Realtime Phase-based Optical Flow on the GPU

Karl Pauwels and Marc M. Van Hulle
Laboratorium voor Neuro- en Psychofysiologie,
K.U.Leuven, Belgium
{karl.pauwels, marc.vanhulle}@med.kuleuven.be

Abstract

Phase-based optical flow algorithms are characterized by high precision and robustness, but also by high computational requirements. Using the CUDA platform, we have implemented a phase-based algorithm that maps exceptionally well on the GPU's architecture. This optical flow algorithm revolves around a reliability measure that evaluates the consistency of phase information over time. By exploiting efficient filtering operations, the high internal bandwidth of the GPU, and the texture units, we obtain dense and reliable optical flow estimates in realtime at high resolutions (640×512 pixels and beyond). Even though the algorithm is local and does not involve iterative regularization, highly accurate results are obtained on synthetic and complex real-world sequences.

1. Introduction

The recovery of visual motion or optical flow from a sequence of images is crucial for autonomous navigation through complex environments. Dense optical flow fields contain a large amount of information that can be used to extract self-motion, to describe the three dimensional structure of the environment, to detect interesting events such as independently moving objects, to recognize actions, etc.

A wide variety of optical flow algorithms have been introduced in the past. An important distinction can be made between local and global methods. The first class only uses the intensity information in a small region surrounding the pixel [10], whereas the second class enforces additional global constraints (e.g. smoothness of the optical flow field) [9]. Local methods are very simple and straightforward to implement, but typically the results are not very precise. Global methods on the other hand deliver highly accurate estimates but require a lot of parameter tuning and iterative optimization. Both classes usually employ the assumption that brightness remains constant over the sequence.

Filter-based approaches, in particular those that employ phase information, provide a middle ground, in that they deliver precise and reliable estimates, that more closely resemble those obtained with global methods, but without the complex parameter tuning and iterative optimization required by the latter.

Phase-based techniques towards the computation of optical flow were introduced by Fleet and Jepson [5] and have been shown to be more accurate than other local methods [2]. This is mainly due to the fact that phase information is robust to changes in contrast, scale, orientation, and speed [5]. The main drawback of phase-based techniques, and the reason why current realtime implementations rely on dedicated hardware [4] (these implementations usually focus on binocular disparity estimation), is the high computational load associated with the filtering operations.

We demonstrate here that, using modern graphics hardware, it is possible to estimate reliable optical flow fields with these techniques at very high resolutions and frame rates. We use a modified version of Fleet and Jepson's algorithm which contains a reliability measure that evaluates the consistency of phase information over time [7]. In addition, this algorithm has been extended with a coarse-to-fine control strategy, as in [13].

We first discuss the algorithm and its implementation in more detail, and then demonstrate its performance on synthetic and real-world sequences.

2. Phase-based Optical Flow

In this section we discuss the efficient filterbank used and we provide details on the multiscale phase-based optical flow algorithm [7, 13].

2.1. Filterbank

For a specific orientation θ, the spatial phase at pixel location $\mathbf{x} = (x, y)^{\mathrm{T}}$ can be extracted using 2D complex Gabor filters:

$$G(\mathbf{x}, \mathbf{f}_\theta) = e^{-|\mathbf{x}|^2/\sigma^2} e^{i\,\mathbf{x}\cdot(2\pi\mathbf{f}_\theta)}\,, \tag{1}$$

with peak frequency $\mathbf{f}_\theta = (f_{x,\theta}\,,\, f_{y,\theta})^{\mathrm{T}}$. A highly efficient spatial-domain implementation of a Gabor filterbank

978-1-4244-2340-8/08/$25.00 ©2008 IEEE

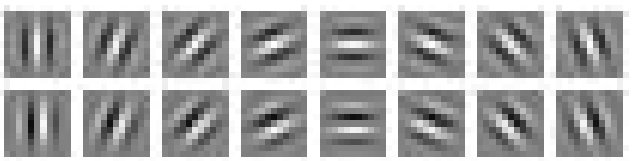

Figure 1. Even (top row) and odd (bottom row) filters used to extract spatial phase at eight orientations. The entire filterbank response can be obtained by appropriately combining the responses of 24 1D convolutions with 11-tap filters.

has been proposed in [11]. The filters introduced there are tuned to a very high frequency ($|\mathbf{f}_\theta| = 1/4$ pixels^{-1}), and separability [8] and symmetry considerations were exploited to obtain responses at four orientations using only 12 1D convolutions with 11-tap filters. Since the optical flow algorithm presented here relies on an orientation voting mechanism, responses at a larger number of orientations are preferable. For this reason, we compute phase at eight orientations. Following a similar procedure as in [11], the required responses can be obtained on the basis of only 24 1D convolutions (again using 11-tap filters). The effective filterbank is depicted in Fig. 1.

2.2. Phase-based Optical Flow using Spatial Filtering

The filter response, obtained by convolving the image, $I(\mathbf{x})$, with the oriented filter from Eq. 1 can be written as:

$$\begin{aligned} R(\mathbf{x}) &= (I * G)(\mathbf{x}) && (2)\\ &= \rho(\mathbf{x}) e^{i\,\phi(\mathbf{x})} && (3)\\ &= C(\mathbf{x}) + i\,S(\mathbf{x})\,. && (4) \end{aligned}$$

Here

$$\rho(\mathbf{x}) = \sqrt{C(\mathbf{x})^2 + S(\mathbf{x})^2}\,, \tag{5}$$

and

$$\phi(\mathbf{x}) = \arctan[S(\mathbf{x})/C(\mathbf{x})]\,, \tag{6}$$

are the amplitude and phase components, and $C(\mathbf{x})$ and $S(\mathbf{x})$ are the responses of the quadrature filter pair. The $*$ operator depicts convolution. Phase-based techniques center around the assumption that constant phase surfaces evolve according to the motion field [5]. Accordingly, points on an equi-phase contour satisfy $\phi(\mathbf{x}, t) = c$, with c a constant. Similar to the brightness constancy assumption, this phase constancy assumption can be transformed into a phase gradient constraint:

$$\nabla\phi \cdot \mathbf{v} + \psi = 0\,, \tag{7}$$

where $\nabla\phi = (\delta\phi/\delta x\,,\, \delta\phi/\delta y)^{\mathrm{T}}$ is the spatial phase gradient, $\mathbf{v} = (v_x, v_y)$ is the optical flow, and ψ the temporal phase gradient, $\delta\phi/\delta t$. Due to the aperture problem, only the velocity component along the spatial phase gradient can be computed (normal flow):

$$(\nabla\phi \cdot \mathbf{v})\,\frac{\nabla\phi}{|\nabla\phi|} = -\psi\,\frac{\nabla\phi}{|\nabla\phi|}\,. \tag{8}$$

This is identical to:

$$|\nabla\phi|\,\mathbf{c} = -\psi\,\frac{\nabla\phi}{|\nabla\phi|}\,, \tag{9}$$

where $\mathbf{c}$ is the velocity component along the spatial phase gradient. We then obtain:

$$\mathbf{c} = \frac{-\psi}{|\nabla\phi|}\,\frac{\nabla\phi}{|\nabla\phi|}\,. \tag{10}$$

Under a linear phase model, the spatial phase gradient can be substituted by the radial frequency vector, $2\pi\mathbf{f}_\theta$ [6]. In this way, the component velocity, $\mathbf{c}_\theta(\mathbf{x})$, at pixel $\mathbf{x}$ and for filter orientation θ, can be estimated directly from the temporal phase gradient, $\psi_\theta(\mathbf{x})$:

$$\mathbf{c}_\theta(\mathbf{x}) = -\frac{\psi_\theta(\mathbf{x})}{2\pi|\mathbf{f}_\theta|}\,\frac{\mathbf{f}_\theta}{|\mathbf{f}_\theta|}\,. \tag{11}$$

At each location and for each orientation, the temporal phase gradient is estimated on the basis of the evolution of the spatial phase in time. It is obtained by solving the following linear model in the least-squares sense:

$$\phi_\theta(\mathbf{x}, t) = a + \psi_\theta(\mathbf{x})t\,. \tag{12}$$

The intercept, a, is discarded. To cope with the periodicity of the phase, the phase is first unwrapped sequentially from $\phi_\theta(\mathbf{x}, 2)$ up to $\phi_\theta(\mathbf{x}, n)$, where n is the number of frames in the short sequence used ($n > 2$). We use a simple unwrapping technique that only considers the (unwrapped) phase at the previous time instance. The reliability of each component velocity is measured by the mean squared error (MSE) of the linear fit, $\sum_t \Big(\Delta\phi_\theta(\mathbf{x}, t)\Big)^2/n$:

$$\Delta\phi_\theta(\mathbf{x}, t) = \Big(a + \psi_\theta(\mathbf{x})t\Big) - \phi_\theta(\mathbf{x}, t)\,. \tag{13}$$

Note that this reliability measure examines the consistent evolution of phase information in *time*, and is thus more closely related to the concept of motion than most other measures that examine the variability of intensity in *space*. A component velocity is considered reliable if its MSE is below a certain threshold, τ_l (the phase linearity threshold). Each reliable component velocity provides a constraint on the full velocity:

$$|\mathbf{c}_\theta(\mathbf{x})| = \mathbf{v}(\mathbf{x})^{\mathrm{T}}\,\frac{\mathbf{c}_\theta(\mathbf{x})}{|\mathbf{c}_\theta(\mathbf{x})|} = \mathbf{v}(\mathbf{x})^{\mathrm{T}}\,\frac{\mathbf{f}_\theta}{|\mathbf{f}_\theta|}\,. \tag{14}$$

If several reliable component velocities with different orientations are present, the corresponding constraints can be combined to estimate the full velocity. Provided a minimal number of component velocities at pixel $\mathbf{x}$ are reliable, they are integrated into a full velocity by means of an intersection-of-constraints procedure:

$$\mathbf{v}^*(\mathbf{x}) = \underset{\mathbf{v}(\mathbf{x})}{\arg\min} \sum_{\theta \in O(\mathbf{x})} \left(|\mathbf{c}_\theta(\mathbf{x})| - \mathbf{v}(\mathbf{x})^{\mathrm{T}} \frac{\mathbf{c}_\theta(\mathbf{x})}{|\mathbf{c}_\theta(\mathbf{x})|} \right)^2 , \tag{15}$$

where $O(\mathbf{x})$ is the set of orientations for which reliable component velocities have been estimated. Contrary to most other approaches, estimating the full velocity from component velocities requires information from the current pixel only in this algorithm. This is possible since spatial information has already been collected in the filtering stage.

The entire procedure discussed in this section requires three parameters: the number of frames in the short sequence, the minimal number of reliable component velocities, and the phase linearity threshold. In the remainder of the paper, the first two parameters are fixed to five and four respectively. Only the linearity threshold is adjusted according to the noise level of the sequence.

2.3. Multiscale Optical Flow

Due to phase periodicity, phase-based techniques can only detect shifts up to half the filter wavelength. To extend this range, a coarse-to-fine control strategy can be used [6]. An efficient solution involves the use of a Gaussian pyramid [1], in which each level is separated by an octave scale. Starting from the original image resolution at pyramid level $k = 1$, the next level, $k + 1$, is obtained by blurring the images with a Gaussian kernel, $g(\mathbf{x})$, and subsampling:

$$I^{k+1}(\mathbf{x}) = (\mathcal{S}(g * I^k))(\mathbf{x}) . \tag{16}$$

The subsampling operator, $\mathcal{S}$, reduces the image resolution to half the resolution of the previous level. The original filters (Eq. 1) are now applied to each level of the pyramid:

$$R^k(\mathbf{x}) = (I^k * G)(\mathbf{x}) . \tag{17}$$

By applying the original filters to the lower resolution images, the largest detectable shift effectively doubles at each pyramid level.

The control strategy starts at the top of the pyramid, level k. Using the optical flow estimate obtained at that resolution, $\mathbf{v}^k$, the phase estimate at the next higher resolution, ϕ^{k-1}, is warped in such a way that the estimated motion is removed [3]:

$$p^{k-1}(\mathbf{x}, t) = \phi^{k-1}\Big(\mathbf{x} - 2 \cdot \mathbf{v}^k(\mathbf{x}) \cdot (t_c - t)\,,\, t\Big) , \tag{18}$$

where $t_c = (n + 1)/2$ is the index of the center frame (assuming an uneven number of frames). Since the optical flow estimate has been computed at the lower resolution, it needs to be doubled first. The factor $(t_c - t)$ ensures that each pixel in the sequence $(t = 1, 2, \ldots, n)$ is warped to its corresponding location in the center frame $(t = t_c)$. Bilinear interpolation is used to perform these warps with subpixel accuracy. Next, the warped phase, p^{k-1}, is used to compute the residual motion. Since a large component of the motion has now been removed, this residual motion is more likely to be within the range of the filters applied to that level. The new optical flow estimate, $\mathbf{v}^{k-1}$, is then obtained by adding the residual motion to $2 \cdot \mathbf{v}^k$. This process is repeated until the pyramid level corresponding to the original image resolution is reached.

The optical flow algorithm presented here is particularly suitable for this warping strategy since it uses strictly local information. Only optical flow vectors that can be computed reliably (obtained on the basis of a sufficient number of reliable component velocities) at the highest resolution are retained. In other words, if the refinement made at the highest resolution to a lower resolution estimate (that was reliable at that lower resolution) is unreliable, the flow vector is discarded. In this way, overly smooth flow fields are avoided.

3. GPU Implementation

The algorithm explained in the previous section is ideally suited for a GPU implementation for a number of reasons. First, it relies extensively on convolution operations. If we compare convolution times using the CUDA Software Development Kit (CUDA SDK) implementations for GPU and CPU, we obtain a 40-fold speed-up with a Geforce 8800 GTX compared to a 2.4 GHz Core 2 Quad processor (using a single core) when filtering a 640×512 image with an 11 taps filter (the speed-up factor goes up to about 120 at higher resolutions). Second, the Gabor image representation is much larger than the image itself (our filterbank returns for each scale 16 values per pixel). Although the richness of this representation enables a variety of applications, it also requires a large bandwidth. Third, the coarse-to-fine control strategy involves many warping operations, which can be handled by the GPU's texture units. We next discuss the implementation of the algorithm in more detail.

3.1. Gabor Pyramid

All 2D separable convolutions required to construct the image and Gabor pyramids are performed using the highly efficient implementation available in the CUDA SDK. A number of simple kernels are used to subsample the image for the image pyramid construction, and to combine the 1D convolution responses in the different ways required for the Gabor filter response construction. The filtering sequence and filter combination operations have been organized so as

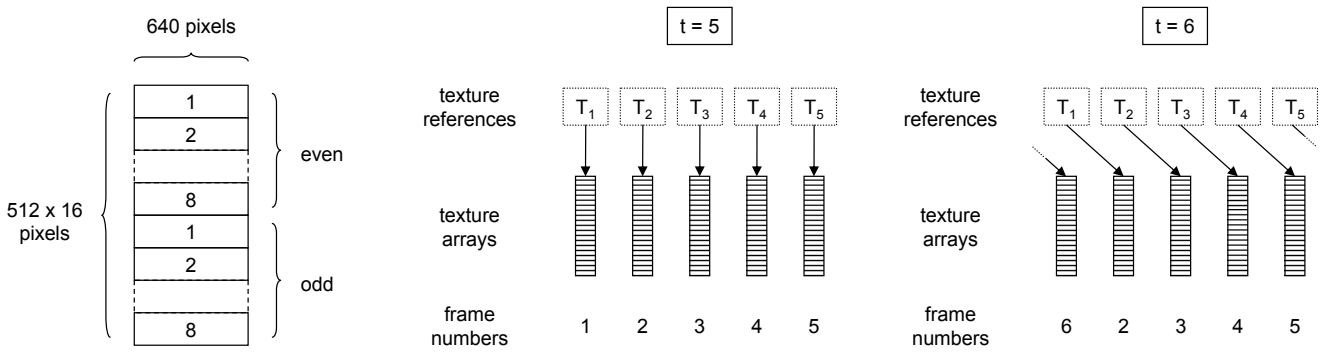

Figure 2. **(A)** Layout of the texture array containing the even and odd filter responses for 8 orientations at resolution 640×512 pixels. **(B)** Circular buffer state at t=5. The five texture arrays contain the filter responses of the first five frames and the texture references point to the consecutive arrays. **(C)** At the next time instance, t=6, the texture array containing the oldest frame's filter responses is overwritten with the new frame's filter responses. The texture references are remapped to accommodate this: T_1 now points to array 2, T_2 to array 3, ..., T_5 to array 1.

to minimize memory transfers. All Gabor filter responses for a particular frame, at a particular scale, are tiled in a single texture array. The layout of this texture array is illustrated in Fig. 2(A) for an example resolution of 640×512.

3.2. Optical Flow

After the filtering stage, the phase is computed, warped and unwrapped, and the component velocities (and their reliability) are computed and integrated into the full velocity. All these steps are performed using a single kernel that operates on a single pixel. During the different steps of component velocity computation, the five subsequent phase values are stored in shared memory. The phase warping step relies heavily on the GPU's texture units. In our implementation, we do not warp the phase, but rather the Gabor filter responses (and recompute phase afterwards). In this way, bilinear interpolation can be used to achieve subpixel accuracy, resulting in much higher precision estimates. This warping transformation depends on the previous scale optical flow estimates and is thus not known in advance. This could result in slow memory access. However, since the spatial locality is very high in this operation, the GPU's texture cache ensures a high bandwidth in this crucial data-intensive stage of the algorithm.

Since a total of five frames are used on all occasions to estimate the temporal phase gradients, the optical flow kernel operates on five texture arrays (at each scale). As illustrated in Figs. 2(B,C), by dynamically changing the texture references to these arrays, we implement a circular buffer. In this way, only data corresponding to the current frame needs to be updated at all times, and memory transfers are minimized.

image resolution	320×256	640×512
Gabor pyramid	5.4	11.7
optical flow	2.4	8.9
total **(msec)**	7.8	20.6
frames per second	**127.6**	**48.5**

Table 1. GPU processing times (in msec) and frame rates obtained on a Geforce 8800 GTX. *Gabor pyramid* involves copying one image from CPU to GPU memory, constructing the image pyramid for this image, applying the filterbank from Fig. 1 to this image pyramid, and copying the Gabor pyramid from linear device memory to the texture array. *Optical flow* consists of the computation of phase, component velocity and full velocity (at all scales), and the subsequent copy of the final optical flow field back to CPU memory. Three scales were used at 320×256 and four scales at 640×512.

4. Results

We first report on the processing times and then discuss results obtained when applying the algorithm to synthetic and real-world sequences. In the remainder, the number of frames used to estimate the temporal phase gradient is always equal to five.

4.1. Processing Times

Table 1 contains the processing times for the different steps discussed in the previous section for two resolutions: 320×256 and 640×512. In addition to these, the pyramid always contains the resolutions 160×128 and 80×64. In both cases, we have observed a frame rate well above 40 frames per second. The computation times and frame rates

image resolution	320×256	640×512
Gabor pyramid	0.1	0.5
optical flow	1.1	4.8
total **(sec)**	1.2	5.3
frames per second	**0.8**	**0.2**

Table 2. CPU processing times (in sec) and frame rates obtained on a 2.4 GHz Core 2 Quad processor (using a single core only) using the same dataset as in Table 1.

	without sky		with sky	
τ_l	AAE	density	AAE	density
0.02	2.09°	63 %	3.22°	54 %
0.05	2.35°	82 %	3.99°	76 %
0.10	2.67°	91 %	4.67°	88 %

Table 3. Optical flow error and density obtained on frame 9 of the Yosemite sequence (with and without the sky region) for different settings of the reliability threshold τ_l. A sequence of five frames has been used on all instances. AAE = average angular error.

reported in Table 1 have been measured by processing a complex real-world sequence (100 frames in length) from CPU memory, always including the CPU to GPU and GPU to CPU memory transfers, and averaging the results (see Fig. 5 for an example frame of the sequence used).

Filtering is currently performed one scale at a time. This however does not fully saturate the GPU at the resolutions used here, resulting in the comparatively slow Gabor pyramid construction at 320×256. This could be improved by filtering the entire pyramid at once. The optical flow computation scales much better to lower resolutions.

We have performed the identical timing experiment using a reference CPU implementation. The processing times (in seconds this time) and frame rates are reported in Table 2. Particularly time-consuming on the CPU are the filter response interpolation and the component velocity computation, which have to be performed for each orientation. We observe over a 150-fold increase in performance with the GPU implementation at both resolutions.

4.2. Yosemite Sequence

We have first applied the GPU implementation of the optical flow algorithm to the well-known Yosemite sequence. The top row of Fig. 3 contains (from left to right) frame 9 of this sequence, the ground truth optical flow associated with this frame, and the color coding used to visualize the optical flow (hue and intensity indicate respectively the direction and magnitude of the optical flow vector). The bottom row of Fig. 3 shows optical flow fields obtained from the five frame sequence consisting of frames 7, 8, 9, 10 and 11, for different values of the reliability measure τ_l (unreliable estimates are shown in white). As expected, the density increases when the phase linearity measure is increased. The estimated flow field closely resembles the ground truth in all instances.

Table 3 provides a quantitative evaluation of the results. The difference between the estimated and ground truth optical flow field is measured by the average angular error (AAE) [2]. Small errors are observed in all instances. We can see from this table that, both when including and excluding the sky region in the evaluation, the AAE (and density) decreases when the phase linearity threshold is decreased. This confirms that the reliability measure correctly rejects unreliable estimates.

When using five frames to estimate the temporal phase gradient, the algorithm yields good performance on a variety of sequences. We have observed that when this number is reduced to three, the MSE is no longer a good measure of reliability and the results significantly deteriorate.

4.3. Real-world Sequences

We next demonstrate the performance of the algorithm on real-world sequences. Ground-truth optical flow fields are unavailable for these sequences, and therefore only a qualitative evaluation is possible. Since real-world sequences contain a lot more noise than the synthetic Yosemite sequence, the reliability threshold has been set to 0.5 in the remainder. Note that although the sequences differ greatly in terms of noise-level, range of motions, camera motion, interlacing, etc., the same threshold has been used for all sequences. This again demonstrates the simplicity of the algorithm.

Figure 4 shows the center images of the sequences used (left) and the estimated optical flow fields (right) for the publicly available *Ettlinger Tor* (top row) and *Rheinhafen* (bottom row) sequences. These sequences have resolutions of 512×512 and 688×565 pixels respectively. Again, the color coding from Fig. 3 has been used, which means that black pixels correspond to zero flow and white pixels to unreliable estimates. Even though strong interlacing artifacts are present in both sequences, all moving objects, and their direction of motion, are clearly visible. The motion boundaries are also reasonably sharp. These results are very similar to results obtained with state-of-the-art algorithms and realtime implementations [12, 16].

We next applied the algorithm to a more difficult sequence recorded from inside a moving car. These images have been recorded at 640×512 pixels. The scene layout in this sequence is much more complex. Besides the self-motion of the car, a number of additional independently moving objects are present as well. Figure 5(A) contains the

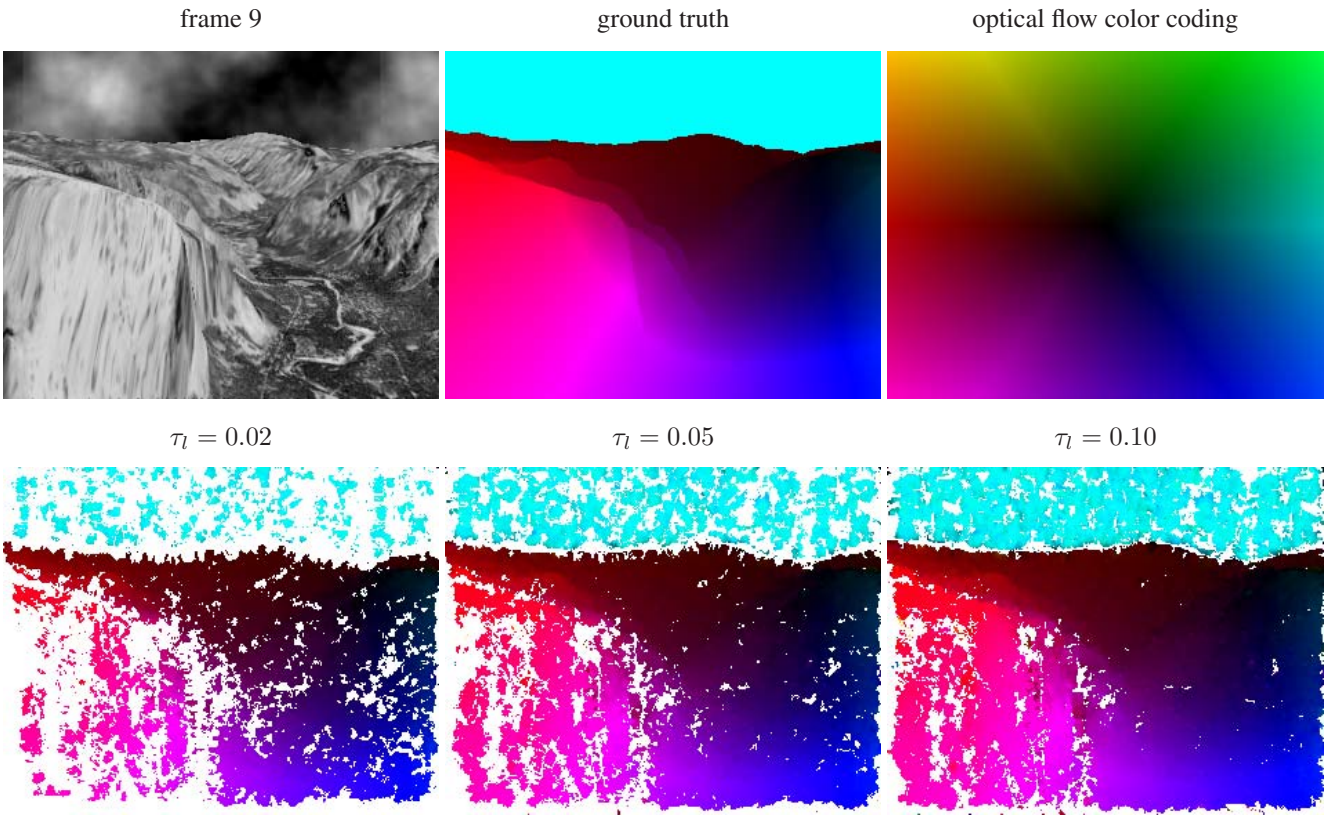

Figure 3. The top row contains the center frame and associated ground truth optical flow field for the Yosemite sequence, and the color coding used to visualize the optical flow fields (the image center corresponds to zero flow). The bottom row shows flow fields, estimated from a five frame sequence, for different values of the phase linearity threshold. Unreliable estimates are shown in white.

center frame of the five frame sequence used, and Fig. 5(B) the same frame with the (subsampled) estimated optical flow field superimposed. The unreliable estimates at the bottom of the frame result from the reflection of the car's ventilation outlets, which remains static over the sequence. Real-world optical flow fields are highly complex, and in this example the camera rotation appears to dominate the optical flow field. However, if we look in more detail at the horizontal and vertical components of the optical flow fields (Figs. 5C and 5D), it is clear that the effect of the translational camera motion component (looming) is also visible. The scene structure (traffic signs, slanted road and buildings) and moving objects are clearly discernible from these figures, and are again separated by reasonably sharp boundaries.

5. Conclusion

We have demonstrated a high-performance implementation of a phase-based optical flow algorithm. The algorithm is characterized by simplicity, robustness, and speed.

Future extensions will concentrate on the incorporation of stabilization in the algorithm, which becomes crucial when moving to higher resolutions [13], and on alternate uses of the readily available filter responses. The high performance of our implementation leaves ample resources available for the simultaneous use of the rich Gabor image representation for other purposes, such as disparity or orientation estimation [14], object recognition [15], etc.

Acknowledgments

Karl Pauwels and Marc M. Van Hulle are supported by the Excellence Financing program of the K.U.Leuven (EF 2005), the Belgian Fund for Scientific Research – Flanders (G.0248.03, G.0234.04), the Flemish Regional Ministry of Education (Belgium) (GOA 2000/11), the Belgian Science Policy (IUAP P5/04), and the European Commission (NEST-2003-012963, STREP-2002-016276, IST-2004-027017, and IST-2007-217077).

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA Engineer*, 29(6):33–41, 1984.

[2] J. Barron, D. Fleet, and S. Beauchemin. Performance of optical flow techniques. *IJCV*, 12(1):43–77, 1994.

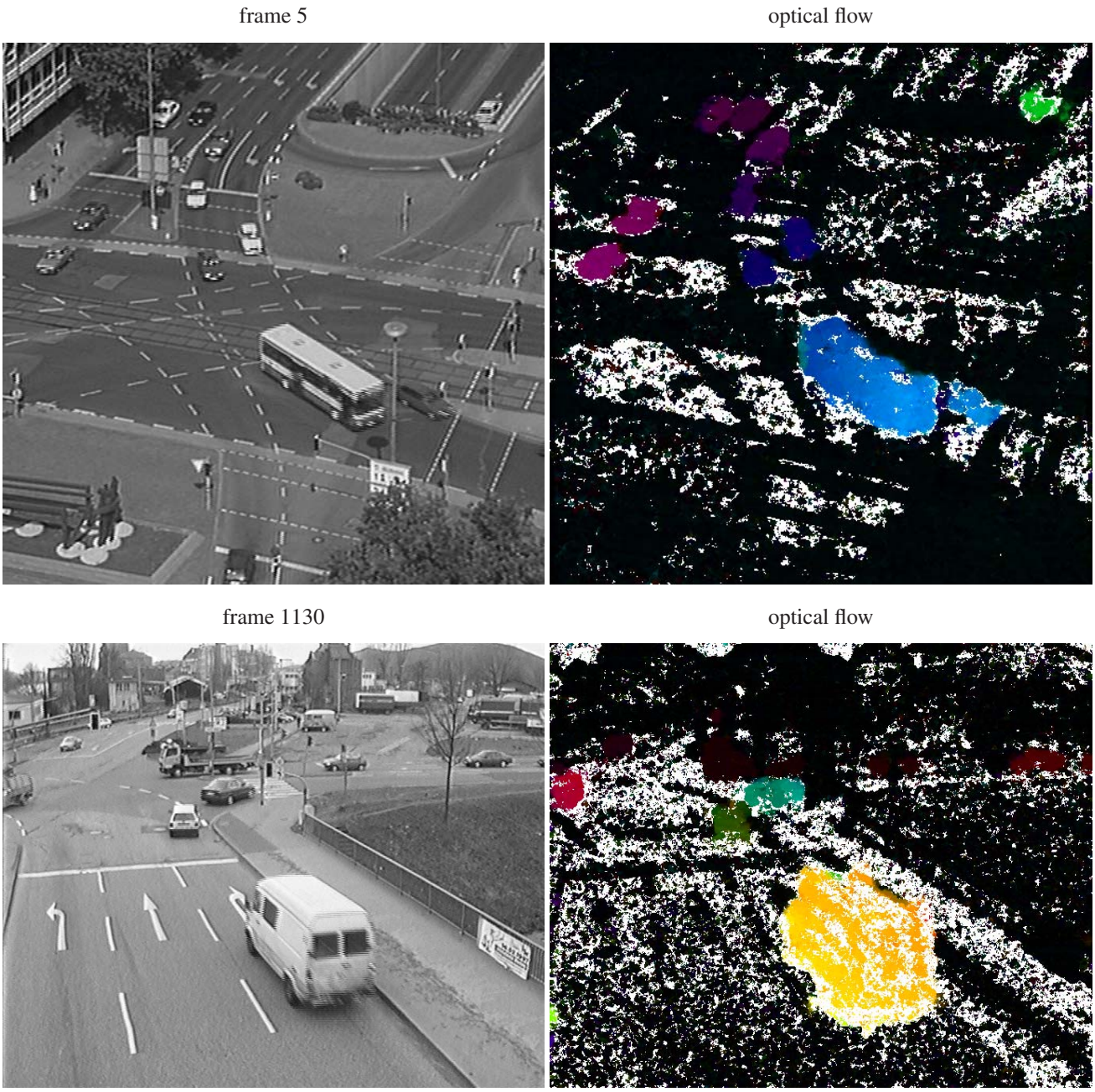

Figure 4. Center images (left column) and estimated optical flow fields (right column) for the *Ettlinger Tor* (top row) and *Rheinhafen* (bottom row) sequences. Unreliable estimates are shown in white.

[3] J. Bergen, P. Anandan, K. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *ECCV*, pages 237–252, 1992.

[4] J. Diaz, E. Ros, R. Carrillo, and A. Prieto. Real-time system for high-image resolution disparity estimation. *IEEE Transactions on Image Processing*, 16(1):280–285, 2007.

[5] D. J. Fleet and A. D. Jepson. Computation of component image velocity from local phase information. *IJCV*, 5:77–104, 1990.

[6] D. J. Fleet, A. D. Jepson, and M. R. M. Jenkin. Phase-based disparity measurement. *CVGIP-Image Understanding*, 53(2):198–210, 1991.

[7] T. Gautama and M. Van Hulle. A phase-based approach to the estimation of the optical flow field using spatial filtering. *IEEE Transactions on Neural Networks*, 13(5):1127–1136, 2002.

[8] D. Heeger. Model for the extraction of image flow. *Journal of the Optical Society of America A*, 4(8):1455–1471, 1987.

[9] B. K. P. Horn and B. G. Schunck. Determining optical-flow. *Artifical Intelligence*, 17:185–203, 1981.

(A) center frame

(B) optical flow

(C) v_x

(D) v_y

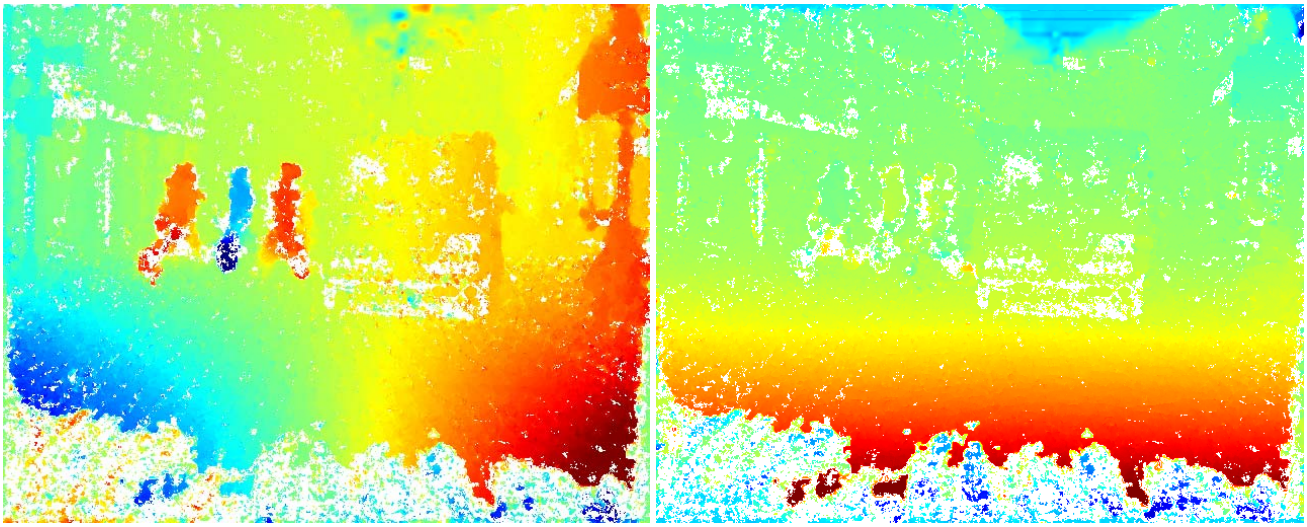

Figure 5. **(A)** Center frame, **(B)** estimated optical flow field (subsampled 10 times), and **(C)** horizontal and **(D)** vertical components of the estimated optical flow fields obtained on a complex real-world sequence involving self-motion and independent motion. The flow components are color-coded from blue (small) to red (large). As before, unreliable estimates are shown in white.

[10] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. DARPA Image Understanding Workshop*, pages 121–130, 1981.

[11] O. Nestares, R. Navarro, J. Portilla, and A. Tabernero. Efficient spatial-domain implementation of a multiscale image representation based on Gabor functions. *Journal of Electronic Imaging*, 7(1):166–173, 1998.

[12] N. Papenberg, A. Bruhn, T. Brox, S. Didas, and J. Weickert. Highly accurate optic flow computation with theoretically justified warping. *IJCV*, 67(2):141–158, 2006.

[13] K. Pauwels and M. Van Hulle. Optic flow from unstable sequences through local velocity constancy maximization. *Image and Vision Computing*, 2008, in press.

[14] S. Sabatini, G. Gastaldi, F. Solari, K. Pauwels, M. Van Hulle, J. Diaz, E. Ros, N. Pugeault, and N. Krueger. Compact (and accurate) early vision processing in the harmonic space. In *VISAPP*, pages 213–220, 2007.

[15] T. Serre, L. Wolf, S. Bileschi, M. Riesenhuber, and T. Poggio. Robust object recognition with cortex-like mechanisms. *PAMI*, 29(3), 2007.

[16] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime TV-L1 optical flow. In *DAGM-Symposium*, pages 214–223, 2007.