# BriefMatch: Dense binary feature matching for real-time optical flow estimation

Gabriel Eilertsen[1], Per-Erik Forssén[2], and Jonas Unger[1]

[1] Department of Science and Technology
[2] Department of Electrical Engineering
Linköping University, Sweden
{gabriel.eilertsen,per-erik.forssen,jonas.unger}@liu.se

**Abstract.** Research in optical flow estimation has to a large extent focused on achieving the best possible quality with no regards to running time. Nevertheless, in a number of important applications the speed is crucial. To address this problem we present BriefMatch, a real-time optical flow method that is suitable for live applications. The method combines binary features with the search strategy from PatchMatch in order to efficiently find a dense correspondence field between images. We show that the BRIEF descriptor provides better candidates (less outlier-prone) in shorter time, when compared to direct pixel comparisons and the Census transform. This allows us to achieve high quality results from a simple filtering of the initially matched candidates. Currently, BriefMatch has the fastest running time on the Middlebury benchmark, while placing highest of all the methods that run in shorter than 0.5 seconds.

**Keywords:** Optical flow, Feature matching, Real-time computation

## 1 Introduction

Optical flow estimation is a fundamental problem within computer vision. It is useful in a wide range of applications, from temporal filtering to structure-from-motion. Due to its applicability, a huge body of work has been devoted to the topic. However, the great majority of methods do not focus on real-time, and it still remains a difficult challenge to determine robust and high quality flow fields in live applications. Real-time optical flow is essential e.g. for detecting moving objects on moving platforms, obstacle avoidance and gesture recognition. It can also be used in live video streaming, in order to perform frame interpolation, video stabilization, rolling-shutter correction etc.

In this work, we use local pixel matching, with binary robust independent elementary features (BRIEF) [11] as similarity criterion. We match these using principles from the PatchMatch algorithm [5]. Running on the GPU, this combination can efficiently estimate a candidate optical flow field. The field contains few outliers as compared to using direct patch comparisons or a Census transform [35]. This makes it possible to use only low-level filtering to obtain a high quality optical flow field, and thus avoid expensive global optimization.

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

- We propose to combine BRIEF and PatchMatch, for efficient dense feature matching in order to find a candidate flow field with few outliers.
- We show that BRIEF features improve both matching quality and efficiency, compared to the Census transform and direct pixels comparisons.
- By comparing to existing real-time and offline methods we show that our robust optical flow exhibits a very good trade-off between quality and running time. It is thus well-suited for real-time applications.

## 2 Background

While classical methods for optical flow estimation generally optimize globally in a coarse to fine setting, modern approaches tend to increasingly use local pixel matching. This means that fine detailed motion at large displacements can be better recovered. The matching can be performed using a sparse set of features [10, 28, 33, 34], and then propagated to neighboring pixels to give a per-pixel flow estimation. Alternatively, the matching can be done densely, comparing per-pixel local patch distances or dense features [2, 4, 12, 21, 23]. However, a dense correspondence field (CF) contains many outliers, and some post-processing is inevitably needed in order to find a final optical flow. For example, the method proposed by Bailer et al. [2] performs a hierarchical correspondence field search, followed by outlier filtering. We also make use of a per-pixel correspondence search. However, in order to relieve the need for post-processing we perform the matching using feature descriptors that makes for relatively few outliers.

The first use of binary patch descriptors were the local binary pattern descriptors (LBP) [26], also known as the Census transform [35] which is the name we will use. The Census transform encodes the local properties around a pixel by comparing it to the pixels in a local neighborhood. Census matching is common in stereo where it often is used together with a Semi Global Matching (SGM) in order to find disparities between images [13, 17, 18]. The Census transform has also been used in recent state-of-the-art for optical flow estimation [2], and in order to promote real-time performance [25, 31]. It has a number of favorable features that makes it well-suited for determining flow vector candidates [16, 32]. A different formulation of binary features is used in the BRIEF descriptor [11], where random pixel-pairs are compared in a local neighborhood. This descriptor has been extensively used as an efficient alternative to e.g. SIFT or SURF. There is also at least one example of it being used for stereo matching [36]. However, to our knowledge BRIEF has not been used in the context of optical flow estimation, and not in combination with the PatchMatch algorithm. We will show that using the BRIEF descriptor for real-time per-pixel matching has significant advantages over the Census transform, both in terms of robustness and speed.

The PatchMatch algorithm [5] is an efficient and well-known method to estimate the correspondence field between two images. The method alters random searches and a propagation scheme, in order to efficiently find approximate nearest neighbors. There are examples of the method being used both for optical flow evaluation [4, 21, 23, 34] and stereo matching [7, 13]. By combining BRIEF and PatchMatch we are able to achieve a robust optical flow estimation in real-time.

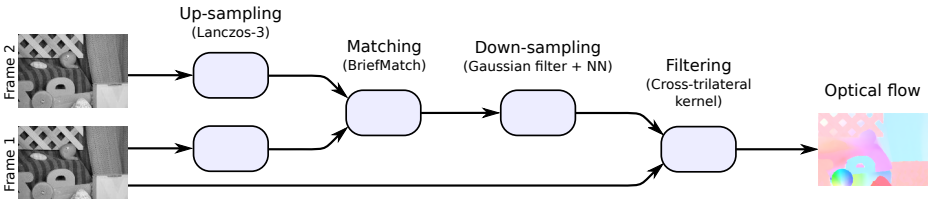



**Fig. 1.** BriefMatch optical flow pipeline.

## 3 Method

The proposed pipeline for BriefMatch optical flow estimation is outlined in Figure 1. It has two main components: 1) per-pixel matching between images using BRIEF descriptors and PatchMatch, to obtain an approximate correspondence field (CF), and 2) outlier filtering of the CF using a cross-trilateral filter kernel. This section describes the pipeline, as well as implementation specific details.

### 3.1 Correspondence field search

A BRIEF descriptor is constructed by performing $N$ pair-wise comparisons in a $S \times S$ pixels local neighborhood [11]. A binary feature vector is then built as a bit string, setting the bits based the comparisons. Denoting local coordinates as $\boldsymbol{q}$ and $\boldsymbol{q}'$, the descriptor at pixel $\boldsymbol{p}$ in the image $I$ can be formulated as

$$F(\boldsymbol{p}) = \sum_{i=1}^{N} 2^{i-1} \left(0.5 \operatorname{sign}(I(\boldsymbol{p}+\boldsymbol{q}_i) - I(\boldsymbol{p}+\boldsymbol{q}'_i)) + 0.5\right). \tag{1}$$

In BRIEF, the sampling of point pairs for the descriptor is done using a normal distribution, $\boldsymbol{q}, \boldsymbol{q}' \sim \mathcal{N}(0, \sigma)$, where the standard deviation is set to $\sigma = S/5$. The Census transform instead compares all pixels of a patch to the central pixel [35]. This can also be described using (1), if we define $\boldsymbol{q}$, $\boldsymbol{q}'$ as

$$\begin{aligned} &\boldsymbol{q} = (0,0), \\ &\boldsymbol{q}' = (i,j), \quad i,j = \{-S/2, -S/2+1, \ldots, S/2-1, S/2\}, \quad \boldsymbol{q} \neq \boldsymbol{q}'. \end{aligned} \tag{2}$$

Matching on descriptor fields $F(\boldsymbol{p})$ is performed by comparing the descriptors with the Hamming distance. This can be done efficiently using a `bitcnt` operation on the bitwise exclusive or (XOR) of two descriptors. In order to find a correspondence field between two images, we use the search strategy from PatchMatch [5]. This was originally designed to search for approximate nearest neighbors, comparing in terms of mean absolute pixel differences over patches. PatchMatch uses the fact that the offset to a nearest neighbor of a pixel is also often a good candidate for its adjacent pixels. By iterating random searches and a propagation scheme, the algorithm is able to find a good correspondence field in about 4–5 iterations. The algorithm is also directly applicable for finding nearest neighbors in terms of binary feature distances.

| | Pixel comparisons | Census transform | BRIEF |
|---|---|---|---|
| Handles light changes | ✘ | ✔ | ✔ |
| Outlier robust | ✘ | ✔ | ✔ |
| Robust to pixel noise | ✔ | ✘ | ✔ |
| Scales well with patch size | ✘ | ✘ | ✔ |

**Table 1.** Properties of different matching criteria. The binary descriptors provide invariance to light changes and a greater outlier robustness as compared to direct pixel comparisons. Census depends heavily on the central pixel of a patch, and is thus sensitive to noise. Only BRIEF is independent of the particular patch size used.

### 3.2 Properties of comparison criteria

In the previous subsection three different comparison criteria were mentioned for performing a correspondence field search – direct pixel comparisons, a Census transform and BRIEF descriptors. Figure 2 shows examples of the resulting correspondence fields from matching in terms of these criteria. The main differences are due to four distinct properties, which are summarized in Table 1. These are:

*Light changes:* The most obvious advantage of using binary features, such as Census and BRIEF, is the invariance to global changes in intensity. Direct pixel comparisons on the other hand, depend on absolute pixel values.

*Outlier robustness:* Encoding the local information around a pixel with binary features also results in a more robust matching that is less sensitive to outliers. For example, an inherent problem with patch comparisons is matching on edges where the background is moving relative to the foreground (see Figure 2(a)). With binary features, outlier pixels have less influence compared to direct pixel comparisons. In order to further improve matching on edges, we also tested to include edge-aware binary matching, that only uses comparisons on either foreground or background [36]. However, this did not improve on the result to a large extent, so given the limited time budget for real-time performance we did not include this in the algorithm.

*Robustness to pixel noise:* A weakness of the Census transform is that it relies heavily on the value of the center pixel of the patch, and it is thus sensitive to that pixel's variance. This is not the case for BRIEF and direct patch comparisons.

*Patch size:* While the complexity of direct patch comparisons and the Census transform are directly related to the size of the local neighborhood, BRIEF is only dependent on the number of binary feature comparisons ($N$ in Equation 1).

### 3.3 Flow refinement

In order to improve on the correspondence field search described in Section 3.1, we make two modifications. First, we up-sample the input frames in the spatial

(a) Pixel comparisons (b) Census transform (c) BRIEF

**Fig. 2.** Examples of correspondence fields estimated using different comparison criteria, where the images have been up-sampled by a factor 3 before matching (see Section 3.3). Using BRIEF results in the least amount of outliers.

dimension, as illustrated in Figure 1, to achieve sub-pixel accuracy and to provide a higher number of flow vector candidates. For the up-sampling the interpolation strategy is crucial, where a Lanczos-3 kernel improves significantly on bicubic or bilinear interpolation. After matching has been performed on the up-sampled frames, the resulting correspondence field is down-sampled to the original size. This is done by first filtering the flow vectors with a Gaussian filter, followed by a nearest neighbor down-sampling.

While the up-sampling strategy is able to refine the flow candidates, there are still many outliers in the flow field. Many existing patch-based methods for optical flow estimation use higher level optimization in order to refine the candidates, e.g. determining motions in segmented regions with RANSAC [12]. However, since we have a tight time budget to allow for real-time performance, we rely purely on local filtering of the correspondence field. While a median filter is able to remove many outliers, it also results in over-smoothed edges. Instead, we propose to use a cross-trilateral filter that in addition to spatial distance incorporates distances both in terms of flow vectors and image intensity,

$$\begin{aligned} \bar{u}(\boldsymbol{p}) &= \text{medianFilter}(\, u(\boldsymbol{p})\,), \\ d_{EPE}(\boldsymbol{p},\boldsymbol{q}) &= \sqrt{(u(\boldsymbol{q}) - \bar{u}(\boldsymbol{p}))^2 + (v(\boldsymbol{q}) - \bar{v}(\boldsymbol{p}))^2}, \\ d_I(\boldsymbol{p},\boldsymbol{q}) &= I(\boldsymbol{q}) - I(\boldsymbol{p}), \\ \bar{\bar{u}}(\boldsymbol{p}) &= \frac{1}{W} \sum_{\boldsymbol{q}\in\Omega} u(\boldsymbol{q}) G_{\sigma_1}\left(d_{EPE}(\boldsymbol{p},\boldsymbol{q})\right) G_{\sigma_2}\left(d_I(\boldsymbol{p},\boldsymbol{q})\right) G_{\sigma_3}(\boldsymbol{q}-\boldsymbol{p}). \end{aligned} \tag{3}$$

Here, $\boldsymbol{q}$ runs in a neighborhood $\Omega$ of the pixel $\boldsymbol{p}$. The distance $d_{EPE}$ incorporates the differences in the flow field. It is computed by comparing flow vectors to the median filtered version of the flow, $\bar{u}(\boldsymbol{p})$, to increase outlier resistance. The distance $d_I$ is a weighting term computed from the original image $I$. $G_{\sigma_{\{1,2,3\}}}$ are Gaussian kernels, which are normalized through the weight $W$. The filtering is performed in the same manner for both $u$ and $v$. The final optical flow estimation $\bar{\bar{\boldsymbol{r}}} = (\bar{\bar{u}}, \bar{\bar{v}})$ has some advantages over a separate median filter, where it better preserves corners and boundaries of the flow as can bee seen in Figure 3.

(a) CF (b) Median filtered (c) Cross-trilateral filtered

**Fig. 3.** The result of filtering the correspondence field (CF) using (b) a median filter and (c) the cross-trilateral filter in Equation 3. The up-samling ratio is 3, and BRIEF-64 features have been used, resulting in a total running time of about 65 ms.

### 3.4 Temporal propagation

Since the aim is to provide a real-time optical flow algorithm for processing frames in a video sequence, we can explore between-frame correlations. One simple modification is to initialize the nearest neighbor search using the information from the previous matching,

$$\boldsymbol{r}_t(\boldsymbol{p})_0 = \boldsymbol{r}_{t-1}(\boldsymbol{p} + \boldsymbol{r}_{t-1}(\boldsymbol{p})). \tag{4}$$

Here, $t$ is the current frame and 0 indicates the initial correspondences. Now, the number of iterations of the correspondence search can be cut in half without sacrificing performance. This makes for a significant reduction in runnning time.

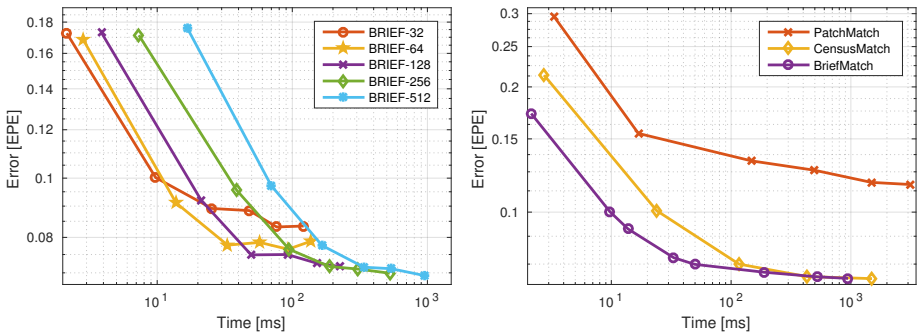
### 3.5 Implementation

The described method is well-suited for parallel implementation. The only exception is the serial propagation of nearest neighbors in the PatchMatch algorithm. In order to approximate this on the GPU we use a jump flooding scheme [29]. All the steps in Figure 1 have been implemented using CUDA, and the performance we report throughout this paper has been evaluated running on an Nvidia Geeforce GTX 980. For a typical setup, the running times of the different stages in Figure 1 are given in Table 2.

For the filtering step (Equation 3) we use a $13 \times 13$ pixels median filter. This involves sorting an array of 169 values for each pixel, which is expensive. Instead we choose to approximate the filter with a separable median computation, which significantly reduces running time without sacrificing quality to a large extent.

| Up-sampling | Matching | Down-sampling | Filtering | Total time | Framerate |
|---|---|---|---|---|---|
| 6.1 ms | 33.1 ms | 1.9 ms | 26.5 ms | 68.1 ms | 14.7 fps |

**Table 2.** Running time for the different steps in Figure 1, given a typical parameter calibration (this can be changed to trade-off quality and speed). Times are estimated using a $640 \times 480$ resolution sequence, and running on a Geeforce GTX 980.




(a) Different lengths of BRIEF descriptors (b) Different matching criteria

**Fig. 4.** Correspondence field search time vs. final error for the *RubberWhale* sequence. Sampling points have been estimated at different up-sampling factors before matching.

## 4 Results

In order to validate the performance of BriefMatch we perform a set of comparisons on the Middlebury training and test data [3]. In order to measure quality we use the average endpoint error (EPE), where the EPE is defined as the distance between estimated and ground truth flow vectors, $d_{EPE}(\boldsymbol{p}) = ||\boldsymbol{r}(\boldsymbol{p}) - \boldsymbol{r}_{gt}(\boldsymbol{p})||$.

*Impact of feature length:* With BriefMatch we have the option to trade off quality for running time by specifying the length $N$ of the binary features in Equation 1. Figure 4(a) shows the error on the *RubberWhale* sequence for a selection of descriptor sizes. The times specified are only for the matching, while the error is after performing the filtering in Equation 3. For each descriptor length the up-sampling factor has been set in the range $[1, 6]$. Since the method has a random search component, the outcome may be slightly different between runs, and thus the results are averaged over 30 separate runs. From the results it is clear that the optimal descriptor length depends on the up-sampling factor.

*Comparison to Census:* In order to show that the BRIEF descriptor performs better than direct pixel comparisons or a Census transform, Figure 4(b) shows the same comparison as in Figure 4(a). However, it is now made between different comparison criteria (see Section 3). All criteria have been matched with the same settings of the correspondence search, and the results have been filtered with the same calibration of the filter in Equation 3. For the direct pixel comparisons and the Census transform, the patch size has been tuned to achieve the best possible performance. For the BRIEF comparisons the descriptor length has been chosen for best performance, so that the plot is the lower envelope of the plots in Figure 4(a). It is clear that the performance when using binary descriptors is improved to a large extent, compared to direct pixel comparisons. The difference between using Census and BRIEF is smaller, although significant for shorter running times. For example, it takes about 3 times as long time for Census to reach the quality of BRIEF, when BRIEF runs in the range of 20–60 ms.

| Method | Time (s) | Avg rank | Army | Mequon | Schefflera | Wooden | Grove | Urban | Yosemite | Teddy |
|---|---|---|---|---|---|---|---|---|---|---|
| NNF-Local [12] | 673 | **3.2** | **0.07** | **0.15** | **0.18** | **0.10** | **0.41** | **0.23** | **0.10** | **0.34** |
| OFLAF [19] | 1530 | 9.5 | 0.08 | 0.16 | 0.19 | 0.14 | 0.51 | 0.31 | 0.11 | 0.42 |
| MDP-Flow2 [34] | **342** | 10.2 | 0.08 | 0.15 | 0.20 | 0.15 | 0.63 | 0.26 | 0.11 | 0.38 |
| BriefMatch | **0.068** | **66.2** | **0.09** | **0.21** | **0.25** | **0.20** | 0.93 | 1.69 | 0.25 | 1.25 |
| Rannacher [27] | 0.12 | 74.8 | 0.11 | 0.25 | 0.57 | 0.24 | **0.91** | 1.49 | **0.15** | 0.69 |
| Bartels [6] | 0.15 | 79.3 | 0.12 | 0.22 | 0.35 | 0.28 | 0.97 | 1.20 | 0.20 | 0.91 |
| FlowNet [14] | 0.5 | 81.5 | 0.11 | 0.30 | 0.62 | 0.27 | 1.04 | **0.46** | 0.17 | 0.75 |
| FlowNet2 [20] | 0.091 | 82.2 | 0.22 | 0.67 | 0.61 | 0.28 | 0.97 | 0.59 | 0.19 | **0.60** |
| PGAM+LK [1] | 0.37 | 118.6 | 0.37 | 1.08 | 0.94 | 1.40 | 1.37 | 2.10 | 0.36 | 1.89 |

**Table 3.** Results from the Middlebury benchmark. The list includes the top 3 performers and the six methods that run in under 0.5 seconds. The colors indicate how well BriefMatch performs relative to the compared methods.

*Middlebury benchmark:* The Middlebury online benchmark[1] currently comprises 125 methods. In terms of the average EPE, BriefMatch places in the middle of these, while being the fastest of all. Table 3 lists 3 of the top-performing methods from the benchmark, as well as the ones that run in $< 0.5$ seconds. BriefMatch performs very well for 3 of the sequences (green), with an error that is not very far from the top-performing methods while being about 4 orders of magnitude faster. However, for 2 sequences (yellow) the result is approximately equivalent to the best of the fast methods, and for 3 of the sequences (red) the quality is worse than many of the fast methods. In Section 5 we try to analyze why this is the case.

*Comparison to real-time methods:* A comparison with existing real-time methods for optical flow is listed in Table 4. The table includes three methods from OpenCV's CUDA library. The first is a Lucas-Kanade solver [24] in a pyramidal implementation [8]. The second is Farnebäck's method [15], that is based on polynomial expansion to approximate the neighborhood of each pixel. The third is the method proposed by Brox et al. [9], using a variational model and a warping technique. We also include the recent real-time method presented by Kroeger et al. [22], which uses a dense inverse search (DIS) for finding patch correspondences. All methods have been executed with constant parameters over the sequences, selected in an effort to give the best quality in approximately 50 ms. However, the pyramidal LK and the polynomial expansion methods do not provide viable options to trade quality for time. For these, increasing the number of iterations or pyramid scaling increases time, but quality does not scale well, and it is a better trade-off to run in shorter time. The times reported are estimated on a machine equipped with an Intel Xeon X5680 (3.33 GHz) CPU and

[1] `http://vision.middlebury.edu/flow/eval/results/results-e1.php`

| Method | Time (ms) | Avg EPE | R.Whale | Hydrangea | Grove2 | Grove3 | Urban2 | Urban3 |
|---|---|---|---|---|---|---|---|---|
| BriefMatch | 49 | **0.514** | **0.079** | **0.152** | **0.164** | **0.687** | 0.606 | 1.396 |
| Pyr. LK [8] | 25 | 1.312 | 0.306 | 0.563 | 0.394 | 1.367 | 2.301 | 2.940 |
| Pol. exp. [15] | **22** | 0.823 | 0.250 | 0.323 | 0.354 | 1.175 | 0.913 | 1.921 |
| Warp OF [9] | 54 | 0.565 | 0.175 | 0.232 | 0.326 | 0.911 | **0.580** | **1.167** |
| DIS [22] | 57 | 0.800 | 0.230 | 0.261 | 0.379 | 0.976 | 1.344 | 1.607 |

**Table 4.** Comparison to real-time methods using the Middlebury training data. The colors indicate how well BriefMatch performs relative to the compared methods.

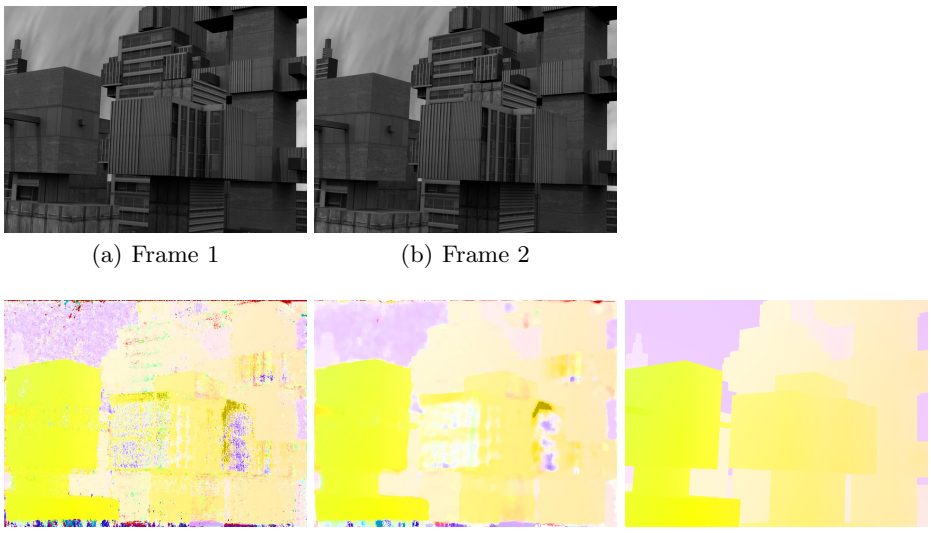
a Geeforce GTX 980 GPU. From the results we can see that BriefMatch reduces the error to 45–75% for four of the sequences (green), as compared to the second best method. The only sequences where another method yields slightly better quality are the *Urban2* and *Urban3* datasets (yellow). We discuss the reason for this in the next section.

## 5 Limitations

Looking at the results in Table 3 and 4, BriefMatch is not consistent in how well it performs relative to other methods. For some sequences it performs very well (marked with green), and for others there are many outliers (marked with red). Elaborating on the cause for this, we can discern the following reasons:

1. **Repetitive patterns:** Image structures that occurs repetitive can cause many outliers in the correspondence field. This is for example the case in the *Urban* sequences in Table 3 and 4. These sequences are also computer generated, which potentially may increase the problem. In order to successfully deduce the motion in areas of repetitive patterns, a global optimization is inevitably needed, which would make real-time performance difficult.
2. **Occlusion:** Many outliers can be created in areas that are occluded from one frame to the next and vice versa, e.g. close to image boundaries in a sequence with camera motion. This is the case for the *Urban* and *Teddy* sequences. To alleviate this problem a global optimization would also be needed.
3. **Z-motion:** In comparing patches between images – either directly or in terms of binary features – there is no invariance to image scale. This is a problem if objects or the camera are moving perpendicular to the image plane, as in the *Yosemite* sequence. In order overcome this problem, matching may need to be performed at multiple scales.

Figure 5 exemplifies the two first problems, using the *Urban3* sequence. In the mid regions of the images the building facades are highly repetitive, causing a large number of outliers. The problem with occlusion can e.g. be seen close to the top and bottom image borders, caused by a vertical camera motion.



(a) Frame 1 (b) Frame 2

(c) CF (d) Filtered CF (e) Ground truth

**Fig. 5.** Repetitive patterns and occluded areas result in a large fraction of outliers in the correspondence field (CF). These are difficult to completely remove with a local outlier filtering technique.

## 6 Discussion

We have shown that for real-time optical flow estimation BRIEF has a significant advantage over the Census transform, and that binary features in general are much better suited for the problem than direct comparisons of pixels. Furthermore, our optical flow algorithm offers a substantial increase in quality compared to existing real-time methods. Also, comparing to offline state-of-the-art methods it can in some circumstances perform on a par with these in terms of quality, while being about 4 orders of magnitude faster.

Although BriefMatch shows promising results, problems occur for repetitive patterns and occlusions. These problems would be the main focus for improving the method, investigating how they can be alleviated without using an expensive global optimization formulation. Another possibility is to use BriefMatch in offline applications. Since a number of the best performing methods use direct patch comparisons, from our investigation we expect that using BRIEF for these methods has the potential of increasing quality and/or reducing running time.

Other straightforward improvements include bidirectional matching, color matching, multiple neighbors in the correspondence search, improved temporal considerations, etc. The up/down-sampling strategy may also be subject to improvement, exploring other interpolation kernels and sampling schemes. We also expect that the current implementation can be improved on, for example using Halide[2] and by adapting the implementation to better use shared memory

[2] `http://halide-lang.org/`

and thread cooperation. Finally, for the BRIEF descriptor we have used random patch comparisons, were pixel pairs are drawn from a normal distribution, as in the original BRIEF formulation. Similar performance could probably be obtained with smaller descriptors, by using mining of feature pairs, as was done for e.g. the ORB descriptor [30].

**Acknowledgments.** This project was funded by the Swedish Foundation for Strategic Research (SSF) through grants IIS11-0081 Virtual Photo Sets and RIT 15-0097 SymbiCloud, and by the Swedish Research Council through grants 2015-05180, 2014-5928 (LCMM) and 2014-6227 (EMC2).

## References

1. A. Alba, E. Arce-Santana, and M. Rivera. Optical flow estimation with prior models obtained from phase correlation. In *Proc. of ISVC'10*, pages 417–426, 2010.
2. C. Bailer, B. Taetz, and D. Stricker. Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proc. of ICCV'15*, December 2015.
3. S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. *IJCV*, 92(1):1–31, 2011.
4. L. Bao, Q. Yang, and H. Jin. Fast edge-preserving patchmatch for large displacement optical flow. In *Proc. of CVPR'14*, June 2014.
5. C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. *ACM Trans. Graph.*, 28(3):24:1–24:11, July 2009.
6. C. Bartels and G. de Haan. Smoothness constraints in recursive search motion estimation for picture rate conversion. *IEEE TCSVT*, 20(10):1310–1319, Oct 2010.
7. M. Bleyer, C. Rhemann, and C. Rother. Patchmatch stereo - stereo matching with slanted support windows. In *Proc. of BMVC*, pages 14.1–14.11, 2011.
8. J.-Y. Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 5(1-10):4, 2001.
9. T. Brox, A. Bruhn, N. Papenberg, and J. Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. of ECCV'04*, pages 25–36, 2004.
10. T. Brox and J. Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Trans. PAMI*, 33(3):500–513, March 2011.
11. M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. In *Proc. of ECCV'10*, pages 778–792. Springer, 2010.
12. Z. Chen, H. Jin, Z. Lin, S. Cohen, and Y. Wu. Large displacement optical flow from nearest neighbor fields. In *Proc. of CVPR'13*, June 2013.
13. J. H. Cho and M. Humenberger. Fast patchmatch stereo matching using cross-scale cost fusion for automotive applications. In *Proc. of IEEE IV'15*, pages 802–807, June 2015.
14. A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. of ICCV'15*, 2015.
15. G. Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proc. of SCIA'03*, pages 363–370, 2003.

16. D. Hafner, O. Demetz, and J. Weickert. Why is the census transform good for robust optic flow computation? In *Proc. of SSVM'13*, pages 210–221, 2013.
17. H. Hirschmuller and D. Scharstein. Evaluation of stereo matching costs on images with radiometric differences. *IEEE Trans. PAMI*, 31(9):1582–1599, Sept 2009.
18. M. Humenberger, T. Engelke, and W. Kubinger. A census-based stereo vision algorithm using modified semi-global matching and plane fitting to improve matching quality. In *Proc. of CVPR'10 Workshops*, pages 77–84, June 2010.
19. T. Hyun Kim, H. Seok Lee, and K. Mu Lee. Optical flow via locally adaptive fusion of complementary data costs. In *Proc. of ICCV'13*, December 2013.
20. E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. Flownet 2.0: Evolution of optical flow estimation with deep networks. Technical report, arXiv:1612.01925, Dec 2016.
21. O. U. N. Jith, S. A. Ramakanth, and R. V. Babu. Optical flow estimation using approximate nearest neighbor field fusion. In *Proc. of ICASSP'14*, pages 673–677, May 2014.
22. T. Kroeger, R. Timofte, D. Dai, and L. V. Gool. Fast optical flow using dense inverse search. In *Proc. of ECCV'16*, 2016.
23. J. Lu, H. Yang, D. Min, and M. N. Do. Patch match filter: Efficient edge-aware filtering meets randomized search for fast correspondence field estimation. In *Proc. of CVPR'13*, June 2013.
24. B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of IJCAI'81*, volume 81, pages 674–679, 1981.
25. T. Müller, C. Rabe, J. Rannacher, U. Franke, and R. Mester. Illumination-robust dense optical flow using census signatures. In *Proc. of DAGM'11*, pages 236–245, 2011.
26. T. Ojala, M. Pietikainen, and D. Harwood. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proc. of ICPR'94*, volume 1, pages 582–585, Oct 1994.
27. J. Rannacher. Realtime 3d motion estimation on graphics hardware. *Undergraduate Thesis, Heidelberg University*, 2, 2009.
28. J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. Epicflow: Edge-preserving interpolation of correspondences for optical flow. In *Proc. of CVPR'15*, June 2015.
29. G. Rong and T.-S. Tan. Jump flooding in gpu with applications to voronoi diagram and distance transform. In *Proc. of I3D'06*, pages 109–116, 2006.
30. E. Rublee, V. Rabaud, K. Konolige, and G. Bradski. ORB: An efficient alternative to SIFT or SURF. In *Proc. of ICCV'11*, pages 2564–2571, Nov 2011.
31. F. Stein. Efficient computation of optical flow using the census transform. In *Proc. of DAGM'04*, pages 79–86, 2004.
32. C. Vogel, S. Roth, and K. Schindler. An evaluation of data costs for optical flow. In J. Weickert, M. Hein, and B. Schiele, editors, *Proc. of GCPR'13*, pages 343–353, 2013.
33. P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proc. of ICCV'13*, December 2013.
34. L. Xu, J. Jia, and Y. Matsushita. Motion detail preserving optical flow estimation. *IEEE Trans. PAMI*, 34(9):1744–1757, Sept 2012.
35. R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. In *Proc. of ECCV'94*, pages 151–158, 1994.
36. K. Zhang, J. Li, Y. Li, W. Hu, L. Sun, and S. Yang. Binary stereo matching. In *Proc. of ICPR'12*, pages 356–359, Nov 2012.