LOW COMPLEXITY OPTICAL FLOW USING NEIGHBOR-GUIDED SEMI-GLOBAL MATCHING

Jiang Xiang[*], Ziyun Li[§], David Blaauw[§], Hun Seok Kim[§] and Chaitali Chakrabarti[*]

[*]School of Electrical, Computer and Energy Engineering, Arizona State University
[§] Department of Electrical Engineering and Computer Science, University of Michigan
jiang.xiang@asu.edu, liziyun@umich.edu, blaauw@umich.edu, hunseok@umich.edu, chaitali@asu.edu

***Abstract*—This paper presents Neighbor-Guided Semi-Global Matching (NG-fSGM), a new method for optical flow. It is based on SGM, a popular dynamic programming algorithm for stereo vision, where 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. Unlike conventional SGM, NG-fSGM operates on a subset of the search space that has been aggressively pruned based on neighboring pixels' information. Our proposed method achieves a fast approximation of SGM with significantly simpler cost aggregation and flow computation. Compared to a prior SGM extension for optical flow, the proposed NG-fSGM provides about 9x reduction in the number of computations and 5x reduction in the memory requirement with only 0.17% accuracy degradation when evaluated with Middlebury benchmark test cases.**

***Keywords*—*Optical flow; SGM; fSGM; Low complexity*.**

I. INTRODUCTION

Accurate optical flow is essential for many computer vision applications such as object detection/tracking, video compression, and autonomous navigation. The dense optical flow algorithm calculates a motion vector for each pixel of the image. The major challenges for optical flow estimation include occlusion, perspective distortion, transparent objects, low/uniform textures, and repeated patterns. For emerging power-critical mobile applications such as autonomous navigation of unmanned aerial vehicles, there is the added challenge of real-time performance with stringent memory and computational resource constraints.

Most existing optical flow algorithms optimize a global energy function in the form of the weighted sum of *data term* (consistency in optical flow) and *prior term* (flow field's favor such as flow smoothness), as stated in the taxonomy of Baker et al. [1]. Of the optimization strategies, the continuous optimization algorithm, such as those by Baker and Matthews [2], Bruhn et al. [3] and Zimmer et al. [4], typically require a large number of nonlinear computations. In contrast, discrete optimization algorithms, often used in stereo matching, enhance search efficiency by approximating the solution space. See, for example, Lempitsky et al. [5], Cooke [6], and Lei and Yang [7]. However, directly applying a discrete optimization to 2D optical flow is challenging because, unlike stereo, the complexity increases quadratically with flow search range. For large flow displacements, therefore, discrete optimization algorithms are typically embedded into a coarse-to-fine approach [9] that incurs inevitable accuracy degradation due to resolution loss at higher hierarchical levels.

Addressing this technical challenge, this paper introduces a new optical flow method, the Neighbor-Guided Semi-Global Matching (NG-fSGM). The proposed method is based on SGM [8], a popular concept in stereo matching, and fSGM [9], a prior work that applies SGM to optical flow. Our objective is to achieve performance comparable to fSGM with significantly higher computational efficiency. The algorithm-level techniques include: (1) reducing the search space size by exploring flow similarity of neighboring pixels; (2) approximating aggregated cost array and embedding pixel-wise cost computation and flow computation in aggregation. We show that the proposed NG-fSGM method provides robust optical flow accuracy comparable to fSGM. Furthermore, these techniques help in significant reduction both in the computational complexity and in the memory space, thus making this method suitable for implementation in a mobile platform.

II. BACKGROUND

The SGM proposed by Hirschmüller for stereo [8] achieves very high accuracy by applying dynamic programming based cost function optimization over the entire 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 that penalizes abrupt disparity changes. An extension of SGM algorithm for optical flow, fSGM, was introduced in [9]. fSGM extends the search space from 1D stereo to 2D flow. It is probably the closest approach to ours, hence we briefly review fSGM here.

Step 1: Computation of pixel-wise matching costs $C(\boldsymbol{p},\boldsymbol{o})$ 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. The cost function can be based on Rank, Census [10] and mutual information [11].

Step 2: Application of an additional constraint on matching costs to get the smoothness of flow image. This is done 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 as

$$L_r(\boldsymbol{p},\boldsymbol{o}) = C(\boldsymbol{p},\boldsymbol{o}) + Z - \min_{\boldsymbol{k}} L_r(\boldsymbol{p}\text{-}r,\boldsymbol{k}) \tag{1}$$

with the cost regularization summand

$$Z = \min_{\boldsymbol{i}}\{L_r(\boldsymbol{p}\text{-}r,\boldsymbol{o}) + P_l\|\boldsymbol{i}\text{-}\boldsymbol{o}\|_1\} \tag{2}$$

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.

$$S(\boldsymbol{p},\boldsymbol{o}) = \textstyle\sum_r L_r(\boldsymbol{p},\boldsymbol{o}) \tag{3}$$

Step 3: Flow computation. It uses winner-takes-all strategy by selecting $\boldsymbol{o}$ with the minimum cost $S(\boldsymbol{p},\boldsymbol{o})$.

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. It motivates the necessity of lower complexity alternatives. We propose a new optical flow algorithm, Neighbor-Guided fSGM (NG-fSGM) that achieves accuracy comparable to fSGM but with significantly reduced memory usage and reduced number of arithmetic/logic operations.

III. Optical Flow with Neighbor-Guided Semi-Global Matching

NG-fSGM reduces the complexity by aggressively pruning the search space based on information from neighbors. Using neighborhood information to prune the search space has also been used in [13] and [14] in the context of block matching for motion estimation. We extend the idea to semi-global cost aggregation and also modify flow computation functions to reduce the overall complexity.

A. Flow Subset Selection

The possibility that neighboring pixels in the image have an identical or slowly changing flow vector is high since they tend to belong to the same object, and thus have similar motion. Small flow variation usually occur due to slanted surface of objects, spinning objects, camera position, etc. Large flow variations can occur in the edge of objects and are typically due to occlusion and motion discontinuity. NG-fSGM exploits this property by selecting a subset of search space, $\boldsymbol{O_p}$ for each pixel $\boldsymbol{p}$, based on its neigbor pixels' flow results prior to the computation of pixel-wise matching costs. The selection strategy is inspired by PatchMatch [12], which initializes a random search space and propagates good 'guesses'.

The subset selection for each pixel $\boldsymbol{p}$ is guided by its neighboring pixels along every path in SGM, as shown in Fig. 1. Let $\boldsymbol{Q}$ denote a set of flow vectors. For pixel $\boldsymbol{p}$, the best N flow vectors $\boldsymbol{Q_{p\text{-}r}}$ of previous pixel along path r with the minimum cost $L_r(\boldsymbol{p}\text{-}r,\boldsymbol{o})$, are selected into the search subset. We choose the best N vectors (and not just one), for robustness to errors caused by accumulated cost variation along a path and localized abnormality of pixel-wise matching cost. Since the SGM applies a low aggregation penalty when the flow varies smoothly, eight adjacent flow vectors surrounding each of these best vectors are selected for cost evaluation as well. To enable the algorithm adaptation 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. To allow simple and fast initialization, the initial subset at these pixels is selected randomly from a uniform distribution.

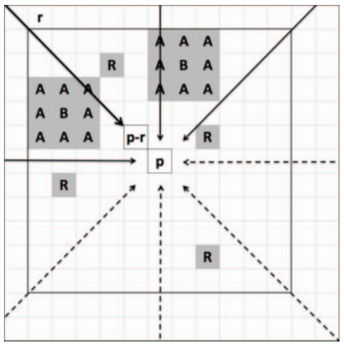

Fig. 1. Subset selection. For the center pixel $\boldsymbol{p}$, the thick square represents flow range. The solid arrows represents path directions in forward scan while dash arrows represents path directions in backward scan. The selected flow vectors, guided by neighbor $\boldsymbol{p}\text{-}r$ along path r, is the combination of B, A and R. B corresponds to the N = 2 best flow vectors and A corresponds to their adjacent flow vectors. R corresponds to M = 4 random flow vectors.

Typically, SGM approaches are implemented in two scans, forward and backward, and paths are divided into two groups as in Fig. 1. The forward scan processes every pixel from top-left to bottom-right of the image in the raster scan order, while the backward scan processes pixels in reverse order. As a result, pixel p has different flow vector subset, $\boldsymbol{O}_{\boldsymbol{p}1}$ and $\boldsymbol{O}_{\boldsymbol{p}2}$, and aggregated cost, $S_1(\boldsymbol{p},\boldsymbol{o})$ and $S_2(\boldsymbol{p},\boldsymbol{o})$, for forward scan and backward scan, respectively. The overall subset $\boldsymbol{O_p}$ should be union of $\boldsymbol{O}_{\boldsymbol{p}1}$ and $\boldsymbol{O}_{\boldsymbol{p}2}$ and the overall aggregated cost $S(\boldsymbol{p},\boldsymbol{o})$ should be sum of $S_1(\boldsymbol{p},\boldsymbol{o})$ and $S_2(\boldsymbol{p},\boldsymbol{o})$. We propose in III-C an approximation strategy to combine forward and backward aggregation. In the backward scan, N best flow vectors with the forward scan minimum cost $S_1(\boldsymbol{p},\boldsymbol{o})$ and their eight adjacent vectors (located in a 3 × 3 window) are added to construct $\boldsymbol{O}_{\boldsymbol{p}2}$ to increase algorithm accuracy. The purpose is to prevent wrong selection in single scan since flow could be inconsistent in certain directions.

The flow vectors chosen by different aggregation paths may be redundant since neighboring pixels' best vectors can be identical and 3×3 adjacent windows (i.e., B's in Fig.1) can overlap. If redundancy is ignored (worst case), the total number of vectors in the search subset is $T = N \times (P / 2 + 1) \times 9 + M$, where P is the number of aggregation paths. The complexity of the following steps in NG-fSGM is $O(WHT)$, which is independent of flow search range ($D = d^2$). In actual implementation, images with larger flow displacements are more likely to experience this worst case condition, while smaller flow images would have reduced average complexity because of redundancy in the flow vectors from neighbors.

B. Pixel-wise Matching Cost

For pixel-wise matching cost, we adopt Hamming distance of Census transform [10]. The Census transform has been proven to represent image structure well and to be robust to environment variations [15]. 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 within a pre-defined window. The cost is then computed by Hamming distance between corresponding pixels.

In a typical implementation for SGM, the costs $C(\boldsymbol{p},\boldsymbol{o})$ are precalculated and stored in an integer (7 bit) array of size $W \times d \times D$. However in the proposed NG-fSGM, since a subset of flow vectors is selected, the calculation of $C(\boldsymbol{p},\boldsymbol{o})$ is part of the cost aggregation step and performed only when $\boldsymbol{o}$ is selected. We do not store the array of precomputed pixel-wise matching cost $C(\boldsymbol{p},\boldsymbol{o})$ in the memory. Instead, the Census transform of two images is precalculated and stored in memory ($W \times d \times 30$ bytes for a 11 × 11 Census window).

C. Cost Aggregation and Flow Computation

For cost aggregation, a linear or a truncated linear function works relatively well [9]. We use single-step Potts model instead since a simple linear function over-regularizes and truncation of the linear function is expensive.

We modify the cost regularization summand Z as

$$\begin{aligned} Z = \min\{ & L_r(\boldsymbol{p}-r,\boldsymbol{o}), \\ & \min_{|\boldsymbol{i}-\boldsymbol{o}|^2 \le 2} L_r(\boldsymbol{p}-r,\boldsymbol{i}) + P_1, \\ & \min_{\boldsymbol{j}} L_r(\boldsymbol{p}-r,\boldsymbol{j}) + P_2\} \end{aligned} \tag{4}$$

where P_1 and P_2 are regularization penalties ($P_1 \le P_2$). The modification penalizes neighboring eight flow vectors by a smaller penalty (smoothness constraint) and all other vectors by a larger 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 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 $L_r(\boldsymbol{p}\text{-}r,\boldsymbol{o})$ is not available in $\boldsymbol{Q}_{p\text{-}r}$ along a certain direction r, it is assigned the minimum value in $\boldsymbol{Q}_{p\text{-}r}$ 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 $\boldsymbol{O}_p$ 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 $\boldsymbol{B}_p$ with 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 $W \times H \times N$.

In the backward scan, for each pixel $\boldsymbol{p}$, cost aggregation is directly followed by flow computation, where the total number of flow vectors is determined by 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: 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. Finally, the output flow vector $\boldsymbol{o}$ is the one corresponding to the minimum cost $S(\boldsymbol{p},\boldsymbol{o})$.

D. Post Processing

After the raw flow results are computed, post-processing steps are applied to refine the flow image. We apply a median filter on both channels (horizontal and vertical components) of the flow image to remove errors and smoothen flow fields. If the accuracy requirement is high, a consistency check between previous and current frame (similar to left-right check for stereo [8]) can be applied to get the confidence map. Optical flow of low-confidence pixels can be interpolated from surrounding high-confidence pixels. For the evaluation results shown in Section IV, we only consider a simple 3x3 median filter.

IV. Results

We conducted comprehensive experiments on the Middlebury optical flow benchmark [1] to evaluate the performance of our method. The main objective is to quantify the impact of various algorithm parameters on the accuracy and complexity. The accuracy is quantified in terms of endpoint error (radius = 2) percentage. The algorithm complexity is measured in terms of the memory size and the number of arithmetic/logic operations. A high-level summary of our findings is given in Table I.

To identify the impact of pixel-wise cost function, we evaluated different values of Census transform window size given $N = 3$, $M = 3$ and $P = 8$. The mean error percentage is 7.00%, 5.26%, 4.71% and 4.82% for Census transform window size of 7x7, 9x9, 11x11 and 13x13, respectively. Since the memory requirement, is a function of the window size, we conclude that 11 × 11 Census transform window provides a reasonable tradeoff point for balancing complexity and accuracy. For $N = 3$, $M = 3$ and 11 × 11 Census transform window size, we study the effect of number of paths P. We see that the mean error percentage is 11.17%, 5.24%, 4.71% and 4.68% while the relative execution time on a 2.6 GHz Intel Core i5 processor changes from 1x, 1.37x, 1.72x to 2.64x for P values of 2, 4, 8 and 16, respectively. Thus $P = 8$ provides a reasonable tradeoff point. For the 11 × 11 Census transform window, $P = 8$ configuration, we found that P_1 = 20, and P_2 = 60 gives the best performance and so in the rest of this study, we choose these parameter settings.

TABLE I. SENSITIVITY OF ALGORITHM PARAMETERS

Parameters	*Impact on Accuracy*	*Impact on Complexity*
N	moderate	low
M	low	low
Census Window Size	low	moderate
The Number of Paths	moderate	high

In NG-fSGM, *N* and *M* are the key parameters that control the algorithm accuracy and complexity by changing the number of selected flow vectors in the search space. Table II shows the algorithm performance and complexity metrics for different values of *N*, when *M* = 4. We also provide the percentage of selected flow vectors over the entire search space to show the impact of different values of *N*. Note that the complexity values in Table II are not the worst-case but average values based on simulations on different images. Table III shows the algorithm accuracy for different values of *M*. While there is significant improvement in accuracy compared to when *M* = 0, the relative improvement diminishes with increasing M. Unlike *N*, the value of *M* has much smaller impact on the number of operations and almost no impact on memory requirement. From the results in Tables II and III, we see that *N*=3, *M*=3 provides good accuracy with modest architectural cost.

TABLE II. PERFORMANCE AND COMPLEXITY OF NG-FSGM FOR DIFFERENT VALUES OF *N*

N	*Selected Flow Vectors*	*Endpoint Error*	*Memory Space (MB)*	*Number of Giga Operations*
1	6.15%	5.79%	2.37	2.17
2	7.83%	4.94%	3.38	2.96
3	9.27%	4.71%	4.39	4.33
4	10.40%	4.71%	5.40	6.31

TABLE III. PERFORMANCE AND COMPLEXITY OF NG-FSGM FOR DIFFERENT VALUES OF *M*

N = 1			*N = 3*		
M	*Endpoint Error*	*Number of Giga Operations*	*M*	*Endpoint Error*	*Number of Giga Operations*
0	7.30%	2.14	0	5.15%	4.25
1	5.85%	2.14	1	4.76%	4.27
2	5.77%	2.15	2	4.75%	4.29
3	5.76%	2.16	3	4.71%	4.31

Table IV shows the accuracy and complexity of NG-fSGM (*N* = 3, and *M* = 3) compared to fSGM and Lucas-Kanade [16], both with the same post-processing (Section III.D) and a single-level pyramid scheme. 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, cost regularization summand (eqn.(4)), P_1 = 40, and P_2 = 200. Penalties in fSGM are larger since NG-fSGM is more likely to get larger values from other flow vectors (3rd term in eqn. (4)). Our algorithm provides significant benefit compared to fSGM in complexity since NG-SGM only utilizes 10% flow vectors to achieve comparable algorithm accuracy. We also observe that NG-fSGM significantly outperforms Lucas-Kanade in accuracy at the cost of increased memory area requirement and a slightly higher number of computations.

To visualize the accuracy difference, Fig. 2 shows the flow maps of image 'Mequon' on Middlebury for each algorithm. NG-fSGM and fSGM both outperforms Lucas-Kanade. The raw flow map output of NG-fSGM shows blurry results 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, after post processing, this difference becomes insignificant. Table IV confirms that the overall accuracy of NG-fSGM after post processing is almost identical to the original fSGM. Since NG-fSGM achieves an order of magnitude complexity reduction, it is certainly an attractive alternative to the original fSGM for low power and resource limited applications.

TABLE IV. COMPARISON OF NG-FSGM, FSGM AND LUCAS KANADE

Algorithm	*Endpoint Error*	*Memory Space (MB)*	*Number of Giga Operations*
fSGM	4.54%	20.68	37.53
Lucas-Kanade	15.78%	1.47	3.15
NG-fSGM	4.71%	4.39	4.31

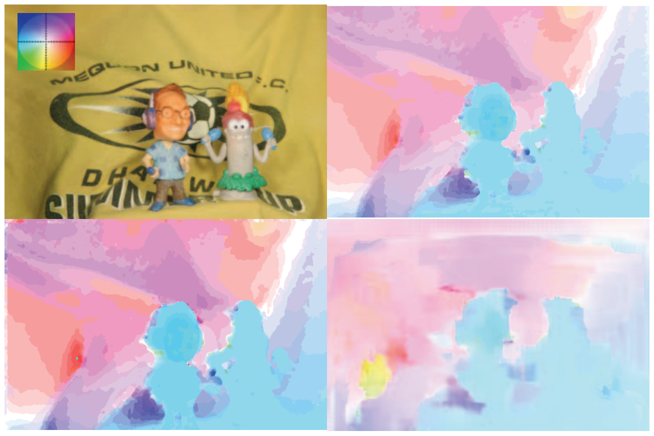

Fig. 2. Colored flow maps of 'Mequon' using different algorithms. Top-left: input previous frame and color legend; top-right: NG-fSGM, *N* = 3; bottom-left: fSGM; bottom-right: Lucas-Kanade.

V. CONCLUSIONS

This paper presented NG-fSGM, a low complexity method for optical flow. The complexity reduction is achieved by aggressively pruning the flow vector search space using the information from neighbor pixels. The cost aggregation and flow computation steps have been optimized for further complexity reduction. NG-fSGM has been evaluated on the Middlebury optical flow dataset. The evaluation results show that ND-fSGM has comparable performance with an order of magnitude reduction in complexity compared to a prior work fSGM, and greatly outperforms other similar-costly methods like Lucas-Kanade. These advantages make NG-fSGM an attractive algorithm for real-time and mobile applications.

References

[1] 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.

[2] S. Baker and I. Matthews, “Lucas-Kanade 20 Years On: A Unifying Framework,” International Journal of Computer Vision, Vol. 56(3), pp. 221-225, 2004.

[3] A. Bruhn, J. Weickert and C. Schnörr, “Lucas/Kanade meets Horn/Schunck: combining local and global optic flow methods,” International Journal of Computer Vision, Vol. 61(3), pp. 211–231, 2005.

[4] 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.

[5] 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.

[6] T. Cooke, “Two applications of graph-cuts to image processing,” Digital Image Computing: Techniques and Applications, pp. 498–504, 2008.

[7] C. Lei, Y.-H. Yang, “Optical flow estimation on coarse-to-fine region-trees using discrete optimization,” International Conference on Computer Vision, pp. 1562–1569, 2009.

[8] H. Hirschmueller, “Stereo processing by semiglobal matching and mutual information,” Pattern Analysis and Machine Intelligence, Vol. 30, pp. 328–41, 2008.

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

[10] R. Zabih and J. Woodfill, “Non-parametric local transforms for computing visual correspondance,” European Conference on Computer Vision, pp. 151–158, 1994.

[11] P. Viola and W. M. Wells, “Alignment by maximization of mutual information,” International Journal of Computer Vision, Vol. 24(2), pp. 137–154, 1997.

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

[13] Christoph Stiller, “Motion—Estimation for Coding of Moving Video at 8 kbit/s with Gibbs Modeled Vectorfield Smoothing” Proc. SPIE 1360, Visual Communications and Image Processing '90: Fifth in a Series, 468 (September 1, 1990).

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

[15] H. Hirschmüller and D. Scharstein, “Evaluation of stereo matching costs on images with radiometric differences,” IEEE Trans. Pattern Analysis Machine Intelligence, Vol. 31, pp. 1582–1599, 2009.

[16] B.D. Lucas and T. Kanade, “An iterative image registration techinique with an application to stereo vision,” International Joint Conference on Artificial Intelligence, Vol. 81, pp. 674-679, 1981.