Low Complexity Optical Flow Using Neighbor-Guided Semi-Global Matching

by

Jiang Xiang

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2017 by the
Graduate Supervisory Committee:

Chaitali Chakrabarti, Chair
Lina Karam
Hun Seok Kim

ARIZONA STATE UNIVERSITY

May 2017

ABSTRACT

Many real-time vision applications require accurate estimation of optical flow. This problem is quite challenging due to extremely high computation and memory requirements. This thesis focuses on designing low complexity dense optical flow algorithms.

First, a new method for optical flow that is based on Semi-Global Matching (SGM), a popular dynamic programming algorithm for stereo vision, is presented. In SGM, the disparity of each pixel is calculated by aggregating local matching costs over the entire image to resolve local ambiguity in texture-less and occluded regions. The proposed method, Neighbor-Guided Semi-Global Matching (NG-fSGM) achieves significantly less complexity compared to SGM, by 1) operating on a subset of the search space that has been aggressively pruned based on neighboring pixels' information, 2) using a simple cost aggregation function, 3) approximating aggregated cost array and embedding pixel-wise matching cost computation and flow computation in aggregation. Evaluation on the Middlebury benchmark suite showed that, compared to a prior SGM extension for optical flow, the proposed basic NG-fSGM provides robust optical flow with 0.53% accuracy improvement, 40x reduction in number of operations and 6x reduction in memory size. To further reduce the complexity, sparse-to-dense flow estimation method is proposed. The number of operations and memory size are reduced by 68% and 47%, respectively, with only 0.42% accuracy degradation, compared to the basic NG-fSGM.

A parallel block-based version of NG-fSGM is also proposed. The image is divided into overlapping blocks and the blocks are processed in parallel to improve throughput, latency

and power efficiency. To minimize the amount of overlap among blocks with minimal effect on the accuracy, temporal information is used to estimate a flow map that guides flow vector selections for pixels along block boundaries. The proposed block-based NG-fSGM achieves significant reduction in complexity with only 0.51% accuracy degradation compared to the basic NG-fSGM.

To my Mother and Father

ACKNOWLEDGMENTS

I would first like to express my sincere gratitude to my thesis advisor Dr. Chaitali Chakrabarti, for the continuous support of my master study and research, for her patience, motivation, enthusiasm, and immense knowledge. The door to Prof. Chakrabarti office was always open whenever I ran into a trouble spot or had a question about my research or writing. I could not have imagined having a better advisor and mentor for my M.S. study.

Besides my advisor, I would like to express my gratitude to the rest of my thesis committee, I would like to thank Dr. Lina Karam for her teaching in the Image Processing class, where I developed my interests in computer vision, and referring me to the research opportunities in Prof. Chakrabarti's lab. I would like to thank Dr. Hun Seok Kim of the EECS department at University of Michigan. I enjoyed the opportunity to work closely with him in the past years. His valuable ideas and insightful comments have helped guide my research.

I would also like to thank Ziyun Li at University of Michigan. He has been a great team mate in this work. As a member of the Low Power Systems Lab at Arizona State University, I would like to thank my lab mates, Hsing-Min Chen, Siyuan Wei, Manqing Mao, Jian Zhou and Shunyao Wu, for their excellent friendship.

Lastly but mostly importantly, I would like to thank my parents, for giving birth to me at the first place, raising me with unconditionally love, providing me with unfailing support during my master study.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

Table Page

Table Page

LIST OF FIGURES

Figure Page

Figure Page

CHAPTER 1. INTRODUCTION

1.1 Motivation

Optical flow is a very popular algorithm in computer vision that is used to compute motion vectors of image pixels from a sequence of images. It is used in the early stages of many applications such as virtual reality, object detection/tracking, video compression and autonomous driving. While several definitions of optical flow from the literature highlight different properties of optical flow, a more recent and well-accepted one by Horn [1] defines optical flow as "the apparent motion of brightness patterns observed when a camera is moving relative to objects being imaged".

In computer vision, information is obtained by spatially and temporally sampling the incoming light to the camera. Changes of visual patterns in the scene are represented by a series of image frames, where each image frame is represented by a 2D array of pixels. Optical flow captures the changes between two frames through a 2D vector field, where each vector (consisting of one vertical component and one horizontal component) defines the point correspondence of a pixel. Figure 1.1 shows three frames of a video and the two corresponding optical flow maps.

Dense optical flow computes a motion vector for each pixel of the image and can be acquired in different ways. Most existing algorithms compute flow by optimizing a global energy function in the form of weighted sum of a *data term* and a *prior term*, as stated in

the taxonomy of Baker et al. [2]. The *data term* measures consistency of optical flow in the input images; the *prior term* refers to flow field's preference such as flow smoothness.

The optimization strategies used by the global energy function can be categorized into two different classes: continuous optimization and discrete optimization. Continuous optimization algorithms, such as those by Baker and Matthews [3], Bruhn et al. [4], and Zimmer et al. [5], typically require a large number of non-linear computations. In contrast, discrete optimization algorithms, often used in stereo vision, enhance search efficiency but sacrifice fidelity by approximating the solution space. See Lempitsky et al. [6], Cooke [7], and Lei and Yang [8], for more details. Also, unlike discrete optimization methods in stereo, 2D flow fields make discrete optimization significantly more challenging. Therefore, for large flow displacements, both continuous and discrete optimization algorithms are

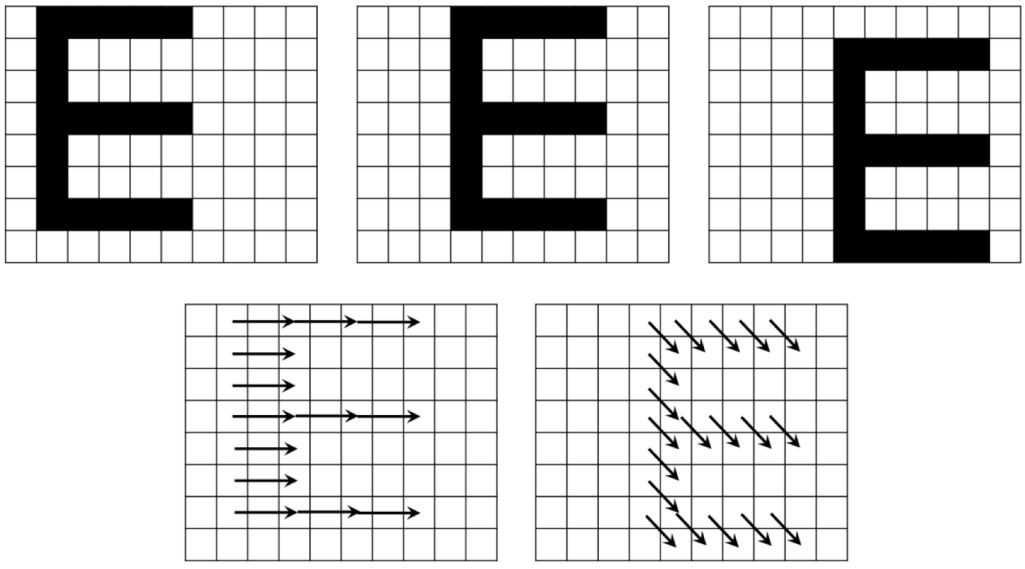

Figure 1.1 Example of three frames which show the movement of letter "E", and the corresponding optical flow maps. 1st row: frame 1, frame 2, frame 3; 2nd row: optical flow between frame 1 and 2, optical flow between frame 2 and 3.

typically embedded into a coarse-to-fine approach [9] to improve efficiency of convergence and avoid local minima. However, the hierarchical approach incurs inevitable accuracy degradation due to resolution loss at higher levels.

Estimating accurate optical flow is challenging in the following areas: occlusion, transparent objects, low/uniform textures, perspective distortion and repeated patterns. In recent years, for emerging real-time and power-critical mobile applications including ADAS (advanced driving assistance system) and autonomous navigation of MAVs (micro aerial vehicles), there is the added challenge of real-time processing with stringent memory and computational resource constraints. However, for most existing optical flow algorithms, memory space and complexity typically increase quadratically with flow search range. Hence, for even a moderate search range (e.g. 64×64) and VGA 60fps, state of the art methods require large memory (~100 MB), high memory bandwidth (~10 GB/s) and very high computational complexity (~2T op/s). So, there is a need to develop new schemes for accurate optical flow estimation with significantly reduced complexity.

1.2 Contribution

In this thesis, a novel optical flow algorithm NG-fSGM (Neighbor-Guided Semi-Global Matching) is presented. This work was done in collaboration with Ziyun Li and Professor Hun Seok Kim at the University of Michigan. An earlier version of this work appeared in [10]. The algorithm is based on SGM [11], a popular concept in stereo matching, and fSGM [9], a prior work that applies SGM to optical flow. Both SGM and fSGM first compute pixel-wise matching costs for all disparities/flow vectors in the search space. Then a

smoothness constraint is added by aggregating costs from multiple directions in image. The complexity of SGM and fSGM increases with the size of search space. Thus for optical flow, the complexity increases quadratically with flow range, and is too high for applications with extreme memory and computational constraints.

Our objective is to achieve performance comparable to fSGM with significantly higher efficiency. Compared to conventional SGM, the proposed method, NG-fSGM, use the following techniques to reduce complexity: 1) A subset of flow vectors is selected by exploring flow similarity of neighboring pixels. This reduces the search space significantly resulting in significant reduction in memory size and computational complexity. 2) A simple cost aggregation function is used, which makes logic/arithmetic operations hardware-friendly. 3) The aggregated cost data is approximated, and pixel-wise matching cost computation and flow computation is embedded in cost aggregation. Since not all costs need to be stored, the memory requirement is reduced. 4) Dense flow map is estimated from sparse flow vectors using interpolation. Since flow estimates are computed for sampled pixels, both memory size and number of operations are reduced. Of the 4 techniques, the baseline method presented in [10] uses technique 1, 2 and 3 to reduce complexity.

The proposed method was evaluated on the Middlebury dataset [2]. The impact of algorithm parameters such as Census size, number of best/random flow vectors and number of paths, on performance metrics such as accuracy, memory footprint and number of computations, was studied. This analysis was used to help find the best set of parameters.

We show that the proposed basic NG-fSGM provides robust optical flow accuracy with 0.53% accuracy improvement compared to fSGM. Furthermore, NG-fSGM achieves about 40x reduction in the number of operations and 6x reduction in the memory requirement. We also show that with sparse-to-dense flow estimation, number of operations and memory size are further reduced by 68% and 47% respectively with only 0.42% accuracy degradation, compared to the basic NG-fSGM.

A parallel block-based version of NG-fSGM is also presented. An earlier version of the work appeared in [12]. The image is divided into overlapping blocks and the blocks are processed in parallel to improve throughput, latency and power efficiency. To minimize the amount of overlap among blocks with minimal effect on the accuracy, temporal information is used to estimate a flow map that guides flow vector selections for pixels along block boundaries.

We show that the proposed block-based NG-fSGM achieves significantly gain in complexity with only 0.51% accuracy degradation compared to the basic NG-fSGM when evaluated on the Middlebury dataset. We also conduct a performance analysis by varying the size of block and overlap, and show that using inertial guidance helps in reducing the size of overlapped blocks while preserving the accuracy.

1.3 Organization

The rest of the thesis is organized as follows. Chapter 2 presents background information that is related to the work in this thesis. Chapter 3 presents the proposed optical flow

method, NG-fSGM (Neighbor-Guided Semi-Global Matching). Chapter 4 presents its parallel version, the block-based NG-fSGM. Chapter 5 presents the accuracy results on the Middlebury optical flow dataset along with the accuracy, memory size requirement and computational complexity. Chapter 6 summarizes the thesis.

CHAPTER 2. BACKGROUND

2.1 Optical Flow

Dense optical flow estimation has been studied for more than 30 years. There is a large body of work and so here we first describe the popular objective functions used in optical flow and the optimization procedures in Section 2.1. Next we describe Semi-Global Matching, a discrete optimizing strategy, which forms the basis of our proposed method.

2.1.1 Objective Functions

Horn and Schunck introduced a global formulation for the optical flow model in 1981 [13]. It assumes that the brightness is constant and apparent motion varies smoothly everywhere in the image. Following Horn and Schunck's model, several other models have been proposed and they are all referred to as Horn-Schunck models. Schulman and Herve [14] applied the theory of robust statistics to obtain a convex regularization to address motion discontinuities to compute robust optical flow. Black and Anandan [15] introduced a robust framework to deal with violations of the brightness constancy and spatial smoothness assumptions caused by multiple motions. Sun et al. [16] developed a statistical model that takes into account both brightness constancy errors and spatial properties of optical flow.

There are also many models beyond the Horn-Schunck models. Zimmer et al. [5] introduced the concept of complementarity between data and smoothness term to avoid undesirable interference. Lei and Yang [8] estimated optical flow by representing the input image as a tree of segmented regions and optimizing a region-based energy function. Wedel et al. [17] used adaptive regularization which adoptively favor rigid motion and

motion discontinuities that coincide with discontinuities of the image structure. The proposed optical method in this thesis follows the Horn-Schunck model.

2.1.2 Optimization Methods

Many methods are proposed to optimize the objective function. Some of the popular ones with good performance are listed here. Recall that the optimization methods can be divided into two groups, continuous optimization and discrete optimization. Continuous optimization methods typically require non-linear computation and have extremely high complexity. Baker and Matthews [3] used steepest descent which takes steps in the direction of the negative gradient. Black and Anandan [15] used gradient descent with modified step size. Bergen et al. [18] and Brox et al. [19] presented coarse-to-fine approaches to improve convergence rate and avoid local minima. Compared to continuous optimization methods, discrete optimization methods usually use approximations to improve search efficiency. Lempitsky et al. [6] used binary graph-cut optimizations to refine the current flow estimate. Glocker et al. [20] and Lei and Yang [8] sparsely allocate states for possible flow at each location as a coarse version of the problem. Wedel et al. [17] used a median filter to remove outliers after each incremental estimation step.

The method proposed in this thesis adopts the discrete optimization strategy and is based on Semi-Global Matching (SGM) [11]. Next we briefly review SGM and one of its extensions to optical flow, fSGM [9].

2.2 Semi-Global Matching

SGM (Semi-Global Matching) method for stereo vision was proposed by Hirschmuller [11]. SGM performs a fast approximation of a global cost function optimization by path-wise optimizations along all directions in the image. It first computes pixel-wise matching costs of corresponding pixels in two frames for all disparities in the search space. This is followed by cost aggregation along a finite number of paths in different directions which penalizes abrupt disparity changes to support a smoothness constraint. Finally, it combines costs in every direction and selects the disparity with the minimum cost as output. Additionally, post processing is done to remove outliers.

An extension of SGM method for optical flow, fSGM, was proposed by Hermann and Klette [9]. fSGM extends the search space from 1D stereo to 2D flow. It is probably the closest approach to ours hence we review it here.

Step 1: Computation of pixel-wise matching cost $C(\boldsymbol{p},\boldsymbol{o})$ between pixel $\boldsymbol{p} = (x,y)$ in the previous image frame and pixel $\boldsymbol{q} = \boldsymbol{p} + \boldsymbol{o}$ in the current image frame, for all flow vectors $\boldsymbol{o} = (u,v)$, where u is the horizontal component and v is the vertical component of the flow vectors. The cost function can be based on many forms such as Rank, Census [21] and mutual information [22].

Step 2: Application of an additional constraint on matching costs to get the smoothness of flow image. This step approximates the 2D minimization of a global energy function by aggregating matching costs in 1D from all directions equally as shown in Figure 1.2. These

paths through flow vector space are projected as straight lines into the base image (previous frame). The smoothness constraint is achieved by penalizing abrupt changes of adjacent pixels' flow offsets. The cost $L_r(\boldsymbol{p},\boldsymbol{o})$ of the pixel $\boldsymbol{p}$ for a flow vector $\boldsymbol{o}$ accumulated along a path in the direction r is defined recursively as

$$L_r(\boldsymbol{p},\boldsymbol{o}) = C(\boldsymbol{p},\boldsymbol{o}) + Z - \min_{\boldsymbol{k}} L_r(\boldsymbol{p}-r,\boldsymbol{k}) \tag{2.1}$$

with the cost regularization summand

$$Z = \min_{\boldsymbol{i}}\{L_r(\boldsymbol{p}-r,\boldsymbol{o}) + P_l\|\boldsymbol{i}-\boldsymbol{o}\|_1\} \tag{2.2}$$

where P_l is the penalty factor and $\|\boldsymbol{i}\text{-}\boldsymbol{o}\|_1$ is the L_1 norm of two flow vectors. Since the full linear model may result in over-regularization, [9] also suggests optional truncation of the linear model. The aggregated cost $S(\boldsymbol{p},\boldsymbol{o})$ is the sum of $L_r(\boldsymbol{p},\boldsymbol{o})$ over all paths.

$$S(\boldsymbol{p},\boldsymbol{o}) = \sum_r L_r(\boldsymbol{p},\boldsymbol{o}) \tag{2.3}$$

Step 3: Flow computation. It uses winner-takes-all strategy by selecting $\boldsymbol{o}$ with the minimum overall aggregated cost $S(\boldsymbol{p},\boldsymbol{o})$ as the output flow vector for pixel $\boldsymbol{p}$ in the base

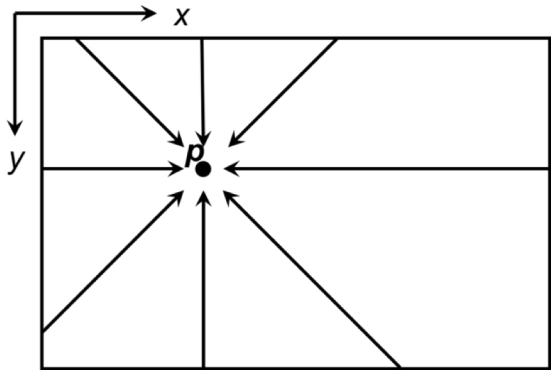

Figure 2.1 Aggregation of costs from all directions r when number of paths is 8.

image.

The complexity of typical SGM-based methods is $O(WHD)$, where W is the width, H is the height and D is the size of search space. Complexity of fSGM, therefore, increases quadratically with the flow range ($D = d^2$ where d is one dimensional search range), making the algorithm rather inefficient for a relatively large search range (e.g., D = 10000 for ±50 pixel search range per dimension). Addressing this issue, fSGM is typically combined with a hierarchical coarse-to-fine approach that incurs inevitable accuracy degradation due to resolution loss at higher hierarchical levels. This motivates the necessity of designing lower complexity alternatives which have comparable accuracy.

CHAPTER 3. OPTICAL FLOW USING NEIGHBOR-GUIDED SEMI-GLOBAL MATCHING

Recall that fSGM consists of the following four steps including pixel-wise matching cost computation, cost aggregation, flow computation and post-processing. The proposed optical flow method performs a flow subset selection step before the four main steps of fSGM [9]. The flow subset selection step reduces the computational and space complexity by aggressively pruning the search space based on the information provided by neighbors. We call this method Neighbor-Guided Semi-Global Matching, or NG-fSGM. Additional methods to reduce the overall complexity include modification of cost aggregation function, approximation of aggregated cost array, and embedding pixel-wise matching cost computation and flow computation in aggregation. It was originally presented in [10]. A block diagram of the proposed optical flow method is shown in Figure 3.1; the parameters used in this work are listed in Table 3.1.

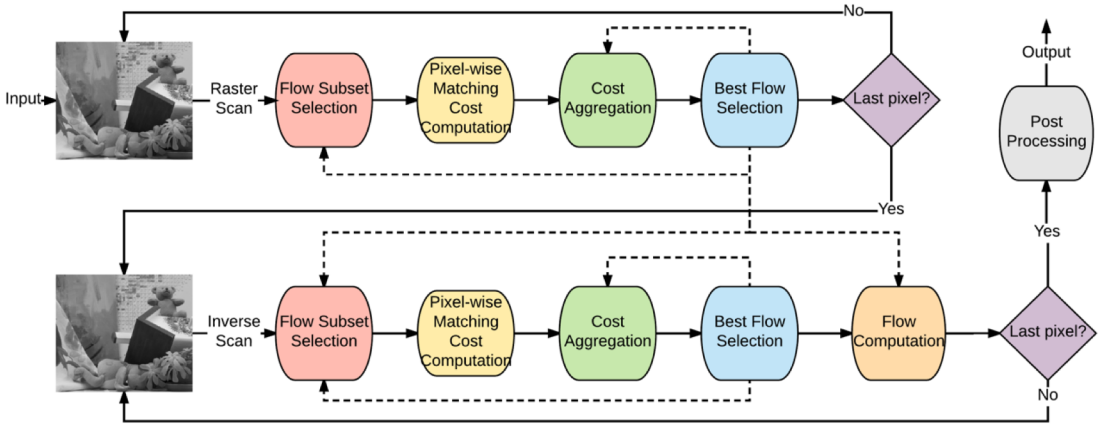

Figure 3.1 Overview of proposed Neighbor-Guided Semi-Global Matching

Table 3.1 Parameters in NG-fSGM and Block-based NG-fSGM

W	Image width	H	Image height
N	# best flow vectors	M	# random flow vectors
P	# paths per scan	K	# flow vectors in search window
C	Census window size	T	Search subset size
P_1	Small penalty	P_2	Large penalty
n	Non-overlapping block size	d	Flow range
m	Overlapping block size	l	Overlap size

The rest of chapter is organized as follows. Details of each step of NG-fSGM are presented in Section 3.1 – 3.4. Complexity analysis of NG-fSGM is presented in Section 3.5. Additionally, an optimization method based on sparse-to-dense flow estimation that further reduces complexity is introduced in Section 3.6.

3.1 Flow Subset Selection

Flow subset selection is the first step of optical flow estimation algorithms. Proper selection reduces the search space size based on the information from neighboring pixels and is the key to complexity reduction. Using neighborhood information to prune the search space was used two decades ago in [23] and [24] in the context of block matching for motion estimation. C. Stiller (1990) [23] introduced a motion-estimation coding algorithm that combined contour coding of regions which have similar displacements with predictive coding of the vectors inside each region. This allowed the estimator to work with decreased block size, resulting in high efficiency. G. Haan et al. (1993) [24] used neighboring block information in a block erosion post-processing method that effectively eliminated block structures from the generated vector field. As a result, visible block structures and artifacts in the motion results were reduced significantly. Recently, PatchMatch [25] used a similar strategy for correspondence matching. It defined a nearest-neighbor field to quickly find

correspondences between small square regions of an image. Our proposed selection strategy is inspired by [25].

Neighboring pixels in the natural image tend to have an identical or similar flow vector since they typically belong to the same object or belong to adjacent objects with similar motion. This property explains why most algorithms use a flow smoothness constraint as the prior term. If two pixels belong to the same object in the scene and the object surface is parallel to the image plane, their optical flow should be identical. Small flow variations usually occur due to slanted surface of objects, spinning objects, camera position, etc. If the image resolution is higher or if the number of frames per second is higher, the difference between flow vectors of two neighboring pixels is smaller. Large flow variations can occur in the edge of objects and are typically due to occlusion and motion discontinuity.

NG-fSGM exploits the property that neighboring pixels have similar flow vectors. It selects a subset of search space, $\boldsymbol{O_p}$ for each pixel $\boldsymbol{p}$, where the selection is based on neighbor pixels' results of flow vectors and their corresponding cost. This step is done prior to the computation of pixel-wise matching cost and is embedded into the dynamic programming scheme of SGM.

The subset selection for each pixel $\boldsymbol{p}$ is guided by its neighboring pixels along every path in SGM, as shown in Figure 3.2. Let $\boldsymbol{Q}$ denote a set of flow vectors. For pixel $\boldsymbol{p}$, the best N flow vectors $\boldsymbol{Q_{p\text{-}r}}$ of the previous pixel along path r with minimum cost $\boldsymbol{L_r(p\text{-}r,o)}$, are added into the search subset $\boldsymbol{O_p}$. We choose the best N vectors denoted by 'B' in Figure 3.2. N is

chosen to be greater than one for robustness to errors caused by accumulated cost variation along a path and localized abnormality of pixel-wise matching cost. Since SGM applies a low aggregation penalty when the flow varies smoothly, adjacent flow vectors denoted by 'A' in Figure 3.2 within a search window centered at each of the *N* best vectors, are selected for cost evaluation as well. To enable the algorithm to adapt to rapid flow variation (e.g., occlusion and object discontinuity), *M* random flow vectors are added to the subset. Note that for pixels along the boundary of the image, some of the neighboring pixels are not available. The flow subset for these pixels are selected randomly to allow simple and fast initialization.

Since SGM approaches are typically implemented in a dynamic programming scheme which consists of two scans, forward scan and backward scan, paths are divided into two

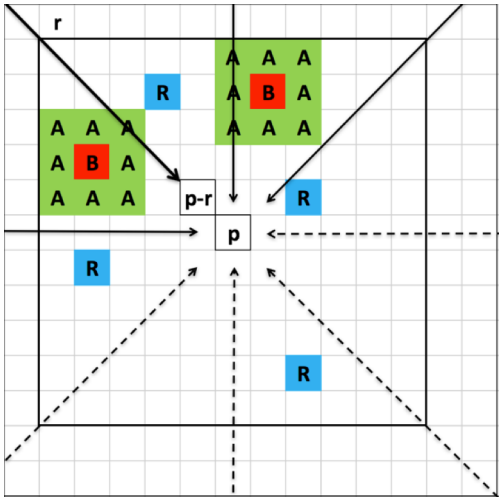

Figure 3.2 Flow Subset Selection. For the center pixel ***p***, the thick square represents flow range. The solid arrows represent path directions in forward scan while dash arrows represent path directions in backward scan. The selected flow vectors, guided by neighbor ***p***-*r* along path *r*, is the combination of flow vectors from B, A and R. B corresponds to the $N = 2$ best flow vectors and A corresponds to their adjacent flow vectors ($K = 9$) and R corresponds to $M = 4$ random flow vectors.

groups as well, as shown in Figure 3.2. The forward scan processes every pixel from top-left to bottom-right of the image in raster scan order, while the backward scan processes pixels in reverse order. In each scan (forward and backward), each pixel performs flow subset selection, pixel-wise matching cost computation, cost aggregation and best flow selection; the flow computation is only performed in the backward scan. As a result, each pixel $\boldsymbol{p}$ has two different flow vector subsets, $\boldsymbol{O}_{p1}$ and $\boldsymbol{O}_{p2}$, and two different aggregated costs, represented by $S_1(\boldsymbol{p},\boldsymbol{o})$ and $S_2(\boldsymbol{p},\boldsymbol{o})$, for each flow vector, one for each scan. The overall subset $\boldsymbol{O}_p$ is the union of $\boldsymbol{O}_{p1}$ and $\boldsymbol{O}_{p2}$, and the overall aggregated cost $S(\boldsymbol{p},\boldsymbol{o})$ is the sum of $S_1(\boldsymbol{p},\boldsymbol{o})$ and $S_2(\boldsymbol{p},\boldsymbol{o})$.

In case the aggregated cost from one scan is missing for a certain $\boldsymbol{o}$, we propose in Section 3.4 an approximation strategy to estimate the missing cost and combine forward and backward aggregation. Additionally, in the backward scan, N best flow vectors from the forward scan with minimum cost $S_1(\boldsymbol{p},\boldsymbol{o})$ and their adjacent vectors (located in the search window) are added to $\boldsymbol{O}_{p2}$ to increase algorithm accuracy. Basically this prevents wrong selection in single scan since flow could be inconsistent in certain directions.

The flow vectors chosen by different aggregation paths may not be distinct since neighboring pixels' best vectors can be identical and the search windows (i.e., B's and A's in Figure 3.2) can overlap. If redundancy is ignored (worst case), the total number of vectors in the search subset is $T = N \times (P + 1) \times K + M$, where P is the number of aggregation paths per scan and K is number of flow vectors in the search window. The complexity of the steps after flow subset selection in NG-fSGM is $O(WHT)$, which is

independent of flow search range ($D = d^2$). Thus in order to reduce the overall complexity, we need to select the smallest possible values for N, P, K and M while maintaining the accuracy. In actual implementation, images with larger flow displacements have higher percentage of selected flow vectors because of lower redundancy, while smaller flow images have higher redundancy and thus lower complexity.

3.2 Pixel-wise Matching Cost

The pixel-wise matching cost computation is an important step, since it is the source of all other types of aggregated cost. The cost value is a local dissimilarity measurement between two corresponding pixels in the previous and current images. Let $C(\boldsymbol{p},\boldsymbol{o})$ represent the pixel-wise matching cost between pixel $\boldsymbol{p} = (x,y)$ in the previous image and pixel $\boldsymbol{q} = \boldsymbol{p} + \boldsymbol{o}$ in the current image, where $\boldsymbol{o} = (u,v)$ is the flow vector whose horizontal component is u and vertical component is v. Various pixel-wise matching cost measurements have been proposed including BCA (brightness constancy assumption) [13], NCC (normalized cross correlation) [26], Mutual information [11] and Census Transform [21]. A comprehensive review of different cost measurements is presented in [27]. Using a combination of different cost measures results in improved overall optical flow accuracy. In our proposed method, we generate the pixel-wise matching cost by combining two measures: BCA and Census Transform [21].

BCA [13] cost is probably the simplest and most common cost measure. It assumes that when a pixel moves from the previous image to the current image, its intensity and color

does not change. The BCA cost penalizes the intensity changes of moving pixels in the image. It is defined as

$$C_{BCA}(\boldsymbol{p}, \boldsymbol{o}) = |I(\boldsymbol{p}, t) - I(\boldsymbol{p} + \boldsymbol{o}, t + 1)| \tag{3.1}$$

Census Transform [21] has been proven to represent image structure well and to be robust to environment of variations [28]. In the Census Transform of an image, a bit string is assigned to every $\boldsymbol{p}$, where each bit is 1 (or 0) if the intensity of $\boldsymbol{p}$ is larger (or smaller) than $\boldsymbol{p}$'s neighboring pixels $\boldsymbol{p}_n$ within a pre-defined window $\boldsymbol{N}_p$. It can be expressed as

$$Census(\boldsymbol{p}) = \otimes \xi(\boldsymbol{p}, \boldsymbol{p}_n), \quad \boldsymbol{p}_n \in \boldsymbol{N_P} \tag{3.2}$$

$$\xi(\boldsymbol{p}, \boldsymbol{p}_n) = \begin{cases} 1, & if |I(\boldsymbol{p})| < |I(\boldsymbol{p}_n)| \\ 0, & otherwise \end{cases} \tag{3.3}$$

where $\otimes$ is the concatenation operator. The Census Transform based matching cost between $\boldsymbol{p}$ in the previous image and $\boldsymbol{p}$+$\boldsymbol{o}$ in the current image in defined as the Hamming Distance between two bit strings of Census Transform of $\boldsymbol{p}$ and $\boldsymbol{p}$+$\boldsymbol{o}$

$$C_{Census}(\boldsymbol{p}, \boldsymbol{o}) = Hamming(Census(\boldsymbol{p}), Census(\boldsymbol{p} + \boldsymbol{o})) \tag{3.4}$$

We choose the final pixel-wise matching cost to be a weighted linear combination of the BCA cost and Census Transform cost.

$$C(\boldsymbol{p}, \boldsymbol{o}) = \alpha \cdot C_{BCA}(\boldsymbol{p}, \boldsymbol{o}) + C_{Census}(\boldsymbol{p}, \boldsymbol{o}) \tag{3.5}$$

where α is the weight for the BCA cost. We choose the value of α to be proportional to the Census window size. For instance, through simulation results, we find that 0.06 for a 9×9 window and 0.1 for a 11×11 window gives the best performance.

In a typical implementation of SGM, the costs $C(\boldsymbol{p},\boldsymbol{o})$ are pre-calculated and stored in an integer array of size $W \times d \times D$. However, in the proposed NG-fSGM, since a subset of flow vectors is selected, only a small number of pixel-wise matching costs need to be calculated. Thus the calculation of $C(\boldsymbol{p},\boldsymbol{o})$ is part of the cost aggregation step and is performed only when $\boldsymbol{o}$ is selected.

There are typically two options to choose what data should be stored. One option is to store two original images, and compute both BCA cost and Census Transform cost on the fly. This option requires smaller memory ($W \times H \times 2$ bytes*)* but more computational cost since Census Transform is performed twice for each $C(\boldsymbol{p},\boldsymbol{o})$. Another option is to store two original images and pre-calculate Census Transform of two images. This option requires larger memory ($W \times H \times 32$ bytes for a 11×11 Census window) but has less computational burden. We make a tradeoff between these two options to address both memory size and number of operations. We store two original images and only a strip ($W \times d$) of Census Map. Thus for a 11×11 Census window, the storage requirement is $W \times d \times 15 + W \times H \times 2$ bytes. However, the Census Transform is now computed twice for each pixel due to two scans of the image.

3.3 Cost Aggregation and Flow Computation

Using pixel-wise matching cost to directly compute flow always results in large errors since a wrong flow vector can easily have a lower cost than the correct one. This is especially true in certain conditions such as non-texture, repeated pattern and occlusion regions. So pixel-wise matching costs need to be modified based on an additional constraint. Flow

smoothness is a popular constraint used to penalize changes of neighboring flow vectors. The proposed NG-fSGM use a single-step Potts model [29]. The pixel-wise matching cost and the smoothness constraint are embedded in a global energy function E that depends on the flow map F:

$$E(F) = \sum_{\boldsymbol{p}} [C(\boldsymbol{p}, \boldsymbol{o_p}) + \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}} \; and \; |\boldsymbol{o_p} - \boldsymbol{o_q}|^2 \leq 2} P_1 + \sum_{\boldsymbol{q} \in N_{\boldsymbol{p}} \; and \; |\boldsymbol{o_p} - \boldsymbol{o_q}|^2 > 2} P_2] \quad (3.6)$$

The first term in equation (3.6) is the sum of all pixel-wise matching costs for all pixels $\boldsymbol{p}$ and their corresponding flow vectors $\boldsymbol{o}$ in F. The second and third terms map the smoothness constraint onto the energy function. The second term adds a small penalty P_1 to all pixels $\boldsymbol{q}$ in the neighborhood $N_{\boldsymbol{p}}$ of $\boldsymbol{p}$, if $\boldsymbol{q}$'s flow vector $\boldsymbol{o_q}$ is different from but adjacent to $\boldsymbol{p}$'s flow vector $\boldsymbol{o_p}$. A smaller penalty allows smaller flow changes due to slanted surface, spinning objects etc. The third term adds a large penalty P_2 if there are any large changes in the neighboring flow vector. A larger penalty enables the algorithm to adapt to motion discontinuation. Penalties P_1 and P_2 are both external constant parameters.

To compute dense optical flow, we need to find the flow map that minimizes the global energy function, which is an NP-complete problem. Fortunately, the Semi-Global Matching method [11] can be used to aggregate matching costs in 1D from all directions equally. Since 1D optimization can be performed efficiently in polynomial time, the problem is tractable. Details of the SGM aggregation method was introduced in Section 2. In the proposed NG-fSGM, the cost $L_r(\boldsymbol{p},\boldsymbol{o})$ of the pixel $\boldsymbol{p}$ for a flow vector $\boldsymbol{o}$ accumulated along a path in the direction r is defined recursively as

$$L_r(\boldsymbol{p}, \boldsymbol{o}) = C(\boldsymbol{p}, \boldsymbol{o}) + Z - \min_{\boldsymbol{j}} L_r(\boldsymbol{p} - r, \boldsymbol{j}) \tag{3.7}$$

with the cost regularization summand

$$Z = \min\left[L_r(\boldsymbol{p} - r, \boldsymbol{o}), \min_{|\boldsymbol{i}-\boldsymbol{o}|^2 \leq 2} L_r(\boldsymbol{p} - r, \boldsymbol{i}) + P_1, \min_{\boldsymbol{j}} L_r(\boldsymbol{p} - r, \boldsymbol{j}) + P_2\right] \tag{3.8}$$

The cost regularization summand adds the smallest sum of cost of the previous pixel $\boldsymbol{p}$-r of the path r and the corresponding penalty.

Typical SGM-based methods store $L_r(\boldsymbol{p},\boldsymbol{o})$ in an array of size $W \times P \times D \times 2$ (for 8 paths) in order to compute the cost regularization summand Z. NG-fSGM uses the best N flow vectors $\boldsymbol{Q}$ for each path and their costs to approximate the original array so that the array size can be reduced to $W \times P \times N \times 2$. If the cost $L_r(\boldsymbol{p}$-$r,\boldsymbol{o})$ is not available in $\boldsymbol{Q}_{\boldsymbol{p}\text{-}r}$ along a certain direction r, it is assigned the minimum value in $\boldsymbol{Q}_{\boldsymbol{p}\text{-}r}$ plus P_2.

To compute flow, typical SGM-based methods store the overall aggregated cost $S(\boldsymbol{p},\boldsymbol{o})$ for all searched flow vectors O_p in an array of size $W \times H \times D$, and update the values by accumulating path-wise aggregated cost $L_r(\boldsymbol{p},\boldsymbol{o})$. NG-fSGM avoids such a large memory usage by storing only the N best flow vectors $\boldsymbol{B}_p$ and their corresponding aggregated cost $S_1(\boldsymbol{p},\boldsymbol{o})$ from forward scan to approximate $\boldsymbol{O}_{p1}$ in backward scan. As a result, the array size is reduced from $W \times H \times D$ to $W \times H \times N$.

To avoid the storage of aggregated cost in the backward scan, for each pixel $\boldsymbol{p}$, cost aggregation is directly followed by flow computation. The total number of flow vectors is

the union of best N vectors, $\boldsymbol{B_p}$, from forward scan and the neighbor-guided vectors from backward scan, $\boldsymbol{O_{p2}}$. For the vectors whose cost has not been calculated in either the forward or the backward scan, the following rules are applied to derive the missing costs. Specifically, the missing costs in forward scan are assigned the maximum cost in $\boldsymbol{B_p}$ plus P_2, while the missing costs in backward scan are assigned the maximum cost in $\boldsymbol{O_{p2}}$. The overall cost $S(\boldsymbol{p},\boldsymbol{o})$ is the sum of cost from two scans.

$$S(\boldsymbol{p},\boldsymbol{o}) = \sum_r L_r(\boldsymbol{p},\boldsymbol{o}) \tag{3.9}$$

Finally, the output flow vector $\boldsymbol{o}$ is the one corresponding to the minimum cost $S(\boldsymbol{p},\boldsymbol{o})$.

3.4 Post Processing

The initial estimated optical flow maps have outliers due to mismatches and occlusions. Additionally, semi-global methods are not robust to large areas with homogeneous, textureless and repeated patterns. Thus a post-processing step is needed to refine the flow map. Of the many post-processing methods that have been proposed in recent years, we choose the median filter. Specifically, after the initial flow results are computed, a 3 × 3 median filter is applied on both channels (horizontal and vertical components) of the flow map to remove errors and smoothen flow fields.

For higher accuracy, especially at object boundaries, a more complex method is used. A consistency check between previous and current frame (similar to left-right check for stereo [11]) can be applied to determine occlusions and false matches. If the current frame is at t

and the previous frame is at t-1, then a flow map $F_{t\text{-}1,t}$ is calculated by treating frame at t-1 as the base image and a flow map $F_{t,t\text{-}1}$ is calculated by treating frame at t as the base image. The invalid flow vectors are identified by comparing each flow vector in $F_{t\text{-}1,t}$ to its corresponding flow vector in $F_{t,t\text{-}1}$. If the flow vector at pixel (x,y) in $F_{t\text{-}1,t}$ is (u,v) and the flow vector at pixel $(x+u,y+v)$ is not $(-u,-v)$, then the flow vector in $F_{t\text{-}1,t}$ is set to be invalid. If the majority of the neighboring flow vectors of an invalid flow vector are valid, the flow vector is flagged as a mismatch; otherwise, it is considered as occlusion. A 3×3 median filtering operation is done around each mismatched flow vector to compute its value.

3.5 Complexity Analysis

We analyze the complexity of the proposed NG-fSGM with respect to memory size and number of operations. The parameters are as follows: the image size is $W \times H$, the number of best flow vectors is N, the number of random flow vectors is M, the number of paths per scan is P, the number of flow vectors in the search window is K, the maximum search space subset size is T where $T = KN(P+1) + M$, the flow search range is d, and the Census transform window size is $C \times C$. Also see Table 3.1. We provide the formulas to calculate both the memory size and the number of operations. We also provide numbers for a typical scenario with $W = 640$, $H = 480$, $N = 3$, $M = 3$, $P = 4$, $K = 9$, $d = 63$ and $C = 11$.

To estimate the minimum memory size, we analyze different data types and their life span in the algorithms. Let each step of the algorithm be denoted by a number: 1 – Flow subset selection; 2 – Pixel-wise matching cost computation; 3 – Cost aggregation; 4 – Best flow selection; 5 – Flow computation. The memory footprint is dominated by the second scan,

since the search space size in the second scan is larger. The larger search space is due to extra KN flow vectors from the first scan. Thus here we provide the memory size analysis for the second scan.

Table 3.2 presents the memory analysis. Two grayscale input images need to be stored all the time in order to compute BCA costs and Census transform. Census map for one pixel needs to be stored for the previous frame, while Census map for pixels within a $W \times d$ sliding window needs to be stored for the current frame. The Census map for each pixel requires $(C^2\text{-}1)/8$ bytes. Each flow vector requires 4 bytes – 1 byte for the horizontal component, 1 byte for the vertical component and 2 bytes for the cost. Thus all the memory to store cost-related data is scaled by a factor of 4 (see rows 3-7 in Table 3.2). Pixel-wise matching costs, aggregated costs along paths and overall aggregated costs need to be stored only temporarily. Since the maximum number of flow vectors is T, the corresponding memory size is proportional to T (see rows 4-6 in Table 3.2). For aggregated cost along paths, memory size is also proportional to number of paths per scan P, making the memory size $4PT$. In order to guide flow selection, a row of best N flow vectors along paths needs to be stored, thus the corresponding memory size is proportional to WPN. The overall best N flow vectors for every pixel need to be stored all the time, so the memory size for this step is proportional to WHN. The final flow map has 2 bytes per flow vector, one for the horizontal component and one for the vertical component.

For a typical scenario, we see that the memory size is dominated by the memory required to store overall best flow (3600 KB), followed by input images (600 KB), flow map (600

Table 3.2 Memory Size Analysis of NG-fSGM

Data Type	Life Span	Memory Size / B	Example/ KB
Input Images	1-5	$2WH$	600
Census Map	1-5	$(Wd+1)(C^2-1)/8$	590.64
Pixel-wise Matching Cost	2-3	$4T$	0.54
Aggregated Cost along Paths	3-4	$4PT$	2.16
Overall Aggregated Cost	3-5	$4T$	0.54
Best Flow along Paths	1-5	$4WPN$	30
Overall Best Flow	1-5	$4WHN$	3600
Flow Map	1-5	$2WH$	600

KB) and Census map (590.64 KB). Since the image size and flow range are fixed, the overall memory size is dominated by number of best flow vectors N and Census window size C.

Table 3.3 shows the computation analysis. Census transform requires $2C^2$ operations for each pixel, and thus $4C^2WH$ operations for each scan. The number of operations for flow subset selection, pixel-wise matching cost, cost aggregation and best flow selection is proportional to the flow subset size, which is T-KN in the first scan and T in the second scan. The flow subset selection step first selects flow vectors using a hash set with $NP(2+3K)+5M$ operations in the first scan and $NP(2+3K)+5(M+KN)$ in the second scan. Then it initializes temporary cost arrays with $(6+2P)$ operations for each selected flow vector. The pixel-wise matching cost for each flow vector is computed by calculating the Hamming distance with $2C^2$ operations and adding the BCA cost with 6 operations. The cost aggregation step compares the current flow vector with the best N flow vectors from

Table 3.3 Computation Analysis of NG-fSGM

Algorithm Steps		Number of Operations	Example / 10^9
1st Scan	Flow Subset Selection	$[(T\text{-}KN)(6+2P)+NP(2+3K)+5M]WH$	0.59
	Census Transform	$4C^2WH$	0.15
	Pixel-wise MC Computation	$(T\text{-}KN)(6+2C^2)WH$	8.46
	Cost Aggregation	$(T\text{-}KN)P(6N+18)WH$	4.91
	Best Flow Selection	$(T\text{-}KN)(P+1)(4N+2)WH$	2.39
2nd Scan	Flow Subset Selection	$[T(6+2P)+NP(2+3K)+5(M+KN)]WH$	0.71
	Census Transform	$4C^2WH$	0.15
	Pixel-wise MC Computation	$T(6+2C^2)WH$	10.51
	Cost Aggregation	$TP(6N+18)WH$	6.10
	Best Flow Selection	$TP(4N+2)WH$	2.37
	Flow Computation	$(4TN+3T)WH$	0.64

every path, thus the number of operations is proportional to P and N. The best flow selection step selects $N(P+1)$ flow vectors in the first scan and NP flow vectors in the second scan so the number of operations is proportional to $N(P+1)$ and NP, respectively. The flow computation first combines best N flow vectors in the first scan and T flow vectors in the second scan using $4TN$ operations, and then selects the output flow vector using $3T$ operations.

Based on this analysis, we see that the overall number of operations is dominated by pixel-wise matching cost computation and cost aggregation steps. Since $T = KN(P+1) + M$, the overall computation complexity is controlled by number of best flow vectors N, number of paths P, Census window size C and number of flow vectors in search window K.

3.6 Sparse-to-Dense Optical Flow Estimation

The proposed baseline optical flow algorithm has high complexity as shown in the previous section. Thus it is not applicable for some computer vision applications with extreme

computational, memory and power constraints. We address the complexity problem by estimating dense optical flow from sparse optical flow using interpolation techniques, where sparse flow vectors are computed by performing NG-fSGM on sampled pixels. Since the algorithm complexity of NG-fSGM is proportional to number of pixels, dense optical flow can be computed more efficiently using the proposed sparse-to-dense NG-fSGM method.

The idea of sparse-to-dense estimation is not new. Previous works always estimated sparse flow vectors based on tracked features [30]. Unfortunately, feature-based matching is not hardware-friendly; it introduces irregularities and typically has higher cost for both memory size and number of computations. Also, due to outliers in the sparse optical flow and uneven covering of the images, generic interpolations techniques such as linear interpolation and polynomial interpolation do not work well and more expensive learning-based interpolators are needed [31]. In this work, we propose to use simple interpolation of optical flow vectors of sampled pixels to address both complexity and accuracy concerns.

The sparse-to-dense NG-fSGM method starts by estimating sparse flow vectors for sampled pixels. Sampled pixels must be able to form a new image while maintaining their relative positions in the original image, so that cost aggregation can be done correctly. This puts a restriction on the number of possible subsampling methods. Figure 3.3 shows some examples of sampling patterns with different values of f_1 and f_2, where f_1 is the horizontal sampling rate and f_2 is the vertical sampling rate. After the new image is generated from sampled pixels, NG-fSGM computes a flow vector for each pixel in the new subsampled

image and then maps them to the previous location in the original image. Note that the pixel-wise matching cost used for computing the sparse flow vectors is based on the Census transform of the original image to avoid resolution loss.

Extending the sparse flow vectors to dense flow vectors is a regression problem. Interpolation is a good method for approximating the value of an unknown point given the values of its neighboring points. The proposed sparse-to-dense NG-fSGM method performs interpolation in two steps. Assuming that neighboring pixels tend to have identical or similar motion, to compute flow vector $\boldsymbol{o_p}$ at pixel $\boldsymbol{p}$, the algorithm selects the known flow vectors which have the minimum pixel distance as candidates. It is similar to nearest-neighbor interpolation which locates the nearest point and assigns the same value to the unknown point. This method is favored for high-dimensional interpolation because of its speed and simplicity. Second, assuming that neighboring pixels with identical color tend to belong to the same object and thus have similar motion, the proposed method selects the pixel $\boldsymbol{q}$ with the smallest intensity difference from pixel $\boldsymbol{p}$ among the candidates in the first step, and assigns the same flow vector $\boldsymbol{o_q}$ to pixel $\boldsymbol{p}$.

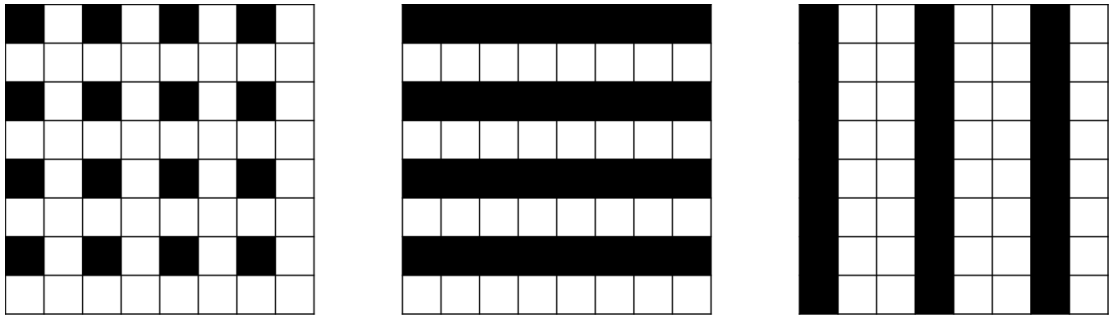

Figure 3.3 Sampling pattern examples where grey pixels are sampled. Left: $f_1 = ½, f_2 = ½$; Middle: $f_1 = ½, f_2 = 1$; Right: $f_1 = 1, f_2 = 1/3$.

The sparse-to-dense NG-fSGM method reduces memory size and number of operations. According to Table 3.2, the memory size is dominated by the storage required for overall best flow vectors. Through this method, that memory is reduced by a factor of $f_1 f_2$. Also, the memory for best flow vectors along path which stores flow vectors and costs for a row is reduced by a factor of f_2. The memory size for other data types remain the same. For the typical scenario, the total memory size is reduced from 5423 KB to 2708 KB, if $f_1 = 2$ and $f_2 = 2$.

According to Table 3.3, the number of operations for all steps except Census transform for the current frame is proportional to image size. Since the image size has effectively reduced by $f_1 f_2$, and the number of additional operations for interpolation step is negligible, the total number of operations is only $f_1 f_2$ of the original. For the typical scenario, the total number of operations is reduced from 36.97×10^9 to 9.46×10^9, if $f_1 = 2$ and $f_2 = 2$.

CHAPTER 4. PARALLEL BLOCK-BASED NG-FSGM

In order to reduce memory space requirement and to improve latency, throughput and power efficiency of NG-fSGM, we present a parallel block-based NG-fSGM method. An earlier version of this work appeared in [12]. Recall that the memory size and number of operations in NG-fSGM is a function of the image size. Thus for the block-based method, the memory space requirement and latency are significantly smaller since they are proportional to the block size. Also, the throughput linearly improves with the number of parallel block processing cores. So for the same throughput, multiple parallel block processing cores can operate at lower clock frequency and lower voltage, thereby achieving lower power.

In the proposed block-based NG-fSGM method, input images are divided into overlapping blocks and the blocks are processed in parallel. While larger overlap between blocks improves the accuracy, it results in larger memory and higher computational complexity. So we propose to minimize the amount of overlap among blocks and address the accuracy issues by using a new method called inertial guidance. In this method, temporal information is used to estimate a flow map and guide flow vector selections along the block boundary.

4.1 Image Block Partitioning for Parallel Processing

In the naive block-based implementation, the input frames are partitioned into non-overlapping blocks and the NG-fSGM algorithm is applied to each block in parallel. Unfortunately, this non-overlapping block approach reduces the algorithm accuracy. The

performance degradation is due to several factors. The first factor is due to the flow subset selection step of NG-fSGM. For pixels at the image (block) boundary, NG-fSGM uses random flow vectors to initialize the search subset so it takes some time (in term of pixel propagation) until correct flow vectors appear in neighbor guided propagation. Second, since the aggregating paths accumulate information from boundary to inner pixels, for pixels at the boundary, not enough flow and cost information is accumulated from certain directions making the cost aggregation relatively unreliable. The third factor applies to all global optical flow algorithms using block partitioning for parallel processing. The flow smoothness constraint is interrupted at the boundary between two blocks, which results in wrong flow estimations especially for blocks with ambiguous texture.

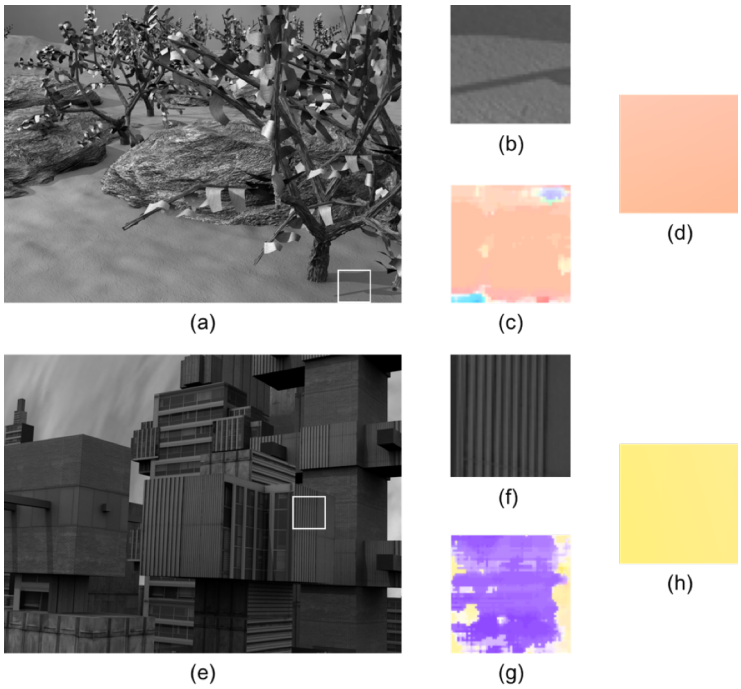

Figure 4.1 Two types of blocks extracted from the images in Middlebury dataset. (a), (e) Grove3 and Urban3; (b), (f) zoomed-in blocks of Grove3 and Urban3; (c), (g) colored flow maps obtained by directly applying original NG-fSGM on the marked block; (d), (h) colored flow maps using original NG-fSGM on the entire image.

Figure 4.1 shows two types of blocks with either low-texture or repeated-patterns extracted from the images in the Middlebury dataset. When NG-fSGM is applied to the highlighted block (Figure 4.1(b)) in Grove3 image (Figure 4.1(a)), errors appear near the boundary of the block (Figure 4.1(c)). For comparison, Figure 4.1(d) shows the result with no errors when the NG-fSGM is applied to the entire image without block segmentation. The block with highly repeated patterns, Figure 4.1(f), captured from Urban3 (Figure 4.1(e)) suffers from ambiguity during pixel-wise matching in non-overlapping block processing. Incorrect flow estimates at block boundaries propagate over the block (Figure 4.1(g)) while the original NG-fSGM over the entire image correctly estimates the flow for the same region (Figure 4.1(h)).

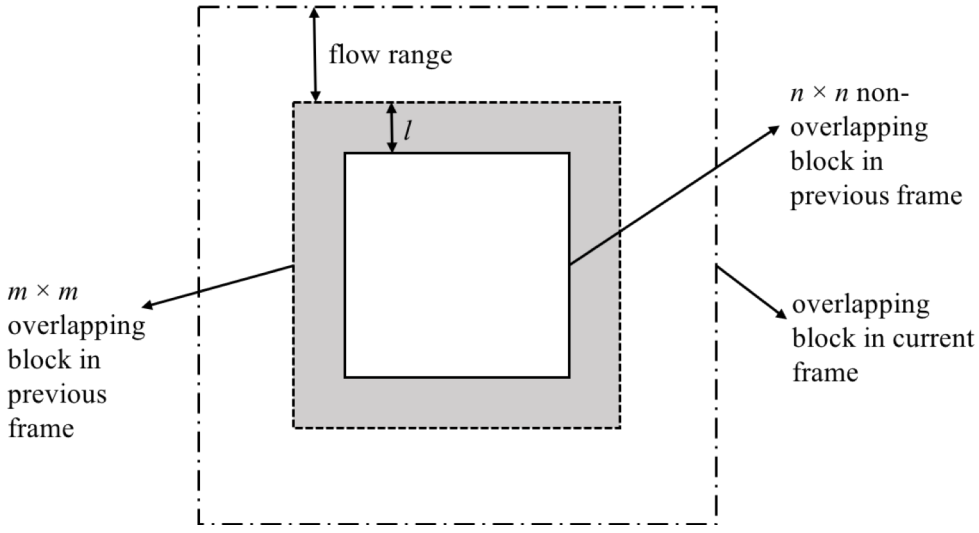

Figure 4.2 An example of an $n \times n$ non-overlapping block in previous frame. The corresponding $m \times m$ overlapping block in previous frame is obtained by extending l pixels along four sides, and the corresponding overlapping block in current frame is obtained by further extending $d/2$ pixels along four sides.

Table 4.1 Influence of Block Size $n \times n$ and Extension l on Accuracy, Latency and Architectural Complexity

Parameters	Accuracy	Latency	Complexity
$n \uparrow$	$\uparrow$	$\uparrow\uparrow$	$\downarrow$
$l \uparrow$	$\uparrow\uparrow$	$\uparrow\uparrow$	$\uparrow\uparrow$

In order to improve the accuracy of block-based NG-fSGM, the three factors described earlier need to be addressed. We propose an overlapping block partitioning scheme where the previous frame is first divided into $n \times n$ non-overlapping blocks, and then each block is extended by l pixels along four sides, resulting in $m \times m$ overlapping blocks, where $m = n + 2l$. The current frame is divided into overlapping blocks as well, but with block size of $m + d - 1$ per dimension, where d is the flow search range. The output flow map of the entire image is built using the flow map of each $n \times n$ block. Figure 4.2 shows an example of non-overlapping block and its extensions. Since the flow vectors of the boundary pixels in each overlapping block are not considered in the final flow map, errors corresponding to the first two factors, namely flow subset initialization and unreliable cost aggregation, can be reduced. Additionally, flow smoothness constraints can be imposed across the block boundary.

Generally, larger size of non-overlapping block and larger number of extended pixels result in better flow accuracy, at the expense of higher latency, computational cost and memory requirement. For a certain image size, the parameter n defines the number of blocks to be processed, and together with l, n also defines the complexity that is a function of the memory size, memory bandwidth and the number of operations. Table 4.1 summarizes the effect of parameters n and l on flow accuracy, latency and complexity. The accuracy is

based on experimental evaluation on the Middlebury dataset. The latency is linear in the size of overlapping block, $m = n + 2l$. Larger n and l both increase the memory and computation requirements per block, though larger n reduces the number of blocks to be processed.

While overlapping of the blocks can significantly improve the flow accuracy, it also increases the algorithm complexity from $O(n^2)$ to $O(m^2)$, where $m = n + 2l$. In Section 4.2, we describe a method to reduce the overlapping overhead by utilizing temporal information from the previous frame to guide the flow vector selection at the block boundary.

If sparse-to-dense flow estimation is applied to the block-based NG-fSGM, the block size will influence the overall accuracy. The smaller block size results in lower accuracy of estimated sparse flow vectors and thus less reliable interpolation due to more outliers of the sparse estimation. So the smaller block size can only support lower subsampling levels (i.e. smaller f_1 and f_2). Since the complexity of the sparse-to-dense block-based NG-fSGM is proportional to m, f_1 and f_2, we need to make a tradeoff between block size and subsampling level.

4.2 Inertial Flow Vector Guidance using Multiple Frames

The overlapping region in the block-based NG-fSGM method can be reduced if good flow vector candidates are included in the search space of the boundary pixels. We propose to use temporal information from the previous pair of frames to provide good candidates. Using information from multiple frames to compute optical flow has been introduced in

many methods. Irani [32] presented an approach for simultaneous estimation of optical flow across multiple frames based on the observation that the set of all flow fields across multiple frames reside in a low-dimensional linear subspace. Based on Irani's method, Chia-Ming et al. [33] introduced a feature-based algorithm based on the image flow constraint equation in a local patch across multiple frames. Recently, Kennedy and Taylor [34] proposed an optical flow estimation method of dealing with temporal information in multiple frames by using 'inertial estimates' of the flow. They first used a base algorithm to estimate flow for three pairs of frames, then combined these estimates using a classifier-based fusion scheme, which significantly improves results. Our method is inspired by 'inertial estimates' in [34] but we use our base algorithm only once for each pair of frames.

Assuming that each object in the scene is carried by inertia and thus moves at a constant velocity, and also moves parallel to the image plane, then the flow map $F_{t,\,t+1}$ between frames [t, t+1] can be estimated from the flow map $F_{t-1,\,t}$ between frames [t-1, t]. Let (i, j) denote pixel index, and let (u, v)$_{(I,\,j)}$ and (u', v')$_{(I,\,j)}$ denote flow of pixel (i, j) between time frames [t, t + 1] and [t – 1, t], respectively. Then the following relationship holds.

$$(u, v)_{(i+u', j+v')} = (u', v')_{(i,j)} \tag{4.1}$$

We use the equation to provide the inertial guided flow estimates. For each pixel in the extended region (grey area in Figure 4.2) in the $m \times m$ block, inertial guided flow estimates are more reliable, in general, than neighbor guided flows especially when the neighbors are closer to the block boundary where flow vectors are randomly initialized. Thus, we add the inertial guided flow vector to the search subset. This approach helps in reducing the size of

Table 4.2 Memory Size Analysis of Block-based NG-fSGM per Block

Data Type	Life Span	Memory Size / B	Example/ KB
Input Images	1-5	$(m+C)^2+(m+d+C)^2$	27.54
Census Map	1-5	$[(m+d)d+1](C^2-1)/8$	124.60
Pixel-wise Matching Cost	2-3	$4T$	0.54
Aggregated Cost along Paths	3-4	$4PT$	2.16
Overall Aggregated Cost	3-5	$4T$	0.54
Best Flow along Paths	1-5	$4mPN$	3.38
Overall Best Flow	1-5	$4m^2N$	60.75
Flow Map	1-5	$2m^2$	10.13
Inertial Estimation	1-5	$2(m^2-n^2)$	2.13

the extended region for overlapped block processing with very little degradation in algorithm accuracy.

4.3 Complexity Analysis

The complexity of the block-based NG-fSGM for each overlapping block is derived along the same lines as the analysis in Section 3.5. However there are some minor differences. First, if Census map within a strip is computed on the fly, the block boundary is extended by $C/2$ pixels, where $C \times C$ is the Census window size. The block boundary in current frame is further extended by $d/2$ pixels, where d is the flow search range. Thus the memory size for input image blocks is increased to $(m+C)^2$ in previous frame and $(m+d+C)^2$ in current frame. Second, in order to implement inertial flow vector guidance, the previous flow map requires additional memory for the overlapping pixels. Third, since only one additional inertial guided flow vector is considered for only boundary pixels, their additional complexity is negligible.

Table 4.3 Computation Analysis of Block-based NG-fSGM per Block

Algorithm Step		Number of Operations	Example / 10^6
1st Scan	Flow Subset Selection	$[(T\text{-}KN)(6+2P)+NP(2+3K)+5M]m^2$	9.9
	Census Transform	$2C^2m^2+2C^2(m+d)^2$	5.7
	Pixel-wise MC Computation	$(T\text{-}KN)(6+2C^2)m^2$	142.7
	Cost Aggregation	$(T\text{-}KN)P(6N+18)m^2$	82.9
	Best Flow Selection	$(T\text{-}KN)(P+1)(4N+2)m^2$	40.3
2nd Scan	Flow Subset Selection	$[T(6+2P)+NP(2+3K)+5(M+KN)]m^2$	11.9
	Census Transform	$2C^2m^2+2C^2(m+d)^2$	5.7
	Pixel-wise MC Computation	$T(6+2C^2)m^2$	177.4
	Cost Aggregation	$TP(6N+18)m^2$	103.0
	Best Flow Selection	$TP(4N+2)m^2$	40.1
	Flow Computation	$(4TN+3T)n^2$	10.7

Table 4.2 and 4.3 shows the analysis of memory size and number of operations respectively. We also provide the numbers for a typical set of parameters: $n = 64$, $m = 72$, $N = 3$, $M = 3$, $P = 4$, $K = 9$, $d = 63$ and $C = 11$. Like the basic NG-fSGM, the memory size is dominated by the memory required to store Census map, overall best flow and input images. The distribution is as follows: Census map 54%, overall best flow 26% and input images 12%. The number of operations is dominated by pixel-wise matching cost computation (51%) followed by cost aggregation (29%).

CHAPTER 5. EXPERIMENTAL RESULTS

In this section, we evaluate the proposed algorithms on the Middlebury optical flow dataset [2]. Our baseline optical flow method is denoted as NG-fSGM. When the proposed sparse-to-dense flow estimation, basic block-based processing and inertial flow vector guidance techniques are used, NG-fSGM is appended with "S" (sparse-to-dense), "B" (block) and "I" (inertial), respectively.

5.1 Benchmarks and Evaluation Methodology

We evaluated the proposed algorithm with respect to accuracy and complexity on the Middlebury optical flow benchmark [2]. Average endpoint error percentage was used as the metric for accuracy. Here we briefly review the Middlebury benchmark and the evaluation methodology.

5.1.1 Middlebury

Middlebury optical flow benchmark was introduced by S. Baker et al. in 2010 [2] for a new generation of optical flow algorithms. It provided four types for data to test different

Table 5.1 Attributes of Middlebury Training Set

Scene Name	**Number of Frames**	**Type**	**Max Flow Range**
Dimetrodon	2	Hidden Texture	11
Grove2	8	Synthetic	11
Grove3	8	Synthetic	31
Hydrangea	8	Hidden Texture	25
RubberWhale	8	Hidden Texture	11
Urban2	8	Synthetic	45
Urban3	8	Synthetic	37
Venus	2	Stereo	21

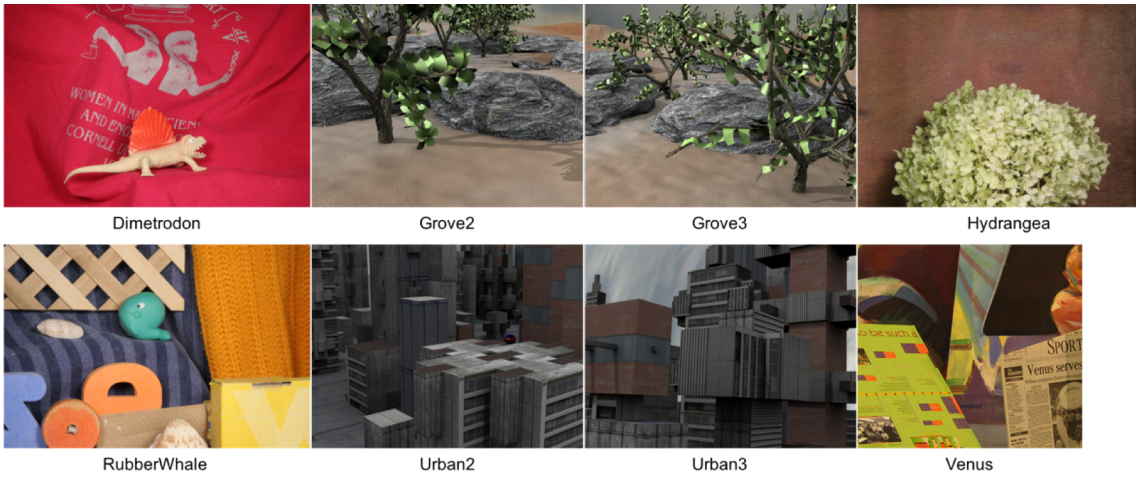

Figure 5.1 Middlebury training set with public ground truth

aspects of optical flow algorithms: (1) sequences with non-rigid motion where the ground-truth flow is determined by tracking hidden fluorescent texture, (2) realistic synthetic sequences, (3) high frame-rate video used to study interpolation error, and (4) modified stereo sequences of static scenes.

There are eight scenes (shown in Figure 5.1) whose ground-truth flow is publicly available, thus we used these scenes for the evaluation of the proposed algorithms. The scenes provide dense and subpixel accurate ground truth with occlusion mask. Table 5.1 summarizes the attributes of each scene.

5.1.2 Evaluation Methodology

The accuracy evaluation methodology proposed in [2], [35] has been used here. It includes (1) the performance measure, (2) the statistics, (3) the sub-region of the images considered.

The Endpoint error (EE) between a flow vector (u, v) and the ground-truth flow (u_{GT}, v_{GT}) is defined as

$$EE = \sqrt{(u - u_{GT})^2 + (v - v_{GT})^2} \tag{5.1}$$

The statistics of the performance measure results, such as averages and standard deviations, are computed for robustness of evaluation. The error percentage, R*X*, defined as the percentage of pixels that have an error measure above *X*, is commonly used [2]. Also, region masks are often used in evaluation since it is difficult to compute flow in certain regions such as motion discontinuities and textureless regions.

In this thesis, we used endpoint error percentage (EEP) to evaluate the estimation accuracy of optical flow [2]. We computed R2.0 (pixels) for the endpoint error for all pixels except the occlusion region where flow vector is impossible to obtain, and pixels at image boundary.

The complexity is measured in terms of the memory size and the number of arithmetic/logic operations. The number of operations listed are not the worst-case values that were presented in Table 3.2 but average values based on the simulations on different images. Note that the number of operations depends on the percentage of selected flow vectors over the entire search space.

Table 5.2 Sensitivity of Algorithm Parameters of NG-fSGM

Parameters	**Impact on Accuracy**	**Impact on Memory**	**Impact on Operations**
Census Size	low	low	moderate
P	moderate	low	high
N	moderate	high	high
M	high	low	moderate
Search Size	moderate	low	high

5.2 NG-fSGM Results

5.2.1 Impact of Algorithm Parameters in NG-fSGM

We conducted comprehensive experiments to quantify the impact of various algorithm parameters on accuracy, memory size and number of operations. A high-level summary of our findings is given in Table 5.2.

The method to find the set of parameters with the best accuracy was as follows: 1) initialize each parameter with a reasonable value ($C = 11$, $P = 4$, $N = 3$, $M = 3$, $K = 9$) [10]; 2) update one parameter value at a time such that the accuracy improves; 3) repeat step 2 until

Table 5.3 Effect of Census Window Size $C \times C$ on the Accuracy, Memory Size and Number of Operations

$\boldsymbol{C \times C}$	**7 × 7**	**9 × 9**	**11 × 11**
Dimetrodon	0.00	0.00	0.00
Grove2	1.54	1.62	1.72
Grove3	6.97	7.21	7.39
Hydrangea	0.69	0.74	0.78
RubberWhale	0.63	0.71	0.87
Urban2	4.26	4.03	3.78
Urban3	11.02	10.15	11.48
Venus	2.07	2.18	2.34
Mean	**3.40**	**3.33**	**3.54**
Memory / MB	3.06	3.12	3.19
Operations / 10^9	0.71	0.91	1.16

parameter values have stabilized. We found that the following parameters gave the best performance in terms of accuracy: $C = 9$, $P = 4$, $N = 2$, $M = 4$, $K = 1$, $P_1 = 12$, $P_2 = 45$, $\alpha = 0.06$. Next we provide results showing the impact of each parameter on the accuracy, memory and computation. We vary the value of one parameter at a time; the rest of the parameters are assigned values corresponding to the ones for best accuracy.

Table 5.3 shows the results for different Census window size. The Census window size that provides the highest accuracy varies for different scenes due to different image structures. The result showed that a 9×9 window provides the best average accuracy. As for complexity, the Census window size has a low impact on memory size and a moderate impact on number of operations.

In NG-fSGM, number of paths P, number of best flow vectors N, number of random flow vectors M and number of flow vectors in the search window K are the key parameters that

Table 5.4 Effect of Number of Paths P per Scan on the Accuracy, Memory Size and Number of Operations

P	**2**	**4**	**8**
Dimetrodon	0.00	0.00	0.00
Grove2	1.69	1.62	1.61
Grove3	7.36	7.21	7.37
Hydrangea	0.71	0.74	0.67
RubberWhale	0.99	0.71	0.68
Urban2	4.08	4.03	3.70
Urban3	11.48	10.15	10.48
Venus	2.14	2.18	2.21
Mean	**3.56**	**3.33**	**3.34**
Memory / MB	3.11	3.12	3.14
Operations / 10^9	0.64	0.91	1.53

Table 5.5 Effect of Number of Best Flow Vectors *N* on the Accuracy, Memory Size and Number of Operations

N	**1**	**2**	**3**	**4**
Dimetrodon	0.03	0.00	0.00	0.00
Grove2	2.10	1.62	1.49	1.44
Grove3	7.96	7.21	7.02	7.00
Hydrangea	1.40	0.74	0.65	0.57
RubberWhale	1.42	0.71	0.63	0.59
Urban2	4.65	4.03	3.77	3.86
Urban3	7.24	10.15	12.83	14.12
Venus	2.83	2.18	2.07	2.10
Mean	**3.45**	**3.33**	**3.56**	**3.71**
Memory / MB	2.13	3.12	4.11	5.10
Operations / 10^9	0.59	0.91	1.29	1.73

control the algorithm accuracy and complexity by changing the number of selected flow vectors in the search space. Table 5.4, 5.5, 5.6 and 5.7 show the effect of *P*, *N*, *M* and *K* on the accuracy, memory size and number of operations, respectively.

Table 5.4 shows the results for different number of paths *P*. Recall that *P* decides the number of neighboring pixels explored in the flow subset selection, and the number of directions aggregated in the cost aggregation. We see that *P* has almost no impact on memory size but the number of operations increases with *P*. The results showed that the accuracy converges when $P = 4$ and thus 4 paths per scan provide a reasonable design point.

Table 5.5 shows the results for different values of *N*, the number of best flow vectors. We see that the algorithm has the best performance when $N = 2$. However the mean accuracy is dominated by scene Urban3 whose accuracy decreases when *N* becomes larger. For the other scenes, the accuracy tends to converge when $N = 3$. As for complexity, *N* has a high

Table 5.6 Effect of Number of Random Flow Vectors *M* on the Accuracy, Memory Size and Number of Operations

M	**0**	**1**	**2**	**3**	**4**	**5**
Dimetrodon	0.03	0.00	0.00	0.00	0.00	0.00
Grove2	1.75	1.67	1.61	1.65	1.62	1.65
Grove3	34.89	7.99	7.19	7.3	7.21	7.13
Hydrangea	5.32	0.88	0.72	0.76	0.74	0.69
RubberWhale	2.01	0.85	0.81	0.89	0.71	0.7
Urban2	28.16	4.35	3.95	4.08	4.03	3.94
Urban3	30.23	12.09	11.13	11.51	10.15	9.95
Venus	3.39	2.28	2.22	2.25	2.18	2.26
Mean	**13.22**	**3.76**	**3.45**	**3.56**	**3.33**	**3.29**
Memory / MB	3.12	3.12	3.12	3.12	3.12	3.12
Operations / 10^9	0.45	0.55	0.67	0.81	0.91	1.04

impact on number of selected flow vectors and thus a high impact on both memory size and number of operations.

Table 5.6 shows the results for different values of *M*, the number of random flow vectors. While there is a significant improvement in accuracy compared to when $M = 0$, the relative improvement diminishes with increasing *M*. Unlike *N*, the value of *M* has almost no impact on memory size. However it has a moderate impact on number of operations. This is

Table 5.7 Effect of Number of Flow Vectors in Search Window *K* on the Accuracy, Memory Size and Number of Operations

K	**1**	**5**	**9**
Dimetrodon	0.00	0.01	0.04
Grove2	1.62	1.54	1.52
Grove3	7.21	7.8	7.91
Hydrangea	0.74	0.71	0.74
RubberWhale	0.71	0.64	0.64
Urban2	4.03	4.61	4.97
Urban3	10.15	12.3	13.18
Venus	2.18	2.72	1.98
Mean	**3.33**	**3.79**	**3.99**
Memory / MB	3.12	3.12	3.12
Operations / 10^9	0.91	2.78	4.22

because a random flow vector is less likely to be identical to other selected flow vectors and thus a larger *M* results in an increase in the number of selected flow vectors.

Table 5.7 shows the result for different *K*, the number of selected flow vectors in the search window. Since *K* is related to the search subset size, it has a high impact on both memory size and number of operations. Surprisingly, the simulations show that $K = 1$ provides the highest accuracy. There are two possible reasons for this. First, with a small search subset and a simple cost aggregation function, small *K* adds a flow smoothness constraint. Second, there is not a large amount of small motion variations in the Middlebury dataset.

In our previous evaluation [10], *K*, the number of flow vectors in the search window, was fixed at 9. In that scenario, *M* was only 3 and had a low impact on both accuracy and complexity. However, when *K* is smaller, *M* has a higher impact on accuracy and complexity as shown here. Thus we conclude that in order to get the same accuracy, *M* should be large when *K* is small, and *M* should be small when *K* is large. But since the complexity is dominated by *K*, a combination of large *M* and small *K* is preferred.

5.2.2 Comparison between NG-fSGM and Other Methods

We compared NG-fSGM with Lucas-Kanade [36], probably the most popular optical flow method, and fSGM [9], a prior work using Semi-Global Matching, both using a single-level pyramid scheme and the same post-processing step (Section 3.4). Note that all three methods can be embedded in a hierarchical scheme if necessary. For fSGM, the parameters used were: 11 × 11 Census transform window, 8 aggregation paths, one-step cost

Table 5.8 Comparison of NG-fSGM, fSGM and Lucas-Kanade

Algorithm	**NG-fSGM**	**fSGM**	**Lucas-Kanade**
Dimetrodon	0.00	0.00	1.64
Grove2	1.62	1.56	4.67
Grove3	7.21	7.11	20.02
Hydrangea	0.74	0.62	6.84
RubberWhale	0.71	0.81	4.12
Urban2	4.03	4.51	16.14
Urban3	10.15	14.26	24.46
Venus	2.18	1.99	9.35
Mean	**3.33**	**3.86**	**10.91**
Memory / MB	3.12	20.68	1.47
Operations / 10^9	0.91	37.53	3.15

regularization summand, $P_1 = 40$, and $P_2 = 200$. Penalties in fSGM are chosen to be larger since NG-fSGM is more likely to get larger values from other flow vectors.

Table 5.8 shows the comparison of NG-fSGM, fSGM and Lucas-Kanade with respect to accuracy and complexity. Our algorithm achieves higher accuracy compared to fSGM since neighbor-guided flow vector selection performs well with a simple cost aggregation function. Neighbor-guided flow selection eliminates wrong flow vectors in the first place, especially for scene Urban3, where large areas of repeated patterns results in mismatches using pixel-wise matching cost. NG-fSGM also provides significant benefit in complexity since the number of flow vectors in the search subset of NG-fSGM is only 14 for each pixel. We also observe that NG-fSGM significantly outperforms Lucas-Kanade in accuracy and number of operations at the cost of increased memory size.

To visualize the accuracy difference, Figure 5.2 shows the flow maps of three types of scenes on Middlebury for each algorithm. NG-fSGM and fSGM both outperforms Lucas-Kanade. The flow map output of NG-fSGM shows more noise, especially along object

edges, but has fewer error patches compared to fSGM. The neighbor dependency in NG-fSGM is less reliable at the object edges when we apply aggressive search space pruning. However it suffers less from the problem of repeated patterns, such as in Urban2 and Urban3. Table 5.8 confirms that the overall accuracy of NG-fSGM is higher than fSGM.

Figure 5.2 Colored flow maps for 3 types of scenes using different algorithms. 1st column: RubberWhale; 2nd column: Urban2; 3rd column: Venus. 1st row: input previous frame; 2nd row: NG-fSGM; 3rd row: fSGM; 4th row: Lucas-Kanade. Color legend is at bottom-left corner.

Table 5.9 Accuracy, Memory Size and Number of Operations of NG-fSGM+S

f_1	**1**	**2**	**1**	**2**	**3**	**2**	**3**
f_2	**1**	**1**	**2**	**2**	**2**	**3**	**3**
Dimetrodon	0.00	0.01	0.00	0.02	0.03	0.03	0.02
Grove2	1.62	1.77	1.76	2.01	2.02	2.16	2.08
Grove3	7.21	7.38	7.46	8.08	8.95	9.16	10.23
Hydrangea	0.74	0.89	0.79	0.99	1.47	1.39	1.59
RubberWhale	0.71	0.91	0.93	1.05	1.22	1.28	1.22
Urban2	4.03	3.95	4.27	4.68	6.59	10.95	8.04
Urban3	10.15	10.47	10.12	10.95	11.06	11.61	10.09
Venus	2.18	2.2	2.18	2.21	2.46	2.54	2.65
Mean	**3.33**	**3.45**	**3.44**	**3.75**	**4.23**	**4.89**	**4.49**
Memory / MB	3.12	2.13	2.13	1.64	1.48	1.48	1.37
Operations / 10^9	0.91	0.50	0.50	0.29	0.22	0.22	0.17

5.2.3 Sparse-to-Dense NG-fSGM Results

We evaluated the sparse-to-dense NG-fSGM with different sampling rates f_1 and f_2, as shown in Table 5.9. The memory size and number of operations are reduced significantly at the expense of accuracy degradation. The sampling rates, $f_1 = 2$ and $f_2 = 2$, provides a reasonable tradeoff between accuracy and complexity. The memory size is reduced by 47% and number of operations is reduced by 68% with only 0.42% increment in error percentage.

Scenes with large flow displacements and refined image structures, such as Grove3 and Urban2, suffer more from subsampling than scenes with small flow displacement and unrefined image structures, such as Dimetrodon and Grove2. Figure 5.3 shows two examples for good and bad performance of sparse-to-dense flow estimation when $f_1 = 3$ and $f_2 = 3$.

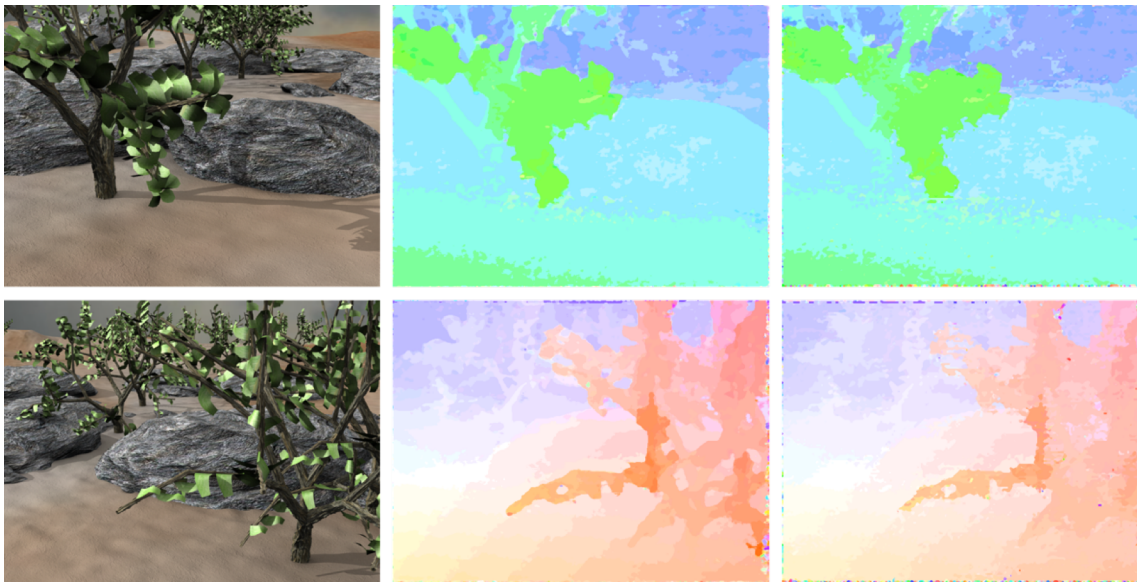

Figure 5.3 Colored flow maps for two types of scenes using NG-fSGM and sparse-to-dense NG-fSGM. 1st row: Grove2, small flow range and unrefined image structures; 2nd row: Grove3, large flow range and refined image structures. 1st column: input previous frame; 2nd column: NG-fSGM; 3rd: sparse-to-dense NG-fSGM where $f_1 = 3$ and $f_2 = 3$.

5.3 Block-based NG-fSGM Results

5.3.1 Impact of Parameters in Block-based NG-fSGM

We evaluate the effect of parameters of block-based NG-fSGM, n (block size) and l (overlap size), on the algorithm performance in terms of accuracy, memory size and number of operations. The other parameters are set to: $C = 9$, $P = 4$, $N = 2$, $M = 4$, $K = 1$, $P_1 = 12$, $P_2 = 45$, $\alpha = 0.06$. The parameter d is chosen based on the maximum flow range. This set of parameters is found to give the best performance in Section 5.2.1.

Table 5.10 Effect of Block Size *n* on the Accuracy, Memory Size and Number of Operations

n	**20**	**40**	**60**	**80**	**100**	**120**
Dimetrodon	0.06	0.02	0.00	0.00	0.00	0
Grove2	1.66	1.66	1.61	1.61	1.58	1.61
Grove3	7.60	7.64	7.22	7.35	7.30	7.25
Hydrangea	0.78	0.75	0.75	0.75	0.71	0.7
RubberWhale	0.76	0.70	0.75	0.67	0.75	0.77
Urban2	9.38	5.92	4.60	5.14	5.17	4.47
Urban3	22.73	17.56	15.76	15.29	13.87	13.86
Venus	2.44	2.26	2.30	2.39	2.25	2.18
Mean	**5.68**	**4.56**	**4.12**	**4.15**	**3.95**	**3.86**
Memory / KB	41.97	72.12	111.65	160.56	218.84	286.50
Operations / 10^6	4.57	11.06	20.37	32.50	47.45	65.23

First, we evaluate the basic block-based processing (NG-fSGM+B) with block size *n* varying from 20 to 120, when overlap size *l* fixed to 8. The results of accuracy, memory size and number of operations per block are shown in Table 5.10. We see that larger *n* results in higher accuracy but also higher complexity. Scenes with large flow range (such as Urban2 and Urban3) requires larger *n* for good accuracy. The accuracy of scenes with small flow range (such as Grove2 and Hydrangea) did not change much with *n*. When *n* is

Table 5.11 Effect of Overlap Size *l* on the Accuracy, Memory Size and Number of Operations

l	**0**	**4**	**8**	**12**	**16**	**20**
Dimetrodon	0.05	0.00	0.00	0.00	0.00	0.00
Grove2	1.80	1.62	1.64	1.63	1.64	1.64
Grove3	7.70	7.57	7.02	7.29	7.13	7.21
Hydrangea	0.85	0.81	0.73	0.77	0.73	0.73
RubberWhale	0.83	0.74	0.70	0.73	0.71	0.72
Urban2	7.39	5.75	5.21	4.74	5.01	4.29
Urban3	15.94	17.40	14.69	13.37	13.69	13.49
Venus	2.37	2.30	2.37	2.17	2.17	2.25
Mean	**4.62**	**4.52**	**4.05**	**3.84**	**3.90**	**3.79**
Memory / KB	86.81	103.00	120.68	139.87	160.56	182.75
Operations / 10^6	14.45	18.28	22.57	27.31	32.50	38.14

Table 5.12 Comparison Between NG-fSGM+B and NG-fSGM+BI

Algorithm	**NG-fSGM+B**			**NG-fSGM+BI**		
l	**4**	**8**	**16**	**2**	**4**	**8**
Grove2	1.66	1.61	1.58	1.59	1.60	1.62
Grove3	7.64	7.22	7.30	7.12	7.16	6.82
Hydrangea	0.75	0.75	0.71	0.72	0.73	0.72
RubberWhale	0.70	0.75	0.75	0.66	0.65	0.66
Urban2	5.92	4.60	5.17	5.53	4.64	4.33
Urban3	17.56	15.76	13.87	13.71	13.95	13.67
Mean	**5.71**	**5.12**	**4.90**	**4.89**	**4.79**	**4.64**
Memory / KB	103.00	120.68	160.56	95.75	105.12	125.18
Operations / 10^6	18.28	22.57	32.50	16.31	18.28	22.57

larger than 60, the mean accuracy improvement becomes smaller, thus n = 60 provides a good balance between accuracy and complexity. Since the image size is typically a multiple of 64, n is set to 64 for the rest of the evaluation.

Next we evaluate NG-fSGM+B with overlap size l varying from 0 to 20. The results are shown in Table 5.11. We see that, similar to block size n, larger l results in higher accuracy and higher complexity, especially for the scenes with large flow range. The accuracy tends to converge when l = 12.

5.3.2 Block-based NG-fSGM using Inertial Guidance Results

We evaluate the effect of inertial guidance on 6 images that have multiple frames. Figure 5.4 shows an example of use of inertial guidance. Since the dataset has small motion and small occlusion region, the performance of inertial estimates from [t – 1, t] is good except for occlusion region, implying that the inertial estimates help guide flow vector selection well.

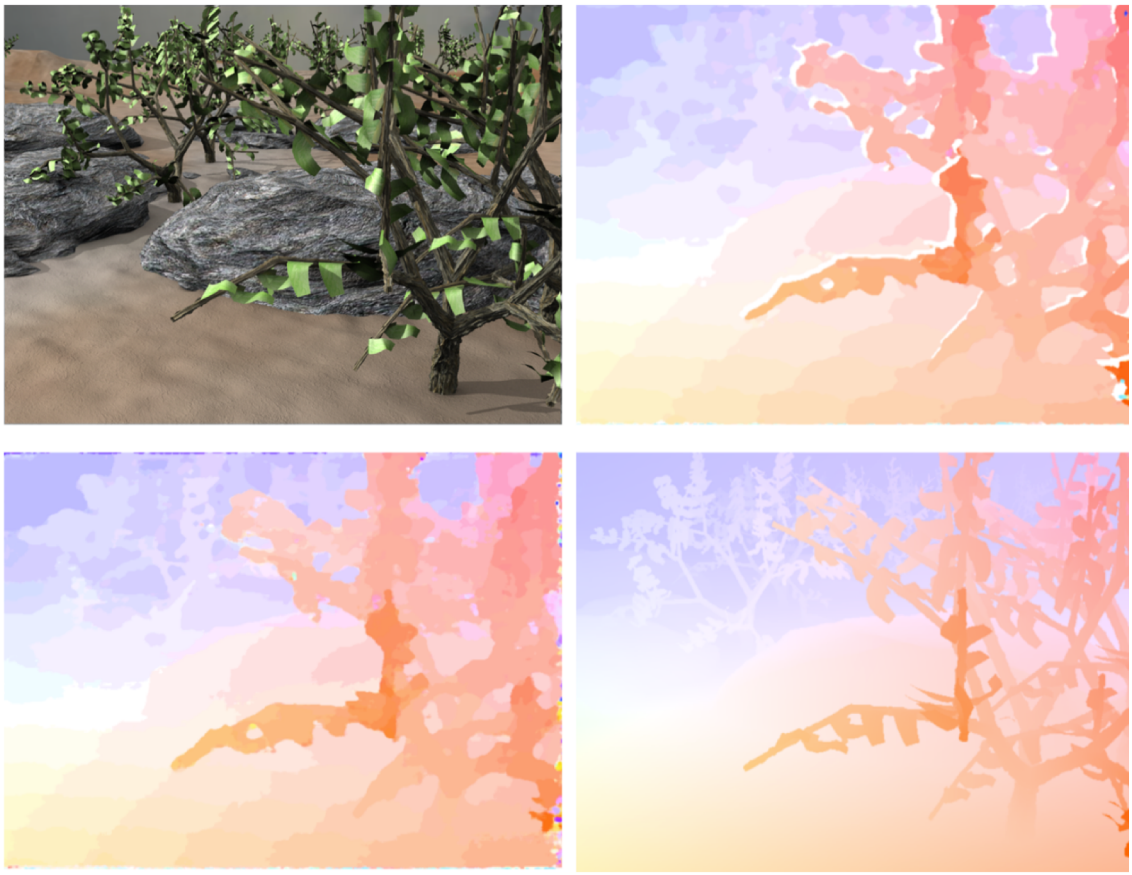

Figure 5.4 An example of inertial guidance in Grove3. Top left: Frame at time t. Top right: Inertial estimates from [t-1, t]. Bottom left: Output flow using NG-fSGM+BI. Bottom right: Groudtruth flow.

Table 5.12 shows the performance of NG-fSGM+BI, compared to NG-fSGM+B. The initial guidance consistently improves the accuracy especially for images with large flow range (Grove3, Urban2 and Urban3). With inertial guidance, l can be reduced from 16 to 2 with almost identical accuracy (see 4th and 5th columns of Table 5.12). This also helps achieve lower complexity (memory size reduced by 40%, number of operations reduced by 50%).

Table 5.13 Accuracy, Memory Size and Number of Operations of NG-fSGM+BIS

n	64				128			
f_1	**1**	**2**	**1**	**2**	**1**	**2**	**1**	**2**
f_2	**1**	**1**	**2**	**2**	**1**	**1**	**2**	**2**
Grove2	1.62	1.80	1.86	1.89	1.64	1.78	1.79	1.95
Grove3	6.82	7.46	7.22	7.93	7.07	7.63	7.51	8.15
Hydrangea	0.72	0.87	0.74	1.03	0.76	0.92	0.75	1.00
RubberWhale	0.66	0.93	0.95	1.03	0.67	0.95	0.81	1.03
Urban2	4.33	5.31	4.94	6.03	3.73	4.20	4.22	4.32
Urban3	13.67	13.97	13.30	14.15	12.72	13.57	12.42	13.42
Mean	**4.64**	**5.06**	**4.84**	**5.34**	**4.43**	**4.84**	**4.58**	**4.98**
Memory / KB	125.18	100.18	98.93	86.43	324.68	243.68	241.43	200.93
Operations / 10^6	22.57	12.35	12.35	7.23	73.13	39.93	39.93	23.32

5.3.3 Sparse-to-Dense Block-based NG-fSGM Results

We evaluate the sparse-to-dense flow estimation method on the block-based NG-fSGM with inertial guidance. The results are shown in Table 5.13. Unlike sparse-to-dense flow estimation on the basic NG-fSGM, here the block size n plays an important role. We can see without sampling, n = 64 and n = 128 provide similar accuracy. But when sampling rates f_1 and f_2 increase, the accuracy degrades more dramatically when n = 64. Thus larger n can support more aggressive sampling level (see columns 4 and 9 in Table 5.13).

For the same accuracy, a smaller block size has lower complexity per block but more blocks per frame, while a larger block size with sampling has higher complexity per block but fewer blocks per frame. However, a larger block size with sampling results in smaller total number of operations per frame and smaller total memory size. Columns 2 and 8 in Table 5.13 show that, for similar accuracy, the 128 × 128 block with sampling (f_1 = 1, f_2 = 2) reduces the total memory size and number of operations by 52% and 56%, respectively, compared to the 64 × 64 block.

CHAPTER 6. CONCLUSION

This chapter summarizes the main contributions of this thesis and suggests possible future work.

6.1 Contributions

This thesis presents novel dense optical flow methods based on Semi-Global Matching [10] [9] for applications with extreme memory and computational constraints. The main contributions of this thesis are summarized below:

- A dense optical flow method Neighbor-Guided Semi-Global Matching (NG-fSGM) is presented. The proposed method has better accuracy and significant reductions in memory size and number of operations, compared to a prior work using SGM [9]. The key features are as follows.
    1. A flow vector subset selection strategy based on exploring flow similarity of neighboring pixels is proposed. It leads to a significant reduction in the search space size and makes algorithm complexity independent of flow range.
    2. A simple cost aggregation function, which reduces number of operations, is proposed. When used in adjunction with flow subset selection, it outperforms more complicated cost aggregation functions.
    3. Pixel-wise matching cost and flow computation are embedded in cost aggregation. This saves the memory for storing pixel-wise matching cost, aggregated cost along paths and overall aggregated cost in the second scan.

4. Sparse-to-dense optical flow estimation is applied to further reduce complexity. An interpolation method is introduced to maintain accuracy while operating NG-fSGM on sampled pixels.
5. An evaluation of NG-fSGM was conducted on Middlebury dataset [2]. Impact of algorithm parameters was analyzed to help find best set of parameters. The results showed that the method using all proposed techniques outperforms fSGM [9] in accuracy with 129x reduction in computation and 12x reduction in memory size.

- Next a parallel block-based optical flow method is proposed based on NG-fSGM.

This method is desirable for multi-core architectures, achieving higher throughput, lower latency and lower power. The key features are as follows.

1. Images are divided into overlapping blocks and a study of the extent of overlap on the accuracy and hardware cost is conducted.
2. The overlap size in minimized by using inertia to help guide flow vector selections for boundary pixels.
3. An evaluation of block-based NG-fSGM was conducted on Middlebury dataset [2]. The results showed that with inertial guidance, the proposed method achieves significantly improved throughput, latency and power efficiency, while preserving the accuracy.

6.2 Future Work

The following items summarize possible research directions based on the work presented in this thesis.

- Since SGM is applicable to both stereo matching and optical flow, a unified architecture could provide solutions to both problems. Depth information could help optical flow estimation and flow estimation could help stereo matching. Thus a depth and optical flow joint estimation could result in better accuracy.
- Computing flow vectors for every pixel is wasteful for certain regions such as background and large object surface. An adaptive support region could be determined to reduce the number of total flow vectors that are computed and adjust cost aggregation penalties at boundaries of the support region. This would help reduce the complexity and improve the accuracy.
- While the proposed block-based NG-fSGM method is easy to parallelize, for GPU implementations, we found that the bottleneck is the cache size in each core. Thus GPU-specific optimizations could be developed to accelerate block-based NG-fSGM on GPUs.
- In recent years, it is shown that learned features obtained from multi-layer convolutional neural network (CNN) are very useful in computer vision tasks. Several papers have used deep learning to compute matching cost to improve accuracy. A CNN-based matching cost, trained with a sufficient dataset, could lead to a better optical flow accuracy.

REFERENCES

[1] Horn B.K.P, Robot Vision, MIT Press, 1986.

[2] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black and R. Szeliski, "A Database and Evaluation Methodology for Optical Flow," *International Journal of Computer Vision,* vol. 92, pp. 1-31, 2011.

[3] S. Baker and I. Matthews, "Lucas-Kanade 20 Years On: A Unifying Framework," *International Journal of Computer Vision,* vol. 56, no. 3, pp. 221-225, 2004.

[4] A. Bruhn, J. Weickert and C. Schnorr, "Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods," *International Journal of Computer Vision,* vol. 61, no. 3, pp. 211-231, 2005.

[5] H. Zimmer, A. Bruhn, J. Weickert, L. Valgaerts, A. Salgado, B. Rosenhahn and H.-P. Seidel, "Complementary optic flow," *Energy Minimization Methods in Computer Vision and Pattern Recognition,* pp. 207-220, 2009.

[6] V. Lempitsky, S. Roth and C. Rother, "Fusion flow: discrete-continuous optimization for optical flow estimation," *Computer Vision and Pattern Recognition,* pp. 1-8, 2008.

[7] T. Cooke, "Two applications of graph-cuts to image processing," *Proceedings of digital image computing: techniques and applications,* pp. 498-504, 2008.

[8] C. Lei and Y.-H. Yang, "Optical flow estimation on coarse-to-fine region-trees using discrete optimization," *Internatonal Conference on Computer Vision,* pp. 1562-1569, 2009.

[9] S. Hermann and R. Klette, "Hierarchical scan line dynamic programming for optical flow using semi-global matching," *Computer Vision-ACCV Workshops,* vol. 7729, pp. 556-567, 2012.

[10] J. Xiang, Z. Li, D. Blaauw, H. S. Kim and C. Chakrabarti, "Low complexity optical flow using neighbor-guided semi-global matching," *IEEE International Conference on Image Processing,* pp. 4483-4487, 2016.

[11] H. Hirschmuller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 30, no. 2, pp. 328-341, 2008.

[12] J. Xiang, Z. Li, H. S. Kim and C. Chakrabarti, "Hardware-Efficient Neighbor-Guided SGM Optical Flow for Low Power Vision Applications," *IEEE International Workshop on Signal Processing Systems,* pp. 1-6, 2016.

[13] B. Horn and B. Schunck, "Determining optical flow," *Artificial Intelligence,* vol. 17, pp. 185-203, 1981.

[14] D. Shulman and J.-Y. Herve, "Regularization of discontinuous flow fields," *Proceedings. Workshop on Visual Motion,* pp. 81-86, 1989.

[15] M. Black and P. Anandan, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," *Computer Vision and Image Understanding,* vol. 63, no. 1, pp. 75-104, 1996.

[16] D. Sun, S. Roth, J. Lewis and M. Black, "Learning optical flow," *European Conference on Computer Vision,* vol. 3, pp. 83-97, 2008.

[17] A. Wedel, T. Pock and D. Cremers, " Structure and motionadaptive regularization for high accuracy optic flow," *IEEE International Conference on Computer Vision,* pp. 1663-1668.

[18] J. Bergen, P. Anandan, K. Hanna and R. Hingorani, "Hierarchical model-based motion estimation," *European Conference on Computer Vision,* pp. 237-252, 1992.

[19] T. Brox and J. Malik, "Large displacement optical flow: Descriptor matching in variational motion estimation," *IEEE Transaction on Pattern Analysis Machine Intelligence,* vol. 33, no. 3, pp. 500-513, 2011.

[20] B. Glocker, N. Paragios, N. Komodakis, G. Tziritas and N. Navab, "Optical flow estimation with uncertainties through dynamic MRFs," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-8, 2008.

[21] R. Zabih and J. Woodfill, "Non-parametric local transforms for computing visual correspondance," *European Conference on Computer Vision,* vol. 2, pp. 151-158, 1994.

[22] P. Viola and W. M. Wells III, "Alignment by Maximization of Mutual Information," *International Journal of Computer Vision,* vol. 24, no. 2, pp. 137-154, 1997.

[23] C. Stiller, "Motion Estimation for Coding of Moving Video at 8 kbit/s with Gibbs Modeled Vectorfield Smoothing," *Visual Communications and Image Processing '90: Fifth in a Series,* pp. 468-476, 1990.

[24] G. d. Haan, P. Biezen, H. Huijgen and O. Ojo, "True-motion estimation with 3-D recursive search block matching," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 3, no. 5, pp. 368-379, 1993.

[25] C. Barnes, E. Shechtman, A. Finkelstein and D. B. Goldman, "PatchMatch: a randomized correspondence algorithm for structural image editing," *ACM Transactions on Graphics (Proc. SIGGRAPH),* vol. 28, 2009.

[26] M. Werlberger, T. Pock and H. Bischof, "Motion estimation with non-local total variation regularization," *IEEE Conference on Computer Vision and Pattern Recogintion,* pp. 2464-2471, 2010.

[27] C. Vogel, S. Roth and K. Schindler, "An Evaluation of Data Costs for Optical Flow," *German Conference on Pattern Recognition,* pp. 343-353, 2013.

[28] H. Hirschmuller and D. Scharstern, "Evaluation of stereo matching costs on images with radiometric differences," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. 31, no. 9, pp. 1582-1599, 2009.

[29] R. B. Potts, "Some generalized order-disorder transformations," *Mathematical Proceedings of the Cambridge Philosophical Society,* pp. 106-109, 1952.

[30] D. Gibson and M. Spann, "Robust optical flow estimation based on a sparse motion trajectory set," *IEEE Transactions on Image Processing,* vol. 12, no. 4, pp. 431-445, 2003.

[31] J. Wulff and M. J. Black, "Efficient sparse-to-dense optical flow estimation using a learned basis and layers," *IEEE Conference on Computer Vision and Pattern Recognition,* pp. 120-130, 2015.

[32] M. Irani, "Multi-frame optical flow estimation using subspace constraints," *The Proceedings of the Seventh IEEE International Conference on Computer Vision,* pp. 173-194, 1999.

[33] C.-M. Wang, K.-C. Fan and C.-T. Wang, "Estimating Optical Flow by Integrating Multi-Frame Information," *Journal of Information Science and Engineering,* pp. 1719-1731, 2008.

[34] R. Kennedy and C. J. Taylor, "Optical Flow with Geometric Occlusion Estimation and Fusion of Multiple Frames," *Energy Minimization Methods in Computer Vision and Pattern Recognition,* pp. 364-377, 2015.

[35] J. L. Barron, D. J. Fleet and S. S. Beauchemin, "Performance of optical flow techniques," *International Journal of Computer Vision,* vol. 12, no. 1, pp. 43-77, 1994.

[36] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," *Proceedings of the 7th international joint conference on Artificial intelligence,* vol. 2, pp. 674-679, 1981.

[37] A. Blake and A. Zisserman, Visual reconstuction, MIT Press, 1987.