2008

Video modeling via implicit motion representations

Yunfei Zheng
West Virginia University

Video Modeling via Implicit Motion Representations

Yunfei Zheng

Dissertation submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Electrical Engineering

Xin Li, Ph.D., Chair
Donald A. Adjeroh, Ph.D.
Oscar D. Escoda, Ph.D.
Arun A. Ross, Ph.D.
Natalia A. Schmid, Ph.D.

Department of Computer Science and Electrical Engineering

Morgantown, West Virginia
2008

Keywords: Video Modeling, Implicit Motion Representations, Video Processing

ABSTRACT

Video Modeling via Implicit Motion Representations

Yunfei Zheng

Video modeling refers to the development of analytical representations for explaining the intensity distribution in video signals. Based on the analytical representation, we can develop algorithms for accomplishing particular video-related tasks. Therefore video modeling provides us a foundation to bridge video data and related-tasks. Although there are many video models proposed in the past decades, the rise of new applications calls for more efficient and accurate video modeling approaches.

Most existing video modeling approaches are based on explicit motion representations, where motion information is explicitly expressed by correspondence-based representations(i.e., motion velocity or displacement). Although it is conceptually simple, the limitations of those representations and the suboptimum of motion estimation techniques can degrade such video modeling approaches, especially for handling complex motion or non-ideal observation video data. In this thesis, we propose to investigate video modeling without explicit motion representation. Motion information is implicitly embedded into the spatio-temporal dependency among pixels or patches instead of being explicitly described by motion vectors.

Firstly, we propose a parametric model based on a spatio-temporal adaptive localized learning (STALL). We formulate video modeling as a linear regression problem, in which motion information is embedded within the regression coefficients. The coefficients are adaptively learned within a local space-time window based on LMMSE criterion. Incorporating a spatio-temporal resampling and a Bayesian fusion scheme, we can enhance the modeling capability of STALL on more general videos. Under the framework of STALL, we can develop video processing algorithms for a variety

of applications by adjusting model parameters(i.e., the size and topology of model support and training window). We apply STALL on three video processing problems. The simulation results show that motion information can be efficiently exploited by our implicit motion representation and the resampling and fusion do help to enhance the modeling capability of STALL.

Secondly, we propose a nonparametric video modeling approach, which is not dependent on explicit motion estimation. Assuming the video sequence is composed of many overlapping space-time patches, we propose to embed motion-related information into the relationships among video patches and develop a generic sparsity-based prior for typical video sequences. First, we extend block matching to more general kNN-based patch clustering, which provides an implicit and distributed representation for motion information. We propose to enforce the sparsity constraint on a higher-dimensional data array signal, which is generated by packing the patches in the similar patch set. Then we solve the inference problem by updating the kNN array and the wanted signal iteratively. Finally, we present a Bayesian fusion approach to fuse multiple-hypothesis inferences. Simulation results in video error concealment, denoising, and deartifacting are reported to demonstrate its modeling capability.

Finally, we summarize the proposed two video modeling approaches. We also point out the perspectives of implicit motion representations in applications ranging from low to high level problems.

Acknowledgments

Many people have inspired, guided, helped and support me during the four and half years I spent at West Virginia University, and I would like to thank them all for a great graduate school experience. I first want to thank Professor Xin Li, my graduate advisor. It was he who started me on my path as a researcher on image and video processing. His encouragement and advices were paramount in providing a well rounded experience consistent with my long-term career goals. I think I was fortunate enough to be a student who can receive Ph.D. training under his guidiance.

I also would like to thank all the faculty members in Lane Department of Computer Science and Electrical Engineering for providing an excellent education and research environment. Especially, I want to thank Professor Natalia Schmid, Professor Arun Ross and Professor Donald Adjero, who were my dissertation committee. Their careful reading and inspiring suggestions drove me to dig deeper in my research topics. I also thank Dr. Oscar Divorra Escoda and Dr. Peng Yin in Thomson Inc. for arranging one year internship for me. Their mentorship helped me a lot to explore deeper and more interesting research topics. Dr. Oscar Divorra Escoda deserve a special note of appreciation for his careful guidaince on my dissertation as a committee member.

I must also thank all the students and friends in West Virginia University for their assistance on my research and life in Morgantown. Especially, I would like to thank Dr. Congxia Dai, who gave me countless helps in my research, life, and job

hunting. I really owe him my heartfelt appreciation. I also want to thank Qiang Hao, Lierong Zhu, Luyi Wang, and Xu Geng for their warmhearted helps and patiences to be audiences of my defense rehearsals.

I owe a huge debt of gratitude to my parents for their love, support, and encouragement throughout not only the course of my Ph.D. studies, but throughout my entire life. Thanks Mom and Dad.

Table of Contents

Abstract ii

Acknowledgments iv

List of Tables viii

List of Figures x

1 Introduction 1

1.1 Video Modeling by Explicit Motion Representations 3

1.2 Video Modeling by Implicit Motion Representations 9

1.3 Contributions 13

2 Video Modeling via Spatio-Temporal Adaptive Localized Learning 16

2.1 Introduction 16

2.2 Duality between 2D image contour and 3D motion trajectory 17

2.3 Video Modeling via STALL 20

2.3.1 From Explicit to Implicit Motion Representation 20

2.3.2 Model Parameter Setting to Capture Motion 25

2.4 Video Modeling by Spatio-Temporal Resampling and Bayesian Fusion 29

2.4.1 Distributed representation of a video 29

2.4.2 Bayesian Fusion of Linear Regression Results 32

2.5 Conclusion 35

3 Applications of STALL **37**

3.1 Introduction 37

3.2 Video Error Concealment 39

3.3 Video Denoising 44

3.4 Video Super-Resolution 47

3.4.1 Motion trajectory orientation resolution invariance 51

3.4.2 Super-Resolution Algorithm 52

3.4.3 Experimental Results in Super resolution 54

3.5 Conclusion 55

4 Video Modeling via Patch-based Sparse Representation **59**

4.1 Introduction 59

4.2 Patch Clustering 62

4.3 Sparsity Representation 66

4.4 Bayesian Inference with Sparsity-Based Priors 69

4.5 Model Structure 73

4.6 Conclusion 74

5 Applications of Patch-based Sparsity model **76**

5.1 Introduction 76

5.2 Video Error Concealment 77

5.3 Video Denoising 81

5.3.1 Impulse Noise Removal 82

5.3.2 Guassian Noise Removal 83

5.4 DeArtifacting 86

5.5 Conclusion 91

6 Conclusions and Perspectives **95**

6.1 Conclusions 95

6.2 Future Work on the Proposed Models 96

6.3 Perspectives on Implicit Motion Representations 98

Bibliography **100**

List of Tables

3.1 PSNR performance comparisons in error concealment - 1 42

3.2 PSNR performance comparisons in error concealment - 2 43

3.3 PSNR performance comparison in denoising 48

3.4 Spatial and temporal dependency tradeoff 54

5.1 PSNR performance comparison in error concealment 80

5.2 PSNR performance of nonparametric model in denoising 83

5.3 PSNR performance of nonparametric model in denoising 85

5.4 PSNR improvement by patch-based sparsity model in deartifacting . 94

List of Figures

1.1 Example of video frames and optical flow 5

1.2 Block Matching Algorithm 6

2.1 Duality between edge contour and motion trajectory 19

2.2 Iso-intensity constraint along motion trajectory 19

2.3 Motion trajectory and point displacement 20

2.4 Example of the filter support topology of STALL 22

2.5 Topology of training window parameters 25

2.6 PSNR evolution with variations of training window parameters 26

2.7 Localization vs video resolution 27

2.8 Model parameters (filter support and training window) 28

2.9 Layered representation of a video frame 28

2.10 Example of spatio-temporal resampling 30

2.11 Real Example of spatio-temporal resampling 32

2.12 Distributed representation of a video provided by resampling 33

2.13 Modeling procedure by resampling and fusion 35

3.1 Examples of model parameters settings 38

3.2 Visual quality comparison in error concealment – *container* 42

3.3 Visual quality comparison in error concealment – *mobile* 43

3.4 Visual quality comparison in error concealment – *garden* 44

3.5 Denoising performance vs impulse noise percentage 46
3.6 Visual quality comparison in denoising – *coastguard* 47
3.7 Visual quality comparison in denoising – *tennis* 48
3.8 Visual quality comparison in denoising – *bus* 49
3.9 The relationship between high resolution and low resolution frame . . 50
3.10 Three typical steps in super-resolution problem 50
3.11 Correspondence between LR and HR covariance in 2D and 3D 52
3.12 PSNR performance comparisons in super resolutionv - 1 55
3.13 Visual quality comparisons in super resolution 57
3.14 PSNR performance comparison in super resolution - 2 58

4.1 Video and Patch 63
4.2 Example of change sensitivity 64
4.3 Accuracy evaluation of 2D and 3D patches 66
4.4 The overlapping possiblilty comparison between 2D and 3D 67
4.5 Enforcing sparsity constraint in patch level 69
4.6 Source of multiple hypotheses of a pixel 70
4.7 Patch-based sparsity video modeling 74

5.1 Visual quality comparison of error concealment - *tennis* 80
5.2 Visual quality comparison of error concealment - *carphone* 81
5.3 PSNR evolution curve in impulse noise removal 83
5.4 Visual quality comparison of impulse noise removal - *foreman* 84
5.5 Visual quality comparison of impulse noise removal - *tennis* 85
5.6 Visual quality comparison of gaussian noise removal - *coastguard* . . . 87
5.7 Visual quality comparison of gaussian noise removal - *garden* 88
5.8 Examples of video coding artifacts 89
5.9 Visual quality comparison in deartifacting - *foreman* 91

5.10 Visual quality comparison in deartifacting - *silent* 92
5.11 Visual quality comparison in deartifacting - *paris* 93

Chapter 1

Introduction

Video modeling is to build analytical models for explaining the intensity distribution in video signals. Based on the models, we can develop algorithms for accomplishing particular video-related tasks, in which the information offered by video can be processed, analyzed, and exploited. Unlike an image where only spatial information is recorded, a video captures a dynamic evolution of a scene, in which motion is often included. As we have known, motion plays crucial roles in video related applications which range from low level to high level vision tasks[1]. Efficiently and reliably exploiting motion information is therefore pursued by most video modeling approaches.

A task of video modeling is to find a suitable motion representation so as to exploit motion information efficiently. A large amount of video modeling approaches employ the correspondence-based motion representations, which is motivated by the physical representation of motion. Specifically, the motion of a point is represented

[1]The low level vision tasks include all tasks that directly process, exploit or deal with local pixel intensity, color, orientation, and scale. For example, denoising, interpolation, and low level compression belongs to this category. At the other extreme end, the high level vision tasks include the tasks to understand or analyze the content or objects inside an image or video, e.g. recognition, scene interpretation or understanding. The tasks lie between these two ends belong to mid-level vision tasks, which, for example, include segmentation, edge detection, and etc.[1]

by its velocity or displacement vector, which points to its correspondence in another frame[2]. The velocity or displacement field can be estimated by certain motion estimation(ME) techniques in order to serve for video modeling. Although this explicit representation is consistent with the format in physics, it is not always the most efficient or accurate way to describe motion recorded by the digitized source. In a lot of cases, the correspondence-based representations can even be the source of difficulties for exploiting motion information[3]. Furthermore, the ME which supports the representation is still an open problem. Thus video models with such explicit motion representations can probably achieve a suboptimal solution in many cases.

Fortunately, when serving for some applications in which the motion vector field is not a necessary output, such explicit representation and the corresponding ME tend to be unnecessary and can be avoided. This observation motivates the other class of video approaches which do not depend on explicit motion representations. In these video models, although motion is not explicitly described by a velocity or displacement vector, it can also be implicitly exploited by embedding into spatio-temporal dependency in pixels or patches. Such video models circumvent the error-prone correspondence-based representations and the suboptimal motion estimation. Therefore they exhibit potentials to handle more general video signals especially in low level vision tasks.

In this chapter, we will give a brief overview of the two classes of video modeling approaches in which motion is differently represented. Then we present a sketchy description of our contributions in developing video models with implicit motion representations.

1.1 Video Modeling by Explicit Motion Representations

Explicit motion representation aims at describing the motion in video by velocity or displacement vector field. The estimation of motion field is often assumed to be the essential step of video modeling with explicit motion representations. With the motion field, motion information is then exploited for various applications. In order to discuss the limitation of the explicit motion representation, it is necessary to know how motion field can be estimated by the ME techniques.

Most motion estimation techniques are derived from the following brightness constancy constraint,

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{1.1}$$

where I is the intensity value of a scene point in a frame; x,y and t are the spatial and temporal coordinates in a video sequence; $(\delta x, \delta y)$ are spatial location change of the point caused by either camera or object motion, δt is the time interval between two frames taken at time t and $t + \delta t$. Eq.(1.1) shows the intensity value of a scene point in the imaging plane keeps constant in different time slots even though its location changes. A lot of motion estimation methods are generated from the constancy constraint in Eq.(1.1). We can classify them into differential-based and matching-based techniques.

One example of differential based ME techniques is the known Lucas-Kanade method[4]. It tries to calculate the motion between two image frames which are taken at time t and $t + \delta t$ at each pixel position. Assuming motion are small between two consecutive frames, Eq.(1.1) can be easily transformed into an optical flow equation (OFE) by taking the first order *Taylor* expansion:

$$I_t + \nabla I \cdot [V_x \ V_y] = 0 \tag{1.2}$$

where $I_t = \frac{\partial I}{\partial t}$, $\nabla I = [\frac{\partial I}{\partial x} \ \ \frac{\partial I}{\partial y}]$, $V_x = \frac{\partial x}{\partial t}$ and $V_y = \frac{\partial y}{\partial t}$ are the velocities along two spatial directions. In this method, motion is explicitly represented by the velocity vectors $[V_x \ \ V_y]$. Since there are two unknown variables but just one equation, it is an ill-posed problem[2] to solve V_x and V_y directly by Eq.(1.2). Fortunately, we observed that the motion field is generally locally smooth, that is, surrounding pixels of (x, y, t) have similar motion characteristics. We can incorporate the motion smoothness constraint into Eq.(1.2) to make the problem well-posed. Then the problem is reduced to a Least Square problem to estimate $[V_x \ \ V_y]$. By solving the velocities of all pixels in a video frame, we can obtain a so called "optical flow" field. Fig.1.1 shows an example of the optical flow estimated by this method.

The above Lucas-Kanade method works well on texture and edge corners. But it produces large estimation error in the following cases. When motion is relatively large which makes the small motion assumption invalid, *Taylor* expansion does not hold, which invalidates Eq.(1.2). In this case, we can introduce the multiscale idea to hierarchically apply Lucas-Kanade method to solve this problem[5]. In another case where the surrounding pixels have different motions (e.g., moving object boundary), the spatial motion smoothness constraint does not hold. This case motivates the regularization based methods, in which a global constraint of smoothness is introduced to solve such problem[6]. In even worse case, the brightness constancy constraint is not satisfied. For example, the illumination happens to change frame to frame. In such case, the differential-based method cannot estimate the optical flow correctly. We can opt to matching-based motion estimation methods.

One example of matching based ME techniques is based on feature tracking. In these approaches, deterministic parametric motion models are usually chosen, such

[2]The mathematical term well-posed problem is defined by Hadamard as a problem where there exists a unique solution and the solution of which dependes continuously on the data in some reasonable topology. Problmes that are not well-posed in the sense of Hadamard are termed ill-posed.

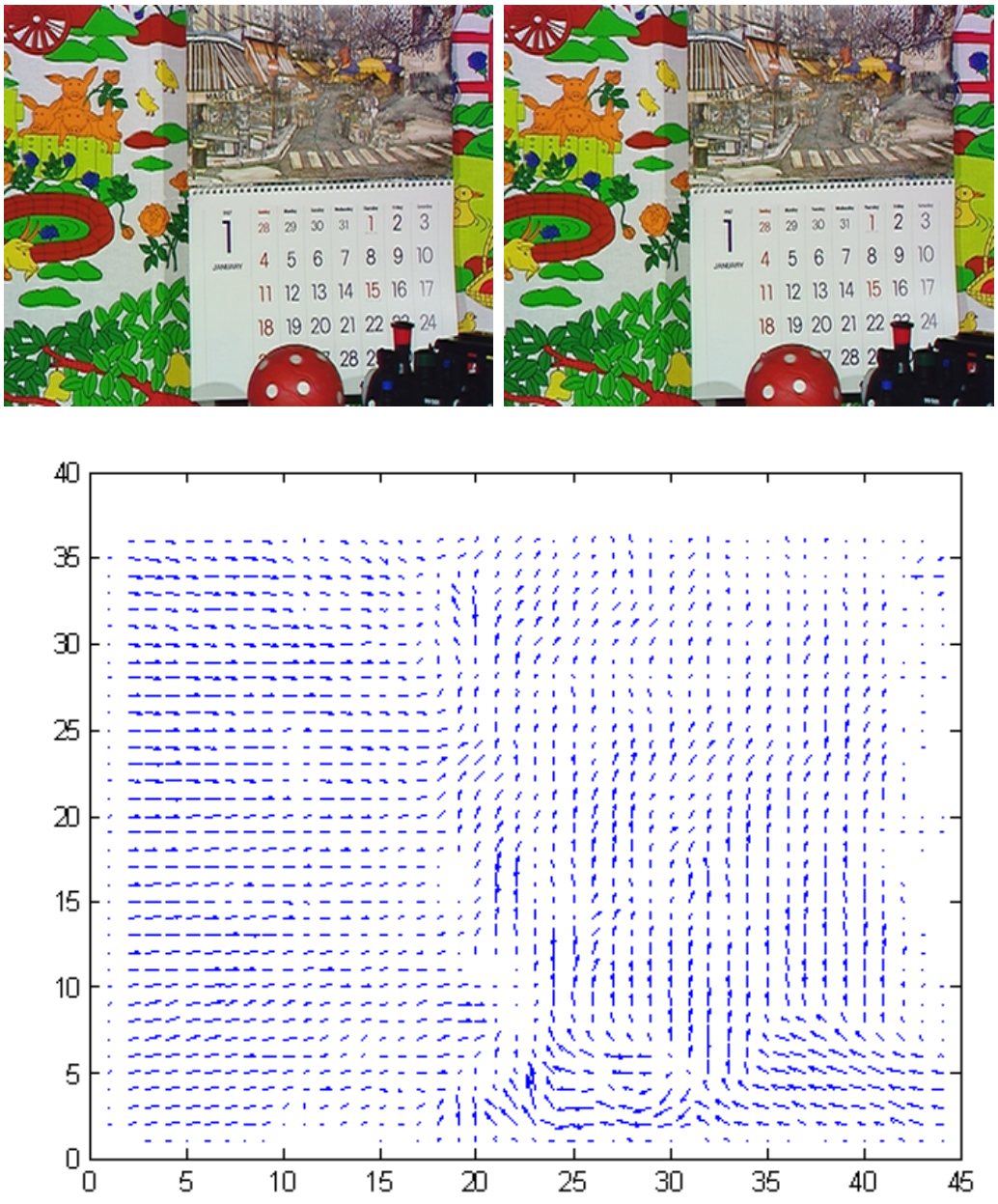


Figure 1.1: Above: two consecutive video frames of *mobile*, bottom: estimated optical flow between the two frames

as affine or perspective[2]. Then suitable matched feature points in different frames (usually two frames) are selected to establish a well-posed solution for parameter estimations. Due to the deterministic model and the limited number of feature points, these kind of approaches are usually able to obtain either a sparse motion field or an estimation of global motions. Thus they are widely used to estimate global camera motion [7] or image registrations[8]. The limitations of these approaches come from two sources. The first is their parametric model. Due to the complexities of natural motion, the deterministic imaging models are far from enough or correct. The second is that it is highly dependent on feature selection and tracking techniques. Although much work on this problem has been done in the computer vision community[9],[10], feature selection and tracking are still an open problems, which limits the performance of these matching based ME techniques.

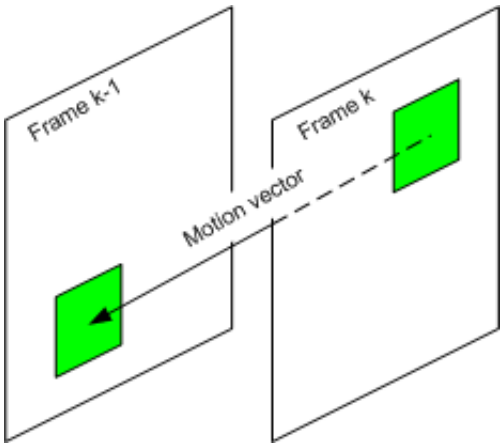


Figure 1.2: Block Matching Algorithm

Another typical matching based technique is the widely used block matching based method[11],[12],[13],[14]. Although rooting from the same brightness constancy constraint in Eq.(1.1), block matching based method solves the problem in a nonlocal way. As shown in Fig.1.2, this kind of method is to nonlocally find a matching of one block in reference frame(s) instead of solving the local differential equations. The relative location of the matched block in the reference frame is taken as motion vector.

The matching criteria is defined by a similarity measures, which can be a function of sum of squared error(SSE), sum of absolute difference(SAD), or some correlation based distance. Using different similarity measure, block matching method can handle the case where the brightness constraint does not hold due to the non-uniform illumination[15]. By introducing nonlocal information, block matching method can get better accuracy for large motion estimation even though it could incur higher complexity by increasing the searching range. Since the operation unit is a pixel block, motion smoothness constraint is implicitly included in this technique. In the case that the motions in the block have different characteristics, subdivision or segmentation can be introduced to split the block firstly, then the matching is done for each subblock independently[16],[17].

Recently, a new motion estimation technique appeared, which can be regarded as the combination of optical flow based and matching based approach. They call it *particle video*, since the video motion is represented using a set of particles[18]. Each particle is an image point sample with a long duration trajectory and other properties. The number and location of particles are optimized by measuring point-based matching along the particle trajectory and distortion between the particles. In [18], five steps (i.e., propagation, linking, optimization, pruning, and addition) are defined to keep refining the distributions of particles in a video sequence. Unlike the traditional optical flow representations that usually consider only two frames, particle video enforce long-range correspondence cosnsistency which is useful for many applications, like restoration, segmentations. Comparing to other matching based approaches, particle video can achieve a relatively dense motion field since the particle can be a pixel unit and can be non-uniformly distributed over video frames.

Although video modeling by explicit motion representation is conceptually simple, it has some limitations coming from the motion representation and corresponding motion estimation techniques. We summarize them as follows:

First, due to the limited sampling rate of video capture devices in space and time, the brightness constancy constraint that most of the explicit models originate from is only approximately valid in the discrete domain. For example, the brightness along a motion trajectory can be roughly regarded as constant in a relative smooth area with slow motion. However, in a texture region with fast motion, it's another story because of the impact of spatial and temporal aliasing. Once the brightness constancy constraint is not accurate, the correspondence is hard or not able to be found, which contributes to the motion estimation inaccuracy.

Second, motion estimation with explicit representation could become very difficult and inaccurate when motion becomes rather complicated. As we have known, most explicit motion representations suffer a lot from the two notorious problems, occlusion and aperture problem[2]. In occlusion problem, the object can be blocked or unblocked by others in future frames, which makes the correspondence hard to find. Aperture problem goes to the other end where a block or point can find more than one correspondence in other frames, which could be caused by the intensity homogeneity or structure similarity. Besides these problems, the natural motion includes more complex patterns: when zooming happens, we even cannot give an accurate definition of the correspondence; given a deformable motion(e.g. smoke and water's motion), the correspondence point can disappear forever in other time slots(frames). All these examples imply that the explicit correspondence based motion representation is far from complete to describe the characteristics of natural motion, which causes the inaccuracy of the video models based on it.

Moreover, the applicability of ME into incomplete or noisy observation data is still questionable. In most of the video models with explicit motion representation, the observation is assumed to be complete and noise free. However, such assumption could be invalid in many real cases. In inpainting [19] or error concealment[20] problem, we just have the video data with block or pattern loss; in video denoising[21], the obser-

vation data are the corrupted version of the original one; in video superresolution[22], we only have low resolution video frames. In these applications, ME becomes very challenging and tends to be suboptimal or even incorrect. The video modeling approaches based on them are therefore not reliable.

New video models are expected to appear based on a deeper understanding of video signals. Recently, a new class of video modeling approaches have sprouted. These approaches avoid the correspondence based motion representation and opt to represent motion implicitly. Comparing to an explicit model, such implicit model enjoys many advantages especially in low-level vision applications, which we will elaborate in the following sections.

1.2 Video Modeling by Implicit Motion Representations

In the physical world, it is reasonable to describe the motion of a point by displacement or velocity. However, in digital world, correspondence based description of motion is questionable. The correspondence based motion representations usually assign probability one to a single motion vector. This hard-assigned motion vector is often not optimal due to the imperfectness of video source. For example, the correspondence of a point can disappear through frames because the limited sampling rate in space and time[3]. In this case, we may not find the corresponding point in other frames but we can still observe motion. And the existance of occlusion and aperture problems in video source makes such representations not effecient. Moreover, the noisy observation can degrade the performance of ME techniques. However,

[3]Although it long-range constraint is enforced in particle video[18], to build the particle field, we still need enough sampling rate to maintain the continuity of motion trajectories.

we find in some low-level vision applications that the motion field is not the necessary output, we do not have to explicitly represent the motion and take the motion estimation as necessities for exploiting motion information. In some literatures[23], authors even argue the explicit motion estimation may harm the restorations tasks. They agree that motion can be exploited in some implicit ways. In the past decades, some video modeling approaches with implicit motion representations are proposed. With promising performance, these approaches generally have higher flexibilities in many applications when compared to the conventional explicit methods.

The spatio-temporal autoregressive(STAR) model is a kind of implicit video modeling approaches[24]. This model expresses each pixel as a linear combination of surrounding pixels lagged in space and time. With the neighborhood size and topology defined, the model parameters (combination coefficients) can be globally learned from the given image sequence. Although it is not explicitly represented or estimated as motion vector, motion information is embedded into the model parameters, which can be used for temporal texture synthesis or recognition tasks. A similar model, dynamic textures, was proposed in[25]. In this work, the spatial correlation is incorporated without imposing causal restrictions, which led to catch more complex motions, especially when the STAR model is ineffective, such as rotation, acceleration and some non-translational motions.

In [26], motion is represented by *object tunnels*, which is a volume carved out by each moving object in the video domain. The motion modeling and segementation problems can be formulated by a 3D volume competition problem. Then level-set methodology is applied to solve the problem. This approach naturally embeds object's temporal smoothness through 3D curvature and consider jointly space and time over multiple frames.

Other implicit video modeling approaches are mostly patch-based. A patch of a video sequence is a data unit with a non-zero size and a topological structure over

space and time. A 2D image block that is widely used in image and video coding[17] is a special example of a patch with the temporal dimension equals to one and spatial topology is a rectangle. Video epitome[27] is an example of implicit video modeling approaches. The epitome of a video is a spatially and/or temporally compact representation of the video. It retains the video's essential textural, shape, and motion information. Video epitome is introduced as a patch-based probability model that is learned by compiling together a large number of examples of pathes from an input video. Under a probabilistic generative model, any other video patches can be considered to have come from the video epitome, since it includes the essential content of a video. In [27], video epitome was applied to solve low-level vision problems, like video inpainting, denoising, and super-resolution, and showed promising performance.

In [28], [29], and [23], a nonlocal means approach is proposed for image and video restorations. In these approaches, motion estimation is circumvented in restoration tasks. The temporal redundancy is exploited by nonlocal patch matching. In the work, the patch searching can have multiple outputs, which is unlike the traditional block matching with singling out one most similar block. Thus the *aperture problem* is even exploited as an advantage to get more correspondences for restoration tasks, since more hypotheses are usually preferred in restoration problems. Another space-time patch-based method for image sequence restoration is proposed in [30]. In this work, there is no explicit motion estimation, but motion information is implicitly exploited by adapting the topology of space-time neighborhood based on the local analysis of the bias-variance trade-off. The authors suggests their approach can also work together with motion compensation to cope with very large displacements due to camera motion.

In [31], the image denoising method reported in their prior work K-SVD [32] is extended to video denoising. Their work relies on sparse and redundant representations of small patches in the video sequence. The explicit motion representation

and estimation are avoided. Motion information is implicitly embeded in the patch dictionary that is trained from current and neighboring frames. The work VBM3D proposed in [33] is also a patch-based sparsity restoration approach. Unlike the basis pursuit approach in [31], VBM3D adapt signal to a set of fixed basis by generating a higher dimensional signals with block clustering in space and time. The sparsity constraint is enforced to the generated high dimensional signal by thresholding transform coefficients. In this work, motion estimation is still not involved. But motion information is exploited by space-time block clustering.

As we can see from the above review about various implicit motion representations, the motion-related information is not exploited by the format of motion vectors. It can be embedded into the model parameters other than motion vectors, geometric characteristics of video data, relationships among pixels or patches, and etc. In many applications, these implicit models enjoy their flexibilities and effficiencies, especially in restoration problems. However, they still have limitations. Either STAR or dynamic textures model is based on the stationarity assumption of a video source. The model parameters are invariant over space and time. So such global learned parameters cannot efficiently describe the motion information in natural video signals, which is generally non-stationary. This limits the applications of these models. The *object tunnels* concept is noval but it still depends on explicit motion model (e.g., affine model in [26]) to finally estimate motion information given the segmentation surfaces, which can limit the capabilities as we address. And the segmentation of *object tunnel* is still problem when the observation is noisy. Although the patch-based implicit models have received increasing attention, they are limited to denoising application alone(except video epitome) and lack theoretically convincing interpretation about their effectiveness.

Although these approaches have their potentials and limitations at the emerging peroid of implicit motion model, these pioneering works do open our minds to explore

more accurate and efficient motion representation approaches. In our work, we advocate the implicit motion representations and target to pursue more video processing models based on them.

1.3 Contributions

In this thesis, we continue to explore new video modeling methods with implicit motion representations. Specifically, we proposed a Least Squares (LS) based parametric model and a patch-based nonparametric model. Both models cast away the correspondence based motion representations. Motion-related temporal dependency can be embeded into either filter coefficients or relationships among video patches.

In the first model, we formulate video modeling as a LS based auto regression problem. Through a local spatio-temporal training, local motion information can be carried by the regression parameters implicitly. Without explicit motion representations, the model can be exploited to support a variety of low level vision tasks. When compared to the existing learning-based STAR model[24], our STALL model can be viewed as a localized version that updates the model parameters on a pixel-by-pixel basis. Such localization allows our model to handle a wider range of motion than STAR and achieve higher computational efficiency. When compared to explicit models, STALL is often preferred in the scenario where a dense and subpixel motion field is not affordable(e.g., in video coding) or feasible(e.g.,in video denoising).

Inspired by the works in[34] and[35], we propose patch-based video model. This patch-based model does not depend on the explicit motion representations, but exploits the motion related dependency through a patch clustering and a generic sparsity model. Specifically, we propose to extend the block matching (neareset neighbor search) into patch clustering (k-neareset neighbors search) and apply an adaptive sparsity constraint to each patch cluster. Under a variational Bayesian framework,

we treat both patch clustering result and unobervable data as latent variables and solve the inference problem via an iterative algorithm. Repeating the process to all patches in the video sequence, we can obtain a distributed representation of the video due to the overlap of patches. Then a sparsity based Bayesian fusion scheme is introduced to fuse the multiple hypotheses for the final inference.

In the following chapters of the thesis, the theories of the above implicit video modeling methods will be introduced. We also demonstrate their capabilities and performances in a variety of low level applications. The remaining of the thesis is organized as follows:

In chapter 2, we propose the theory of STALL model by building up the duality between the edge contour in 2D image and the motion trajectory in 3D video sequence. Then we investigate the relationship between the model parameters and the video source. Based on the relationship, we introduce a new concept of spatio-temporal resampling together with an empirical Bayesian fusion approach to facilitate the task of video modeling.

In chapter 3, we apply the STALL model on several low-level vision problems - namely video error concealment/inpainting, video denoising, and video super-resolution. As we can see, STALL model provide a class of unified solutions to attack these problems by adjusting the model parameters for different applications.

In chapter 4, we cover the theory of our patch-based implicit model. We first extend the block matching (nearest neighor search) into patch clustering (k nearest neighbor search). Then we illustrate how to establish the adaptive sparsity constraint for exploiting the information in each patch cluster. We will introduce how to solve the inference problem via an iterative algorithm under a Bayesian framework with the patch clustering result and sparsity constraint.

In chapter 5, we will apply the patch-based model to three video processing problems, video error concealment, denoising, and compression artifacts removal. We will

demonstrate its processing performance by comparing with both parametric model and other nonparametric model.

Chapter 6 provides concluding remarks on our proposed models. We also discuss the perspectives of implicit motion representations in applications ranging from low to high level vision problems.

Chapter 2

Video Modeling via Spatio-Temporal Adaptive Localized Learning

2.1 Introduction

In this chapter, we propose a parametric video modeling method which is not based on explicit motion representations. Motivated by the duality between a 2D edge contour and a 3D motion trajectory, we formulate the video modeling problem as a Least-Square based adaptive filtering problem. The filter coefficients (model parameters) can be estimated in a pixel by pixel fashion through a spatio-temporal adaptive localized learning (STALL). The adaptation can be achieved by the localized learning, which makes the model more flexible than the explicit models.

Similar models as our STALL model are the STAR and dynamic textures model proposed in [24] and [25] respectively. However, both STAR model and dynamic textures model attempt to capture the motion characteristics by certain global learnings, which are based on the stationarity assumption of the video source. Unfortunately,

this assumption is inaccurate or even invalid for most natural video signals. When the motion becomes complicated, these models try to capture more details by increasing the number of model parameters. This incurs higher computational cost. Although introducing more model parameters can make the model more accurate, it is still in expedient, which does not solve the problem from the root, i.e. the inaccurate stationary assumption and the global learning. Unlike the global learning by STAR and dynamic textures model, STALL catches motion information through a localized space-time adaptive learning. Due to the localization characteristics, the number of model parameters can be reduced to less than ten, which is easy to implement and provide chances to develop fast algorithms. In our STALL model, we just assume the motion is stationary within a very small local region. The model parameters can be updated in a pixel by pixel fashion, which provides a way to adaptively model the non-stationary video signals.

The rest of this chapter will be organized as follows: in section 2.2, we introduce the duality between 2D image contour and 3D motion trajectory in video, by which we work out our STALL model in section 2.3. In section 2.3 we analyze the impact of motion type to the settings of model parameters. Based on the analysis, in section 2.4 we introduce a spatio-temporal resampling and a empirical Bayesian fusion technique for STALL to facilitate the video modeling task. In section 2.5, we conclude the parametric model and provide the orientation of future research.

2.2 Duality between 2D image contour and 3D motion trajectory

The duality between edge contours in still image and motion trajectories in video can be best understood in the continuous space. Loosely speaking, edge contours are planar curves characterizing the boundary of an object in an imaging plane; motion

trajectories are 3D curves showing the projected positions of the same physical point of an object onto image plane along the temporal axis. Radiometric modeling of imaging process tells us that the irradiance E of a point in the imaging plane (intensity value before digitization) is proportional to the radiance L of an area in an object [36]. If we assume the object surface is approximately Lambertian (ignore specular effects), it can be shown that the surface radiance of any point in the object is jointly determined by the surface characteristics (i.e., geometry and material characteristics) and incident radiance. So the variation of intensity field could caused by change of either surface characteristics or incident radiance, which can form edge contours in the image plane.

Generally an isotropic distribution of the surface characteristics and incident radiance will form a smooth area in the image plane, while an anisotropic distribution will shape edges or textures. We can observe that the intensity field will vary dramatically in the orientation crossing an edge while it will keep constant along the edge, which is the so-called geometric constraint. Similarly, in video scenario, if the relative geometric relationship between incident radiance and scene is unchanged (e.g., slow and smooth motion), the irradiance E of the same physical point would remain constant along the temporal axis. Such observation gives rise to another type of geometric constraint - i.e., iso-intensity constraint along the smooth motion trajectory, which is shown by a synthetic example in Fig.2.1. We can find that the iso-intensity constraint along the motion trajectory is the same as brightness constancy constraint introduced in Chapter 1.1.

Although it is generally difficult to display the iso-intensity profile in 3D, visualization becomes easier when motion is constrained to horizontal camera panning. If we denote a video sequence by $I(x, y, t)$. As shown in Fig.2.2, we can cut along a fixed vertical index $y = y_0$ and obtain a spatio-temporal slice $I(x, y_0, t)$. The flow-like patterns in such slice convey the information of motion trajectory and we can easily

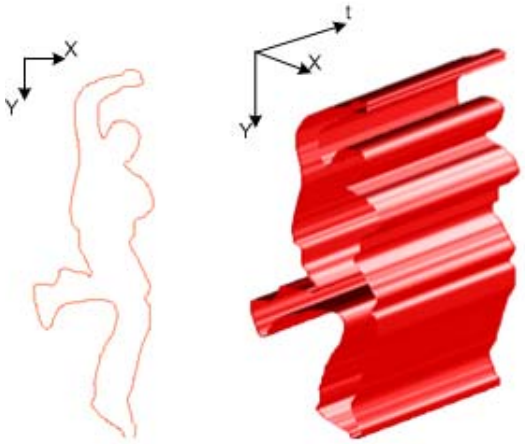


Figure 2.1: Duality between edge contour and motion trajectory

check the validity of motion-related iso-intensity constraint.

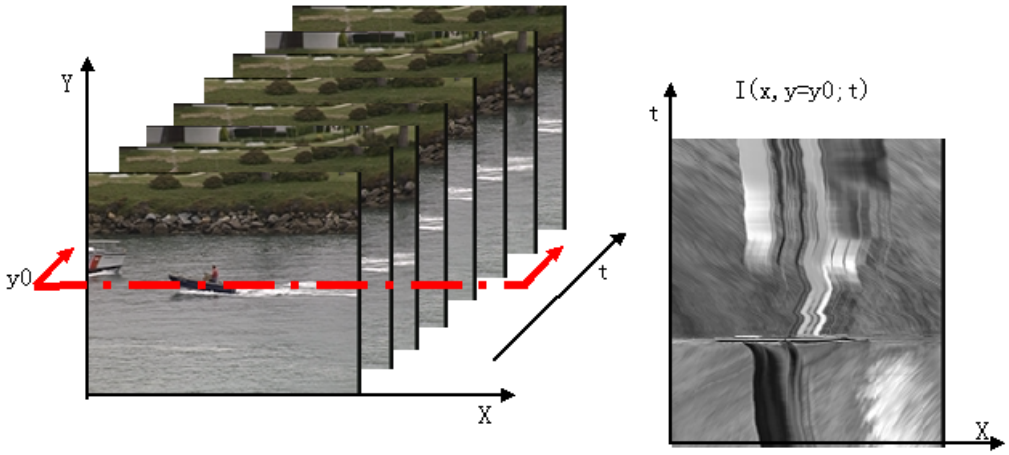


Figure 2.2: Iso-intensity constraint along motion trajectory can be seen more clearly by analyzing spatio-temporal slices.

The above duality provides us a way to model a video signal by exploiting the motion trajectory iso-intensity constraint as the approach to model an image signal by the edge geometric constraint. We will derive and explain our model in the following sections.

In the discrete space, we acknowledge that spatial and temporal aliasing do have different impact on motion-related geometric constraints. In this work, we assume

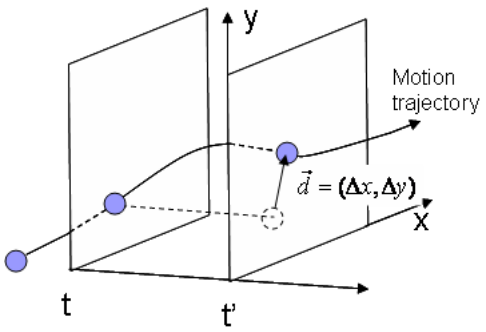


Figure 2.3: Motion trajectory and point displacement

that the spatial and temporal sampling rate are high enough to ensure that motion-related geometric constraint is approximately satisfied. Here we need to notice that such assumption is also implicitly required by most of the explicit motion models, in which the extraction of accurate motion information is always dependent on the enough sampling rate both in space and time. Even in other implicit video models, the severe aliasing does impact the modeling capability, though it is not stated explicitly.

2.3 Video Modeling via STALL

2.3.1 From Explicit to Implicit Motion Representation

Suppose $\{s(x, y, t)\}$ is the video in a continuous space where (x, y) and t are the spatial and temporal coordinates respectively. As shown in Fig.2.3, explicit ME is based on the formulation of following correspondence problem:

$$s(x, y, t) = s(x - \Delta x, y - \Delta y, t - \Delta t) \tag{2.1}$$

where $(\Delta x, \Delta y)$ is the displacement vector of the considered point from $t - \Delta t$ to t (current timing). Using 1^{st}-order Taylor expansion, we can get

$$\Delta x \frac{\partial s}{\partial x} + \Delta x \frac{\partial s}{\partial y} + \Delta t \frac{\partial s}{\partial t} = 0 \tag{2.2}$$

If we divide both sides of Eq. (2.2) by Δt and let $\Delta t \to 0$, the classical optical flow equation follows:

$$\frac{\partial s}{\partial t} = \frac{\partial s}{\partial x}\vec{v}_x + \frac{\partial s}{\partial y}\vec{v}_y \tag{2.3}$$

where $\vec{v}_x$ and $\vec{v}_y$ are the velocities of a point in x and y directions respectively. However, since our interest is not to estimate displacement (Δx,Δy) or velocity vector ($\vec{v}_x$,$\vec{v}_y$) at (x, y), we opt to expand the Eq. (2.2) by incorporating the estimation of spatial gradients. For example, finite difference scheme [37] gives[1]

$$\begin{aligned} \frac{\partial s}{\partial x} &= \frac{s(x+h,y) - s(x-h,y)}{2h} \\ \frac{\partial s}{\partial y} &= \frac{s(x,y+h) - s(x,y-h)}{2h} \\ \frac{\partial s}{\partial t} &= \frac{s(x,y,t) - s(x,y,t-h)}{h} \end{aligned} \tag{2.4}$$

where h is the spatial sampling grid increment. Substituting Eq. (2.4) into Eq. (2.2), we obtain

$$\begin{aligned} s(x,y,t) = s(x,y,t-h) + \frac{\Delta x}{2\Delta t}s(x-h,y,t) - \frac{\Delta x}{2\Delta t}s(x+h,y,t) \\ + \frac{\Delta y}{2\Delta t}s(x,y-h,t) - \frac{\Delta y}{2\Delta t}s(x,y+h,t) \end{aligned} \tag{2.5}$$

Eq.(2.5) can be viewed as the convolution of s with a five-point kernel. The topology of the kernel is shown in Fig.2.4. More generally, if Taylor expansion with higher-order derivatives is used, finite difference approximation will produce the convolution of s with a linear kernel with a larger support

$$s(x,y,t) = \int_{(x',y',t') \in \mathcal{N}_c(x,y,t)} a(x',y',t')s(x'-x, y'-y, t'-t)dx'dy'dt'. \tag{2.6}$$

[1] note that such approximation is not unique

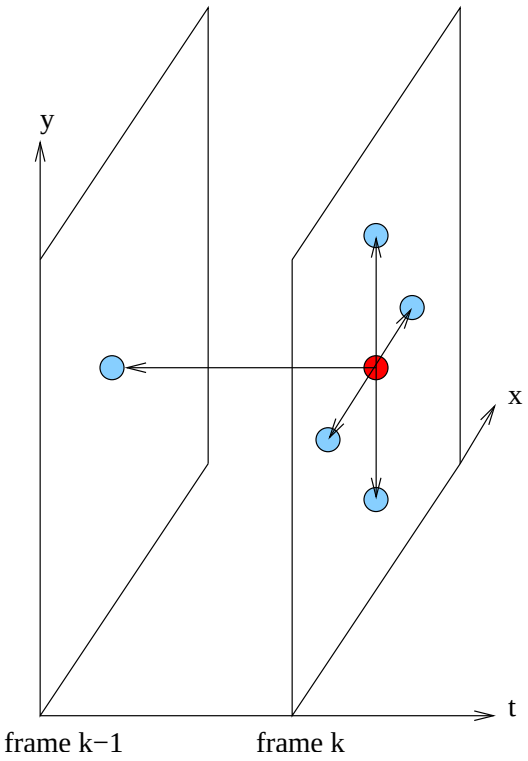


Figure 2.4: Five-point kernel topology

where kernel $a(x', y', t')$ includes all coefficents of the terms on the right side of Eq. (2.5). Note that all terms related to displacement (motion) $\Delta x, \Delta y$ are all assimilated into the linear kernel.

When compared with Eq. (2.1), we argue that Eq. (2.6) has the following advantages: 1) it better fits the discrete approximation - linear convolution simply becomes linear filter. Although there is no additional interpolation kernel involved, we can obtain sub-pel accuracy motion information adaptively. 2) it improves the robustness to observation noise because of the increased support size (due to higher-order expansion).

In the discrete space, we use $s(r, c, t)$ to denote video signals, where $r \in [1, R]$ and $c \in [1, C]$ are two spatial coordinates (row and column index respectively) and $t \in [1, T]$ is the temporal one. For the simplicity of notation, we denote the position of current pixel by $\vec{n}_0 = (r, c, t)$ and its surrounding neighbors within the support of

filtering kernel $\vec{a}$ by $\vec{n}_i$ $(i = 1, 2, ..., N)$. Then Eq. (2.6) becomes

$$s(\vec{n}) = \sum_{i=1}^{N} a_i s(\vec{n}_i). \tag{2.7}$$

which is closely related to the spatio-temporal autoregressive (STAR) model proposed in [24]. However, unlike STAR in which model parameters are trained globally from the given sequence, we propose to update the set of AR coefficients $\vec{a} = [a_1\ a_2\ ...\ a_N]$ within a local space-time training window on a pixel-by-pixel basis.

We denote the training window centered at $\vec{n}$ by $\mathcal{M}(T_s, T_t) = [r - T_s, r + T_s] \times [c - T_s, c + T_s] \times [t - T_t, t + T_t]$ ($M' = (2T_s + 1)^2(2T_t + 1)$ is the total number of training sample candidates). Our training samples are selected from these candidates based on the availability of the pixels within the training window. Assume there are M samples selected and organized into a vector $\vec{y}$. The N neighbors in $\mathcal{N}(\vec{n}_i)$ for each sample selected can be packed into a $1 \times N$ row vector X_i. An $M \times N$ data matrix D can be generated as

$$D = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_M \end{bmatrix}$$

The derivation of locally optimal AR coefficients $\vec{a}$ follows the standard Least-Square formulation

$$\vec{a} = \underset{\vec{a}}{argmin} ||\vec{y}_{M\times 1} - D_{M\times N}\vec{a}_{N\times 1}||^2 \tag{2.8}$$

Once the number of training data M is larger than the number of neighbors N, it has closed-form solution, which is given by

$$\vec{a} = (D^T D)^{-1}(D^T \vec{y}) \tag{2.9}$$

where $D^T D, D^T \vec{y}$ are empirical estimation of local covariances. Such result can be viewed as the extension of traditional 2D Kalman filtering [38] into 3D. Motion information, as we have shown in our derivation, is implicitly embedded into the filter coefficients $\vec{a}$.

Our implicit motion model is particularly appealing to low-level vision tasks such as video interpolation, filtering, and prediction for the following three reasons. First, since implicit model is only based on the smoothness constraint of motion trajectories, it has improved capability on handling complex video material than explicit motion models. When the accuracy of ME becomes questionable (e.g., due to block approximation, ill-posed nature, non-ideal observation, suboptimal motion representation), ME-based techniques produce poor performance. By contrast, the proposed implicit motion model can handle most typical video material containing slow and smooth motion including camera panning, rotation and zoom [2]. Second, the filtering perspective in STALL allows us to adaptively achieve the tradeoff between spatial and temporal dependency. Such tradeoff is important to model the video source with occlusion and nonrigid motion. In the extreme case of either little temporal dependency or little spatial dependency, we will see the model can automatically switch to exploit the information from the other side by spatio-temporal learning. Moreover, the sub-pel accuracy motion estimation is often obtained by spatial invariant interpolation in most explicit motion estimation techniques[39][40]. However, STALL implicitly includes the spatial and temporal adaptive interpolation to achieve higher modeling accuracy.

---

[2]Fast and rapid motion is handled by incorporating the space-time resampling and Baysian fusion technique into STALL, which will be introduced in section 2.4

2.3.2 Model Parameter Setting to Capture Motion

A. Localized learning for adaptation

The main difference between STALL and STAR[24] lies in its localization. STALL model can update the model parameters in a pixel by pixel fashion by a localized learning instead of a global learning adopted by STAR. The reasons that we prefer this localized approach are mainly from the nonstationarity of video signals. We perform the following conceptual test to verify the necessity of this localized learning.

Define the topology of filter support as shown in Fig.2.4. We look the training window paramaters T_s and T_t as two variables. So the size of training window varys with T_s and T_t over space and time, as shown in Fig.2.5. The goal of this test is to predict current pixel (i.e., pixel in red in Fig.2.4 and Fig.2.5). We are going to watch the variations of prediction performance (PSNR) with the change of model parameters T_s and T_t.

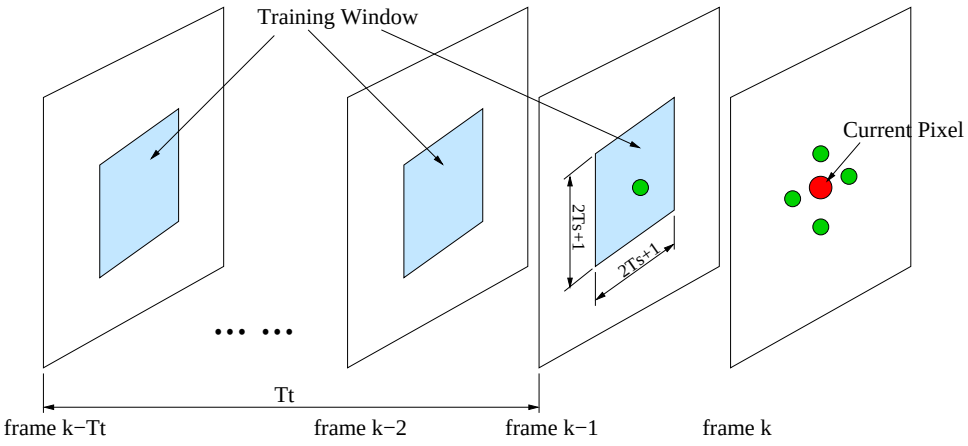


Figure 2.5: Topology of training window parameters

Fig.2.6 shows the PSNR evolution with the variations of training window parameters. We can see the performance acheives the maximum only in a relative local range. When the training window becomes larger in space and time, more pixels will be incorporated into training. However, due to the nonstationarity of video signals, more and more outliers could also be included, which could degrade the performance.

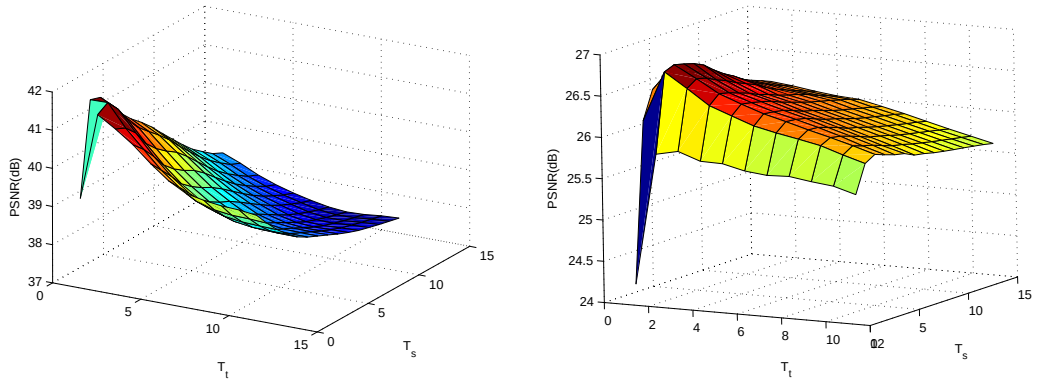


Figure 2.6: PSNR evolution with variations of training window parameters T_s and T_t. Left: salesman-qcif, right: garden-qcif

In general, the optimal training window is dependent on many factors, such as video content (i.e., motions and scene contents) and video resolution. Roughly speaking, we can choose relative larger training window for the video with slow motion and smooth scene than that with fast (or complicated) motion and textural scene. We also observe the optimal training window size will become large with increasing the video resolution. For example, both plots in Fig.2.7 are tested on *coastguard* sequence, but in different resolutions. The PSNR peak moves towards the relative larger training window size in higher resolution video (i.e., cif). It is reasonable since there are more capturing sensors involved in imaging. So we can find more pixels with the same motion characteristics for a better training result.

B. Topology Adaptation to motion

As shown in section 2.3.1, the parameters of STALL model are the filter support $\mathcal{N}(\vec{n})$ and the training window $\mathcal{M}(T_s, T_t)$. Filter support and training window can incorporate both spatial and temporal pixels. The size and topology of them can be changed over space and time in order to make STALL model capture different motions, which provides a foundation to build up a class of unified algorithms for different applications. In usual case, we fixed the size of the filter support and training

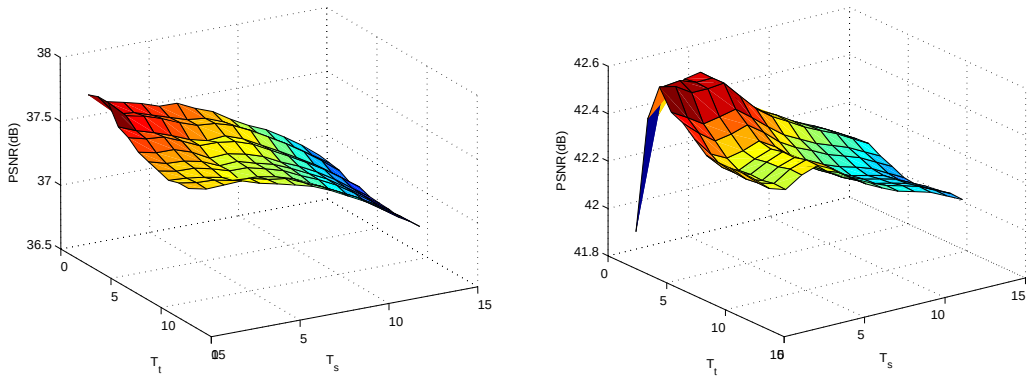


Figure 2.7: PSNR evolution with variations of training window parameters T_s and T_t in different resolution. Left: coastguard-qcif(176×144), right: coastguard-cif(352×288)

window throughout the whole sequence for computational simplicity, though it is not necessitated. In the following, we will introduce how to make STALL model to fit different motion characteristics through adjusting the *shape* of the filter support and training window.

In a lot of video sequences, especially in some high definition(HD) video, the dominant motion is relatively slow.[3] When the dominant motion in the video sequence is relatively slow, we can just use a *straight* filter support and training window, like the left in Figure 2.8 throughout the whole sequence, since the slow motion trajectory is relatively *straight* and will not step out these area in a local region. However, when the motion in a video sequence becomes fast, the motion trajectory could change dramatically so that the straight filter support and training window cannot cover it in the local region. In this case, we need to adjust the setting by *skew* the filter support and training window along the orientation of motion trajectory, which is illustrated in the right of Figure 2.8.

However, in a general video sequence, there could be both slow and fast motion

[3]Although the motion itself can be large, it could be relatively slow in consecutive frames because of high temporal sampling rate (frame rate). The temporal sampling rate is about 24 to 30 frames per second(fps) for standard definition(SD) video, 60 fps for HD video.

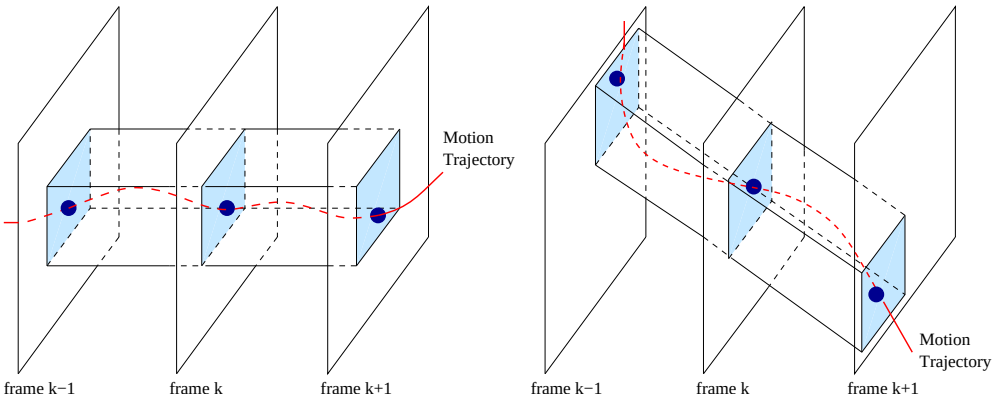


Figure 2.8: Model parameters (filter support and training window setting for capturing different motion characteristics: left is for relatively slow motion, right is for fast motion.)

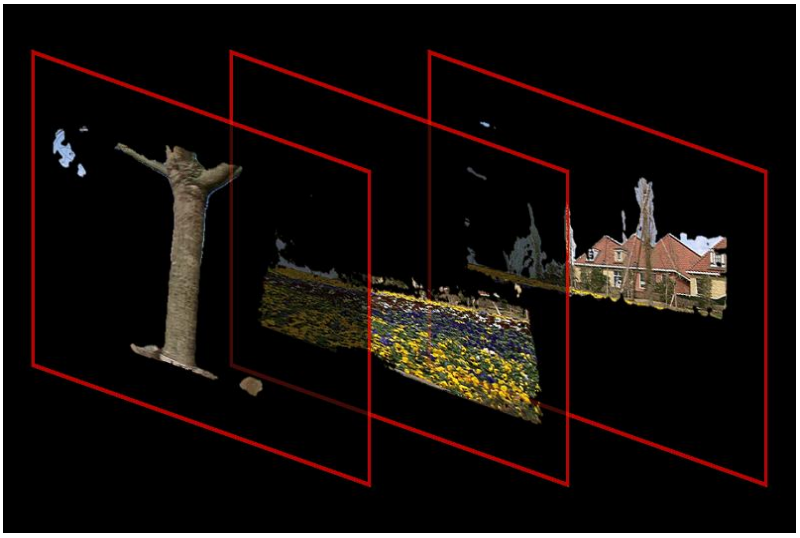


Figure 2.9: Layered representation of a video frame

components existing at the same time (frame), which could be caused by the difference of scene depth or velocities of the objects in it. In this case, we would like to request the parameter setting can adaptively fit the requirement of the local motion characteristics. This adaptation need the prior information of motion, such as high level motion segmentation results. If we can obtain a layered representation of a video by some motion segmentation approaches[41], as shown in Figure 2.9, we can easily realize the adaptiveness by using the right model parameter settings for each layer.

However, such prior motion information is usually based on the video modeling result which is what we are pursuing now. To escape from this dilemma, we introduce a resampling and a Bayesian fusion technique to facilitate STALL model to solve this problem, which dramatically enhances the modeling capacity of STALL model. We will elaborate these techniques in section 2.4.

2.4 Video Modeling by Spatio-Temporal Resampling and Bayesian Fusion

2.4.1 Distributed representation of a video

As mentioned in section 2.3.2, the effectiveness of STALL largely depends on the choice of model support $\mathcal{N}$ and training window $\mathcal{M}$. When both $\mathcal{N}$ and $\mathcal{M}$ are fixed, STALL can only handle the class of video containing single motion velocity that fits the parameters setting. Although adaptive selection of $\mathcal{N}$ and $\mathcal{M}$ by layered representation [42] is conceptually appealing, practically implementing such idea faces the obstacle of layer decomposition or motion segmentation especially in view of the uncertainty with the suboptimal segmentation results.

An alternative solution to achieve spatio-temporal adaptation is to realize the fundamental tradeoff between space and time. Specifically motion is a *relative* concept. The perception of motion arises from the spatial displacement of the same physical point with respect to the camera. Therefore, a moving object might appear still if the camera is moving in parallel to the object at the same speed. Such observation motivates us to introduce a class of spatio-temporal resampling techniques for video modeling. By analogy to resampling in statistics[43], we propose to obtain spatio-temporally resampled signal by "reversibly" transforming the original video into another perceptually meaningful one. For example, temporally reversing a video

sequence is a valid resampling operation (reversed signal is physically infeasible but perceptually convincing); while spatially or temporally shuffling a sequence usually destroys the motion continuity and therefore is not a valid resampling candidate for capturing motion characteristics.

In this work, we consider the class of effective resampling via spatio-temporally warping as shown in Figure 2.10. Specifically, the nth frame is shifted by $[(n-1)v_{cx}, (n-1)v_{cy}]$,($v_{cx}, v_{cy}$ are integers). Note that such warping is readily reversible since no interpolation is involved [44]. The impact of warping can be intuitively understood by referring to Figure 2.10 - when the warping direction matches the motion orientation of interest, a slant trajectory could be transformed into a straight one (therefore better fit the STALL model with the fixed and straight setting of $\mathcal{N}$ and $\mathcal{M}$). The warped video can be looked as captured by a *virtual* camera which records the same scene but with a different **camera** speed $\vec{v}_c = (v_{cx}, v_{cy})$. Note that the above resampling strategy does not affect the motion continuity and therefore the warped video is still perceptually meaningful (ignoring boundary artifacts).

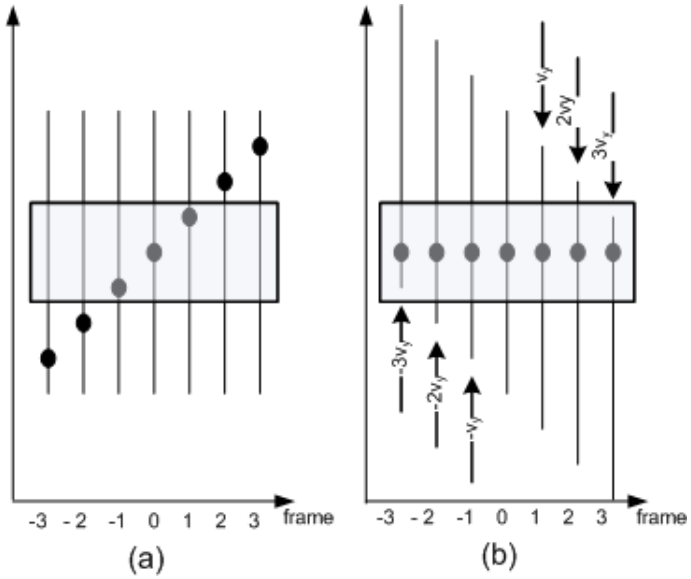


Figure 2.10: An example of spatio-temporal resampling: vertical axis denotes y direction ($v_{cx} = 0$)

Given a set of camera speeds $\mathcal{V}_c = \{\vec{v}_{ci}\}_{i=1}^{m}$, resampling produces a redundant or distributed representation for a video signal. To manage the computational complexity, we make some assumption about high-level knowledge about video such as camera motion type (e.g., panning vs. zooming), which is often available from video segmentation or can be estimated from some global models like phase-correlation[2]. Such high-level information is useful to select resampling parameters, i.e. virtual camera speed set. For instance, we can set $v_{cy} = 0$ to each element $\vec{v}_{ci}$ in the virtual camera speed set for the video sequence with only horizontal camera panning. Figure 2.11 shows the temporal slices of original and resampled video for *garden* (up) and *mobile* (down) sequence. It can be clearly seen that the fast panning tree in *garden* sequence moves slower and slower as the warping parameter v_{cx} increases. The general virtual camera speed with (both v_{cx} and v_{cy} are not equal to zero) is used for *mobile* sequence, in which the motion change can also been observed.

Such observation with the relativity of motion provides an effective distributed modeling approach, which can dramatically enhance the modeling capability of STALL model. It suggests an alternative approach of achieving adaptation by soft fusion instead of hard decision as in layer-based motion segmentation. Note that even when a fixed training window is used, spatio-temporal adaptation can be achieved by resampling because the training samples vary from one resampled video to another. Apparently, we might choose the optimal training window (in some virtual camera) for a pixel, which assigns a deterministic label to each pixel (the layer index). Such strategy can be shown equivalent to a maximum likelihood (ML) approach of making hard decisions for every pixel. Next, we will show how to softly combine the regression results from distributed virtual cameras under a Bayesian framework.

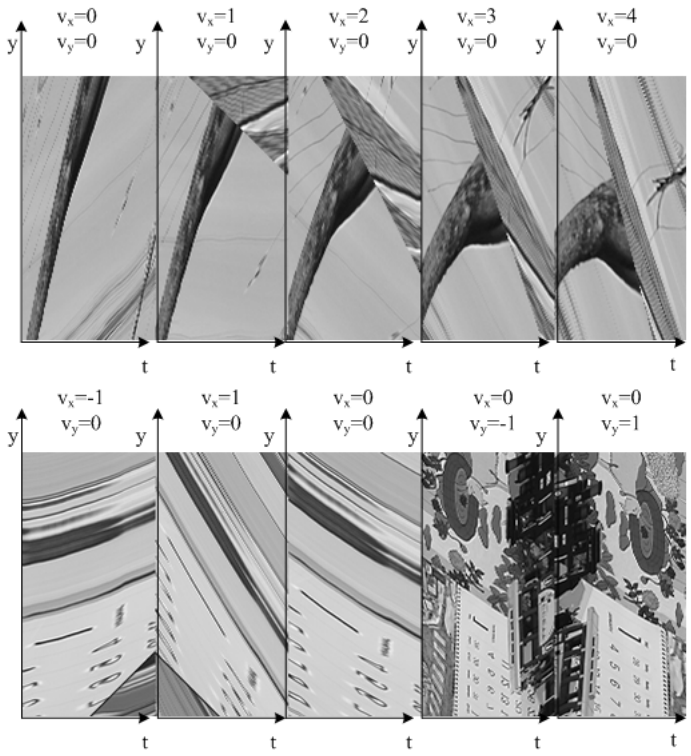


Figure 2.11: Example of spatio-temporal resampling applied to *garden* sequence containing horizaontal camera panning (up) and *mobile* sequence containing camera zooming

2.4.2 Bayesian Fusion of Linear Regression Results

The spatio-temporal resampling can provide us a distributed representation of the observed video sequence as shown in Fig.2.12. We denote the resample set by $\mathcal{S} = (S_1, S_2, ..., S_K)$. For each resample S_i, we can apply STALL on it to obtain a hypothesis for current pixel, which provides us a way to highly redundantly describe the video signal.

We use $(X_1, X_2, ..., X_K)$ to denote the linear regression result by applying STALL to K resampled sequences respectively, denote the true intensity value of current pixel by X and the estimate of X by $\hat{X}$ (to simplify the notation, we drop 3D coordinated $\vec{n}$ from now on). We consider the Baysian Least-Square estimation given by the

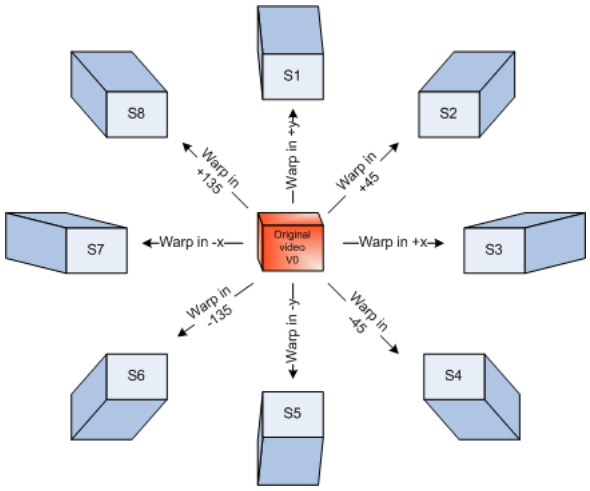


Figure 2.12: Distributed representation of a video provided by resampling

conditional expectation, i.e.,

$$E[\hat{X}|X] = \sum_{k=1}^{K} \alpha_k X_k \tag{2.10}$$

where the linear weighting coefficients $\alpha_k = P(X_k|X)$ denote the posterior probability of inferring X from the $k-$th resampled sequence S_k by $X_k = E[\hat{X}|X, k]$. By Bayesian rule, the weighting coefficients can be calculated by

$$\alpha_k = P(k|X) = \frac{P(X|k)P(k)}{\sum_{i=1}^{K} P(X|i)P(i)} \tag{2.11}$$

where $P(k)$ is the prior probability and $P(X|k)$ is the likelihood function of observing X in the k-th resampled sequence. Using STALL[45], we can empirically model $P(X|k)$ by the regression error e' through Eq.(2.12) for the k-th regression result X_k.

$$e'(\vec{n}_0) = X(\vec{n}_0) - \sum_{i=1}^{N} a_i X(\vec{n}_i) \tag{2.12}$$

Since the original data is what we are modeling, Eq.(2.12) is not available to us. But we can exploit the training error which happens at the regression period to estimate

the real error e' as

$$e(\vec{n}_0) = ||\vec{y} - D\vec{a}||_2 \tag{2.13}$$

Assuming a Gaussian probability function[4], we have

$$P(X|k) = P(e_k) \propto exp(\frac{-e_k^2}{2\sigma^2}) \tag{2.14}$$

where e_k is the regression error of k-th virtual camera by Eq.(2.13) and σ^2 is a constant determined by heuristics (similar to those used in bilateral filtering [46]). So Eq.2.11 can be simplified as

$$\alpha_k = P(k|X) = \frac{exp(\frac{-e_k^2}{2\sigma^2})}{\sum_{i=1}^{K} exp(\frac{-e_i^2}{2\sigma^2})} \tag{2.15}$$

It is easy to see that smaller regression errors lead to larger conditional probabilities in the fusion model Eq.(2.11), which matches our intuition that α_k should reflect the confidence about the modeling result of $k-$th resampled sequence. Intuitively, as long as the virtual cameras speed set is sufficiently large, any segment of a smooth motion trajectory is likely to be warped to the straight position (aligned with the straight training window) in some resampled sequence. Upon the alignment, localized regression by STALL will produce smallest errors and therefore make the largest contribution during Bayesian fusion. When compared with hard-decision based layer representation, our distributed model systematically pools together the inferring results from the resampled sequences and avoids the penalty of the uncertainty with any suboptimal labeling process.

Our modeling procedure is shown in Fig.2.13. Firstly we resample the video signal by a set of virtual cameras which have different speeds. After getting the distributed representation of the processing video, we apply STALL to each resample to obtain

[4]The function here is not unique. Actually we can choose other functions to modeling $P(X|k)$

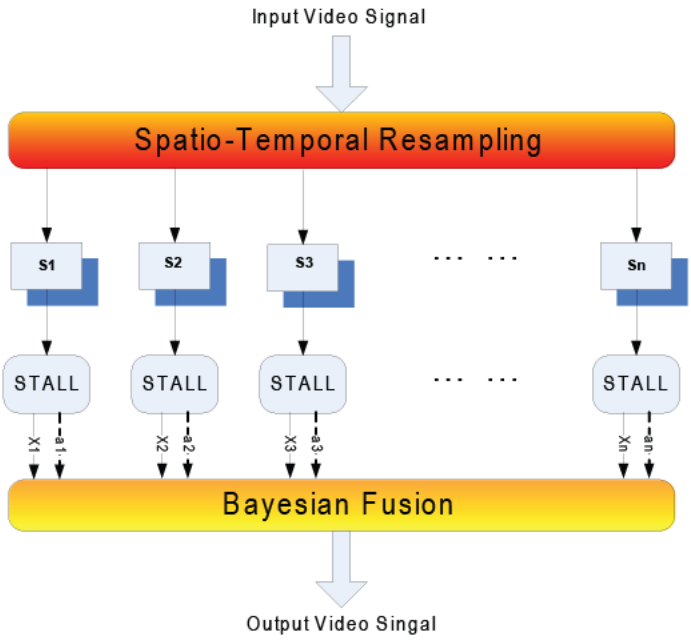


Figure 2.13: Modeling procedure by resampling and fusion

a modeling hypothesis and a weight which comes from the training error. Finally we fuse all modeling hypotheses by the weights to get the final modeling result.

2.5 Conclusion

In this chapter, motivated by the observation of duality between edge contours in images and motion trajectories in video sequences, we proposed STALL, a parametric video model with an implicit motion representation. By formulating the video modeling problem as a linear regression problem, STALL model exploits the motion information through an adaptive space-time localized learning. Such space-time localization and adaptation provide STALL powerful modeling capability for handling more general video sources.

More importantly, STALL model possesses flexibilities to capture different motion characteristics by setting different model parameters, i.e. model support and

training window. To further enhance the modeling capability of STALL to model the video sequences with mixing or complex motions, a distributed modeling technique by spatio-temporal resampling is introduced based on the relativity of motion. Instead of using hard-decision based adaptation scheme to select the best regression result of a certain resampled sequence, we use a Bayesian based soft fusion technique to combine all available regression results, which avoid the penalty of the uncertainty with any suboptimal labeling process.

The importance of STALL model lies not only in the above mentioned flexibilities, but also in the fact that it provides us a unified model for different of applications. So under this framework, we can apply the same model for different applications. The only thing we are required to do is to provide appropriate model parameter settings to fit the requirements of the corresponding applications. In next chapter, we will apply STALL model to different low-level applications to show the promising modeling capability of STALL.

Chapter 3

Applications of STALL

3.1 Introduction

As mentioned in the previous chapter, STALL model provides us a flexible video modeling approach, in which motion is implicitly represented without explicit motion estimation. It gives us a foundation to build a class of unified algorithms for low-level vision problems. Thus we can apply the same model to various applications by providing different model parameter settings (e.g., the causality or topology of model support and training window)[45]. As the examples shown in Fig.3.1, in interpolation problem, we use non-causal model support and training window; in interpolation problem, we generate the support and training window based on the low resolution data; in prediction problem, we just change them to causal ones without changing the backbone of the algorithm.

In this chapter, we consider three applications of STALL model: video error concealment[47], video interpolation and video denoising, which cover two low-level vision problems: filtering and interpolation. All of these problems have non-ideal observations, either incomplete (in error concealment[20] and supper-resolution[48] problems) or corrupted (in denoising problem[49]).

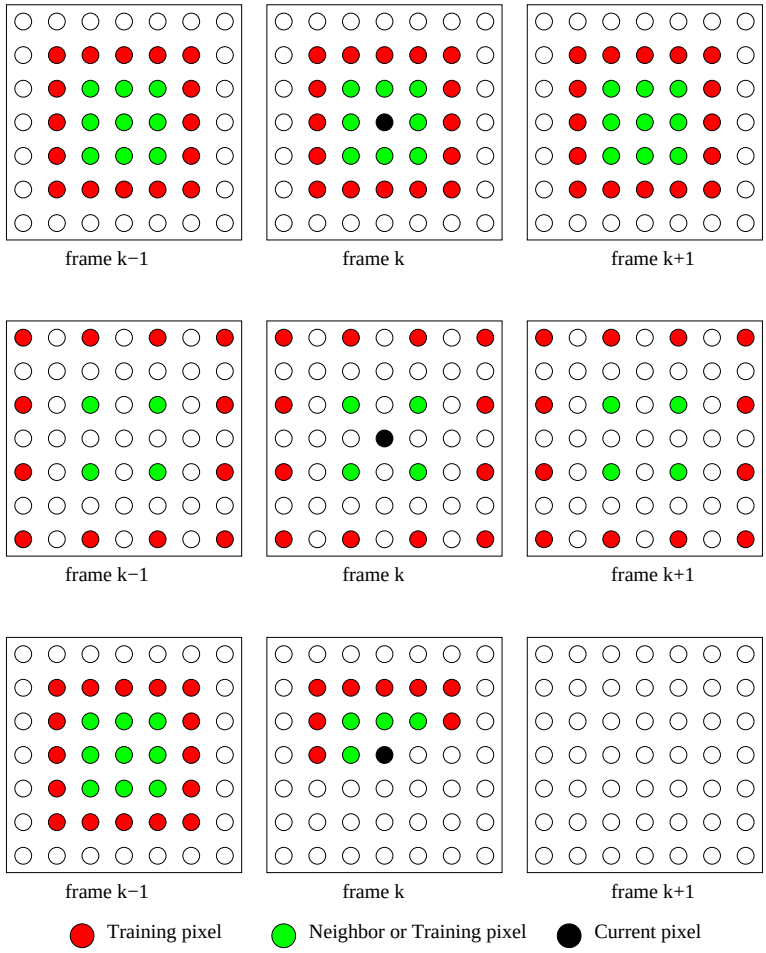


Figure 3.1: Examples of model parameters settings for filtering(Top),interpolation(Middle), and prediction(Bottom)

Under the conventional framework, those problems are attacked by explicitly estimating the motion information, which is then exploited to resolve the intensity uncertainty. However, they cannot always get accurate motion estimation based on these non-ideal observations, which therefore makes the solutions suboptimal.

In this chapter, we will see how the proposed STALL model lead to a class of unified solution algorithms that only differ in the consideration of model parameters in different applications. The performance of derived algorithms will be demonstrated for test sequences with various motion types including camera panning, zooming and nonrigid object motion. Then we will show the modeling capability enhancement

induced by incorporating the spatio-temporal resampling and Bayesian fusion techniques.

3.2 Video Error Concealment

Video error concealment or also known as video inpainting by computer vision community is a missing data problem. The missing data which might be caused by package loss or damage when transmitting a video over an error prone channel[20], or by deliberate removal by human[50]. The goal of error concealment is to unify the missing data with the surrounding available ones by exploiting spatial and temporal redundancies.

Some video error concealment approaches require the side information from encoding procedure. The common side information could be motion vectors, block modes, and/or some other overhead for rubustness[20]. However, when trasmitting over an error prone channel, all these side information could also be lost or corrupted. Although some approaches are developed to recover these side information, especially the motion field firstly[51] and then use them to resolve the intensity uncertainties, the final error concealment performance is highly dependent on the effectiveness of this side information recovery process. Some other approaches switch to spatio-temporal extrapolation[52] or texture synthesis techniques[53] when motion information is lost. However, most of these approaches can often get good perceptual quality but not objective performance. In this section, we apply our STALL to attack the error concealment problem. We design the error concealment process as a post-processing procedure. Thus the additional side information is not required. STALL model can conviniently use the available data for recovery.

There are mainly two issues to be addressed when applying STALL model into error concealment: scanning order (i.e., which missing pixel gets repaired first?) and

parameter settings (i.e., model support and training window). In this section, we only consider the recovery of damaged blocks assuming video is transmitted through a block-based coding system such as MPEG and H.264[17]. The damaged blocks can be isolated or consecutive (we also consider the entire frame loss). However, the proposed technique should be easily generalized to work for repairing a region with arbitrary shape in the spatio-temporal space.

The selection of scanning order has been extensively studied in the scenario of image inpainting[54],[55]. Edges or structures are often given higher priority than smooth regions to be recovered first due to their perceptual importance. As we move to video inpainting, we need to take care of the existence of temporal neighbors in both directions (past and future), which suggests a variety of scanning orders - e.g., unidirectional vs. bidirectional. Since bidirectional interpolation suffers from the increased complexity of implementation due to the interference of missing pixels in training, we choose for each pixel the direction with larger number of training data in that side to get the recovery, because the larger number of training data has higher probability to provide more robust information for recovering.

The key issue in determining model support is causality - given the selected scanning order in space and time, we can only access causal neighbors during the recovery. One example of recovery based on thirteen causal neighbors (four spatial + nine temporal) is shown in Figure 3.1(bottom). After choosing the neighbor $\mathcal{N} = \{\vec{n}_i\}_{i=1}^{N}$ for a pixel located at $\vec{n}_0 = (r, c, t)$, spatio-temporal training set includes all *valid* pixels within the cube $[r - T_s, r + T_s] \times [c - T_s, c + T_s] \times [t - T_t, t - 1]$ (assuming temporally forward direction). T_s and T_t are spatial and temporal extension of the training window. A pixel is declared to be *valid* if its N neighbors defined by $\mathcal{N}$ are all available either from observation or from previous recovery. With the assumption that damaged blocks are dispersed across space and time, a training window should be also across space and time with the size (the total number of valid samples - M) larger

than N. Only under rare occasions (e.g., flat regions), the covariance matrix $D^T D$ in Eq.(2.9) might be ill-conditioned and we need to switch back to ad-hoc interpolation such as local average of the neighbors.

In this section, we are going to compare the error concealment results by our STALL model with the edge-directed image error concealment approach to show how effective our STALL model exploit the temporal redundancy. Then we are also going to compare results of STALL with and without incorporating spatio-temporal resampling and Bayesian fusion techniques to show the modeling capability enhancement by these two techniques. Our test sequences include a variety of motion characteristics which is classified into three categories: 1) object moving (e.g., *container* and *foreman*); 2) camera zoom (e.g.,*mobile* and *tennis*); and 3) camera panning (*garden* and *coastguard*). Except for *tennis* in which the 24th-33rd frames (where camera zoom starts) are used, we ran our test with first ten frames in all sequences.

Three types of error concealment scenarios are simulated for recovering the damaged sequence: 1) isolated $8 \times 8 \times 1$ block loss; 2) spatial consecutive block loss (rectangular stripes with the height of 8 pixels). For the first two cases, we report the comparison between two concealment techniques: spatial interpolation[56] and spatio-temporal interpolation (this work). It can be observed from Table 3.1 that exploiting temporal redundancy significantly boosts the PSNR performance of concealment techniques. Gain for slow motion sequences (e.g., *container* and *mobile*) is larger than that for fast motion sequences (e.g., *garden* and *coastguard*). Figure 3.2 and 3.3 includes the examples of simulated frames sequences with block loss along with recovered frames by different concealment techniques. Visual quality improvements are obvious around the pole region in *container* and texture region in *mobile*.

We show in Table 3.2 that the modeling capability of STALL model is improved by introducing the spatio-temporal resampling and Baysian fusion schemes, that is,

Sequence	8×8 isolated		consecutive	
	S+T	S	S+T	S
coastguard	30.79	22.65	31.36	27.75
container	37.48	22.28	37.14	20.62
garden	20.17	16.75	19.13	16.48
mobile	26.27	15.55	25.93	14.44
tennis	29.63	21.35	26.91	20.40
foreman	35.79	29.36	29.99	25.00

Table 3.1: PSNR(dB) performance comparisons of space only(S) and space+time(S+T) error concealment

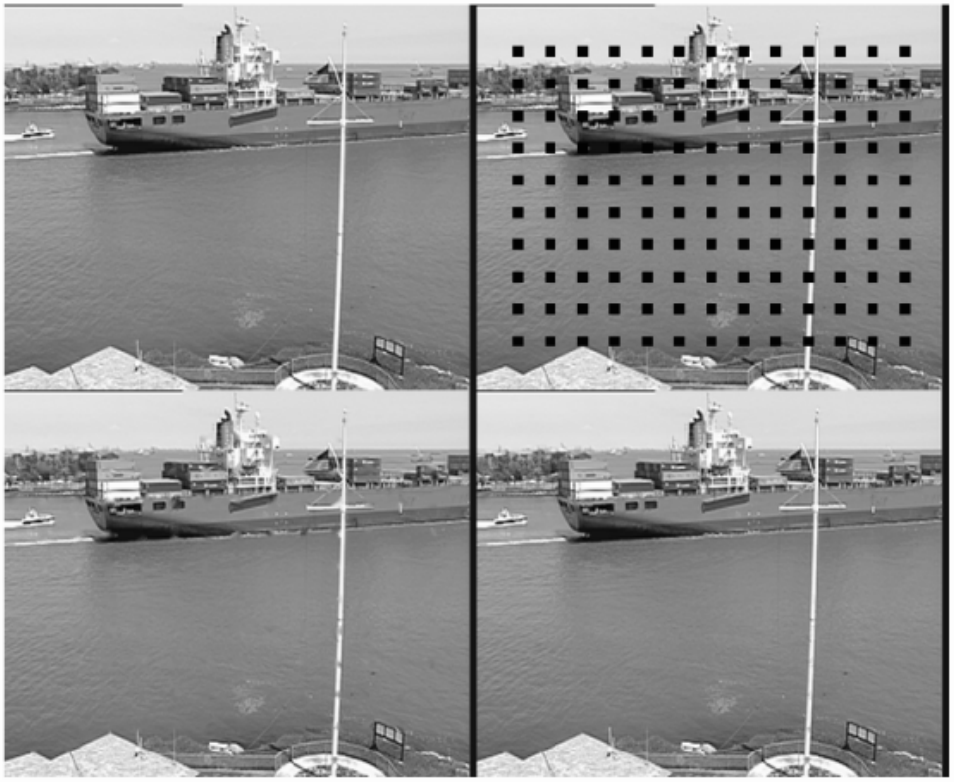


Figure 3.2: Original frame of *container* (up-left); damaged frame of (up-right); recovered frame by S only (bottom-left); recovered frame by S+T (bottom-right)

it can handle a wider range of video signals with more general motion characteristics, such as slow, fast, panning, zoom, and even mixed motions. We use the same model parameters (model support and training window) as the above. The experiment sequence is SIF-*garden* and CIF-*bus* since each of them includes both slow and fast motions. It assumes the block loss at the same spatial location occurs for three

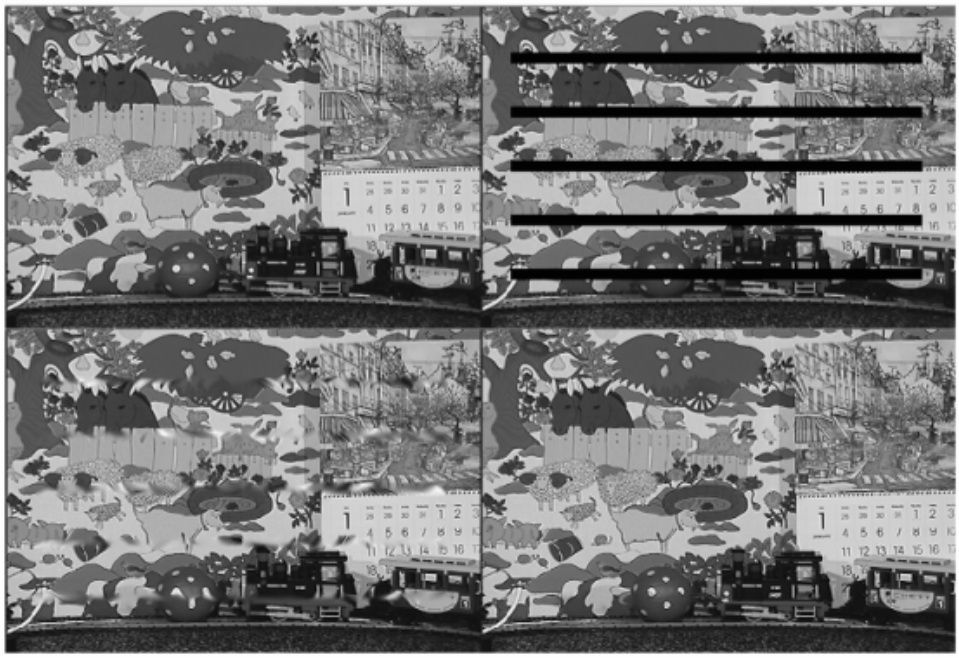


Figure 3.3: Original frame of *mobile* (up-left); damaged frame of (up-right); recovered frame by S only (bottom-left); recovered frame by S+T (bottom-right)

Sequence	Without resample & fusion	With resample & fusion
garden	25.93	31.87
bus	28.34	34.20
mobile	27.77	31.64
tennis	32.54	34.45

Table 3.2: PSNR(dB) performance comparisons of with and without resampling and fusion in error concealment

consecutive frames, which is particularly challenging for fast motion video because the lost content varies from frame to frame due to the fast motion. For camera panning sequences, we use a horizontal virtual camera speed set in which $v_y = 0$; for camera zoom sequences, we use a general virtual camera speed set. The PSNR comparison is shown in Table 3.2 while the visual comparison is provided in Fig.3.4. We can observe the dramatic PSNR and visual quality gains brought by the new introduced schemes.

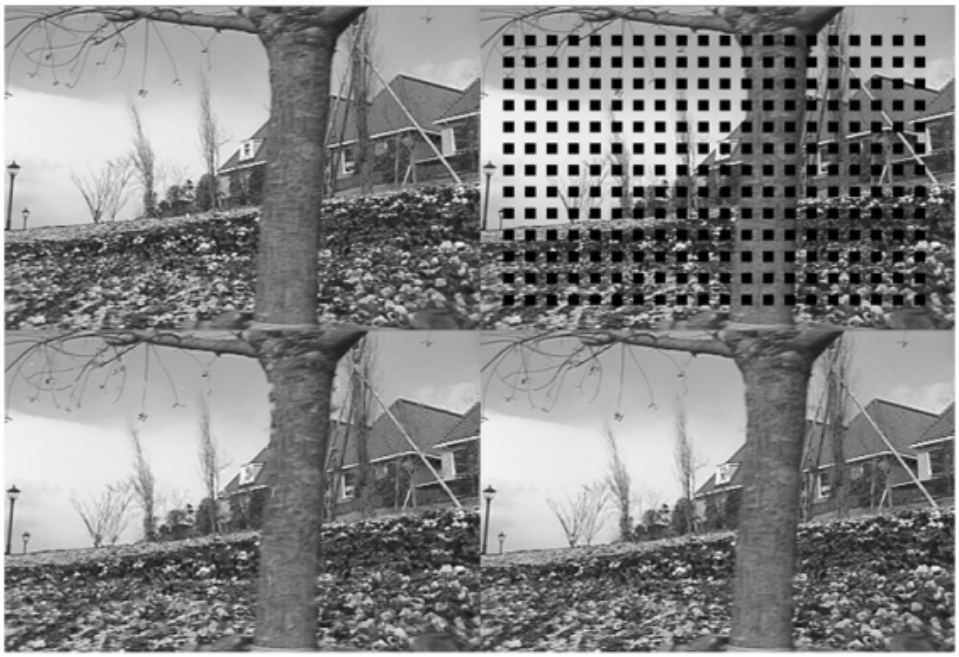


Figure 3.4: Comparison of error concealment results. Top-left: original frame; top-right: damaged frame(note that the previous and next frames are also damaged at the same locations); bottom-left: concealed frame without resampling and fusion schemes(PSNR= 25.46dB); bottom-right: concealed frame with resampling and fusion schemes(PSNR= 30.51dB)

3.3 Video Denoising

In this section, we assume the observed video frames are corrupted by a small percentage of random-valued impulses uniformly distributed between [0,255]. In such a denoising problem, explicit ME becomes difficult since the noisy observation invalidates the noise-free assumption which is depended on by most motion estimation approaches. However, by STALL model, we can attack this problem by easily exploiting clean and filtered pixels while shunning the contaminated ones.

Two steps including noise detection and filtering are often involved to remove impulse noise. The problem of impulse noise detection has been widely studied for still images (e.g.[21]). In our experiments, we assume the availability of noisy detection result, i.e. the correct locations of noisy pixels, since our emphasis is on comparing different filtering strategies in the second step.

In filtering step, we compare the performance of STALL-based filtering with 3D median filtering since median filtering is known to be an effective tool for removing impulse noise[49]. In median filtering, noisy pixel can be replaced by the median of a local spatio-temporal window(e.g. $3 \times 3 \times 3$ space-time window in our simulations); while in STALL-based filtering, noisy pixel is sequentially replaced by the estimate from the STALL model with suitable model parameter settings. Specifically, we give each noisy pixel a filtering priority which is proportional to the number of available (clean or filtered) pixels around it (e.g. within a $5 \times 5 \times 3$ local window in our simulations). Then STALL filter works sequentially from high priority pixel to low priority one, by which the available information is expected to be fully used. To each noisy pixel, all of the available pixels around it within the $3 \times 3 \times 3$ local window shape are its neighbor ($N \leq 26$). With this neighborhood definition, we define that a pixel is *good* if and only if it and its neighbors do not include unavailable (noisy) pixels. Then the training set consists of all *good* pixels within the $5 \times 5 \times 3$ window ($M \leq 74$). In the case when the size of training set is smaller than the neighbor size, i.e. there is no enough pixels for training, we then fall back on the median filtering result.

As shown in Fig.3.5, we can observe that in the light noisy cases(noise percentage is lower than 30%), the filtering performance of STALL model is much better than 3D median filter. With increasing the noise level, more noisy pixels will be filtered by 3D median filter at the begining of denoising because there are few *good* pixel that can be used to establish the training by STALL. More and more pixels can be filtered by STALL when enough pixels become *good*. But these recovered pixels carry errors which can be propogated during localized training. So the final performance of STALl will be degraded and converges to that of 3D median filter as shown in Fig.3.5.

Figs.3.6 and 3.7 include denoising results of two sequences (*coastguard* and *tennis*) containing camera panning and zoom respectively. STALL-based denoising technique

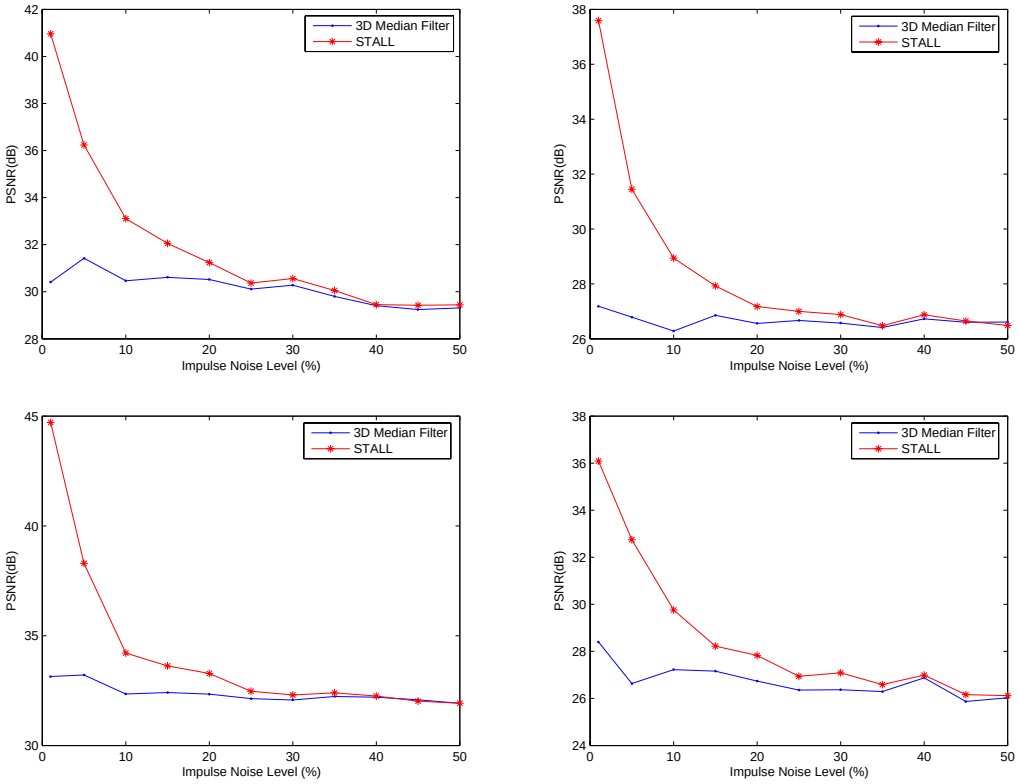


Figure 3.5: Denoising performance vs impulse noise percentage: Top Left - *foreman*, Top Right - *container*, Bottom Left - *silent*, Bottom Right - *table*

achieves around $2.57dB$ and $1.93dB$ better PSNR performance than 3D median filter for *coastguard* and *tennis* respectively. Note that PSNR improvement for *tennis* sequence mainly comes from the background since fast-moving objects (ping-pong ball and arm) are outside the reach of STALL model with fixed filter support and training window, which can be cured by incorporating the resampling and fusion techniques.

The following experimental results show the denoising performance of STALL can be dramatically improved by incorporating spatio-temporal resampling and Bayesian fusion especially for video sequences containing complex or mixed motions. Four test sequences as in video error concealment are used in our experiments: two containing fast camera panning (SIF-*garden* and CIF-*bus*) and two containing camera zoom (SIF-*tennis* and CIF-*mobile*). For camera panning sequences, we use a horizontal

Figure 3.6: Video Denoising result for *coastguard* sequence with neighbor size at most $3 \times 3 \times 3$ and training set size at most $5 \times 5 \times 3$: Top-Left: original frame; Top-Right: noisy frame(10% impulse noise); Bottom-Left: median filtering result (PSNR=38.51dB); Bottom-Right: STALL filtering result (PSNR=41.08dB).

virtual camera speed set where $v_y = 0$; for camera zoom sequences, we use a general virtual camera speed set. Both of them include nine virtual camera speeds. Table 3.3 includes the PSNR performance comparison for impulse noise removal without and with resampling and Bayesian fusion. We observed dramatic improvement ($> 2dB$) brought by the proposed resampling and fusion scheme. Figure 3.8 contains the subjective quality comparison of the denoising results for the *bus* sequence.

3.4 Video Super-Resolution

Video superresolution (SR) addresses the problem of enhancing the spatial resolution of video by exploiting the tradeoff between space and time. Since the sampling

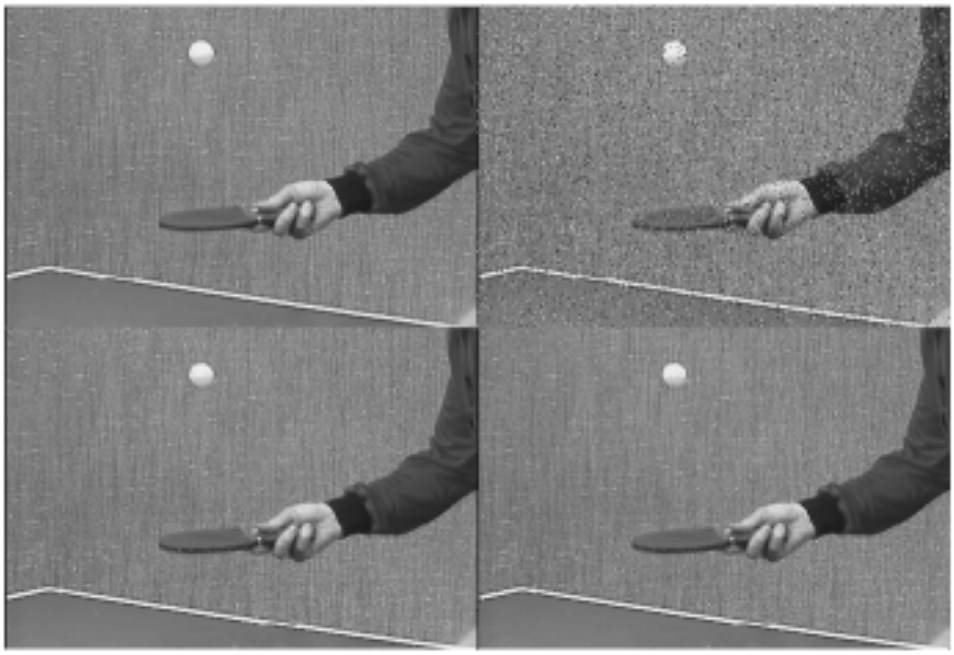


Figure 3.7: Video Denoising result for *tennis* sequence with neighbor size at most $3 \times 3 \times 3$ and training set size at most $5 \times 5 \times 5$: Top-Left: original frame; Top-Right: noisy frame(10% impulse noise); Bottom-Left: median filtering result (PSNR=34.62dB); Bottom-Right: STALL filtering result (PSNR=36.54dB).

Sequence	10% Impulse noise	
	Without resample & fusion	With resample & fusion
garden	32.14	35.54
bus	33.19	36.55
mobile	33.52	37.13
tennis	37.90	39.93

Table 3.3: PSNR(dB) performance comparison between STALL-based impulse noise removal algorithm

lattices induced by the motion along the temporal axis are likely to cover fractional locations, it is convenient to obtain a high-resolution (HR) frame by fusing the data in multiple low-resolution (LR) frames. When compared with optics-based approaches, SR represents a low-cost computational alternative towards HR imaging and has attracted increasingly more attention in recent years.

In this problem, we often assume the observation model that the low resolu-

Figure 3.8: Comparison of Video impulse noise removal results. Top-left: original frame; Top-right: noisy frame (10% impulse noise); Bottom-left:removal result of STALL without resampling and fusion (PSNR= 31.54dB); Bottom-right:removal result of STALL with resampling and fusion (PSNR= 35.01dB)

tion(LR) frame is a filtered and downsampled version of a high resolution(HR) frame, as shown in Fig.3.9. Based on the observation model, there are three steps involved in a lot of SR techniques: registration, interpolation, and restoration, as shown in Fig.3.10[22]. The performance of most existing approaches SR heavily rely on advanced motion estimation (ME) techniques for registration[57], [58], [59] for inferring the subpixel displacement among LR frames. Unfortunately, the accuracy of existing subpixel ME techniques is still limited especially when the motion contained in LR frames is more complex than parametric geometric transformation (e.g., planar homography [60]). Even under simplified assumption with camera motion (e.g., panning only), global planar homography is insufficient for characterizing the relationship among LR frames unless the scene is constrained to a planar object. Moreover, formu-

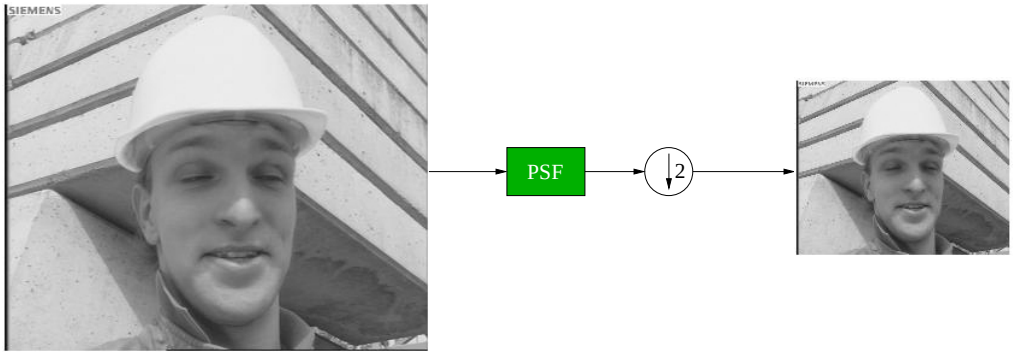


Figure 3.9: The relationship between high resolution and low resolution frame

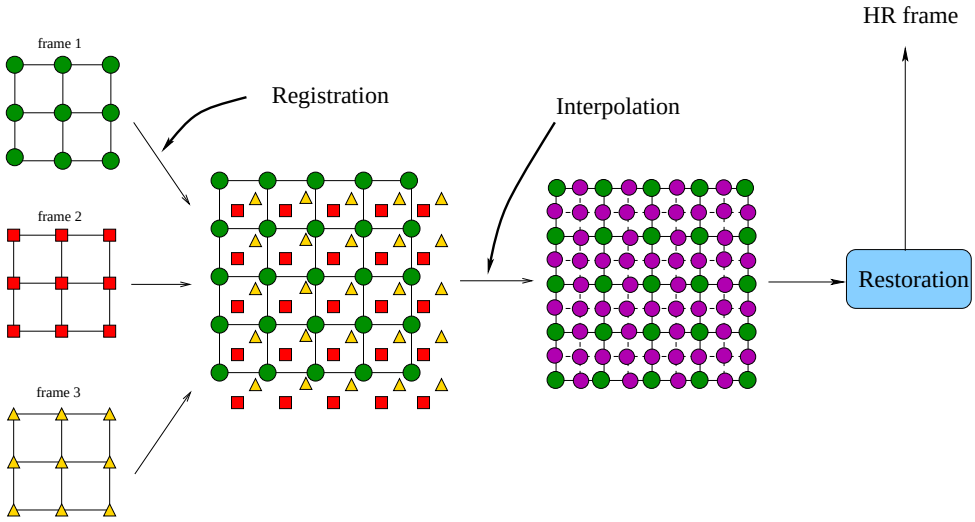


Figure 3.10: Three typical steps in super-resolution problem

lating SR as an inverse problem is at the mercy of observation models whose validity becomes difficult to justify as motion gets complex. How to achieve good performance under generic motion appears to be a major challenge to SR based on our own assessment.

We note that ME is not an indispensable step towards SR. In this section, we apply our STALL model to SR problem, in which no explicit ME involves. Based on the assumption that video is acquired by a stable camera [1], we can exploit the

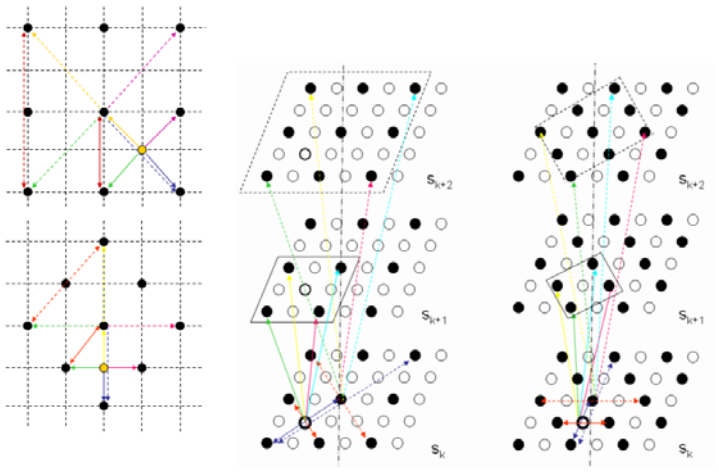
[1]We note that various video stabilization techniques exist in the literature, e.g., [**?**], [**?**]

duality of edge contour and motion trajectory to extend the 2D edge orientation scale invariance to 3D motion trajectory orientation resolution invariance. Thus we can generalize the correspondence of LR and HR covariances observed in [61] from 2D to 3D. Our experimental results will show that our approach achieves not only better visual quality but also higher PSNR performance.

3.4.1 Motion trajectory orientation resolution invariance

The main obstacle with applying STALL model into SR scenario is the missing data problem. How is it possible to obtain HR covariance matrix to drive the spatio-temporal interpolation with only LR data available? The 2D version of such problem was considered in [61] for edge adaptive interpolation of still images. The duality introduced in section 2.2 serves a good foundation for extending the ideas presented in [61] into 3D scenario. In 2D images, the resolution obstacle is overcome by the resolution (scale) invariance of edge orientation; in 3D video, we observe that the orientation of motion trajectory has a similar scale invariance property. Next, we will elaborate on how such scale invariance leads to a new class of spatio-temporal interpolation techniques.

To facilitate the illustration, we opt to briefly review the edge-adaptive spatial interpolation in 2D first. The key motivation behind the work [61] is that when the target of modeling is constrained to a class of regular edges, scale invariance property of edge orientation facilitates the solution to classical Wiener filtering (MMSE estimation) problem at HR suffering from the missing data. An estimation problem at HR can be translated to a dual problem at LR which can be easily solved. An alternative interpretation is that HR covariance estimates are replaced by their LR covariance counterparts as shown in Fig.3.11a. In view of the duality between edge contour and motion trajectory, it is natural to consider a 3D extension of such covariance correspondence.



a) b)

Figure 3.11: Correspondence between LR and HR covariances in a) 2D (from top to bottom: step 1 and step 2) and b) 3D (from left to right: step 1 and step 2): the corresponding pair is denoted by the same color (HR - solid lines, LR - dashed lines).

In spatio-temporal interpolation, the treatment of temporal neighbors should not be separated from that of spatial neighbors. This is because of the interaction between spatial and temporal sampling. As spatial sampling distance doubles, we need to proportionally increase the temporal sampling distance to achieve the scale invariance of motion trajectory orientation (refer to Fig.3.11b). Such adjustment is important to ensure the consistency of spatial and temporal neighbors in STALL model (if we drop the time attribute, the only difference between temporal and spatial neighbors is the location of sampling lattice). We note that the adjustment of temporal sampling distance does not affect the selection of training samples of D but only the support of $\vec{a}$.

3.4.2 Super-Resolution Algorithm

Now, we are ready to present the adaptive spatio-temporal interpolation algorithm. Similar to the algorithm presented in [61], resolution enhancement of video is decomposed into two steps: we first interpolate the pixels along the diagonal direction (i.e.,

r, c are both even numbers) and then interpolate the pixels of the quincunx lattice ($r+c$ is odd) based on the other quincunx lattice ($r+c$ is even). The duality between 2D and 3D scenarios can be clearly seen from Fig. 3.11b. To save space, we only plot the temporal neighbors along one direction. In our implementation, the order of interpolation N is 12, which includes the four nearest spatial neighbors in previous, current and next frames respectively.

When compared to existing ME-based approaches, our spatio-temporal adaptive interpolation based on STALL model has several advantages. First, our assumption about motion type is relaxed. Smoothness constraint along motion trajectory is valid for a wide range of camera motions and object motions. By contrast, the accuracy of ME has been a major limiting factor in the performance of ME-based registration and interpolation especially when complex motion is involved.

Another salient feature of the STALL-based interpolation is its capability of achieving the tradeoff between spatial and temporal dependency. Due to the complexity of motion in generic video material, the dominance of spatial or temporal dependency varies dynamically. For instance, in the case of rigid textures undergoing translational motion, temporal dependency dominates spatial one; while it goes the other way around in the presence of nonrigid textures with deformable motion. More frequently, both spatial and temporal dependency contribute to resolve the intensity uncertainty and it is difficult to achieve a good tradeoff between them under the traditional ME-based framework. By contrast, our STALL model trains the spatio-temporal interpolation coefficients locally, which facilitates the exploitation of such tradeoff.

To illustrate such feature of STALL model, we provide some examples of locally trained interpolation coefficients $\vec{a}$ under three different conditions: temporal-dominant, spatial-dominant and jointly-impacting in Fig. 3.4. The first seven frames of each sequence are used in the training and the 4th frame is assumed to be current.

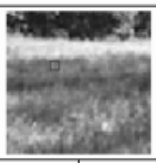
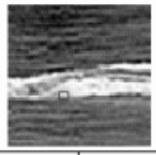
Test points / Support Coeff.			*Forest*		*Coastguard*		*Mobile*	
								
Frame3	a(1)	a(3)	**0.9344**	-0.2217	-0.0828	-0.1098	**0.2978**	0.0529
	a(2)	a(4)	0.0649	**-0.7341**	0.1102	0.1147	0.0807	0.0089
Frame4	a(5)	a(7)	0.0407	0.5579	**0.2714**	**0.3943**	-0.1371	**0.1493**
	a(6)	a(8)	0.0644	0.1028	**0.1898**	**0.3578**	**0.1405**	-0.1262
Frame5	a(9)	a(11)	**-0.5928**	-0.2544	-0.1207	0.0953	0.0053	0.1167
	a(10)	a(12)	0.0828	**0.9655**	-0.1322	-0.0827	0.0106	**0.4031**

Table 3.4: Interpolation based on STALL model achieves the tradeoff between spatial and temporal dependency (coefficients in the table correspond to the marked position in the frame and the four coefficients with largest magnitude are highlighted in bold for each sequence).

The first sequence $forest$ contains the translational motion of rigid texture in which temporal dependency is stronger than spatial one. It clearly shows the dominance of temporal neighbors over spatial neighbors after LS training. The second sequence *coastguard* contains deformable motion of flowing water, for which we observe the dominance shifts to spatial neighbors instead. For the translating edge in the third sequence *mobile*, both temporal and spatial dependency is useful and we observe a nearly uniform distribution of weights along the direction of edge contour and motion trajectory.

3.4.3 Experimental Results in Super resolution

In following experiments, the STALL model order N of spatio-temporal adaptive interpolation is chosen to be 12 (four surrounding neighbors in the previous, current and next frame respectively); the size of training window is set to be 147 (a $7 \times 7 \times 3$ cube). Such fixed interpolation support and training window is for the simplicity of implementation and sufficient for most video sequences with slow motion. To handle smooth areas which often give rise to ill-conditioned covariance matrix, we check the

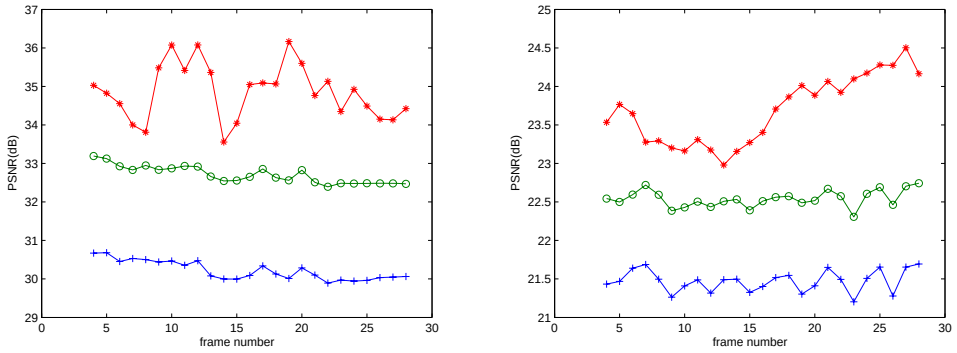


Figure 3.12: PSNR comparison among bilinear ('+'), cubic spline ('o') and this work ('*') for *foreman* (left) and *mobile*(right) sequences.

condition number of $D^T D$ and use the local average as the interpolated value when $D^T D$ is ill-conditioned. The implementations of two steps described in this section are similar as we can see from Fig.3.11. For color sequences, the luminance and chrominance channels are independently interpolated. In Fig.3.12, 3.13, and 3.14 we compare the performance of our spatio-temporal adaptive interpolation scheme with the space only (bilinear, cubic-spline, and edge-directed interpolation[61]) for showing the improvements due to temporal information. We also compare our results with current state of the art work[48] in Fig.3.14. Note here we add a simple restoration step after the STALL-based interpolation for fair comparison. This step includes a LS-based blurring kernel estimation with considering phase and a deburring step. It can be observed that ours achieve the best performance for *forest* sequence which does not contain severe spatial aliasing.

3.5 Conclusion

In this chapter, we applied our STALL model into three applications, i.e., error concealment, denoising, and super-resolution. All of the algorithms in these applications are located in the same framework, which is constructed by the STALL-based im-

plicit video model. We showed that the only difference between these algorithms is the model parameter settings. During the demonstrations, the flexibility and capability of STALL model are exhibited. We also showned the performance enhancement by incorporating the space-time resampling and Bayesian fusion techniques in interpolation and filtering problems.

Compared to the algorithms based on explicit motion models for the same applications, our STALL model shows the following advantages: first, STALL model achieves good tradeoff between spatial and temporal information through localized learning, which is important for handling more complex motion cases, like occlusion and texture; second, STALL model implicitly embeds the spatio-temporal adaptive interpolation by LS training, which is favorable for many applications like super-resolution[22] and video coding[40]; moreover, STALL model can easily handle the non-ideal observation data problems by circumventing the noisy or missing data in localized learning. All of these advantages contribute the performance of STALL model for handling low-level vision problems in video processing.

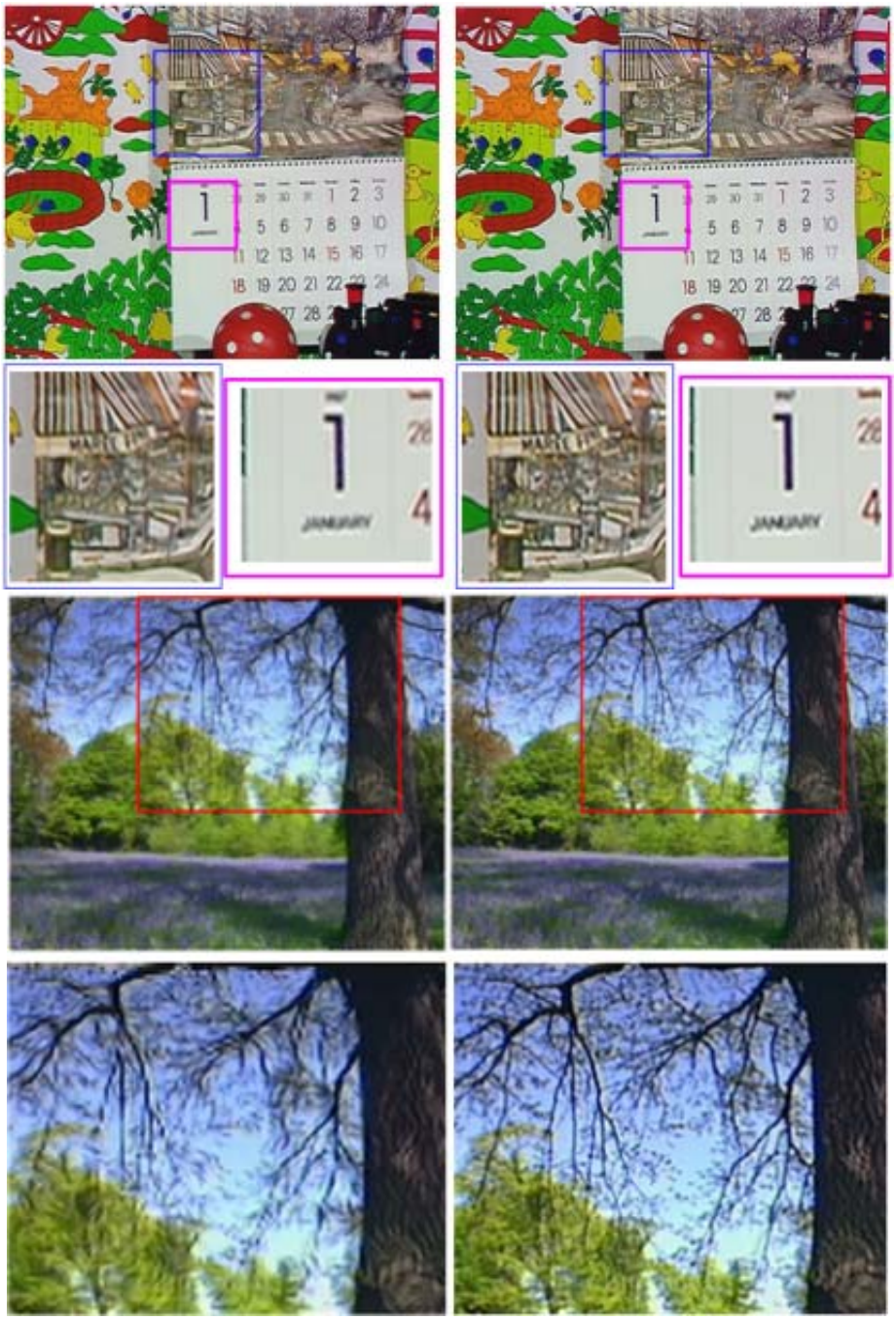


Figure 3.13: Reconstructed HR images and zoomed portions by spatial-only interpolation (left) and spatio-temporal interpolation (right). Top: *mobile*; Bottom: *forest*

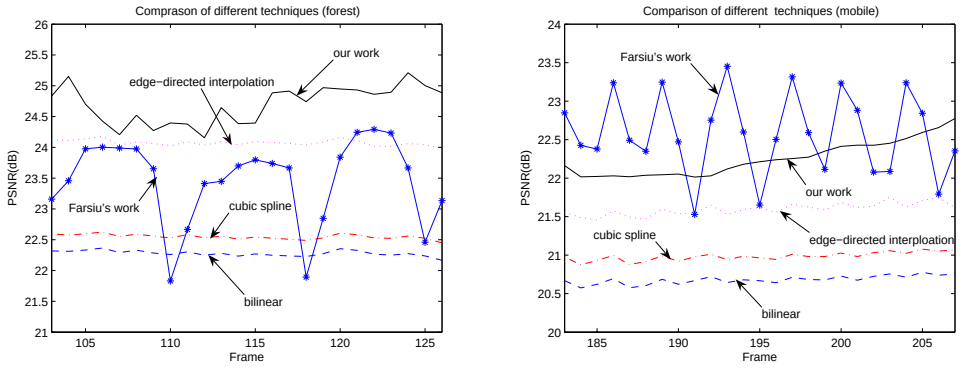


Figure 3.14: PSNR comparison among spatio-only interpolation (bilinear, cubic spline, edge-directed interpolation [61]), UCSC-SR (Farsiu et al.) [48] and this work for *forest* (left) and *mobile*(right) sequences.

Chapter 4

Video Modeling via Patch-based Sparse Representation

4.1 Introduction

The performance of parametric models is highly dependent on the matching between the observation data and the model[62]. Although parametric models can provide us an approach for accurately understanding video signals, their limitations are also well-known. Parametric models always assume the observation data are drawn from an underlying distribution, such as *Gaussian*. The modeling error will occur when the observation data are either out of the assumption or noisy. The localized adaptive strategies (like STALL) can alleviate the problem to some extent but at the price of rapidly increasing complexity. In video modeling, due to the complexity of natural motion[1], sometimes, the task is extremely difficult to assume an underlying distribution or distribution form of the observation data. And we also often suffer from

[1] Although the motions in physics are well defined, they have not been well defined when projecting to the imaging plane to form a video signal.

the noisy or incomplete observation data, which make the problem harder to solve by parametric models.

Fortunately, data-driven nonparametric modeling approaches can provide us some ways to circumvent the dilemma. Nonparametric models differ from parametric ones in that the model is not dependent on any underlying distribution or distribution form but is instead determined from observed data. As nonparametric models often make fewer assumptions, their applicability is much wider than the corresponding parametric methods. Also due to the reliance on fewer assumptions, non-parametric models are more robust to noisy observation data. The typical non-parametric models are k-nearest neighbor(kNN) and kernel regression technique, which are widely used in image and video processing[63] and computer vision problems[64]. In video modeling, some nonparametric models, such as [27],[65], appeared recently and exhibited promising performance in handling low-level vision problems. In this chapter, we propose another nonparametric video modeling approach, which does not depend on any underlying distribution assumption of the observation data.

The proposed nonparametric video modeling approach is neither dependent on explicit motion representations. In STALL model, the motion information is embeded into the relationships among local neighboring pixels. Unlike STALL, here we embed the motion-related information into the relationships among video patches[66]. We define a patch as a connected pixel set which can cover both space and time within a video signal, which is a 3D extension of a 2D block. We consider the 3D patches as the atoms of video representation. The patch space is then established by defining the distance measurement between patches. We assume a video sequence is composed of overlapping space-time patches. Based on these definitions, we extend the nearest neigbhor search (e.g. block matching) to k nearest neighbor search (i.e. patch clustering), through which motion related information is represented in an implicit and distributed fashion.

In our approach, we apply a generic sparsity-based prior for typical video sequences. The sparsity constraint of a video patch cluster is exploited by increasing the dimensionality. Such counter-intuitive strategy is the consequence of patch clustering that makes the signal representations adapt to fixed bases instead of basis pursuit[67](find bases to adapt fixed signal representations). We propose a weighted average strategy of fusing diverse inference results under patch-based video models. Most existing works on patch-based image/video processing[27][68] simply take the uniform average of overlapped regions as the final outcome. We show this is a special case of general weighted averaging that could handle arbitrary set of hypotheses in redundant representations. In our fusion strategy, we will show how to exploit the location and sparsity information to estimate weighting coefficients.

When applying the above model to the problems involving non-ideal observations, we enter a dilemma where on one hand, it is hard to obtain an accurate patch clustering result because of the non-ideal observations (e.g. noisy or incomplete data); on the other hand, we have to resolve the uncertainty through patch clustering. In our approach, we propose to treat both of them as latent variables and solve the Bayesian inferenence problem via an iterative algorithms.

The rest of this chapter is organized as follows: in section 4.2, we extend block matching which is widely used in explicit motion representations into patch clustering and highlight the benefits of such generalization. Then in section 4.3, we extend the block-based motion compensation into a sparsity-based constraint in the patch space. We also show in section 4.4 the patch clustering and the sparsity-based constraint jointly specify a generic prior for video to support Bayesian inference. Finally in section 4.5, we propose the algorithms for iteratively solving the inference problem in restoration applications. Section 4.6 summarizes the patch-based sparsity model.

4.2 Patch Clustering

A patch is a connected pixel set which can cover both space and time within a video signal. If the time dimension is restricted to one and the spatial topology is rectangular, a patch is a 2D block that is widely used in motion compensated video processing. In our approach, we keep the spatial rectangular restriction for a patch but allow the time dimension be larger than one. Thus a patch is a 3D cube which is the atom of our video representation, as shown in Fig.4.1. Actually, the spatial rectagular restriction of a patch can also be released which makes the arbitrary spatial shaped patches. Although the arbitrary shape can help us to get sparser representations of the signal which we will introduce in the following sections, to define suitable spatial shape of a patch need additional image information as priors, which is usually not available or hard to obtain in a lot of tasks. So in our approach, we keep this rectangular restriction. We assume the dimension of a patch as $N = (2T_r+1)\times(2T_c+1)\times(2T_t+1)$, where T_r, T_c, T_t are the patch size parameters in row, column, and time repectively. With boundary extension and maximum overlapping, we can obtain $M = \prod_{i=1}^{3} N_i$ patches (N_1, N_2, N_3 are the dimension of the video) and store them into a collection $\mathcal{P}$ which we call *patch database*.

With the patch definition, we assume a patch as a symbol in a patch space with dimension of N. The patch space is defined under a certain distance measurement. Since the goal of our patch clustering is to exploit the repetitive spatio-temporal patterns within a video signal, a suitable distance measurement is important. The distance measurement is not unique. It can be defined in spatial domain like the common-used sum of squared error (SSE) or sum of absolution difference (SAD) of color information between two cubes. Or it can be defined in transform domain to better capture the periodic patterns for textures. Furthermore, it can be defined by some function which combines the measurement in both spatial and transform domain. Although it is also an interesting work to mining features, it is not the main

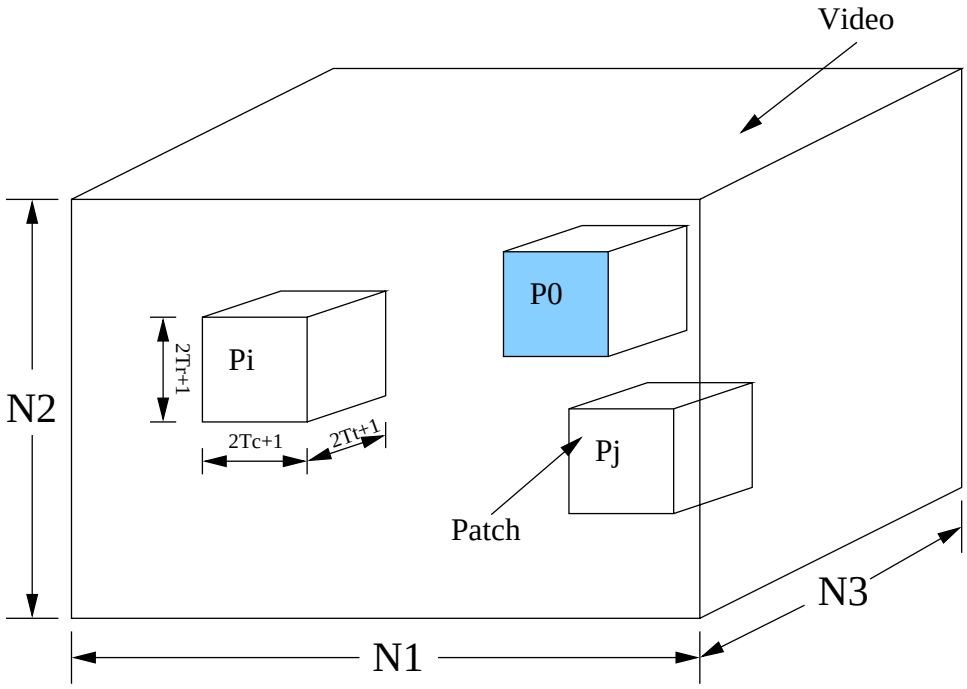


Figure 4.1: Video and Patch

work in our model. So for simplicity, we choose spatial domain approach to measure the patch distance by taking the spatial and temporal continuities into account, because human vision system (HVS) is also sensitive to motion.

So we define a new distance measure with taking motion into account. We denote a 3D space-time patch by P_i with size N, the subscript i represents the index in the whole patch database $\mathcal{P}$ that consists of all overlapping space-time patches of the video sequence. Secondly, we use the normalized L^2-norm of the difference of two mean removed patches as the criteria to calculate the patch distance:

$$d(P_i, P_j) = \frac{||(P_i - \bar{P}_i) - (P_j - \bar{P}_j)||_2}{\sqrt{N}}; \tag{4.1}$$

where $d(P_i, P_j)$ is the distance between patch P_i and P_j, $\bar{P}_i$ and $\bar{P}_j$ are the mean of them respectively. The mean value subtraction allows for improved matching of patches with similar structure but different mean values since HVS is shown more sensitive to changes. Fig.4.2 illustrates an example(in 1D) that very different variation

behaviors can lead to the same SSE score[50]. The function $f(t)$ has a noticeable change. Yet, its SSE score relative to a similar looking function $g_1(t)$ is the same as the SSE score of $f(t)$ with a flat function $g_2(t)$: $\int (f-g_1)^2 dt = \int (f-g_2)^2 dt$. However, perceptually, $f(t)$ and $g_1(t)$ are more similar, as they both encode a change.

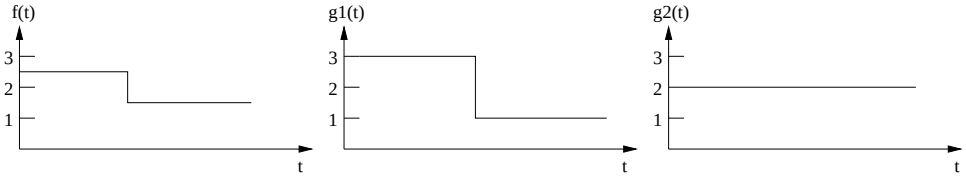


Figure 4.2: Example of change sensitivity

With the definitions of patch and patch distance, we can extend the block matching into patch clustering. Instead of searching for the best motion vector (nearest neighbor search in patch space), we look for k nearest neighbors (kNN) of a given patch P_0. The searching window can be the whole patch database $\mathcal{P}$. However, for simplicity, we search for them within a relative local window $\mathcal{W} = [-K_r, K_r] \times [-K_c, K_c] \times [-K_t, K_t]$, where K_r, K_c, K_t are the ranges of searching window extending over row, column and time respectively. If kNN search within the searching window $\mathcal{W}$ can successfully obtain k nearest neighbors, we can get a similar patch set $S(P_0)$ with k patches for each given patch P_0. If kNN search is performed for each patch in patch database $\mathcal{P}$, we can have an array $\mathbf{T}$ sized $N_1 \times N_2 \times N3 \times k$, which we call "kNN array". Based on the patch distance definition, we define the following similar patch set of a patch P_0:

$$S(P_0) = \{P_i \in \mathcal{P} | d(P_i, P_0) < \tau_{match}\} \tag{4.2}$$

In order to avoid obvious mismatch being included into the similar patch set $S(P_0)$ due to non-ideal observations, we set a threshold τ_{match} in Eq.(4.2) to pick out the erroeous patches during clustering. Usually the size of the similar patch set defined by Eq.(4.2) can be very large because of the highly redundant characteristics of a video.

Although the more similar patches (i.e. larger k value), the more information can be exploited to achieve better modeling performance once we set suitable threshold τ_{match}. However, we need to pay for it by the incurring computational cost. In our work, we fix the maximumn value of k, which is about 30 and usually enough for our test video.

Although the extension from 2D block to 3D patch increase the dimensionalty of the patch space, 3D patches can achieve better match than 2D blocks because of the increased matching features. We can show this by the following simulations. If we fix the similar patch set size to 30, we perform the clustering based on 2D block and 3D patches and calculate the average matching error for several sequences. As shown in Fig.4.3, we found 3D patch clustering is always more accurate than 2D block clustering in these sequences. Another advantage of the extension from 2D patch to 3D patch is the increased overlapping possiblities. Since the overlapping will provide addtional hypotheses for current pixel which is really helpful for restoration tasks, we prefer more overlapping on a pixel. As shown in Fig.4.4, if we use 2D patch, the overlapped patches are only from spatial matches (i.e., the temporal matched patches cannot overlap at current pixel). But if 3D patch is used, spatial and temporal matched patches will be possible to contribute addtional hypotheses for current pixel. Furthermore, 3D patch clustering takes the temporal constraint into consideration, which can arguably better characterize the motion-related temporal dependency than 2D block.

Another extension in our work is to extend the conventional block matching into the patch clustering. The conventional block matching assigns a probability of one to the established correspondence; by contrast, kNN-based patch clustering provides a distributed representation of motion information, which in turn will enjoy the benefits brought by such multiple hypotheses[3]. Another advantage of considering kNN is to facilitate the exploitation of sparsity constraint by sparsifying transforms as we discuss

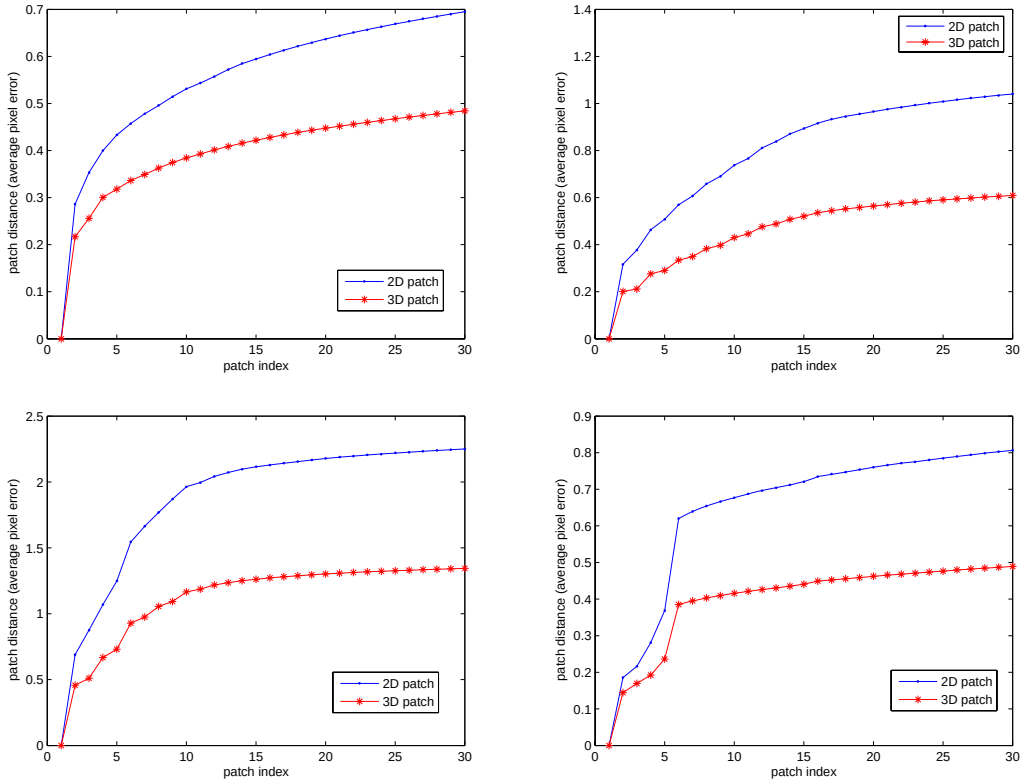


Figure 4.3: Accuracy evaluation of 2D and 3D patches: Top Left - *foreman*, Top Right - *coastguard*, Bottom Left - *garden*, Bottom Right - *tennis*

in the next section.

4.3 Sparsity Representation

The extension from the nearest neighbor search based block matching to kNN-based patch clustering provides us a distributed way to represent motion information. The similar patch set $S(P_0)$ usually includes both local and nonlocal information. But the problem is how to make use of these information to facilitate our modeling. When kNN of a given patch is available, it is natural to consider the linear combination as an improved higher-order approximation:

$$\hat{P}_0 = \sum_{i=0}^{k} w_i P_i \tag{4.3}$$

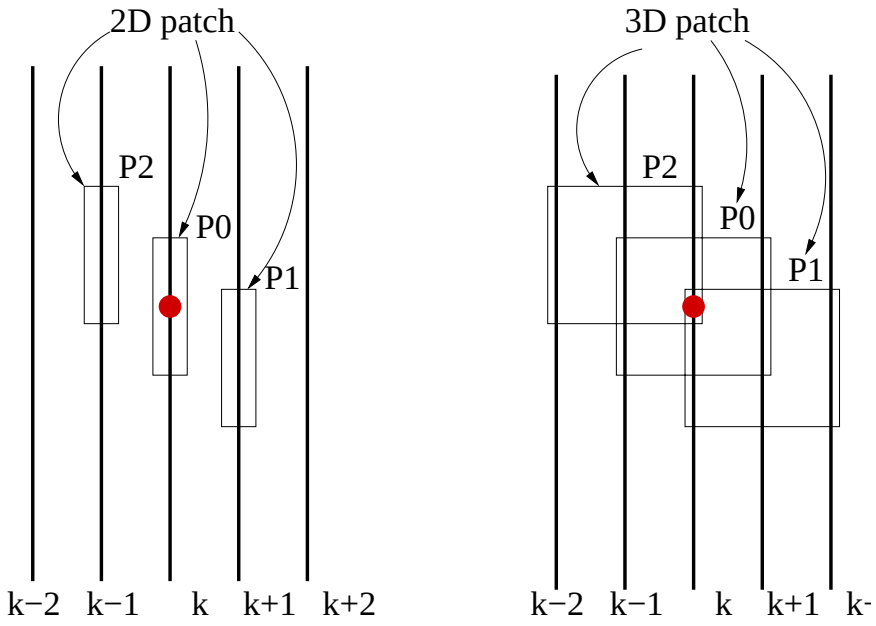


Figure 4.4: The overlapping possiblilty comparison between 2D and 3D

where P_i is a patch within the similar patch set $S(P_0)$, the weighting coefficients $\{w_i\}$ can be solved by standard Least-Square(LS) method. Such linear expansion strategy is at the fundation of locally linear embedding(LLE)[69] and also related to our STALL model in previous chapters.

However, since the atom in this model is patch not pixel, a patch P_i could include corrupted or incomplete data, which is hard to avoid like that in our STALL model. Once the outliers or noisy data contaminate the training data, the standard LS approach could lead to an unreliable result. One remedy is to formulate a total least square(TLS) as pursued in [70]. Inspired by the recent advance in patch-based image restoration(e.g., BM3D[71] and [72],[73]), we propose an alternative transform-based approach of obtaining a sparsity constraint in the patch space. Specifically, the k patches in the similar patch set are sorted firstly based on the clustering distortion in Eq.(4.1) in order to maximize the sparisty. Then we pack the k patches into a four dimensional data array $D = [P_0, P_1, P_2, \ldots, P_{k-1}]$. We denote the fourth dimension of D by s, which implicitly stores the motion-related information - i.e., the entries in the kNN array $\mathbf{T}$ store the spatio-temporal locations of similar patches. Because array D

contains the patches with high similarity, it shows high correlation along the fourth dimension. That means it can be decorrelated by standard bases(e.g., FFT,DCT,DST, and wavelet transforms) to obtain a sparse representation. We define the following four dimensional transform:

$$D_T = \mathcal{T}D = \sum_{x=0}^{2T_c+1} \sum_{y=0}^{2T_r+1} \sum_{t=0}^{2T_t+1} \sum_{k=0}^{K-1} D(x,y,t,k)\phi(x,y,t,k) \tag{4.4}$$

where (x, y, t, k) is the coordinates of a pixel in a similar patch set D. The tranform $\mathcal{T}$ can be aribitrary four dimensional transform that can sparsely decompose the signal D. But based on our experience, the choice of the transform is not very critical for the whole modeling procedure. It is because that the patch clustering makes the fourth dimension almost stationary, which is very easy to acquire a sparse signal even using simple standard basis, like Fourier basis.

At the first sight, such sparsification of data arrays is counter-intuitive because the dimensionality and the number of pixels apparently increased. However, the majority of transform coefficients in transformed signal D_T will be small due to patch clustering, thus we still can achieve sparsification. When compared with the previous works on either pre-fixed transform or adaptive bases[72],[74] such dimensionality increase strategy can be viewed as a hybrid approach - we still use pre-fixed bases but achieve adaptation by transforming signal representations with the aid of patch clustering result $\mathbf{T}$. In other words, instead of adapting bases to the signal in a space with same dimension, we adapt the signal to bases by representing it in a higher dimensional space where basis pursuit is not needed any more.

With the sparse representation in transform domain, the sparsity constraint can be applied by coring operators[75] (e.g., Wiener filtering[76], soft/hard thresholding[77],[73]) to restore a collection of clustered patches, as shown in Fig.4.5. The generic information of a similar patch set is extracted and the disturbing noise can be removed

through coring operation.

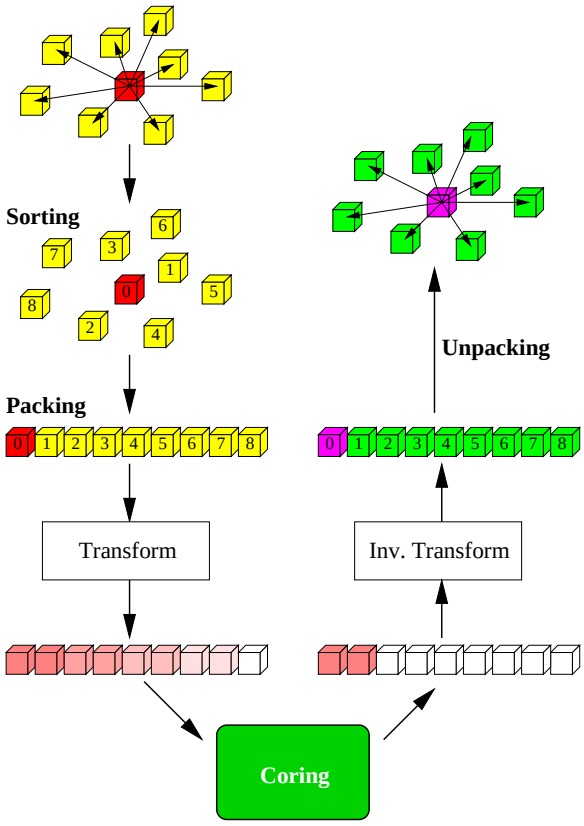


Figure 4.5: Enforcing sparsity constraint in patch level

4.4 Bayesian Inference with Sparsity-Based Priors

In this section, we are going to solve inference problem in pixel level. So we use x,y to denote a pixel in idealistic (i.e., clean and complete) and realistic (e.g., noisy or incomplete) observation respectively. Many restoration tasks can be posed as Bayesian estimation problems and it is desirable to pursue a maximumn posterior (MAP) solutions. In our model, we exploit the patch clustering to manage video patches adaptively. The clustering index $\mathbf{T}$ can be updated based on the signal. By treating both x and $\mathbf{T}$ as two latent variables, we can approximately solve the MAP estimation problem by iteratively update the estimate of x and $\mathbf{T}$. In the following,

we will first assume $\mathbf{T}$ known and derive the Bayesian Least Square estimation $E(x|y)$ and then present a iterative approach to deal with unknown $\mathbf{T}$.

If the ground truth of $\mathbf{T}$ is known, the restoration problem is reduced to obtain Bayesian Least Square estimation of x given y, which is $E(x|y)$. Since the patches in our model can overlap with each other, one pixel can be included into multiple patches, each patch can be further included into multiple similar patch sets, as shown in Fig.4.6[2]. After enforcing sparsity constraint for each of these similar patch sets, each set will provide at least one processed version of the pixel, which we call a **hypothesis** of the pixel. In the literature(e.g.,[68], [70]), the ad-hoc uniform averaging strategy is often adopted to fusing the multiple hypotheses. In our work, we present a weighting average approach under the Bayesian framework.

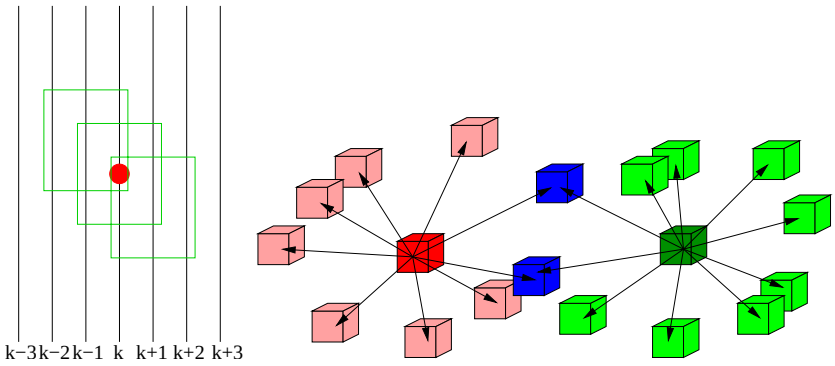


Figure 4.6: Source of multiple hypotheses of a pixel

We note that multiple hypotheses arise at two levels in our patch-based video model: 1) in the pixel level, any pixel x can be covered by more than one patches, say $\{P_z\}_{z=0}^{Z}$; 2) in the patch level, any patch P_z can be associated with multiple data arrays, say $\{D_{z'}\}_{z'=0}^{Z'}$. The posterior distribution $p(x|y)$ can be formulated as

$$p(x|y) = \sum_{z=0}^{Z} p(x, P_z|y) = \sum_{z=0}^{Z} p(x|P_z)p(P_z|y), \tag{4.5}$$

[2]For simplicity of illustrating the patch overlapping, we assume the patch in the left plot is 2D, the data array D is 3D. 3D patch case can be easily extended.

since y is the nonideal version of x, $y \in P_z$. Based on Bayesian rule, the probablity $p(P_z|y)$ is given by

$$p(P_z|y) = \frac{p(y|P_x)}{\sum_{w=0}^{Z} p(y|P_w)}, \tag{4.6}$$

where $p(y|P_z)$ is the likehood function. Since we have no further prior information of difference patches, we adopt uniform prior $p(P_z)$ in Eq.(4.6).

Similarly, in patch level, the probability $p(x|P_z)$ can be given as

$$p(x|P_z) = \sum_{z'=0}^{Z'} p(x, D_{z'}|P_z) = \sum_{z'=0}^{Z'} p(x|D_{z'})p(D_{z'}|P_z), \tag{4.7}$$

where $p(D_{z'}|P_z)$ is given by

$$p(D_{z'}|P_z) = \frac{p(P_z|D_{z'})}{\sum\limits_{w'=0}^{Z'} p(P_z|D_{w'})}, \tag{4.8}$$

where $p(P_z|D_{z'})$ is the likelihood function and a uniform prior of $p(D_{z'})$ is adopted as in pixel level.

Substituting Eq.(4.7) into Eq.(4.5), we can have the following simplified result

$$p(x|y) = \sum_i a_i p(x|D_i), \tag{4.9}$$

where subscript i stands for the $i-th$ hypothesis (it is the compounding result of pixel level and patch level diversities) and weighting coefficient is

$$a_i = p(P_z|y)p(D_{z'}|P_z). \tag{4.10}$$

Multiplying both sides of Eq.(4.9) by x and taking the integration, we have

$$E[x|y] = \sum_i a_i E[x|D_i], \tag{4.11}$$

which relates the Bayesian Least Square estimate at the pixel level to that at the patch level.

The estimate of the weighting coefficient $\{a_i\}$ is very important to the fusion of multiple hypothese. In Eq.(4.10), we observe that the contributions to a_i come from pixel($p(P_z|y)$) and patch($p(D_{z'}|P_z)$) level. In our work, denoting the location of pixel x in the video by (r, c, t), the posterior probablity $p(P_z|y)$ is modeled by a function of the pixel's relative location $(\delta r, \delta c, \delta t)$ within a patch, say $f(\delta r, \delta c, \delta t)$. For example, we can set larger probabilities to the pixels in the center of a patch than those around the patch boundaries(e.g., using *Gaussian* or *Bessel* function); or we can just use uniform value over the pixels within a patch. In our simulations, we use uniform function to model $p(P_z|y)$ for simplicity. In patch level, the posterior probability $p(D_{z'}|P_z)$ is modeled by a function of the sparseness of an array $D_{z'}$, say $g(sp(D_{z'}))$, where $sp(D_{z'})$ means the sparseness of $D_{z'}$. Since we observe that the sparser a data array is, the more influence it should have on the final outcome. For example, we can approximate $p(D_{z'}|P_z)$ by $\frac{1}{N_{nz}}$, where N_{nz} denotes the number of significant coefficients in $\mathcal{T}D_{z'}$. Therefore, the weighting coefficients in Eq.(4.10) and Eq.(4.11) are given by

$$a_i(r, c, t) = f(\delta r, \delta c, \delta t)g(sp(D_{z'})), \tag{4.12}$$

Where we leave the function $f(\delta r, \delta c, \delta t)$ and $g(sp(D_{z'}))$ generic here, though we use $g(sp(D_{z'})) = \frac{1}{N_{nz}}$, and uniform $f(\delta r, \delta c, \delta t)$ in our simulations.

The above derivation assumes a known $\mathbf{T}$, which is not feasible in restoration problem due to the non-ideal observations. Therefore, we have to treat it as another variable just like x. Motivated by recent advances on variational Bayesian methods

(e.g., [78]), we can formulate the following dual MAP estimation problems

$$\begin{aligned} \hat{\mathbf{T}} &= \underset{\mathbf{T}}{argmax} \sum_{x} p(\mathbf{T}, x|y) \\ \hat{x} &= \underset{x}{argmax} \sum_{\mathbf{T}} p(x, \mathbf{T}|y) \end{aligned} \tag{4.13}$$

which can be solved by an iterative algorithm by keeping updating $\mathbf{T}$ and x. Specifically, we first conditionally search kNN at iteration $n+1$ based on the result at iteration n to get $\hat{\mathbf{T}}^{(n+1)}$. Based on $\hat{\mathbf{T}}^{(n+1)}$, we can calculate $\hat{x}^{(n+1)} = E[x|y]$ by the procedure illustrated in Fig.4.5 and Eq.(4.11). Because the $\hat{x}^{(n+1)}$ is likely better than $\hat{x}^{(n)}$ and $\hat{\mathbf{T}}^{(n+1)}$ is also likely more accurate than $\hat{\mathbf{T}}^{(n)}$, we can hope the algorithm converge to the final inference x^*. As can be seen from our experimental results, the iterative algorithm does converge.

4.5 Model Structure

We have already introduced all the components in our nonparametric model in previous sections. In this section, we will connect all the components and show the modeling flow. Then we are going to present the algorithm structure that is depended by the applications in next chapter.

As shown in Fig.4.7, we first perform kNN-based patch clustering for each processing patch to get the kNN array $\mathbf{T}$. Then for each patch cluster, as shown in Fig.4.5, we generate a 4D data array D by packing all similar patches and enforce the sparsity constraint on D by forward transforming, coring, and inverse transforming. Then we unpack the processed patches from data array $\hat{D}$(processed one) and send them back to the original locations. After all patch cluster being processed, combine the multiple hypotheses to get the final inference by weighting averaging.

Although the details of algorithm are highly application dependent, we still pre-

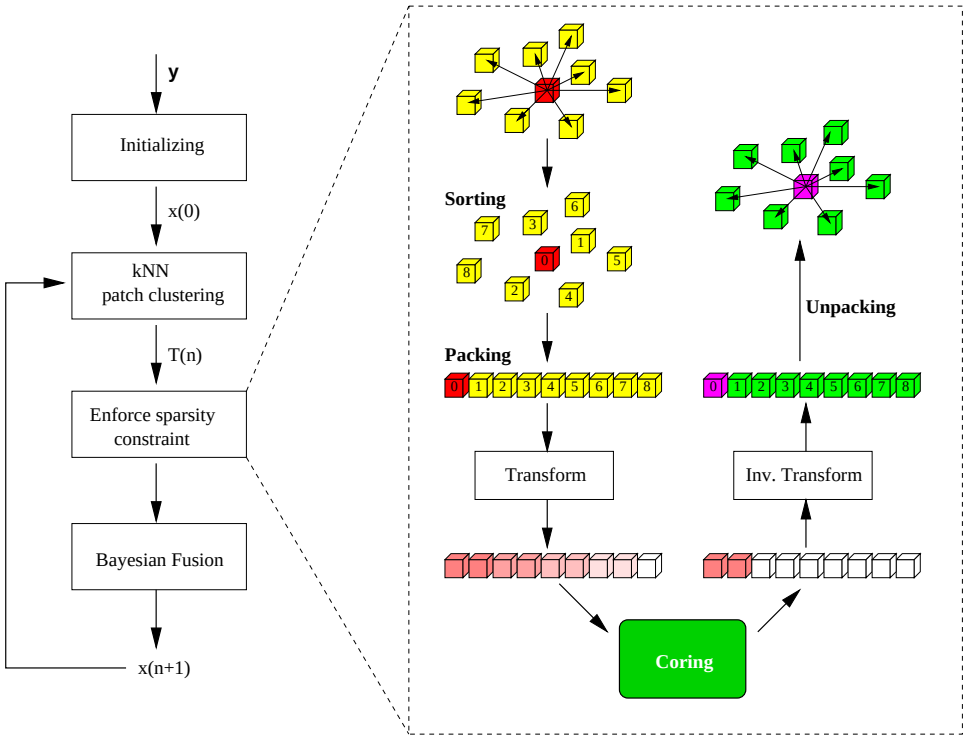


Figure 4.7: Patch-based sparsity video modeling

senst the algorithm structure in Algorithm 1 to help understanding the basic procedure of solving x and $\mathbf{T}$ iteratively. The detailed algorithms and implementations for each application will be elaborated in the next chapter.

4.6 Conclusion

In this chapter, we propose a nonparametric video modeling approach, which is not dependent on explicit motion estimation. Assuming the video sequence is composed of a lot of overlapping space-time patches, we propose to embed motion-related information into the relationship among video patches and develop a generic sparsity-based prior for typical video sequences. First, we extend block matching to more general kNN-based patch cluster, which provide an implicit and distributed representation for motion information. We propose to enforce the sparsity constraint on a higher-dimensional data array signal, which is generated by packing the patches in the similar

Algorithm 1 Nonlocal Patch-based Video Modeling

- Initialization: set $\hat{x}^{(0)} = y$ and perform kNN search for patches to obtain $\hat{\mathbf{T}}^{(0)}$;
- Outer Loop: for $n = 1, 2, ...$
  - Inner Loop: for all processing patches
    - Generate data array D based on $\hat{\mathbf{T}}^{(n-1)}$;
    - Forward transform, coring, and inverse transform.
    - Unpack the recovered data array $\hat{D}$; Put the patches in it to their original locations;
  - Combine multiple-hypotheis inference by weighting averaging to get $x^{(n)}$
- Update the estimation of $\hat{\mathbf{T}}^{(n)}$: update kNN array based on $x^{(n)}$

patch set. Then, we present a Bayesian fusion approach to fuse multiple-hypothesis inferences by updating the kNN array $\mathbf{T}$ and the wanted signal x iteratively.

We note that the sparisty-based prior in our approach has two silent features. First, it transforms patch-based video representations into a higher-dimensional space to eliminate the need of basis pursuit. Second, the kNN array $\mathbf{T}$ is dynamically varying according to the signal, which make possible the iterative inference algorithm.

This nonparametric modeling approach is another implicit video modeling technique, in which the motion information is exploited by the space-time patch definition and the patch clustering. So it enjoys many advantages of the implicity, especially in handling non-ideal observation problems, like incomplete or noisy data. Moreover, it is a data-driven model, in which the model structure can implicitly and adaptively fit the input data. This avoids the mismatch between data and model, which is an inherent issue for parametric modeling approaches. In next chapter, we will apply this model into some restoration applications to show its good subjective and objective performances.

Chapter 5

Applications of Patch-based Sparsity model

5.1 Introduction

In this chapter, we apply the nonlocal patch-based sparsity model to several low level vision problems including error concealment/inpainting, denoising, and deartifacting. Many works in these applications have been done based on implicit nonparametric models, in which motion continuity is often maintained by spatio-temporal patch definition and space-time sampling instead of explicit motion estimation. For example, Wexier *et.al.*[50] presented a video completion method extending to space-time the pioneering image synthesis work in [79]. In their work, they filled in the missing regions by sampling spatio-temporal patches from other portions of the input video based on local structures and formulated the completion problem as a global optimization problem with a well-defined objective function. Cheung *et.al.*[27] exploited the "*epitome*" models that learns a mapping from the input video into a smaller volume-*epitome* which are then used for various tasks including video inpainting and video super resolution.

In this chapter, we will show our patch-based sparsity constraint provides a unified prior for various restoration tasks. The main difference lies in the derivation of $E[x|D]$. Even if our emphasis here is the versatility rather than optimization of patch-based processing algorithms for a specific task, we can see they have achieved highly competitive performance in these applications, which strongly supports the effectiveness of patch-based video models.

The rest of this chapter is organized as follows: in section 5.2, we focus on error concealment task. We first introduce the algorithm details, and then compare the performance to STALL. In 5.3, we apply this model to video denoising problem. We study on two noise type: impulse noise and gaussian noise, and compare our results with STALL and video epitome model. In 5.4, we introduce how to use our model to removing compression artifacts. In section 5.5, we summarize our model based on the applications.

5.2 Video Error Concealment

In this section, we are trying to attack video error concealment (i.e., video inpainting) by our patch-based sparsity model that proposed in previous chapter. The main algorithm structure is introduced in section 4.5. The compoment that need to be addressed here is the procedure to estimate $E[x|D]$, i.e., to get the estimation $\hat{D}$. In this application, we enforce the sparsity constraint by a recursive thresholding strategy similar to the one used by overcomplete DCT-based concealment of still images in [73]. Specifically, we can successively approximate $E[x|D]$ by hard thresholding transform coefficients D_T with a decreasing model of threshold τ.

The threshold τ is chosen based on two observations. First, the closer the patches are in the patch space, the sparser the data array D including the patches shows. So in this case, we will use small threshold, since there is small probablity to include

corrupted or missing pixels. We use the variance along the fourth dimension[1] to evaluate the similarity between patches in a data array. Denote the average variance along the fourth dimension of D by V_p,

$$V_p = mean(var(D, 4)), \tag{5.1}$$

where $var(D, 4)$ means to calculate the 4-th dimensional variance of each pixel location in a patch; $mean(.)$ is to the averaging operation of a 3D array. Second, with increasing the iteration times, the inpainting result will be better and better. So the threshold should become smaller and smaller to prevent oversmoothing. Thus, we present the following model for the hard thresholding parameter τ.

$$\tau(iter, V_p) = c\frac{\sqrt{V_p}}{iter}, \tag{5.2}$$

where $iter$ means the iteration times; c is a constant which is empirically fixed to 0.1.

We denote the inpainting domain by Ω_I, in which the data are missed. We simply replace all pixels in Ω_I^c by their original values, since undamaged pixels could be modified after enforcing the sparsity constraint. The complete video inpainting algorithm is summerized in Algorithm 2

In our experiments, the patch size is $11 \times 11 \times 3$, similar patch searching region centering with the current patch with size $51 \times 51 \times 7$. The maximum number(k) of patches in the similar patch set is 30. τ_{match} in Eq.(4.2) is set to 5. We tested *coastguard*, *mobile*, *tennis*, and *carphone* sequences, which includes deformable motion + panning, structure + zoom, texture + zoom, and jittering respectively. Except for the *tennis* sequence in which we use $11 - 40$-th frames, we use $1 - 30$-th frames. It

[1]The fourth dimension is the dimension s, along which we pack the similar patches. If the patch is 2D block, it should be the third dimension.

Algorithm 2 Video Inpainting by Nonlocal Patch-based Video Modeling

- Initialization: choose an appropriate $\hat{x}^{(0)}$ and perform kNN search on inpainting domain Ω_I for patches to obtain $\hat{\mathbf{T}}^{(0)}$;
- Outer Loop: for $n = 1, 2, ...$
  - Middle Loop: for $iter = 1, 2, ...$
    - Inner Loop: for all processing patches
      - Generate data array $D^{(iter)}$ based on $\hat{\mathbf{T}}^{(n-1)}$;
      - Estimate V_p by Eq.(5.1) and calculate τ by Eq.(5.2)
      - Forward transform
      - Thresholding $D_T^{(iter)}$ by τ
      - inverse transform.
      - Unpack the recovered data array $\hat{D}$; Put the patches in it to their original locations;
    - Combine multiple-hypotheis inference by weighting averaging to get $\hat{x}^{(n)}$
    - Replace the pixels in Ω_I^c by original pixels
- Update the estimation of $\hat{\mathbf{T}}^{(n)}$: update kNN array based on $\hat{x}^{(n)}$

assumes the block loss with size $8 \times 8 \times 3$ at the same spatial location occurs for three consecutive frames in the middle of the 30 frames sequence. Although we can start with any initial estimate of the missing block, we still use the STALL model result (without resampling and fusion) as the initial estimate to reduce the iterations in the algorithm.

Table 5.1 shows the PSNR performance comparison between this nonparametric model with STALL model without resampling and fusion. In average, this nonparametric model improves the PSNR by over 2dB. Fig.5.1 and 5.2 shows the perceptual comparisons of two sequences. Note that even in the *carphone* sequence that includes jittering which is very challenge for STALL model without fusion and resampling, our nonparametric model can provide good estimates.

Compared to other video error concealment algorithm, ours has the following ad-

Sequence	PSNR(dB)	
	STALL	This Model
coastguard	25.18	29.00
mobile	21.21	22.50
tennis	24.45	29.62
carphone	29.24	32.67

Table 5.1: PSNR(dB) performance comparison between STALL and Patch-based sparsity model

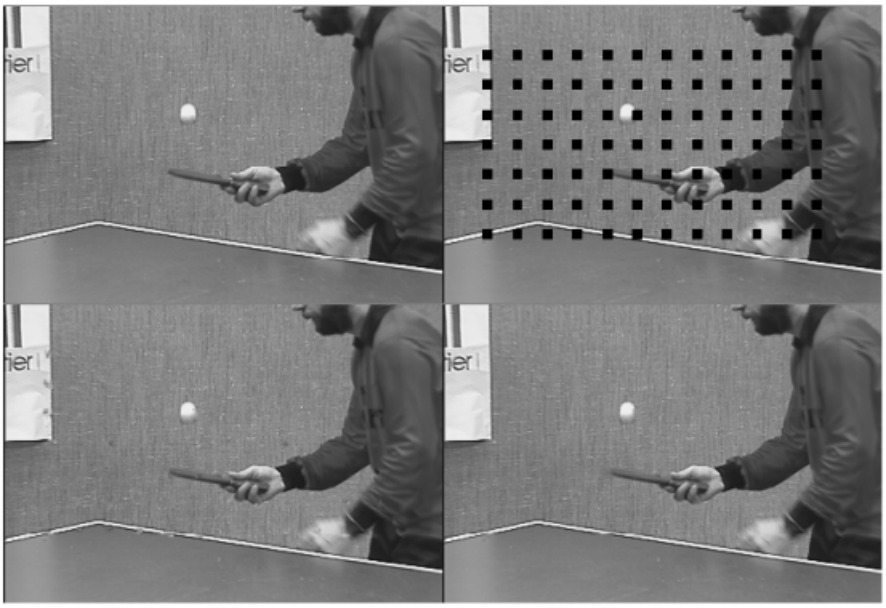


Figure 5.1: Comparison of error concealment results(*tennis_sif*). Top-left: original frame; top-right: damaged frame(note that the previous and next frames are also damaged at the same locations); bottom-left: concealed frame by STALL without resampling and fusion schemes; bottom-right: concealed frame by nonparametric model

vantages: first, we exploit the motion information in an implicit way, which avoid the suboptimum brought by the motion representation and estimation. The motion information is implicitly represented by the kNN array which can be adaptively refined. Second, we recover the missing data in a parallel way. Most of the video error concealment techniques recover the missing data in a sequential way, which cause the notorious error propagation problem once the missing block or area become large[54] [80] [47]. By contrast, our technique has potential to get away from this issue and

Figure 5.2: Comparison of error concealment results (*carphone_cif*). Top-left: original frame; top-right: damaged frame(note that the previous and next frames are also damaged at the same locations); bottom-left: concealed frame by STALL without resampling and fusion schemes; bottom-right: concealed frame by nonparametric model

can produce artifact-free results.

5.3 Video Denoising

In this section, we apply the patch-based sparsity model to video denoising problem. We consider two kinds of noise: impulse noise and white *Gaussian* noise, since they are typical noises in denoising problems.

5.3.1 Impulse Noise Removal

In this problem, we assume a video is corrupted by $[0, 255]$ uniform distributed impulse. The observation model is as follows,

$$Y = \begin{cases} X & w.p. \quad p & X : clean\ pixel \\ W & w.p. \quad (1-p) & W : impulse\ noise\ U[0,255] \end{cases}$$

where p shows the noise level(i.e., the percentage of noisy pixels). Similar with the STALL application in impulse noise removal, we also assume the availability of noisy detection result, i.e. the correct locations of noisy pixels. So the problem changes to the error concealment problem with the concealment domain Ω_I uniformly distributed over the video sequence (i.e., the noisy pixel locations).

So we use the same algorithm and parameter settings for inpainting in section 5.2. We tested four QCIF (176×144) sequences, which are *coastguard*, *garden*, *foreman*, and *tennis*. In impulse noise removal case, the noise level is $10\%, 20\%, 30\%, 50\%$, In this application, we choose to use the output of median filter as the initial estimate.

Fig.5.3 shows the PSNR evolution of the filtered sequences during the processing. It verifies that our algorithm do converge. We can see the staircase corners marked with arrow is a new round of the outer loop in Algorithm 2 (i.e., at this point, $\mathbf{T}$ will be updated by latest $\hat{x}$); From Fig.5.3, we can see the iterative algorithm does work to improve the restoration performance.

Table 3.3 reports the PSNR of the impulse noise removal results on the four video sequences. Fig.5.4 and 5.5 show the perceptual results by median filter and our nonparametric model. We can see our model outperforms the median filter in objective and subjective quality.

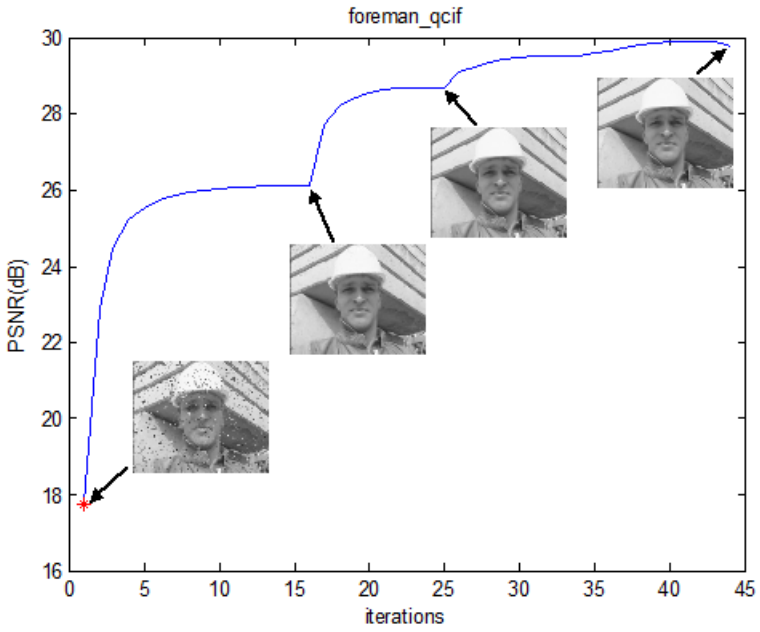


Figure 5.3: PSNR evolution curve in impulse noise removal application,*foreman_qcif*, 50% impulse noise uniformly distributed in $[0, 255]$

sequences	Impulse Noise Level							
	10%		20%		30%		50%	
	Median	Ours	Median	Ours	Median	Ours	Median	Ours
coastguard	26.12	34.80	24.59	32.46	23.03	30.86	19.37	29.14
garden	26.80	28.00	24.90	26.78	22.02	25.23	17.28	23.68
foreman	19.92	34.88	19.39	32.43	18.45	31.80	16.22	29.30
tennis	25.76	36.57	24.84	34.19	23.79	33.93	20.78	32.92

Table 5.2: PSNR(dB) performance of Patch-based sparsity model in impulse noise removal

5.3.2 Guassian Noise Removal

We assume the original sequence is corrupted by an additive white Gaussian noise with standard deviation σ_W. So we have the following observation model:

$$y = x + W, \tag{5.3}$$

where x and y are the original and noisy pixels respectively; W is a white *Gaussian* noise with standard deviation σ_W.

Figure 5.4: Comparison of impulse noise removal results (*foreman_qcif*). Top-left: original frame; top-right: 50% impulse noise corrupted frame; bottom-left: restored frame by median filter with window 3×3; bottom-right: restored frame by our nonparametric model

The main algorithm structure was introduced in section 4.5. In this problem, we will introduce how to estimate $E[x|D]$. There are two popular coring strategies for enforcing sparsity constraint to obtain $E[x|D]$: thresholding and filtering. In the former case, $E[x|D]$ is simply obtained by hard thresholding the transform coefficients D_T[77]; in the latter case, MMSE estimation is given by the classical *Wiener* filtering formula[75] under the assumption that noise variance is given.

Since *Wiener* filtering requires the knowledge about the variance of clean signal. In our work, we propose to iteratively estimate the signal variance in the following fashion. The iterative *Wiener* filtering operates in transform domain by

$$D_T^{(iter+1)}(r,c,t) = \frac{(\sigma_x^{(iter)}(r,c,t))^2}{(\sigma_x^{(iter)}(r,c,t))^2 + \sigma_w^2} D_T^0(r,c,t), \tag{5.4}$$

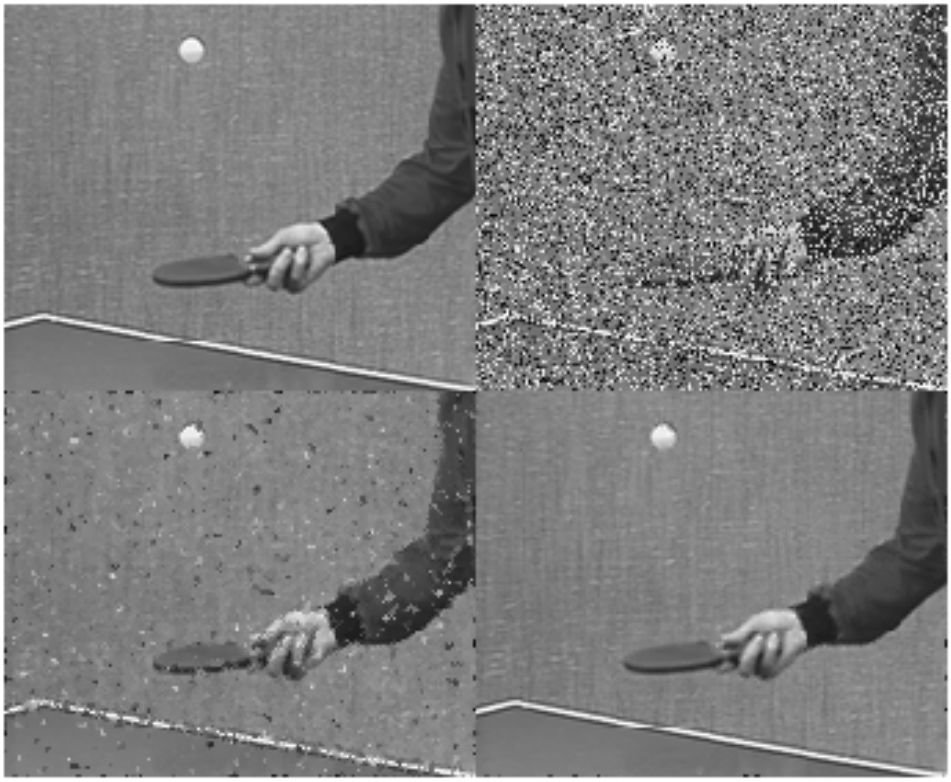


Figure 5.5: Comparison of impulse noise removal results (*tennis_qcif*). Top-left: original frame; top-right: 50% impulse noise corrupted frame; bottom-left: restored frame by median filter with window 3×3; bottom-right: restored frame by our nonparametric model

where $\sigma_x^{(iter)}(r,c,t)^2 = max[0, (D_T^{(iter)})^2 - \sigma_w^2]$ is the maximum-likelihood estimation of signal variance. We summerize the denoising algorithm in Algorithm 3

In this application, we use patch size $11 \times 11 \times 3$, searching region $51 \times 51 \times 7$, the matching threshold $\tau_{match} = 5$. We tested four QCIF sequences, which are *coastguard*, *garden*, *foreman*, and *tennis*. The standard deviation $\sigma_W = 25$. In this application, we choose to use the noisy sequence as the initial estimate.

sequences	σ_w	epitome[27]	Ours
coastguard	25	25.7212	30.4867
garden	25	19.3833	25.6774
foreman	25	25.4452	32.6508
tennis	25	27.5898	31.6768

Table 5.3: PSNR(dB) performance of Patch-based sparsity model in denoising application

Algorithm 3 Video Denoising by Nonlocal Patch-based Video Modeling

- Initialization: set $\hat{x}^{(0)} = y$ and perform kNN search to obtain $\hat{\mathbf{T}}^{(0)}$;
- Outer Loop: for $n = 1, 2, ...$
  - Middle Loop: for $iter = 1, 2, ...$
    - Inner Loop: for all processing patches
      - Generate data array $D^{(iter)}$ based on $\hat{\mathbf{T}}^{(n-1)}$;
      - Forward transform
      - Estimate $(\sigma_x^{(iter)}(r, c, t))^2 = max[0, (D_T^{(iter)})^2 - \sigma_w^2]$
      - Filtering by Eq.(5.4)
      - Inverse transform
      - Unpack the recovered data array $\hat{D}^{(iter)}$; Put the patches in it to their original locations
    - Combine multiple-hypotheis inferences by weighting averaging to get $\hat{x}^{(n)}$
- Update the estimation of $\hat{\mathbf{T}}^{(n)}$: update kNN array based on $\hat{x}^{(n)}$

Fig.5.6 and 5.7 show the restoration results by video epitome and our nonparametric model. It is obvious that our result outperforms that of video epitome with sharp edges in space and well-maintained motion continuity in time.

5.4 DeArtifacting

Many video compression strategies employ block-based transforms(e.g., DCTs) and motion compensation among their compression tools. Coarse quantization of transform coefficients, and the use of different reference locations or different reference pictures by neighboring blocks in motion compensated prediction can give rise to visual artifacts such as distortion around edges, textures or block discontinuities. Fig.5.4 shows some video frames which are compressed with H.264/AVC enocoder without deblocking filter. We can easily find the artifacts around edges, textures, and the high motion areas.

Figure 5.6: Comparison of Gaussian noise removal results (*coastguard_qcif*). Top-left: original frame; top-right: noisy frame corrupted by additive Gaussian noise with $\sigma_W = 25$; bottom-left: restored frame by epitome; bottom-right: restored frame by our nonparametric model

In the H.264/AVC video compression standard[81], an in-loop deblocking filter[82] has been adopted to attenuate artifacts arising along block boundaries. Specifically, it applys low pass filtering at both vertical and horizontal block boundaries. The filtering strength and support length is adaptive to the encoding mode(e.g.,intra or inter coded block), the boundary locations(e.g., macroblock boundary or $4x4$ block boundary), and the source of predictions(i.e., whether the predictions come from the same reference frame or not). It really improves both subjective and objective video quality by removing blocky artifacts. However, blocky artifacts are not the only ones present in compressed video. Coarse quantization is also responsible for other artifacts such as ringing, edge distortion or texture corruption. Since the low pass filtering operation assumes a smooth image model, it is not suited for removing

Figure 5.7: Comparison of Gaussian noise removal results (*garden_qcif*). Top-left: original frame; top-right: noisy frame corrupted by additive Gaussian noise with $\sigma_W = 25$; bottom-left: restored frame by epitome; bottom-right: restored frame by our nonparametric model

image singularities such as edges and textures. Recently some advances have achieved in removing the compressed aritifacts by incorporating in-loop or out-loop filters. Sparsity-based non-linear in-loop filters have been proposed in the literatures[83] and [84]. In [85], *Wiener* filter-based post filters which are trained at encoder based on local spatial variances are sent to decoder as overhead to facilitate the deartifacting.

Although all the above approaches improve both subjective and objective video quality, all evaluation measures are based on single frame quality (i.e., PSNR or visual quality of a single frame). The motion consistency has not been addressed in these works, since the filtering operations are performed within a single frame. In this section, we apply our patch-based sparsity model to deartifacting problem. The generalization of patch definition to 3D and the space-time patch clustering provide us

Figure 5.8: Examples of video coding artifacts: Top-Original frame, Bottom-Compressed frame without deblocking or deartfacting.

an implicit way to handle the motion consistency. We design our approach as a post-processing filter, which takes the deblocked video as input. We will use the slightly modified algorithm with Algorithm 3 in section 5.3.2 for deartifacting. In *Gaussian* denoising algorithm, we assume the noise power σ_w is given. In this deartifacting problem, we don't know the exact noise power. However, since we have original video at encoder, we can estimate σ_w by using the blocky and original videos and send the estimate to decoder as an overhead. Comparing to the work in [85] which needs to send a set of filter coefficients as overhead, our work cut the amount of overhead to one parameter σ_w, which is more preferred especially in low bit rate applications. We summerize the algorithm in Algorithm 4

In this application, we use patch size $11 \times 11 \times 3$, searching region $41 \times 41 \times 3$,

Algorithm 4 Video DeArtifacting by Nonlocal Patch-based Video Modeling

- Get σ_w: Estimate σ_w if at encoder; Parse σ_w from bitstream if at decoder;
- Initialization: set $\hat{x}^{(0)} = y$ and perform kNN search to obtain $\hat{\mathbf{T}}^{(0)}$;
- Outer Loop: for $n = 1, 2, ...$
  - Middle Loop: for $iter = 1, 2, ...$
    - Inner Loop: for all processing patches
      - Generate data array $D^{(iter)}$ based on $\hat{\mathbf{T}}^{(n-1)}$;
      - Forward transform
      - Estimate $(\sigma_x^{(iter)}(r, c, t))^2 = max[0, (D_T^{(iter)})^2 - \sigma_w^2]$
      - Filtering by Eq.(5.4)
      - Inverse transform
      - Unpack the recovered data array $\hat{D}$; Put the patches in it to their original locations
    - Combine multiple-hypotheis inference by weighting averaging to get $\hat{x}^{(n)}$
- Update the estimation of $\hat{\mathbf{T}}^{(n)}$: update kNN array based on $\hat{x}^{(n)}$

the matching threshold $\tau_{match} = 5$, maximum number of patches in a similar patch set is set 30. Table 5.4 shows the simulation results of the patch-based sparsity model in deartifacting. All the testing sequences are compressed as *intra* pictures by H.264/AVC reference software JSVM[86] with quantization parameter $QP = 37$ (i.e., in low bit rate range), which corresponds to the most coarse quantization step defined in VCEG standardized test range for *intra* pictures. We can observe from Table 5.4, even in such large quantization step case, our approach can acheive up to $0.7dB$ PSNR gain. Fig.5.9 to Fig.5.11 show the visual quality comparisons between our approach and H.264/AVC adaptive deblocking filter. Our approach can also improve the subjective quality obviously.

Figure 5.9: Visual quality comparison between H.264/AVC deblocking and patch-based sparsity model - *foreman-qcif*: Original frame(top-left), Artifacting frame(top-right),H.264/AVC deblocked frame(bottom-left),patch-based sparsity model filtered frame(bottom-right)

5.5 Conclusion

In this chapter, we apply our nonlocal patch-based sparsity model to three video processing tasks: error concealment(/inpainting), denoising, and deartifacting. All these applications include either incomplete or corrupted information, which is challenging for the models with explicit motion respresentations. In our model, motion is not explicitly represented by motion vectors but implicitly embedded into the relationships among patches, which represents motion in a distributed fashion.

We observed the effectiveness of this model by comparing with parametric and other nonparametric models. More importantly, we have shown in this chapter that

Figure 5.10: Visual quality comparison between H.264/AVC deblocking and patch-based sparsity model - *silent-qcif*: Original frame(top-left), Artifacting frame(top-right),H.264/AVC deblocked frame(bottom-left),patch-based sparsity model filtered frame(bottom-right)

our patch-based sparsity constraint provides us a unified prior for various restoration tasks. For all these applications, they share the similar model structure shown in section 4.5. The application dependent part lies in the derivation of $E[x|D]$. We designed different approaches to estimate $E[x|D]$ based on the applications. Specifically, in error concealment, we use hard thresholding to enforce the sparsity constraint. The threshold model is designed as a function of the patch-dimension variance and iteration times. In denoising, we adopt *Wiener* filtering as the coring strategy and present an iterative method to estimate signal variance. The similar algorithm for denoising can be easily revised for deartifacting by estimating the noise variance. Although our

Figure 5.11: Visual quality comparison between H.264/AVC deblocking and patch-based sparsity model - *paris-cif*: Original frame(top-left), Artifacting frame(top-right),H.264/AVC deblocked frame(bottom-left),patch-based sparsity model filtered frame(bottom-right)

emphasis in this chapter is the versatility rather than optimization of patch-based processing algorithm for a specific task, we can see our model still achieved highly competitive preformance in all these applciations.

Table 5.4: PSNR improvement by patch-based sparsity model over H.264/AVC in deartifacting

Sequences	Resolution	QP	H.264/AVC Deblocking Filter	Our Filter	PSNR Improvement
container	176×144	22	41.8	42.5	0.7
		27	38.0	38.7	0.7
		32	34.5	35.1	0.6
		37	31.0	31.7	0.7
foreman	176×144	22	41.7	42.2	0.5
		27	38.0	38.6	0.6
		32	34.6	35.3	0.7
		37	31.5	32.2	0.7
silent	176×144	22	41.3	42.5	1.2
		27	37.2	38.2	1
		32	33.5	34.3	0.8
		37	30.6	31.2	0.6
foreman	352×288	22	42.0	42.3	0.3
		27	38.6	38.9	0.3
		32	35.5	35.9	0.4
		37	32.7	33.3	0.6
mobile	352×288	22	41.1	41.3	0.2
		27	36.5	36.8	0.3
		32	32.1	32.4	0.3
		37	28.1	28.5	0.4
paris	352×288	22	41.6	42.3	0.7
		27	37.5	38.1	0.6
		32	33.5	34.1	0.6
		37	30.0	30.5	0.5
tempete	352×288	22	41.2	41.6	0.4
		27	36.9	37.4	0.5
		32	32.7	33.3	0.6
		37	29.1	29.7	0.6
night	1280×720	22	42.7	43.0	0.3
		27	39.1	39.5	0.4
		32	35.6	36.1	0.5
		37	31.7	32.2	0.5
crew	1280×720	22	43.3	43.4	0.1
		27	40.4	40.5	0.1
		32	38.0	38.1	0.1
		37	35.2	35.4	0.2

Chapter 6

Conclusions and Perspectives

The existence of motion in video source gives a video potentials to include more information than a still image. The high sensitivity of human eyes to motion drives video enter more and more fields in our life today. Although we already have systematic knowledge about motion in physical domain, we have little understanding about it in the digitized world. So how to exploit motion information to facilitate the tasks in the increasing video applications is still an open problem in video related research. The goal of this thesis is to study on mathematical models to represent and exploit motion information for practical applications.

6.1 Conclusions

In this thesis, we first analyzed the limitations of the popular motion representations and propose to represent motion from another perspective. We pointed out in Chapter 1 that video modeling by explicit motion representations is not always suitable for many applications because of the suboptimum of motion estimation and the non-ideal observation data. In this thesis, we advocate the implicit motion representations by proposing two video approaches in which motion information is embeded into the

relationships among either pixels or patches. Such distributed motion representations are likely to more closely match the mechanism employed by human vision[3].

The first model is a local parametric model, which achieves adaptation by a learning within a spatio-temporal training window. Such localized learning helps to achieve the balance between space and time, which is critical to exploit the motion information. The second model is a patch-based approach. It exploits a generic sparsity-based prior by a kNN based patch clustering and a dimensionality increasing strategy. Such approaches adapt signal representations to fixed bases instead of basis pursuit approaches. In both models, multiple hypotheses fusion strategies are developed to resolve the model uncertainties, which are proved to be useful for more general video signals. The experimental results in low level vision tasks have clearly indicated the versatilities, flexibilities, and competence of these implicit motion models.

6.2 Future Work on the Proposed Models

Although the favorable flavors of such implicit models have been tasted, many future works is possibly required in order to make the proposed models popular in practical applications:

- The performance of STALL model is largely dependent on the fidelity of training data set. As we have known, the localized learning is sensitive to the outlier and noise. First, we need to avoid outliers involving into the training. It is generally not a simple problem due to the complexity of video signals. Second, for noisy training data, we can consider to incorporate total least square (TLS) techniques to take the noise into acount.

- In STALL model, the relationship between the model order and the topology of training data set is critical but not investigated in detail. With increasing the model's order, we hope it will be more accurate to describe the motion.

However, to make the problem stable, we need to include more training data, which may require to extend the training data set in either space or time. As we have shown in chapter 2.3.2, extension of training data set can drop the performance because of the nonstationarity of video signals. So to find the optimal model order and the optimal traning set are also a possible research orientation in both theory and practice.

- In the patch-based sparsity model, 2D blocks are generalized into 3D patches. Although we have many intuitions and some reseasonable simulation results to show the benefits brought by 3D patches, the theoretic evidence and explaination are not addressed. It introduces the question, what is the best patch topology given a video sequence? In our work, we just use rectangular patches. But it is just a complexity oriented not a performance oriented choice. So this problem could be another possible research orientation in the future.

- Some critical parameters in the patch-based sparsity model should be investigated deeper. For example, the kNN patch clustering parameter k, which says how diverse the motion is represented. In our work, we fixed it empirically, which is far from optimal. Since the sparsity of data array D is determined by the simiarity of all patches in the similar patch set, it is desirable to adaptively choose the value of k.

- The complexity of the patch-based sparsity model is prohibitive for industrial applications. The high complexity of this model mainly comes from the kNN-based 3D patch clustering and the forward and backward transform. In our works, we simplify the transform to *Fourier* transform and adopt FFT in our implementations. But for kNN search, fast algorithms can be exploited to acheive better trade-off between performance and complexity.

6.3 Perspectives on Implicit Motion Representations

As we said at Chapter 1, we advocate the implicit motion representations because of their flexibilities and effeciencies to exploit motion information in digital video source. We proposed two models and applied them into several low-level vision problems, such as denoising, inpainting, and interpolation. However, the applications of implicit motion representations are much wider than these. Actually we believe many problems ranging from low to high level vision problems can benefit from such implicit motion representations.

In low-level vision problems, some implicit motion representations have been applied to video compression[87] and [88]. Due to the implicit characteristics, we have no motion vectors that have to be transmitted to decoder as overhead under the explicit motion representation based framework. Motion information is exploited through implicitly utlizing the correlations between neighoring pixels over space and time. To extend the implicit motion representations to video compression, the most challenge is to build a suitable compression framework for them, since most current compression frameworks (like H.264 and MPEG4) are designed based on explicit motion respresentations. Similiarly, video texture synthesis can also benefit from implicit motion representations as we use them into video inpainting problems.

In middle-level vision problems, we have seen the *object tunnel* concept was applied in object segmentation and occlusion detection. Actually our STALL model can also have potentials to be applied into motion segmentation if we can cluster the pixels based on analysis of model parameters' characteristics. Although we do not include our very preliminary results in this field, they still show the potentials of our STALL models.

In the other end, high-level vision problems, the implicit motion representations

have also shown its possiblilities. For example, geometric pattern of spato-temporal slice (STS) is analyzed for detecting scene or shot change in [89]. Their work shows again the explicit motion estimation can also be avoided once the final output is not motion vector or field. But this time, the evidence is from high level vision problem. Their work is dependent on the observations: the discontinuity of color and texture represents the occurence of a new event; the orientation of texture depicts camera or object motions. No motion estimation is involved, but motion information is implicitly exploited by texture analysis in STS.

Another possible perspective of the implicit motion representations is in multi-view video modeling[90]. Multi-view video is often simultaneously captured by multiple video cameras in different viewpoints of the scene. Multi-view video can be applied to create realistic 3D depth impression of the observed scene[91]. In this application, one more dimension - view dimension, is introduced. We can also regard the relationship between different views is a kind of motion that caused by the difference of imaging device's position and pose. Like the motion along temporal axis, we argue the motion along the view axis can also benefit from implicit representations, i.e., the redundancy between different views can also be exploited by impolicit motion representations.

Bibliography

[1] E. H. Adelson, J. Y. A. Wang, and S. A. Niyogi, "Mid-level vision: new directions in vision and video," in *Proceedings of IEEE International Conference on Image Processing*, vol. 2, (Austin, TX), pp. 21–25, Nov. 1994.

[2] A. M. Tekalp, *Digital Video Processing*. Prentice Hall PTR, 1 ed., August 1995.

[3] E. Simoncelli, *Distributed Representation and Analysis of Visual Motion*. Ph.D. thesis, MIT, Jan 1993.

[4] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of 7th Intl Joint Conf on Artificial Intelligence*, pp. 674–679, 1981.

[5] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, pp. 221–225, Feb. 2004.

[6] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, no. 17, pp. 185–203, 1981.

[7] F. Dufaux and J. Knorad, "Efficient, robust and fast global motion estimation for video coding," *IEEE Transactions on Image Processing*, vol. 9, pp. 497–501, March 2000.

[8] R. Szeliski, "Video mosaics for virtual environments," *IEEE Computer Graphics and Applications*, vol. 16, pp. 22–30, March 1996.

[9] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[10] J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, no. 21-23, pp. 593–600, 1994.

[11] R. Li, B. Zeng, and M. L. Liou, "A new three-step search algorithm for block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 438–442, August 1994.

[12] L.-M. Po and W.-C. Ma, "A novel four-step search algorithm for fast block motion estimation," *IEEE Transactions on Circuits and Systems for Video Technology*, pp. 313–317, Jun. 1996.

[13] B. Girod, "Motion-compensation prediction with fractiona-pel accuracy," *IEEE Transactions on Communications*, pp. 604–612, 1993.

[14] M. T. Orchard and G. J. Sullivan, "Overlapped block motion compensation: An estimation-theoretic approach," *IEEE Transactions on Image Processing*, pp. 693–699, 1994.

[15] J. Boyce, "Weighted prediction in the h.264/mpeg4 avc video coding standard," in *ISCAS*, IEEE, May 2004.

[16] D. E. O., P. Yin, C. Dai, and X. Li, "Geometry-adaptive block patitioning for video coding," in *Acoustics, Speech and Signal Processing, 2007, ICASSP 2007*, vol. 1, April 2007.

[17] I. E. Richardson, *H.264 and MPEG-4 Video Compression: Video Coding for Next-Generation Multimedia*. John Wiley and Sons, Ltd., 2003.

[18] P. Sand and S. Teller, "Particle video:long-range motion estimation using point trajectories," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2, pp. 2195–2202, 2006 2006.

[19] M. Bertalmio, A. L. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. I–355–I–362, 2001.

[20] Y. Wang and Q. Zhu, "Error control and concealment for video communication: a review," in *Proceedings of the IEEE*, vol. 86, pp. 974–997, May 1998.

[21] T. Chen and H. Wu, "Adaptive impulse detection using center-weighted median filters," *IEEE Signal Processing Letters*, pp. 1–3, January 2001.

[22] S. C. Park, M. K. Park, and M. G. Kang, "Super-resolution image reconstruction: a technical overview," *Signal Processing Magazine, IEEE*, pp. 21–36, May 2003.

[23] A. Buades, B. Coll, and J.-M. Morel, "Nonlocal image and movie denoising," *International Journal of Computer Vision*, vol. 76, no. 2, pp. 123–139, 2008.

[24] M. Szummer and R. W. Picard, "Temporal texture modeling," in *Proceedings of International Conference on Image Processing(ICIP)*, pp. 823–826, 1996.

[25] G. Doretto, A. Chiuso, Y. Wu, and S. Soatto, "Dynamic textures," *International Journal of Computer Vision (IJCV)*, no. 2, pp. 91–109, 2003.

[26] M. Ristivojevic and J. Konrad, "Space-time image sequence analysis: object tunnels and occlusion volumes," *IEEE Transactions on Image Processing*, vol. 15, pp. 364–376, Feb 2006.

[27] V. Cheung, B. J. Frey, and N. Jojic, "Video epitomes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 42–49, 2005.

[28] A. Buades, B. Coll, and J. M. Morel, "On image denoising methods," *SIAM Multiscale Modeling and Simulation*, 2005.

[29] M. Mahmoudi and G. Sapiro, "Fast image and video denoising via nonlocal means of similar neighorhoods," *IEEE Signal Processing Letters*, pp. 839–842, 2005.

[30] J. Boulanger, C. Kervrann, and P. Bouthemy, "Space-time adaptation for patch-based image sequence restoration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1096–1102, 2007.

[31] M. Protter and M. Elad, "Sparse and redundant representations and motion-estimation-free algorithm for video denoising," in *SPIE (Wavelet XII)*, (San Diego CA), August 2007.

[32] M. Elad and M. Aharon, "Image denoising via learned dictionaries and sparse representation," in *Proc. of the International Conference on Computer Vision and Pattern Recognition(CVPR)*, June 2006.

[33] K. Dabov, A. Foi, and K. Egiazarian, "Video denoising by sparse 3d transform-domain collaborative filtering," in *in Proceedings of 15th European Signal Processing Conference*, 2007.

[34] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising with block-matching and 3d filtering," in *in Electronic Imaging'06, Proc. SPIE 6064*, no. 6064A-30, 2006.

[35] O. G. Guleryuz, "Nonlinear approximation based image recovery using adaptive sparse reconstructions," in *IEEE International Conferenece on Image Processing*, vol. 1, 2003.

[36] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentile Hall, 2002.

[37] G. Aubert and P. Kornprobst, "Mathematical problems in image processing: Partial differential equations and the calculus of variations," *Applied Mathematical Sciences*, 2002.

[38] J. Woods and C. Radewan, "Kalman filtering in two dimensions," *IEEE Transactions on Information Theory*, pp. 473–482, 1977.

[39] T. Wedi and H. G. Musmann, "Motion- and aliasing-compensated prediction for hybrid video coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 577–586, July 2003.

[40] T. Wedi, "Adaptive interpolation filter for motion compensated prediction," in *Proceedings, 2002 International Conference on Image Processing*, vol. 2, pp. II–509–II–512, 2002.

[41] J. Wang and E. Adelson, "Layered representation for motion analysis," in *Computer Vision and Pattern Recognition, Proceedings CVPR'93*, (New York, NY, USA), pp. 361–366, Jun 1993.

[42] J. Wang and E. Adelson, "Representing moving images with layers," *IEEE Transactions on Image Processing*, pp. 625–638, 1994.

[43] B. Efron, "The jackknife, the bootstrap and other resampling plans," in *CBMS-NSF Regional Conference Series in Applied Mathematics,Philadelphia: Society for Industrial and Applied Mathematics(SIAM)*, 1982.

[44] D. Taubman and A. Zakor, "Multi-rate 3d subband coding of video," *IEEE Transactions on Image Processing*, pp. 572–588, 1994.

[45] Y. Zheng and X. Li, "Video modeling via spatio-temporal adaptive localized learning (stall)," in *40th Asilomar Conference on Signals, Systems, and Computers*, pp. 979–983, Oct.-Nov. 2006.

[46] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *IEEE International Conference on Computer Vision*, pp. 839–846, 1998.

[47] Y. Zheng, X. Li, and C. Dai, "Video error concealment based on implicit motion models," in *Proceedings of the SPIE*, vol. 6015, pp. 60150F1–60150F10, 2005.

[48] S. Farsiu, M. Robinson, M. Elad, and P. Milanfar, "Fast and robust multiframe super resolution," *IEEE Transactions on Image Processing*, pp. 1327–1344, 2004.

[49] G. Pok, J.-C. Liu, and A. S. Nair, "Selective removal of impulse noise based on homogeneity level information," *IEEE Transactions on Image Processing*, vol. 12, pp. 85–92, January 2003.

[50] Y. Wexler, E. Shechtman, and M. Irani, "Space-time completion of video," *IEEE Transactions on Pattern Anal. Mach. Intell.*, no. 3, 2007.

[51] J. Zhang, J. Arnold, M. Frater, and M. Pickering, "Video error concealment using decoder motion vector estimation," in *TENCON '97, Proceedings of IEEE Speech and Image Technologies for Computing and Telecommunications*, vol. 2, pp. 770–780, December 1997.

[52] K. Meisinger and A. Kaup, "Spatiotemporal selective extrapolation for 3-d signals and its applications in video communications," *Transactions on Image Processing*, vol. 16, pp. 2348–2360, September 2007.

[53] S. Kumar, M. Biswas, S. Belongie, and T. Nguyen, "Spatio-temporal texture synthesis and image inpainting for video applications," in *IEEE International Conference on Image Processing*, vol. 2, pp. II–85–8, Sept. 2005.

[54] A. Criminisi, P. Perez, and K. Toyama, "Object removal by exemplar-based inpainting," in *Computer Vision and Pattern Recognition*, vol. 2, pp. 721–728, Jun.

[55] J. Sun, L. Yuan, J. Jia, and H. Shum, "Image completion with structure propagation," in *ACM SIGRAPH*, vol. 24, pp. 861–868, 2005.

[56] X. Li and M. T. Orchard, "Novel sequential error-concealment techniques using orientation adaptive interpolation," *IEEE Transactions on Circuits and Systems for Video Technology*, no. 10, pp. 857–864, 2002.

[57] Q. Tian and M. Huhns, "Algorithms for subpixel registration," *Computer Vision, Graphics, and Image Processing*, pp. 220–233, 1986.

[58] P. Thévenaz, U. Ruttimann, and M. Unser, "A pyramid approach to subpixel registration based on intensity," *IEEE Transactions on Image Processing*, pp. 27–41, 1998.

[59] H. Foroosh, J. Zerubia, and M. Berthod, "Extension of phase correlation to subpixel registration," *IEEE Transactions on Image Processing*, pp. 188–199, 2002.

[60] D. Capel and A. Zisserman, "Computer vision applied to super resolution," *IEEE Signal Processing Magazine*, pp. 75–86, May 2003.

[61] X. Li and M. T. Orchard, "New edge directed interpolation," *IEEE Transactions on Image Processing*, pp. 1521–1527, October 2001.

[62] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*. John Wiley and Sons, Inc., 2 ed., 2000.

[63] H. Takeda, S. Farsiu, and P. Milanfar, "Kernel regression for image processing and reconstruction," *IEEE Transaction on Image Processing*, pp. 349–366, 2007.

[64] Shakhanarovish, Darrell, and Indyk, *Nearest-Neighbor Methods in Learning and Vision*. The MIT Press, 2005.

[65] W. Freeman, T. Jones, and E. Pasztor, "Example-based super-resolution," *IEEE Computer Graphics and Applications*, pp. 56–65, 2002.

[66] X. Li and Y. Zheng, "Patch-based video processing: a variational bayesian approach," *IEEE Transaction on Circuits and System for Video Technology*, In Press.

[67] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, pp. 31–61, 1998.

[68] W. Freeman, E. Pasztor, and O. Carmichael, "Learning low-level vision," *International Journal of Computer Vision*, vol. 40, pp. 25–47, October 2000.

[69] S. Roweis and L. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *Science*, pp. 2323–2326, 2000.

[70] K. Hirakawa and T. Parks, "Image denoising using total least squares," *IEEE Transactions on Image Processing*, pp. 2730–2742, 2006.

[71] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Image denoising by sparse 3-d transform-domain collaborative filtering," *IEEE Transaction on Image Processing*, vol. 16, pp. 2080–2095, Aug 2007.

[72] O. G. Guleryuz, "Nonlinear approximation based on image recovery using adaptive sparse reconstructions and iterated denoising-part i: Theory," *IEEE Transactions on Image Processing*, no. 3, pp. 539–554, 2006.

[73] O. G. Guleryuz, "Nonlinear approximation based on image recovery using adaptive sparse reconstructions and iterated denoising-part ii - adaptive algorithms," *IEEE Transactions on Image Processing*, vol. 15, pp. 555–571, March 2006.

[74] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Transaction on Image Processing*, vol. 15, pp. 3736–3745, December 2006.

[75] E. P. Simoncelli and E. H. Adelson, "Noise removal via bayesian wavelet coring," in *Proc 3rd IEEE Int'l Conf on Image Proc*, vol. I, pp. 379–382, September 1996.

[76] X. Li and M. Orchard, "Spatially adaptive image denoising under overcomplete expansion," in *IEEE International Conference on Image Processing*, 2000.

[77] D. L. Donoho and I. M. Johnstone, "Ideal spatial adaptation by wavelet shrinkage," *Biometrika*, vol. 81, pp. 425–455, 1994.

[78] M. Beal, *Variational algorithms for approximate bayesian inference*. PhD thesis, University College London, Gatsby Computational Neuroscience Unit, 2003.

[79] A. Efros and T. Leung, "Texture synthesis by non-parametric sampling," in *ICCV*, pp. 1033–1038, 1999.

[80] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester, "Image inpainting," in *SIGGRAPH 2000*, pp. 417–424, 2000.

[81] ITU-T, *ITU-T Recommendation H.264*, Mar 2005.

[82] P. List, A. Joch, J. Lainema, G. Bjontegaard, and M. Karczewicz, "Adaptive deblocking filter," *IEEE Transaction on Circuits and System for Video Technology*, vol. 13, pp. 614–619, July 2003.

[83] O. G. Guleryuz, "A nonlinear loop filter for quantization noise removal in hybrid video compression," in *IEEE International Conferenece on Image Processing*, vol. 2, 2005.

[84] C. C. Dorea, O. D. Escoda, P. Yin, and C. Gomila, "A direction-adaptive in-loop deartifacting filter for video coding," in *IEEE International Conference on Image Processing*, October 2008.

[85] M. Karczewicz, W.-J. Chien, P. Chen, and Y. Ye, "Post-filter sei message extensions," tech. rep., Qualcomm Inc., Berlin, Germany, July 2008. VCEG-AI34.

[86] ISO/IEC JTC1/SC29/WG11 and ITU-T SG16 Q. 6, *JSVM 6 Software*, Apr 2006.

[87] X. Li, "Least-square based prediction for backward adaptive video coding," *EURASIP Journal on Applied Signal Processing*, vol. 2006, pp. 126 – 126, 2006.

[88] Y. Suzuki, C. S. Boon, and T. K. Tan, "Inter frame coding with template matching averaging," in *IEEE International Conference on Image Processing*, vol. III, (San Antonio, TX), pp. 409–412, Oct. 2007.

[89] C.-W. Ngo, T.-C. Pong, and H.-J. Zhang, "Motion-based video representation for scene change detection," *International Journal of Computer Vision*, vol. 50, pp. 127–142, November 2002.

[90] M. Flierl and B. Girod, "Multiview video compression," *IEEE Signal Processing Magazine*, pp. 66–76, Nov. 2007.

[91] M. Tanimoto, "Ftv (free viewpoint television) creating ray-based image engineering," in *Proceedings of the IEEE International Conference on Image Processing*, (Genova, Italy), Sept. 2005.