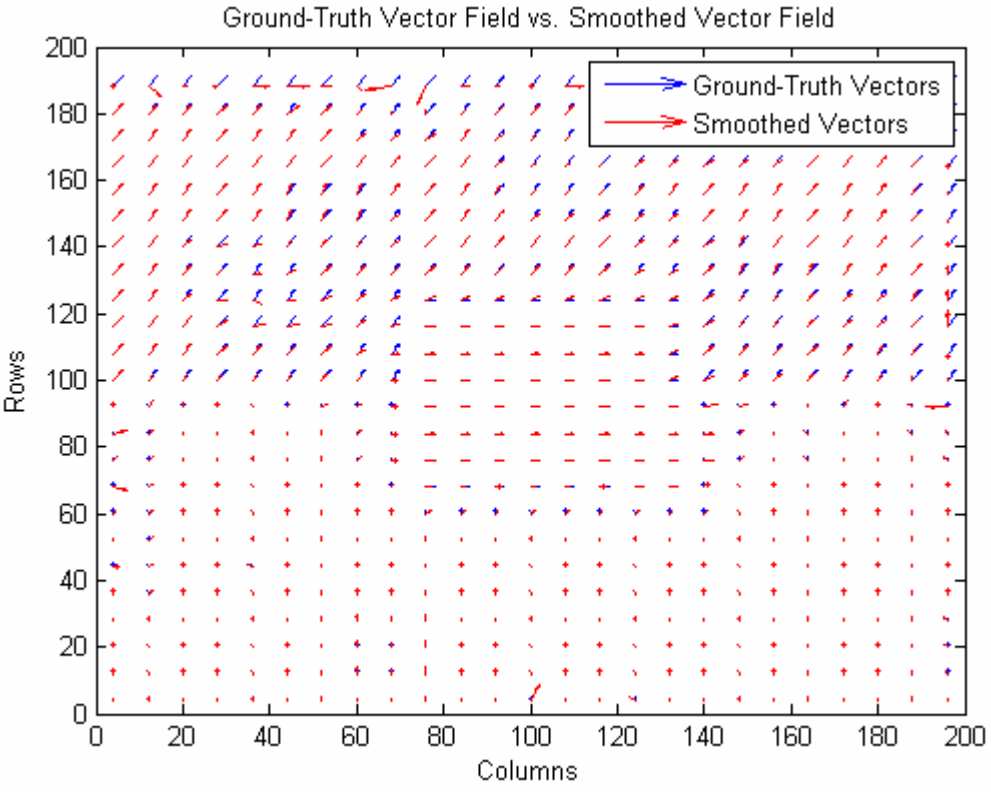

Figure 5.6. Ground Truth vs. Smoothed MVF for Mean Filtering of IS–III

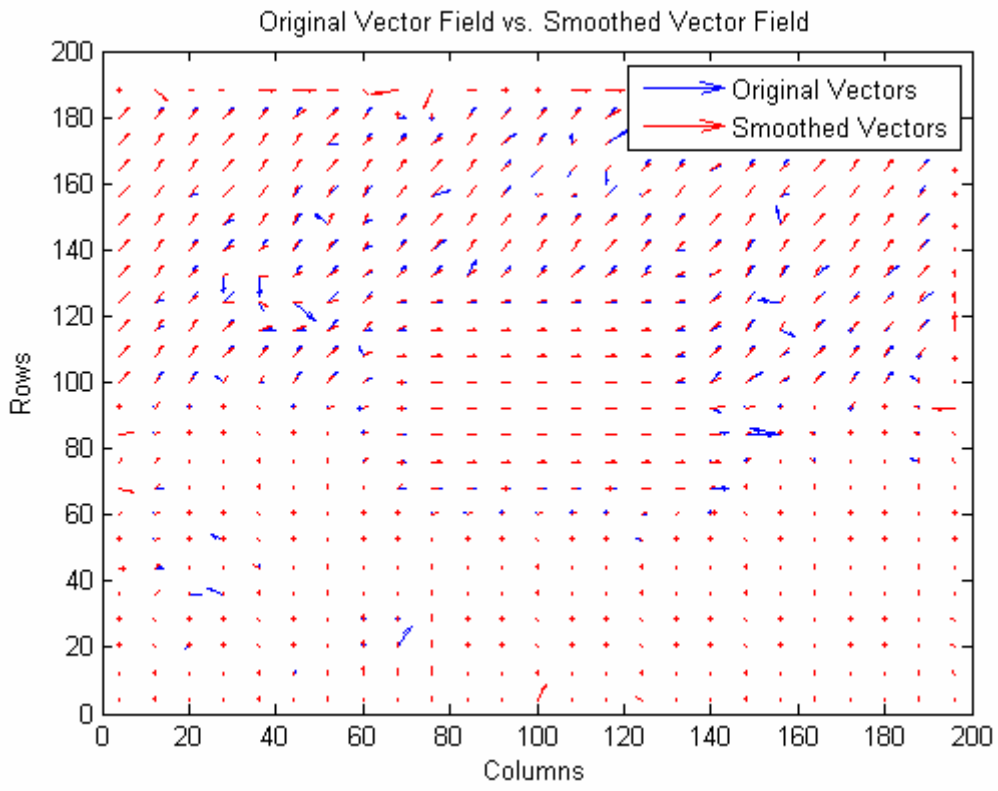

Figure 5.7. Original vs. Smoothed MVF for Mean Filtering of IS–III

5.2.2. Median Filtering Strategy

Median filters are very effective in terms of the removal of impulsive distortions and their edge preserving characteristic increases their attractiveness when they are compared with mean filters. For filtering multidimensional data, such as motion vectors, each channel of the data is processed separately and then the medians of the each channel are put together to find the final output.

Table 5.2. Smoothness and MSE scores for Median Filtering

	Image Sequence - I		**Image Sequence - II**		**Image Sequence - III**	
	Original MVF	Filtered MVF	Original MVF	Filtered MVF	Original MVF	Filtered MVF
MSE Score	0.9526	0.5672	29.5482	15.1531	2.1719	0.4170
Smoothness Score	1.3246	3.0972	0.1645	0.5218	0.8254	2.2108

Table 5.2 shows the effect of median filtering on the utilized MSE and Smoothness metrics. As seen, the median filtering improves both of them.

When the outputs of the median filter are compared with the corresponding ground-truth vector fields, our observations showed that for most of the cases the filtering operation eliminates the erroneous vectors without propagating any error to the neighbors (look at figures 5.9 and 5.13). However, if the vector field is too much noisy then the filter fails and the filter outputs do not converge to the ground-truth ones for most of the cases as seen in Figure 5.11.

Simulation results showed that median filter has a superior motion discontinuity capability when compared to the mean filters. However, the outputs also showed that the median filter also fails in certain discontinuity regions as seen in figures 5.8 and 5.12. The two main reasons of these failures are the separate processing of each data channel and the lack of a mechanism to measure the reliability of input vectors.

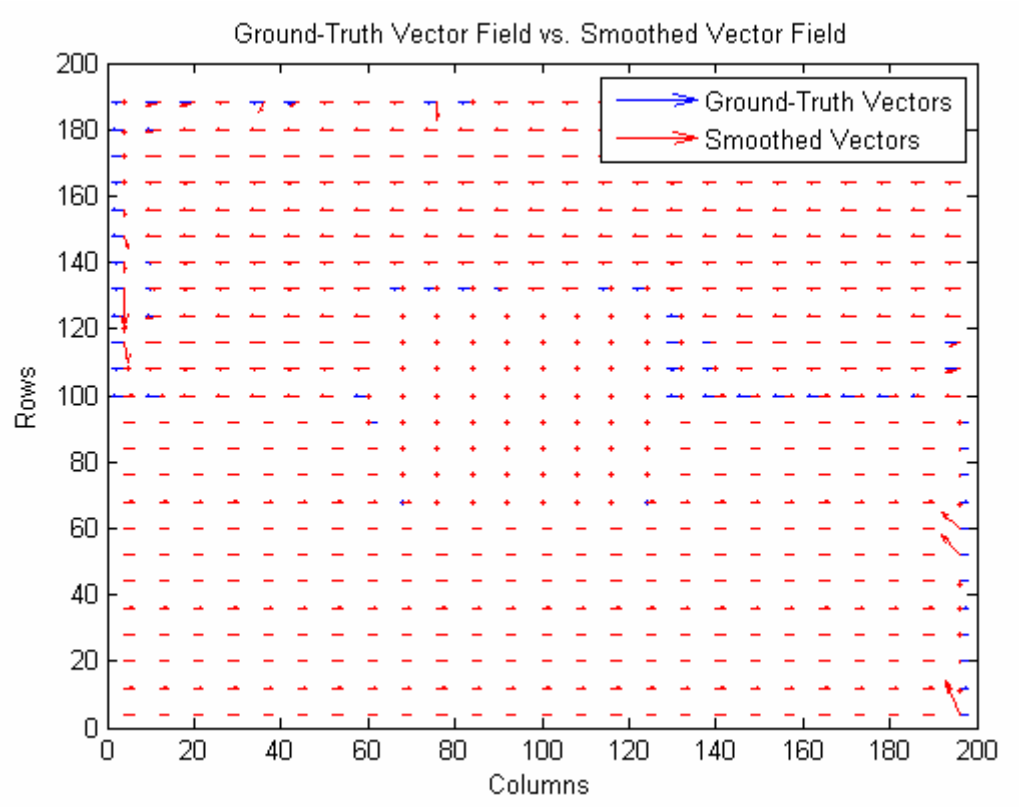

Figure 5.8. Ground Truth vs. Smoothed MVF for Median Filtering of IS–I

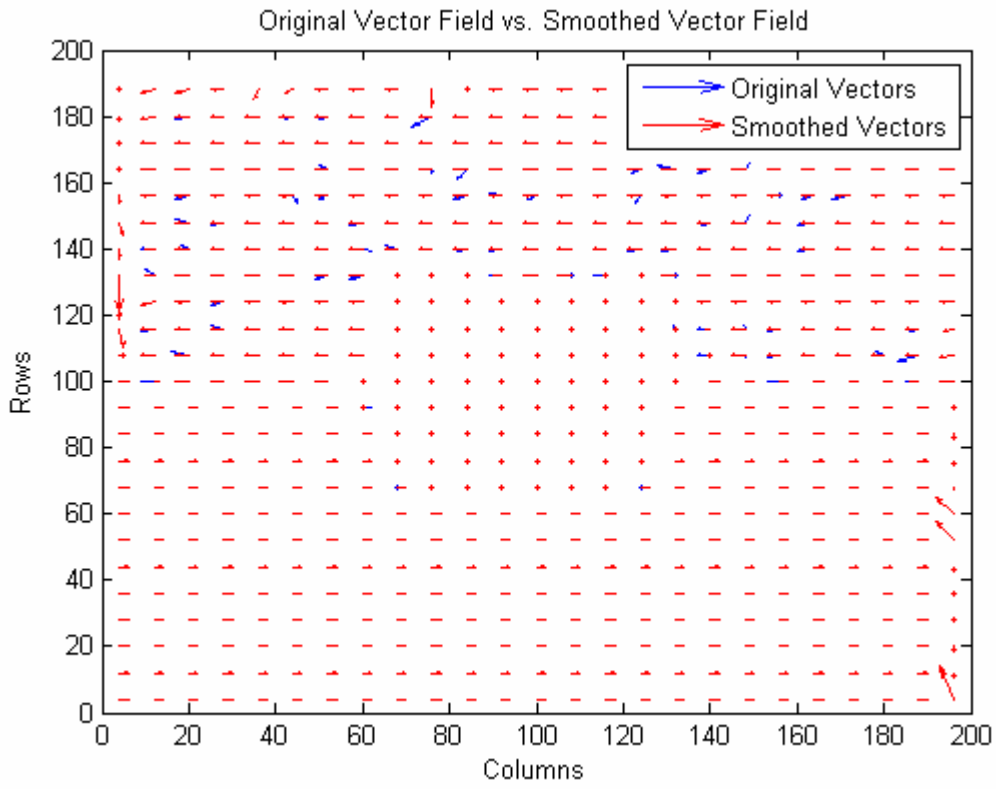

Figure 5.9. Original vs. Smoothed MVF for Median Filtering of IS–I

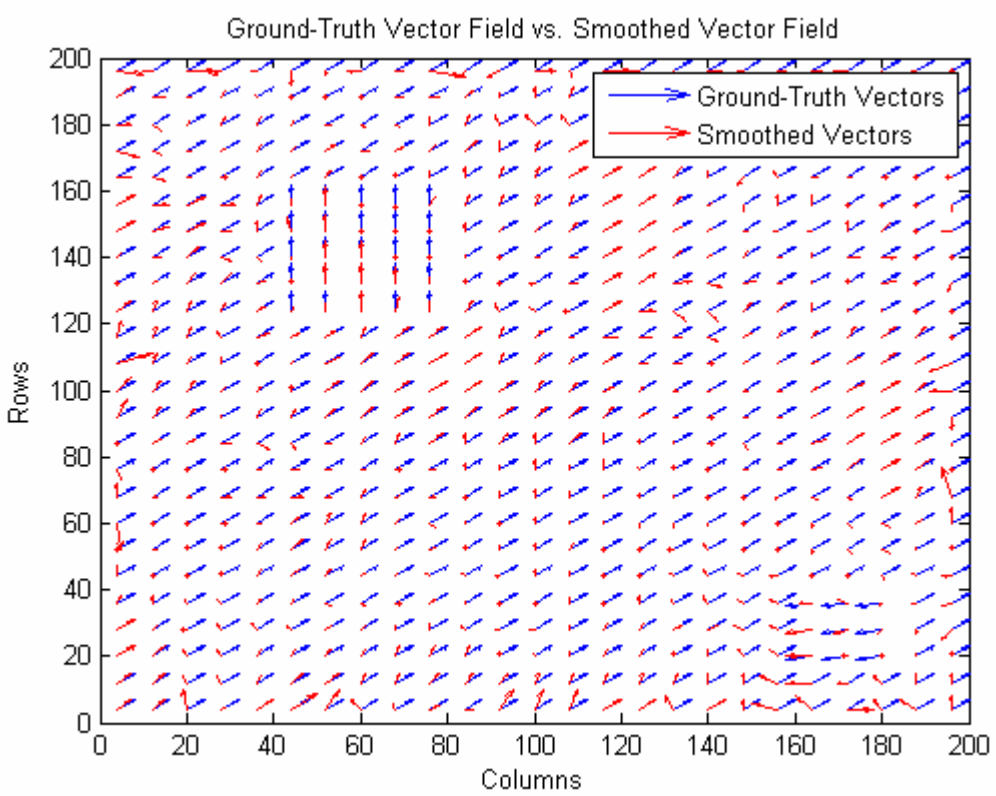

Figure 5.10. Ground Truth vs. Smoothed MVF for Median Filtering of IS–II

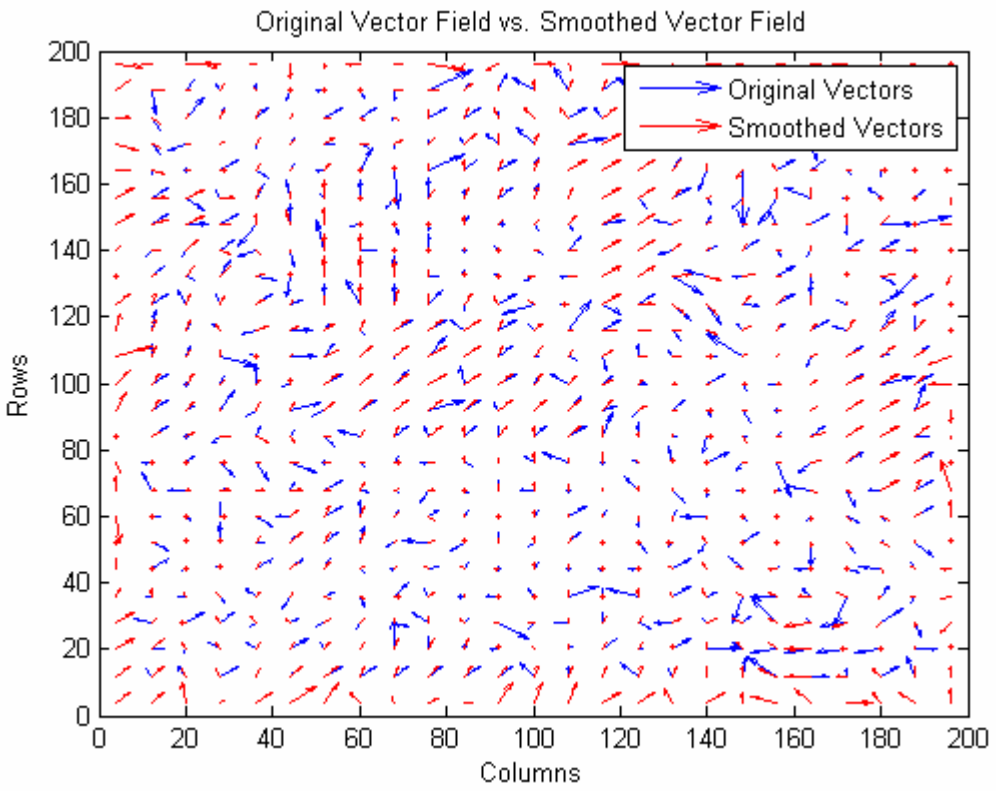

Figure 5.11. Original vs. Smoothed MVF for Median Filtering of IS–II

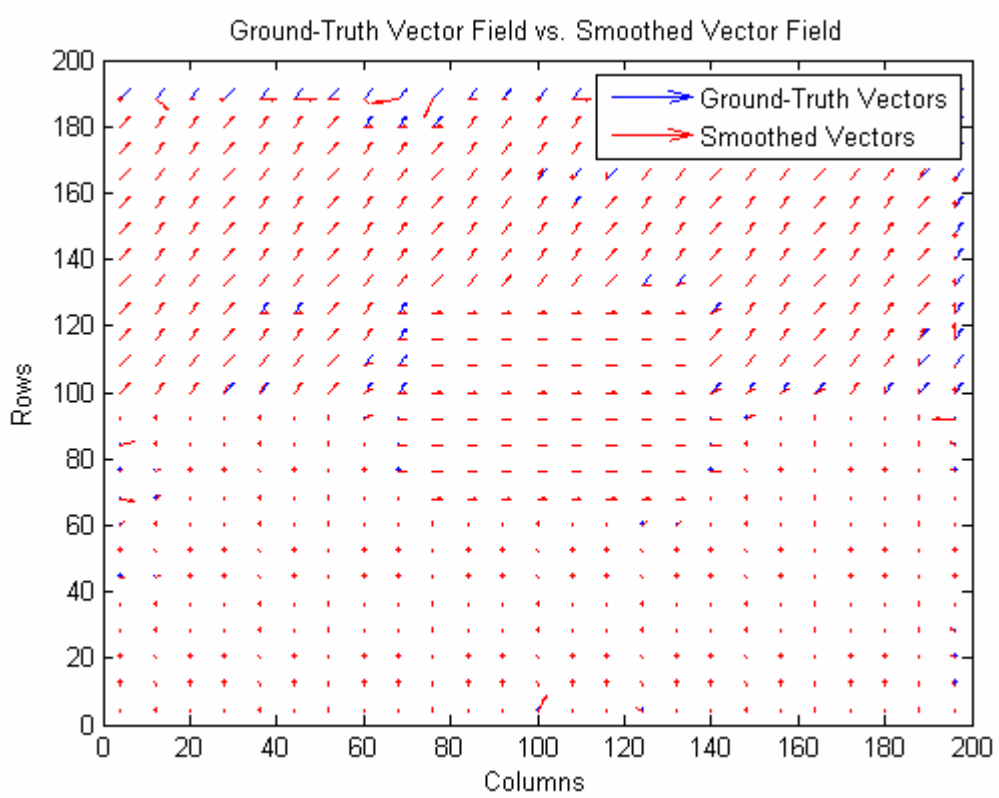

Figure 5.12. Ground Truth vs. Smoothed MVF for Median Filtering of IS–III

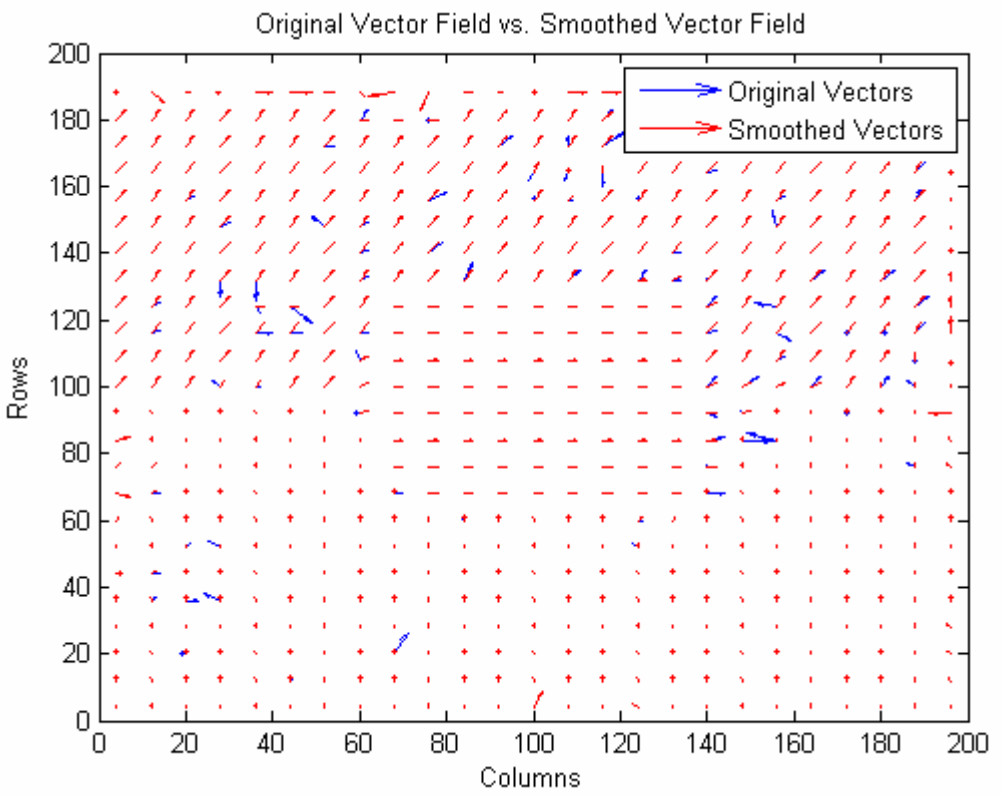

Figure 5.13. Original vs. Smoothed MVF for Median Filtering of IS–III

5.2.3. Vector Median Filtering (VMF) Strategies

After the observation of the shortcomings of classical median filters on the multidimensional data processing, vector median filters were proposed as a solution to the existing problems. Unlike classical median filters, VMF processes the input data without separating them into their channels as is discussed in section 3.2.2.

VMF methods generally utilize a norm to transform the multidimensional data to a scalar value and then they process these scalar values to find the desired filter outputs. By this way, it is guaranteed to generate an output data which is identical to one of the input vectors. There exist many alternatives for the utilized norm operation and in the case of motion vector processing the Euclidian distance, angular distance and their combination are the most preferred ones.

In order to observe the effects of the most commonly used norms on the performance of MVF smoothing operation several simulations were run on our synthetic contents. The

MSE and Smoothness scores of the VMF filters which are constructed by using different norm are given in Table 5.3.

Table 5.3. Comparison of the various VMF methods

IS - I	Original	VMF (Dist1)	VMF (Dist2)	VMF (Dist3)	VMF (Angular)	VMF (Ang.&Dist.)
MSE Score	0.9526	0.5731	0.5632	0.5771	0.7194	0.4625
Smoothness Score	1.3246	3.0877	2.3603	3.0528	2.8467	3.0413

IS - II	Original	VMF (Dist1)	VMF (Dist2)	VMF (Dist3)	VMF (Angular)	VMF (Ang.&Dist.)
MSE Score	29.5482	15.6257	16.9887	16.2098	20.6616	15.1059
Smoothness Score	0.1645	0.4654	0.4551	0.4148	0.2499	0.4072

IS - III	Original	VMF (Dist1)	VMF (Dist2)	VMF (Dist3)	VMF (Angular)	VMF (Ang.&Dist.)
MSE Score	2.1719	0.4269	0.9209	0.4486	0.3775	0.2984
Smoothness Score	0.8254	2.1659	1.6853	2.1834	2.0507	2.2793

The VMF methods, which are listed in Table 5.3, use the following norm criteria:

- VMF (Dist1): Total Euclidian distance to other input vectors
- VMF (Dist2): Euclidian distance to the input vector set's mean
- VMF (Dist3): Euclidian distance to the input vector set's median
- VMF (Angular): Total angular distance to the other input vectors
- VMF (Ang.&Dist.): Combination of the total Euclidian distance with the total angular distance

Obtained results showed that combining the angle information with the distance information gives the best result for most of the times. However, since the computational

complexity of VMF (Ang.&Dist.) is very high compared to the VMF (Dist1) and their performance scores are comparable we will use the VMF (Dist1) as the representative of the vector median filtering methods.

If the vector fields which are processed with VMF are compared with the ground-truth ones, it is seen that its performance level is very similar with the classical median filtering. If Figure 5.12 and Figure 5.18 are compared, it is easy to observe that the VMF prevents the occurrence of output vectors which do not belong to the input vectors. On the other hand, similar to the classical median filtering VMF could not preserve edge structure in some special cases such as seen in Figure 5.14.

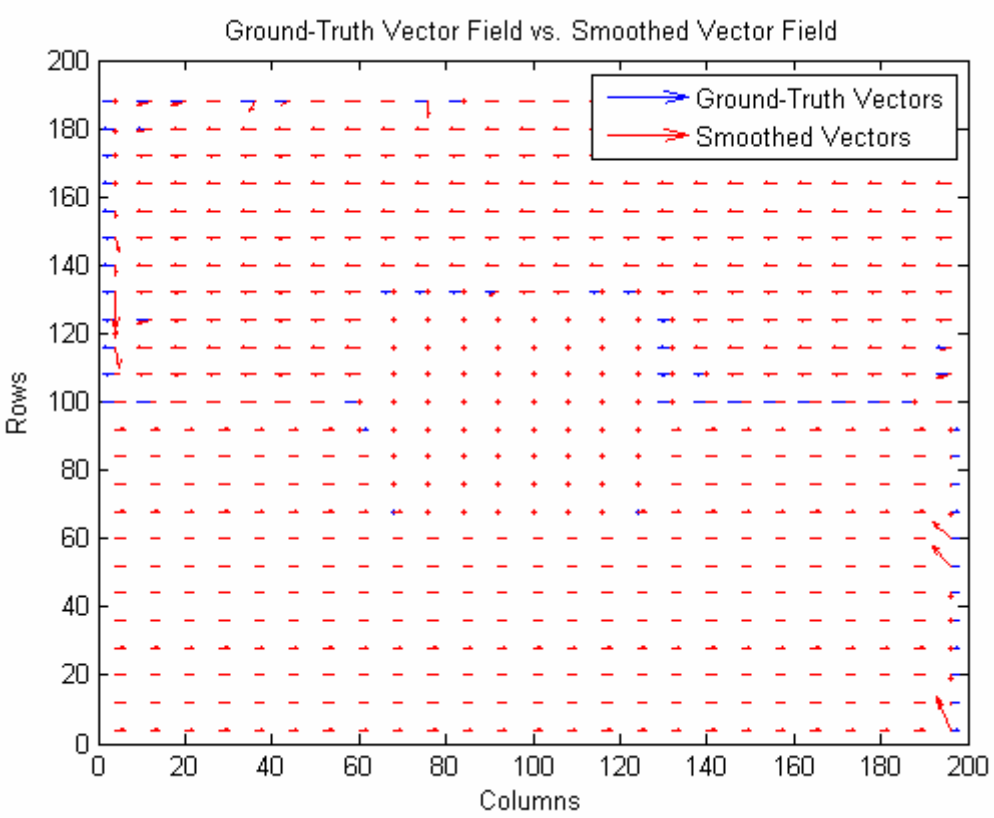

Figure 5.14. Ground Truth vs. Smoothed MVF for Vector Median Filtering of IS–I

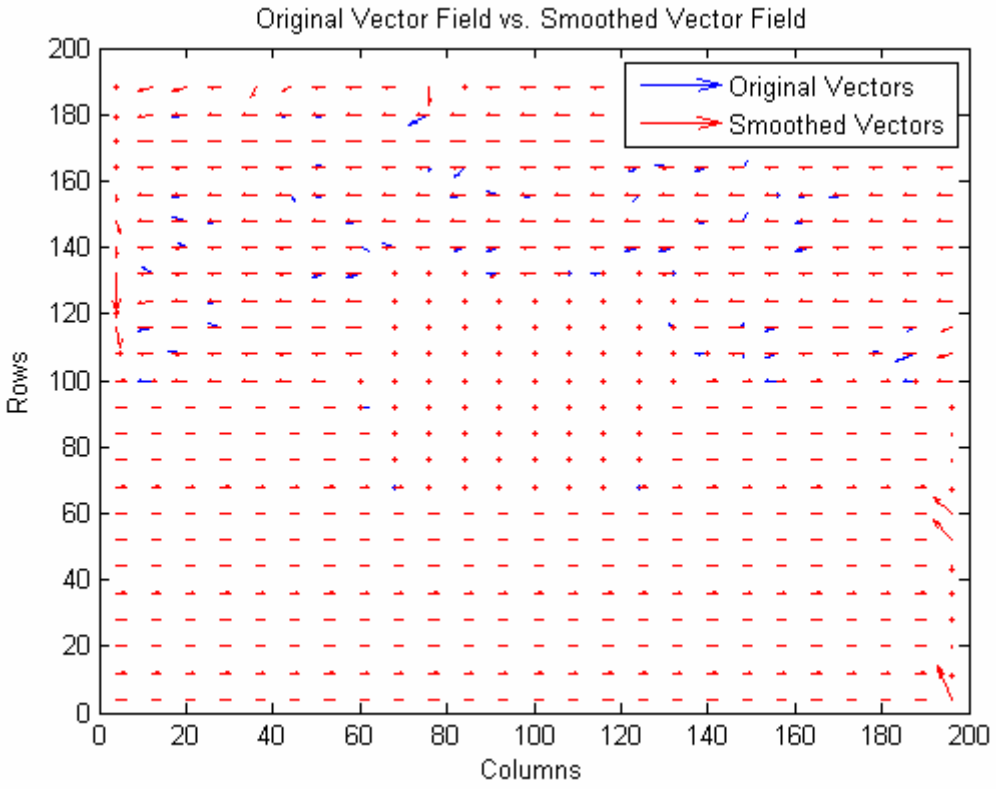

Figure 5.15. Original vs. Smoothed MVF for Vector Median Filtering of IS–I

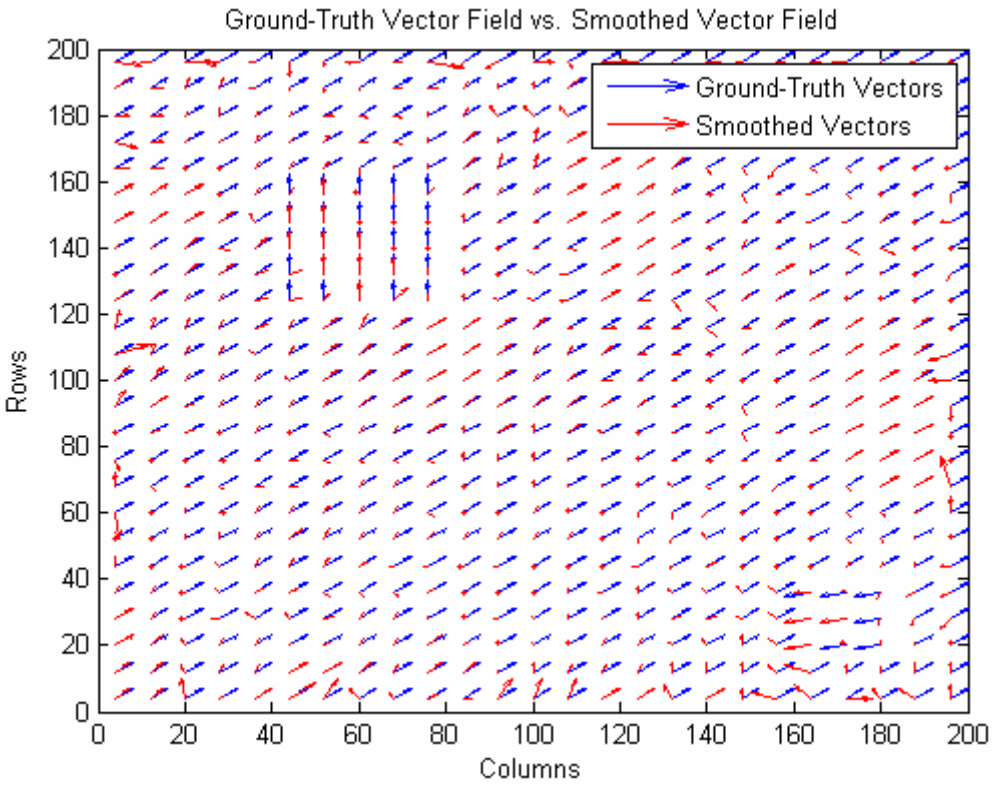

Figure 5.16. Ground Truth vs. Smoothed MVF for Vector Median Filtering of IS–II

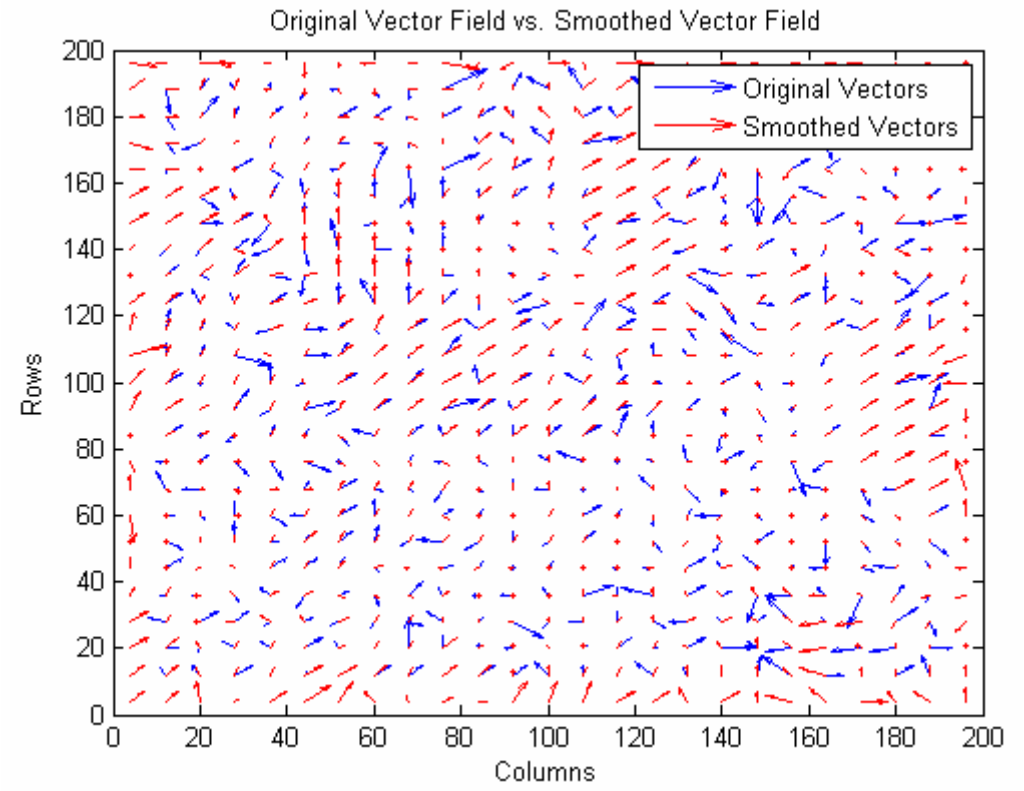

Figure 5.17. Original vs. Smoothed MVF for Vector Median Filtering of IS–II

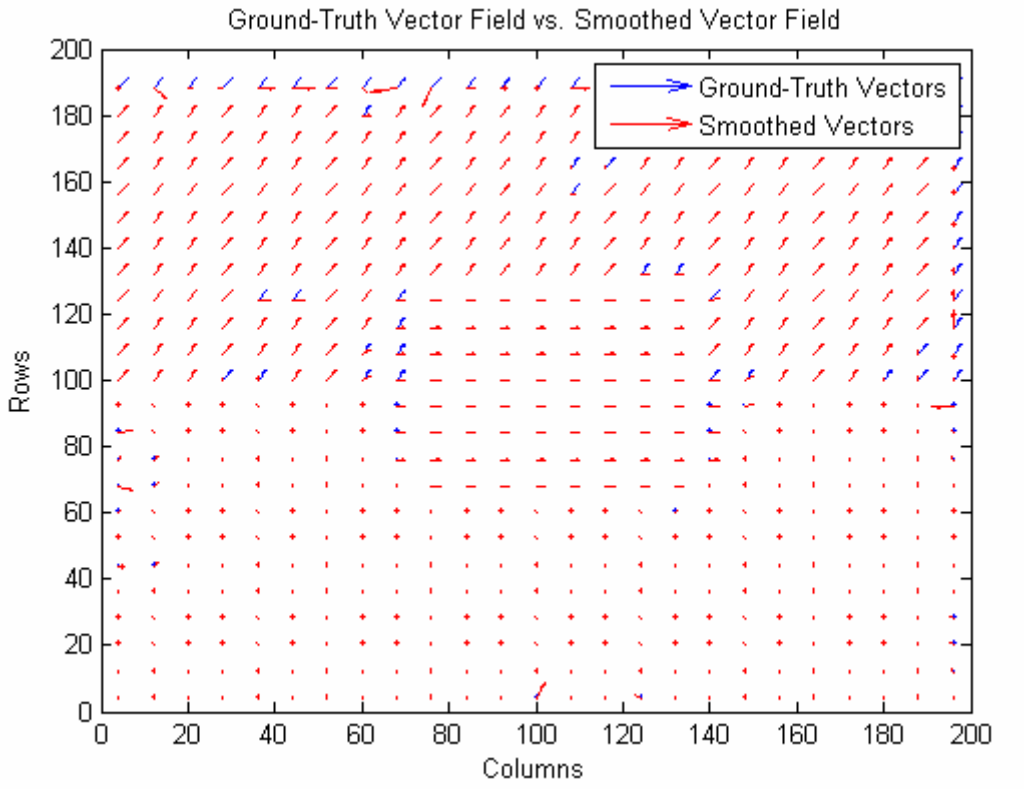

Figure 5.18. Ground Truth vs. Smoothed MVF for Vector Median Filtering of IS–III

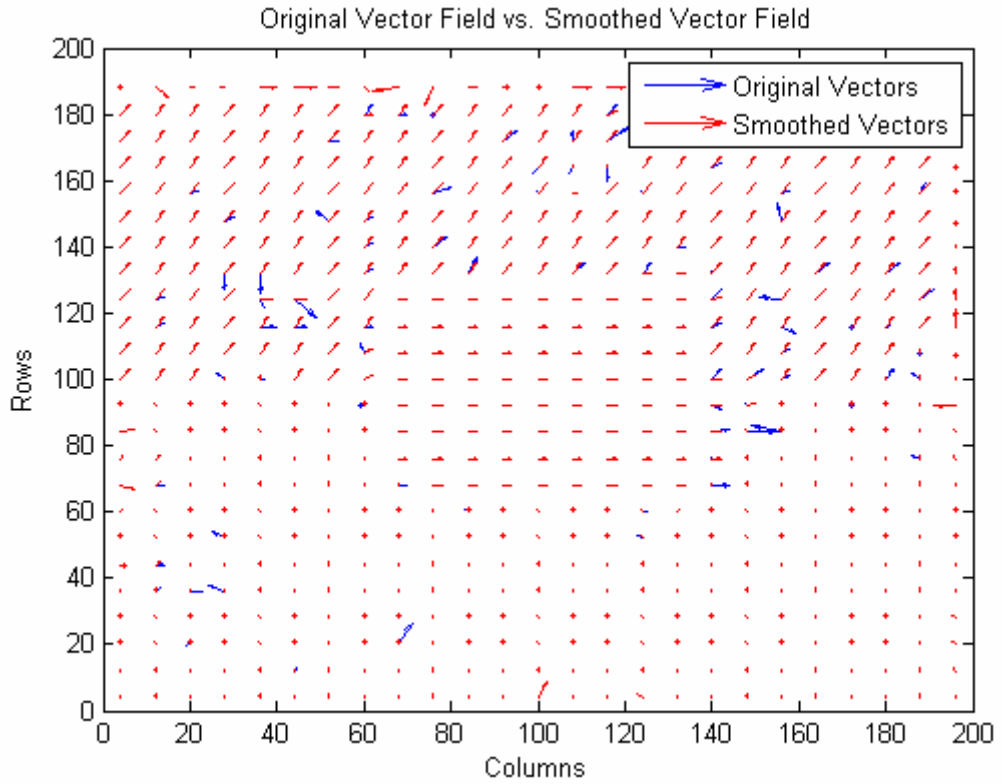

Figure 5.19. Original vs. Smoothed MVF for Vector Median Filtering of IS–III

5.2.4. Motion Discontinuity Based VMF Strategy

The proposed Motion Discontinuity Based VMF approach detects the motion characteristic of current filtering window before starting to process vectors and then, based on the detected characteristic; it only uses the motion vectors of the correlated search blocks. In other words, if the current region is detected as a motion discontinuity region then the region is segmented into two sub-regions and only the vectors which belong to the search blocks of the same sub-region (i.e. the sub-region which contains the center block) are selected as an input set to the main filtering step.

Table 5.4 gives the Smoothness and MSE metric scores for the proposed approach. From the obtained values it is seen that the performance of the method, when it is compared with the basic VMF, changes depending on the considered content. In some cases, probably the motion discontinuity and segmentation blocks fail and this produces performance degradation in terms of the MSE score.

Table 5.4. Smoothness and MSE scores for Motion Discontinuity Based VMF

	Image Sequence - I		**Image Sequence - II**		**Image Sequence - III**	
	Original MVF	Filtered MVF	Original MVF	Filtered MVF	Original MVF	Filtered MVF
MSE Score	0.9526	0.3202	29.5482	24.4726	2.1719	0.6858
Smoothness Score	1.3246	2.7005	0.1645	0.1940	0.8254	1.7174

If Figure 5.20 is compared with Figure 5.14, it seen that the proposed method performs better at the lower corners of the square shaped region that is located at the center. At considered corner locations, the basic VMF chooses a motion vector which actually does not fit to the current position. On the hand, the proposed method detects the existing motion discontinuity and with the help of the utilized segmentation only the vectors which are suitable for the current location are used.

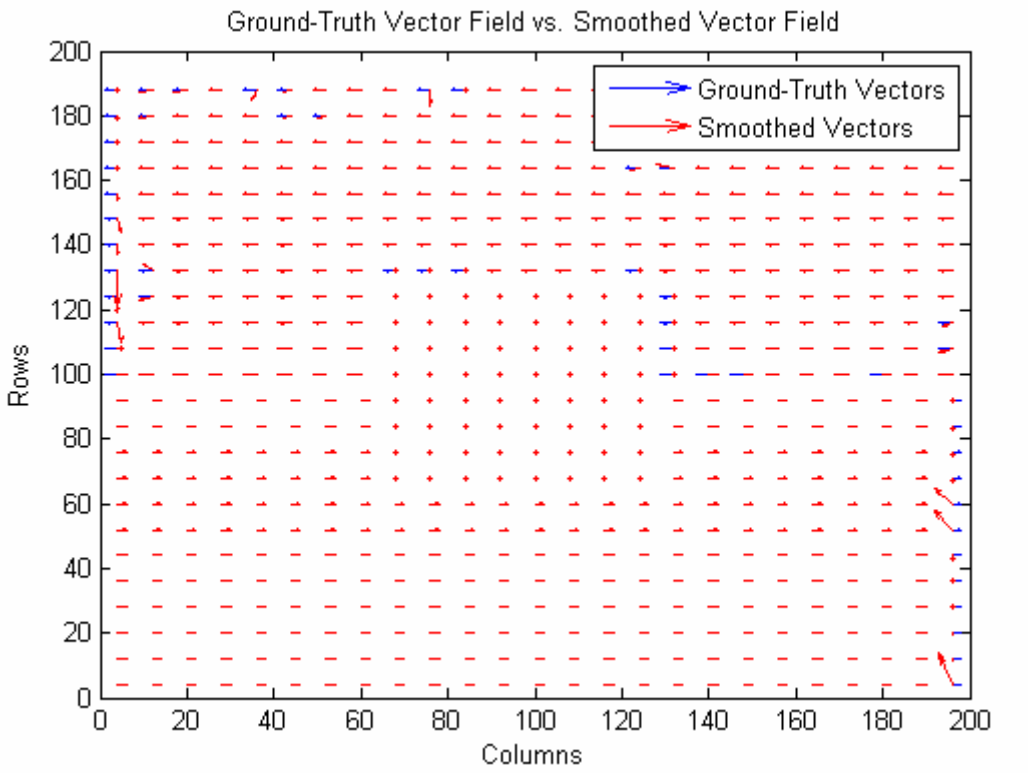

Figure 5.20. Ground Truth vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–I

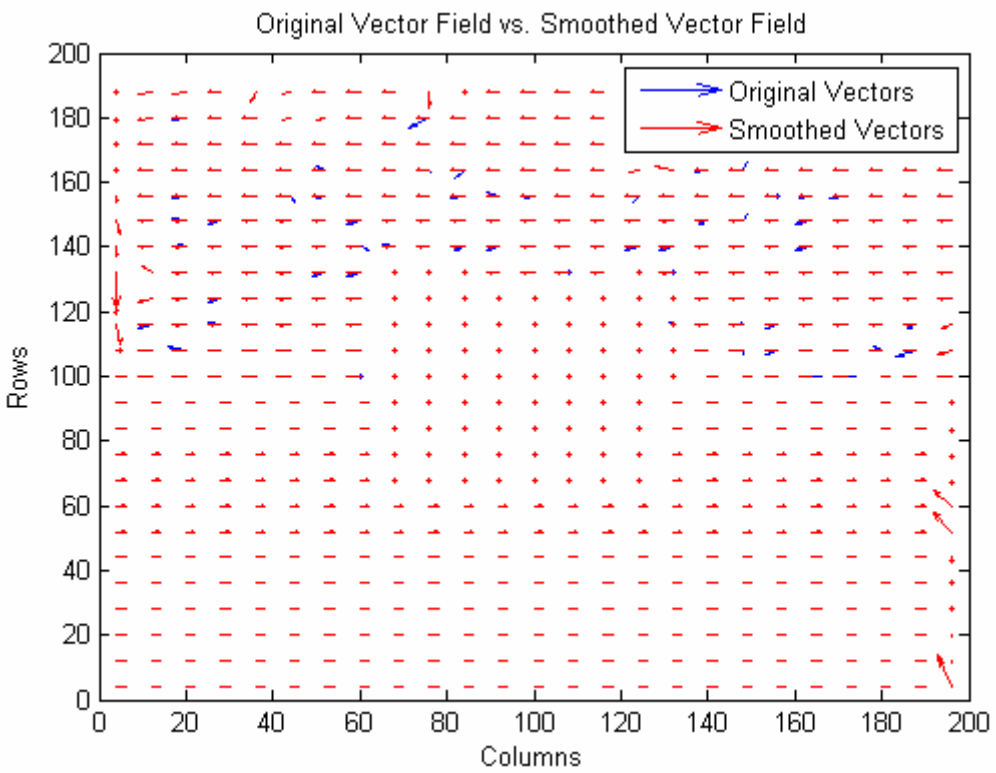

Figure 5.21. Original vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–I

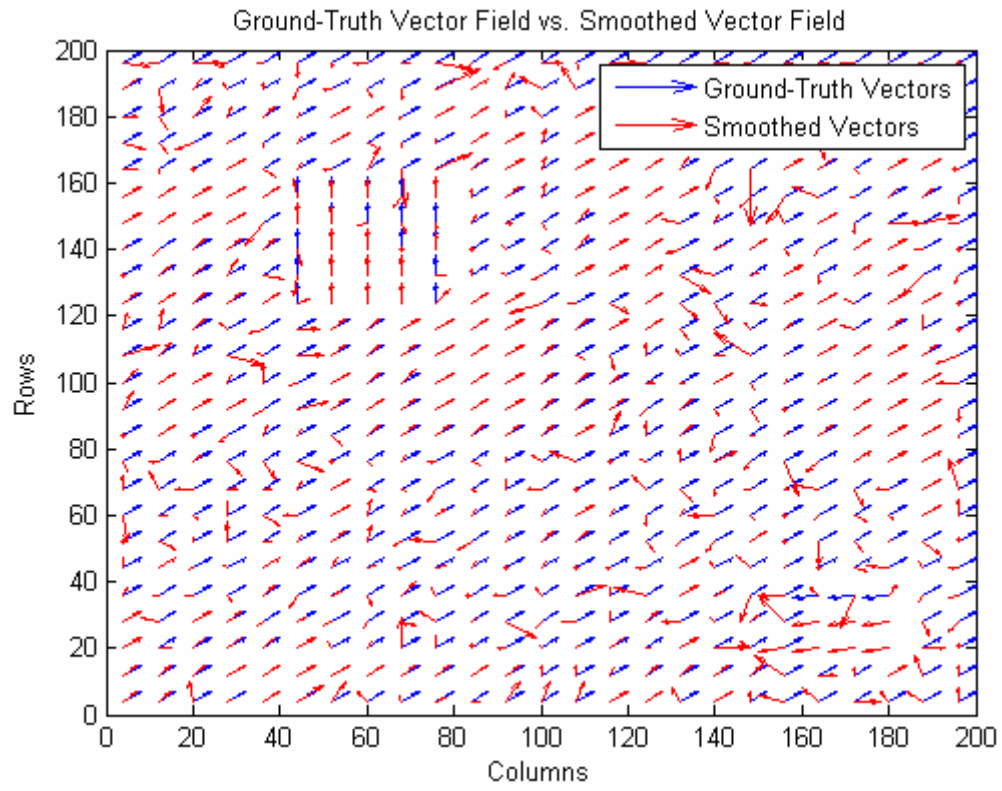

Figure 5.22. Ground Truth vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–II

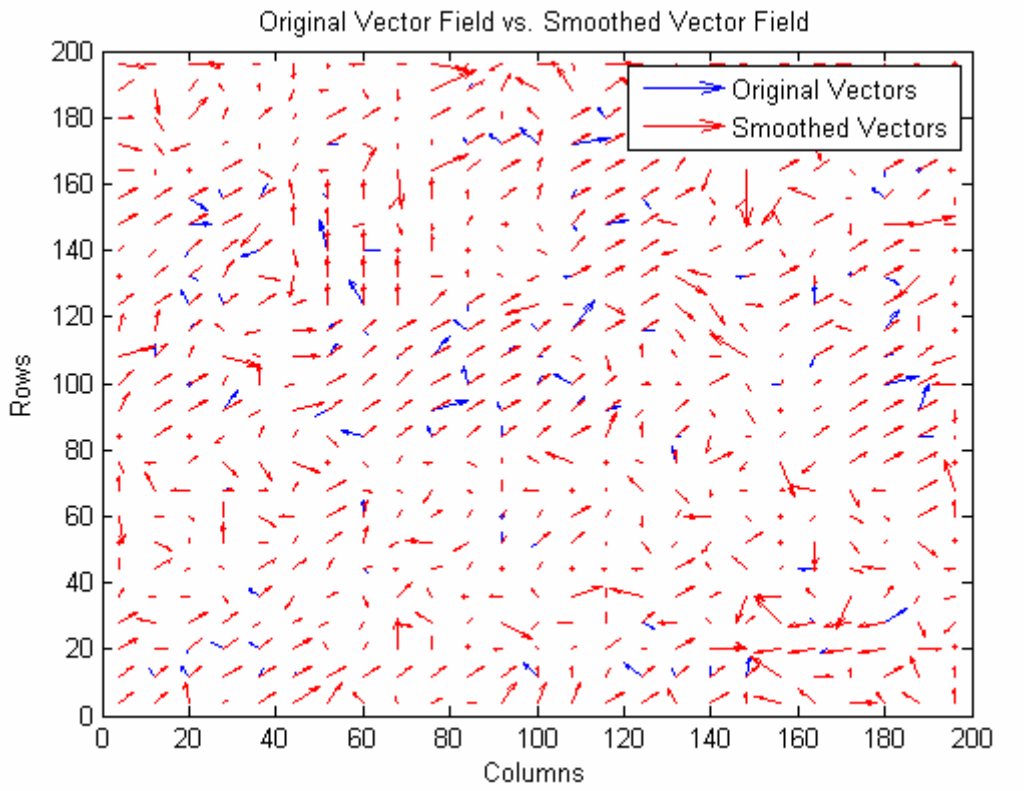

Figure 5.23. Original vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–II

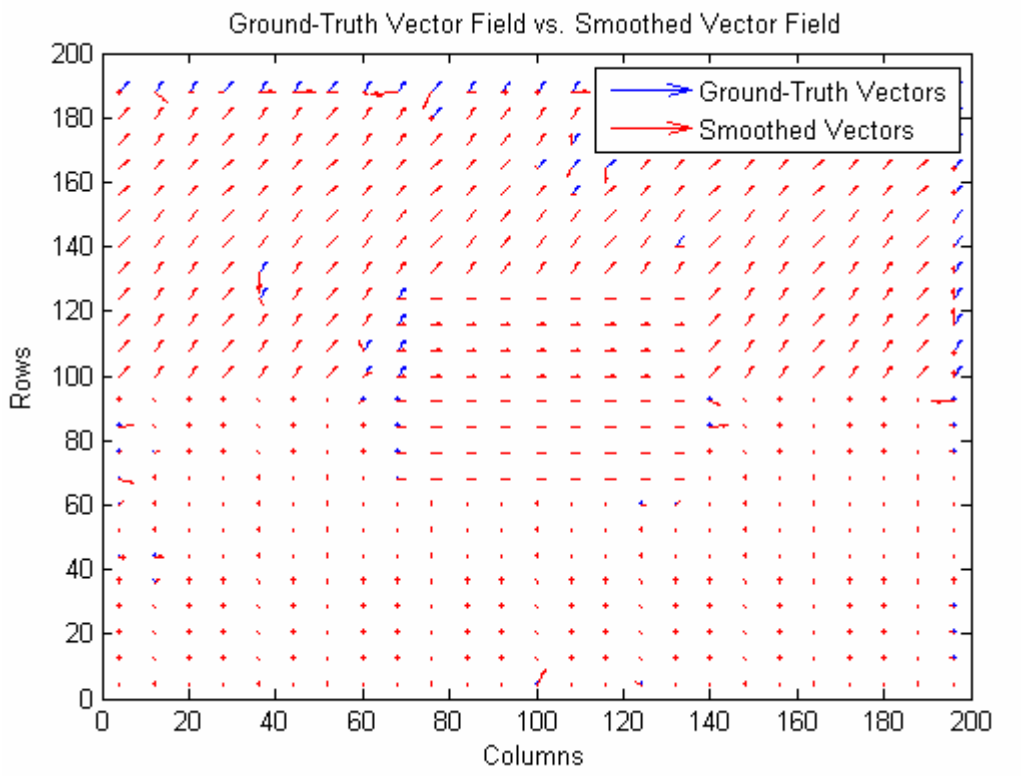

Figure 5.24. Ground Truth vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–III

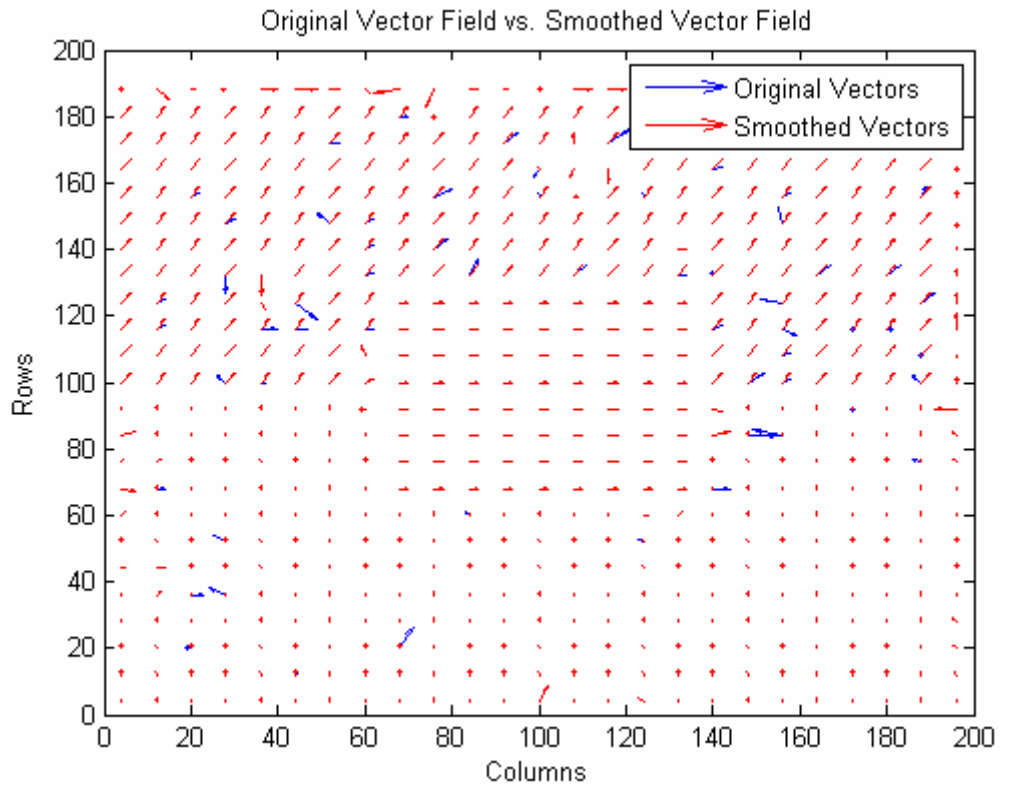

Figure 5.25. Original vs. Smoothed MVF for Motion Discontinuity Based Vector Median Filtering of IS–III

5.2.5. Weighted Vector Median Filtering (WVMF) Strategy

Weighted vector median filters were proposed in order to have a control on the operations of median filters and also to improve their performance. Similar to the classical vector medians, WVMF also utilizes the distance information between the input vectors. However, instead of using the calculated distance values directly they are weighted by using some kind of reliability information.

Table 5.5. Smoothness and MSE scores for WVMF

	Image Sequence - I		**Image Sequence - II**		**Image Sequence - III**	
	Original MVF	Filtered MVF	Original MVF	Filtered MVF	Original MVF	Filtered MVF
MSE Score	0.9526	0.3399	29.5482	5.7561	2.1719	0.1206
Smoothness Score	1.3246	2.9809	0.1645	0.5143	0.8254	2.6406

Table 5.5 gives the Smoothness and MSE scores of the WVMF and the results show that the considered filtering method performs better than all the other methods which are evaluated in the previous sections.

If the vector fields which are created by the WVMF are compared with the ground-truth ones (Figure 5.26, Figure 5.28 and Figure 5.30), it is observed that the filter outputs generally converge to the true ones independent of whether the current location has a motion discontinuity or not and most of the problems, which are observed with classical median and vector median filters, do not exist.

Although it is very obvious that the performance of WVMF is very high compared to the all other considered strategies, its computational complexity is also very high.

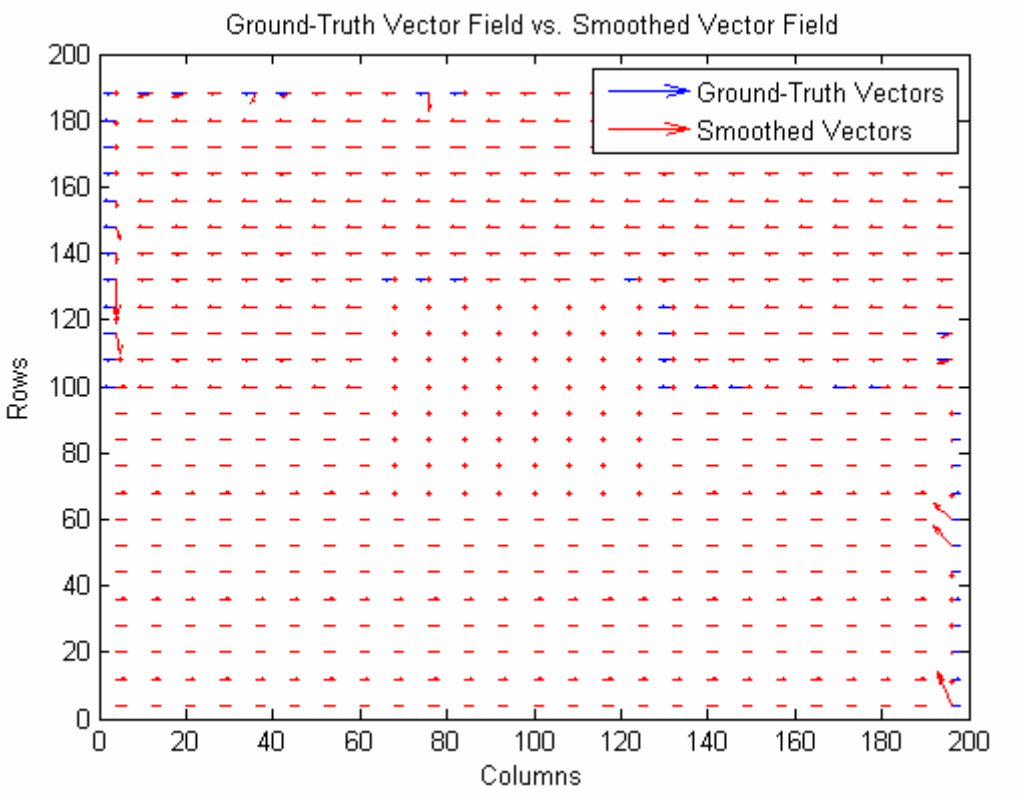

Figure 5.26. Ground Truth vs. Smoothed MVF for WVMF of IS–I

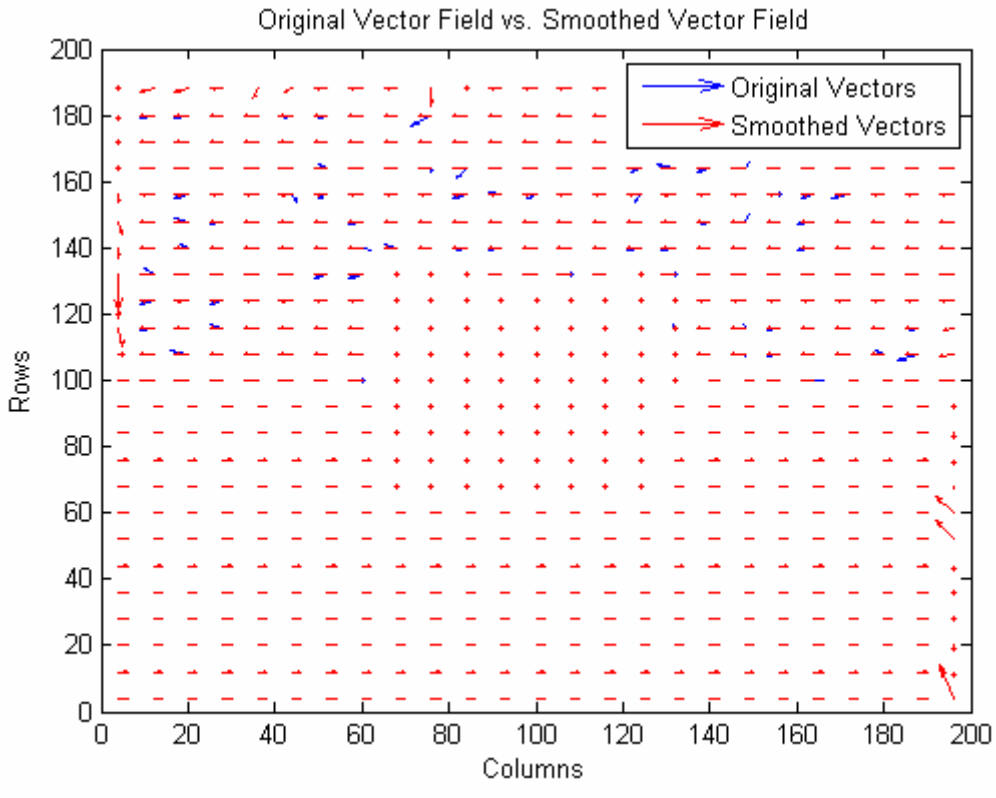

Figure 5.27. Original vs. Smoothed MVF for WVMF of IS–I

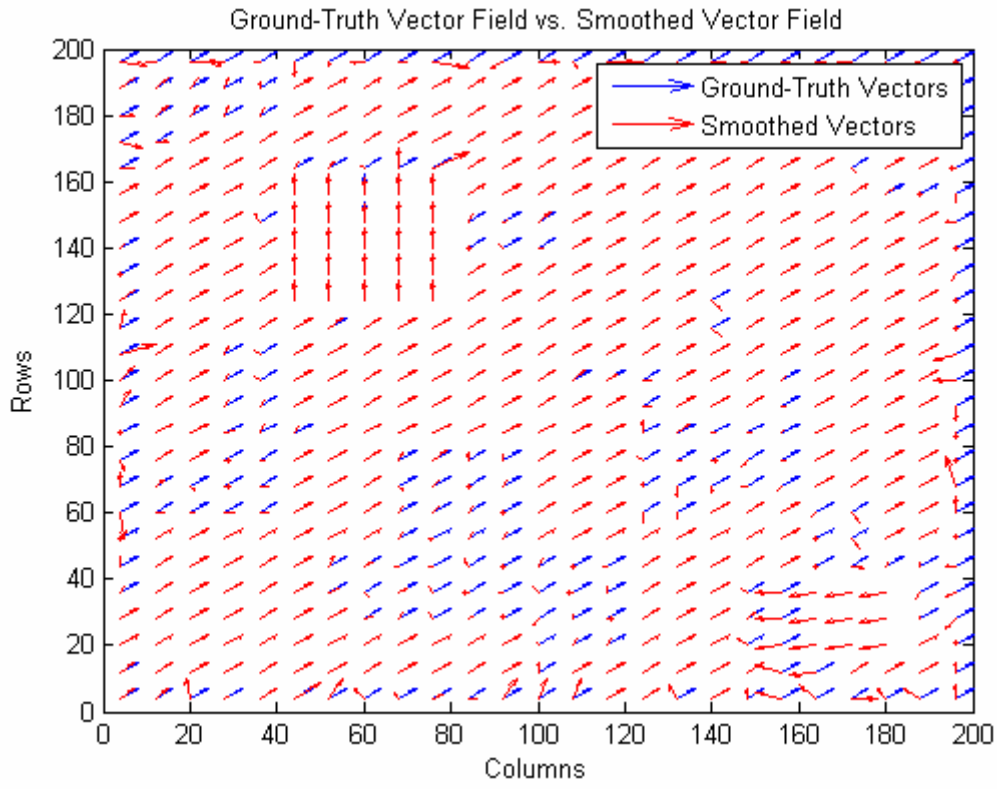

Figure 5.28. Ground Truth vs. Smoothed MVF for WVMF of IS–II

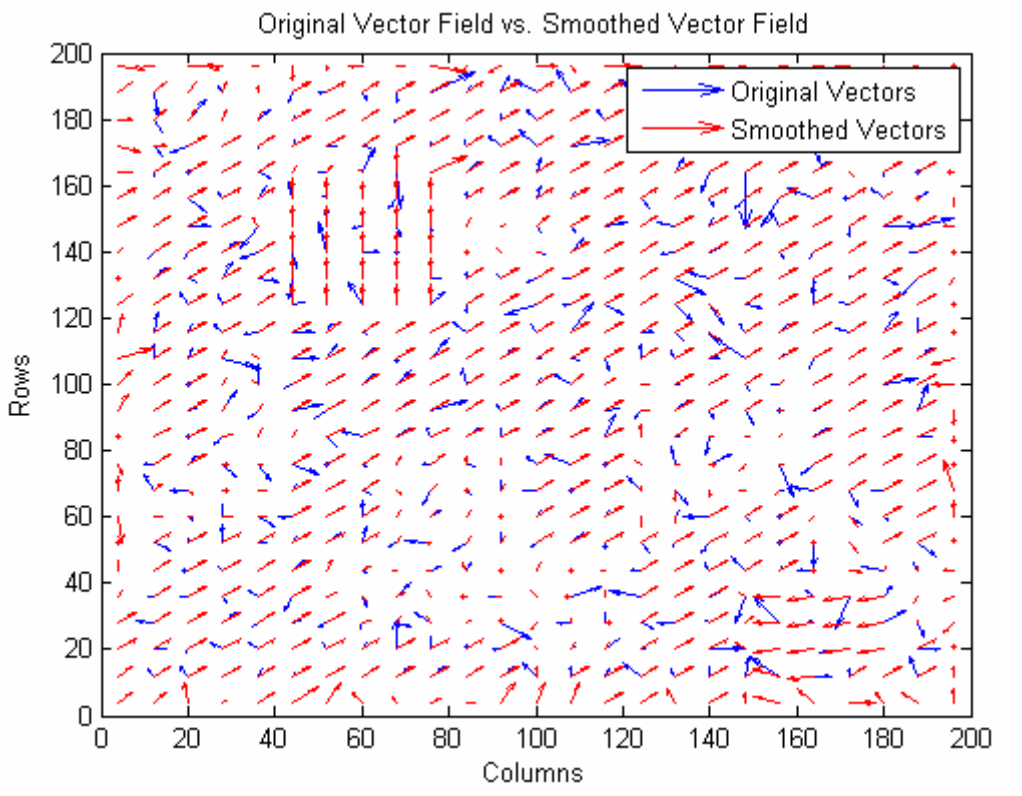

Figure 5.29. Original vs. Smoothed MVF for WVMF of IS–II

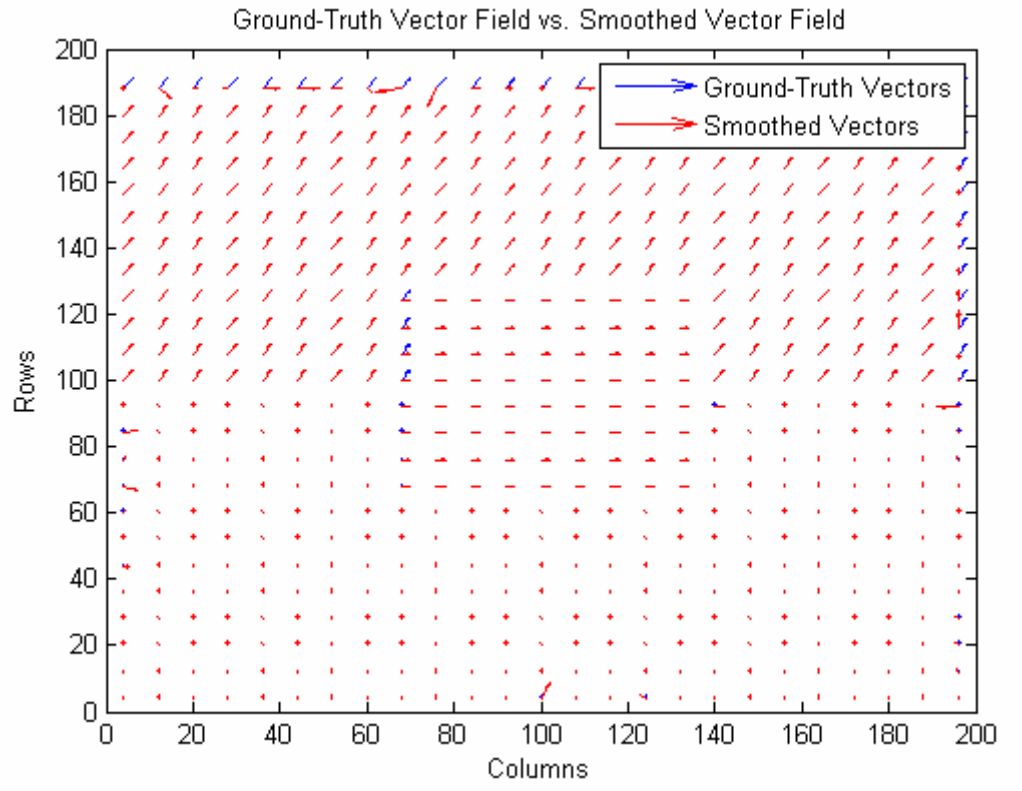

Figure 5.30. Ground Truth vs. Smoothed MVF for WVMF of IS–III

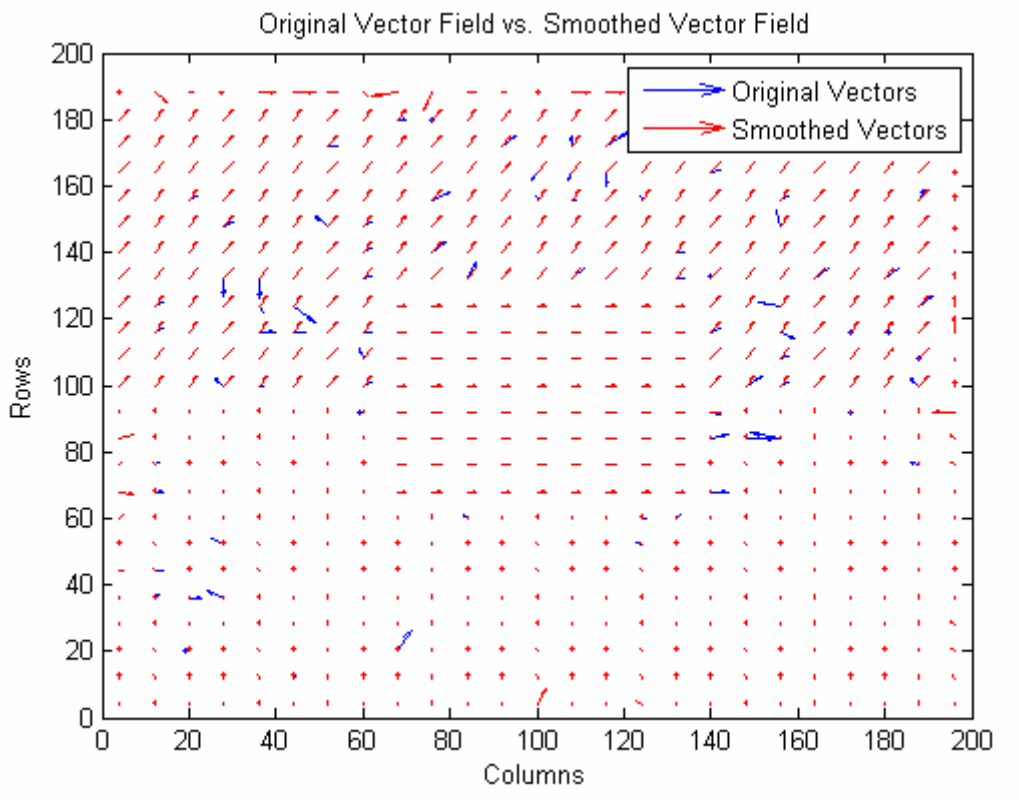

Figure 5.31. Original vs. Smoothed MVF for WVMF of IS–III

5.2.6. Low Complexity WVMF Strategy

Their superior performance makes weighted vector medians more popular in the area of vector filtering. However, the realization of the algorithm is not possible or very difficult especially for low cost applications because of the high computational complexity of the method. The Low Complexity WVMF method, which is proposed in this thesis, aims to obtain the same or at least very similar performance level with the original WVMF.

Below table and the following figures show that the performance of the proposed method is very similar, and as shown in section 4.2 this method has lower complexity.

Table 5.6. Smoothness and MSE scores for Low Complexity WVMF

	Image Sequence - I		**Image Sequence - II**		**Image Sequence - III**	
	Original MVF	Filtered MVF	Original MVF	Filtered MVF	Original MVF	Filtered MVF
MSE Score	0.9526	0.3577	29.5482	7.4064	2.1719	0.1107
Smoothness Score	1.3246	3.0074	0.1645	0,3609	0.8254	2.6337

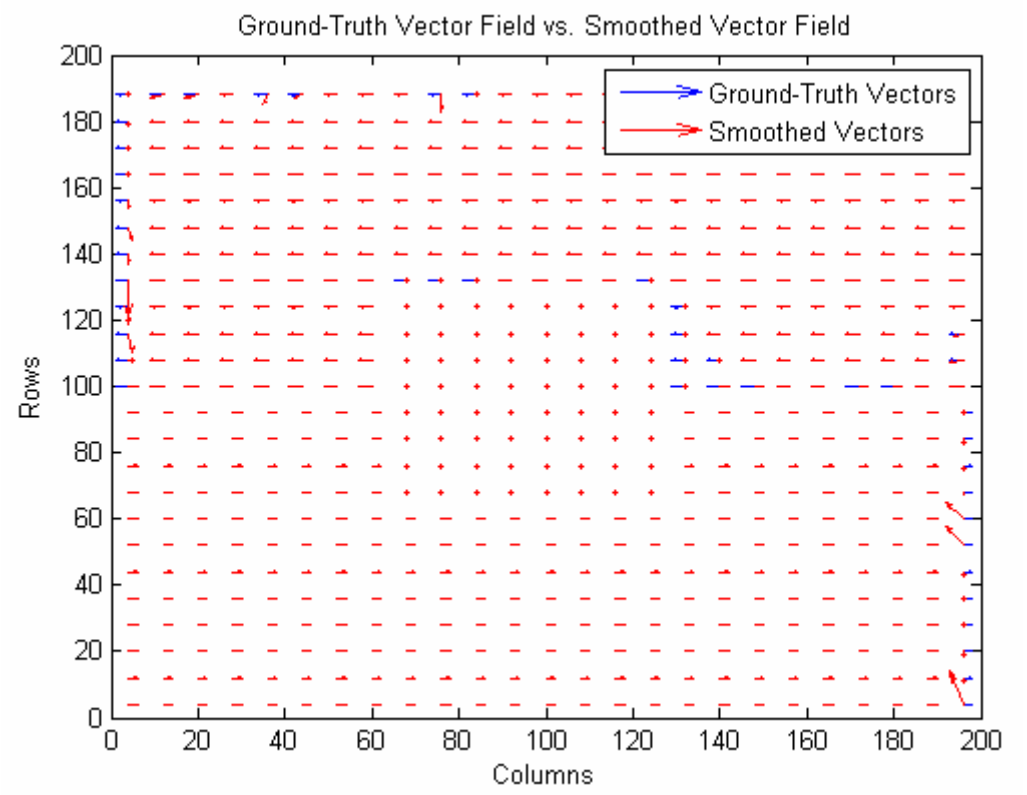

Figure 5.32. Ground Truth vs. Smoothed MVF for Low Complexity WVMF of IS–I

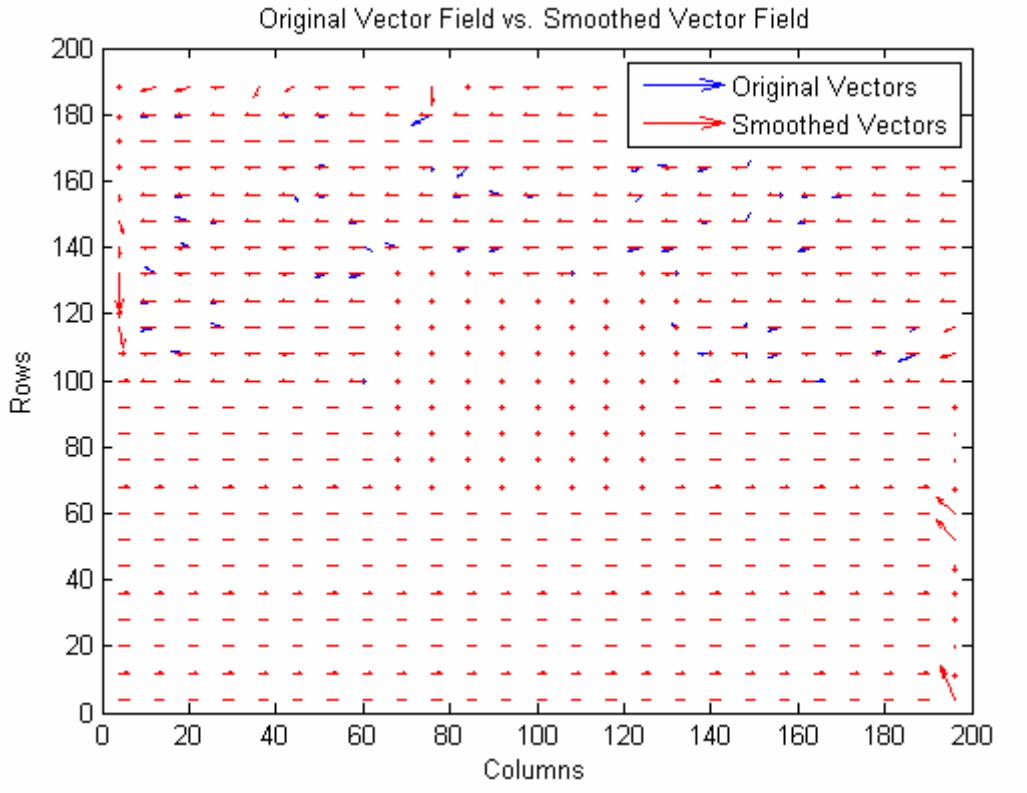

Figure 5.33. Original vs. Smoothed MVF for Low Complexity WVMF of IS–I

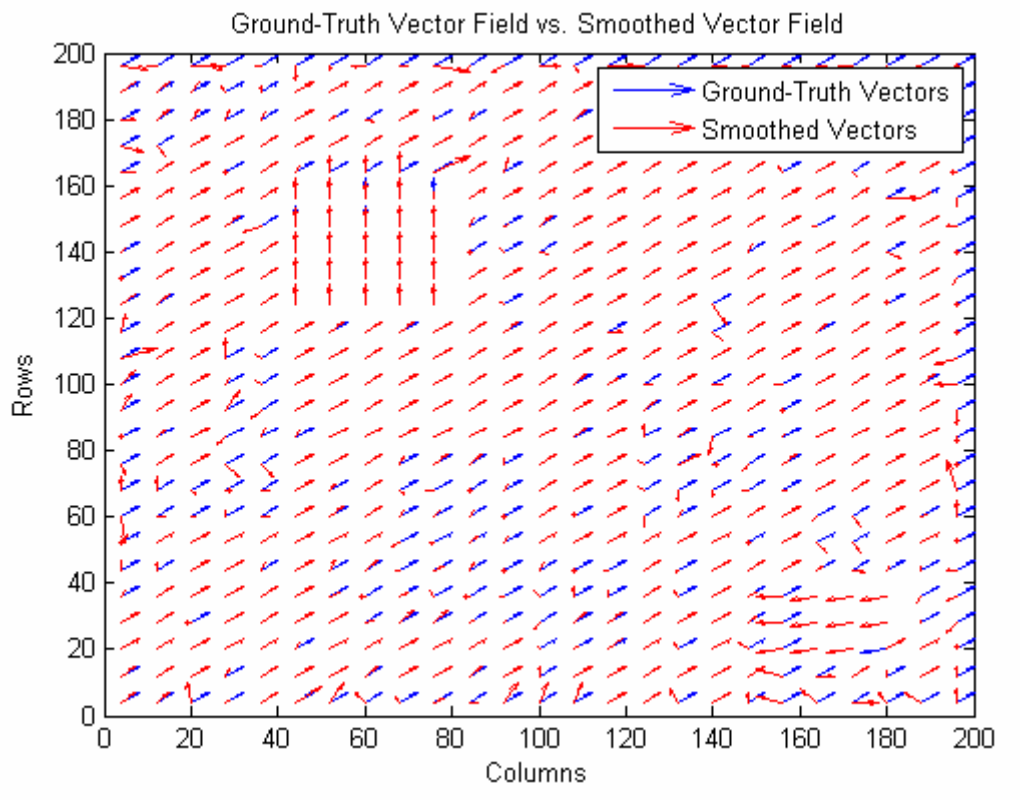

Figure 5.34. Ground Truth vs. Smoothed MVF for Low Complexity WVMF of IS–II

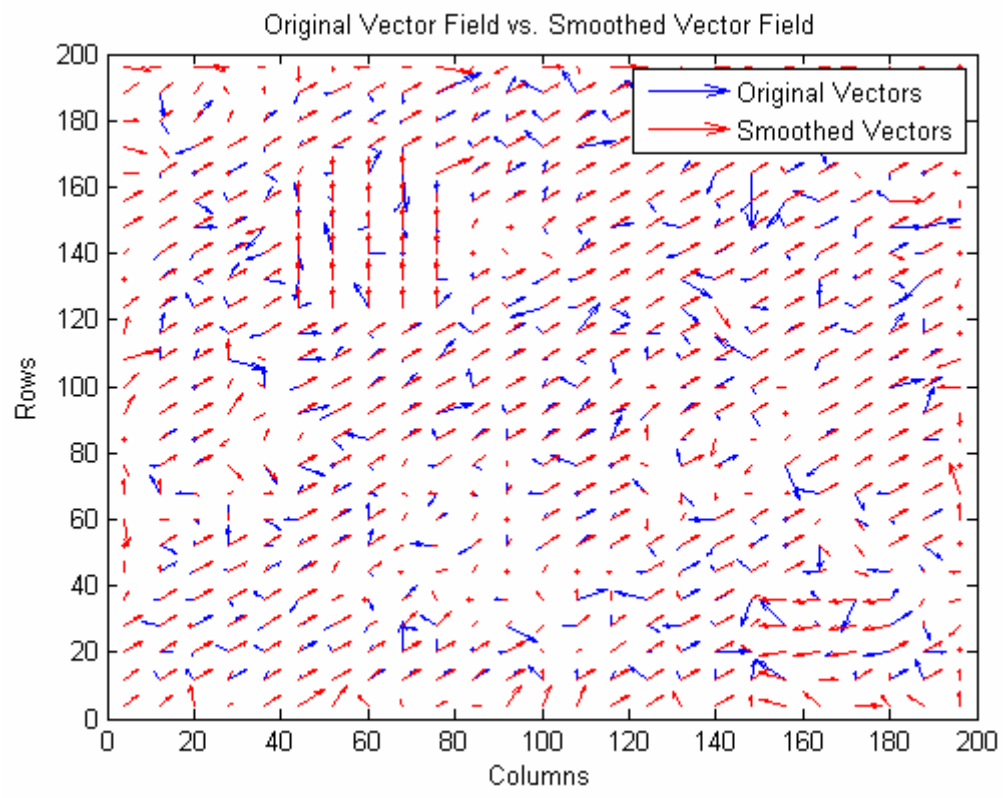

Figure 5.35. Original vs. Smoothed MVF for Low Complexity WVMF of IS–II

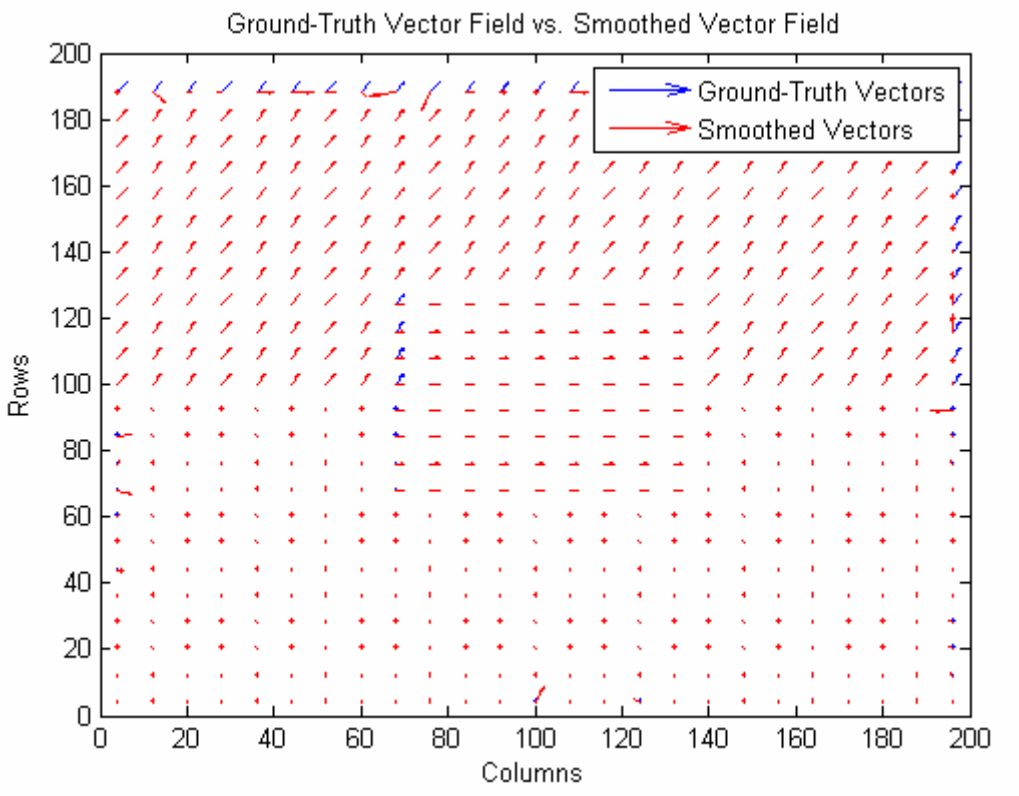

Figure 5.36. Ground Truth vs. Smoothed MVF for Low Complexity WVMF of IS–III

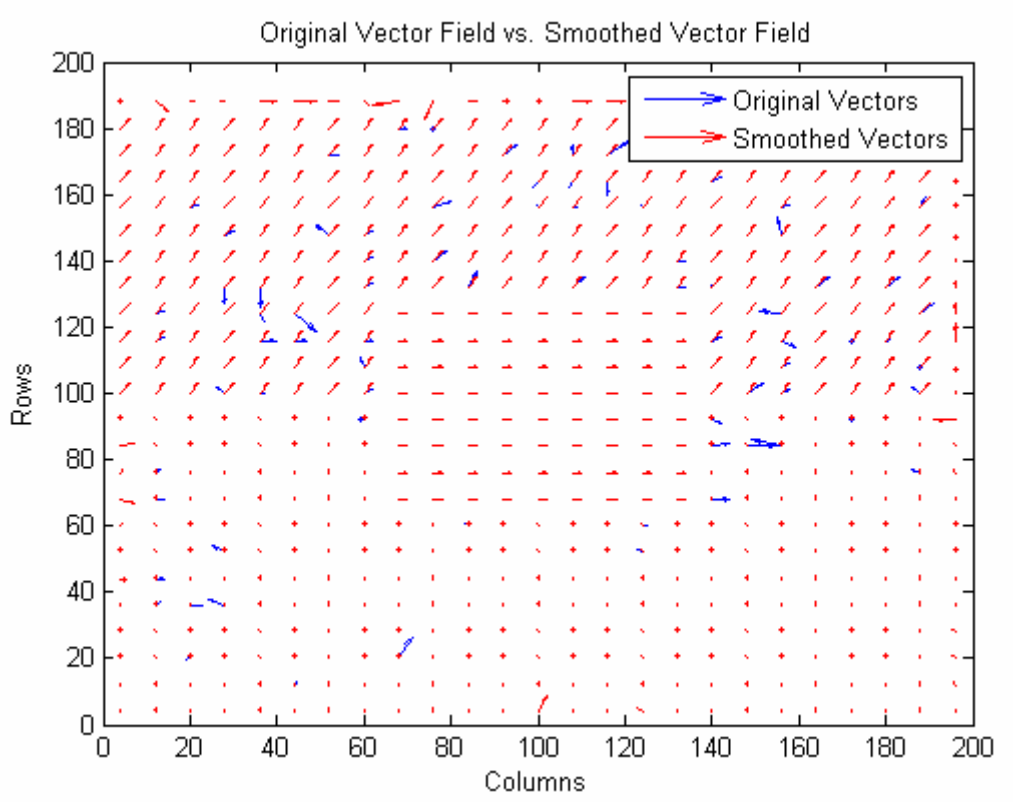

Figure 5.37. Original vs. Smoothed MVF for Low Complexity WVMF of IS–III

5.2.7. Comparison of Implemented Methods

In this section, we will give the Smoothness and MSE scores of all the previously discussed vector field processing methods in a single table (Table 5.7) and we will make a few comments on the obtained results.

The values which are given in Table 5.7 prove that the performance levels of the Weighted Vector Median Filter (WVMF) and its simplified version (Low Complexity WVMF) are significantly higher than the other methods. If the vector fields which are produced by these methods are observed it is seen that these methods have a good discontinuity detection mechanism and, as a result of this successful detection, the probability of converging to the ground-truth data is high with these methods. The average scores for the proposed Low Complexity WVMF are a little bit lower than the scores of original WVMF. However these differences are negligible if they are compared with the performance level differences with other methods such as Mean, Median or VMF. Furthermore, the complexity level of the proposed algorithm is adjustable. By using a version which is a little bit more complex than the currently utilized one, it is possible to obtain exactly the same performance level with the original WVMF method.

Table 5.7. Smoothness and MSE scores for all methods

IS - I	Original	Mean	Median	VMF	Discont. Based VMF	WVMF	Low Comp. WVMF
MSE Score	0.9526	0.6601	0.5672	0.5731	0.3202	0.3399	0.3577
Smoothness Score	1.3246	2.3426	3.0972	3.0877	2.7005	2.9809	3.0074

IS - II	Original	Mean	Median	VMF	Discont. Based VMF	WVMF	Low Comp. WVMF
MSE Score	29.5482	15.9282	15.1531	15.6257	24.4726	5.7561	7.4064
Smoothness Score	0.1645	0.5876	0.5218	0.4654	0.1940	0.5143	0.3609

IS - III	Original	Mean	Median	VMF	Discont. Based VMF	WVMF	Low Comp. WVMF
MSE Score	2.1719	0.8755	0.4170	0.4269	0.6858	0.1206	0.1107
Smoothness Score	0.8254	1.7059	2.2108	2.1659	1.7174	2.6406	2.6337

For the remaining methods (i.e. Mean, Median, VMF and Discontinuity Based VMF), the scores of utilized metrics are in the comparable ranges. However, observations on the plots of produced vector fields have shown that the overall performance level of the median filtering based strategies are better than the mean filters. The main reason for this decision is its better discontinuity preserving characteristic.

Among the median filtering based methods (i.e. Median, VMF and Discontinuity Based VMF), the basic median filtering has a lower performance level especially in the discontinuity regions. The main reason of this poor performance level is the occurrence of output vectors which do not exist actually in the input set because of the processing of each data channel separately. Although vector medians perform better than the scalar median operations for most of the time, they still have some problems as is discussed in section 4.1.

Vector median filters are a good compromise between low complexity methods with poor performances (such as Mean and Median) and high complexity methods with better performance levels (such as WVMF and Low Complexity WVMF). As discussed in sections 3.2.2 and 5.2.3, vector median filters have many variations with a little bit different performance levels. Table 3.1 and Table 5.3 give a rough idea about the computational complexities and MVF processing performances of selected vector median strategies. From the given values, it could be said that the Euclidian distance based vector median filtering is a good choice if the tradeoff between the complexity and quality is taken into account.

5.3. Frame Rate Conversion Application

In order to observe the effects of MVF post-processing algorithms (i.e. Smoothing and Refinement) on a complete system, we have implemented a testbench environment. The implemented environment is actually a Frame Rate Conversion (FRC) application with a configurable flow. The details for configuring and running the environment are given in Appendix A.

The implemented application has three operational blocks: motion estimation, post-processing and frame interpolation.

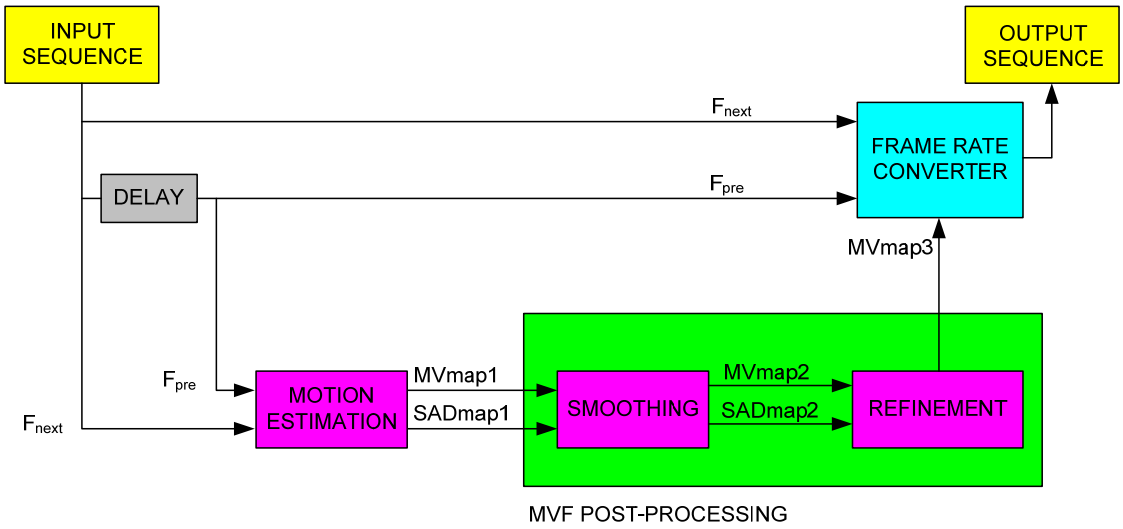

Figure 5.38. Block diagram of the implemented FRC application

The testbench environment provides few different customization options for each block. By this way it becomes very easy to evaluate the effect of each block on the overall system performance and to compare the performances of different configurations.

The following subsections will give the details of the implemented testbench environment and illustrative frames from the outputs will be given to show the performance of different configurations.

5.3.1. Motion Estimation Block

It is possible to choose one of the four available motion estimation strategies (Full search, Hierarchical search, Hexagonal search and 3D Recursive search) by using the GUI of testbench environment. The choices contain three error minimization based methods (Full search, Hierarchical search and Hexagonal search) with different computational complexity levels and also a true-motion estimator (3D Recursive search). The details of the implemented algorithms are given in section 2.

In addition to the selection of the desired motion estimation algorithm, it is also possible to play with the size of the search blocks and search range by using the provided GUI.

The selected motion estimation algorithm has an effect on the overall performance of the FRC application. Since all the following blocks of the FRC application are built on the information of motion vectors which are created by the estimator block, the quality of the MVF (i.e. smoothness and closeness to the true motion) directly affects the quality of the final output video. While making a choice on the motion estimation it should be known that the method which has the highest complexity or the minimum matching performance does not always correspond to the one best method in terms of closeness to the true-motion.

Figure 5.39 gives two consequent frames from "Bus.cif" sequence. In the scene, the bus and the TV logo are stationary and all the other objects have a uniform motion to the right of the scene. If the motion vectors which are created for this scene by different motion estimators are observed (see Figure 5.40, 5.41, 5.42 and 5.43), it is seen that although the Full Search and Hierarchical Search algorithms give the best scores in terms of block matching scores, the output MVF of the 3D Recursive Search is clearly better in terms closeness to the true-motion. It is also seen that the low complexity Hexagonal Search method produces a very noisy MVF because of the previously discussed (see section 2.1.3) local minimum problem of the fast search strategies.

Figure 5.39. 1st and 2nd frames of "Bus.avi" sequence

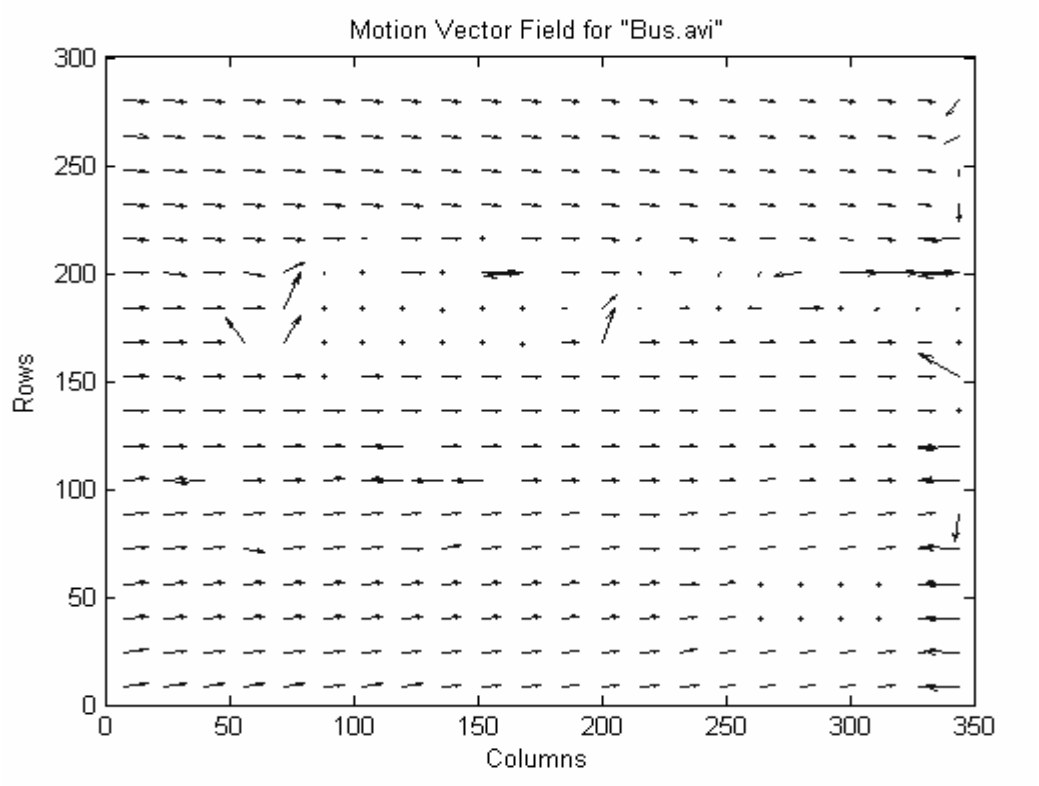

Figure 5.40. MVF produced by Full Search

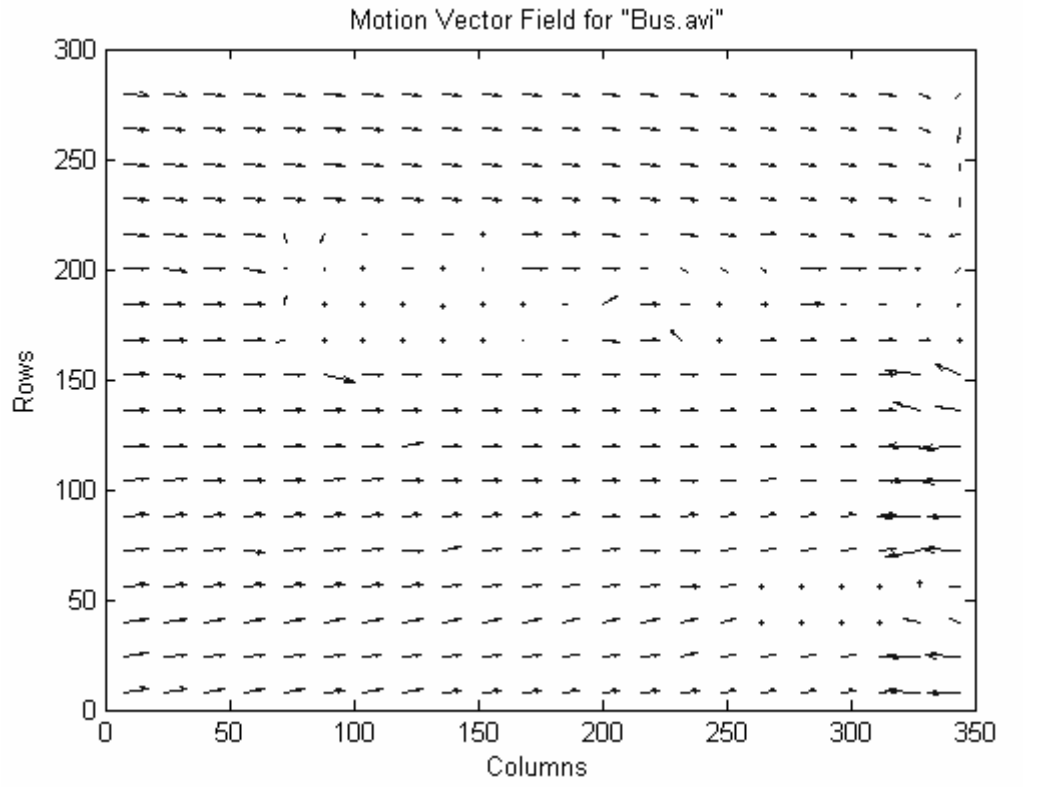

Figure 5.41. MVF produced by Hierarchical Search

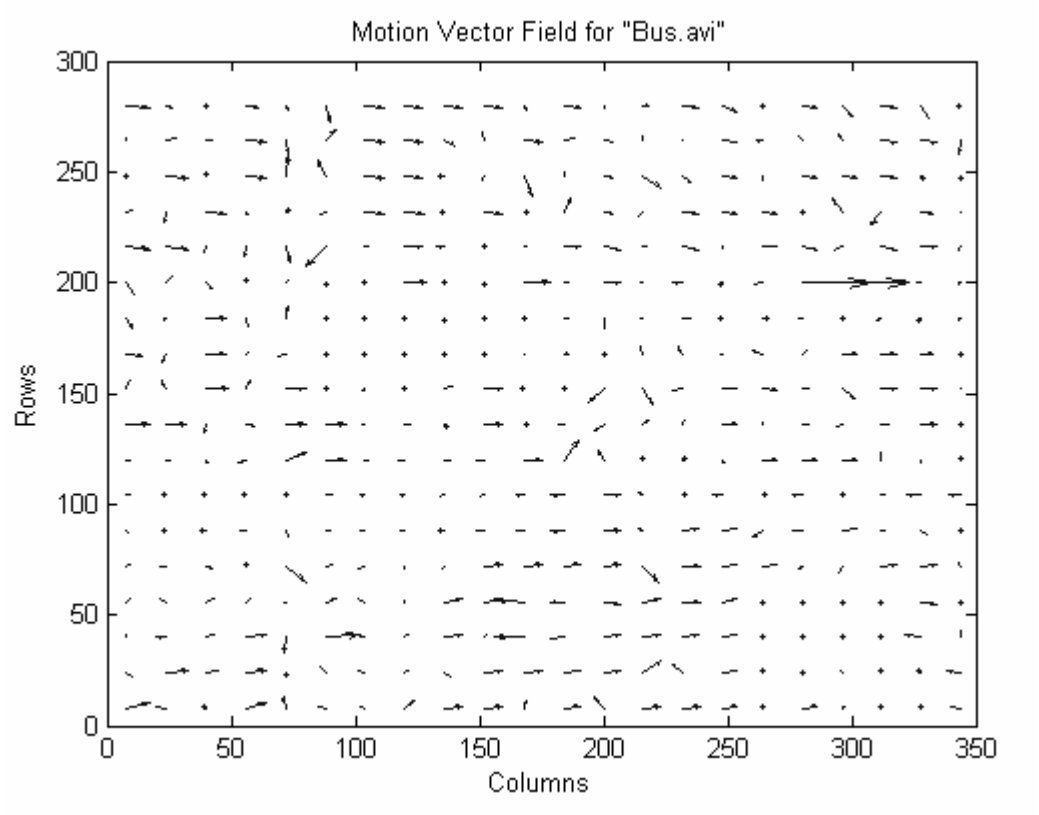

Figure 5.42. MVF produced by Hexagonal Search

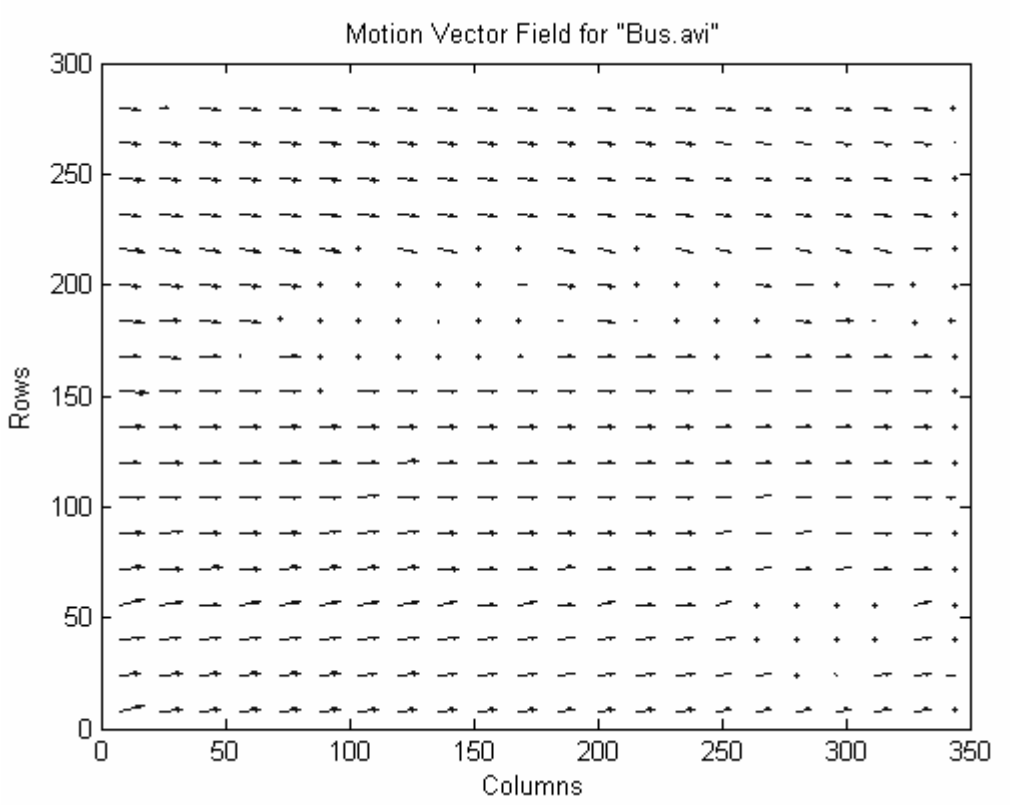

Figure 5.43. MVF produced by 3D Recursive Search

5.3.2. MVF Post-Processing Block

By using the provided GUI it is possible to choose one of the four possible options for the post-processing of the vector field which is created by the motion estimation block:

- Option-1: Applying Smoothing and Refinement
- Option-2: Applying only Smoothing
- Option-3: Applying only Refinement
- Option-4: No processing on MVF

The list of available Smoothing and Refinement methods and also the details for configuring the testbench environment with the desired post-processing settings can be found in Appendix A.

The methods which are listed under the Smoothing title generally focus on obtaining a smooth vector field by removing the noisy vectors. These vectors are generally generated because of the local minimum problems of the search strategies or in some cases the wrong vectors are created since the true motion vector does not always correspond to the candidate vector with the minimum matching error. During the smoothing of the vector field, it is also important to preserve the structure in motion discontinuity regions. Performed simulations showed that the applications which utilize smoothing methods with explicit or implicit discontinuity detection mechanisms produce outputs with better visual quality levels.

In order to observe the effect of MVF smoothing we have performed simulations on many different video sequences. The outputs showed that the smoothing operation improves the visual quality of the implemented FRC application significantly. Figure 5.44 and Figure 5.46 are two illustrative frames which are extracted from the output videos that are created by using MVF smoothing strategy. If these figures are compared with the ones that are created without using any MVF smoothing strategy (Figure 5.45 and Figure 5.47)

it is obvious that the most of the artifacts those exist in Figure 5.45 and 5.47 are not visible in the Figure 5.44 and 5.46.

Figure 5.44. Frame which is interpolated between the 157th and 158th frames of the "Foreman.avi" sequence by using the MVF strategy

Figure 5.45. Frame which is interpolated between the 157th and 158th frames of the "Foreman.avi" sequence without using the MVF strategy

Figure 5.46. Frame which is interpolated between the 4th and 5th frames of the “Bus.avi” sequence by using the MVF strategy

Figure 5.47. Frame which is interpolated between the 4th and 5th frames of the “Bus.avi” sequence without using the MVF strategy

The methods which are listed under the Refinement title desire to produce denser vector fields. Most of the commonly used motion estimators are block based and since the sizes of the utilized blocks are very small when compared to the frame size, they assume that a single motion vector is enough to represent the motion characteristic of all the pixels which belong to the considered block. However, this assumption generally fails at motion discontinuity regions and the refinement methods especially focus on these regions in order to solve the problems. The common approach for fixing the problem is dividing the currently utilized search blocks into smaller sub-blocks and then assigning appropriate motion vectors to the new sub-blocks. By this way, the refinement algorithm creates a denser vector field and the visual performance of the final application is increased especially at motion discontinuity regions.

Figure 5.48. The effect of Refinement on the reduction of *Blocking Artifacts*: a) the output with using refinement, b) the output without using refinement

Figure 5.49. The effect of Refinement on the reduction of *Stair Artifacts*: a) the output with using refinement, b) the output without refinement

Two of the most commonly seen artifacts which appear because of the utilization of block based processing are Blocking Artifacts and Stair Artifacts. Figure 5.48 and Figure 5.49 are the illustrative examples of the two artifacts. From given figures, it is observed that the refinement strategy successfully decreases the visibility of considered artifacts.

5.3.3. Motion Compensated Frame Interpolation Block

The last block for the implemented FRC application is the frame interpolation block. By using a motion compensation based strategy, this block interpolates new frames between the consequent frames and it doubles the frame rate of the input video. For performing this task, the implemented testbench environment provides four alternative ways:

- Bidirectional Frame Interpolation Method
- Graceful Degradation Based Interpolation Method
- Basic & Bidirectional Interpolation Method
- Basic & Graceful Degradation Method

The bidirectional frame interpolation method (see Figure 5.50) is based on the following assumption: the correct motion vector for the block B'(x,y) can be found inside the vector set which contains the motion vector of B(x,y) and its eight neighbor motion vectors. Then the appropriate motion vector (D"(x,y)) is found by calculating the matching scores for all available vectors and dividing the one with the minimum matching score by two. After finding the best vectors, it is straightforward to obtain the interpolated frame by using the following formula:

$$B'(x,y) = \frac{1}{2}\left(B_{pre}(x\text{-}u,y\text{-}v) + B_{next}(x+u,y+v)\right) \qquad (4.11)$$

where B_{pre} and B_{next} are blocks from previous and next frames respectively. Also u and v are the horizontal and vertical components of the motion vector D"(x,y).

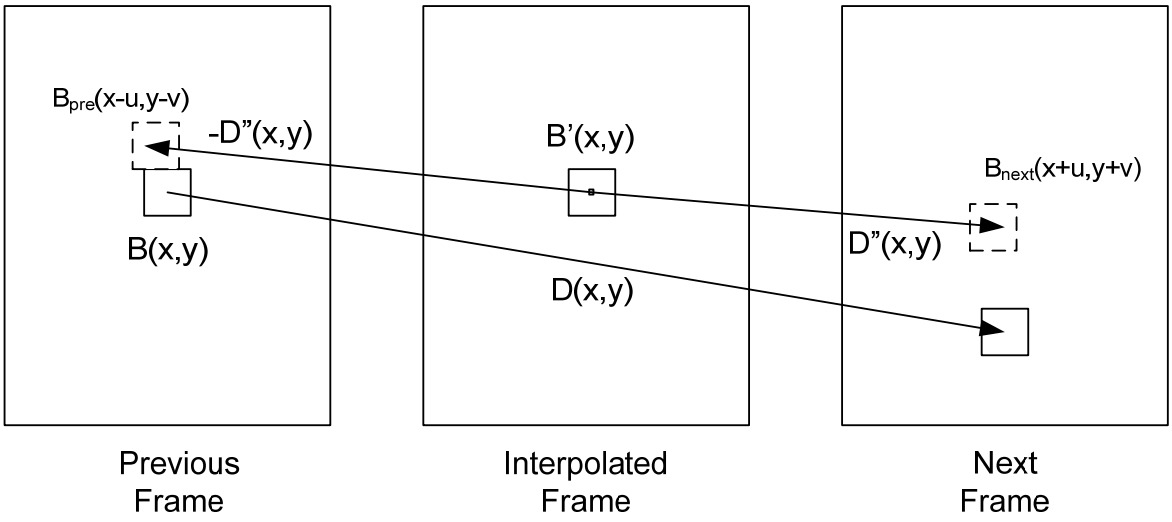

Figure 5.50. Bidirectional Frame Interpolation

The Graceful Degradation based frame interpolation performs a pixel by pixel interpolation. (see Figure 5.51) For each pixel location, the value which will be assigned to the current position is found by a simple median operation:

$$P'(x,y) = \text{median}\left\{P_{pre}(x\text{-}u,y\text{-}v), P_{next}(x+u,y+v), \frac{1}{2}\left(P_{pre}(x,y) + P_{next}(x,y)\right)\right\} \qquad (4.12)$$

where u and v are the half of the horizontal and vertical components of the motion vector D(x,y):

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{D(x,y)}{2} \tag{4.13}$$

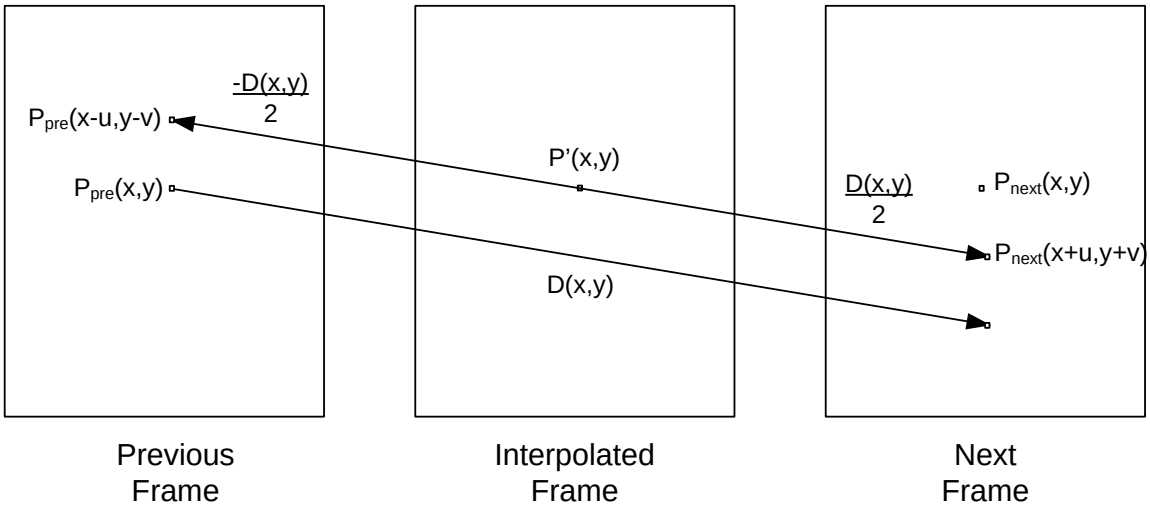

Figure 5.51. Graceful Degradation based frame interpolation

The basic interpolation method transfers the pixels/blocks of previous frame to the interpolated frame by using their corresponding motion vectors and then the positions which are not filled at the end of the operation are interpolated by using the previously discussed Bidirectional or Graceful Degradation methods.

For the possible choices, our observations showed that on the average Bidirectional Interpolation strategy gives the best results.

All the outputs which are given in the other sections of the thesis are created by using the Bidirectional Frame Interpolation method.

5.3.4. Effects of MVF Post-Processing on FRC Performance

In this section the effects of MVF post-processing methods on the performance of the FRC application will be investigated by using the PSNR and SSIM metrics. The PSNR

metric is a variation of previously discussed MSE metric and it is calculated by using the formula:

$$\text{PSNR} = 10\log_{10}\left(\frac{255^2}{\text{MSE}}\right) \tag{4.14}$$

In order to use the PSNR and SSIM metrics, it is necessary to have a reference data. Then, we have sub-sampled our test videos in temporal domain and by using our FRC environment we have interpolated the missing frames by using different configurations.

The following figures give the PSNR plots of different FRC configurations on different video sequences. The "None" option represents the case for which no MVF post-processing operation is applied and the "Joint" option corresponds to the case for which both the refinement and smoothing operations are applied. For other cases, only the defined operation is applied during the processing steps.

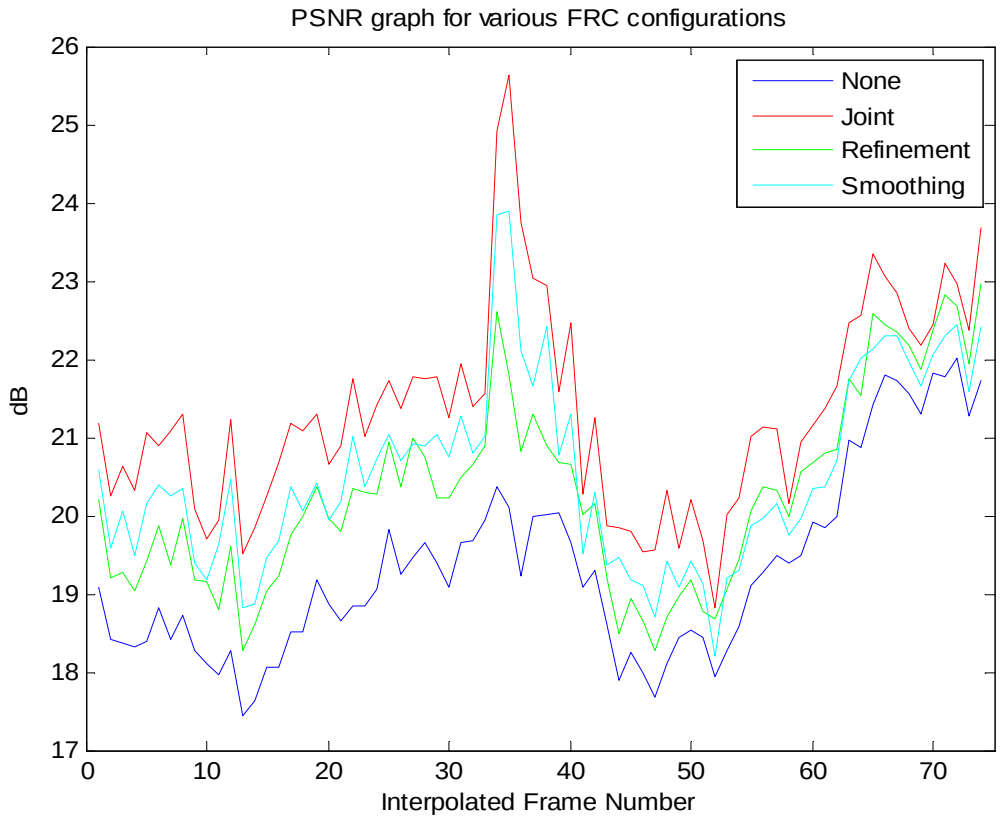

Figure 5.52. PSNR plots for "Bus.avi" sequence

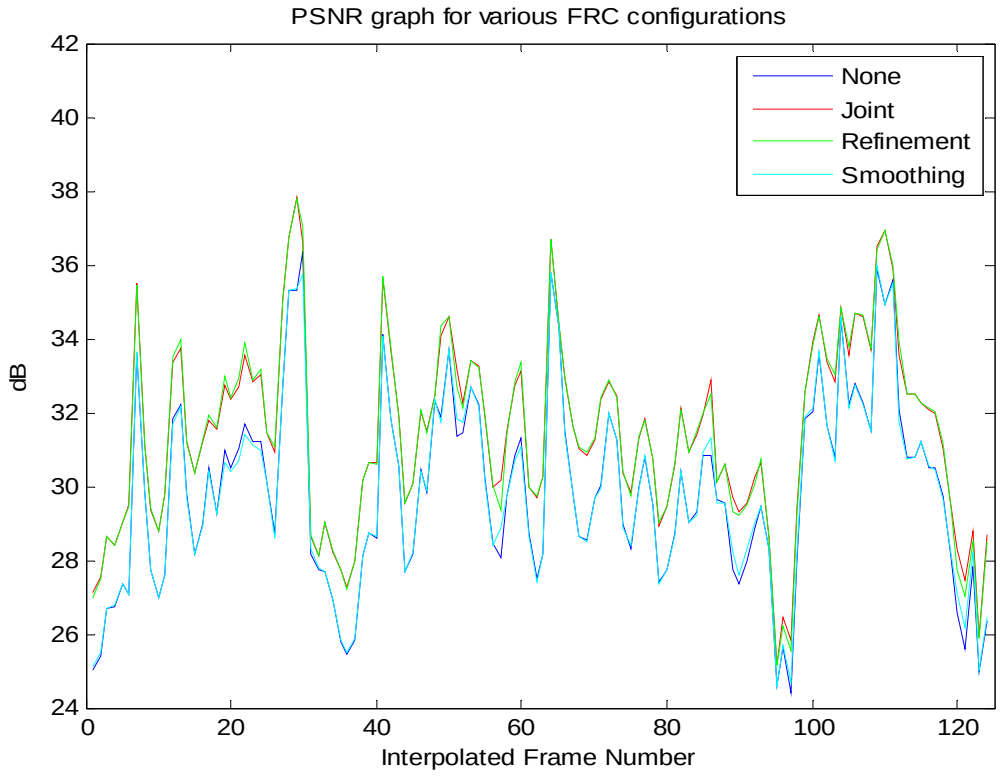

Figure 5.53. PSNR plots for "Foreman.avi" sequence

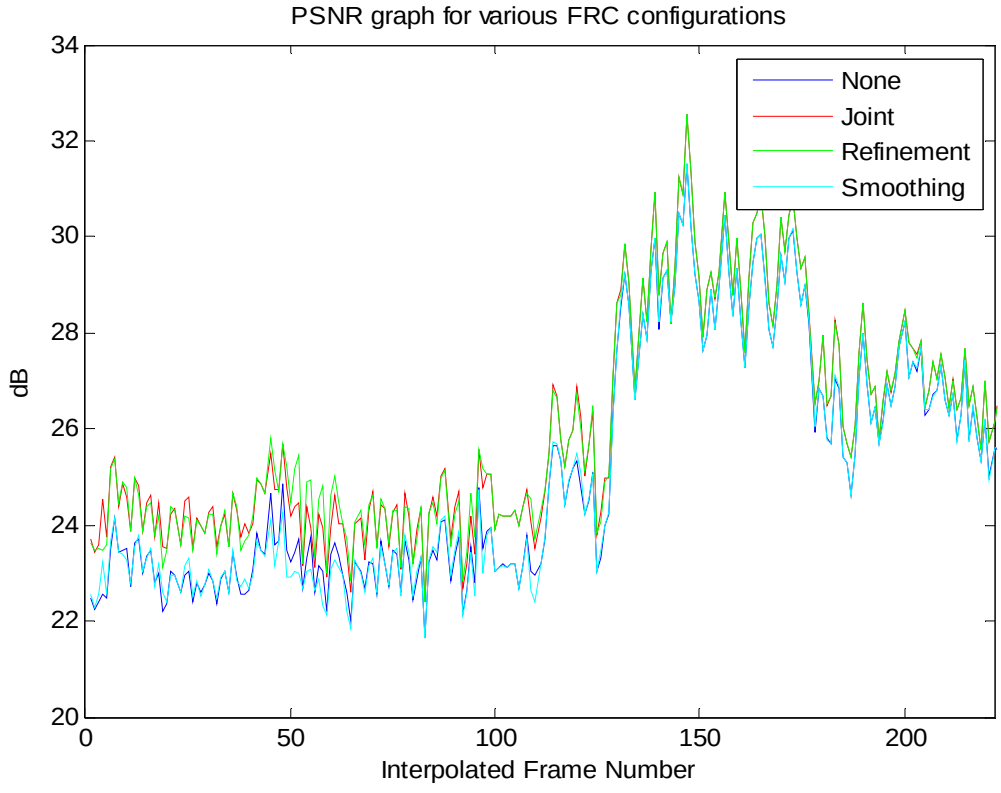

Figure 5.54. PSNR plots for "MobileCalender.avi" sequence

If the Figure 5.52, 5.53 and 5.54 are examined, it is seen that the PSNR scores of the interpolated frames which are created by using the post-processing operations are generally 1,5-2dB higher than the ones which are created without using any post-processing operations. Especially from Figure 5.53 and 5.54, it is seen that the main reason of the PSNR improvement is the *Refinement* operation and the *Smoothing* operation generally does not make any significant change on the PSNR score.

Although a better PSNR score is a desirable property, it does not guarantee to have better visual quality for all the times. In order to evaluate the visual quality we have utilized the structural similarity (SSIM) metric. This metric proposes that the similarity of two contents in terms of their visual quality may be evaluated by using three parameters: average gray levels, variance of the gray levels and the similarity of existing edge structures. The following figures give the obtained results for three distinct test videos.

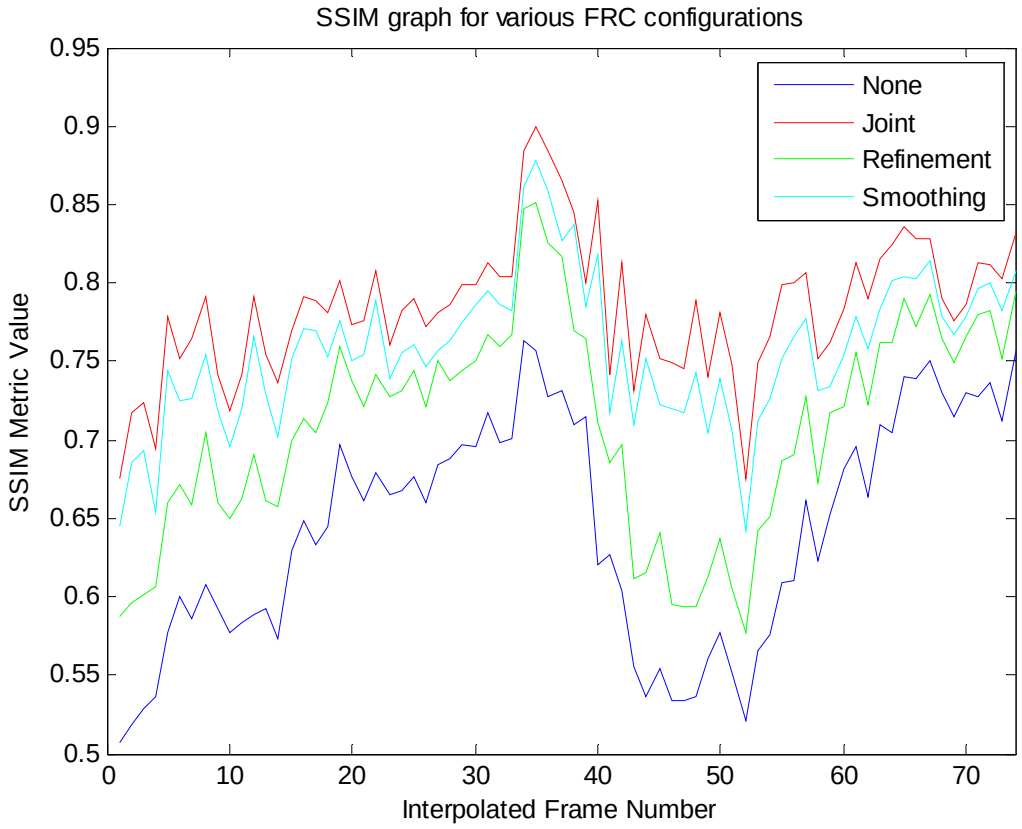

Figure 5.55. SSIM plots for "Bus.avi" sequence

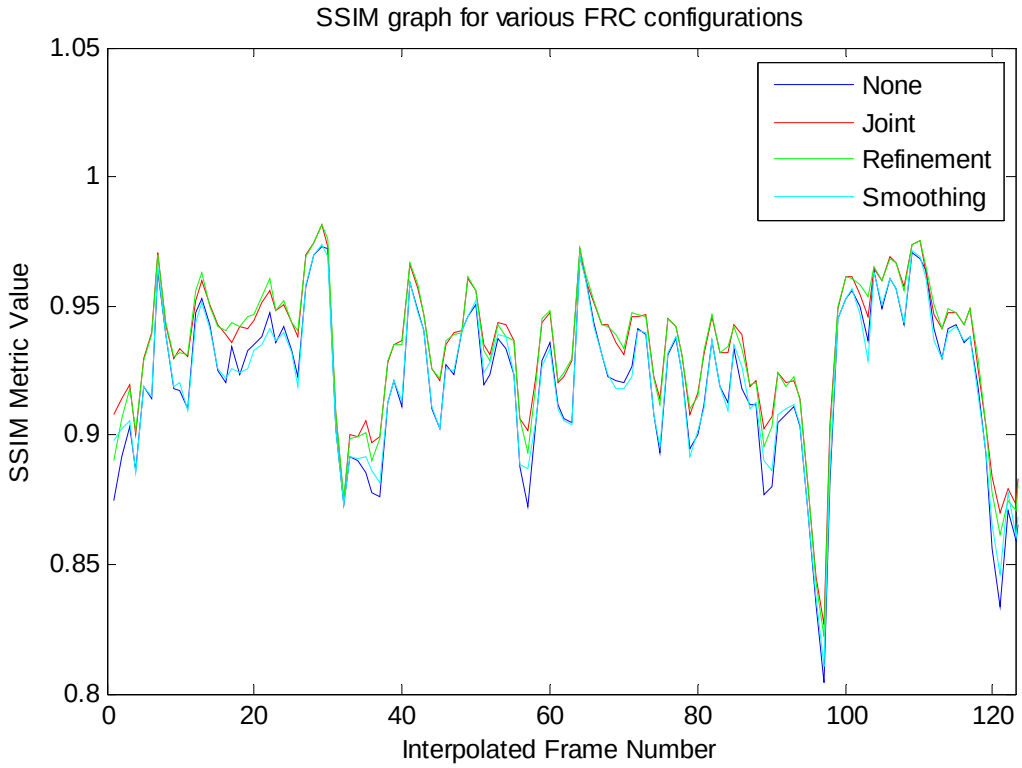

Figure 5.56. SSIM plots for "Foreman.avi" sequence

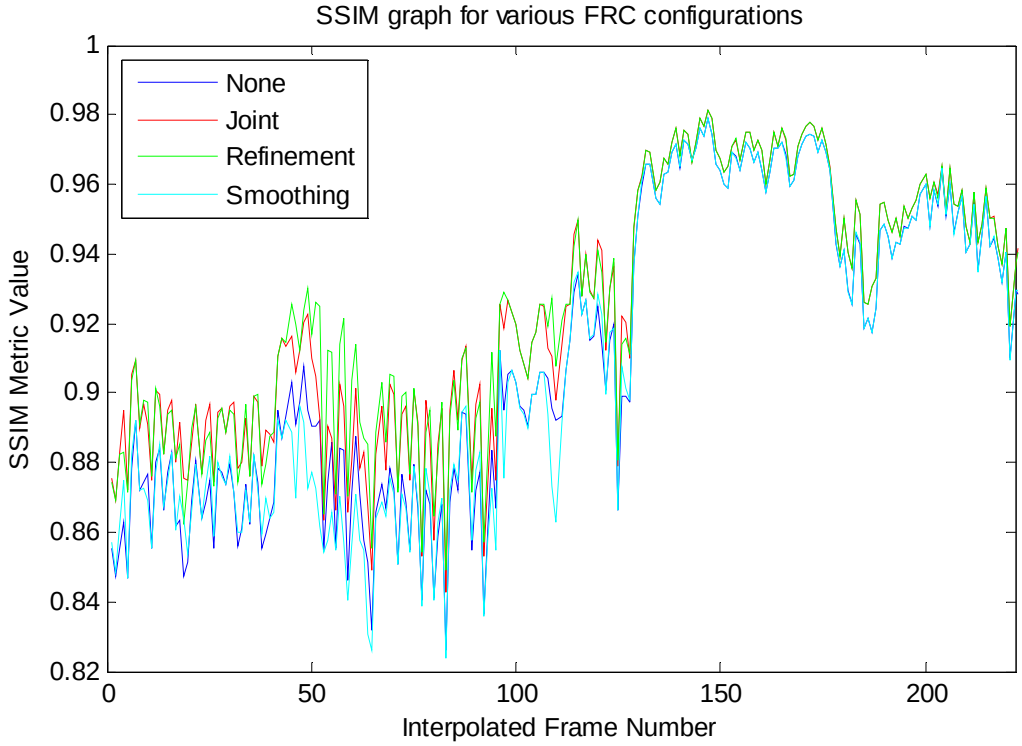

Figure 5.57. SSIM plots for "MobileCalender.avi" sequence

The plots, which show the SSIM metric results for various video sequences, prove that the MVF post-processing operations have a positive effect on the visual quality of the output videos which are created by our FRC testbench. Especially for relatively fast moving scenes, such as ‘Bus.avi’, the SSIM metric score difference between the videos which utilize the post-processing strategies and the ones which do not use any post-processing strategies deviates is very significant.

6. CONCLUSION AND SUGGESTIONS FOR FUTURE WORK

As previously summarized in section 2, there are various motion estimation methods in the literature which have different complexity levels and estimation qualities. Some of these methods focus only on the minimization of utilized matching metric (such as SAD and MSE) without caring about the true motion related issues, whereas the rest target to obtain the best motion vectors in terms of true motion. Actually, these two groups of motion estimators have different application areas. While the cost minimization based estimators are generally utilized in the coding/compression applications (such as MPEG2, MPEG4, H.263 and H.264), the true motion estimators are preferred for the video processing/enhancement applications (such as FRC and DI).

The work on existing motion estimation methods show that the true motion estimation algorithms have a significantly higher computational complexity levels than the cost minimization based motion estimators since they have to deal with some true motion related issues, such as the correlations between neighbor vectors, in addition the matching related ones. On the other hand, most of the consumer electronic devices, which contain video processing steps in their data flow, have very limited computational power and embedding true motion estimators inside them is generally not possible. At that point, utilization of the motion vectors which are available inside the compressed video sequences is a good idea instead of performing a search. However, simulation results show that the direct use of motion vectors which are created by a matching metric minimization based motion estimation method produces disturbing artifacts on the outputs of the considered video processing application such as seen in Figure 5.45, 5.47, 5.48 and 5.49. Our studies also show that the insertion of post-processing blocks in order to smooth and refine the available motion vectors has a positive effect on both of the visual (SSIM) and error (PSNR) metrics since the erroneous vectors or the problematic ones in terms true motion constraints are cleaned up during the post-processing operations. (see figures from 5.52 to 5.57)

The two contributions in this thesis, which are Motion Discontinuity Based Vector Median Filter and Low Complexity Weighted Vector Median Filter, introduced alternative vector field post-processing strategies.

As discussed in sections 4.1 and 5.1.7, the vector median filters are a good compromise between the low complexity methods with poor performances (such as Mean and Median) and the high complexity methods with better performance levels (such as WVMF). However, they have some obvious problems such as the one discussed in section 4.1 and the proposed *Motion Discontinuity Based VMF* successfully solves the considered problem as seen if the Figure 5.14 and 5.20 compared. On the other hand, from the Table 5.3 and 5.4, it is seen that MSE performance of the proposed method is generally lower than the basic VMF. The main reason for this decrease is possibly the wrong motion discontinuity decisions on uniform motion regions and simulations show that although the MSE score is lower the visual performance of the output videos are not affected too much in these regions. Further work on this algorithm should include the investigation of a better discontinuity detector and/or a checking mechanism for measuring the reliability of the detection in order to increase the performance in uniform motion areas.

The second contribution focuses on the Weighted Vector Median Filters and it proposes a simplification strategy (see Figure 4.4 and 4.5) for the basic WVMF method. The achieved simplification level by using the proposed *Low Complexity WVMF* strategy may be judged by looking at Table 4.1. If the simulation results which are given in tables 4.5 and 4.6 are compared it is seen that our approach saves a significant amount of computation power without producing an important performance degradation. Additionally, it is possible to adjust the complexity level of our algorithm by making few simple modifications inside the flow. The outputs of the utilized MSE and Smoothness metrics show that if the complexity level of our algorithm is increased one step further (it still has lower complexity than WVMF), then exactly the same performance level is achieved with the WVMF. Further work on this algorithm may include finding simpler weighting strategies and may also include observing the effects of simplified distance metrics.

To sum up, obtained outputs clearly show that the post-processing of motion vectors, which are created by classical motion estimators, has positive effects on increasing both of objective and subjective evaluation metrics.

APPENDIX A: IMPLEMENTED FRC TESTBENCH ENVIRONMENT

The testbench environment, which is created by using the MATLAB software, is used for simulating the full flow of a typical Frame Rate Conversion application and also it is used for observing/comparing the effects of each sub-block (such as the motion estimation block, the vector field processing block or the motion compensated frame interpolation block) on the overall system performance.

Figure A.1. GUI of implemented FRC Testbench Environment

Figure A.1 shows the GUI of the implemented testbench environment. The GUI has several panels on it and each panel can be used for configuring the different blocks of the FRC application.

Three panels, which are located on the upper part of the GUI, are used for defining the main flow of the processing application. The user is free to choose anyone of the:

- 4 distinct motion estimation methods: Full Search, Hierarchical Search, Hexagonal Search and 3D Recursive Search

- 4 vector field processing strategies: Smoothing&Refinement, only Smoothing, only Refinement and None for no MVF processing

- 4 Frame Rate Conversion techniques: Bidirectional Interpolation, Graceful Degradation Based Interpolation, mixture of Basic & Bidirectional Interpolation and mixture of Basic & Graceful Degradation Based Interpolation

By using “Motion Search Parameters” panel, the user can change the values of the utilized block size and the search range parameters of the motion estimation algorithms.

The next panel (Vector Processing Settings) can be used for selecting the specific MVF smoothing and refinement strategies. The possible smoothing strategies are:

- Mean Filtering: the details can be found in section 3.2.1

- Scalar Median Filtering: the details can be found in section 3.2.2

- Vector Median Filtering (Distance-1): utilizes the total Euclidian distance to all other vectors and is discussed in section 3.2.2

- Vector Median Filtering (Distance-2): utilizes the Euclidian distance between the considered vector and the mean of the input vectors and is discussed in section 3.2.2

- Vector Median Filtering (Distance-3): utilizes the Euclidian distance between the considered vector and the scalar median of the input vectors and is discussed in section 3.2.2

- Vector Median Filtering (Angle): utilizes the total angular distance to all other vectors and is discussed in section 3.2.2

- Vector Median Filtering (Angle&Distance): utilizes the combination of the total Euclidian distance and total angular distance to all other vectors and is discussed in section 3.2.2

- Weighted Vector Median Filtering: the details can be found in section 3.3.2

- Motion Discontinuity Based VMF: this is the first proposed smoothing method and the details of it can be found in section 4.1

- Low Complexity WVMF: this is the second proposed method and the details of it can be found in section 4.2

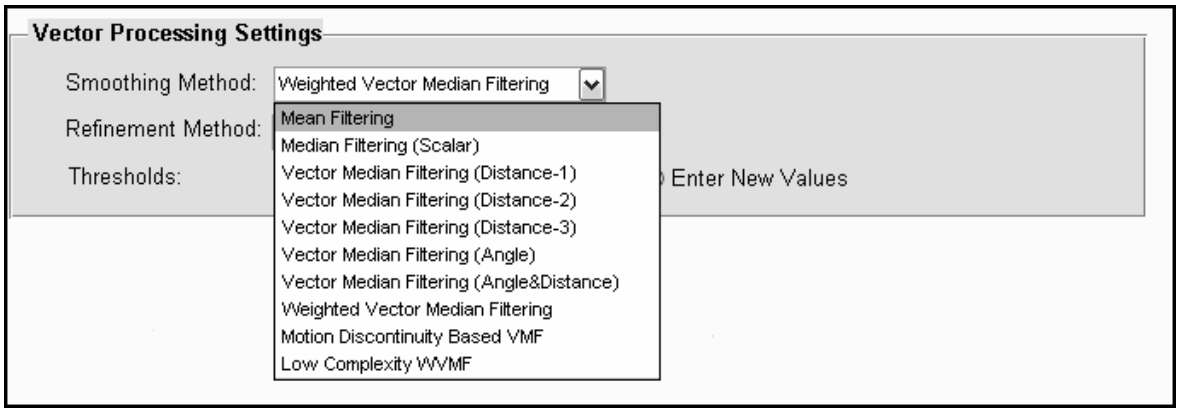

Figure A.2. Listbox for the implemented MVF smoothing methods

Also, the possible refinement strategies are:

- Simple Median : which is discussed in section 4.3 and is given in publication [24]

- Detailed Search : which is discussed in sections 3.3.3 and 4.3, and is given in publication [43]

- Selective Search : which is created from the combination of previous two methods, and the details are given in section 4.3

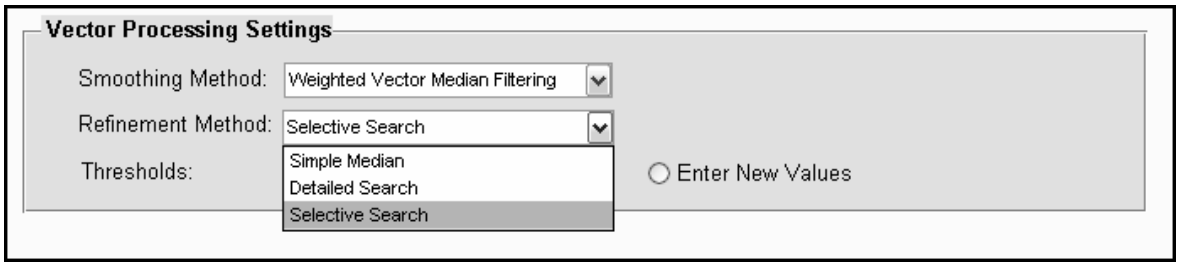

Figure A.3. Listbox for the implemented MVF refinement methods

In addition to the selection of the post-processing strategies, user is free to choose entering new values for the utilized thresholds, which are used for making some decisions during processing operations, by using the same panel.

The last panel is related with the simulation related parameters: the user chooses the input video file by using the browse button and, if he/she wants, it is possible to define the limits for the frames which will be processed during the simulation.

There exist two buttons at the bottom side of the GUI:

- Implemented application stores the configuration of the currently running simulation and by using the 'Load Latest Configuration' button it is possible to restore the lastly used configuration settings and parameter values when the application is started in the next time.

- 'Run' button starts the simulation by using the currently selected configuration and entered parameter values.

APPENDIX B: SYNTHETIC IMAGE SEQUENCES

The synthetic image sequences which are used for the performance evaluation of MVF smoothing methods and their corresponding ground-truth motion information are as follows:

Figure B.1. Synthetic Image Sequence – I

For the image sequence that is seen in Figure B.1, the upper half part of the image sequence moves to the left side and has a motion vector of [0,-3] and the lower half part of the image sequence moves to the right side with a motion vector of [0,2]. Additionally, the square block which is located at the center of the images has no motion and its motion vector is [0,0]. The corresponding ground-truth motion vector field is given in Figure B.4.

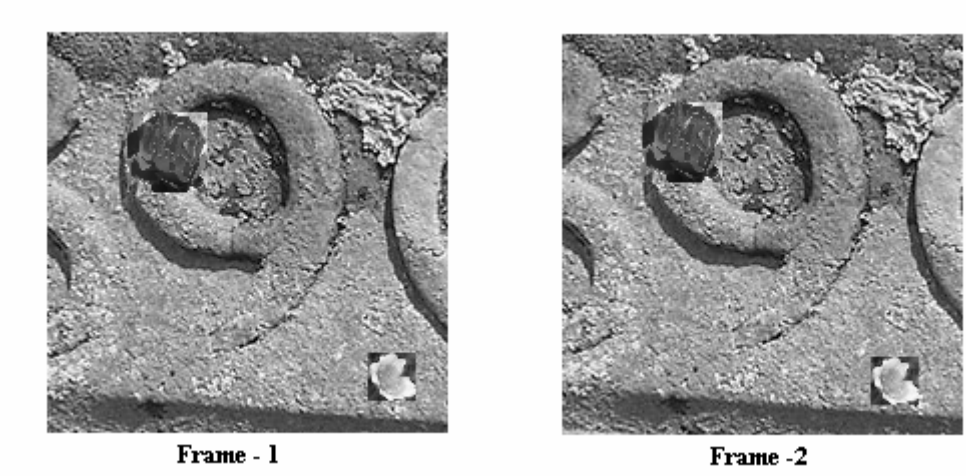

Figure B.2. Synthetic Image Sequence – II

For the second image sequence the background has a motion in north-east direction with a motion vector of [-3,4]. Also, there exist two small objects with different motion characteristics. The red colored object moves with a motion vector of [-6,0], and the motion vector of the other object is [1,-6]. The corresponding ground-truth motion vector field is given in Figure B.5.

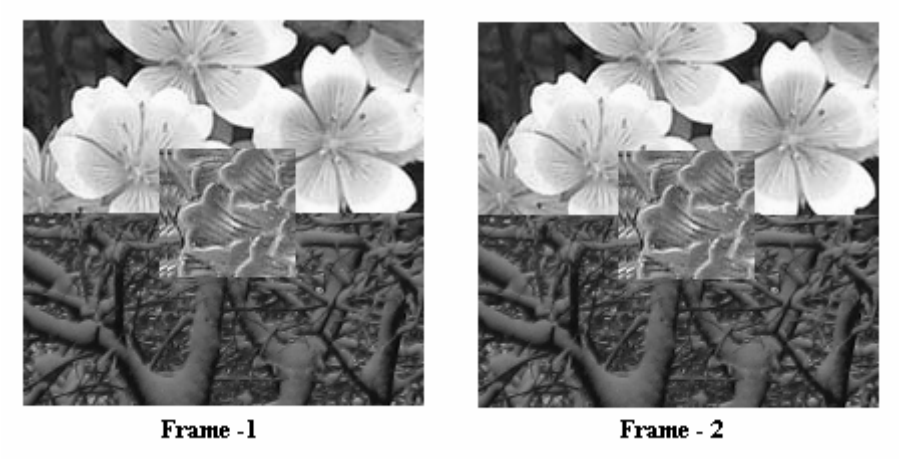

Figure B.3. Synthetic Image Sequence – III

The third image sequence contains three different motion characteristics. The upper half part of the image has a motion vector of [-3,2] and the motion vector for the lower half part is [-1,0]. The block which is located at the center of the image moves toward the right with a motion vector of [0,2]. The corresponding ground-truth motion vector field is given in Figure B.6.

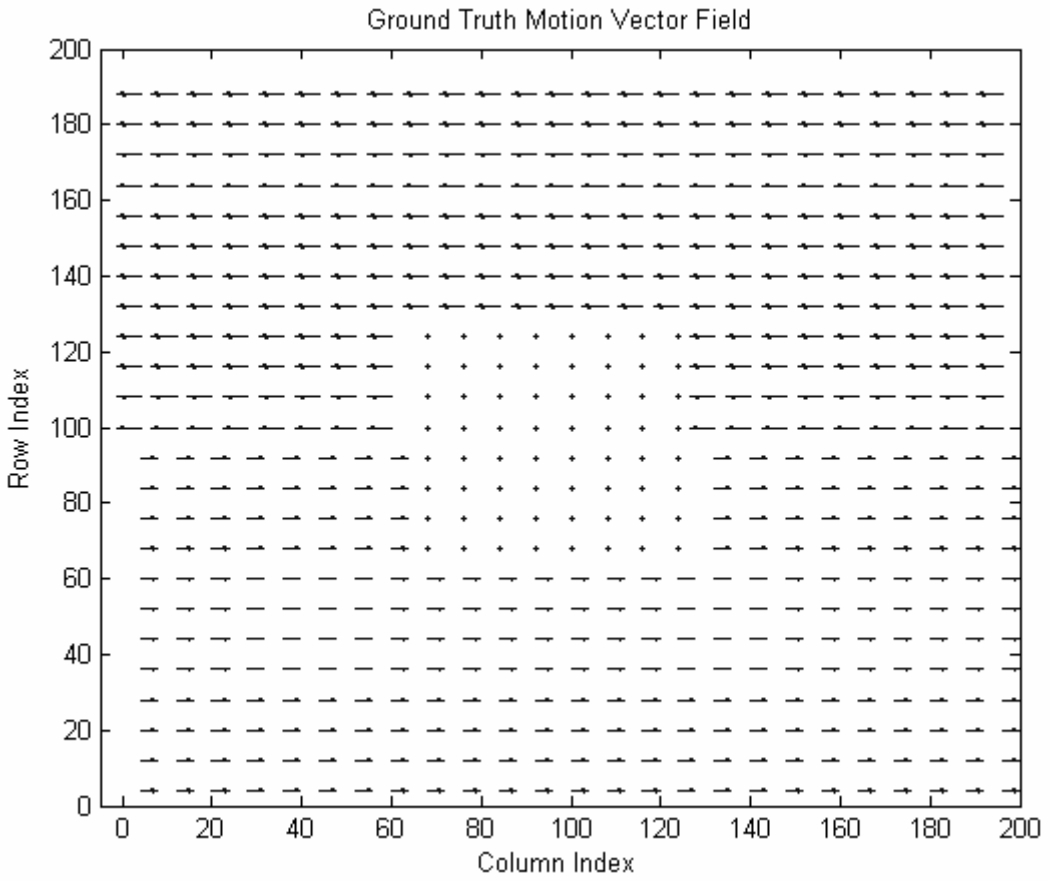

Figure B.4. True motion vectors for Synthetic Image Sequence – I

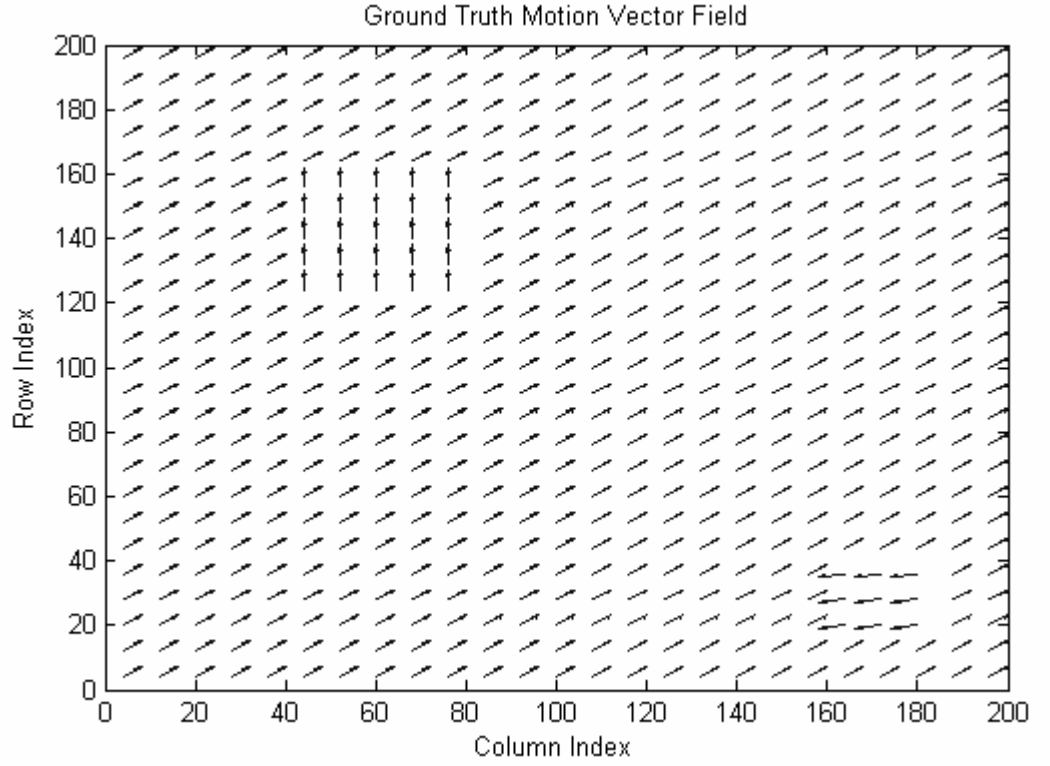

Figure B.5. True motion vectors for Synthetic Image Sequence – II

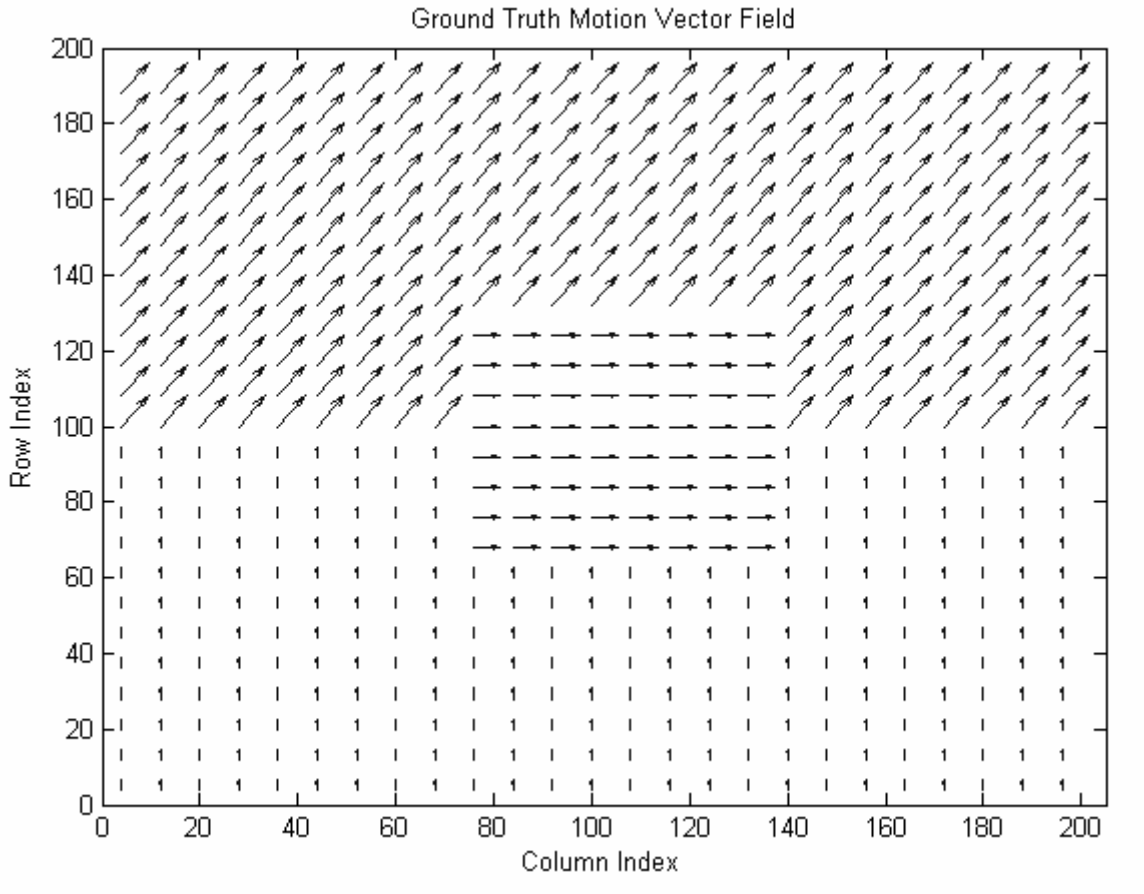

Figure B.6. True motion vectors for Synthetic Image Sequence – III

REFERENCES

1. Dufaux, F. and F. Moscheni, “Motion Estimation Techniques for Digital TV: A Review and a New Contribution”, *Proceedings of IEEE*, Vol. 83, No. 6, 1995.

2. Zaccarin, A. and B. Liu., "Fast algorithms for motion estimation", *IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. III, pp. 449-452, March 1992.

3. Heijden, H., F. Wenzel and R. Grigat, “Strategies for Fast True Motion Block Matching”, *1st International Conference on Computer Vision Theory and Application (VISAPP)*, Vol. 2, pp. 359-363, 2006.

4. Koga, T., K. Iinuma, A. Hirano, Y. Iijima, and T. Ishiguro, “Motion compensated interframe coding for video conferencing”, *Proceedings of National Telecommunications Conference*, pp. 961-965, New Orleans, LA, Nov./Dec. 1981.

5. Li, R., B. Zeng and M. Liou, “A New Three-Step Search Algorithm for Block Motion Estimation”, *IEEE Trans. Circuits And Systems For Video Technology*, Vol. 4, No. 4, pp. 438-442, August 1994.

6. Po, L. and W. C. Ma, “A Novel Four-Step Search Algorithm for Fast Block Motion Estimation”, *IEEE Trans. on Circuits and Systems for Video Technology,* Vol. 6, No. 3, pp. 313-317, June 1996.

7. Jain, R. and K. Jain, “Displacement Measurement and Its Application In Interframe Image Coding”, *IEEE Transactions on Communication*, Vol. 29, No. 12, December 1981.

8. Zhu, S. and K. Ma, “A new diamond search algorithm for fast block matching motion estimation”, *IEEE International Conference on Information, Communications and Signal Processing*, pp. 292–296, Sept., 1997.

9. Cheung, C. H. and L. M. Po, "A Novel Cross-Diamond Search Algorithm for Fast Block Motion Estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, No. 12, Dec. 2002.

10. Jia, H. and L. Zhang, "A New Cross Diamond Search Algorithm for Block Motion Estimation", *IEEE Proceedings of Acoustics, Speech, and Signal Processing*, Vol. 3, pp. 357-360, 2004.

11. Zhu, C., X. Lin and L. Chau, "Hexagon-based search pattern for fast block motion estimation", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 12, pp. 349–355, May 2002.

12. Zhu, C., X. Lin and L. Chau, "An Enhanced hexagonal search algorithm for block motion estimation", *IEEE International Symposium on Circuits and Systems*, Vol. 2, pp. 392-395, May 2003.

13. Barron, J. L., D. J. Fleet, and S.S. Beauchemin, "Performance of Optical Flow Techniques", *International Journal of Computer Vision*, Vol. 12, pp. 43-77, 1994.

14. Horn, B. K. P. and B. G. Schunk, "Determining Optical Flow", *Artificial Intelligence*, Vol. 17, pp. 185-204, 1981.

15. Lucas, B. and T. Kanade, "An Iterative Image Registration Technique with Applications in Stereo Vision", *Proceedings of the DARPA Image Understanding Workshop*, pp. 121-130, 1981.

16. Anandan, P., "A Computational Framework and an Algorithm for the Measurement of Visual Data", *International Journal of Computer Vision,* Vol. 2, pp. 283-310, 1989.

17. Singh, A. “An Estimation-Theoretic Framework for Image-Flow Computation”, *IEEE Proceedings of International Conference on Computer Vision,* pp.168-177, 1990.

18. Singh, A. “Optic Flow Computation: A Unified Perspective”, *IEEE Computer Society Press,* 1992.

19. Heeger, D. J., “Model for Extraction of Image Flow”, *Journal of the Optical Society of America,* pp. 1455-1471, 1987.

20. Heeger, D. J., “Optical Flow Using Spatiotemporal Filters”, *International Journal of Computer Vision,* pp. 279-302, 1988.

21. Fleet, D. J. and A. D. Jepson, “Computation of Component Image Velocity from Local Phase Information”, *International Journal of Computer Vision,* pp. 77-107, 1990.

22. Fleet, D. J., *Measurement of Image Velocity*, Kluwer Academic Publishers, Norwell, 1992.

23. Fleet, D. J. and A. D. Jepson, “Stability of Phase Information”, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 15, pp. 1253-1268, 1993.

24. Haan, G. D. and A. C. Biezen, “True-Motion Estimation with 3-D Recursive Search Block Matching”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 5, Oct. 1993.

25. Haan, G. D. and A. C. Biezen, “An Efficient True-Motion Estimator Using Candidate Vectors from a Parametric Motion Model”, *IEEE Transactions on Circuits and Systems for Video Technology*, Vol.:8, No:1, Feb. 1998.

26. Braspenning, R. and G. D. Haan, "True-Motion Estimation Using Feature Correspondences", *Proceedings of SPIE on Visual Communications and Image Processing*, pp. 396-407, 2004.

27. Heijden, H., F. and R. R. Grigat, "Strategies For Fast True Motion Block Matching", *International Conference on Computer Vision Theory and Applications*, Vol. 2, pp. 359-363, 2006.

28. Yin, H. B., X. Z. Fang, H. Yang, S. Y. Yu and X. K. Yang, "Motion Vector Smoothing For True Motion Estimation", *IEEE Proceedings of Acoustics, Speech, and Signal Processing*, Vol. 2, 2006.

29. Elias, D. P. and N. G. Kingsbury, "An Efficient Block Segmentation Algorithm For True Motion Estimation", *IEEE International Conference on Image Processing and Its Applications*, Vol. 1, July 1997.

30. Chen, Y. K., Y. T. Lin and S. Y. Kung, "A Feature Tracking Algorithm Using Neighborhood Relaxation With Multi-Candidate Pre-Screening", *IEEE Proceedings of International Conference on Image Processing*, Vol. 2, Sep. 1996.

31. Barnett, V., "The Ordering of Multivariate Data", *Journal of the Royal Statistical Society*, Vol.139, pp. 318-354, 1976.

32. Astola, J., P. Haavisto and Y. Neuvo, "Vector Median Filters", *IEEE Proceedings*, Vol. 78, No. 4, 1990.

33. Barni, M. and V. Capellini, "On the Computational Complexity of Multivariate Median Filters", *Signal Processing*, Vol. 71, No: 1, pp. 45-54, 1998.

34. Evans, A. N., "Vector Area Morphology for Motion Field Smoothing and Interpretation", *IEE Proceedings in Vision, Image and Signal Processing*, Vol. 150, pp. 219-226, 2003.

35. Westenberg, M. A. and T. Ert, "Vector Wavelet Thresholding for Vector Field Denoising", *IEEE Proceedings of the Conference on Visualization*, Vol. 10, pp. 25-26, 2004.

36. Kelly, A. R. and E. R. Hancock, "Vector Field Smoothing Via Heat Flow", *IEEE International Conference on Pattern Recognition*, Vol. 2, pp. 94-97, 2004.

37. Bednar, J. B. and T. L. Watt, "Alpha-Trimmed Means and Their Relationship to Median Filters", *IEEE Transactions on Acoustics, Speech and Signal Processing,* Vol. 32, pp. 145-153, 1984.

38. Dane, G. and T. Q. Nguyen, "Motion Vector Processing for Frame Rate Up Conversion", *IEEE International Conference on Acoustics, Speech and Signal Processing,* Vol. 3, pp. 309-312, 2004.

39. Alparone, L., M. Barni, F. Bartolini and V. Capellini, "Adaptively Weighted Vector Median Filters for Motion-Fields Smoothing", *IEEE Proceedings of Acoustics, Speech and Signal Processing,* Vol. 4, pp. 2267-2270, 1996.

40. Alparone, L., M. Barni, F. Bartolini and R. Caldelli, "Regularization of Optic Flow Estimates by Means of Weighted Vector Median Filtering", *IEEE Transactions on Image Processing,* Vol. 8, pp. 1462-1467, 1999.

41. Ascenso, J., C. Brites and F. Pereira, "Improving Frame Interpolation with Spatial Motion Smoothing for Pixel Domain Distributed Video Coding", *EC-SIP-M,* 2005.

42. Blume, H., G. Herczeg, O. Erdler and T. G. Noll, "Object Based Refinement of Motion Vector Fields Applying Probabilistic Homogenization Rules", *IEEE Transactions on Consumer Electronics,* Vol. 48, No. 3, pp. 694-701, 2002.

43. Ching, S. and F. Hu, *Refinement of Block Motion Vectors to Achieve a Dense Motion Field*, US Patent, Patent Number: 5748247, 1998.

44. Pal, N. R. and S. K. Pal, "A review on image segmentation techniques", *Pattern Recognition,* Vol. 26, pp. 1277-1294, 1993.

45. Haan, G. D., *Video Processing For Multimedia Systems*, University Press, Third Edition, Eindhoven, 2003.

46. Wang, Z., L. Lu and A. C. Bovik, "Video Quality Assessment Using Structural Distortion Measurement", *IEEE International Conference on Image Processing*, Vol. 3, pp. 65-68, 2002.

REFERENCES NOT CITED

Haan, G. D., *Motion Estimation and Compensation: An Integrated Approach To Consumer Display Field-Rate Conversion,* Natuurkunding Laboratorium, ISBN: 90-74445-01-2, September 1992.

Chen, Y. K., *True Motion Estimation – Theory, Application, and Implementation*, Ph.D. Thesis, Princeton University, November 1998.