Real-Time Estimation of Optical Flow Based on Optimized Haar Wavelet Features

Jan Salmen[1], Lukas Caup[1], and Christian Igel[2]

[1] Institut für Neuroinformatik, Ruhr-Universität Bochum, 44780 Bochum, Germany
{Jan.Salmen, Lukas.Caup}@ini.rub.de
[2] Department of Computer Science, University of Copenhagen, 2100 Copenhagen Ø, Denmark
igel@diku.dk

Abstract. Estimation of optical flow is required in many computer vision applications. These applications often have to deal with strict time constraints. Therefore, flow algorithms with both high accuracy and computational efficiency are desirable. Accordingly, designing such a flow algorithm involves multi-objective optimization. In this work, we build on a popular algorithm developed for real-time applications. It is originally based on the Census transform and benefits from this encoding for table-based matching and tracking of interest points. We propose to use the more universal Haar wavelet features instead of the Census transform within the same framework. The resulting approach is more flexible, in particular it allows for sub-pixel accuracy. For comparison with the original method and another baseline algorithm, we considered both popular benchmark datasets as well as a long synthetic video sequence. We employed evolutionary multi-objective optimization to tune the algorithms. This allows to compare the different approaches in a systematic and unbiased way. Our results show that the overall performance of our method is significantly higher compared to the reference implementation.

1 Introduction

Optical flow can be defined as the "the distribution of apparent velocities of movement of brightness patterns in an image" [1], which result from the projection of a 3D scene onto an image plane. Estimating optical flow is common to many computer vision applications since the resulting flow field is very valuable, amongst others, for detection and tracking of objects, deriving structure from motion, estimation of ego-motion, and collision avoidance. These problems often arise in the context of real-world applications (e.g., robotics, surveillance, automotive) where hard time constraints have to be met. Therefore, there is a special interest in flow algorithms that offer good accuracy while being computationally efficient.

In this study, we propose to employ evolutionary multi-objective optimization for tuning flow algorithms with respect to these two partially conflicting objectives. This allows analysis of different possible trade-off solutions and therefore systematic comparisons.

We consider a *feature-based* algorithm [2], which was designed for driver assistance systems. This approach combines computational efficiency, high accuracy, robustness, and ease of implementation. The Census transform [3] is used for the description of small image patches. This allows for table-based matching and tracking of interest points over time, resulting in a sparse flow vector field. In this work, we show how to modify the algorithm in order to make it more flexible and powerful. We suggest to replace the Census transform by Haar wavelet features, which are state-of-the-art for real-time computer vision.

This article is organized as follows. In the next section, we give an overview over related work. In section 3, the reference algorithm is introduced. Section 4 presents our approach. Multi-objective optimization is introduced in section 5. Our experiments are described in section 6, finally the results are discussed.

2 Related Work

For a detailed overview and comparison of many different flow algorithms, we refer to Barron et al. [4] and Baker et al. [5] and references therein. Here, we focus only on approaches closely related to our method, namely real-time capable and feature-based flow algorithms.

Lucas and Kanade [6] proposed an iterative method that was adapted by Bouguet [7] for pyramid based feature tracking. His implementation is publicly available in the *OpenCV* library[3]. We consider this algorithm as a baseline for our experiments. Tomasi and Kanade [8] proposed tracking of features obtained based on eigenvalues of small image patches. Optical flow algorithms based on this principle are fast and accurate [9]. Similar results can also be obtained with correlation-based approaches [10], or block-matching algorithms [11]. Stein proposed the algorithm that serves as starting point for our method [2]. It uses the Census transform to generate signatures for image patches that can be used like *hash values* to handle interesting points. This allows for very efficient temporal analysis. This approach is described in more detail in the next section.

The local, feature-based approaches mentioned above calculate sparse flow vector fields. While these algorithms are able to handle large displacements, they may produce noisy results depending on local structure. It is essential to attach great importance to temporal integration and evaluation of pixel neighborhoods.

Dense optical flow can be obtained based on the variational method proposed by Horn and Schunck [1]. Implementations of this basic technique are likely to fail for large displacements and for small moving objects because of the smoothness constraints. Moreover, such implementations typically are not well suited for real-time applications. However, different improvements and extensions have been proposed recently which address these issues [12–15].

3 Original Method

The Census transform as proposed by Zabih and Woodfill [3] is based on the 3×3 neighborhood $\mathcal{P}$ of a pixel. We denote the center pixel by P_0 and the eight ones surrounding it by $P_1, \ldots, P_8$. The signature $\xi(\mathcal{P})$ is calculated *bitwise*, where the i-th digit

[3] `http://sourceforge.net/projects/opencvlibrary`

ξ_i is set to 0 or 1 depending on the intensities of P_i:

$$\xi_i(P_0, P_i) = \begin{cases} 0, & P_i > P_0 \\ 1, & P_i \leq P_0 \end{cases} . \tag{1}$$

This is extended in [2] in order to distinguish "similar" pixels by introducing a new parameter ϵ and computing the signature as

$$\xi_i(P_0, P_i) = \begin{cases} 0, & P_0 - P_i > \epsilon \\ 1, & |P_0 - P_i| \leq \epsilon \\ 2, & P_i - P_0 > \epsilon \end{cases} . \tag{2}$$

Additionally, Stein proposed to consider larger neighborhoods and therefore not only pixels directly adjacent to the center. For his experiments, he used signatures with up to 20 digits.

This signature is computed for every pixel in a camera image. The positions of all image patches with signature s in frame t are stored in a table $T_t(s)$. Flow hypotheses between frames $t-1$ and t are computed by considering all pairs of entries from $T_{t-1}(s)$ and $T_t(s)$ for all relevant signatures s. In this step, two restrictions are considered: The lengths of flow vectors can be limited and the relative brightness difference between the center pixels is restricted to avoid bad flow hypothesis. All resulting hypotheses in frame t are stored in a list H_t.

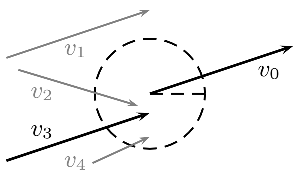

Fig. 1. Temporal analysis in frame t. $v_1, \ldots v_4$ from H_{t-1} are possible predecessors for v_0 from H_t. The vector v_1 is not valid because it is outside the search radius, v_2 because of the angular difference to v_0, and v_4 because of the relative length difference. v_3 is the only valid predecessor.

To increase speed and robustness of the approach, not all possible signatures have to be processed. Stein proposed to use a so-called black-list to exclude signatures that code for infeasible image patches (e.g., homogeneous image regions or regions related to the aperture problem). Furthermore, lists table entries $T_t(s)$ with more than mdp (*maximum discriminative power*) elements are ignored for the calculation of flow hypothesis because the number of possible hypothesis increases exponentially.

The hypotheses calculated between two frames are generally not unique and therefore not reliable. In order to compute final flow vectors for frame t, longer-term temporal analysis is performed based on H_{t-1} and H_t. For each hypotheses h in frame t, it is checked if there was another hypothesis from frame $t-1$ with similar length and similar orientation ending approximately where the new vector starts. Figure 1 illustrates this analysis. Finally, reliable flow vectors are obtained from current hypotheses which have more than p_{min} predecessors.

4 Proposed Algorithm

In this section, we present our new flow algorithm. It is based on the one described in the previous section but is more flexible. In particular, we show that it can optionally allow for sub-pixel accuracy.

4.1 Haar features for index-based matching

Haar wavelet features are state-of-the-art for real-time computer vision. Their popularity is mainly based upon the efficient computation using the *integral image* proposed by Viola and Jones [16]. Haar wavelet features were successfully applied in many computer vision applications, especially for object detection, classification, and tracking [16–19]. Figure 2 shows examples of six basic types of Haar Wavelet features. Their responses can be calculated with 6 to 9 look-ups in the integral image, independently of their absolute sizes.

It stands to reason to consider Haar wavelet features also for the calculation of optical flow. They could offer a corporate preprocessing for even more different applications and therefore serve as a universal basis for real-time computer vision systems.

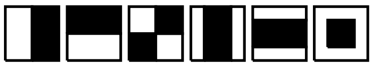

Fig. 2. Basic types of Haar wavelet features.

In the original flow algorithm, a feature-based index is obtained for every pixel through the Census transform. We calculate a similar value based on responses R of a bunch of different Haar wavelet features centered on that pixel. The signature for a pixel is calculated as

$$\xi_i(R) = \begin{cases} 0, & R_i < -\epsilon_i \\ 1, & |R_i| \leq \epsilon_i \\ 2, & R_i > \epsilon_i \end{cases} , \tag{3}$$

where ϵ_i now are individual thresholds used for discretization of the continuous feature responses. We further extend this by allowing for more than one ϵ per feature. For the example of two thresholds, $\epsilon_i^{(0)}$ and $\epsilon_i^{(1)}$, the discretization is

$$\xi_i(R) = \begin{cases} 0, & |R_i| \leq \epsilon_i^{(0)} \\ 1, & \epsilon_i^{(1)} > R_i > \epsilon_i^{(0)} \\ 2, & R_i > \epsilon_i^{(1)} \\ 3, & -\epsilon_i^{(1)} < R_i < -\epsilon_i^{(0)} \\ 4, & R_i < -\epsilon_i^{(1)} \end{cases} . \tag{4}$$

The temporal analysis (i.e., interest point matching and tracking) can be done analogously to the original approach. The Census-based method is embedded in our approach as a special case, because the definition of Haar-like wavelets allows for regions with size of a single pixel. That is, the calculations for the Census transform can be modeled with Haar wavelet features. Of course, in this case, calculation of the integral image would not lead to any gain in speed.

4.2 Optional sub-pixel refinement

Using Haar wavelet features allows to perform flow computation with sub-pixel accuracy by employing an optional refinement step: The position of the end points of all flow hypothesis in frame t are refined in a way that the feature responses are most similar to the responses at the start point in frame $t-1$.

We propose to perform *bilinear interpolation* in the radius of one pixel around the original (pixel-accurate) end point in order to find the best matching position. The resolution between pixels can be chosen arbitrarily. For our experiments, we used steps of one-tenth pixel. To calculate matching costs c for two feature vectors R and S with n components each we used the *sum of absolute differences* $c\left(R,S\right) = |R_1 - S_1| + |R_2 - S_2| + \ldots + |R_n - S_n|$.

The proposed sub-pixel refinement requires that the raw feature responses are either stored for start points of all correspondence hypothesis or that they are calculated anew only for final flow hypothesis. It depends on the problem at hand which of these two solutions is preferable in terms of efficiency. Here, we recalculate feature responses when needed for refinement.

5 Multi-objective optimization

In our experiments, we evaluate the performances of various approaches on different datasets. For a fair and meaningful comparison, all algorithms must be setup "as good as possible". Adapting parameters in such cases is typically done manually in a more or less systematic way. We propose to use rely on evolutionary optimization instead. It allows to do a more extensive search in an impartial manner.

We give a short overview over evolutionary multi-objective optimization with an emphasis on sorting candidate solutions in section 5.1. The variation of candidate solutions is described in section 5.2. Of course, results obtained by automatic optimization have to be watched as critically as results obtained by manual tuning. We discuss these issues in section 6.3.

5.1 Evolutionary multi-objective optimization

Optimizing parameters of sparse optical flow algorithms is not straightforward as their performance cannot be quantified by a single measure: maximization of the number of flow vectors and maximization of their mean accuracy are two *conflicting goals*. Therefore, we suggest to perform *multi-objective optimization* (MOO, *vector optimization*) [20]. The goal of vector optimization is to find a diverse set of solutions that approximates the set of *Pareto-optimal* trade-offs. A solution is Pareto-optimal if it is *not dominated* by another solution. In our application a solution is not dominated if on the one hand no other solution exists with more flow vectors and at least as good accuracy and on the other hand each solution with better accuracy does not have more flow vectors.

We consider evolutionary MOO. The crucial step in this process is selection. Here, we adopt the selection procedure from the MO-CMA-ES [21] for comparing and sorting candidate solutions to determine the parents for the next generation.

For the problem at hand, the vector valued quality of a solution g is given by $\boldsymbol{\Phi}(g) = (n, 1/e)$, where n is the mean number of flow vectors and e is the mean endpoint error as proposed by [5]. Both objectives are to be maximized. A solution g (weakly) dominates another solution g' iff it is better in one of the two objectives and is not worse in the second.

Based on this concept, each candidate solution g_i in a set $M = \{g_0, \dots, g_n\}$ can be assigned a rank $r(g_i)$ that expresses its level of non-dominance. The set M is partitioned into subsets $M_1, M_2, \dots, M_{n_{\text{levels}}}$. Candidate solutions in M_1 are not dominated by any other solution in M, solutions in M_2 are only dominated by members of M_1, solutions in M_3 are only dominated by solutions in $M_1 \cup M_2$, and so on. The rank $r(g)$ of a candidate solution g is defined as the index of the set containing it.

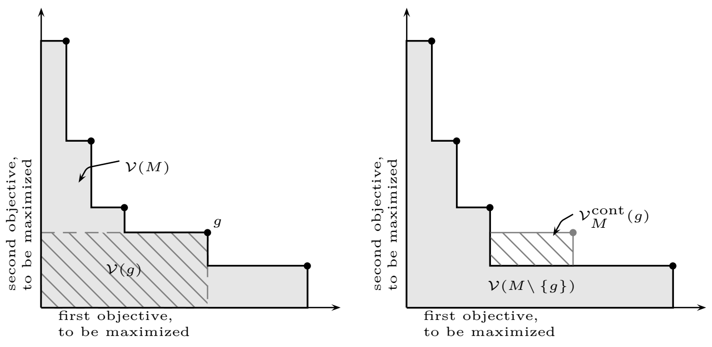
$\mathcal{V}(M)$
g
$\mathcal{V}(g)$
second objective, to be maximized
first objective, to be maximized
$\mathcal{V}_M^{\text{cont}}(g)$
$\mathcal{V}(M \setminus \{g\})$
second objective, to be maximized
first objective, to be maximized

Fig. 3. The concepts of hypervolume and contributing hypervolume. Left: Dominated hypervolume of solution g and dominated hypervolume of set M. Right: Contributing hypervolume of $g \in M$ and dominated hypervolume of set $M \setminus \{g\}$.

A second criterion is needed in order to sort solutions having the same rank index r. We do this based on their *contributing hypervolume* as proposed in [21–23]. The dominated hypervolume $\mathcal{V}(g)$ of a candidate solution g is defined as the volume of all points in the objective space (here $\mathbb{R}^2_{\geq 0}$) that are dominated by $\boldsymbol{\Phi}(g)$. The dominated hypervolume of a set M_i is defined as the volume of the union of the dominated hypervolumes of all points in M_i. The contributing hypervolume $\mathcal{V}^{\text{cont}}_{M_i}(g)$ for a candidate solution $g \in M_i$ is given by the share of the total dominated hypervolume that is lost if the solution is removed: $\mathcal{V}^{\text{cont}}_{M_i}(g) = |\mathcal{V}(M_i) - \mathcal{V}(M_i \setminus \{g\})|$. Figure 3 illustrates this concept.

A set M^* only containing non-dominated solutions is called a *Pareto set*, the corresponding vectors $\{\boldsymbol{\Phi}(g) \mid g \in M^*\}$ a *Pareto front*. After optimization, M^* represents different possible trade-offs. This does not only give insights into the characteristics of the problem at hand, but can be useful in practice, for instance, to adapt the behavior of an application online.

5.2 Mutating solutions

In our optimization problem, we allow the number of features, their individual parameters, and the flow algorithm parameters to vary. This calls for a variable length representation. We use an encoding and variation operators resembling those proposed in [18]. In the following, we briefly outline the tailored mutation procedure using the example of a Haar feature based algorithm.

Each time when mutating an individual, tossing a coin decides whether the parameters of the flow algorithm are varied or the feature set. The flow algorithm parameters comprise mdp, d_{int}, $l_{\max}$, r, d_{ang}, d_{l}, and $p_{\min}$ (see Table 1). They are mutated using standard operators.

If the feature set is mutated, it is decided whether the number of features is changed or the individual features are varied. The former happens with a probability of $p_{\text{size}} = 0.20$. Features are then added or removed with equal probability. New features are created with a random type (see Figure 2) and width and height each between 2 and 12.

Alternatively, with a probability of $1 - p_{\text{size}} = 0.80$, we mutate the features. Each Haar feature has real-valued, integer, and nominal parameters. The mutation operators *changeSize*, *changeType*, and *mutateEpsilons* are applied to each Haar feature with a probability of $p_{\text{mut}} = 0.50$ each. In *changeSize*, the width and height of the feature are changed independently by $+1$ or -1. In *changeType*, a new random basic type is assigned to the feature. In *mutateEpsilons*, the thresholds $\epsilon_i^{(0)}, \epsilon_i^{(1)}, \ldots$ are mutated. This also includes mutating the number of thresholds used. We allowed up to three of these thresholds per feature.

A repair mechanism ensures that only feasible solutions are generated: For each parameter, we defined a minimal and maximal value according to their meaning. For example, it is not reasonable to have negative values for differences (e.g., d_{int} and r) or Haar features smaller than 2×2 pixels.

6 Experiments

We compare the original Census based and our Haar wavelet based approach in two different scenarios. As a baseline for comparison, we also evaluate the performance of the pyramid based Lucas and Kanade algorithm [7] using *good features to track* [9] from *OpenCV*. The setup of our experiments is described in section 6.1, the results in section 6.2. We discuss these results as well as crucial points of our experiments in section 6.3.

Fig. 4. Frames out of the Middlebury datasets considered for experiments.

6.1 Setup

The first comparison is performed on Middlebury benchmark datasets[4]. The Middlebury benchmark [5] was designed to evaluate only dense flow algorithms. However, there are some additional datasets available that we can use for evaluation: *Rubber-Whale*, *Hydrangea*, *Grove2*, *Grove3*, *Urban2*, and *Urban3*. All these sets consist of eight frames and evaluation is performed only for the last frame in each case. We use a common parameter set for all sequences. Figure 4 shows three example frames out of the datasets.

The second comparison is performed on an *Enpeda* benchmark sequence[5]. Long synthetic video sequences have recently been introduced by Vaudrey et al. [24]. We consider the second sequence which has complex textures and hard shadows. Optimization is performed on frames 80 to 180. The subsequent frames are used after optimization to test how the optimized solutions generalize. Figure 5 shows three example frames out of the sequence.

Fig. 5. Frames out of the synthetic Enpeda sequence considered for experiments.

For the Middlebury as well as the Enpeda benchmark, we perform MOO for all three approaches. For the *OpenCV* algorithm, there are 8 parameters to be optimized. For the Census based as well as the Haar feature based approach, we optimize the features used and the algorithm parameters. As the temporal analysis is the same in both methods, many parameters that are optimized are identical, see Table 1. For the Census based approach, we used a setup with 12 digits and therefore additionally optimized individual

[4] `http://vision.middlebury.edu/flow`

[5] `http://www.mi.auckland.ac.nz/index.php?option=com_content&view=article&id=52`

thresholds for each digit. For the new approach based on Haar features, we additionally optimized the thresholds $\epsilon_i^{(0)}, \epsilon_i^{(1)}, \ldots$, types, and sizes of features as described in section 5.2.

Table 1. Parameters of the flow algorithm used.

Description		unit
Maximum discriminative power	mdp	
Maximum intensity difference	d_{int}	[%]
Maximum vector length	l_{max}	[pixel]
Radius for predecessors	r	[pixel]
Maximum angular difference	d_{ang}	[°]
Maximum length difference	d_{l}	[%]
Minimal number of predecessors	p_{min}	

For the Census based and the Haar feature based approach, we do not use a blacklist as proposed in the original work. The setup of such a list would make statistical analysis necessary, which would be performed after optimization and only improve the results. As this is the same for both methods, no bias is introduced in our comparison.

As framework for the evolutionary multi-objective optimization, we use the *Shark* open-source machine learning library [25], which is freely available[6]. It provides implementations of efficient algorithms for computing the (contributing) hypervolume used for selection.

We performed 5 independent optimization runs for each experiment. In every optimization run, we considered 1000 generations. We used elitist $(\mu+\lambda)$-selection. In each generation, $\lambda = 15$ new candidate solutions result from recombination and mutation of $\mu = 15$ parent solutions. We made sure that all parameters remain within a reasonable domain. Solutions generated during optimization which are not feasible regarding our application, namely solutions having less than 500 flow vectors or mean endpoint error larger 5 pixels, are discarded. We selected the best solutions (according to the definitions from Sec. 5.1) from all runs in order to obtain final sets for comparison. For further details, we refer to the source code and its documentation which is made publicly available.

6.2 Results

The resulting sets of optimized solutions for the Middlebury datasets are shown in Figure 6. They visualize different trade-offs between Φ_1, the mean number of flow vectors, and $1/\Phi_2$, the mean endpoint error.

Using the original Census based approach, it was possible to find up to $22,286$ flow vectors per frame (with mean endpoint error of 4.95). Using the new Haar feature based approach, it was possible to find up to $46,999$ flow vectors per frame (with mean endpoint error of 2.89). Sub-pixel refinement allowed to further increase the accuracy

[6] `http://shark-project.sourceforge.net`

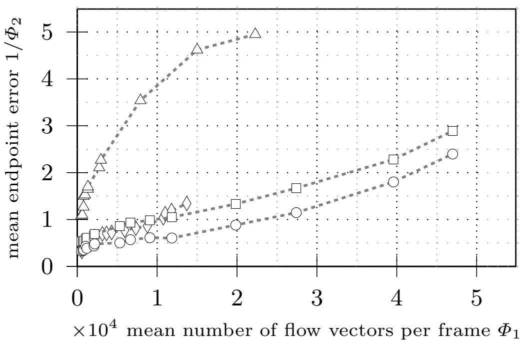

Fig. 6. Results of optimization for Middlebury datasets: Pareto fronts for Census based approach (triangles), Haar feature based approach with pixel (squares) and sub-pixel (circles) accuracy, and *OpenCV* algorithm (diamonds).

of our approach. The *OpenCV* approaches allowed to find up to $13,669$ flow vectors per frame (with mean endpoint error of 1.46).

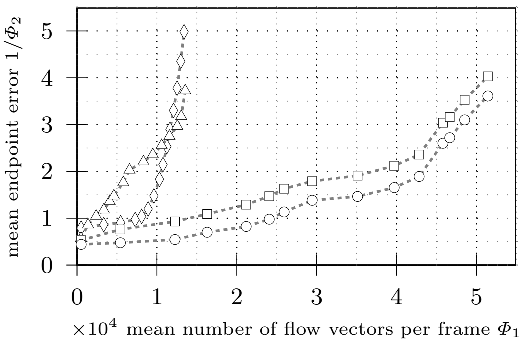

Fig. 7. Results of optimization for Enpeda sequence: Pareto fronts for Census based approach (triangles), Haar feature based approach with pixel (squares) and sub-pixel (circles) accuracy, and *OpenCV* algorithm (diamonds).

Figure 7 shows the Pareto fronts after optimization for the Enpeda sequence. The *OpenCV* algorithm allowed to find up to $12,415$ flow vectors. For the Census- based approach the maximum number of flow vectors found was $13,545$ and for the Haar feature based approach $51,434$. Sub-pixel refinement improved the mean endpoint error in our approach by 0.4 pixels on average. Optimized Haar wavelet features for two different solutions are shown in Figure 8. Table 2 shows the optimized parameter settings for different trade-offs for both approaches.

The generalization performances of the solutions on the rest of the sequence are shown in Figure 9. The Census based approach and the *OpenCV* algorithm performed

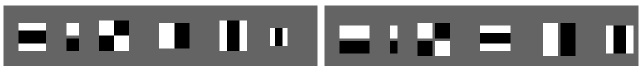

Fig. 8. Optimized feature sets from two different solutions. The features have sizes of (a) 11×11, 5×11, 12×12, 12×10, 11×12, and 7×7. (b) 12×11, 3×11, 13×13, 12×10, 13×13, and 11×11.

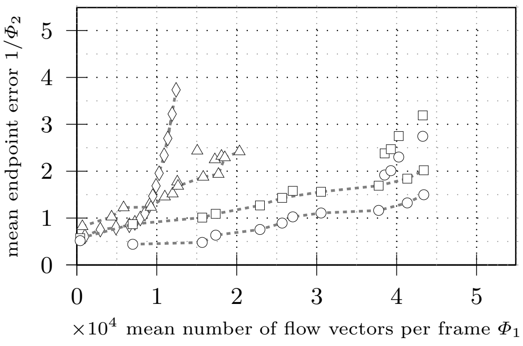

Fig. 9. Generalization results of optimized solutions on the rest of the Enpeda sequence: Census based approach (triangles), Haar feature based approach with pixel (squares) and sub-pixel (circles) accuracy, and *OpenCV* algorithm (diamonds).

significantly better in the test than in the training. Therefore one can conclude that this latter part of the sequence is “easier”. The Haar feature based approach performed slightly better for solutions up to 4×10^4 flow vectors and slightly worse for solutions with more flow vectors.

Table 2. Optimized parameters from different trade-off solutions for the Enpeda sequence.

Φ_1	$1/\Phi_2$	mdp	d_{int}	$l_{\max}$	r	d_{ang}	d_{l}	$p_{\min}$
Census based								
502	0.57	33	1.2	43	0.8	0.32	0.11	2
6569	2.03	70	0.5	40	5	0.05	0.1	1
13545	3.72	70	1	45	5	0.23	0.1	1
Haar feature based								
521	0.53	33	0.6	80	0.6	0.23	0.1	3
12264	0.93	70	1.6	40	1.6	0.05	0.15	3
25940	1.63	70	2	40	5	0.05	0.1	2
39709	2.12	70	2	40	3.3	0.05	0.1	1
51435	4.03	69	2	46	4.6	0.15	0.74	1

The optimized solutions for the Census based approach result in runtimes between $200\ ms$ and $400\ ms$ on a standard desktop PC with $2.2\ GHz$. The solutions of the Haar wavelet based approach result in runtimes between $250\ ms$ and $600\ ms$. It is not surprising that these runtimes are comparable as the implementation of temporal analysis is identical for both approaches. The optimized solutions for the *OpenCV* algorithm resulted in runtimes between $200\ ms$ and $1100\ ms$.

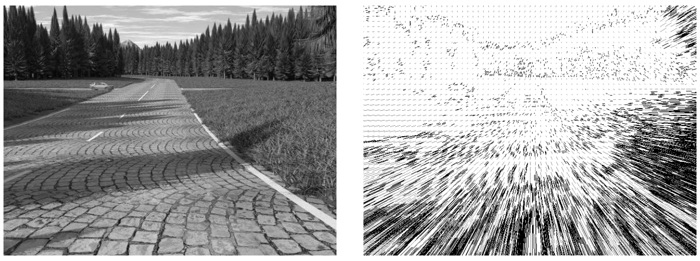

Fig. 10. Camera image from the Enpeda sequence (left) and resulting flow field (right), final flow vectors shown in black.

A typical result of the new Haar feature based flow algorithm is shown in Figure 10, where an optimized solution was considered that combines very small mean endpoint errors and an adequate number of flow vectors. This setup is suitable for real-world applications in the context of driver assistance systems, for example, for object segmentation and tracking, collision avoidance, or ego-motion estimation.

6.3 Discussion

In our experiments, the three algorithms were optimized in the same way for the same scenarios. The Census based and the new Haar feature based approach share most of their parameters because the temporal analysis is the same. Therefore the significant differences in performance shown in our experiments solely result from the different feature basis. The Haar feature based approach is more powerful in general: The mean endpoint errors were the smallest for every number of flow vectors in all experiments, especially when the optional sub-pixel refinement was performed. Our method additionally allowed to have a significantly higher number of flow vectors compared to the other approaches, in particular almost the fivefold amount in the Enpeda benchmark.

One could argue that the Haar feature based approach has the most degrees of freedom (as feature types and sizes can be adapted) and therefore profits most from optimization. While this might be true, it is exactly this flexibility that makes the approach particularly attractive. Despite this flexibility, most of the optimized solutions showed good generalization, there was no *over-fitting*.

The execution speed was not an objective in the optimization performed here, therefore the solutions found lead to comparatively slow execution speed, still most of them meet real-time constraints. All optimized solutions for the new method use 6 Haar features which was the maximal allowed number in the optimization. Nevertheless, the algorithm can generally be used in real-time applications requiring higher frame rates, especially as the table-based matching and the temporal analysis are suited very well for implementation on embedded hardware.

Evaluation of different optimized solutions also allows for statements regarding on how the parameters for the proposed flow algorithm should be set. Most of the optimized features are relatively large, $85\,\%$ in the final solutions have widths and heights larger than 10 pixels. This shows that considering relatively large neighborhoods is preferable in the given framework. Using Haar features, this is possible without any drawbacks regarding computational cost. This is an advantage over other filters. Most parameters are correlated to the trade-off between high number of flow vectors and high accuracy in some degree. Nevertheless, dependencies are manifold and the interplay is complex. Therefore, manual adjustment (e.g., for a fixed desired trade-off) is challenging and evolutionary optimization has shown to be a valuable tool.

Appropriate parameters are problem class specific. They clearly have to be different for an automotive application (with many relatively large displacements) and for entrance hall surveillance. Therefore the parameters found here are well suited for the scenarios considered but have to be changed for other contexts. This can easily be done for the proposed method.

7 Conclusion

Many computer vision applications benefit from fast and accurate estimation of optical flow. We considered a popular feature-based method which relies on the Census transform and table-based matching. We proposed to employ Haar wavelet features within this framework in order to make it more flexible and powerful.

Haar wavelet features are state-of-the art for real-time computer vision, particularly for object detection and classification. It is highly desirable to use them in more tasks within the same application to benefit from a shared preprocessing. We showed how Haar wavelet features can be applied for computation of optical flow after appropriate discretization. The resulting approach is versatile and allows for solutions that satisfy different trade-off requirements.

The performance of the new method was systematically assessed and compared to the original approach considering two different benchmarks. This was possible by evolutionary multi-objective optimization, which provides the opportunity to handle the two conflicting goals *high number of flow vectors* and *high accuracy*. In our experiments, the new approach significantly outperformed the original one. For any bound on the maximum mean error, the method can find a number of flow vectors several times larger compared to the original approach.

Our experiments suggest how to set up the new flow algorithm in general. In particular, considering relatively large neighborhoods has shown to be most successful. This favors Haar wavelets features, because the computational cost of calculating them is independent of their sizes.

References

1. Horn, B.K.P., Schunck, B.G.: Determining optical flow. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA (1980)
2. Stein, F.: Efficient computation of optical flow using the census transform. In: Proceedings of the DAGM Symposium, Springer LNCS (2004) 79–86
3. Zabih, R., Woodfill, J.: Non-parametric local transforms for computing visual correspondence. In: Proceedings of the European Conference on Computer Vision, Springer LNCS (1994) 151–158

4. Barron, J.L., Fleet, D.J., Beauchemin, S.S.: Performance of optical flow techniques. International Journal of Computer Vision **12**(1) (1994) 43–77
5. Baker, A., Scharstein, D., Lewis, J., Roth, S., Blak, M., Szeliski, R.: A database and evaluation methodology for optical flow. In: Proceedings of the IEEE International Conference on Computer Vision, IEEE Press (2007) 1–8
6. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence. (1981) 674–679
7. Bouguet, J.Y.: Pyramid implementation of the Lucas Kanade feature tracker. In: OpenCV Documentation, Intel Corporation Microprocessor Research Labs (2000)
8. Tomasi, C., T.Kanade: Detection and Tracking of Point Features. Technical Report CMU-CS-91-132, Carnegie Mellon University (1991)
9. Shi, J., Tomasi, C.: Good features to track. In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE Press (1994) 593–600
10. Camus, T.: Real-time quantized optical flow. Journal of Real-Time Imaging **3** (1995) 71–86
11. Zhang, D., Lu, G.: An edge and color oriented optical flow estimation using block matching. In: Proceedings of the IEEE International Conference on Signal Processing. Volume 2., IEEE Press (2000) 1026–1032
12. Bruhn, A., Weickert, J., Kohlberger, T., Schnörr, C.: A multigrid platform for real-time motion computation with discontinuity-preserving variational methods. International Journal of Computer Vision **70**(3) (2006) 257–277
13. Zach, C., Pock, T., Bischof, H.: A duality based approach for realtime TV-L1 optical flow. In: Proceedings of the DAGM Symposium, Springer LNCS (2007) 214–223
14. Brox, T., Bregler, C., Malik, J.: Large displacement optical flow. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE Press (2009) 41–48
15. Steinbruecker, F., Pock, T., Cremers, D.: Large displacement optical flow computation without warping. In: Proceesings of the IEEE International Conference on Computer Vision, IEEE Press (2009) 1609–1614
16. Viola, P., Jones, M.: Robust real-time object detection. International Journal of Computer Vision **57**(2) (2001) 137–154
17. Papageorgiou, C., Poggio, T.: A trainable system for object detection. International of Journal Computer Vision **38**(1) (2000) 15–33
18. Salmen, J., Suttorp, T., Edelbrunner, J., Igel, C.: Evolutionary optimization of wavelet feature sets for real-time pedestrian classification. In: Proceedings of the IEEE Conference on Hybrid Intelligent Systems, IEEE Press (2007) 222–227
19. Grabner, H., Bischof, H.: On-line boosting and vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, IEEE Press (2006) 260–267
20. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers (1999)
21. Igel, C., Hansen, N., Roth, S.: Covariance matrix adaptation for multi-objective optimization. Evolutionary Computation **15**(1) (2007) 1–28
22. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research **181**(3) (2007) 1653–1669
23. Suttorp, T., Hansen, N., Igel, C.: Efficient covariance matrix update for variable metric evolution strategies. Machine Learning **75**(2) (2009) 167–197
24. Vaudrey, T., Rabe, C., Klette, R., Milburn, J.: Differences between stereo and motion behaviour on synthetic and real-world stereo sequences. In: Proceedings of the Conference on Image and Vision Computing New Zealand, IEEE Press (2008) 1–6
25. Igel, C., Glasmachers, T., Heidrich-Meisner, V.: Shark. Journal of Machine Learning Research **9** (2008) 993–996