# Design of a High-Speed Architecture for Stabilization of Video Captured Under Non-Uniform Lighting Conditions

Ming Zhu Zhang
*Old Dominion University*

# DESIGN OF A HIGH-SPEED ARCHITECTURE FOR STABILIZATION OF VIDEO CAPTURED UNDER NON-UNIFORM LIGHTING CONDITIONS

by

Ming Zhu Zhang
B.S. December 2004, Old Dominion University
M.S. August 2005, Old Dominion University

A Dissertation Submitted to the Faculty of
Old Dominion University in Partial Fulfillment of the
Requirement for the Degree of

DOCTOR OF PHILOSOPHY

ELECTRICAL AND COMPUTER ENGINEERING

OLD DOMINION UNIVERSITY
December 2008

Approved by:

Dr. Vijayan K. Asari (Director)

Dr. James Leathrum, Jr. (Member)

Dr. Jiang Li (Member)

Dr. Shunichi Toida (Member)

## ABSTRACT

# DESIGN OF A HIGH-SPEED ARCHITECTURE FOR STABILIZATION OF VIDEO CAPTURED UNDER NON-UNIFORM LIGHTING CONDITIONS

Ming Zhu Zhang
Old Dominion University, 2008
Director: Dr. Vijayan K. Asari

Video captured in shaky conditions may lead to vibrations. A robust algorithm to immobilize the video by compensating for the vibrations from physical settings of the camera is presented in this dissertation. A very high performance hardware architecture on Field Programmable Gate Array (FPGA) technology is also developed for the implementation of the stabilization system. Stabilization of video sequences captured under non-uniform lighting conditions begins with a nonlinear enhancement process. This improves the visibility of the scene captured from physical sensing devices which have limited dynamic range. This physical limitation causes the saturated region of the image to shadow out the rest of the scene. It is therefore desirable to bring back a more uniform scene which eliminates the shadows to a certain extent. Stabilization of video requires the estimation of global motion parameters. By obtaining reliable background motion, the video can be spatially transformed to the reference sequence thereby eliminating the unintended motion of the camera.

A reflectance-illuminance model for video enhancement is used in this research work to improve the visibility and quality of the scene. With fast color space conversion, the computational complexity is reduced to a minimum. The basic video stabilization

model is formulated and configured for hardware implementation. Such a model involves evaluation of reliable features for tracking, motion estimation, and affine transformation to map the display coordinates of a stabilized sequence. The multiplications, divisions and exponentiations are replaced by simple arithmetic and logic operations using improved log-domain computations in the hardware modules. On Xilinx's Virtex II 2V8000-5 FPGA platform, the prototype system consumes 59% logic slices, 30% flip-flops, 34% lookup tables, 35% embedded RAMs and two ZBT frame buffers. The system is capable of rendering 180.9 million pixels per second (mpps) and consumes approximately 30.6 watts of power at 1.5 volts. With a 1024×1024 frame, the throughput is equivalent to 172 frames per second (fps).

Future work will optimize the performance-resource trade-off to meet the specific needs of the applications. It further extends the model for extraction and tracking of moving objects as our model inherently encapsulates the attributes of spatial distortion and motion prediction to reduce complexity. With these parameters to narrow down the processing range, it is possible to achieve a minimum of 20 fps on desktop computers with Intel Core 2 Duo or Quad Core CPUs and 2GB DDR2 memory without a dedicated hardware.

## ACKNOWLEDGMENTS

First of all, I would like to thank Dr. K Vijayan Asari for his guidance and efforts as my dissertation director. I am also very grateful for such an opportunity to explore on a broader view of the chosen topic. I would also like to thank Dr. James Leathrum, Jr, Dr. Jiang Li, and Dr. Shunichi Toida for their invaluable time and consideration in serving on my dissertation committee. I greatly appreciate their efforts. Finally, I would like to thank my family and friends for all their support.

# TABLE OF CONTENTS

# List of Tables

## List of Figures

# CHAPTER 1

# INTRODUCTION

The theme of this dissertation focuses on reducing the complexity of certain calculations in video stabilization by decomposition and the structural representation of the dataset into smaller sub-features. This methodology enables us to overcome the drawback of conventional performance-resource trade-off in hardware designs by concentrating the computation on the most distinct sub-feature and sustaining a one-on-one throughput rate. The main contributions of this dissertation are listed in section 1.3 followed by the organization of this book in section 1.4.

## 1.1 Motivation of the Research

Video Stabilization is an essential part of the video processing technology for scenes captured under shaky conditions. From the perspective of an audience, extraction of information from such a video source can be distracting, thus making it very difficult to concentrate and exhausting to track the target of interest from the scenes. In extreme cases, it is impossible to identify the details from such a scene with large variations when the frames are averaged through our eyes' perception. However, the vibrative motion of the camera is not the only problem. Videos captured under non-uniform lighting conditions are mainly contributed from the limitation of physical sensing devices. Due to the limited dynamic range of the sampling circuitry, the brighter region of the image

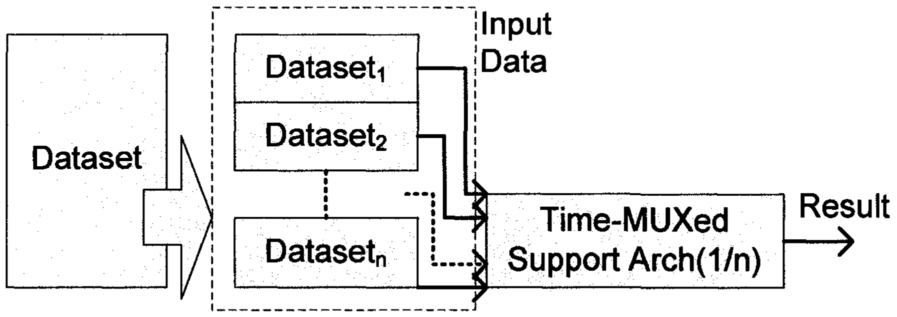
Format of this dissertation is *IEEE Transactions on Computers* saturates the photo site of sensing elements, causing the device to compensate itself and shadow out slightly darker parts of the scene. While there are several image enhancement algorithms available, the method which is capable of simultaneous rendering of the luminance and contrast components of the color images is not currently available for efficient design of the architecture.

The motivation of this dissertation is to find a robust algorithm to immobilize the video by compensating for the background motion of the camera. Another objective is to develop a high performance system architecture in FPGA technology for the stabilization of video sequences captured under non-uniform lighting conditions. In this research, we apply a reflectance-illuminant model for video enhancement to improve the visibility and quality of the scene. With fast color space conversion, the computational complexity is reduced to a minimum, further simplifying the hardware design. The basic video stabilization model is formulated and simplified for implementation. Such a model involves evaluation of reliable features to track, feature measure and tracking, motion estimation, and affine transformation to map the display coordinates of stabilized sequences. Novel architectures for performing these calculations are also proposed in this dissertation. With improved log-domain computation, all multiplications, divisions and exponentiations are replaced by simple arithmetic and logic operations. On a Xilinx's Virtex II 2V8000-5 FPGA platform, the prototype system consumes 59% logic slices, 30% flip-flops, 34% lookup tables, 35% embedded RAMs and two ZBT frame buffers. The system is capable of rendering 180.9 million pixels per second (mpps) and consumes approximately 30.6 watts of power at a 1.5 volt internal operating voltage. With a $1024 \times 1024$ maximum frame, the throughput is equivalent to 172 frames per second (fps).

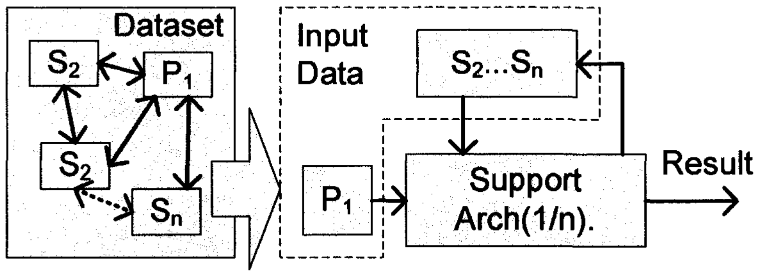
## 1.2 Proposed Theme of the Dissertation

We often face the decision of performance-resource optimization due to hardware constraints and the performance needs of specific applications. The performance parameter usually has an inverse relationship with the amount of assisting hardware necessary to achieve certain calculations within a given time. To minimize the resource, conventional methods usually compute the partial results on a timeslot shared architecture and construct a set of distributed queues to hold the partial results which are accumulatively combined to produce a more complete output. This concept can readily be illustrated in Fig 1a. A dataset is first uniformly divided into $n$ subsets, where $n$ is the reduction factor of computing elements, $1/n$ is the throughput parameter. The subsets are fetched into the support architecture in a timely manner. Only one subset may occupy the data path to the support architecture at a clock cycle. The partial results are properly saved to the distributed queues to be accumulated in subsequent cycles. It requires $n$ clock cycles to complete the evaluation of an entire dataset.

To sustain the peak performance of a system while reducing hardware complexity, we propose to represent the dataset by sub-features in a structured constellation as shown in Fig 1b. The full dataset is first decomposed into a primary sub-feature, $P_1$, and a set of secondary sub-features, $S_{2..n}$. The criteria of evaluating the sub-features are application dependent; however, the general rule is to extract the most distinctive characteristics for the primary sub-feature and select sub-optimal regions to be secondary sub-features. With the coordinates of the sub-features obtained from the process of feature decomposition, these sub-features form the distance and angle relationships with the primary sub-feature in a structure which identifies the complete dataset. The support architecture for partial evaluation is similar to the time-multiplexed architecture in Fig 1a. Nonetheless, only $P_1$ is being evaluated at all time with respect to the region of interest. With the successful evaluation of the primary sub-feature, subsequent measure of secondary sub-features is enabled at the proper spatial locality predetermined by the structure representing the dataset. The secondary sub-features, $S_{2..n}$, are evaluated only once for every valid measure of $P_1$. A successful measure of all sub-features contributes to a correct



(a) Time-multiplexed evaluation of a complete dataset.



(b) Distinctive characteristics oriented partial evaluation with sub-feature representation.

Figure 1: Decomposition and structural representation of the dataset.

evaluation of the dataset. This method allows us to evaluate a larger dataset on a dedicated hardware with limited computing elements and at the same time provides roughly a one-on-one system performance.

## 1.3 Main Contributions

The main contributions of this dissertation can be summarized as follows:

1. Model of feature representation by constellation. The sub-feature methodology decomposes a feature into sub-features based on the criteria of distinctive characteristics (textural optimality in our case) of the dataset. A region with the most distinctive characteristics is assigned as primary sub-feature along with a set of non-overlapped secondary sub-features having sub-optimal regions. A brute force evaluation of the complete feature is eliminated with the computational power concentrated on the primary sub-feature. For every successful measure of the primary sub-feature, the secondary sub-features are evaluated to confirm the existence of such structure which represents the complete feature. By representing the full feature in primary sub-feature and a set of secondary sub-features, the amount of necessary calculations can be dramatically reduced. For hardware design, it is not necessary to build a huge architecture to evaluate the complete feature; instead, one only needs to implement a support system for the sub-feature dimension. The model also encapsulates the spatial deformation or distortion of the full feature through the angle and distance relationships among the sub-features within the constellation. With further extension, this idea can be readily applied for adaptive tracking of rigid objects.

2. Design of various subsystems for video stabilization, namely, the potential feature selection, feature measure and tracking, and the angle calculation for inlier motion estimation. The single most important component, however, is the feature measure which involves the computation of 2D normalized cross correlation (NCC). The first step to reduce complexity is to perform a partial measure of NCC on primary sub-features. Another important aspect of NCC architecture is its ability to sustain the peak performance without the performance-resource trade-off. With the assistance of sub-feature measures, the representation of the full feature served as confirmation of primary sub-features; the NCC architecture virtually appears to handle much larger feature templates. It is interesting to note that such a design is the direct extension of our previous research based on the generic architecture of 2D convolution.
3. Design of low complexity architecture for video enhancement. While we already have several implementations of various algorithms, the architecture presented in this dissertation has the lowest complexity of all previous approaches. The idea of generalized 2D convolution with quadrant symmetry property from master thesis has also proven to be very flexible in deploying the concept to various image enhancement architectures. Virtually all image processing algorithms involve some kind of filtering operations that often has quadrant symmetric kernels.
4. Design of improved logarithmic modules. A simple bit-level error correction is introduced to increase the precision of $Log_2$ and $iLog_2$ modules with improved pipelining. The hardware complexity for 32-bit numbers is also further reduced while maintaining only 8-bit fixed point for the fractions. The research done from

the master thesis, "A multiplier-less architecture for high speed computation of multi-dimensional convolution", has proven log-domain computation to be very useful for reducing the complexity in hardware designs.

5. Global motion evaluation with triangular order of search. By calculating the angles between the vectors alone, we can quickly estimate the background motion. It is accomplished by searching, without any redundant calculation, for the most outstanding element within a collection of motions. The outstanding element can be applied to further narrow down the motions of subsequent video frames, hence, forming a star constellation based on the stability of the outstanding element in relation to other nodes within itself.
6. Basic model of feature evaluation based on different types of textures. Due to hardware related issues, only a single layer of texture already available from the literature has been selected. The uniqueness of the features proved to be least useful as the required processing bandwidth can become highly non-uniform which is not very suitable for hardware implementation.
7. Application of fast color space conversion to speed up the video enhancement on desktop computers to 30 fps with 3.2GHz Intel P4H processor and 1.5GB DDR1 memory.

Future work should not focus on further improving the performance of hardware system since the throughput is already excessively high; however, it should optimize the performance and the resource to meet the specifics based on the nature of the applications.

Future development should also extend the great potential of the model into finer grains for extraction and adaptive tracking of moving objects as our model inherently encapsulates the attributes of spatial distortion and motion prediction to reduce complexity. With these parameters to narrow down the processing range, it is possible to achieve a minimum of 20 fps on desktop computers with Intel Core 2 Duo or Quad Core CPUs and 2GB DDR2 memory without dedicated hardware.

## 1.4 Organization of Forthcoming Chapters

The remainder of this dissertation is organized as the following. A brief survey of image enhancement, feature evaluation and motion estimation is discussed in chapter 2 regarding the fundamental problems involved with the stabilization. Chapter 3 introduces the issues of complexity of certain operations commonly applied in image processing. Chapter 4 addresses the theoretical model formulation and the simplification toward designing hardware efficient high-speed architecture. The design of different subsystem modules is illustrated in-depth in chapter 5. The simulation results and error analysis along with the parameters of performance and resource allocation are given in chapter 6. The conclusion and the comments regarding future development of the video stabilization system are presented in chapter 7.

# CHAPTER 2

# ALGORITHMIC BACKGROUND

In this chapter, we briefly describe the fundamental issues involved with the task of stabilizing the video sequence. One of the issues is video enhancement for which one must compensate the visual quality of the scenes captured from cameras with limited dynamic range. The second issue requires the detection of reliable features to establish the correspondence between the video frames. Various approaches for estimation of background motion are addressed for derivation of parameters necessary for stabilization.

## 2.1 Necessity of Image and Video Enhancement

Physical limitations exist in the sensor arrays of imaging devices, such as CCD and CMOS cameras. Often the videos captured by these devices cannot properly represent scenes that have both very bright and dark regions. The sensor cells are commonly compensated with the amount of saturation from bright regions fading out the details in the darker regions. Image enhancement algorithms [1], [2], [3], [4] provide good rendering to bring out the details hidden due to dynamic range compression of the physical sensing devices. For applications in color images these algorithms may fail to preserve the color relationship among RGB channels which result in distortion of color information after enhancement. Thus, there is still room for improvement. The recent development of a fast converging neural network based learning algorithm called Ratio Rule [5], [6] provides an excellent solution for natural color restoration of the image after gray-level image enhancement. Hardware implementation of such algorithms is absolutely essential to parallelize the computation and deliver real time throughputs for color images or video processing containing extensive transformations and a large volume of pixels. Implementation of window related operations such as convolution, summation, and matrix dot products which are common in enhancement architectures demands enormous amount of hardware resources [7], [8]. Often a large number of multiplications/divisions is needed [9]. Some designs compromise this issue by effectively adapting the architectures to very specific forms [7], [8], [10] and cannot operate on different sets of properties related to the operation without the aid of dynamic reconfiguration in an FPGA based environment. We proposed the concept of log-domain computation in [11] to solve the problem of multiplication and division in the enhancement system to significantly reduce the hardware requirement while providing a high throughput rate.

Algorithms developed under the reflectance-illuminance category of the image processing models are not unique. The theorization of such a model for visual representation originated in the early 1970's [12] with stochastic image processing in [13] to reduce the salt-and-pepper noise (black and white dots imposed from poor quality sensing device available at the time). In classical approaches, homomorphic processing operates exclusively on the grayscale images. Recently the concept has become popular for adapting the model to color image representation. Although the concepts for a number of exotic approaches are generalized by Kimmel *et al* [14], dedicated architectures for such algorithms are generally unavailable; thus, comparison is limited to existing designs relevant to the subject. One of the few well explored and adapted techniques (in both hardware and software) in this category is Multi-Scale Retinex (MSR) related model developed by Jobson's research team [2], [3], [15]. By the nature of the algorithmic procedure, MSR is suitable for DSP based implementation discussed in [16] where the fast Fourier transform (FFT) and inverse FFT (IFFT) may be readily plugged in from the DSP library [17], [18]. Further improvement on MSR can be made for better color consistency to minimize the influence from background color. Within the same category, we presented a hardware-efficient architecture in [19] for enhancement of the digital color images with non-uniform darkness using a Ratio learning algorithm [5] [6] for color distortion correction. We also proposed the nonlinear enhancement architecture in [20] based on [21] which results with similar quality on the output images. As far as efficiency is concerned, tweaking of the enhancement processes needs to be further exploited for potential speed up and hardware reduction.

## 2.2 Evaluation of Good Tracking Features

Modeling of artificial neural networks (ANN) to solve real-world problems is inspired by biological neural systems. Such systems are simplified for ANN where the neurons are characterized solely by the biologic machinery but the ability to adapt, learn, and generalize in response to given types of information within the network architectures are governed by certain learning rules. The successors of such models mimic the biological functionality of the systems quite well. Virtually all forms of modification of the synaptic weights between neurons are in some ways variations of Hebbian or Delta rule in ANN whether the networks are feed-forward or recurrent [22].

While feed-forward architectures such as perceptron and adaline [23] [24] [25] have strict limitation where no feed-back or back-propagation exists for error correction, recurrent networks significantly increase the dynamics of the network. One of those earliest recurrent networks was introduced by Anderson and Kohonen in [26] [27] and generalized by Hopfield in 1982 [28] with primary applications for associative memory which remembers the patterns and pattern recognition. Examples of applications include optimization in power systems [29]. For classification, the unsupervised Fuzzy Adaptive Resonance Theory, Fuzzy-ART, neural network, introduced in 1976 by biological phenomena [30] can be useful. Fuzzy-ART is capable of clustering documents with the ability to mine data and discover knowledge dynamically by a wide variety of techniques [31]. It can also be applied for rapid stable learning to categorize and recognize the patterns [32]. The supervised Fuzzy-ART called Fuzzy ARTMAP can learn incrementally for category recognition with new minima learning rule [33].

Inspired by the concepts of adaptive resonance theory based neural networks and Hopfield recurrent network, a new neural architecture is desirable to fuse different characteristics [34] for automatic extraction and selection of a set of unique features from a video stream. The same network should also be able to track the features to maintain the correspondence between video frames and minimize iterative error measures. Such features would be useful for estimation of motion parameters. While the ANN has the capacity to support pattern related classes of applications, the iterative nature of the process itself imposes the bottleneck of non-constant bandwidth access of the storage components in dedicated high performance system architectures. Nonetheless, the specific textures can be considered reliable for tracking. In general, good features are characterized by the distinctiveness of different types of textures.

Scale invariant feature templates (SIFT) can be very helpful for object detection [34][74]. To identify the correspondent coordinates of a feature in another picture, the image must be re-sampled into different resolutions to construct a pyramid of images. Within each resolution, feature selection is performed based on certain criteria. The most consistent features of the pyramid are extracted as the scale invariant templates for subsequent processing. In conjunction with rotation invariant features through affine normalization, the multi-resolution feature extraction has proven to be vital for the construction of image descriptors and the accumulation of its database for autonomous object detection [75]. Our main focus, however, is to identify reliable features with respect to a current video frame under the legitimate assumption that variation of scale and rotation are gradual within a video sequence. Hence, it is not necessary to represent the features according to image descriptors with scale and histogram orientation of certain key points.

## 2.3 Evaluation of Motion Parameters

Evaluation of the features alone may seem insignificant; however, such a step is crucial when combined with a variety of motion analysis and estimation. Motion estimation (ME) is a process of evaluating the relationship between the frames such that the contents of the frames are approximately stationary with respect to the reference frame through transformation of motion parameters. Global motion estimation (GME) is an instance of ME which involves the monitoring of background or dominant motion. ME has a very broad applications in video processing technology. In video compression, the ability to accurately estimate the motions, not necessarily global motion alone, determines the compression ratio, resulting in smaller video files [35]. In segmentation, motion information helps in distinguishing between moving objects from the background [36]. In registration and mosaicing applications, motion vectors contribute to the key components in identifying orientation for stitching the frames into a more complete scene [37].

ME search algorithms can be divided into three categories based on their complexity. The full search algorithms (FSA) contribute to the most optimal match yet impractical with overwhelmed complexity O($n$), where $n$ is the search range. The cost for block search is O($n/m$), where $m$ is the block range, and may be as low as O(log($n$)) with logarithmic search [38]. In video compression the blocks are usually divided into macro-blocks to further reduce the search range with a trade off of increasing the distortion and the assumption of block-wise uniform motion [39]. The hybrid search serves to balance the complexity and accuracy [40].

The class of gradient/differential based ME algorithms is commonly modeled by (2.1), where $I_t(\mathbf{x})$ is the current frame with coordinate vector $\mathbf{x}=[x,y]^T$, $I_{t-1}$ is previous frame or reference, $G$ function is the affine transformation by motion vector **M**, and err(**x**) is the error. The coefficients $a_i$ and $b_i$ are the rotation and scale of affine parameters while $d_x$ and $d_y$ are the displacement or translational motion between the frames. Two well known cost functions for error measure defined in (2.2) are mean absolute error (MAE) and mean square error (MSE) [41] for which **M** has the solution of least squares regression.

The error minimization has a first order Taylor expansion of (2.3). The resolution pyramid is often constructed to iteratively estimate the motion parameter for convergence from course to fine resolutions. Such differential techniques assume that the intensity of the images is conserved reducing its reliability to subtle change by noise and illumination. Higher orders of Taylor expansion further assumes that the subsequent gradients be conserved which implies the ill-posed problems to rotation, scaling as well as sources of noise [42]. The approach essentially becomes less stable as the constraint is overly specified. A more troublesome part of the gradient descending ME is that the search algorithm fails when trapped into local minima [43]. Lucas introduced weighting to error measure defined in (2.4) to give more influence to centre pixel under the window [44]. This concept can be extended to increase the reliability of selected regions for motion measure. Such confidence factor can be modeled with Bayesian statistics through observation over time to enhance regions with low noise and small aperture problems or suppressing otherwise [45]. Multi-frame buffering and frame sub-sampling are usually required for ME with differential techniques. They translate to greater memory usage with added complexity to buffer flow in hardware realization. The potential of measuring global motion can be explored through consistency of motions to avoid iterative error measure and minimize frame buffering. Once the motion parameters are sustained and compensated by intended motion, standard affine transforms may be performed to minimize shakiness of the scene, hence, stabilizing the video. While the motion parameters may be estimated through measure of residues with respect to the reference frame, we should examine the basic properties of calculating the coherence of different motions.

$$\begin{aligned} I_t(\mathbf{x}) &= I_{t-1}\left(G(\mathbf{x},\mathbf{M})\right) + err(\mathbf{x}) \\ G(\mathbf{x},\mathbf{M}) &= \begin{bmatrix} a_1 & a_2 \\ b_1 & b_2 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} d_x \\ d_y \end{bmatrix}, \; or \\ & \begin{bmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \end{bmatrix} \times \begin{bmatrix} a_1 & a_2 & d_x & b_1 & b_2 & d_y \end{bmatrix}^T \\ &= \mathbf{C}(\mathbf{x}) \times \mathbf{M} \end{aligned} \tag{2.1}$$

$$\begin{aligned} err(\mathbf{x}) &= \arg\min_{\mathbf{M}} \left\{ E\left( \left| I_t(\mathbf{x}) - I_{t-1}\left(G(\mathbf{x},\mathbf{M})\right) \right| \right) \right\} \\ err(\mathbf{x}) &= \arg\min_{\mathbf{M}} \left\{ E\left( I_t(\mathbf{x}) - I_{t-1}\left(G(\mathbf{x},\mathbf{M})\right) \right)^2 \right\} \end{aligned} \tag{2.2}$$

$$I_t(\mathbf{x}) = I_{t-1} + \sum_i \frac{\partial I_{t-1}}{\partial M_i} M_i \tag{2.3}$$

$$\sum_{\mathbf{x}\in\Omega} W(\mathbf{x})^2 \left[ \nabla I(\mathbf{x},t) \cdot \mathbf{M} + I_t(\mathbf{x},t) \right]^2 \tag{2.4}$$

## 2.4 Summary

In this chapter, we discussed the fundamental limitation of physical sensing devices for which the cameras had narrow dynamic range. Saturation in part of the image tended to shadow out the details in other regions of the scene. Different image enhancement algorithms often required several separate operations for contrast improvement, luminance enhancement, color correction and color restoration. We found it necessary to apply a simpler model and at the same time eliminate color correction and restoration. Such a model should minimize the number of multiplications to reduce complexity while produced reasonable image quality to improve the visibility of the scene. Moreover, we intended to design an enhancement subsystem capable of fine-tuning certain parameters to meet the need. In computer vision, most feature evaluation concepts available in the literature often model the scale, rotation and orientation as part of the efforts to recognize certain invariant features. It required a significant degree of computation frequently too difficult to implement in hardware. The processing nature of certain calculations would be highly non-linear, therefore, extremely difficult to implement. Our interest would be to only evaluate reliable features with respect to current conditions within the scene. It is desirable to derive a simple model for feature evaluation which has low complexity with minimum storage space. Hence the evaluation technique has to support feature extraction on the fly. A class of gradient based motion evaluation techniques was also evaluated in this chapter. These approaches are iterative nature in the process of obtaining motion parameters. The assumption of conservative image intensity further poses the ill nature in the presence of noise. Similar to the feature evaluation these techniques often require the storage of entire video frames for which certain prediction must be iteratively measured by means of residual errors. In theory, if the feature evaluation works well, it is not mandatory to extract motion parameters from the entire frames. In the forthcoming chapters, a simpler mean of video enhancement, feature evaluation and motion estimation suitable for hardware realization are analyzed for the stabilization of video sequence.

# CHAPTER 3

# COMPUTATIONAL COMPLEXITY REDUCTION

In advance to in-depth discussion of the theory behind the algorithm, let us begin by a briefing on the fundamental problems of the complexity itself regarding most commonly used operators such as division, multiplication, exponentiation and some form of summation equivalent to window/kernel based operations such as matrix multiplication.

## 3.1 Redundancy of the Operator

As introduced in chapter 2, window based operations are very common in video processing technology such as generalized 2D convolution. Often, coefficients associated with these kernels are non-arbitrary and exhibit interesting properties. It is a waste of the computational power and resource allocation from hardware designers' perspective to not take advantage of certain symmetries within the kernels. Such symmetries are very common in the design of digital filters. In particular, we utilize the quadrant symmetry (QS) of the kernels to support convolution operations (digital filtering). This preprocessing ideally saves close to 75% of the multiplications in addition to the replacement of the hardware multipliers discussed in [46]. Such optimization results with the architecture that is neither too specific nor generic while focusing the essential computation to a single quadrant. It maintains the flexibility of redefining the filter characteristics at run-time (soft upgrade) without recompiling and dynamically reconfiguring the architecture (hard upgrade) by external systems. Examples of the filters qualified for QS property include both separable [47] and non-separable kernels. QS also encapsulates circularly symmetric kernels such as Gaussians and Laplacian of Gaussians used for smoothing and edge detection, respectively. In summary, one can minimize the computational power by simply exploiting the redundant properties of certain operators. The reliance on redundancy alone, however, is insufficient for hardware implementation of the dedicated architectures which demand relatively complex calculations.

## 3.2 Concept of Logarithmic Domain Computation

This section describes the basics of logarithmic approximation. A common technique which relies on piecewise straight lines for error correction to various precisions is also illustrated in the subsections. We propose bit-level curve fitting as a mean to generate the correcting coefficients and achieve similar precision compared to other approaches.

### 3.2.1 Simplicity of Approximation and Its Benefits

Multiplications and divisions become additions and subtractions with logarithmic transformations logically defined by (3.1) which require significantly less computational power. A number to the power $n$ becomes a matter of arithmetic shift operation achievable within single clock cycle for $n$ equals to power of two, or multiplication operation for any finite $n$ in general. Eq. (3.1) states that the $\log_2$ scale of $V$ can be calculated by concatenating the index $I_V$ of leading 1's in $V$ with the fractions (remaining bits after $I_V$th bit). The reversed process holds true as well, except the leading 1's and fractions, $L_f$, are shifted to the left by $L_i$ (integer of $L$) bits as shown in (3.1). This definition is generalized to integer values as well as fraction numbers. For example, $\log_2(0000.0110)$ binary becomes -1.5 or (-2 + 0.5) in decimal since the position of $I_v$ is -2 (two places after decimal point) with fraction 0.10 in binary. The correct value should be -1.415 which results with 6% error from the approximation for worst case scenario. Application of this concept eliminates most costly components just described for hardware designs. Thus, it is crucial to implement efficient logarithmic estimation modules in such a way that is very compact in its design, reduces large amount of hardware resource, and provides very high throughput rate [11] [19]. Designs based on the concept presented in [48] which employees unrolled pipeline architectures such as [20], [49] and [50] may not be efficient for replacement of multiplications and divisions in window related architectures for FPGA based implementation. Particularly in filters, such architectures usually require a large number of multiplications and the amount of hardware resource allocated for the unrolled pipeline stages usually can come close to the cost of the multipliers on FPGA technology. Our implementation of the estimation modules packs the resolution-dependent unrolled pipeline style design into a few stages regardless of its resolution and at the same time optimizes the component count, power and speed. It is about 10 times reduction in the resource and 170% performance boost in FPGA environment. We generalized the modules to support both integer and fraction numbers without introducing hardware complexity. These modules are also insensitive to the bit-resolution that exists in hardware multipliers in which the performance is inversely proportional to the number of bits in the multipliers. We have demonstrated the use of log domain computations in [19], [46] for image processing applications with a figure of 60% hardware reduction in addition to the applicable QS based architecture. The error correction to enhance such approximation is discussed in next section.

$$\log_2(V) \cong \{I_V\} + \{(V - I_V) \gg I_V\} \Leftrightarrow \log_2^{-1}(L) \cong \{1 \ll L_i\} + \{L_f \ll L_i\} \tag{3.1}$$

### 3.2.2 Improvement of Precision with Piecewise Straight Lines

Mitchell's logarithmic converter proposed in [48] was derived based on binary representation of a number N in

$$N = \sum_{i=j}^{k} b_i 2^i \tag{3.2}$$

as a summation of binary coefficients, $b_i$ with respect to the placement, $2^i$. The k is an index (aka characteristic of $\log_2 N$) for which the most significant bit (MSB) of N in binary equals to 1's. Given $b_k=1$, the term $2^k$ can be factored out to simplify (3.2) by

$$N = 2^k(1+f), \tag{3.3}$$

where f is the fraction of the remaining terms of (3.2). The $\log_2$ scale of (3.3) is defined by

$$\log_2 N = k + \log_2(1+f) \tag{3.4}$$

and can be approximated by

$$\log_2 N' = k + f, \tag{3.5}$$

with the slop of line equal to one between consecutive points of exactly power of two. A different perspective to the approximation using power series can eventually reach the same conclusion for fix-point N in addition to strictly integer values [46]. The error is measured by the difference between exact $\log_2 N$ and approximated $\log_2 N'$:

$$Err(N) = \log_2 N - \log_2 N'. \tag{3.6}$$

Eq (3.5) only requires arithmetic addition and may be implemented completely free of any multiplication. Mitchell demonstrated the design with counter and shift register in serial form which requires minimum hardware resources at the expense of the largest number of clock cycles necessary. The result of Mitchell's approximation is shown in Fig 2a along with the difference error in Fig 2b, which is quite symmetrical with x-axis in $\log_2$ scale. The periodic nature of the difference error makes it possible for bit-level error correction as an alternative to piece-wise linear approaches [49][50][51][52][53].

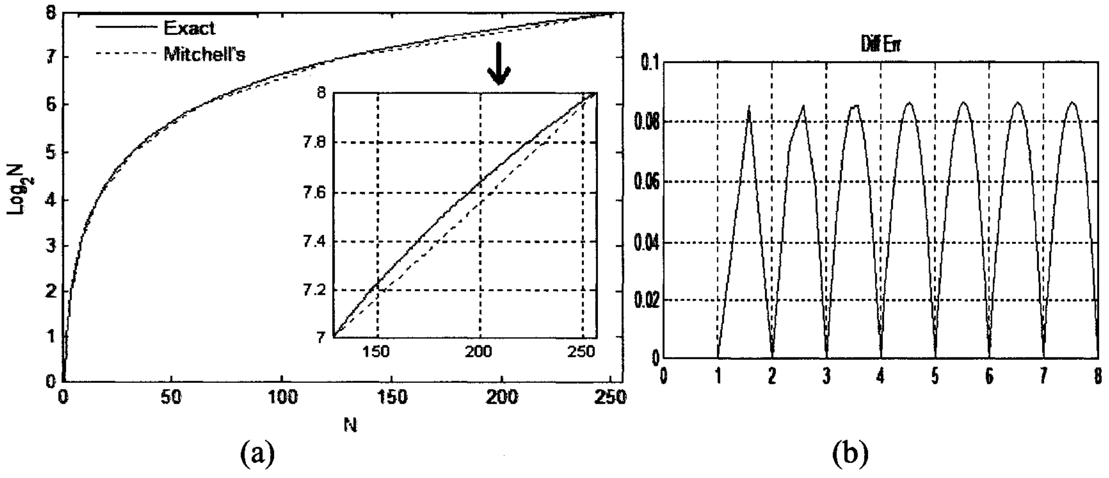


Figure 2: Mitchell's $\log_2$ approximation (a) and the difference error (b).

In general, piecewise linearly corrected logarithmic converters maintain the following form:

$$\log_2 N' = k + f + \left(f \times \sum c_R + \sum d_R\right), \tag{3.7}$$

where $c_R$ is the single-bit slop in power of two to eliminate real multiplications and minimize error, and R denotes the divided regions for such linearization. The number of binary coefficients in $\sum c_R$ is determined to be fewest possible for realization of low complexity circuitry. In 1999, SanGregory proposed the two-region piecewise linear correcting factor [52]. The single-bit coefficient was selected by minimizing root-mean-square error. His correction algorithm can be summarized as follows:

$$\log_2 N' = k + f + \begin{cases} 2^{-2} f_{4MSB}, & for\, 0 \le f < 1/2 \\ 2^{-2} \overline{f}_{4MSB}, & for\, 1/2 \le f < 2/2 \end{cases}, \quad (3.8)$$

where $\overline{f} \approx 1\text{-f}$, or the descending part of the error curve in Fig 2b. The difference error with $0 \le f < 1$ is approximately symmetrical around the midway in linear scale. SanGregory chose to only incorporate 4 MSBs of the fraction to generate a three bits correcting factor to improve accuracy yet maintain very low hardware overhead. Dated back to 1965, Combet also improved Mitchell's algorithm with a four region error correction in serial architecture with increased circuit complexity [51]. His algorithm was based on trial and error in selecting the straight lines and can be defined as:

$$\log_2 N' = k + f + \begin{cases} (2^{-2} + 2^{-4}) f, & for\, 0 \le f < 1/4 \\ 2^{-4} + 2^{-6}, & for\, 1/4 \le f < 2/4 \\ 2^{-3} \overline{f} + 2^{-6} + 2^{-7}, & for\, 2/4 \le f < 3/4 \\ 2^{-2} \overline{f}, & for\, 3/4 \le f < 4/4 \end{cases}. \quad (3.9)$$

Hall also adapted Combet's idea with more coefficients for better accuracy yet seemed to defeat the desire for solutions with a simple hardware requirement [54]. In 2003, Abed refined the work done in [52] and extended the piecewise straight line approach to offer two, three, and six region error correction algorithms for 32-bit integer numbers. His formulation for the implementation presented in [53] can be summarized as follows with two, three, and six-bit correcting factors, respectively:

$$\log_2 N' = k + f + \begin{cases} 2^{-2} f_{3MSB}, & for\, 0 \le f < 1/2 \\ 2^{-2} \overline{f}_{3MSB}, & for\, 1/2 \le f < 2/2 \end{cases}, \quad (3.10a)$$

$$\log_2 N' = k + f + \begin{cases} 2^{-2} f_{4MSB}, & for\, 0 \le f < 1/4 \\ 2^{-4} + 2^{-6}, & for\, 1/4 \le f < 3/4 \\ 2^{-2} \overline{f}_{4MSB}, & for\, 3/4 \le f < 4/4 \end{cases}, \qquad (3.10b)$$

$$\log_2 N' = k + f + \begin{cases} 2^{-2} f_{6MSB}, & for\, 0 \le f < 1/16 \\ 2^{-2} f_{6MSB} + 2^{-6}, & for\, 1/16 \le f < 4/16 \\ 2^{-4} + 2^{-7} + 2^{-8}, & for\, 4/16 \le f < 6/16 \\ 2^{-4} + 2^{-6} + 2^{-7}, & for\, 6/16 \le f < 10/16 \\ 2^{-4} + 2^{-7}, & for\, 10/16 \le f < 12/16 \\ 2^{-2} \overline{f}_{6MSB}, & for\, 12/16 \le f < 16/16 \end{cases}, \qquad (3.10c)$$

In general, increasing the number of regions results with smaller approximation error defined in (3.6) at the cost of additional logic gates and adder cells. In the case of six-region method, two cascaded adder arrays are needed which can reduce the overall performance of the logarithmic converters. We now present the idea of bit-level curve fitting to generate a three-bit correcting factor. Furthermore, we apply it beyond integer values to include fixed-point representation, given consideration of the precision.

### 3.2.3 Bit-level Curve Fitting

The process of calculating inverse-$\log_2$ is to undo the $\log_2$ conversion which has the following relationship:

$$\log_2(N) \cong \{k_N\} + \{(N - 2^{k_N}) \gg k_N\} \Leftrightarrow \log_2^{-1}(L) \cong \{(1 + L_{\text{frac}}) \ll L_{\text{int}}\}, \qquad (3.11)$$

where $\gg$ and $\ll$ denote the opposite data bus shifting operation. Note that neither (3.5) nor (3.11) restricted us from defining negative index, $k < 0$; however, only the integer portion (assuming non-negative k) has been exploited in the literature to our best knowledge. For a fixed-point decimal of 8/8 (8-bit integer and 8-bit fraction), the same rule holds true. We need to find a mechanism to express the k in two's complement. By using the standard priority encoder, we found that the single bit-inversion of the MSB at the output of the encoder does the trick. A single bit of inverter logic generalizes the architecture to accept both integer and decimal values. By including fraction values, it may seem to complicate the problem that shifting operation of (3.11) can go either way depending on the sign of k; the logic shift remains unidirectional in the implementation. To avoid real computation and minimize delay in realizing high speed parallel architectures, one is often left with very few choices. Besides the linear methods of (3.8) to (3.10), curve fitting at the logic level can also achieve a high degree of precision without introducing complex circuits.

Rather than applying piecewise straight lines, we examined the dataset of the difference error shown in Fig 2b to determine a close fit for generating such correcting factor. At the same time, the correction should not be dependent on all fraction bits to minimize circuit complexity. Examples of binary logarithmic conversion are shown in Table I for 5-bit integers with index value k = 4. On the rightmost column is the error pattern without the correcting factor. Given a finite set of data points and the coherent near-symmetric error bits, one can utilize a large ROM table to correct the error to its best precision. Although not entirely impractical, such an approach does not work well in its scalability as the size of ROM storage exponentiates with the increasing resolution of input integers. It is therefore wise to focus solely on the bits which contribute to the largest magnitudes of error. Based on the simulation analysis at higher precision with consideration of rounding, it was determined that the last five bits of fraction coefficients can provide sufficient improvement for 8-bit fixed-point representation. Furthermore, optimization on highlighted bits of the error coefficients, EC, (on last column of Table I) shows best trading of higher precision with a reasonably small set of logic gates. Unlike piecewise straight line methods discussed in previous section, bit-level curve fitting needs to be optimized at a much higher resolution for more accurate representation by logic gates.

Table 1: Example of $\log_2$ converter with 5-bit integers and the index $k = 4$.

| **Input_bin** | **Log2_dec** | **Log2_bin** | **Log2appx_bin** | **Log2 - appx** |
|---|---|---|---|---|
| 10000: | 4 | 100.00000000 | 100.**00000**000 | 0.000**000**00 |
| 10001: | 4.0875 | 100.00010110 | 100.**00010**000 | 0.000**001**10 |
| 10010: | 4.1699 | 100.00101011 | 100.**00100**000 | 0.000**010**11 |
| 10011: | 4.2479 | 100.00111111 | 100.**00110**000 | 0.000**011**11 |
| 10100: | 4.3219 | 100.01010010 | 100.**01000**000 | 0.000**100**10 |
| 10101: | 4.3923 | 100.01100100 | 100.**01010**000 | 0.000**101**00 |
| 10110: | 4.4594 | 100.01110101 | 100.**01100**000 | 0.000**101**01 |
| 10111: | 4.5236 | 100.10000110 | 100.**01110**000 | 0.000**101**10 |
| 11000: | 4.585 | 100.10010101 | 100.**10000**000 | 0.000**101**01 |
| 11001: | 4.6439 | 100.10100100 | 100.**10010**000 | 0.000**101**00 |
| 11010: | 4.7004 | 100.10110011 | 100.**10100**000 | 0.000**100**11 |
| 11011: | 4.7549 | 100.11000001 | 100.**10110**000 | 0.000**100**01 |
| 11100: | 4.8074 | 100.11001110 | 100.**11000**000 | 0.000**011**10 |
| 11101: | 4.858 | 100.11011011 | 100.**11010**000 | 0.000**010**11 |
| 11110: | 4.9069 | 100.11101000 | 100.**11100**000 | 0.000**010**00 |
| 11111: | 4.9542 | 100.11110100 | 100.**11110**000 | 0.000**001**00 |

The binary logic fitting analysis of precision-circuit trading pinpoints to the generation of $EC_{-4}$ to $EC_{-6}$ depending on the fraction bits $f_{-1}$ to $f_{-5}$ and the potential rounding of EC itself. Based on the results, it was concluded that the following simple logic equations reduce the average magnitude of error to approximately one tenth of Mitchell's estimation:

$$\begin{aligned}
\mathrm{EC}_{-4}(f) &= (\bar{f}_{-1} \vee \bar{f}_{-2}) \wedge (f_{-1} \vee f_{-2} \vee f_{-4}) \wedge (f_{-1} \vee f_{-2} \vee f_{-3}) \\
\mathrm{EC}_{-5}(f) &= (\bar{f}_{-3} \vee \bar{f}_{-4}) \wedge (\bar{f}_{-1} \vee f_{-2}) \wedge (\bar{f}_{-2} \vee f_{-1}) \wedge (\bar{f}_{-1} \vee \bar{f}_{-3} \vee \bar{f}_{-5}) \\
&\quad \wedge (f_{-2} \vee f_{-3} \vee f_{-4}) \\
\mathrm{EC}_{-6}(f) &= (f_{-1} \vee f_{-2} \vee \bar{f}_{-4}) \wedge (f_{-2} \vee \bar{f}_{-3} \vee \bar{f}_{-4}) \wedge (\bar{f}_{-1} \vee \bar{f}_{-2} \vee \bar{f}_{-4} \vee \bar{f}_{-5}) \\
&\quad \wedge (f_{-1} \vee f_{-3} \vee f_{-4} \vee f_{-5}) \wedge (\bar{f}_{-1} \vee \bar{f}_{-2} \vee \bar{f}_{-3} \vee f_{-4} \vee f_{-5}) \quad .
\end{aligned} \tag{3.12}$$

The range of error is expected to be $-2.5f_{-8} < Err < 3.5f_{-8}$ for 8-bit integers comparing to double precision $\log_2$ values. Since the meaningful precision is limited to 8-bit fixed-point for the fraction, it is also subjected to additional bit of error from rounding for N with greater bit resolution. The analysis also shows that the precision-circuit optimized logic equations reduced the range of approximation error to [-0.0096, 0.0128] with an average error 11.8 times smaller than Mitchell's for 8-bit integers.

Given eight-bit fixed point fraction, $f_{-1}$ to $f_{-8}$, it is obvious that appending any fraction beyond an eighth bit contributes very little to improve the precision with dominant source of error in higher significant bits. Hence, the $\log_2$ converter can be simplified further to reduce hardware components. Instead of the full one-to-one mapping from input to the output, one only needs to construct the data paths relevant to eight output nodes, whether it be 8, 16 or 32-bit resolution. The same concept applies to inverse-$\log_2$ converter. In summary, we replace the portion of piecewise lines with three-bit coefficients to improve the precision with eight bits fraction while in logarithmic scale:

$$\log_2 N' = k^* + f_{8MSB} + EC \tag{3.13a}$$

$$\log_2^{-1} L' = 2^{k^*}\left(1 + f_{8MSB} - EC'\right), \tag{3.13b}$$

where k* is specific to the input of either integer or fixed-point decimal(2's complemented k). In next chapter, we discuss the theoretical model for video stabilization and illustrate how redundancy and log-domain computation help reduce complexity of the design.

# CHAPTER 4

# THEORETICAL MODEL

Improvement of visibility, evaluation of reliable features and estimation of motion parameters are inseparable integral of the effort to formulate the model for stabilization of video sequence captured under non-uniform lighting conditions. The theory underneath these three subjects are discussed to the fullest extend in this chapter. At the end of each topic, the relevant part of the model should be simplified to the point that is reasonably achievable for realization of such hardware architecture.

## 4.1 Low Complexity Video Enhancement

The main objective of improving the quality of visibility includes enhancing the contrast and luminance components of the image for a more uniform appearance of the scene. Ideally, noise reduction should be part of the effort to correct noise induced from capturing devices. However, we do not deal with this issue as the magnitude of noise source is acceptable. In this section, we discuss and evaluate a more effective approach as an alternative to the methods introduced in chapter 2.1 to significantly reduce hardware requirement while achieving similar fidelity in the enhanced image/video. The new architecture should be capable of improving the brightness and contrast simultaneously to minimize shadow regions of the image. It processes the images and streaming video in HSV-domain with the homomorphic filter (Homomorphic model is a developed concept in computer science field mostly for grayscale image processing and cannot be applied directly for color images) and converts the results back to RGB representation with fast conversion factor [55] instead of full transformation [56]. The following sections discuss on the reflectance-illuminance enhancement model and the simplification for boosting the performance.

### 4.1.1 Homomorphic Processing in HSV-Domain

Color distortion correction can be avoided for color image enhancement in HSV-domain where the color (H), intensity (V) and saturation/color density (S) components can be rendered separately without introducing the distortion. HSV is a conical representation of the color as opposed to cubical representation in RGB space. To remove the shadows in the image, only the V component in HSV needs to be enhanced instead of boosting separate RGB channels which results with loss of color consistency without correction. Extraction of the V component is defined by

$$V(x,y) = \max(R(x,y), G(x,y), B(x,y)) \qquad (4.1.1)$$

where the R, G, and B are the original color components of the input image. The V-component is enhanced by a homomorphic filter defined as

$$V_{enh}(x,y) = \exp\left(\ln\left(\frac{V(x,y)}{2^P}\right) * h(x,y)\right) \times D \qquad \text{or} \qquad (4.1.2a)$$

$$V_{enh}(x,y) = 2^{\left(\log_2\left(\frac{V(x,y)}{2^P}\right) * h(x,y)\right)} \times D \qquad (4.1.2b)$$

for logarithmic based two expression where the * denotes convolution operation, $h(x, y)$ is the time-domain filter coefficients from a high-boosting transfer function, $P$ is the resolution of the pixels, $D$ is the de-normalizing factor, and $V_{enh}(x, y)$ is enhanced intensity value of the image. This enhancement model assumes that the detail (reflectance components) in the image is logarithmically separable [13], [3], [55]. Hence the model belongs to reflectance-illuminance category. The convolution or digital filter operation can be defined by

$$V_{enhl}(x,y) = \sum_{m=-a}^{a} \sum_{n=-a}^{a} V_{nl}(x-m, y-n) \times h(m,n), \tag{4.1.3}$$

where $a = (K-1)/2$ for $K \times K$ filter kernel, $V_{nl}$ is the normalized logarithmic scaled version of $V(x, y)$ and $V_{enhl}$ is the result from performing 2D convolution. The quadrant symmetry property of the homomorphic filter operation defined in (4.1.2a) and (4.1.2b) allows us to optimize (4.1.3) to reduce the number of multiplications approximately by 75% as summarized in section 3.1. The folded version of (4.1.3) can be expressed as

$$V_{enhl}(x,y) = \sum_{m=0}^{\frac{K}{2}-1} \sum_{n=0}^{\frac{K}{2}-1} V_{nl}\left(x \pm m + \frac{K}{2}, y \pm n + \frac{K}{2}\right) \times h(m,n) + V_{nl}(x,y) \times h\left(\frac{K}{2}, \frac{K}{2}\right) \tag{4.1.4a}$$

$$\begin{aligned} V_{enhl}(x,y) = \sum_{m=0}^{\frac{K-1}{2}} \sum_{n=0}^{\frac{K-1}{2}} \Bigg[ & V_{nl}\left(x+m-\frac{K}{2}+1, y+n-\frac{K}{2}+1\right) + V_{nl}\left(x-m+\frac{K}{2}, y+n-\frac{K}{2}+1\right) \\ & + V_{nl}\left(x+m-\frac{K}{2}+1, y-n+\frac{K}{2}\right) + V_{nl}\left(x-m+\frac{K}{2}, y-n+\frac{K}{2}\right)\Bigg] \times h(m,n) \end{aligned} \tag{4.1.4b}$$

for odd and even size kernels respectively. The enhanced image can now be transformed back to RGB representation by mapping the following set according to the value of $i$:

$$\{R'G'B'\}_n = \{\{e,p,t\},\{n,e,t\},\{t,e,p\},\{t,n,e\},\{p,t,e\},\{e,t,n\}\} \text{ for } i \text{ in } \{\{0\},\ldots\{5\}\}, \tag{4.1.5}$$

where $t = 1 - S$, n $= 1 - S \times f$, $p = 1 - S \times (1 - f)$, $e = 1$, and $\{R'G'B'\}_n$ is the normalized enhanced RGB components. The $i$ and $f$ are the integer and fraction portions of H which is the angular representation of color component in HSV-domain defined by (4.1.6). The S component in HSV domain is defined to be (4.1.7). The final output, $\{R'G'B'\}$, can be calculated as (4.1.8) with the denominator approximately equal to one for non-uniform scenes or images which contain bright parts, where $V_{enh} = 2^{V_{enhl}} \times D$. Equations (4.1.1)-(4.1.8) provide basic framework for the design of HSV-domain enhancement system.

$$H = \begin{cases} 0 + (G - B)/(V - \min(RGB)), & \text{if } V = R \\ 2 + (B - R)/(V - \min(RGB)), & \text{if } V = G \\ 4 + (R - G)/(V - \min(RGB)), & \text{if } V = B \end{cases} \tag{4.1.6}$$

$$S = \frac{V - \min(RGB)}{V} \tag{4.1.7}$$

$$\{R'G'B'\} = \frac{\{R'G'B'\}_n \times V_{enh}}{\max(\{R'G'B'\}_n)} \tag{4.1.8}$$

## 4.1.2 HSV-Domain Enhancement with Fast Color Space Conversion

We have shown the concept of enhancing color images in HSV-domain in a previous section. It reduces the processing bandwidth needed in hardware design to focus on one channel (V-component) rather than concurrently processing on all RGB channels followed by color distortion correction. This approximately cuts the hardware resource by 2/3 compared to the machine learning approach discussed in [19]. As Li Tao *et al* demonstrated in the color image enhancement algorithms [21], [55], the color restoration process can be further simplified. She showed that if H and S components in HSV space remain constant, the equations (4.1.5)-(4.1.8) needed for inverse transformation can be replaced by (4.1.11). This approach should moreover reduce the hardware complexity since full implementation of the transformations between HSV and RGB representations is not mandatory.

$$\{R'G'B'\} = \frac{\{RGB\}}{V} \times V_{enh} \tag{4.1.11}$$

### 4.1.3 Comparison of Visual Quality with Relevant Algorithms

The results from algorithmic simulation are shown in Fig 3 for visual judgment. The original image is illustrated in Fig 3a. This type of non-uniform image is typically the consequence of saturating the bright parts of the scene (Low lighting condition intensifies the effect in this case). We enhanced the image with the algorithm discussed in the last section. The more uniform result is shown in Fig 3b. It is trivial that most shadow regions with reasonable darkness (e.g. not completely dark) are removed while the bright parts maintain the fidelity. It should be noted that the discoloring in the dark regions of the enhanced image is natural since the color information is very weak with V component close to the tip of the HSV cone shown in Fig 3c.

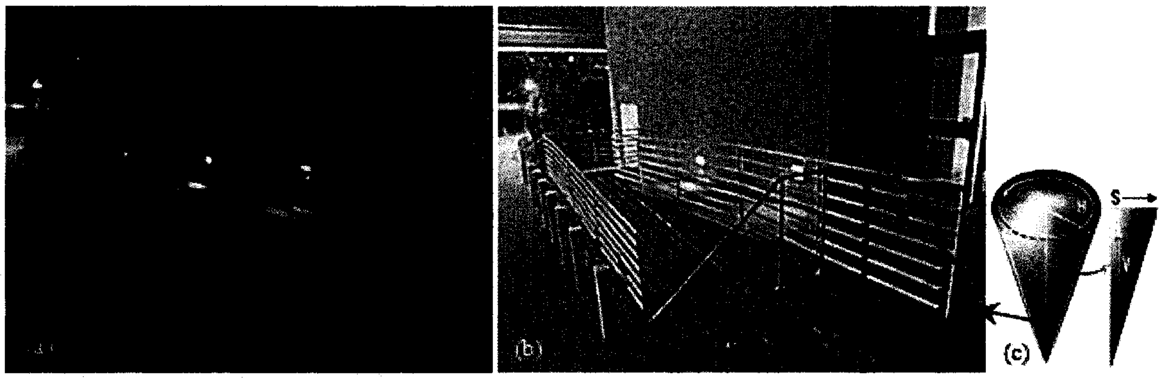


Figure 3: Algorithm simulation: (a) original, (b) enhanced (c) conical representation of HSV color space. No useful color information can be obtained with V component too close to the tip of the cone. Hence the excessively dark regions appear pale in the enhanced image.

While discussion of the other enhancement algorithms is outside the scope of this research, it is important to illustrate the results since we will compare the hardware utilization and the performance for the available implementation of the algorithms. The original test image is shown in Fig 4a. After enhancing the image on separate RGB channels, more details are revealed as shown in Fig 4b; however, the image appears pale due to loss of the color relationship between the channels. The result of enhancement by Multi-Scale Retinex with Color Restoration (MSRCR) [15], which is based on human perception, is illustrated in Fig 4c. This approach corrects the color distortion but still appears grayish in certain areas depending on the background color and lighting condition. In this case the background has a mild influence on the image. Thus further improvement can be made. The hardware implementation of this algorithm can be done, but the large scale kernel of the filters makes it impractical to achieve in time domain. Shown in Fig 4d is the output of the Luminance Dependent Nonlinear Image Enhancement (LDNE) algorithm presented in [21] which we implemented the hardware system in [20]. It is clear that the color is consistent which is obvious on the color of the hair of the man shown in the figure. Fig 4e is the output of correcting Fig 4b with Ratio Rule which is a machine learning algorithm [5], [6]. We also implemented it in [19]. The output for the design to be implemented is illustrated in Fig 4f. It has similar characteristics with Fig 4d and 3e and is somewhere between the two. With carefully chosen homomorphic transfer function, it can be hard to distinguish by human eyes. Nonetheless, the difference between these designs in terms of the performance and hardware utilization is quite dramatic. Design of this simplified system architecture is discussed in next chapter where we show the architectural realization of the equations (4.1.1), (4.1.2b), and (4.1.11) in the color image enhancement system.

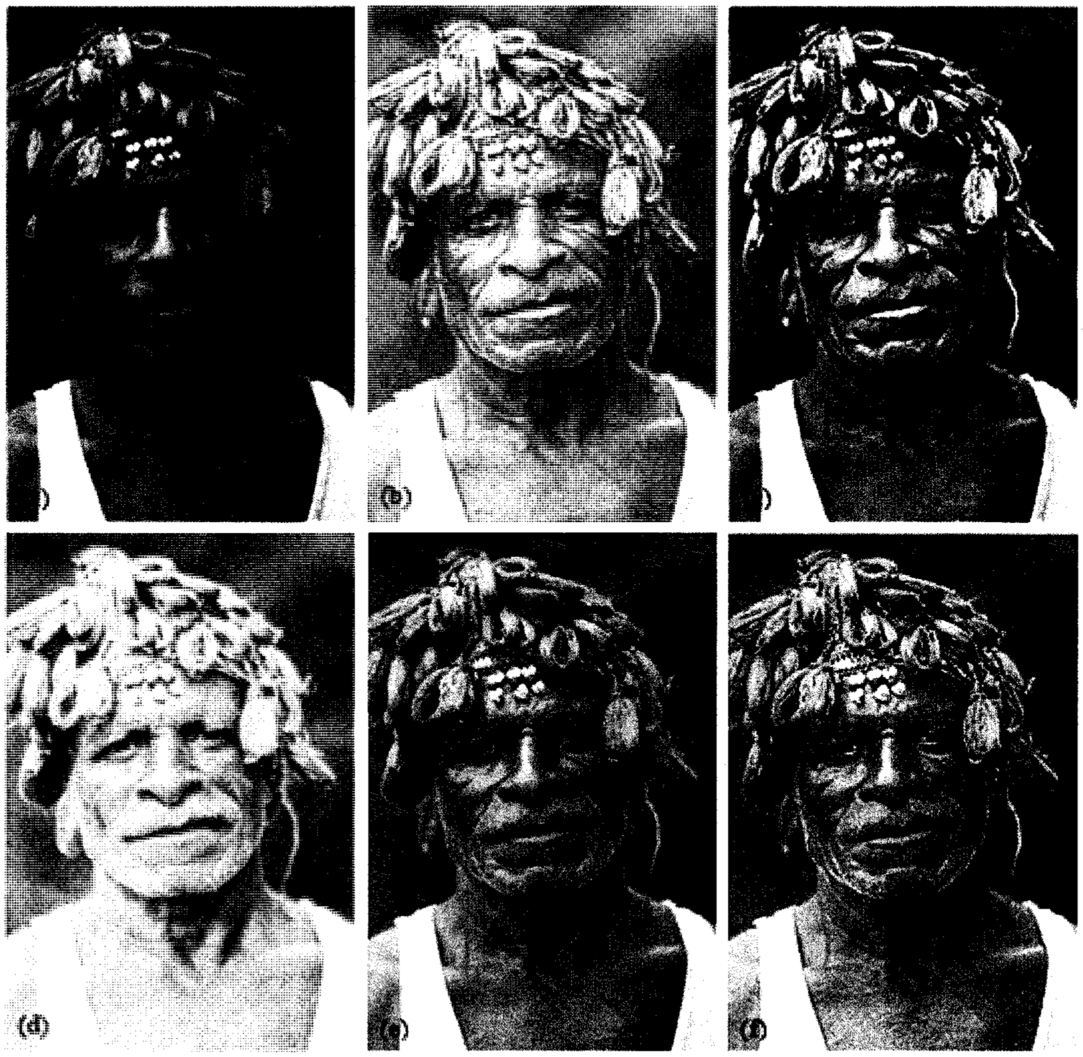


Figure 4: Algorithm Comparison: (a) original image taken from [15], (b) enhanced Separate RGB channels without color correction, (c) enhanced by MSR with color correction [15], (d) enhanced with LDNE [20] [21], (e) enhanced with RR [5] [6] [19], (f) enhanced with the approach we proposed.

## 4.2 Feature Selection and Tracking

The basic concept of feature evaluation framework is described in this section. The overview of the structure, the formulation of different components needed to evaluate texturally optimal features, and the preliminary simulation results are discussed in details to reveal the drawback of the framework. Simplifications are provided to reduce the complexity and make the calculations feasible for hardware realization.

### 4.2.1 Overview of the Framework

The new framework comprises mainly three functional levels. An overview of the network is illustrated in Fig 5 for image data represented by red, green blue color bands. The RGB color components are connected to level 1, $l_1$, of the neural network where $\beta$ layers of textures are extracted based on desired criteria such as edges, lines, and corners [57] in feed-forward configuration [23]. The $\beta$ layers of textures are then weighted through distance dependent modular network [58] and merged into single layer feature at $l_2$. The feature selection of this fused texture layer is considered in the descending order from the most optimal regions. The regional feature measure is performed and extracted to evaluate the periodicity of the potential candidates. The data involved with the measure can be from $l_1$. If the feature is indeed unique at $l_3$, we say that the network converges with good feature to track by different aspects of the textures and its distinctness within its region. Otherwise the weight memory of $l_2$ is modified similarly to Kohonen's learning rule [59] to suppress the regions resembling to the disqualified candidates. In latter case, the network converges if it determines a good feature or that the weight memory of $l_2$ converges to zero which implies that there is no reliable or traceable feature for tracking.

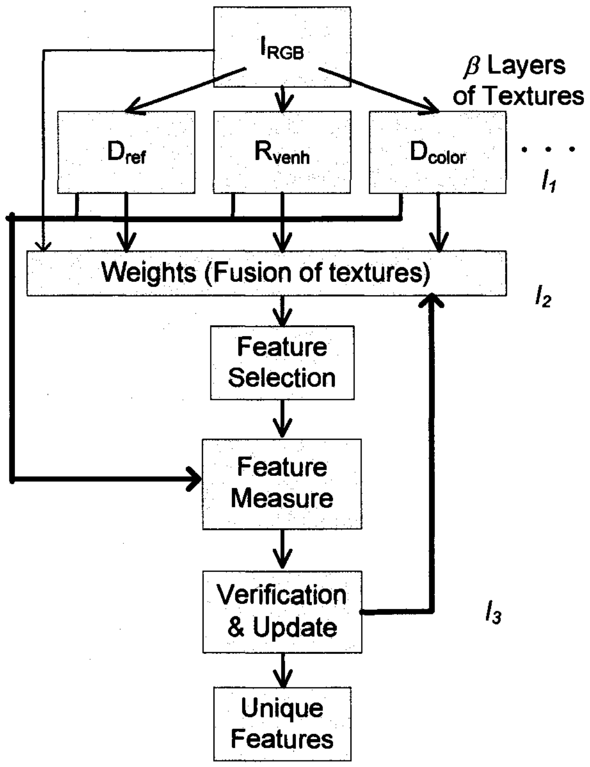


Figure 5: Block diagram of the framework for automatic feature selection for tracking.

### 4.2.2 Model Formulation

This section presents the calculation for different layers of textures and is a means to obtain the unique features. The basic process requires the extraction of different types of textures, the fusion of the textures, and the potential feature evaluation based on the uniqueness criteria.

#### 4.2.2.1 Extraction of $\beta$ Layers of Textures

The type of desired texture is strongly impacted by the nature of the problem. In image processing domain, edges, lines, and corners are the common textures. They can be considered as separate texture layers for the neurons defined by

$$C_i(x,y) = H_i\{I(x,y)\} \quad for\, 1 \le i \le \beta, \tag{4.2.1}$$

where $I(x,y)$ denotes the input data (the RGB color components in this case), $H_i$ is a form of transformation response to specific type of textures, and $C_i(x,y)$ is the $i^{\text{th}}$ texture layer or the activation values in $\beta$ layer extraction. We utilize the reflectance component [15] [55], ratio-relationship [5], and color variation to be the three distinct layers of textures in *a priori* feed-forward network at $l_l$ in Fig 5. One can often assume that the image is composed of the logarithmically separable reflectance (details) and illuminance (lighting sources) components under a reflectance-illuminance model. The model is especially helpful for image enhancement where these components can be enhanced for more uniform visual quality as demonstrated earlier. Hence, the variation of the reflectance component is illumination independent and can be a good source of texture which is defined by

$$C_1(x,y) = V_{ref}(x,y) * K_D(m,n), \tag{4.2.2}$$

where $V_{ref}(x,y)$ is the reflectance of the intensity of the color image, $K_D(m,n)$ is the derivative function, and $*$ is the filter operation. The exponentially separated details can be defined by

$$V_{ref}(x,y) = \exp\left(V_{nl}(x,y) * k_h(m,n)\right) \times D, \tag{4.2.3}$$

where $V_{nl}(x,y)$ is the normalized logarithmically scaled image intensity, $K_h(m,n)$ is the high-pass filter, and $D$ is the de-normalizing factor. This component can be obtained as the intermediate component from the part responsible for video enhancement. Another type of texture which is inspired by ratio learning algorithm can be useful by maximizing the neighborhood dependent ratio relationship. Instead of preserving the relationship between RGB components in fully connected network as discussed in [5] [60], we maximize the magnitude of the ratio among the neighbors within the intensity of the image defined by

$$C_2(x,y) = \frac{1}{MN}\sum_m \sum_n \frac{\max\left(V_{enh}(x,y), V_{enh}(x-m,y-n)\right)}{\min\left(V_{enh}(x,y), V_{enh}(x-m,y-n)\right)} \tag{4.2.4}$$

where $V_{enh}(x,y)$ is the enhanced intensity similar to (3.3.3) but boosted by filter $K_b(x,y)$, and $M \times N$ is the grid dimension of the inputs. The maximized ratio texture is rotation-invariant and considers the contribution of illuminance. The third layer of textures is dedicated for color variations within the RGB components and between the channels defined by

$$C_3(x,y) = D_{RGB}\left\{D_R(x,y), D_G(x,y), D_B(x,y)\right\}, \tag{4.2.5}$$

where $D_x$ denotes the derivative operator. $C_3(x,y)$ maximizes the regions where a sufficient variety of color information is available. The activation function of each neuron of $l_1$ is defined same as the activation values within 0 to 255 for 8-bit image but saturates outside the range. We simply refer to (4.2.1) to be the activation function.

#### 4.2.2.2 Weight Matrix for Fusing $\beta$ Layers of Textures

The structure of interconnects at $l_2$ is similar to modular architecture discussed in [58] as shown in Fig 6 with weights initialized to 1's. The weights of neighboring neurons, $w_i^{m,n}$, are connected to the central neuron through distance-based weighting described by (4.2.6) with $(m \times n) \times \beta$ neurons connected from $l_1$ layer. The resultant nodes of $\beta$ layers of textures are combined to produce fused data with the weights which emphasizes the global significance of each type of textures. We treat the activation function of this layer to be the activation values. The update model of the weight memory is to be discussed in $l_3$. This feature space is utilized for initial optimal feature selection.

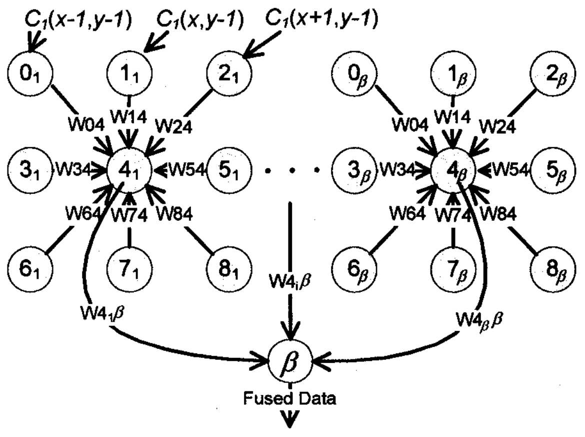


Figure 6: Architecture of interconnects for fusing $\beta$ layers of textures.

$$F_{fused}(x,y)=\left\{\sum_i w_i^{m,n}C_i(x,y)\right\} \tag{4.2.6}$$

### 4.2.2.3 Feature Selection

The initial coordinates of optimal features which maximize the textures in feature space can be described as

$$F_{jR}[.]=\arg\max_{jR}\left\{F_{fused}(x,y)\right\} for\, 1\le j\le J, \tag{4.2.7}$$

for $J$ features in the regions of interest where $F_{jR}[.]$ contains the coordinates of texturally maximal features. Local maxima in each region of interest become candidates in the descending order by magnitude subject to further examination for uniqueness. The initial features are defined by

$$F_j(m,n)=T_j\left\{I(F_{jR}[.])\right\}, \tag{4.2.8}$$

where $T_j$ defines the desired transformations of input data $I(F_{jR}[.])$ at the coordinates of the candidates and $F_j(m,n)$ is the $j$th $M\times N$ feature centered at $F_{jR}[.]$. The $F_{jR}(x,y)$ refers to $T_j$ transformed domain for feature $j$ at the region $R$. The result of $T_j$ may be a combination of the outputs from $l_1$. For simplicity, we defined the $T_j$ to contribute the intensity of the color image.

### 4.2.2.4 Feature Measure

The measure for uniqueness and potentially growing pattern (signatures) of the feature candidates in relation to the neighbors can be computed with normalized correlation defined by

$$\phi_{jR}(x,y)=\frac{\sum_{m,n}\left[\left[F_{jR}(x+m,y+n)-\bar{F}_{jR}(x,y)\right]\times\left[F_j(m,n)-\bar{F}_j\right]\right]}{\left[\sum_{m,n}\left[F_{jR}(x+m,y+n)-\bar{F}_{jR}(x,y)\right]^2\sum_{m,n}\left[F_j(m,n)-\bar{F}_j\right]^2\right]^{1/2}}, \tag{4.2.9}$$

where $\bar{F}_{jR}(x,y)$ and $\bar{F}_j$ are the expected values at $(x,y)$ under $(m,n)$ range for the region of interest and feature candidates, respectively, and $\phi_{jR}(x,y)$ is the similarity space. Lewis pointed out in [61] that despite a variety of template matching methods are available for feature measure, normalized cross correlation (NCC) remains the default choice. The covariance may be computed instead of NCC with the draw-back that the result is not normalized; hence the level of confidence may be questionable.

### 4.2.2.5 Verification and Update of Weight Memory of $l_2$

Useful information can be extracted from (4.2.9) to verify the uniqueness and analyze the potential signatures of the feature candidates with respect to its neighbors in $F_{jR}$. Suppose there exists a function defined by

$$\psi_{jR}(x,y)=\eta_{jR}(d)\phi_{jR}(x,y)\Big|_{\left(D_a^1\phi_{jR}=0,\,D_a^2\phi_{jR}<0,\,\left|D_{\perp a}^1\phi_{jR}\right|<\alpha\right)}, \tag{4.2.10}$$

where $\eta_{jR}(d)$ is the distance dependent weight function which emphasizes the importance of the most dominant candidates. Let us assume $\eta_{jR}(d)=1$, where the measure of the candidates appeared in $\psi_{jR}(x,y)$ is treated equally. The $D_a^1\phi_{jR}, D_a^2\phi_{jR}$ are the first and second order directional derivatives of (4.2.9) in $\vec{a}$. The $D_{\perp a}^1\phi_{jR}$ denotes the first order directional derivative orthogonal to $\vec{a}$ with the magnitude bounded by $\alpha$. This positive scalar defines the range for which the rate of the change along $\overrightarrow{a_\perp}$ is considered desirable for the signatures associated with the candidates. Graphical visualization of (4.2.10), which contributes to the growing pattern of the candidates with respect to the neighbors, is essentially a directed concave function of (4.2.9). The $D_{\perp a}^1\phi_{jR} = 0$ condition, subject to $D_{\perp a}^2\phi_{jR} < 0$, suffices the uniqueness test where the local maxima indicate the periodicity of the candidate. Eq (4.2.10) can be threshold to binary form defined by

$$\delta_{jR}(x,y) = \begin{cases} 1, \mathit{for}\ D^1\psi_{jR} = 0, D^2\psi_{jR} < 0, \psi_{jR} > T_{\psi_{jR}} \\ 0,\ \mathit{otherwise} \end{cases}, \tag{4.2.11}$$

where $\delta_{jR}=1$ for all the local maxima that satisfy threshold value $T_{\psi_{jR}}$ defined by the global maxima in the region of interest. The uniqueness can be determined by minimizing the distances between non-zero samples in (4.2.11) or by frequency of occurrence defined in (4.2.12). For a distinct feature, $\upsilon$ should be equal to one.

$$\upsilon = \sum\nolimits_{jR} \delta_{jR} \tag{4.2.12}$$

The update of the weight memory in $l_2$ is defined as

$$w_i^{m,n}(t+1) = \frac{1}{\upsilon} w_i^{m,n}(t) \times \left(1 - G(\delta_{jR}, \sigma)\right), \tag{4.2.13}$$

where $G(\bullet)$ is the Gaussian function characterized by (4.2.11), and standard deviation, $\sigma$, for neighborhood dependent iterative modification similar to [59]. The rate of the convergence of $w_i^{m,n}$ for specific feature is approximately inversely proportional to periodicity of the features.

#### 4.2.2.6 Convergence of the Network

The process is terminated if $\upsilon=1$ is found or that the weight memory converges to zero. The latter case implies that different measures of the textures resulted with either a completely periodic feature or empty set. In the rare worst case scenario, the network converges at approximately $k^{\text{th}}$ iteration where

$$\sum_k \upsilon_k = \mathbb{R}^{jR}, \tag{4.2.14}$$

and $\mathbb{R}^{jR}$ is the dimension of the region of interest. Usually, a couple of iterations are sufficient for the convergence.

### 4.2.3 Preliminary Simulation

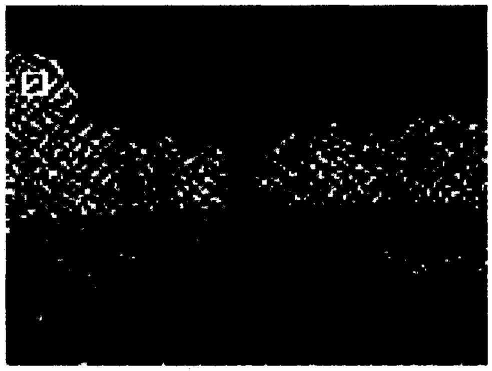
Preliminary results from software simulation for automatic feature selection and tracking are presented in this section to demonstrate how the periodicity of certain features affects the reliability of tracking.

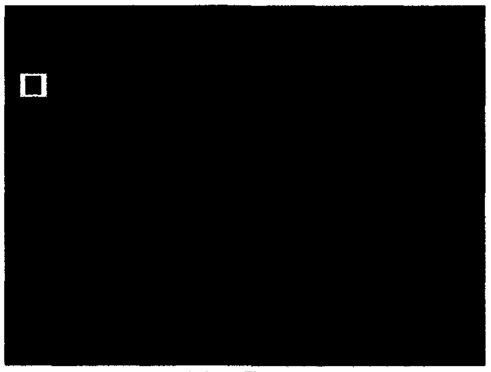
#### 4.2.3.1 Automatic Feature Selection

The input data and the relevant outputs of the architecture are illustrated in Fig 7(a)-(i). The image is fed into the network where $C_{1\text{-}3}(x,y)$ are computed with the results shown in Fig 7(b)-(d), respectively. The fused data after passing through the weight memory is illustrated in Fig 7e along with the dominant feature candidates in Fig 7f where the textures are strong enough. The result of feature measure for currently best candidate (marked with square box) at selected region of interest (We only demonstrate on one region for simplicity.) is illustrated in Fig 7g with unique local maxima shown in

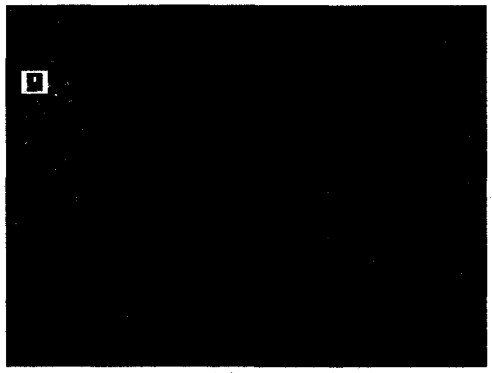
Fig 7h. In this case, the optimal feature is found in single iteration as boxed with the blue square in Fig 7i along with the optimal points by corner criteria [62] in pink dots. An example of the input with complete periodic textures is illustrated in Fig 8a with well distributed dominant feature candidates shown in Fig 8b. It is obvious with numerous local maxima plotted in Fig 8d that the potential candidates are not reliable for tracking unless additional distinguishable geometry is incorporated from (4.2.10) with $\left|D^{1}_{\perp a}\phi_{jR}\right| < \alpha$ condition where the signatures associated with the features exist.

(a): $I_{RGB}(x,y)$

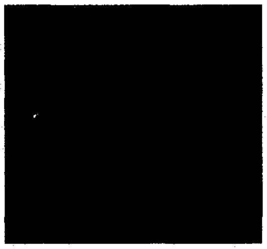

(b): $C_1(x,y)$

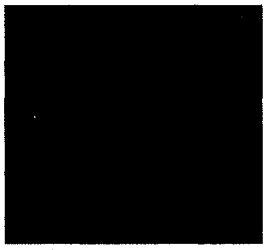

(c): $C_2(x,y)$

(d): $C_3(x,y)$


(e): $F_{fused}(x,y)$

(f): dominant feature candidates


(g): $\phi_{jR}(x,y)$


(h): $\delta_{jR}(x,y)$

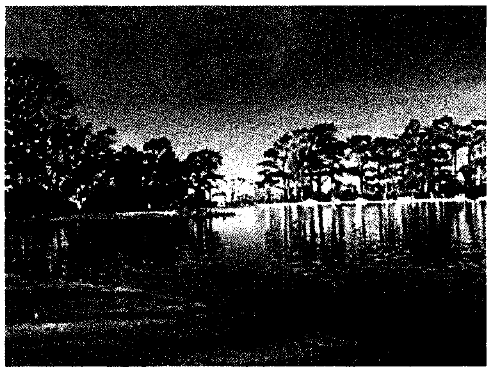
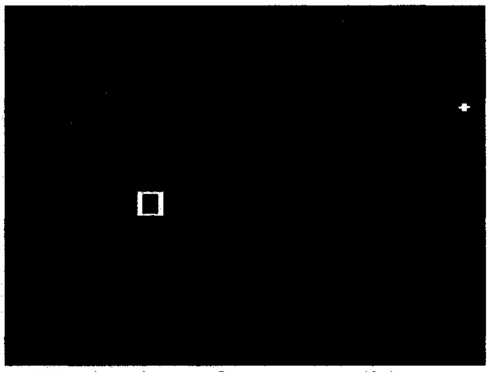
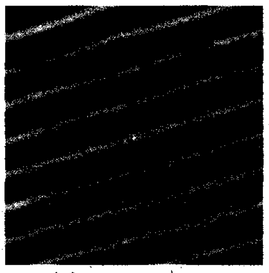

(i) Selected feature in blue box

Figure 7: Input color image and the outputs of the network at different stages are illustrated in (a)-(i). White dot in (f) shows initial dominate candidate with uniqueness test ($\upsilon$=1 according to (4.2.12)) shown in (h). Pink dots are important corner features evaluated by [62].

(a): $I_{RGB}(x,y)$ of periodic textures


(b): dominant feature candidates


(c): $\phi_{j_R}(x,y)$

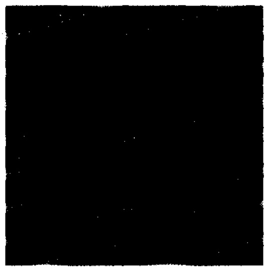

(d): $\delta_{j_R}(x,y)$

Figure 8: An example of input data with periodic textures where no feature is reliable for tracking confirmed by large $\upsilon$ computed from (d).

#### 4.2.3.2 Feature Tracking

The automatic feature selection is performed by the framework discussed in section 4.2.2 on frame 1 of a $360 \times 240$ video sequence. Once the optimal feature (optimal according to its uniqueness with respect to $\beta$ layers of textures) is selected, subsequent frames can be passed directly to the entry of 'Feature Measure' shown in Fig 5 with proper transformations, $T_j$, of the input data as discussed in section 4.2.2.3. Since the feature is unique at the time it is selected, the regional global maxima may be treated as new coordinate of the feature from previous frames while it remains relatively unique. Snapshots at frames 1, 56, 78, 102, 135, 161, 180, 220, and 237 are illustrated in Fig 9. After about 20 seconds, it slowly drifts away from the targeted feature because the network at $l_3$ does not compensate the temporal deformation with insufficient information represented by (4.2.11). The drift becomes obvious after frame 135 where the intended feature is severely rotated out of plane. We consider extending the framework in the future to accommodate the deformation utilizing the distinguishable geometry of the signatures related to the features represented by (4.2.10).

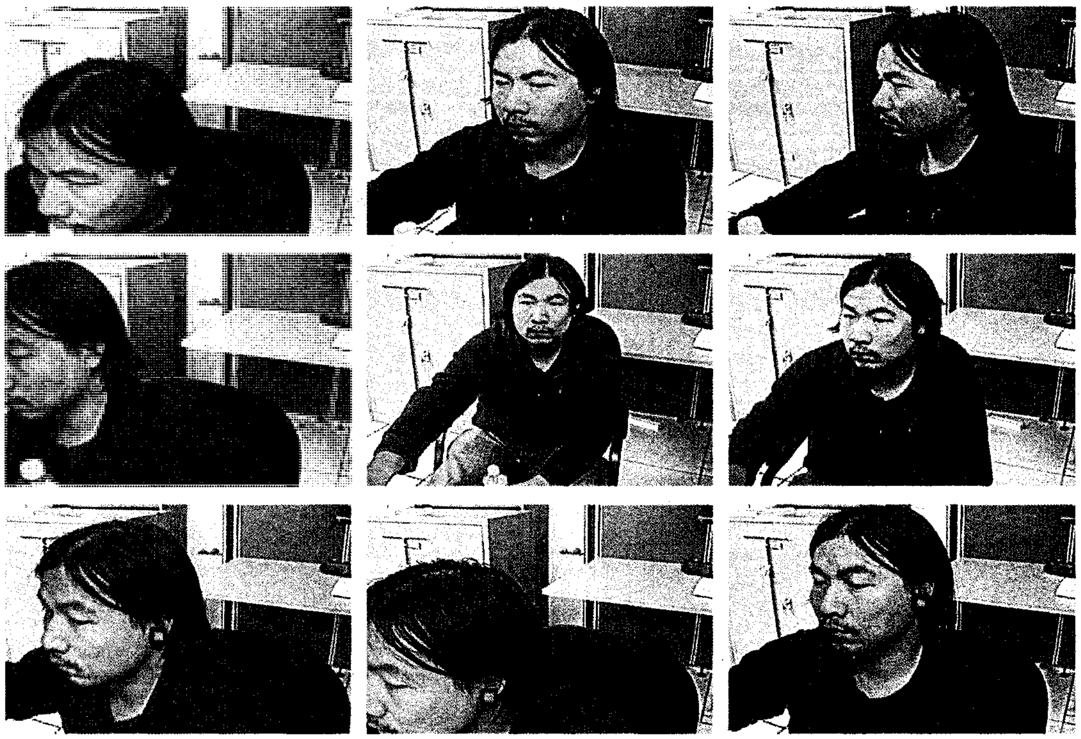


Figure 9: Results from feature tracking after automatic feature selection scheme. Shown from top-left to bottom-right are snapshots at frames 1, 56, 78, 102, 135, 161, 180, 220, and 237.

### 4.2.4 Potential Extension of the Framework

We presented a new framework of recurrent neural network for automatic feature selection by textures and uniqueness for tracking. Preliminary simulation showed that different types of textures could be extracted and fused, that feature measure played a distance-based learning rule for convergence of the network to unique and texturally maximized features. Feature tracking was also demonstrated by the network with a small tweak. The tracking results indicated that the framework is acceptable to in-plane rotation, scale change to certain extend. Research can be extended to make it more adaptive to out-of-plane temporal deformation. One may also fully explore the signatures associated with the neighbors of selected features to adapt the network to deformable circumstances and minimize the drift effect for more accurate feature tracking. Those signatures may also be used to estimate the numeric point spread function for motion deblur subject to further evaluation of the scene [63].

### 4.2.5 Simplification

The intension of selecting unique features overly constraints the problem for which one has to seek in descending fashion over the potential set of features. By the iterative nature of the framework, this implies that one has to buffer the search space for each region associated with the feature. Hence the bottleneck of non-constant memory access will compromise its performance. To not sacrifice our objective of designing a high performance system, we must remove the need for uniqueness of the features from the framework. This section serves to simplify the structure to the point feasible for hardware realization.

#### 4.2.5.1 Single Trivial Layer of Texture

Rather than fusing a set of texture layers to obtain more optimal features, only a single trivial layer is selected to reduce computation. The corner-ness criteria seem to suffice our need according to earlier work by Harris in [64]. Given a point in the image, the auto-correlation of V component with adjacent pixels is defined by

$$ac(x,y)=\sum_{\vartriangle x,\vartriangle y} W_{x,y}\left(V(x+\vartriangle x, y+\vartriangle y)-V(\vartriangle x,\vartriangle y)\right)^2 . \qquad (4.2.15)$$

With small $(\vartriangle x,\vartriangle y)$, the Taylor expansion of first order simplifies (4.2.15) to

$$\begin{aligned}ac(x,y) &= \sum_{\Delta x,\Delta y} W_{x,y}\left(V(x+\Delta x, y+\Delta y) - V(\Delta x,\Delta y)\right)^2, \\ & V(x+\Delta x, y+\Delta y) \approx V + [V_x V_y][\Delta x,\Delta y]^T \\ &= \sum_W \left(V_x V_y [\Delta x,\Delta y]^T\right)^2, \\ &= [\Delta x,\Delta y]\begin{bmatrix} \sum_W V_x^2 & \sum_W V_x V_y \\ \sum_W V_x V_y & \sum_W V_y^2 \end{bmatrix}[\Delta x,\Delta y]^T \\ &= [\Delta x,\Delta y]\begin{bmatrix} A & C \\ C & B \end{bmatrix}_M [\Delta x,\Delta y]^T \end{aligned} , \qquad (4.2.16)$$

where W(.) is the window function chosen to be summation of 3x3 neighbors to avoid multipliers, $V_x$ and $V_y$ are the first order derivatives, and M matrix encapsulates the shape structure characterizing the point. The corner-ness response is defined by

$$\begin{aligned} R &= Det(M) - k \times Tr(M)^2, \\ & Det(M) = A \times B - C^2 , \\ & Tr(M) = A + B \end{aligned} \qquad (4.2.17)$$

where Det(.) and Tr(.) are the determinant and trace of M, and k is the empirical constant. The best response with respect to particular region of the image is simply the maximum of R in the range.

#### 4.2.5.2 Sub-feature Representation

The feature measure of (4.2.9) with broader search ranges in the region of interest can be quite expensive for large feature templates. With a bigger set of features, the required computational power for searching and tracking becomes problematic. Equation (4.2.9) therefore does not scale well and can consume excessive amounts of resources in hardware implementation. To cut back the amount of calculations per feature template without compromising its performance, each feature is divided into sub-feature regions similar Stefano's [65].

***Sub-feature Detection:*** The concept of sub-feature measure in a nutshell is to evaluate the likelihood of the resemblance significant enough to provoke a full measure of the complete template. The two sub-feature based measure of [65] seems to hold promise of minimizing the number of calculations for exhaustive template matching; however, the search itself (even if the range of search remains regional) has its own setback in that the threshold of first sub-feature must be determined from the template in advance. The bigger drawback is not the threshold of first sub-feature; rather, it is the significance of that sub-feature. Supposed the first sub-feature is not sufficiently texturally specific or optimal, it may generate an overwhelming number of responses to initiate full measures; hence, it has the tendency of approaching the complexity in the context of full a search. To overcome this obstacle, we introduce a constellation to link between the primary sub-feature and a set of secondary sub-features. In addition to minimizing the number of calculations with smaller sub-feature space, the scores of secondary sub-features serves the purpose of confidentiality in supporting the primary sub-feature.

***Feature Representation by Constellation:*** To reduce the number of calculations with feasibility of such hardware realization in mind, the complete template is first decomposed into sub-features with the primary sub-feature containing the most complex texture. As shown in Fig 10a, the locality of remaining sub-feature set is determined by its sub-optimal textures and directly connected with primary sub-feature to form a star constellation. With such structure constructed, we may ignore the rest of that complete feature template. To search for the feature in an image with relatively static spatial locality around the region, the local maxima of primary sub-feature is first calculated in (4.2.11).

The secondary sub-features are evaluated if and only if the primary sub-feature and the already estimated secondary sub-features scored sufficiently high. In other words, the star constellation that represents the full feature template has a cascaded search sequence for which any failed score will terminate subsequent search on the maxima of primary sub-feature. In this manner, the exhaustive search in the region of interest is only needed on any primary sub-feature. The test for secondary sub-features is not really a search; it only exists to verify the spatial relationship of the constellation specific to the full feature remains legitimate. This concept can readily be applied to the tracking of rigid objects since the spatial deformation of such objects is also encapsulated in the constellation. In fact, the process of evaluating secondary sub-features generates the byproduct of attributes related to spatial distortion. Let us not be lost in this very promising model; the only piece of information necessary to solve part of our problem is really the coordinate of primary sub-feature. What that means is the precise locality of sub-feature set is not crucial. The score of (4.2.9) in image processing is often gradual for which the match around the maxima is relatively sub-optimal to conclude the existence of a particular sub-feature. Hence the representation is also tolerable to spatial deform to certain extend.

While such star constellation can be effortlessly constructed on desktop computers, the structure itself unfortunately poses the demand for buffering of full feature and bookkeeping of sub-features. To bypass such a tedious process on low level hardware implementation, a single path straight line constellation is selected with the primary sub-feature on the top, ignoring the sub-optimal-ness of the textures in the sub-feature set as shown in Fig 10b.

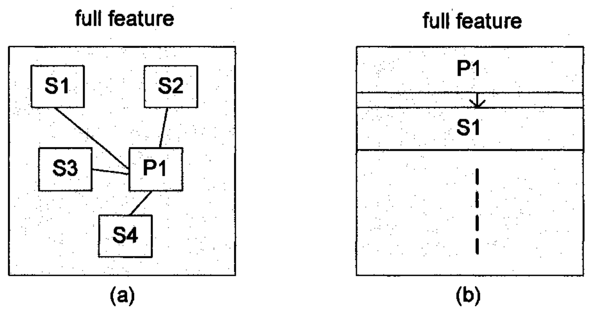


Figure 10: Representation of full feature by texturally optimal sub-features in a constellation. (a) A start constellation constructed to encapsulate the spatial relationship of sub-features. (b) A straight line structure to simplify hardware realization.

## 4.3 Estimation of Motion by Consistency of Motion Parameters

In this chapter, we propose ME by measuring the consistency of motion present in selected features. A minimum of two frame buffers is necessary to extract motion parameters. By reducing frame buffering, system delay is also minimized. GME calculates the camera or scene motion which can be modeled in 2D or 3D spatial coordinates. We concentrate on 2D which is accurate for 6-parameter camera motion: rotation, translation and scaling. To incorporate off-axes zooming and change of viewpoint, the 3D spatial model is far more precise. In [66], Huang used corner detection for GME to improve the edge detection based approach presented in [67]. The author calculates the motion parameter by detecting the corresponding cross-points which are iteratively grouped into inlier or outliers based on their velocities and residual error. If the resultant inlier group contributes to prediction error below a certain threshold, then parameters calculated within that inlier represents the global motion. The approach has a couple drawbacks. GME by Edge or cross-point is not very robust to motion blurness(or appeture problem) and sources of noise. It also depends on the quality of edge/corner detectors; however, the cornerness may be solved with a large ratio of eigenvalues in x-y directions which was the primary reason that we chose corner-ness for texture layer in section 4.2.5 [64]. Given corresponding feature points, an alternative mean of grouping inlier and outliers can be determined based on preserved properties of affinity through angle and distance relation to other features. The affined model to compensate translation before rotation and scaling matrix can be defined as:

$$G(\mathbf{x},\mathbf{M}) = \alpha \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}. \tag{4.3.1}$$

### 4.3.1 Estimation of Inlier Motion

Affine transformations distort the distance, angle, as well as area or volume; however, they preserve three important geometric properties. One is the collinearity for which the sample points laying on a line remain on the line after the transformation since translation, rotation, and scaling are affine subspaces. The second property is the parallelism where the lines parallel to each other remain parallel. Lastly, the ratios between the sample points on a line are preserved constant. Supposed that we have derived a set of feature points, $S_{t-1}$, uniformly distributed in frame $I_{t-1}$, and another set $S_t$ in $I_t$. By collinearity, each line between two points in $S_{t-1}$ maps to $S_t$ and can be grouped into an element in a set of $K$ motions:

$$\overset{i\to j}{S_{t-1}} = \mathbf{M}_k \overset{i\to j}{S_t}, i \neq j, i, j \in S, \tag{4.3.2}$$

for which we define the element in $K$ with most sample points to be the inlier motion. In this approach the outcome represents the global motion given sufficient uniformly distributed samples. For a limited set of points, the inlier does not guarantee finding of global motion.

Despite that the angle relationship is distorted by affinity, the consistency of the angles between vectors $\mathbf{S}_{t-1}$ and $\mathbf{S}_t$ remains coherent given two conditions:

1. At least one element in $K$ contains multiple vectors.
2. Each element satisfying 1 possesses constant angle between the samples within the element iff the locality among the samples remain relatively stationary.

The second condition implicates the existence of rigid regions or static background within the video. And the most dominant element in $K$ strongly correlates to global motion. The angular argument and the direction of rotation of (4.3.2) can be evaluated by (4.3.3). It is not necessary to explicitly calculate cosine term since the intension is to check the consistency and group the rotational motions. With 2D motion model, the sign of cross product also suffices the direction of rotation.

$$\cos(\theta) = \frac{\mathbf{S}_{t-1}\mathbf{S}_t^T}{|\mathbf{S}_{t-1}||\mathbf{S}_t|}, \quad dir = sign(\mathbf{S}_{t-1} \times \mathbf{S}_t) \tag{4.3.3}$$

The consistency in rotational motion provides good insight in estimating dominant rigid regions; however, the integrity of the constant ratio must be sustained since the error along the exact axis of the line is undetected. The rejection mechanism for each element in $K$ can be defined by (4.3.4) for which the samples outside the threshold are discarded from the group based on absolute difference from expected value. The $T_d$ constant is fixed to tolerate a source of error such as distortion from camera lens. Once the inlier is estimated, linear regression can be applied to calculate motion matrix. An alternative is to assign the angle found in (4.3.3) and the accumulated scale change as $\left(\prod_t E_k\right) / E_k(t_0)$ in (4.3.4) to rotation matrix and scalar value (assume uniform zooming) of (4.3.2), respectively.

$$R_k = \left| E_k\left[ |\mathbf{S}_{t-1,k}| \big/ |\mathbf{S}_{t,k}| \right] - |\mathbf{S}_{t-1,i}| \big/ |\mathbf{S}_{t,i}| \right| > T_d, \quad i \in k \in K \tag{4.3.4}$$

## 4.3.2 Estimation of Intended Motion

In addition to estimation of inlier motion in section 4.3.1, it is also necessary to evaluate the scene of interest for which the intended camera motion can be incorporated. Without compensating the intended motion, the scene is fixed as an absolute coordinate in space. Once the camera is deviated outside the range, there is nothing else but blank screen. Given the signed progressive angle $\theta$ of inlier calculated by (4.3.3), the accumulated rotation can be defined by (4.3.5) subject to the window of rotation defined in (4.3.6). The $T_\theta$ is a constant that separates the range of vibrative (unintended) camera rotation with respect to desired movement. Likewise, the window of translation is defined. The accumulated angle and translation are particularly useful for non-static camera setting.

$$\theta_{acc}(t) = \theta_{acc}(t-1) + \theta(t) \tag{4.3.5}$$

$$\theta_{acc}(t) = \begin{cases} \theta_{acc}(t), & if\, |\theta_{acc}(t)| < T_\theta \\ sign(\theta_{acc}(t))T_\theta, & otherwise \end{cases} \quad (4.3.6)$$

## 4.3.3 Simplification by Order of Search

While grouping of a set of k elements of motions in (4.3.2) implicates a full measure of all possible combination of nodes (selected features), one can avoid such fully connected topology by eliminating certain redundant connectivity to the nodes. For example, the affine attributes obtained from node a to b is the same as b to a. It is, therefore, pointless to group both motion of $\vec{ab}$ and $\vec{ba}$ under particular element in K. Neither are we interested in obtaining a complete collection of k motions. We propose a triangular search scheme to reduce the number of calculations to minimum. Although such a search scheme is not quite computationally intensive or time sensitive and is considered negligible comparing to the complexity of feature measures in section 4.2, this modification is essential for the design of simpler architecture. The concept of estimating inlier motion can easily be illustrated in Fig 11. Given a set of feature coordinates $F_{1...n}$ obtained from $\delta_{jR}$ of (4.2.11), one can calculate the angle with (4.3.3) between two vectors, $\mathbf{S}_{t-1}$ and $\mathbf{S}_t$, from two pairs of points/nodes. At the bottom of the triangle, the leftmost point-pair is used to calculate the angular relationship to other point-pairs in the list. If the majority of such angles are non-constant, we may assume the test point-pair belongs to any outliers and reject it from the list. In this fashion, the point-pairs of outliers are being progressively eliminated until the outstanding element dominates and terminates the search. Notice that there is absolutely no redundant calculations from the

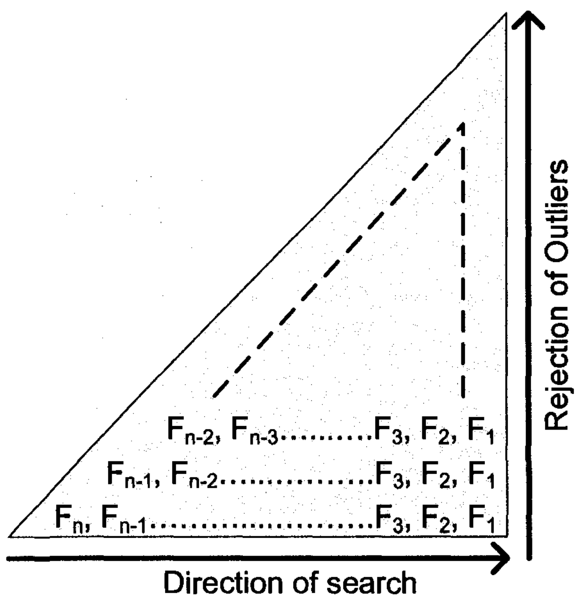

$F_{n-2}, F_{n-3}$..........$F_3, F_2, F_1$
$F_{n-1}, F_{n-2}$...............$F_3, F_2, F_1$
$F_n, F_{n-1}$......................$F_3, F_2, F_1$
Rejection of Outliers
Direction of search

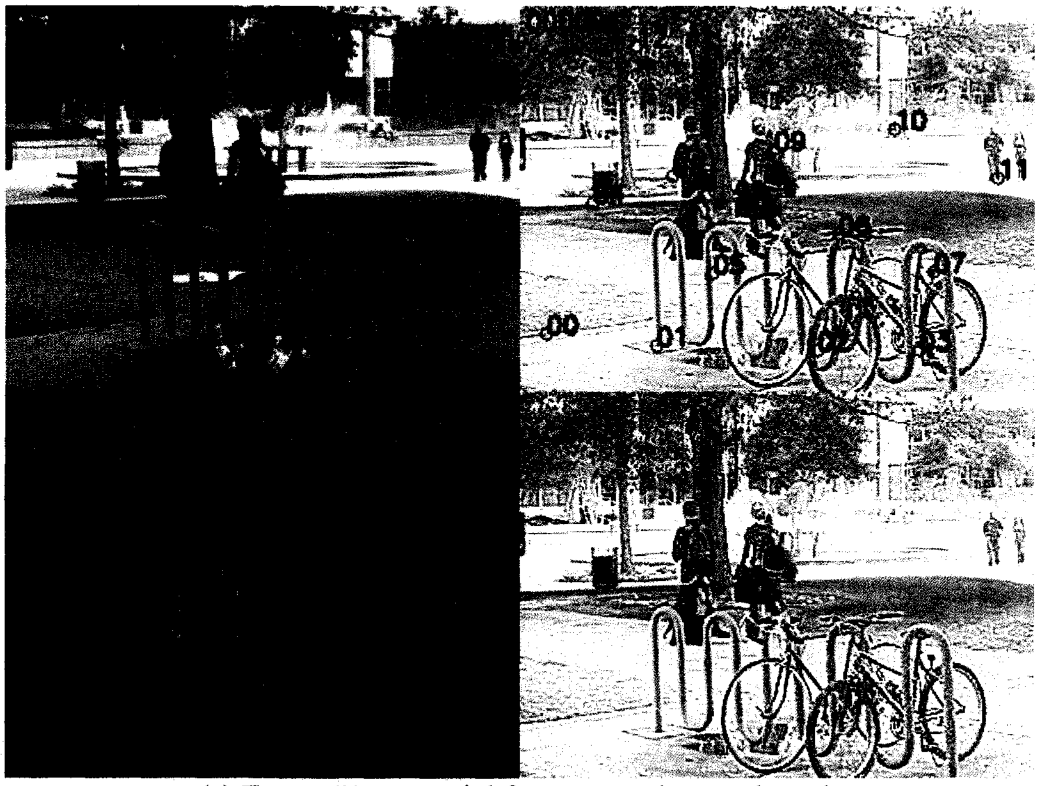
Figure 11: Triangular order of search to minimize the number of calculations and to identify the inlier and reject the outliers.

bottom level up within the triangle. And that the search terminates as soon as the outstanding element is discovered. The very first point in the point-pair within that outstanding element also has inherently the most stable spatial locality relating to other points within the element. In other words, if we were going to maintain such a list of known and reliable background features, the leftmost point on the dominant element naturally forms a star constellation by the same concept that we learned in section 4.2.5.2. That means we can readily use the most stable point(s) to calculate subsequent incoming video frames to better estimate any new stationary features as well as narrowing down the search range of (4.2.9) to a greater extent. For the sake of simplicity, we ignore such efforts and concentrate solely on sorting out the inlier motion parameters. Furthermore, the measure of constant ratio of (4.3.4) between new and reference frames is ignored since the order of search also inherently rejects non-constant distant ratios. We also found there is no need to implement the zooming factor of (4.3.1) and rejection mechanism of

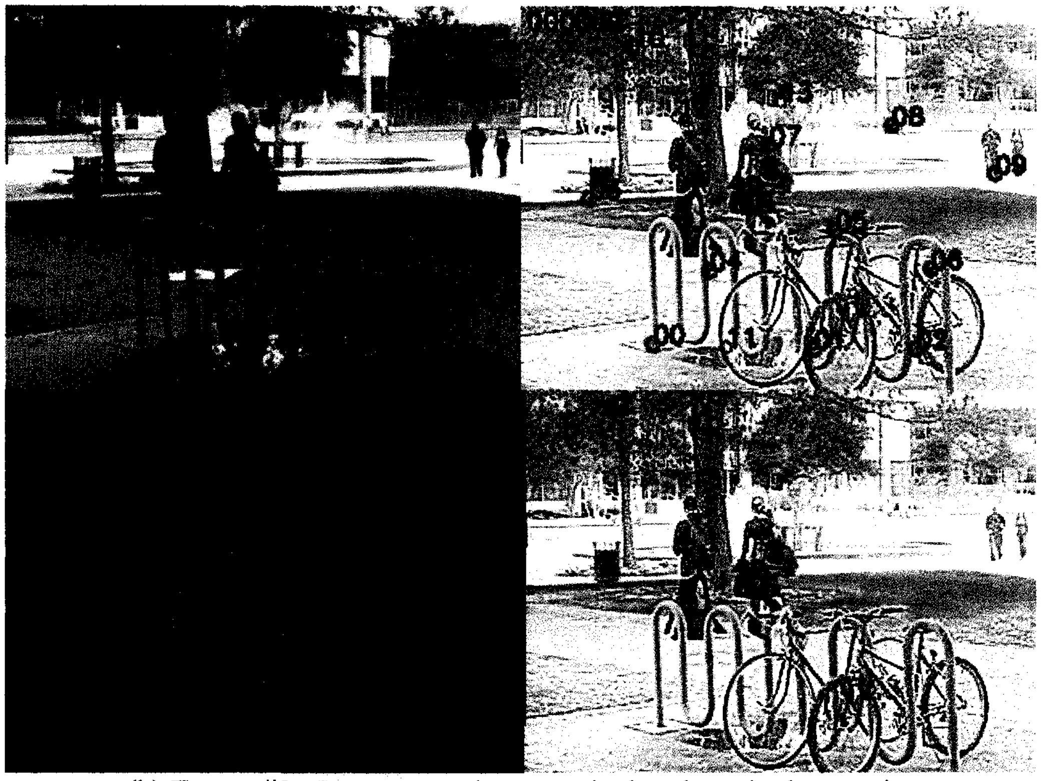
(4.3.4) at this point until further development of the algorithm. In the next section, the preliminary results of the algorithmic simulation of motion evaluation are combined with the material from sections 4.1 to 4.2 to illustrate the working prototype closely resembles what we would expect from hardware simulation.

### 4.3.4 Algorithmic Simulation of the Stabilization Prototype

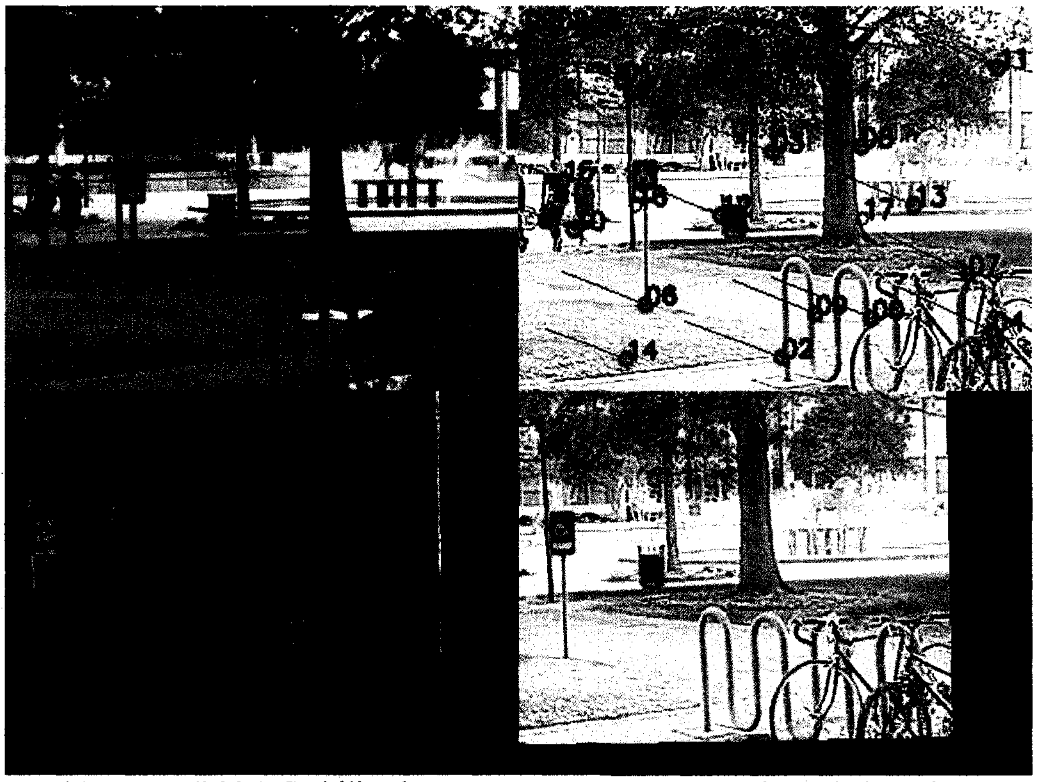
The snapshots from stabilization of scene with non-uniform lighting conditions are illustrated in this section to briefly demonstrate the outcome of the prototype. Fig 12a shows the very first frame of the video. On the top left corner is the original image. The feature selection and tracking results are circled in the top right frame. The bottom left frame illustrates the moving object other than the background motion (separated by several frames). It also reveals the influence of motion blurness and lens distortion on a wide-angle camera. The frame on the bottom right corner shows the result of a stabilized sequence. Notice that the first frame immediately selects texturally optimal potential features (thin blue circles) in different regions of the image. All of the features with static background motion have been detected in the second frame of Fig 12b (thick green circles). Also notice the initial feature on the left most person in the image is already rejected in Fig 12b as his motion is significant enough to deviate away from background motion; however, features #9 and #11 perceived by our eyes as moving objects are not immediately rejected due to the fact that both remain relatively stationary between the frames. In frame #236 of Fig 12c, the cameraman already shifted the scene to the left and is now detected as intended motion since the accumulated velocity/translation runs



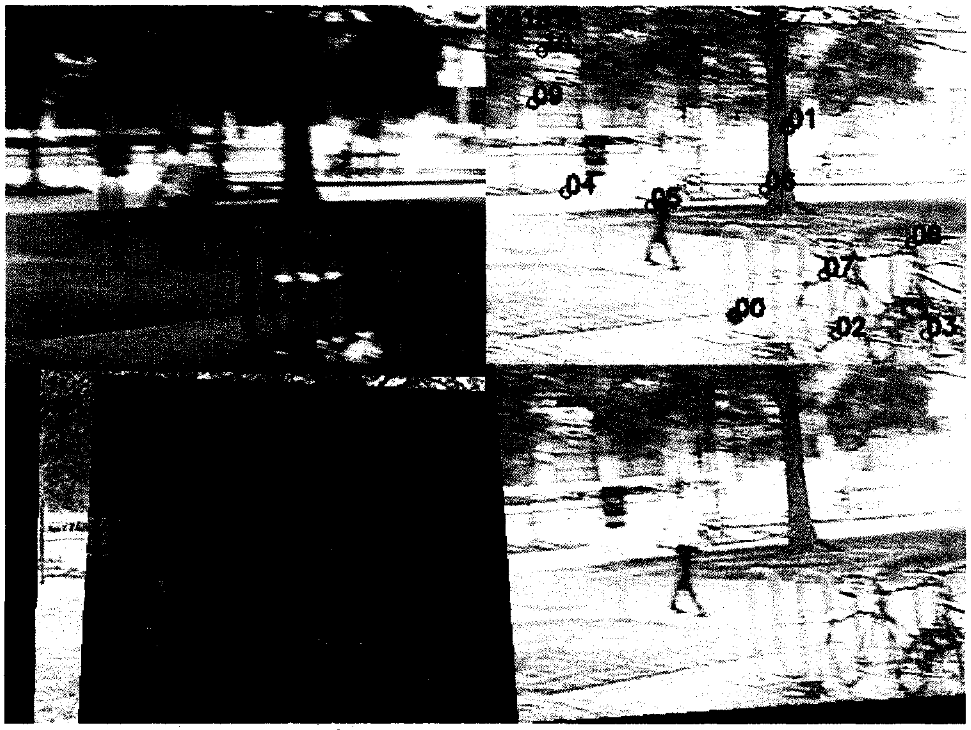
(a) Frame #1: Potential features are been selected.



(b) Frame #2: Features are been tracked and marked as static.



(c) Frame # 236: Stabilized sequence compensates the intended motion.



(d): Frame #1018: The scenario where feature measure fails to detect due to severe motion blurness.

Figure 12: Snapshots of enhanced video and stabilized sequence to show different stages of the event.

outside the window of monitor. The scene is therefore compensated to gradually transition as directed by the cameraman. The scenario where the feature measure fails to detect and track due to severe motion blurness is also illustrated in Fig 12d. The new potential features are selected but remained untraceable as well in frame #1019.

## 4.4 Summary

The subjects of complexity of commonly used operators and the formulation of a simple model necessary for stabilization of the video sequence were analyzed in this chapter to reduce complexity and establish a series of steps feasible to implement in hardware. Exploitation of redundancy properties inherent in the operators helped us to focus on the essential computational power and eliminate unnecessary calculations. The logarithmic domain computation further lowered the complexity of multiplication, division and exponentiation related operators for realization of multiplier-less architectures. The basic concept of video enhancement with fast color space conversion was also illustrated; however, it should be noted the homomorphic processing was a well established concept in literature. The basic model of feature selection and tracking was presented with in-depth analysis of its drawbacks. A more hardware-friendly solution was exploited to minimize the amount of calculations involved with feature evaluation and measure. The advancement was to select a known reliable texture to accompany the representation of the full feature into a constellation of sub-features. While we did not explore the more complete characteristics of the model, the framework posed the potential for future expansion to finer grains capable of analyzing certain spatial properties. In section 4.3, we further proposed a simple technique to evaluate the inlier motion by a triangular order of search. This method progressively rejected the outliers to discover the most outstanding element which is equivalent to the background motion. And finally the snapshots from algorithmic simulation were provided to illustrate certain steps along the stabilization of the sequence. The case in which the model failed to stabilize under extreme motion-blurness was also presented. In chapter 5, hardware realization of different components necessary for the calculations introduced in this chapter is discussed.

# CHAPTER 5

# DESIGN OF HIGH PERFORMANCE ARCHITECTURE

Various aspects of the design are introduced in this chapter which covers the overview of system architecture and the realization of different subsystems. The main portions of the chapter focus on the design of logarithmic modules with correction, the video enhancement module, the feature selection, the feature measure and correspondence management, the motion evaluation and the affine transformation. Further modifications are introduced as necessary to simplify the architectures.

## 5.1 Overview

The overall block diagram of the system architecture is shown in Fig 13. The “Stream Line Buffers” consists of eight line buffers to support buffering of nine video lines streamed in from “Stream Vin”. This component creates the internal parallel bus for concurrent processing of other core engines. The data on the parallel bus are simultaneously fed into the blocks of “Single-layer Feature Selection”, “Partial NCC” and “Video Enh” as well as the storage blocks. While these blocks operate in parallel paths, their coordinate system is just slightly off each other due to the difference in pipeline latencies. The “Frame Coord. States” block serves to generate the coordinate states suitable to other blocks. It is basically composed of counters and some registered adders to accommodate the offsets. The “Video Enh” engine basically enhances the RGB components of incoming stream and sends the result into one of the two pipelined storage RAMs for full video frame buffering through the “Frame Switch” block. The switch block consists of multiplexers to alternate the write/read paths between the frames. The core engine for feature selection evaluates texturally optimal features by corner-ness criteria and saves the feature into its storage space. At the same time, the partial normal correlation is being computed. Under normal scan mode, the "Partial NCC" block only evaluates on primary sub-features. It gives higher priority only if the coordinate of secondary sub-features coming into the testing range. Similar the "Potential Feature Storage", the partial NCC storage takes snapshots of the sub-features with each pass of better correlation score. Once the entire feature set has been evaluated, the "Motion Evaluation" block takes over the list of feature coordinates and begins the process of inlier estimation in a stack fashion which mimics the triangular order of search. With the obtained global motion parameters, the "Affine Transform" block generates the memory address of stabilized video sequence and grabs the data for display.

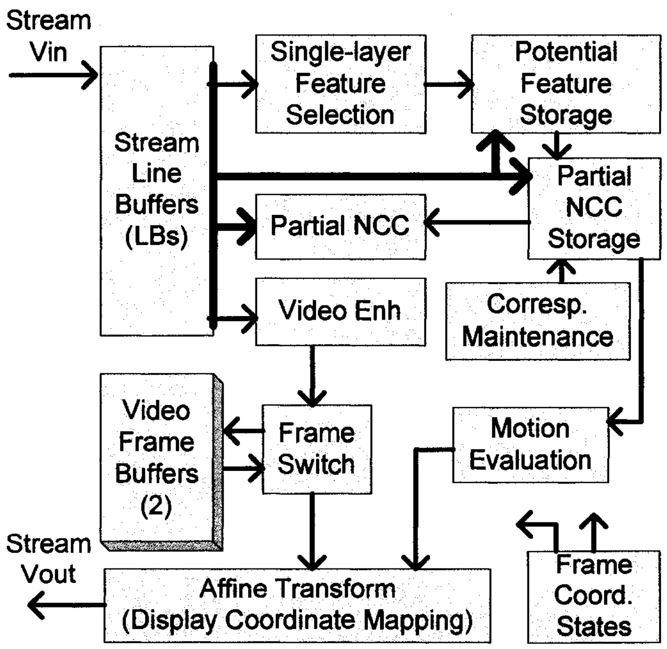


Figure 13: Block diagram of the system architecture.

## 5.2 Logarithmic Domain Computation

This section covers the design of logarithmic modules. It mainly illustrates the realization of approximation modules and the placement of error correcting coefficients on a functional level; however, more efficient circuitry at the transistor level is currently not available.

### 5.2.1 Architecture of $\text{Log}_2$ Module

The $\log_2$ architecture consists mainly of the $N$-bit standard priority encoder and a modified barrel shifter (MBS). The general architectural design for $\log_2$ is shown in Fig 14. The priority encoder provides the index output based on the logic '1' of the highest bit in the input value. As indicated in Fig 14a where $N$ equals 16, the input of priority encoder is capable of encoding any 16-bit real number. If the input value is strictly a positive integer, the index output maps directly to the integer portion of $\log_2$ scale, binary 0000 to 1111 in this example. The infinity is bounded to index 0 as it is the logical function of priority encoder, and there is no need of defining $\log_2(0) = -\infty$ for our application. If the input value has both integer and fractional parts, the MSB of the index on the output of priority encoder is inverted to determine the actual integer part of $\log_2$ scale. For example, the index value is now mapped to [7, -8] instead of [15, 0] integer input value. Index 0 now corresponds to -8 in 2's complement. For the same reason, $\log(0) = -\infty$ is bounded to -8.

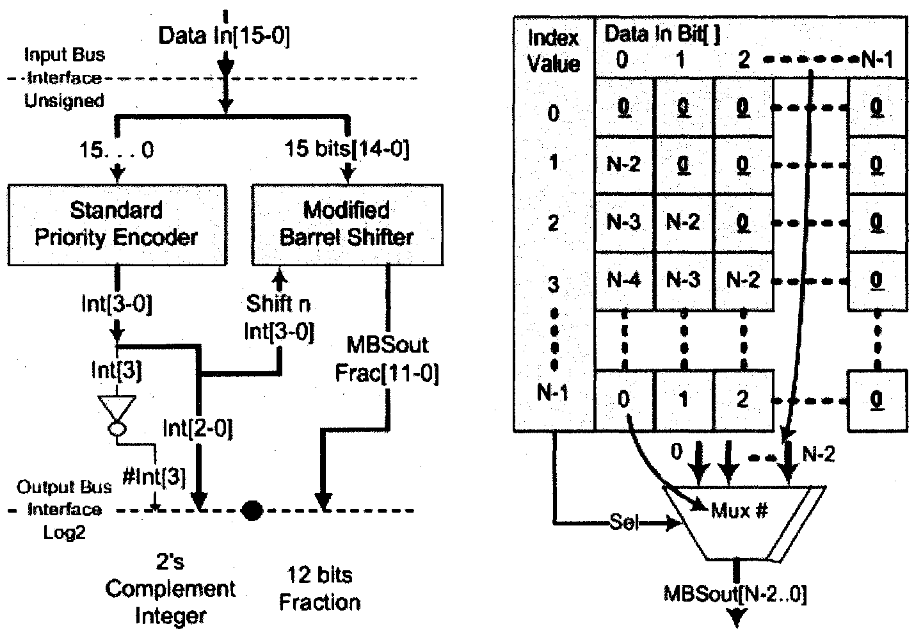


Figure 14: (a) Architecture of $\log_2$, (b) Mapping of multiplexers in MBS.

The fractional bits are extracted with a modified barrel shifter. It is composed of *N*-1 *N*-to-1 multiplexers at the most where *N* is the number of bits to be shifted according to the given index. The logical functional view for mapping the set of multiplexers is that given the index, it always shifts the bit stream at the index position to be the first bit at its output. In standard barrel shifter, the output can be linearly or circularly shifted by *n* positions from index 0 or *N*-1; however, the modified barrel shifters in both $\log_2$ and inverse-$\log_2$ exhibit the reverse mapping. The mapping of *N*-1 multiplexers is indicated in Fig 14b. The index value along the vertical axis represents the index that specifies *n* shifts. It is directly connected to the select lines of multiplexers. So for the binary combination of *n* shifts, the corresponding input *n* is enabled. The outputs of multiplexers are one-to-one mapping to the *N*-1 bit output bus. The index on the horizontal axis represents the bit value of the input at the corresponding bit location. The values within the horizontal and vertical grid specify the multiplexer numbers where the corresponding bit values of the input are mapped to. For example, with the index value of three, bit values at locations 0 to $N$-1 of the input are mapped to the third set of inputs of the multiplexer numbered $N$-4 to 0. The third set of inputs (marked as '$\underline{\mathbf{0}}$' in the grid) of the multiplexers outside the mapping bit range of the input is padded with zeros for simplicity. The net number of inputs of the multiplexers can be reduced by half when the architecture of MBS is optimized, eliminating the zero-padded inputs. The fraction on the output of MBS occupies $N$-$\log_2(N)$ bits with the fixed point $\log_2(N)$ bits down from the MSB. Note that the whole fraction up to $N$-1 bits can be preserved as needed.

The maximum propagation delay of the $\log_2$ architecture is computed based on the critical path of the combinational network in priority encoder and modified barrel shifter where the modified barrel shifter depends on the index from priority encoder to perform $n$ shifts. Note that the arrangement of multiplexers is completely in parallel such that the overall latency comprises a single multiplexer. The depth of propagation delay is significantly less compared to non-pipelined conventional multipliers. It implies the architecture can provide very high speed operations.

### 5.2.2 Architecture of iLog$_2$ Module

Structural mapping of inverse-$\log_2$ is the reverse of $\log_2$, as illustrated in Fig 15b. The inverse-$\log_2$ architecture is simpler than $\log_2$ architecture since it is not necessary to have the decoder to undo the priority encoding where the integer serves as $n$ shifts to the reverse of the modified barrel shifter (RMBS). The inverter is not needed for the inverse-$\log_2$ architecture shown in Fig 15a if the inputs are unsigned positive numbers. Note that negative values of $\log_2$ scale indicate the inverse-$\log_2$ result in linear scale should be a fraction.

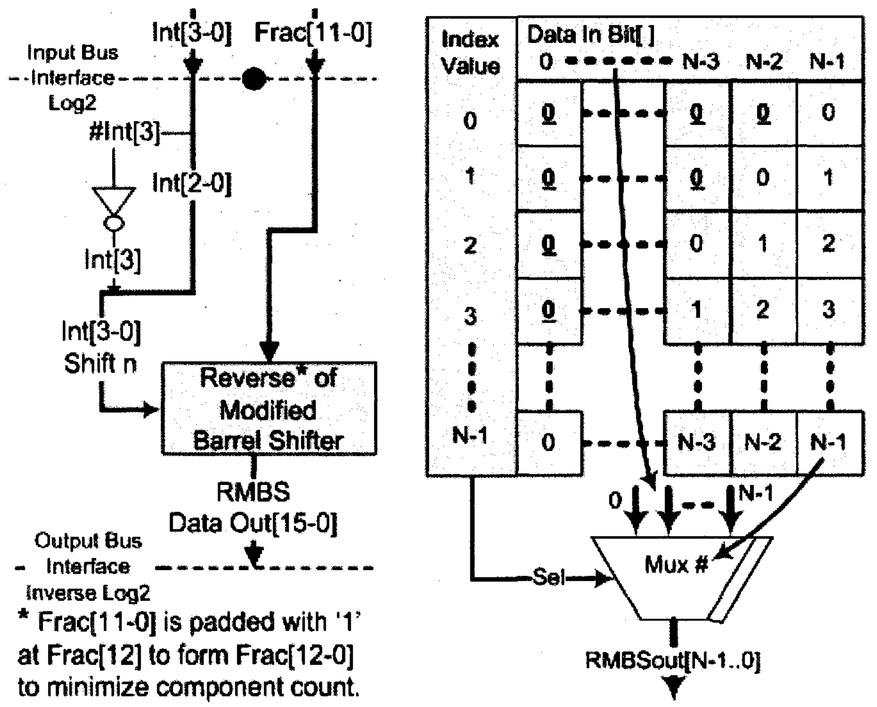


Figure 15: (a) Architecture of inverse-$\log_2$, (b) Mapping of multiplexers in RMBS.

For applications where such small numbers are insignificant, the hardware resource can be reduced by half for the conversion of signed inverse-$\log_2$ scale to linear scale. Another important point is the fraction bits fed to the reverse of the modified barrel shifter should be padded with logic '1' at the MSB such that the magnitude of index can be restored in binary. It is the equivalence to performing the OR operation between the decoded bit and the unpadded fraction bits if the decoder was included in the architecture to reverse the operation of $\log_2$ architecture. The operating frequency of inverse-$\log_2$ architecture is estimated to be twice that of the $\log_2$ architecture as the propagation delay of the critical path is reduced to half.

## 5.2.3 Error Correction

The error correction factor has the form of (3.12) and can readily be combined with the fraction bits of (3.13):

$$\left(\log_2 N'\right)_{binary} = \overline{k}^{*}_{msb} k_{msb-1} \ldots k_0 . f_{-1} \ldots f_{-8} + 0 \ldots 0.000 EC_{-4} EC_{-5} EC_{-6} \tag{5.2.1}$$

Likewise the inverse-log$_2$ is just the reverse of this process. The simple correction circuit that improves the precision is shown in Fig 16. It only requires a total of 16 logic gates to generate the correction factor and two bits full adder (FA) and four bits half adders for addition of (5.2.1). The placement of correction circuitry is shown in Fig 17 with the logarithmic modules fully optimized. A few more pipeline stage is introduced to each module as a result of incorporating the EC factor. Although not shown in Fig 17, we managed to eliminate the padding shown in Figs 14 and 15. The number of mux/demux necessary for mapping is reduced to eight sets. The advantage of these modules will become clear once we apply it in the following sections.

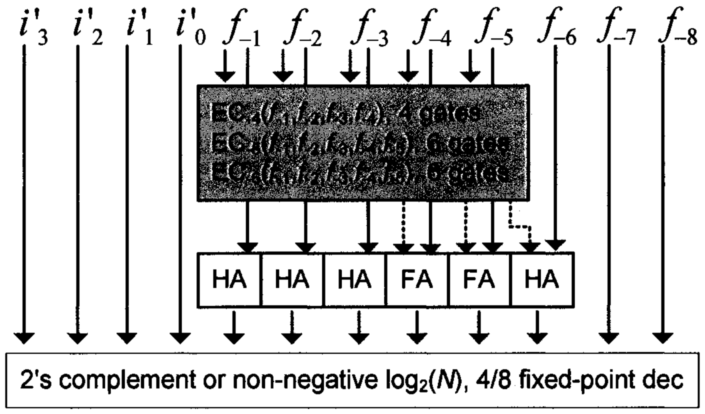

$i'_3$ $i'_2$ $i'_1$ $i'_0$ $f_{-1}$ $f_{-2}$ $f_{-3}$ $f_{-4}$ $f_{-5}$ $f_{-6}$ $f_{-7}$ $f_{-8}$
$EC_{-4}(f_{-1}, f_{-2}, f_{-3}, f_{-4})$, 4 gates
$EC_{-5}(f_{-1}, f_{-2}, f_{-3}, f_{-4}, f_{-5})$, 6 gates
$EC_{-6}(f_{-1}, f_{-2}, f_{-3}, f_{-4}, f_{-5})$, 6 gates
HA
HA
HA
FA
FA
HA
2's complement or non-negative log$_2$($N$), 4/8 fixed-point dec

Figure 16: Error Correction Circuitry.

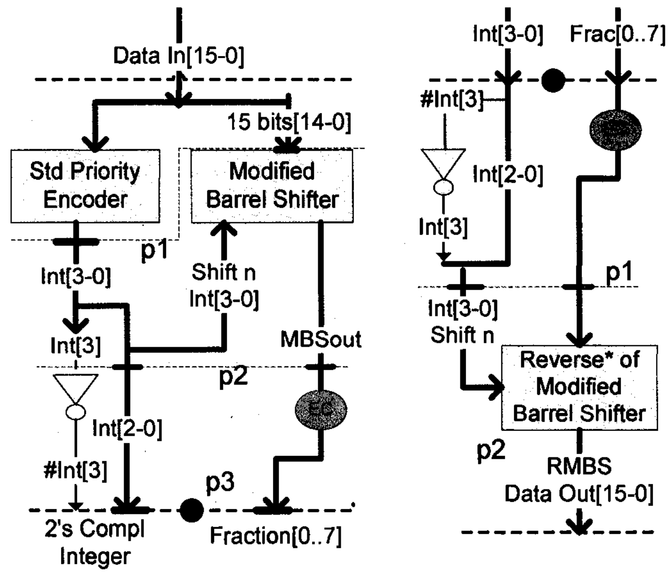


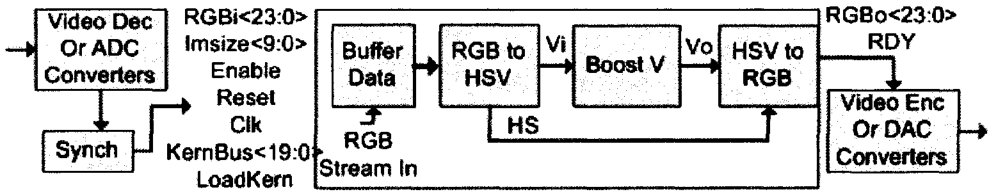
Figure 17: Fully optimized architectures of $Log_2$ and $iLog_2$ with error correction factor.

## 5.3 Video Enhancement Module

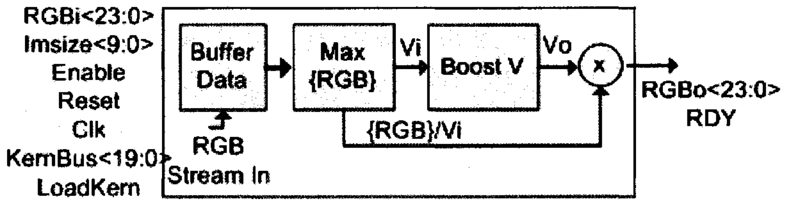
One of the main tasks in the stabilization system is the video enhancement which compensates the physical limitation of image sensing devices. This section discuses the design of various components which contribute to a tightly coupled system architecture capable of sustaining a very high throughput rate.

### 5.3.1 Overview of Computational Sequence

A brief overview of the enhancement system with full color space transformations is shown in Fig 18a along with its interface signals which are connected to the supporting circuitry (the 'Synch' block) synchronous to an off-shell video input chip. The input source can be from a video decoder, assuming progressive scan mode for which odd and even video fields are not interlaced, or from VGA source digitized by analog-to-digital converters (ADCs). The output is achieved likewise with video encoder or digital-to-analog converters (DACs), one for each color component. The core architecture features RGB streaming input with the options of specifying the image width on 'Imsize' bus, and reloading of the kernel coefficients through 'KernBus' for the convolution operation. The output buses include the enhanced RGB components. The computational sequence takes place as follows. The input pixels are buffered just enough to create an internal parallel data bus (PDB) to maximize the fine grained parallelism for massive parallel processing. This bus is also common to other core engines to be discussed in later sections. RGB to HSV color space transformation is calculated and followed with enhancement of V component. Finally, HSV to RGB conversion is performed with enhanced V before being sent to video output circuitry for testing and storing in the frame buffer for video stabilization. While the video I/Os are fixed by off-shell components, the complexity of the enhancement core can be simplified as discussed in section 4.1. The V component can be directly extracted for enhancement and normalizing original RGB components as shown in Fig 18b. The final output is computed by simply merging the enhanced V and normalized RGB. Hence one division and multiplication approximates the full two-way color transformation in the computational sequence.



(a): Computational sequence of video enhancement system with full color space transformations.



(b): Simplified core computing sequence.

Figure 18: Block diagram illustrates the overall sequence of computation alone with simplification.

## 5.3.2 Tightly Coupled System Architecture

The tightly coupled system architecture is illustrated in Fig 19. It mainly consists of three units, the data buffer unit (DBU), the homomorphic filter unit (HFU), and the fast HSV to RGB conversion (HRC) arithmetic for which H and S components are never calculated. The integration of these units contributes to consistent and highly parallel-pipelined design to maximize hardware utilization and deliver very high peak performance which might be degraded in a loosely coupled or unevenly pipelined system. The design of these units is discussed in greater detail, keeping in mind the computational sequence, in the following sub-sections along with notations introduced as they appear.

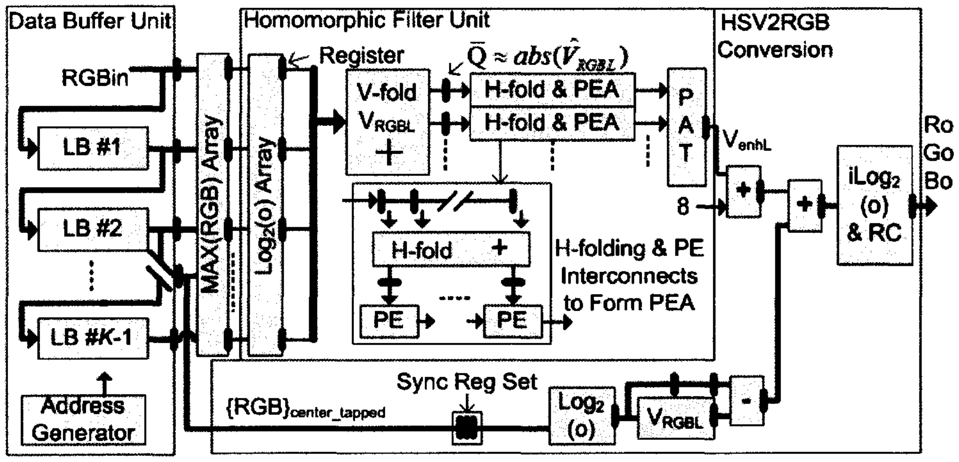


Figure 19: System architecture illustrates the coupling of three main units to achieve very high performance with simplicity in the design.

## 5.3.3 Data Buffer Unit

The DBU is implemented with the dual port RAMs (DPRAMs) as shown in Fig 20. One set of DPRAMs is utilized to form line buffers (LBs) and store just enough lines of image in the LBs to create massive internal parallelism for concurrent processing. The pixels are fetched into the DBU in raster-scan fashion which requires unity bandwidth for the input data. The DPRAM based implementation has the advantage of significantly simplifying the address generator compared to commonly known first-in-first-out (FIFO) based approach. The address generator with the DPRAMs based implementation makes scalability of DBU consistent and simple. It consists of two counters to automatically keep track of the memory locations to insert and read the data to internal PDB for extraction of V-component. Data bus A (DBA) of ($K$-1) $\times P_{RGB}$ bits wide, which is formed with just enough number of DPRAMs in parallel, is used to insert pixel values through write-back paths to the memory location designated by address bus A (ABA). For eight-bit pixel resolution, $P_{RGB}$ is 24 bits. The data bus B (DBB) is used to read the pixel

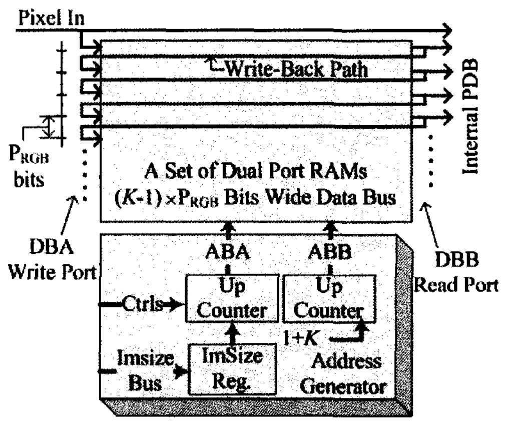


Figure 20: Detail architecture of the DBU shown in Fig 19. The data bus of ($K$-1) $\times P_{RGB}$ bits wide is grouped into a number of 24-bit paths to form effective LBs for 8-bit pixel resolution.

values onto internal PDB and write to the write-back paths. Only one address generator is necessary in DBU.

### 5.3.4 Extraction of V-component

The V-component is extracted by a max filter. The concept was simplified from the architecture for separable filters suitable 2D uniform filters [19]. For 1D max filter, which is what we need in this design, a pipelined adder tree (PAT) style can be utilized. A generalized 1D max filter architecture for *N* nodes is shown in Fig 21. The design utilizes the signs from subtractions in the PAT structure to successively filter and merge until a maximum value is found at the end of last pipeline stage. An array of *K* 3-to-1 max filters is necessary as illustrated in MAX(RGB) Array block of Fig 19. This architecture works for min finder as well by swapping the inputs fed to 2-to-1 multiplexers.

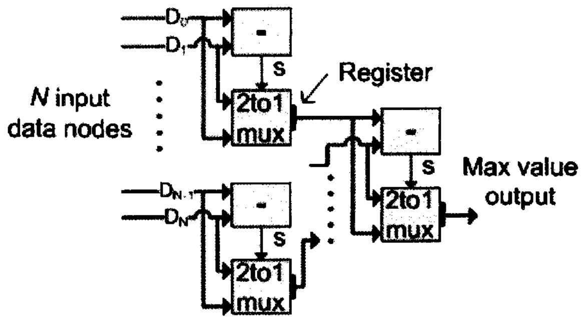


Figure 21: Elementary architecture of the max filter is used to extract the V-component. *K* elements of 3-to-1 max filters are needed in the MAX(RGB) Array as shown in Fig 19.

### 5.3.5 Architecture of Homomorphic Filter

The HFU coupled with an array of the $\log_2$ scaled version of extracted V-component is illustrated in Fig. 19. The quadrant symmetry property of the 2D convolution operation indicated by (4.1.4a) and (4.1.4b) allows the computation to concentrate on one quarter of the kernel through folding. The vertical folding of data is accomplished by linearly folding the data from the last stage of internal PDB (the $\log_2$ scaled version of V component array) with adders. This halves the processing bandwidth. To normalize a value $v$ ($log_2(v/2^N) = log_2(v) - N$), which is negative, given the fact that image pixels are positive and $\log_2$ of negative number is undefined, the absolute value can be logically approximated by taking the inverted output ($\overline{\mathrm{Q}} \approx N - log_2(v) = \overline{log_2(v)}$) of the registered result from vertical folding. This procedure inherently utilizes the V-fold pipeline stage rather than introducing an additional stage and resource to compute the absolute value of the normalized $v$. To reduce the processing bandwidth by another half, the horizontal folding defined by (4.1.4a) and (4.1.4b) is translated to (5.3.1a) and (5.3.1b) and performed with respect to even and odd dimension kernels, taking account of the inherent delay in the systolic architecture. The H-fold denotes the results from horizontal folding, and HQ is a set of horizontal shift registers for vertically folded data. The registered results of the H-fold stage are sent to arrays of processing elements (PEs) for successive filtering. The partial results from the PE arrays (PEAs) are combined together by a pipelined adder tree (PAT). The overall output of the homomorphic filter is kept in the $\log_2$ scale for the fast color space conversion in the HRC architecture as shown in Fig 19.

$$\text{H-fold}(k) = \begin{cases} \text{HQ}[0] + \text{HQ}[2k+1], & \text{for odd } K, k \neq 0 \\ \text{HQ}[0], & \text{for odd } K, k=0 \end{cases} \tag{5.3.1a}$$

$$\text{H-fold}(k) = \text{HQ}[0] + \text{HQ}[2k], \qquad \text{for even } K, \forall k \tag{5.3.1b}$$

The design of the PE in the homomorphic filter utilizes the log-domain computation to eliminate the need of hardware multipliers. The data from the H-fold register is pre-normalized without extra logics by shifting the bus. It is then converted to $\log_2$ scale as shown in Fig 22 and added with $\log_2$ scaled kernel coefficients (LKC) in LKC register set which is initialized through chained bus with 'LKCin' and 'LKCout' signals. The result from last stage is converted back to linear scale with range check (RC). If the overflow or underflow occurs given the desire range, the holding register of this pipeline stage is set or clear, respectively. Setting and clearing contribute the max and min values representable to $N$-bit register. The output of this stage is de-normalized, likewise by bus shifting, before it is successively accumulated along the accumulation line.

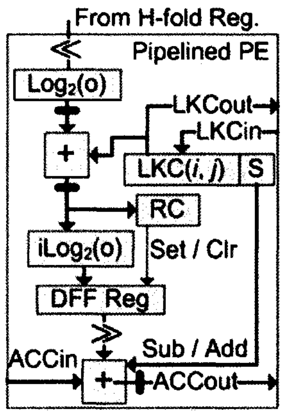


Figure 22: Architecture of the PE in the homomorphic filter.

## 5.3.6 Fast HSV to RGB Color Space Conversion

The HRC unit inverse transforms the enhanced image in HSV color space back to RGB representation without computing the H and S components. As illustrated in Fig 19, the center-tapped RGB components from DBU pass through synchronization register set to compensate the latencies associated with HFU. The synchronized RGB components are converted to $\log_2$ scale. Furthermore, the V-component at this node is also determined with the architecture shown in Fig 21. The $V_{enhl}$ output is first de-normalized by adding constant 8 for 8-bit pixel resolution in log-domain which is equivalent to multiplication of de-normalizing factor $D = 2^8$. The division in (4.1.11) is calculated by subtraction in log-domain as illustrated in Fig 19. The final output of the enhanced RGB components is computed by taking the inverse-$\log_2$ of the sum of the resultant subtraction and $V_{enhl}$. This completes the discussion on the design of image enhancement system.

## 5.4 Single Layer Feature Selection

The design of architecture for computing the corner-ness response is covered in this section. Furthermore, the memory layout and the conditions which trigger the events of capturing and flushing of potential features are also described with respect to the score of the texture.

### 5.4.1 Overview of Feature Selection & Storage

Feature selection involves evaluation of the score from its texture. The block diagram of this subsystem is shown in Fig 23. It basically has two main components: calculation of feature score and the storage memory. Given the new score, R, the coordinate and the feature in the memory is updated if a better score is observed, thereof, overwriting any preexisting sub-optimal potential features. The most important part of the design for this subsystem is the calculation of texture score.

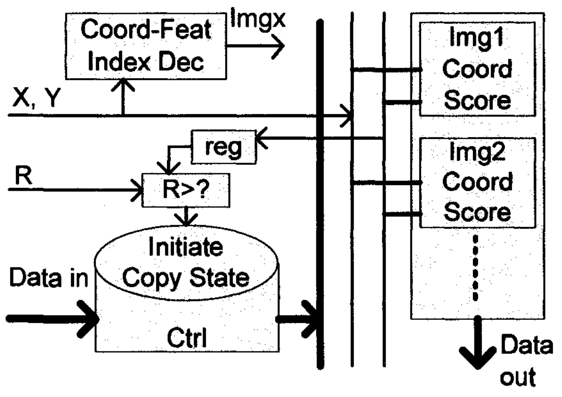


Figure 23: Block diagram of feature selection and storage subsystem.

## 5.4.2 Components of Auto-correlation Matrix

The task to obtaining the texture score is to compute the auto-correlation matrix, **M**, of (4.2.16) on the first step. Only then can the response, R, be calculated from (4.2.17). The detail diagram for the design of the architecture is shown in Fig 24. A 3x3 kernel is utilized for the calculation of the derivatives, *Dx* and *Dy*, from the grayscale image. The kernel mask equivalent to differentiation of discrete samples has the coefficients of

$$Dx = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad Dy = Dx^T. \tag{5.4.1}$$

Since all coefficients are exactly power of two, no real multiplication takes place. Due to 2D summation of a 3x3 kernel, we must utilize three separate derivatives to concurrently calculate the A, B, and C components of **M** matrix. For the *Dx* shown in Fig 24, vertical folding is applied to merge and reduce computing nodes into one by summing the adjacent lines on LBs and the left shifted (multiplied by two) version of the LB in the middle. This result is then subtracted with the delayed/earlier version of the partial results. By the same token of folding within the architecture as of Fig 19, only three adds/subtract are needed. Likewise the $Dy_1$ to $Dy_3$ are calculated with minor difference. Instead of folding three nodes vertically, the folding is done horizontally. Although not exactly as depicted in Fig 24, the adder node with left shifted data is actually added from the last delay, Z, component due to pipeline delay of the three-node adder. The fact that we utilize logarithmic modules forces us to extract the signs of $Dx_{1\text{-}3}$ and $Dy_{1\text{-}3}$. This is not necessarily undesirable as one must compute the squares of *Dx* and *Dy*. Once the $\log_2$ scale values are obtained the buses are wired to the left for effect of squaring the numbers for A and B components of **M** matrix. The summation of $\text{ilog}_2$ values does the job of combining the results. The C component is calculated slightly different in that the signs of *Dx* XORing *Dy* are recombined to form signed numbers for a 3x3 summation. Since the real computing bandwidth mandatory is the moving video lines, a full 2D summation is a bit of exaggeration for what the bandwidth really demands. Although the dimension is rather small in this case for the requirement of hardware resource to be negligible, we should see in a moment that 2D integral summation is the most efficient implementation in such a scenario. With the folding style of Fig 19 alone, it can easily be accomplished by folding vertically and then horizontally through its own delay lines as shown on the right side of Fig 24. For the same reason of pipeline timing effect, addition of the middle node of horizontal delay registers is really the value of the rightmost register for computing the final results. As far as programming is concerned, the compiler will generally simplify it to eliminate unnecessary registers.

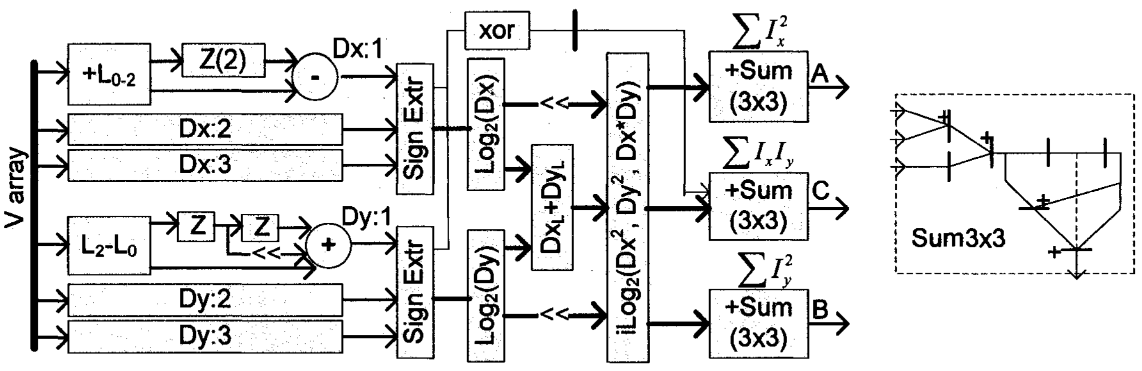


Figure 24: Architecture for calculation of A, B, and C components of M matrix.

## 5.4.3 Logarithmic Corner-ness Response

Given the A, B, and C components from the previous section, the computation of corner-ness response seems effortless. To obtain the components of AxB, $C^2$, and $k(A+B)^2$, we simply calculate the $\log_2$ version and manipulate the buses with adders and subtracters as needed. The results converted back to linear scale may then be merged to form the final score, as illustrated in Fig 25. Due to excessive word length of the scalar score, the response, R, is converted back to $\log_2$ scale, $R_L$, which occupies only 13 bits for storage memory. A 13-bit scalar that represents the magnitude of 32-bit number is

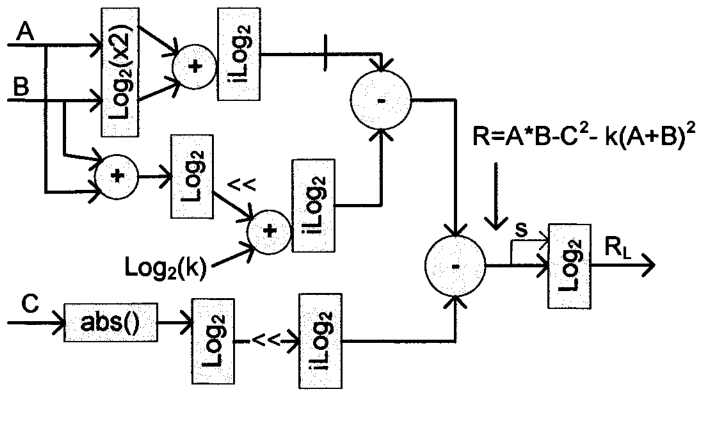


Figure 25: Architecture for calculation of the response in $\log_2$ scale to reduce word length.

efficient enough for our purpose. The storage layout and the condition for capturing and flushing the potential features are discussed in next section.

### 5.4.4 Storage of Potential Features

The timing which initiates the copy state to capture or flush the potential features depends on the coordinate XY values of its subsystem. The copy state for the capturing of potential features is triggered by the textural score which improves the preexisting features. The condition stated in Fig 26 is as follows. The existing score output from memory 'Scro' is subtracted from $R_L$. The sign bit indicates one instance to initiate the copy state, given the potential feature module is currently active, 'pfActive'. This copy state captures the primary sub-feature of the potential feature as shown on the top circuit of Fig 26. Due to the pipeline latency of dual port RAM for our primary storage, values written to the BRAM are not immediately available, and the delayed version of this new score, $R_L(1)$ is also used to trigger the copy state which first terminates the copy state machine previously executed. The second condition of initiating copy state becomes active when the calculated XY coordinate is exactly at the center of secondary sub-features of the potential features. The '$Sc_{1..2}XY$' constants are used to compensate the pipeline delay of the subsystem. Due to the layout of storage ram, the copy state is really just the assertion of the write signal to the BRAMs for a certain number of circles determined by the counter. Flushing of the potential features from the BRAM is accomplished in a similar manner.

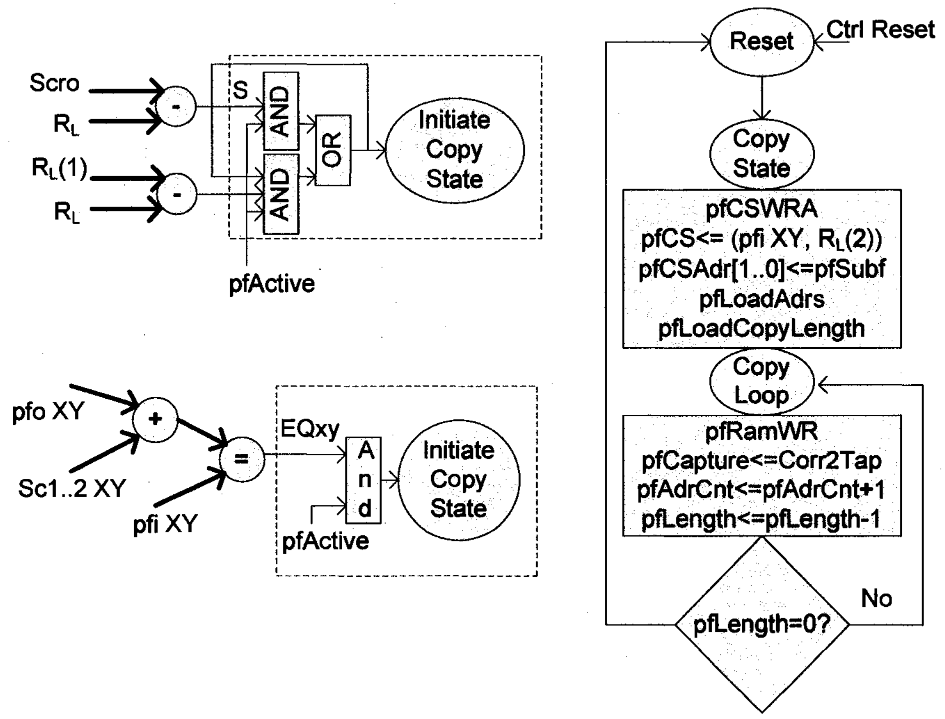


Figure 26: Conditions of initiating copy state.

The storage layout of feature selection subsystem mainly comprises the BRAMs in single port mode and the XY coordinate with score in a separate BRAM operated in dual port mode to share storage of two components related to energy calculations. As illustrated in Fig 27, a sufficient number of BRAMs must be stacked together to form the required number of bits for each column within the sub-feature. Since we already know the bandwidth of moving video lines, one does not need to take the snapshot of the feature in a single clock cycle. This essentially implies the minimum use of FPGA resource solely allocated for the routing of the bus. The 'pf capture' input bus comes from part of the NCC architecture for which a single column can be captured into the pf storage at proper timing. Likewise, 'pf flush' pumps out the pf features column by column for the NCC storage circuit to capture into its own storage. To the left of Fig 27, the XY coordinates and the scores are constantly updated on Bank0 of the BRAM. The

Bank1 is dedicated to capturing and flushing of energy components of the NCC calculation. Due to slight pipeline differences the 'Sync Regs' are introduced with more delays to synchronize inputs/outputs with late-read/write policy. As far as addressing the BRAMs is concerned, five bits are dedicated to store maximum of 32 columns per sub-feature, another five bits are assigned directly as the index of the complete feature, and finally two bits are used to select sub-feature index. In this fashion no real address to feature index translation is required as in fully utilized memory locations. It is also the reason to configure BRAMs with 12 bit address lines. The cost of such convenience comes with a quarter waste of the total storage space under current design with total of three sub-features. In the next section, it should become clear on how the process of capture and flush takes place.

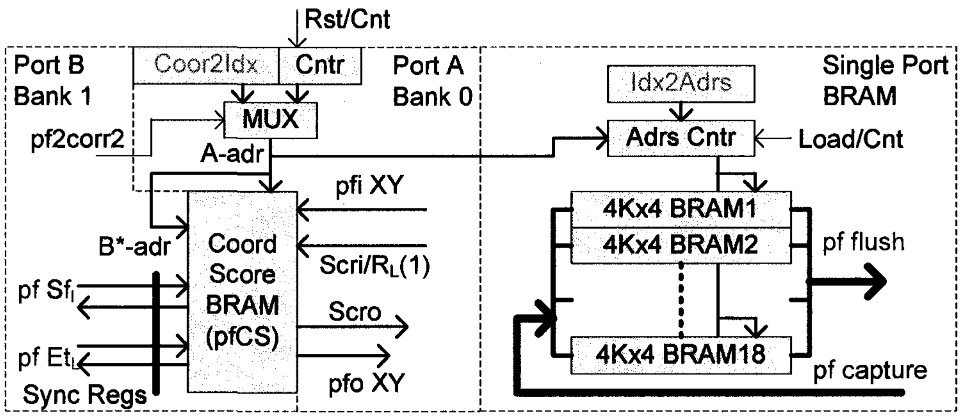


Figure 27: Storage layout of feature selection subsystem.

## 5.5 Feature Measure & Tracking with Improved NCC Architecture

The design of feature measure and tracking subsystem addresses the issues of efficient realization of NCC architecture which include the partial calculation of NCC score, the 2D summation and correlation, and the essential layout of the memory to meet the bandwidth requirement for feature storage and retrieval.

### 5.5.1 Overview of Shared NCC Architecture

The overview of a shared NCC architecture is illustrated in Fig 28. Given the input column array, the NCC score can be evaluated by pumping the data through 'Partial NCC' block. This block must be tightly coupled with the feature storage shown on the right side of the figure. The close coupling is absolutely necessary due to the extreme processing bandwidth and the minimum waste cycles within the 'Partial NCC' block. Similar to the feature selection subsystem, the NCC has another copy state to capture or release the data to NCC storage memory. The ownership of 'Partial NCC' block depends on the score of current results. The 'Corresp. Maint.' regulates the dataflow and updating between the storage and 'Partial NCC' blocks.

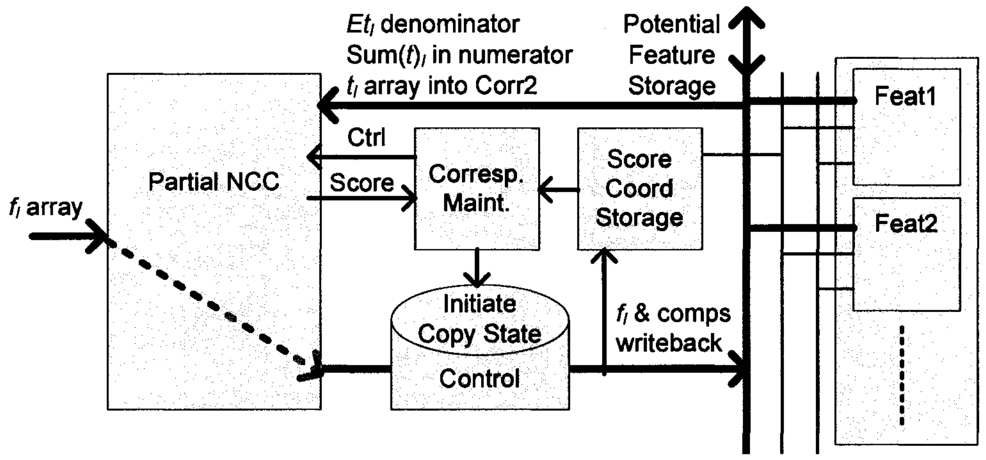


Figure 28: Block diagram of NCC and storage subsystems.

### 5.5.2 Partial NCC as Parallel Filters

The earlier implementation of NCC architecture we presented in [68] which computes (4.2.9) directly has a very high performance yet with extremely inefficient utilization of hardware resource. This inferior design which allocates an excessive amount of resource also has no flexibility on potential scalability regarding the dimension of the feature template. Although the demand on resource may be cut back with performance trade-off as in [69], the response time with respect to dynamic mobility of the templates is still unacceptable. The reason is trivial while the change of templates executes, the correlation scores being calculated are virtually meaningless as they are direct resultants of blending the templates. This creates large blind spots if one must simultaneously scan several features. While the simplification of section 4.2.5.2 may significantly cut back the number of necessary calculations, hardware implementation of partial NCC may still be achievable to sustain unity search bandwidth for the measure over several features simultaneously. To get around with the problem without compromising its performance for the gain of resource, we adhere to the assumption of the observation of sub-optimal scores around the search targets. The blind spot must be minimal by avoiding a timeslot multiplexed resource-performance trade-off.

The earlier NCC architecture was implemented as cascaded filters with component evaluation of (4.2.9) followed by average filters. The serial nature of calculation sequence imposes data dependency which requires a set buffer space to hold the partial results. It is, therefore, wise to realize the design as parallel filters with the following simplification:

$$\begin{aligned}
\phi(x,y) &= \frac{\sum_{u,v}\left[\left[f(x+u,y+v)-\bar{f}_{u,v}\right]\times\left[t(u,v)-\bar{t}_{u,v}\right]\right]}{\left[\sum_{u,v}\left[f(x+u,y+v)-\bar{f}_{u,v}\right]^2\sum_{u,v}\left[t(u,v)-\bar{t}_{u,v}\right]^2\right]^{1/2}} \\
&= \frac{\sum_{u,v}\left[f(.)\times t(u,v)+\bar{f}_{u,v}\bar{t}_{u,v}-\bar{f}_{u,v}t(u,v)-\bar{t}_{u,v}f(.)\right]}{\left[\sum_{u,v}\left[f(.)^2+\bar{f}_{u,v}^2-2\bar{f}_{u,v}f(.)\right]\sum_{u,v}\left[t(u,v)^2+\bar{t}_{u,v}^2-2\bar{t}_{u,v}t(u,v)\right]\right]^{1/2}} \\
&= \frac{\sum_{u,v}f(.)\times t(u,v)+UV\bar{f}_{u,v}\bar{t}_{u,v}-UV\bar{f}_{u,v}\bar{t}_{u,v}-UV\bar{t}_{u,v}\bar{f}_{u,v}}{\left[\left[\sum_{u,v}f(.)^2+UV\bar{f}_{u,v}^2-2UV\bar{f}_{u,v}^2\right]\left[\sum_{u,v}t(u,v)^2+UV\bar{t}_{u,v}^2-2UV\bar{t}_{u,v}^2\right]\right]^{1/2}} \\
&= \frac{\sum_{u,v}[f(.)\times t(u,v)]-UV\bar{f}_{u,v}\bar{t}_{u,v}}{\left[\left[\sum_{u,v}f(.)^2-UV\bar{f}_{u,v}^2\right]\left[\sum_{u,v}t(u,v)^2-UV\bar{t}_{u,v}^2\right]\right]^{1/2}},\ UV\bar{f}_{u,v}^2=UV\bar{f}_{u,v}\bar{f}_{u,v}=\frac{1}{UV}\left(\sum f_{u,v}\right)^2 \\
&= \frac{C-\frac{1}{UV}S_tS_f}{\left[\left[S_{f^2}-\frac{1}{UV}S_f^2\right]\left[S_{t^2}-\frac{1}{UV}S_t^2\right]\right]^{1/2}}
\end{aligned} \tag{5.5.1}$$

The correlation, C, may now be computed in parallel with components of energy normalization. The sum of *f*, S<sub></sub>$S_f$, and the square of the sum *f*, $S_f^2$ can be treated as a single component while the sum of squared f, $S_{f^2}$ may also be calculated in parallel. Equation (5.5.1) also makes the design easier in that all partial results are non-negative.

### 5.5.2.1 Architecture of Normalizing Factors

The architecture of feature measure is illustrated in Fig 29 with the detailed portion dedicated for energy normalization of the correlation results. When the data of moving video lines pump through entry nodes, $f_{u,v}$, from common PDB (internal parallel data bus discussed in section 5.3) of the system, the square of *f* is calculated with bus shift of readily available $\log_2$ scale version of $f$, $f_l$. At the output node, the column array of $f^2$ is merged to compute the 2D summation by integral sum. Concurrently, the sum of $f$ is been computed and then also converted to logarithmic domain for division with UV factor which is the constant of $\mathrm{U}\times\mathrm{V}$ sub-feature dimension. Since the pipeline latency of $S_f^2/UV$ stage is one more than the $S_{f^2}$ stage, a register is included to synchronize the partial results for subtraction operation to once again merge the nodes and obtain a more complete partial result. The output of $\log_2$ scale $E_f$ is then added with $E_t$ to form the denominator portion of (4.2.9). While this part of the architecture performs the calculation of denominator, the factor $S_tS_f/$ UV is also being computed in a parallel data path. The earlier node which has the result of $S_f$ in $\log_2$ scale is also being merged (through synchronizing registers) with the addition (linear scale multiplication) of $S_t$ component and subtraction (linear scale division) of UV constant. Before converting to $\log_2$ scale for the final energy normalization sequence, the result of $S_tS_f/$ UV is combined with the output from 'Corr2' correlation block which has the most expensive computing power of the entire NCC architecture. The nodes in red color labels are used to capture or flush the current state of the sub-features between NCC subsystem and the pf subsystem. The blue nodes have the same function but are interfaced with local storage for cycling between current and previous video frames. The final scores of NCC output are tagged with a specific feature index for further processing. Note the calculation of energy components is quite hardware resource friendly. The real burden of the design is in the Corr2 block along with its storage.

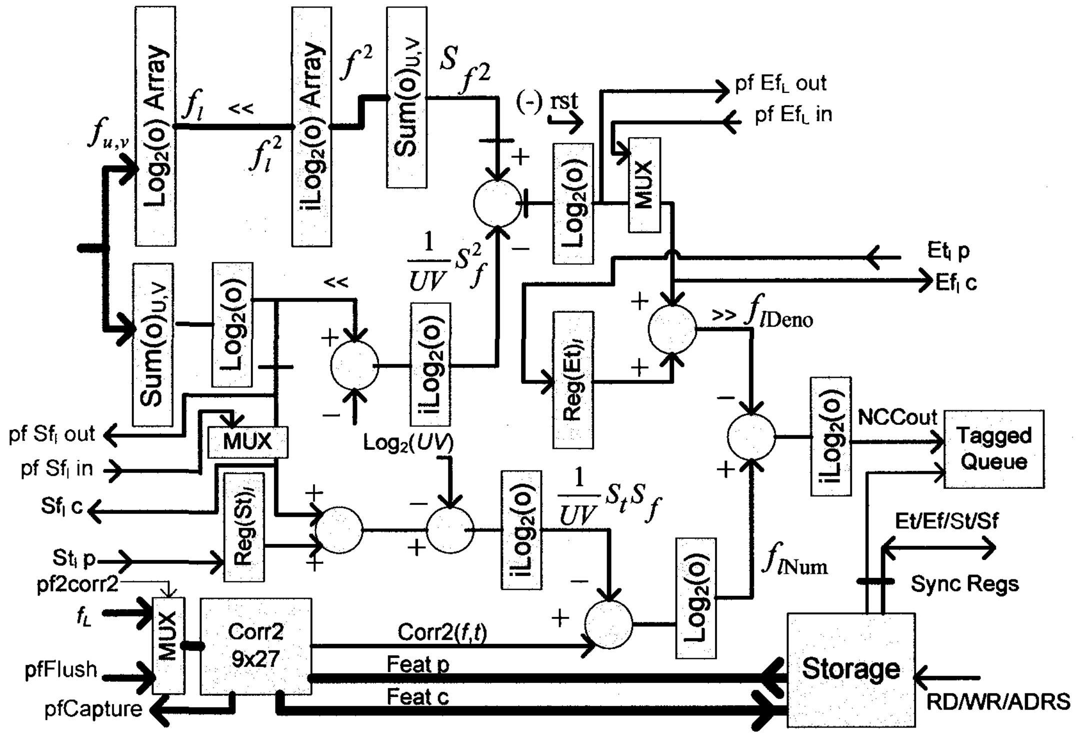


Figure 29: Architecture of feature measure by NCC.

### 5.5.2.2 2D Integral Summation

One of the most efficient components includes the design of 2D integral summation. Although integral sum is widely used in computer vision to reduce the complexity of certain calculations on the desktop computer, the same functionality conventionally implemented in hardware is really accomplished by either fully pipelined adder tree (PAT) structure or modularized processing elements (PEs). The average cost is one adder per node/PE which requires $U \times V$ nodes given such dimension of the sub-features. With 2D integral sum, the cost exclusively depends on the number of moving rows or video lines. For the example of nine elements in a column array, as illustrated in Fig 30, the PAT structure is needed to reduce processing power to a single node. This partial result then flows through the 27 registers to be applied for summation under of a

$9\times27$ window. While the PAT result is been registered through a series of registers, it is also sent for the accumulation of preexisting values. On the other hand, the output of 27 tap registers at node B is applied to deccumulate the accumulated result from node A in Fig 30. The final output of a $9\times27$ summation can simply be obtained from partial results on registers A and B. Nodes A and B are essentially a single node which performs an add-and-subtract operation but are decomposed to reduce the delay path between the registers. The 2D summation in this example only requires ten adders and one subtractor as opposed to 242 adders of a full scale architecture.

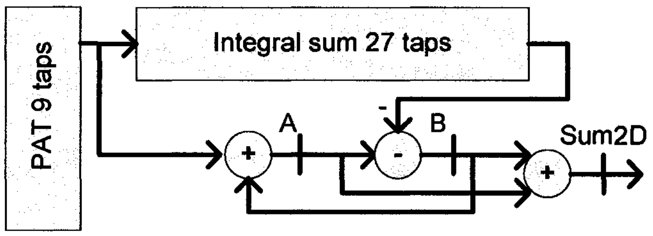


Figure 30: Architecture of 2D integral summation.

#### 5.5.2.3 2D Correlation

While the simplicity of integral sum is fascinating, one would wish to copy such idea and paste it into the design of the summation in Corr2 block. The true processing power of the Corr2, however, is not one column array per clock. Rather, it requires 27 column arrays per cycle in our example. It is, therefore, inevitable to implement a full scale PAT to merge all partial results and produce the final output on cycle basis. The architecture of Corr2 is shown in Fig 31 along with the PE. Each PE basically calculates the multiplication of *f* and *t* before the PAT structure. Although log-domain computation simplifies the burden to a certain extent, a $9\times27$ Corr2 architecture still demands 243 8-

bit adders, 243 16-bit iLog$_2$ modules and 242 16-bit pipeline stage expanded adders. The Corr2 module consumes a majority of the resource in our video stabilization system; however, it should also be noted that a $9 \times 27$ dimension sub-feature in our design represents virtually a limited number of full features. When the sub-features are combined, the processing power is virtually a multiple of base dimensions. For the example of a primary sub-feature accompanied by two (or more) secondary sub-features, the virtual computational complexity becomes $27 \times 27$ at the expanse of the base dimension. That is something infeasible in the earlier design regardless of resource optimization.

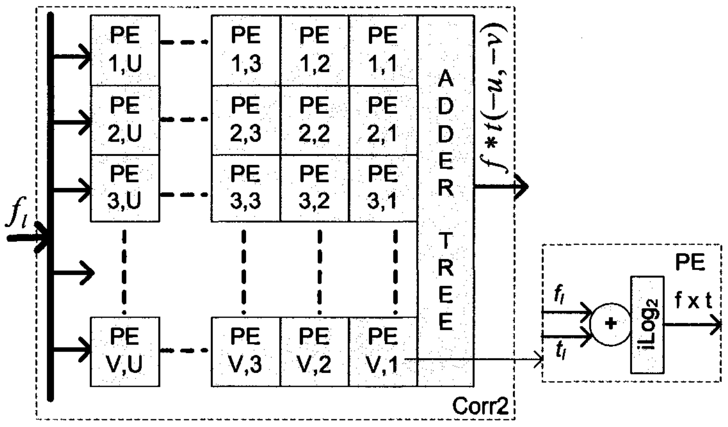


Figure 31: Architecture of Corr2 module.

## 5.5.3 Storage Layout

The storage layout of the NCC module demands the highest bandwidth for which the snapshots of the new sub-features may be captured on cycle basis. To cope with such bandwidth, the queues are localized to distribute the storage of sub-features with respect to the kernel nodes of the NCC architecture. As shown in Fig 32, the dual port BRAMs are layout in a way that the input and output nodes of PortA and PortB are completed attached to NCC block with one-to-one connectivity. This configuration makes the capturing of sub-features possible within a single clock cycle. Due to the available layout of the embedded BRAM itself on the FPGA, the nine-bit address lines are reduced to eight bits in dual port setting to minimize memory waste and cut back the required number of BRAMs by half. Similar to the pf storage in section 5.4.4, total of eight-bit address lines are needed: five-bit for feature index, two bits for sub-features, and one bit for toggling between previous and current feature sets. Given such a memory layout, there is no real index to address translator as shown in the address field of Fig 32.

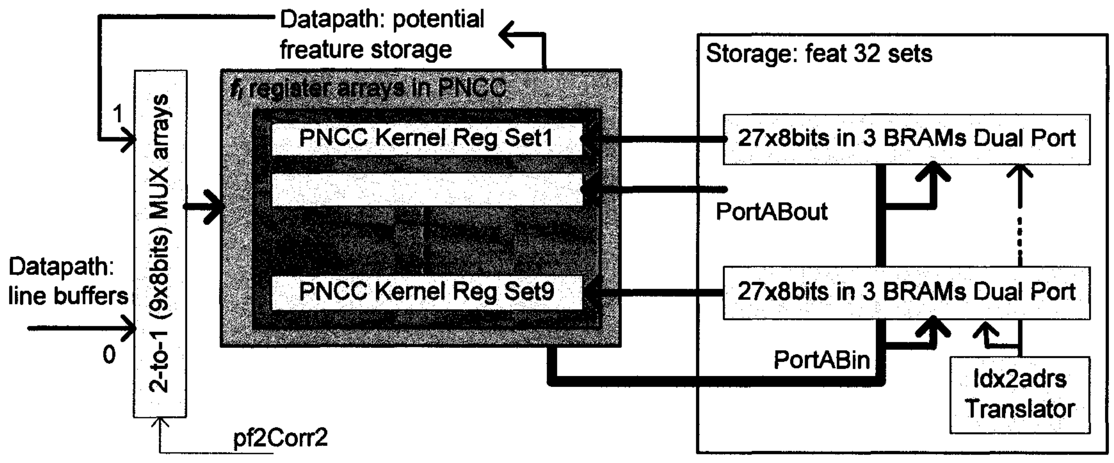

Datapath: potential freature storage
$f_i$ register arrays in PNCC
PNCC Kernel Reg Set1
PNCC Kernel Reg Set9
2-to-1 (9x8bits) MUX arrays
1
0
Datapath: line buffers
pf2Corr2
Storage: feat 32 sets
27x8bits in 3 BRAMs Dual Port
PortABout
27x8bits in 3 BRAMs Dual Port
PortABin
Idx2adrs Translator

Figure 32: Storage layout of the NCC architecture.

## 5.6 Correspondence Management

The 'Correspondence Maintenance' subsystem of Fig 13 serves to manage the results between the NCC module and coordinate of the features as illustrated in Fig 33. The BRAM memory block stores the XY coordinate and the score of each feature similar to the layout of pf storage block. When running into the leftmost coordinate of each moving video line, the Y coordinate of previous feature, fxp, is read out to check if the search range should be active in the 'Range Check' block. Given the 'A-adres' (5-bit feature index) and the in-range flag, 'InRng', the XY coordinate is written to one of 8 index register table, 'IdxRegArr' which serves as small cache for fast target scanning. The actual write to a particular register is enabled by the 'Rotate' signal which contains the one logic '1' bit with in the eight-bit serial ring. The purpose of such a cache is to eliminate waste cycles. Given 32 feature sets, one would be required to read these features in 32 clock cycles, regardless of its range. This mechanism very passively creates undesirable blind spots for computing the NCC scores. The cache solves the problem by retreating the searchable coordinates in advance to the actual feature measure, making it more scalable to larger set of features. The '#RngFlg' generated by each register in the table is calculated by the distance of range and the current X coordinate of the subsystem. This flag is used as a feature read signal, 'featRD', propagated to the 'PNCC & Storage' block to activate the loading of sub-feature into kernel registers. Prior to reaching the PNCC subsystem, however, the address is used to retreat the score, 'fcScr', on port B. The 'fcScr calculates whether the scan should be performed on primary or secondary sub-features based on the logic specified in Fig 33. If the fcScr is below 0.5 constant, the scan mode is designated for primary sub-feature (sf00). On the other hand if the score of the primary target is sufficiently high, the secondary sub-features are determined by both the XY ranges and assigned two-bit sub-feature address as sf01 or sf10. In this manner, the future expansion of additional sub-features can be readily incorporated into the design. When the feedback scores, 'Scr' become available with valid 'ScrTag' from the 'PNCC' subsystem, the address portion is sent back to port A to retreat the 'fcScr' values for comparison with current results. If any score signals a better match of primary sub-feature from the sign bit, 'sp', new XY coordinate is updated along with the 'Scr' and the 'sfxx' tag. For the passing score, 'ss', of secondary sub-features, the original information remains with only the 'sfxx' tags updated to enable the search of the next sub-features. In the case of sub-optimal scores of the primary sub-feature, a lock bit generated from 'IdxRegArr' becomes active when current XY coordinate comes into short range, inhibiting the scanning of other coordinates within the table. This concludes the basic design of correspondence management subsystem.

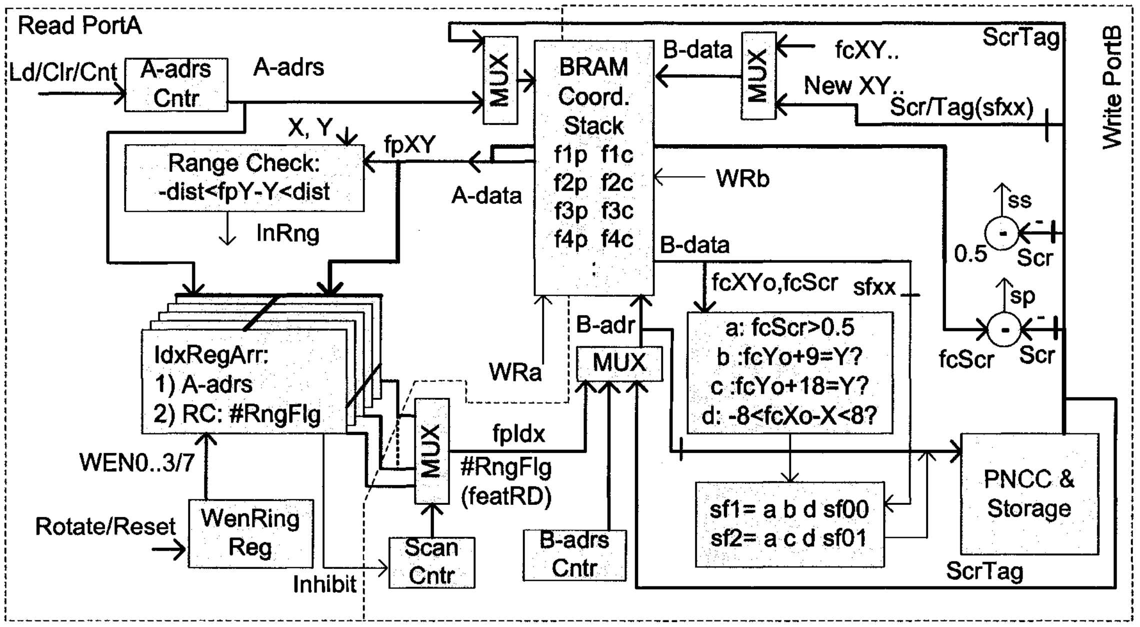


Figure 33: NCC correspondence management subsystem.

## 5.7 Motion Evaluation Module

Due to the order of search introduced in section 4.3.3, the evaluation of inlier motion boils down to the search of coherent angles between feature coordinates stored in the stack ram of Fig 33. The architecture of inlier estimation illustrated in Fig 34 calculates the angles of rotation. Given 'A-adrs Cntr' and 'B-adrs Cntr' of Fig 33, one can incrementally simulate the order of search by utilizing 'A-adrs Cntr' as the base node and 'B-adrs Cntr' as the search node. If the 'B-adrs Cntr' reaches the last feature coordinate for example, the 'A-adrs Cntr' would increment by one to initiate and move upward in the triangular order. With the data fetched from the dual port BRAM, the vectors V and $V_0$ can be calculated by subtraction. The next pipeline stage extracts the sign bits since $\log_2$ modules are limited only to positive numbers. To compute the angle given in (4.3.3), the following components must be calculated:

$$\cos(\theta) = \frac{x_1 x_0 + y_1 y_0}{\left(x_1^2 + y_1^2\right)^{1/2}\left(x_0^2 + y_0^2\right)^{1/2}} \qquad (5.7.1)$$

The signed extracted vectors are first converted to $\log_2$ scale. The squares and the square roots in the denominator are calculated with bus shifts with proper logarithmic conversions. The numerator can be computed slightly different in that $x_1 x_0$ is inverted at the output of $i\mathrm{Log}_2$ module, approximating the 2's complement format. This value is then either added or subtracted from $y_1 y_0$, depending on its own sign, $s_{y1}$ XOR $s_{y0}$. The calculation of angle along is relatively simple. A more interesting part comes for the direction of rotation in which the complete evaluation is unnecessary. The direction of (4.3.3) can be simplified by separating out the sign bits as follows

$$\begin{aligned}
&S_a \times |x_1 \times y_0| - S_b \times |y_1 \times x_0| < 0 : \mathit{clockwise} \\
&S_d = \mathit{sign}\left(|x_1 + y_0|_L - |y_1 + x_0|_L\right) \\
&S_b(+) : S_a(|x_1 \times y_0| - |y_1 \times x_0|) < 0, \quad S_b(+) : S_a \oplus S_d \\
&S_b(-) : S_a(|x_1 \times y_0| - |y_1 \times x_0|) > 0, \quad S_b(-) : \sim(S_a \oplus S_d) \\
&S_o = (S_a \wedge S_b \wedge S_d) \vee (S_a \wedge \overline{S}_b \wedge \overline{S}_d) \vee (\overline{S}_a \wedge S_b \wedge \overline{S}_d) \vee (\overline{S}_a \wedge \overline{S}_b \wedge S_d)
\end{aligned} \quad , \qquad (5.7.2)$$

where the $s_a$ and $s_b$ are the signs of XORing previously extracted signs of $x_1 \times y_0$ and $y_1 \times x_0$, the $s_d$ bit is another sign bit from subtraction of $x_1 \times y_0 - y_1 \times x_0$ directly in $\log_2$ scale and $s_0$ is the direction of rotation. With $\log_2$ version of input components readily available, it only took two adders, one subtractor and five gates to determine the direction of rotation. In doing so we eliminated two $\mathrm{iLog}_2$ modules at the cost of two adders and five gates. At the output of (4.3.3) shown in Fig 34, the incoming angles and directions are compared against designated values for coherence with respect to the reference. If the boundary error is within expected range, the index tag is saved to the queue for which its size determines the vote for inlier motion.

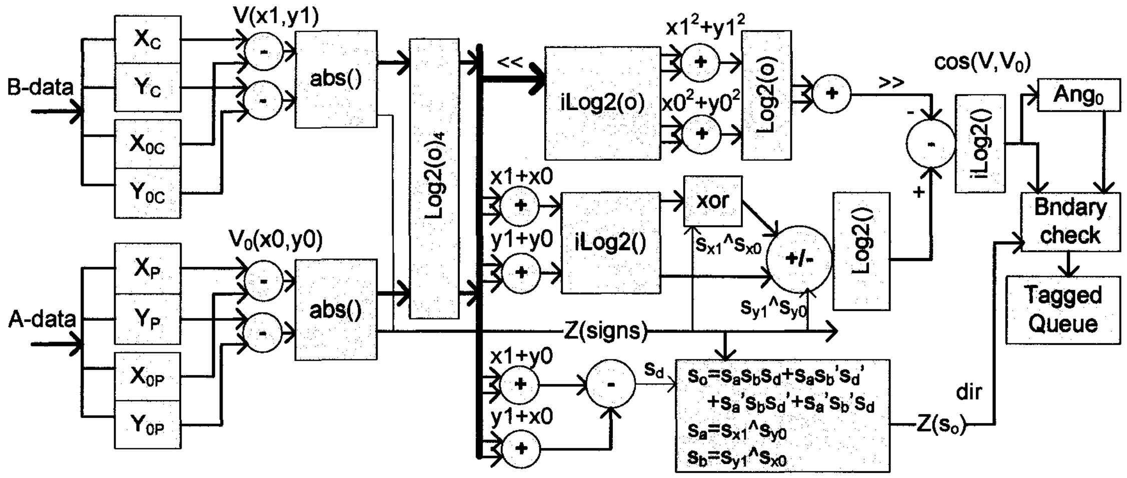


Figure 34: Architecture of inlier motion evaluation subsystem.

## 5.8 Affine Transformation

With given inlier motion, the final step is to generate the coordinate address of the stabilized video. From the display perspective of the output video, the XY counters generate the orderly coordinates of the expected display. Another coordinate to memory address translation is ignored in our design. The complete architecture is capable of rendering $1024 \times 1024$ frame size, thus, eliminating the need for such address translation given the XY coordinates are both perfect 10-bit numbers. The CORDIC rotated XY coordinates, therefore, are directly applicable to the address lines of the frame buffers as illustrated in Fig 35. The multiplication of K gain factor is needed although we implemented it with adders to a limited number of non-zero coefficients in K. With 10-bit address space, the precision of our logarithmic modules is currently not accurate enough to replace multiplication in this particular case. With the computed coordinate related to current frame, the RGB components can now be retrieved from the buffer for display. This concludes the mapping of display coordinate.

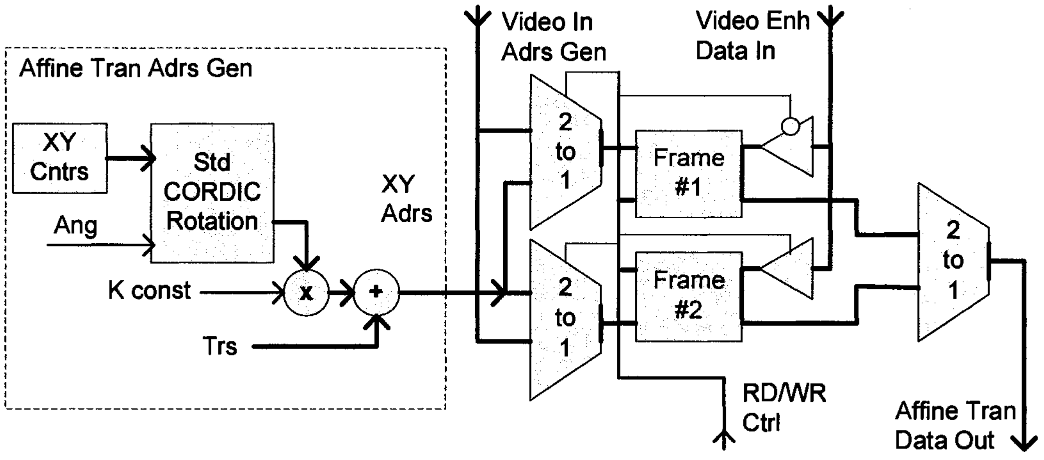


Figure 35: Affine transformation to map the display coordinates.

## 5.9 Summary

The design of various modules for video stabilization system was presented in this chapter along with appropriate discussion of further modifications to simplify the architectures. A thorough description and illustration of the logarithmic modules was provided along with the error correction to demonstrate how these modules could be applied to reduce the complexity of the design in other subsystems. A tentative discussion of video enhancement subsystem was also presented in section 5.3 to reveal necessary steps for generating the scenes with more uniform lighting. The design of single layer feature selection was also proposed with a modification which stored the corner-ness response in logarithmic scale to compress the word length and reduce storage space. The idea of designing NCC architecture as parallel filters was also illustrated along with the efficient design of the subcomponents such as energy normalizing factors, 2D integral sum and correlation. The architecture for correspondence management was also presented which served to regulate the dataflow between the computing module and its storage device. The module designed to evaluate the inlier motion was also discussed in which its main function was to calculate the angles between the vectors and save the coordinates with consistent motion. And finally, the subsystem for affine transformation was presented for which its main function was to generate the display coordinate through standard CORDIC rotation. This concludes the design of different subsystems. The subjects of simulation and performance related characteristics are discussed in chapter 6.

# CHAPTER 6

# RESULTS AND ANALYSIS

The chapter covers basic timing of various events which happen within each video frame to illustrate how different subsystems are synchronized to perform the tasks. Simulation results and error analysis are also discussed in detail for logarithmic modules, the video enhancement and stabilization subsystems along with the performance variables and the resource allocation. The power consumption of various resources is also presented based on Xilinx's XPower Estimator [76].

## 6.1 Timing Overview

Due to excessive duration of timing involved with the simulation, the sequence of events can be better illustrated in the following steps in the perspective of video frames as shown in Fig 36. The [1]initial latency of the system is 4 video lines to fill up the LBs. When the XY frame coordinate becomes (0,0), the [2]video enhancement subsystem starts kicking in to render the RGB components. Since the buffering on LBs is circular by design, the padding around the bothers contributes no useful information regarding the potential features and the feature measure[3]. However, the idle time of the boundary on the left is not necessarily a waste[4]. During these cycles, the search coordinates are fetched into the search table[5], thereof, freeing up the port access. At the bottom of the searchable range, 26 video lines are used for a couple events. A short duration of this period is used for evaluating the inlier motion[6]. Because the pf subsystem was design to capture sub-features column by column, most of the cycles are occupied to flush (pfFlush) the potential features into the NCC storage[7]. Near the end of the frame, the Affine transformation is performed by generating the stabilized coordinate and fetching the RGB components located in that memory location[8]. When the XY coordinate reaches the very last pixel of the frame, the above event flags are reset for cycling the states in next frame[9]. At the same time the frame toggle bit is performed to swap the current and previous state of the frame buffers and related storages[9]. In this manner, the absolute minimum frame buffers are required while still achieving reasonable stability in the stabilized video sequence.

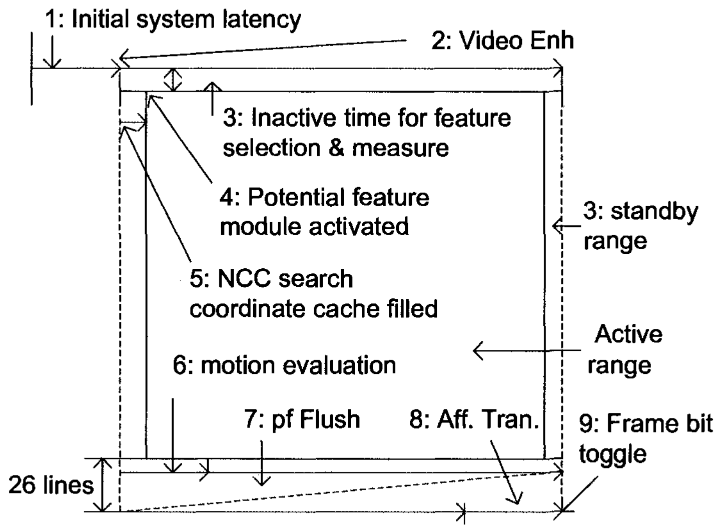

1: Initial system latency
2: Video Enh
3: Inactive time for feature selection & measure
4: Potential feature module activated
3: standby range
5: NCC search coordinate cache filled
Active range
6: motion evaluation
7: pf Flush
8: Aff. Tran.
9: Frame bit toggle
26 lines

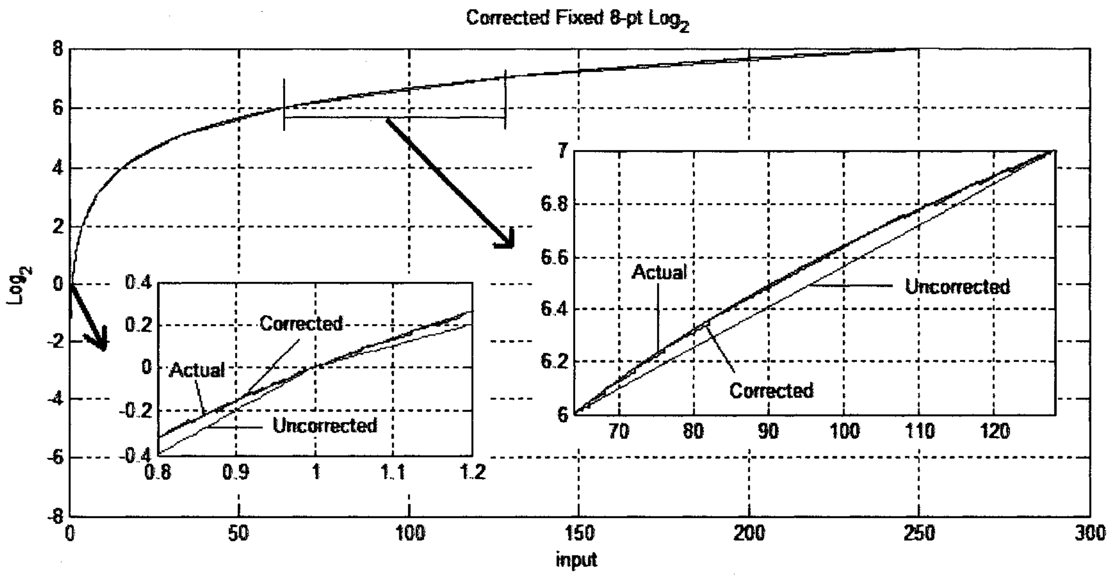
Figure 36: Timing events within the video frame.

## 6.2 Simulation And Error Analysis

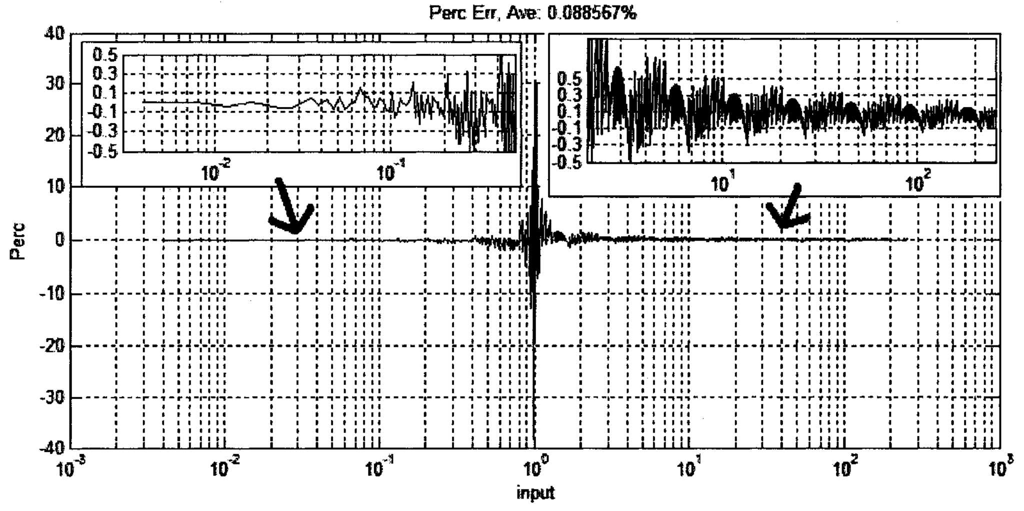
The results from simulation are discussed in this section along with the error analysis. The magnitude of error is measured based on the difference or the percentage deviated from the expected values of double precision.

### 6.2.1 Logarithmic Approximation

The fact that we heavily rely on the logarithmic modules to reduce complexity deserves a closer examination of the errors from such approximation and error correction. The double precision $\log_2$, uncorrected and corrected $\log_2$ approximations are plotted in Fig 37a for eight-bit fixed point decimal (8 bits integer and 8 bits fraction). The



(a)



(b)

Figure 37: Plot of double precision, uncorrected and corrected $\log_2$ calculations are shown in (a) with the percentage error in (b) for 8-bit fixed point decimals.

architecture with error correction produces $\log_2$ values very close to the double precision. The average percentage error over the entire range is 0.0886% (corresponds to average magnitude of difference error 0.0053) compared to 0.936% without correction as shown in Fig 37b. The measure of such error on a percentile basis is not very helpful although it may appear to increase the precision. A more meaningful illustration is to take only the difference error since the error cycle is periodic with constant peak around the mid-points of integer digits as illustrated in Fig 2 of chapter 3.

The comparison of difference errors on 16-bit integers is shown in Fig 38 with five other implementations. Michell's difference error serves as the reference for measure of improvement. SanGregory improved Mitchell's by dividing the error curve into two regions and used straight lines for correction which reduce the magnitude of error to 0.02 as oppose to 0.86. Abed's approach utilizes the same method for two, three, and six region corrections, however, minimizes the non-zero coefficients as the slope of the lines. Clearly, dividing the error curve into more linear regions has the trade-off of further complicating the correction mechanism. So there is a limit to the number of piecewise lines that bridges between simplicity and accuracy. Abed's six-region method seems more optimal with two adder arrays to reduce peak error to 0.013. On the other hand, both Combet's and Hall's achieve greater accuracy yet require complex circuits which defy the goal of approximation. The difference error of bit-level fitting is also shown in Fig 38. Our method generates a correction factor from 16 logic gates and reduces the error to 0.0177 with the majority in the range of -0.005 to 0.01. This mechanism is far more accurate than the two and three region methods and comparable to six region approach as shown in the last plot of Fig 38. In summary, Table 2 shows the number of bits involved for deriving the correction factor and its complexity. The last two columns summarize the range of error and the average magnitude of error for 16-bit integers. Clearly the bit-level curve fitting has the average error between three and six linear regions.

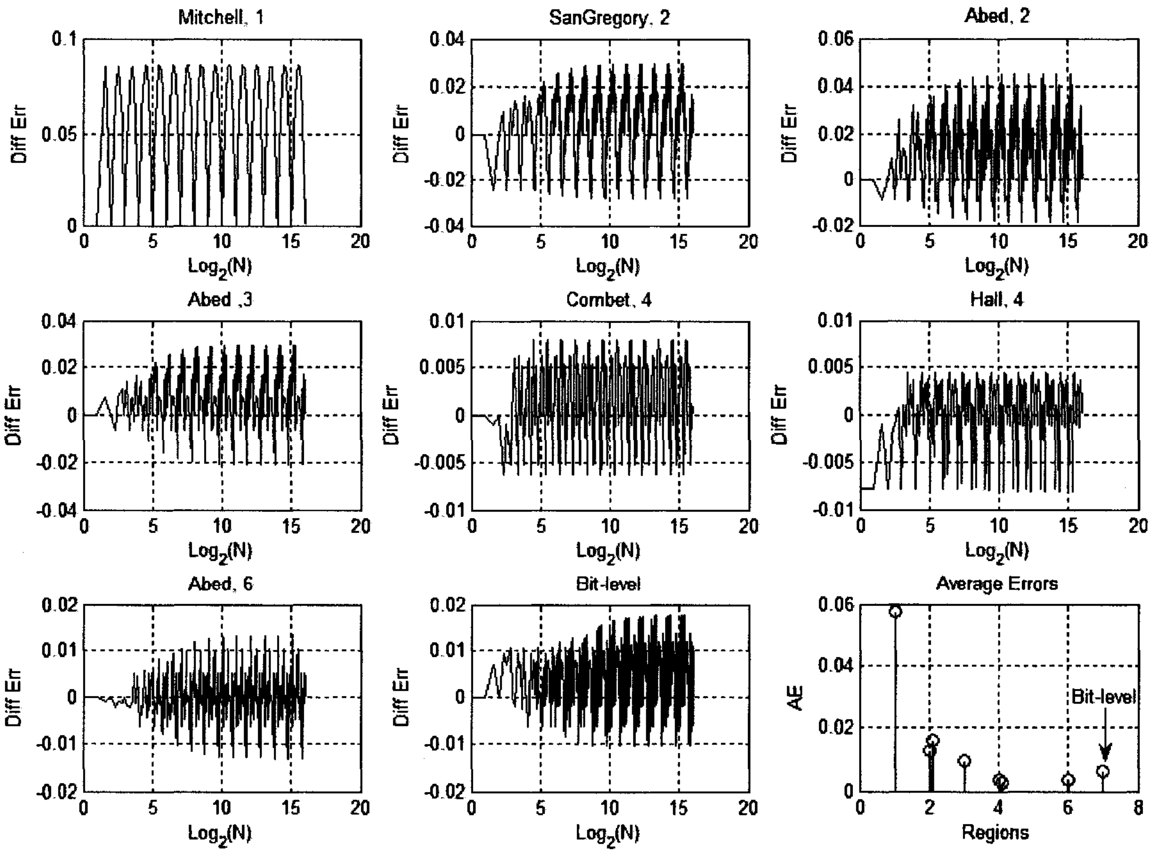


Figure 38: Comparison of difference errors with 5 other designs.

Abed's six-region method has superior average error at the cost of two adder arrays and a small number of logic gates for error correction. On the smaller scale for the actual usage of the logarithmic modules, his design has the advantage of providing more accurate results. For the implementations on larger scale such Corr2 architecture inside the NCC or the filter of the video enhancement, even the slightest improvement multiplies to minimize the resource. Given Abed's method it would require additional 243 8-bit adders with the correction logics for a $9 \times 27$ Corr2 kernel.

Table 2: Comparison of the error range and average magnitude with other designs.

| **Methods** | **Rgns** | **Correction factor** | **Corr. circuit** | **Error bound** | **Ave. mag. of error** |
|---|---|---|---|---|---|
| **Mitchell[48]** | 1 | none | none | $0 \le Err \le 0.0861$ | 0.0573 |
| **SanGregory[52]** | 2 | 3 bits | simple | $-0.0280 \le Err \le 0.0293$ | 0.0127 |
| **Abed[53]** | 2 | 2 bits | simple | $-0.0183 \le Err \le 0.0449$ | 0.0158 |
| **Abed[53]** | 3 | 3 bits | simple | $-0.0208 \le Err \le 0.0293$ | 0.0096 |
| **Combet[51]** | 4 | all bits | complex | $-0.0062 \le Err \le 0.0080$ | 0.0036 |
| **Hall[54]** | 4 | all bits | very complex | $-0.0082 \le Err \le 0.0044$ | 0.0024 |
| **Abed[53]** | 6 | 6 bits | simple | $-0.0130 \le Err \le 0.0132$ | 0.0033 |
| **Bit-level** | N/A | 3 bits | simple | $-0.0102 \le Err \le 0.0177$ | 0.0061 |

While the comparison of error for $iLog_2$ is not available in literature, it is a good practice to roughly sneak a peak over its range. Fig 39a illustrates the corrected $iLog_2$ has similar accuracy with Fig 37a for 8-bit signed numbers (4-bit integer and 8-bit fraction in 2's complement). It is more appropriate to grasp the magnitude of error in percentile for $iLog_2$ as the difference error exponentiates with the number itself. As shown in Fig 39b, the peak magnitude is bounded to 1.56% with 0.437% average magnitude of error. Apparently, the average error escalates by roughly five times due to the exponentiation. It should be noted that we use the same coefficient from $log_2$ correction. The range and the average magnitude of errors may be further reduced.

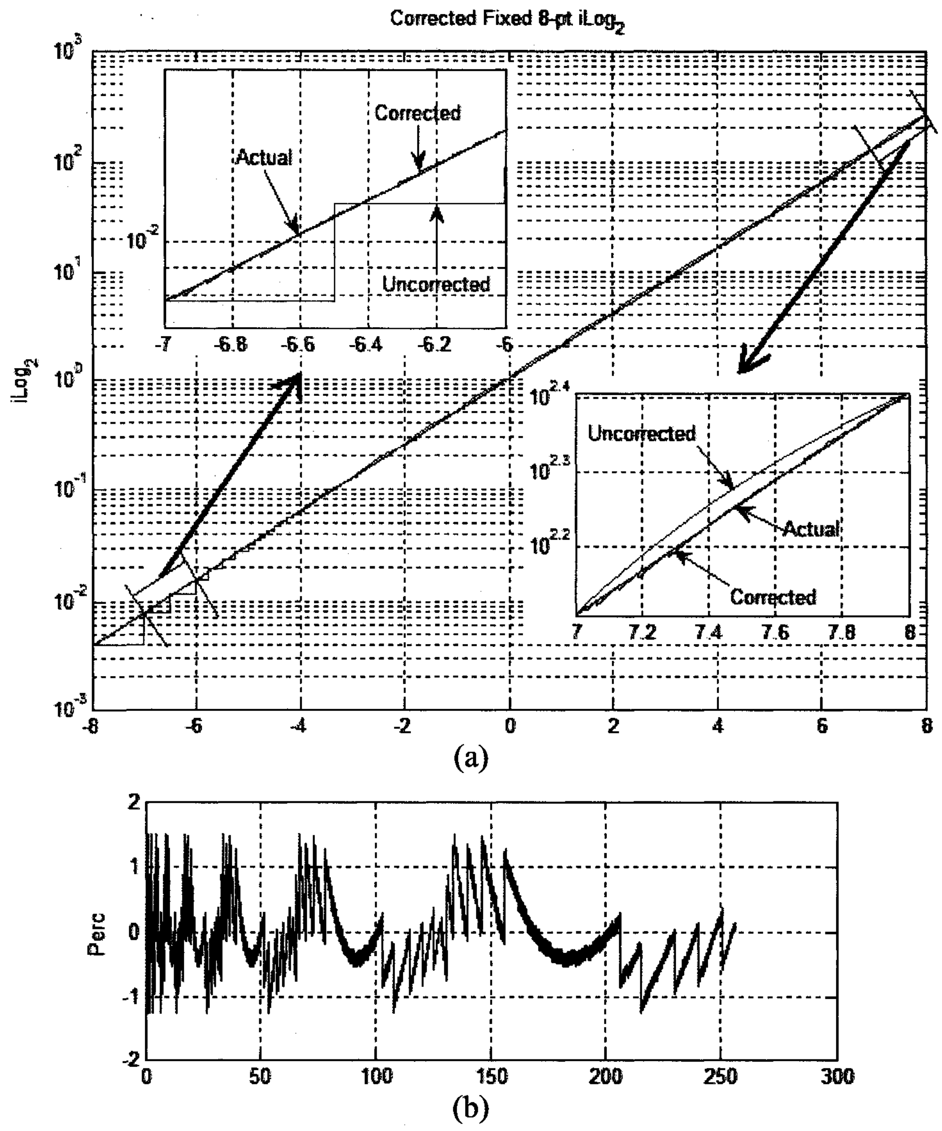


Figure 39: Plot of 8-bit fixed point (4-bit integer and 8-bit fraction in 2's complement) iLog$_2$ (a) and its percentage error (b).

## 6.2.2 Video Enhancement

The images are sent to the architecture pixel by pixel in raster scan fashion which is common for video streaming in progressive scanning mode. After the transient state, as indicated in Fig 36, the output becomes available and is collected for error analysis. The overall output of the enhancement architecture is recorded to give a graphic view of the enhanced image for quick evaluation of the visual quality. Typical test images are shown in Fig. 40 (1st row) where the shadow regions exist as the consequence of the saturation in bright regions. The outputs of the system produced by software and hardware simulations are illustrated on 2nd and 3rd rows, respectively. As one can see the majority of the details hidden in the dark regions are brought out while the natural color is preserved. The enhanced images produced by the hardware are slightly brighter than the ideal results. The difference is contributed by both logarithmic approximation and the limited bits representation in the architecture. Overall, the visual quality, in terms of brightness and contrast, is very satisfied with fewer shadow regions.

The error introduced from replacing equations (4.1.5)-(4.1.8) by (4.1.9) is shown in 1st row of Fig. 41 scaled by 50 times. The simplification induces a negligible magnitude of error at extremely dark regions of the images. Typical histograms of the error between ideal and hardware outputs from Fig. 40 (2nd and 3rd rows) are illustrated in 2nd row. The average errors of the system are 2.97, 2.61, and 3.79 pixel intensities with respect to the test images. Simulation with a large set of images shows a majority of the errors in this system is below 10 with the average error around 3.5. While the hardware simulation shows very attractive results, the efficiency of hardware utilization and its performance is also very important. This subject along with its performance on a desktop computer will be discussed in section 6.3.2. It may become trivial that there is no need for such architecture for small video frames.

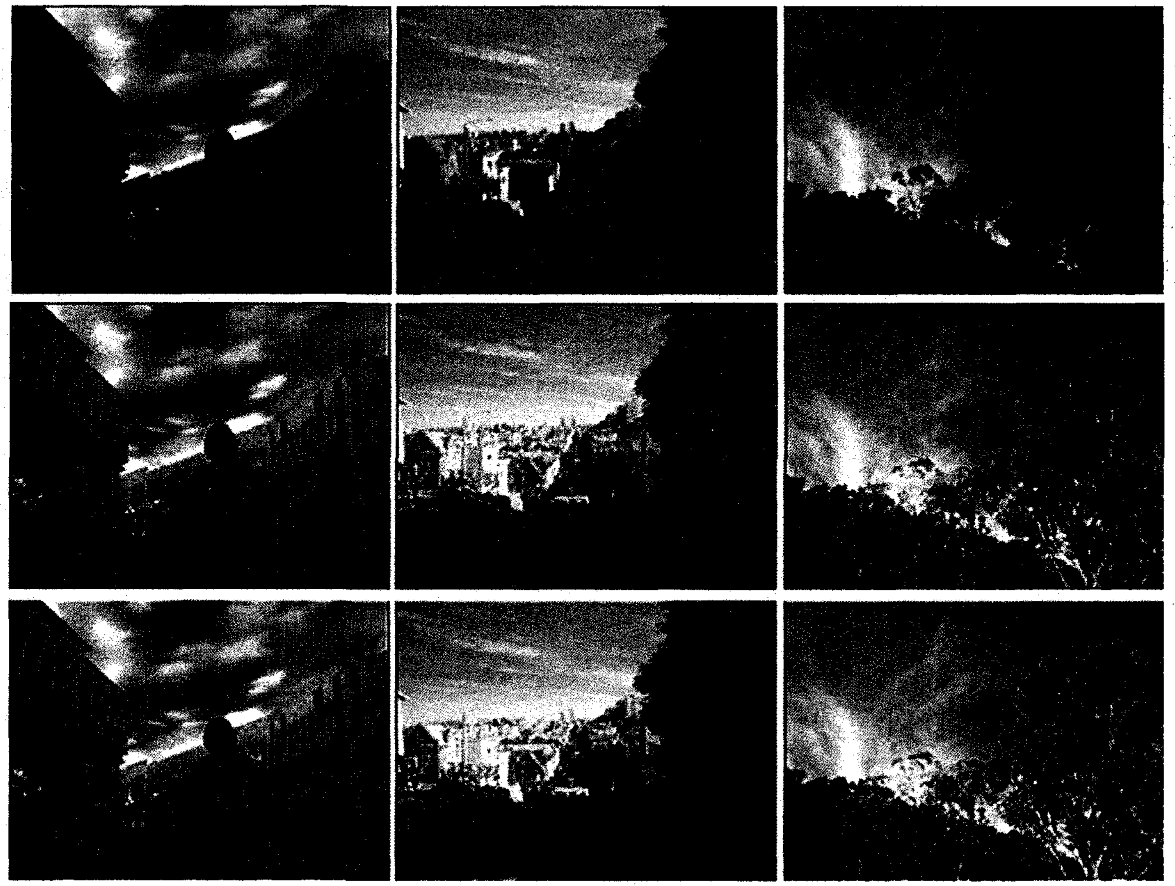


Figure 40: Images shown on 1st row are the test color images with non-uniform darkness. Results from software and hardware simulations are illustrated on 2nd and 3rd rows, respectively.

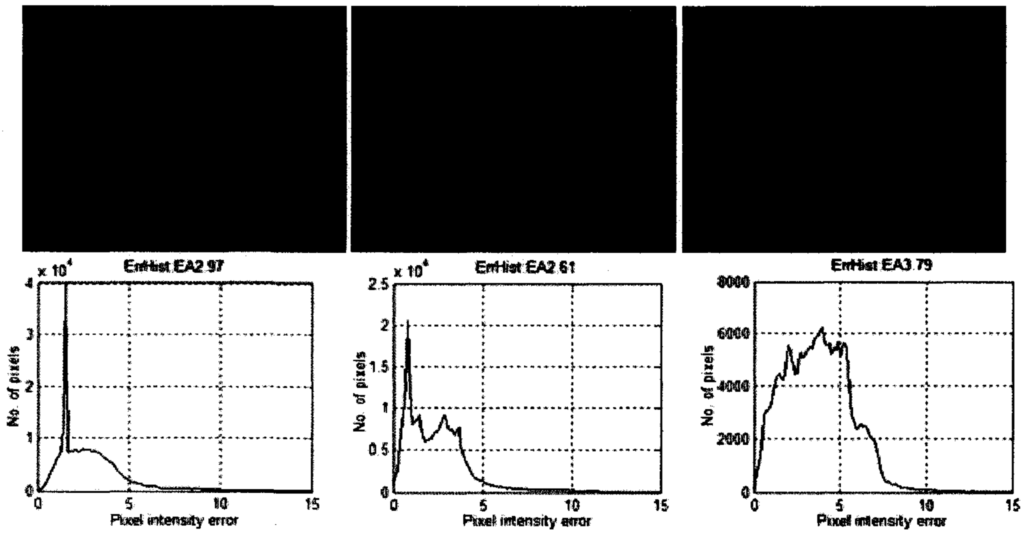
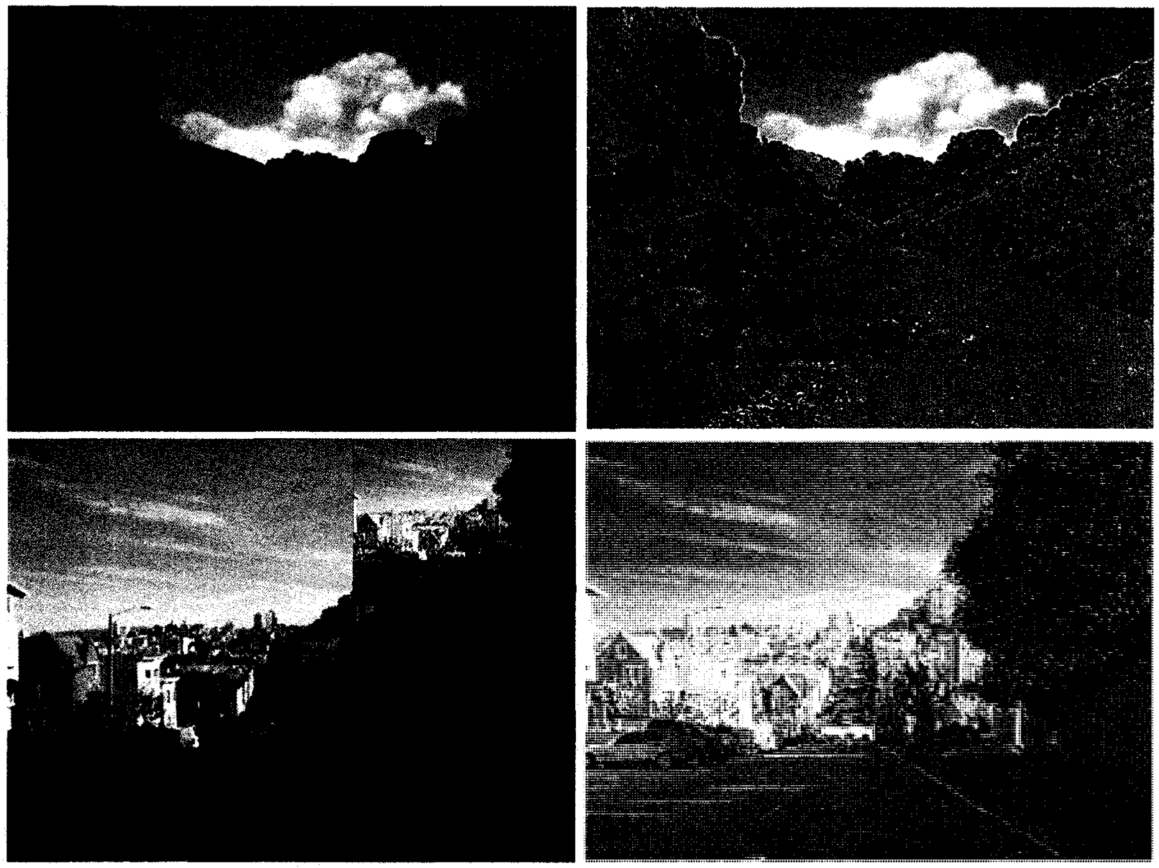


Figure 41: Error characteristics: The errors introduced for utilizing fast conversion factor are illustrated on the 1st row (50x). Error histograms are graphed on the 2nd row with average errors of 2.97, 2.61, and 3.79 pixel intensities.

### 6.2.2.1 Fine-tuning Transfer Function

While the design is very hardware efficient with decent performance as we should discuss in a moment, the biggest advantage, however, is not solely the impression of its colorful and uniform output of the images. Rather, it is the ability to fine-tune the filter coefficients suitable for different transfer functions so long as the functions have quadrant a symmetry property. More examples of output images are illustrated in Fig 42 with fine-tune on the luminance component alone, causing the scenes to be brighter than those in Fig 40.



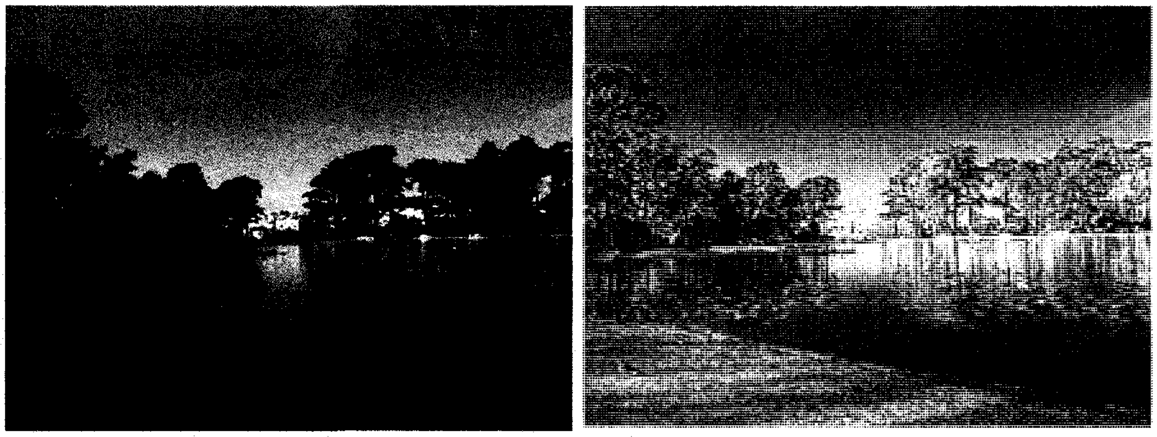


Figure 42: Examples showing the flexibility of fine-tuning the transfer function for visually more clear view of the scenes.

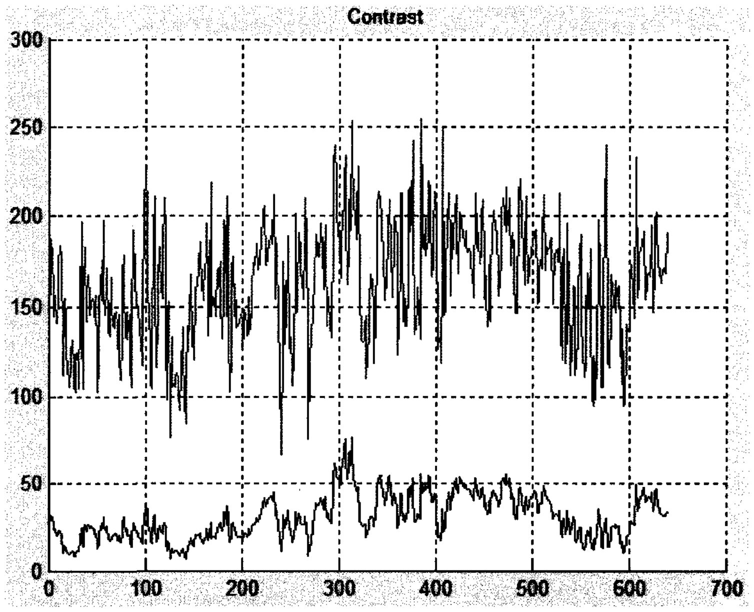


Figure 43: The kernel registers of the architecture can also be fine-tuned to enhance the contrast (sharpness) component of the image as illustrated.

### 6.2.3 Video Stabilization

Due to a tremendous amount of time involved with the simulation and debugging, only three initial frames were simulated in the process of obtaining the final results.

These frames are shown in Fig 44. Illustrated on the left column are the expected outputs from the algorithm. The results from hardware simulation are shown in the second column. The dark squares in the figure indicate the outcome from feature tracking with update from selection of potential features. The exact coordinates of potential features, however, is different from expectation. Due to the rounding from overflows of the architecture shown in Fig 24 and 25, more optimal choices were trimmed to the limited range. For this reason more features were selected and included for further processing. A majority of the stable features being measure, however, agreed with the results produced on C++ version of the algorithm. Notice the slight difference shown (last row of Fig 44) in the stabilized sequence from hardware simulation. The black strip was off by a few pixels compared to the expected image on the left column. The error is common since the coordinate of memory address generated by affine transformation was truncated to integers, resembling the effect of standard nearest neighbor interpolation. Although this distortion is not visually trivial from the test frames, it is expected to increase the severity directly proportional to the angle of rotation from reference frame. A better method is to apply bilinear or bicubic interpolation to minimize the distortion. The average errors of a potential feature selection and the feature measure are shown in Table 3. The measure of error was performed with respect to the resulting coordinates from the simulation. For an example of the feature selection, a direct comparison from the coordinate chosen by the algorithm does not help due to the limitation within the architecture itself. These sources of error can be tolerable as the outcome of the stabilized video mainly depends on the video enhancement and the precision of affine transform.

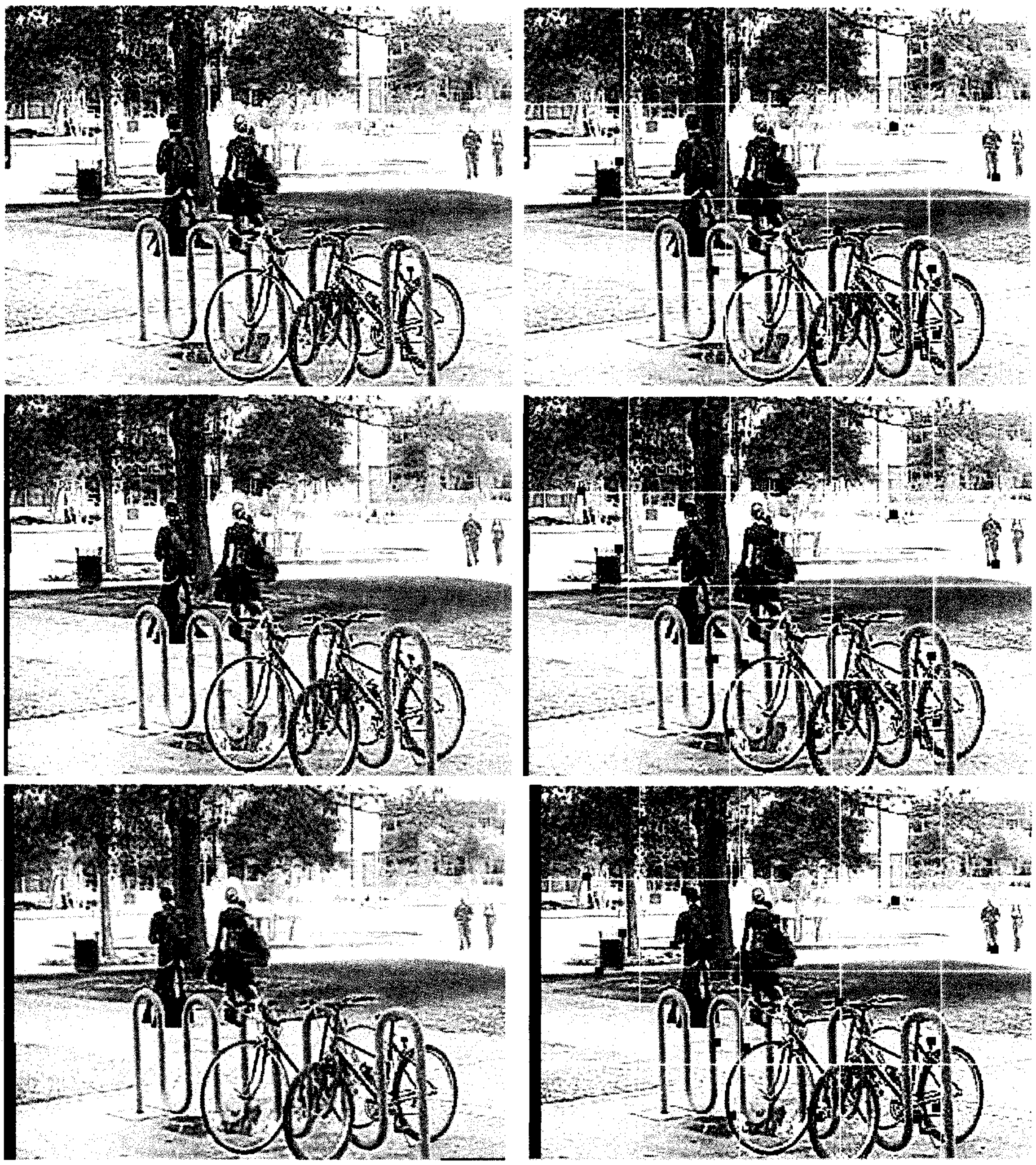


Figure 44: A short sequence of stabilized frames from the algorithm (1st column) and the hardware simulation (2nd column).

Table 3: Average errors of feature selection and measure subsystems.

| Frame Error | Feature Selection | Feature Measure |
|---|---|---|
| #1 | -0.4% | -- |
| #2 | -0.5% | 3.1% |
| #3 | -0.6% | 1.1% |

Ideally, a more concise magnitude of error should be compared between the improved NCC architecture and the design proposed in [68]. Unfortunately there was a design error in the earlier publication. The block diagram of the earlier design shown in Fig 45a contains an error in which the input 8-bit data, *f*, was subtracted from average $\bar{f}$ but excessively delayed through another set of internal LBs in block 3. This creates the impact of a moving average $\bar{f}$ for every set of *f* values within the 2D window (kernel). The correct implementation, as illustrated in Fig 45b, should have been the fixed $\bar{f}$ for every set of *f* values under the kernel without any LBs in block 3. And the subtraction should be performed inside block 3 right before the Corr2 operation. The implication with reference to the current design ($U \times V = 9 \times 27$ kernel dimension), is that another set of 243 $Log_2$ and $iLog_2$ modules are necessary, due to data dependency, to operate in full 2D processing bandwidth for correct calculation. The correct implementation may increase the resource by 60%. Although the 8.7% average error seemed reasonable for logarithmic modules without correction, the peak error could be as high as 62% of the expected score. The architecture in Fig 45a still produced the correct coordinates. But it does not mean that the NCC score is exactly right, aside from approximation error of logarithmic modules. The implementation is clearly equivalent to (6.1) rather than the true normal correlation of (4.2.9). The reason that an earlier design still produces correct coordinates is that moving average tends to change relatively slow.

$$\phi(x,y) = \frac{\sum_{u,v}\left[\left[f(x+u,y+v) - \bar{f}_{u,v}(x+u,y+v)\right] \times \left[t(u,v) - \bar{t}_{u,v}\right]\right]}{\left[\sum_{u,v}\left[f(x+u,y+v) - \bar{f}_{u,v}(x+u,y+v)\right]^2 \sum_{u,v}\left[t(u,v) - \bar{t}_{u,v}\right]^2\right]^{1/2}} \tag{6.1}$$

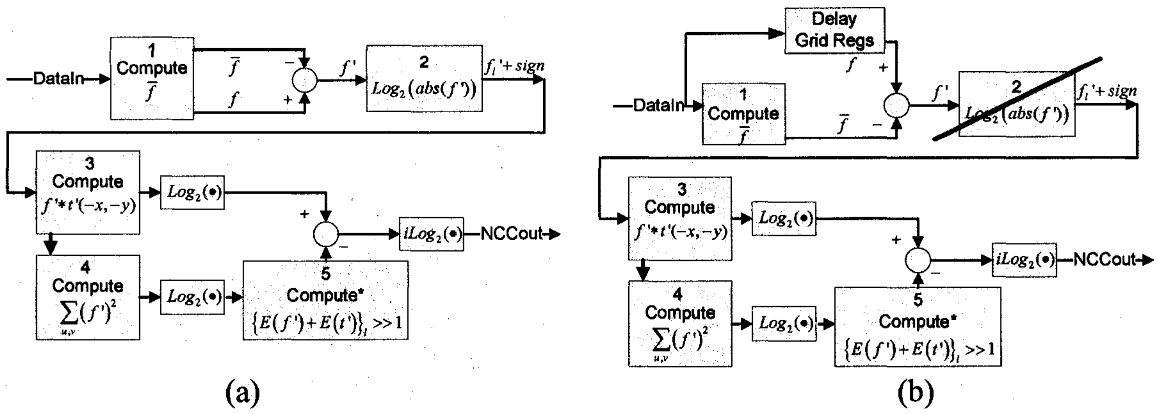


Figure 45: Design error in earlier implementation of NCC architecture.

## 6.3 Performance Analysis and Resource Utilization

The performance and resource allocation of various subsystems are presented in this section along with discussion. The performance and the hardware resource parameters are mainly characterized on Xilinx's Virtex II FPGA technology.

### 6.3.1 Log$_2$/iLog$_2$ Modules

The performance of logarithmic modules improves with one additional pipeline stage compared to earlier implementations. On Xilinx's Virtex II 2V2000FF896-4 FPGA, the performance of error corrected fully pipelined log$_2$ and iLog$_2$ modules can produce the throughput of 203.6 and 304.6 million outputs per second (MOPS) for 8-bit format, respectively. For the 32-bit numbers, both modules are able to sustain above 200 MOPS data rate as shown in Tables 4 and 5. The performance of log$_2$ doubles (203.6 vs. 100 MOPS) for 32-bit format at the expense of 31 bits register. The modules with improved precision utilize a greater number of logic slices (LSs) and lockup tables (LUTs) than the designs without correction. The simultaneous reduction in resource and precision gain become obvious in 32-bit numbers. A better figure of such resource reduction may be obtained at transistor (VLSI) level of implementation.

Table 4: Performance and resource utilization for $\text{Log}_2$ module.

| **Description** | **Resolution (Correction/No Correction)** | | |
|---|---|---|---|
| | **8** | **16** | **32** |
| **Logic Slices** | 22/11 | 48/43 | 94/166 |
| **LUTs** | 32/19 | 74/76 | 161/289 |
| **$F_{max}$(MHz)** | 203.6/205 | 203.6/121.5 | 203.6/100 |

Table 5: Performance and resource utilization for $\text{iLog}_2$ module.

| **Description** | **Resolution (Correction/No Correction)** | | |
|---|---|---|---|
| | **8** | **16** | **32** |
| **CLB Slices** | 19/12 | 57/44 | 155/164 |
| **LUTs** | 30/19 | 71/70 | 151/268 |
| **$F_{max}$(MHz)** | 304.6/305.6 | 234.5/235.4 | 212.2/212.2 |

### 6.3.2 Comparison of Video Enhancement Architectures

Due to the comparison made to earlier designs, the hardware resource utilization is characterized based on the Xilinx's multimedia platform with Virtex II XC2V2000-4ff896 FPGA and the Integrated Software Environment (ISE) [70], [71]. The particular FPGA chip we targeted has 10,752 logic slices, 21,504 flip-flops (FFs), 21,504 lookup tables (4-input LUTs), 56 block RAMs (BRAMs), and 56 embedded 18-bit signed multipliers in hardware; however, we do not utilize the built-in multipliers. The resource allocation for various sizes of the kernels in homomorphic filter is shown in Table 6. For $9 \times 9$ kernels in homomorphic filter, the computational power is approximately 81 multipliers which is significantly less compare to [19] with similar setting where 243 multipliers and 150 dividers are needed if a conventional approach is taken. With the alternative approach introduced in this design and the concept of log-domain computation, the amount of hardware resource necessary for the implementation is tremendously reduced. The maximum windows can be utilized on target FPGA consumes 85% of the logic slices (4 slices is equivalent to 1 configurable logic block), 51% of the FFs, 49% of LUTs and 22 BRAMs. Larger kernels are not necessary in practice. Testing conducted in section 6.2.2 shows that a $5 \times 5$ filter kernel is sufficient to remove most shadows of reasonable darkness. Only 13% of the logic slices is needed in this case. The proposed design uses approximately 71.7% and 73.6% (does not include embedded multipliers used in [20]) less logic slices with a great performance boost compared to the architectures presented in [19] and [20] ($1024 \times 1024$ frame size), respectively.

Table 6: Hardware resource utilization for various sizes of the kernels with corresponding throughput rate.

| **Kernel Size** | **Logic Slices** | **Slice FFs** | **LUTs** | **BRAMs** | **Perf (MOPS)** |
|---|---|---|---|---|---|
| $5 \times 5$ | 13% | 8% | 7% | 6 | 182.65 |
| $9 \times 9$ | 30% | 18% | 17% | 11 | 182.65 |
| $13 \times 13$ | 53% | 32% | 31% | 16 | 182.65 |
| $17 \times 17$ | 85% | 51% | 49% | 22 | 182.65 |

The critical timing analysis of Xilinx's ISE shows that 182.65 MOPS is the most optimal throughput achievable with the maximum clock frequency of 182.65 MHz on Xilinx's Virtex II technology. Further evaluation of pipelining the critical path suggests that increasing the level of pipeline does not gain significant throughput rate. This directly indicates the impact of the design with tightly coupled and well pipelined system. Given $1024 \times 1024$ image frame, it can process over 174.2 frames per second at its peak performance without frame buffering, which is very suitable for video streaming applications. This tremendous gain in the performance while consuming significantly less hardware resources would have been extremely difficult to achieve without the algorithmic simplification, efficient filter design and log-domain computation. The additional benefit is that the filter coefficients are not hardwired, which gives the highest flexibility in reloading the coefficients without the need of dynamic reconfiguration for different characteristics of the transfer functions. The performance of the proposed approach increases to 124% and 273% when compared to the designs we presented in [19] and [20] ($1024 \times 1024$ frame size), respectively. Due to massive parallelism, it is also far superior to those DSP based approaches discussed in [16], [20], [72] which utilize a limited number of functional units. A comparison of the proposed work with other implementations most relevant to the model is listed in Table 7. While the throughput of the FPGA based architectures significantly out performs those of DSP processors (by more than 80 times), it should be pointed out that the DSP processors are largely constraint to the available functional units with associated resource. For instance, the FFT/IFFT operations are accomplished through reuse of the fixed N (N samples are padded to power of two prerequisite to use the FFT/IFFT) points FFT/IFFT blocks where the fully parallel-pipelined architectures could consume more than two high-end highly dense FPGAs such as Xilinx's Virtex II Pro 70 with 33,088 logic slices and an enormous number of embedded RAMs and multipliers [73]. So the full level parallelism cannot be exploited. For this reason, the processors usually operate at higher clock frequency to achieve minimum real-time criteria with limited video resolution. The operating frequency to throughput ratio can be more than two orders of magnitudes (i.e. 100 clocks/pixel). For standard NTSC ($720 \times 480$ at 30 fps) video with the algorithms fully exploited, the DSP processors need to operate at GHz scale where the memory access of the systems becomes the bottleneck without the assistance of improved memory

Table 7: Comparison of the proposed work with other implementations most relevant to reflectance-illuminace model. Note that $256 \times 256$ frame size (should be power of 2) is employed so the performance is not penalized for [16] and [72] to utilize FFT and IFFT.

| **Hardware Platforms** | **Operating Frequency** | **Nature of Design** | **Resource Utilization** | **Frame Buffers** | **Throughput Rate** | **Frame Rate** |
|---|---|---|---|---|---|---|
| **FPGA[1][20]: XC2V2000** | 67MHz | Systolic-parallel | 49.3% logic slices[8] | Ext. 133MHz ZBTRAMs | 1ppc[5] 67mpps[6] | 1022 fps[7] |
| **FPGA[1][19]: XC2V2000** | 147.3MHz | Systolic-parallel | 46% logic slices[9] | None | 1ppc[5] 147mpps[6] | 2248 fps[7] |
| **DSP[2][16]: C6711** | 150MHz | VLIW (256-bit) | DSP+DSK[3] support | Ext. 100MHz SDRAMs | 0.009ppc[5] 1.36mpps[6] | 20.7 fps[7] |
| **DSP[2][16]: C6713** | 225MHz | VLIW (256-bit) | DSP+DSK[3] support | Ext. 90MHz SDRAMs | 0.008ppc[5] 1.84mpps[6] | 28 fps[7] |
| **DSP[2][72]: DM642** | 600MHz | VLIW (256-bit) | DSP+EVM[4] support | Ext. 133MHz SDRAMs | 0.004ppc[5] 2.24mpps[6] | 34.1 fps[7] |
| **FPGA[1, 11, 12]: XC2V2000** | 182.65MHz | Systolic-parallel | 13% logic slices[9, 10] | None | 1ppc[5] 182mpps[6] | 2787 fps[7] |

**Notes:** [1]Xilinx's Virtex II XC2V2000-4ff896 FPGA on multimedia platform [70], [71].
[2]Texas Instruments' DSPs in TMS320 family with appropriate platforms and 2 levels cache support.
[3]SDK: TI's platform supporting C6711 and C6713 DSP chips [16].
[4]EVM: TI's platform supporting DM642 DSP chip [72].
[5]Ppc is unit for pixels per processor clock.
[6]Mpps is unit for million pixels per second, equivalent to MOPS.
[7]Fps is unit for frames per second.
[8]Use of all embedded multipliers not included.
[9]Architecture is multiplier-less.
[10]If multiplier-less architecture (also utilizes logarithmic modules) for color space conversions (RGB to HSV, and HSV to RGB [47]) is fully implemented in the design, it consumes additional 108 logic slices and 430 LUTs.
[11]Proposed work in this paper utilizes same technology as [19] and [20].
[12]The C++ executable version can sustain 29.85 fps (with the frame size of $360 \times 240$ or quarter NTSC) on the laptop with Intel CPU P4H@3.2GHz, 1.5GB DDR1 memory, IEEE1943 firewire, and 38% CPU load as opposed to 26 fps with 98% CPU load when fast color space conversion in (4.1.11) is not incorporated. The design is also at least 70 times faster than the software version.

management [72]. Despite the drawback, DSP based implementations are still well adapted for lower end applications (low pixel volume) where the performance of the systems is not critical. The performance of [20] was limited to uneven pipelining (Unregistered arithmetic operations are followed by high precision multipliers.) and the setup of external Zero Bus Turnaround (ZBT) RAMs which are coupled with the core module. This bottleneck does not impose on the proposed architecture since the throughput is sufficient to enhance the video on the fly at the constant rate as the streamed video in progressive scanning mode. Overall, the new design achieved similar output quality with reduced hardware resource while boosting the performance.

### 6.3.3 Video Stabilization Subsystems

Xilinx's Virtex II 2V2000 series platform has insufficient resource for our system. The hardware resource and the performance parameters were recorded based on 2V8000 chip with a speed grade -5C which is slightly faster than the platform discussed in section 6.3.2. The resource allocation to different subsystems is listed in Table 8. As expected the feature measure 'NCC$9 \times 27$' consumes a majority of the available CLB slices (44%), flip-flops (22%, LUTs (24%) and BRAMs (16%). Even though such architecture is resource friendly, the tremendous number of adders given a $9 \times 27$ sub-feature dimension demands extremely high volume of logic elements. The remainder of subsystems utilizes only 6% of the LUTs and 10% LSs.

A number of performance parameters of subsystems is also listed in Table 8. These components have a very high throughput rate, too excessive for conventional video applications. The overall system performance depends on the slowest modules in Table 8. It is interesting to note the performance of each module is mainly affected by the number of bits in the adders and subtractors. The feature selection subsystem has a throughput rate of 180.9 MOPS with its internal 28 bit arithmetic logic units. We should also point out the performance gain of a video enhancement subsystem is not the result of further improvement in pipelining; rather, it is the technology of selected FPGA (speed grade 4 vs. 5). Thus, one should not mistakenly treat the technological parameter with the advancing of architecture itself. With the simplification to memory address translation, the system was really designed for $1024 \times 1024$ video frames. At the peak performance (one output per cycle), the processing power is equivalent to 172 fps. There is no need for such bandwidth in current applications. The resource and excessive bandwidth should be traded for future improvements to suit the design according to the nature of specific applications [69].

Table 8: Resource allocation and the performance of subsystems.

| **Components** | **Resource** | | | | **Performance (MHz)** |
|---|---|---|---|---|---|
| | **Slices** | **Flip-Flops** | **LUTs** | **BRAMs** | |
| **Line Buffers** | 0% | 0% | 0% | 6% | 267.2 |
| **Video Enh.** | 3% | 2% | 2% | 0% | 210.1 |
| **Feat. Sel.** | 5% | 2% | 3% | 11% | 180.9 |
| **NCC9$\times$27** | 44% | 22% | 24% | 16% | 199.8 |
| **Corr. Mgmt** | 0% | 0% | 0% | 0% | 260.4 |
| **Mot. Eval** | 2% | 0% | 1% | 0% | 212.9 |
| **Aff. Trans.** | 0% | 0% | 0% | 0% | 192.6 |
| **System[1]:** | 59% | 30% | 34% | 35% | 180.9 |

**Notes:** [1]Two frame buffers with ZBT RAMs not included.

## 6.4 Scalability of the System

The system produces constant throughput of one output per clock cycle. Decomposition of full features into the constellation makes the one-on-one data rate possible without the classic method of trading the performance for reduction of hardware resource in the NCC subsystem. The feature measure is also flexible in terms of scalability of the sub-features since the measure of secondary sub-features is only activated at the fixed spatial locality with successful detection of primary sub-feature. Interestingly, the parameters of performance on clock basis and resource utilization have the least to do with the dimension of the frame in our subsystems since the criterion to operate the system is dependant on the parallel data provided by the LBs. For the system to operate on a larger video frame, only the length of LBs needs to be increased along with the sufficient frame buffers; however, the frame rate will be reduced. Depending on the layout of frame buffers, it may become necessary to construct the coordinate to memory address translators. Future development over ultra high frame resolution should consider down-sampling the image to reduce search range and map the coordinate back to the original image (short range course to fine search mechanism). It does not necessarily require frame buffers for the down-sampled images.

### 6.4.1 Feature Measure

As we know the feature measure does not scale well with conventional implementation. To put the performance-resource trade-off into perspective, Table 9 illustrates certain requirements for the calculation over a single $128 \times 128$ feature. A common technique of trading the performance has an inverse relationship for reduction of hardware resource. Given a $64\times 128$ time-multiplexed support architecture, it requires two clock cycles ($64\times 128\times 2$ cycles$\rightarrow 128\times 128$) to produce one complete output with the gain of reducing the computing elements by half (8192). To further reduce the resource by eight, a $16\times 128$ support architecture is needed with the throughput rate of one output per eight clock cycles. To sustain the performance while utilizing minimum resources, the design eventually becomes technology dependent in the sense that the core engine has to run at a much higher frequency. The sub-feature representation on the other hand, has the advantage of reducing the number of calculations at the same time without compromising its performance. A $128\times 128$ feature represented by 8 $16\times 128$ sub-features requires the same amount of computing elements as a $16\times 128$ time-multiplexed architecture. However, the throughput rate remains one per clock cycle. The $16\times 128$ support architecture computes on a $16\times 128$ primary sub-feature which contains the most

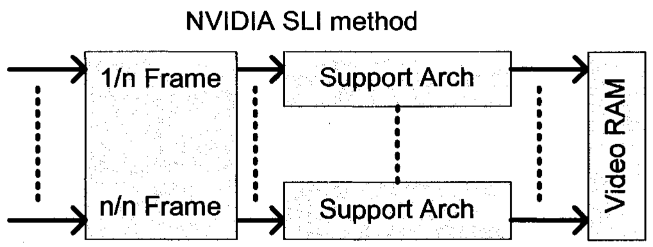
Table 9: Comparison of conventional method and sub-feature representation.

| **Feature Size** | **Perf. Per Cycle** | **Support Arch.** | **Processing Elements** | **Storage Capacity** | **Resource Req.** |
|---|---|---|---|---|---|
| $128\times 128$ **Time-MUXed** | 1/2 | $64\times 128$ | 8192 | 16384 | 16384 adders 8192 $\mathrm{iLog_2s}$ |
| $128\times 128$ **Time-MUXed** | 1/4 | $32\times 128$ | 4096 | 16384 | 8192 adders 4096 $\mathrm{iLog_2s}$ |
| $128\times 128$ **Time-MUXed** | 1/8 | $16\times 128$ | 2048 | 16384 | 4096 adders 2048 $\mathrm{iLog_2s}$ |
| $128\times 128$ **Sub-features** sf(8): $16\times 128$ | 1 | $16\times 128$ | 2048 | 16384 | 4096 adders 2048 $\mathrm{iLog_2s}$ |
| $128\times 128$ **Sub-features** sf(8): $9\times 27$ | 1 | $9\times 27$ | 243 | 1944 | 486 adders 243 $\mathrm{iLog_2s}$ |

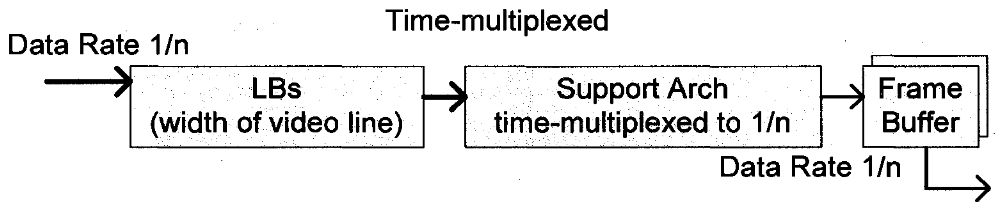
distinct characteristics of all other sub-features. With the known structure of the constellation, the success of primary sub-feature enables the measure of a set of secondary sub-features with predetermined spatial locality. In this fashion, the computing power is dramatically minimized over the sub-features. With the structure formed by the sub-features, it is not necessary to allocate the storage for the complete feature.

### 6.4.2 Frame Size and Rate

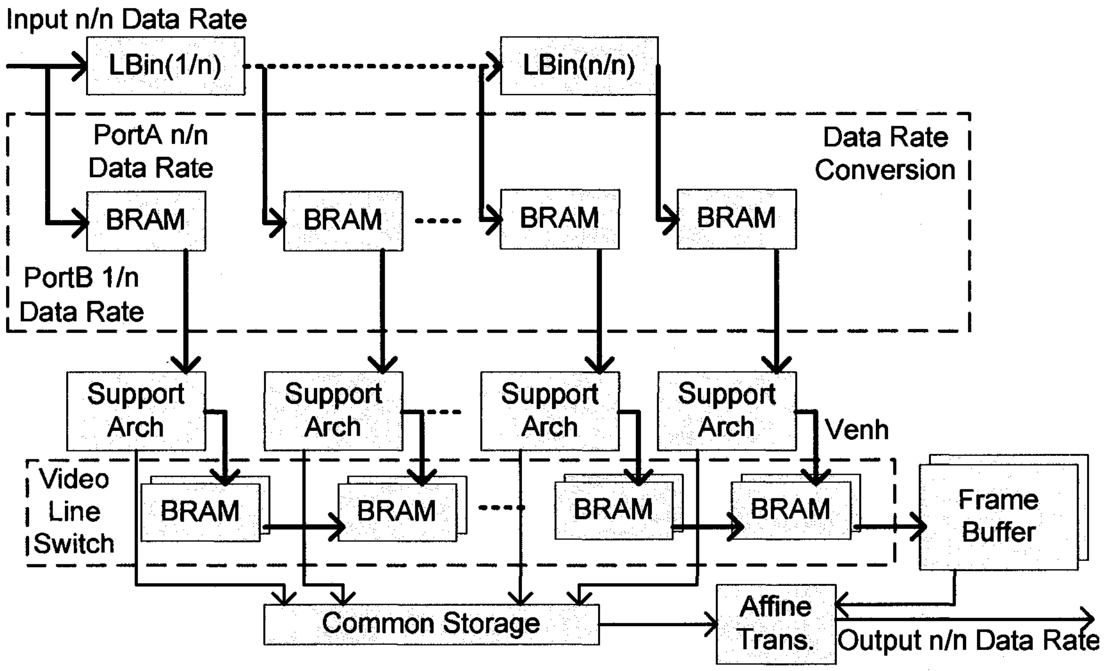
The frame rate and dimension essentially translate to the pixel rate. To achieve the same frame rate given a different frame size, the support architecture must compensate its performance by either increasing or reducing the pixel rate on clock basis. With NVIDIA SLI graphics technology, the larger frame can be divided into upper and lower sub-frames. This mechanism allows rendering of graphic contents at the same pixel rate with two graphic cards operating in parallel. The end result is a system which sustains a relatively steady frame rate at twice the frame size. This method assumes the concurrent input video stream is available. The direct application of NVIDIA SLI technology to our n-sub-frame architecture of Fig 46a can achieve a similar goal for upscale of the frame size; however, it would require a half-frame buffer space to create two concurrent streams. Although the horizontal frame division solves the problem of scaling the y-dimension in our design, the system cannot compensate the change in x-dimension since the pixel rate remains constant. This argument brings out the scalability of current design for adapting the performance to different pixel rates.



(a) NVIDIA SLI approach with full frame divided into even blocks.


Time-multiplexed
Data Rate 1/n
LBs (width of video line)
Support Arch time-multiplexed to 1/n
Frame Buffer
Data Rate 1/n

(b): Frame size downscale: Time-multiplex rendering with input/output pixel rate at 1/n of internal operating clock.



(c) Frame size upscale: Increase in the dimension of video frame requires higher pixel rate to sustain same frame rate.

Figure 46: Scalability of frame size and rate: (a) NVIDIA method to increase the size with constant rate, (b) reduced size or rate with multiplexed architecture to cut back resource, (c) larger size or higher rate with de-multiplexed support to obtain proper data rate from the stream video and sustain same frame performance (c).

Suppose we have a stream video with smaller frame size (number of pixel elements decreased by a factor of 1/n) with respect to our system, the pixel rate will drop to 1/n given a constant frame rate. While the system clock may be slowed down to minimize power dissipation and match the pixel rate, the conventional method of time-multiplexing can be applied for performance-resource optimization as illustrated in Fig 46b. The main advantage of time-multiplexed architecture over the current design can reduce the number of processing elements and the adders of Corr2 architecture shown in Fig 31 by 1/n.

While the system clock may be adjusted accordingly with reduced frame size (or rate), the design modification for supporting larger frames is a bit different. The scalability of the current system can be illustrated by Fig 46c to support a higher pixel rate (bigger frame size at constant frame rate or vise versa). In order to produce a higher pixel rate with multiple support units operating at lower frequency, the input sample rate must be converted to the internal frequency of the support architecture. The conversion can be accomplished by capturing the data (propagated through LBin(.) at n/n sample rate) into a dual port BRAMs at specific cycles. Suppose that a single line of video is buffered in the LBin storage space at the input/output video rate, the capture cycle activates at only 1/n of the cycles per video line. The conversion allows a higher rate to be de-multiplexed into a lower frequency which cannot be achieved by manipulating the video lines in NVIDIA SLI scheme (constant pixel rate). As a result, the entire frame is divided into a number of vertical blocks with the BRAMs' 'PortA' operating at n/n sample rate and 'PortB' at 1/n internal speed. In order to convert the outputs back to n/n data rate, the results must be pre-assembled into the video line in its entirety. Unlike the data rate conversion from the frontend, the outputs cannot keep up with the moving video line at n/n rate as doing so will discard (n-1)/n of the entire line of pixels with duplicates of adjacent values. A video line switch with two LBs can be incorporated to solve the problem with a difference in data rates. While one LB assembles the results from multiple support units, another LB is free to stream out the previous video line. The affine transformation subsystem should be capable of operating at n/n rate with two frame buffers to produce display coordinates. If technology does not permit such performance for coordinate transformation, the same concept can be applied to convert the data rates and merge the results. The hardware overhead to operate multiple support units in parallel is the additional buffer space of four video lines.

### 6.4.3 Technological Advancement

The advancement in technology results with devices which improve the attributes of silicon area, performance and power. We focus on performance parameter of the support platform. With FPGAs (such as Xilinx's Virtex-5 family) capable of operating at a higher frequency, the immediate benefit is a large increase of pixel rate given the same system. This gain directly translates to a higher frame rate with constant frame size, or larger frame size with fixed frame rate. While technological improvement, frame rate and dimension appear to be separate subjects of scalability, they can be characterized by a single parameter of pixel rate on clock basis. Given the specific constraints of technology, frame size, and frame rate with respect to the nature of applications, the system can be optimized by time-multiplexing to reduce hardware resource for smaller frame size or rate with better technology. Conversely, multiple units can be deployed to operate in parallel proportional to a larger frame size or higher frame rate with technical limitation as illustrated in Fig 46. The ability to support a higher performance system with low speed design depends on the conversion of sample rates between the input/output interface and its internal frequency of the computing cores. Hence, the scalability of the system with respect to the technology is also linear. For the implementation on VLSI level, it is likely to enhance the system's performance since the routing overhead can be independent of specific FPGA technology and multiple nodes (i.e. in adder tree) can be pumped through within a pipeline stage.

## 6.5 Power Consumption

The power consumption of the subsystems is listed in Table 10 based on Xilinx's toolbox XPower Estimator [76]. Given the attributes of different types of resource utilization in Table 8 and the switch rate, the toolbox generates the power estimation according to the technology of particular FPGA. It turns out the embedded BRAM drains on the average of 27mW with 100% read and 1% write rates at 180.9MHz operating frequency. The majority of the power dissipation, however, stretches on the NCC subsystem which utilizes the most logic slices (20,424mW) and BRAMs (756mW) at 1.5V internal operating voltage. Currently the typical power estimation for ZBT RAMs is not available from manufactures' datasheets (both Samsung and Cypress).

Table 10: Power consumption of subsystems at 180.9 MHz system clock.

| Components | Line Buffers | Video Enh | Feat Sel | NCC $9\times27$ | Corr Mgmt | Mot. Eval | Aff Trans | ZBT RAM |
|---|---|---|---|---|---|---|---|---|
| Power (mW) | 473 | 1923 | 3049 | 21180 | 597 | 1365 | 507 | --- |

Power dissipation characterized by resources is shown in Table 11 as a whole system. It includes the internal quiescent drain power at 1.5V and the auxiliary power at 3.3V. Likewise, the CLB logics consume roughly 90% of the total power since the system utilizes 59% of the logic slices. The input/output pads drain about 1,034mW with 12mA LVTTL standard drivers also operating at 180.9MHz, according to Xilinx's design reference [77]. The actual current may vary depending on the capacity load of the external device and track impendence of the printed circuit board (PCB). The stabilization system currently requires 30.6W or 20A at 1.5V to operate, excluding the external storage device. The power utilization is approximately 0.17W per MHz operating frequency.

Table 11: Power consumption of the system by FPGA resources.

| Source Name | Power (mW) |
|---|---|
| Vccint Quiescent | 90 |
| Vccaux | 330 |
| CLB Logics | 27643 |
| BRAMs | 1451 |
| Multipliers | 0 |
| Digital Clock Mgmt | 8 |
| Input/Output Pads | 1034 |
| Total | 30622(~19918mA@1.5V) |

## 6.6 Summary

The basic timing of events in the perspective of video frame was illustrated in this chapter. Results from simulation and error analysis of the logarithmic modules, video enhancement and stabilization subsystems were presented along with the characteristics of performance, resource utilization and power consumption of different subsystems. The simulation indicated the improved $\mathrm{Log}_2$ and $\mathrm{iLog}_2$ modules had an average error of 0.09% and 0.44% compared to double precision for 16-bit numbers, respectively. The results also indicated the average error of three pixel intensities for the video enhancement subsystem. Upon closer examination, the uniform scenes appeared slightly brighter than the expected outputs. The result of video stabilization subsystem also indicated a small error of -0.5% and 2.2% for feature selection and measure introduced by the logarithmic modules and the rounding limitation of the architecture. As a result the texturally optimal regions might not be selected. It was also expected that a greater degree of error from feature measure subsystem did not alter the outcome of the coordinates of the features. The performance of various subsystems was one-on-one on a clock cycle basis with the slowest feature selection subsystem limited by the resolution of adders. With a $9 \times 27$ sub-feature dimension, the NCC subsystem consumed the most CLB slices (44%) and BRAMs (16%) as well as the power supply (21.2 watts).

# CHAPTER 7

# CONCLUSION & FUTURE WORK

Stabilization of video sequence captured under non-uniform lighting conditions requires analysis of several components. These include the video enhancement which improves the quality and visibility of the image in a scene with uniform lighting. The second component of the stabilization process is to evaluate reliable features for feature measurement and tracking in the third step. The fourth task is to estimate the global motion parameter of a given scene. This motion parameter can then be applied to generate the display coordinates of stabilized video frames to produce the final sequence.

We have established a simplified model in this research for the video enhancement and stabilization. The algorithm was constructed to reduce complexity and make feasible for implementation with the Xilinx's FPGA technology. A number of concepts were developed along the design process. This included the log-domain computation to reduce hardware complexity, the generalized 2D convolution architecture with quadrant symmetry property for video enhancement, the generic 2D NCC architecture for the support of feature measure, the feature representation by a set of sub-features in the constellation that captured the spatial relationships, and a fast search mechanism for estimation of background motion of the camera.

The goals of this dissertation were to develop a simple video stabilization algorithm reasonable to implement on FGPA technology. We applied the homomorphic filtering with fast color space conversion in the video enhancement subsystem to eliminate two complete color space converters between the RGB and HSV color spaces. With the folding in the architecture to eliminate redundant calculations, only 9 processing elements were needed to realize a $5 \times 5$ kernel of the boosting transfer functions for the enhancement application. Unlike most existing designs, it allowed us to fine-tune the luminance and contrast components within the architecture without any modification to the structure. A model for extraction of features based on textural optimality and the argument of uniqueness was constructed; however, the complexity and the drawback of non-uniform processing bandwidth forced us to concentrate on a single texture layer already available in the literature and remove the uniqueness criterion. Moreover, the full feature was decomposed into a primary sub-feature and a set of secondary sub-features based on textural optimality of sub-features. A star constellation was constructed to represent the full feature with the distance and angle relationships among the sub-features. In doing so we minimized the trigger for the measurement of secondary sub-features. For the detection of every local maxima of the primary sub-feature, the secondary sub-features served to confirm the existence of proper structure in the constellation. Due to hardware limitation, a straight line constellation was chosen to reduce buffering of video lines. In this fashion, the number of calculations involved was considerably reduced since the measure of a complete feature was never performed. With a 2D model of the scene captured by the camera, we also constructed a very efficient search mechanism to quickly estimate the inlier motion from a set of corresponding points of the adjacent video frames. The search technique progressively rejected the outlier motions and terminated with the discovery of an outstanding element equivalent to the background motion. The approach only required the calculation of angles between the vectors of point-pairs from the feature coordinates. The novel architectures for the computation of logarithmic corner-ness response, and the angle calculation between the point-pair vectors were important. The most important portion of the subsystems, however, was the design of NCC architecture as it consumed a majority of the hardware resource. The process of energy normalization was simplified to tolerate the bandwidth of moving video lines rather than demanding full 2D processing power. By computing the summation relatively independent of the 2D correlation and energy components, the architecture for calculation of the 2D integral sum was applied which only utilized 11 adders instead of the 243 adders in full bandwidth for a $9 \times 27$ kernel. The concept of sub-features further reduced the complexity of NCC design. Rather than computing with a $27 \times 27$ kernel or larger, only partial NCC was implemented. The conventional concept of time multiplexing to trade the performance for the gain of resources had the drawback of reducing the throughput rate inversely proportional to reusability of the functional units. The very idea of our sub-feature representation also demanded a similar need for the processing power; however, it completely focused on the primary sub-feature, eliminating the full calculation of partial results otherwise wasted in the event of a failed measure. The decomposition of features and the modification to the data dependency of the NCC calculation made the entire system possible to sustain the performance of a one-on-one throughput rate without compromise. In addition to the aforementioned results, the improved version of logarithmic modules was employed to remove the need of embedded hardware multipliers, dividers and exponent related operations.

The $\log_2$ module had better fixed point precision with the average magnitude of error around 0.09% (equivalent to a difference error of 0.0053) which is an order of magnitude lower than the uncorrected module. The $\mathrm{iLog}_2$ module had a 0.44% average error. Both modules were more precise than the two or three region correction methods compared to other implementations that relied on piecewise lines to generate error coefficients. The accuracy of the bit-level curve fitting technique also came between the three and six region methods with relatively fewer resources. The video enhancement subsystem had the average error of three pixel intensities which is barely noticeable to human eyes. Upon close examination, however, the results produced by hardware were slightly brighter than the expected image computed with double precision. If desirable, this effect could be compensated by fine-tuning the luminance component of the transfer function. The range of errors from the test sequence for feature selection and measure subsystems was around -0.5% and 2.2%, respectively. Because of the rounding limitation within the architecture, the feature selection subsystem did not necessarily select the most optimal regions as we would expect on the software; however, the majority of the potential features were consistent. While comparison of the feature measure subsystem with previous implementation might not be completely available, it should be clear the new approach was superior in the aspects of accuracy, performance, and resource utilization.

The precision of the current stabilization system mainly depends on the video enhancement and the affine transformation. The quality of the video is expected to degrade with the increase in angle of rotation. The system is expected to sustain 180.9

MOPS or equivalently 172 fps with a $1024 \times 1024$ frame size on Xilinx's Virtex II 2V8000-5 FPGA technology. It consumes 59% logic slices, 35% embedded rams and two external ZBT frame buffers and dissipates roughly 30.6 watts of power at 1.5 volts supply with 3.3V auxiliary power.

Future work will concentrate on extending the great potential of such a model into finer grains for extraction and adaptive tracking of moving objects since our model encapsulates these attributes with lower computational complexity in the aspect of both algorithmic and hardware development. From these attributes, the angle and distant relationships within the constellation become useful for analysis of spatial distortion/deformation. The obvious benefit is its ability to determine the 3D structures of the objects to a certain extent from the 2D video frames. This concept should be exploited to a greater extent in the near future. With these parameters to narrow down the processing range, the bandwidth demand becomes highly non-linear and concentrated which makes it possible to achieve 20 fps or greater on desktop computers with Intel Core 2 Duo or Quad Core CPUs and 2GB DDR2 memory without the demand of dedicated hardware for video frames of conventional size.

## REFERENCES

[1] V. Caselles, J.L. Lisani, J.M. Morel, and G. Sapiro, "Shape preserving local histogram modification," IEEE Trans. Image Process. vol. 8, no. 2, pp. 220–230, 1999.

[2] D.J. Jobson, Z. Rahman, and G.A. Woodell, "A multi-scale retinex for bridging the gap between color images and the human observation of scenes," IEEE Trans. Image Process. vol. 6, no. 7, pp. 965–976, 1997.

[3] E. H. Land and J. J. McCann, "Lightness and retinex theory," *Journal of the Optical Society of America*, vol. 61, no. 1, pp. 1-11, 1971.

[4] R. Mekle, A.F. Laine, and S.J. Smith, "Evaluation of a Multi-Scale Enhancement Protocol for Digital Mammography," Image-Processing Techniques For Tumor Detection, R. N. Strickland, Ed., Marcel Dekker, New York, NY, pp. 155-186, 2001.

[5] M.J. Seow, and V.K. Asari, "Ratio rule and homomorphic filter for enhancement of digital color image," Journal of Neurocomputing, vol. 69, no. 7-9, pp. 954-958, 2006.

[6] M.J. Seow, and V.K. Asari, "Associative memory using Ratio rule for multi-valued pattern association," Proceedings of the IEEE International Joint Conference on Neural Networks, Portland, Oregon, pp. 2518–2522, 2003.

[7] A.F. Breitzman, "Automatic derivation and implementation of fast convolution algorithms," PhD Dissertation, Drexel University, 2003.

[8] E. Jamro, "Parameterised automated generation of convolvers implemented in FPGAs," PhD Dissertation, University of Mining and Mentallurgy, 2001.

[9] A. Wong, "A new scalable systolic array processor architecture for discrete convolution," MS Thesis, University of Kentucky, 2003.

[10] J. Yli-kaakinen, and T. Saramaki, "A systematic algorithm for the design of multiplierless FIR filters," Proceedings of the IEEE International Symposium Circuits and Systems, Sydney, Australia, vol. 2, pp. 185–188, 2001.

[11] M. Z. Zhang, H. T. Ngo, and K. V. Asari, "Multiplier-less VLSI architecture for real-time computation of multi-dimensional convolution," Journal of Microprocessors and Microsystems, vol. 31, pp. 25-37, 2007.

[12] T. G. Stockham Jr., "Image processing in the context of a visual model," Proceedings of IEEE, vol. 60, pp. 828-842, July, 1972.

[13] R.W. Fries, and J.W. Modestino "Image enhancement by stochastic homomorphic filtering," IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP-27, no. 6, pp. 625-37, 1979.

[14] R. Kimmel, M. Elad, D. Shaked, R. Keshet (Kresch), and I. Sobel, "A variational framework for retinex," the International Journal on Computer Vision, vol. 52, no. 1, pp. 7-23, April 2003.

[15] Z. Rahman, D. J. Jobson, and G. A. Woodell, "Retinex processing for automatic image enhancement," J. Electronic Imaging, vol 13, pp. 100-110, 2004.

[16] G. D. Hines, Z. Rahman, D. J. Jobson, and G. A. Woodell, "DSP implementation of the retinex image enhancement algorithm," Visual Information Processing XIII, Proc. of SPIE 5438, pp. 13-24, 2004.

[17] C. Chung, and O. Sohm, "Signal Processing Examples Using TMS320C64x Digital Signal Processing Library (DSPLIB)", Texas Instruments, September 2003.

[18] S. Qureshi, "Embedded Image Processing on the TMS320C6000$^{TM}$ DSP: Examples in Code Composer Studio and Matlab," Springer, 2005.

[19] M.Z. Zhang, M.J. Seow, and V.K. Asari, "A high performance architecture for color image enhancement using a machine learning approach," International Journal of Computational Intelligence Research – Special Issue on Advances in Neural Networks, vol. 2, no. 1, pp. 40–47, 2006.

[20] H. T Ngo, M. Z. Zhang, L. Tao, and V. K. Asari, "Design of a high performance architecture for real-time enhancement of video stream captured in extremely Low lighting environment," International Journal of Embedded Systems: Special Issue on Media and Stream Processing (accepted for publication), 2008.

[21] L. Tao and V. K. Asari, "Modified luminance based MSR for fast and efficient image enhancement," IEEE International Workshop on Applied Imagery and Pattern Recognition, AIPR - 2003, Washington DC, USA, pp. 174-179, 2003.

[22] B. Krose, and P. van der Smagt: "An introduction to neural networks," University of Amsterdam, 1996.

[23] F. Rosenblatt, "Principles of Neurodynamics," Spartan, New York 1959.

[24] M. Minsky and S. Papert, "Perceptrons: An Introduction to Computational Geometry," The MIT Press, 1969.

[25] B. Widrow, and M.E. Hoff Jr., "Adaptive switching circuits," IRE WESCON Convention Record, pt. 4, pp. 96-104, 1960.

[26] J.A. Anderson, "Neural models with cognitive implications – Basic processes in reading perception and comprehensive models," Hillsdale, NJ, pp. 27-90, 1977.

[27] T. Kohonen, Associative Memory: A System Theoretic Approach, Springer, Berlin, 1977.

[28] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities," Proceedings of the National Academy of Sciences of the USA, vol 79, pp. 2554-2558, 1982.

[29] R. S. hartati and M. E. El-Hawary, "A summary of applications of Hopfield neural network to economic load dispatch, 2000.

[30] S. Grossberg, "Adaptive pattern classication and universal recoding II:Feedback, expectation, olfaction and illusions," Biological Cybernetics, vol 23: pp. 187–202, 1976.

[31] A.K. Jain, M.N. Murty and P.J. Flynn, "Data Clustering: A review," ACM Computing Surveys, vol 31, no. 3: pp. 264-323, 1999.

[32] G. A. Carpenter, S. Grossberg, and D. B. Rosen, "Fuzzy-ART: Fast stable learning and categorisation of analog patterns by an adaptive resonance system," Neural Networks, vol 4, no. 6, pp. 759–771, 1991.

[33] G. A. Capenter, S. Grossberg, N. Markuzon, J. H. Reynolds, and D.B. Rosen, "Fuzzy ARTMAP: A neural network architecture for incremental supervised learning of analog multidisional maps," IEEE Transactions On Neural Networks, vol. 3, no. 5, pp. 9, 1992.

[34] D. G. Lowe, "Distinctive Image Features for Scale-Invariant Keypoints," International Journal of Computer Vision, vol 60, no. 2, pp. 91-110, 2004.

[35] B. Furht, J. Greenberg, and R. Westwater, "Motion estimation algorithms for video compression," Kluwer Academic Publishers, 1997.

[36] D. Toth, T. Aach, and V. Metzler, "Illumination-Invariant Change Detection," Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation, pp. 3, 2000.

[37] A. Smolic and J. Ohm, "Robust global motion estimation using a simplified M-estimator approach," In Proc. IEEE International Conference on Image Processing, vol 1, pp. 868-871, 2000.

[38] J. R. Jain and A. K. Jain, "Displacement measure and its application in interframe image coding," IEEE Trans. Commun., vol 29, pp. 1799-1806, 1981.

[39] H. H. Nagel, "Displacement vectors derived from second-order intensity variations in image sequences," Comput. Vision, Graphics, Image Processing, vol. 21, no. 1, pp. 85-117, 1983.

[40] C. K. Cheung and L. M. Po, "”A hybrid adaptive search algorithm for fast block motion estimation,” IEEE International Symp. Signal Proc. and its Appl., vol. 1, pp. 365-368, 1996.

[41] R. Dahyot, and A. Kokaram, “Comparison of two algorithms for robust m-estimation of global motion parameters,” in Proc. Irish Machine Vision and Image Processing, pp. 224-231, 2004.

[42] J. L. Barron, D.J. Fleet, Beauchemin, S., and Burkitt, “Performance of optical flow techniques,” IEEE Conference on Computer Vision and Pattern Recognition, Champaign, pp. 236-242, 1992.

[43] W. Qi, H. J. Zhang, and Y. Zhong, “New robust global motion estimation approach used in mpeg-4,” Journal of Tsinghua University Science and Technology, 2001.

[44] B. D. Lucas “Generalized Image Matching by the Method of Differences,” PhD Dissertation, Dept. of Computer Science, Carnegie-Mellon University, 1984.

[45] E. P. Simoncelli, E. H. Adelson and D. J. Heeger, “Probability distributions of optical flow,” IEEE Proceddings of CVPR, Maui, pp. 310-315, 1991.

[46] M. Z. Zhang and K. V. Asari, “An efficient multiplier-less architecture for 2-D convolution with quadrant symmetric kernels,” Integration, the VLSI Journal, vol. 40, no. 4, pp. 490-502, 2007.

[47] V. Lakshmanan, “A Separable Filter for Directional Smoothing,” IEEE Transaction on Geoscience and Remote Sensing Letters, vol. 1, no. 3 pp. 192-195, 2004.

[48] J. N. Mitchell, “Computer Multiplication and Division Using Binary Logarithms,” IRE Transactions on Electronic Computers, pp. 512-517, 1962.

[49] K. H. Abed, and R. Siferd, “CMOS VLSI Implementation of 16-bit logarithm and anti-logarithm converters”, IEEE Midwest Symposium on Circuits and Systems, vol. 2, pp. 776-779, 2000.

[50] D.J.Mclaren, "Improved Mitchell-based logarithmic multiplier for low-power DSP applications," SOC Conference, 2003. Proceedings. IEEE International [Systems-on-Chip] , pp. 53- 56, 2003.

[51] M. Combet, H. Zonneveld, and L. Verbeek, “Computation of the Base Two Logarithm of Binary Numbers,” IEEE Trans. Electronic Computers, vol. 14, pp. 863-867, 1965.

[52] S.L. SanGregory, C. Brothers, D. Gallagher, and R. Siferd, “A Fast, Low-Power Logarithm Approximation with CMOS VLSI Implementation”, Proceedings of the IEEE Midwest Symposium on Circuits and Systems, vol. 1, pp. 388–391, 1999.

[53] K. H. Abed, and R. E. Siferd, "CMOS VLSI Implementation of a Low-Power Logarithmic Converter," IEEE TRANSACTIONS ON COMPUTERS, vol. 52, no. 11, pp. 1421-1433, 2003.

[54] E.L. Hall, D. Lynch, and J. Dwyer III, "Generation of Products and Quotients Using Approximate Binary Logarithms for Digital Filtering Applications," IEEETrans. Computers, vol. 19, pp.97-105, 1970.

[55] L. Tao and K. V. Asari, "An efficient illuminance-reflectance nonlinear video stream enhancement model," Proceedings of the IS&T/SPIE Symposium on Electronic Imaging: Real-Time Image Processing III, San Jose, CA, vol. 6063, pp. 60630I-1-12, 2006.

[56] Color Space Conversions: http://www.cs.rit.edu/~ncs/color/t_convert.html

[57] J. Shi and C. Tomasi, "Good Features to Track," Procedding IEEE Conference on Computer Vision and Pattern Recognition, pp. 593-600, 1994.

[58] M.J. Seow and V.K. Asari, "Learning using distance based training algorithm for pattern recognition," Pattern Recognition Letters, vol. 25, no. 2, pp. 189-196, 2004.

[59] T. Kohonen, K. Makisara, K. Saramaki, "Phonotopic maps – insightful representation of phonological features for speech recognition," Proceedings IEEE 7th International Conference on Pattern Recognition, pp. 182-185, 1984.

[60] M.Z. Zhang, M.J. Seow and K.V. Asari, "A hardware architecture for color image enhancement using a machine learning approach with adaptive parameterization," Proceedings of the IEEE International Joint Conference on Neural Networks – IJCNN 2006,Vancouver, BC, Canada, pp. 35-40, 2006.

[61] J.P. Lewis, "Fast normalized cross-correlation," Industrial Light & Magic, 2003. (http://www.idiom.com/~zilla/Work/nvisionInterface/nip.pdf)

[62] OpenCV: GoodFeaturesToTrack function.

[63] T. F. Chan and J. Shen, "Theory and computation of variational image deblurring," IMS Lecture Notes, 2006. (http://www.math.umn.edu/~jhshen/Mars/ChanShenIMS.pdf )

[64] C. G. Harris and M. Stephens, "A combined corner and edge detector," In Proc. 4th Alvey Vision Conf., Manchester, pp. 147-151, 1988.

[65] L.D. Stefano, S. Mattoccia, and M. Mola, "An efficient algorithm for exhaustive template matching based on normalized cross correlation," Proceedings of the 12th International Conference on Image Analysis and Processing, pp. 322, 2003.

[66] J.C. Huang and W.S. Hsieh, "Automatic feature-based global motion estimation in video sequences," IEEE Trans Consum Electron, vol. 50, no. 3, pp. 911-915, 2004.

[67] B. Ahmad and T.S. Choi, "Edge detection-based block motion estimation," IEE Electronics Letters, vol. 37, no. 1, pp. 17-18, 2001.

[68] M. Z. Zhang and V. K. Asari, "A hardware-efficient high performance digital architecture for run-time computation of normalized cross correlation," WSEAS Transactions on Circuits and Systems, vol. 5, no. 9, pp. 1416-1423, 2006.

[69] M. Z. Zhang and V. K. Asari, "A design methodology for performance-resource optimization of a generalized 2D convolution architecture with quadrant symmetric kernels," Lecture Notes in Computer Science, Advances in Computer Systems Architecture, Proceedings of the Twelfth Asia-Pacific Computer Systems Architecture Conference - ACSAC 2007, vol. 4697/2007, pp. 220-234, 2007.

[70] Xilinx Inc.: Virtex II Characteristics: http://direct.xilinx.com/bvdocs/publications/ds031.pdf

[71] Xilinx's FPGA board: http://www.xilinx.com/bvdocs/userguides/ug020.pdf

[72] G. D. Hines, Z. Rahman, D. J. Jobson, and G. A. Woodell. "Single-scale retinex using digital signal processors," in Global Signal Processing Conference, pp. 1-6, 2004.

[73] B. Stuber, and D. Zardet: "Ultra-high-speed spectral analysis in Xilinx FPGAs: A 32k-point FFT with 2-GSPS data throughput in a single FPGA establishes a new level of performance," Xcell Journal, pp. 56-59, 2007.

[74] K. Mikolajczykand, and C.Schmid: "Indexing based on scale invariant interest points," In ICCV, pp. 525–531, 2001.

[75] J.Matas, P. Bilek, and O. Chum: "Rotational invariants for wide-baseline stereo," Proceddings of ICPR, vol. 4, pp. 363-366, 2002

[76] Xilinx's XPower Estimator: http://www.xilinx.com/cgi-bin/power_tool/power_Virtex2

[77] S. Bapat: Synthesizable 200 MHz ZBT SRAM interface, XAPP136, 2000.

# VITA

**Ming Zhu Zhang**

## Education:

Doctor of Philosophy in Electrical & Computer Engineering, Old Dominion University, Norfolk, Virginia, December 2008.

Masters of Science in Computer Engineering, Old Dominion University, Norfolk, Virginia, August 2005

Bachelor of Science in Computer Engineering, Old Dominion University, Norfolk, Virginia, December 2004

Bachelor of Science in Electrical Engineering, Old Dominion University, Norfolk, Virginia, December 2004

Associate in Science Degree in Engineering, Thomas Nelson Community College, Hampton, Virginia, August 2002

Associate in Applied Science Degree in Electronics Technology, Thomas Nelson Community College, Hampton, Virginia, August 2002

## PhD Level Courses:

ECE 783: Digital Image Processing, A, Spring 2004.
ECE 882: Digital Signal Processing II, A, Spring 2005.
ECE 795: VLSI Array Processor Design (Ind.), A, Summer 2005.
ECE 847: High Performance Computer Architecture, A, Fall 2005.
ECE 897: VLSI for Signal Processing Systems (Ind.), A, Fall 2005.
ECE 651: Statistical Analysis and Simulation, A, Spring 2006.
ECE 848: Distributed Computer Simulation, C+, Spring 2006.
ECE 897: Adv. Topics in Computer Vision (Ind.), A, Spring 2006.

## Candidacy Examination:

1. Dengwei Fu, and Alan N. Willson, "A Two-Stage Angle-Rotation Architecture and Its Error Analysis for Efficient Digital Mixer Implementation," IEEE Transactions on Circuits and Systems -I: Regular Papers, Volume 53, Number 3, March 2006.
2. Chao Cheng, and Keshab K. Parhi, "High-Speed Parallel CRC Implementation Based on Unfolding, Pipelining, and Retiming," IEEE Transactions on Circuits and Systems -II: Express Briefs, Volume 53, Number 10, October 2006.

## Scholarships and Honors:

Best Ph.D. Researcher 2008
GAANN fellowship 2007
ETA KAPPA NU 2005
Tau Beta Pi, 2003
Phi Kappa Phi, 2003
Staurt Russell Scholarship, 2003
Kovner Scholarship, 2003
Golden Key, 2002
National Dean's List, 2002/2003
Hastings Award (TNCC), 2002

## Research Publications:

### Journals:

1. Ming Z. Zhang, Ming-Jung Seow, Li Tao, and K. Vijayan Asari, "A tunable high-performance architecture for enhancement of stream video captured under nonuniform lighting conditions," Journal of Microprocessors and Microsystems, vol. 32, no. 7, pp. 386–393, October 2008..
2. Hau T. Ngo, K. Vijayan Asari, Ming Z. Zhang, and Li Tao, "Design of a systolic-pipelined architecture for real-time enhancement of color video stream based on an illuminance-reflectance model," Integration, the VLSI Journal, vol. 41, no. 4, pp. 474-488, July 2008.
3. Ming Z. Zhang, Hau T. Ngo, Adam R. Livingston, and K. Vijayan Asari, "A high performance architecture for implementation of 2-D convolution with quadrant symmetric kernels," International Journal of Computers and Applications (in print).
4. Hau T. Ngo, Ming Z. Zhang, Li Tao, and K. Vijayan Asari, "Design of a high performance architecture for real-time enhancement of video stream captured in extremely low lighting environment," International Journal of Embedded Systems: Special Issue on Media and Stream Processing (in print).
5. Ming Z. Zhang and K. Vijayan Asari, "An efficient multiplier-less architecture for 2-D convolution with quadrant symmetric kernels," Integration, the VLSI Journal, vol. 40, no. 4, pp. 490-502, July 2007.
6. Ming Z. Zhang, Hau T. Ngo, and K. Vijayan Asari, "Multiplier-less VLSI architecture for real-time computation of multi-dimensional convolution," Journal of Microprocessors and Microsystems, vol. 31, pp. 25-37, February 2007.
7. Ming Z. Zhang and K. Vijayan Asari, "A hardware-efficient high performance digital architecture for run-time computation of normalized cross correlation,"

WSEAS Transactions on Circuits and Systems, vol. 5, no. 9, pp. 1416-1423, September 2006.

8. Ming Z. Zhang, Ming-Jung Seow and K. Vijayan Asari, "A high performance architecture for color image enhancement using a machine learning approach," International Journal of Computational Intelligence Research - Special Issue on Neurocomputing and Applications, vol. 2, no. 1, pp. 41-48, 2006.

**Book Chapters:**

1. Ming Z. Zhang and K. Vijayan Asari, "A design methodology for performance-resource optimization of a generalized 2D convolution architecture with quadrant symmetric kernels," Lecture Notes in Computer Science, Published by Springer-Verlag Berlin/Heidelberg (ISSN: 0302-9743), Advances in Computer Systems Architecture, Proceedings of the Twelfth Asia-Pacific Computer Systems Architecture Conference - ACSAC 2007: (ISBN: 978-3-540-74308-8), vol. 4697/2007, pp. 220-234, August 23-25, 2007.

2. Ming Z. Zhang, Li Tao, Ming-Jung Seow, and K. Vijayan Asari, "Design of an efficient flexible architecture for color image enhancement," Lecture Notes in Computer Science, Published by Springer-Verlag Berlin/Heidelberg (ISSN: 0302-9743), Advances in Computer Systems Architecture, Edited by C. Jesshope and C. Egan: Proceedings of the Eleventh Asia-Pacific Computer Systems Architecture Conference - ACSAC 2006: (ISBN: 3-540-29643-3), vol. 4186/2006, pp. 323-336, July 2006.

3. Ming Z. Zhang, Hau T. Ngo, and K. Vijayan Asari, "Design of an efficient multiplier-less architecture for multi-dimensional convolution," Lecture Notes in Computer Science, Published by Springer-Verlag Berlin/Heidelberg (ISSN: 0302-9743), Advances in Computer Systems Architecture, Edited by T. Srikanthan, J. Xue and C. H. Chang: Proceedings of the Tenth Asia-Pacific Computer Systems Architecture Conference - ACSAC 2005: (ISBN: 3-540-29643-3), vol. 3740/2005, pp. 65-78, October 2005.

**Conference Proceedings:**

1. Ming Z. Zhang and K. Vijayan Asari, "A new framework for automatic feature selection for tracking," Proceedings of the Twentieth International Joint Conference on Neural Networks - IJCNN 2007, Orlando, Florida, pp. pp. 3104-3109, August 12 -17, 2007.

2. Ming Z. Zhang, Ming-Jung Seow, Li Tao and K. Vijayan Asari, "Design of an efficient architecture for enhancement of stream video captured in non-uniform lighting conditions," Proceedings of the International Symposium on Signals Circuits and Systems - ISSCS 2007, Romania, vol. 2, pp. 1-4, July 13-14, 2007.

3. Ming Z. Zhang and K. Vijayan Asari, "A fully pipelined multiplierless architecture for 2D convolution with quadrant symmetric kernels," Proceedings of the IEEE Asia Pacific Conference on Circuits and Systems - APCCAS 2006, Singapore, pp. 1561-1564, December 4-7, 2006.

4. Hau T. Ngo, Ming Z. Zhang, Li Tao and K. Vijayan Asari, "Design of a digital architecture for real-time video enhancement based on illuminance-reflectance model," Proceedings of the 49th IEEE International Midwest Symposium on Circuits and Systems - MWSCAS 2006, Puerto Rico, pp. 3179-1-5, August 6-9, 2006.

5. Ming Z. Zhang, Ming-Jung Seow and K. Vijayan Asari, "A hardware architecture for color image enhancement using a machine learning approach with adaptive parameterization," Proceedings of the IEEE International Joint Conference on Neural Networks - IJCNN 2006,Vancouver, BC, Canada, pp. 35-40, July 16-21, 2006.

6. Ming-Jung Seow, Ming Z. Zhang and K. Vijayan Asari, "Natural color representation using Ratio learning algorithm for enhancement of digital color images," IS&T Proceedings of the 30th International Congress of Imaging Science - ICIS'06, Rochester, New York, pp. 625-628, May 7-11,2006.

7. Li Tao, Hau T. Ngo, Ming Z. Zhang, Adam Livingston, and K. Vijayan Asari, "Multi-sensor image fusion and enhancement system for assisting drivers in poor lighting conditions," IEEE Computer Society Proceedings of the International Workshop on Applied Imagery and Pattern Recognition, AIPR - 2005, Washington DC, pp. 106-113, October 19 - 21, 2005.

8. Hau Ngo, Li Tao, Adam Livingston, Ming Z. Zhang, and K. Vijayan Asari, "A visibility improvement system for low vision drivers by nonlinear enhancement of fused visible and infrared video," IEEE 1st Workshop on Computer Vision Applications for the Visually Impaired - CVAVI 2005: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition - CVPR 2005, San Diego, CA, pp. 25, June 20 - 25, 2005.

9. Ming Z. Zhang, Hau T. Ngo, Adam Livingston, and K. Vijayan Asari, "An efficient VLSI architecture for 2-D convolution with quadrant symmetric kernels," IEEE Computer Society Proceedings of the International Symposium on VLSI - ISVLSI 2005, Tampa, Florida, pp. 303-304, May 11 - 12, 2005.

10. Adam Livingston, Hau T. Ngo, Ming Z. Zhang, Li Tao, and K. Vijayan Asari, "Design of a real time system for nonlinear enhancement of video streams by an integrated neighborhood dependent approach," IEEE Computer Society Proceedings of the International Symposium on VLSI - ISVLSI 2005, Tampa, Florida, pp. 301-302, May 11 - 12, 2005.

**Published Abstracts:**

1. Ming Z. Zhang and K. Vijayan Asari, "A robust algorithm for real-time stabilization of shaky video captured under low lighting conditions," ODU-NSU-EVMS-VTC Research Exposition Day: Research Expo 2007 - 400 Years of Discovery, Ted Constant Hall, Norfolk, VA, (Poster), p. 43, April 05, 2007.
2. Li Tao, Ming Z. Zhang, and K. Vijayan Asari, "Image fusion for visibility improvement of digital color images," ODU-NSU-EVMS Research Exposition Day: Research Expo 2006 - Global Challenges and Local Solutions, Ted Constant Hall, Norfolk, VA,(Poster), p. 34, April 05, 2006.
3. Hau T. Ngo, Li Tao, Ming Z. Zhang, Adam Livingston, and K. Vijayan Asari, "Driver's assistant for visibility improvement," ODU-NSU Research Exposition Day: Research Expo 2005, Ted Constant Hall, Norfolk, VA, (Poster) April 06, 2005.